

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**

**\*\*\*DATE\*\*\***

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 mathematics 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 once they fall behind.   
  
During the implementation of this project, research into the primary school curriculum through books 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 books and online resources to create a fun, interactive and informative application that will aid both students and teachers in primary schools.

The project was initially 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 seeing is there any improvement 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 its 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

\*\*\*DATE\*\*\*

Acknowledgements

Body text

Table of Contents

[1. Introduction 6](#_Toc37180996)

[1.1. Project Background 6](#_Toc37180997)

[1.2. Project Description 7](#_Toc37180998)

[1.3. Project Aims and Objectives 8](#_Toc37180999)

[1.4. Project Scope 9](#_Toc37181000)

[1.5. Thesis Roadmap 9](#_Toc37181001)

[2. Literature Review 10](#_Toc37181002)

[2.1. Introduction 10](#_Toc37181003)

[2.2. Alternative Existing Solutions to The Problem 10](#_Toc37181004)

[2.3. Technologies Researched 13](#_Toc37181005)

[2.4. Technologies Used 14](#_Toc37181006)

[2.5. Other Research Conducted 17](#_Toc37181007)

[2.6. Existing Final Year Projects 20](#_Toc37181008)

[2.7. Conclusions 20](#_Toc37181009)

[3. Experiment Design 21](#_Toc37181010)

[3.1 Introduction 21](#_Toc37181011)

[3.2. Software Methodology 21](#_Toc37181012)

[3.3. Overview of System 25](#_Toc37181013)

[3.4. Software Test plan 26](#_Toc37181014)

[3.5. Front-End 27](#_Toc37181015)

[3.6. Middle-Tier 43](#_Toc37181016)

[3.7. Back-End 43](#_Toc37181017)

[3.8. Conclusions 44](#_Toc37181018)

[4. Experiment Development 45](#_Toc37181019)

[4.1. Introduction 45](#_Toc37181020)

[4.2. Software Development 45](#_Toc37181021)

[4.3. Front-End 45](#_Toc37181022)

[4.4. Middle-Tier 48](#_Toc37181023)

[4.5. Back-End 61](#_Toc37181024)

[4.6. Conclusions 64](#_Toc37181025)

[5. Evaluation 64](#_Toc37181026)

[5.1. Introduction 64](#_Toc37181027)

[5.2. Software Evaluation 64](#_Toc37181028)

[5.3. Specific Evaluation 64](#_Toc37181029)

[5.4. Questionnaires and Interviews Evaluation 64](#_Toc37181030)

[5.5. Conclusions 64](#_Toc37181031)

[6. Conclusions and Future Work 66](#_Toc37181032)

[6.1. Introduction 66](#_Toc37181033)

[6.2. Conclusions 66](#_Toc37181034)

[6.3. Future Work 66](#_Toc37181035)

[7. Bibliography 67](#_Toc37181036)

[[1] 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: 67](#_Toc37181037)

[http://www.iaeng.org/publication/WCECS2016/WCECS2016\_pp150-153.pdf?fbclid=IwAR1iFguPdUgTeTq4pAUzfkYheh7g1EVxyUmrFUI8SE6hXfLr6E5DdhYSafg [Accessed 8 Dec. 2019]. 67](#_Toc37181038)

# 1. Introduction

## Project Background

Many sources state that children, even from young ages, can learn a lot by using technology and can reinforce information already covered in their school curriculums through gamification in apps on phones or tablets. 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”.* [1] 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. It found that *“pre-service and in-service teachers saw value in integrating iPads into Maths education as a tool to promote student learning”* [2] 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 PC’s in many homes but yet there are *“three million teenagers have Java enabled cell phones”*  [3] which would strongly benefit from mobile learning or m-learning applications due to the inaccessibility of web apps in that region of the world.

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 PC’s because almost everyone nowadays already owns a mobile phone. This cuts the cost of textbooks and on building expensive computer rooms for students. [4]

## 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 off the third class maths curriculum focusing on simple operators: addition, subtraction, multiplication and division, all following the learning outcomes put in place by the Irish Department of Education.

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 then join and the teacher can then view all of their students weekly progress in one place. 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.

Following a similar structure to Duolingo (shown in figure \*\*\*\*), an application some primary students may already be familiar with, students will earn experience points for whatever operator they are currently playing in and through these points the student will unlock increasingly more difficult levels relating to the current operator. The students will be unable to play more difficult levels until they have first completed all levels leading up to the one they are attempting to complete to ensure that they 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.

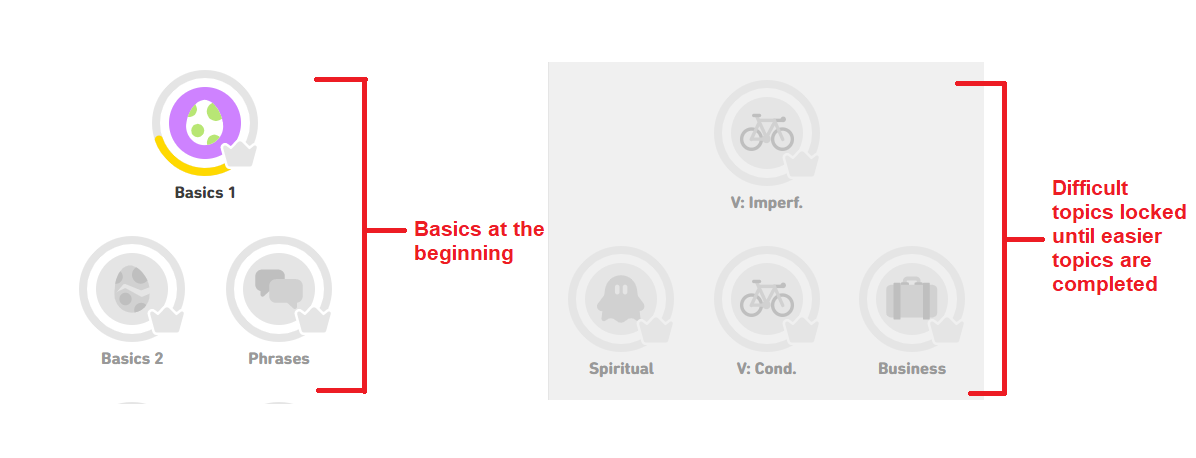


Figure 2 Comparison of easy and difficult topics on Duolingo

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. Unfortunately for reasons expanded upon later in the document, it is currently only possible to run the mobile application on both android and iOS emulators and not physically on mobile devices (**see section \*\*\*\***).

## 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 will include minigames and helpful maths-related topics that will 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 6). 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.

## Project Scope

Maths is 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. The student either knows 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 sub section 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 actually 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 really 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.

## 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 as well as why these other applications are not the be-all and end-all solution to the problem that this project is trying to solve. Additional research on exactly what students and teachers want and need from an app such as this will be conducted. Technologies researched and also a review of two previous computer science dissertations will be created.

## 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.[12] The free content that is available 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.

|  |  |  |
| --- | --- | --- |
| Figure AB Maths minigame | Figure Minigame paper prototype | Figure Minigame wireframe |

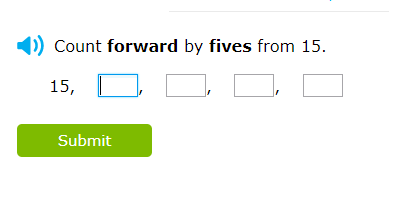
**Mathematics Skill Builder Primary for iOS:**

Opposite to AB Maths mentioned above, this app actually was developed for primary school maths students, but it is very outdated and cartoonish.[13] It also seems to deal with topics that could be deemed too easy for primary school students. Examples of this 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.

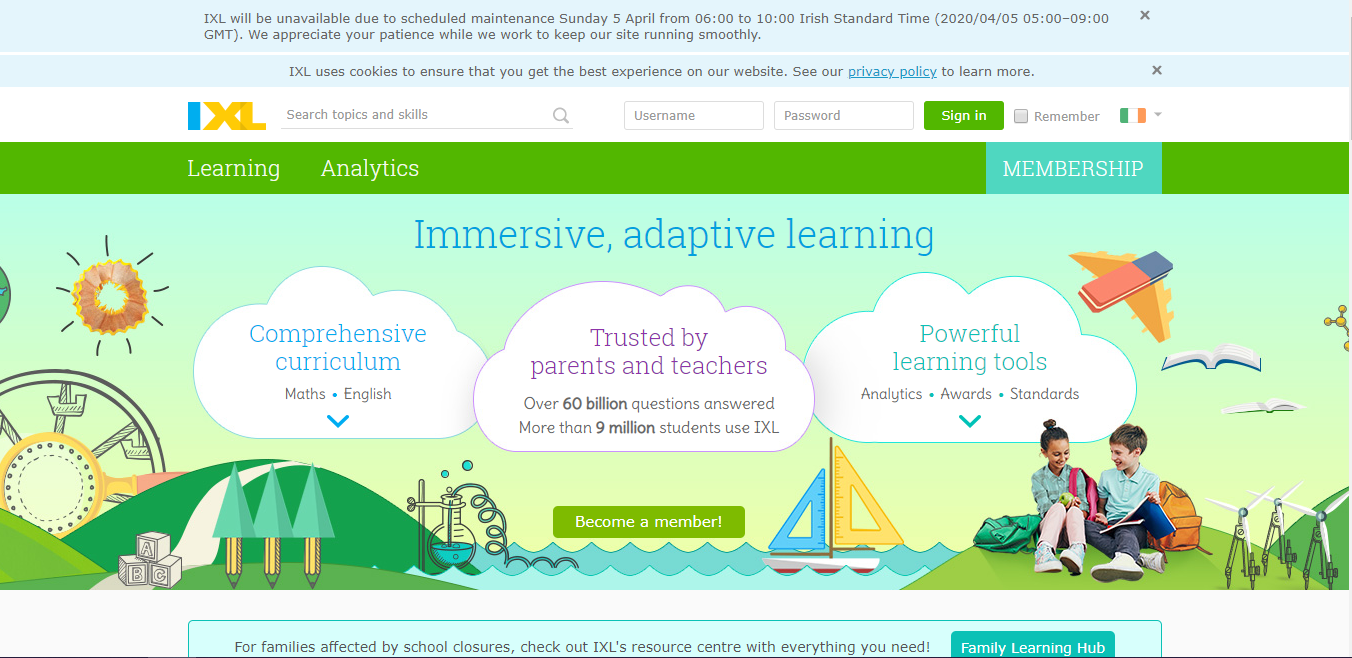
|  |  |
| --- | --- |
|  |  |
| *Figure #: Mathematics skill builders primary homepage* | *Figure #: Skill builder primary basic question example* |

**ixl.ie website:**

This website has hundreds of different quizzes that are developed around the different years of curriculum for Irish primary school students.[14] The screenshot in figure **\*\*FIND FIGURE NUMBER\*** shows a nice, clean minigame found on the website but looking at the screenshot in figure **\*FIND FIGURE NUMBER \*** can seem like a bit of a sensory overload with all of the text 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 #: ixl example question*



*Figure #: IXL Home Page*

All of the apps and websites mentioned above are definitely in the same vein as the Primary Mathletes application created for this project. Whether 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 were looked at 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 almost never enough working desktop computers available for primary school students in their schools but most schools nowadays have access to tablets whether they’re Android or iOS.[15][16][17][18]

Some research also went into using Java [19] and the Android Studios IDE [20] because in 3rd year of computer science there is a module on mobile app development using both of these 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, there are many modules that 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.

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. [21] 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, it doesn’t need to be their own personal device and also makes things like progress analytics easier for students.

Some research also went in to 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.

**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 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.[]

**JSFiddle.net:**

After working outside of college on a different project I became quite confident in my JavaScript programming and so I used JSFiddle to create the first draft of the algorithm that I would later use in the minigame portion of the project. 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 back to a previous version of the code that I was confident that worked and also in case my laptop crashed and I lost my files I could pull them from Github from another device but thankfully this didn’t happen.[]

**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: calculator, windows media player etc.

**Airtable:**

Late in the development of the application and before I started testing, I began using 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 this 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 off all of 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 face images, the logo, back buttons etc.

**Audacity:**

This program was used to edit and change file formats of the sound files for 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 with 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.

**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 actually necessary for this application to be successful. It was also used later on in the testing phase that will be discussed further on in the document.

**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 usable 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. [22]

- 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. [23]

- NCCA Maths Development: This website provides information on the development of the new Primary Mathematics Curriculum. [24]

- 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. [25]

Below is a short document that I 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 can also be seen below:

**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 devices 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 to have 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?**

A few days after this questionnaire was sent to the teachers, responses began coming back and the collective document below was created. It gathers all the key wants and needs that an application like the one developed for this project should have based on the teacher's feedback:  
  
**Things noted after speaking to primary school teachers about the app:  
- Most primary school students nowadays are happy to use tablets/iPads. (steering away from desktop computers and laptops depending on schools)  
- Most primary schools have poor internet connection so the ability to use the app without an internet connection is preferred but not essential.  
- The app could be used for assessment as a group in the classroom after finishing a section of the curriculum. (Work on seeding so every student in the class gets the same exam questions)  
- Primary school curriculum:**

**• Twinkl  
• Transum Maths  
• Curriculum online [26]  
• Haylock maths for primary teachers [27]  
• Folens online [28]  
• SEN [29]  
• NCCA Mathematics curriculum online  
• Mental Maths [30]  
• NZ maths [31]  
• Bee Bots (for teaching directions) [32]**

**- Maths Tables section of the app would be good.  
- Tooltips / Glossary for what certain keywords mean.  
- App should be aimed for 2nd class and over but some have said even junior infants would benefit.  
- To help with engagement:**

**• Bright/contrasting colours  
• Badges  
• Hierarchical system that gets more difficult as you go along  
• Positive praise  
• Good sound design for correct and incorrect answers   
• Pictures  
• Points system  
• Allow them to create their own avatar/upload an image**

**- The app would benefit teachers in terms of more advanced children, the teacher can assign additional work for students that are abler when it comes to maths.  
- It would also allow teachers to assign additional work to students who are struggling so that they could catch up.  
- The addition of an app into the student’s life would be a breath of fresh air and help with engagement in class.  
- Mini-game ideas:**

**• Odd-One-Out  
• Matching Games  
• Time trial mode / Beat the clock  
• NRich maths online (problem-solving questions)  
• ixl.ie (online quizzes for Irish maths curriculum)  
• Word problems**

**- Not every student would be provided an iPad/tablet to use on their own so group activities might be necessary, but this is an issue for the school rather than for the development of the app.  
- Combine all curriculum strands to make quizzes at the end of each section.  
- Users should be able to see a graph of the user's progress at the end of the week/month/year/all time.**

All of this information 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

## 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. Experiment Design

## 3.1 Introduction

In this section, some common software methodologies are discussed, followed by a description of the one finally decided upon and why and how this decision was made. There is also a thorough overview of the system and an explanation of the front, middle and back tier of the three-tier system implemented through the design of this project.

## 3.2. Software Methodology

Scrum –

Scrum [33] is a methodology that is comprised of five values: commitment, courage, focus, openness, and respect. It usually involves someone taking over the role of scrum master who organizes daily stand-ups and points out places in the development team 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 of time. 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 than the last. Scrum is a methodology that is usually used when working with a team on a project but due to the fact that this is a solo project, it did not seem like it would really be a suitable choice for this project.

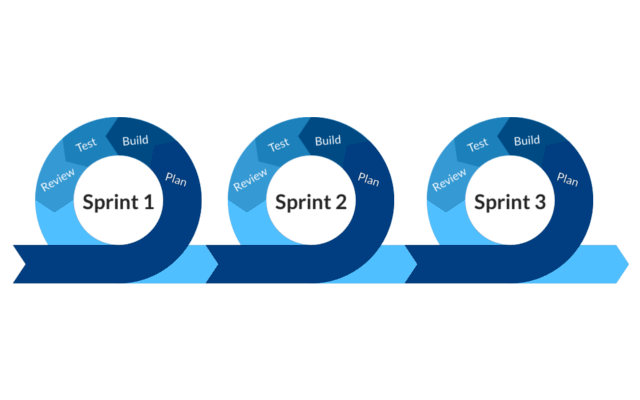


Figure 7 Sprints lifecycle as part of the Scrum Methodology

Agile –

Agile [34] 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. 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.

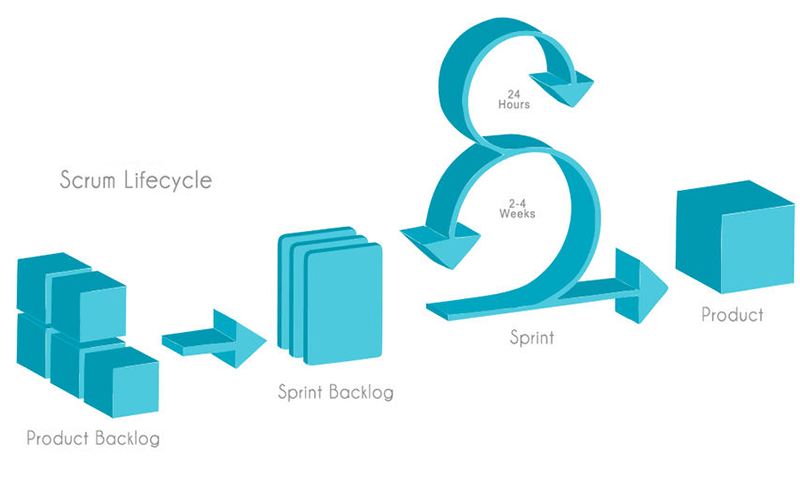


Figure 8 Scrum lifecycle as part of the Agile Methodology

Kanban Methodology –

A Kanban board is one version of the Kanban project management methodology [35] 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 of time 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 [36] 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 it was a programming task or just a researching or writing task.

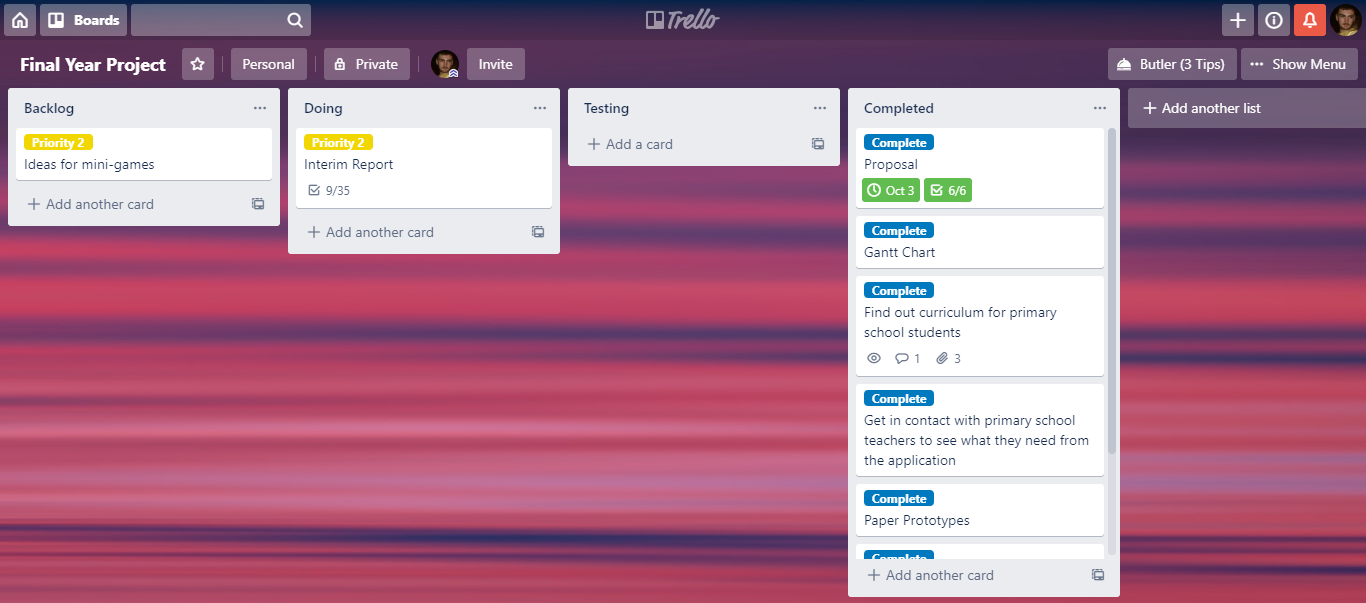


Figure 9 Project Trello board

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 like you see in the image below but I also have cards where I have added useful links to pages I have researched added to the card themselves, as well as comments and descriptions that would aid me in completing the task to the best of my ability.  
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 all be grouped into one card anyway.

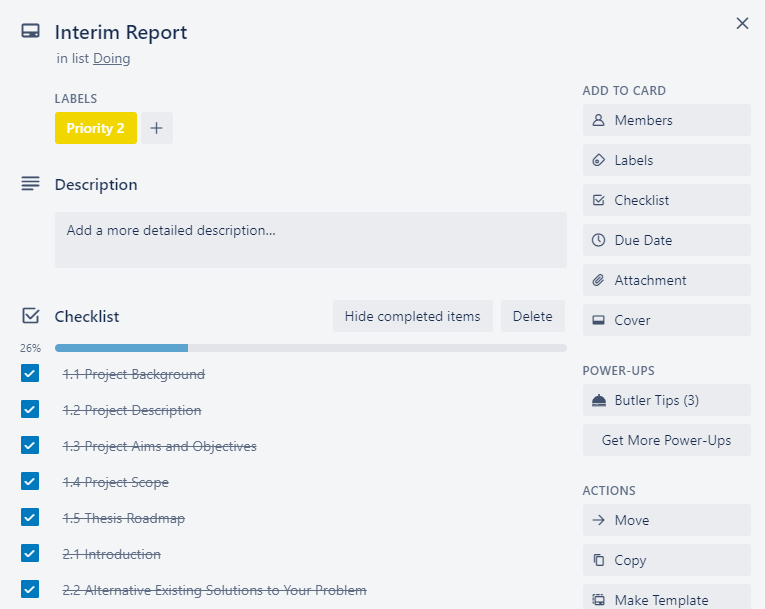
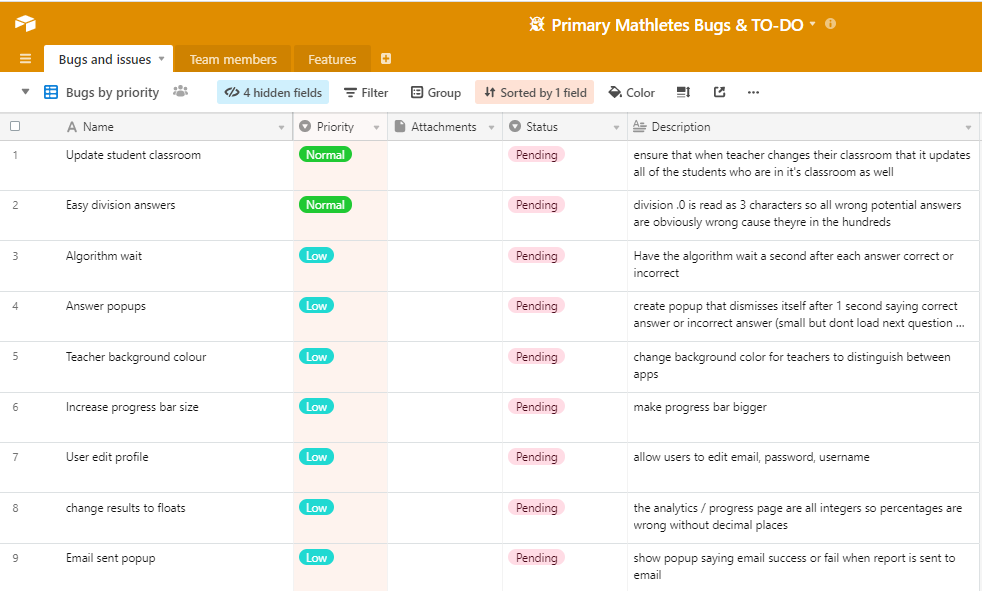


Figure 10 Card details for Interim Report

**Airtable:**

Airtable is another example of a Kanban board and the one seen in Figure \*\*\*\* below shows the one that I used as a bug list while I was personally testing that everything worked after initial development was finished on the app.



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 then to use Airtable as another means of organization to hold all the information I had for bugs and small programming tasks that had not been implemented yet.

## 3.3. Overview of System

In figure \*\*\*\* you can see the proposed three-tier system diagram that was created earlier in the academic year. It includes both android and iOS mobile devices and in figure \*\*\* you can see 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 these 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 in all of the operators, 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.

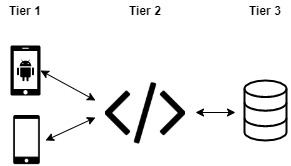


Figure 11 Three-Tier System

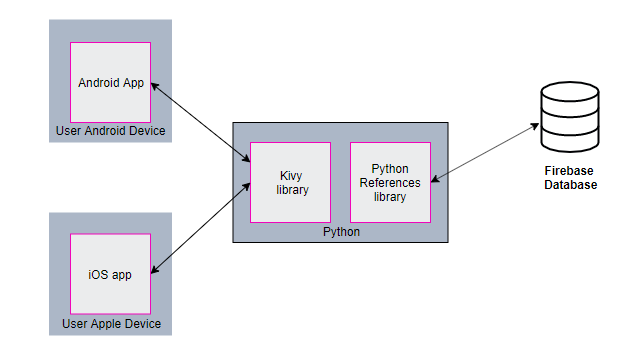


Figure 12 Technical Architecture Diagram

## 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 conversations and run throughs of the application with my family. Luckily one of which, my younger sister, is a primary school student themselves, so this was a big plus for this portion of the testing.

**White Box Testing:**

This was conducted by myself and this was made easier by the fact that Kivy offers unit testing for both graphical and non-graphical aspects of the library and this will be discussed further in section 5.

It is unfortunate that the original plan of using actual groups of primary students was not possible because 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 color scheme, font choices and other aspects of the application that are discussed further in this section.

## 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 2.5. and this helped with moving forward in development as I received input from the target audience of the application.

**Register Page:**

As seen in figures x,x and x 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 #: Register Paper Prototype |
| |  |  | | --- | --- | |  |  | |
| Figure #: Register Wireframe Prototypes |
|  |
| Figure #: 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 x,x and x 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 #: Login Paper Prototype** |
|  |
| **Figure #: Login Wireframe Prototype** |
|  |
| **Figure #: Final App Login Page** |

**Main Menu:**

In figures x, x and x 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 x 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 coulduse 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.

|  |
| --- |
|  |
| **Figure #: Main Menu Paper Protoype** |

|  |
| --- |
|  |
| Figure #: Main Menu Wireframe Prototype |

|  |
| --- |
|  |
| Figure #: Final App Main Menu |



Figure #: Duolingo Level Tree

**Minigame Page:**

The evolution of the minigame page changed slightly over the course of development. Figures x and x show that the paper prototype and the final version of the minigame page aren’t too different. 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 as well as the addidion of a 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 aks whether they would like to leave the current minigame or not. Figure x shows a more detailed wireframe prototype that shows an exit button in the opposite corner that is now found in the final version of the app as well as 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 #: Minigame Paper Prototype | Figure #: Minigame Wireframe Prototype |

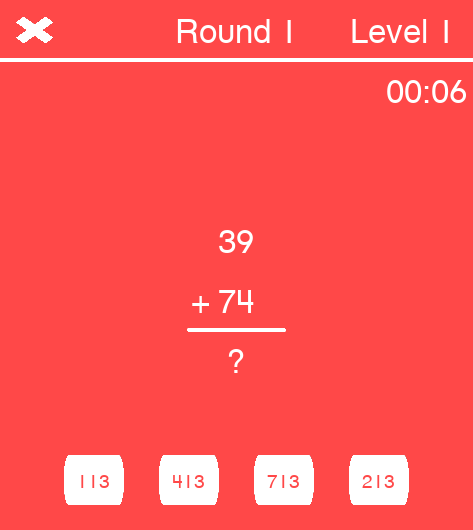


Figure #: Final App Minigame Page

**Results Page:**

Early in development the results page was called the finish page and there was a drastic change between the original versions seen in figure x and x and the final app version seen in figure x. Originally the results page was going to have very minimal amount of information on it as well as displaying which questions the user actually got correct and incorrect. After the number of questions per minigame were 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 for the operator they were currently playing minigames within. 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 #: Results Paper Prototype | Figure #: Results Wireframe Prototype |

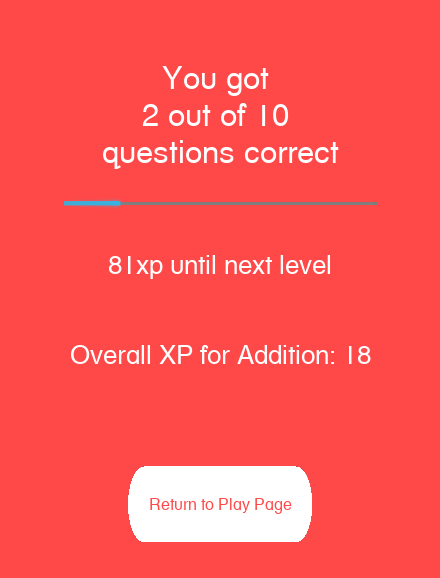


Figure #: Final App Results Page

**Progress Page:**

This page probably changed the most over the course of 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 and something text based like the last two versions seen in figures x and x 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 run down 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 really serve any purpose. A better and more complex version of the progress page was then proposed and created and that is the final version seen in figure x. 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 are able to see their total play time 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 actually see if progress is being made or not which was not possible in previous iterations of the progress page. 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 #: Progress Paper Prototype | Figure #: Progress Wireframe Prototype |

|  |  |
| --- | --- |
|  |  |
| Figure #: Old version of Progress Page | Figure #: Final App Progress Page |

**Other screens:**

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

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

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

- Figure x 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 is gained. Level 2 unlocks at 100xp Level 3 unlocks at 200xp etc.

- Figure x is what the teacher sees when they enter the progress screen. It shows all of the students that are in that teachers 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 x shows the edit profile page. The final version of this page currently only has the ability to change the user’s username or delete their account but in future updates to the application there may be an option to set a profile picture, edit color scheme of the app, update email address etc.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Figure #: Title Page | Figure #: Classroom Page | Figure #: Play Page |
|  |  |  |
| Figure #: Level Selection Page | Figure #: Teacher’s Student List | Figure #: Edit Profile Page |

**Font:**

The font chosen for the text found inside the application was decided on late into development because I wanted the different pages to all be connected so the different font choices could be seen 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 on which font they liked most and the results of these questions showed that people older than the age of 12 actually liked the default Kivy font the most for its simple and stylish design that can be seen in figure x but interestingly, people 12 and younger liked the font seen in the final version of the application and the reason behind this will be discussed in section 5.

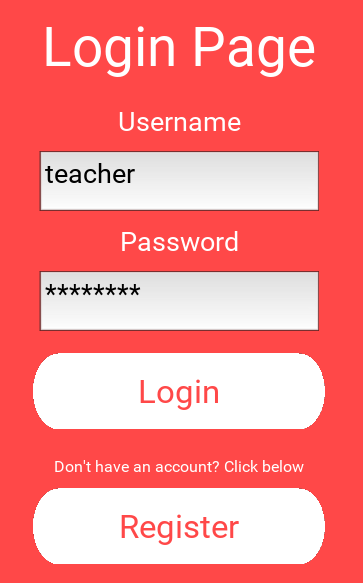


Figure #: Login Page using Kivy Default Font

**Button Design:**

After looking into other apps and reading articles about why humans like rounded buttons more than sharp-edged rectangular buttons I decided to design my own button and replace the default kivy button. [] Comparing figure x to figure x it’s clear to see why rounded buttons were chosen. They’re softer on the eyes and bring the focus inside the button. On screens like the level selection 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 #: Default Kivy Button | Figure #: Custom Rounded Button |

There were other custom buttons designed like the ‘X’ button found in the minigame page (figure x) 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 individual icon meant but they could definitely 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 in figures x – x.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Figure #: ‘X’ Button | Figure #: Unused Back Button | Figure #: Unused Log Out Button | Figure #: Final App Button Example |

**Transitions Between Screens:**

Figure x shows all of the applications screens and how they interact with each other.

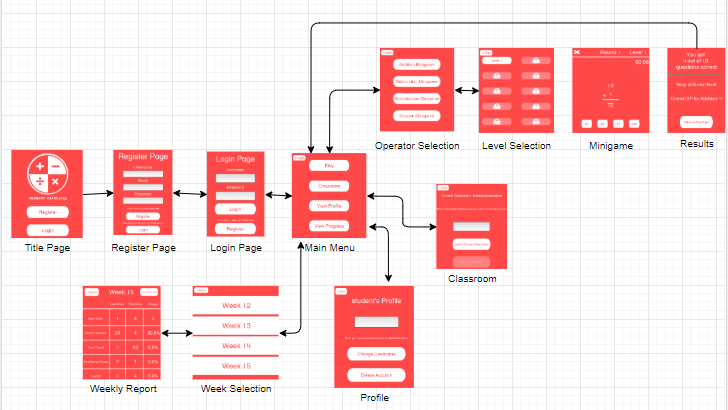


Figure #: Screen Flow of App

**Popup Design:**

Popups are used in various places within the app to ensure that entered information is correct and screen transitions are intentional.   
Figure x shows a popup shown to the user if they don’t input a suitably strong password when registering an account. Figure x 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 log out button by accident.

|  |  |
| --- | --- |
|  |  |
| Figure #: Unsuitable Password Popup | Figure #: Log Out Popup |

Popups shown in figures x and x 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. This idea came from inspiration after reading an article on positive reinforcement and how to encourage students.[] 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 #: Correct Answer Popup | Figure #: Incorrect Answer Popup |

There are other popups shown to the user throughout the application for reasons like: a username or email address was previously used for a different account and so the user must choose a different one when registering.

**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. [] 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. To solidify their learning of a particular topic.

**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.

**Use Cases:**

The five 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 log in, register and log out (figure x). After this, in the second iteration (figure x), 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, 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.

Teachers on the other hand, for the second iteration (figure x), 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 (figure x).

|  |  |
| --- | --- |
|  |  |
| Figure First iteration use case for both students and teachers | Figure Second iteration use case for students |

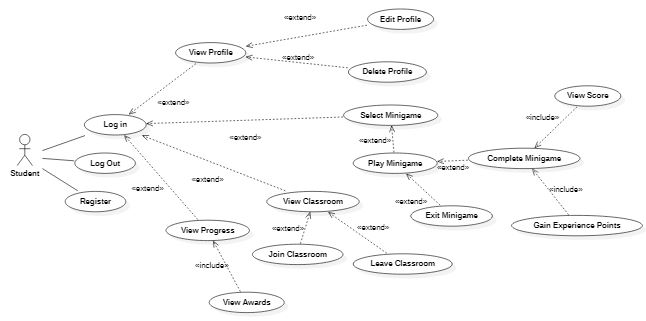


Figure 15 Third iteration use case for students

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 to register, log in 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 x. 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.

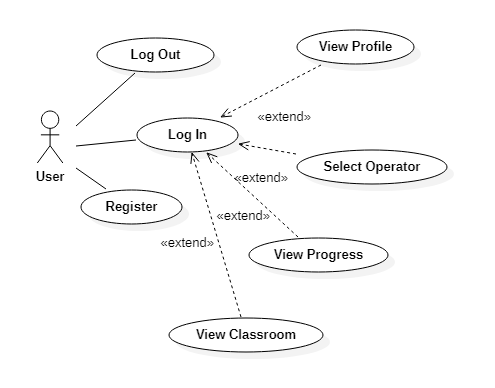


Figure #: Final Second Iteration for User and Teacher

For the third iteration of the use cases, the students and teachers part ways in some respects in order to give each of them individual control of their own version of the app. In their third iteration, students have the ability to 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 analysis their progress from the previous week and send themselves an email of their report. Originally students were going to be rewarded with badges/icons (seen in figure x OLD student third iteration) 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.

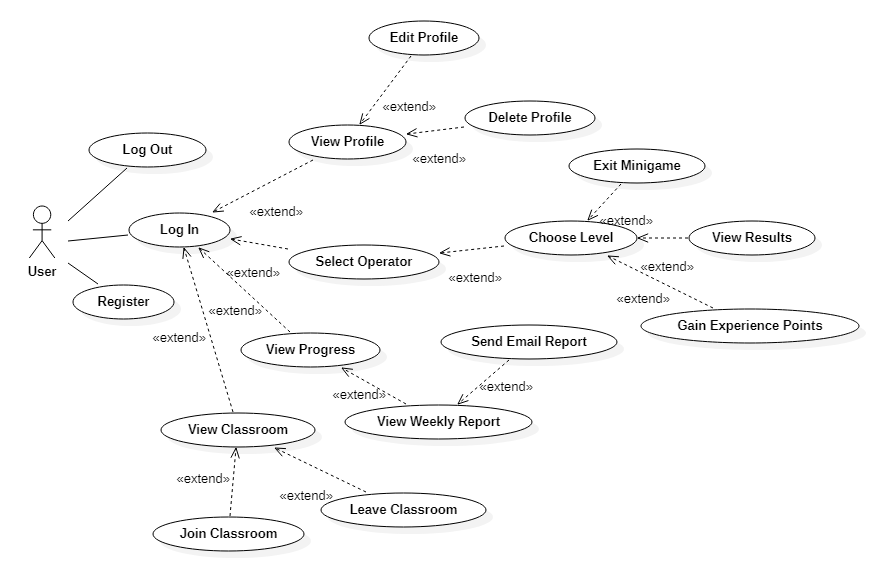


Figure #: Third Iteration User Case for Students

Looking at the teachers third iteration use case in figure x shows that teachers now have the ability to 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 teachers virtual classroom. Teachers also have the ability to create and delete classrooms and remove unwanted students from their classroom rather than leave and join classrooms.

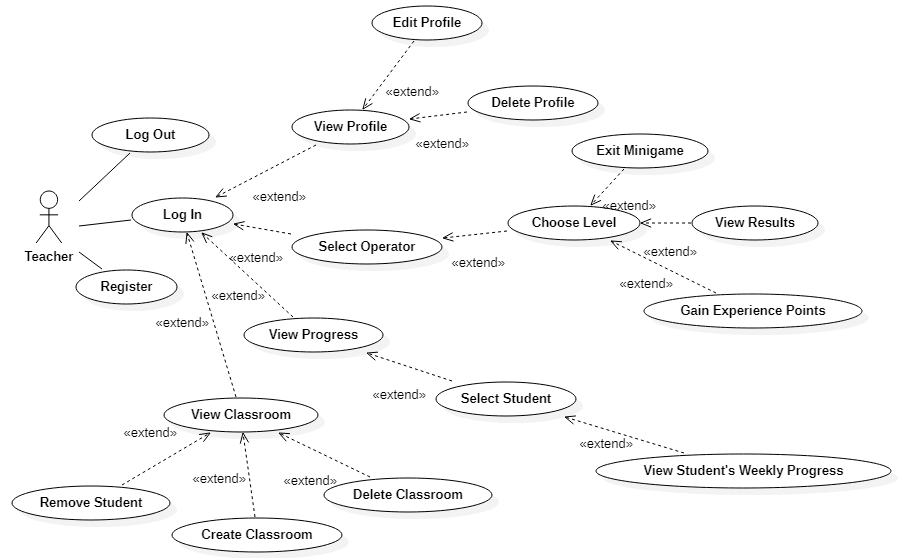


Figure #: Third Iteration Use Case for Teachers

## 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 initially 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 x shows the original Entity Relationship Diagram that was designed before I had a proper understanding of how the back end was actually going to work. Comparing this to the ERD that is used in the final version, its 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 is discussed in section 4.

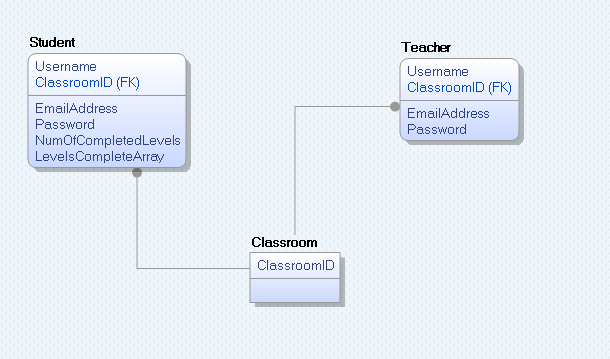


Figure 18 ERD showing relationships between tables

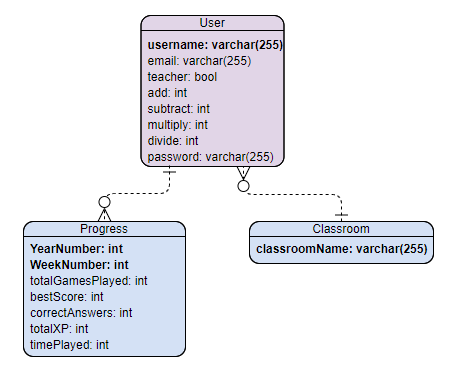


Figure #: Final Version of ERD

## 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. It showed the thought process behind choosing the Kanban board as the methodology for the project as well as an overview of what the system aims to be for users.

# 4. Experiment Development

## 4.1. Introduction

In this section there will be a demonstration of the working version of pages that have been created so for the front and back end of the application as well as how the middle tier of the system helps 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 and finally there will be some screenshots 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. Software Development

## 4.3. 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 x 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 first prototype to ensure that development using Kivy could continue as the project progressed.

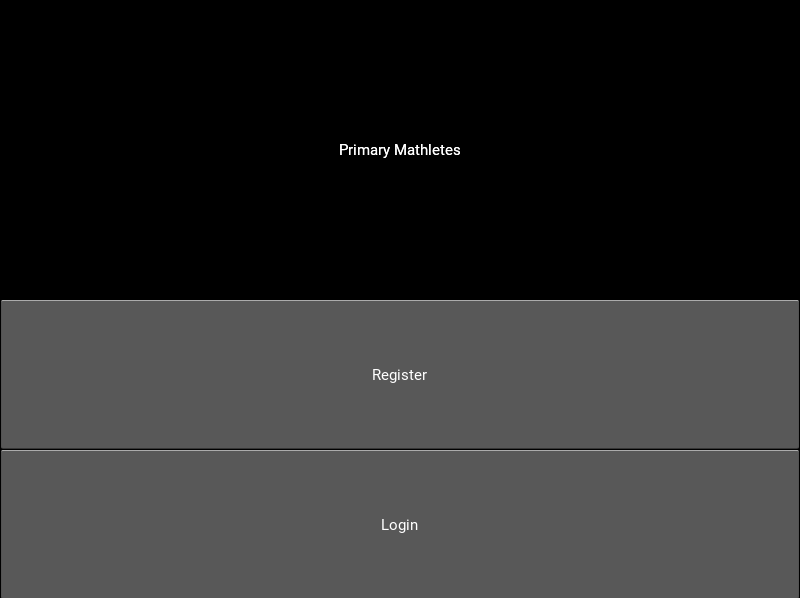


Figure 19 Title Page of the app

**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 4.4.)

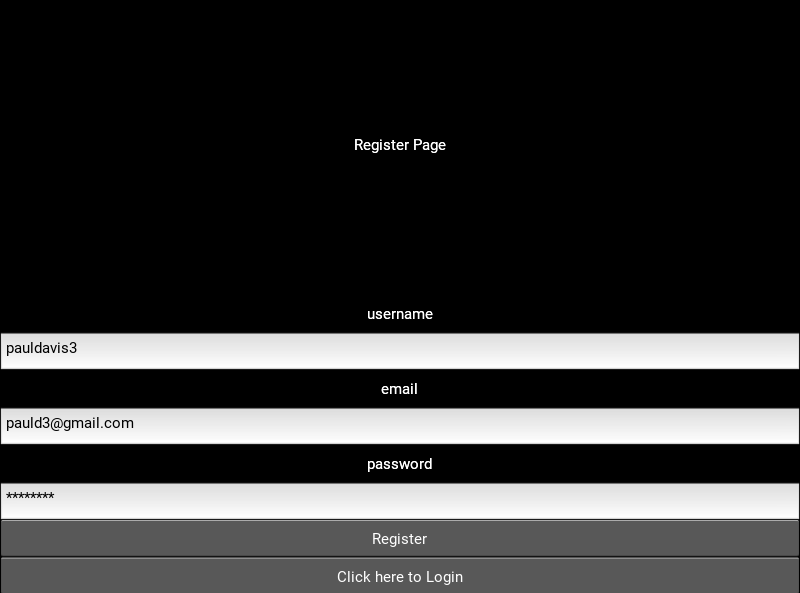


Figure 20 Register Page of the app

**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.

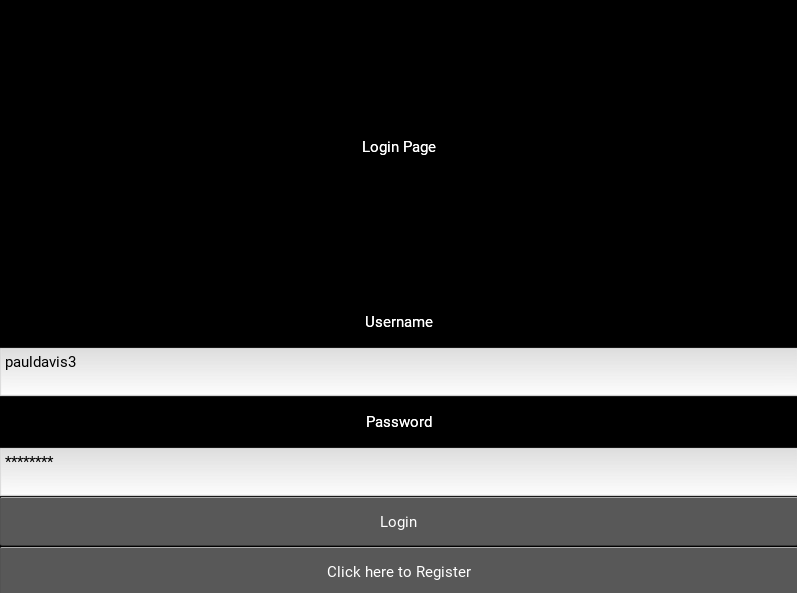


Figure 21 Login Page of the app

This kind of development continued for the creation of the rest of the screens before the design revamp was started and the final look of the application that can be seen in section 3 was finished.

## 4.4. 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.



Figure 23 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 there were only two firebase functions 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 x shows a code snippet showing the importing of the firebase library and the function used to connect to the projects Realtime firebase database.

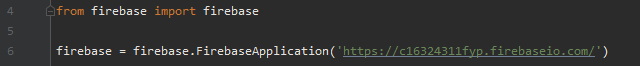


Figure 24 Snippet of code showing firebase library being used

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 create a random string that can be used to identify individual users in the database. An example of this can be seen in figure x.



Figure #: Firebase Post Example

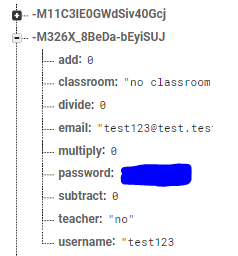


Figure #: Example of random strings created by Firebase POST method

- Reading 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 x. 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.



Figure #: Firebase GET Example

If a second parameter is given, the function will only return whatever was asked for. Figure x 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.



Figure #: 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 x where the green highlight shows new information being created in the database and orange highlights mean that a field is being updated.



Figure #: Firebase PUT Example



Figure #: 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 x shows the database deleting user ‘index’ from the users table.



Figure #: 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 x 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.

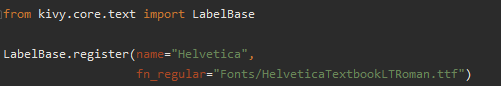


Figure #: Label Base Code Snippet

**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 x shows the 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.

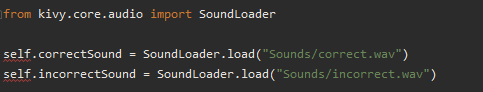


Figure #: Sound Loader Code Snippet

**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 x 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. One example of random numbers in use can be seen in figure x where a random number decides what potential answer found along the bottom of the minigame screen is going to be the actual answer to the question and then also used to choose random wrong answers for the rest of the potential answers (more on this function will be discussed further in this section).

|  |  |
| --- | --- |
|  |  |
| Figure #: Random Library Code Snippet | Figure #: 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.  
Near the top of the function seen in figure x the subject and the body of the email are 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. 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 x shows an example of an email received using this function.

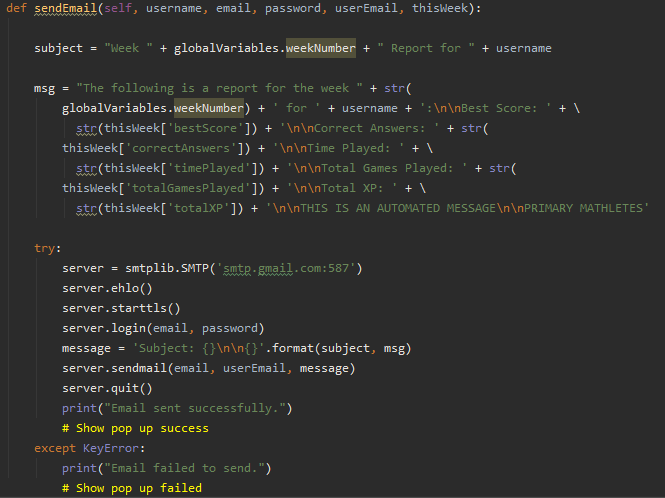


Figure #: Send Mail Function Code Snippet

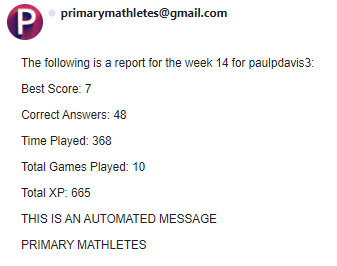


Figure #: Email received from the Send Email Function

**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 x 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. 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.



Figure #: Datetime Code Snippet

**Minigame – How it works:**

When the user clicks on an operator (function x) and then a level number (figure x), 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 #: Clicking On An Operator | Figure #: 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 harder and a single level easier 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.

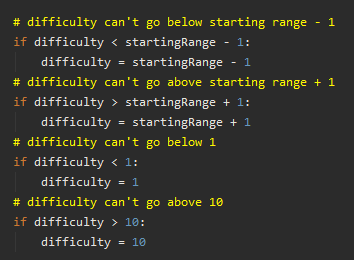


Figure #: Algorithm File Difficulty Setup

Depending on what operator was chosen, the range of numbers that can be randomly selected varies. This is because, by 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 x shows the minimum and maximum range being capped on different numbers depending on what the difficulty chosen was but never exceeding the expected knowledge of third class students.

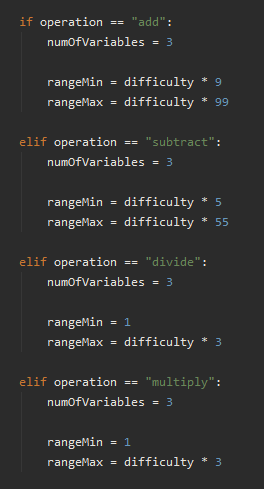


Figure #: Setting Maximum and Minimum Ranges

Next, a while loop runs to ensure that the answer to the question does not exceed the maximum number 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.

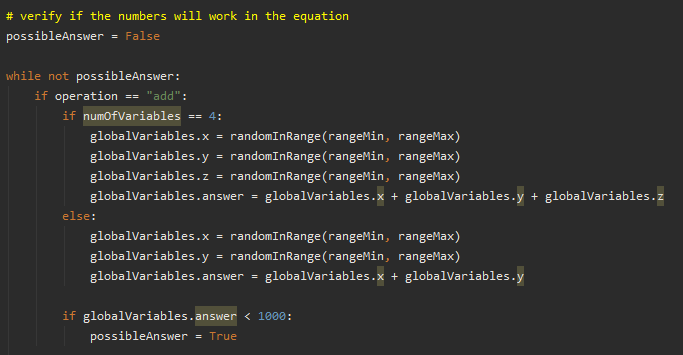


Figure #: Ensure Answer Doesn’t Exceed 1000

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 straight-forward equation.

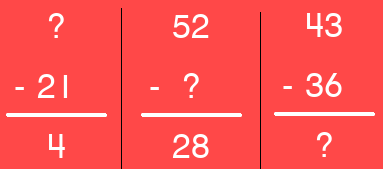


Figure #: Examples of ‘blank’ Being 0, 1 and 2

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.

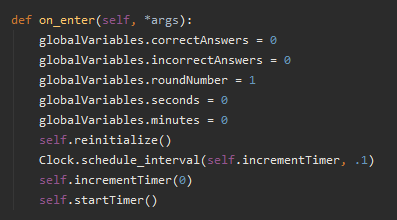


Figure #: Minigame Initialization

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 get new numbers for the next round of questions and the update text function runs.

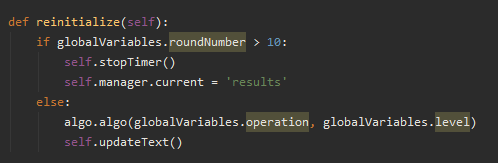


Figure #: Reinitialize Function

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 x.



Figure #: 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 and no matter what answer they choose the round number is incremented and the reinitialize function runs again after this check.

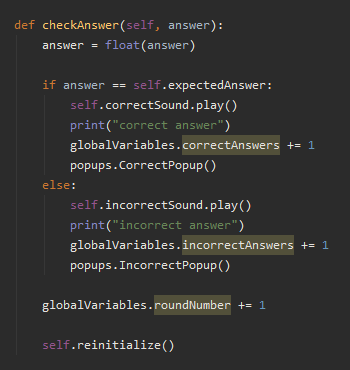


Figure #: Check Answer Function

The getPotentialAnswer function is the most complex part of this minigame section of the code. 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 expected answers length and creates a check to make sure the potential answer is suitable.

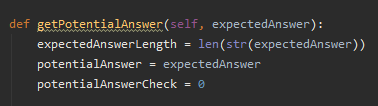


Figure #: Get Potential Answers Code Snippet

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 and it basically flips the numbers 1-10 backwards to 10-1. Instead of 1 getting mapped to 10, it instead starts at 9 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 interesting 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.



Figure #: 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 #: Difficulty Mapped Numbers 1-10

This difficulty mapped variable is then used to decide how random the potential answers will be. Looking at figures x and x 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 actual 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 actually is the correct one.

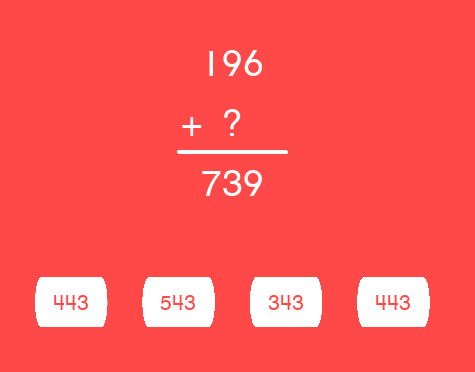


Figure #: Level 8 Addition Potential Answer Example

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 the answer away straight away without actually having to understand the entire question.

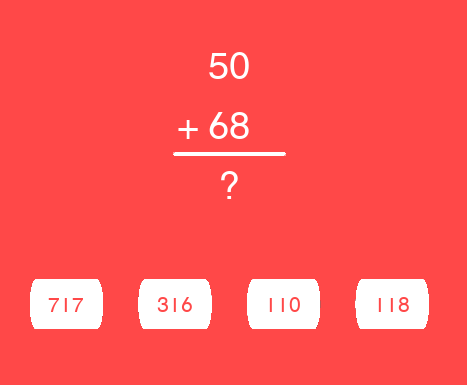


Figure #: Level 2 Addition Potential Answer Example

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.



Figure #: Potential Answer Code Snippet

**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 x shows that it is just a list of every available screen.

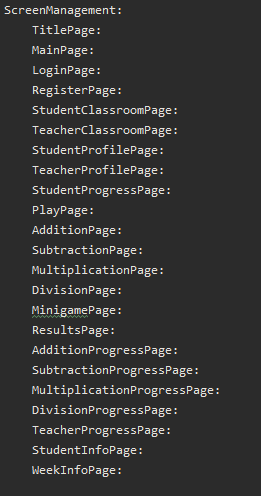


Figure #: Screen Manager Code Snippet in Kivy File

**Basic Screen Structure:**

Figure x 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.

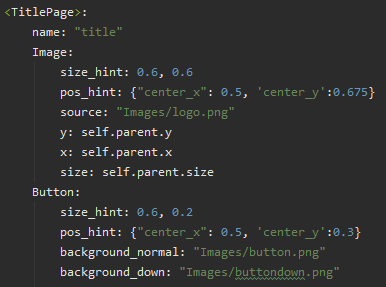


Figure #: Basic Screen Structure in Kivy File

**Images:**

Images as attributes are only used as a handful of times in the application but figure x 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 go 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 figure x and x.   
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.

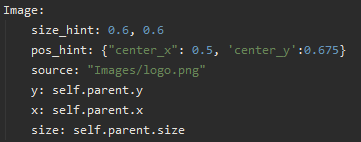


Figure #: Image Code Snippet in Kivy File

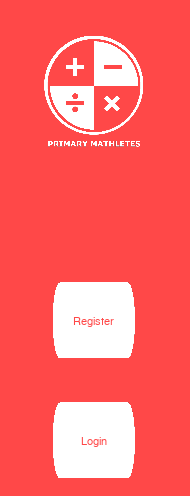


Figure #: Logo Adapting 1

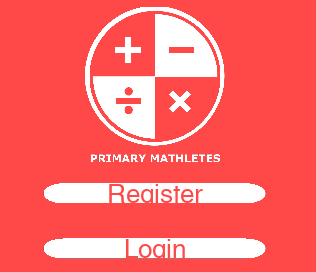


Figure #: Logo Adapting 2

**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 x, 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.

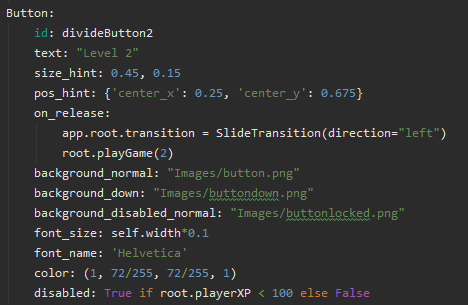


Figure #: Button Code Snippet in Kivy File

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 x 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 #: 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 x shows that labels share many attributes with the other entities such as size and position hint, as well as font size and font name.

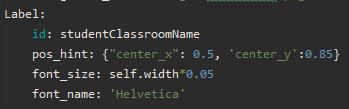


Figure #: Label Code Snippet in Kivy File

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 x 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.

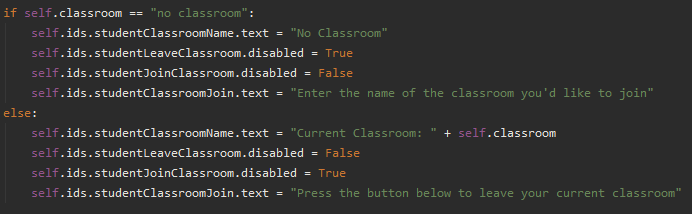


Figure #: Dynamic Update of Label Text

**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 utilizied 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.

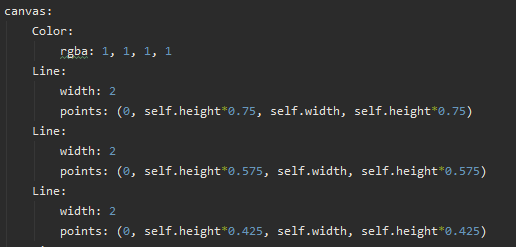


Figure #: Canvas Code Snippet in Kivy File

|  |  |
| --- | --- |
|  |  |
| Figure #: Canvas Line Styling 1 | Figure #: Canvas Line Styling 2 |

## 4.5. 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 x. Each user in this prototype only had a username, password and email address.

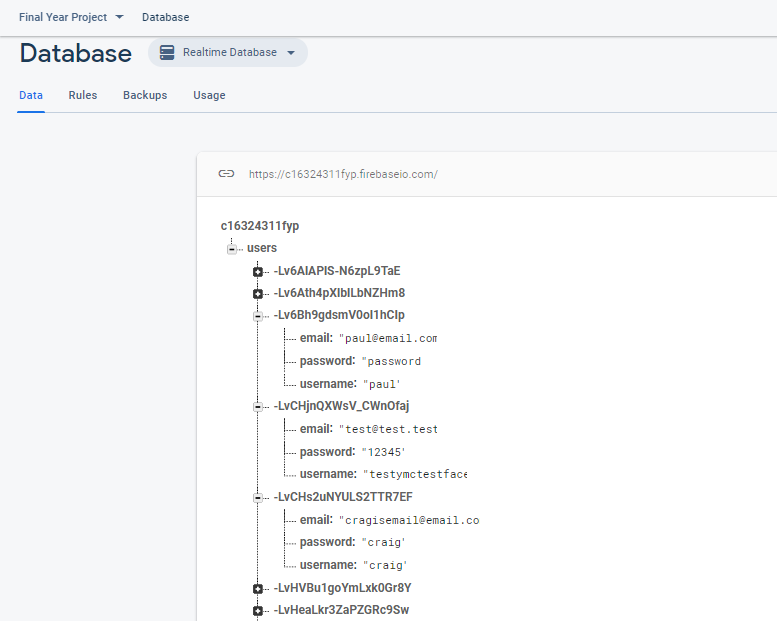


Figure #: Old Database Prototype

Throughout development more and more attributes and tables were added and removed from the database but the final version can be seen in figure x. The final version of the database shows that there is now a classrooms table which 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. 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 users 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

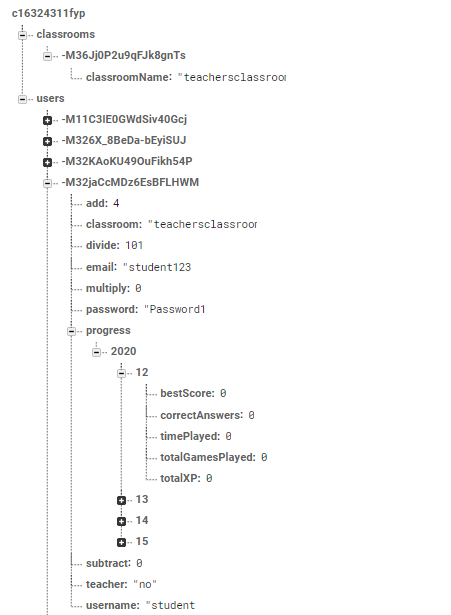


Figure #: Final Version of the Database

**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, uppercase letter, 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 its looking for like .isupper(), .islower(), isnumeric() etc. Then the if statement seen in figure x 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.

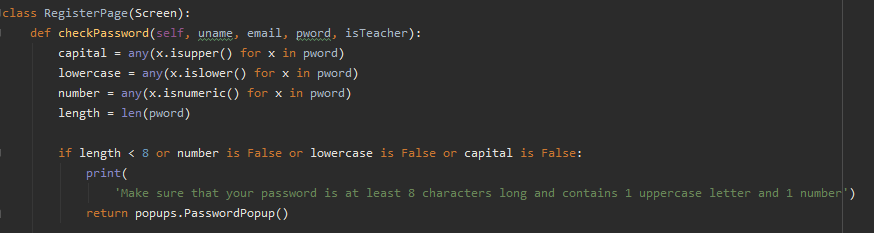


Figure #: Password Check on Register Page

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 users email address does not meet this structure then a popup similar to the password popup is shown on screen instructing the user to enter a valid email address.

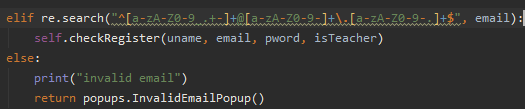


Figure #: Email Check on Register Page

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.



WEEK CREATION IF IT DOESN’T EXIST MIGHT BE SOMEWHERE ALREADY BRING IT DOWN HERE

**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 x shows a code snippet importing the json store library, setting it up and the using it to both retrieve information (“ScreenManagement.store.get”) and put information into it (“ScreenManagement.store.put”).



Figure #: JSON Store Code Snippet

## 4.6. Conclusions

# 5. Evaluation

## 5.1. Introduction

## 5.2. Software Evaluation

## 5.3. Specific Evaluation

## 5.4. Questionnaires and Interviews Evaluation

## 5.5. Conclusions

# 6. Conclusions and Future Work

## 6.1. Introduction

## 6.2. Conclusions

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 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 actually 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.

## 6.3. Future Work

As seen in section ***\*find section with airtable in it \****Airtable was used to document any bugs that were found while testing and there are still a number of 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 a second after each correct or incorrect answer to guard the user from 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 necessary to come up with a different system for storing the users data. The firebase database that is used currently in this project works but as seen in the ***\*section where the fact that the app doesn’t compile because of firebase is mentioned\**** section, it was not possible to compile this project into a useable APK for android or IPA for iOS due to the inclusion of the Firebase library which is not supported by Buildozer for some reason. Due to this, turning the application into a web application and developing a Django database using MySQL or Oracle might not be a bad idea in the future because even if this project did compile with the Firebase library, their 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.

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 actually has knowledge about recently studied aspects of the curriculum or whether the teacher needs to revisit these sections again.

# 

# 7. Bibliography

# [1] 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:

# <http://www.iaeng.org/publication/WCECS2016/WCECS2016_pp150-153.pdf?fbclid=IwAR1iFguPdUgTeTq4pAUzfkYheh7g1EVxyUmrFUI8SE6hXfLr6E5DdhYSafg> [Accessed 8 Dec. 2019].

[2] Matthew Kearney and Damian Maher - MOBILE LEARNING IN MATHS TEACHER EDUCATION: USING IPADS TO SUPPORT PRE-SERVICE TEACHERS’ PROFESSIONAL DEVELOPMENT: <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> [Accessed 8 Dec. 2019].

[3] Machdel Matthee, Jacobus - Mathematics on the Move: Supporting Mathematics Learners through Mobile Technology in South Africa: <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> [Accessed 8 Dec. 2019].

[4] Al-Zeidi, A, Al-Kindi, K, Al-Khanjari, Z. A. - SQU Future: From E-Learning to M-Learning Application Development: <https://www.researchgate.net/profile/Youcef_Baghdadi/publication/236310002_Web-enabled_services_development_with_respect_to_service-orientation_paradigm/links/5528c2640cf29b22c9bcb350.pdf#page=185>   
[Accessed 8 Dec. 2019].

[] Anthony, UX Movement – Why Rounded Corners are Easier on the Eyes: <https://uxmovement.com/thinking/why-rounded-corners-are-easier-on-the-eyes/> [Accessed 6 April. 2020]

[] Kendra Cherry, Very Well Mind – What is Positive Reinforcement: <https://www.verywellmind.com/what-is-positive-reinforcement-2795412> [Accessed 6 April. 2020]

[] Azhar, UX Design – How to Pick Colors for Your App Without a Struggle: <https://uxdesign.cc/how-to-pick-colors-for-your-app-without-a-struggle-bc46c5e19574> [Accessed 6 April. 2020]

[] Renk Etikisi, The Effect of Colors on Chidlren - :<http://renketkisi.com/en/the-effects-of-colors-on-children.html> [Accessed 6 April. 2020]

Appendix:

- put the questionnaire done by teachers here

- put instructions on how to download and run the app here