

https://dspace.org/

Evaluating the development and effectiveness of grit and growth mindset among high school students in a computer programming project

Kench, Delia Joan

2017

http://hdl.handle.net/10539/23461

Kench, Delia Joan (2017) Evaluating the development and effectiveness of grit and growth mindset among high school students in a computer programming project, University of the Witwatersrand, Johannesburg, <http://hdl.handle.net/10539/23461>

Downloaded from a DSpace repository. https://dspace.org/

EVALUATING THE DEVELOPMENT AND EFFECTIVENESS OF GRIT AND GROWTH MINDSET AMONG HIGH SCHOOL STUDENTS IN A COMPUTER PROGRAMMING PROJECT

Delia Kench

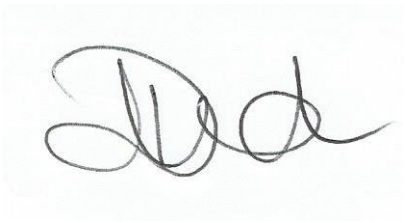
Student number: 8550970

A Dissertation submitted to the Faculty of Science, University of the Witwatersrand, in fulfilment of the requirements for the degree of Master of Science

Johannesburg 2016

DECLARATION

I declare that this Dissertation is my own, unaided work. It is being submitted for a Master of Science degree at the University of Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University.



Delia Kench

10th day of May 2017 in Johannesburg

ABSTRACT

This dissertation investigates grit “passion and perseverance” for a long-term goal and growth mindset in grade 11 high school students as they code a non-trivial programming project in Java over a six-week period. Students are often challenged by the complexities of programming and can be overwhelmed when they encounter errors causing them to give up and not persevere. The programming project includes scaffolding with frequent feedback to increase the motivation of students. The study used mixed methods research that used both quantitative and qualitative data to find answers to the research questions. Whilst the correlation between grit, mindset and the project results were moderate, that students submitted their project numerous times showed an indication to perseverance. The data gathered from the interviews further indicated that the students’ perseverance led them to employ their own problem-solving strategies when they encounter problems.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation for the support and guidance of Dr. Femi Otulaja and Prof. Scott Hazelhurst who without their supervision this Dissertation would not have been possible.

I also express my gratitude to the Executive Headmaster, Mr. Andre Oosthuysen, and the Headmaster of St Benedict's College, Mr. Dave Jeffrey, who have made this study possible by allowing me the time needed to complete this degree.

I am indebted to the 2015 Grade 11 Information Technology students who took part in this study. Their willingness to participate and their honesty in the inter-views provided me with valuable data for this study.

Lastly, I am so grateful to my family, particularly my husband, Mr. Wayne Kench, my children, Jamie and Liam for bearing with me and giving me the time and space to complete this study.

THANK YOU

LIST OF FIGURES

Figure 1. Activity Theory Analysis from Researcher's Perspective 29

Figure 2. Waterfall Model 31

Figure 3. The Gogga Grid Displaying the Results of Java Code 35

Figure 4. Explanatory Sequential Design 40

Figure 5. Sequence of Quantitative and Qualitative Data Collection and Analysis with Emphasis on Quantitate Research 40

Figure 6. PAT Box Plot 70

Figure 7. Grit vs PAT 77

Figure 8. Mindest-8 vs PAT 78

Figure 9. *Perseverance* vs PAT 79

Figure 10. Number of Submissions vs PAT 80

LIST OF TABLES

Table 1. Mindset Questionnaire 59

Table 2. Comparison of the Mindset-10 and Mindset-8 Questionnaires 66

Table 3. Summary Results of Quantitate Data 71

Table 4. *Mindset-8* Tally 73

Table 5. Mindset Classification 73

Table 6. Correlation Coefficients 74

Table 8. Bootstrapping Results 76

Table 7. Correlation Comparison 76

Table 9. Results of Students Interviewed 81

Table 10. Grade 11 and Grade 12 PAT Scores 97

Table 11. Results Summary 113

Table 12. Mindset-10 Raw Scores 114

Table 13. Mindset-8 Raw Scores 115

Table 14. Grit Raw Scores 116

ABBREVIATIONS

IT	Information Technology
IDE	Integrated Development Environment
IEB	Independent Examination Board
GUI	Graphical User Interface
OOP	Object-Oriented Programming
SQL	Structured Query Language
PAT	Performance Assessment Task

An extract of some of the work presented here was published in:

ICT Education

45th Annual Conference of the Southern African Computer Lecturers' Association, SACLA 2016, Cullinan, South Africa, July 5-6, 2016, Revised Selected Papers

Editors: Stefan Gruner

ISBN: 978-3-319-47679-7 (Print) 978-3-319-47680-3 (Online)

Kench D., Hazelhurst S., Otulaja F. (2016) Grit and Growth Mindset Among High School Students in a Computer Programming Project: A Mixed Methods Study. In: Gruner S. (eds) ICT Education. SACLA 2016. Communications in Computer and Information Science, vol 642. Springer, Cham

TABLE OF CONTENT

DECLARATION ii

ABSTRACT iii

ACKNOWLEDGEMENTS iv

LIST OF FIGURES v

LIST OF TABLES vi

ABBREVIATIONS vii

CHAPTER ONE – INTRODUCTION 1

1.1 General Introduction 1

1.2 The Background of the Study 1

1.3 The Purpose of the Study 2

1.4 Context of the Study 3

1.5 Problem Statements 5

1.5.1 Main problem 5

1.5.2 Sub-problems 5

1.6 Rationale for the Study 6

1.7 Significance of the Study 7

1.8 Research Questions 8

1.9 Limitations and Assumptions of the Study 8

1.10 Summary of the Chapter 9

CHAPTER TWO - LITERATURE REVIEW, THEORETICAL/CONCEPTUAL FRAMEWORKS 11

2.1 Introduction 11

2.2 Literature Review 11

2.2.1 What is Grit? 11

2.2.2 What is the Growth Mindset? 15

2.2.3 Feedback 19

2.2.4 Scaffolding Combined with Feedback 22

2.2.5 Gender Inequality in Computer Science 24

2.3 Theoretical Frameworks 25

2.3.1 Theoretical Framework Applied to this Study 28

2.4 Conceptual Framework 30

2.4.1 Software Engineering 30

2.4.2 Object-Oriented Programming (OOP) 33

2.5 Summary of the Chapter 36
CHAPTER THREE – RESEARCH METHODOLOGY 37
3.1 Introduction 37
3.2 Mixed Methods Research 37
3.3 Research Participants 41
3.4 Research Design 41
3.4.1 Summary of the Journal 46
Day 1 – double-period lesson. 47
Day 2 – single-period lesson. 47
Day 3 – double-period lesson. 47
Day 4 – double-period lesson. 47
Day 5 – single-period lesson. 48
Day 6 – single-period lesson. 48
Day 7 – single-period lesson. 48
Day 8 – double-period lesson. 48
Day 9 – single-period lesson. 48
Day 10, 11 and 12 – four lessons. 49
Day 13 – double-period lesson. 49
Day 14 – single-period lesson. 50
Day 23 – 30 –eleven lessons. 51
3.5 Trustworthiness, Reliability and Validity of Research 51
3.5.1 Use of Mixed Method Research in Computer Science 51
3.5.2 Discussion of using a Mixed Method Methodology for this Study 53
3.5.3 Quantitative Research — Principles to Test Validity and Reliability 53
3.5.4 The Data Collecting Instruments 56
Grit Questionnaire. 56
Mindset Questionnaire. 58
3.5.5 Qualitative Research — Principles to Establish Credibility 60
3.6 Summary of the Chapter 62
CHAPTER FOUR – DATA COLLECTION, ANALYSIS, RESULTS AND DISCUSSION 63
4.1 Introduction 63
4.2 Data Collection 63
4.3 Analysis of Collected Data 65
4.3.1 Quantitative Data Analysis 66
4.3.2 Qualitative Data Analysis 67

4.4 Research Findings and Discussion 70
4.4.1 Quantitative Data 70
Grittiness. 71
Number of Submissions. 72
Mindset. 73
Correlation Coefficients. 74
Grit vs PAT. 77
Mindset-8 vs PAT. 77
Perseverance vs PAT. 79
Number of Submissions vs PAT. 80
4.4.2 Qualitative Data 81
Summary of the Interviews 82
Discussion 91
4.5 Summary of the Chapter 97
CHAPTER FIVE – CONCLUSION 99
5.1 Introduction 99
5.2 Limitations of study 100
5.3 Future Work 101
5.3 Conclusion 104
REFERENCES 105
APPENDIX #1 - Mindset Questionnaire 110
APPENDIX #2 - Short Grit Scale 111
APPENDIX #3 – Results 113
APPENDIX #4 - Scripts in R Studio 117
Grit VS PAT 117
Grit vs PAT Scatter Plot 117
Mindset-8 vs PAT 117
Mindset-8 vs PAT Scatter Plot 118
Perseverance vs GRIT 118
Passion vs GRIT 118
Perseverance vs Passion 119
Perseverance vs Number of Submissions 119
Perseverance vs PAT 120
Perseverance VS PAT Scatter Plot 120
Number of Submissions vs PAT 120

Number of Submissions vs PAT Scatter Plot 121
Grit vs PAT Bootstrapped 121
Mindset-8 vs PAT Bootstrapped 122
Perseverance vs PAT Bootstrapped 122
Number of Submissions vs PAT Bootstrapped 122
APPENDIX #5 - Semi-Structured Interview Questions 124
APPENDIX #6 – Coded Interviews 125
Interview – Student A 125
Interview – Student B 128
Interview – Student C 135
Interview – Student D 143
Interview – Student E 148
Interview – Student F 154
Summary of the Codes 160
Version 1 160
Version 2 161
Version 3 163
Version 4 165
Version 5 166
Version 6 168
APPENDIX #7 - Ethics Clearance Certificate 170
APPENDIX #8 - IEB Rubric 171
APPENDIX #9 – Personal Journal 176

CHAPTER ONE – INTRODUCTION

1.1 General Introduction

Learning to program is a challenging and complex process for high school students. Students may give up rather than persevere. The concepts of grit and growth mindset are used in this study to describe and measure a student's attitude to their work regardless of failure. This dissertation investigates the grit and mindset in a group of high school students as they produce a complex, non-trivial, programming project in Java. Students were required to produce a project over a six-week period. The effect of each student's grit, and mindset was investigated to determine whether a relationship exists between grit and mindset and academic performance. This academic performance was measured in terms of a programming project that the students were required to produce over a six-week period. A mixed-methods research design was used where the students' grit and mindset were measured quantitatively using surveys to produce numerical data. The result of the programming project called a Performance Assessment Task was recorded numerically in addition to the number of submissions by each student. Qualitative data were produced by interviewing a sample of the students. Both quantitative and qualitative data were analysed to investigate whether grit and mindset have positive or negative effects on academic performance.

1.2 The Background of the Study

High school students[1] experience difficulties in programming. Programming can be difficult for students to grasp because of its complexity (Jenkins, 2001), (Cutts et al., 2010). In order to master the skills of programming a student needs to persevere to fix the many different errors and mistakes that may occur while producing code. The logic in programming is similar to mathematics where the skills accumulate over time with each concept building on previous concepts (Blackwell,

[1] The term 'student' will denote grade 11s who are 16 to 17 years of age studying computer programming at high school level at the time of this study.

Trzesniewski & Dweck, 2007). The construction of knowledge is recursive building on the existing knowledge of the student (Ben-Ari, 1998). The process of determining the best data structure, coding a solution and eliminating errors requires patience, perseverance and persistence. In addition, the complexities of learning to program can be influenced by the choice of programming language and the teaching methods of the assigned teacher, such as whether object-oriented programming is taught earlier or later in the course (Gries, 2008).

Students developing programming projects are often frustrated, anxious and lose self-confidence when they encounter errors (Ben-Ari, 1998), (Cutts et al., 2010) finding difficulty in completing complex programming tasks. Programming, in itself, is difficult and requires discipline with constant practice. In order to master the skills of programming, a student needs to persevere regardless of the type of errors they encounter. They need to seek out alternate strategies (McCartney, Eckerdal, Moström, Sanders & Zander, 2007) and be prepared to fail without reacting negatively. This perseverance is known as grit, i.e. *passion* and *perseverance* for a long-term goal which are predictors of success over and above IQ (intelligent quotient) (Duckworth, Peterson, Matthews & Kelly, 2007).

Linked to the lack of grit is the students' perception of their intellectual capability as being fixed and their failure to achieve as something that they cannot control (Blackwell et al., 2007). If they experience problems in their programming code, it is often perceived as their lack of ability that tends to restrict them from succeeding rather than their lack of efforts. Students with a growth mindset, who believe that their intelligence can be changed with *perseverance* and efforts, will more likely succeed. Learning to program can easily produce a fixed mindset (Cutts et al., 2010) since there are so many ways a student can get stuck. This tendency can induce a student to give up. Students with a growth mindset are more likely to employ alternate strategies to address their problems as the problems arise.

1.3 The Purpose of the Study

The purpose of this study is to investigate the importance of mindset and grit in developing programming skills. Students may easily give up when faced with the

complexity of programming. As a personal goal, I would like to determine what makes one student persevere and another not to persevere. Based on years of experience as a teacher, I have witnessed many intelligent and able students fail to produce a project which they are capable of doing. I have also witnessed many students become successful through their *perseverance*. This study is aimed at creating a deeper understanding of the student's motivation, *perseverance* and problem-solving strategies when they encounter obstacles by investigating grit and mindset.

1.4 Context of the Study

The study took place at a private single gender parochial high school in a metropolitan city in South Africa over a period of six weeks. The school was chosen for expediency since the author was a teacher at the school. This situation introduced bias into the research with both positive and negative consequences. On the positive side, the author knew the participants personally enriching the qualitative data. On a negative side, the author's desire for the students to achieve could have had an impact on the study although measures were taken to eliminate bias as will be described later on in this paper. As a result of the single gender school, the participants of the study were restricted to male students in grade 11 who chose IT as a subject and developed a programming project from September, 2015 to November, 2015. The differences between the male and female students studying computer-related courses is significant (Sidiropolous, 2016) and may have an effect on findings of this study, which may not be transferable to females in a similar educational context. The gender difference in computer science will be discussed in the literature review.

The students had chosen information technology (IT) as a subject in their grade 10 year where they are introduced to programming in Java. In order to take IT as a subject, the students needed to achieve more than 60% mark in mathematics in their grade 9 year and above 60% mark in IT in an exam that was written in July of their grade 9 year. IT was not offered as a subject prior to grade 9, and they had no experience of programming. To introduce the students to programming, a two-day

introductory workshop using a programming language called *Robomind*[2] was taught in their grade 9 year with the purpose of exposing students to basic programming commands of sequencing, repetition, selection statements and methods (procedures). After the workshop, which was held in June, the students wrote their *Robomind* practical exam in July of the same year. The exam was one hour long and required the students to use nested loops, 'if' statements and procedures.

The school is a member of the Independent Examination Board (IEB) which provides the IEB certification of grade 12 students. All grade 12 students write the IEB exams at the end of their matric (grade 12) year and need to meet certain criteria in order to achieve a result that is deemed a pass. The school values and expects a high level of academic performance and has had a 100% matric pass rate for over thirty years. Students are annually awarded colours based on their academic performance over the year in the form of half and full colours. In grade 11 an average of above 70% for all subjects is rewarded with a simple half colours scroll to be displayed on the student's blazer. Full colours are awarded for an average above 80% with a more decorative scroll and contrasting braiding on the sleeves and edges of the blazer. At the end of each of the three terms the students are ranked according to their averages in each grade and badges are awarded to students ranked in the top 20 according to these averages. These badges are usually worn on a student's blazer for the duration of the term. The top 20 ranking is valued among students and parents.

IT as a subject has consistently achieved excellent results with the students achieving over an 80% average for the subject for the last 15 years. There is an expectation upon entering the class that students will be expected to work hard yet will achieve good results as a result of this effort. Students participating in the subject have all achieved 60% in maths, have passed the July programming test and could either possibly perceive themselves or be perceived by others to be smart. Consequently, the class could be considered to be homogeneous in ability in terms of programming

[2] *Robomind* has been developed by MIT and provides commands for a robot-like car to manoeuvre around obstacles in a two dimensional space which is called a world.

and mathematical skills and students who gain access to the class could consider themselves to be talented in terms of mathematical and programming ability.

The project created by the grade 11 students is modelled on the grade 12 PAT project. The PAT project is a task that forms part of the IEB assessment and counts a total of 25% by the end-of-the-year result forming a significant part of the year's assessment. Students are exposed to the projects produced the students in the grade above with opportunities to interact with the projects and students who coded them. The standard produced by grade 12 students is high and grade 11 students are aware that their grade 11 PAT will form the basis for their grade 12 PAT. In order to familiarise the grade 11 students with the process of developing a PAT, a similar rubric is used but with reduced sections and a reduced total. A sample of the grade 11 rubric is provided in Appendix #8.

1.5 Problem Statements

1.5.1 Main problem

The main problem addressed in this study is the inability of many students to persevere when they encounter problems in programming and thus leading to poor results. The purpose of this study is to measure the grit and growth mindset among high school students in a computer programming project to determine their *perseverance* when encountering errors in programming. Although computer programming consists of many facets, the focus of this study is on programming errors in particular. This includes the use of a problem-solving strategy when attempting to correct the errors in the program.

1.5.2 Sub-problems

Students display lack of grit when programming. They do not persevere when they encounter difficulty when coding programs. They do not employ problem-solving strategies to correct errors.

Students tend to have a fixed mindset and do not believe that by persevering, they are able to improve their intelligence. Once they have failed to solve a problem, they often think it is because of their limited intelligence.

The relationship between grit and mindset and a student's PAT results needs to be investigated to determine whether these personality traits impacts student performance when coding a complex programming project.

1.6 Rationale for the Study

Intuitively the harder one works, the better one gets at whatever one is doing; however, not much research has been done regarding the combination of grit and growth mindset in high school students, in South Africa, whilst they develop a significant programming project. Most practitioners do not address grit and/or mindset in programming. Students who find value in efforts and *perseverance* tend to have a growth mindset (Hochanadel & Finamore, 2015) and will continue with a task regardless of the number of times it fails. Growth mindset can be termed as resilience where a student will bounce back after failure and not give up. In addition to resilience, they need to have unswerving interests and passion for a task or subject that spans a large period of time (Perkins-Gough, 2013). Grit is not a synonym for growth mindset. Grit is one mechanism for developing growth mindset. Grit can be identified through the students' *perseverance* when working on a task.

The reason for this study is to determine the effect of a student's grit (or perseverance) and growth mindset on their programming. Students with high scores in grit and growth mindset will tend to persevere when they encounter problems and employ a variety of strategies to solve the problem. Growth mindset and grit have not been studied together, nor have they been applied to high school students developing a large programming project.

My personal goal is to determine the relationship between grit and growth mind-set and the student's Performance Assessment Task (PAT) results with the hope of ultimately improving their PAT results.

Each student is required to develop a significant project called a PAT which constitutes 25% of their year-end mark. I would like to determine why some students do not achieve high results enabling me to provide a better way to guide the students through the process of developing a large scale programming project. By finding a

clear link between their project marks and their grit and mindset, I can plan an intervention at a later stage, which would be beyond the scope of this research report. By interviewing students with respect to the topic I hope to gain a deeper understanding of the problems faced by high school students when programming a PAT project. I hope to be able to identify with their context and perspectives in order to inform future teaching and learning in my class.

My practical goal is to determine the relationship between grit and growth mind-set and the student's PAT results through questionnaires and interviews. I will need to collect data about grit and growth mindset whilst the students develop their PAT programming project; and investigate the relationship between the data obtained and the PAT results. I will also record the number of submissions of each student.

1.7 Significance of the Study

This study is significant in determining whether a student's grit and mindset have an influence on a student's PAT results. Since mindset and the student's self-belief and their perception of their capability can impact their results, this provides hope that students are not restricted by the mere chance of their so-called intelligence. If ability can be changed with *perseverance* and hard work, students will have greater control over their success and teachers will be positioned to create a positive effect through their daily encounters. There is hope for students and teachers alike that students are not limited by their intelligence and that it is not fixed to some predetermined genetic intelligence roulette. Our idea of academic ability needs to be questioned and the types of lessons we, as teachers, deliver needs to change to allow students to persevere. Testing or assessing for the sake of measuring academic ability needs to be revisited. Teachers need to design tasks that provide opportunities for students to demonstrate *perseverance* or be given opportunities to develop a growth mindset. As teachers we need to re-evaluate our concept of failure and see it not as a final destination, but as a journey through which a student can progress. This means valuing the process and not the product.

1.8 Research Questions

How do grit and growth mindset influence/shape the learning of high school students in a programming project (PAT)?

Is grit and growth mindset related to high school student's PAT results?

How do grit and growth mindset impact on student's performance in their PAT?

How do the qualitative data explain the qualitative results?

Underlying these questions are the questions of how do I recognise and then measure grit and growth mindset. How can grit and growth mindset be quantified? What data can I use to quantitatively determine whether as student has grit or a growth mindset?

1.9 Limitations and Assumptions of the Study

This pilot study addresses the development of a programming project once the required programming skills had been taught before September 2015. Although grit and growth mindset was important in the months that lead up to the project, they were only measured once the project commenced.

Whilst debugging a program is important, this skill was not measured, only the grit and *perseverance* when debugging a program. A student may have improved their debugging skills during the study, but this result did not form part of the study, nor their approach to the process of software engineering.

The study takes place in a school with only male students. The student demographics comprised a majority of white students with a small representation of Asian, Indian and black students. Since the school is a private school, the students tended to be affluent. The demographics and socioeconomic factors could have affected the outcome of the study as parents have made a large investment in the education of their children and may place importance in the study. The school val-

ues academic achievement and students are rewarded when they achieve high aggregates across all their subjects. Outstanding academic achievement is rewarded by colours where students may attach scrolls to their blazers, and if the aggregate reaches an average above 70%, the student may differentiate themselves from their peers by the addition of braiding to their blazer. This obvious visual reward could have inspired students to be more invested in the study with the motive of improving their subject aggregate.

By the start of this project I assumed that the students were familiar with the programming concepts that had been taught in the beginning of the grade 11 academic year and are able to code objects, arrays of objects, a basic Graphical User Interface (GUI) in Java and to create a database in Access. I assumed that they had basic programming skills to debug code, fix run time and logical errors for small programs. I also assumed that the students were able to complete the questionnaires by being reflective in their learning and provide honest answers.

In the next chapter, a literature review will discuss the various studies on grit and mindset relevant to the educational setting of computer programming in a high school.

1.10 Summary of the Chapter

Owing to the complexity of programming, students often feel frustrated and may give up instead of persevering. This study aimed to investigate the effect of grit and mindset on a student's performance in a significant programming project. Grit being made up of two components namely, *passion* and *perseverance* which are considered to be better predictors of success than IQ. Students with a growth mindset are more likely to believe that their intelligence can be changed. These two factors are investigated in students when facing the complexities of programming. The study takes place in an all-male high school with grade 11 students where academic achievement is valued. The programming project forms a large part of the grade 11 IT syllabus and is marked with a detailed rubric.

The role of grit and mindset in programming has not been previously explored in terms of programming, neither have grit and mindset studied together in relation to *perseverance* and problem-solving strategies. The importance of this study is to explore a student's *passion* and *perseverance* and mindset when coding a solution to a non-trivial programming project at high school level.

CHAPTER TWO - LITERATURE REVIEW, THEORETICAL/CONCEPTUAL FRAMEWORKS

2.1 Introduction

In this section, I review the researches that have been done on grit, growth mindset, scaffolding and providing feedback to students. These concepts are relevant to the developments of the PAT project. In the theoretical framework section, I engage with constructivism and its role in computer science education. Scaffolding is linked to the Zone of Proximal Developments (ZPD). Activity theory is used to unpack the study and scaffolding is further linked to situated learning and cognitive apprenticeship. In the conceptual framework, the pedagogical decisions around the teaching of programming by introducing objects (a type of programming construct) before methods is also discussed. Finally, the process of software engineering is described to explain how large projects are coded when there is more than one programmer coding a solution.

2.2 Literature Review

2.2.1 What is Grit?

Simply put, grit is *perseverance* and *passion* for a long-term goal; it is how much someone is willing to stick with something regardless of failure. According to Duckworth *et al.* (2007), grit is a non-cognitive quality that emphasises the long-term stamina of an individual who will finish tasks and pursue an aim over a period of years. It is a marathon and not a sprint. It is not a single entity; it is made up of efforts and continued interests despite failure or adversary. Students achieve their goals by pursuing their aims over many years. Grit differs from self-control which may only be evident in the short term. Grit is not the same as the need for achievement where there is instant feedback on performance. This statement regarding feedback on performance is interesting and will be addressed later.

In a longitudinal study involving the 2005 Scripps National Spelling Bee and freshman candidates who entered the United States Military Academy, West Point, in July 2004, Duckworth and Seligman (2014) investigated the idea that IQ is less of

a predictor of academic performance than self-discipline. While there is a vast amount of research relating IQ to academic achievement, these studies are far less on the non-intellectual strengths of an individual compared to his/her academic achievement. Although IQ and the outcomes it predicts can be reliably measured, little is known about other factors that could predict academic performance. In studying why some individuals accomplish more than other people with similar intelligence, there is need to consider other attributes of an individual (Duckworth *et al.*, 2007). Research has indicated that individuals identified as possessing grit often demonstrate sustained efforts and interests over many years regardless of failures and setbacks (Duckworth, 2016). Achievement has been likened to running a marathon, characterised by stamina. Individuals who possess grit are not adversely affected by failure or the timeframe in which the goal must be achieved. The students understand that they are in for the long haul and they often set extremely long-term objectives. It has been shown that grit can be measured using a self-reporting questionnaire, such as the 8-point Grit Scale (Duckworth *et al.*, 2007).

The study of grit as predictor of success has been developed to include a grit scale which is a tool to measure grit in individuals (Duckworth & Quinn, 2009). Duckworth created a 12-point and an 8-point scale (Duckworth, 2016) and was evaluated by Duckworth and Quinn (2009). The former consists of 12 questions and the latter consists of 8 questions. Because the 8-point questionnaire was sufficient to gauge grit; and many of the questions in the 12-point scale are repetitive, the 8-point grit scale proved to be more stable over time and did not differ between genders (Duckworth & Quinn, 2009) (see Appendix #2).

Grit consists of two parts which are *passion* or consistency of interest and *perseverance* of efforts henceforth termed as *passion* and *perseverance*. *Passion* is interpreted as "consistency over time" (Duckworth, 2016) rather than an intense emotion. It is the commitment and sustained interest to the task over a long period of time. *Perseverance* is characterised by working hard and recovering from setbacks. It is a person's resilience to setbacks in that the individual does not give up and continues with the task. Grit does not indicate mindlessly following the same strategy regardless of the output. It is not a simple case of "try, try and try again" but

rather “try, try again, then try something different” implying a change in strategy but still persevering with the task (Duckworth, 2016).

In her book “Grit: The power of *passion* and *perseverance*”, Duckworth (2016) uses a 10-point grit scale similar to the 8-point grit scale. Two questions have been added, namely “My interests change from year to year” and “I have overcome setbacks to conquer an important challenge”. The questions in the 10-point scale have been rearranged slightly differently and are not in the exact order as the 8-point grit scale. Two questions have been extended in the 10-point grit scale to further qualify their meaning. “Setbacks don’t discourage me” has been extended to include “I don’t give up easily” and “I am diligent.” had “I never give up” included. In the 10-point grit scale the odd numbered items indicate the *passion* score and the even numbered items indicate the *perseverance* score. Returning to the 8-point grit scale, questions “2. Setbacks don’t discourage me”, “4. I am a hard worker”, “7. I finish whatever I begin” and “8. I am diligent” indicate *perseverance* and questions “1. New ideas and projects sometimes distract me from previous ones”, “3. I have been obsessed with a certain idea or project for a short time, but later lost interest”, “5. I often set a goal but later choose to pursue a different one” and “6. I have difficulty maintaining my focus on projects that take more than a few months to complete” indicate scores for *passion*. Duckworth (2016) states that a person’s score for *passion* shows the individual's ability to stay focused on a goal over time and not switch to a new goal while *perseverance* score indicates the individual’s ability to work hard and recover from setbacks (Duckworth, 2016). The idea that grit can be entirely divided into subsections and the combination of these two parts, namely *perseverance* and *passion* can lead to loss of significance in the ability to predict the performance of an individual (Credé, Tynan & Harms, 2016). Credé et al (2016) claims that *perseverance* predicts performance more accurately than *passion* and that these two constructs should be separated with *perseverance* being that factor to be used on its own as criteria to predict performance. They go on to suggest that grit researchers should shift their focus to studying *perseverance* as an area of research with more potential to predict performance.

Whilst the concept of grit is relatively new, it is can be compared to conscientiousness and even work ethic. Conscientiousness forms one of the big five model of personality traits along with factors of extraversion, openness to experience, emotional stability, and agreeableness with older individuals having been more conscientious than younger individuals. Credé *et al.* (2016) found conscientiousness to have a very strong correlation with grit and in particular self-control ($p = 0.84$) - a facet of conscientiousness. They question whether the concept of grit is merely "old wine in new bottles" with grit being redundant with conscientiousness (Credé *et al.*, 2016).

Work ethic is defined as the multidimensional set of values that reflect the significance of work in a person's life. These sub-values being self-reliance, morality or ethics, leisure, hard work, centrality of interest, wasted time and delay of gratification. Work ethic refers to when there is a relationship between effort and performance, which includes academic performance (Meriac, Slifka & LaBat, 2015). Grit and work ethics are similar in that they both relate to a person's effort toward accomplishing a task; however, work ethics are oriented toward work in general and staying engaged in tasks relating to performance. Grit is focused partly on persistence when encountering adversity. In particular, the work ethic aspect of hard work is related to the *perseverance* aspect of grit (Meriac *et al.*, 2015).

The relationship between *perseverance* and grit has been validated using the short Grit scale in a non-Western collectivist setting with college and high school Filipino students in a study performed by Datu, Valdez and King (2015). They explored the influence of culture on grit stating that in a collectivist culture like the Philippines, where interpersonal harmony and the quest of common goals is emphasised over personal autonomy and the pursuing self-set goals of people in individualistic cultures (Datu, Valdez & King 2015). *Perseverance* is strongly endorsed in both collectivist and individualistic cultures, whereas the pursuit of interests over time is more common in Western cultures. Their study proves that *perseverance* was more positively associated with academic performance than consistency of interests (*passion*) stating that an individual does not have to show consistency of interests to be considered gritty. It is possible that people in collectivist cultures will demonstrate

a "context-sensitive self" emphasising the adjustment of an individual to social-contextual conditions. A gritty student may continue to work toward a high school diploma even if he was not interested in graduating, but is doing so as it is expected from his family (Datu *et al.*, 2015). Interestingly, Duckworth's father fell a prey to the same ideal. He became a chemist at the urging of his father as these skills were required by the family textile business. Unfortunately, the Communist Revolution ended the family textile business and when the family moved to the Unites States, Duckworth's father was employed by DuPont where he retired as one of the top ranking scientists in the company thirty-five years later making a case for considering practicality over *passion* (Duckworth, 2016). Duckworth mentions culture in a different context by talking about being part of a gritty culture. If you want to be grittier then become part of a culture that shares those values. She quotes Dan Chambliss, a sociologist who studied swimmers, who stated "The real way to become a good swimmer is to join a great team". There is a reciprocal effect of a team's culture on a person who joins the team. If everyone in the team gets up early in the morning to train then it is easier to adopt such a habit (Duckworth, 2016).

Grit is an important factor in measuring the *perseverance* and *passion* of an individual, but stops short in finding a way of developing or altering a student's grit. The work performed by Blackwell *et al.* (2007) in developing growth mindset addresses the issue of altering a student's belief about their ability to improve their results.

2.2.2 What is the Growth Mindset?

Growth mindset is a significant factor in explaining a student's academic achievement or lack thereof independent of their intellectual capability. The student's belief about their ability informs their motivation for achievement. Since the difficulties experienced in programming produce emotional reactions such as frustration, anxiety and a loss in self-confidence (Ben-Ari, 1998), the study of mindset seeks to explore the effect of self-belief on a student's intellectual ability and hence, their academic performance. The study performed by Blackwell *et al.* (2007) showed that adolescent's beliefs about their intelligence is a key belief that informs their motivation for achievement. Students could hold two different "theories" about the

nature of their intelligence. Some students believed that intelligence is fixed or unchangeable, (fixed "entity"), which is termed an entity theory. Others believed that intelligence can change and be developed, which they termed an incremental theory. Their research on the theory of intelligence showed that students with equal intellectual ability influenced by their theory of intelligence react similarly to an academic challenge. In a study of students during a high school transition, Henderson and Dweck (1990) discovered that students who had a more incremental theory of intelligence had a distinct advantage over those with a more entity theory of intelligence stance; and had achieved better results in their first year of junior high (Henderson & Dweck, 1990) (as cited in Blackwell *et al.*, 2007).

Incremental theorists focus more on learning goals (goals that will increase abilities) as opposed to performance goals (goals that focus on documenting abilities) (Dweck & Leggett, 1988). Incremental theorists believe in the use of efforts to handle difficult tasks even if students have low ability and prefer to attribute failure to lack of efforts and display mastery-oriented strategies as opposed to helpless strategies (efforts withdrawal) when setbacks occur.

The longitudinal study previously mentioned on theories of intelligence in adolescents' mathematical achievement by Blackwell *et al.* (2007) was conducted in two parts. They first determined whether students held a belief that their intelligence was fixed (entity) or malleable (incremental). They then linked the student's theory of intelligence to the student's achievements in mathematics over a two-year period.

The first part of the study followed four sets of students as they progressed through grade seven and grade eight. In the beginning of their first term, the participants entering grade seven completed a motivational questionnaire which assessed their theory of intelligence, goals and beliefs about efforts and helplessness vs mastery–oriented responses to failure (Blackwell *et al.*, 2007). In addition, previous mathematics results from students entering grade seven class were recorded. At the end of each grade, in the first and second terms, the mathematics marks were recorded for all the grade six and seven students who took part in the study. Their research focused on student's learning of mathematics, which is similar to programming since the outcomes can be accurately measured; where the skills accumulate over

time (Blackwell et al., 2007), which has an impact on a student's future learning. Similarly, in programming, each concept builds on the previous concept, and if students fail to understand a concept, it will impact on their future understanding and hence learning.

The first study concluded that students with an incremental theory of intelligence diverged significantly from those with an entity theory of intelligence outperforming those with an entity theory of intelligence. These students developed positive learning goals, made fewer negative attributes when faced with setbacks, made less attributions about failure due to their lack of ability and were more likely to invest more efforts and choose strategies in solving problems.

Investigating the relationship between the theories of intelligence and motivational constructs, which they addressed in the second intervention study, (Blackwell *et al.*, 2007) taught incremental theory to an experimental group of students and then assessed whether these students show a more positive motivation and greater efforts in the classroom. Students displaying an incremental theory of intelligence were likely to be positive about challenging tasks and will use more efforts to overcome the difficulty of the tasks. Applying this theory to programming, since programming is considered difficult with many challenges, students with incremental theory are more likely to overcome their problems.

Blackwell's study concluded that students with a fixed or entity theory of intelligence will believe that their intelligence cannot be controlled (is fixed) and will be in favour of giving up or reducing efforts when encountering problems. Students who believe that their capability can be developed through their efforts (will hold an incremental theory) and will be in favour of pursuing more challenging tasks that develop skills. They are more likely to accept the use of efforts to overcome the challenges they experience. The study concluded that adolescents who endorse an incremental theory of intelligence endorsed stronger learning goals, held a positive belief about efforts, made more positive statements about their ability and created strategies based on efforts in relation to failure, which as a result boosted their achievement in mathematics. There are similarities in learning mathematics and

programming (Blackwell *et al.*, 2007) where skills are accumulated over time and skills are built on previously learned concepts.

A student with a growth mindset are most likely to believe that through their efforts, their capability can be developed. This effort implies continuous attempts at a task and not giving up despite failure or hang-in-there no matter what; and this is referred to as the *perseverance* dimension of "grit" (Duckworth et al., 2007). It takes *perseverance* to continue striving; and with this *perseverance*, a person's ability can be enhanced. Grit refers to the attitude toward a task whilst mindset indicates that with efforts, intelligence can be enhanced.

Growth mindset has been applied to teaching programming (Cutts et al., 2010) with first year university students in an introductory programming course over a six-week period with test scores being used as a measure of effectiveness. They discovered that when students struggle with programming problems, they can develop a fixed mindset unless they are encouraged with alternative strategies, such as referring to a crib sheet as an alternate resource. The researchers used three intervention strategies: mindset training; a rubric and crib sheets. The mindset training was delivered in a series of four 10-15 minute lessons covering both the mindset performances and learning goals, responses to feedback, role models and the neuroscience supporting growth mindset. The crib sheet contained a list of questions, hints and pointers that provided aid to the student. Each item on the crib sheet was linked to a wiki entry with more details and with relevant examples. A wiki entry is an entry in an online encyclopaedia. Students were encouraged to use the crib sheets, although the improvement in their results could not be attributed to the use of crib sheets since all the students in the study each had a crib sheet. Lastly, a rubric intervention encouraged students to use their feedback positively by highlighting to overcome their current problem. After six weeks, there was a positive effect for those who received growth mindset training. On the average, students who were taught about mindset showed a shift toward a growth mindset and those who were not taught about mindset showed a shift toward a more fixed mindset during the course. Students who were not exposed to the mindset training developed a shift to

a more fixed mindset as a result of their frequent exposure to failure in a programming environment (Cutts et al., 2010).

The researchers' focus on detailed formative feedback together with the combination of mindset instruction was to address the possibility that those with fixed mindset would ignore feedback as they would not believe that their efforts could improve their outcomes. Their study was conducted at an introductory level programming course and rather than at a more advanced programming level where students need to use their skills to complete a more detailed project. It was also conducted with university students and not high school students.

Murphy and Thomas (2008) describe individuals having a fixed mindset as being more likely to demonstrate a more helpless response to substantial challenges, avoid risks and abandon strategies as opposed to mastery-oriented response of those with a growth mindset. Students in an introductory computer science programming course face a constant bettering of negative feedback in terms of unfamiliar tools and environments, syntax and runtime errors that do not make sense and logical errors producing incorrect output. Students with a growth mindset will view these errors as challenges and opportunities for learning while those with a fixed mindset will view the negative feedback produced by these errors as a challenge to the intellectual ability and avoid similar conditions in the future (Murphy & Thomas, 2008).

Formative feedback goes a step further in improving a student's motivation. With regular and descriptive feedback, students can be motivated to improve. A growth mindset with a positive self-belief can overcome a negative reaction to failure. A student may experience failure during their project leading to negative emotions during a programming task. It is worthwhile investigating feedback as a mechanism of encouraging student's motivation and *perseverance* through their project.

2.2.3 Feedback

Feedback is significant in allowing students to answer questions about their progress through a project. They needed to know how well they are doing and whether

they are on the right track. These questions can be identified as the "What knowledge or skills do I aim to develop?", "How close am I now?", "What do I need to do next?" (Brookhart, 2008). With the use of the IEB rubric, which clearly denotes the skills and level of competency required at each stage, the student will be able to determine their success in the project they are programming. Feedback needs to address two factors: cognitive and motivation. According to the Brookhart (2008), students needed to know where they are in their journey and what the next step will be. This leads to the feeling of having control over their progress, which could be the motivating factor (Brookhart, 2008).

Not all feedback can be constructive, which could be a factor of the classroom culture. If the classroom culture values the process of improving rather than the end result of simply getting it right, inferring that the solution is either right or wrong, then feedback may not be used positively but rather to produce negative results. Students also need an opportunity to use the feedback given to help them improve their projects; and they have the opportunity to resubmit for assessment.

Feedback was initially linked to behaviourism where positive and negative feedback was used to shape an individual student's behaviour. This theory, using stimulus and response, was disregarded in favour of considering the students' needs to make meaning of their work and not just respond to stimuli. To make meaning of something requires that student use and control their own thought processes through self-regulation (Duckworth et al., 2007).

According to Butler and Winne (1995) as cited in (Brookhart, 2008), external feedback given by a teacher and internal feedback as in students' self-evaluation both have an effect on the students' knowledge and beliefs. External and internal feedback help students with self-regulation to determine their next learning goals, to devise tactics and strategies to reach them, and to produce work. Having grit implies that a student will try to solve the problem, but after repeated failed attempts an change in strategy is required (Duckworth, 2016). When an error is found, if the same strategy is repeatedly used to solve a problem with no positive effect, then a student could become frustrated, anxious and lose self-confidence (Ben-Ari, 1998). To direct a change in strategy, feedback can be used to direct a student to choose a

more effective strategy. Vandewalle (2012) indicated that a person with a growth mindset would be more likely to receive feedback and interpret the information positively to diagnose their shortcomings thereby enhancing their attention to their own mistakes. Feedback by the teacher should be an input that when combined with the students' own internal input can aid the students in measuring their success against the learning goals. It also should help them strategise what to do next. This process of encountering an error, diagnosing the cause, seeking alternate answers and coding the solutions are repetitive and requires *perseverance* or grit.

Feedback can be divided into four types or levels according to Hattie and Timperley (2007). Feedback can be about 1) the task (such as feedback about whether answers were right or wrong or directions to get more information), 2) the processing of the task (such as feedback about strategies used or strategies that could be used), 3) self-regulation (such as feedback about student's self-evaluation or self-confidence), and 4) the student as a person (such as pronouncements that a student is "good" or "smart"). Feedback, about the first two is the most effective; however, feedback about the second two can be harmful if the student does not hear it in a positive way. The fourth type of feedback is not recommended by Dweck (2007). She concluded that labelling a student as good or smart aligns more with a fixed mindset. The students seeing themselves as smart will likely consider that their intelligence cannot be changed and may become despondent should they not be able to do something even though they have been told they are smart. A comment such as "Good girl!" will not focus a students' attention to their learning.

In my opinion students should be able to obtain feedback as often as they need to continuously evaluate their development. This implies that students will have opportunities to submit their tasks numerous times. The rubric becomes a vital artefact for students to judge their performance. Each time student submits their projects, the rubric should be updated with their latest scores as well as be provided with qualitative feedback about their progress.

When a large task is given to students it should be scaffolded so the students are guided through the process with relevant support materials and structures.

2.2.4 Scaffolding Combined with Feedback

Contemporary learning theories support the two ideas: that knowledge is constructed and that learning and development are processes that are embedded in our culture and supported socially (Shepard, 2005). Scaffolding and formative assessment help move a student through the zone of proximal development (ZPD) according to Vygotsky (1978). Scaffolding is the support given by teachers to students in the form of hints, encouragement, and reminders to ensure the successful completion of a task. An important part of scaffolding is for the teacher to remove the scaffolding as the students become more competent so that the students can ultimately successfully complete the task on their own. Formative assessment should use insights about the student's current understanding to change and customise the instruction in order to develop the students' competence. Formative assessment is collaborative and involves the negotiation of what is meant and required between the teacher and the students so that the students can improve their performances (Shepard, 2005).

In a study conducted on teaching software engineering to 16 to 18 year-old high school students, the importance of positive feedback in increasing motivation is emphasized (Köhler, Gluchow & Brügge, 2012). The researchers recognised that motivated students can outperform a more talented student with less motivation. They linked motivation to interest in the topic by the students; and the researcher providing reasons why something should be done to maintain the motivation. The teacher then needed to determine whether the choice was feasible and what additional skills were needed for the students to be able to develop their project (Köhler *et al.*, 2012). The gap in their knowledge was bridged by scaffolding the form of tutorials and articles. This process is supported by ideas of cognitive apprenticeship (Brown, Collins & Duguid, 1989) that uses situated learning (Wegner, 1998) and scaffolding to create independence in the student with mediated learning (Piaget, 1968) from the teacher. For scaffolding to be successful it must offer the required help that will enable the students to reach the desired goal. The more capable the students become the less the scaffold is applied eventually leading students to reach the goal with little instruction (Köhler *et al.*, 2012).

In programming, students will frequently experience problems with their coding; either in syntax, run time or logical errors. Successful students will employ a repertoire of strategies to get 'unstuck' when programming (McCartney *et al.*, 2007); linked to this success is the ability to persevere when coding problems occur. A student may need to use a variety of strategies, such as, 1) getting help from other sources (peers, the Internet and books); 2) working on similar examples; 3) try to understand the problem by representing it using diagrams or breaking it down into smaller parts; and lastly, 4) by 'using the force'. The force being described as the student telling himself/herself to remember, think and persevere (McCartney *et al.*, 2007).

Scaffolding is the support given to students as they progress through the task with aim of the students achieving independence (Shepard, 2005). However, some tasks are too big for a student to take on at once. A large programming project can take weeks to design and code; and high school students do not have any previous experience in this process. It is necessary to break up the task into something that is achievable. Software engineering is a practice that deals with large complex software systems. It is a science whose aim is to engineer reliable, cost efficient and timeous software (Schach, 1990). A high school computer project that is developed by individual student falls into the category of software engineering, therefore, the principles of sub-dividing a large project can be applied.

In conclusion, much research has been done on the individual aspects of this study. Grit and growth mindset have been used to assess and improve *perseverance*; feedback and scaffolding can be used as useful tools to positively guide a student whilst they develop a project. The importance of grit and growth mindset as factors to determine the success of a programming project has been the starting point of this dissertation. Strategies that can be used to support students when developing a large project such as hint sheets and feedback have significant effects on improving results. This study aims to understand the significance of grit and growth mindset and the relationship with the final PAT results whilst supported with scaffolding, formative assessment and positive feedback.

2.2.5 Gender Inequality in Computer Science

The context of this study is an all-male school which may have a significant impact on the study. The South African woman is underrepresented in the IT field. In the 1990s, only 30% of degrees in computer science were achieved by woman with 27% of IT employees being a woman (Bovée, Voogt & Meelissen, 2007). Whilst the reasons for this disparity are many and not the place for this study, some ideas are put forward by Goode, Estrella and Margolis (2006) as a result of their study of high school students in Los Angeles, USA. They cited that in 2001-2002 as little as 18% of computer science bachelor's degrees were awarded to women in the United States.

Their study provided four themes to explain this inequality between the genders in computer science. Firstly, there were few schools that provide opportunities for learning computer science, rather settling for lower-level skills such as computer literacy. Secondly, the idea of relevance plays an important part in choosing computer science as a career path. A student's perception of what computer science is and what computer scientists do influences a student's decision for academic careers. Educators fail to make the connection between computer science and academic pathways, particularly for female students. Thirdly, females who take computer science have negative experiences in the classroom. They are not as tech-savvy as their male counterparts who are more likely to have acquired their computer science knowledge at home or through playing computer games. Lastly, the interpretation of the computer science by teachers often does not focus on higher-order thinking bypassing critical thinking and problem-solving (Goode, Estrella & Margolis, 2006).

The number of girls studying IT in South African IEB schools reported similar statistics. Of the 10,212 IEB candidates in 2015, a mere 785 were IT students currently studying the subject at a school compared to the more computer literacy-orientated subject computer applications technology which attracted 1,420 candidates, almost double that of IT. Of these 785 IT candidates, only 18% were girls, an increase from 16% in 2013 compared to 44% of girls taking computer applications technology. The inability to attract girls to the subject did not permeate to the results achieved

by the girls. The girls studying IT achieved similar results as the boys with 29% of both genders achieving distinctions and 82% of the girls achieving a result above 50% compared the 84% of the boys (Sidiropolous, 2016).

Applying these themes to this study, it is likely the participants in this study are more tech-savvy being males than females. IT as a subject has been taught at the school from grade 0 and the boys have had weekly exposure to computers through their years at the school. The computer skills taught from grade 0 to grade 8 include the use of various applications to support the content taught in other subjects. Coding is introduced in grade 9 from the perspective of problem-solving by teaching the underlying concepts of sequence, selection and iteration.

2.3 Theoretical Frameworks

Since the students involved in this study are social actors, and the learning of computer programming is taking place in the social context of the classroom and involves knowledge production guided by the teacher and the self-learning of the students, the social learning theory of cognitive constructivism developed by Piaget (1968) (as cited in Teaching Guide for GSIs, 2011), in addition to other related theories, will help me in answering my research questions. I am focusing on the mental processes instead of the observable behaviours as described by behaviourists (Teaching Guide for GSIs, 2011). Knowledge is constructed by students based on previous knowledge. Experiences are interpreted from the student's existing knowledge, cultural background, and personal history; using these factors helps students integrate new knowledge. Knowledge is hence constructed as students make sense of what they are learning based on their individual perspectives and cultural environment.

There is less research into constructivism applied to programming as there is in mathematics and physics education. In fact, computer science education may have more in common with engineering education (Ben-Ari, 1998). According to Ben-Ari (1998) when constructivism is applied to programming, certain factors are highlighted. Knowledge is gained in a recursive manner. Information that is gained through the senses is combined with existing cognitive structures and become the

basis of further constructed knowledge. Students need to understand the underlying cognitive model inherent in computer programming. For example, they need to have a mental model of an object or an array to understand the implication for their programs in using such data structures. Constructivism is the opposite of passive learning as it requires the students to be active; but if the tasks chosen are merely group projects or hands-on activities; they will fail if the teacher does not allow students to build a mental model that they can work with based on their existing knowledge (Ben-Ari, 1998).

A teacher's role, therefore, would be to facilitate students' accommodation of new information into existing knowledge so that they can make changes to their existing intellectual framework in order to assimilate this new information (Piaget, 1968). Effective learning would be achieved by the construction of mental models that successfully represent the new information. It is the role of the teacher to facilitate the acquisition of a working mental model by understanding the student's current mental model. Since it is not possible for the teacher to understand each and every student's cognitive structure (mental model), the student will also need to be supported by members of the class. According to Vygotsky (1978), anyone who interacts with the student during the learning process must be taken into account in the social world of the student (Liu & Chen, 2010). The teacher's role is thus extended to not only ensure that the student will be able to learn actively, but that the classmates are able to interact with the student socially providing many opportunities for the student to learn. Additional resources such as the Internet must be accessible to enrich the e-learning environment. These resources can guide students to assimilate new knowledge and help move students from the known to the unknown through a constructive sequence of steps. These steps will be described by the stages of a programming development described by the loosely-based software engineering model, the rubric and the phases where each component of the project is taught. The students will begin the project knowing the programming data structures and related processes. They will use this knowledge to create a larger programming project that solves a problem chosen by the students themselves. The sequence of steps to be followed to develop the solution will be scaffolded and guided from the

known tasks they have completed during the first half of the year toward the more complex project in the second half of the year.

The scaffolding and guidance provided the students align with the zone of proximal development (ZPD) as conceptualized in the Vygotskian theory of social constructivism (Vygotsky, 1978). Vygotsky (1978) argued that cognitive functions originate in and can be understood as a result of social interactions. Learning was not merely the accommodation of new knowledge into old knowledge, but a process through which learners were integrated into a knowledge community of practice (Wenger, 1998). Vygotsky (1978) emphasized the role of language and culture in cognitive development. Language and culture are the frameworks that humans use to communicate, to experience and to understand the world around them. Vygotsky articulated two levels in development; the level of actual development is the level that the student has already reached and where the student is capable of solving problems unaided. In this study, the grade 11 students spent the first six months of the year coding smaller programs using arrays, objects, text files and the combination of arrays and objects. They would not have been expected to combine these concepts on a larger scale.

Vygotsky's second level is the level of potential development or "zone of proximal development" (ZPD), which is the level that the student is able to reach with the guidance of a more knowledgeable other (MKO), for example, a teacher or collaborating peers. Students would need the assistance of a teacher and/or their peers to move them through the ZPD so that later on, without assistance, they would be able to produce their own programming projects. The teacher or peer would guide the student through a series of steps in order to move from the problem they were able to solve toward the unknown larger solution. The process was similar to mediated learning where the development was from inter-physical (teacher and student solve the problem together) to intra-physical (where the student can solve the problem on their own) (Vygotsky, 1978). In a discussion of the teaching implications of moving computer science students to a more malleable mindset by Murphy and Thomas (2008), they referred to Vygotsky's ZPD to push students beyond their independent capabilities supported by scaffolding. Students and teachers with a malleable view

of their intelligence will often benefit from moving through ZPD as both will be more likely to believe that the student is able to develop their abilities and that the student will accept scaffolding positively. Students with a fixed mindset will enjoy completing tasks within their range of abilities but are more likely to become frustrated when extended in the ZPD and even possibly disengage (Murphy & Thomas, 2008). Scaffolding provided by the teacher requires the teacher to engage, motivate and guide. Scaffolding becomes difficult when the student has a fixed mindset and only feels content with the tasks s/he can easily accomplish. When students with fixed mindset come across difficulties, they are less likely to accept offers of help from their peers or teacher. In programming, a student could either view bugs as an opportunity to learn or something to avoid. A teacher could also try to stimulate a growth mindset by encouraging and praising efforts rather than student's ability or intelligence (Murphy & Thomas, 2008). The language in the classroom needs to change to be focused on the work done, the efforts and persistence demonstrated by a student.

2.3.1 Theoretical Framework Applied to this Study

In this study, the students were given time in class to discuss their projects; they will be free to walk around and talk, ask for advice and demonstrate code. When they are not at school, they will have access to a WhatsApp group that will facilitate communication with their peers. The rubric, Battleship example, document templates, programming Integrated Development Environment (IDE), and/or programming language are all artefacts as in activity theory (Engestrom, 1999). Activity theory is a set of principles that form part of a conceptual system (Kaptelinin & Nardi, 1997), and it is a method of analysing a situation. The activity is hierarchical in nature which includes a subject moving towards an object through actions. A subject is the person involved in the activity; in this study, they are the students studying computer programming. The object (or objective) inspires the activity providing the direction, which in this case is producing the programming project whilst increasing grit and growth mindset (Nardi,1996). Accordingly, actions are then processes whose goals are to realise the object. These actions are mediated by tools or artefacts which help move the subjects to the desired object.

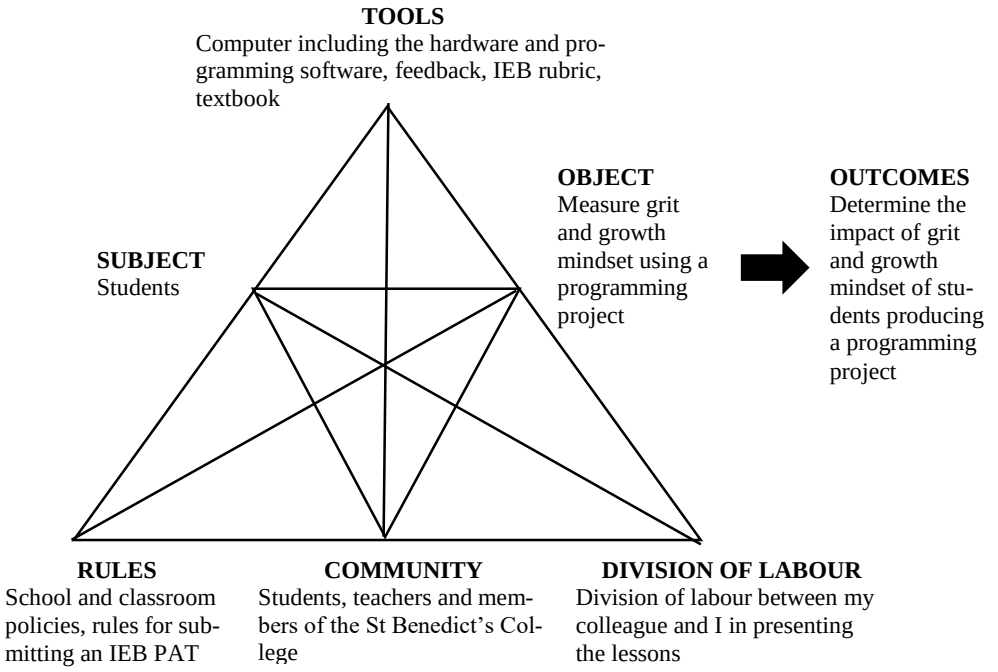


Figure 1. Activity Theory Analysis from Researcher's Perspective

The activity is controlled by a set of rules and takes place in a community. The work in the activity has to be divided into specific tasks for the people in the community. The rules mediate between the subject and the community and the division of labour mediates between the community and the object. Activity theory (Engestrom, 1999) is not only helpful in analysing the situation of this learning of programming and the relationship to grit and mindset, it is also useful in analysing the tensions that exist in this situation. Figure 1 serves to aid the researcher to unpack the issues relevant to the study and gain understanding of the context surrounding the research.

The choice of topic for the project is not important as long as it is authentic and legitimate for each student. If the project is chosen by the student, then they may be more intrinsically motivated to complete the project. Situated learning (Lave, 1991) is a theory of knowledge acquisition that has been applied in schools for technology-based learning activities that focus on problem-solving skills. Lave and Wenger (1991) identified two principles: 1) knowledge needs to be presented in an

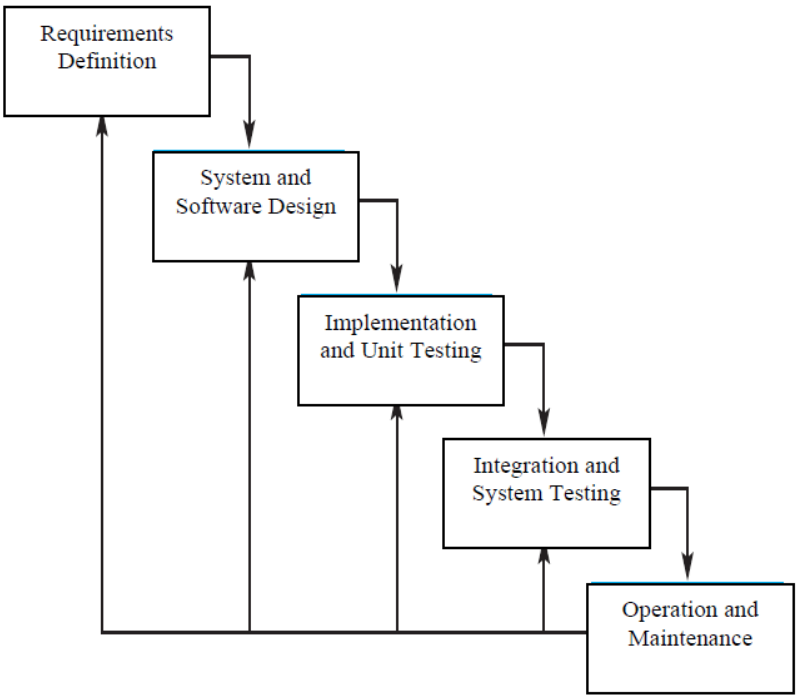
authentic context, and 2) learning requires social interactions and collaboration. Instead of viewing learning as the acquiring of knowledge, Lave and Wenger (1991) placed learning in a community of practice and asserted that learning takes place by participating in the community of practice where newcomers move toward old-timers in a community through "legitimate peripheral participation" (Lave, 1991). The learning is configured in the process of becoming a full participant in the community. Situated learning underlines the importance of students choosing a relevant and authentic project that will have meaning beyond the classroom.

2.4 Conceptual Framework

2.4.1 Software Engineering

Building a large software project is a complex task involving many lines of code and many people. Coordinating the development of a large scale project is similar to traditional engineering and should follow similar engineering principles such as design, testing and manufacturing (Schach, 1990).

The process of developing a software product is termed software engineering; and it is considered to be a life cycle. It is iterative since once a product is developed any new changes or alterations will start a new cycle. The waterfall model shown in Figure 2 (Schach, 1990) is a more classical organisational model for developing reliable software. The process is divided into clear and separate phases. Each phase being the input for the next phase. The process can often be shown using a diagram where each phase "flows" into the next phase (hence the term, waterfall). The names of the phases may differ slightly, but the process, in essence, remains the same.



(Nabil, Ali & Govardhan, 2010)

Figure 2. Waterfall Model

The phases are:

- system requirements (where the client's needs are analysed)
- specifications (the client's requirements are represented in a document stating what needs to be done)
- design (the project is broken up into modules during the architectural design and each module is detailed in detailed design)
- implementation (each module is coded)
- integration (the modules are combined and the project is tested as a whole)
- maintenance (any changes to the product in terms of repairs or enhancements once the client has accepted the product) and lastly
- retirement (when the product is removed from service). (Schach, 1990)

Whilst the waterfall model provides an excellent structure for dividing a project into clearly defined phases and developing good habits, such as, designing before

coding, it has a few drawbacks. It can be too linear and inflexible. The time between requirements and coding may be too long resulting in delayed deadlines. It is unrealistic to expect the requirements to be accurate so early in the project, which can produce errors as the project develops.

There are many models that have been designed to deal with the reliable development of software. A more iterative model allows for more flexible development. The project is divided into smaller parts and each is designed and coded to provide more immediate feedback as to how the project is progressing. (Nabil *et al.*, 2010)

Software engineering is particularly useful when dealing with many people coding a complex, large solution where each person's code is related and dependant on others producing multiple possible points of failure. The process of developing a project at the high school level with a single author cannot be termed software engineering in the true sense since there is only one author of the code and the scale of the project is not large enough. However, it is important to develop the project in stages in order to reduce the complexity of the project and to develop the habit of designing before coding (Nabil *et al.*, 2010).

When coding a large programming project with a single or multiple authors, object orientation becomes necessary in creating individual units that function as blocks used to subdivide the problem into sections so that these blocks can be designed and tested individually before they are combined to create the solution. Software engineering theory has been taught to high school students who were not familiar with programming in an intense three-day course. According to (Köhler *et al.*, 2012), the projects were small and students were led through the process by scaffolding the tasks and providing hint sheets to provide guidance to the students, if they were stuck. The hint sheet provided the most important instructions for the programming language.

Applying the principles of software engineering to a programming project can serve to enhance scaffolding as the teacher provides support in breaking down the task into achievable sized sections. The teacher can lead the students through specific phases such as GUI design and database access which would be common to most

of the student's projects. Once the students have achieved these parts, the scaffold will be lessened whilst they start to develop their own individual code to solve their particular task. Using the rubric and feedback, the student could continue to be supported whilst they develop their project and the project is formatively assessed.

In developing large programming projects, software techniques have been created that can break up a complex problem into well-defined sections with strict rules for their behaviour and interaction. These are called objects and the process of designing a software project with objects is called object-oriented programming (OOP). OOP design is beneficial in subdividing student's projects into achievable chunks; however, the implementation is complex, which will have an impact of the student's projects and processes.

One of the benefits of OOP is inheritance where objects can extend other objects, adding functionality. The advantage being that code can be reused and the objects being used should be more reliable as they should have been tested. Inheritance is outside the scope of this study as the grade 11 students will not have been taught inheritance.

2.4.2 Object-Oriented Programming (OOP)

Object orientation is an important concept in programming. The code is structured into objects which describe a real world object in the problem domain. An object consists of fields and methods. The fields store the details of an object such as a person's name, date of birth and address while the methods store the actions of an object. For example, using the Person object, the actions could be to sleep, eat and exercise.

In teaching programming, there is a variety of thought as to when to teach objects. Some feel it is important to teach objects first (Gries, 2008), while others prefer to teach a more procedural approach before objects (Burton & Bruhn, 2003). A procedural approach is characterised by breaking the program into smaller and smaller sections in a divide and conquer approach. Unlike objects, the focus is on what is

to be done procedurally with the data being incidental to the method. Object encapsulates the data (called fields) and the methods into an organisational unit. The advantage of a procedural approach is that the student has more experience with the fundamentals before encountering the complexities of objects. The OOP first argues that objects are fundamental to programming and should be dealt with as soon as possible (Gries, 2008). A third approach combines both by teaching of OOP in stages. The first stage would be to focus on the procedural aspects of the language and then to introduce the object-oriented programming once the concepts of procedural programming have been understood.

Java is an object-oriented language by nature where programmers using predefined objects are required to code calls to objects by prefixing a method call by the object name. Students are not expected create their own object classes, but must be able to call them correctly. In grade 10, students are introduced to the use of objects with the *Gogga* class as an introduction to programming in Java. This is a visual component that can be imported into a Java program that allows the programmer to move a *Gogga* (a local name for a bug in South Africa) object given certain commands. The class contains a *Gogga* object that has fields to store the values of the objects *x* and *y* position, size of trail, colour and direction. The students use method calls such as *move*(), *turnLeft*(), *turnRight*(), *setPostition*(*x,y*) to manipulate a bug object to draw shapes. Emphasis is placed on the use of the *Gogga* class and the terminology in and around the use of objects is constantly reinforced. For example, students are taught to instantiate the *Gogga* object, the use of different constructors to assign values of the fields of the *Gogga* object and the prefix each method call with the object variable. Students are required to learn the terminology and are assessed on their knowledge in cycle tests and examinations. They are also given practical examinations in which to demonstrate their ability to code a working solution in a given time limit.

The *Gogga* class was introduced to grade 10 students as a visible way to understand the workings of Java. The feedback is immediate, and since it is visual, the student can see the result of their code through the shapes that the *Gogga* object draws. The

Gogga object was based on the LOGO turtle, but is simplified to remove the complexities of geometry. The object can only turn 90 degrees easily producing shapes like square, rectangles and zig-zag lines.

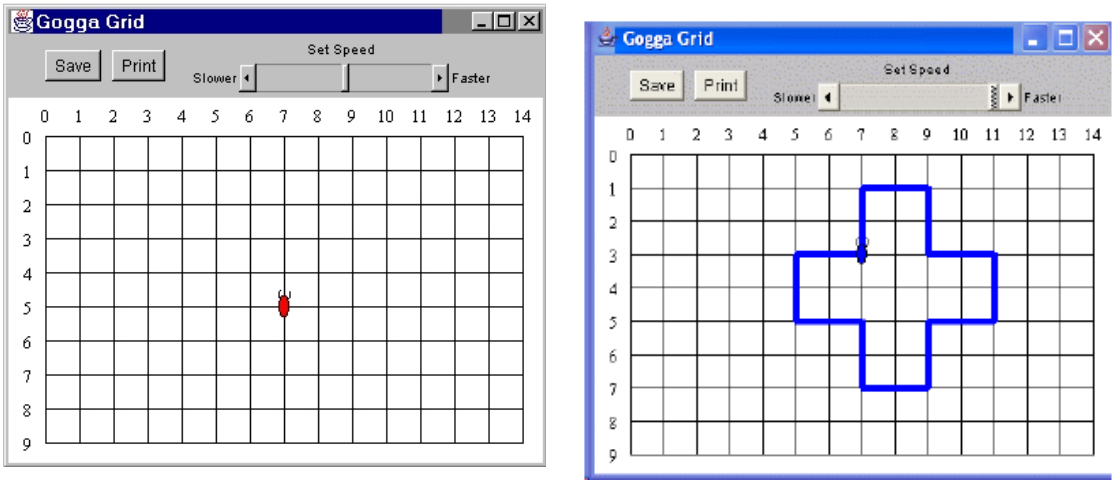


Figure 3. The Gogga Grid Displaying the Results of Java Code

(Kench, 2014)

In grade 11, the students code their own objects and use the concepts taught in grade 10 relating to object calls with *Gogga* to understand their own objects. The students coded many classes initially. They created a variety of classes to perform string manipulation, classes to process arrays, classes to store details of an individual, such as an employee and finally classes to store an array of objects. The grade 11 students wrote their July practical exam in which they were required to code a solution using an array of objects. The paper led the student through the design by indicating the class in which the students were required to place their code. The paper was graded for complexity starting with a simple base class with constructor, accessor, mutator and toString (a method for combining the values of the fields into a string type) methods proceeding to sorts and searches in the array class.

Using objects in a programming project provides structure to their solution. Together with scaffolding and feedback, the students can be led through the process of choosing, designing and coding their objects. The details of the study will be discussed in the next chapter where the intricacies of the process will be explained.

2.5 Summary of the Chapter

In this chapter, the concepts of grit and mindset were explored in greater detail. In particular, the sub-concepts of *passion* and *perseverance* of grit were discussed in relation to work ethics and conscientiousness. Both these concepts describe a student's response to learning situations. Since this study took place in the context of a programming project, feedback provided by the teacher is important in increasing motivation and guiding the students. The programming project provides the opportunity for the students to demonstrate their grit and mindset which is managed by the teacher. The project is scaffolded to help the students to move though the ZPD and combined with feedback, the students should have opportunities to become more motivated. The principles of software engineering are applied to the design of the project and provide a basis for the scaffolding of the task. The role of the teacher in this project is to facilitate the student's accommodation of new knowledge into their existing knowledge. The use of objects in the programming project aids in dividing the project in clearly identifiable sections so that students are able build their solution in 'blocks'.

CHAPTER THREE – RESEARCH METHODOLOGY

3.1 Introduction

This study takes place in an educational context where learning is an intensely personal journey. Each student has their own background and culture which influence the way in which they assimilate new knowledge. The process of investigating the effect of grit and mindset needs to take into account these personal experiences and record this qualitative information in a meaningful way. It also needs to quantify numerically a score to describe the level of grit and the mindset of each individual. It would be a disservice to the study to only use quantitative or qualitative data. Instead to gain a richer understanding, a combination of quantitative and qualitative data needs to be used. Mixed methods research is a suitable research methodology to combine both types of data effectively and in a significant manner.

3.2 Mixed Methods Research

Mixed methods can be described as "multiple ways of seeing" (Creswell & Clark, 2011) so that the results of the study can be seen from various angles. Both quantitative data and qualitative data have their strengths and limitations which can be overcome by the combination of both. The method uses collection, analysing, and mix of both quantitative and qualitative data in a single study. In essence it seeks to find multiple meaning incorporating a diversity of viewpoints (Creswell & Clark, 2011).

Mixed method design is considered a separate research design for collecting, analysing and reporting research by integrating both quantitative and qualitative data (Cresswell, Plano-Clark, Gutmann & Hanson, 2003). Mixed methods gather both quantitative and qualitative data and analyses this data rigorously and then integrates or mixes both types of data. This can be performed either sequentially where one builds on the other, or by embedding one form of the data in the other, giving priority to either quantitative or qualitative data or both depending on what is emphasised by the researcher. The researcher may use both research methods in a single study or a series of studies, framing the procedures within theoretical lenses and

within a philosophical worldview to form a research design as a plan for conducting the study (Creswell & Clark, 2011). Mixed methods can also be termed a "concurrent triangulation method design" where there is a triangulation of data collection, analysis of both quantitative and qualitative data and the integration or combination of the data during the discussion stage of the report (Cresswell *et al.*, 2003).

By combining both quantitative and qualitative data, the assumption is that the combined strength of both data collections will produce a better understanding of the problem (Cresswell, 2015). Quantitative data provides a more general understanding of the problem by examining a large number of people and determining their response to variables. The data can be efficiently analysed and the relationships between the variables can be investigated to determine possible cause and effect. Quantitative data does not explore the context of the research or the setting of the participants. The individual voice of each individual participant is not heard. Qualitative data provides a more detailed response to the problem by studying a few individuals and exploring their perspectives. The voice of an individual can be captured and understood in the context of the problem. It is focused on the participant and not the researcher (Creswell & Clark, 2011).

Mixed methods research overcomes the limitations of qualitative and quantitative research since quantitative research restricts the understanding of the context and perspective of an individual. The quantitative researcher's personal bias and interpretation may not be addressed either. Qualitative research, on its own, has become an accepted form of inquiry in the social sciences as it provides detailed information about the context of the study (Cresswell *et al.*, 2003). In qualitative method, research is restricted by the sample size of participants and creates difficulty in generalising the findings of the research. Mixed methods research helps to answer questions that cannot be answered by one type of research method alone. (Creswell & Clark, 2011).

Mixed methods research includes a description of procedural guidelines with the use of visual models, a notation system and a specification of the types of designs. This system was developed by Morse, a nursing researcher (Cresswell *et al.*, 2003). A visual model depicts the flow of research activities. The model depicts both the

qualitative and qualitative methods as boxes and uses arrows to show the sequence of the events in the study. The diagram could be linear where the quantitative data collection precedes the qualitative data collection followed by the results. Alternatively the diagram could be at right angles showing that the qualitative and quantitative data collections are combined to produce the results. To show the importance of one type of collection over the other, capital letters can be used so that "QUAN" may indicate quantitative data collection has an emphasis over "qual" qualitative data collection. Morse (1991) included the terms *simultaneous* and *sequential* to describe whether the data collection was performed one after the other in the case of sequential or simultaneous where data collection is performed at the same time. Research design varies in terms of the sequence of the data collection and the priority given each form of data collection. The sequence used to collect the data is determined by the intent of the researcher. If the researcher wishes to first understand and explore the problem being addressed in the study and then follow up with the examination of the problem, the order of the research will be qualitative data followed by the quantitative data. The qualitative data could be used to develop a hypothesis and be interrogated using the quantitative data. Conversely, when the quantitative data is collected first, followed by qualitative data, a researcher could explore the relationship between variables using a large sample and then investigate a limited set of cases in the qualitative part of the data collection. If data collection is performed simultaneously the researcher's aim is to identify similar findings in both sets of data that supports the research hypothesis (Cresswell *et al.*, 2003).

In this study, an explanatory sequential design was chosen, the quantitative method was first applied, followed by the quantitative interviews which will be used to explain the qualitative results in more depth. The quantitative research was used to investigate the relationship between grit, mindset and academic performance. Since there was not a strong correlation between these variables, the significance of the number of submissions was also explored. Using the results of the quantitative data, a few participants were selected and interviewed to gain a deeper understanding of the context in relation to the research questions posed.

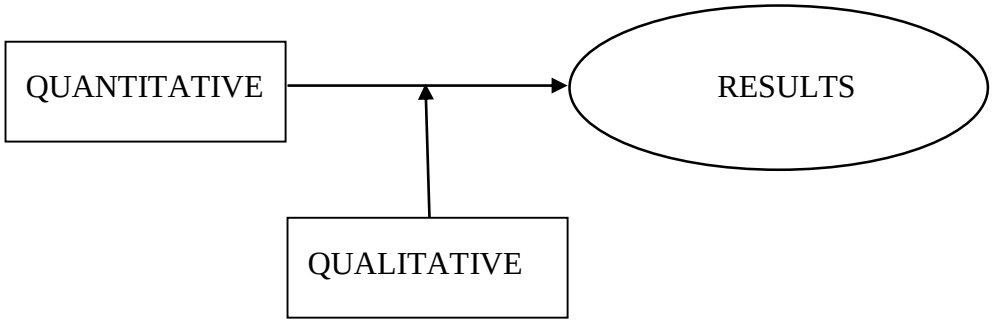


Figure 4. Explanatory Sequential Design

By first employing quantitative method, the intention is to explore the problems and the variables engaging the participants and then follow up with a more in-depth investigation of a select number of participants. The priority is on the quantitative data as it will be used to inform the choice of the students to provide the qualitative data in the form of interviews. The two methods of research will be integrated during the interpretation of the data. The study takes place in two phases: the quantitative phase followed by the qualitative phase. A group of students would be selected from the original sample and the data collected from the interviews will be used to interrogate the quantitative results.

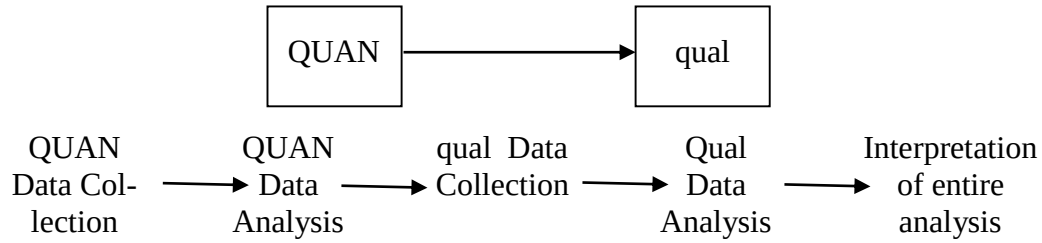


Figure 5. Sequence of Quantitative and Qualitative Data Collection and Analysis with Emphasis on Quantitate Research

The research study used a mixed methods approach combining quantitative and qualitative research methods. The results of the grit and mindset questionnaires, the

number of submissions and the PAT scores were used to produce quantitative numeric results. The students were given rubrics at the beginning of their project. Each time they submitted their project, their progress was recorded on their rubric by assigning marks to each section and their number of submissions was recorded.

Once the students had submitted their final project a sample of students was interviewed. These interviews were transcribed, coded and analysed for patterns such as similar words or phrases to provide qualitative data. The analysis was used to uncover themes in which students strategise when they encountered setbacks and how they overcame the difficulties, if and when they encountered them.

3.3 Research Participants

The participants were grade 11 male students who were beginning their PAT task in IT in July 2015. There were 29 students split between two classes, one taught by myself of 14 participants and one taught by a colleague with 15 participants. The students were 17 to 18 years of age at the beginning of the study. All the members of both the IT classes were invited to take part in the study with 29 out of the 39 students agreeing to participate. Students were invited by means of a participation information sheet and participants who wished to take part in the study signed a consent form including their guardian's consent.

In the study to manipulate mindset to influence introductory programming performance performed by Cutts *et al.* (2010) the class of 170 students were divided into 12 smaller groups for practical work with a typical size of 14 students, which is similar to group size used in this study.

3.4 Research Design

The participants of this study were pre-tested in grade nine to determine whether they met the criteria to take IT. During grade 10, the beginning of the three-year programming course, the participants were taught Java using a procedural approach and were introduced to objects using the *Gogga* class. In grade 11, they were taught object-oriented programming in the first half of the year. In September of 2015, the grade 11 students began their programming project.

The students were divided into two equally-sized classes. The one class was taught by the researcher and the other by a colleague of the researcher. The researcher and colleague met frequently to ensure that both classes were taught the same content and performed the same tasks and kept to similar submission schedule for the PAT.

Each student had been taught the content required to develop the project but needed to do so under guidance (Teaching Guide for GSIs, 2011). They all had been taught the skills needed to be combined to solve their programming tasks. In July of 2015, they wrote a 3-hour practical programming examination, which assessed their use of objects and arrays of objects including searching and sorting. This knowledge was needed to identify and select the correct programming tools that were required for their tasks. These programming tools needed to be combined to solve a larger problem. For example, the use of a text file as opposed to a database would affect their programming code or the use of an array, a 2D array or a simple object. Students needed the assistance of a teacher and their peers to move them through the ZPD so that later on, without assistance, they were able to produce their own projects. The teacher guided the students through a series of steps in order to move from the problem they were able to solve toward the unknown larger solution. The process was similar to mediated learning where the development was from inter-physical (teacher and student solve the problem together) to intra-physical (where the student can solve the problem on their own) (Vygotsky, 1978). The project was broken into manageable sections, and the class as a whole developed the peripheral sections, namely, the help screen and the user log on process, which provided access to the database. All students began with a similar project since each project required a help screen and user authentication. Once this had been achieved, the more general aspects of the project were discussed, such as, the different types of data structures required for a card game, board game or quest. The choice of scaffolding was loosely-based on an iterative software engineering model.

Applying activity theory (Engestrom, 1999) and Vygotsky's (1978) tools, the importance of the tools that were used to mediate the learning of students were discussed with the students. Students were required to become more independent learners as they tried to determine possible solutions to an error. A WhatsApp group

was created for each class with the members of the class joining the group to aid each other when help was needed. The Integrated Development Environment (IDE) also aided the students in identifying errors. The IDE supplied descriptions for errors detected in the students' code in the form of error messages, which were mostly helpful, and thus added to the student's resources to help them solve their problems. The IDE provided regular and immediate feedback when coding. The level of feedback is more detailed for the simpler syntax errors, slightly less detailed for run time errors and non-existent for logical errors. The assessment rubric which was provided by the IEB provided information about the steps involved in the process of developing a project as well as the components required such as a user interface, the primary and secondary storage.

In the previous year of grade 10, programming had been taught using a procedural programming approach using Java. This method of teaching was both teacher- and learner-centred. The grade 10 programming concepts were taught using the "Exploring IT" series of textbooks (Kench, 2011) where each new section was explained and sometimes demonstrated by the teacher coding on a computer linked to a data projector so all students could see the code. The teaching of the concepts and coding was delivered in a teacher-centred manner. Once a concept had been introduced, students were given activities listed in the textbook to complete. The activities were designed to illustrate a particular concept. The students were required to type in the code and make changes to determine the effects of the changes. The activities, progress through concepts and after a concept had been explored using the activities, exercises were assigned by the teacher so that students could demonstrate their learning. At particular points, the students were drawn together to revise all the concepts taught by the teacher.

In January 2015, there were two classes of students studying programming. They were in their second year of their three-year course and had been taught program-

ming structures such a simple data types, sequencing, selection and iteration statements in Java. In the first six months, students were taught databases with SQL[3] statements, object and arrays of objects. Once the students had completed their programming syllabus, their grit and growth mindset were pre-tested in the beginning of the third term of their grade 11 year. The grit short scale questionnaire and the growth mindset questionnaire were used. Both produced numerical values to signify the level of grit or growth mindset in an individual. By recording the number of submissions that the student made of their project, a further numerical value was added and recorded for each student. A selection of students was each interviewed to identify their grit and growth mindset when encountering errors or problems and the process they followed to solve these problems. This led to discovering that each student had developed their own individual problem-solving strategy combined with the use of resources that they are able to use effectively.

The pedagogical approach used to teach grade 10 and grade 11 programming leading up to the programming project used in the study was a blend of activity-based and a hands-on approach where the textbook and computer were used to mediate learning (Engestrom, 1987); Vygotsky (1978)). The teacher tried to limit the time given for instructing students and maximise the time the students had to work on the activities and exercises. Students were able to collaborate with fellow students to understand concepts and verbalise their experiences. This teaching style was inculcated into the classroom culture and students had become accustomed to learning in this manner. The same culture and style of teaching and learning was used throughout the programming project.

In the third term of 2015, the students were given a topic for a programming project that was within their capability. The students had written an examination in July that assessed their programming ability with respect to the content required for the third term. Each student was required to code their own project and was allowed to choose their own topic. The teachers checked that the student was able to code the

[3] SQL – Structured query language is a high level language that used to extract data from a database or manipulate the data by editing, inserting or deleting data.

project with/using the knowledge they were taught in the beginning of the year. The project was assessed using a rubric that was adapted from the Independent Examination Board (IEB) rubric for matric assessment. The rubric had some of the assessment criteria required for grade 12 removed so as to meet more simplified needs of grade 11. The grade 11 rubric was marked out of a total of 70.

Although the projects developed by the students were single-author and was too small to meet requirement for software engineering, its process was employed to ensure a structured solution. The model used was iterative where the project was broken down into sections. After the requirements for each student's project had been established, students developed their projects in sections. Students designed, coded and tested each section (Graphical User Interface, classes, database) before moving on. The project was developed over six weeks with students attending seven thirty-five minute long lessons per week with some lessons being double periods and some single from Monday to Friday. The project was scaffolded by the teacher by jointly proving the code, database and GUI design for the help and user authentication part of the project.

A sample game called Battleships was coded with the class which used a 10 by 10 grid and hides a selection of ships. The user is required to "find" the ships by clicking on the buttons in a 10 by 10 grid layout each button representing a square in the grid. Each time the user clicks on a button the score is decreased unless the user detects a ship in which case his score will be increased based on the type of ship. The object of the game was to locate all the ships before the score was depleted. The purpose of this example is to demonstrate the use of 2D arrays which is often used in board games. Students have the choice of coding a game based on a board such as snakes and ladders. The solution to Battleships is developed with the class in order to provide them a working solution with code snippets that they could incorporate into their project. The student would need to be able to identify the relevant code and adapt it to their solution. The documentation for the code is often a challenge for students who are not familiar with describing a solution in words. Documentation samples serve as templates and are digitally available consisting of a table of contents, heading styles for the appropriate headings linked to the IEB

rubric and the description. The documentation templates provided were documents provided by students from a previous grade who had achieved excellent PAT results. The students were instructed not to copy the documents but to use them as an indication of layout and content. Since each project had to be original the teachers were able to identify if a student had in fact lifted text from the document templates.

Students were aware of methods to find and correct errors in their code. In grade 10 they were taught how to detect errors. They were shown how to use the IDE to trace through their programs. This method requires the student to "watch" certain variables in a separate window in the IDE and then step through each line of code, line by line and noticing the changes in the variables as the code executes. This process is time consuming and becomes more complex when the program has more than one class. Students could trace their programs on paper listing variables as columns and writing down the values to each variable as they progress through the code line by line. Some students prefer to select a few variables and then perform calculations for a specific section of code after identifying the problematic section. Another method that is popular with students is to place many output statements in the code to view the values of variables as the code executes. These methods are time consuming and require patience and *perseverance*. The process of racing a program will be discussed with the students in their interviews to understand how they approached detecting and correcting errors.

The project was assessed at the end of grade 11 forming the foundation for their grade 12 project. Many grade 11 students adapt, extend or modify their grade 11 PAT projects for their grade 12 submissions. The PAT has a significant weighting of 25% of the year mark.

3.4.1 Summary of the Journal

I kept a journal (Appendix #9) briefly describing the events that took place each day. My colleague and I worked closely together on this study which was aided by the sharing of the same classroom. This allowed us to synchronise the lessons for both groups. Both classes had the same lesson delivered on the same or next day if the timetable permitted. On most days a lesson was thirty five minutes long, but on

occasional periods would be shortened according to the school's needs such as an additional assembly. If my colleague was unsure of anything, I was able to step in to provide the same instructions that were given to my class.

Day 1 – double-period lesson. The study began with a double-period lesson showing the students how to create a user and help table in MySQL using NetBeans. They were shown how to create a GUI with the purpose to authenticate a user when logging on. A second GUI was created to ultimately display help topics extracted from the help table in the database. The GUI had multiple buttons, one for each help topic. Each student was required to begin his own project which was the beginning of their PAT project. No code was written for any of the GUIs although their function was explained to the students.

Day 2 – single-period lesson. Students were given time to complete the tasks from the previous lesson and were shown where to access the document templates. They were shown examples of PAT documentation produced by students from previous grades. The specification document was explained and the deadline for the submission of the specification spelling error document was set for day 7, one week later.

Day 3 – double-period lesson. Students were shown how to extract the data from the help table in the database and display the relevant text on the screen based on the button that was clicked. The same method was used for all the help topic buttons. Students had to develop an algorithm to authenticate a user by accessing the user table to check if the user exists before the user can be authenticated by testing the user's password. If the user does not exist then the user needs to be added. The students were provided with the code for the database access in a class entitled DB with methods to query and update tables in a database. The query and update methods would be needed in their user authentication algorithm.

Students were given time to complete the questionnaires in class.

Day 4 – double-period lesson. Students were given time in class to complete the tasks from the previous lesson.

Day 5 – single-period lesson. A separate 2D grid class was introduced as a data structure for Battleships together with the constructor and toString methods. The constructor placed the ships in random positions on the grid. The students had the option to extend the constructor to add a variety of ships of different sizes. All the students coded this class, even if they knew they did not require it for their own PAT.

Once completed, the students continued to work on their own projects and complete their help and log in GUIs.

I checked the responses from the mindset and grit scores ensuring that each student had completed the questionnaires and calculated the grit and mindset scores.

Day 6 – single-period lesson. A GameGUI for the Battleships problem was created with code to check whether clicked button on the GUI was the location of a ship by calling a related method in the 2D grid class. I demonstrated how this code could be replicated for many blocks. Students were given time to work on their 2D grid class.

Day 7 – single-period lesson. The progress of the students was checked. The students were reminded that they should have the two tables in the database created and populated with data, a working help and log on GUI, a DB and a 2D grid class**.**

Students submitted their specification document.

Day 8 – double-period lesson. Each student's code was run by the teacher to ensure each student had achieved the requirements of the previous lesson. If a student had not completed the required tasks they could use the time in class to consult with their peers, fix their errors and demonstrate the corrected code to the teacher. The student's progress was monitored to ensure the students remained up-to-date and was on track.

Day 9 – single-period lesson. The specifications were returned to the students. A mark sheet was provided and a mark out of 14 was awarded using the rubric. Comments were written on each submission regarding the suitability of their project and

areas for improvement. The teacher ensured that the projects did not overlap extensively so that students could not hand in identical projects. General comments on their submissions were discussed with the class, including common errors. Students were encouraged to use these comments to submit an improved version. Some students were encouraged to change their topics if they were considered unsuitable.

Students had to extend their project to include code for Battleships using front-end GameGUI with buttons representing each square of the back-end 2D grid class. Each time a button is clicked on the GameGUI, a method in the 2D grid must be called, passing parameters and returning a value. The button on the gird needed to react based on what was stored in the 2D grid by changing colour according to what ship was detected.

Day 10, 11 and 12 – four lessons. Students were given time to work on their GameGUIs calling the methods in the 2D grid class. A deadline was issued for Day 12's lesson to check that the GameGUI and 2D grid was working along with all the previous requirements of the project.

Day 13 – double-period lesson. Each student's code was run and checked to determine if their GameGUI and 2D grid was working. A random button was clicked on their GameGUI and the button had to respond based on the type of ship stored in the 2D array. If the student had not achieved any of the previous tasks checked on day 8 they were given another opportunity to demonstrate their coded solutions to the required tasks.

Students submitted the second draft of their specifications. A record was kept of their number of submissions on their mark sheet. Each time a student submitted they had to keep the original mark sheet and indicate what changes they had made in their new submission. Students normally highlighted their changes which sped up the marking process. If a student had achieved full marks out of 14 for their specifications, the mark sheet and the specification document was stored in accordance with the school's policy for storing evidence of student's work and ensuring they are not lost. These documents are returned to the students in grade 12 when they revisit their PAT for their grade 12 submission to the IEB.

Students who were able to produce printed versions of their specifications with demarcated changes had their specifications remarked in class and an updated mark awarded.

Day 14 – single-period lesson. The layout and function of the design document was described and supported by the document template and previous examples. The relationship between the specification document and design document was explained. Students were given the remainder of the lesson to work on their projects.

Day 15, 16 and 17 – four lessons. Students were given time in class to work on their projects, ether their code or the documentation. Some students asked for advice on the design in particular, the class diagrams. The deadline was issued for students to submit their design documents on day 18.

Day 18 – 22. Students were taught the remaining theory lessons to complete the theory syllabus. No homework was given allowing the students the opportunity to work at home on their projects.

Students submitted design documents which were returned two days later with a mark awarded based on their evidence. Again, comments were written on their design document and the class diagrams were carefully checked to ensure the project would satisfy the separation of code requirements. Some students continued to submit their requirements documents if they had not achieved full marks. These were normally marked in the presence of the student if the changes made were small and quick to read. The type of feedback provided was relevant to each project. The first level of feedback was regarding the suitability of the design, once the student had chosen a correct database structure, class diagrams, sequencing and algorithms, the feedback became more specific by commenting on grammar, document layout and finer details.

Students were given time in class to answer the grit and mindset questionnaires for the second time.

A final due date was given for day 30. No late submissions would be accepted.

Day 23 – 30 –eleven lessons. Students were given time in class to complete their projects. The lessons became informal and students could continue to submit their specification and design documents, although by this stage most had achieved full marks for their specifications and were working on their design documents. Students were able to move around the classroom and ask their peers or the teacher for help should they need assistance. Each submission was marked in class during this stage with the student present. Feedback was verbal and relevant to the student's progress and achievements.

Once the design document was completed and full marks awarded the mark sheet and the design document was stored together with the student's specification document.

For the technical part of the project, the student's project was run by the teacher. To establish the use of parameters and separation of code, the code of all the classes was printed and checked. The total number of submissions was calculated by totalling on the number of submissions for the specification, design and technical phases of the project. The printouts were stored along the rest of the documentation and the mark sheets.

In the week after the day 30, the boys were again given time in class to complete the questionnaires. Checks were made to ensure each student had completed the questionnaire and the participation forms were all completed and filed.

3.5 Trustworthiness, Reliability and Validity of Research

The research method described above is comprehensive and covers all the aspects of the study, but the appropriateness of the type of research for this particular study needs to be investigated. By using mixed method approach, I am incorporating two distinct types of methods into a single study.

3.5.1 Use of Mixed Method Research in Computer Science

Mixed method research has moved into the realm of computer science research. The limitations of qualitative research has been understood and could be compared

to the Garbage In - Garbage Out where an illusion is created by using hard numbers to ascribe authority to research (Lister, 2005). During the process of operationalisation the original research question may be changed from something vague like "Did my students learn better" to "Did my students earn better marks?" whilst this may not be a good example of operationalisation it does show how a research question may not measure what is originally intended. A $p < 0.05$ may not indicate the relevance of the research if the result is caused by an extra uncontrolled variable. Improved results could even be caused by the Hawthorne Effect where students respond to the additional attention the received as part of the experiment (Lister, 2005).

Similarly, qualitative research methods can also prove to be effective or ineffective. If used effectively, qualitative data can be used to provide rich, detailed, descriptions of the human processes and clarify the operationalisation process by identifying the variables that would best be used in the quantitative method. This sequential design would require qualitative research to be performed before quantitative research in order to identify the variables of the study (Lister, 2005).

Ineffective, or poorly executed qualitative research can lend credibility to a contemporary idea that does not have merit. According to Lister, this problem has occurred in educational research which in his terms is "fashion prone" where a new "fashion" is supported by qualitative research that has been badly implemented (Lister, 2005).

One of the primary goals of computer science research is to identify the factors that explain why some people are able to program while other cannot and, in essence try to identify "predictors of success" (Lister, 2005) which is a similar theme in this study. Whilst this research has been conducted over many years, more recent work has used quantitative methods with linear regression models to account for only half of the variation in the performance of computer science students. To account for the other half, qualitative methods needs to be used where students are interviewed and observed to gain new understandings (Lister, 2005).

Lister (2005) concludes that bad research or the inappropriate use of research methods is the cause of invalid research. The research methods in themselves are not bad, merely incorrectly chosen and applied to the situation.

3.5.2 Discussion of using a Mixed Method Methodology for this Study

The quantitative research is validated by considering the reliability and validity of the instruments used. These criteria are well-established for assessing quantitative research; and are less open to interpretation having dominated the field of research study (Anney, 2014). Quantitative research is positivistic with only one reality converging as a result of the research and this reality can be divided into variables. The researcher is independent of the objects being studied and knowledge is gained through scientific methods producing an absolute generalisable truth. In contrast, qualitative researchers believe that there is no one reality, reality is divergent as their part of the inquiry that may or may not be related to each other. The researcher and the participants in the research either depend on each other or have a relationship that is influential. The research does not produce an absolute truth, inquiries cannot be generalised and are focussed on individual cases (Anney, 2014). By combining both quantitative and qualitative research methods in this study, the strength of the study will be improved although both methods need to be validated. Since they are both so different, the criteria for validating each methodology differ.

3.5.3 Quantitative Research — Principles to Test Validity and Reliability

The grit and mindset questionnaires that were used to assess the students needed to be validated. These tools allow for an objective means of collecting data about the student's attitudes and beliefs to their grit and mindset. Both tools were taken from existing questionnaires for mindset (Dweck & Blackwell, 2015) and grit (Duckworth & Quinn, 2009).

A questionnaire consists of a set of questions that are designed to collect and record data. There should be a clear purpose that is related to the purpose of the study and the use of results should be clear from the onset. The result of the questionnaire should be used quantitatively and is often associated with social research

(Bolarinwa, 2015). A questionnaire should be interrogated in terms of the validity and reliability of the research instrument. Validity is an expression of the extent to which the measurement tool sets out to measure. Validity consists of four types, namely face validity, construct validity, content validity and criterion validity, and are divided into internal and external components. Internal validity is how accurately the data obtained actually quantifies what the tool was designed to measure. External validity is whether the study can be applied or generalised to another sample, group or setting and produce similar results (Venkatesh, Brown & Bala, 2013). Reliability is the degree to which the measurement can be replicated with similar results. Reliability consists of three aspects — equivalence, stability and internal consistency (Bolarinwa, 2015).

According to Bolarinwa (2015), face validity involves experts looking at the questionnaire and either agreeing or disagreeing whether each of the items of the questionnaire measure what is required by the study. Face validity is considered to be casual and may not be an accurate measure of validity. Content validity measures the degree to which the questionnaire measures the construct of interest, in other words the measure to which extent the questions in the questionnaire represent the possibility of attitudes or behaviours in the problem domain. A content valid instrument would typically be evaluated by experts who are familiar with the research subject. A drawback of content validity is that like face validity, it is highly subjective. Criterion validity determines the relationship of the scores on the questionnaire to a specific criterion. This can be achieved by either assessing against a highly credible standard and is termed concurrent validity. An example quoted by Bolarinwa (2015), describes comparing the results of a questionnaire to eliciting information about a diabetic patients' blood sugar level compared a reading of blood glucose level for the patient obtained in a laboratory. Predictive validity, a subset of criterion related validity, is the ability of the questionnaire to predict future events and is assessed using a correlation coefficient. In this study the measure of grit and mindset are used to predict academic achievement.

Bolarinwa (2015) alluded to the fact that construct validity is the degree to which the questionnaire measures the theoretical construct it is intended to measure. It is

one of the most important but difficult to determine of the validity aspects. It is the measure of how meaningful the instrument is used in practice. Construct validity considers different types of evidence, namely 1) convergent validity, 2) discriminant validity, 3) known-group validity, 4) factorial validity and 5) hypothesis-testing validity and the type chosen is dependent on the research problem. Convergent validity is supported by evidence that the same concept will yield similar results if it is measured in different ways. For example, use of questionnaires and observations are two methods that could measure the same behaviour. Discriminant validity determines if there is evidence that one concept is different to concepts that are closely related. For example the exposure to TV programmes that are entertaining should differ to the exposure to TV health programmes. These concepts are closely related but are significantly different. In known-group validity the outcome of a group whose attribute has not been established is compared to a group with an established attribute of the outcome. For example, when exploring clinical diagnosis of depression of two groups of people, one group who is diagnosed with depression and one who has not. The construct of depression should score higher in the questionnaires for those who have been clinically diagnosed with depression that with those who have not (Bolarinwa, 2015). Factorial validity validates the contents of the construct using a statistical model called factor analysis. This method is appropriate where there are more than one dimension to the construct and each form different domains of the attribute. In the grit score, the dimensions of *passion* and *perseverance* relate highly to the overall grit score and to each other. Hypothesis–testing validity provides evidence that the relationship between one variable and the other variable described by the theory is supported (Bolarinwa, 2015). By determining whether there is a positive correlation between grit and mindset and PAT results, evidence can be provided to support the research questions of this study.

As previously stated, the reliability tests of a questionnaire determines the extent to which the instrument produces the same results in repeated sets of trials. It can also be described as to how stable or consistent the scores are over time or for different raters. This test does not apply to a person, but the score produced by the person rating the questionnaire, called a rater. For example, in a platform diving event, the

extent to which judges produce similar scores for the same dive would be an indication of reliability. Reliability can be determined using 1) test-retest reliability, 2) alternate-form reliability and 3) internal consistency reliability (Bolarinwa, 2015). Test-retest reliability provides an indication of stability over a period of time. Stability occurs when the scores are similar or the same after repeatedly testing the same group of respondents from one time to the next. Alternate-form reliability or equivalence is the extent to which two or more research instruments or questionnaires agree when measures at a nearly identical point it time. These could be two similar questionnaires that are worded differently that measure the same attribute or construct. Instruments are considered equivalent if there is a high degree of correlation between the two. Internal consistency, reliability or homogeneity is the extent to which items on the test are measuring the same concept. A test could be divided into two parts such as odd or even items and administered to the same group of individuals with responses being correlated. The advantage of such a test is that it eliminates having to test over a period of time. The reliability value will increase as the length of the test increases since the more items there are to measure the more reliable the scale becomes. However, it would not be effective to have extremely long tests so a point has to be found where the test is considered to measure the construct of interest as efficiently as possible (Bolarinwa, 2015).

3.5.4 The Data Collecting Instruments

In adapting the instruments used for collecting data in this study, the following were some of the guiding principles considered.

Grit Questionnaire. The grit score was validated by Duckworth (2009) although there were three versions of the grit scale. Initially she used the 12 point grit scale (Grit-O) which was reduced to the 8-point grit scale (Grit –S) (Duckworth & Quinn, 2009). The 12-point grit scale was divided into 6 questions relating to consistency of effort or *passion* and the other six related to *perseverance* of effort or *perseverance*. The 12-point scale included two extra *passion* questions "My interests change from year to year" and "I become interested in new pursuits every few months" and two extra *perseverance* questions "I have achieved a goal that took years of work"

and “I have overcome setbacks to conquer an important challenge”. These four descriptors were eliminated in the Grit-S score since they were considered to be similar to the 8 existing questions in the Grit-S score and duplicated the concepts being questioned. Her investigation consisted of six studies where she returned to the West Point Academy and the National Spelling Bee and included the Ivy League graduates in the first study. She concluded the Grit-S was a good fit in all three cases (Duckworth & Quinn, 2009). After removing the two questions from the two sub-scales of *passion* and *perseverance* the Grit-S scale demonstrated internal consistency across the samples.

In her second study, she investigated the factor structure of Grit-S over a large population sample of Americans, the relationship between the Big Five personality traits and the predictive validity for changes in career and attainment in education. She found support for the Grit-S scale across both *passion* and *perseverance* in a large sample of the population with little variation across gender. The relation to the Big Five dimensions of education, age, gender and career changes showed that the Grit-S correlation with conscientiousness was higher than any other Big Five Indicator for personality traits. Controlling for conscientiousness, individuals with more grit attained more education compared to other individuals of the same age. The Grit-S scores again did not have any significant difference in gender, although adults tended to be grittier than younger individuals which may be attributed to life experience. She determined that Grit-S has an inverse relationship with career changes where grittier individuals tended to have fewer changes in careers (Duckworth & Quinn, 2009).

In her third study, she validated the shorter scale using participants along with their friends and family whose purpose was to describe the individual. Each participant had supplied one friend and one family member using their email addresses as a link. The email address was used to ensure the participants were unique. The results suggested that the grit score of the informants of the participant was consistent with an individual’s own grit scale (Duckworth & Quinn, 2009).

The fourth study tested the Grit-S score using a sample population of high-achieving, middle and high school students. The study was longitudinal and tested whether

Grit-S was able to predict school grades and the number of hours watching television during the academic year. The study proved that Grit-S was stable over the time of the longitudinal study with scores that did not differ between the genders. The grit scores predicted the GPA and was inversely proportional to the number of hours watching television (Duckworth & Quinn, 2009).

Study five investigated the retention of candidates at West Point. West Point has rigorous admission criteria however many drop out. Grit predicted the retention of candidates better than their whole candidate score (Duckworth & Quinn, 2009).

The last study investigated the finalists in the National Spelling Bee testing the predictive validity of Grit-S to their behavioural measure of their performance. This test also determined whether the effect of grit on an individual's achievement was mediated by the cumulative effort of an individual. The results showed the scores on Grit-S predicted the individuals who score high on the Grit-S were 38% more likely to proceed to further rounds. The grittier individuals performed better that those with less grit as they had more practice in spelling and that experience in previous competitions was a mediator between grit and the final round of the Spelling Bee (Duckworth & Quinn, 2009).

In conclusion of her investigations she determined that Grit-S was a more efficient measure of *passion* and *perseverance* for long-term goals. *Perseverance* was a better predictor of GPA, extracurricular activities and predicted television watching among adolescent students inversely. *Passion* was a better predictor of career change in adults, the final round of the National Spelling Bee and the retaining of West Point cadets. She found evidence that *passion* and *perseverance* were both required to succeed in a challenging sphere. The total Grit-S score predicted the final round of the National Spelling Bee and the retaining of West Point cadets better than *passion* and *perseverance* on its own (Duckworth & Quinn, 2009).

Mindset Questionnaire. The mindset questionnaire used in this study was adapted from the Mindset Works website (Dweck & Blackwell, 2015). The original web site had 8 questions, where the odd numbered questioned described a growth mindset and the even numbered questions described a fixed mindset. Each question had

six possible responses, “Disagree a lot”, “Disagree”, “Disagree a little”, “Agree a little”, “Agree” and “Agree a lot”. Questions testing a growth mindset such as “No matter how much intelligence you have, you can always change it a good deal.” or “I like my work best when it makes me think hard” scored a maximum for “Agree a lot” ranging from 0 for “Disagree a lot”. Questions relating to fixed mindset were scored in reverse. Various forms of the mindset questionnaire have appeared on the Internet with some going to as many as 20 questions. The questionnaire used in this study consisted of 10 questions with a four-point scale from “Strongly Agree”, “Agree”, “Disagree” to “Strongly Disagree” each related in pairs.

Table 1. Mindset Questionnaire

Question Number	Question
1	No matter how much intelligence you have, you can always change it a good deal
2	You can learn new things, but you cannot really change your basic level of intelligence
3	I like my work best when it makes me think hard
4	I like my work best when I can do it really well without too much trouble
5	I like work that I'll learn from even if I make a lot of mistakes.
6	I like my work best when I can do it perfectly without any mistakes
7	When something is hard, it just makes me want to work more on it, not less
8	To tell the truth, when I work hard, it makes me feel as though I'm not very smart
9	If I cannot solve a problem quickly, I give up easily
10	I like to work on problems even if it takes a long time

Question 1 was a negative rewording of Question 2. Question 3 and Question 4 were similar statements that were not a direct rewording of each other. Question 5 and Question 6 were negative rewording of each other. Question 7 and Question 8 are related, but not a negative rewording of each other. Question 9 and Question 10 are also related but not a negative rewording of each other. These slight differences

in the question pairs may prove to produce a questionnaire that is not valid or reliable. The mindset questionnaire contained the first 8 questions ("What 's My Mindset ? Mindset Questionnaire Scoring," n.d.); I added the last two questions as an attempt to improve the validity of the questionnaire. The validity of the both the mindset and grit questionnaires will be evaluated in the next chapter.

3.5.5 Qualitative Research — Principles to Establish Credibility

The credibility of qualitative research is the extent to which the research findings inferred from the original data obtained from the participants is correctly interpreted from the participant's original views (Anney, 2014). Credibility can be achieved through the following means: prolonged engagement, reflexivity, time sampling, triangulation, establishing the authority of the researcher, member checking, peer examination, interview technique and structural coherence.

According to Anney (2014), prolonged engagement involves the researcher becoming part of the participants' world to gain awareness and understanding of the context of the study. This should reduce the distortions or bias resulting of the researcher being present in the study. Peer examination provides the researcher with support from other professionals who can provide guidance for the researcher on an academic level. This can be done by presenting their findings for comments and reviews from professional peers. Triangulation enriches the research by using a variety of methods, sources and theories to corroborate the evidence; it helps to reduce the bias of the researcher and verifies the truthfulness of the responses given by the participants. Triangulation can be achieved in three ways (Anney, 2014). The first is to use multiple investigators to research the same problem thereby strengthening the integrity of the findings by incorporating multiple perceptions. The second is to use different sources of data or research instruments such as interviews, focus groups and questionnaires or a variety of informants thereby enhancing the quality of data obtained from different sources. The third is to use different research methods to ensure the quality of the data. Member-checking allows for the participants to be included in the process of analysing and interpreting the data. The participants are invited to evaluate the interpretations made by the researcher and can suggest

changes if they do not agree with the outcome thus removing any conflicts or inconsistencies. Should the results of the inquiry contradict the expectations of the researcher then a negative case analysis should be conducted which may result in a change in the research questions and improving the rigour of the study (Anney, 2014).

Transferability describes the extent to which the qualitative results can be transferred to other contexts with other participants. This can be done by reporting a detailed or thick description of the research process and the participants who took part in the study. A thick description allows for replication of the research process in similar conditions but a different setting. The sampling of the participants needs to be purposefully associated with answering the research questions. The participants may be chosen for their knowledge about the issue that is being researched, providing a richer data for the research (Anney, 2014).

A research study is considered as dependable if the findings and the interpretation thereof are supported by the data collected. Dependability can be established using an audit trail, triangulation, step wise replication, code-recode strategy and peer examination. The audit trail should provide details regarding the raw data, notes about interviews and observations made, and other documents and record that have been collected in the study and test scores. Stepwise replication can be achieved by two or more researchers analysing the raw data individually and comparing their results so that inconsistencies can be addressed. When coding the data in order to find meaning, the code-recode strategy requires the researcher to code the data and then after a couple of weeks, to recode the data. The two sets of codlings are compared to determine any discrepancies. Peer examination can be used in the same way as member-checking where a researcher discusses the strategies used in the research with someone experienced in qualitative research and is not involved with the study to identify any parts of the research that has not been addressed (Anney, 2014).

Confirmability establishes the degree to which the results of the research can be confirmed. The research needs to be clearly derived from the data and this can be done by the use of an audit trail, reflexive journal and triangulation (Anney, 2014).

The integrity of the research needs to be established to determine whether the data provided by the researcher were not fabricated by the researcher.

In the next chapter the details of the results and findings of the research are discussed and analysed in relation to the research questions.

3.6 Summary of the Chapter

The details of the research methodology were explored in this chapter together with the justification of using mixed methods. The importance of the process of mixed methods is explained in terms of producing reliable and valid research. In this study, quantitative data will be collected first and then a sample of the students were interviewed to produce qualitative data. Both methods of data collection were used to answer the research questions.

The study was described in detail by including the process of the study, the participants and the programming project. My personal journal was described the daily lessons that took place during the study to support the process described in the study. The principles of trustworthiness, reliability and validity of the research were explored to outline the requirements of quantitative research that is deemed to be valid. In particular, the research regarding data collection instruments were investigated with reference to the grit and mindset questionnaires. The credibility of qualitative research was discussed to explore the factors that should be considered when collecting qualitative data. The principles for both quantitative and qualitative research were explored to validate the credibility of this study.

CHAPTER FOUR – DATA COLLECTION, ANALYSIS, RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, I am going to discuss how my data was collected using the Grit-8 questionnaire and an adapted mindset questionnaire (called Mindset-8) as described in Chapter Three from the works of Duckworth (2016) and Dweck (2007). The questionnaires were administered online at three different times during the study. The PAT scores and number of submissions were collected at the end of the study to supplement the quantitative data. From the data collected, the data was analysed using the Pearson coefficient to determine whether a linear relationship existed between, grit and PAT, *perseverance* and PAT, Mindset-8 and PAT and number of submissions and PAT. The significance of the results were calculated using the *p*-value of the pairs of data. The Spearman rank correlation was used to investigate the similarities in the rank order of the pairs of numbers. Bootstrapping was used to resample the data to investigate the measure of accuracy of the results. The qualitative interviews were summarised and analysed for common trends that could relate to the research questions. Lastly, I will discuss the outcome of the analysis of both the quantitative and qualitative research.

4.2 Data Collection

The data collected in this study were of two types, namely: quantitative and qualitative. The students were pre-tested at the beginning of the project and then after they had completed their design and at the end when their projects were completed. The questionnaires were completed online using separate Google forms for the grit and mindset questionnaire. I chose the first test to serve as a pre-test before they started the project to produce a baseline value. The second test was administered after they have completed their design, which was one of the most challenging and frustrating parts of the project. If the design was incorrect, then the student could not proceed. The third test was a post-test at the end of the project. In completing each questionnaire students were reminded to the questionnaires as honestly as possible with time given in class to facilitate the completion. The questionnaires for

grit and mindset were separate and the URL was communicated to the students using a simplified bit.ly[4] address. The same two Google forms, one for grit and one for mindset, were used each time they answered the questionnaire. The forms recorded the pseudonym of the student and date/time stamp when they completed the questionnaires. The results were recorded in a Google sheet which was exported into Excel. Both the grit and mindset questionnaires produced numerical values to signify the level of grit or growth mindset in an individual. The grit measured as a score out of 5 expressed to one decimal place; and the mindset was expressed as a result out of 24 points which is then converted to percentages. The grit score was divided into the two dimensions of *passion* and *perseverance*. The scores for *perseverance* was obtained from questions 2,4, 7 and 8 and the score for *passion* was obtained from questions 1,3,5 and 6 of the grit questionnaire. Each sub-score consisted of 4 questions with a maximum result of 5 similar to the grit scores. A score for *perseverance*, *passion* and grit was recorded for each student. To obtain a single score for grit, *passion*, *perseverance* and mindset for each student, the three scores taken over the 6-week period of the study were averaged.

In addition, the number of submissions of each student was also recorded each time a student submitted their project to be assessed. The students kept their IEB mark sheet and each time their project was assessed, the score was updated on the mark sheet by the teacher. In most cases the teacher assessed the project in the presence of the student ensuring the student had immediate feedback detailing where they went wrong and what they could do to improve. The students' average *passion* score, average *perseverance* score, average grit score, average mindset score, their number of submissions and their final PAT result were recorded in a spreadsheet which is included in Table11 in Appendix #3. These variables were recorded to provide an answer to the research question of how grit and growth mindset is related to high school student's PAT results and in particular whether grit and growth mindset have impacts on student performance in their PAT. To answer the research question whether qualitative results explains the quantitative results, six students were

[4] A bit.ly address is a shortened, more user-friendly URL (Uniform Resource Locator)

selected and interviewed to further investigate mindset, grit, *passion*, *perseverance* and the number of submissions in relation to the students' PAT scores.

To perform statistical calculations the programming language R was used along with the more user-friendly RStudio front-end. The language provided functions to calculate correlation coefficients, draw scatter plots, determine lines of best fit and summary statistical data most of which was not available in Excel. The data stored in Excel was saved in csv[5] format and imported into R. A list of all RStudio scripts used to produce statistical results are listed in Appendix #4.

4.3 Analysis of Collected Data

The trustworthiness of this research study was achieved through a combination of factors. Prolonged engagement in the field was achieved by my teaching IT to grade 10, 11 and 12 students for twenty years. Each student has at least five IT periods a week requiring almost daily contact with their teachers namely myself and a colleague. We were able to observe how the students program, the processes they follow when they are stuck and have looked at the errors in an individual's code. The context of programming as part of a subject at high school level is an important element of this study. The complexity of programming cannot be underestimated, nor a student's reaction to learning this new skill. Over the years I have had many informal conversations with the students discussing their approaches to programming and how they feel about IT as a subject. The study comprised two classes, one taught by myself and the other by the aforementioned colleague. Whilst I did not teach the one class, I was often present when the lessons were delivered and consulted frequently with my colleague. We shared an open relationship where his input and opinions about the study were often discussed. The risk of this relationship was that his thoughts could influence the study although his participation in the study could be considered as a peer against whom I could test my insights into the

[5] CSV – Comma Separated Values file stores data in a package independent text file where each value is stored with a comma separating them.

research. His support and participation in the study provided a sounding board as the study progressed (Anney, 2014).

4.3.1 Quantitative Data Analysis

Both questionnaires were checked for internal consistency using the Cronbach alpha function provided by the R package. Beginning with the mindset questionnaire, the data from each student's three mindset questionnaires were combined using the median function in Excel. The median of each question of the three tests was calculated to produce a single result for each question for each student-participant. This was repeated for each of the 10 mindset questions and each of the 8 grit questionnaire results. In the case of the mindset questionnaire the median function produced results ending in 0.5 which were truncated. The results listed in the Table 12 in Appendix #3 were imported into R and the scales on the even numbered question were reversed. The values of the items ranged from 0 to 3; however, the inclusion of a 0 value for an item affects the results of the Cronbach alpha test (Bolarinwa, 2015). The values were adjusted from a scale of 0 to 3 on a scale of 1 to 4 by simply adding 1 to each value. The Cronbach alpha scored a low 0.41 indicating that the internal consistency reliability was low. If the last two questions were removed so that the questionnaire contained the original 8 questions (Table 13 in Appendix #3) from Dweck and Blackwell (2015), the Cronbach alpha for internal consistency increases to 0.51. This questionnaire with 8 questions will be termed Mindset-8 and the questionnaire with 10 questions will be termed Mindset-10.

Table 2. Comparison of the Mindset-10 and Mindset-8 Questionnaires

	Mindset-10	**Mindset-8**
Average	50.9	53.9
Standard Deviation	8.52	13.01
Maximum	72.2	84.7
Minimum	38.9	31.9
Cronbach Alpha	0.41	0.51

The average produced for the Mindset-8 questionnaire was 3 percent higher and had a greater standard deviation (SD) producing a greater measure of variety of

scores which is supported by larger maximum and minimum cores. The Mindset-8 questionnaire has a better validity than the original Mindset-10 questionnaire and the data produced from this questionnaire which ignored the last two questions "9. If I cannot solve a problem quickly, I give up easily." and "10. I like to work on problems even if it takes a long time." will be used in the quantitative analysis.

Besides using the Cronbach alpha calculation to validate the mindset questionnaire, I needed to find other forms of validation for the mindset questionnaire. Dweck refers to the mindset questionnaire extensively in her studies, in her book and in her web site (Blackwell *et al.*, 2007; Dweck, 2006; Dweck & Blackwell, 2015).

The grit questionnaire scored higher (listed in Table 14 of Appendix #3) on the Cronbach for internal consistency. A value of 0.80 was achieved, which affirms the reliability of the test. None of the questions were reverse-worded and the scale of the test began at 1, making the analysis easier to perform. Duckworth had asserted the validity of the 8 point Grit-S scale which attested to the quality of the questionnaire.

4.3.2 Qualitative Data Analysis

The mixed methods approach has allowed for triangulation of both quantitative and qualitative data to answer the research questions. The data from the quantitative research was investigated from the qualitative interviews. The participants interviewed were chosen from among those who provided the quantitative data. The interviews were semi-structured. The interview with each participant took place late in November of 2015 whilst the students were writing formal end-of-year examinations. The participants were invited to participate in the interview at a time that was convenient to them. Each interview was semi-structured and followed the questions listed in Appendix #5. These question were asked in more or less detail depending on the participant's responses. The participants may have found the interview uncomfortable as it is not customary for teachers to interview a student, particularly regarding their performance on a task. Some may have felt defensive or apologetic if they felt they were not satisfied with their PAT results. However, the

teacher (researcher) made concerted effort to make the student-participants as comfortable as possible. The interviews took place in the school's coffee shop where the students often spend time. Each interview had a clear purpose which was to determine the student's attitude towards *perseverance*, problem-solving, adhering to deadlines, what he did when he got stuck and his overall process that he followed. At this stage I had marked the projects of the participants in my class and the other class was marked by my colleague. For each student interviewed, I was aware of his questionnaire scores, his number of submissions and final PAT score. To put the student at ease at the beginning of the interview, I asked each student to describe his project enabling me to understand how he perceived the project. The students were animated when describing their projects regardless of the scores achieved. Each spoke about his process to problem-solving, their enjoyment of the task and what they learnt from the project. As a researcher, I gained insight into how each student approached the task. Their methods varied and may be related to their personalities and ease in asking for help as influenced by their background, though not part of this study.

Extended interaction with the participants could not be achieved as there are regulations regarding interactions with the students and, more importantly, the time frames dictated by the school's academic and extra mural program did not allow for a lot time to interact with students. Most interactions occurred during class. The students were given a significant amount of class time to work on their projects despite the class time being insufficient. Most settled into a routine of working at home and in class.

I did not perform member-checking by allowing the students to participate in the analysis and interpretation of the data. The students were not made aware of any of their grit or mindset scores during or after the study. I consider releasing the results to the students as a discussion before they matriculate at the end of grade 12 in 2016. I chose not to inform the students of their grit and mindset scores during the study so that they would not be influenced by their previous results. I chose not to inform them after the study as I was unsure of the consequences of the students knowing their results psychologically particularly since most were minors. Looking

at the data obtained I was concerned that the students were not capable of the level of reflection required to complete the questionnaires and by displaying these results to the students, they may lack the maturity needed to interpret their results positively. I discussed the results at length with my colleague and we both spent time analysing the results together. We both reflected on the process of the study and the significance of the results.

The study should be transferable to a similar context with participants of a similar description. The detailed description provided in section 3.4.1 describing the research design provides a thick description of the study supported by the journal description of the daily events.

The qualitative interviews once transcribed, were coded using a software program called AQUAD 7. Relevant line or line(s) of text was coded. The interviews were revisited frequently and the codes were refined. Initially codes were assigned to statements in the interviews such as "fix errors as coding" and "feedback" as listed in Version 1 of the Summary of Codes in Appendix #6. The initial codes gave an overview of the types of codes that were produced across all six interviews. The codes assigned were revisited and categories of codes were considered. If a code fitted into a category it was prefixed with the category followed by the code. Version 2 of the codes was produced by revisiting the interviews and the code categories. These codes were analysed and further condensed. The categories chosen were related to the *perseverance*, deadlines, code development, difficulties, documentation, emotional, errors, feedback, PAT, problem-solving (PS) strategy and tracing. The interviews were revisited and the codes were analysed and simplified. With each iteration the codes were refined until the final version, Version 6, produced categories for deadlines, difficulties, documentation, emotional, feedback, PAT, *perseverance*, programming, PS strategy, scaffolding and tracing. The choice of codes was discussed with the supervisors of this study. The final choice of category codes was allocated to determine the student's response to deadlines, scaffolding of the task and feedback. Emotional, difficulty, tracing, programming, PS strategy was assigned to record a student's response and strategy when they encountered errors. PAT and documentation were used to describe the student's response to the PAT

and the process of writing the documentation. Through this process of refining and categorising the codes created a deeper understanding of the patterns present in the qualitative interviews and their relationship to grit and mindset. The results of this study were frequently discussed with my supervisors as well as the interpretation of these results.

4.4 Research Findings and Discussion

4.4.1 Quantitative Data

As shown in Table11 in Appendix #3, the results of the PAT produced by the students were mostly above 80% with only 2 students achieving below 80%. The majority of the class (18 students) achieved 100% for the project. This narrow range of scores makes it difficult to differentiate between the student's scores.

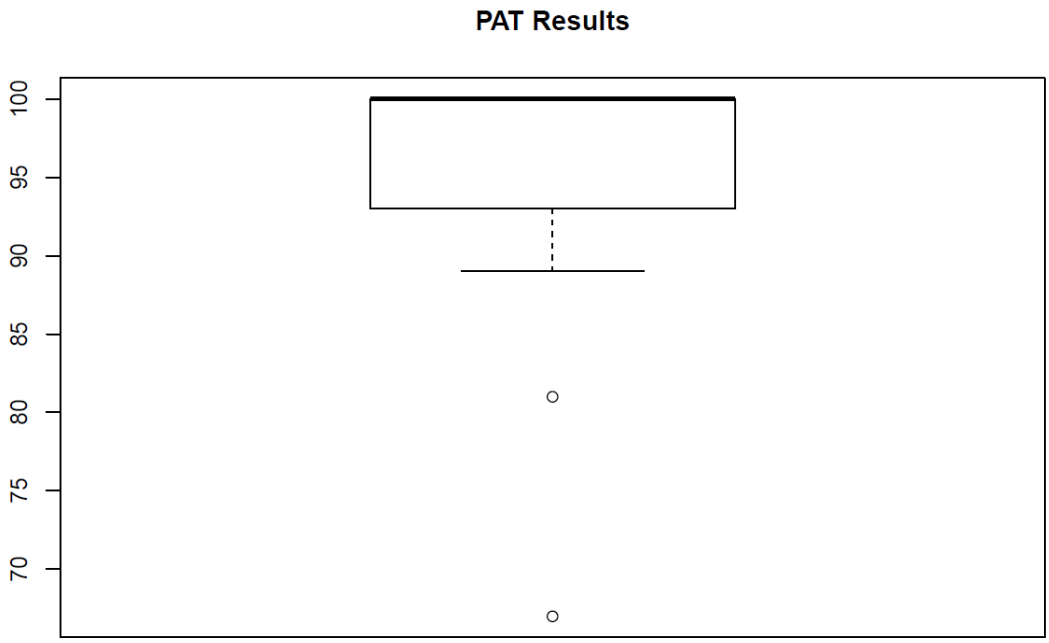


Figure 6. PAT Box Plot

Figure 6 displays the box plot of the PAT results showing the extent of the skewed results. The mean is 100 which is also the maximum result. Since the maximum and mean are the same, there is no upper quartile and the box plot is positively skewed.

This one-sided box plot shows two outliers with the minimum of 89. Table 3 summarises the quantitative data in terms of grit, Mindset-8, number of submissions and the final PAT scores.

Table 3. Summary Results of Quantitate Data

	Average	**Highest**	**Lowest**	**Std Dev**
Grit out of 5	3.4	4.3	2.3	0.56
Perseverance	3.8	4.8	2.4	0.66
Passion	3.1	4.2	1.6	0.65
Mindset-8	53.9%	84.7%	31.9%	13.01%
Number of Submissions	7.2	11	4	1.78
Final PAT Score	96%	100%	67%	7.28%

Grittiness. The average grit score was 3.4 out of 5 which does not initially appear to be particularly gritty. However, 3.4 is similar (the graph depicts a value between 3.4 and 3.5) to the average produced from 25 to 34 year old adults taken from a large sample of Americans (Duckworth, 2016). Although it may not be reasonable to use American adults as an effective comparison to 17 year old South African boys, considering the different culture and age, it is does provide a general indication that the class could be considered to be particularly gritty for their age. Duckworth states that grit improves with age and individuals with a higher level of education tend to be more gritty (Duckworth et al., 2007). When she breaks down her measurement of grit and graphs it against age, the 25 to 34 year old group have an average grit of between 3.4 and 3.5, the average grit value of the population sample lying between 35 and 44 years is between 3.5 and 3.6 which is a mere increase of 0.1 over a decade difference in age. With two decades difference in age, the grit average score improves by a further 0.1 lying between 3.6 and 3.7. A sharp increase occurs from the age of 64 years old and above reaching an average above 3.9. This signifies an approximate 0.5 increase in grit over three decade difference in age. Unfortunately the data produced by Duckworth does not include ages less than 25 years of age. It would be pure speculation to guess from the data provided what the average grit score would be for an average American in the 15 to 24 years of age

range. However, in South African, with among grade 11 learners age between 17 and 18 years of age the average grit score was 3.4.

At the time of writing, there is no available data for the average grit score of American students 15 to 24 years of age using a similar sample of Americans, although Duckworth (2009) does provide average grit scores of high-achieving, middle and high school students (N=279) in her longitudinal study using Grit–S to predict grade at school and the number of hours in a school year spent watching television. She also includes the average grit score of finalists in the 2006 Scripps National Spelling Bee with (N=190). Both studies coincidentally produced an average grit score of 3.4 which could be classified as high considered the participants in both studies were chosen for their perceived grittiness.

The grit score was further analysed by determining a value for *passion* and *perseverance* using related questions in the questionnaire. This average score for *perseverance* was greater than the average score for *passion* (Table 3). This indicates that most students were able to persevere more than sustain interest (as described by the term *passion*) in a topic over a long period of time. Only two participants had a *passion* score higher than their *perseverance* score and both of these student participants were categorised as having a fixed mindset with some growth. Both participating students achieved results above 90% in their PAT. Since this study took place over the short period of 6 weeks, the score for *passion* may or may not reveal long term commitment to a task.

Number of Submissions. The number of submissions was recorded each time students handed in their task to be assessed. The average number of submissions was 7.2 with a maximum of 11 and a minimum of three. Each student was required to hand in three submissions being the specifications, the project design and a technical document along with the working code. No student handed in less than the minimum three submissions. Each student was required to submit a specification document, a design document and a technical submission. The number of submission value does not directly reflect the *perseverance* of a student since a student will discontinue submissions once the task has been evaluated to achieve 100%, meaning completion.

Mindset.

Table 4. *Mindset-8* Tally

Mindset description	Tally
Strong growth mindset	2
Growth mindset with some fixed	8
Fixed mindset with some growth	18
Strong fixed mindset	1

The average mindset score was 53.9% with 2 students recording a strong mindset. Eight students recorded a growth mindset with some fixed mindset producing 10 of the 29 students being classified as having mostly a growth mindset. The majority of students participants (18) were classified as having a fixed mindset with some growth and 1 was classified as having a string fixed mindset. This range of classifications is given in Table 5.

Table 5. Mindset Classification

	Percentage
Strong growth mindset	75-100
Growth mindset with some fixed	56-74
Fixed mindset with some growth	35-55
Strong fixed mindset	0-34

The classification was adapted from a blog written by Emily Diel which used a mindset score out of 60 (Emily, 2008), this score has been converted to a percentage to accommodate my data.

Correlation Coefficients.

Table 6. Correlation Coefficients

	Grit	**Perse-verance**	**Passion**	***Mind-set-8***	**No of Sub-missions**	**PAT**
Grit	1.00					
Persever-ance	0.86	1.00				
Passion	0.86	0.48	1.00			
Mindset-8	0.41	0.44	0.28	1.00		
No of Sub-missions	0.24	0.26	0.16	0.47	1.00	
PAT	0.48	0.43	0.40	0.21	0.52	1.00

The correlation coefficients were calculated using R using the Pearson test. The Pearson test is used to analyse whether a linear relationship exists between selected variables such as whether a linear relationship exists between Grit and the PAT scores. A positive coefficient would indicate as grit increases so does PAT which would indicate the grit has a positive effect on PAT performance. The closer this value is to 1 would show the strength of this relationship. The significance values were determined by calculating *p*-values based on a one tailed test. A one-tailed test was chosen to determine the significance of the hypothesis that grit and mindset are related or even predicts academic performance in terms of the PAT results. If a linear relationship did not exist, the Spearman rank test was performed to analyse a monotonic relationship between the variables using the ranking of the variables.

The purpose of this study was to analyse the relationship between grit and mindset in predicting academic performance measured by the student's PAT scores. Five correlations were analysed: grit vs PAT, *passion* vs PAT, *Mindset-8* vs PAT, number of submissions vs PAT and *perseverance* vs PAT. With five correlations being analysed, the likelihood of these correlations being random forces a stricter *p*-value threshold than 0.05. The statistical significance of a test assumes that 1 in 20 or $p = 0.05$ is the threshold of accepting or rejecting the null hypothesis. In this study the null hypothesis would be grit and mindset do not have an effect on a student's PAT

scores. A *p*-value > 0.05 indicates that the null hypothesis may in fact be true, and the correlation may be produced by chance.

In this study, five comparisons or tests are performed, Grit vs PAT, *Mindset-8* vs PAT, *Perseverance* vs PAT, *Passion* vs PAT and Number of submissions vs PAT. Each is selected using a different variable to determine its effect on the PAT scores. With each additional test, the probability of a significant result simply due to chance increases, or to put it another way, the more tests that are performed, the more likely a result with a *p*-value < 0.05 will be produced. The Bonferroni correction was introduced to reduce the *p*-value by a factor related to the number of tests performed. This correction takes into account multiple testing and adjusts to the *p*-value to be correspondingly smaller. The Bonferroni correction is divides the *p*-value (α) by the number of tests. A *p*-value of 0.01 will be used which is a result of dividing 0.05 by the five, one for each test. The Bonferroni correction is very conservative and can produce strict significance levels. Since the population sample is small, two statistical procedures were applied to the sample: 1) the sample was resampled numerous times using bootstrapping and 2) the highest and lowest values were removed from a pair of variables, such as grit vs PAT, to recalculate the Pearson correlation and Spearman rank-order correlation.

Considering the small sample of the population, the correlation coefficients may fluctuate from sample to sample. The magnitude of these variances can be used to assess the margin of error through a technique called bootstrapping (Singh & Xie, 2010). This technique relies on resampling the data by creating many new estimate samples based on the existing sample by a process known as replacement. Random samples are drawn from the existing sample and the population statistic recalculated one thousand times. The purpose of the bootstrapping process is to estimate the bias of the resampling and report the standard deviation of the bootstrapped value. The bootstrapping was performed using the Pearson correlation coefficient since the most significant relationships between the variables appear to be linear. The results of the bootstrapping are listed in Table 8.

Table 8. Bootstrapping Results

Pearson Correlation	Original	Bias	Std. error
Grit vs PAT	0.48	-0.018	0.18
***Mindset-8* vs PAT**	0.21	0.0003	0.17
Perseverance vs PAT	0.43	-0.033	0.17
No of Submissions v PAT	0.52	-0.0138	0.14

The Pearson coefficient with its *p*-value and the Spearman rank-order coefficient was recalculated after the top and bottom results were removed from each set. The top and bottom results are not considered as outliers, they are merely the top and bottom. The point of performing the second set of calculations is to interrogate the robustness of the results. This procedure was not performed on grit vs *passion*, grit vs *perseverance*, *passion* vs *perseverance* and *passion* vs number of submissions since these comparisons were not directly related to the research questions. The results are shown in Table 7.

Table 7. Correlation Comparison

	Pearson r	*p*-value	Spearman r_s
Grit v PAT	0.48	0.007	0.34
Grit vs PAT Top and Bottom Removed	0.46	0.02	0.27
***Mindset-8* vs PAT**	0.21	0.28	0.26
***Mindset-8* vs PAT Top and Bottom Removed**	0.07	0.74	0.16
Grit vs Passion	0.86	0	0.84
Grit vs Perseverance	0.86	0	0.85
Passion vs Perseverance	0.48	0.007	0.45
Perseverance vs Number Submissions	0.26	0.17	0.19
Perseverance vs PAT	0.43	0.02	0.34
Perseverance vs PAT Top and Bottom Removed	0.24	0.23	0.23
No of Submissions vs PAT	0.52	0.003	0.43
No of Submissions vs PAT Top and Bottom Removed	0.48	0.01	0.34

Grit vs PAT. The correlation between grit and PAT was r=0.48, with the significance of this relationship producing a *p*-value = 0.007 showing a significant moderate positive relationship between grit and the PAT result. The Spearman rank correlation produced a weak value of $r_s = 0.34$ indicating that the relationship is closer to a linear relationship. With the top and bottom outliers removed, the Pearson correlation was reduced by 0.02 with a p-value = 0.007 and r_s remaining at 0.34. The bootstrapped result was slightly lower by 0.018 with a standard error of 0.18.

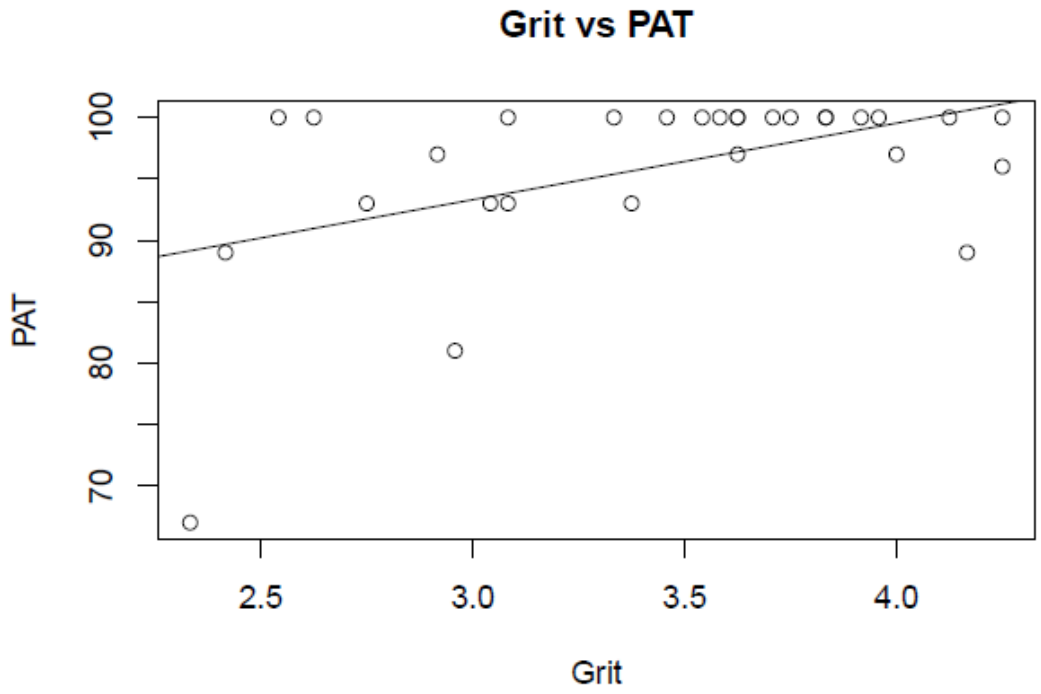


Figure 7. Grit vs PAT

Mindset-8 *vs PAT.* The correlation between Mindset-8 and PAT was a weak positive relationship with 0.21 with a significance of *p*-value = 0.28 which does not provide strong evidence for the effect of mindset on academic performance, furthermore the Spearman rank correlation is a weak $r_s = 0.26$. The top and bottom outliers removed, the Pearson correlation is significantly lowered to r=0.07 with a *p*-value = 0.74 and a weak $r_s = 0.16$, proving an even weaker relationship between the two variables. The bootstrapped results are similar to the results produced by the original sample with a low bias of 0.0003 and a standard error of 0.17.

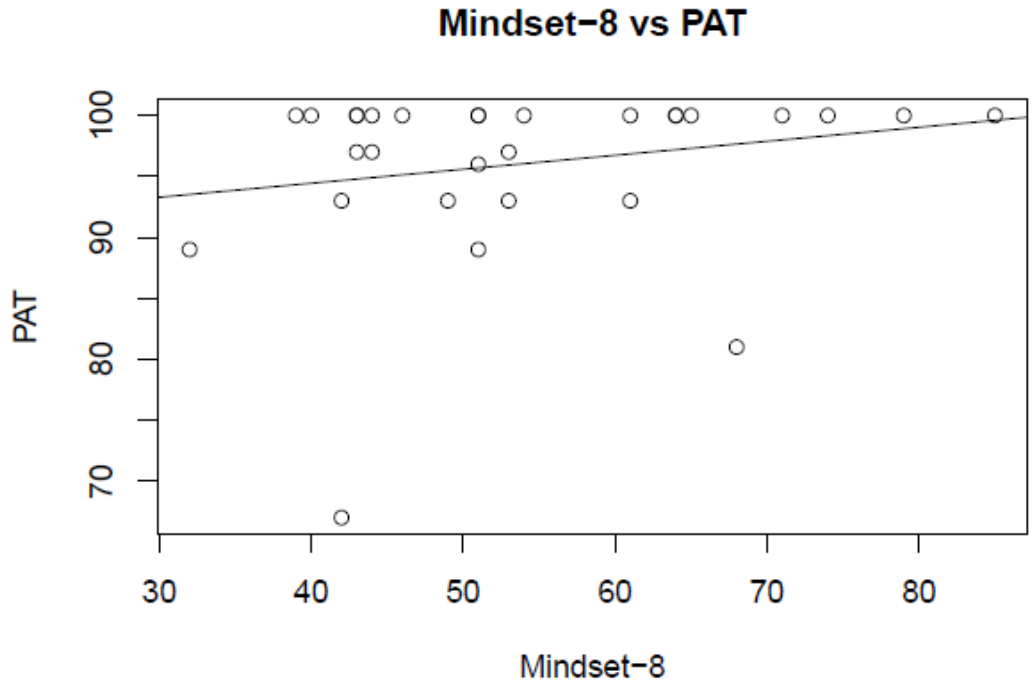


Figure 8. Mindest-8 vs PAT

Considering the grit contained two sub-scores of *perseverance* and *passion*. The relationship between these two components and the PAT score needed to be analysed. The correlation and between grit and *passion* and grit and *perseverance* was both high at 0.86 both with *p*-value = 0.00 with $r_s = 0.84$ and $r_s = 0.85$ respectively. This is to be expected as both are dimensions of the grit score. Unexpectedly, the correlation between *passion* and *perseverance* was much lower at 0.48 with a *p*-value = 0.007 indicating that students were less likely to have both *passion* and *perseverance*. *Passion* was defined by Duckworth (2016) as "consistency over time" which is the ability to stay focused on a task over time a considerable period of time. *Perseverance* was the ability to rebound from failure and work hard. I considered that *perseverance* would be closely related to the number of submissions since the number of submissions could be an indicator of *perseverance*. The more times a student submits his project the more they have persevered. The number of submissions was, however, restricted, because a student would stop submitting his project when he achieved 100% and thus could not be an accurate correlation for

grit. The correlation between *perseverance* and the number of submissions was even lower at a weak 0.26 with *p*-value =0.17 which is too high to be statistically significant for considering that students' *perseverance* will be related to the number of submissions. This result could be attributed to the number of submissions being constrained when the student achieved full marks for the PAT.

Perseverance vs PAT. The correlation between *perseverance* and PAT was worth analysing since the project was run over a short period of six weeks contradicting the definition of *passion* being consistent interest over a longer period of time. A moderate positive correlation of 0.43 was found between *perseverance* and PAT with a *p*-value = 0.02 which is above our threshold of 0.01 proving that this correlation is not significant. The Spearman rank coefficient was a weak $r_s = 0.34$ indicating a more linear relationship between *perseverance* and PAT. The Pearson correlation was reduced to 0.24 when the top and bottom outliers were removed with a *p*-value = 0.01 and $r_s = 0.23$. After bootstrapping, the bias was -0.033 showing a slightly lower result with a standard error of 0.17.

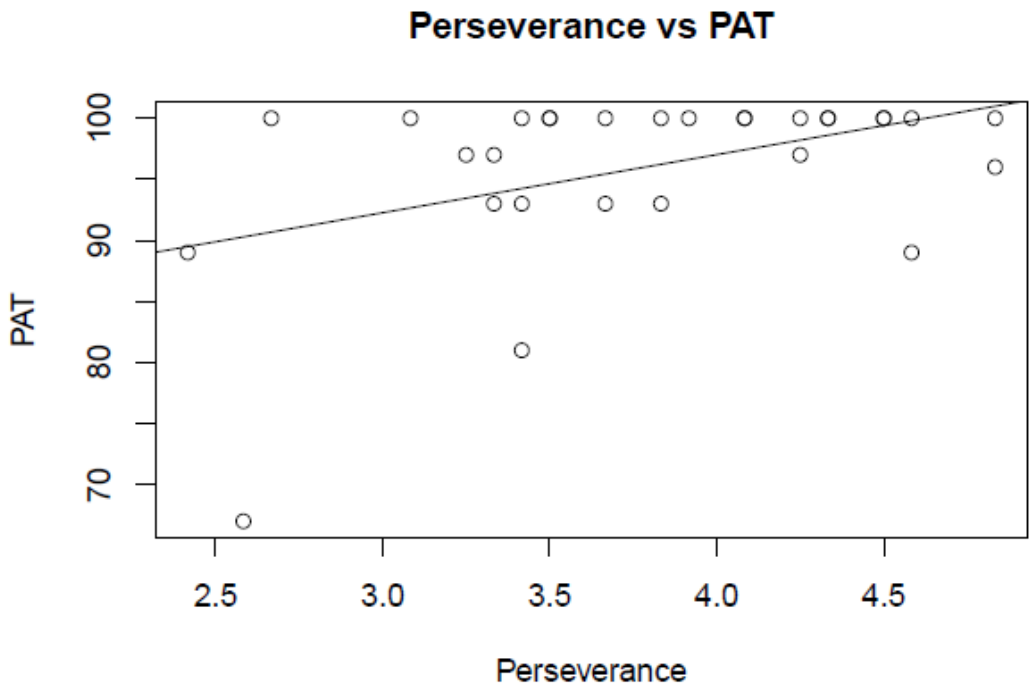


Figure 9. *Perseverance* vs PAT

Number of Submissions vs PAT. Lastly, the relationship between the number of submissions and the PAT score was determined to be 0.52 which proved to be significant with a *p*-value = 0.003. The Spearman rank coefficient was a moderate r_s = 0.43. The Pearson correlation coefficient was reduced by 0.04 to 0.48 with p-value=0.01 and r_s = 0.34. The bias resulting from bootstrapping was -0.0138 with a standard error of 0.14.

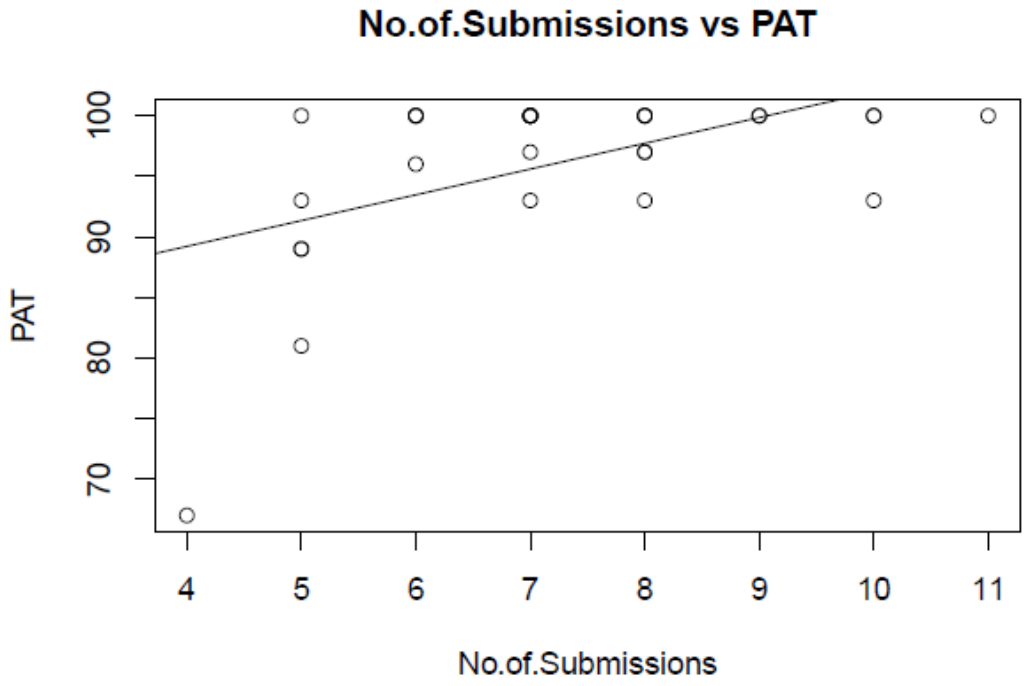


Figure 10. Number of Submissions vs PAT

These tests show that there is a moderate correlation between grit and performance in the PAT, *perseverance* and PAT and the number of submissions and PAT. The highest correlation lies between number of submissions and PAT indicating that the more a student submitted his project the higher his PAT result is likely to be. Whilst a higher correlation would have been desired, it is worthwhile to remember that the PAT result was skewed as most students achieved 100% and that the number of submissions stopped when a boy achieved full marks. Further investigation needs to be done regarding these correlations to gain deeper understanding of these relationships.

Six students were selected to be interviewed to gain their perspectives and understanding regarding the study. The details of these interviews will be described in analysing the qualitative data.

4.4.2 Qualitative Data

The above quantitative results indicate the strength of the correlation between the research variables in the study. This correltation does not imply causation, merely that a relationship exists. To further inverstigate and understand this relationship, qualitative data was used.

Six (6) student participants were selected and interviewed. They were selected by considering a range of grit, mindset and number of submissions. These values were plotted on a 3D plot and students whose data was represented on the extremes and in the middle of the scatter plot were selected. The purpose of the interviews was to understand the small correlation between the grit, mindset, number of submissions and PAT results. The students were asked questions to identify their process in producing the project, what type of errors they encountered, the strategy they used to correct the errors; whether they gave up or persevered when they encounter errors and lastly, what they would do to improve their projects.

Table 9. Results of Students Interviewed

Pseudonym	Grit	Perseverance	Passion	Mindset-8	Number submissions	PAT
Student A	2.6	2.7	2.6	39	6	100%
Student B	3.1	3.8	2.3	42	7	93%
Student C	4.2	4.6	3.8	85	5	89%
Student D	2.9	3.3	2.5	44	8	97%
Student E	4.3	4.8	3.7	64	8	100%
Student F	4.0	4.3	3.8	54	7	97%

The number of submissions was further investigated during the interviews. All the students interviewed encountered problems during the coding phase. Each student displayed *perseverance* by devising problem-solving strategies to correct their codes.

Summary of the Interviews

In the following section, I will briefly summarise the interview recorded from each student as an overview. In the following discussion, I will analyse the themes and patterns that have emerged in the interviews and discuss the extent to which these themes and patterns support the research questions.

Student A

Student A achieved 100% after 6 submissions, had similar scores for *perseverance* and *passion* with 2.7 and 2.6 and an overall grit score of 2.6. This student's mindset was categorised as fixed with some growth mindset with a score of 43. The student is a quiet boy who seldom participates in class activities. He is not a speaker of English as English is not his first language. Student A found that he enjoyed the project, but was not satisfied with his result. He coded a multi-level game with a path through which the user must navigate in order to obtain a reward. When the user encountered a different object the user was expected to perform a task.

Student A found the project was long and tiring and it took a long time to work out how to code the path which required research. The student used YouTube for research, planned algorithms to determine the logic of the program. When student A was stuck with the code he tried to diagnose the errors by writing algorithms or adding output statements to print out the values. This approach solved some of the errors, but not the big errors. The student did not ask the members of the class for help, which could be attributed to his shyness and language barrier. He found it embarrassing to ask questions which was why he did not use the WhatsApp group, although he did ask one particular student who he felt he could talk to. He was not comfortable with asking questions in class which was a problem he had experienced for quite some time.

He learned a lot from the process in terms of solving his own problems, particularly since the project was more than just homework. Student A spent one week not working on the project since he was considering changing the topic but was too shy to ask. The deadlines pushed him to stay on task and the Battleship example and

documentation templates were useful to help him achieve a high mark for the documentation. He wrote the documents before he would code a section, although he discovered his code did not always work like he planned in the documents which was when he considered changing his project. He found the previous coding learnt in the first half of the year a bit confusing and was struggling to apply the principles learnt to the project.

For his grade 12 submission, Student A will consider changing the path into something a little bit easier like a grid that allows the user to click on tiles in the grid. He liked the idea of the grid as he had seen the implementation of a gird with a 2D array in the Battleships example. He feels that in future he would start with an easier concept and should have asked for help when he was struggling with his topic.

Student B

Student B achieved 93% for his PAT after 7 submissions. His overall grit score was 3.1 with a *perseverance* score of 3.8 and a significantly lower score for *passion* (2.3). His mindset score was 42. Student B was involved in sport as an extramural which was fortunately not too onerous as he was not participating in any of the premier teams and his practises were limited in time and proximity. He is an easy going character who is not particularly passionate about IT as supported by his low *passion* score. Student B wrote a quiz game and was pleased with the aesthetics of his program. He manually typed in all the questions using his own data and did not consider searching the Internet for a possible downloadable set of questions. All the questions were stored in a database and he provided pictures to improve the aesthetics of the program.

He was not happy with the final result as he had not successfully manage to separate the code into a back-end that provided all the logic and front-end that provided the GUI with method calls to the back-end although his game did work well. He was embarrassed in describing his project in the interview which could be attributed to his perceived lack of expertise demonstrated in the PAT. He had managed to solve the problem but had not adhered to the design criteria. Once he realised he had made this error, he was not able to correct the design in the short time remaining. Since

the number of marks allocated to this design criteria was limited to 5 marks, in his case he was satisfied with his overall result considering the magnitude of the error and time required to fix it. He does comment that if had more time, he probably could have fixed the error. He programmed the PAT and documented simultaneously at the start, but then experienced his errors in design. Once he realised his error he chose to finish the game and then come back and fix the error. He patched up the errors in the design document by looking at code from past projects and using the Internet. His design document and PAT did not match he admits he "I kind of made up some rubbish" for his design.

Student B expressed frustration when fixing errors in his code as once he fixed one error then another would occur. He attributes this frustration in his inability to fix his design error. He admits to being an avid gamer and alternated gaming with coding his PAT. When he became frustrated "you fix one problem and the next one would come up and it just eventually got really irritating" with one he would switch to the other, although he found the gaming more frustrating as he was playing a new game against experienced players. He was able to code for long periods of time and achieve something before he switched to the game. When he was stuck in the PAT, he used Google and YouTube as resources to help him problem-solve. He asked his friends in his class to help him with his GUIs and they obliged by looking at his code and instructed him how to correct some problems. Student B was able to replicate the corrections throughout his code. He admits to sometimes not listening in class as a possible reason for not knowing how to code some sections.

Student B did not use the WhatsApp groups for help. He would first use the Internet then ask a person directly usually in class. He found trace tables and printing out the variables of a program tedious and frustrating. Instead he would ask a member of his family for help. This family member was not good at coding but was able to troubleshoot problems. Student B discovered that by explaining his logical errors he was able identify his own errors by talking through them. He did not experience many syntax errors compared to the logical errors. When he did experience problems, he would try to fix the error quickly and if that did not work he would "chill" and come back to it later.

Student B found the deadlines and rubric helpful combined with the multiple submissions. The feedback from the rubric illuminated his errors and focused his attention to where he could make improvements. He edited the code from the Battleships project provided in class and he used the document templates as a reference for his documentation.

In future Student B would improve his design of his code by separating the front-end from the back-end. He would like to focus more on the rubric to be more strategic in fixing his project instead of submitting his code repeatedly in a trial and error fashion. He would like to add a timer to improve the scoring system. A user would achieve a higher score for answering in a shorter amount of time.

Student C

Student C achieved 89% for his PAT after 5 submissions. His grit score was 4.2 with a score of 4.6 for *perseverance* and 3.8 for *passion*. His mindset score of 85 is the highest among the participants. He is a quiet, dedicated student who enjoys the subject. He started his project with many ambitious ideas which were beyond his expertise. He was generally pleased with his PAT result and found the deadlines to approach rather quickly. He enjoyed the project and involved his family members in testing the code. He struggled with the documentation and tended to focus rather on his code. He was able to apply the lessons learnt in the first six months of the year and referred back to the work when necessary. His project was a riddle game where all the riddles were stored in a database. He spent a lot of time designing the distractors to the riddle questions with his family members providing input on the choice of riddle question. He built in a mathematical formula which randomly calculated whether the user could continue answering questions based on the number of previous correct answers.

He had a unique method for solving problems in his code. Each time he experienced an error he would code multiple solutions to the problem and then test which solution was in his opinion the best. He also used many output statements to print out any error he experienced. He preferred using output statements to a traceable as he

found trace tables to be too time consuming. He developed a method of taking random test values of variables and then working out what the result would be if the problematic section of code used this value. He also tracked the changes he made to the code on paper since he often lost track of the changes he made to his code. He recorded each change he made and the options he had for each change. Student B developed this strategy for coding in the second term of grade 11 but was often frustrated by the process of making and recording the changes. He described his process as building from the inside out where he would start with smaller parts of the problem and build up to larger parts. Since one of his major problems was keeping track of errors and versioning, he admitted that a versioning tool would have been a great help to him in keeping track of his project.

The deadlines and feedback with multiple submissions helped Student C in particular the feedback as it provided him with information regarding the level he needed to achieve. The feedback was detailed and helped him with areas like pseudocode and general documentation. Student C struggled to write the documentation that described the project and found coding the project to be easier. He found the section where he had to describe the sequence of his project particularly challenging. He persevered with his documentation, although he found it difficult, and gave up on perfecting the class diagrams conceding two or three marks in the design document.

Upon reflection, Student C found that by having an idea beyond his capabilities, it undermined his confidence when trying to code the solution. In future, he would advise to have ideas and then have backup ideas in case the original ideas don't work out. He would improve his PAT for grade 12 by ranking his riddles according to difficulty and changing the way the user entered their answers.

Student D

Student D achieved 97% for is PAT with 8 submissions and scored 2.9 for grit overall. His *perseverance* was 3.3, his *passion* was 2.5 and his mindset score was 50. Although Student D is a quiet student, he frequently asks questions in class. These questions are mostly directed to a fellow student who achieves extremely

high academic results. He enjoyed the project but found it challenging as he struggled with the programming. He liked the idea that he could code a game. He coded a memory game which allowed the user to select two tiles which could be flipped over. If the tiles were a match, then the tiles would disappear with the object of the game being to remove all the tiles. Student D struggled with programming and mostly experienced logical errors which he solved by writing down what he wanted the output to be and then checking the program to see if the output was as he predicted. If not, he would try to work out what the error was and fix it. Before he coded he would plan the solution in pseudocode. He used the online tracing in NetBeans to help him find and fix the errors even though he found the process time consuming. When he found an error, he would attempt to fix the error and if he was not able to, he would ask his friends or a family members who helped with fixing logical errors. He found that asking for help worked well and he was able to proceed.

The deadlines motivated him to finish the project and he found the multiple submissions to be beneficial. Each time he handed in his project, he used the feedback to improve so he could achieve a better mark. He felt the deadlines were reasonable and if stretched any longer he would have procrastinated. Student D found the documentation easier than the code and did the documentation before the code, although he lost 2 marks for the documentation on his final mark. If he changed anything in the code, he would make the related changes in his documentation. When asked about how he persevered with the project, Student D felt he could have persevered more and made the game a lot better and achieved full marks of the documentation. He progressed through the project by breaking it up into definable steps and working through them. He used some of the code provided in the Battleships example.

Student D would like to improve his game by using a bigger grid, adding themes with a more colourful display. He would also like to use the game to educate people.

Student E

Student E submitted his project 8 times and achieved 100%. His grit score was a high 4.3 with an extremely high *perseverance* of 4.8 and *passion* of 3.7. His mindset scored lower at 58. Student E is passionate about programming and is recognised for his leadership abilities. He takes part in many of the school's activities including sport and cultural activities. Student E coded a themed game where, due to global warming and other catastrophes, the user is required to grow a crop of four plants in a grid of two by four. Half the grid was in the sun and half was in the shade. The aim was to water the plants and move them in and out of the shade to prevent them burning or freezing. The object of the game was to grow the plants to maturity at which one could progress to the next level of a total of six levels. Student E found the game to be enjoyable. He appreciates IT as a subject. He likes being creative and the possibility that he could sell the game he produced. Initially, he was stuck for a topic for the game, but then came across the idea of farming. He enjoyed the challenges he experienced and the satisfaction of solving the problems. He worked out how to use 2D arrays on the Internet before it was taught in class following the advice of a grade 12 student IT in his sports team. Student E had a fairly good idea of how to code the project before he began. He also worked out how to use a timer (a concept not taught in class) using the Internet.

When Student E experienced problems he would use the Internet to search for a solution. He referred to Google as being his "best friend". Alternatively, he would print out the values in the problem area of the code to figure out what was causing the error. He preferred to use this manual method of tracing to that provided by NetBeans. Since he had a good understanding of his code he was able to easily identify where the error occurred. When asked if he ever walked away from the project out of frustration, he answered that he never did. If he could not work out how to fix the error he would comment it out and then redo the code in another way so he could fix the error. Student E did not ask his classmates for help as he did not think they were able to help him. He would have asked the teacher and if he felt the teacher was unable to help, he would turn to Google.

In terms of the deadlines and feedback he preferred being able to make many rough drafts which were marked with feedback being provided. He found the deadlines to

be reasonable, although if he had more time he would have introduced more concepts to his code. He designed and coded his project before he wrote his code and then worked backwards.

He found the templates for the documentation helpful in providing a structure format or layout to be followed. He edited the templates to produce his own documentation. He did not use the Battleships code as he already understood how to use 2D array which he found relatively easy to understand. He found the logic in the back-end easier than creating the aesthetics of the front-end. He describes himself as a perfectionist and wanted everything to look nice. He spent a lot of time working out how to place images on the buttons that were transparent and lining up the components to be correctly sized. He found the GUI to be "finicky".

He found the previous knowledge taught to be helpful in designing his data structure. He originally had many arrays that he combined into one array of plant objects which led to his separation of code. He found that by applying his knowledge form the beginning of the year he could improve the data structure

To improve his project, Student E like to add a game pause option, a rate the game option whereby the user can provide a numerical ranking of the game, improve the graphics, add in plant types or categories which requires a varying amount of water and sun.

Student F.

Student F scored 97% for his PAT after 7 submissions; he had a grit score of 4.0 with 4.3 for *perseverance* and 3.8 for *passion*. His mindset was scored at 51. Student F is extremely quiet and reserved; he has struggled through school and battles to finish his exams. He is a kind and gentle person who participated in school activities throughout the duration of the project. He has a pre-existing challenge of taking a bit longer to work through things and was assessed for extra time for IEB exams at the time of writing this paper. Student F found the task interesting, enjoyable and creative. He had a few ideas for the project, which he realised were too ambitious before he settled on coding a riddle game where the user was required to

solve a number of riddles in a limited amount of time. If the user achieved this objective they could move onto the next level accumulating points as the user proceeded.

Student F frequently experienced problems with his project. He frequently used Google on the Internet to search for solutions to his problems. If he could not solve his problem, he relied on his peers and the teacher. Most of his errors were logic errors since he could easily deal with the syntax errors by looking up the error on the Internet. After using Google he would try to debug the code by outputting the variables and tracing mentally He would then ask his friends for help with the logical errors if he could not fix them himself during class or using WhatsApp depending on the time of day. Finally, if he still could not fix the logical error he would resort to changing the problematic code. He found this method to be more expedient than wasting time trying to fix the code, allowing him to move onto the next part of the code.

He managed his time by using the class time to ask questions so that he could be more efficient at home when he coded. He tried to reduce the errors he experienced when coding by asking his friends and using the resources available at school during class. He also wrote his documentation at home.

He found the deadlines useful in keeping up to date with the task and prioritising other tasks. By submitting the specifications and design before the coded solution, he was able to focus on the design of his game. Any changes he made to the code, he went back and fixed in his documentation. The feedback combined with the deadlines provided him with a road map for knowing where was going, what he did wrong, and where he had to improve with the aim of getting 100%.

He found the Battleship example useful in separating the front-end from the back-end. He appreciated the concept of separation of code and understood the elegance of the design. He was able to apply the knowledge taught at the beginning of the year to his project and understood its relevance.

Student F would like to improve his project by fixing a logical error that occurred right at the end of the game which he did not have time to do. He would also improve the user friendliness of the program and possibly add a multiplayer option.

Discussion

Perseverance. The purpose of the interviews was to further explore the relationship between grit, mindset, the number of submissions and the resulting PAT score. Each of the students displayed some level of *perseverance* when they experienced errors with their code. No-one gave up as soon as they experienced an error. Some persevered more than others. Student A persevered by using algorithms and output statements, but did not ask his peers for help owing to his shyness and language barrier. In his words he had to "figure things out by himself". His lack of self-confidence could have attributed to his low grit score of 2.6 and a mindset score of 43. His number of submissions was below the average of 6; however, he achieved 100% for the task which would have limited his submissions and reduced the correlation between number of submissions and his PAT score.

Student B persevered to solve most of the errors, but ignored the main error of separation of code. When he discovered this error, he did not attempt to fix it. His number of submissions was similar to the average of the group at 7; but by his own admission he did not use the submissions strategically. He merely made changes and submitted on a trial and error basis without paying much heed to the detailed criteria on the rubric.

> "I would focus more on the rubric. I kind a looked at it briefly and was like ok this needs to be done, this needs to be done, get this done and then that was it. I looked at my rubric maybe once or twice and then every time I got something wrong I would look at it, fix it, hand it back in immediately and see if I could get that right. So it was a real big kind of trial and error kind of thing."

His number of submissions may have been inflated and hence also affecting the correlation between his number of submissions and his PAT result. Student B had

a large difference between his *perseverance* 3.8 and *passion* 2.3 score, which could explain his tendency to continue with the project even though it was not correctly done. His mindset score was the lowest of the sample interviewed at 42 which may support his lack of diligence in his project and his tendency to switch between tasks (his project and his game) when he experienced problems. Student B's lack of diligence in correcting his code to separate the front-end and back-end could be related to his low mindset score. Once he encountered a challenge he was not inspired to overcome it by working through the difficulty of redesigning his solution. "I tried to make my life a bit easier with the GUI". He tended to enjoy coding the easier parts of his program such as the visually appealing GUI which is more rewarding as opposed to the logic surrounding the more complex back-end. Student C had the highest mindset score of 72 and a high grit 4.2. He developed an exhaustive method of solving his problem where he considered and coded each possible outcome before selecting the one he considered to be best. This method was time consuming and required a great deal of patience and *perseverance* and could explain his below average number of submissions since he was busy coding multiple solutions. His continued effort in dealing with setbacks is supported by his high grit score and his high mindset score indicates his ability to work on something that was hard and challenging. Student D had a below average grit score of 2.9 and a mindset of 50. He found programming to be a challenge and was not confident in his ability. He submitted his PAT 8 times, which was above average. An indication of Student D's grit was that he was one of the few students who persevered with the tedious process of tracing his program using the NetBeans tool. This tool is complex to use and painstaking particularly when he had many classes. Student E who is an extremely adept programmer did not give up when he encountered errors and managed to solve his own problems. He is intensely passionate about programming which is not reflected in his *passion* score of 3.7, although he has a high grit score of 4.3 with a near perfect score of 4.8 for *perseverance*. Student F has a suspected learning disability and has shown evidence of not finishing exams within the time allocated. He used the deadlines to manage his time, but had to choose to let errors go when he was aware of time running out. He submitted the PAT 7 times and scored 97%,

which could be interpreted as evidence of his *perseverance* considering his challenges regarding the pace of work. His grit score was a high 4.0 which supports his continued effort in the task. He used his time strategically well by allocating class time for seeking help for his errors and using time at home to implement the fixes he had learnt and writing his documentation which he could do independently.

Problem-solving. The problem-solving strategy chosen was personal and peculiar to the students. Each student worked through a variety of strategies to solve their problems and in differing ways. Most used the Internet as the port of call by using Google, YouTube or other online resources. The Internet is the most accessible resource to students particularly when they are at home and late at night. Student A, C, E and F preferred to use output statements to locate errors. This method takes time and patience, but is not as tedious as using the debugger tool in NetBeans. Student D persevered with the NetBeans debugging tool and Student B found both processes frustrating and turned to a family member for help. Both Student A and Student E did not ask their classmates during class or use the WhatsApp group for help, but instead traced through their programs to find their errors. The reasons for these choices was Student A's shyness and Student E did not want his classmates to think he could not solve the problems. Student E is a high achieving student who has academic, sporting and cultural colours and was selected to be a leader in the school. Student B did not use the WhatsApp group preferring to talk to people face-to-face. This may indicate a reluctance to explain the errors using WhatsApp, or a reliance on his peers to fix the errors. "When I was at school, then I didn't have to phone them and they could almost do it for me but show me at the same time". His use of a family member to talk through the logical errors was unusual, but appeared to work effectively. His avoidance of the WhatsApp group and use of friends and a family member to solve his problems may point to a form of extroversion and comfort among people. Both Student D and Student B engaged friends and a family member for help in unravelling logical errors, although Student D first attempted to determine the error through lengthy online tracing and writing algorithms. Student E, who described Google as his best friend, used a process of commenting out the erroneous code once having identified the source of the error and then rewriting the code in another way to solve the problem. His ability to code allowed him to quickly

identify errors and then correct them. Student F, who was challenged by time, used the Internet, printed out variables, the WhatsApp group and his friends in class to support him when he was stuck.

Deadlines. All the students appreciated the deadlines and found them motivational. Student A kept to the deadlines even though he was unsure of his topic. He persevered even though he knew the topic was problematic. This can be attributed to his grittiness in that he persevered with the task despite his uncertainty. Student B kept to the deadlines, but frequently procrastinated while playing his game. He could have used this time to fix his separation of code error. Student C found the deadlines to occur faster that he anticipated. "It's more just the how quickly the deadlines approach on you" however, he used the deadlines of each submission to gain feedback and then improve his work. Student D found the deadlines to be motivational and a mechanism to improve his marks for the next submission. "It gave us like a, a motive to finish it…. so, and also being allowed to hand it in multiple times, also allowed us to get better marks". Student E demonstrated his thorough approach to the project by creating multiple drafts which he used to be assessed and improve.

> "I did a lot of rough drafts and stuff and I had like three or four for each little thing and I like that you could go and hand it in, and it gets they like look at it and mark it and you could come and change your stuff,"

Student F used the deadlines to prioritise his other school task and stay on track. "…when I find when I have deadlines and it helps me keep up to date with what I need to get done and it also helps me separate … what certain tasks I need to do first." The adherence to deadlines is linked into an increase in motivation shown by the students' response by improving their work after each submission. This demonstrates a growth mindset where students are willing to persevere by resubmitting their projects.

Use of templates. Students A, D and F made use of the Battleships example and were able adapt their code using the example as a starting point. The documentation templates were useful in helping the students structure their documentation, although the order in which they coded and documented was varied. Students who

were more comfortable with code tended to code first and document later while others documented first. No student produced all the documentation before coding the solution. Most did it in parallel with a tendency of either coding or documentation running slightly before the other depending on their perceived strengths. The manner in which the students coded and documented could highlight their mindset and grit. By analysing each student's approach to what they found difficult (coding or documenting), and the order in which they worked on the tasks they found challenging, an insight into their mindset and grit can be obtained. A fixed mindset prefers to do something that is easy and would try to avoid challenging tasks. If a student is more comfortable with coding, then the act of coding would not be perceived as challenging and student would more likely gravitate to what is easier to do, in this case coding, and can be achieved quickly.

It is worth examining what the students did with the sections they themselves found challenging. Student A faced many challenges in terms of his shyness and language barrier. He discovered after writing his documentation and attempting the code that the code did not work as he planned in the documentation. He considered changing his project. In his situation the challenge would have been to approach the teacher and alter his project, which he chose not to do. Instead, he did persevere to complete the project to achieve full marks. Student B demonstrated a fixed mindset by avoiding challenges. He did not correct his separation of code error and even found the process of using output statements or tracing to be too tedious. He opted to explain the code to a family member to help him find his errors. The development of the back-end would have been a challenge for him, which he avoided by focussing on the graphical aspect of the front end which was visually rewarding. He spent a good deal of time typing out he questions of his board game, but did not consider searching the Internet to download the question which he would be a process that was unfamiliar to him. Student C found the code easier to the extent that he developed a complicated system of coding multiple solutions to errors. He found the documentation a challenge, but persevered although he lost 8 marks for documentation. For his code, he achieved full marks. One could argue that he could have used his time more strategically by spending less time of the code and more time on the documentation; however, but one has to appreciate the extent of his *perseverance*

in not only solving his errors but checking whether the solution he had chosen was actually the best. Student D found the code challenging. He experienced many errors, but systematically worked through the process of finding his errors by writing algorithms, using output statements and even using the tedious tracing tool in NetBeans. Student E prepared himself for the project by ensuring he had acquired the knowledge and skills of using 2D arrays before the project began. His coding ability was excellent, but he also ensured that his documentation was perfect to score full marks. He spent a lot of time perfecting his GUI to a level beyond the requirements of the task. Student F was facing possible learning disability challenges (as previously mentioned) and struggled to complete tasks on time and balance the load of other subjects. He struggled to code and also worked through a process of tracing, using Google, and output statements to identify and fix the errors. If he could not fix the error he would rewrite the code as a last resort, He was diligent in synchronising any changes in his code to be reflected in his documentation which requires care and thoughtfulness.

Improvements. When asked how each student could improve their project, the students were all able to identify areas that could be improved for this year's submission and for next year. Students A would simplify the maze of consider a change in topic, Student B would "focus more on the rubric", correct the back-end problem, add a timer and improve the points system. Student C identified improvements by stating he "would rank my riddles according to difficulty" and "change how …answers are entered in". Student D would like to increase complexity by adding themes and colours and increasing the size of the grid. Student E would add a pause button and improve the graphics. Student F would have liked to correct a logical error he found at the end of the program and add multiplayer functionality. Considering each student's intention to improve their projects, it is worthwhile considering how each student fared in grade 12. In grade 12, each student was required to add more functionality to their project and the project was marked out of 100 as opposed to the reduced 70 in grade 11. In short, a higher standard was required for grade 12. Comparing the results in Table 10, each student, with the exception of Student A and Student D, either maintained their result or improved slightly. Student D's result was a slight reduction of 1% and Student A achieved 5% less than his grade 11

result. In general, each student maintained a similar result the following year. This could be interpreted a demonstration of *passion* where the student's interest was "consistent over time" (Duckworth, 2016).

Table 10. Grade 11 and Grade 12 PAT Scores

	Grade 11 PAT	Grade 12 PAT
Student A	100	95
Student B	93	93
Student C	89	89
Student D	97	96
Student E	100	100
Student F	97	100

The qualitative data revealed that each student A, C, D, E and F showed a greater level of *perseverance* and growth mindset than Student B. The way in which they persevered was personal and based on their strengths. For example, if the student found coding easier than documentation, then they coded first and then worked out the documentation. Student B tended to avoid challenges and even procrastinated. Student E was exceptional in his *passion* for the subject and is intent on studying the subject further and will enter an IT competition with this project in 2016. Student A and F were challenged with language or learning barriers, but both endeavoured to produce good PAT results.

Students A and E achieved 100% for the project, with the lowest score being 89% achieved by Student C which all could be considered excellent results. The students who did not achieve 100% expressed regret or willingness to attempt to achieve 100%. This desire to achieve high marks could be attributed to the classroom culture of excellence, matched by the teacher's high expectations.

4.5 Summary of the Chapter

The details of how the quantitative and qualitative data was collected was described and analysed. The quantitative data was collected using questionnaires, recording each student's number of submissions and final PAT score. The questionnaires were checked for internal consistency using the Cronbach alpha function. The grit

questionnaire was considered to be valid and reliable with a result of 0.8 and the mindset questionnaire was reduced to 8 questions based on the Cronbach alpha result of 0.51 compared to 0.41 using the original 10 questions.

The correlation between grit and PAT, *perseverance* and PAT, *passion* and PAT, Mindset-8 and PAT and number of submissions and PAT was investigated. Since the relationship between a variety of variables was investigated, a Bonferroni correction was used to decrease the *p*-value to 0.01. Whilst the PAT scores and the number of submissions were slightly skewed, the correlation between PAT and number of submissions was the highest at 0.52 followed by grit vs PAT (0.48) and *perseverance* vs PAT (0.43). These three comparisons were deemed to be significant with p-values < 0.01.

The qualitative data analysis supported the quantitative data in providing evidence of *perseverance*, grit, and growth mindset. The students demonstrated their *perseverance*, adhered to deadlines, were able to problem-solve, and were able to reflect on future improvements.

CHAPTER FIVE – CONCLUSION

5.1 Introduction

This research sought to investigate the relationship between grit and mindset and academic performance in a programming context among high school students. A mixed method study was used to gather both quantitative and qualitative data. A classification of grit into *perseverance* and *passion* was used to determine that this study was essentially about measuring a student's *perseverance* when coding a programming project. The sub-scale of *perseverance* was recorded using questionnaires and the number of submissions of each student was tracked to gain numerical data to describe *perseverance*. The correlation between number of submissions and PAT was the highest at 0.52 which was proven to be significant with a *p*-value 0.002. To further understand the relationship between *perseverance* and PAT, six student participants were selected and interviewed. These six students provided a more comprehensive view of their *perseverance* in response to the problems and errors they encountered. This was demonstrated in the emergence of common themes, namely: adherence to deadlines, multiple submissions, problem-solving strategies and ideas for future improvements. The results of this study reveal that there is a relationship between *perseverance* and academic performance measured in the form of a programming project.

The results of this study implies that a non-trivial programming project, structured with deadlines, regular and thorough feedback, a detailed rubric that follows software engineering principles, students will be provided with the opportunity to demonstrate *perseverance* resulting in improved academic performance. The study was performed in a context that combined these factors to create a situation where students could succeed.

The study is limited by the skewed PAT scores and the number of submissions. The size and demographics of the participants were a limiting factor. The use of deadlines may have obscured or at least influenced a student's *perseverance*. The

teacher's dual role as researcher and teacher could have introduced bias. And finally, the students who participated in the interview may not have been a good representation of the group.

In summary, future studies should conducted with a larger and more representative sample of participants. The sample should include both genders, a larger variety of academic abilities and both private and public schools. The culture of a "gritty" classroom could be investigated in terms of producing higher academic results. A student's *perseverance* could be more effectively measured by using more rigorous testing measures and the complexity of a student's problem-solving strategies needs to be further investigated. The limitations of this study will be discussed in more detail in the next section.

5.2 Limitations of study

Whilst this study produced some interesting findings, it was limited by a few key factors that may have skewed the results. The number of students in the study was small and the study took place in a male, parochial (Catholic) school in an affluent area of a metropolitan city in South Africa. The participants were selected to participate in the IT course. They were required to have achieved 60% in Maths in grade 9 and to have passed a programming test to be part of the class in grade 10. The culture of the class was one of high expectations for both the students and the teacher. In addition, the school has a high academic standard and visibly rewards students with academic achievement by the use of braided colours emblazoned on the school blazers to differentiate students who have maintained a high average across their subjects.

The use of deadlines may have influenced *perseverance*. All the students interviewed found the deadlines helpful in keeping them on track. Each student submitted their PAT the required number of times or more perhaps for fear of being punished according the school codes of conduct. This adherence to school rules may influence their *perseverance* and combined with the exposure in class to a culture of high expectations, the students have felt external pressure to conform to continue working long after they felt the need to persevere.

The students who were chosen to be interviewed may not have been the best representation of the sample group. Although the participants had a variety of grit and mindset scores, perhaps a better range could have been selected. For example, the student with the lowest PAT declined to be interviewed possibly due embarrassment over his PAT result. The students may not have been as truthful as possible in the interviews owing the researcher, who is the head of department of information technology, conducting the interviews. The interviews could either be conducted by my colleague or we could have interviewed each other's students so the students were not interviewed by their own teacher. This may have allowed the students to feel more comfortable when discussing their shortcomings in their PATs.

By having a dual role of researcher and teacher, there may have been bias in conducting this study. The researcher's own personal involvement in the study may have been influenced by a desire to determine that grit and mindset have positive effects on academic performance in an information technology class.

5.3 Future Work

This study has produced results that are of interest, but need further work to create a better understanding of the effects of grit and mindset on academic performance. Since the study involved a limited number of participants, it could be expanded to include a large sample. This sample could extend across more than one grade from grade 10 to grade 12 seeing as a PAT, in the form of a programming project, is required in information technology each academic year. Although the PAT would change in academic rigour from one year to the next, the relationship between grit and mindset shaping academic performance could be measured. To increase the sample size significantly, the sample should include participants from both private and government schools in metropolitan and rural areas. It must also include members from both genders in single sex and co-educational environment. Information technology is a subject that attracts mostly male students and the effect of grit and mindset on academic performance and/or activity may be differ in female students, although this is not supported by the evidence supplied by the IEB matric results over the past three years (Sidiropolous, 2016). Using a larger population sample in South Africa, a database could be created to track the grit score and mindset scores

of students across, grade, gender and culture. The data could reveal what the average level of grit would be for a 16 year old female student in a government school with a particular cultural background. These values could act as indicators for students and teachers who perform the test to gain an understanding in measuring themselves against the norms obtained from a large population sample. Tracking could also take place for an individual over time. The grit and mindset could be measured going into high school at grade 8 level and then measured again at grade 12. This evidence could be used to predict academic performance and even plan interventions. However, care has to be taken not to turn this into how Intelligent Quotient has been used in the past to classify and marginalise groups of people.

The culture of the classroom, the school and the cultural background of students has effects on their grit and mindset. If the classroom culture can be measured and considered to be gritty, it could affect the grit score of each student in the class. Furthermore, the individual cultural background of each student may predetermine the grit and mindset of that student. This database could be extended to include data from other countries and cultures. It may be interesting to compare how a South African students measures up to a West Point candidates of the same age, gender, cultural and socioeconomic background.

The study could be developed by using a sample that has a wider range in academic abilities and with no prerequisite to gaining access to the information technology course. If a larger range of students' ability were used in the sample then a closer correlation may exist between grit, mindset and academic performances. Using the data obtained in investigating the effect of grit and mindset on performance, an intervention could be planned to influence the grit and mindset of the students in the hope that an increase in academic performance would be produced. Such a study would require that the participants are provided with lessons explaining the effect of grit, *perseverance* and mindset on academic performance and how both can be grown. These lessons would need to be meticulously carried out so that students can gain a thorough understanding of the effect of grit and mindset on their work. These lessons need to run concurrently with lessons being taught and the project should span more than one academic year. Issues relating to dealing with failure,

perseverance, *passion* and embracing a growth mindset need to become the common and everyday language of the classroom. Effort needs to be praised above results so that students will be comfortable with their errors and feel encouraged to continue trying.

More works need to be done in the measurement of validity of the grit and mindset questionnaires in a South African context with high school students. The Grit-S scale has been validated by Duckworth using American students many who are from elite schools (Duckworth & Quinn, 2009). The grit scores of South African students across a range of ages and cultures would help in providing an index to determine the grittiness of a South African student. The large variety of mindset questionnaires creates confusion along with different number of options for each question. A single validated mindset questionnaire with similar data for a South African population would be helpful in determining a consistent mindset score.

The *perseverance* of a student could be measured more definitively by setting up an experiment where a student is required to debug a program. The program will contain many errors and will need many attempts to eliminate all three syntax, run time and logical errors. Each time the student attempts to eliminate an error, the number of attempts will be increased. The purpose of this experiment will be to determine how many times the student tried to solve the errors, not whether the student was successful. The student would be given time to complete the test and will have the option to stop when they felt necessary. The test would need to ensure that the errors could not completely be eliminated in the period of time. The data collected would be a quantitative description of *perseverance*. The hypothesis of such an experiment would be that students with more grit would have a higher number of attempts in debugging the code.

Further work needs to be done into the problem-solving strategies developed by the students. Since no problem-solving strategy was formally taught, there is clear evidence that the problem-solving strategies were developed independently by the students and each were varied. While students may adopt the problem-solving strategy demonstrated by the teacher, more work needs to be done in investigating teaching formal problem-solving strategies to students. This study has indicated that

problem-solving strategies are personal and possibly linked to the personality of a student. This concept of personality influencing problem-solving strategy needs to be clarified through research. In addition, it may be determined that teaching a general one-size-fits-all problem-solving strategy may not be beneficial to all students. It may prove that rather than teaching a single problem-solving strategy, a more customised solution may be required that fits each student's needs, personality, culture and prior learning.

5.3 Conclusion

This study aimed to determine the effect of a student's grit and mindset on their academic performance using a mixed method research technique. A significant statistical correlation was found between grit and the PAT scores and *perseverance* and PAT scores. This relationship was further investigated by interviewing a selection of the participants of this study. The results of the qualitative data collected provided further evidence of grit, growth mindset and *perseverance*. The relationships between these variables needs to be further investigated with the intention of improving academic performance.

REFERENCES

Anney, V. N. (2014). Ensuring the quality of the findings of qualitative research: looking at trustworthiness criteria. *Journal of Emerging Trends in Educational Research and Policy Studies*, *5*(2), 272–281.

Ben-Ari, M. (1998). Constructivism in computer science education. *ACM SIGCSE Bulletin*, *30*(1), 257–261.

Blackwell, L. S., Trzesniewski, K. H., & Dweck, C. S. (2007). Implicit theories of intelligence predict achievement across an adolescent transition: a longitudinal study and an intervention. *Child development*, *78*(1), 246–63. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/17328703

Bolarinwa, O. (2015). Principles and methods of validity and reliability testing of questionnaires used in social and health science researches. *Nigerian Postgraduate Medical Journal*, *22*(4), 195. Retrieved from http://www.npmj.org/text.asp?2015/22/4/195/173959

Bovée, C., Voogt, J., & Meelissen, M. (2007). Computer attitudes of primary and secondary students in South Africa. *Computers in Human Behavior*, *23*(4), 1762–1776.

Brookhart, S. M. S. (2008). How to Give Effective Feedback to Your Students. *Association for Supervision and Curriculum Development*. Retrieved May 20, 2015, from http://books.google.com/books?hl=en&lr=&id=nKks5TlC_zEC&pgis=1%5Cnhttp://books.google.com/books?hl=en&lr=&id=nKks5TlC_zEC&oi=fnd&pg=PA1&dq=How+to+give+effective+feedback+to+your+students&ots=kY1WiEUKZQ&sig=NSyhijIQpcuWXbPVGp-NL9GIt00

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated Cognition and the Culture of Learning. *Educational Researcher*, *18*(1), 32–42.

Burton, P. J., & Bruhn, R. E. (2003). Teaching programming in the OOP era. *ACM SIGCSE Bulletin*, *35*(2), 111. Retrieved from

http://portal.acm.org/citation.cfm?doid=782941.782993

Credé, M., Tynan, M. C., & Harms, P. D. (2016). Much Ado about Grit: A Meta-Analytic Synthesis of the Grit Literature. *Ournal of Personality and Social Psychology, in press*.

Cresswell, J. W. (2015). *A Concise Introduction to Mixed Methods research*. United States of America: Sage Publications.

Cresswell, Plano-Clark, Gutmann, & Hanson. (2003). Advanced Mixed Methods Research Designs. *Handbook of Mixed Methods in Social and Behavioral Research*, 209–240. Retrieved from http://www.sagepub.com/upm-data/19291_Chapter_7.pdf

Creswell, J. W. L., & Clark, V. L. P. (2011). The nature of mixed methods research. *Designing and Conducting Mixed Methods Research*, 1–18.

Cutts, Q., Cutts, E., Draper, S., O'Donnell, P., & Saffrey, P. (2010). Manipulating mindset to positively influence introductory programming performance. *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE '10*, 431. New York, New York, USA: ACM Press. Retrieved from http://portal.acm.org/citation.cfm?doid=1734263.1734409

Datu, J. A. D., Valdez, J. P. M., & King, R. B. (2015). Perseverance Counts but Consistency Does Not! Validating the Short Grit Scale in a Collectivist Setting. *Current Psychology*, 121–130. Retrieved from http://link.springer.com/10.1007/s12144-015-9374-2

Duckworth, A. L. (2016). *Grit: The power of passion and perseverance*. Penguin Random House Ebury Publishing.

Duckworth, A. L., Peterson, C., Matthews, M., & Kelly, D. (2007). Grit: perseverance and passion for long-term goals. *Journal of personality and social psychology, 92*(6), 1087–1101.

Duckworth, A. L., & Quinn, P. D. (2009). Development and validation of the short

grit scale (grit-s). *Journal of personality assessment*, *91*(2), 166–74. Retrieved July 24, 2014, from http://www.ncbi.nlm.nih.gov/pubmed/19205937

Dweck, C. S. (2006). *Mindset: How you can fulfil your potential*. London: Robinson.

Dweck, C. S., & Blackwell, L. S. (2015). Take the Mindset Assessment to Learn More About Your Mindset. *Mindset Works*. Retrieved August 20, 2016, from http://blog.mindsetworks.com/my-mindset?force=1&Itemid=908

Emily, D. (2008). Motivating Students with Mindset coaching and How Brains Work (Dweck). *Classroom 2.0*. Retrieved August 28, 2016, from http://www.classroom20.com/forum/topics/motivating-students-with

Engestrom, Y. (1987). *Learning by Expanding: An Activity-Theoretical Approach to Developmental Research*. Helsinki: Orienta-Konsultit.

Goode, J., Estrella, R., & Margolis, J. (2006). Lost in Translation:\nGender and High School Computer Science. *Aspray, W. & Cohoon, J. M. (Eds.) Women in IT: Reasons on the Reasons of Under-Representation. Cambridge, MA: MIT Press.* Retrieved from http://outoftheloop.gseis.ucla.edu/pdf/LostInTranslation.pdf

Gries, D. (2008). A Principled Approach to Teaching OO First. *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, *14853*(607), 31–35.

Hochanadel, A., & Finamore, D. (2015). Fixed And Growth Mindset In Education And How Grit Helps Students Persist In The Face Of Adversity. *Journal of International Education Research*, *11*(1), 47–50.

Jenkins, T. (2001). Teaching Programming – A Journey From Teacher to Motivator. *Proceedings of 2nd Annual conference of the LSTN Centre for Information and Computer Science*.

Kaptelinin, V., & Nardi, B. (1997). Activity Theory: Basic Concepts and

Applications. *CHI 97 Electronic Publications: Tutorials*, 158–159. Retrieved from http://old.sigchi.org/chi97/proceedings/tutorial/bn.htm

Kench, D. (2014). *Exploring IT: Grade 10 Java* (Fourth.). Johannesburg: Funworks.

Köhler, B., Gluchow, M., & Brügge, B. (2012). Teaching Basic Software Engineering To Senior Highschool Students. *International Association for Development of the Information Society Conference Proceedings*, 11–18.

Lave, J. (1991). Situating Learning in Communities of Practice. *American Psychological Association*, *xiii*(429), 63–82.

Lister, R. (2005). CS Research Mixed Methods: Positivists are from Mars, Constructivists are from Venus. *ACM SIGCSE Bulletin*, *37*(Number 4).

Liu, C., & Chen, I. (2010). Evolution of constructivism. *Contemporary Issues in Education Research (…*, *3*(4), 63–66. Retrieved from http://www.cluteonline.com/journals/index.php/CIER/article/view/199

McCartney, R., Eckerdal, A., Moström, J. E., Sanders, K., & Zander, C. (2007). Successful Students ' Strategies for Getting Unstuck, 156–160.

Meriac, J. P., Slifka, J. S., & LaBat, L. R. (2015). Work ethic and grit: An examination of empirical redundancy. *Personality and Individual Differences*, *86*, 401–405. Elsevier Ltd. Retrieved from http://dx.doi.org/10.1016/j.paid.2015.07.009

Murphy, L., & Thomas, L. (2008). Dangers of a fixed mindset: implications of self-theories research for computer science education. *ACM SIGCSE Bulletin*, *40*(3), 271–275. Retrieved from http://dl.acm.org/citation.cfm?id=1384271.1384344%5Cnpapers3://publication/doi/10.1145/1384271.1384344

Nabil, M., Ali, M., & Govardhan, A. (2010). A Comparison Between Five Models Of Software Engineering. *International Journal of Computer Science Issues*,

7(5), 94–101.

Nardi, B. (1996). Studying context: A comparison of Activity Theory, Situated Action Models , and Distributed Cognition. *Context and Conciousness: Activity Theory and Human Computer Interaction*, 69–102.

Perkins-Gough, D. (2013). The Significance of Grit: A Conversation with Angela Lee Duckworth. *Educational Leadership*, *71*(1), 14–20.

Schach, S. (1990). *Software Engineering*. United States: Aksen Associates.

Shepard, L. a. (2005). Linking Formative Assessment to Scaffolding. *Educational Leadership*, *63*(3), 66–70. Retrieved from http://eric.ed.gov/?id=EJ745460

Sidiropolous, H. (2016, August 29). Email communication.

Singh, K., & Xie, M. (2010). Bootstrap : A Statistical Method. *International Encyclopedia of Education*, 46–51.

Teaching Guide for GSIs. (2011). Learning : Theory and Research Overview of Learning Theories. *Graduate Student Instructor*. Retrieved May 24, 2015, from http://gsi.berkeley.edu/gsi-guide-contents/learning-theory-research/

Venkatesh, V., Brown, S. A., & Bala, H. (2013). Research Essay Bridging The Qualitative – Quantitative Divide : Guidelines For Conducting Mixed Methods In Information Systems. *Management Information Systems Quaterly*, *37*(1), 21–54.

Vygotsky, L. (1930). *Mind in Society: The Development of Higher Psychological Processes*.

What ’ s My Mindset ? Mindset Questionnaire Scoring. (n.d.). *Mindset Works*. Retrieved from http://pvcsd.org/superintendent/pdf/Handout-Whats_My_Mindset.pdf

APPENDIX #1 - Mindset Questionnaire

Participants are asked to answer the following questions on a 4-point scale, strongly agree, agree, disagree and strongly disagree.

1. No matter how much intelligence you have, you can always change it a good deal.
2. You can learn new things, but you cannot really change your basic level of intelligence.
3. I like my work best when it makes me think hard.
4. I like my work best when I can do it really well without too much trouble.
5. I like work that I'll learn from even if I make a lot of mistakes.
6. I like my work best when I can do it perfectly without any mistakes.
7. When something is hard, it just makes me want to work more on it, not less
8. To tell the truth, when I work hard, it makes me feel as though I'm not very smart.
9. If I cannot solve a problem quickly, I give up easily.
10. I like to work on problems even if it takes a long time.

For Questions 1, 3, 5, 7, 9, the scoring is 3 for *Strongly Agree* down to 0 for *Strongly Disagree.* For the other questions, the scoring is 0 for *Strongly Agree* up to 3 for *Strongly Disagree.* Total out of 30 converted to a percentage

APPENDIX #2 - Short Grit Scale

Here are a number of statements that may or may not apply to you. For the most accurate score, when responding, think of how you compare to most people -- not just the people you know well, but most people in the world. There are no right or wrong answers, so just answer honestly!

1. New ideas and projects sometimes distract me from previous ones.
   - Very much like me
   - Mostly like me
   - Somewhat like me
   - Not much like me
   - Not like me at all

2. Setbacks don't discourage me.
   - Very much like me
   - Mostly like me
   - Somewhat like me
   - Not much like me
   - Not like me at all

3. I have been obsessed with a certain idea or project for a short time but later lost interest.*
   - Very much like me
   - Mostly like me
   - Somewhat like me
   - Not much like me
   - Not like me at all

4. I am a hard worker.
   - Very much like me
   - Mostly like me
   - Somewhat like me
   - Not much like me
   - Not like me at all

5. I often set a goal but later choose to pursue a different one.
   - Very much like me
   - Mostly like me
   - Somewhat like me
   - Not much like me
   - Not like me at all

6. I have difficulty maintaining my focus on projects that take more than a few months to complete.
   - Very much like me
   - Mostly like me
   - Somewhat like me
   - Not much like me
   - Not like me at all

7. I finish whatever I begin.
   - Very much like me
   - Mostly like me
   - Somewhat like me
   - Not much like me
   - Not like me at all

8. I am diligent
   - Very much like me
   - Mostly like me
   - Somewhat like me
   - Not much like me
   - Not like me at all

Scoring:

1. For questions 2, 4, 7 and 8 assign the following points:
   5 = Very much like me
   4 = Mostly like me
   3 = Somewhat like me
   2 = Not much like me
   1 = Not like me at all

2. For questions 1, 3, 5 and 6 assign the following points:
   1 = Very much like me
   2 = Mostly like me
   3 = Somewhat like me
   4 = Not much like me
   5 = Not like me at all

Add up all the points and divide by 8. The maximum score on this scale is 5 (extremely gritty), and the lowest score on this scale is 1 (not at all gritty).

APPENDIX #3 – Results

Table 11. Results Summary

Pseudonym	Grit	Perseverance	Passion	Mindset	Total Drafts	Overall
Arteezy	2.4	2.4	2.4	39	5	89
The Doctor	2.8	3.4	2.1	44	8	93
Solid Snake	2.5	3.5	1.6	43	8	100
Luffie	2.6	2.7	2.6	43	6	100
Bing Bong	3.6	3.3	4.0	40	8	97
Keith T. Maxwell	3.9	4.3	3.6	67	7	100
Supernatural	3.4	3.7	3.1	47	5	93
Ducky	3.1	3.8	2.3	42	7	93
Sassy the SaSquatch	3.7	3.9	3.5	60	10	100
Thaumaturge	4.2	4.6	3.8	72	5	89
Delta7736	4.3	4.8	3.7	63	6	96
Big LEZ	3.6	3.1	4.2	47	7	100
Black Panther	3.5	3.5	3.4	61	7	100
Megamind	2.3	2.6	2.1	44	4	67
Jimmy	3.8	4.3	3.3	56	7	100
Archeus	3.8	4.5	3.0	46	8	100
Zoro	3.6	4.5	2.8	46	9	100
Panda2.0	3.1	3.4	2.8	47	6	100
Mac	2.9	3.3	2.5	50	8	97
Hercules	3.6	4.1	3.1	60	7	100
Goblin	3.8	4.1	3.6	61	9	100
I heart Jgrasp	4.3	4.8	3.7	58	8	100
Pan	4.1	4.3	3.9	51	6	100
Deadpool	3.5	3.7	3.4	46	11	100
MJ	3.3	3.8	2.8	49	5	100
Krusty Krab	3.0	3.3	2.8	48	10	93
BigThatcher	4.0	4.6	3.3	44	10	100
Etzio Auditore	4.0	4.3	3.8	51	7	97
Barry Aaron	3.0	3.4	2.5	51	5	81

Table 12. Mindset-10 Raw Scores

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	1	0	1	1	0	2	2	2
2	2	1	0	2	1	1	2	2	1
3	0	1	1	2	0	1	2	2	1
1	1	1	1	2	1	1	1	2	2
2	2	2	1	1	0	2	1	1	1
3	2	3	2	3	2	3	3	1	0
2	2	2	1	2	0	2	1	1	1
2	2	2	0	2	0	1	0	1	1
2	2	1	2	2	2	2	2	2	1
3	2	3	2	3	1	3	3	1	0
2	0	3	2	3	2	3	3	1	0
2	1	2	1	2	1	2	2	0	1
2	2	2	2	2	2	2	3	1	1
2	2	0	0	2	0	1	2	1	3
2	2	2	1	2	1	2	2	1	1
2	1	2	0	2	0	2	2	0	1
3	1	2	0	3	0	2	1	0	0
2	1	2	0	2	0	2	2	2	2
2	2	1	0	2	0	1	2	2	2
3	2	2	1	2	2	2	2	1	1
3	2	2	2	2	1	2	3	1	1
1	1	2	2	2	1	3	3	1	1
3	1	2	1	2	1	2	3	0	0
2	2	1	0	2	0	1	2	1	1
2	2	2	1	1	1	2	2	1	1
1	1	1	1	1	0	1	3	2	2
3	0	3	0	3	0	3	0	1	0
2	1	2	1	2	0	2	3	1	1
2	1	2	1	2	1	2	1	2	1

Table 13. Mindset-8 Raw Scores

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
1	1	1	0	1	1	0	2
2	2	1	0	2	1	1	2
3	0	1	1	2	0	1	2
1	1	1	1	2	1	1	1
2	2	2	1	1	0	2	1
3	2	3	2	3	2	3	3
2	2	2	1	2	0	2	1
2	2	2	0	2	0	1	0
2	2	1	2	2	2	2	2
3	2	3	2	3	1	3	3
2	0	3	2	3	2	3	3
2	1	2	1	2	1	2	2
2	2	2	2	2	2	2	3
2	2	0	0	2	0	1	2
2	2	2	1	2	1	2	2
2	1	2	0	2	0	2	2
3	1	2	0	3	0	2	1
2	1	2	0	2	0	2	2
2	2	1	0	2	0	1	2
3	2	2	1	2	2	2	2
3	2	2	2	2	1	2	3
1	1	2	2	2	1	3	3
3	1	2	1	2	1	2	3
2	2	1	0	2	0	1	2
2	2	2	1	1	1	2	2
1	1	1	1	1	0	1	3
3	0	3	0	3	0	3	0
2	1	2	1	2	0	2	3
2	1	2	1	2	1	2	1

Table 14. Grit Raw Scores

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
2	3	2	2	3	2	3	2
2	3	2	3	2	2	4	4
1	3	1	4	2	1	5	3
3	2	2	3	3	2	3	3
3	3	5	4	4	4	3	3
5	5	3	4	4	3	5	4
3	3	3	4	3	3	4	4
2	4	3	4	1	2	3	4
4	3	3	4	4	3	4	4
3	4	4	5	4	4	5	4
3	5	4	5	4	4	4	5
4	3	5	3	4	4	4	3
4	2	3	4	4	4	4	4
2	2	2	3	4	1	3	3
3	4	3	5	3	4	4	5
3	4	3	5	1	4	4	5
3	3	1	5	4	4	5	5
3	4	3	4	2	3	2	3
2	2	3	4	3	2	3	4
4	4	2	5	3	3	4	4
2	5	4	4	4	4	4	4
3	4	4	5	3	4	5	5
4	2	3	5	4	5	5	5
4	3	4	4	4	3	4	4
4	3	2	4	3	3	4	4
3	3	2	3	2	3	4	3
1	5	5	5	4	5	5	5
4	4	4	4	4	4	4	4
2	4	2	3	2	3	4	4

APPENDIX #4 - Scripts in R Studio

Grit VS PAT

```
> cor.test(Results$Grit,Results$PAT,method=c("pearson"))

    Pearson's product-moment correlation

data:  Results$Grit and Results$PAT
t = 2.8734, df = 27, p-value = 0.007818
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.1427289 0.7223163
sample estimates:
      cor
0.4839203

> cor.test(Results$Grit,Results$PAT,method=c("spearman"))

    Spearman's rank correlation rho

data:  Results$Grit and Results$PAT
S = 2652.1, p-value = 0.06534
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.3467754
```

Grit vs PAT Scatter Plot

```
plot(Results$Grit, Results$PAT,main="Grit vs PAT",
xlab='Grit',ylab='PAT')
line1 <- lm(Results$PAT ~ Results$Grit)
abline(line1)
```

Mindset-8 vs PAT

```
> cor.test(Results$Mindset..8,Results$PAT,method=c("pearson"))

    Pearson's product-moment correlation

data:  Results$Mindset..8 and Results$PAT
t = 1.0971, df = 27, p-value = 0.2823
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.1730251  0.5327503
sample estimates:
      cor
0.2065808

> cor.test(Results$Mindset..8,Results$PAT,method=c("spearman"))

    Spearman's rank correlation rho
```

```
data:  Results$Mindset..8 and Results$PAT
S = 3004.6, p-value = 0.1732
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.2599484
```

Mindset-8 vs PAT Scatter Plot

```
plot(Results$Mindset..8, Results$PAT,main="Mndset-8 vs PAT",
xlab='Mindset-8',ylab='PAT')
line2 <- lm(Results$PAT ~ Results$Mindset..8)
abline(line2)
```

Perseverance vs GRIT

```
> cor.test(Results$Perseverance,Results$Grit,method=c("pearson"))

    Pearson's product-moment correlation

data:  Results$Perseverance and Results$Grit
t = 8.8445, df = 27, p-value = 1.847e-09
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7247219 0.9336719
sample estimates:
      cor
0.8622116

> cor.test(Results$Perseverance,Results$Grit,method=c("spearman"))

    Spearman's rank correlation rho

data:  Results$Perseverance and Results$Grit
S = 621.61, p-value = 6.966e-09
alternative hypothesis: true rho is not equal to 0
sample estimates:
     rho
0.846895
```

Passion vs GRIT

```
> cor.test(Results$Passion,Results$Grit,method=c("pearson"))

    Pearson's product-moment correlation

data:  Results$Passion and Results$Grit
t = 8.7509, df = 27, p-value = 2.291e-09
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.7203405 0.9324860
sample estimates:
```

```
      cor
0.8598406

> cor.test(Results$Passion,Results$Grit,method=c("spearman"))

    Spearman's rank correlation rho

data:  Results$Passion and Results$Grit
S = 632.98, p-value = 8.744e-09
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.8440935
```

Perseverance vs Passion

```
> cor.test(Results$Passion,Results$Perseverance,method=c("pearson"))

    Pearson's product-moment correlation

data:  Results$Passion and Results$Perseverance
t = 2.8642, df = 27, p-value = 0.007993
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.1412198 0.7215789
sample estimates:
      cor
0.4827399

> cor.test(Results$Passion,Results$Perseverance,method=c("spear-
man"))

    Spearman's rank correlation rho

data:  Results$Passion and Results$Perseverance
S = 2203.6, p-value = 0.01264
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4572523
```

Perseverance vs Number of Submissions

```
> cor.test(Results$Perseverance,Results$No.of.Submis-
sions,method=c("pearson"))

    Pearson's product-moment correlation

data:  Results$Perseverance and Results$No.of.Submissions
t = 1.41, df = 27, p-value = 0.1699
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.1157254  0.5733606
sample estimates:
```

```
     cor
0.261888

> cor.test(Results$Perseverance,Results$No.of.Submis-
sions,method=c("spearman"))

    Spearman's rank correlation rho

data:  Results$Perseverance and Results$No.of.Submissions
S = 3258.1, p-value = 0.3044
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.1975132
```

Perseverance vs PAT

```
> cor.test(Results$Perseverance,Results$PAT,method=c("pearson"))

    Pearson's product-moment correlation

data:  Results$Perseverance and Results$PAT
t = 2.4782, df = 27, p-value = 0.01975
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.07595804 0.68837769
sample estimates:
      cor
0.4304796

> cor.test(Results$Perseverance,Results$PAT,method=c("spearman"))

    Spearman's rank correlation rho

data:  Results$Perseverance and Results$PAT
S = 2688.5, p-value = 0.0731
alternative hypothesis: true rho is not equal to 0
sample estimates:
     rho
0.337807
```

Perseverance VS PAT Scatter Plot

```
plot(Results$Perseverance, Results$PAT,main="Perseverance vs PAT",
xlab='Perseverance',ylab='PAT')
line3 <- lm(Results$PAT ~ Results$Perseverance)
abline(line3)
```

Number of Submissions vs PAT

```
> cor.test(Results$No.of.Submissions,Re-
sults$PAT,method=c("pearson"))
```

```
    Pearson's product-moment correlation

data:  Results$No.of.Submissions and Results$PAT
t = 3.1782, df = 27, p-value = 0.003696
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.1919935 0.7456863
sample estimates:
     cor
0.521783

> cor.test(Results$No.of.Submissions,Re-
sults$PAT,method=c("spearman"))

    Spearman's rank correlation rho

data:  Results$No.of.Submissions and Results$PAT
S = 2300.6, p-value = 0.01886
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4333479
```

Number of Submissions vs PAT Scatter Plot

```
> plot(Results$No.of.Submissions, Results$PAT,main="Number of Sub-
missions vs PAT", xlab='Number of Submissions',ylab='PAT')
> line4 <- lm(Results$PAT ~ Results$No.of.Submissions)
> abline(line4)
```

Grit vs PAT Bootstrapped

```
> library(boot)
> f <- function(data, i){
+   d2 <- data[i,]
+   test <- cor(d2$Grit,d2$PAT,method=c("pearson"))
+   return(test)
+ }
> bootcorr <- boot(GritvPAT, f, R=1000)
> print(bootcorr)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = GritvPAT, statistic = f, R = 1000)


Bootstrap Statistics :
     original      bias    std. error
t1* 0.4839203 -0.01771385   0.1820437
```

Mindset-8 vs PAT Bootstrapped

```
> library(boot)
> f <- function(data, i){
+   d2 <- data[i,]
+   test <- cor(d2$Mindset..8,d2$PAT,method=c("pearson"))
+   return(test)
+ }
> bootcorr <- boot(MindvPAT, f, R=1000)
> print(bootcorr)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = MindvPAT, statistic = f, R = 1000)


Bootstrap Statistics :
     original       bias    std. error
t1* 0.2065808 0.0003739417   0.1726084
```

Perseverance vs PAT Bootstrapped

```
> library(boot)
> f <- function(data, i){
+   d2 <- data[i,]
+   test <- cor(d2$Perseverance,d2$PAT,method=c("pearson"))
+   return(test)
+ }
> bootcorr <- boot(PersvPAT, f, R=1000)
> print(bootcorr)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = PersvPAT, statistic = f, R = 1000)


Bootstrap Statistics :
     original      bias    std. error
t1* 0.4304796 -0.03309941   0.1734504
```

Number of Submissions vs PAT Bootstrapped

```
> NumSubvPAT <- read.csv("C:/Users/delia.SBC/Google Drive/Mas-
ters/Data from Boys/Data for R/NumSubvPAT.csv")
>   View(NumSubvPAT)
> library(boot)
> f <- function(data, i){
+   d2 <- data[i,]
+   test <- cor(d2$No.of.Submissions,d2$PAT,method=c("pearson"))
```

```
+   return(test)
+ }
> bootcorr <- boot(NumSubvPAT, f, R=1000)
> print(bootcorr)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = NumSubvPAT, statistic = f, R = 1000)


Bootstrap Statistics :
    original      bias    std. error
t1* 0.521783 -0.01378103   0.1462041
```

APPENDIX #5 - Semi-Structured Interview Questions

Please describe your project?

What did you do when you were stuck?

What improvements would you make to your project?

Did the deadlines help?

Did you use the templates and sample code?

Were you able to use the code and concepts taught in the beginning of the year?

APPENDIX #6 – Coded Interviews

Interview – Student A

```
int_6.atx
 1  DK: Please just tell me your name and tell me what you
 2  did in your project just briefly describe your project for
 3  me.
 4  SA:  Uhm... my name is Student A uhm... I tried to make a
 5  quiz game by using a maze when uhm... an object hits a
 6  .... When the player hits an object in the maze it
 7  generates a quiz frame and then you uhm... have three
 8  lives and then you get scores for answering questions,
 9  and if you don’t you loses a life and if you are not able
to
10  get to the dragon before you loses all your life the game
11  is over. And if you get to the dragon you get specific
12  points and it adds to your score and your score is, you
13  can use your score to get into different levels.
14  DK: Ok... and did you enjoy it how did you find the actual
15  task, I mean the whole process for you?
16  SA:  I think it was quite tiring uhm... it took me a long
      -> (913-245): emotional - depressing
           -> (913-245): PS strategy - research
17  time to figure out how to do a maze and it took me
18  quite a lot of research but I still could not manage it out
19  so it was quite depressing in the end.
20  DK: Ok so what did you do when you were stuck?
21  SA: Uhm... I looked up on YouTube or stuff like that and
      -> (1226-187): perseverance - did not give up
           -> (1216-234): PS strategy - plan solution
22  tried to uhm.. draw algorithms and stuff or figure out
23  the logic and stuff... what to do but I always get stuck.
24  DK:  Did you use println's?
25  SA: Ya I tried to use println's.
      -> (1452-40): PS strategy - plan solution
           -> (1452-40): tracing - println
26  DK: Did it work?
27  SA: It sorted out problems but it didn’t sort out the big
28  problem.
29  DK: And did you ever decide to come to school and ask
30  the students?
31  SA: Uhm... no.
32  DK: Do you think you should've?
33  SA: I thought I should've but it's just the communication
34  between my class mates.
35  DK: Is it a language thing?
36  SA: A little bit and I don't know I am very shy.
      -> (1892-56): PS strategy - too shy to ask for help
37  DK: Ok fair enough Uhm... and the WhatsApp did you
```

use the WhatsApp to ask for help or does that also not work for you?
SA: I didn’t try to use WhatsApp.
DK: You know the group the class group, you didn’t want to use that?
SA: No uhm.. for me I think it is uhm... also my problem.
DK: Just being shy?
SA: Ya I feel quite embarrassed when asking questions.
DK: You are not the only one to worry about that so don’t worry. So did you enjoy the process did you learn from the process?
SA: I learnt from the process, I learnt a lot from the
-> (2546-479): perseverance - did not give up
-> (2546-479): PS strategy - too shy to ask for help
process because Uhm... to figure things out by myself more than just to do homework and this project you have to think for yourself and create stuff by yourself but I still didn’t ask questions when I got into trouble, so
Uhm... I enjoyed it but the result wasn’t quite satisfying.
DK: So now you are going to carry on with this project next year?
SA: Uhm... ya but I think the maze for me might be a
-> (3027-192): PAT - improvements
little bit too hard. I am thinking of changing the maze to easier stuff so I can get something I can deal with.
DK: And do you think you would ask more questions or do you think you would still not be able to ask questions?
SA: Uhm... I am just thinking uhm.. it depends like if I am
-> (3362-234): PS strategy - asked friends
close enough to speak Uhm.. I speak a lot to my friends and I speak a lot with Student Y and my other classmates I don’t really.
DK: Are you in boarding?
SA: No it is just Student Y is always in the same class as me and he likes to talk to me so.
DK: So your issue was just not feeling comfortable with asking questions.
SA: Ya it has been quite an issue for a long time even in China I don’t speak to teachers quite often.
DK: Ya that is fine but you maybe need to move towards finding people in your class that you can talk to, to help you that is maybe what you need to do for next year.
SA: Yes I know.
DK: And then what I wanted to know is when you were stuck you would go to YouTube and you would go to Google and then maybe print out a few things, Uhm... and then just give up?
SA: Uhm... sort of like I had a week like didn’t work on
-> (4397-384): deadlines - adhere to

-> (4397-384): difficulties - topic choice
my game that's all because I was thinking should I change it or should I not and because that was before the deadline for the PAT specification document and I was thinking should I change my topic but I was also too shy to ask if I could, so I stayed on the same game.
DK: So now the deadlines and the graphs did that help you, like submitting you got to have this by this date and that by that date?
SA: it pushed me because after the week I was like I
-> (4944-192): deadlines - adhere to
wasn't really feeling like doing that after that week because like it was the deadline so we had to work on it.
DK: So do you think you got a better mark because of the deadlines?
SA: Yes
DK: Ok cool, and the templates did that help, the Battleships template and the documentation of how to do the specs and the design did that help?
SA: Yes that helped a lot because I looked at a lot of
-> (5417-228): Scaffolding documentation useful
examples from last years and uhm... the formats and stuff like that, that's why I was able to get a high mark because of the documents.
DK: And then when you were coding did you, how did you do the documents in the code did you do documents and then the code or the code and then the documents or bits and pieces in between?
SA: I did the documents first because when I was doing
-> (5875-186): documentation and coding - simultaneous
the documents I was thinking about the code, s uhm... I was just logically thinking what should happen.
DK: And then did you battle to actually code what you wanted to happen?
SA: It didn't happen like as the document went so.
-> (6154-58): difficulties - coding
DK: So you battled to actually code like the document said it would be, you didn't think of changing the documents?
SA: I did think of changing the documents but like I was wanting to change it into like a different thing.
DK: You wanted to change the topic.
SA: Ya like I was thinking of like not doing a maze but to
-> (6530-302): difficulties - coding
-> (6530-302): difficulties - topic choice
do a grid then you click on stuff until you click on the dragon like if you click on other stuff you get questions that you need to do, I was thinking of that because we practised on the grid so.
DK: Oh ok I know what you are saying, and then did the
-> (6834-230): prior knowledge - useful

first two terms work help you, everything you did up until this project was that good enough to get you started.
SA: Uhm... yes.
DK: But you are saying maybe not because you still needed a bit more.
SA: Yes because it's the understanding of the theory and
-> (7155-150): difficulties - coding
also the coding, so uhm... it is the logic that I am not getting.
DK: And then if you had to tell another person similar to you how would you go about it or if a grade 10 came to you and asked you for advice what would you tell them?
SA: Uhm... I would tell them that I didn't do good so ask someone better than me.
DK: What if they say how did you find it or what should I do or what should I not do?
SA: Uhm... Don't make a game too hard like make it easy and just do hard like more specific like don't do something you can't do.
DK: That is good advice that is very good advice, and then uhm... so for next year we are going to carry on so you are going to change your topic now for next year?
SA: Yes, I think so, I will probably change it to a good one, is it fine?
DK: Yes it is fine, so that will be good and then come speak to me next time and if you are stuck tell them and then we can change it, it doesn't really matter, it really doesn't matter as long as you have a project, nobody knows what you are planning to do nobody knows, so if you change the plan or update the code as you go along, and even if you code the whole thing then we can go back and change the plan. Ok so don't worry about it but thank you, thanks so much for your time it is very interesting what you are saying...

Interview – Student B

int_4.atx
SB
SB: Hi, my name is StudentB. My project was who wants to be a millionaire.
DK: What was it about?
SB: Uhm…it was a game exactly like the TV show … uhm… I used a database to get all the questions and all the answers in. it took me a while, I got all the pictures that look really nice. It worked really well. Uhm ya.
DK: Ok, so were you happy with the project?
SB: I was …uhm… I didn't have a back-end cause I
-> (481-203): difficulties - design
-> (481-203): perseverance - gave up

kind of messed up cause I sort of started and then sir was like "Back end needed" and I was like "Oh OK but mine works already". And I couldn't, I did try cause I
-> (684-281): difficulties - coding
-> (684-281): difficulties - design
-> (684-281): perseverance - gave up
finished it about 4/5 days before we had to hand it in and I started doing it but I just couldn't get it to work so I handed in what worked. And I got, I lost 5 marks for not doing a back-end which was ok.
DK: And if you had had more time to fix the back-end would you have done it?
SB: I think so, if I had had maybe another weekend
-> (1071-422): deadlines - not able to adhere
-> (1071-422): difficulties - coding
-> (1071-422): emotional - frustrated
or whatever, where I could of just sat down I would have been able to get it and it was small things…uhm… you fix one problem and the next one would come up and it just eventually got really irritating so I was like ok good enough just hand in. Uhm…but definitely if I had had more time I would of done it.
DK: And obviously you were rowing? So that's why you are mentioning weekends? How was your rowing schedule?
SB: It was ok, uhm…this year being opens everything kind of gets dumped to the little people so I would go home after my race, I would put my boat of the trailer and go home so I would be home sometimes 2/3 o clock. Uhm…one day we didnt make the final, I was home really early so ya. I was home 1 o clock?
DK: Oh alright so your weekends weren't to bad?
SB: No, no, no. not too bad (mumbled)
DK: Sorry?
SB: Also gaming. I am a really big gamer so that did
-> (2142-234): deadlines - not able to adhere
-> (2142-234): perseverance - gave up
-> (2142-234): perseverance - procrastination
take up a bit of my time cause it's like yes turn on computer first thing is to go and play a game and it's like but project…ahhhh… it's a problem.
DK: So what did you do?
SB: Uhm…I would game until I got frustrated and then I would go and code until I got frustrated and then I would go and game so I would kind of…
DK: So which was longer? The gaming or the coding?
SB: Well…uh…I think the gaming got me more
-> (2662-416): deadlines - not able to adhere
-> (2662-416): perseverance - gave up
-> (2662-416): perseverance - procrastination

irritated more quickly …uhm…because I was playing a game that I had just started and I had no idea what I was doing …uhm… so it was short games but they were, it was also really, really good guys who I was playing with so it was really, really quick and I had no idea what I was doing so that really irritated me but
  -> (3025-212): emotional - frustrated
the coding was long periods but it would be long periods of getting stuff done and then as soon as I'm, I got stuck on something I'd Google it and if that didn't
  -> (3237-36): PS strategy - search Google
work then I'd start YouTubing and then sometimes I
  -> (3301-33): perseverance - did not give up
    -> (3301-33): PS strategy - search YouTube
get side tracked but most of the time I ended up
  -> (3342-79): perseverance - did not give up
    -> (3342-79): perseverance - procrastination
solving the problem but if I'd get annoyed and I'd
  -> (3421-69): emotional - frustrated
    -> (3421-69): perseverance - did not give up
      -> (3421-69): perseverance - procrastination
move on to something else.
DK: So now friends? Did you ask friends for help?
SB: Uhm…yes. Whenever, for my GUIs. To get them
  -> (3563-288): perseverance - did not listen in class
to open and close when you click on buttons, 'play the game' takes you to the game and closes the menu screen, I didn't know how to do that and I wasn't paying attention to what Mr Gill was telling us so…uhm…I asked Keith T. Maxwell, he came to my desk and showed me how to do it and I was like oh. So
  -> (3917-160): PS strategy - asked friends
he showed me for one and I did, I think I had 4 GUIs so I got it to work perfectly, ya.
DK: So, your strategy would be, you get stuck, Google, YouTube, got frustrated and moved on to something else? When would you call the friends and that strategy?
  -> (4287-231): PS strategy - asked friends
SB: When I was at school, then I didn't have to phone them and they could almost do it for me but show me at the same time…uhm… ya. That was only when I used friends, it was only for that one piece.
DK: So you weren't running WhatsApp groups and asking stuff all the time with some people posting things?
SB: No, I'm not really one to post questions and stuff. I Google it and if I cant Google it, I ask ya.
DK: And what about trace tables and printlns?
SB: Ah no that's so tedious to do. So uhm…
  -> (4850-321): emotional - frustrated

-> (4850-321): tracing - no trace tables
normally I do need to do a trace table it would like, I would start doing it and then I would get through like one loop and I would go this is too much work and I'd stop. If I got really frustrated and I needed to get through it, I would ask my dad. Uhm… He's not really
-> (5171-448): PS strategy - asked friends
-> (5170-449): tracing - verbally
good with coding but troubleshooting he is extremely good with its one of his few talents. I don't know why but he can sit me down and be like explain to me how this works and I would start explaining to him and then I would be like oh my gosh there's the problem, fix it and then he will get up and go. Doesn't even have to do anything.
DK: Oh so you just need to talk through it?
SB: Ya.
DK: OK so that's a strategy. And then syntax errors? Did they bother you at all or could you solve those?
SB: Uhm…I don't think I had any syntax errors
-> (5834-116): PS strategy - plan solution
cause I did kind of think it out as I went through…uhm… I think I had one problem where it would bring my questions up before I had started the game…ah… which was a problem because you need to…
DK: So those are more like logical errors?
-> (6180-281): difficulties - coding
DM: Ya, that was the only kind of big problem I had. Sometimes my buttons wouldn't work and then I would be like why and then I would realise cause I had disabled it before I needed to, disabled it the wrong place…ya. So ya no syntax errors.
DK: And then objects? How did you deal with the objects and the arrays of objects? Well cause you didn't have a back-end you sort of by passed a bit of it.
SB: Ya..mmm… I didn't exactly have an array for my
-> (6662-71): difficulties - coding
questions. Ma'am if you had to read my project you
-> (6733-377): difficulties - design
-> (6733-377): difficulties - coding
would probably cringe cause it was that bad but ah… what happened was I set it up so that my database had all the questions, the correct answer (the letter for the correct answer) and all 4 correct answers and then I just read it in each time I would read a new question which…
DK: Well it works. It just doesn't have a back-end, it's a fine solution but its forcing you into a certain design but you just solved the problem as it was which is fine had you just not had to read the rubric.

SB: So ya uhm…that was it.
DK: So now tell me about your code and your documentation? Which one did you do first?
SB: Uhm…I started on the specs, the specs
 -> (7520-657): difficulties - design
  -> (7520-657): documentation and coding - simultaneous
   -> (7520-657): difficulties - coding
document and I handed that in and then I started my code…uhm… and then once I got the specs document back I was a bit, quite a bit far in my code and then I got the design document to do and then I was like "OK cool" do the whole design document and see if I can get anywhere and then ran in to problems so then I was like ok hold on I will finish the game first and then I will do the design document to the game and … So in the end it was kind of like bits and pieces put together in the end like made the project.
DK: So and the pseudo code issue how did you how did you cope with the pseudo code?
DM: Uhm… I don't battle with pseudo code so I kind of
 -> (8293-336): difficulties - documentation
what happened was I didn't, my first hand in for my design I didn't have pseudo code it was just a blank page. Uhm… Mr Gill just gave me a skeef <funny> look
 -> (8488-336): documentation and coding - code first
and moved on, So after that I finished my code and went back and went line by line and I wrote it into word and I kind of used the past project to see the format and what I needed to do. If I didn't know how to take something from pseudo code I just went and
 -> (8825-41): perseverance - did not give up
  -> (8825-41): PS strategy - search Google
Googled it.
DK: So you also used, your other resource was the past example papers the templates, ok. Uhm…talking about perseverance and that. So you telling me when you got to an error you did have a strategy that you would try and do something, you would try and fix it…
DM: Ya uhm… it wasn't ah there's an error ok stop,
 -> (9185-267): perseverance - did not give up
  -> (9185-267): perseverance - take a break
and move on it was try and fix it to my best ability and as quickly as possible but if I did get really stuck then it would be kind of like lets chill and come back to that later.
DK: But you would come back?
DM: Mmm.
DK: Ok, and the only place where it kind of let you
 -> (9518-273): deadlines - not able to adhere

down was towards the end when you ran out of time to complete the back-end cause you would have had to redesign the whole code. You would have to go back for like a month to fix it.
DM: Mmm… if I had the time I would have been able to do it definitely.
DK: And what improvements would you do for next year cause obviously we want to reuse this project it matric.
-> (10028-545): feedback - did not read rubric
-> (10028-545): PAT - improvements
SB: Uhm… next year I would definitely do a back-end this time. I would focus more on the rubric. I kind a looked at it briefly and was like ok this needs to be done, this needs to be done, get this done and then that was it. I looked at my rubric maybe once or twice and then every time I got something wrong I would look at it, fix it, hand it back in immediately and see if I could get that right. So it was a real big kind of trial and error kind of thing.
DK: So that goes with the marking with the deadlines, did that help? I mean multiple submissions and deadlines running?
DM: Ya definitely. I think if I, if it was just one hand in I
-> (10731-300): feedback - multiple submissions helpful
would have gotten like 13 for almost all of it. Uhm… the hand ins definitely helped cause you can see where you've gone wrong and gone right and …uhm ya. So definitely the hand ins really did help.
DK: How many times did you hand it?
SB: My code I handed in twice I think, my specs was 4 and my design was 3.
DK: So you did have quite a few hand ins.
SB: Well ya. I got full marks for both documents and then…
DK: But surely your class diagram should have had the back-end in it? Or did you fudge it?
-> (11424-108): documentation - UML
SB: Ya, I kind of made up some rubbish here and there to see what it would look like and then…
DK: Well you see that's where you went wrong so if you had followed your design into your thing you would have had a back-end. So actually your code and your design didn't match?
SB: Not exactly , no.
DK: Haha, ok so future improvements? What would you do besides getting the design right and getting the code right, what would you add on?

-> (11960-352): PAT - improvements
SB: Uhm…probably to do this game again. I don't think I would change anything but maybe add a timer so that there's a better point system cause my point system was if you got the question right you get that amount of money added to your high score and then… So there was a max high score that you could get so it wasn't like there was very different things. So I think I would of added, like if there was a person in the first 3 seconds who got the question right or whatever you would get so many points and depending on how quickly you answer it and that gets timed by the money and then added to. So the score would be more different instead of just, if you got to round five and got all of them right then you got this amount…
DK: So there's a timer. Anything else you would want to add?
SB: Uhm…no? Nothing really cause it's a simple game so… Something I thought oh hey this will work quickly. I had the board game at home as well so I just used the questions on those card … uh. That was easiest to do as well, to type out every single card. There were 150 questions that I typed out.
DK: You didn't think of downloading it?
SB: only once I had done like 130 of them I was like
-> (13324-203): PS strategy - asked friends
Student Z was, when I asked Student Z for help he was like why don't you just download them I was like… I didn't think of that ok. But it was fun I
-> (13527-296): perseverance - did not give up
-> (13533-814): PS strategy - found shortcuts
coded a GUI to enter the questions for me so I would type it out instead of entering SQL statements which really helped cause I eventually found shortcuts in that as well to set the money value to this question, it would increment itself and stuff like that like so. I tried to make my life a bit easier with the GUI I guess.
DK: That is actually quite a good idea. Alright anything else you want to add?
SB: Uhm… no.
DK: If you had to give advice to somebody else doing this project next year?
SB: Make sure uh… the back-ends the big one. Uhm… also do the design document first cause if you start your code the you code gets a head your design documents not what it needs to be. Uhm… I've told Student X, he asked me is there anything. I said just start thinking, get ideas, get the done quickly cause once you got that its easy but actually thinking of a game actually took me 2/3 weeks to actually think ok we will do who wants to be a millionaire and then

```
261  when that was it there was a week and a half maybe to
262  do the coding and documentation, I had done the
263  documentation to an extent uhm so ya. It was a real
264  mess up in the beginning but it came together in the
265  end.
266  DK: And then uhm…did it help what we taught? Did
267  you go back to the first term knowledge that we had
268  done? Did you work on them?
269  SB: I used Battleships to help with GUIs a bit cause
      -> (15166-180): Scaffolding Battleships useful
270  the help GUI and all the stuff like that all the code taught
by your colleague
271  Gill taught us using Battleships, I used that. I pretty
272  much copied and pasted and used that exactly. Uhm…
273  DK: And the template? The document templates?
274  DM: Ya, I used them for the specs document. I copied
      -> (15480-294): Scaffolding documentation useful
275  that document and just edited it to what I needed.
276  Uhm… and then the design document I typed up
277  completely but using the old one as a reference to get
278  the right headings and format and stuff like that.
279  DK: Cool, thanks babes
280  SB: Pleasure
```

Interview – Student C

```
int_1.atx
 1  Student C - SC
 2  DK: Tell me about your PAT? How did it go?
 3  SC: Overall I think it went very well Ma'am. In the
      -> (85-359): PAT - too ambitious
 4  beginning I was quite ambitious, I was planning to
 5  have a whole bunch of moving objects on the screen,
 6  but I threw that idea away in the end because I
 7  realised on the first day working that it, it was
 8  just way too ambitious for my expertise. So uh… ya.
 9  DK: what did you, what did you do when you got stuck
10  in the project. How did you deal with it.
11  SC: Uhm… I'd actually copy the program multiple
      -> (561-420): perseverance - did not give up
12  times and then I'd have all possible different
      -> (572-421): PS strategy - diagnosing problems
13  solutions err… in the code, and I would see which
14  one worked best then I would test all the
15  functionality every function of the program using
16  that solution if it worked perfectly, I would choose
17  the one that I thought worked the best or looked
18  best on paper.
19  DK: that's quite clever, and syntax errors how did
20  you deal with those?
21  SC: Uhm I just used a lot of println's ma'am, I just
```

22 printed everything out so if I had errors I would
-> (1095-193): PS strategy - diagnosing problems
-> (1095-193): tracing - println
23 print out my SQL statement and I would just
24 basically use that.
25 DK: Uhm… and trace tables did you not use that.
26 SC: Not really trace tables I find them a bit too
27 time consuming.
-> (1359-71): PS strategy - diagnosing problems
-> (1359-71): tracing - no trace tables
28 DK: And even the online tracing.
29 SC: Nah its just… what I tend to do is take a piece
30 of paper and jot down random value, like randomly
-> (1486-274): PS strategy - diagnosing problems
-> (1486-274): tracing - no trace tables
-> (1486-274): tracing - using paper
31 put a value and just do what is happening to that
32 value in my brain, and then it will come up with
33 what happens and what should happen.
34 DK: And in terms of the project overall are you
35 satisfied with your result?
36 SC: yeah uhm… I was, I had, I was a bit time pressed
-> (1855-140): deadlines - not able to adhere
-> (1855-140): documentation - achievable
37 for the, for the documentation but overall I think I
38 did well. uhm… It's more just the how quickly the
39 deadlines approach on you, so then you got to
-> (2006-68): deadlines - adhere to
40 quickly get different sections of your code up and
41 running, so you know how that bit of your code is
-> (2128-214): documentation and coding - simultaneous
42 going to work, before you can talk about any
43 documentation, but overall I think it went well.
44 DK: So you tried to code it and then document it?
45 SC: I would code and document more or less at the
-> (2393-195): perseverance - did not give up
46 same time, so then if I came across an error I would
47 fix anything that related to that bit of code in my
-> (2413-195): difficulties - coding
-> (2413-195): documentation and coding - simultane-
ous
48 documentation.
49 DK: Then how did you the - what you learnt in the
50 first two terms, was that easy to apply or was it
51 difficult?
52 SC: It was relatively easy. Every now and then I
53 have to go back just to double check what parameters
54 to pass and what format I should be using in it.
-> (2762-307): prior knowledge - useful
-> (2762-307): PS strategy - diagnosing problems
55 Overall it was quite easy to remember,… just

remember the general rules and stuff we learnt like substring or…
DK: So you kind of used your beginning stuff to support you and you went back to it and the objects the arrays of objects and passing the objects.
SC: I didn't find that an issue at all.
DK: So you didn't have like errors where one object is calling another or…
SC: No I would…
DK: had it in a loop or something?
SC: I had a quite annoying error when it came to
  -> (3451-324): difficulties - coding
    -> (3452-458): perseverance - did not give up
just the creation of my GUISs, where it forms the main method uhm… that would just be an extra because previous methods I had was testing and it was sending uhm… making a random object in that and it was giving me all these errors, that I couldn't figure out why turns out it was just that little bit of extra code, in the main method and then…
DK: So how did you fix that error?
SC: I eventually just got I gave up and started
  -> (3969-30): perseverance - gave up
    -> (4000-226): tracing - using paper
scrolling for what it could possibly be, I lost track of like on a piece of paper I kept track of what changes I've made, and I like added this bit of segment using copy and paste that segment so I'll just have to go back to the original, and I'll have that so obviously I just lost track of it there, so I created this fake object here.
DK: So you are actually running a system where you tracking changes like kind of manually?
SC: Yeah tracking my progress and whenever I can
  -> (4541-155): perseverance - did not give up
make a change, I just see different options for that change and then choose one of them.
DK: Have you always coded like this?
SC: Yeah it's, I find it, I find it the best way
  -> (4755-197): PS strategy - diagnosing problems
because its. I go and I try one method and I delete it if it doesn't work so I'll try another or I'll just do it and I won't like it and I'll lose track
  -> (4948-214): difficulties - lost track
    -> (4948-214): emotional - frustrated
of what worked once or almost worked and. I'll try change that and it becomes horrible and I can't really get back so I don't really like that method
DK: When did you start developing this method.
SC: Uhm… last year in the middle of second term.
DK: so obviously it worked very well for this?

99 SC: Yeah it did.
100 DK: Didn't you find it when you had one solution so
101 you had ten options and then that option had ten
102 options.
103 SC: I would also find it like I'd work, I'd work in
-> (5511-441): difficulties - coding
-> (5510-442): perseverance - did not give up
104 order from like going to the depths of the program
105 so like go through the small niggly things, get
106 those perfect and find the best option for that and
107 work my way up to the bigger things uhm… it came to
108 just, to just entering and checking passwords in the
109 database I'd have that then properly go to making
110 the user object.
111 DK: So you kind of tested before you kind of
112 designed?
113 SC: I built my way from the inside out really.
-> (6040-44): difficulties - coding
-> (6040-44): PS strategy - fix errors from bottom
up
114 DK: OK that is interesting Uhm… so just quickly
115 briefly describe your project.
116 SC: Err… it's a riddle game but it works on leniency
117 so the riddles are in a database but as you progress
118 so... Uh… this thing is just giving you these
119 riddles, there is code basically to take how many
120 answers you have given correctly over all in that
121 round of the game. I think it was divided by 1000
122 then times by a random number. I think the random
123 number was between 0 and 10 and cause less than a
124 certain amount…uhm…it wouldn't be lenient, if you
125 did the answer wrong it won't give you another try,
126 greater than it would be lenient but I made it so
127 that on a scale from 0 to 100 if it was less than 80
128 it wouldn't be lenient so greater than that so…to
129 make it more improbable that person would have that
130 leniency cause if you just started the game, and you
131 get one wrong it gives you that leniency already
132 that's not really worth it.
133 DK: So… you basically build and intelligence behind
134 like a basic quiz kind of thing?
135 SC: Yes, cause like basically the way the code works
136 is as soon as you pass over 20, 20 riddles sorry.
137 Then there was the actual possibility that when that
138 number was divided by 1000 was actually, it could
139 actually be greater than that one amount. Anything
140 less then that couldn't be greater than that on
141 amount.
142 DK: Ok, so that is quite clever. So if you had to
143 tell another child, give them advise about this
144 project what would you say.

SC: Uhm…start as soon as you can and when it comes to ideas first think of what you can do, what your capabilities are because … uhm… so may guess when I did I base…Had very basic knowledge of how to do anything not even, not even real collisions so know your capabilities and then don't go mad thinking this massive idea. Oh, it's going to be amazing whereas I don't have the capabilities, cause once you try and start working on it kind of undermines your confidence a bit cause oh wow I had this idea and it all thought out but I have to start again with an idea I don't have. So have ideas and backup ideas.

DK: And then marking? Having it remarked and the deadlines? How did that work for you?

SC: That, that helped quite a bit cause it gave me
 -> (8701-165): deadlines - adhere to
  -> (8701-743): deadlines - motivational
an idea of what level you were working at and when you needed to improve like in the documentation. Mr Gill would tell us Uhm… you need to prove him this
 -> (8881-86): feedback - provided
area or rather move this , this, or rather move this subject to go down here so it makes sense with all
 -> (9029-122): feedback - respond
of the document, but also like explaining what I was doing to my pseudocode it was overall easier because
 -> (9166-156): feedback - learn
it taught me about actual ordering, and have to jump in between methods of pseudocode so that was pretty good. You really just have to learn to talk code in
 -> (9323-123): programming - achievable
a document that is actually logical and easy to follow.

DK: So which was the hardest part doing the document or doing the code?

SC: Uhm… my code I found easiest with that method I
 -> (9549-171): difficulties - documentation
  -> (9548-799): perseverance - did not give up
used like Uhm… when it came to err… putting my code into a form that I could use in my documentation, that threw me off like as I said just now uhm… my
 -> (9777-185): difficulties - design
  -> (9777-185): difficulties - documentation
order of my pseudocode jumping in between methods like I'd have methods in my GUI which would use five maybe six other methods between different objects, so besides I would actually find alternatives which
 -> (9973-138): difficulties - design
  -> (9973-138): difficulties - documentation
   -> (9973-138): PS strategy - alternate

just became too much it was more jut jumping in between methods, because I would have to use this method but first I have to introduce using pseudocode so yeah that, that, that confused me.
DK: So that sequencing section didn't help you really, it was confusing?
SC: I started in the beginning and it was fine but as soon as I started putting in my methods and
 -> (10446-127): difficulties - documentation
  -> (10446-127): difficulties - documentation
fixing problems, it threw me all over the place because I had methods I had to refer to.
DK: Something I was supposed to ask you… Oh! So a project about grit and perseverance and that kind of stuff so tell me how you felt about persevering and did you persevere?
SC: Uhm…
DK: Did you give up easily and what did you do?
SC: For the programming side of things I didn't
 -> (10875-80): programming - achievable
really have much trouble Uhm… I found it quite easy its just logically plan things. When it came to the
 -> (11023-109): difficulties - documentation
documentation on the other hand though uhm… It got a bit of trouble, it became quite troublesome as I said the pseudocode because that really troubled me but my class, my class diagram as well, I'd be
 -> (11241-164): difficulties - design
  -> (11241-164): difficulties - documentation
changing things so often that I had to actually change a lot of information in my class diagram and documentation but by then I had lost track of the
 -> (11445-65): difficulties - documentation
  -> (11445-65): difficulties - lost track
   -> (11445-65): perseverance - gave up
things there in writing things down, and in changes
 -> (11513-103): difficulties - design
  -> (11513-105): difficulties - documentation
   -> (11513-105): emotional - frustrated
I had made so then I kind of got fed up with all the class diagrams so, so I lost two or three marks in that part.
DK: So then did you give up and not come back or did you persevere?
SC: I didn't give up I was just like I am so lost
 -> (11771-65): difficulties - coding
  -> (11771-66): difficulties - documentation
right now and I can't even go backwards and I'm tsss bleh and gave up eventually, I had just lost track
 -> (11898-19): difficulties - documentation
  -> (11898-19): perseverance - gave up

-> (11927-51): difficulties - documentation
of all the changes I had made like adding extra objects extra variables everything like when did that even get in there sort of thing
DK: And in terms of programming how often do you if you come across an error how often do keep trying and having a go or do you just try once and walk away.
SC: Do you mean like as I am programming.
DK: Programming, yes.
SC: Uhm… whenever it comes to an error I'll first
-> (12399-99): programming - correct errors
-> (12399-99): PS strategy - diagnosing problems
-> (12392-687): perseverance - did not give up
figure out what if it is me causing the error because sometimes I use the wrong Uhm… type like string instead of integer that's just because bad muscle memory uhm…uhm… when it comes to my syntax errors or errors like that as soon as I come across an error, I basically make a way of testing that so
-> (12758-153): programming - correct errors
-> (12758-153): PS strategy - diagnosing problems
-> (12758-153): PS strategy - alternate
I am sorting something I will have a way on the side of my users to immediately get to that, instead of having to go through the entire process so I get
-> (12975-104): programming - correct errors
-> (12975-104): PS strategy - diagnosing problems
-> (12975-104): PS strategy - alternate
straight to the problem, so there is nothing else that is giving me issues.
DK: So obviously it sounds like you got really good strategies in trying to code and keep track of everything and the class diagram fell apart when your strategy failed.
SC: No uhm… I was keeping track of what methods and
-> (13313-86): difficulties - coding
-> (13313-86): programming - correct errors
-> (13313-86): PS strategy - diagnosing problems
what alterations I was making to methods, but not in extreme detail I wouldn't know if I was incorporating any of your private variables or static variables.
DK: so if you had to go back and fix that problem if you coded again how would you fix that?
SC: I would probably keep track of what I have added
-> (13672-79): programming - correct errors
-> (13672-79): programming - kept track
-> (13672-79): PS strategy - diagnosing problems
as I just said uhm… like if I was adding a new variable in this class or a method that refers to

253 another method, I would kind of keep track of what
-> (13864-149): programming - correct errors
-> (13864-149): programming - kept track
-> (13864-149): PS strategy - diagnosing prob-
lems
254 changes I make to the variables as well and which
255 variables are used by the different methods, and
-> (14014-334): programming - correct errors
-> (14014-334): programming - kept track
-> (14014-334): PS strategy - diagnosing prob-
lems
256 when it comes to methods themselves what those
257 methods actually do because I was coding a method
258 called sort, meanwhile it's called getSort and I
259 would be calling this getSort method thing to sort
260 meanwhile I have made this method to add numbers
261 together or something.
262 DK: What happens if you could find and I think there
263 is tools that could do versioning and changes and
264 what if you go look in NetBeans.
265 SC: Like… like I know there is a class diagram
-> (14520-108): documentation - UML
-> (14520-108): programming - correct errors
-> (14520-108): PS strategy - diagnosing prob-
lems
266 generator I only found that out at the end though.
267 DK: Uhm… but I am say you found a tool on NetBeans
268 that could track changes and put dates next to stuff
269 and…
270 SC: I… I would quite like that uhm… because it would
-> (14780-196): difficulties - coding
-> (14780-196): programming - correct errors
-> (14780-196): PS strategy - diagnosing prob-
lems
271 be nice to be in an orderly change, instead of
272 jumping around like see this J with little arrows
273 running around uhm.. ya I would like that but when
-> (14979-186): difficulties - coding
-> (14979-186): emotional - build confidence
-> (14979-186): programming - correct errors
274 it comes to the actual tool to working out errors I
275 kind of like doing it myself because I feel it helps
276 me build my confidence.
277 DK: Alright anything else you want to add?
278 SC: not really Ma'am that's all.
279 DK: Did you enjoy it?
280 SC: Ja I did, it was great fun, I had my
-> (15304-84): emotional - enjoyed PAT
281 grandparents testing the riddles out.
282 DK: They are good users to try things on, apparently
283 they are good test subjects.

```
284  SC: I remember my laptop died so my Gran had been
285  staring at a black screen for a while.
286  DK: Shame you got to consider that in your design
287  code. It is always important to consider all…
288  SC: I think next year when I revisit my PAT uhm… I
      -> (15726-265): PAT - improvements
289  would rank my, I would rank my riddles according to
290  difficulty but I would also change how they would be
      -> (15856-453): difficulties - time consuming
291  entering, how answers are entered in, because the
292  way I was doing it, is I would have to come up with
      -> (15987-279): difficulties - time consuming
           -> (15987-279): PAT - improvements
293  possible answers for the riddles and they had to be
294  ones that could actually be considered and that was
295  quite time consuming so to just think of possible
296  answers, so basically trying to answer a riddle with
297  something that could be right, but it's wrong and
      -> (16311-248): difficulties - time consuming
           -> (16311-248): PAT - improvements
298  that wasted a lot of time like I would be sitting
299  once I had done my work in class with an exam pad
300  out and some riddles that I had written out trying
301  to come up with possible but wrong answers.
302  DK: You could just google them.
303  SC: Yes you could get possible answers but, like but
304  you never get things like the correct answers but
305  you never get possible answers. Ja so, that was
306  quite time consuming, it helped asking my
307  grandparents because they would be like do this. So
308  ja, that helped just taking exam pads and doing
309  riddles
310  DK: Wooh thanks sweetheart.
311  SC: No problem Ma'am
```

Interview – Student D

```
int_3.atx
 1  SD
 2  SD: I'm SD and I have done a memory game.
 3  DK: So tell me about your game.
 4  SD: So basically the user will play the game and the
 5  user selects two… tiles. The tiles flip over uhm… if it is
a
 6  match they will disappear, and if it isn't they will flip
back
 7  over and the user will play until all the tiles are
 8  disappeared.
 9  DK: And how did you find the project?
10  SD: It was, it was tough uhm… I struggled a little bit
      -> (477-110): difficulties - coding
```

```
          -> (477-110): perseverance - did not give up
11  with the programming but yeah I got through it.
12  DK: So what did you struggle with?
13  SD: The actual programming like understanding how
      -> (643-182): difficulties - coding
           -> (643-182): difficulties - design
14  everything works and, like I didn't struggle with it but it…
15  it was not struggling it was figuring out a solution.
16  DK: Ok so syntax errors is that okay?
17  SD: Yeah
18  DK: And then you started getting runtime errors.
19  SD: Um… it was more the way I'm doing it so logical
      -> (958-69): difficulties - coding
20  errors.
21  DK: Ok so when you got a logical error, what would you
22  do?
23  SD: Well then I would sit down right, write what I am
      -> (1122-134): perseverance - did not give up
           -> (1122-134): PS strategy - plan solution
24  doing, what I want to do, how I approached it, and all
25  that.
26  DK: Then what were your steps you took from there?
27  SD: So basically what I would do is I would write down
      -> (1328-234): programming - correct errors
           -> (1328-234): perseverance - did not give up
                -> (1328-234): PS strategy - plan solution
28  what I wanted to do, then I would see what I had done, if
29  I have an error I would see what I have done, see, try
30  figure out what the error is and fix it.
31  DK: You are trying to work, so you wrote down what you
32  wanted to output?
33  SD: Yes, and what I wanted to do, so what I want, the
      -> (1668-85): programming - correct errors
           -> (1668-85): perseverance - did not give up
                -> (1668-85): PS strategy - diagnosing problems
34  output I wanted it to be.
35  DK: Ok so now you get an error what would you do next,
36  so it's not working, so did you start line by line, or did
37  you did you do a trace table?
38  SD: No, well I would first write out in pseudo code or
      -> (1938-130): programming - correct errors
           -> (1938-130): perseverance - did not give up
                -> (1938-130): PS strategy - pseudocode
39  something like that, what I wanted to do.
40  DK: ok, alright.
41  SD: And then when I have got what I wanted to do,
      -> (2078-214): programming - correct errors
           -> (2078-214): perseverance - did not give up
                -> (2078-214): tracing - using NetBeans
42  which is generally what is generally what my code is,
43  then I would try find the error using this, um the
```

debugging thing on NetBeans.
DK: Ok so you use the debugger a lot.
SD: Yes .
DK: And did that help?
SD: It did yeah.
DK: It was not too time consuming?
SD: Well we had quite a bit of time so…
  -> (2469-42): programming - correct errors
    -> (2469-42): difficulties - time consuming
      -> (2469-42): perseverance - did not give up
DK: You were fine with that?
SD: Yeah.
DK: So what kinds of errors did you find, what is your
main errors?
SD: Just logical.
DK: Logical, so you wanted to do x and it was doing y.
SD: Yeah.
DK: And objects and all the stuff we had done up until
that date did it help did you go back to it?
SD: Yeah, definitely, especially the parallel arrays and
working with that.
DK: And then, and the 2D arrays.
SD: Yes.
DK: So um… if you came across an error did you give up
straight away or did you sit down and…
SD: Um… it depends I guess how my mood is um…
  -> (3160-222): perseverance - did not give up
    -> (3160-222): PS strategy - asked friends
generally I would try a little bit and after a while I would
give up or maybe go ask for help from somebody else.
DK: And who did you ask for help?
  -> (3390-158): perseverance - did not give up
    -> (3390-158): PS strategy - asked friends
SD: Um… multiple people I sometimes ask my dad
because my dad is quite good with logical um… problems
otherwise just friends in general.
DK: And did that help?
SD: Yes.
DK: Did it work?
SD: Definitely yeah.
DK: So you kind of created a support structure around
you that you could go to.
SD: yes
DK: or, so first you try yourself.
SD: Yes
DK: And then you would go to your friends, not the other
  -> (3841-41): perseverance - did not give up
    -> (3841-41): PS strategy - not friends help
way around
SD: No
DK: Ok Perfect, so then um, so, did you enjoy the

project?
SD: I enjoyed it, it was tough though, yeah, but it was
  -> (4022-107): emotional - enjoyed PAT
    -> (4022-107): perseverance - learnt to code project
enjoyable, learn how to make a game, yes.
DK: And um, so how did the marking and deadlines help
you?
SD: It kind of gave us like a, a motive to finish it.
  -> (4217-55): deadlines - adhere to
    -> (4217-55): deadlines - motivational
DK: Ok
  -> (4294-109): deadlines - adhere to
    -> (4294-109): feedback - multiple submissions
SD: so, and also being allowed to hand it in multiple
times, also allowed us to get better marks.
DK: And did, did you work with that?
SD: Yes.
DK: So every time you got it marked you went home and
  -> (4477-100): deadlines - adhere to
    -> (4477-100): feedback - learn
      -> (4477-100): feedback - multiple submissions
fixed it, and all of the rest of it.
SD: Yes.
DK: And how was the documentation?
SD: What do you mean?
DK: So some people said they got the program running
well, but the documentation was difficult, or vice versa,
which was easier for you?
SD: Documentation was much easier.
  -> (4846-36): documentation - achievable
DK: And the code you battled with?
SD: Yes.
  -> (4938-6): difficulties - coding
DK: And which did you, which order did you do it in?
Code it first then document?
SD: I did it, I did the documents first, but also built up,
  -> (5053-186): documentation and coding - simultaneous
built them up together. So if I did change something in
my program, I would then change it on my document.
DK: Ok. At the same time?
SD: Yes.
DK: So you didn’t leave it and come back later, oh hell?
  -> (5301-75): documentation and coding - simultaneous
SD: no
DK: And the documentation was fine.
SD: yes
DK: And you didn’t really battle with that? And the
pseudo code?
SD: um, it was ok, it’s a little bit hard to well, not
  -> (5529-174): difficulties - documentation
    -> (5529-174): perseverance - did not give up

understand but to actually write it, it's quite challenging to remember which words are for which.
DK: Ok my project is about perseverance and carrying on and all the rest of it, do you think you persevered because it sounds like you did?
SD: Not as much as what I could have, I think I could
-> (5882-112): perseverance - could have done more
-> (5882-112): perseverance - did not give up
have made the game a lot better then what it is.
DK: What did you get for your project in the end?
SD: Um… I think eighty something.
DK: So you think you could have done well?
SD: I think I could have got full marks, I think I lost two marks on documentation and then… no I got full marks for my programming and lost two for documentation so I got ninety, but I'm sure I could have fixed documentation.
DK: So what was it that stopped you time or just…
SD: I think I was just fed up with constantly having to
-> (6493-72): perseverance - gave up
do it.
DK: So if we had given you longer time would it have helped you?
SD: I don't think so.
-> (6659-23): deadlines - adhere to
-> (6659-23): perseverance - gave up
DK: Would you have just been fed up?
SD: I would have just left it for longer periods.
-> (6735-52): perseverance - procrastination
DK: Okay and what improvements would you have liked to if you could've. I mean you obviously just mentioned that you could have fixed your documentation but if you had to do it for the matric what would you like to add in?
SD: To my game?
DK: yeah.
SD: um well, it's quite, I think the games, it is basic, so I
-> (7101-375): PAT - improvements
would like to make it more complex, so adding more, maybe making the grid a bigger, or adding different themes to it, or something like that, so making um so instead of matching colours, you will match computer components, and things like that, to educate people.
DK: Ok, did you know how to do all the things before you started?
SD: With what?
DK: So when you started this project, did you have, I could do all of this? Or did you start and think oh jeez I'm going to…
SD: I kind of worked in the steps, so I would say ok, this

-> (7748-438): PS strategy - asked friends
-> (7748-438): PS strategy - plan solution
is what I want to do, I want to do a memory game, and then kind of said ok so what first do I need to do? If I can do it, I would try, and if I couldn't I would go ask for help.
DK: So did the, were those steps that you defined, did you define them, or did they come from the rubric and the way we were going in class?
SD: The programming, I did it myself, but for the
-> (8196-132): PS strategy - plan solution
documentation I will go through and do each one step by step.
DK: So did the class lessons help? We didn't do many, most of the time you were just working on your own. That gave you a kind of structure to work in.
SD: For the Battleships?
-> (8518-104): Scaffolding Battleships useful
-> (8518-104): PS Strategy - use other code
DK: yeah
SD: um, I did use some other code to build my game.
DK: And that was nice. Is there anything else you want to say?
SD: Nag not really.
DK: Ok as long as you put your name in.
SD: *mumble*
DK: I just need to record you, to know who you were

SD:

Interview – Student E

int_2.atx

SE: Student E that's my name and my nickname was Student E. Let me tell you about the project.
DK: Describe what your project did.
SE: Ok so my project was, it was called the Last Farmer, and basically you were the last farmer on earth due to global warming and global catastrophes or whatever, and you had to, you had four little plants and you had a little grid of two by four, and then half the grid was in the sun and half was in the shade, and you had to water your plants and move them in and out of the sun and the shade to prevent them from burning and freezing. And if all four of your plants grew to adulthood then you passed the level, but if one of them died then you didn't pass the level and there about six levels and yeah high scores and you could rate the game, that's in general, yeah.
DK: Ok, did you enjoy it, tell me about how you felt about the project.

18 SE: Um, like, I enjoyed it, like I enjoy IT in general and I
-> (1074-99): emotional - enjoyed PAT
19 enjoy programing and all that stuff, I also like the creative
-> (1175-156): difficulties - topic choice
20 part of it where you could like sell your own game and
21 everything at first I didn't really know what I wanted to do
22 I didn't know if I wanted to do a game, then I had the idea
23 and I was like you know I might as well just do it, it would
24 be fun so I did enjoy it uhm… obviously there were times
-> (1517-231): difficulties - coding
-> (1517-231): perseverance - did not give up
25 where I was a little bit stuck but I just went through it and
26 I think I actually enjoyed when you were stuck and then
27 become unstuck I guess so you enjoy that part as well.
28 DK: So what did you do when you were stuck what was
29 the strategy because everyone has a strategy.
30 SE: Well say for example I didn't know how to do
-> (1879-457): difficulties - coding
-> (1879-457): PS strategy - alternate
-> (1879-457): PS strategy - search Google
31 something like if I wanted to get this value from the array
32 or whatever or from the action events then I would go
33 and google it if I didn't know how to do something but for
34 example say I was trying to shift my plants in the 2D array
35 and something was just going wrong and I knew that it
-> (2247-418): difficulties - coding
-> (2247-418): PS strategy - alternate
-> (2247-418): PS strategy - search Google
36 wasn't necessarily an error, a programming error it was a
37 logical error so then a logical error I would just kind of use
38 system.out.println statements and just kind of force it
39 um… just, just force almost trying to figure out what was
40 wrong it was, it was throwing like some unknown
41 exception that I would google that google was my best
42 friend, ha ha.
43 DK: The online tracing, the tracing thing, the watches on
44 the variables that makes you trace that gets the computer
45 to trace, and then.
46 SE: I don't think I have ever done that.
-> (2866-39): tracing - not using online
47 DK: Then you just slip it out the printer.
48 SE: Yeah well yeah normally so like for example, it like I
-> (2972-348): difficulties - coding
-> (2972-348): difficulties - design
-> (2971-630): perseverance - did not give up
49 originally didn't have an object array, I had a bunch of 2D
50 arrays, like parallel arrays, then I decided to make it an

object array, but then like say for example I forgot, so I switched the plants sun button and water button and all that stuff, then I forgot to switch something else, then it wouldn't work, and I wouldn't know why, so then I would
  -> (3397-207): difficulties - coding
    -> (3397-207): tracing - println
go and print out the state of which plant this is and which different thing , and then I would kind of figure out from there, I guess.
DK: Then you would just println …
SE: Yeah, like I had a I had a good idea of how my code works, so I could kind of pick out errors that were going on and it felt like I had just copied and pasted stuff and didn't really know how it was working.
DK: And when did you walk away, and when was it like just not working, would you?
SE: OOO sounds cheesy but never I , I figured I, if I
  -> (4016-208): perseverance - did not give up
    -> (4016-211): programming - correct errors
couldn't figure it out I would go and comment it out and redo it and see if the error still comes and if it doesn't I don't know.
SE: Yeah
DK: And then the marking and the deadlines did they work.
SE: Uhm… it did I did a lot of rough drafts and stuff and I
  -> (4339-370): deadlines - adhere to
    -> (4339-370): feedback - learn
      -> (4339-370): feedback - provided
had like three or four for each little thing and I like that you could go and hand it in, and it gets they like look at it and mark it and you could come and change your stuff, I didn't think the deadlines were too early or late or anything, I think they were fine.
DK: And if you had more time to do this project would you have done any better?
SE: Uhm… I could have added some more stuff I guess if
  -> (4826-239): deadlines - adhere to
    -> (4826-239): PAT - improvements
I had more time but I want to save some stuff for matric so I think the time I had was fine, well that is from me so some other guys might struggle with it.
DK: So what did you do first document or code or run them in parallel?
SE: I know you are supposed to do the design or specs
  -> (5169-569): documentation and coding - code first
document or *mumble* one of the document first and then do your thing and then, what I did was I literally I think just coded it, and then I worked backwards to get

90 the documents, just cause I had an idea of how I wrote
91 the code and I just did it, well I don't know if its lazy, it's
92 just, the way I think you are supposed to do it is just get
93 your specs document then your design then go code from
94 there, but I just coded first, yeah.
95 DK: So perseverance when you got two errors your thing
96 was you were going to kill it, you were going to find out…
97 SE: Yeah, yeah and if I was really super stuck, go and go
-> (5887-130): difficulties - coding
-> (5887-131): perseverance - did not give up
98 and comment it out and redo it, see what might be
99 wrong.
100 DK: Did you ever ask people in the class?
101 SE: Um, the classes that I was doing, I don't know if a lot
-> (6085-230): difficulties - coding
-> (6085-230): perseverance - did not give up
-> (6085-230): PS strategy - ask teacher
102 of the guys could have helped me, but I might have asked
103 the teacher if it was an unknown , or my teacher if it was an
104 unknown, like error coming up, and if he wasn't there I
-> (6317-93): difficulties - coding
-> (6317-93): perseverance - did not give up
-> (6317-93): PS strategy - search Google
105 would go and Google it just because it might be easier to
106 ask the teacher cause he might give you a nice, um,
-> (6422-190): difficulties - coding
-> (6422-190): perseverance - did not give up
-> (6422-190): PS strategy - not friends help
107 simple description, whereas the thing, but nah I didn't
108 really ask for anyone's help, not really, I think I did at all.
109 DK: And, what improvements would you do if you could?
110 SE: Um well, what happened was I did the documents, I
-> (6690-957): PAT - improvements
111 didn't have a pause button, for my game so that obviously
112 time is running, and then so I put the pause button in, so
113 it could like stop the times and restart them, that was the
114 one thing I added, I don't think it was in my documents
115 though, I will probably add that in, next year whatever, I
116 have already made a rate the game, I might change, I
117 think I am going to change the graphics to look a bit
118 smoother and better, and I might change, like I might add
119 in types of plants, so this type of plant will need more
120 water than sun, maybe, I will see how complicated it is,
121 otherwise I will find something else to add on, but I think
122 the game, the game is pretty much complete maybe I can
123 just add something cool, maybe something with a bit of
124 vibrancy. I don't know.
125 DK: And did you know, did you have an idea of how to do

this game before you started?
-> (7758-479): PS strategy - learn to code independently
SE: um, I had heard of 2D arrays, and then I kind of looked it in but I didn't really know how to use them, you need to know how to do, one of the guys I played polo with and swimming, he says I must learn to use 2D arrays properly so then I just went on the Internet and I just found that I could, could do it quite easily but it was just, wait what was the question again…
DK: Did you have quite a good idea of how to do this before you started?
SE: Yeah, I did, I think the idea for it came, like the rules
-> (8305-520): perseverance - did not give up
-> (8300-525): PS strategy - learn to code independently
of the game were there quite solidly uhm… I just needed to put it onto the code and I knew how to do the, to do the code, it was just a few things like I didn't know how to use the timer class, so I just Googled it and figured out how to do it, it wasn't really that hard, I didn't, I didn't not know how to do it and when I just went it was just fixing little errors as I was going through.
DK: And templates, what about the templates, did they help?
SE: Uhm… the templates for the documents?
DK: The Battleships and the documents.
SE: Uhm… the documents template really helped me a
-> (9015-413): Scaffolding documentation useful
lot, just because I didn't know the format really I didn't know exactly how exactly how they wanted it set out, um then I go based on the template but then the templates, when I hand it in it might now be necessary then I go make my own, edit the templates to be like proper , um and the Battleships, I didn't really need , I think I could
-> (9429-268): Scaffolding Battleships useful
have got on without the Battleships, well I don't know if some of the guys in my class could have, but I think I could have , I could have done my thing without even doing Battleships, the 2D arrays it's not super hard to, um I'm going to come back to that later, so that the template I defiantly needed just so to help me set it out and stuff, but I don't think I need the Battleships.
DK: Cool, anything else?
SE: Um not really, I enjoyed it, I like making my own
-> (9975-76): emotional - enjoyed PAT
-> (9975-1409): perseverance - did not give up

164 little projects, I think the 2D arrays it's good that, like
that
-> (10052-421): Scaffolding data structure useful
165 you making them do that 2D arrays in the PAT cause it is
166 like useful to use and stuff, you know what I mean, um
167 obviously it's useful to all games you are going to use,
so if
168 we were like doing a PAT without a 2D array it wouldn't
169 be as much of a learning experience as if you were doing
170 it with a 2D array, uhm… yeah just I struggled like I know
-> (10473-580): difficulties - coding
-> (10473-580): PAT - able to sep back-end
-> (10473-580): perseverance - did not give up
171 with games the logic or the back-ends is the easiest part,
172 it's the frontend where you have to make it look pretty,
173 I'm just quite a perfectionist so I want everything to look
174 nice I don't want there to be funny things hanging out and
175 to see the buttons and stuff so I think I spent a lot of
the
176 time figuring out how to get an image to be on the button
177 and how to make the button like see through so it doesn't
178 look like it's a button it looks like a actual thing, how
can
-> (11051-309): difficulties - coding
-> (11051-309): PAT - able to separate back-end
-> (11051-309): perseverance - did not give up
179 get this background like this, how can I prevent the thing
180 from shortening the background or making it bigger or
181 whatever so the a lot of the time was spent looking at the
182 graphical part of it which I think is, is the graphical
part is
183 just finicky.
184 DK: And like everything we did in the first two terms, did
185 that help you, was it easy to refer back to that
186 knowledge?
187 SE: Uhm… I think the object orientated programming
-> (11518-1117): difficulties - coding
-> (11518-1117): PAT - able to sep back-end
188 helped because what I was doing is I was using parallel
189 arrays for all the different plant things and I was like
why
190 don't I just make an array of plants, I used to have, I
mean
191 like also the splitting into the front and back-end and
also
192 helped, because I had the one class that was really
193 complicated, and I thought it was going to be even more
194 complicated to change into the back-end but you just
195 copy and paste, it's really easy, so like I had, I used to
196 have an array of the plants grow state, sun state, the
197 whatever state, the buttons , the whole things, and I just

198 went and put in one, the buttons and the, the buttons
199 were still their own separate thing but all the *mumble*
200 in the other one, it didn't take me long to figure out, I
just
201 realised that I had so many arrays and I could just make
it
202 one, and I see you have been doing that the whole year,
203 like ok I'm going to do that, *mumble*
204 DK: Well done, yeah
205 SE: um, not really, I enjoyed it , um was the diary for
206 you?
207 DK: yeah, its got to go with this to see what you thought.
208 SE: oh ok. I tried to do as much of the diary as I could
209 DK: so yeah, its fine. Let me turn this off.
210

Interview – Student F

int_5.atx
1 DK: And tell me about your project.
2 SF: I'm Student F …umh…my project. Mine was
3 quite challenging at times …umh… there would be a lot
-> (144-115): difficulties - coding
-> (144-115): perseverance - did not give up
4 of times where I would also come to problems where I
5 find I would have to try and research obviously to try
-> (260-69): PS strategy - search Google
6 and fix the problems…umh… I would also ask other
-> (330-116): PS strategy - ask teacher
-> (330-116): PS strategy - asked friends
7 resources such as other people…uhm… Mr Gill as well as
8 a resource. Uhm.
9 DK: So, what did your project do?
10 SF: My project. Mine was, it was a game that, that was a
11 riddle game. So, you would have a certain amount of
12 time to answer a certain amount of riddles and then as
13 soon as you finished the game you've won and if you
14 don't get the certain amount of each riddle in each
15 round correct you don't move on to the next round.
16 Uhm…and you have to get points accumulated for each
17 riddle that you answer correctly obviously. Uhm…ja.
18 DK: And are you happy with it?
-> (1037-182): emotional - enjoyed PAT
-> (1037-182): PAT - creativity
19 SF: OH, no. Yes I was, I was. It was really fun. It was an
20 interesting task. Uhm…it's definitely something that gets
21 you more interested in creating and stuff.
22 DK: Ok. So tell me …uhm… the deadlines and the
23 marking. How did that help you?
24 SF: Uhm… when there were usually when I find when I
25 have deadlines and it helps me keep up to date with

what I need to get done and it also helps me separate like what certain tasks I need to do first. So like let's just say you need to get to your obviously your research task of, well the part we had like the specs and stuff usually helps you to first also design your game for the design
-> (1733-454): deadlines - motivational
-> (1733-454): feedback - learn
-> (1733-454): feedback - multiple submissions helpful
document and then also you can do your game and see where you can go wrong cause what I realised half way through was that I had, I had done my game and my specs and had to often, once coding it, realised that I had to go back and change my design or specs document cause they weren't the same as I had actually landed up programming.
DK: So, did you do it straight away or did you leave it when you made that change?
SF: Uhm…I first did the documents, well the one
-> (2297-401): documentation and coding - simultaneous
document, the specs document. Then I did the programming...uhm… and then the design…like together. So I landed up doing the programming and then going back to the design and saying ok, I need to change things up cause I did it differently in the game and same with the specs document as well.
DK: So you sort of did the design, did the code. The code changed. Went back to the design.
SF: Ja
DK: And that was ok?
SF: Ja, it was fine, ma'am. I still landed up with the right end product.
DK: So you were kind of working with them in parallel and switching between them?
SF: Ja, I keep on updating them.
DK: And the marking and having multiple submissions and all that? Did that work?
-> (3199-351): documentation and coding - simultaneous
-> (3199-351): feedback - learn
-> (3199-351): feedback - multiple submissions
SF: Ja, it also helps. It does cause then you know that, ok look I know where I am going, I'm going right and I just see where I am going wrong and try and improve it obviously for the next time and just keep on trying to improve it until we get to like a 100% of what you think your project will be.
DK: Ok so now tell me. What did you do when you can to errors? How did you deal with errors?
SF: It depends on the error cause sometimes, most the
-> (3672-337): difficulties - coding

-> (3672-337): perseverance - did not give up
-> (3672-337): PS strategy - research
time I try and sit down and try and get it to work with the errors that I get. A lot of them were logical errors because the syntax errors I would get I would just look up and I would get the answer. So I would be able to correct it.
DK: So where would you look it up? In google or the textbook?
SF: I would use google obviously.
-> (4099-36): difficulties - coding
-> (4099-36): perseverance - did not give up
-> (4099-36): PS strategy - research
DK: OK
SF: Uhm, ja and then. Well either google or maybe I
-> (4158-517): perseverance - did not give up
-> (4158-517): PS strategy - asked friends
-> (4158-517): PS strategy - search Google
would ask like a friend or something for help or to fix a syntax error. Logical errors I would sometimes, I would find myself having a logical error and then like I would try and correct it but I can't and then I would land up, land up having to change my whole code just because of a logical error. Uhm…so I wouldn't, I wouldn't know how to do it like I don't know how to get the answer so…ya.
DK: So you just, so you just changed the code? You couldn't fix it?
SF: Ja, I would just change my code but usually I would
-> (4771-373): deadlines - not able to adhere
-> (4771-373): perseverance - did not give up
-> (4771-373): PS strategy - research
sit down and actually try or look it up, try and find answers but if I can't really or find that I don't really have enough time to get it done cause obviously there's time limits and stuff then I would just move on to the next thing and also try and change my code.
DK: You didn't trace tables or println out stuff or…
SF: Oh, yes I would do that as well obviously I would try
-> (5216-328): PS strategy - search Google
and debug my program as well by myself but then if I couldn't find it as well then I would just change it.
DK: So what was your strategy? You would find the error and try and fix the error and then you would go to google.
SF: Yes.
DK: Then you would println out?
-> (5564-39): tracing - println
SF: Yes.
DK: And then trace table?
-> (5630-43): tracing - not using online
SF: Yes.

104 DK: Did you use the online tracing thing?
105 SF: No uhm, I would just use printlns and stuff. I would
-> (5735-98): tracing - using paper
-> (5735-98): tracing - not using online
106 try trace it in my head almost.
107 DK: Ok
108 SF: Uhm ja like loops and printlns and stuff. I would use
-> (5858-160): tracing - using paper
-> (5858-160): tracing - not using online
109 that to help myself and then if I can't find it then I would
110 just change my code.
111 DK: You wouldn't go to friends next?
112 SF: No, I would have done that like second.
113 DK: Hahaha, so friends are second?
114 SF: Ja
115 DK: Ok, so then you try fix it yourself? And when would
116 you ask friends? In class? Or would you ask them at
117 home and email them?
118 SF: Well I would ask them over message like WhatsApp
-> (6344-168): PS strategy - asked friends
119 or during class. It depends what I need to fix at a certain
120 time and where they are obviously.
121 DK: So who you can get hold of at the time?
122 SF: Ja
123 DK: And then in terms of persevering and carrying on
124 and trying. What would you… How long would you carry
125 on, on an error before you try something else? Ok,
126 obviously your strategy was, ok I have tried all these
127 options I am going to give up and try another way of
128 solving it.
129 SF: Well, I think when you get an error like this there is
-> (6929-370): perseverance - did not give up
-> (6929-370): perseverance - gave up
130 only a certain amount of time you can on that error
131 before you realise, ok you're not going to get it right
132 let's try and find a different way to try and get your end
133 product. You know what I mean? So, like ja it's to do
134 mainly with time that what I would think.
135 DK: And sports wise are you really busy?
136 SF: uhm, ja I was . So, I had to think about all my other
-> (7358-702): perseverance - did not give up
-> (7358-702): PS strategy - diagnosing problems
137 school things and stuff and sort of lengthen it out. See
138 when I am about to work with certain parts of the
139 project, when will it be better…uhm…like with the
140 coding most of it I did I would work with like asking
141 questions mainly about my code actually during school
142 and trying to understand where my problems are during
143 school cause that's a lot easier and when I got home I
144 would try and actually just code it properly without the

errors, well I would try and fix the errors proper…uhm…and then I would also do documentation at home.
DK: So you would do it in class cause you would have access to your friends in class rather than at home?
SF: Ya, I have access to resources and stuff that can help
 -> (8195-93): perseverance - did not give up
      -> (8195-93): PS strategy - research
with the syntax errors.
DK: Uhm…and the improvements if you had to improve it for next year what would you do?
SF: Uhm… for me there was in my project I couldn't get
 -> (8405-746): perseverance - did not give up
      -> (8405-746): PS strategy - diagnosing problems
a…there was a logical error that I landed up having at the end, it's really small, you only pick it up at the end of the project cause it only shows really at the last round. Uhm…but I would try and fix that, I think it is also to do with my time limit and stuff, maybe that messed it up. I will try and fix that up but that was also cause I didn't fix that up because I didn't have time to do that. So I'd probably work on that thing, the logical error …uhm… that's to try and make it a little more user-friendly, I guess wherever I could. It's probably the best thing you can do.
DK: But would you want to add more stuff on to it like network it or do something really weird?
SF: Well I mean you could do a, maybe like an
 -> (9276-444): PAT - improvements
multiplayer kind of thing where you can have 2 answering the riddles at the same time see how many can get the right answers, compare the answers and stuff, how many they get. So you could do something like that I guess. The thing is you would have to be strict with the riddle quiz game cause there is not much you can do also.
DK: You mean that it's difficult to extend?
SF: Ya, its…
DK: Now the templates, did that help you? Like the Battleships game did that help and the templates of the documents?
SF: Uhm…I think its helps with you being able to set the front-end and back-end …uhm…it keeps, it gives you that concept of really keeping the user interface and stuff on one side and your hard core programming on the one side cause then that makes it a lot easier for other programmers to read and for you to work with it. When you are mixing them with each other then it doesn't really …uhm…its harder to work with. It makes it less dynamic in a way …uhm… so the templates did help for

that reason, I mean it also helped for the normal coding of like using arrays and stuff …uhm… My game wasn't ready to, wasn't similar to that so it maybe just helped with the front and the back-end.
DK: More like the conceptual idea of how to set it out?
 -> (10829-200): Scaffolding documentation useful
SF: Ja, I mean if I was doing maybe a collision detection thing it might of helped even more with that kind of game. It all depends on what kind of game you are doing.
DK: And then anything else you want to mention about the project? Or how you felt it went? Or …
SF: Uhm…I don't really have anything else. I think it was
 -> (11146-92): emotional - enjoyed PAT
just really fun.
DK: Oh sorry I didn't ask. The first and second term did that help you? What we did in…the coding and all that. Did that help you get this thing going or was it quite a struggle?
SF: With all the arrays and stuff?
DK: Ja, the arrays and the objects and the …
SF: Umm…I think, I think it did cause by time you get to
 -> (11556-561): emotional - enjoyed PAT
  -> (11556-561): Scaffolding documentation useful
the end of…uhm…terms you know all the code already in your head cause you do a lot of it as repetition, you do work quite often. So, it definitely helps with the speed of how you can program and also your thought because also when you have a lot more code to use it is a lot easier…uh…cause you can find different methods as well. It's not just one route to get an answer of something. And also that really did help.
DK: And then…did you know how to do this project when you started? Did you have quite a good idea of how to do it or was it…uh?
SF: Well my first idea, I dint know what I was going to do
 -> (12276-628): perseverance - did not give up
  -> (12276-628): PS strategy - plan solution
…uhm… but as soon as I started thinking about it then I realised what kind of areas I could start with …uhm…Like I said when I did my first document I just kind of had the basis and then as soon as I did it I thought of things like maybe I can add things on, maybe I can make that a bit more user-friendly by doing that. So I questioned it along the way in other words. So, uhm…I kind of had an idea of what I was going to start with and I just build on to it as it goes.
DK: What was your first idea?
SF: Uhm…well I may different other project ideas as well. I mean there was I stage when I was …uhm… there was a stage when I was going to do some other games, I

don’t know what they were.
DK: Why did you discard them?
SF: I discarded them cause I realised they were too
-> (13211-343): PAT - too ambitious
hard and realised I wouldn’t have time to get through all the coding. Ya, and then I just choice the easiest option which is the riddle cause then I also know about databases and stuff so I could incorporate that in with my program.
DK: So you knew you were able to code with this so that is what you chose?
SF: Ya, because so at the begin I wanted to do something with collision detection but then I looked it up and I thought it might just take I little bit too long to understand and because I am a little bit better with the quiz type game cause all the coding around it I decided to just go after that kind of game first.
DK: Now, last thing. If you had to tell somebody, some poor grade 10 came to ask you for advice what would you tell them?
SF: I would say if you are gonna do a project, your first project always start with something you will be able to manage. Don’t over shot, I mean a lot of people they aspire to have great ideas like to do amazing projects, make like a call of duty or something but it takes ages to do that. You need to start really small. Like it’s the basics and the hard programming you need to do small simple programs but that work efficiently. Those are the best to start with and then obviously over the years you can try and get better, improved.
DK: Cool, thank you sweetheart.
SF: Pleasure
DK: You got great ideas.
SF: Thank you

Summary of the Codes

Version 1

adherence to deadlines
alternate ways of doing things
build confidence
create pseudocode
creativity in programming
delete methods with errors
determine cause of error
did not give up
documentation
enjoyed project
error experienced when coding

feedback
fix errors as coding
fix errors from bottom up
fixing errors
fixing problems in pseudocode
gave up
generate UML
gets frustrated
got stuck
help from google
incorrect data structure
invent a strategy
keeping track
learn from feedback
lose track
manually tracing
no problems in coding
not sure of PAT topic
not use trace tables
online tracing not using
order of methods in pseudocode
parallel coding and documenting
print errors
problem-solving
problems coding
problems converting code to pseudocode
problems in documentation
problems with representing class diagrams
project too ambitious
recheck content
respond to feedback
scaffolding
time consuming
traced manually
use of content taught
versioning tool
ways to improve PAT

Version 2

code development - kept track
code development - lost track
code development - stuck
correct - errors
correct - recheck content
deadlines - achievable
deadlines - adhere to
deadlines - motivational
deadlines - not able to adhere
deadlines - not enough time
diagnosing problems
difficulties - class diagrams

difficulties - coding
difficulties - converting code to pseudocode
difficulties - creating pseudocode
difficulties - documentation
difficulties - logical err
difficulties - matching design to code
difficulties - sep back end
difficulties - time consuming
difficulties - versioning
difficulties - finding solution
documentation
documentation - achievable
documentation - UML
documentation and coding - code first
documenting and coding - simultaneous
emotional - build confidence
emotional - depressing
emotional - enjoyed PAT
emotional - enjoyed project
emotional - frustrated
errors - coding
errors - correcting
errors - incorrect data structure
errors - pseudocode
feedback - did not read rubric
feedback - learn
feedback - multiple submissions
feedback - multiple submissions helpful
feedback - provided
feedback - respond
initiative - learn code independently
PAT - able to sep back end
PAT - consider changing game
PAT - creativity
PAT - improvements
PAT - too ambitious
PAT - unsure of topic
PAT battleships - useful
PAT data structure - useful
PAT templates - useful
perseverance - learnt to code project
perseverance - could have done more
perseverance - did not give up
perseverance - did not listen in class
perseverance - gave up
perseverance - procrastination
perseverance - take a break
prior knowledge - useful
programming - achievable
PS strategy - ask teacher
PS strategy - asked friends

```
PS strategy - comment out
PS strategy - compare output
PS strategy - delete methods with errors
PS strategy - determine cause of error
PS strategy - fix errors from bottom up
PS strategy - found shortcuts
PS strategy - invent
PS strategy - not friends help
PS strategy - paper plan
PS strategy - plan out
PS strategy - pseudocode
PS strategy - research
PS strategy - search Google
PS strategy - search YouTube
PS strategy - too shy to ask for help
PS Strategy - use other code
PS strategy - work in steps
PS strategy - worked out solution
pseudocode - easy
pseudocode - method order
scaffolding
strategy - alternate
tracing - manually
tracing - no tracetables
tracing - not using online
tracing - println
tracing - using NetBeans
tracing - verbally
```

Version 3

```
code development - achievable
code development - correcting errs
code development - correct errors
code development - kept track
code development - lost track
code development - stuck
deadlines - achievable
deadlines - adhere to
deadlines - motivational
deadlines - not able to adhere
deadlines - not enough time
difficulties - class diagrams
difficulties - coding
difficulties - converting code to pseudocode
difficulties - creating pseudocode
difficulties - documentation
difficulties - incorrect data structure
difficulties - logical err
difficulties - matching design to code
difficulties - method order
```

difficulties - pseudocode
difficulties - sep back end
difficulties - time consuming
difficulties - versioning
difficulties - finding solution
documentation - achievable
documentation - UML
documentation and coding - code first
documentation and coding - simultaneous
emotional - build confidence
emotional - depressing
emotional - enjoyed PAT
emotional - enjoyed project
emotional - frustrated
feedback - did not read rubric
feedback - learn
feedback - multiple submissions
feedback - multiple submissions helpful
feedback - provided
feedback - respond
PAT - able to sep back end
PAT - consider changing game
PAT - creativity
PAT - improvements
PAT - too ambitious
PAT - unsure of topic
PAT battleships - useful
PAT data structure - useful
PAT documentation - useful
perseverance - learnt to code project
perseverance - could have done more
perseverance - did not give up
perseverance - did not listen in class
perseverance - gave up
perseverance - procrastination
perseverance - take a break
prior knowledge - useful
PS strategy - alternate
PS strategy - ask teacher
PS strategy - asked friends
PS strategy - comment out
PS strategy - compare output
PS strategy - delete methods with errors
PS strategy - determine cause of error
PS strategy - diagnosing problems
PS strategy - fix errors from bottom up
PS strategy - found shortcuts
PS strategy - invent
PS strategy - learn to code independently
PS strategy - not friends help
PS strategy - paper plan

```
PS strategy - plan out
PS strategy - pseudocode
PS strategy - recheck content
PS strategy - research
PS strategy - search Google
PS strategy - search YouTube
PS strategy - too shy to ask for help
PS Strategy - use other code
PS strategy - work in steps
PS strategy - worked out solution
pseudocode - easy
tracing - no tracetables
tracing - not using online
tracing - println
tracing - using NetBeans
tracing - using paper
tracing - verbally
```

Version 4

```
code development - achievable
code development - correct errors
code development - kept track
code development - lost track
code development - stuck
deadlines - adhere to
deadlines - motivational
deadlines - not able to adhere
difficulties - coding
difficulties - design
difficulties - documentation
difficulties - time consuming
documentation - achievable
documentation - UML
documentation and coding - code first
documentation and coding - simultaneous
emotional - build confidence
emotional - depressing
emotional - enjoyed PAT
emotional - frustrated
feedback - did not read rubric
feedback - learn
feedback - multiple submissions
feedback - multiple submissions helpful
feedback - provided
feedback - respond
PAT - able to sep back end
PAT - consider changing game
PAT - creativity
PAT - improvements
PAT - too ambitious
```

```
PAT - unsure of topic
PAT battleships - useful
PAT data structure - useful
PAT documentation - useful
perseverance - learnt to code project
perseverance - could have done more
perseverance - did not give up
perseverance - did not listen in class
perseverance - gave up
perseverance - procrastination
perseverance - take a break
prior knowledge - useful
PS strategy - alternate
PS strategy - ask teacher
PS strategy - asked friends
PS strategy - comment out
PS strategy - compare output
PS strategy - delete methods with errors
PS strategy - diagnosing problems
PS strategy - fix errors from bottom up
PS strategy - found shortcuts
PS strategy - invent
PS strategy - learn to code independently
PS strategy - not friends help
PS strategy - paper plan
PS strategy - plan out
PS strategy - pseudocode
PS strategy - research
PS strategy - search Google
PS strategy - search YouTube
PS strategy - too shy to ask for help
PS Strategy - use other code
PS strategy - work in steps
tracing - no tracetables
tracing - not using online
tracing - println
tracing - using NetBeans
tracing - using paper
tracing - verbally
```

Version 5

```
deadlines - adhere to
deadlines - motivational
deadlines - not able to adhere
difficulties - coding
difficulties - design
difficulties - documentation
difficulties - time consuming
difficulties - topic choice
documentation - achievable
```

documentation - UML
documentation and coding - code first
documentation and coding - simultaneous
emotional - build confidence
emotional - depressing
emotional - enjoyed PAT
emotional - frustrated
feedback - did not read rubric
feedback - learn
feedback - multiple submissions
feedback - multiple submissions helpful
feedback - provided
feedback - respond
PAT - able to sep back end
PAT - creativity
PAT - improvements
PAT - too ambitious
PAT battleships - useful
PAT data structure - useful
PAT documentation - useful
perserverance - learnt to code project
perseverance - could have done more
perseverance - did not give up
perseverance - did not listen in class
perseverance - gave up
perseverance - procrastination
perseverance - take a break
prior knowledge - useful
programming - correct errors
programming - achievable
programming - kept track
programming - lost track
PS strategy - alternate
PS strategy - ask teacher
PS strategy - asked friends
PS strategy - comment out
PS strategy - compare output
PS strategy - delete methods with errors
PS strategy - diagnosing problems
PS strategy - fix errors from bottom up
PS strategy - found shortcuts
PS strategy - invent
PS strategy - learn to code independently
PS strategy - not friends help
PS strategy - paper plan
PS strategy - plan out
PS strategy - pseudocode
PS strategy - research
PS strategy - search Google
PS strategy - search YouTube
PS strategy - too shy to ask for help

```
PS Strategy - use other code
PS strategy - work in steps
tracing - no tracetables
tracing - not using online
tracing - println
tracing - using Netbeans
tracing - using paper
tracing - verbally
```

Version 6

```
deadlines - adhere to
deadlines - motivational
deadlines - not able to adhere
difficulties - coding
difficulties - design
difficulties - documentation
difficulties - lost track
difficulties - time consuming
difficulties - topic choice
documentation - achievable
documentation - UML
documentation and coding - code first
documentation and coding - simultaneous
emotional - build confidence
emotional - depressing
emotional - enjoyed PAT
emotional - frustrated
feedback - did not read rubric
feedback - learn
feedback - multiple submissions
feedback - multiple submissions helpful
feedback - provided
feedback - respond
PAT - able to separate back-end
PAT - creativity
PAT - improvements
PAT - too ambitious
perseverance - learnt to code project
perseverance - could have done more
perseverance - did not give up
perseverance - did not listen in class
perseverance - gave up
perseverance - procrastination
perseverance - take a break
prior knowledge - useful
programming  - correct errors
programming - achievable
programming - kept track
PS strategy - alternate
PS strategy - ask teacher
```

PS strategy - asked friends
PS strategy - comment out
PS strategy - compare output
PS strategy - delete methods with errors
PS strategy - diagnosing problems
PS strategy - fix errors from bottom up
PS strategy - found shortcuts
PS strategy - invent
PS strategy - learn to code independently
PS strategy - not friends help
PS strategy - paper plan
PS strategy - plan out
PS strategy - pseudocode
PS strategy - research
PS strategy - search Google
PS strategy - search YouTube
PS strategy - too shy to ask for help
PS Strategy - use other code
PS strategy - work in steps
Scaffolding Battleships useful
Scaffolding data structure useful
Scaffolding documentation useful
tracing - no tracetables
tracing - not using online
tracing - println
tracing - using NetBeans
tracing - using paper
tracing - verbally

APPENDIX #7 - Ethics Clearance Certificate

Research Office

HUMAN RESEARCH ETHICS COMMITTEE (NON-MEDICAL)
R14/49 Kench

CLEARANCE CERTIFICATE	**PROTOCOL NUMBER: H15/06/31**
PROJECT TITLE	Evaluating the development and effectiveness of grit and growth mindset among High School students in a computer programming project
INVESTIGATOR(S)	Mrs D Kench
SCHOOL/DEPARTMENT	Computer Science/
DATE CONSIDERED	08 July 2015
DECISION OF THE COMMITTEE	Approved unconditionally This protocol has been approved by the HREC (Medical) Sub-Committee
EXPIRY DATE	12 July 2018

DATE 13 July 2015 **CHAIRPERSON** J. Wright
(Professor J Knight)

cc: Supervisor : Professor S Hazelhurst

DECLARATION OF INVESTIGATOR(S)

To be completed in duplicate and **ONE COPY** returned to the Secretary at Room 10005, 10th Floor, Senate House, University.

I/We fully understand the conditions under which I am/we are authorized to carry out the abovementioned research and I/we guarantee to ensure compliance with these conditions. Should any departure to be contemplated from the research procedure as approved I/we undertake to resubmit the protocol to the Committee. **I agree to completion of a yearly progress report.**

_____ _____/_____/_____
Signature **Date**

PLEASE QUOTE THE PROTOCOL NUMBER ON ALL ENQUIRIES

APPENDIX #8 - IEB Rubric

PERFORMANCE ASSESSMENT TASK – EXAMPLE MARK SHEET

Project Specifications				
CONTENT				
Summary			**2 MARKS**	
0 – No summary or completely inadequate	1 – Summary partially done	2 – Summary encompasses all aspects of the problem		
Specifications of Program Function			**3 MARKS**	
0 – No Functions listed	1 – Function list is a single line statement/a list of 4 or less points	2 – Function list is a substantial list of appropriate outcomes	3 – Function List is complete and detailed	
Specifications of User Interface			**2 MARKS**	
0 – User Interface not specified or incorrectly specified	1 – One or two items are inadequately specified	2 – User Interface completely specified		
Specification of Help			**2 MARKS**	
0 – Help not discussed	1 – Help partially discussed with omissions and/or errors	2 – Help completely detailed including menu options and types of help available. Context provided for each help screen as well as storage of help related data has been specified		
Specifications of Data Storage			**3 MARKS**	
0 – No information given on the data to be stored	1 – Only a few items are incorrectly described	2 – Many items are described with a few errors	3 – All data to be stored has been correctly described	
Hardware and Software requirements			**2 MARKS**	
0 – Hardware and Software not discussed	1 – Hardware and software is partially discussed for development, includes detail for what software is needed for what task	2 – Hardware and software is completely discussed for development, software list includes versions		

System Design Document

Taking the template as an example, this document should be around 14 – 20 pages (including title page and table of contents). Its main task is to detail the actual design elements of the program, namely:

- User interface design (what the screens look like and what happens on them)
- Program flow (how the program works – linked to the interface)
- Class design (what the classes are, their fields and methods)
- Database/Storage design (what the persistent storage structure is)

System Design Document				
User Interface Design NB: The GUI screen can be designed in a RAD environment (e.g. NetBeans/Eclipse/Delphi), on paper, or in a graphics program like Paint. Screen mock-ups are possible without writing code, therefore screenshots are acceptable as evidence of design, All data to be displayed on the screen must be listed. The action elements on the screen must be listed and clearly described.			**8 MARKS**	
0 – 2 – No screen design evident/Screen design is cursory	3 – 5 – Screen design is evident but no consideration has been given to good design principles for an effective GUI or inadequate description of on-screen action elements or no indication of progression between screens	6 – 8 – Screen design present, layout good, all on-screen action elements tabulated and described in detail. Data access and error-checking are indicated for all action elements.		
Sequencing – (also known as What Happens When) **In this section you describe the flow of events in the program – planning this can make your program easier/more logical to use (can help you decide on interface elements such as wizards, etc.)** **NB:** The template contains flowcharts. The candidate may use any algorithmic representation of sequencing of events in the flow of the program such as pseudocode.			**5 MARKS**	
0/1 – No sequencing evident/Sequencing is rudimentary or cursory with little detail and large leaps of logic evident	2/3 – Sequencing is substantial but still shows leaps of logic/areas that have not been covered in appropriate detail	4-5 Sequencing is broken into sections to cover all aspects of the functions and features in the Specification document. Flow is clear, well represented and easy to understand. No logic gaps are evident.		
Class Design **The candidates must provide their class design and explain their choice of classes, fields and methods.** **NB:** The template contains tables and this structure must be used for the class design where each field and method is explained.			**8 MARKS**	
0 – 2 – No class design evident/class design is rudimentary or cursory with little detail. Fields are incomplete, methods are minimal/not well thought out/not well described.	3 – 5 – Class design is substantial but still shows obvious gaps in missing fields/methods. Method descriptions more thorough but elements still missing.	6 – 8 – Class design is thorough – all fields and methods are present and well described. Sub-methods are present. Methods and fields clearly relate back to the Specification document.		

Persistent Storage Design **The candidate must provide their storage design** **NB:** Storage design should be done in tables. Screenshots of tables with record structure and field types from database software can be acceptable.			**6 MARKS**	
0/1 – No storage design evident/storage design is rudimentary or cursory (e.g. 'uses a database').	2 – 4 – Storage design is substantial. Some record design and description of fields are evident. Descriptions are, however, cursory and show evidence that they have not been completely thought through. Not all files/tables/relationships covered.	5/6 – Record structure is described – fields are listed, typed and described. Data structure for text/typed files is described. Storage design is appropriate to purpose and matches the Specification document requirements.		
Explanation of Storage Design **The candidate must provide an explanation of their storage design** For example a text file may have been a better solution that a database as the data to be stored is small in value and simple. In this section you describe the way that data is stored so it can be re-accessed when the program is used again. Appropriate storage is what is required. DO NOT mark for quantity. What we need to see is that if, for example, a game is coded then high scores and save games are needed – and maybe other file handling to load appropriate data. DOES NOT have to be database.			**4 MARKS**	
0/1 – No explanation is given about storage or no understanding of the storage design the storage is shown.	2/3 – Explanation is substantial but it is not completely justified and there are some areas of confusion or lack of understanding of the implication of the storage design.	4 – Explanation shows in-depth understanding of the implications of the storage design and is completely justified.		

CODING and Technical Documentation

Taking the template as an example, this document can be anything from 10 – 100+ pages depending on the complexity and extent of the code that the candidate has written.

Emphasis must be placed upon:

CODING NB: This is assessed by examining the actual code – no attention need be paid to documentation/layout/etc.				
Separation of UI from working code			**5 MARKS**	
0 – No separation – all code in the interface class/unit.	1 – 3 – Some separation. There are separate classes/units but work is still done in the UI. Insufficient further breakdown and separation of different aspects of the engine. This includes SQL statements for database centric programs (SQL is separation – you are passing off complex data handling to the database engine).	4/5 – Complete separation. Different classes are separated as well as the engine from the UI. The engine can be 'plugged into' a different UI that uses all the methods appropriately and will work without any issues.		
Inter-Code communication (Typed Methods and Parameters)			**5 MARKS**	
0 – No inter code communication (no typed methods (functions) or parameters.	1 – 3 – Some use of parameters/functions. Marks can be deducted (–1 per error type – multiple instances of the same error do not accumulate deductions) Errors include: Shows errors in comprehension of the concepts– unnecessary use of parameters, incorrect parameters types, parameters specified but not used, incorrect functions types, failing to return values in functions, failing to use the results returned by functions, using variables/fields where the value is best returned by a function.	4-5 Effective and conceptually correct use of parameters and typed methods (functions).		
Good General Techniques			**5 MARKS**	
0 – No techniques.	1 – 4 – Errors in techniques (–1 per error type – multiple instances of the same error do not accumulate deductions). Errors include: No indentation, single level indentation, inconsistent or inaccurate indentation, variable names do not clearly indicate what the variable is used for, multiple variables used instead of arrays, multiple if statements instead of switches, repetition of code (instead of using a procedure/function), code extending beyond the edge of the readable printed page.	5 – Technically perfect. Indentation immaculate. Variable names, data structures, etc. all correct.		

Fulfilment of Specifications **NB:** this can only be assessed by running the compiled program.				**6 MARKS**	
0 – Not achieved.	1 – 3 – Basic implementation of specifications. Obvious omissions in missing functions/significant amount of functions do not work as specified.	4 – 90% of specification achieved. Perhaps all functions are there but do not all work correctly OR almost all functions are there but those that are there work 100%.	5/6 – All specifications complete and working 100%		
User Experience **NB:** this can only be assessed by running the compiled program.				**4 MARKS**	
0 – Program does not execute.	1 – The user is lost – does not know where to start or how to achieve anything when using the program.	2/3 – Most of the program has a good user experience but navigating to some screens/functions is unnecessarily complex/impossible. Any aspect of the design/interaction is confusing or unsatisfying.	4 – An easy to use, completely easy to understand and to navigate program: a wonderful user experience.		

	70 MARKS	

APPENDIX #9 – Personal Journal

1 Thursday 10 September - Double period

Students were instructed to do the following:

- Open a NetBeans project and package
- Create a database in NetBeans using MySQL
- Students had to create the User table with name, password and score field
- Help table in MySQL with fields for topic and description.
- Students had to create a log in GUI with fields for name, password and button to log in and create a new user. If they clicked create new user then the confirm password field should appear.
- HelpGUI with buttons down the side for about 5 help topics

2 Friday 11 September Single period

Students were given time in class to work on the tasks given on Thursday. Gave an overview of the documentation using samples of PATs from Grade 12 students of previous years. Gave the student's access to these Pats on the shared drive.

3 Monday 14 Sept – Double period

Showed students how to access help table from the HelpGUI using the query method in the DB class. Showed how to use the one method for different buttons.

Asked students to write down an algorithm for logging in and creating a new user. They will need to use the update method to insert a new user and the query method to check if the user exists.

Most students has filled in questionnaires and had provided a pseudonym

4 Tuesday 15 September – Double period

Allowed students to work on tasks given in previous period

5 Wed 16 September – Single period

Showed the 2D grid class, the constructor and toString. Allowed them to work on GUI classes.

I calculated their mindset and grit scores.

6 Thursday 17th - Single period

Allowed them to continue working on creating a 2D grid with constructor and toString. All students must do a grid regardless of their project. I demonstrating Battleships and they need to get a working version of Battleships. This can be altered for their project or discarded depending on their project.

7- Friday 18 September

Check on progress. They should have the HelpGUI, Log in GUI that checks or creates a user, User class, DB class, Grid class with constructor and to String.

Students had to hand in the Specification document.

8 Monday 21 September

Checked that each student had the following working in their project.

- User table
- Help Table
- LoginGUI C
- Check existing User
- Create new user
- HelpGUI
- Grid
- GameGUI
- DB
- Check if specifications have been handed in

9 Tuesday 22 September

Returned their marked specifications. Commented on their Specifications documents and explained in general in class where there were common errors. Allowed students to fix their specifications documents.

Students had to extend their project to include the working GameGUI that links to the Grid. Each time an element is clicked on the GUI, a method in the Grids must be called, passing parameters and returning a vale. The component (usually a button) on the gird needs to react.

10 Wednesday 23 September

Time given in class to work on the GUIs calling method in the Grid class.

11 Thursday 24 September

Double period. Time given to work on their Game GUI

12 Friday 25 September

Time given in Class to work on their Game GUI

13 Monday 28 September

Double period. Checked each students Game GUI and tested if it responded to user input by clicking the button and calling the corresponding grid methods. Checked if they has completed any other part of the missing projects.

Students handed in their second version of their Specifications.

Recorded the dates when they submitted their Specs to track how many times they submit. If they had full marks I took the document and mark sheet in.

14 Tuesday 29 September

Described the Design Document and discussed the two templates of students work from previous years. Since the students had demonstrated they can use a grid, help high scores and the DB class. They are allowed to continue with their own project.

15 Wednesday 30 September

Students given time in class to work on their own projects. Some students discussed their class design.

16 Thursday 1 October

Students given time in class to work on their own projects. Some students discussed their class design.

17 Friday 2 October

Students given time in class to work on their own projects. Some students discussed their class design.

18 Monday 5 October

Return to theory lessons to complete syllabus. Double period.

Students need to continue with their project at home. Complete Learning Unit 4 of Theory textbook.

Students submitted design documents.

Tuesday 6 to Thursday 8 October

Started Learning Unit 5 of the theory textbook. Students were given the week to summarise the notes of Learning Unit 5 and Learning Unit 6 the textbook in groups and produce a definition list of the terms discovered in the theory book.

Design documents were returned on Wednesday with feedback.

Friday 9 October

Discussed the summaries of the Learning Units with the students and showed them how to relate to other section of the syllabus.

Monday 12 October to Tuesday 13 October

Learning Unit 7. Went through the concepts in the textbook and updated the definitions list. Completed the Theory syllabus with the conclusion of this Learning Unit.

23 Wednesday 14 October to Friday 30 October

Gave students time to complete their project in class. Documents were marked in class with verbal feedback. The number of submissions were recorded for each phase.

Students were given class time to complete the questionnaires for the second time on the 19th October.

9 November

Student given class time to complete their questionnaires for the third time.