Primary Mathletes – Primary School Mathematics Teaching Aid Application

Final Project Report

DT282
BSc in Computer Science International

Paul Davis

C16324311

Cindy Liu

School of Computer Science
Technological University, Dublin

11th April 2020

Abstract

This project will be focusing on improving primary school teaching through the use of mobile apps. The project will consist of an app that is going to be developed with Kivy, a Python library used to create GUI elements and using Python programming, that will help primary school teachers by acting as a teaching aid for their students to improve their mathematical abilities.

One issue that commonly occurs in classrooms, especially ones with a lot of students, is that no matter how slow the teacher goes, eventually they will have to move on so that they can appropriately tackle all elements of the curriculum before the end of the year. This project aims to aid the teacher and student by giving them the ability to teach the student more of the curriculum in a fun and interactive way, inside and outside of the classroom so that even if they fall behind there is a way of catching up without taking up valuable class time.

The app will hopefully improve the grades of students who may be struggling with maths in school and prevent them from giving up if they fall behind.

During the implementation of this project, research into the primary school curriculum through online resources was conducted as well as conversing with primary school teachers to get an idea of areas where students are struggling with the topics that come up and see if it is possible to make it more accessible to a wider range of students.

Implementation of this project was completed through Python programming using Kivy to make an application that uses use the research conducted through online resources to create a fun, interactive and informative application that will aid both students and teachers in primary schools.

The project was going to be evaluated by getting in contact with a primary school teacher and asking them to use the app during their maths teaching/learning in-class and monitoring any improvements with the use of the app. There were also other plans to get a small group of primary school students to participate in a study group to try and evaluate the apps usefulness but due to the sudden outbreak of Covid-19, by the time the application was ready to be tested, neither of these evaluations were possible due to the country being on lock down. Other testing methods were conducted instead, and these are discussed further within the document.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

_____

Paul Davis

11 April 2020

Acknowledgements

I would like to thank all of my lecturers over the last 4 years at TUD for helping me get to this point in the course and my friends and family for supporting me throughout the duration of this project.

Table of Contents

1. Introduction 10
1.1. Project Background 10
1.2. Project Description 11
1.3. Project Aims and Objectives 12
1.4. Project Scope 12
1.5. Thesis Roadmap 13
2. Literature Review 15
2.1. Introduction 15
2.2. Alternative Existing Solutions to The Problem 15
2.3. Technologies Researched 18
2.4. Technologies Used 19
2.5. Other Research Conducted 22
2.6. Existing Final Year Projects 22
2.7. Conclusions 23
3. Design 24
3.1 Introduction 24
3.2. Software Methodology 24
3.3. Overview of System 28
3.4. Software Test plan 29
3.5. Front-End 30
3.6. Middle-Tier 47
3.7. Back-End 47
3.8. Conclusions 48
4. Experiment Development 49
4.1. Introduction 49
4.2. Front-End 49
4.3. Middle-Tier 59
4.4. Back-End 73
4.5. Conclusions 86
5. Evaluation 87
5.1. Introduction 87
5.2. Software Evaluation 87
5.3. Questionnaires and Interviews Evaluation 97

5.5 Full Use of Application....99
5.6 Conclusions....102
6. Conclusions and Future Work....103
6.1. Introduction....103
6.2. If I Could Do It Again....103
6.3. Conclusions....103
6.4. Future Work....104
7. Bibliography....107
8. Appendices....110

Table of Figures

Figure 1 Comparison of easy and difficult topics on Duolingo....11
Figure 2 AB Maths minigame....15
Figure 3 Minigame paper prototype....15
Figure 4 Minigame wireframe....15
Figure 5 Mathematics skill builders primary homepage....16
Figure 6 Skill builder primary basic question example....16
Figure 7 ixl example question....17
Figure 8 IXL Home Page....17
Figure 9 Sprints lifecycle as part of the Scrum Methodology [15]....24
Figure 10 Scrum lifecycle as part of the Agile Methodology [17]....25
Figure 11 Project Trello board....26
Figure 12 Card details for Interim Report....26
Figure 13 Airtable Bug Table....27
Figure 14 GANTT Chart....28
Figure 15 Three-Tier System....29
Figure 16 Technical Architecture Diagram....29
Figure 17 Register Paper Prototype....31
Figure 18 Register Wireframe Prototype 1....31
Figure 19 Register Wireframe Prototypes 2....31
Figure 20 Final App Register Page....31
Figure 21 Login Paper Prototype....32
Figure 22 Login Wireframe Prototype....32
Figure 23 Final App Login Page....33
Figure 24 Main Menu Paper Prototype....34
Figure 25 Main Menu Wireframe Prototype....34
Figure 26 Final App Main Menu....35
Figure 27 Duolingo Level Tree....35
Figure 28 Minigame Paper Prototype....36
Figure 29 Minigame Wireframe Prototype....36

Figure 30 Final App Minigame Page....36
Figure 31 Results Paper Prototype....37
Figure 32 Results Wireframe Prototype....37
Figure 33 Final App Results Page....37
Figure 34 Progress Paper Prototype....38
Figure 35 Progress Wireframe Prototype....38
Figure 36 Old version of Progress Page....39
Figure 37 Final App Progress Page....39
Figure 38 Title Page....40
Figure 39 Classroom Page....40
Figure 40 Play Page....40
Figure 41 Level Selection Page....40
Figure 42 Teacher's Student List....40
Figure 43 Edit Profile Page....40
Figure 44 Login Page using Kivy Default Font....40
Figure 45 Default Kivy Button....41
Figure 46 Custom Rounded Button....41
Figure 47 'X' Button....41
Figure 48 Unused Back Button....41
Figure 49 Unused Log Out Button....41
Figure 50 Final App Button Example....41
Figure 51 Screen Flow of App....42
Figure 52 Unsuitable Password Popup....42
Figure 53 Log Out Popup....42
Figure 54 Correct Answer Popup....43
Figure 55 Incorrect Answer Popup....43
Figure 56 First iteration use case for both students and teachers....44
Figure 57 Second iteration use case for students....44
Figure 58 Third iteration use case for students....44
Figure 59 Final Second Iteration for User and Teacher....45
Figure 60 Third Iteration User Case for Students....46
Figure 61 Third Iteration Use Case for Teachers....46
Figure 62 ERD showing relationships between tables....48
Figure 63 Final Version of ERD....48
Figure 64 Title Page of the app....50
Figure 65 Register Page of the app....51
Figure 66 Login Page of the app....52
Figure 67 Screen Manager Code Snippet in Kivy File....53
Figure 68 Basic Screen Structure in Kivy File....53
Figure 69 Image Code Snippet in Kivy File....54
Figure 70 Logo Adapting 1....55
Figure 71 Logo Adapting 2....55
Figure 72 Button Code Snippet in Kivy File....56
Figure 73 Locked, Normal and Disabled Button Example....56
Figure 74 Label Code Snippet in Kivy File....57

Figure 75 Dynamic Update of Label Text....57
Figure 76 Canvas Code Snippet in Kivy File....58
Figure 77 Canvas Line Styling 1....58
Figure 78 Canvas Line Styling 2....58
Figure 79 Snippet of code showing Kivy libraries....59
Figure 80 Snippet of code showing firebase library being used....59
Figure 81 Firebase Post Example....60
Figure 82 Example of random strings created by Firebase POST method....60
Figure 83 Firebase GET Example....60
Figure 84 Firebase GET Example with Second Parameter....61
Figure 85 Firebase PUT Example....61
Figure 86 PUT Method Creating and Updating Information....61
Figure 87 DELETE Method Example....61
Figure 88 Label Base Code Snippet....62
Figure 89 Sound Loader Code Snippet....62
Figure 90 Random Library Code Snippet....63
Figure 91 Random Use in Minigame....63
Figure 92 Send Mail Function Code Snippet....64
Figure 93 Email received from the Send Email Function....64
Figure 94 Datetime Code Snippet....65
Figure 95 Clicking On An Operator....65
Figure 96 Clicking On A Level....65
Figure 97 Algorithm File Difficulty Setup....66
Figure 98 Setting Maximum and Minimum Ranges....66
Figure 99 Ensure Answer Doesn't Exceed 1000....67
Figure 100 Examples of 'blank' Being 0, 1 and 2....67
Figure 101 Minigame Initialization....68
Figure 102 Reinitialize Function....68
Figure 103 Update Function Code Snippet....69
Figure 104 Check Answer Function....70
Figure 105 Get Potential Answers Code Snippet....70
Figure 106 Difficulty Mapped Calculation....71
Figure 107 Difficulty Mapped Numbers 1-10....71
Figure 108 Level 8 Addition Potential Answer Example....71
Figure 109 Level 2 Addition Potential Answer Example....72
Figure 110 Potential Answer Code Snippet....73
Figure 111 Old Database Prototype....74
Figure 112 Final Version of the Database....75
Figure 113 Password Check on Register Page....76
Figure 114 Email Check on Register Page....76
Figure 115 Register Loop for Register Page....77
Figure 116 Login Screen On Pre Enter Function....78
Figure 117 Login Loop Function on Login Screen....78
Figure 118 Check Login Function on Login Page....79
Figure 119 Update JSON Loop Function....79

Figure 120 Check Week Function 80
Figure 121 JSON Store Code Snippet 81
Figure 122 Teacher Progress Page and On Pre-Enter Function 82
Figure 123 Create Scroll View Function 83
Figure 124 Breakdown of Week Numbers from Database 84
Figure 125 Creating the Week List for Students Progress 84
Figure 126 No Previous Week Example 85
Figure 127 Previous Week Example 85
Figure 128 Numerical Change for Best Score 85
Figure 129 Percentage Change for Correct Answers 85
Figure 130 Percentage Change for Time Played 85
Figure 131 Percentage Change for Total Games Played 86
Figure 132 Percentage Change for Total XP Earned 86
Figure 133 Check For Zeros Function 86
Figure 134 Increment Timer Function 88
Figure 135 Timer Example 1 88
Figure 136 Timer Example 2 88
Figure 137 Timer Example 3 88
Figure 138 Slow Code Snippet 89
Figure 139 Average Speed of Old and New Code 89
Figure 140 Faster Code Snippet 89
Figure 141 Slow Code Snippet 90
Figure 142 Average Speed of Old and New Code 90
Figure 143 Faster Code Snippet 91
Figure 144 Test Case Table 94
Figure 145 Testing Table 95
Figure 146 Bar Graph Showing Peoples Font Preference 97
Figure 147 Bar Graph Showing People Under 12s Font Preference 97
Figure 148 Bar Graph Showing People Over 12s Font Preference 98
Figure 149 Improvement Through Use of App 102
Figure 150 Future Minigame Prototype 1 105
Figure 151 Future Minigame Prototype 2 105

1. Introduction

1.1. Project Background

Children, even from young ages, can learn a lot by using technology.[1] It can help reinforce information already learned in school curriculums through the gamification of learning in apps on phones or tablets.[2] Below are articles and documents about these sources and also includes some research conducted to find what the best programming language would be to write the Primary Mathletes application using:

One paper discusses the different programming environments that are available for mobile platforms such as Python, C#, and Java through Android Studio. The paper mentions the flexibility of Python for mobile platforms. [3] Originally the Primary Mathletes application was going to be developed using Android Studio which, for a lot of cases is not very user-friendly in comparison to something like Python which lets you do a lot with less code and so this is what was chosen to develop the application of this project.

Another paper discusses how the use of mobile devices such as iPads can be used by teachers to make the teaching of maths more efficient and more involved by having data stores with a piece of software on a mobile device.[4] It found that pre-service and in-service teachers saw value in integrating iPads into Maths education as a tool to promote student learning and it also discusses several mobile learning approaches and how they can affect a student's learning positively or negatively.

Another piece speaks about how in South Africa there is limited access to PCs in many homes but yet there are three million teenagers that have Java enabled cell phones who would strongly benefit from mobile learning or m-learning applications due to the inaccessibility of web apps in that region of the world.[5]

The final paper looked at discusses the differences between e-learning (using a PC to teach students) and m-learning (using mobile devices) such as handheld phones and tablets that are internet-enabled and how m-learning is a lot more accessible because there is no hardware limitation of PCs because almost everyone nowadays already owns a mobile phone.[6] This cuts the cost of textbooks and on building expensive computer rooms for students.

1.2. Project Description

Primary Mathletes is a cross-platform mobile application aimed at primary school students and their teachers to help reinforce their learning of the primary school mathematics curriculum. It provides fun, interactive minigames based on the third-class primary school mathematics curriculum focusing on simple operators: addition, subtraction, multiplication and division, all following the learning outcomes put in place by the Irish Department of Education.[7]

The application is split up into two different modes: a "student mode" and a "teacher mode". In the "student mode", students will be able to register, login, play minigames, join teacher-created virtual classrooms and view weekly progress reports showing improvements over previous weeks.
The "teacher mode" will allow the same functionality as the "student mode" but they also have the ability to create a virtual classroom that students can join and the teacher is able to view all of their students' weekly progress in one place. Teachers can create a classroom using a name and as long as students have this name, they can join the virtual classroom. The teacher can give this name to the students however they wish but the application does not let teachers invite students, only students can join classrooms. Both modes allow for progress reports to be sent to a user-selected email account so, if parents are concerned with their children's progress within the application they can be notified at the click of a button.

Primary Mathletes will follow a similar structure to Duolingo (figure 1). Duolingo is an application some primary students may already be familiar with. Students will earn experience points from completing minigames and through these experience points, unlock increasingly more difficult levels. The students will be unable to play more difficult levels until they have first completed all levels leading up to the one that they want to attempt. This is to ensure that students don't accidentally throw themselves into the deep end of learning. This gamification of learning will hopefully keep students engaged with the application and therefore will also increase the student's retention of information for this operator.

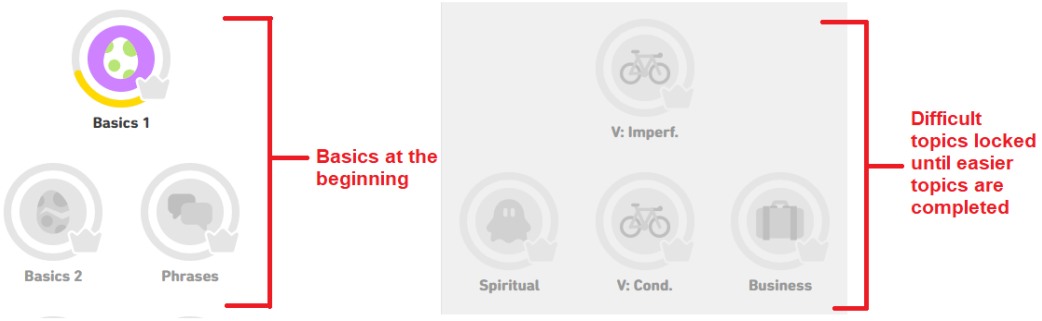


Figure 1 Comparison of easy and difficult topics on Duolingo

The application was originally going to be tested in a few different ways: through black-box testing which involved handing over the application to primary school students in a small group and record their thoughts and suggestions before redeveloping the application before the end of the project deadline but this was not possible due to the recent, sudden outbreak of COVID-19. While some black-box testing was still possible under the circumstances, the original plan was not executed and so the redevelopment segment of this project fell short of what was planned. The other form of testing, white-box testing, was conducted throughout development by myself to ensure that all individual parts of the application worked separately before being integrated into the full app. Testing will be discussed further in section 5.

1.3. Project Aims and Objectives

The overall aim of the project is to help aid primary school students and teachers to improve the students' understanding of the maths curriculum.

The goal for the project is to create a mobile application that will be used by primary school students that includes minigames and helpful maths-related topics that help to improve their overall understanding of the primary school maths curriculum.

Milestones are outlined in the GANTT chart that will be seen later in the report (Section 3). This was created to ensure that targets were set and completed (or at least attempted to be completed) within their given timeframe so that the project would be completed on or before the due date.

The purpose of the project is to see if it is possible to create an application that could be used by primary schools and in turn, would improve the understanding of all students that are studying third-class mathematics. The project would be considered a success if the application could be used by these students and when asked of their opinion, had it improved their maths abilities by some amount.

1.4. Project Scope

Maths can be one of the most hit or miss subjects that are tackled in any level of schooling whether it be primary, secondary or third-level education. Students tend to either know the answers or they don't. So, because of this, there is a market for something that can come away from traditional learning on paper and through verbal teaching and gamify the process of learning to the point where it doesn't feel like learning anymore. With something like maths where there are definitive answers to everything, some students struggle with finding the right process to arrive at these answers. Some students don't enjoy traditional methods of learning and they may lose interest in class which will lead to them falling behind. Even though they could be a hyper-intelligent person, the method in which they learn can throw them off reaching their true level of knowledge.

The application could be expanded to include a huge range of information but for the scope of this project, the functionality and educational content will be limited to a small subsection of the primary school mathematics curriculum. The app will be aimed towards third class students and focusing on basic operators: addition, subtraction, multiplication, and division. The reason for choosing this section of the third class curriculum is because it is the first time students in primary schools begin using numbers and operators whereas before this in first and second class the curriculum focused more on basic counting and recognition of shapes and other things not related to arithmetic so it was decided this would be a good starting point and in future, the application could be further expanded to include educational treatment for all of the primary school curriculum.

1.5. Thesis Roadmap

Research

This section explores background research conducted on apps similar to the one being developed in this project and also research conducted on papers that outline positive and negative aspects of m-learning (mobile learning) and the benefits of using certain programming languages over others when developing an interactive application for younger students.

Design

This section includes use-cases and personas related to the application, a detailed description of the architecture used and testing plans for the application.

Development

This section delves deeper into the development process of the system that was outlined in the design chapter and will also include challenges that became apparent throughout the development of the application.

Testing and Evaluation

This section describes all of the testing and evaluation of the system that was executed. Each part of the testing is described in detail and will have a detailed report of user feedback received during the user's evaluation of the application.

Redevelopment

This section will outline some of the development steps taken as a result of the feedback gained from the user evaluation. The changes made and the importance of these changes will be examined.

Conclusions and Future Work

This section will reflect on the entirety of the project and will discuss the conclusions drawn, personal reflections made, and the future work planned for the project.

2. Literature Review

2.1. Introduction

In this section, applications and websites that are similar to the application being developed in this project will be explored. Discussion on what these applications have done right that could potentially be implemented into the Primary Mathletes app and things that they have done wrong that can be improved upon through this project. Additional research on exactly what students and teachers want and need from an app like Primary Mathletes will be conducted through questionnaires and verbal communication. Information about the technologies researched is given as well as why some technologies were chosen over others. There is also a review of two previous final year projects.

2.2. Alternative Existing Solutions to The Problem

AB maths for Apple iOS:

This app wasn't developed specifically for Irish primary school maths students, but it seems to have a lot of general maths content behind paywalls for around 90% of the content. The available free content shows a similar structure to one of my original ideas for minigames that can be seen in the image shown below along-side a paper prototype and wireframe created for this specific minigame.

Some things that this app does right include good minigames with bright colours that could really draw the attention of younger users. The structure of the equations was eventually adopted by the Primary Mathletes app and so was the idea of using sound as a way to engage users further. However, this app falls short due to the fact that there is a paywall that locks almost all content from the users which gives off the impression that only people who are willing to pay should be given the opportunity to learn. A better strategy for this would to sell advertising space somewhere on the app instead of impeding students learning.

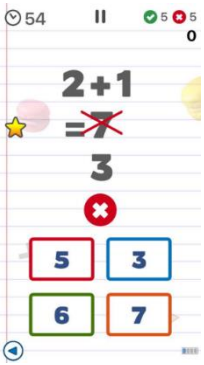


Figure 2 AB Maths minigame

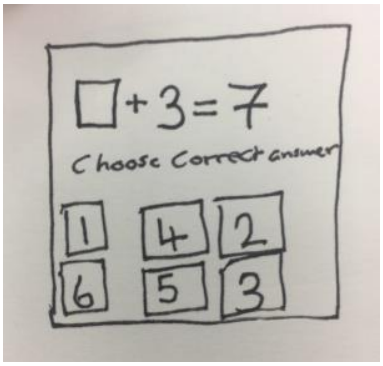


Figure 3 Minigame paper prototype

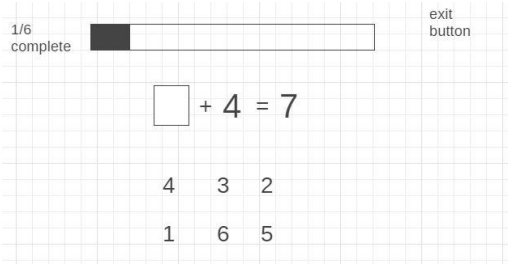


Figure 4 Minigame wireframe

Mathematics Skill Builder Primary for iOS:

Opposite to AB Maths mentioned above, this app was developed for primary school maths students, but it is very outdated and cartoonish. It also seems to deal with topics that could be deemed too easy for the target audience of the Primary Mathletes app. Examples are challenges where a student needs to pick which number is bigger than the other, pick what shape is mentioned, etc. Examples of this can be seen in the images below.

While this application is useful because it does comply to the learning outcomes set out for primary school students issues arise just by looking at the design of the app. This app has a trademark for the year 1997 and if that is when the app was designed it really shows. A cleanser and more readable solution like the one found in the Primary Mathletes app is the way forward especially nowadays where a simple and clean design go a long way. As well as the poor design, the content shown in the application is too basic for the types of students this project is targeting.

Figure 5 Mathematics skill builders primary homepage

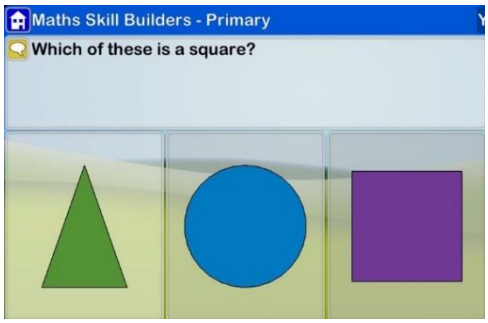


Figure 6 Skill builder primary basic question example

ixl.ie website:

This website has hundreds of different quizzes that are developed around the different levels of curriculum for Irish primary school students. Figure 7 shows a nice, clean minigame found on the website. Figure 8, however, shows the home screen of ixl and this can seem like a bit of a sensory overload with all of the text onscreen which may be off-putting to students visiting the website for the first time. The application created for the Primary Mathletes project has a clean and simple design with few options so that younger students don't get confused when interacting with it, especially for the first time.

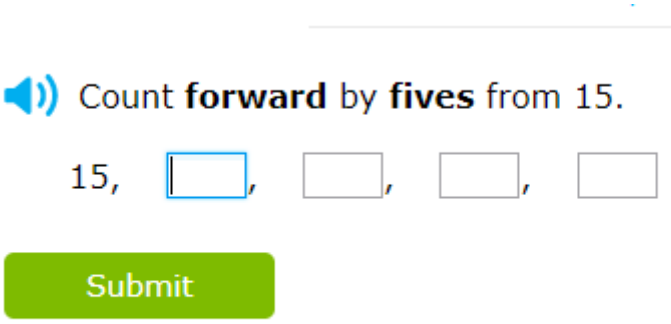


Figure 7 ixl example question

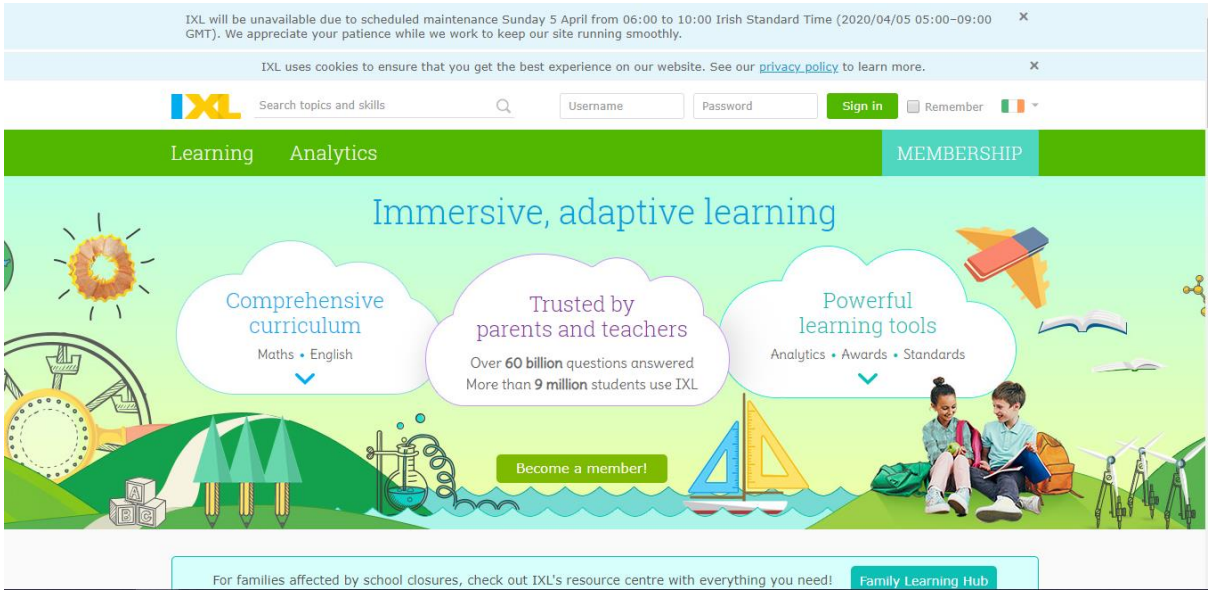


Figure 8 IXL Home Page

The ixl website is a good place for students to learn about the maths curriculum for primary schools. Looking through the website gave me the inspiration for the minigame section of the app that is seen in the final version of this project. The only issue that could be expanded upon with this website is the fact that you need a computer to access it. A mobile version of this website would be ideal as it has all of the learning information and minigames needed to be a great tool for students but its lack of mobility is its downfall.

All of the apps and websites mentioned above are definitely in the same vein as the Primary Mathletes application created for this project. However, whether it's the fact that these apps are too basic, outdated or too cluttered, there seems to be room for improvement in many aspects such as content and user experience through sound, imagery, simplicity, and interactability.

2.3. Technologies Researched

Quite a few programming languages and tools were considered in addition to all the programs and languages that were studied throughout the last three years of computer science.

Firstly, some research went into web-based applications using HTML, CSS, PHP, and JavaScript but in the end, this was ruled out because after some conversations with primary school students and teachers it became clear that there is rarely enough working desktop computers for primary school students. Now, more than ever, there is access to tablets and iPads in primary schools as just this year an additional €50 million was granted to primary and secondary schools to provide extra technologies for E-learning. [8]

Some research also went into using Java and the Android Studios IDE because this is something we briefly learned in 3rd year of computer science and creating an application using something already studied would be easier than learning a new programming language and IDE but, after completing this module with a low grade it was evident that Android Studios would not be the ideal choice for this project. It is difficult to use and is not very forgiving when something goes wrong. Hard to understand error messages also makes the IDE difficult to work with.
In the 4th year of the computer science course, many modules use Python. Researching Python led to the discovery of a plethora of resources for the language and this is where the Kivy library for Python was discovered. The Kivy library can be used to create GUIs based on Python code which is great because Python is very easy to use and there is a lot of great resources online for Kivy as well. Kivy also allows for cross-platform development which is perfect for this project because not every primary school student has access to predominantly iOS or predominantly Android phones or tablets.[9]

For databases, some research went into using either a local SQL database that would store the user's data locally but this led to problems such as if a student lost his or her phone then all of their progress would also be lost. There was also an issue of the teacher not being able to access the child's progress reports without going onto the student's device manually.

In the end, Google's Firebase Database System was chosen because this means that any user can access their account just by logging in on any random device. This means that any student can sign in on any device without being tied to a single device. This is especially useful for situations in primary schools where many users may be sharing the same tablet or iPad in a single day. Firebase was the easy choice since it is free, easy to use, there's almost no setup especially if you already have a Google account and there is no SQL involved so it is quicker than setting up your own database.[10]

Some research also went into what it would take to compile this projects application into a useable APK or API file so that it could be used on mobile devices. The only option currently available to compile Kivy projects is Buildozer which proved to be very difficult to use. Further details on this will be discussed later on in the project document.

2.4. Technologies Used

Many pieces of technology were used in combination to create this entire project and these will be discussed briefly below.

Python

The Python programming language was chosen because of its easy usability and multitude of resources and tutorials found online. Python is now the second-largest programming language in use after JavaScript and since I am comfortable in JS already it made sense to upskill my Python abilities while also using the best language available for this type of application.[11]

Google's Firebase Database:

The Firebase system was chosen because of the ability to connect to the Python code using its own Python library and because it allows users to access their accounts no matter what device they are on without having to set up any complicated MySQL databases.

Kivy library:

The Kivy library was chosen because it allows for easy-to-setup GUIs and it allows for cross-platform development.

Various other Python Libraries:

Other python libraries were used to do various things such as: keep track of the time, allow the import of sounds to the project, create popups for errors and displaying other pieces of information, creating an SMTP link to a Gmail account used to send weekly reports to users, etc.

Pycharm IDE:

Pycharm was chosen because it is easy to use and because it offers suggestions when coding that makes creating readable and concise code more efficient. It also has a built-in virtual device that can be resized to test different screen sizes and a helpful debugging mode that was used throughout the project.[12]

JSFiddle.net:

Working outside of college on different projects I became quite confident in my JavaScript programming abilities and so I used JSFiddle to create the first draft of the algorithm that I would later use in the minigame section of the app. After it was tested and I was happy it worked, I converted it into Python code.

Github:

I used GitHub to back up my project whenever I implemented anything new in case I needed to revert to a previous version of the code that I was confident worked and also in case my laptop crashed and I lost my files I could pull them from Github on another device. Thankfully this didn't happen at any stage during development. (My GitHub link: https://github.com/paulpdavis3/FYP)

Windows 10:

General functionality and programs of the operating system I have on my laptop were used daily throughout the development of the application such as the calculator, windows media player, etc.

Airtable:

Late in the development of the application and before I started testing, I began to use Airtable to document any bugs I found or small things I had forgotten to add in that would benefit the app.

Microsoft Office:

Throughout the development of the project, I used different aspects of Microsoft Office for different things, some of these include:

- Using Word to write down any ideas I had whether they were about the dissertation, the application or just thoughts I had that might be useful for me to read later.

- Using Excel to create the GANTT chart seen further on in the document

- Using PowerPoint to create presentations for the interim and final demo of the application and project.

Trello:

Throughout development, I used Trello to keep track of all the individual parts of the application that needed to be developed as well as deadlines for certain aspects of the project.

GIMP (GNU Image Manipulation Program):

This program was used to create all of the image assets found within the application including button backgrounds, correct and incorrect answer images, the Primary Mathletes logo, back buttons, etc.

Audacity:

This program was used to edit and change file formats of the sound files used within the application.

Wireframe.cc:

This website was used to create quick mock-up versions of some of the pages that would eventually be implemented into the application through Kivy.

StarUML:

This program was used to create the use case and entity-relationship diagrams for the project.

Draw.io:

This website was used to create system tier diagrams.

Python IDLE:

After finishing the JS version of the minigame diagram I then converted it into Python using the IDLE so that I could test it worked on its own before implementing it with the rest of the application. This is the process I used for all pure Python functions that were created to ensure they worked independently of the whole app to limit the amount of technical debt associated with the app.

Social Media:

Initially, social media platforms were used for me to connect with primary school teachers to get their input during the ideation phase of the project to see what was necessary for this application to be successful.

Buildozer:

Buildozer was used to compile the project into an APK for use on android devices. To use Buildozer it was also necessary to use Virtual Box to set up a virtual version of Linux 18.04 as currently, Buildozer is only available on Linux.

Gmail:

An email address was set up with Google to send reports to users on request

2.5. Other Research Conducted

Apart from the other research mentioned in this section there was some other research conducted and these are discussed below:

- Twinkl: This website has a lot of resources, both for primary and secondary students as well as teachers. It allows teachers to create plans for their students and children to follow based on the curriculum given.

- Transum Maths: This website has resources for both students and teachers, but it is not tailored to the Irish primary maths curriculum but instead for the British GCSE's.

- NCCA Maths Development: This website provides information on the development of the new Primary Mathematics Curriculum.

- NRich Maths: This website provides problem-solving questions that help to reinforce things that students have learned after they finish a section of the curriculum.

Appendix 1 shows a short document that was sent to some primary school teachers that contained a revised version of the Abstract and Section 1 of this report just to give an overview of the idea for the project, along with a short questionnaire that is also in Appendix 1. A few days after this questionnaire was sent to the teachers, responses began coming back and the results are discussed further in section 5. All information received from this questionnaire was taken on board when development began on the application but a lot of the things that the teachers mentioned such as graphs and a multitude of minigames were outside the scope of this project.

2.6. Existing Final Year Projects

Alex Brady's final year project on "Fantasy Premier League Predictive Analytics" attempted to create a web application that would be able to accurately predict how many points a player would get in the next football match based on the previous seasons. The student split the project into three sections:

The first of which was a predictive analytics system that was created in Jupyter Notebook using the Python coding language. This part of the project would do the heavy lifting in terms of scraping historical data, sorting the data and generating the predictions. The second section was a Python Flask back end web service that held all of the predictive data which was cloud-hosted.

Finally, the third section of the project was a web application that the student developed with AngularJS and NodeJS. This was the part of the app that the user would see when using the student's project. In the end, the student is quite happy with how all of the features ended up after the proposal at the start of the paper.

Although some features were never implemented the core idea for the project was a success. The student was able to confidently answer the question he put forward at the beginning of the conclusion "can a machine know more than me?" and so I believe that the project was a successful one.

The student also planned to continue to update the app and add features in past the submission of the project to DIT.

Keith Mc Loughlin**,** with his project "Temple of Thoth", set out to create a game environment that tracks users' behaviours and uses this information and apply logic to the data in such a way that it could predict the best way to challenge specific users. It aimed to manipulate a 3D virtual environment to lead the player into traps and to discourage them to reach the end goal. The reason for this was to see could the game itself design a challenging level purely based on the player's previous moves and decision-making patterns and use these against them.

The game aimed to see could it control user behavior by using previous data farmed from global user data and actively use player's minds against current players. The game was created inside the unity engine and was written in C++ and used a MongoDB database to store player information.

In the student's conclusion, they talk about how they now know how to effectively design a game, create different scenes and game objects in unity and persist data that was generated inside of the game. This student also talks about how the project could be further worked on and perhaps more on how the project can track user behavior.

While neither of these projects directly relate to the issue that the Primary Mathletes project sets out to solve, reading them showed me the level of work that was needed to complete and document a good project and researching their project management methodologies led to the decision I made on which methodology to choose for my own project so it was a worthwhile exercise.

2.7. Conclusions

In this section, some previous final year projects that students had written were reviewed. Some additional research conducted through questionnaires and conversations with current primary school teachers was discussed as well as a few websites that have primary school curriculum information. Applications other people have already created that attempt to help primary school students and other students of similar age with the maths curriculum were discussed. Programming languages, databases and APIs were researched and decided upon that best-suited development of the application and a full list of all programs/technologies used throughout the project were listed along with their functionalities and why they were important to the development of the app and project as a whole.

3. Design

3.1 Introduction

In this section, common software methodologies are researched and discussed to find out which one is best suited for this project. An overview of the whole system is given to visualize what aspects of the app fall within the front middle and back end of the system. Systems that were put in place to ensure the project was finished before the deadline are shown. After this, a detailed insight into the design of every aspect of the app are given from the design of the front end that is shown to the user to the structure set out in the database to hold all of the users information.

3.2. Software Methodology

Scrum –

Scrum is a methodology that is comprised of five values: commitment, courage, focus, openness, and respect.[13] It usually involves someone taking over the role of scrum master who organizes daily stand-ups and points out places in development where there may be blockages and see can they be removed in one way or another. It also involves doing weekly or bi-weekly sprints which are fast and efficient pushes of work on one particular task over a short period.[14] After this is finished there is a sprint review and retrospective, where lessons are created that, will ensure that sprints that follow the last will always be more efficient. Scrum is a methodology that is usually used when working with a team on a project but since this is a solo project, it did not seem like it would be a suitable choice.

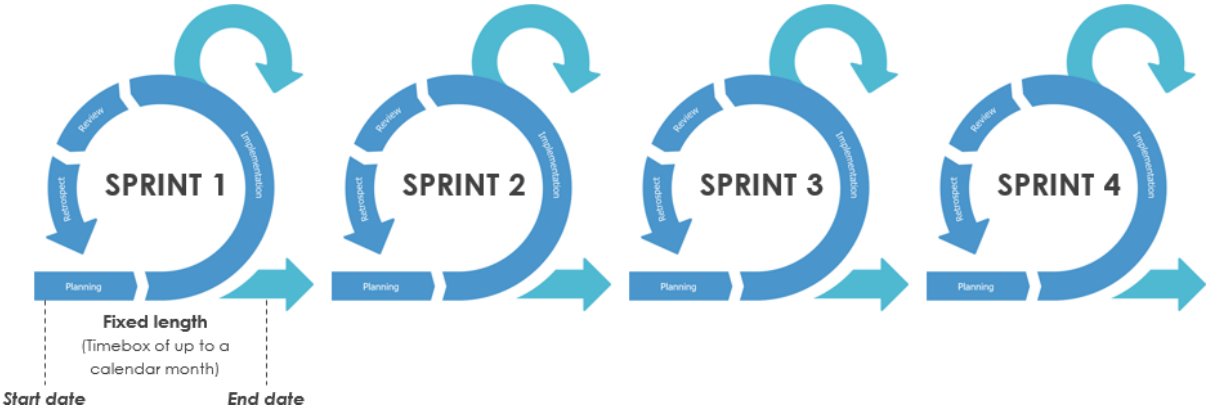


Figure 9 Sprints lifecycle as part of the Scrum Methodology [15]

Agile –

Agile is one of the most famous and recognizable project management methodologies. It was developed by industry leaders that wanted to uncover better ways of developing software by having clear and measurable goals that take into consideration iterative development.[16] It is made up of 4 fundamental values and 12 key principles. Some of the main features include having working code over comprehensive documentation, responding to change over following a set-in-stone plan and being simplistic. I decided not to use Agile but instead opted for a subsection of Agile which can be read below.

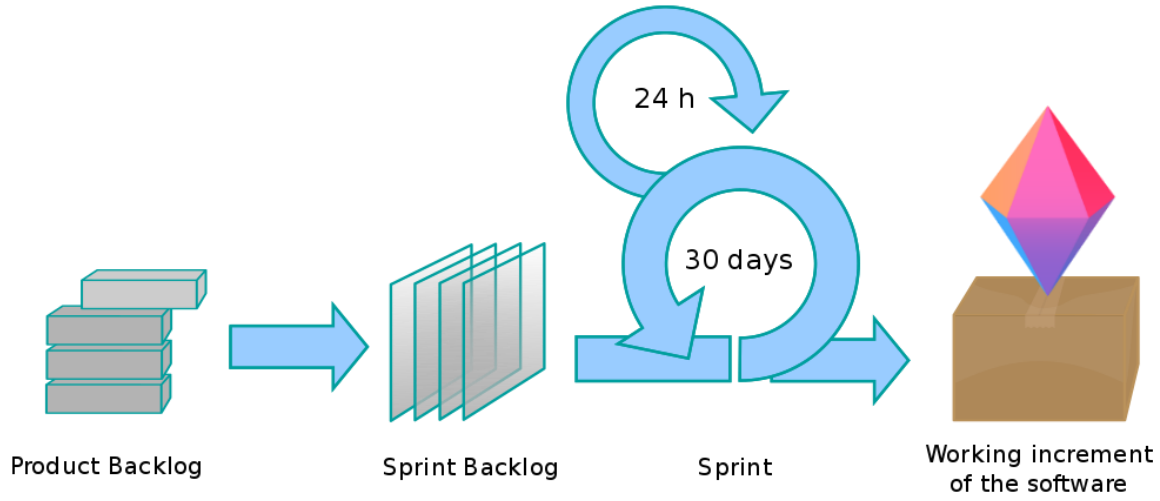


Figure 10 Scrum lifecycle as part of the Agile Methodology [17]

Kanban Methodology –

A Kanban board is one version of the Kanban project management methodology that is used to keep track of tasks to be done during a project. It is a subsection of the Agile approach to project management and they are used to visualize workflow for a given period and helped me a lot throughout the progression of the project as it let me add items to the backlog whenever I thought of something that I may have forgotten to add earlier or a new idea that I thought would make a good addition to the project.

Trello:
The image below shows how I used Trello as my Kanban board for the project. Each task is given its own card and was initially placed into the backlog list. The structure of the Trello board allowed me to pick one task at a time and work on it until it was finished, then it was either added to the "testing" list or the "completed" list depending on whether testing was necessary or not.

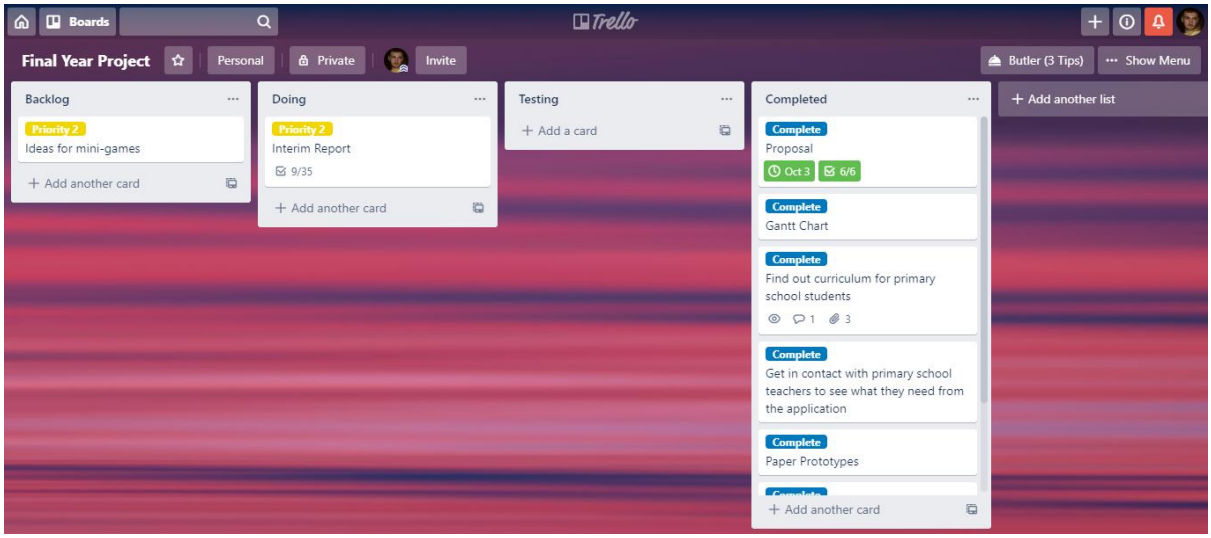


Figure 11 Project Trello board

Below, you can see what is inside of the cards themselves. Usually, inside a card, there would be a breakdown of all the working parts that need to be completed for the overall task to be finished. Normally I would create a checklist as you see in the image below but I also have cards where I have added useful links to pages I have researched. Comments and descriptions that aid in completing the task are also added to the card.
By adding comments and checklists to the cards it helped to visualize the tasks that needed to be complete without cluttering the overall Trello board with hundreds of small tasks that could be grouped.

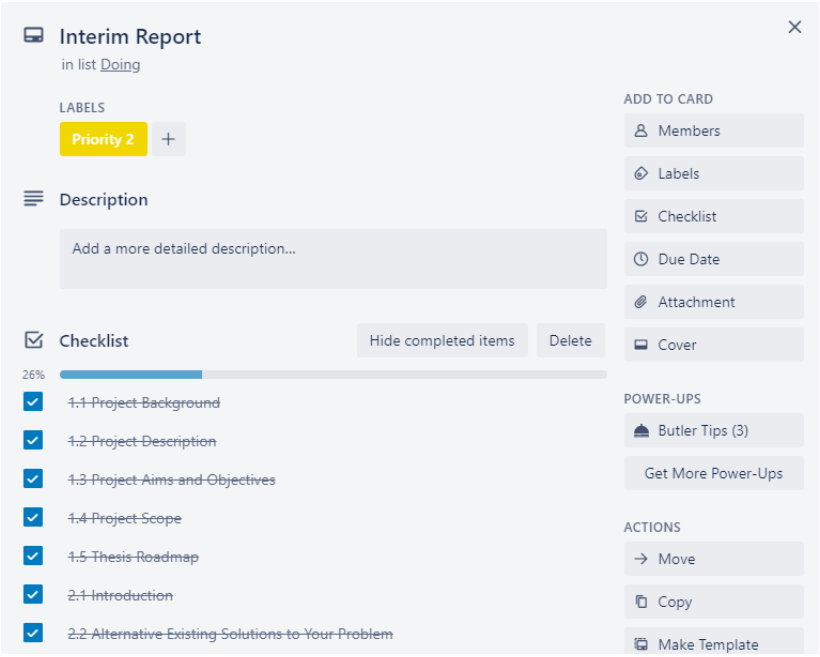
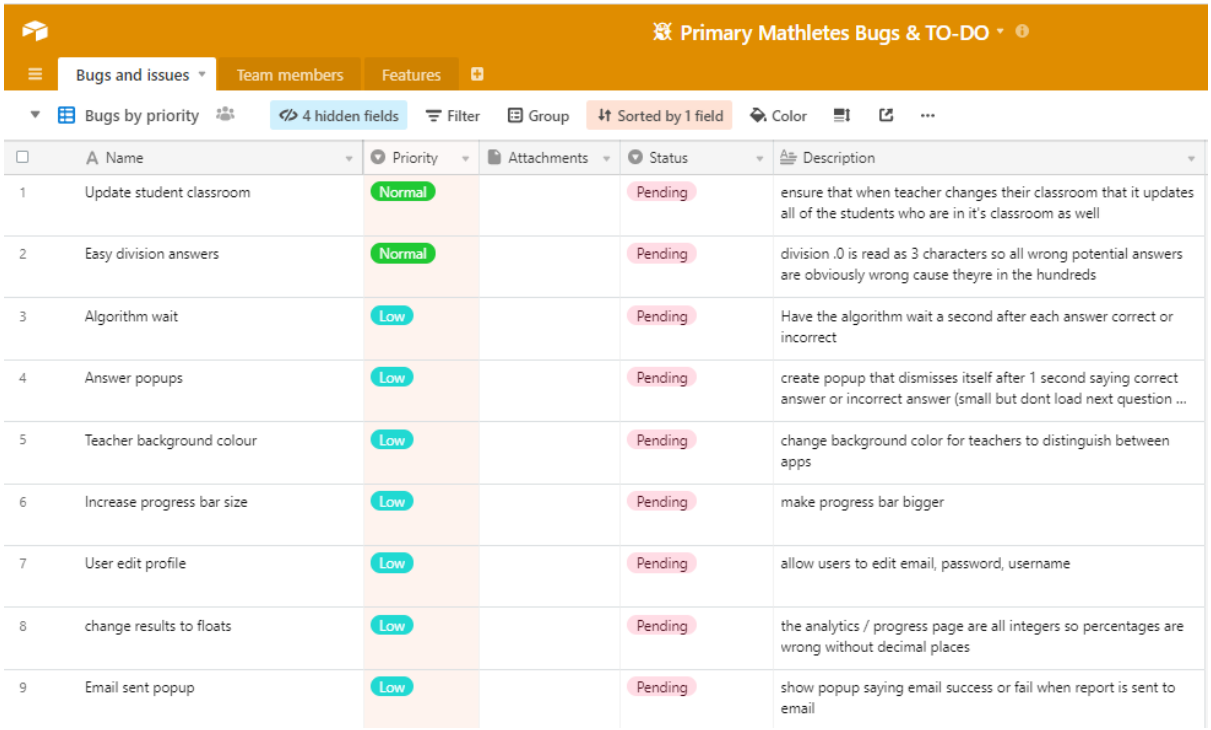


Figure 12 Card details for Interim Report

Airtable:

Airtable is another example of a Kanban board and the one seen in figure 13 below shows the one that I used as a bug list while I was personally testing the functionality of the Primary Mathletes app.



Primary Mathletes Bugs & TO-DO

Bugs and issues | Team members | Features

Bugs by priority | 4 hidden fields | Filter | Group | Sorted by 1 field | Color

	Name	Priority	Attachments	Status	Description
1	Update student classroom	Normal		Pending	ensure that when teacher changes their classroom that it updates all of the students who are in it's classroom as well
2	Easy division answers	Normal		Pending	division .0 is read as 3 characters so all wrong potential answers are obviously wrong cause theyre in the hundreds
3	Algorithm wait	Low		Pending	Have the algorithm wait a second after each answer correct or incorrect
4	Answer popups	Low		Pending	create popup that dismisses itself after 1 second saying correct answer or incorrect answer (small but dont load next question ...
5	Teacher background colour	Low		Pending	change background color for teachers to distinguish between apps
6	Increase progress bar size	Low		Pending	make progress bar bigger
7	User edit profile	Low		Pending	allow users to edit email, password, username
8	change results to floats	Low		Pending	the analytics / progress page are all integers so percentages are wrong without decimal places
9	Email sent popup	Low		Pending	show popup saying email success or fail when report is sent to email

Figure 13 Airtable Bug Table

After researching these methodologies, it was an easy decision to adopt the Kanban board and use Trello to keep track of the overall project and use Airtable as another means of organization to hold all of the bugs and small programming tasks that had not been implemented yet.

GANTT Chart:

Another item used to keep the project on track was the GANTT chart that was first developed at the start of the academic year. Over time the GANTT chart had to be altered to accommodate an extension on the project deadline but for the most part, the plan laid out in the GANTT chart was followed as close as possible to ensure that deadlines were met and to make sure that no one aspect of the project got more attention than others. The final version of the GANTT chart can be seen in figure 14.

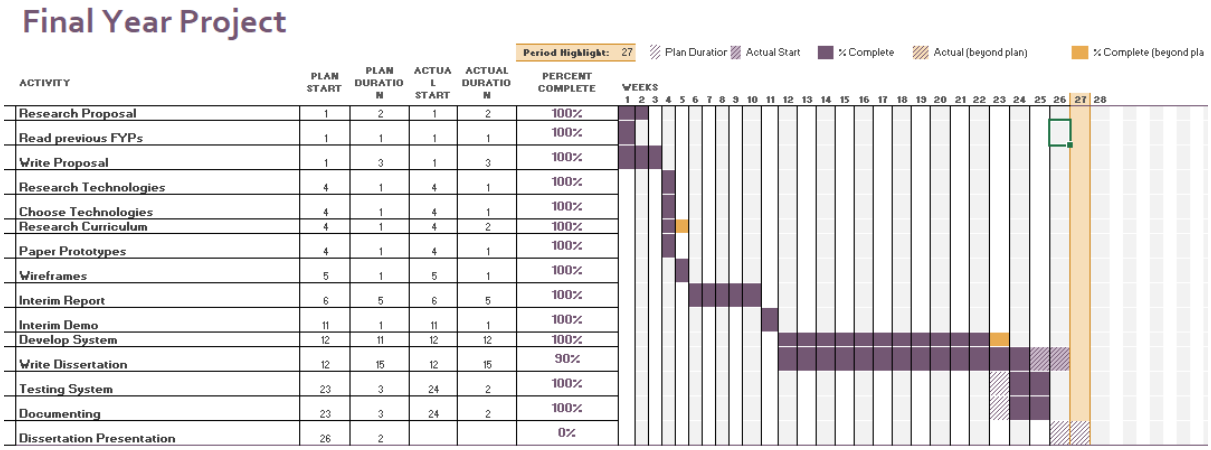


Figure 14 GANTT Chart

3.3. Overview of System

Figure 15 shows the proposed three-tier system diagram that was created earlier in the academic year. It includes both android and iOS mobile devices. Figure 16 shows a simple technical architecture diagram of the application.

Tier one of the system will be everything that the end-user sees from the login screen to the minigames, the progress page and everything else shown on the mobile device.
Tier two of the system is all of the code written in Python that creates the screens that the end-user sees as well as the code that connects to the database that holds all of the users' information for use within the app.
Tier three of the system is the database that holds all of the information that can be accessed by any user on any device. It holds a table of all the users including their overall experience points, their login details, whether they are a teacher or not and whether they are in a classroom or not. It also holds information about the classrooms themselves which lets any user anywhere join a classroom.

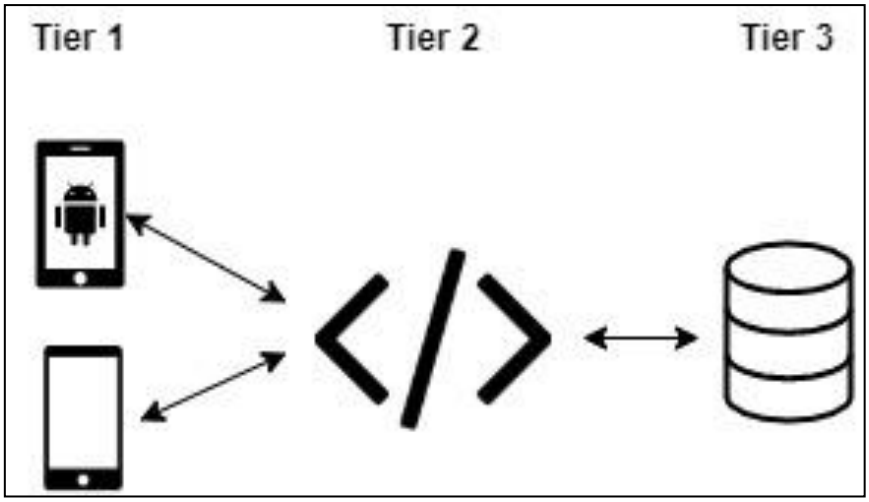


Figure 15 Three-Tier System

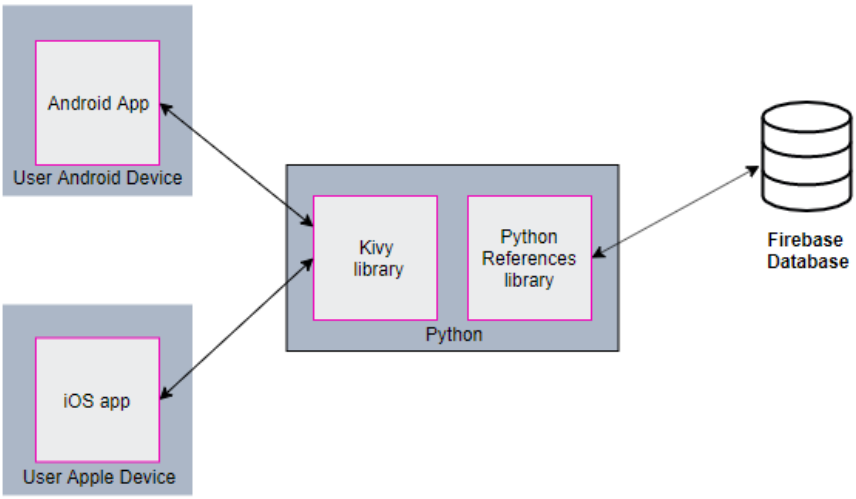


Figure 16 Technical Architecture Diagram

3.4. Software Test plan

Black Box Testing:

As mentioned in the project description (Section 1), the black box testing plan of this project was unfortunately interrupted by the outbreak of COVID-19 and so only a limited amount of this type of testing was possible through interviews and run-throughs of the application with my family. Luckily one of which, my younger sister, is a primary school student herself, so this was a big plus for the testing portion of the app.

The original plan of using actual groups of primary students was not possible and this was a big aspect of the testing plan that would allow me to redevelop a superior version of the application but instead, a lot of the testing and redevelopment suggestions had to come from myself which presumably

impacted the final version of the application seen in this project. Wherever possible, I still managed to get input from students on things such as colour scheme, font choices and other aspects of the application that are discussed further in this section.

White Box Testing:

Using knowledge of the inner workings of the application, test cases were created to test the code in place of a wider group of primary school students testing the application for me. As well as this, manual unit testing was conducted by myself to ensure individual functions of the application worked on their own before being implemented into the wider system technical debt within the project. An Airtable bug list was also set up to keep track of bugs that I knew of myself and could fix whenever I got the time to.

3.5. Front-End

Tier one of the system is the presentation layer and it consists of everything that the user sees when they are using the application. It allows the user to interact with the app through text input boxes and button presses and is both the source of input and output to the system and the user.

Originally, rough prototypes were drawn out by hand and in one way or another they were used as inspiration for the final look of the application. Some were copied over identically, and others were changed to different versions of the same thing. The paper prototypes were included in the document that was sent out to teachers mentioned in section 2 and this helped with moving forward in development as I received input from the target market of the application.

Register Page:

As seen in figures 17 - 20 the register page didn't change much throughout the development of the app. The original paper protoype and the final version seen in the app are very similar but with the addition of the teacher check box and login button in the final version. The wireframe protoype shows that originally the user would have to confirm through a code sent to their email that they were the ones registering for the account. This idea was eventually scrapped as it did not fall within the scope of the project and it was an unnecessary step that the project didn't require.

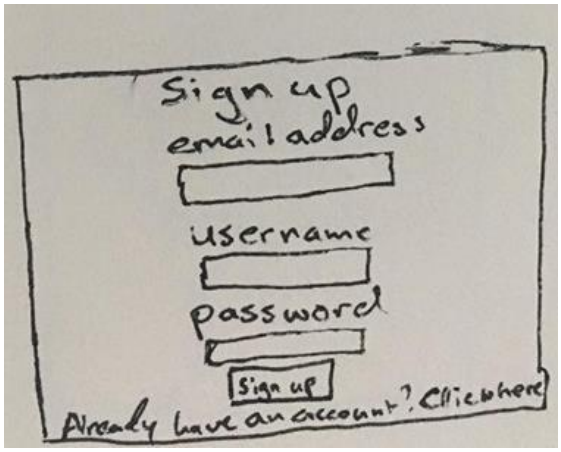


Figure 17 Register Paper Prototype

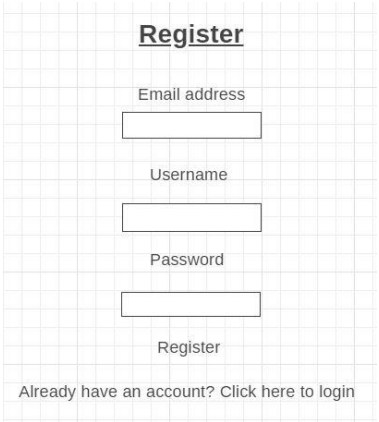


Figure 18 Register Wireframe Prototype 1

Figure 19 Register Wireframe Prototypes 2

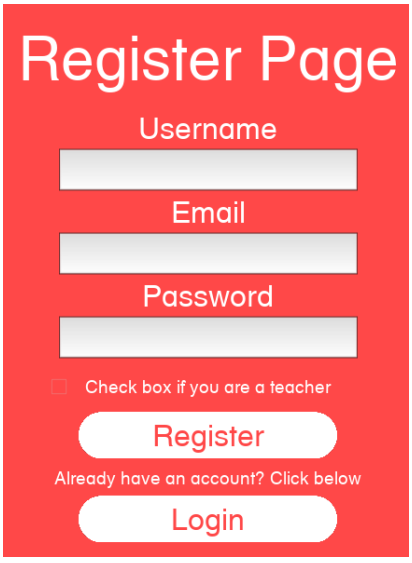


Figure 20 Final App Register Page

Login Page:

Similar to the evolution of the register page, the login page did not change much throughout the development of the application as can be seen in figures 21, 22 and 23 except for the addition of a registration button in place of a clickable word that was originally seen in the paper and wireframe protoypes.

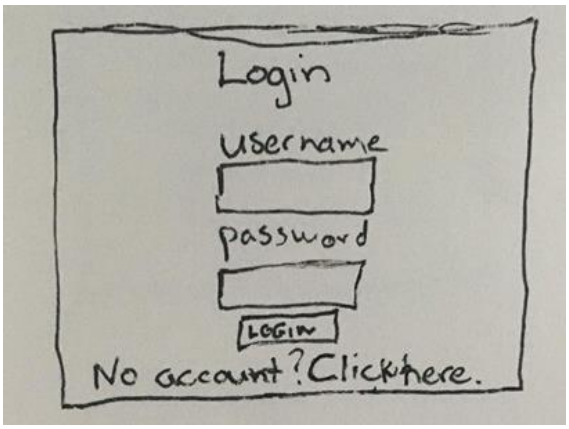


Figure 21 Login Paper Prototype

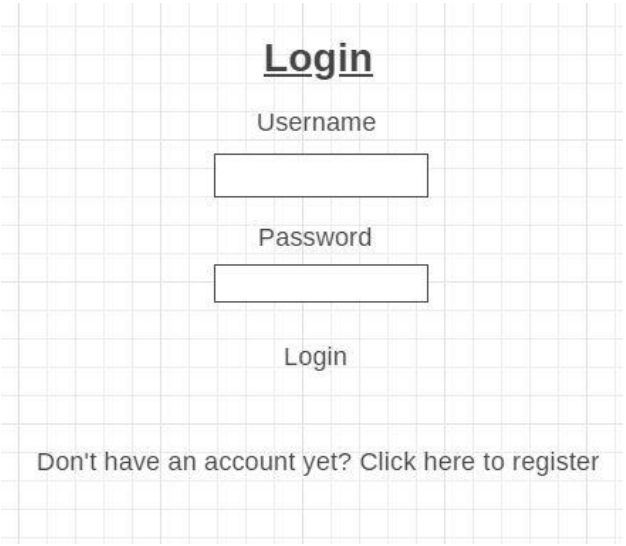


Figure 22 Login Wireframe Prototype

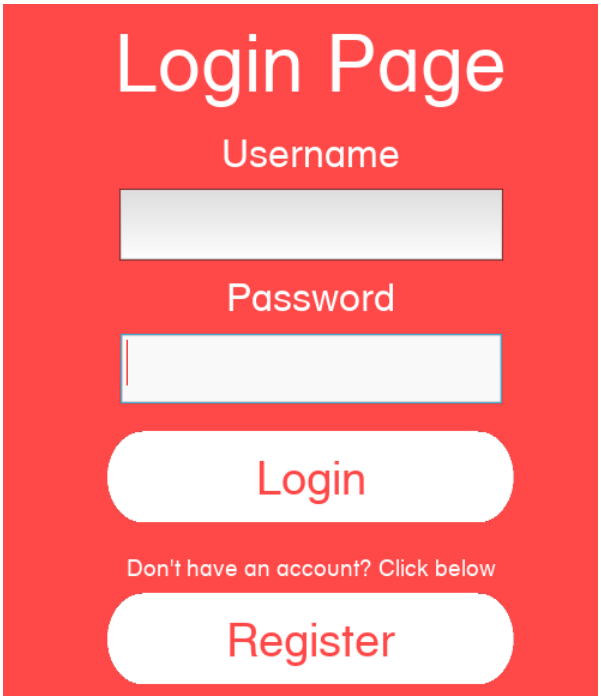


Figure 23 Final App Login Page

Main Menu:

In figures 24, 25 and 26 you can see the drastic changes that the main menu went through while developing the app. Originally in the paper and wireframe prototypes the main menu was going to be based on the Duolingo level tree that can be seen in figure 27 with the users profile being displayed in the top-right corner. Later in development it became clear that this kind of graphical user interface wasn't possible to make within the limitations of Kivy. As well as these limitations it was also decided that a more basic approach with clear indication of where the user will be brought to next was a better option especially when designing an application created for younger children. The final design shown is a clean and easy to understand main menu that even the youngest students in primary school could use comfortably. The final design allows users to click buttons to enter into the operation selection screen, view the classroom screen, see their progress or edit their profile. The main menu also allows the user to log out of the app if they wish which is a feature not seen in the paper and wireframe prototypes.

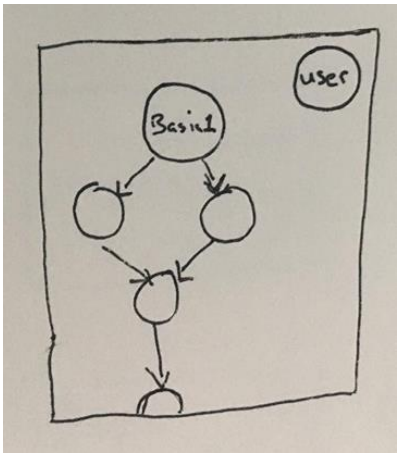


Figure 24 Main Menu Paper Prototype

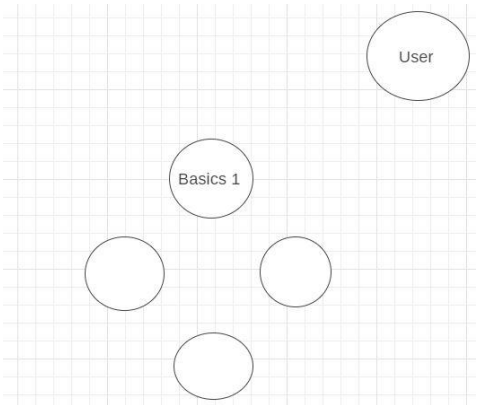


Figure 25 Main Menu Wireframe Prototype

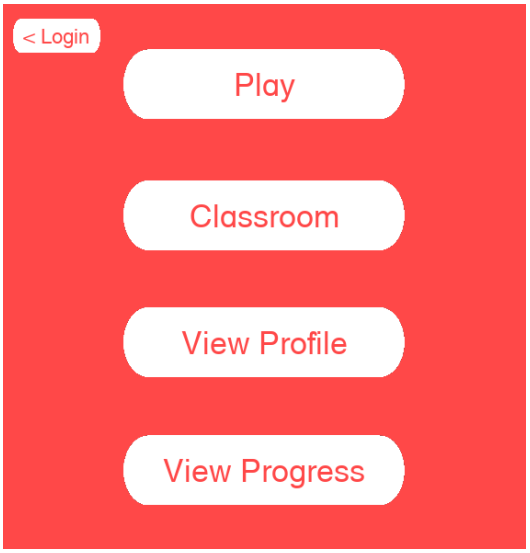


Figure 26 Final App Main Menu

Figure 27 Duolingo Level Tree

Minigame Page:

The evolution of the minigame page changed slightly over the course of development. Comparing figures 28 and 29 to figure 30 show that the paper and wireframe prototypes of the minigame page aren't too different from the final version seen in the app. The final version is a cleaner and more structured version of the paper prototype, more inline with the types of mathematical arithmetic that students would be used to. Students responded better to the numbers appearing in a vertical manner rather than a horizontal one as is seen in the paper and wireframe prototypes. Other additions include the round number, level number, and a timer to keep track of how long the student has spent on a particular level. The user also has the ability to click the 'X' at any time and are prompted with a popup that asks whether they would like to leave the current minigame or not. The wireframe prototype shows a progress bar that shows the users progression from zero to six questions answered. The original number of questions per minigame was 6 but this was increased to ten in the final version of the app. The idea of having 6 multiple choice answers was also cut down to 4 to give the user less options per question and to declutter the page.

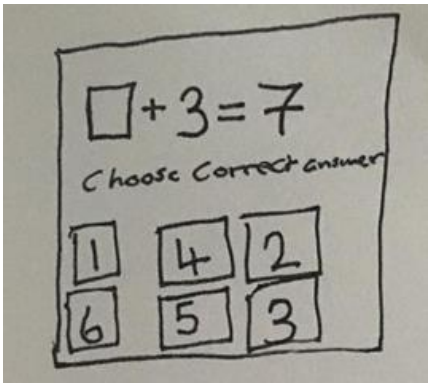


Figure 28 Minigame Paper Prototype

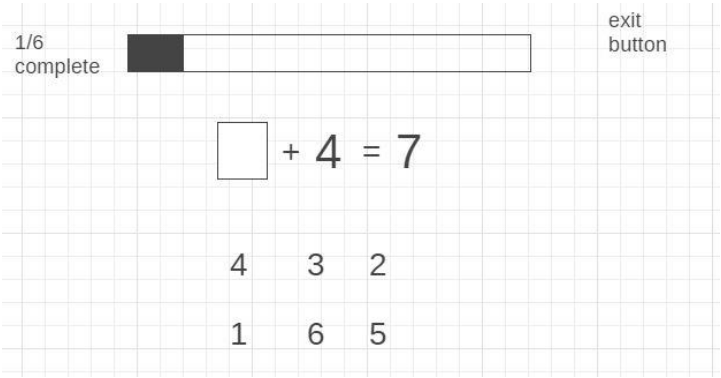


Figure 29 Minigame Wireframe Prototype

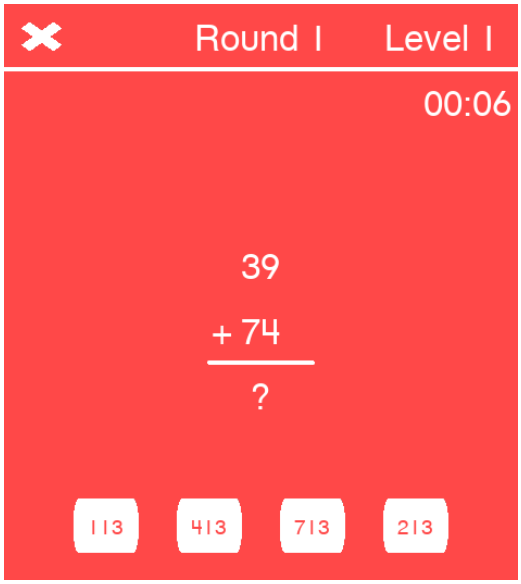


Figure 30 Final App Minigame Page

Results Page:

Early in development, the results page was called the finish page and there was a drastic change between the prototype versions seen in figures 31 and 32 and the final app version seen in figure 33. Originally the results page was going to have a very minimal amount of information on it as well as displaying which questions the user got correct and incorrect. After the number of questions per minigame was increased to ten this information about which questions were correct and incorrect became redundant as the student would be unable to remember 10 questions ago to which question they even got correct or incorrect so this aspect of the results page was removed. Instead, in the final version, a progress bar was added to add to the gamification of the minigames to visually show the users how far away they were from unlocking the next level available. The final results page also shows the user how many experience points they have in total so that they don't have to exit the minigame page and travel to the progress page just to see the same information that can just be shown here. The exit button was also moved to a more central and low position on the device so that it was easy for the user to press this button with their thumb. This made sense because on the previous page (the minigame page) the user would be pressing buttons in the same location as this button and so to press this mandatory button would not disrupt the flow of movement from the user too much.

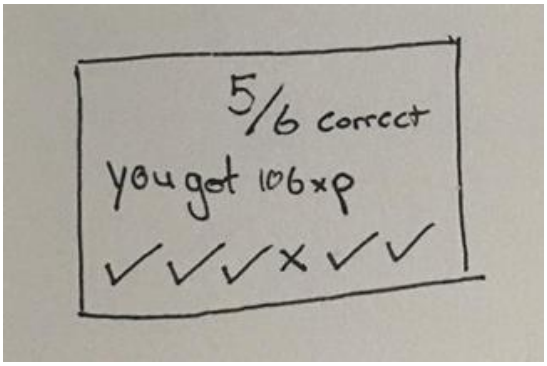


Figure 31 Results Paper Prototype

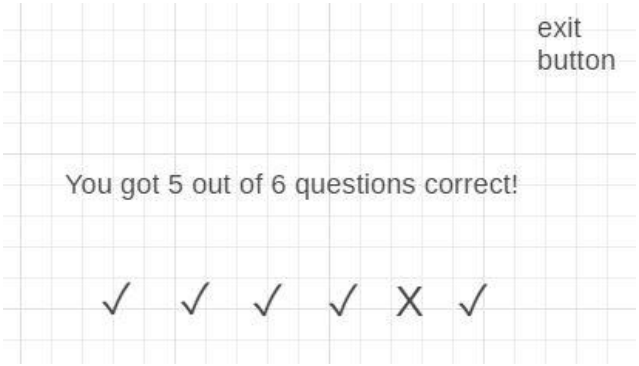


Figure 32 Results Wireframe Prototype

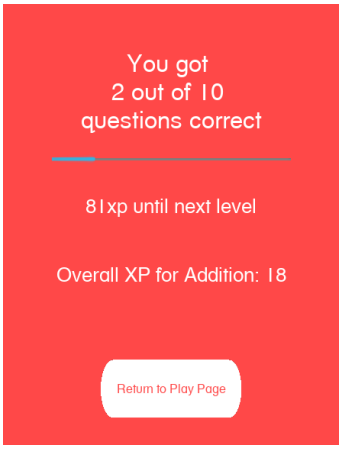


Figure 33 Final App Results Page

Progress Page:

This page probably changed the most throughout development from the paper prototype and wireframe to the first and last version of the in-app page. The main difference between the first and last versions of the app is the absence of the proposed graph that would be on the progress page. After discussing the progress page with various primary school teachers it became apparent that students that young wouldn't understand graphs even if they were simple. Something text-based like the results page seen in figure 36 would be a lot more suitable for both teachers and students. After scrapping the progress graph the first implementation of the progress page was a basic rundown of all the users' experience points and the amount they had overall. This version of the progress page was also eventually scrapped as it was too simple and didn't serve any purpose. The same information could be seen on the results page. A better and more complex version of the progress page was then proposed and created and that is the final version seen in figure 37. This final version allows the users to click on any week of the year and instead of basic experience point values they are given more information about the types of progress they were making. They can see their total playtime for that week, the number of minigames they played, their best score, the number of correct answers they got and the total number of experience points gained that week. As well as this there is also information displayed about the previous week if there is any and a change whether that be positive or negative is displayed beside these values so that students and teachers can see if progress is being made or not. This was not possible in previous iterations of the progress page and so was a welcome change for result analysis. Also on this page is a button that allows the user to send themselves or a parent their weekly progress report and this will be discussed further in section 4.

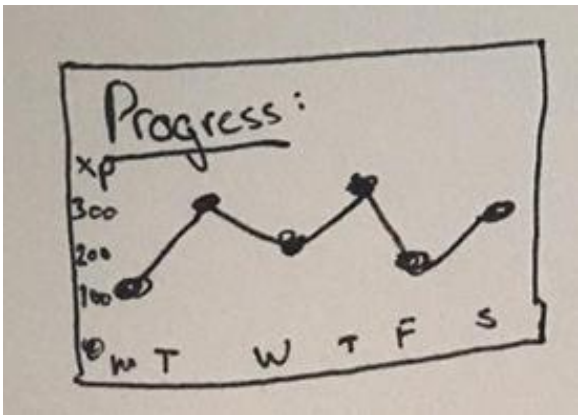


Figure 34 Progress Paper Prototype

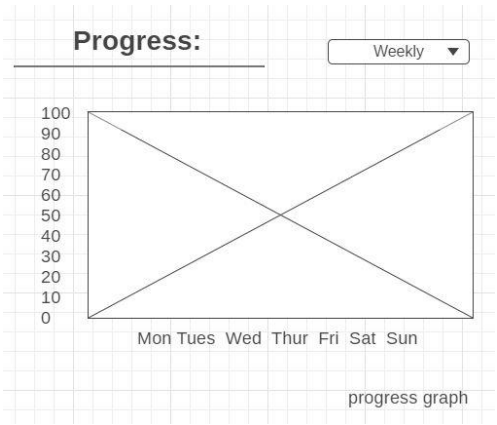


Figure 35 Progress Wireframe Prototype

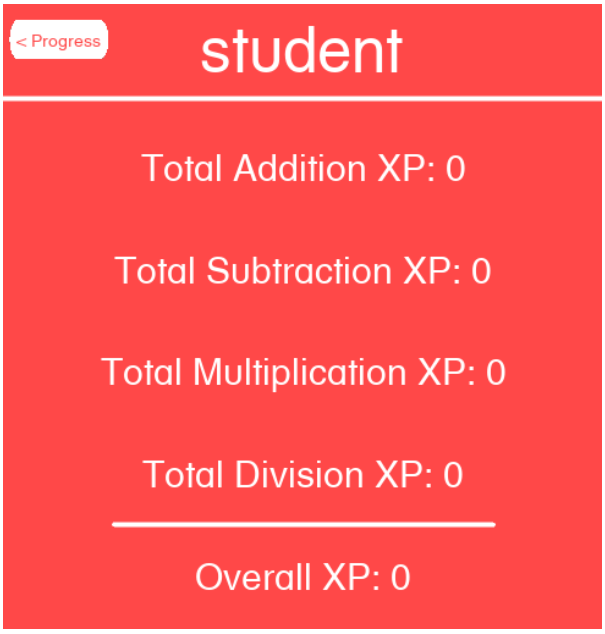


Figure 36 Old version of Progress Page

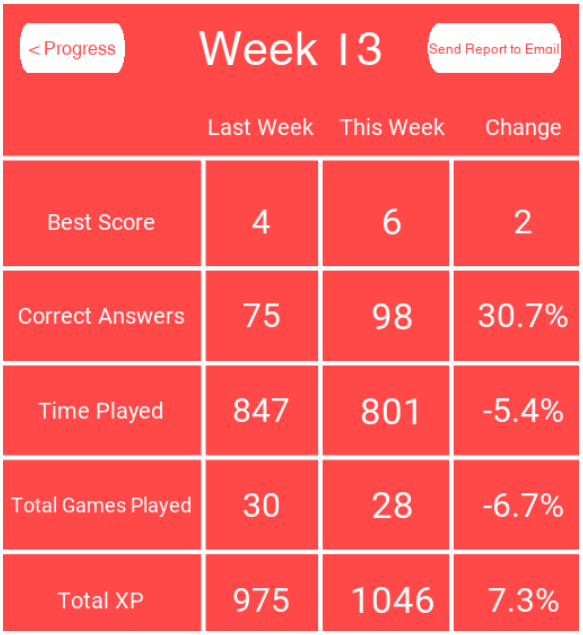


Figure 37 Final App Progress Page

Other screens:

- Figure 38 shows the title page that wasn't designed until development began on the app.

- Figure 39 shows the classroom page that allows teachers to create and delete classrooms and students to join and leave classrooms.

- Figure 40 shows the play page which is the page that allows users to pick what operator they would like to use for the minigame.

- Figure 41 shows the screen that is presented after choosing an operator and shows all of the levels that the user can choose from. All operators have 10 levels and each level unlocks after an additional 100 experience points are gained. Level 2 unlocks at 100xp, Level 3 unlocks at 200xp, etc.

- Figure 42 is what the teacher sees when they enter the progress screen. It shows all of the students that are in that teacher's virtual classroom in a list and the teacher can click on a student and they are brought to the progress page for the selected student. If the teacher does not have a classroom then the list will be empty and they will only have the option to go back to the main menu.

- Figure 43 shows the edit profile page. The final version of this page currently only allows users to edit their username or delete their account but in future updates, there may be an option to set a profile picture, edit the colour scheme of the app, update email address, etc.

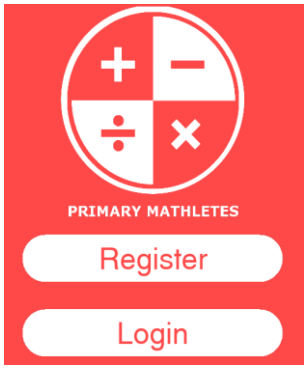


Figure 38 Title Page

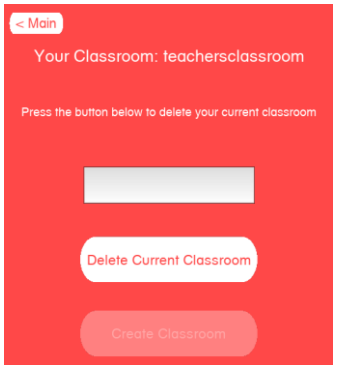


Figure 39 Classroom Page

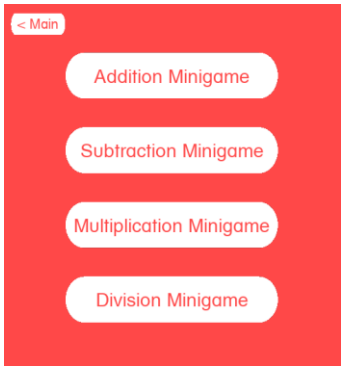


Figure 40 Play Page

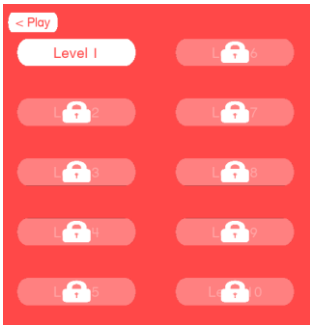


Figure 41 Level Selection Page

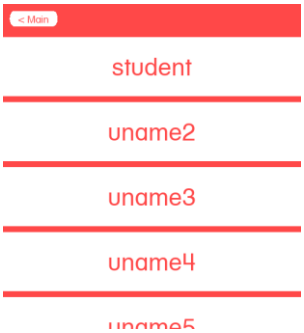


Figure 42 Teacher's Student List

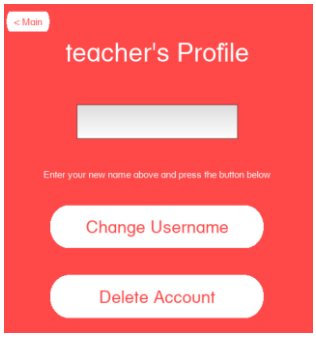


Figure 43 Edit Profile Page

Font:

The font chosen for the text found inside the application was decided late into development because I wanted the different pages to all be connected. This was so the different font choices could be seen throughout the app while the screens could be used in conjunction with each other. I decided on a few fonts that looked good and then looked into articles online about different fonts that were suitable for younger children and came up with a questionnaire that let people choose which font they liked the most. People of all ages were questioned, and the results of this questionnaire can be seen in section 5.

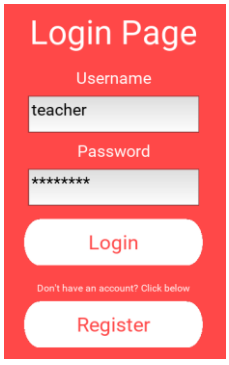


Figure 44 Login Page using Kivy Default Font

Button Design:

Humans enjoy rounded buttons more than straight edge buttons. This is because rounder buttons are softer on the eyes and the non-sharp edge of the button allows users to concentrate on the text found inside the button.[16] On screens like the level selection screen where there are eleven buttons present on screen at once, it is more pleasant to look at round buttons than a grid of rectangular ones and that's why I decided to design a rounded button for use within the app. When the user presses any of the rounded buttons the button transforms into a darker version of itself to give the illusion that the user is pressing an actual button to add to the immersiveness of the application. The custom button also has enlarged text to help younger students see them better on the screen.

Figure 45 Default Kivy Button

Figure 46 Custom Rounded Button

There were other custom buttons designed like the 'X' button found in the minigame page (figure 47) but these were ultimately left out of the app and instead, words were used instead. The reason for this was because younger students might not have the mental know-how to remember what each icon meant but they could read the word and look at the arrow to understand where and what direction the app was going to go once they pressed the button. Examples of used and unused buttons can be seen below.

Figure 47 'X' Button

Figure 48 Unused Back Button

Figure 49 Unused Log Out Button

Figure 50 Final App Button Example

Transitions Between Screens:

Figure 51 shows all of the app's screens and how they interact with each other. The main menu is the central hub that all the other pages connect to.

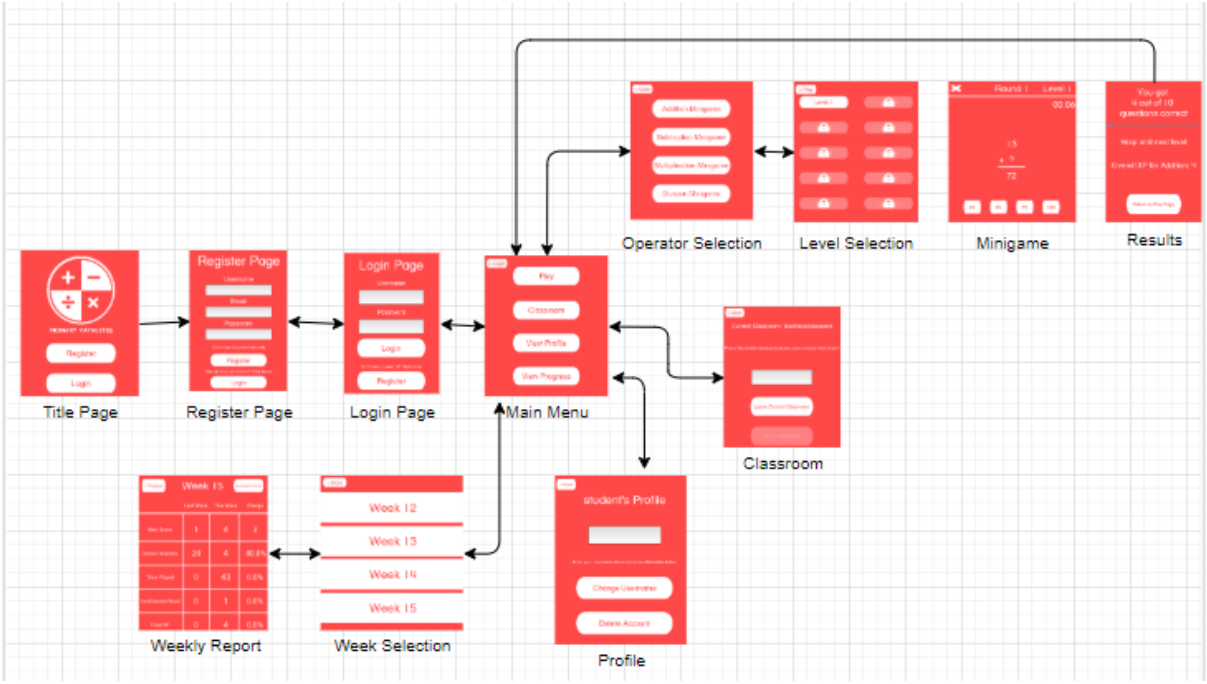


Figure 51 Screen Flow of App

Popup Design:

Popups are used in various places within the app to ensure that entered information is correct and screen transitions are intentional.
Figure 52 shows a popup shown to the user if they don't input a suitably strong password when registering an account. Figure 53 shows a popup that the user will see if they press the log out button. This is to ensure that users aren't logged out instantly if they miss-click the logout button by accident.

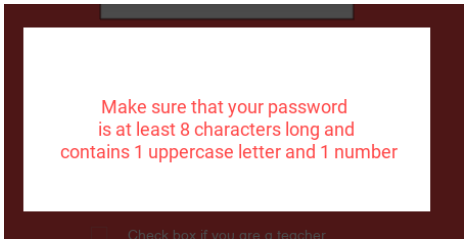


Figure 52 Unsuitable Password Popup

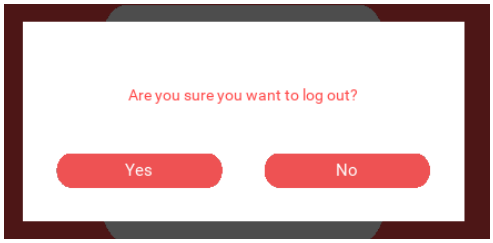


Figure 53 Log Out Popup

Popups shown in figures 54 and 55 are displayed when the user gets a question right and wrong respectively. When the user gets a correct answer, they are shown a smiley face. Positive reinforcement was promoted through the use of imagery and sound in the minigame section of the app. Positive reinforcement has been proven to improve retention of information when the user is told that they did a good job when performing a task.[17] The sad face was eventually added as well as there needed to be some sort of indication, apart from sound, that the user had gotten an answer wrong.

Figure 54 Correct Answer Popup

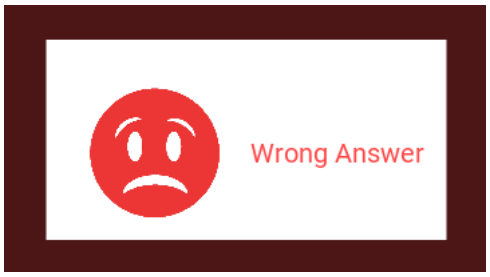


Figure 55 Incorrect Answer Popup

There are other popups shown to the user throughout the application. One example of this is a popup being shown to the user when they register an account with a username that has already been taken by someone else. In this case, a popup is shown to the user explaining the situation and that they should try a different username.

Colour:

The use of bright red backgrounds and contrasting white buttons draws immediate attention to the application from the get-go. The colour red in most western countries indicates alertness, danger and something to pay attention to and it is a strong stimulus. [20] While this colour choice can be straining on the eyes, the application isn't meant to be played for hours and hours at a time but rather, a few minutes at a time to brush up on information that the student already knows and to solidify their learning of a particular topic. Bright and contrasting colors also appeal to younger people so the red and white combination seen throughout the application was suitable.[21]

Sounds:

Two sounds were used within the application and they were used in conjunction with the correct and incorrect popups used in the minigame page to add to the gamification of the application as well as to sonically remind students of how well they were doing in a particular minigame. The sounds were added because some people respond better to affirmative sounds rather than images.[22]

Use Cases:

Some of the original use cases that were created can be seen below. The first iteration of these use cases was suitable for both students and teachers as it was just a basic login, register and log out (figure 56). After this, in the second iteration (figure 57), students were to have the ability to view their profile, select a minigame, view their classroom and view their progress. Finally, in the third iteration (figure 58), students could edit their profile, delete their profile, play minigames, complete minigames, view their scores, gain experience points, exit minigames, leave and join classrooms and view awards.

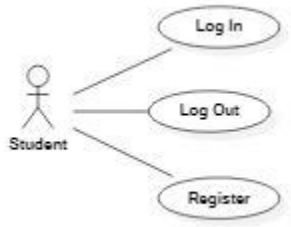


Figure 56 First iteration use case for both students and teachers

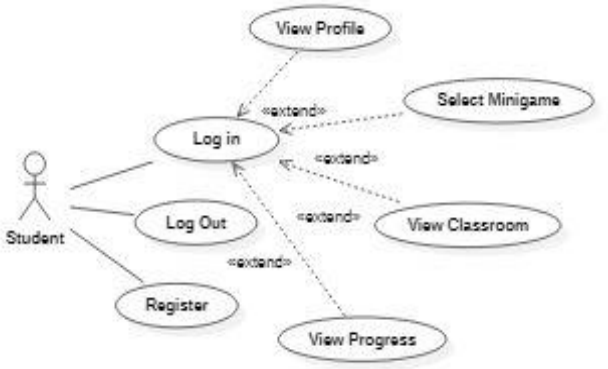


Figure 57 Second iteration use case for students

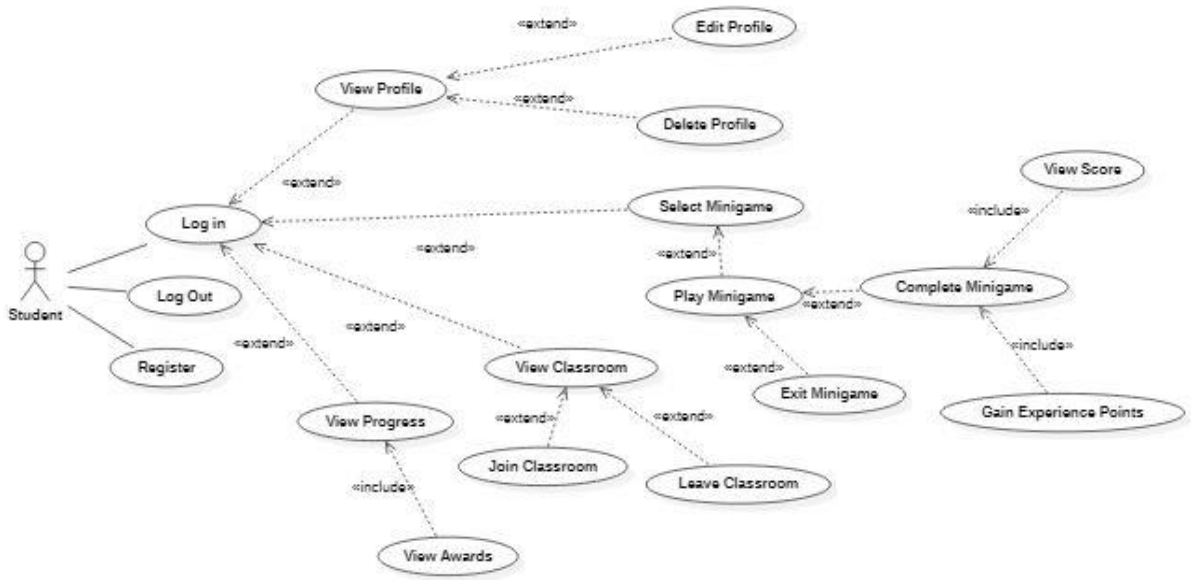


Figure 58 Third iteration use case for students

Teachers on the other hand, for the second iteration, could only view their profile and view their classroom. Then for the third iteration, teachers were also able to edit and delete their profile, create, delete and remove students from their classroom, view individual student progress and manage info about these students.

After the Christmas break when development properly began and after an initial run of questions went out to teachers and users alike, it became clear what was actually necessary for this application and not just what I thought would be suitable in my opinion and so, new use cases had to be created to hold these changes to the original idea.

The original first iteration drawn still stands for the final application. The student and teacher both need the ability to register, login and log out. After some thought it was decided that the ability to play minigames would also benefit the teacher so that they could get comfortable with the system that their students would be using and so both the student and teacher ended up sharing the second iteration use case shown in figure 59. The only change between this use case and the one originally created is that minigame selection doesn't immediately happen on the main page but rather later on in the third iteration use case.

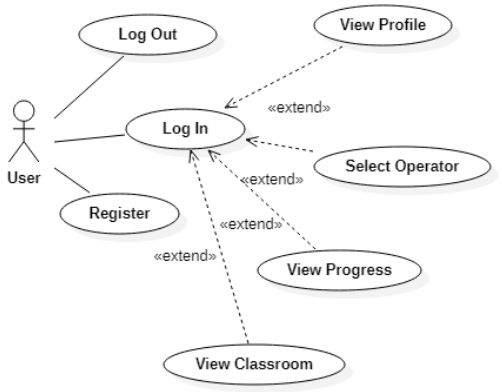


Figure 59 Final Second Iteration for User and Teacher

For the third iteration of the use cases, the students and teachers part ways in some respects to give both of them individual control of their own version of the app. In their third iteration, students can select a minigame level, play the minigame, gain experience, see their results from the minigame, join and leave a classroom, edit or delete their profile, view a weekly report that analyses their progress from the previous week and sends themselves an email of their report. Originally students were going to be rewarded with badges/icons (seen in the original third iteration use case) that would be displayed in the progress menu but this idea was eventually scrapped as it didn't add much to the application and would have required a lot more graphic design which was not within the scope of the project.

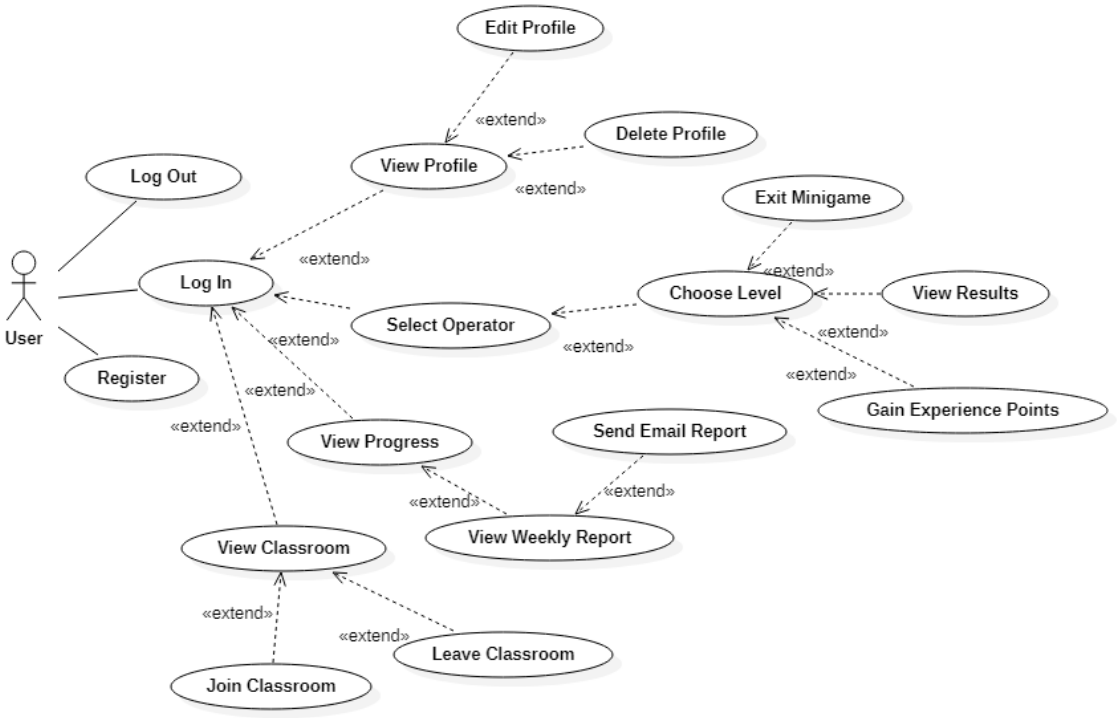


Figure 60 Third Iteration User Case for Students

Looking at the teachers' third iteration use case in figure 61 shows that teachers can now do almost the same things that students can do except they can pick on individual students to then view their weekly progress reports as long as they are in the teacher's virtual classroom. Teachers can also create and delete classrooms and remove unwanted students from their classrooms rather than leave and join classrooms which is an ability only students have.

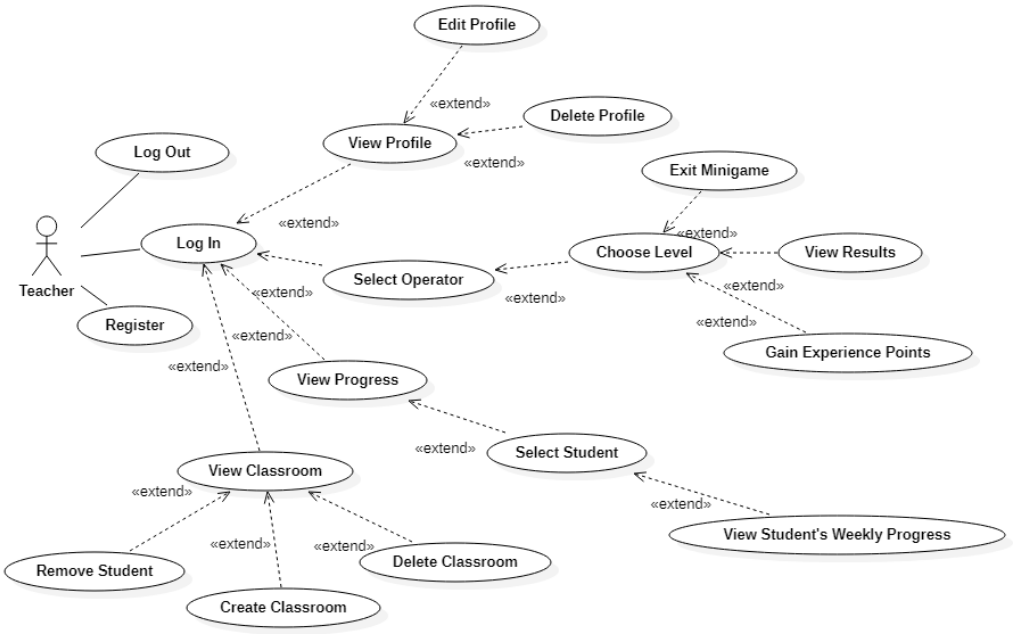


Figure 61 Third Iteration Use Case for Teachers

3.6. Middle-Tier

The middle tier contains all of the functions in the code that drive the application's capabilities. It allows users to input data, manipulate this data in whatever way is necessary and then push information back to the user. It also allows for communication between the front and back end of the application through functionality such as:

- Logging in
- Registering an account
- Editing their profile
- Playing minigames and gaining experience points

It also handles underlying functions that the user might not be aware of such as monitoring the time spent playing minigames, the number of correct and incorrect answers the user has gotten overall and dynamically updating labels throughout the app that change based on who is using it.

More detail on the middle tier of the app will be discussed in section 4.

3.7. Back-End

Figure 62 shows the original Entity Relationship Diagram that was designed before I had a proper understanding of how the back end was going to work. Comparing this to the ERD that is used in the final version, it is possible to see many differences. Firstly, users are just users. There is no difference in the database between a teacher and a student except for the fact that the teacher Boolean is true for teachers and false for students. All other attributes are the same. Users can also have a classroom. Whether the user is a teacher or not will decide who owns the classroom and who is a member of the classroom. Each classroom can have many users but each user can only have one classroom. All users can also have progress entities. These entities hold information about the user on a week-by-week basis throughout the year. Each user can have many progress entities but each progress entity only belongs to one user. This ERD is implemented using Google's Firebase Realtime Database and more information about how the database works will be discussed in section 4.

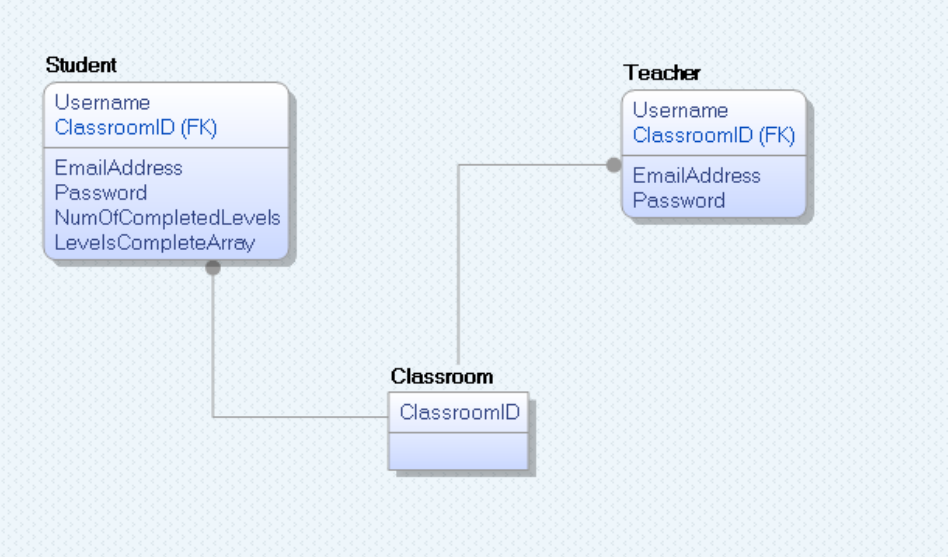


Figure 62 ERD showing relationships between tables

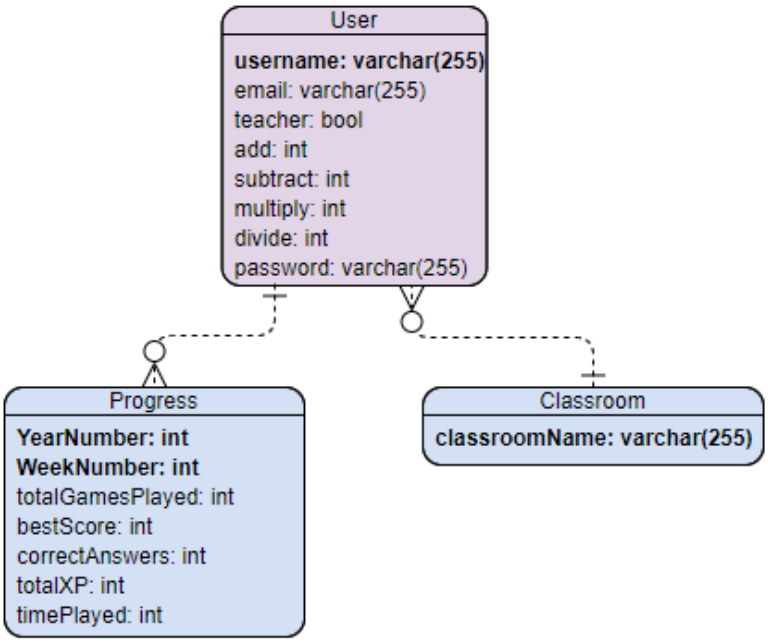


Figure 63 Final Version of ERD

3.8. Conclusions

This section of the project took an in-depth look into the design that shaped how the application sounded, looked and felt to the user. Using this design section moving forward it was clear to see what needed to be implemented and what didn't. The evolution of the changes made to the design from the start of the academic year to the interim and from the interim to the final version of the app were shown and this adaptation of design was crucial to the success of the app.

4. Experiment Development

4.1. Introduction

In this section, there will be a demonstration of the working version of pages that have been created for the front and back end of the application as well as all of the inner workings of the middle tier. The middle tier takes care of all the complex workings of the system as well as helping the front end communicate with the back end for data retrieval from the database. There is a breakdown of the front-end pages that are shown to the user and why they were chosen, code samples of the middle tier that show how the front end is created as well as code samples on how the front end communicates with the database in the back-end. Finally, there will be some images of the database/back-end itself along with a description of its purpose and details on why everything is structured the way it is. The section will cover the key development process and discuss the challenges encountered during the creation of the Primary Mathletes application.

4.2. Front-End

The original prototypes that were created for the front-end of the application were very minimalistic. They were created with the barebones, default appearances of the Kivy library, purely because of time restraints leading up to the interim of the project. There was a big revamp in the visual design of the application towards the end of the project that was discussed in section 3. For the development of small prototypes before the Winter break, there was a need to first research and practice using the library before trying to make anything look, sound and feel fancy.

Title Page

The title page is the page that the user is brought to when they open the app for the first time. It allows the user to decide whether they would like to register if they have never used the application before or if they want to log in if they already have an account. This first page was created for the register and login functions, but it was also created to test the ability to create buttons as well as swapping between screens which is a big part of the Kivy library. The title page seen in figure 64 is a far cry from what is found in the final version of the application but it was the first screen developed using Kivy and was a necessary prototype to ensure that development using Kivy could continue as the project progressed.

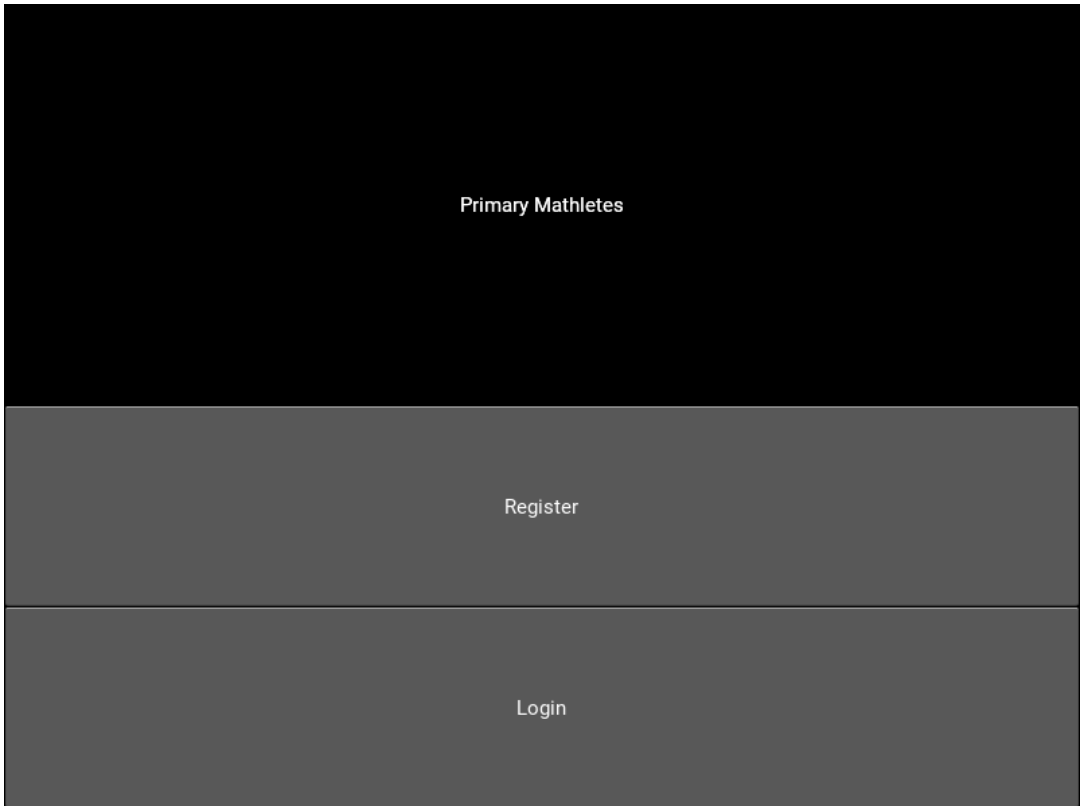


Figure 64 Title Page of the app

Register Page

The register page was then created to try out the screen swapping capabilities of the Kivy library. It allows the user to press a button and be brought to the login screen. The register page was also created to try out the text input option that Kivy offers. Some labels were created to let the user know what information was being asked for and then text boxes were included underneath them for the user to fill out. Once the information was all there, the user can press the register button. The middle tier will then check the database to see if another user has already used this information to create an account. (This will be discussed in more detail in section 4)

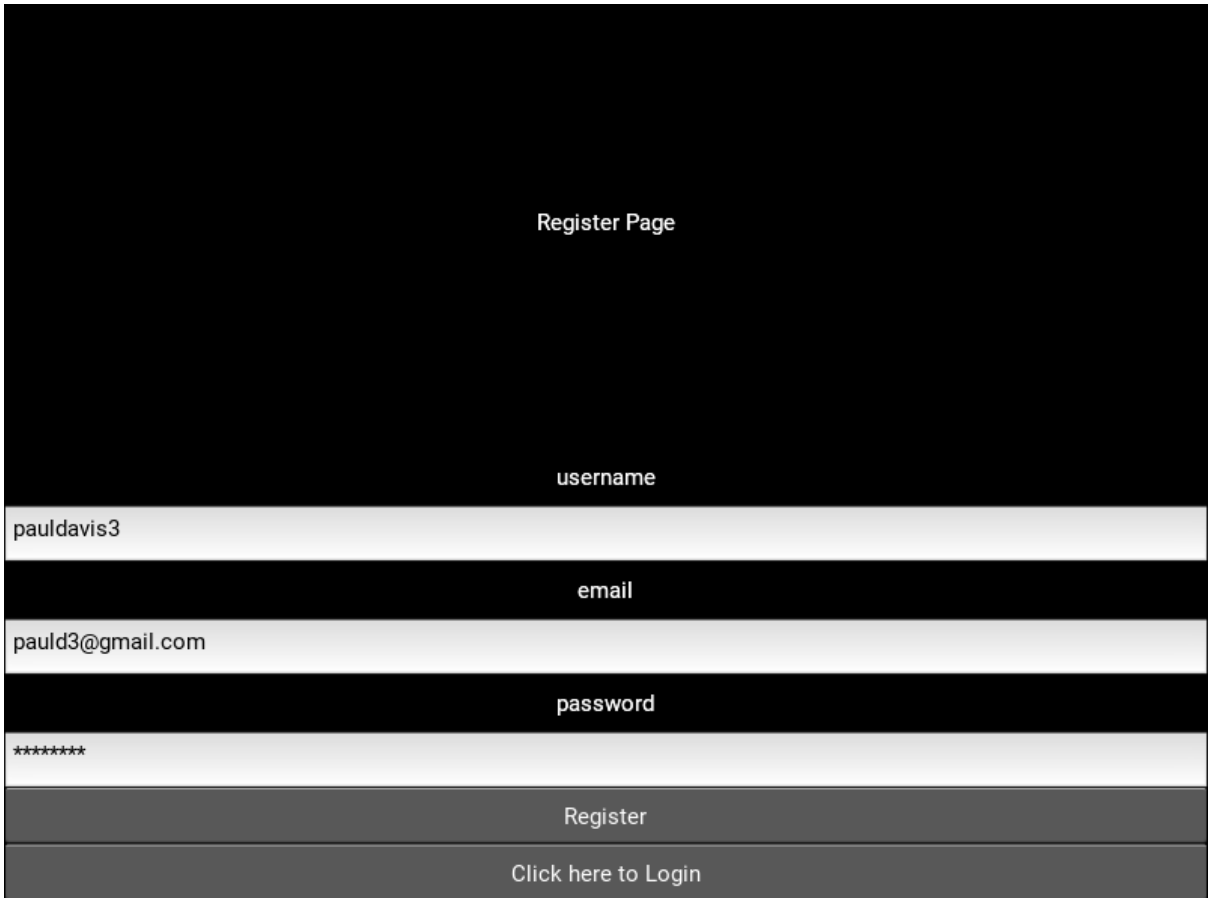


Figure 65 Register Page of the app

Login Page

Then the login page was created. It was very similar to the register page except that there is no text box for the email as this information should already be in the database. The user can enter their details on this page and then press the log in button the be brought to the main page of the app provided that the details they entered match a pair found in the database, otherwise they will stay on the same page.

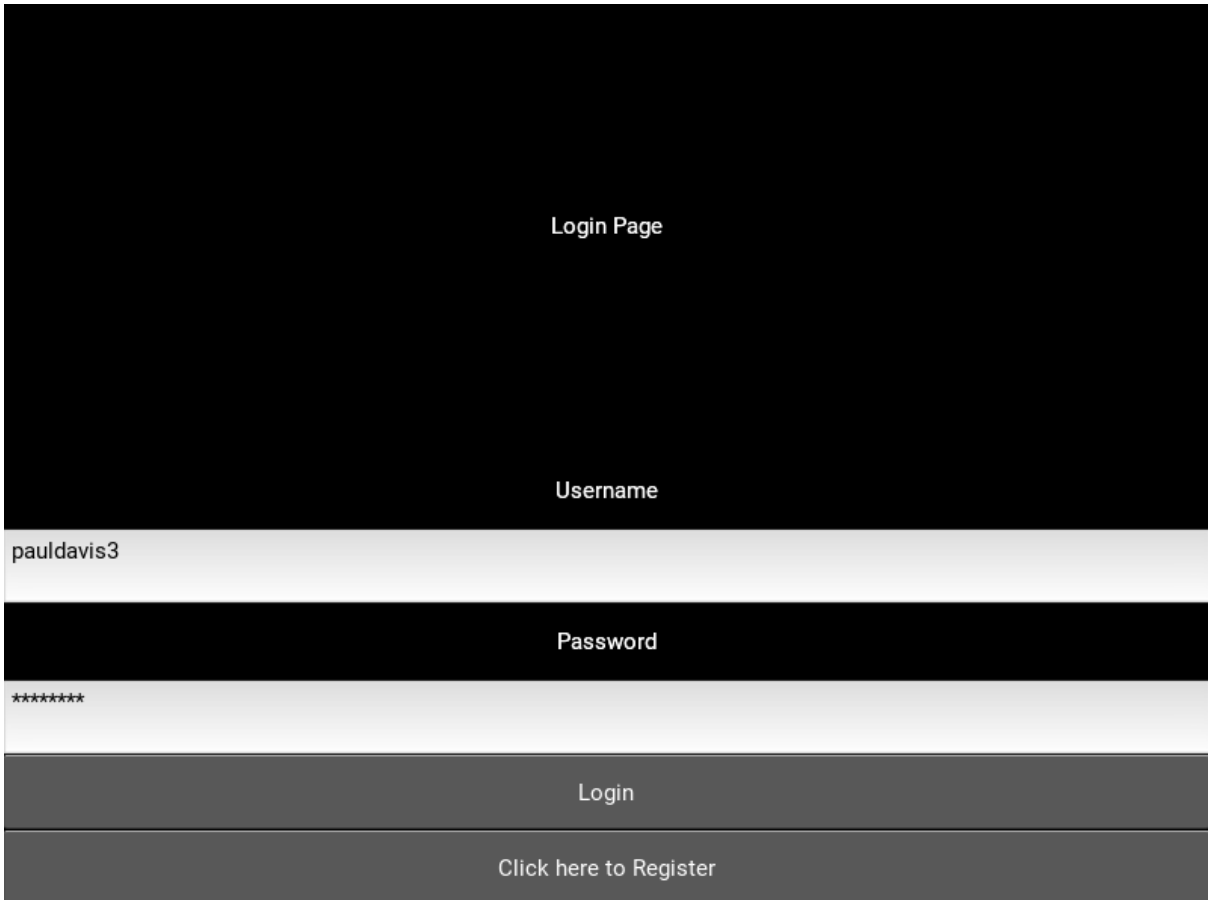


Figure 66 Login Page of the app

Kivy File:

The Kivy or kv file is a separate file to the Python file that holds most of the GUI elements for the app. It is usually used because it is very powerful with minimal amounts of code if you separate the files rather than use the Kivy UI elements in the Python file along with all of the functions. It is possible and sometimes necessary to create Kivy elements in the Python file but where possible it should all be kept in the kv file for readability and organization. All of the pages seen in the final version of the application have some link to the kv file. The following are all of the different aspects of the kv file:

Screen Manager:

This part of the kv file is necessary for all of the screens of the app to link to each other. Figure 99 shows that it is just a list of every available screen.

```
ScreenManagement:
    TitlePage:
    MainPage:
    LoginPage:
    RegisterPage:
    StudentClassroomPage:
    TeacherClassroomPage:
    StudentProfilePage:
    TeacherProfilePage:
    StudentProgressPage:
    PlayPage:
    AdditionPage:
    SubtractionPage:
    MultiplicationPage:
    DivisionPage:
    MinigamePage:
    ResultsPage:
```

Figure 67 Screen Manager Code Snippet in Kivy File

Basic Screen Structure:

Figure 100 shows the basic screen structure found in the Kivy file. At the top, there is the screen name that was listed in the screen manager list and below this is the name it is given so that it can be referenced by other screens if necessary. Then following after the name of the screen there can be any number of elements listed to display different kinds of things on the screen and these will be discussed further on in this section.

```
<TitlePage>:
    name: "title"
    Image:
        size_hint: 0.6, 0.6
        pos_hint: {"center_x": 0.5, 'center_y':0.675}
        source: "Images/logo.png"
        y: self.parent.y
        x: self.parent.x
        size: self.parent.size
    Button:
        size_hint: 0.6, 0.2
        pos_hint: {"center_x": 0.5, 'center_y':0.3}
        background_normal: "Images/button.png"
        background_down: "Images/buttondown.png"
```

Figure 68 Basic Screen Structure in Kivy File

Images:

Images as attributes are only used as a handful of times in the application but figure 101 shows one example of this on the title page to show the logo of the project.
In this particular example the image is given a size of (0.6, 0.6) in other languages this might make the image very small but in the Kivy language, all sizing of elements fall between 0 and 1 and these numbers map between 0 and the full length of the width and height of the screen. So in this example, the image is going to take up 60% of the screen according to the size hint and a demonstration of this adaptiveness can be seen in figures 102 and 103.
The position hint works similarly to the size hint where you give a number between 0 and 1 and it works as a percentage of the screen size. This example says along the center X-axis go 50% across which is half-way across the screen and the other is 67.5% up on the Y-axis so the logo sits slightly above the middle of the screen.
The source is exactly what it says and is the link to the source file of the image which in this case is kept in the Images sub-directory and is called 'logo.png'
The x, y and size values just say to take up exactly the amount of space that was set out in the size hint attribute.

Images can also be used in place of the Kivy default background for buttons and this is what they were mainly used for in this project.

```
Image:
    size_hint: 0.6, 0.6
    pos_hint: {"center_x": 0.5, 'center_y':0.675}
    source: "Images/logo.png"
    y: self.parent.y
    x: self.parent.x
    size: self.parent.size
```

Figure 69 Image Code Snippet in Kivy File

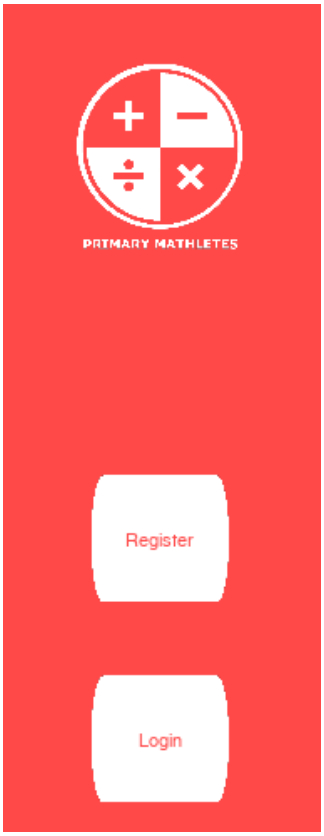


Figure 70 Logo Adapting 1

Figure 71 Logo Adapting 2

Buttons:

Buttons can be given ID's so that they can be referenced in the Python code. They also normally have text which can be sized using font size, styled with font name and coloured with colour. The buttons also have the same size and position hints that images do.
In addition to these attributes buttons also have some unique ones like the on_release attribute which

can be used to fire a function whenever the button is pressed, in the example shown in figure 104, the on_release attribute tells the app to change the current slide transition to left. This determines what direction the app will swipe to get to the next screen. As well as changing the slide transition, this on_release attribute tells the app to go to the root of the button which in this case happens to be the 'divide' page and run the 'playGame' function, passing in a parameter of 2.

```
Button:
    id: divideButton2
    text: "Level 2"
    size_hint: 0.45, 0.15
    pos_hint: {'center_x': 0.25, 'center_y': 0.675}
    on_release:
        app.root.transition = SlideTransition(direction="left")
        root.playGame(2)
    background_normal: "Images/button.png"
    background_down: "Images/buttondown.png"
    background_disabled_normal: "Images/buttonlocked.png"
    font_size: self.width*0.1
    font_name: 'Helvetica'
    color: (1, 72/255, 72/255, 1)
    disabled: True if root.playerXP < 100 else False
```

Figure 72 Button Code Snippet in Kivy File

Buttons can also take three variations of the same attribute which are: background_normal, background_down and background_disabled. These attributes take a source file in the same way the Image entities do and change depending on what state the button is currently in. Figure 105 shows the difference between the button states that can be seen all over the application. Finally, buttons also have a disabled attribute which is a Boolean that can also take inline if statements. The example here sets the disabled to True or False depending on the players number of experience points.

Figure 73 Locked, Normal and Disabled Button Example

Labels:

Anywhere that has text on the application that isn't a button, or an image is using a label. These are used throughout the app to give the users information and guide them to use the app correctly. Figure 106 shows that labels share many attributes with the other entities such as size and position hint, as well as font size and font name.

```
Label:
    id: studentClassroomName
    pos_hint: {"center_x": 0.5, 'center_y':0.85}
    font_size: self.width*0.05
    font_name: 'Helvetica'
```

Figure 74 Label Code Snippet in Kivy File

The label can either be given a text attribute or in this case, the label is given an ID so that the label can be updated dynamically in the python code. An example of this can be seen in figure 107 where depending on the value of 'self.classroom', 'self.ids.studentClassroomJoin.text' will be set to two separate strings of text depending on what is needed from the Label. Changing text like this dynamically saves having to enable and disable two different labels where one will suffice.

```
if self.classroom == "no classroom":
    self.ids.studentClassroomName.text = "No Classroom"
    self.ids.studentLeaveClassroom.disabled = True
    self.ids.studentJoinClassroom.disabled = False
    self.ids.studentClassroomJoin.text = "Enter the name of the classroom you'd like to join"
else:
    self.ids.studentClassroomName.text = "Current Classroom: " + self.classroom
    self.ids.studentLeaveClassroom.disabled = False
    self.ids.studentJoinClassroom.disabled = True
    self.ids.studentClassroomJoin.text = "Press the button below to leave your current classroom"
```

Figure 75 Dynamic Update of Label Text

Canvas:

The canvas is another Kivy entity that allows the drawing of shapes on the screens found within the app. For the Primary Mathletes application, the Line was the only shape utilized for styling and segregating some text as seen in figures x and x but there are also other shapes like rectangles, circles, and triangles available if needed.

```
canvas:
    Color:
        rgba: 1, 1, 1, 1
    Line:
        width: 2
        points: (0, self.height*0.75, self.width, self.height*0.75)
    Line:
        width: 2
        points: (0, self.height*0.575, self.width, self.height*0.575)
    Line:
        width: 2
        points: (0, self.height*0.425, self.width, self.height*0.425)
```

Figure 76 Canvas Code Snippet in Kivy File

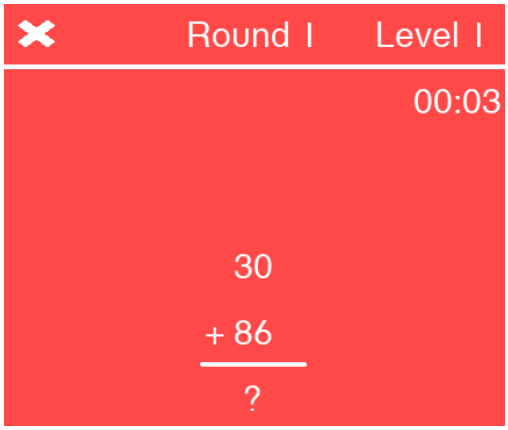


Figure 77 Canvas Line Styling 1

Best Score	0	1	1
Correct Answers	10	20	100.0%
Time Played	0	0	0.0%
Total Games Played	0	0	0.0%
Total XP	0	0	0.0%

Figure 78 Canvas Line Styling 2

4.3. Middle-Tier

Python File:

Inside the main python file, there are a few pieces that are crucial to making the app run. The Kivy library, along with its screen manager sub-library, is responsible for most of the interactive features and functions from the front-end like buttons, labels, text input boxes, etc. as well as allowing the user to traverse between screens, like going from the title page to the login page and from the login page to the main page.

```
1  from kivy.app import App
2  from kivy.uix.screenmanager import ScreenManager, Screen
3  from kivy.lang.builder import Builder
```

Figure 79 Snippet of code showing Kivy libraries

Firebase:

Another import library is the firebase library, this allows for items in the database to be created, read, updated and deleted (C.R.U.D). In the initial prototype of the app, only two firebase functions were being used: post and get but later in development, put and delete functions were also utilized to get the most out use out of the database. GET requests to the database are used twice within the python file: once to ensure that the user has not entered information that has already been used to register their account and then again when the user is logging in to ensure that their username and password pair matches in the database. Figure 68 shows a code snippet showing the importing of the firebase library and the function used to connect to the projects Realtime firebase database.

```
4  from firebase import firebase
5
6  firebase = firebase.FirebaseApplication('https://c16324311fyp.firebaseio.com/')
```

Figure 80 Snippet of code showing firebase library being used

All Firebase requests are written in JSON and the figures below show examples of how the firebase library is used to:

- Create information in the database:

The firebase POST method takes two parameters: the table to go to and the information to put there. POST method creates a random string that can be used to identify individual users in the database. An example of this can be seen in figure 69.

```
firebase.post('/users',
              {'username': uname, 'email': email, 'password': pword, 'teacher': "yes",
               'classroom': "no classroom", 'add': 0, 'subtract': 0, 'multiply': 0, 'divide': 0})
```

Figure 81 Firebase Post Example

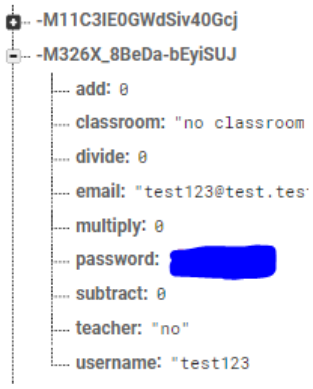


Figure 82 Example of random strings created by Firebase POST method

- Reading the information in the database:

The firebase GET method retrieves information from the database. It takes two parameters: the table to search for and what information to retrieve from that table. The GET method is mostly used to get a list of users from the database like in the snippet shown in figure 71. If the second parameter is set to 'None' like in the snippet below, the GET method will return the entire list of users from the database as well as all of their attributes.

```
results = firebase.get('/users/', None)
```

Figure 83 Firebase GET Example

If a second parameter is given, the function will only return whatever was asked for. Figure 72 shows an example of this where a query is constructed out of concatenated strings and a user's current week is retrieved from the database and saved into the 'check' variable.

```
check = firebase.get('/users/' + index + '/progress/' + str(currentYear) + '/', str(currentWeek))
```

Figure 84 Firebase GET Example with Second Parameter

- Update information in the database:

The firebase PUT method is similar to the POST method mentioned above except that it takes three parameters instead of two. Instead of creating a random string like the POST method, the PUT method takes a second parameter and puts the third parameter (the information) into this part of the database. Using PUT either creates the second parameter in the database if it doesn't exist or else it updates the information to whatever the third parameter is if the information already existed. An example of this is shown in figure 74 where the green highlight shows new information being created in the database and orange highlights mean that a field is being updated.

```
firebase.put('/users/' + index, 'classroom', 'no classroom')
```

Figure 85 Firebase PUT Example

Figure 86 PUT Method Creating and Updating Information

- Delete information from the database:

The firebase DELETE method does as it says and deletes unwanted information from the database. It takes two parameters, the table to search and the item to delete from the table. The example shown in figure 75 shows the database deleting the user 'index' from the users' table.

```
firebase.delete('/users/', index)
```

Figure 87 DELETE Method Example

Label Base:

This library allowed me to import the Helvetica Textbook Font that is used throughout the entire application in place of the default Kivy label that proved unsuitable for younger children as discussed in section 3. Figure 76 shows a code snippet of the library being imported and then the code that reads the font from the Fonts sub-directory of the project folder where the main.py file is found.

```
from kivy.core.text import LabelBase

LabelBase.register(name="Helvetica",
                   fn_regular="Fonts/HelveticaTextbookLTRoman.ttf")
```

Figure 88 Label Base Code Snippet

Sound Loader:

Similar to the Label Base library, Sound Loader allowed me to import sounds into the application for use in the minigame screen of the application. Figure 77 shows a code snippet of the library being imported and then the code that reads in the sounds from the Sounds sub-directory of the project folder where the main.py file is found.

```
from kivy.core.audio import SoundLoader

self.correctSound = SoundLoader.load("Sounds/correct.wav")
self.incorrectSound = SoundLoader.load("Sounds/incorrect.wav")
```

Figure 89 Sound Loader Code Snippet

Random:

The random library is a simple library that allows for random numbers to be generated within a range that I utilized in the minigame section of the app. Figure 78 shows the random library in use in the main and algorithm Python files to get a random number in different ranges depending on what they're needed for. Two examples of random use can be found on the minigame screen. The first random thing is the selection of which potential answer is going to be the expected answer to the question. The second potential answer in this example is chosen. The second example is choosing what values the other potential answers will have since they are not the expected answer and are wrong on purpose to throw off the user from guessing the answer to the question.

```
import random

randomNum = random.randrange(0, 4)

  def randomInRange(min, max):
      return random.randrange(min, max + 1)
```

Figure 90 Random Library Code Snippet

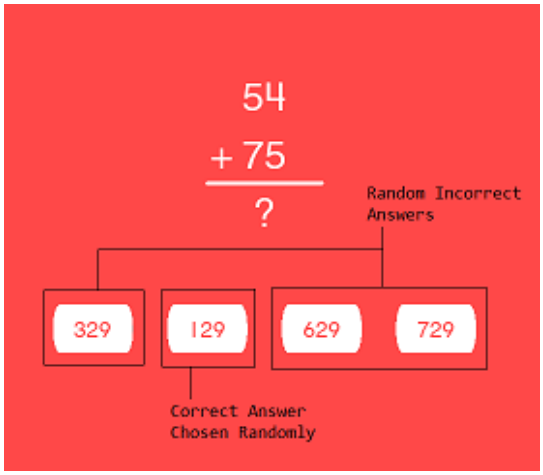


Figure 91 Random Use in Minigame

SMTP Lib:

This library is used to set up a Simple Mail Transfer Protocol link to an email address that I set up purely for this project. It retrieves users' weekly progress and sends a report to the email address that they used to set register their account with.
Near the top of the function seen in figure 80, the subject and the body of the email is constructed by concatenating information from the user's progress table in the database. In the try section, an attempt is made to set up an SMTP link to Googles server address. After it connects successfully it logs into the Primary Mathletes email address with the login credentials provided from a separate Python file called 'emailConfig'. Then the message is constructed, the mail is sent and the SMTP link is killed. If none of this fails the mail is sent successfully, otherwise, the catch code is run, an error message is thrown, and the email will not be sent and received. Figure 81 shows an example of an email received using this function.

```
def sendEmail(self, username, email, password, userEmail, thisWeek):

    subject = "Week " + globalVariables.weekNumber + " Report for " + username

    msg = "The following is a report for the week " + str(
        globalVariables.weekNumber) + ' for ' + username + ':\n\nBest Score: ' + \
          str(thisWeek['bestScore']) + '\n\nCorrect Answers: ' + str(
        thisWeek['correctAnswers']) + '\n\nTime Played: ' + \
          str(thisWeek['timePlayed']) + '\n\nTotal Games Played: ' + str(
        thisWeek['totalGamesPlayed']) + '\n\nTotal XP: ' + \
          str(thisWeek['totalXP']) + '\n\nTHIS IS AN AUTOMATED MESSAGE\n\nPRIMARY MATHLETES'

    try:
        server = smtplib.SMTP('smtp.gmail.com:587')
        server.ehlo()
        server.starttls()
        server.login(email, password)
        message = 'Subject: {}\n\n{}'.format(subject, msg)
        server.sendmail(email, userEmail, message)
        server.quit()
        print("Email sent successfully.")
        # Show pop up success
    except KeyError:
        print("Email failed to send.")
        # Show pop up failed
```

Figure 92 Send Mail Function Code Snippet

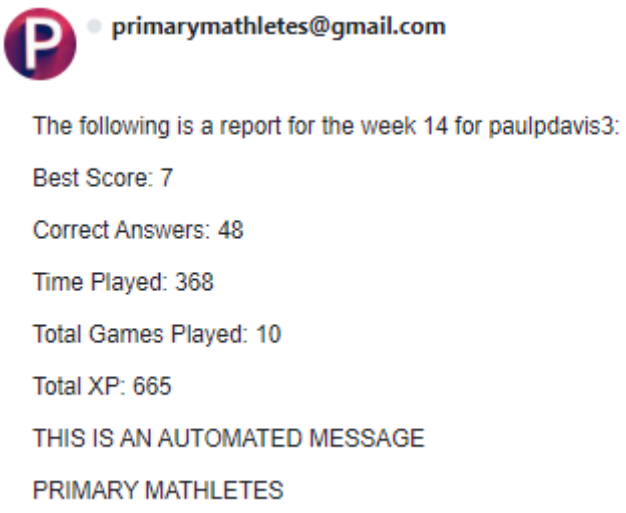


Figure 93 Email received from the Send Email Function

Date Time:

This library is used to get the current Year and week for use within the app. The current day is also gotten using this library that was used for testing purposes, but it is not necessary for the final use of the library. Figure 82 shows the DateTime and isocalender functions being used to format the current

date into human-readable integers. The for-loop being used will be discussed further in this section but basically what it is doing is ensuring the user is only altering their own information in the database. Once a user logs in, the system checks the current week and if the user has any progress for this week already stored in the database. If they do, nothing happens but if the user doesn't yet have any progress for the current week, a new progress entity is created for the user in the database and all of the values are set to 0.

```
def checkWeek(self):
    currentDate = datetime.date.today()
    currentYear, currentWeek, currentDay = currentDate.isocalendar()

    results = firebase.get('/users/', None)

    for index in results:
        if results[index]['username'] == ScreenManagement.store.get('credentials')['username']:

            check = firebase.get('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek), None)

            if check:
                pass
                # print(check)
            else:
                print('Didnt find the current week in the DB')
                firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                             'bestScore', 0)
                firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                             'totalGamesPlayed', 0)
                firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                             'timePlayed', 0)
                firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                             'correctAnswers', 0)
                firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                             'totalXP', 0)
```

Figure 94 Datetime Code Snippet

Minigame – How it works:

When the user clicks on an operator (function 83) and then a level number (figure 84), these values are sent to a separate algorithm file where they are taken and used to set up the minigame that the user is about to see.

Figure 95 Clicking On An Operator

Figure 96 Clicking On A Level

The level number will decide what the difficulty will be for the minigame that is about to begin. Some standard checks are done to ensure each minigame is fair. The difficulty cannot go above 10 or below 1 for starters as this would throw off the whole level selection. As the user answers questions the current difficulty can fluctuate between a single level above and a single level below so that if the user is doing well in one minigame the questions will become slightly harder and if the student is performing poorly in one minigame then the difficulty will be dropped slightly to make it easier for

them. This also cannot go below a difficulty of 1 or above 10 so if the user is on Level 1 the equations can only get harder and if the user is on Level 10 then they can only get easier.

```
# difficulty can't go below starting range - 1
if difficulty < startingRange - 1:
    difficulty = startingRange - 1
# difficulty can't go above starting range + 1
if difficulty > startingRange + 1:
    difficulty = startingRange + 1
# difficulty can't go below 1
if difficulty < 1:
    difficulty = 1
# difficulty can't go above 10
if difficulty > 10:
    difficulty = 10
```

Figure 97 Algorithm File Difficulty Setup

Depending on what operator was chosen, the range of numbers that can be randomly selected varies. This is because, by the third class, students will be more familiar with subtraction over multiplication and division and they will be more familiar again with addition than all the rest. Third class students are expected to be able to perform arithmetic with addition on numbers from 1-1000, subtraction from 1-500, and multiplication and division up to 30. Figure 86 shows the minimum and maximum range being capped on different numbers depending on what difficulty was chosen.

```
if operation == "add":
    numOfVariables = 3

    rangeMin = difficulty * 9
    rangeMax = difficulty * 99

elif operation == "subtract":
    numOfVariables = 3

    rangeMin = difficulty * 5
    rangeMax = difficulty * 55

elif operation == "divide":
    numOfVariables = 3

    rangeMin = 1
    rangeMax = difficulty * 3

elif operation == "multiply":
    numOfVariables = 3

    rangeMin = 1
    rangeMax = difficulty * 3
```

Figure 98 Setting Maximum and Minimum Ranges

Next, a while loop runs to ensure that the answer to the question does not exceed the maximum number that students are expected to be able to perform. Random numbers within the minimum and maximum range determined by the level number are chosen and saved into global variables, along with the answer to be used on the minigame screen when the user is brought there.

```
# verify if the numbers will work in the equation
possibleAnswer = False

while not possibleAnswer:
    if operation == "add":
        if numOfVariables == 4:
            globalVariables.x = randomInRange(rangeMin, rangeMax)
            globalVariables.y = randomInRange(rangeMin, rangeMax)
            globalVariables.z = randomInRange(rangeMin, rangeMax)
            globalVariables.answer = globalVariables.x + globalVariables.y + globalVariables.z
        else:
            globalVariables.x = randomInRange(rangeMin, rangeMax)
            globalVariables.y = randomInRange(rangeMin, rangeMax)
            globalVariables.answer = globalVariables.x + globalVariables.y

        if globalVariables.answer < 1000:
            possibleAnswer = True
```

Figure 99 Ensure Answer Doesn't Exceed 1000

Once a suitable answer has been chosen, another random number is retrieved and this is saved to a variable called 'blank'. This random number can be 0,1 or 2 and it decides which number in the equation will be blanked out for the user to guess in the minigame. This blank element of the minigame keeps the users on their toes and steers clear of the monotonous questions where the answer is always a straightforward equation.

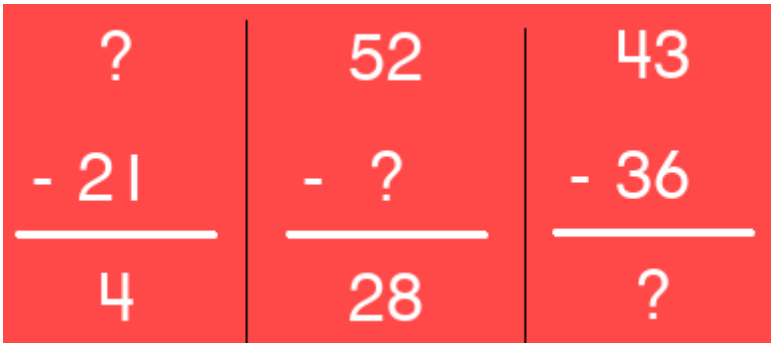


Figure 100 Examples of 'blank' Being 0, 1 and 2

Once the users enter the minigame page, after all of the pregame information has been gathered, all of the variables used in the minigame are initialized and set up for the start of the game. A timer is started and the reinitialize function runs.

```
def on_enter(self, *args):
    globalVariables.correctAnswers = 0
    globalVariables.incorrectAnswers = 0
    globalVariables.roundNumber = 1
    globalVariables.seconds = 0
    globalVariables.minutes = 0
    self.reinitialize()
    Clock.schedule_interval(self.incrementTimer, .1)
    self.incrementTimer(0)
    self.startTimer()
```

Figure 101 Minigame Initialization

At the beginning of every minigame and after a question is answered the reinitialize function runs. It checks if the total number of questions has been answered and if so, stops the timer and moves the user onto the results page. If the total number of questions hasn't been met, the algorithm function in the algorithm file gets run again to retrieve new numbers for the next round of questions and the update text function runs.

```
def reinitialize(self):
    if globalVariables.roundNumber > 10:
        self.stopTimer()
        self.manager.current = 'results'
    else:
        algo.algo(globalVariables.operation, globalVariables.level)
        self.updateText()
```

Figure 102 Reinitialize Function

The update text function prints all of the necessary information to the screen on a round-by-round basis for the minigame such as the operator, round number, timer, level number, and all of the potential answers. Another random number is generated to decide which of the four random numbers

will be the expected answer and which potential answers will be random incorrect answers. This selection can be seen in figure 91.

```
randomNum = random.randrange(0, 4)

if randomNum == 0:
    self.potentialAnswer1 = str(self.expectedAnswer)
    self.potentialAnswer2 = str(self.getPotentialAnswer(self.expectedAnswer))
    self.potentialAnswer3 = str(self.getPotentialAnswer(self.expectedAnswer))
    self.potentialAnswer4 = str(self.getPotentialAnswer(self.expectedAnswer))
elif randomNum == 1:
    self.potentialAnswer1 = str(self.getPotentialAnswer(self.expectedAnswer))
    self.potentialAnswer2 = str(self.expectedAnswer)
    self.potentialAnswer3 = str(self.getPotentialAnswer(self.expectedAnswer))
    self.potentialAnswer4 = str(self.getPotentialAnswer(self.expectedAnswer))
elif randomNum == 2:
    self.potentialAnswer1 = str(self.getPotentialAnswer(self.expectedAnswer))
    self.potentialAnswer2 = str(self.getPotentialAnswer(self.expectedAnswer))
    self.potentialAnswer3 = str(self.expectedAnswer)
    self.potentialAnswer4 = str(self.getPotentialAnswer(self.expectedAnswer))
elif randomNum == 3:
    self.potentialAnswer1 = str(self.getPotentialAnswer(self.expectedAnswer))
    self.potentialAnswer2 = str(self.getPotentialAnswer(self.expectedAnswer))
    self.potentialAnswer3 = str(self.getPotentialAnswer(self.expectedAnswer))
    self.potentialAnswer4 = str(self.expectedAnswer)
```

Figure 103 Update Function Code Snippet

After the user clicks on one of the potential answers, the check answer function runs. If the user clicks the correct potential answer a sound plays to indicate that they guessed the correct answer, a popup is displayed with a smiley face to visually show the user that was the correct answer and the variable that holds the number of correct answers is incremented. If the user guesses incorrectly a different sound goes off indicating an incorrect answer and a sad face is shown in a popup to visually tell the user they got the answer wrong. The incorrect answer variable is incremented in this case. No matter what answer the user chooses, the round number is incremented and the reinitialize function runs again after this check.

```
def checkAnswer(self, answer):
    answer = float(answer)

    if answer == self.expectedAnswer:
        self.correctSound.play()
        print("correct answer")
        globalVariables.correctAnswers += 1
        popups.CorrectPopup()
    else:
        self.incorrectSound.play()
        print("incorrect answer")
        globalVariables.incorrectAnswers += 1
        popups.IncorrectPopup()

    globalVariables.roundNumber += 1

    self.reinitialize()
```

Figure 104 Check Answer Function

The 'getPotentialAnswer' function is the most complex part of this minigame section of the code. This part was the most challenging for me and took probably the most amount of time to get right. If the function didn't work properly there wouldn't be any challenge to the minigames, and every question would be very easy for the user to answer which negates the whole point of the minigames.

The function starts by reading in the expected answer that was retrieved from the algorithm file. Then it sets up the function by creating variables for the potential answer, the length of the expected answer and creates a check to make sure the potential answer is suitable for the minigame.

```
def getPotentialAnswer(self, expectedAnswer):
    expectedAnswerLength = len(str(expectedAnswer))
    potentialAnswer = expectedAnswer
    potentialAnswerCheck = 0
```

Figure 105 Get Potential Answers Code Snippet

After these are set up a variable called 'DifficultyMapped' is set up. It is calculated based on the level number that the user chose before entering the minigame. Essentially, what this function does is flip the numbers 1-10 backward so that they read 10-1. Instead of 1 getting mapped to 10, it instead starts at 9. This is because if a number were to move 10 places away from itself in a single-digit it would loop back around to itself which isn't useful for the way this function works. E.g If the 1 in 61 was mapped 10 places either way from itself the potential answers would be 51 and 71. Since I am only interested in the final digit at the moment 10 is useless so instead 1 is mapped to 9 and both 9 and 10 are mapped to 1 on the other end of the range.

```
difficultyMapped = int((((globalVariables.level - 1) * (1 - 9)) / (10 - 1)) + 9)
```

Figure 106 Difficulty Mapped Calculation

Level	1	2	3	4	5	6	7	8	9	10
Difficulty Mapped	9	8	7	6	5	4	3	2	1	1

Figure 107 Difficulty Mapped Numbers 1-10

This difficulty mapped variable is then used to decide how random the potential answers will be. Looking at figures 96 and 97 we can see that the lower the level number, the more random the potential answers are and so, it's a lot easier for the user to guess which number is correct if they aren't certain of an answer whereas in later levels, the potential answers are quite similar to the expected answer and so this adds another layer of difficulty to the questions the users are being asked.

The example below shows that some thought needs to go into figuring out which potential answer is the correct one.

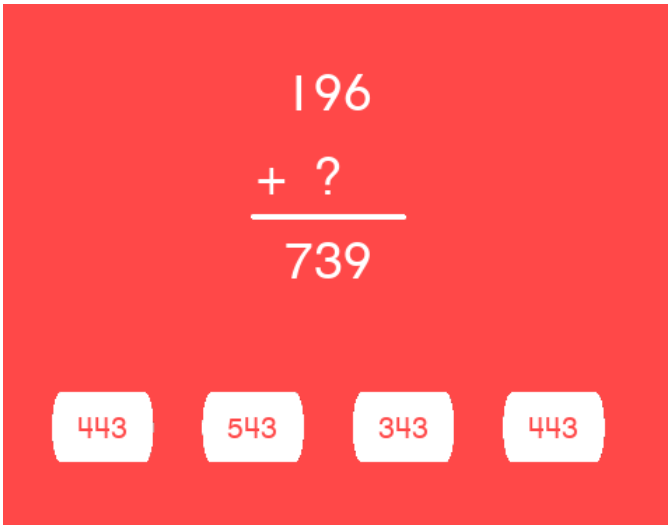


Figure 108 Level 8 Addition Potential Answer Example

This example shows that earlier levels have extremely random potential answers and the final digit of the incorrect potential answers do not match the correct answer so looking at the final digit of the equation pretty much gives away the answer.

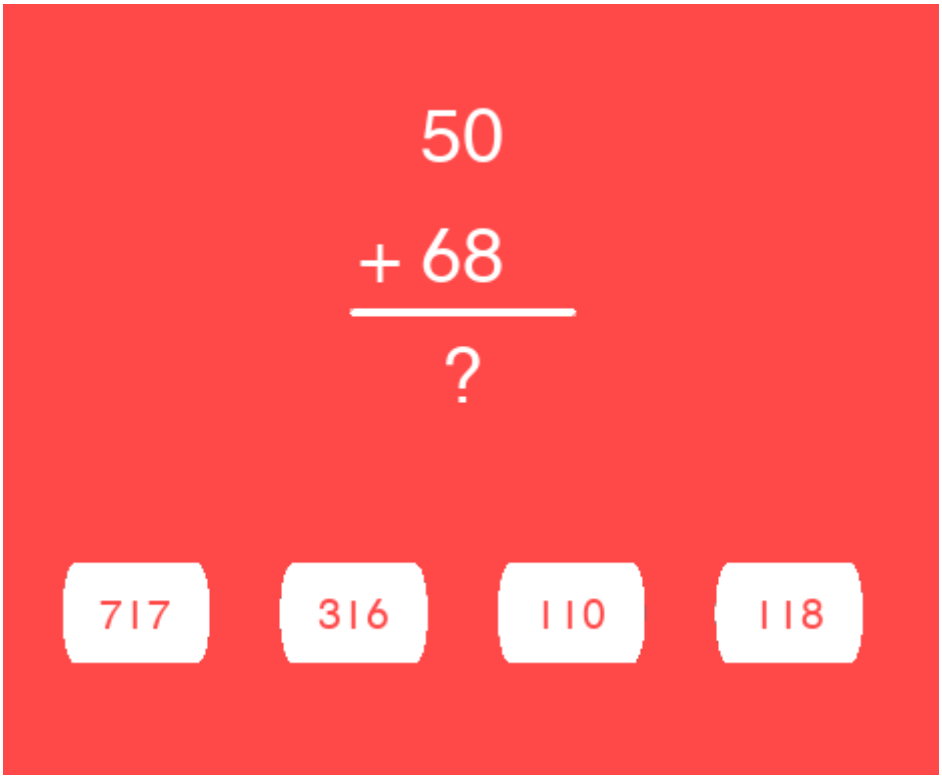


Figure 109 Level 2 Addition Potential Answer Example

Depending on how many digits there are in the expected answer will determine how much calculating will need to be done for the potential answers. The potential answer check mentioned earlier is used to ensure that the potential answer doesn't accidentally spit out an identical number to the expected answer, as this would then increase the chances of the user selecting a random number and getting the answer correct for that question. This is achieved by looping through potential answers and ensuring they don't match the expected answer. If it does match, a new number is continuously generated until it no longer matches and this number is then returned from the function to be used in the minigame.

```
if expectedAnswerLength == 1:
    while potentialAnswerCheck == 0:
        potentialAnswer = expectedAnswer + random.randrange(-difficultyMapped, difficultyMapped)
        if potentialAnswer != expectedAnswer and potentialAnswer > 0:
            potentialAnswerCheck = 1
elif expectedAnswerLength == 2:
    expectedAnswerSingles = expectedAnswer % 10
    expectedAnswerTens = int(str(expectedAnswer % 100)[0])
    while potentialAnswerCheck == 0:
        potentialAnswer = int(
            str(expectedAnswerTens + random.randrange(-difficultyMapped, difficultyMapped)) + str(
                expectedAnswerSingles))
        if potentialAnswer != expectedAnswer and potentialAnswer >= 0:
            potentialAnswerCheck = 1
elif expectedAnswerLength == 3:
    expectedAnswerSingles = int(expectedAnswer % 10)
    expectedAnswerTens = int(str(expectedAnswer % 100)[0])
    expectedAnswerHundreds = int(str(expectedAnswer)[0])
    while potentialAnswerCheck == 0:
        potentialAnswer = int(
            str(int(expectedAnswerHundreds) + random.randrange(-difficultyMapped, difficultyMapped)) + str(
                expectedAnswerTens) + str(int(expectedAnswerSingles) + random.randrange(-difficultyMapped, difficultyMapped)))
        if potentialAnswer != expectedAnswer and 0 <= potentialAnswer < 1000:
            potentialAnswerCheck = 1
```

Figure 110 Potential Answer Code Snippet

4.4. Back-End

For the original prototype of the back-end Firebase database, there was only one table, the user table. It was set out that two additional tables would be created: a teachers table and a classroom table. Eventually, through development, it was decided to scrap the teacher table as teachers and students share all of their attributes just with different values for each. The database prototype can be seen in figure 111. Each user in this prototype only had a username, password and email address.

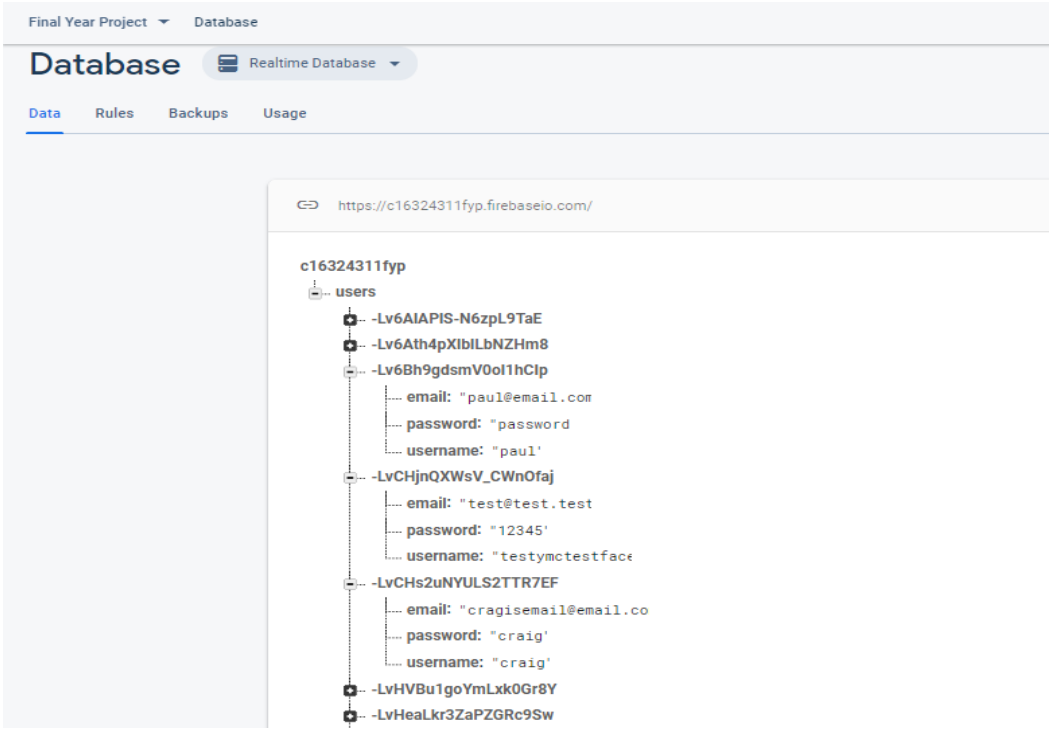


Figure 111 Old Database Prototype

Throughout development, more and more attributes and tables were added and removed from the database but the final version can be seen in figure 112. The final version of the database shows that there is now a Classrooms table that holds all of the classrooms that students can join and teachers have the ability to create. The classrooms currently only need to have one attribute which is their name as all the other relevant information about users is stored in their own table. The Users table now holds both students and teachers. Users have an add, subtract, multiply and divide attribute which keeps track of the overall number of experience points the users have for each of these operators. As well as this, users also have the original username, password, and email attributes. The teacher attribute is what differentiates students from teachers. If the user is a teacher then it will say yes, if they're a student it will say no. Finally, each user has the potential to have a progress attribute which can potentially hold weekly information about every week for every year if necessary. Each year can hold 52 weeks and each week holds the same information about students on a week-by-week basis. This keeps track of:

- The user's best score for that week whether that be zero, ten or anything in between.

- The total number of correct answers the user got that week.

- The amount of time in seconds that the user spent playing that week.

- The total number of games they played that week.

- Total experience points gained that week.

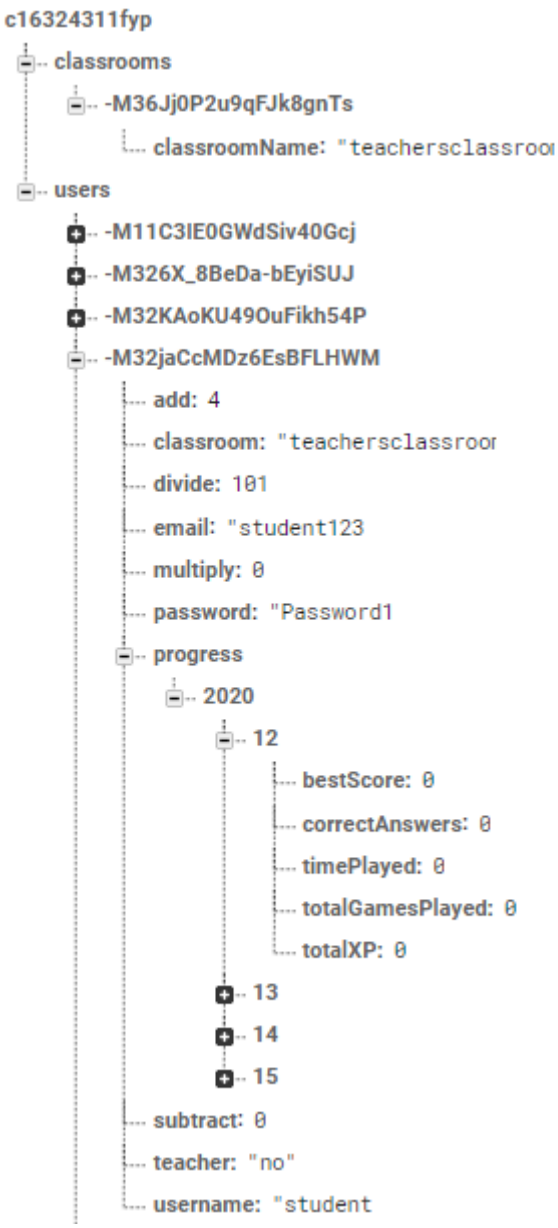


Figure 112 Final Version of the Database

Registering an Account:

When a user is registering an account the first thing that gets checked is if their password is suitable. Going with password standards the password must contain at least one number, an uppercase letter, a lower case letter and it must be at least 8 characters long. This check is achieved using Pythons 'any' function which returns true if it finds something inside that it is looking for like .isupper(), .islower(), isnumeric(), etc. Then the if statement seen in figure 113 checks if any of these are false, if they are a popup is shown to the user telling them they need to have the specific requirements to pass the password check.

```
class RegisterPage(Screen):
    def checkPassword(self, uname, email, pword, isTeacher):
        capital = any(x.isupper() for x in pword)
        lowercase = any(x.islower() for x in pword)
        number = any(x.isnumeric() for x in pword)
        length = len(pword)

        if length < 8 or number is False or lowercase is False or capital is False:
            print(
                'Make sure that your password is at least 8 characters long and contains 1 uppercase letter and 1 number')
            return popups.PasswordPopup()
```

Figure 113 Password Check on Register Page

After the password is confirmed to be suitable the email is then checked. The current app doesn't have any check on the username so users can be free to choose whatever they want for this. The Regex library is used to ensure the user has entered a valid email address that contains a string of letters, followed by an '@' symbol, another string of letters followed by a '.' symbol and a final string of letters. If the user's email address does not meet this structure then a popup similar to the password popup is shown onscreen instructing the user to enter a valid email address.

```
elif re.search("^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$", email):
    self.checkRegister(uname, email, pword, isTeacher)
else:
    print("invalid email")
    return popups.InvalidEmailPopup()
```

Figure 114 Email Check on Register Page

After the username, password and email are confirmed to be suitable they are passed into the Register Loop function where they are checked against all of the current users in the database to ensure that the same username or email aren't being used by two separate people. If the username is found to already be in the database a popup is shown to the user explaining this and the function returns with a value of -1 which tells the register page that the function failed. The same happens if the email happens to already be in the database, a popup is shown explaining this and the function returns -1. If both of these checks return nothing, then the function returns a 1 signifying that these credentials are free and that they should be placed into the database.

```
def registerLoop(self, uname, email):
    while True:
        results = firebase.get('/users/', None)

        for index in results:
            if results[index]['username'] == uname:
                popups.UsernamePopup()
                print("this username is already taken")
                return -1
            elif results[index]['email'] == email:
                popups.EmailPopup()
                print("this email is already taken")
                return -1

        return 1
```

Figure 115 Register Loop for Register Page

After the username and email are confirmed to not be in use with any other account in the database the user's information is POSTed into the firebase database along with some initialized attributes like add, subtract, multiply, divide and whether or not the user is a teacher or not. This same information is also passed into the JSON store that will be discussed further in this section and then the user is brought to the login screen.

Logging In:

Whenever a user is brought to the log in screen, the app will attempt to bring over the data stored in the JSON store mentioned later in the document and attempt to help the user with logging in since it will already know their details from the time they registered or the last time that they logged in. Figure 116 shows the 'on_pre_enter' function that is built into Kivy that runs whenever a screen is being transitioned to from another screen. In this function, there are two try-except blocks. These ensure that if the user does not already have a JSON store on their device that the app won't crash. It attempts to grab the username and password of the user from the store and if it can't find this information then it just passes empty strings into the username and password text input fields.

```
def on_pre_enter(self, *args):
    try:
        ScreenManagement.store.get('credentials')['username']
    except KeyError:
        self.username = ""
    else:
        self.username = ScreenManagement.store.get('credentials')['username']
    try:
        ScreenManagement.store.get('credentials')['password']
    except KeyError:
        self.password = ""
    else:
        self.password = ScreenManagement.store.get('credentials')['password']
```

Figure 116 Login Screen On Pre Enter Function

When the user presses the login button a similar process happens to the one when a user registers an account. A login loop function runs to ensure that the user's credentials are found in the firebase database but instead of checking for an email or username in the database, like on the register screen, this time the login loop ensures that both the username and password are both present in the database together or else the user will not be allowed to log in.

```
def loginLoop(self, uname, pword):
    while True:
        results = firebase.get('/users/', None)

        for index in results:
            if results[index]['username'] == uname and results[index]['password'] == pword:
                return 1

        return -1
```

Figure 117 Login Loop Function on Login Screen

If the login loop returns -1 this means that no username and password combination match anything in the database and so an appropriate popup is shown to the user explaining that either the password or username is incorrect and that they should check the credentials before attempting to log in again.

```
def checkLogin(self, uname, pword):
    if self.loginLoop(uname, pword) == -1:
        print('this username and password do not match anything in the database')
        return popups.UsernameAndPasswordPopup()
    else:
        if self.updateJsonLoop(uname) == 1:
            print("successfully updated JSON file")
        else:
            print("couldn't update JSON file")

        self.checkWeek()
        self.manager.current = 'main'
```

Figure 118 Check Login Function on Login Page

If the login loop returns 1 then that means the username and password combination has been found in the database and the user is safe to login. The JSON store is then updated with the user's login information by running the 'updateJsonLoop' function. This function takes in the username that the system now knows should be allowed to login and stores all of this user's information into the JSON store from the Firebase database before moving on to the last steps of the login function. The reason the JSON file needs to be updated every time the user logs in is that multiple users may be using the same device and if one user logs in on a single device and the JSON file is never updated when another user logs in, the second user will be overwriting the first user's data and neither user will get any benefit out of the application.

```
def updateJsonLoop(self, username):
    while True:
        results = firebase.get('/users/', None)

        for index in results:
            if results[index]['username'] == username:
                ScreenManagement.store.put('credentials',
                                           username=results[index]['username'],
                                           password=results[index]['password'],
                                           email=results[index]['email'],
                                           teacher=results[index]['teacher'],
                                           classroom=results[index]['classroom'],
                                           add=results[index]['add'],
                                           subtract=results[index]['subtract'],
                                           multiply=results[index]['multiply'],
                                           divide=results[index]['divide'])
                return 1

        return -1
```

Figure 119 Update JSON Loop Function

After the user's details have been confirmed and added to the JSON store, a function 'checkWeek' runs. This function needs to know who is logged in before it can run to avoid applying the wrong person's progress to a different user. The function works by getting the current year, week and day and checking the user's information in the database to see if they have any data saved for the current week. If the variable 'check' returns nothing, which it will if the user hasn't logged in on any given week, a new weekly progress entity is created in the database for them. The else statement shows the firebase library entering '0' values for the "bestScore", "totalGamesPlayed", "timePlayed", "correctAnswers" and "totalXP" attributes for the given week that the user has logged in on. This will allow the users to compare weekly progress on a week-by-week basis in the progress section of the app.

```
def checkWeek(self):
    currentDate = datetime.date.today()
    currentYear, currentWeek, currentDay = currentDate.isocalendar()

    results = firebase.get('/users/', None)

    for index in results:
        if results[index]['username'] == ScreenManagement.store.get('credentials')['username']:

            check = firebase.get('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek), None)

            if check:
                pass
                # print(check)
            else:
                print('Didnt find the current week in the DB')
                firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                             'bestScore', 0)
                firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                             'totalGamesPlayed', 0)
                firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                             'timePlayed', 0)
                firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                             'correctAnswers', 0)
                firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                             'totalXP', 0)
```

Figure 120 Check Week Function

After the 'checkWeek' function runs, the user is then finally brought to the main menu screen of the app where they can choose what they want to do next.

JSON Store:

A JSON store was also set up that is local to the device that the user is using. This was utilized to cut down on load times when trying to retrieve information from the database which would get longer and longer as the number of users using the application scaled up.

When a user logs in, their information is saved to the JSON store and this is then used throughout the application when a piece of information is needed from the user that doesn't necessarily change

throughout a single use of the app. Figure 121 shows a code snippet importing the JSON store library, setting it up and then using it to both retrieve information ("ScreenManagement.store.get") and put information into it ("ScreenManagement.store.put").

```
from kivy.storage.jsonstore import JsonStore
    data_dir = App().user_data_dir
    store = JsonStore(join(data_dir, 'storage.json'))
            ScreenManagement.store.get('credentials')['password']
           ScreenManagement.store.put('credentials', username="",
                                      password="",
                                      email="",
                                      teacher="",
                                      classroom="",
                                      add="",
                                      subtract="",
                                      multiply="",
                                      divide="")
```

Figure 121 JSON Store Code Snippet

Creating and Joining Virtual Classrooms:

Teacher's Student List:

This was another challenging part of the development that took a long time. Up until this point I was creating static graphical elements in the kv file and dynamically changing some elements like the text or image but for this section I had to dynamically create a list of buttons. This proved to be very difficult and took some amount of time to figure out but the solution is graceful and works well within the app with minimal load times.

When a teacher creates a virtual classroom students are then able to join it as long as they know the name. As students begin to join the classroom the teacher can go to their own progress page in the app and see the students joining their student list. This list is one of the few places where the kv file is not used to create some sort of graphical section of the application as an entire list needs to be constructed whenever the teacher enters their progress page. Every time the teacher enters their progress screen the 'createScrollView' function runs.

```
class TeacherProgressPage(Screen):
    view = ObjectProperty(None)

    def on_pre_enter(self, *args):
        self.createScrollview(1)
```

Figure 122 Teacher Progress Page and On Pre-Enter Function

The ‘createScrollView’ function first creates an empty list called ‘studentList’ before grabbing all of the users from the database as has been shown before in this document. Then it loops through all of the users and checks does the student's classroom name match the teacher’s classroom name. This is one of those situations where the JSON file comes in handy. The app knows that a teacher is signed in based on the JSON check that happened when they logged in so it knows that it can search the JSON store to find the teacher's classroom name and avoid having to make an unnecessary call to the database. If a user is found to have the same classroom as the teacher they are appended to the student list and the loop moves onto the next user. After all of the users have been checked and the students are either in the student list or not, a grid layout is created that stretches the height of the screen and has only one column. For every student found in the student list, a button is created with the student's username as the text, and it is styled the same way all of the other buttons are styled in the application. The button is also given an ‘on_release’ function that sends the students name that is clicked to the ‘studentInfo’ function. Then this list of buttons is appended to the grid layout and the whole layout is put into a scrolling layout so that if there are many students the teacher can swipe up and down to find the user that they are looking for.

```
def createScrollview(self, dt):

    studentList = []

    results = firebase.get('/users/', None)

    for index in results:
        if results[index]['classroom'] == ScreenManagement.store.get('credentials')['classroom'] and results[index][
            'username'] != ScreenManagement.store.get('credentials')['username']:
            studentList.append(results[index]['username'])

    layout = GridLayout(cols=1, spacing=10, size_hint_y=None)
    layout.bind(minimum_height=layout.setter("height"))

    for studentName in studentList:
        layout.add_widget(Button(on_release=partial(self.studentInfo, studentName),
                                 text=studentName, font_size=self.width * 0.08,
                                 size_hint=(0.8, None),
                                 font_name='Helvetica',
                                 background_normal="button.png",
                                 background_down="buttondown.png",
                                 color=(1, 72 / 255, 72 / 255, 1)))
    scrollview = ScrollView(id="scrollview", size_hint=(1, None), size=(Window.width, Window.height))
    scrollview.add_widget(layout)
    self.view.add_widget(scrollview)
```

Figure 123 Create Scroll View Function

Student Week List:

Similar to the teacher's student list discussed above, this was also a difficult section to complete with the added difficulty of having to parse items taken from the database. In future this might be avoided by structuring the database differently but for now this solution works.
The student's week list is structured pretty much the same way that the teacher's student list. This time however there is one more complicated step that comes around when creating the list of weeks. In the Firebase database, it is not allowed that integers be the name of an attribute and so, before sending the week number to the database it first has to be converted to a string. This makes things more complicated when attempting to convert this string back into a number for use in this weekly list students have. The database is first queried for the total list of users which we have seen before and then the user's information is found in the database. After this, the user's progress section is queried and each element is added to a list before it can become an integer for use in the list. A diagram of this retrieval and parsing can be seen in figure 124.

The numbers first come as a list of strings which are then broken down to individual lists on their own with brackets and quotation marks around them. These individual lists are then turned into strings themselves and are sliced using Pythons string slicer ([2 : -2]) to remove the two characters at the start and two characters at the end to reveal the plain integers for use in the week list.

```
['12','13','14']

['12'],['13'],['14']

  12,    13,    14
```

Figure 124 Breakdown of Week Numbers from Database

```
weekList = []

results = firebase.get('/users/', None)

for index in results:
    if results[index]['username'] == ScreenManagement.store.get('credentials')['username']:
        progress = firebase.get('/users/' + index + '/progress/' + str(currentYear), None)
        for x in range(len(progress)):
            weekList.append(str([key.split(',') for key in progress.keys()][x])[2:-2])
```

Figure 125 Creating the Week List for Students Progress

The process after this is identical to the teacher's student list to construct a scrolling view list of weeks that students can click on and see their weekly report for the selected week.

Weekly Progress Report:

When the user chooses to check the report of a specific week they are brought to a report page where a function checks if there is any report for the previous week. If there isn't a report from the previous week, the report is filled with 'N/A' or 'not applicable' strings to signify to the user that they do not have any data for the previous week to show a progress update (figure 126). If there are results for the previous week then these are shown to the user as well as a change in the results whether these are improvements or if the chosen week has worse results than the last (figure 127).

	Last Week	This Week	Change
Best Score	N/A	0	-
Correct Answers	N/A	0	-
Time Played	N/A	0	-
Total Games Played	N/A	0	-
Total XP	N/A	0	-

Figure 126 No Previous Week Example

	Last Week	This Week	Change
Best Score	10	6	-4
Correct Answers	47	72	53.2%
Time Played	294	357	21.4%
Total Games Played	12	16	33.3%
Total XP	301	500	66.1%

Figure 127 Previous Week Example

To get the change of best score from one week to another it was as simple as taking the best score from the current week and taking away the best score from the previous week (figure 128). The format shown in figures 129 - 132 was used to work out the percentage changes from one week to another. The function checkZeros (figure 133) was used to ensure that no division could be done on zero because Python is one of the programming languages that does not have anything set up to handle the division of zero like something that JavaScript has so I had to create my own function that handled this possibility.

```
self.ids.bestScoreChange.text = str(int(thisWeek['bestScore']) - int(lastWeek['bestScore']))
```

Figure 128 Numerical Change for Best Score

```
self.ids.correctAnswersChange.text = str(round(
    float(self.checkForZeros(int(thisWeek['correctAnswers']) - int(lastWeek['correctAnswers']),
                             int(lastWeek['correctAnswers'])) * 100), 1)) + '%'
```

Figure 129 Percentage Change for Correct Answers

```
self.ids.timePlayedChange.text = str(round(float(
    self.checkForZeros(int(thisWeek['timePlayed']) - int(lastWeek['timePlayed']), int(lastWeek['timePlayed'])) * 100), 1)) + '%'
```

Figure 130 Percentage Change for Time Played

```
self.ids.totalGamesPlayedChange.text = str(round(float(
    self.checkForZeros(int(thisWeek['totalGamesPlayed']) - int(lastWeek['totalGamesPlayed']),
                       int(lastWeek['totalGamesPlayed'])) * 100), 1)) + '%'
```

Figure 131 Percentage Change for Total Games Played

```
self.ids.totalXPChange.text = str(
    round(float(self.checkForZeros(int(thisWeek['totalXP']) - int(lastWeek['totalXP']), int(lastWeek['totalXP'])) * 100), 1)) + '%'
```

Figure 132 Percentage Change for Total XP Earned

```
def checkForZeros(self, first, second):
    return first / second if second else 0
```

Figure 133 Check For Zeros Function

4.5. Conclusions

This section showed off the inner workings of how the front middle and back end of the system works beneath the hood of the brightly coloured GUI that the user sees in the final version of the application. Code snippets break down the development to show exactly what went into bringing the design to life within the app. This was the most challenging section of the project overall and this is visible through complicated code snippets and descriptions of how each working part works behind the scenes of what is shown to the user.

5. Evaluation

5.1. Introduction

This section of the document will look into testing and evaluating the application and project as a whole. Discussion about the testing plan that was unable to be put into practice, as well as what testing was possible is discussed along with a critical examination of extra questionnaires and other external testing that was conducted.

5.2. Software Evaluation

As mentioned in section 3 the test plan originally set out in the interim report was unfortunately unable to go ahead due to the outbreak of COVID-19 and the subsequent lockdown that has impeded me from setting up testing groups with primary school students that were originally planned out through conversations with myself and some primary school teachers that I contacted.

White Box Testing:

White box testing is a name given to the type of tests performed by someone who knows the ins and outs of the application before testing begins.[23] This section will discuss the types of tests that I conducted myself before giving the application over to someone who has never used it before to see if there is anything I could pick up and fix before handing the app off to a potential user.

- **Unit Testing:**

Throughout the development of the application, every time a new feature needed to be added it was tested locally on its own in a separate python file to ensure it worked with different types of available inputs that it would be getting. Every time a function was implemented, I manually went through all of the possible inputs that it could receive from the classes and other functions throughout the application. When working with items that are coming from the database where information could potentially be anything a user wanted like a username, for example, things got slightly tricky for these situations so try-except blocks were used throughout the application to catch any unwanted behaviors from crashing the app or changing unwanted details in the database.

For example, in the case of the Increment Timer function, it was easy to test as I knew that the numbers would be fixed every time that the function gets called. I could leave a minigame running and I would know that the increment worked as I could visually see the timer clicking up and up as time went by.

```
def incrementTimer(self, interval):
    if int(globalVariables.seconds) == 59:
        globalVariables.seconds = 0
        globalVariables.minutes += 1

    if globalVariables.minutes < 10:
        if globalVariables.seconds > 9:
            self.timer = "0" + str(globalVariables.minutes) + ":" + str(round(globalVariables.seconds))
        else:
            self.timer = "0" + str(globalVariables.minutes) + ":0" + str(round(globalVariables.seconds))
    else:
        if globalVariables.seconds > 9:
            self.timer = str(globalVariables.minutes) + ":" + str(round(globalVariables.seconds))
        else:
            self.timer = str(globalVariables.minutes) + ":0" + str(round(globalVariables.seconds))

    globalVariables.seconds += .1
```

Figure 134 Increment Timer Function

Figures 135, 136 and 137 show that the timer function works as intended, it removes the extra zeros when necessary. When the first 60 seconds pass, the code adapts properly to facilitate the 1 minute being shown on screen and if the minigame ever gets to 10 minutes somehow the other 0 is removed and replaced with the number 10 so that the timer imitates a proper stopwatch timer.

Figure 135 Timer Example 1

Figure 136 Timer Example 2

Figure 137 Timer Example 3

Python does have a unit test library available where functions can be tested but for something like this application where there are graphical elements and other objects that aren't native to pure Python this wasn't much use except for the pure Python functions in the code which were few and far between as the main focus of this project was creating a user-friendly visual learning application.

Optimizations:

While testing the application myself I came across two places where there was a reoccurring issue of the application stalling and taking too long to move on to the next screen. Creating a good user experience was one of the main objectives of this project and for the application to take a long time to load something is quite the opposite of that so I set out to improve this problem. One of the places where this stalling was occurring was when the user first logs in. The problem only arises when the

user has logged in for the first time in a single week so it was easy to pinpoint the location of this slowing code. The section of code shown in figure 138 was the issue. For some reason, I was calling 5 separate instances of the firebase put method instead of just making one proper call with all of the necessary data in it. This obviously was a quick solution that I implemented and never returned to fix up and this situation is an example of technical debt that could have been avoided by planning ahead of time.

```
else:
    print('Didnt find the current week in the DB')
    firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                 'bestScore', 0)
    firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                 'totalGamesPlayed', 0)
    firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                 'timePlayed', 0)
    firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                 'correctAnswers', 0)
    firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                 'totalXP', 0)
```

Figure 138 Slow Code Snippet

I decided to time how long this stall would take on average whenever a user logged in for the first time in a single week before fixing the issue. After I fixed the code I timed the section of code running again and compared these results to the initial problem to see how optimized the code was in comparison. Figure 139 shows the results.

	Old Code	New Code
	4.35	2.73
	4.85	2.9
	4.9	2.76
	4.93	2.88
	4.86	2.7
Average Speed:	**4.78**	**2.79**

Figure 139 Average Speed of Old and New Code

These results show that there was a 41.63% increase in speed after the code in figure 140 was put in place instead of the old code that was slowing the system down.

```
firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/', str(currentWeek),
             {'bestScore': 0, 'totalGamesPlayed': 0, 'timePlayed': 0, 'correctAnswers': 0, 'totalXP': 0})
```

Figure 140 Faster Code Snippet

The other place where this issue was arising was anytime the user finishes a minigame. It takes a long time for the app to bring them to the results page, this is again due to multiple firebase calls being used instead of using just one call to cut down on waiting times. This time the problem was worse because not only was I using multiple PUT calls but everytime I was calling PUT I was also calling GET which was doubling the slow down every time. I timed the results again before fixing the issue to get the percentage speed up from fixing this issue. Figure 141 shows the original code and figure 142 shows the table of speeds taken.

```
firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
             'totalXP', int(str(firebase.get(
        '/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek) + '/totalXP',
        None))) + int(xpEarned))

firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
             'totalGamesPlayed', int(firebase.get(
        '/users/' + index + '/progress/' + str(currentYear) + '/' + str(
            currentWeek) + '/totalGamesPlayed', None)) + 1)

firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
             'timePlayed', int(str(firebase.get(
        '/users/' + index + '/progress/' + str(currentYear) + '/' + str(
            currentWeek) + '/timePlayed', None))) + int(globalVariables.seconds) + int(
        globalVariables.minutes * 60))

firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
             'correctAnswers', int(str(firebase.get(
        '/users/' + index + '/progress/' + str(currentYear) + '/' + str(
            currentWeek) + '/correctAnswers', None))) + int(globalVariables.correctAnswers))

if int(globalVariables.correctAnswers) > int(str(firebase.get(
        '/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek) + '/bestScore',
        None))):
    firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/' + str(currentWeek),
                 'bestScore', int(globalVariables.correctAnswers))
```

Figure 141 Slow Code Snippet

	Old Code	New Code
	7.53	1.8
	5.75	1.8
	6.13	1.82
	5.78	1.76
	5.72	1.88
Average Speed:	**6.15**	**1.81**

Figure 142 Average Speed of Old and New Code

```
if results[index]['username'] == ScreenManagement.store.get('credentials')['username']:
    firebase.put('/users/' + index, operator, currentXP)
    if int(globalVariables.correctAnswers) > \
            results[index]['progress'][str(currentYear)][str(currentWeek)]['bestScore']:
        firebase.put('/users/' + index + '/progress/' + str(currentYear) + '/', str(currentWeek), {
            'totalXP': int(
                results[index]['progress'][str(currentYear)][str(currentWeek)]['totalXP']) + int(
                xpEarned),
            'totalGamesPlayed': int(
                results[index]['progress'][str(currentYear)][str(currentWeek)]['totalGamesPlayed']) + 1,
            'timePlayed': int(
                results[index]['progress'][str(currentYear)][str(currentWeek)]['timePlayed']) + int(
                globalVariables.seconds) + int(globalVariables.minutes * 60),
            'correctAnswers': int(
                results[index]['progress'][str(currentYear)][str(currentWeek)]['correctAnswers']) + int(
                globalVariables.correctAnswers),
            'bestScore': int(globalVariables.correctAnswers)
        })
```

Figure 143 Faster Code Snippet

To fix this issue I made one GET call to retrieve all of the information I would need and two PUT calls to add all the necessary information to the database This brought the total number of calls from 10 down to three. The speed increase gained by fixing this code was 70.6% which is an extremely good speed up from a simple fix that didn't take too long to implement.

Test Cases:

I created test cases to test the functionality of the application myself in the absence of having a larger group of primary school students testing the application themselves and then reporting back with bugs, suggestions and general ideas on how to better the application. These test cases were created to ensure that the intended functionality was implemented properly and if it wasn't, what went wrong or what could go wrong and were proper error handling measures put in place. If they weren't, a bug was created in the Airtable bug list that was mentioned before in the document and once testing had concluded, redevelopment of the application began and I worked my way through as many bugs as I could before revisiting the use case and updating it as they were fixed.

Test No.	Test Description	Expected Outcome	Pass?	Comment	Fixed?
1	User tries to register account with already taken username	Popup should be shown to user explaining that username is already in use	Yes	Works as intended	-

2	User tries to register account with already taken email address	Popup should be shown to user explaining that email address is already in use	Yes	Works as intended	-
3	User logs in with wrong username and password combination	Popup should be shown to user explaining that their username and password combo is wrong	Yes	Works as intended	-
4	User clicks log out button	Popup is shown to user that asks are they sure they want to log out	Yes	Works as intended	-
5	User presses back button on teacher progress page	User goes back to main menu page	No	Accidentally hooked up teacher progress back button to addition page	Yes
6	Teacher deletes classroom for everyone in it	Teacher deletes classroom and all student's classrooms automatically update to "no classroom"	No	Teachers classroom is set to "no classroom" but all students still have the classroom name	No
7	User presses leave classroom button	User's classroom should be set to "no classroom" in the database	No	Users classroom name is being saved as "Current Classroom" + classroom name	Yes
8	Teacher presses delete classroom button when they have no classroom	Nothing should happen	Yes	Works as intended	-
9	User attempts to send email to invalid email address	Error message should appear in console	No	Crashes application even though there is a try except block in the code	No
10	User sends email to valid email address	Email sends and user receives email in their inbox	Yes	Works as intended	-
11	User tries typing on minigame screen	Nothing should happen	Yes	Works as intended	-
12	User tries to delete their	Account successfully deletes from	Yes	Works as	-

	account	database		intended	
13	User leaves minigame half-way through	No database information is altered, user is free to attempt minigame again later	Yes	Works as intended	-
14	User attempts to update their username	Users username changes and is updated in the database	Yes	Works as intended	-
15	User spends over 10 minutes on minigame	Timer should update as necessary	No	Minutes stack on top of 0 that was already there so output looks like “012:34”	Yes
16	User gets answer correct in minigame	Popup shown to user with smiley face and correct answer text on it	Yes	Works as intended	-
17	User gets answer wrong in minigame	Popup shown to user with sad face and incorrect answer text on it	Yes	Works as intended	-
18	User gets answer correct in minigame	Correct answer sound plays	Yes	Works as intended	-
19	User gets answer wrong in minigame	Incorrect answer sound plays	No	Accidentally hooked up correct sound to incorrect function	Yes
20	User presses X button when in minigame	User should be prompted with popup asking are they sure they want to leave the minigame	Yes	Works as intended	-
21	User presses Yes when in leave minigame popup	User should be brought back to the play page	No	Accidentally set to bring user back to main menu	Yes
22	User presses no on leave minigame popup	Popup should be dismissed	Yes	Works as intended	-
23	User clicks outside of popup on leave minigame popup	Popup should be dismissed	Yes	Works as intended	-
24	User presses back button on addition level selection screen	User should be brought to operator selection screen	Yes	Works as intended	-
25	User presses back button on	User should be brought to	Yes	Works as	-

	subtraction level selection screen	operator selection screen		intended	
26	User presses back button on multiplication level selection screen	User should be brought to operator selection screen	No	User is brought to main menu	Yes
27	User presses back button on division level selection screen	User should be brought to operator selection screen	Yes	Works as intended	-
28	Results page user xp to next level should be displayed on progress bar	User progress to next level is shown on the progress bar	Yes	Works as intended	-

Figure 144 Test Case Table

Black Box Testing:

Black box testing is the opposite of white-box testing. It can be visualized by thinking about the input going into a black box that no one can see into and then receiving an output on the other side. It refers to the type of testing that users who have never interacted with the application before do.[24] For this type of testing, I decided to hand over the application to members of my family who are very technologically illiterate to test its functionality in place of the larger group of primary school students I had originally intended to use for black-box testing purposes. While members of my family were testing out the application, I observed without giving them any instructions and asked could they complete some set of instructions without my input to test if the application was easy to navigate through. Figure 145 shows a black box testing table that was created for the 3 members of my family and the results were recorded based on different challenges I put forth for them to complete.

Task	User 1	User 2	User 3	Comments
Register an account	YES	YES	YES	If I make a mistake putting in my password, it shows an alert that has no x to get out.
Log in with your account	YES	YES	YES	
Play Level 1 of the multiplication minigame and return to the main menu	YES	YES	YES	1. When choosing an answer I didn't know I had to click screen to return to the game 2. Didn't know to press screen to return to game
Join the classroom called 'teachersclassroom'	yes	YES	YES	1. No indication to users that the classroom name that they entered doesn't exist

Leave your current classroom	YES	YES	YES	No issues
Send a progress email to yourself with your report for the current week	YES	YES	YES	1. Back button wasn't clear. Needs to be on the screen instead of words. Should say back instead of previous pages name 2. Slightly difficult to understand
Change your username to add a '123' to the end of it	YES	YES	YES	1. Change username button should bring user to a new page to change their username rather than be the button that changes it. 2. No issue 3. It wasn't clear that my username actually changed.
Log out of your account	YES	YES	YES	1. Get back to login page it should say log out not log in 2. Log in button instead of log out button
Delete your account	YES	YES	YES	3. It should say, You have successfully deleted your account

Figure 145 Testing Table

After handing the app over to the users who had never seen the app before and allowing them to use it with minimal instructions it was clear what parts of the application were working well and what parts needed some work done. All of the issues that arose from this test were added to the Airtable bug list and were later tackled after testing had concluded. One issue that seemed to come up for every user that tested the application was that they did not know how to handle popups when they appeared on the screen. It took users a while to figure out that they had to click away from the popup to dismiss it and so something to implement in the future is a button that says "dismiss" along the bottom of the popup to indicate to users they must do something to close the popup. This issue was especially prevalent in the minigame section of the app where a popup appears every time the user guesses a question. Many got frustrated when the popup did not automatically close for them and they couldn't figure out how to dismiss it themselves until after a few random clicks on the screen. Another issue that users pointed out was that back buttons were not intuitive to what they were used to when compared to other applications. I thought that by putting the previous screens name in the back button as text it would indicate to the user that they were being brought back to somewhere they were familiar with rather than going back into the unknown but evidently, this proved to be a bad solution to a problem I created myself. Another minor issue that I overlooked during development was the lack of popups to indicate to the user what is happening when they press a button or enter text. While I was testing I could always look at the terminal to see if there were issues or not but users of the app did not have this ability and so they believed they were being left in the dark with certain things like changing their username and sending themselves an email.

Overall, the users seemed happy with the application, they all mentioned how they enjoyed the email system as they got the email sent to their own phones almost instantly after pressing the button. Apart from these issues discussed there isn't much else that the user can interact with so I believe these bugs that came up while testing was happening are the only ones left to fix before moving on to other future work in the project.

Another step to take in the future would be to create a check to see if it's the users first time logging into the application and if it is then a short video might play to teach the user how to use the application and explain what each button does before letting them experience the app themselves. This would ensure that any uncertainties that the user might have about functionality within the app will be removed.

5.3. Questionnaires and Interviews Evaluation

Font Questionnaire and Graphs:

When deciding on a font for the application I selected a few that I thought would be appropriate and eventually brought the number of fonts down to two different ones. The two fonts I decided on were the default Kivy font and a font called Helvetica Textbook. When I showed these two fonts to a group of 19 people, they got to choose which font they would like to see in the application over the other. The results of this questionnaire are shown below in figure 146.

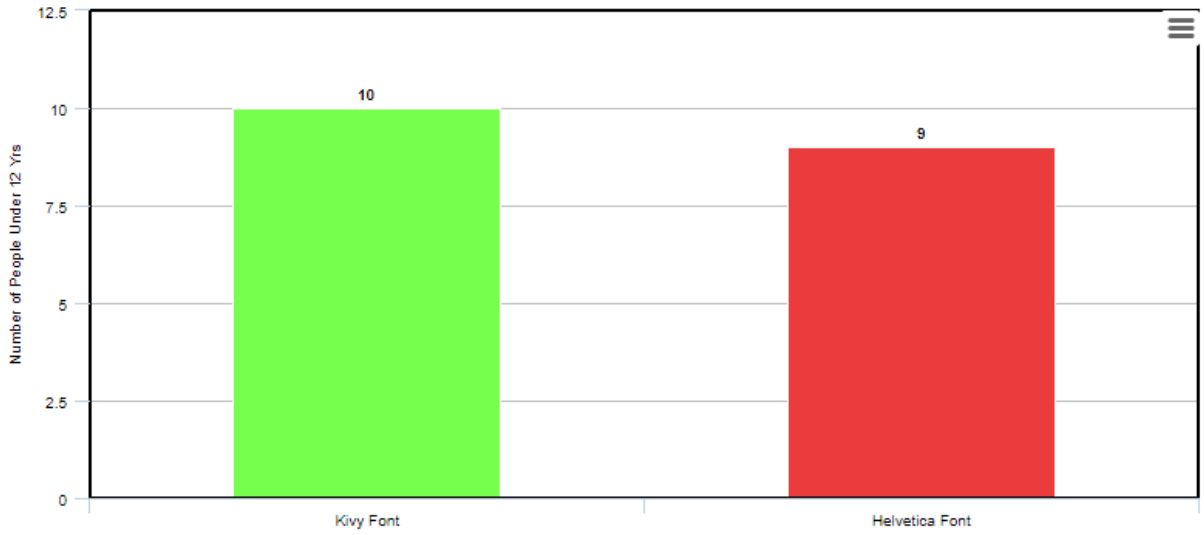


Figure 146 Bar Graph Showing Peoples Font Preference

The interesting part of this questionnaire arises when we look into the different age groups of people when it came to choosing their preferred font. The breakdown of users preference based on age range can be seen in figure 147 and 148.

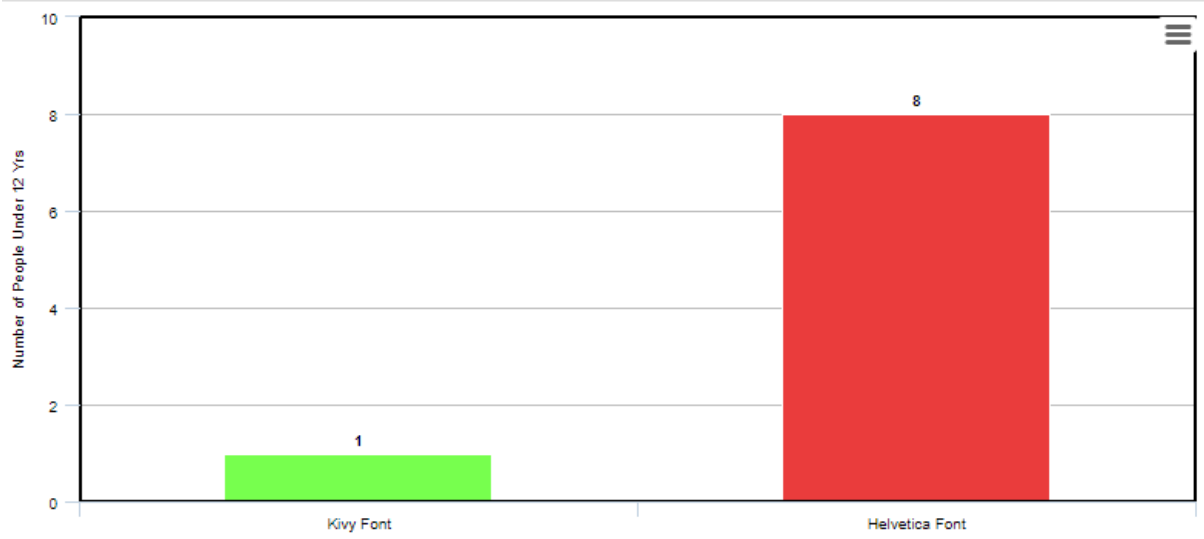


Figure 147 Bar Graph Showing People Under 12s Font Preference

As we can see in figure 147. When people under 12 years of age were asked about their font preference only one person out of the 9 people asked voted for the default Kivy font whereas 8 others voted for the Helvetica font. This means that 89% of people under the age of 12 voted for the Helvetica font and when questioned on this, they responded that they liked the way the font was easy to read and that it reminded them of their textbooks that they have been reading from in school. This proved that the Helvetica font looks like something that they are comfortable with reading and something that they would have no difficulties understanding.

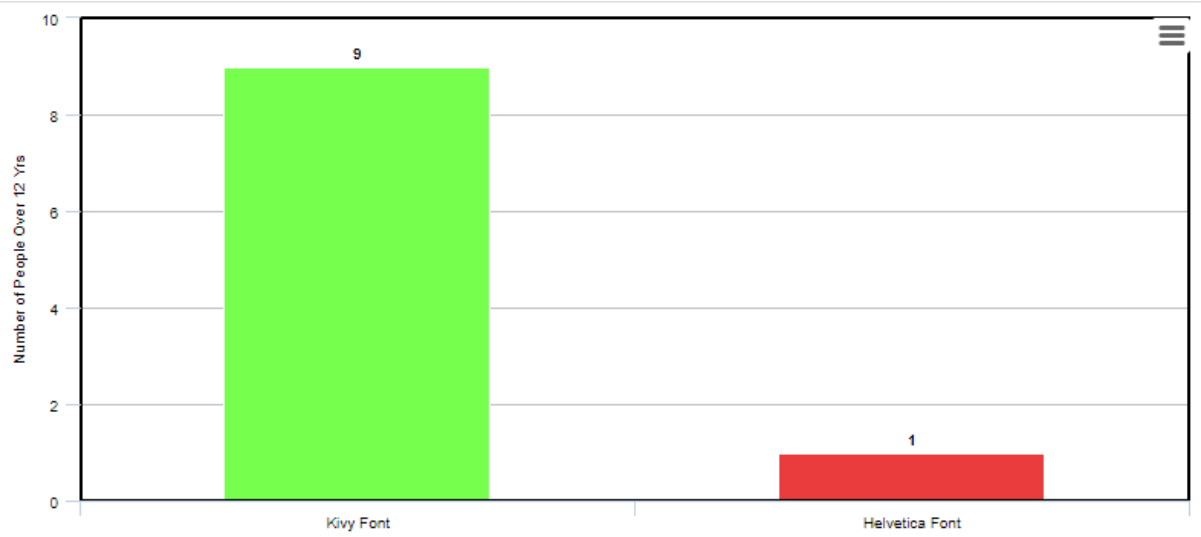


Figure 148 Bar Graph Showing People Over 12s Font Preference

Figure 148 shows almost the opposite result for people over the age of 12s font preference. This time when questioned, the older people responded that they just liked the design of the Kivy font more and felt that the Helvetica font looked too simple and childish. They weren't interested in the readability of the font they were only interested in how appealing it looked in the context of the app.

After hearing these results and taking into account that the target market was for third-class primary school students it was decided that the Helvetica Textbook font would be chosen to go into the final version of the Primary Mathletes application rather than the default Kivy font.

Teacher Proposal Questionnaire:

When the application was in the ideology phase where ideas were just being written down and nothing had yet been set in stone, a questionnaire was sent out to primary school teachers for their opinions on the idea and what they thought would be beneficial to the project.
The proposal/questionnaire document that was sent out to the teachers can be seen in Appendix 1 and those who replied and consented to have their responses included in this project can be seen in Appendices 2 – 5.
According to the teachers questioned, almost all primary schools in Ireland will have access to iPads and/or laptops at some stage throughout their learning week. The application created for this project would be a great addition to their time with these iPads and laptops because as one of the teachers put it, using the app "wouldn't feel like work to them" and so they could be reinforcing learning outcomes of the curriculum while still enjoying themselves.
I also learned of great teacher resources that I used as inspiration for the minigame section of the application that the teachers mentioned such as Twinkl, Transum Maths, ixl.ie, NRich Maths, etc.
Teachers also brought up good points like the fact that not all students will be provided with iPads and laptops at all times while in school but the purpose of the application is not to replace the maths curriculum in school but rather help solidify what the users have already learned so that their retention of information improves.
Using positive reinforcement like the one seen in the correct answer popup with the smiley face was also a suggestion laid out in one of the teachers' answers to the questionnaire and so this was added in as part of the design of the app.
Another thing mentioned but not implemented was allowing users to customize their profiles with profile images and giving out badges/awards. This would have been a good feature to implement as it would keep younger users entertained and engaged within the map but it didn't fall within the scope of the project.

5.5 Full Use of Application

While I was working on the redevelopment of the application to the best of my ability I also tried to answer the most important question this app set out to solve and that is, is it possible for the users mathematical abilities to improve with using the Primary Mathletes application. I asked my younger sister to take part in a two-week test that asked would she spend 20 minutes a day using the application and playing minigames. After this period, I would check back and take a look at her progress page to see was there any change at all between the previous week and the current week that she was working through. If everything went to plan this is the kind of testing, I would have attempted to do with a larger group of primary school students so that I would have a larger dataset to see was
After the first week concluded the user wrote some notes on how to improve the experience of the application. These are discussed below.

Experience Points:

The ability to gain experience points is vital to the gamification of learning that is a big part of the Primary Mathletes project. After using the application for 20 minutes for 5 days straight it became apparent to the user that the number of experience points gained was far too low and the number of experience points to unlock the next level seemed far too low as well. After the first day, the user was able to unlock almost every level for the Addition minigames. Then numbers given for correct answers and fast times were only supposed to be a base but due to the lack of external testing, redevelopment of the experience points system never came about until it was too late to implement something else. This was another example of the redevelopment stage of the project being less productive than it could have been.

Another issue with the experience points system is that there wasn't a big enough drawback to mindlessly clicking potential answers on the minigame screen until it was over. I attempted to add in a penalty for this sort of thing wherein users would receive -10 points for every incorrect answer they gave but users still end up with a net gain of points no matter how poorly they do in a minigame as long as they get one question correct. A feature that might fix this issue is a life point system. This would mean that if users lose all of their lives in a single minigame they will be kicked out, gain 0 experience points and will have to try again if they want to continue gaining points. This would still give the students room to get answers wrong but wouldn't allow them to mindlessly press buttons and end up not learning anything.

Difficulty:

Another issue that arose from the user playing the game every day is that the levels quickly get exponentially harder. Early levels, especially for multiplication and division are exceptionally easy because only numbers from 1-3 are available to work with while the user is such a low level. Implementing a more detailed and thoughtful difficulty curve would benefit users who are looking for a challenge right from the get-go as well as not making the later levels so impossibly hard that even skilled users find them difficult. A balanced difficulty curve from start to finish would secure user retention as well as making users happier when they get something right without just guessing and getting lucky.

Missing Key Features:

As mentioned in the teacher questionnaire section above, the profile section was neglected more than the other features of the application. Originally this section of the app was supposed to feature user-chosen images, badges and achievements all to keep the user interested and engaged in the application. Because these things weren't essential parts of the app and were more so bells and whistles for the finished app they never got added due to time restraints. However, the ability to edit the users username, password and email address all were supposed to be key features of the application. The main issue arises due to the missing 'Edit Email' ability, This is an issue because users may have signed up to the app with a fake or bogus email address and then ended up wanting to send themselves progress report emails and were unable to edit their email so they miss out on this feature. A simple fix would be to add the ability to edit a user's email address from the profile section of the app.

Another solution would be to make the user confirm their email address by sending them an email and asking them to confirm but I am not sure how to do this and it didn't really need to be in the application for this project to be successful.

After the second week finished, I took a look at the results page for the user and noted any comments made. A short, verbal interview took place to bring up any issues, ideas, comments, and suggestions. There was an improvement in most of the categories that are available in the progress report that is shown to the users. One issue that became prevalent to every user is that the 'best score' category caps at 10 and every user is more than likely going to achieve this score once after repeated use of the application during a single week and so this category would probably suit being an average score rather than a standard high score since the highest achievable score is pretty simple to accomplish.

Minigame Variety:

At the beginning of the ideology phase, I had already decided that the game would have multiple minigames. Once development began, I realized how time-consuming this would be after designing and curating the JavaScript version of the minigame algorithm and the idea of multiple minigames became unfeasible. I spoke to Damian Gordon about this and he suggested that by creating one minigame it showed the ability to create more and this is true. With more time, multiple minigames would be possible to create and these would add variety and keep the application fun and interesting for the users. Anyone would get bored of playing the same minigame repeatedly, 20 minutes a day for two weeks so this is very understandable. The test still stood to see if improvements were visible after using the application for a week.

Insufficient Data:

Another thing that cropped up after this second week of development was that the categories that are shown to the user are pretty useless. While users being able to see their statistic is a very useful thing, especially on a week-by-week basis, the categories that are shown in the progress report page simply aren't enough to gain any valuable data from. A breakdown of each minigame played should be able to be analyzed by the user. A general total experience points measure is useless if the user cannot work out how many experience points they have for a single operation in an easy to find location.

Interview Notes and Results:

After the test had finished and the interview had also concluded, there were a few notable things mentioned. The application has a lot of potential. The gamification of learning maths is a good idea in short bursts but playing for extended amounts of time would require some variation in the minigames to keep things interesting for the user. The addition of extra personalization in the profile section would be a good implementation to incentivize students to compare profiles and increase usage of the app. More gamification could be added by letting users other users weekly scores. This would let users

incentivize themselves to play more minigames and in turn, learn more maths in the process. One other small issue was brought up and that was something I was already aware of but never implemented a fix for. Users can just keep playing the early levels and earn the same amount of experience points as they would for the harder levels. In a game where the levels get progressively harder but the user doesn't gain anything extra for completing them it doesn't do much to incentivize users to play the game as it was intended. A quick but unimplemented fix would be to disable experience points gained from earlier levels once the following levels became unlocked and this will be introduced in future work.

Below we can see the user's results after playing 20 minutes a day for two weeks. The application did help the user learn more maths but there still could be improvements made on the application to beef it up and make it into something that real students and teachers would use as a learning/teaching aid.

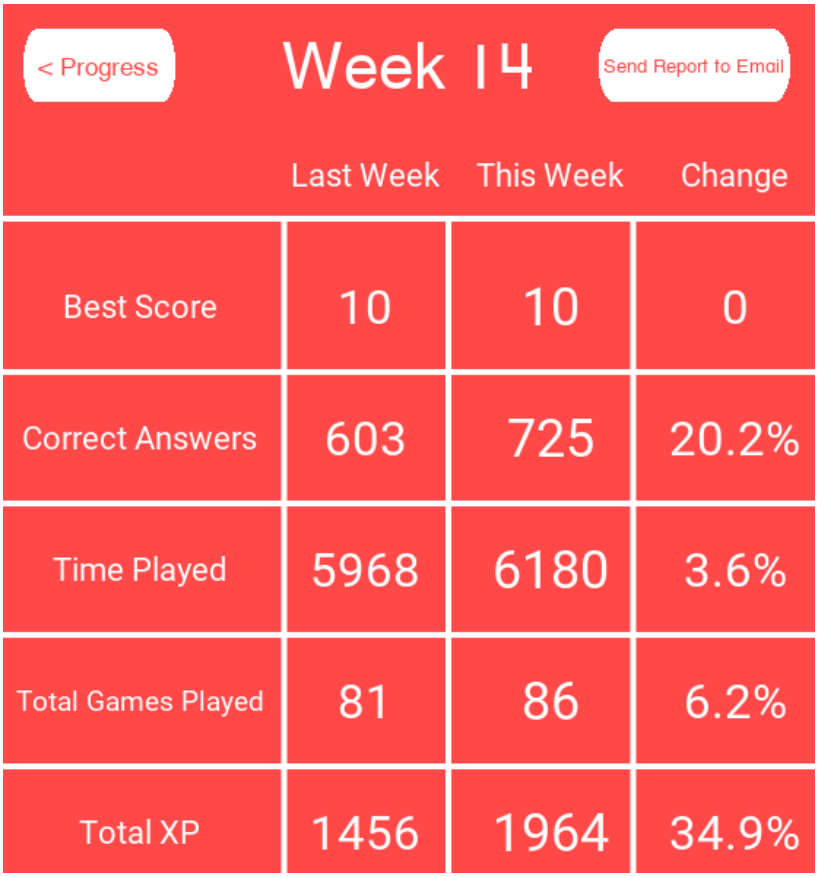


Figure 149 Improvement Through Use of App

5.6 Conclusions

This section went over several different ways that the application was tested as well as an in-depth evaluation of one test that took place over two weeks. The project aimed to create an application that teachers and students could use to reinforce the information that students had already learned in school. In its own way, the application achieved this aim and we can see this by looking at the progress report page of the user that clearly shows an improvement in almost every category. The addition of more users testing the application would have greatly benefitted this section but unfortunately, that was not possible, and I feel that the project suffered somewhat because of this.

6. Conclusions and Future Work

6.1. Introduction

This section will wrap up the document by discussing key points from the project as well as discussing what lies in the near and far future for the continuation of this project. Certain things that did and didn't work for the project are discussed and how things could have been handled differently if I was to start this project again from the start with the knowledge I have now. Future work is discussed with the aid of prototypes that show where the minigame section of the app is going to go and other ideas on where the application could go are discussed as well.

6.2. If I Could Do It Again

Some minor things that I would do straight away is look further into the documentation of Kivy to try and limit the amount of repeated work I would have to do. This was an issue that came up a lot when adding in new features to the application. I would have to replace a lot of different entities in the kv file for example but now that I have learned a lot about the language I know that I could limit a lot of the work that I had to do throughout development.

Another thing I might do would be to look into using a different language all together like maybe React Native which allows for cross-platform applications to be created except that it has built-in standards for both Android and iOS so it is a lot easier to style and a lot easier to compile application files onto devices. One of the biggest issues faced was trying to compile the Kivy application onto a mobile device.

I would begin development sooner than I did this time around, and also begin testing sooner than planned as well.

6.3. Conclusions

This project originally set out to solve a problem that had very limited solutions. To create an application that could help students to improve their maths abilities and to better retain information through the gamification of learning. I believe that the app achieved its goal.
With more time and effort, I think that this application could be put out into the world and be used by teachers, parents, and students to improve the retention of mathematical information. With the recent outbreak of COVID-19, it has become apparent that an application like the one developed for this project could be very useful in times where students are unable to attend a physical classroom for whatever reason they might have, whether that be an illness or even an international quarantine. With further input from actual primary school teachers to ensure the implementation of proper educational information is being presented to the students, as well as more development on the minigames to include different aspects of the curriculum, it would be very possible for the application to be used as

an alternative to reading numbers from a book and instead turning it into an interactive and fun learning experience that could result in further knowledge being gained overall.

The results from the various tests reveal students' interest in the app overall. Some minor issues here and there can be resolved but the idea behind the app is something that students need as a breath of fresh to the mundane task of learning maths from a book.

This project also provided me with the opportunity to learn Kivy, something I never would have looked into if it wasn't for this project. It taught me how to deal with a large-scale issue and how to manage my time over a year-long endeavour in which I was solely responsible. It helped me practice using software methodologies that are used in the real world and it forced me to use everything that I had learned in the last 4 years of computer science. I believe that this project has helped to improve my abilities not only as a programmer but as a critical thinker and problem solver which can now be put to use on different projects in the future.

6.4. Future Work

As seen in section **2,** Airtable was used to document any bugs that were found while testing and there are still several low to normal level bugs that were found too late in the testing phase to be fixed. The application still runs with these issues present but they are quality of life fixes that would be beneficial to implement. When providing an application to younger students it would be especially important to integrate aspects such as having the minigame wait for a second after each correct or incorrect answer to guard the user against miss-clicking the next answer straight away, which is a possibility in the current state of the application so this would be one of the first things to be fixed in the future.

In the future it would also be beneficial to turn the application into a web app by developing a Django database using MySQL or Oracle might not be a bad idea because Firebase's pricing plans are pretty expensive when scaling up with more and more users using the application.

Creating more minigames would be vital to the future success of this project as only having a single type of question that can be asked gets repetitive quite quickly. The addition of more minigames alongside additional educational content from the curriculum outside of basic operators would not only benefit the users currently targeted by this project (3rd class primary students) but could also be scaled up and down to include the entire curriculum needed to teach all primary school students from 1st to 6th class.

Figure 150 shows a mockup of what another minigame could potentially look like in a future version of the app. It is a similar to the structure seen in the Primary Mathletes app now but it would have a sequence of numbers where the student would have to click the correct answer to complete the full sequence. In this example the answer would be 6 because the numbers are going up in 3's every time. This idea of spotting sequences in numbers is something students learn from a young age.

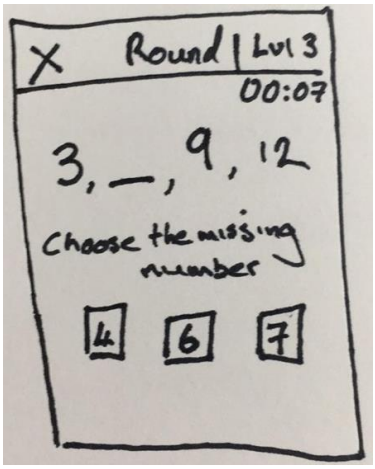


Figure 150 Future Minigame Prototype 1

Figure 151 shows another mockup, this time it is a odd-one-out minigame. This idea stemmed from a suggestion one of the teachers had when responding to the questionnaire I sent out at the start of the academic year so I know it is something that the target market for the app is interested in. This one is similar to the other mockup but instead of completing the sequence this time the user must find the odd one out in the sequence.

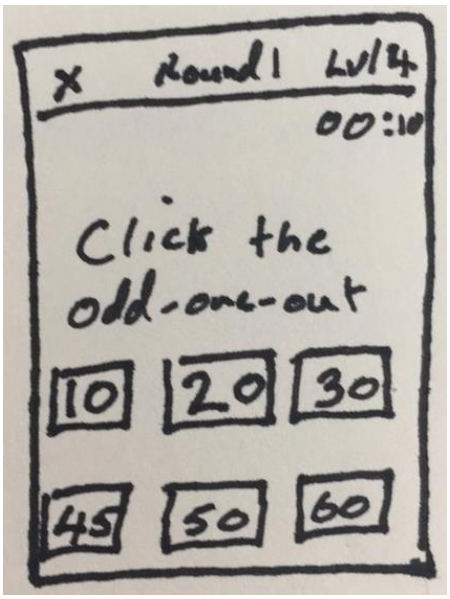


Figure 151 Future Minigame Prototype 2

Other smaller and quicker additions to the application could be made, such as an improved experience system. The one implemented at the moment can be exploited easily by spamming any of the potential answer buttons and, if the user is lucky, no matter what questions they answer they will still end up gaining experience points that will eventually unlock all of the levels available in the game without ever actually learning anything. Such exploits could be detrimental to the success of the

application in the future. Bringing the minigame system that Duolingo uses into the Primary Mathletes system may be beneficial, this is where if a user gets three questions wrong in any given minigame they are automatically kicked out of the minigame with 0 experience points gained so users are forced to think about the answers they are giving and not just mindlessly pressing buttons until they unlock all of the available levels.

Another small feature that could be implemented but that would be very beneficial to teachers using the application as a supplemental teaching aid is seeding. This would involve giving out a seed for every "random" set of questions that could be copied and sent out to each student so that they would all receive the same set of random questions at the same time so that the teacher could observe whether the majority of their class has knowledge about recently studied aspects of the curriculum or whether the teacher needs to revisit these sections.

After conducting tests using a real student it was visible that not enough data was being shown collected for the progress page to be worthwhile. In future updates this will be changed, and teachers will have a lot more data so that they can monitor students performance. Things like what level students are currently on, what types of questions they continuously get wrong, etc.

7. Bibliography

1. Lydia Plowman, Christine Stephen, Joanna McPake. Growing Up With Technology: Young Children Learning in a Digital World. [E-Book]. Books.google.com. [cited 2020 Apr 10]. Available from: https://books.google.ie/books?hl=en&lr=&id=5HiMAgAAQBAJ&oi=fnd&pg=PP1&dq=young+children+learning+with+technology&ots=nHM82RW6Q4&sig=L3z8BRv0E9-EIDUScxEdbrQL6m4&redir_esc=y#v=onepage&q&f=false

2. Kai Erenli. The Impact of Gamification - Recommending Education Scenarios. [Internet]. Learntechlib.org. [cited 2020 Apr 10]. Available from: https://www.learntechlib.org/p/45224/

3. Emetere M. E, Oluwafemi T. J, Akinlusi T, Allen M. C, Ama E. C. Developing a Mathematical Mobile App: A Case Study of an Environmental Model [Internet]. Iaeng.org. [cited 2019 Dec 8]. Available from: http://www.iaeng.org/publication/WCECS2016/WCECS2016_pp150-153.pdf?fbclid=IwAR1iFguPdUgTeTq4pAUzfkYheh7g1EVxyUmrFUI8SE6hXfLr6E5DdhYSafg

4. Matthew Kearney, Damian Maher. MOBILE LEARNING IN MATHS TEACHER EDUCATION: USING IPADS TO SUPPORT PRE-SERVICE TEACHERS' PROFESSIONAL DEVELOPMENT [Internet]. Researchgate.net. [cited 2020 Apr 10]. Available from: https://www.researchgate.net/profile/Matthew_Kearney/publication/285751515_Mobile_learning_in_maths_teacher_education_Using_ipads_to_support_pre-service_teachers'_professional_development/links/568c55ca08ae153299b66628.pdf

5. Machdel Matthee, Jacobus. Mathematics on the Move: Supporting Mathematics Learners through Mobile Technology in South Africa. [Internet]. Researchgate.net. [cited 2020 Apr 10]. Available from: https://www.researchgate.net/profile/Machdel_Matthee/publication/238749401_Mathematics_on_the_Move_Supporting_Mathematics_Learners_through_Mobile_Technology_in_South/links/0a85e53c38801e8c9d000000/Mathematics-on-the-Move-Supporting-Mathematics-Learners-through-Mobile-Technology-in-South.pdf

6. Al-Zeidi, A, Al-Kindi, K, Al-Khanjari, Z. A. SQU Future: From E-Learning to M-Learning Application Development. [Internet]. Researchgate.net. [cited 2020 Apr 10]. Available from: https://www.researchgate.net/profile/Youcef_Baghdadi/publication/236310002_Web-enabled_services_development_with_respect_to_service-orientation_paradigm/links/5528c2640cf29b22c9bcb350.pdf#page=185

7. Department of Education and Skills. Mathematics Curriculum. [Internet]. [cited 2020 Apr 10]. Available from: https://www.curriculumonline.ie/getmedia/9df5f3c5-257b-471e-8d0f-f2cf059af941/PSEC02_Mathematics_Curriculum.pdf

8. Department of Education and Skills. Minister McHugh announces €50m in ICT grant funding for schools. [Internet]. Education.ie. [cited 2020 Apr 10]. Available from https://www.education.ie/en/Press-Events/Press-Releases/2020-press-releases/PR20-01-13.html

9. Kivy. Kivy Website. [Internet] KIvy.org. [cited 2020 Apr 10]. Available from: https://kivy.org/#home

10. Upenda Patrel. Why Firebase is the Best Mobile Backend as a Service. [Internet] tristatetechnology.com. [cited 2020 Apr 10]. Available from: https://www.tristatetechnology.com/blog/firebase-backend-mobile-app/

11. Liam Tung. Python developers now outnumber Java ones. [Internet] zdnet1.com. [cited 2020 Apr 10]. Available from: https://zdnet1.cbsistatic.com/hub/i/2019/04/15/cc0934f1-84ff-4d47-891a-91828e8e7307/slashdataprogramminglanguagesapr19.jpg

12. Medium Solutions. What Is Pycharm IDE. [Internet]. Medium.com. [cited 2020 Apr 10]. Available from: https://medium.com/@mindfiresolutions.usa/what-is-pycharm-ide-cc0735784f64

13. Collab. What Is Scrum. [Internet]. Collab.net. [cited 2020 Apr 10]. Available from: https://resources.collab.net/agile-101/what-is-scrum

14. Yodiz. What is a Sprint. [Internet]. Yodiz.com. [cited 2020 Apr 10]. Available from: https://yodiz.com/help/what-is-sprint/

15. Visual Paradigm. Why Fixed Length Sprints in Scrum? [Internet]. Visual-paradigm.com. [cited 2020 Apr 11]. Available from: https://www.visual-paradigm.com/scrum/why-fixed-length-of-sprints-in-scrum/

16. Atlassian. What is Agile? [Internet]. Atlassian.com. [cited 2020 Apr 10]. Available from: https://www.atlassian.com/agile

17.Dinnie Muslihat. Agile Methodology: An Overview. [Internet]. Zenkit.com [cited 2020 Apr 11]. Available from: https://zenkit.com/en/blog/agile-methodology-an-overview/

18. Anthony. Why Rounded Corners are Easier on the Eyes. [Internet]. Uxmovement.com [cited 2020 Apr 10]. Available from: https://uxmovement.com/thinking/why-rounded-corners-are-easier-on-the-eyes/

19. Kendra Cherry. Very Well Mind – What is Positive Reinforcement [Internet]. Verywellmind.com [cited 2020 Apr 10]. Available from: https://www.verywellmind.com/what-is-positive-reinforcement-2795412

20. Renk Etikisi. The Effect of Colors on Children. [Internet]. Renketkisi.com [cited 2020 Apr 6]. Available from: http://renketkisi.com/en/the-effects-of-colors-on-children.html

21. Azhar. How to Pick Colors for Your App Without a Struggle. [Internet] uxdesign.cc. [cited 2020 Apr 6] Available from: https://uxdesign.cc/how-to-pick-colors-for-your-app-without-a-struggle-bc46c5e19574

22. Nathan Madsen. Tips for Using Sound in Apps and Games. [Internet]. Premiumbeat.com. [cited 2020 Apr 10]. Available from: https://www.premiumbeat.com/blog/tips-for-using-sound-in-apps-and-games/

23. Guru99. What is White-Box Testing. [Internet]. Guru99.com. [cited 2020 Apr 10]. Available from: https://www.guru99.com/white-box-testing.html

24. Guru99. What is Black-Box Testing. [Internet]. Guru99.com. [cited 2020 Apr 10]. Available from: guru99.com/black-box-testing.html

8. Appendices

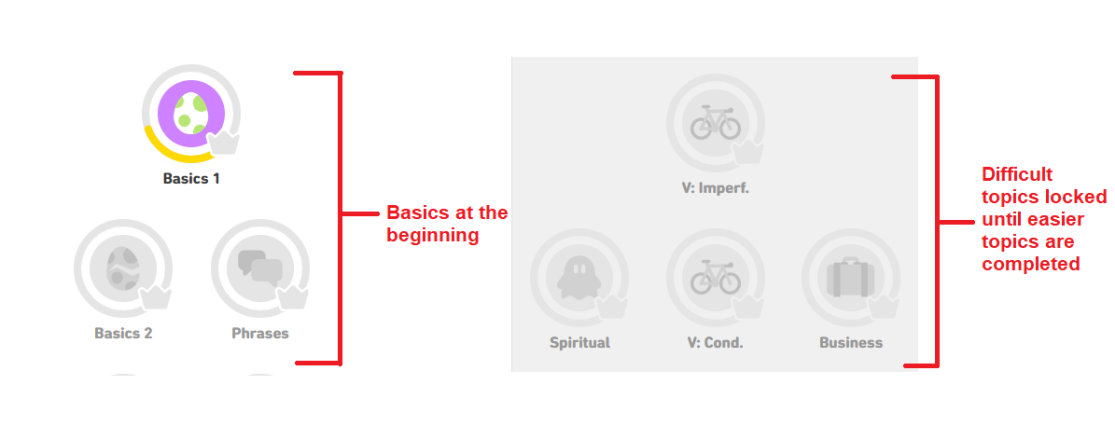
Primary School Maths Teaching Aid

The idea:

I'm making an app and writing a dissertation for my final year in college and I need to do some background research before I get going so I was just wondering if you could take a look at this and let me know if you have any suggestions, comments or ideas about anything I have so far!

The app, hopefully, could be used as a teaching aid for primary school teachers to use when going through the math curriculum. The idea behind this is that every student goes at their own pace and so if the teacher moves on too fast for some students or is going to slow for other students that they can use this app to further their education one way or another.

This app will be similar to Duolingo (but for maths obviously) in the sense that there is a hierarchy of topics/pathways. They will have to complete earlier tasks before they can move on to the more difficult ones.



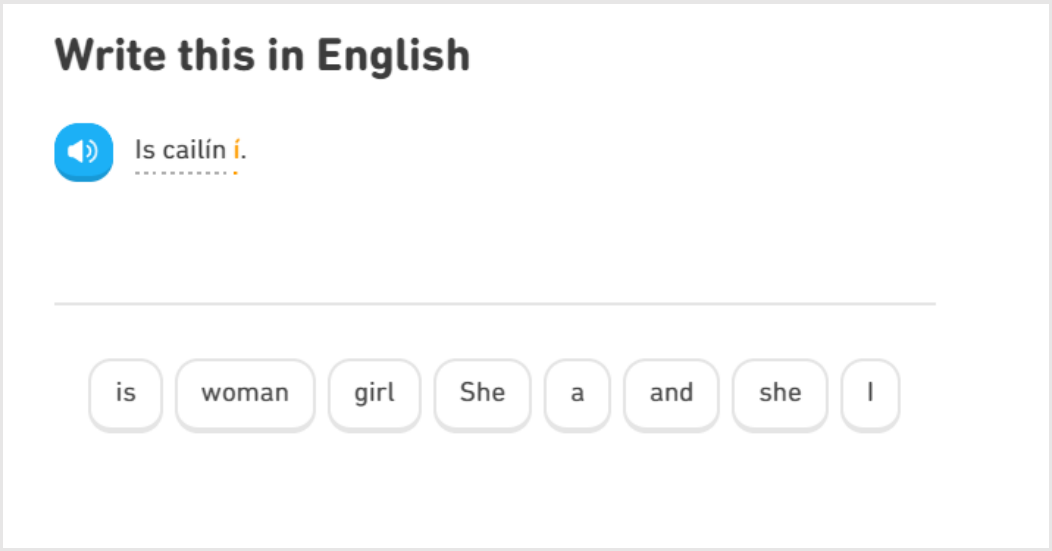
Hierarchal System in Duolingo

Below is an example of a question in Duolingo.
The app I'm developing will follow a similar structure but with basic equations.

For example it could be something like:
[ ] + 7 = 10

[1] [2] [3] [4]

Choose the correct number to complete the equation.



Example of Duolingo Questions

The students will progress through levels, making mistakes and getting correct answers and at the end of the week the app will send the parent, teacher or both an easy-to-read report listing all of the progress the student has made and all of the mistakes that they have made so that revision can be done either in class or at home with a parent depending on where the teacher is in the curriculum.

Your Input:

1)[a] Do you think the app could be useful for you as a primary teacher?
[b] Would students use it?

2) Do you think there are any problems with the idea and if so what do you think could be done to correct them?

3) Do you know of any resources (websites, books or both) that I could look into to access the curriculum for primary school maths?

4) What form of device are primary school students most comfortable with? (e.g Desktop Computer, Laptop, iPad/Tablet, mobile phone)

5) Off the top of your head do you think there is any other kind of minigame-esque questions that would be helpful (and hopefully fun) to engage students more with the app?

6) [a] Do you think having a section of the app dedicated to times tables would be worthwhile creating?
[b] Is there anything else in the same vein as this idea that you think could be essential to have in the app?

7) Is there a specific age the app should NOT be aimed at (junior, senior, 1st class)? Is there any point in targeting students this young?

8) Do you have any suggestions for the name of this app?

9) Can you think of any ideas that would keep the students more engaged in the app?

10) Are there any common difficulties students have with maths that I could address with the app?

Other Suggestions & Additional Feedback:

Appendix 1: Questionnaire Sent to Primary School Teachers

Your Input:

1) [a] Do you think the app could be useful for you as a primary teacher?

YES

[b] Would students use it?

Yes there are many ipads/laptops available for use in many schools now

2) Do you think there are any problems with the idea and if so what do you think could be done to correct them?

No, it would be beneficial for assessment too

3) Do you know of any resources (websites, books or both) that I could look into to access the curriculum for primary school maths?

Twinkl, transum maths, folens online (set up free student account)

4) What form of device are primary school students most comfortable with? (e.g Desktop Computer, Laptop, iPad/Tablet, mobile phone)

Ipads and tablets

5) Off the top of your head do you think there is any other kind of minigame-esque questions that would be helpful (and hopefully fun) to engage students more with the app?

No

6) [a] Do you think having a section of the app dedicated to times tables would be worthwhile creating?

Yes, tables are extremely important for pupils to revise as it comes up in many areas of maths including daily mental maths

[b] Is there anything else in the same vein as this idea that you think could be essential to have in the app?

Notifications in the app to remind pupils of the maths rules and rhymes (e.g for subtraction "more on the floor go next door")

7) Is there a specific age the app should NOT be aimed at (junior, senior, 1st class)? Is there any point in targeting students this young?

In my opinion I would aim for 2nd and above. Infants and 1st class would use a lot more hands on and concrete resources and the app may be too advanced

8) Do you have any suggestions for the name of this app?

"Mathlete" "maths master"

9) Can you think of any ideas that would keep the students more engaged in the app?

Visuals – rewards – unlocking new levels – positive praise

10) Are there any common difficulties students have with maths that I could address with the app?

Tables – long multiplication and long division

Other Suggestions & Additional Feedback:

Good Luck !

Appendix 2: Teacher Questionnaire Response 1

Your Input:
1. [a] Do you think the app could be useful for you as a primary teacher?
[b] Would students use it?
It could be extremely useful and could benefit students. Students love online interactive apps, this wouldn't seem like work to them.

2. Do you think there are any problems with the idea and if so what do you think could be done to correct them?
Not every child in the class will be provided with an iPad at all times to use the app. If each class in the school were to use the app every day, there wouldn't be resources for them to use it. However, it could be an excellent resource in SEN for learning support or resource teachers.

3. Do you know of any resources (websites, books or both) that I could look into to access the curriculum for primary school maths?
NCCA Mathematics curriculum online, NRich maths online, Mental Maths (combine all curriculum strands to make quizzes).

4. What form of device are primary school students most comfortable with? (e.g Desktop Computer, Laptop, iPad/Tablet, mobile phone)
IPads are usually available in most schools, there are usually a set of laptops in schools. However, there won't be enough to provide every child with one.

5. Off the top of your head do you think there is any other kind of minigame-esque questions that would be helpful (and hopefully fun) to engage students more with the app?
NRich maths have brilliant problem solving questions that I use to introduce my lessons and recap on a topic. NZ maths can be useful too.

6. [a] Do you think having a section of the app dedicated to times tables would be worthwhile creating?
[b] Is there anything else in the same vein as this idea that you think could be essential to have in the app?
Yes, this could work well.

7. Is there a specific age the app should NOT be aimed at (junior, senior, 1 st class)? Is there any point in targeting students this young?
Yes definitely, they learn so much from a young age and early ed maths education is such a huge aspect to their school day. It would be important to note what they are learning in maths at this age, e.g. they learn to number 5 in junior infants and 10 in senior infants, they learn to understand that the numbers come after each other and each number has a value, this could be a huge part to the app. Younger children would react best to online games and it might be worth your while looking at Bee Bots, which are used in junior classes to teach directions.

8. Do you have any suggestions for the name of this app?

9. Can you think of any ideas that would keep the students more engaged in the app?
Setting timers, keeping them on task and feeling that they are achieving something. Giving points or positive reinforcements when they achieve a goal.

10. Are there any common difficulties students have with maths that I could address with the app?

Other Suggestions & Additional Feedback:

Appendix 3: Teacher Questionnaire Response 2

Your Input:

1) [a] Do you think the app could be useful for you as a primary teacher?
[b] Would students use it?

I do believe that the app could be useful for a primary teacher. Especially for children who finish work quickly it would allow them to have an add on activity to push themselves further in their learning.

I think students would use it. It would be something different and new for them which makes it exciting.

2) Do you think there are any problems with the idea and if so what do you think could be done to correct them?

I think that creating an app like this there would be a lot of concern from parents in regard to how the children are learning and monitoring their work on the app.

3) Do you know of any resources (websites, books or both) that I could look into to access the curriculum for primary school maths?

Curriculum online

Haylock maths for primary teachers

Folens online (access to planet maths ebook& resources)

4) What form of device are primary school students most comfortable with? (e.g Desktop Computer, Laptop, iPad/Tablet, mobile phone)

Ipad/tablet or phone

5) Off the top of your head do you think there is any other kind of minigame-esque questions that would be helpful (and hopefully fun) to engage students more with the app?

Using pictures, odd one out games, matching games, students will get bored if similar sums are constantly appearing in front of them.

6) [a] Do you think having a section of the app dedicated to times tables would be worthwhile creating?
[b] Is there anything else in the same vein as this idea that you think could be essential to have in the app?

Yes I believe it would be helpful, especially if they could time themselves answering and try and beat their original time, I think breaking down the curriculum into different sections would be helpful as it would give the children a chance to master each of the curriculum skills required.

7) Is there a specific age the app should NOT be aimed at (junior, senior, 1st class)? Is there any point in targeting students this young?

I believe that all the students would benefit, it would take a long time to teach them exactly how to use the app etc.

8) Do you have any suggestions for the name of this app?

Other Suggestions & Additional Feedback:

I think that it's a really worthwhile app and it would help the teachers differentiate their lessons, different settings could give children the chance to catch up with fellow classmates if struggling and the chance for more able students to challenge themselves further.

Appendix 3: Teacher Questionnaire Response 3

Your Input:

1) [a] Do you think the app could be useful for you as a primary teacher? Yes
[b] Would students use it? Yes

2) Do you think there are any problems with the idea and if so what do you think could be done to correct them?

Will the app work without internet access

3) Do you know of any resources (websites, books or both) that I could look into to access the curriculum for primary school maths?

www.nccaplanning.ie – online access to the aims/objectives of the maths curriculum for each class level

www.ixl.ie – maths games linked to the Irish primary maths curriculum (check out some ideas for questions for your own app)

Maths textbooks – Busy at Maths Programme by CJ Fallon (Teacher books contain clear aims & objectives for each topic & class level)

4) What form of device are primary school students most comfortable with? (e.g Desktop Computer, Laptop, iPad/Tablet, mobile phone)

Very dependant on the school – desktop computers, laptops & tablets

5) Off the top of your head do you think there is any other kind of minigame-esque questions that would be helpful (and hopefully fun) to engage students more with the app?

Extra challenges/points/beat the clock activities

6) [a] Do you think having a section of the app dedicated to times tables would be worthwhile creating? Definitely!

[b] Is there anything else in the same vein as this idea that you think could be essential to have in the app?

Incorporate word problems linking to the topic

7) Is there a specific age the app should NOT be aimed at (junior, senior, 1st class)? Is there any point in targeting students this young?

Possibly target 1st – 6th class children (focus on topics from 1st – 6th class)

8) Do you have any suggestions for the name of this app?

9) Can you think of any ideas that would keep the students more engaged in the app?

Let the children design an icon for their profile

Let the children see a graph showing their own progress

10) Are there any common difficulties students have with maths that I could address with the app?

Understanding word problems

Mathematical language specific to each topic

Other Suggestions & Additional Feedback:

Sounds like a great idea!

Appendix 5: Teacher Questionnaire Response 4

- How to run the application on Windows –

Prerequisites: Python 3.7

1- Download and install PyCharm for Windows using the steps provided here:
https://www.jetbrains.com/help/pycharm/installation-guide.html

2- Download Kivy 1.9.1 using the steps provided here:
https://kivy.org/doc/stable/installation/installation-windows.html

3- Download and unzip the project folder from: https://github.com/paulpdavis3/FYP

4- Create a new project in PyCharm and add all of the downloaded files into it.

5- Go to File -> Settings -> Project -> Project Interpreter

6- Click the settings symbol to the right of the project interpreter drop down and select Python 3.7

7- Press the green arrow to run the app

Appendix 6: Instructions on How to Run Application