Mobile Application Focusing on Understanding of Maths Relations

Sonny Pritesh

UP926975

School of Computing

Final-Year Project

Engineering Project

Abstract

Over the years mobile phones have an increase of usage. Many use it for keeping in contact with friends and family, other use it to share their work or use to upload snippets of their life. Many also use mobile phone as a means of education. There are hundreds of applications available to download on smartphones that many in the educational field have started to integrate it into their everyday teaching.

This project showcases the development of a mobile application designed for Android. This is for those struggling with math relations as the application allows for the user to attempt question on the topic within math relations. Requirements for the application were gathered through the analysis of both relevant literature and already existing applications, allowing for the design to take shape. The implementation stage and testing stage are also documented which allows for the artefact to be evaluated against the requirements. The report is the concluded by reflecting on developmental decisions.

Table of Contents

1. Introduction

1.1 Project Background

Many applications exist to help supplement the students learning in education. Many act as a calculator, whilst other are designed to help a broader range of student covering a large variety of subject areas. However, there are too few applications that are design to help student focus on specific topics.

This project aims to produce an Android application that will help those students are finding it hard to learn and understand a specific topic. This project will be designed to help those students that are struggling with the topic are of math relations. Which would hopefully help increase their academic performance.

1.2 Aims and Objectives

This project aims to document and produce an educational mobile application for Android devices tailored towards the focus of mathematic relations. To achieve this, certain objectives will be established:

- Research of student performance during COVID-19 and the affect mobile learning application have on their performance
- Exploration of methodologies and determining the suitable methodology for the development of the application
- Production of requirements from the research
- Produce a design and architecture for the application
- Implementation and testing of the application.
- Evaluation of requirements against the application
- A conclusion of the work and the discussion of potential future work

1.3 Project Overview

This is an overview of each chapter that is included in the report.

Chapter 2 will include a literature review on the topics of student performance during the COVID-19 pandemic along their performance through the of academic learning applications.

Chapter 3 will include information regarding the methodology that I have chosen, including a comparison of different methodologies.

Chapter 4 will include the requirements for this project which have been devised from producing the literature review.

Chapter 5 will include the design and architecture for the mobile application.

Chapter 6 will include the implementation of the application and the testing of its functions based on the requirements of the application

Chapter 7 will include a reflection of the work based on the system requirements and how the developed application has met the requirements.

Chapter 8 will include the project conclusion and the discussion of future work.

2. Literature Review

This literature review will look at and analyse the how mobile applications, specifically revisions applications, affect the performance of students in academia.

2.1 Academic Performance of Students

In recent times, due to the effects of COVID-19, many students have started to learn from home through the use of video calls and pre-recorded lectures. A study done by Urtel (2008) shows that the performance of student in face-to-face is better but as the number of courses increase under analysis, it shows that performance increases through online learning (Iglesias-Pradas et al., 2021).

Many meta-analyses are shown to support the idea that the academic performance of students is usually higher through remote learning than it is through face-to-face teaching. Whilst some also conclude that there is negligible difference for academic performance of students between online learning and face-to-face teaching. In the meta-analyses done by Shachar & Neumann (2003), it is concluded that the results show that distance education (DE) is a more effective form of teaching. However, it was concluded, in the study done by Zhao et al. (2005), that the factors that have been found to have affected effectiveness of face-to-face learning also seem to have affected DE.

As the vaccines for COVID-19 have started to be distributed among the public, schools and universities have also started to adapt by implementing a mix of online teaching and face-to-face teaching called blended learning. In the study done by López-Pérez et al. (2011), the drawn conclusion is that the implementation of blended learning has a positive effect on student performance whereas online learning itself does not have an impact on student performance. Both Bernard et al. (2014) and Means et al. (2013) support the idea that blended learning can result in better academic performances in higher education student and that further supported by the results found in the meta-analyses done by Vo et al. (2017).

2.2 Mobile Applications

As the availability of electronic devices have increased over the year, so has its integration into society, especially into the educational sector. Many educational facilities have, in one way or another, utilised electronic devices into their teaching. Many students also use electronic devices, either PC, laptops, or mobile phones, as a way of supplementing their learning by using applications designed and produced for their specific topic area.

Mobile phones in particular have a unique standpoint as an educational resource as many smartphones can download application related to their studies in a few seconds. It can be seen that student participation increases when they are using a mobile application related to education when it is in the form of a game as seen in the study done by Lim & Wang (2005) where students admitting that they learned more information than they realised after using the application, EcoRangers. Typical comments that students would give were ‘the game gave me an understanding of topics taught in social studies and geography’.

Some researchers have defined the difference between e-learning, a learning process supported digital electronic tools and media and mobile learning (m-learning) is e-learning that uses mobile devices and wireless transmission (Alqahtani & Mohammad, 2015). During a study done by Hamdan & Ben-Chaban (2013), the participating teachers observed that mobile learning applications had a positive impact on students and that they were more engaged in the learning process than before. It is also revealed that according to literacy surveys that teachers are not the only source of information for students as their main sources have become site like Google, YouTube and other online sources.

2.2.1 Microsoft Maths Solver

Microsoft Maths Solver (Microsoft Maths Solver, 2019) is a mobile application released for both iOS and Android and is found in the 'Education' section of both app stores. The application is designed to help users with problems including, arithmetic, algebra, trigonometry, calculus, and many other mathematical topics.

According to the description in the Android app store, the problems that are supported include:

- Elementary Maths: arithmetic, real, complex numbers, lowest common multiple, greatest common divisor, factors and roman numerals
- Pre-Algebra Maths: radicals and exponents, fractions, matrices, determinants
- Algebra: quadratic equations, system of equations, inequalities, rational expressions, linear, quadratic graphs and exponential graphs
- Word problems on maths concepts, number theory, probability, volume, surface area
  Basic Calculus: Summations, Limits, derivatives, integrals
- Statistics: Mean, Median, Mode, Standard Deviation, permutations, combinations

The way this is done is by having the user either write out their problem or scan their problem through their camera. That is then recognised by the AI and a step-by-step explanation is provided to the user. The main focus of the application is to act as a supplement to students and further help them with their studies.

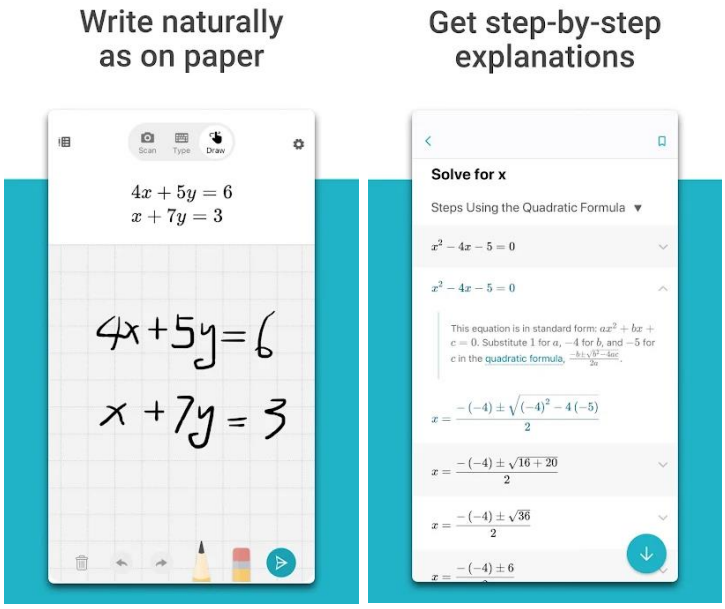


Figure 2.1: Images of Microsoft Maths Solver taken from the Android App Store

The application also provides the user with daily quizzes that they can complete and learning resources that are relevant to the problems the user has.

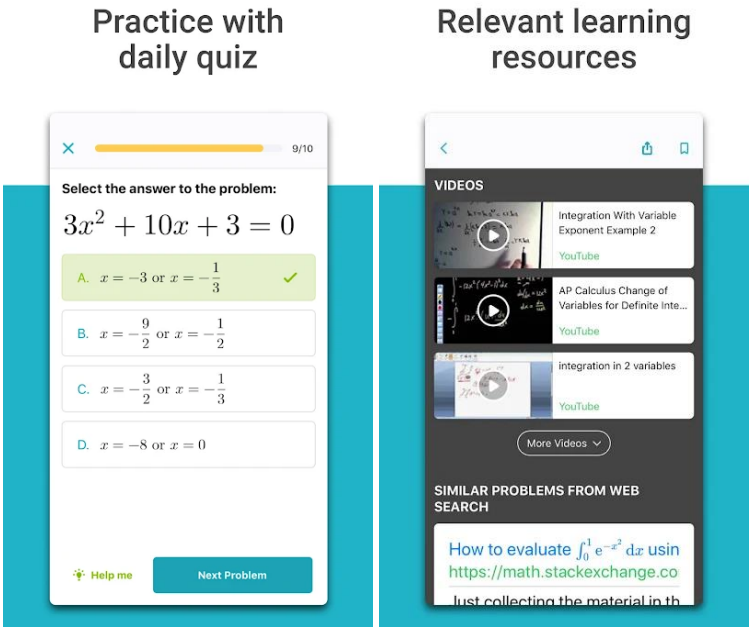


Figure 2.2: Images of Microsoft Maths Solver taken from the Android App Store

2.2.2 Khan Academy

Khan Academy (Khan Academy, 2008) is a mobile application available on both iOS and Android. The application is designed as a supplement revision application for those currently in education. The application covers a wide range of subject areas including maths, science, economics and many more. The application mainly focuses on providing users with videos and interactive practice exercises regarding their topic area. According to the description provided in Android app store, the interactive practice exercises have instant feedback and step-by-step hints.

The application also provides a service where the user can download the resources within application to allow for use offline.

The application is split into three sections; home, explore and bookmarks. The home section will give the user a summary of the latest action, i.e recently attempted questions. The explore section is for the user to find questions and resource that they can use for their studies. Lastly, the bookmarks section is for the user save any specific questions or topics. This section also allows the user to download their bookmarks so that they can be used offline.

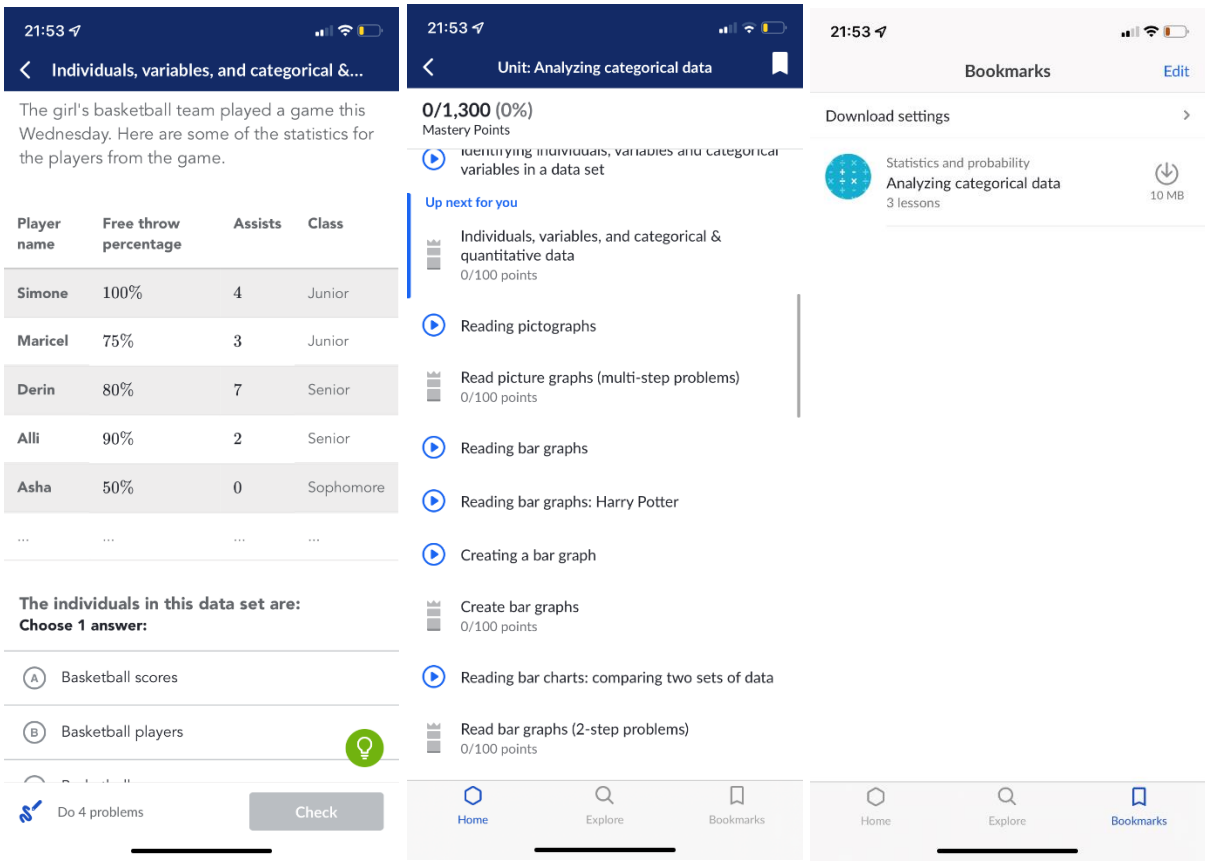


Figure 2.3: iOS Khan Academy Screenshots

After doing some searching on its iOS application, I was unable to find resources related to maths relations however after searching for it on its website I was able to find resource for the topic.

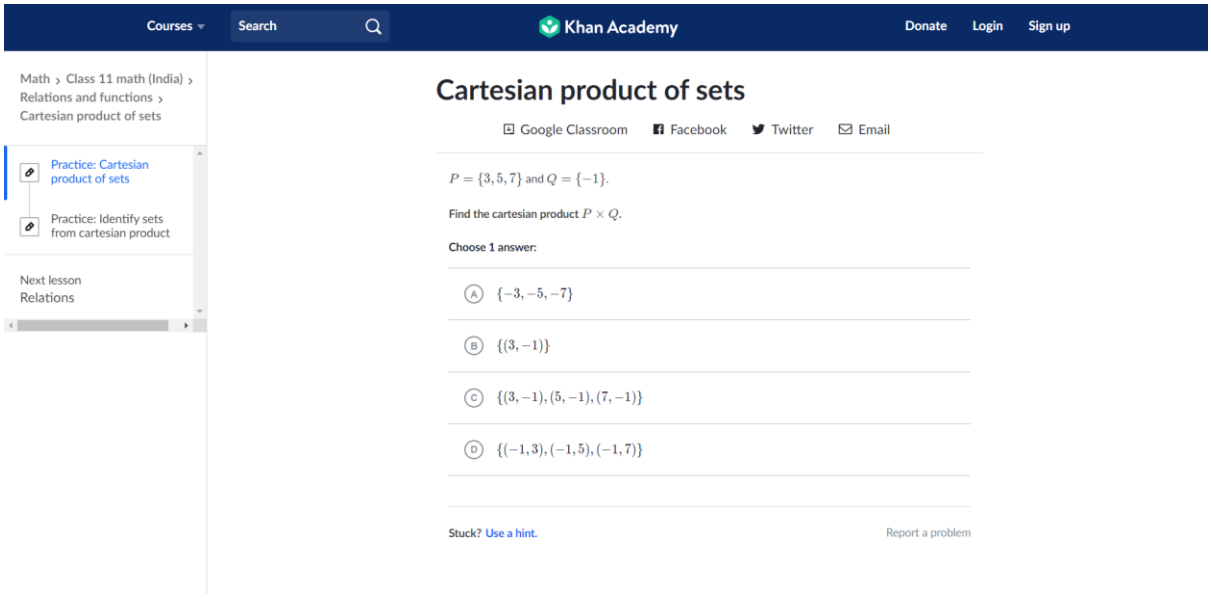


Figure 2.4: Khan Academy Website Screenshot relating to Cartesian Product.

2.2.3 Quizlet

Quizlet (Quizlet, 2007) is an iOS and Android application designed to help students with their revision with the use of flash cards. The application allows its users to create customised flash cards

for their revision and allows them to view publicly shared flash cards made by other users. The application also contains “expert-verified textbook solutions” that the use can view.

According to the description in the Android app store, the Quizlet app can allow the users to:

- Learn with flashcards
- Get expert-verified textbook solutions for your toughest problems
- Share flashcards with friends, classmates, or students
- Get ready for your exams with Learn mode
- Put your memory to the test with Write mode
- Race against the clock in a game of Match
- Learn foreign languages
- Listen to your texts pronounced correctly in 18 languages
- Learn about science, maths, history, coding and more

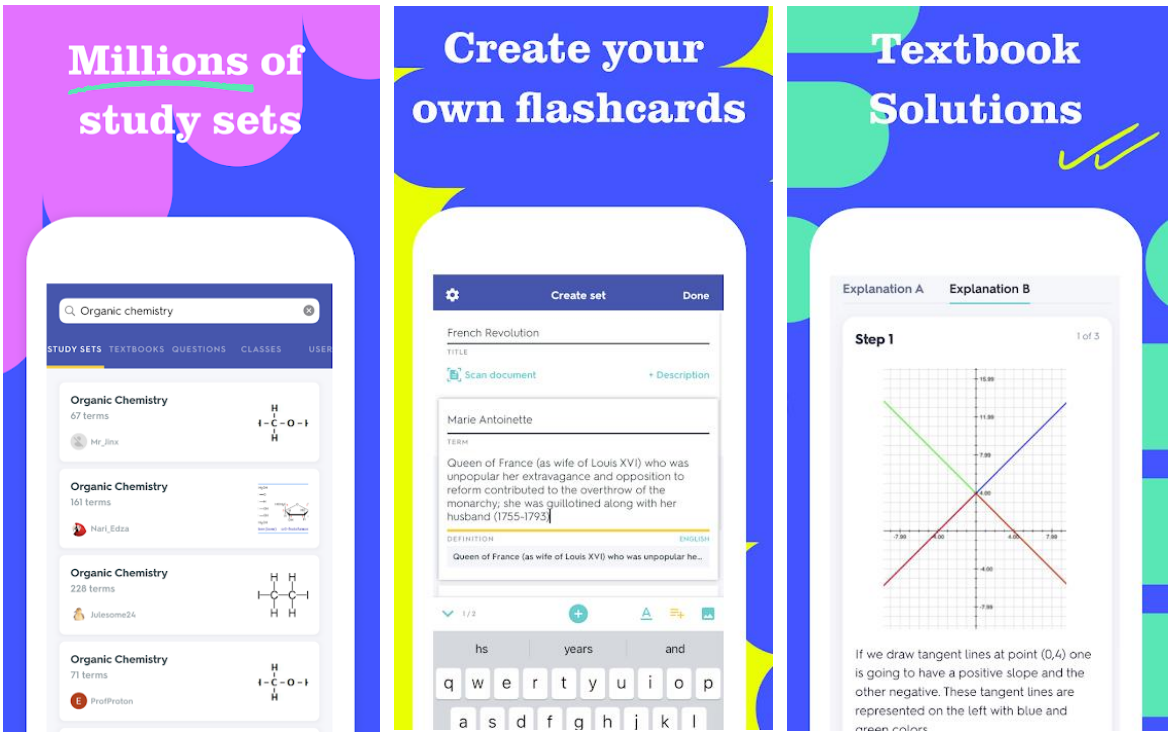


Figure 2.5: Images of Quizlet taken from the Android App Store

2.3 Summary

This chapter has explored the academic performance of students that have been affected by the global pandemic of COVID-19 as well how the use of mobile applications and m-learning has impacted students in education. Mobile learning applications were also discussed showing application for general studies or for supplementing the students learning. However, it can be seen that none of them specialise in mathematic relations specifically. This chapter will be used as a basis for the user requirements.

3. Methodology

The methodology is used to reach the objectives and requirements of the project in a low cost and an effective manner to allow for a smooth process. This chapter will look at existing methodologies that are commonly used for mobile applications and will give a reasonable reason for why the specific methodology was chosen for this project.

3.1 Agile Methodology

The agile methodology starts off similarly to the waterfall methodology where the requirements and design are created, but changes at the development stage. Agile methodology focuses on continuous releases of the artefact where each section of the artefact is released and tested in iterations. Once each iteration is complete then it releases the artefact to the user and moves on to the next iteration. The most important principle in the agile methodology is customer satisfaction by giving fast and continuous delivery of each iteration of the artefact.

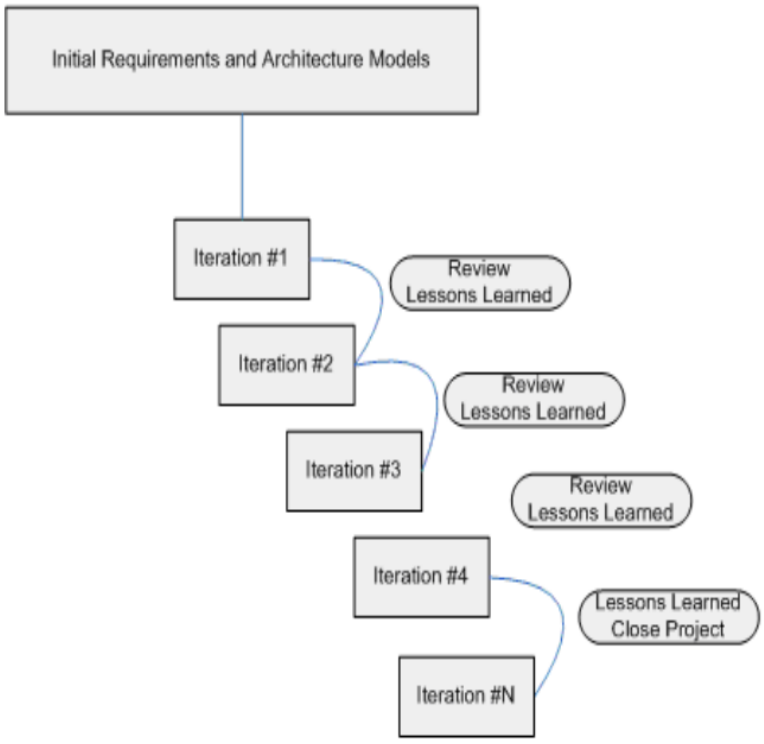


Figure 3.2: Agile Model Life Cycle (Balaji & Sundararajan, 2012)

The biggest advantage for using this methodology is having the ability to respond to changing of user requirements of the project. This allows for the changes to be integrated immediately into the artefact. There is also constant communication between the developer and the client which allow for almost no ambiguity within the work. However if the project size is too large, there is a chance that project can easily go off track due to the ever-changing user requirement, especially when the customer representative(s) do not know what they want.

3.1.1 Extreme Programming

Extreme programming (XP) is an agile methodology framework that was designed to allow for dynamic changes in the system requirements. As this is a derivation of the agile methodology, XP also uses the iterative development model. The main practices of XP are (Powell-Morse, 2017):

- Pair Programming: This is where two developers work on the same system at the same time when developing code.

- Planning Game: This is where there are constant meetings at regular intervals between the developer(s) and client(s).
- Test Driven Development: This is where tests are developed for each and every requirement of the project. Once that is done then the development of the code starts.

3.2 Waterfall Methodology

The waterfall methodology model is a development model that is carried out in a sequence and before going to the next step of the process. The project is broken down into sequential tasks as shown in Figure 3.1.

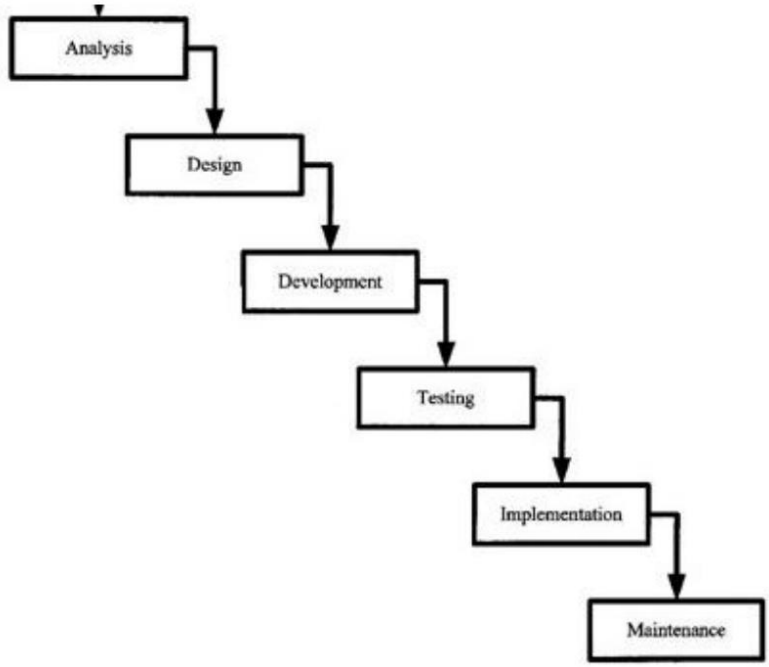


Figure 3.1: Waterfall Model Life Cycle (Balaji & Sundararajan, 2012)

First comes the analysis or requirements where the developer would create a set of requirements in tandem with the client. Then comes the design of the artefact that the developer would create and get approval from the client. Next comes the development phase, where the developer would create the entire artefact and wrap it up before moving onto the testing phase. The developer would then move to testing phase to make sure the developed artefact is working. Then the developer moves onto the implementation phase, then onto the maintenance phase.

One of the major disadvantages of this methodology is that if the client wishes for a change in requirements, especially if it is a large change, that change will not be implemented in the current development process (Balaji & Sundararajan, 2012) due to the methodology being a linear process. Also, if steps are missed then it could lead to problems later in the development process.

3.3 Chosen Methodology

After comparing and analysing the waterfall, standard agile, and XP agile methodology. I have decided to go with the standard agile methodology that I am adapting with features that are present in XP agile methodology. The reason for not choosing the waterfall methodology is due to its linear nature restricting the chance for changes in the requirements. The XP methodology due to the fact it more of a team based oriented methodology.

4. Requirements

This chapter will surmise the requirements for the project.

4.1 Requirements Elicitation

The main method used for the gathering of requirements for this project is the research from the literature review and the analysis of current applications and their features.

4.1.1 Research of Current Applications

The research of current applications has helped establish a base understanding of the requirement for the project. The applications regarding specific help for student gave a wide-ranging spectrum of feature that could potentially be used in the artefact. Applications that were more towards general revision help, gave a better understanding of how to lay out the application to allow for ease of access for the user. Features not present in the available application could be established along with their respective requirements.

4.2 Requirements

Functional Requirements			
ID	Requirements	Description	Priority
1	Login Page	The first page that should show up if the user has not already logged into that application	Must Have
2	Registration	A page where the user can create a new account for themselves	Must Have
3	User Friendly UI	A user interface that is simple to use.	Must Have
4	Database	The application must store all details about the user the associated data, that can only be accessed by the specified user	Must Have
5	Results Display	Where the user can request to see the question they have completed and whether it is correct or not	Must Have
6	Question Display	Where the user will go to answer the question given to them	Must Have
7	User Friendly Question UI	Question UI should be easy to read by the user and is clear about what is a part of the UI.	Should Have
8	Question Verification	The questions should be verified before the user is notified of whether they got the answer correct or not.	Must Have
9	Question database	The questions should be taken from a database and not stored locally in the application.	Must Have
10	User Profile	Implementation of a user profile that displays the statistics of the user.	Could Have
11	Creation of new questions	Question should be created and displayed when requested by the user	Must Have
12	Request help feature	A feature where the use can request for help, usually something that is usable immediately	Should Have
13	Retry question display	Where the user will go when they have requested to retry a question that have previously gotten incorrect	Must Have
14	Question variety	Questions must vary in topic and difficulty.	Should Have

15	Quick Access	Ensure the features of the application can be used as soon as possible after the application is launched by the user.	Should Have
16	Password recovery	Password can be recovered when requested	Should Have
17	Password encryption	Password should be encrypted when the user has registered their account	Must Have

Table 4.1: Functional Requirements

Non-Functional Requirements			
ID	Requirements	Description	Priority
1	Application should only work on touch screen devices	The application will require buttons to be pressed repeatedly on the screen using a touch screen device. Navigation will also require the use of a touch screen device.	Must Have
2	Compatible with Android devices	The application must be compatible on at least 70%-80% of current Android devices, including devices running the most recent version of android	Should Have
3	System Performance is not affected	The average smartphones should be able to run the application without any delays	Should Have
4	Appropriate GUI	GUI should be appealing and not offensive	Must Have
5	Responsiveness	The application should be responsive to any input the user gives, and should be able to return to correct state if error occur	Should Have
6	Availability	The application should be working 99.9% of the time	Must Have

Table 4.2: Non-functional Requirements

5. Design

5.1 Architecture

When creating a mobile application for Android, it is important to design a visual application architecture. This will allow for a visualisation of the navigation within the application and can avoid redundancy when it comes to navigation buttons. The architecture design should also show how the data will be passed through between the many pages of the application.

The project will use Android Studio to develop the application, with Java used for classes and activity files and XML for resource files. Each activity file, which are initialised through the use of intents, has a corresponding XML file, which defines the UI components contained within the activity.

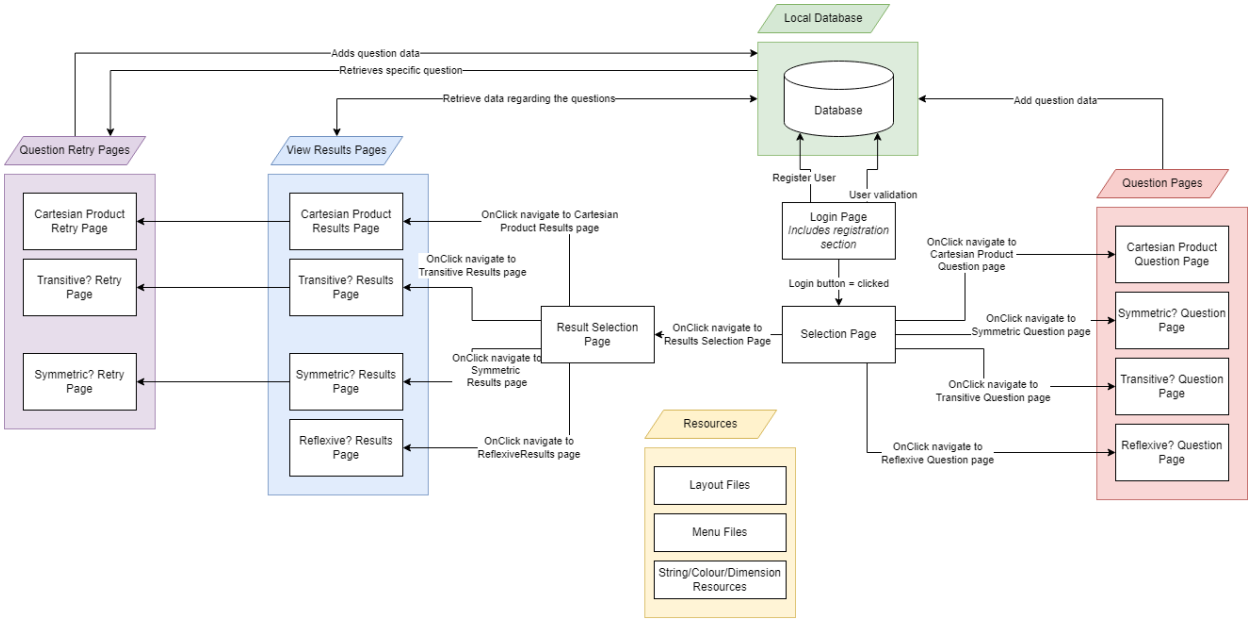


Figure 5.1: Application Architecture

Figure 5.1 shows a high-level view of the application architecture which displays the resource files that will be used and the navigation within the application; Login page, Selection pages, Question pages, Result pages and Question Retry pages. The onClick functions will send the user to the next activity depending on which button is clicked.

5.2 User Interface Design

The user interface (UI) design describes the appearance of the application and how the user will interact with the application, therefore the design of the application will be simple and each component's use will be clear to the user.

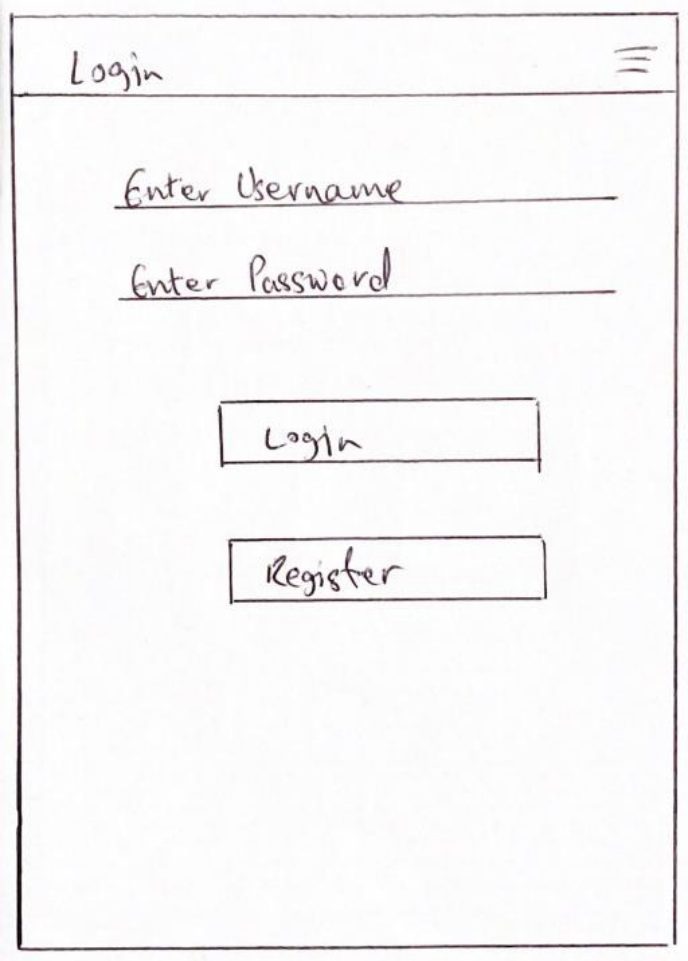


Figure 5.2: Login Activity Page Design

The login page shows two simple text boxes that will allow the user to enter their username and password. There are also two buttons one for login and the other for registration, this will direct the user to the register page.

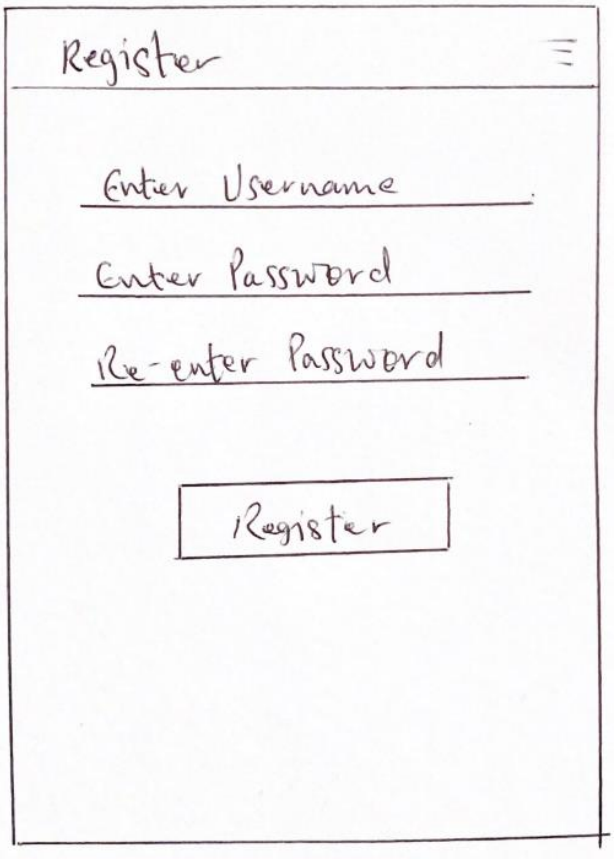


Figure 5.3: Register Activity Page Design

The register page shows three simple text boxes that will allow the user to enter their username and password and re-enter their password for verification. There is also a button below the text boxes which, when clicked, will register the user and the details to the database, it would then redirect the user back to the login page, where user can log in to their account using the newly created account.

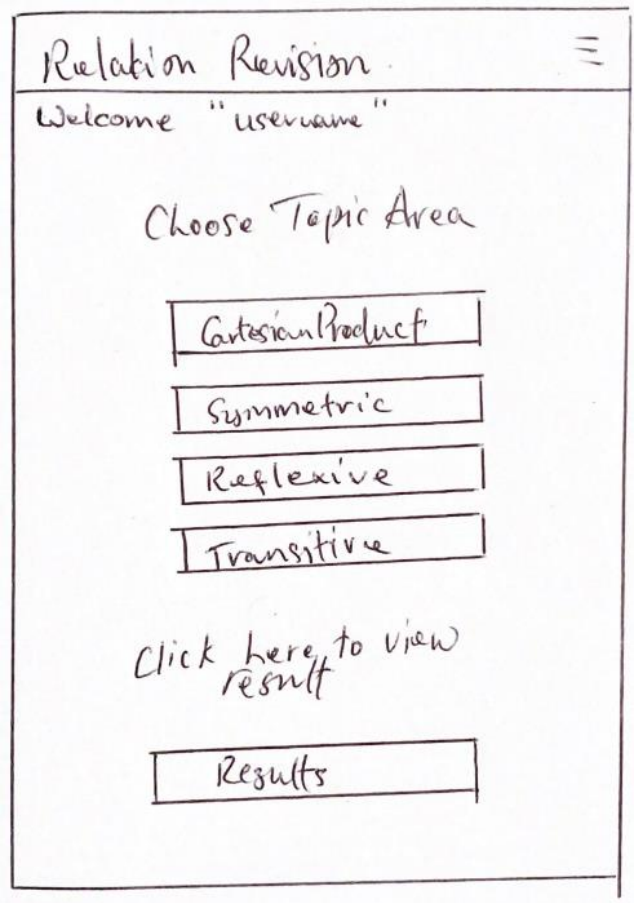


Figure 5.4: Choose Activity (Home) Page Design

Figure 5.4 represents the page where user will first be directed after they have logged in. The page will welcome the user and give them four choices regarding the topic area they wish to test. At the bottom there is also a button for the user to direct them to results page.

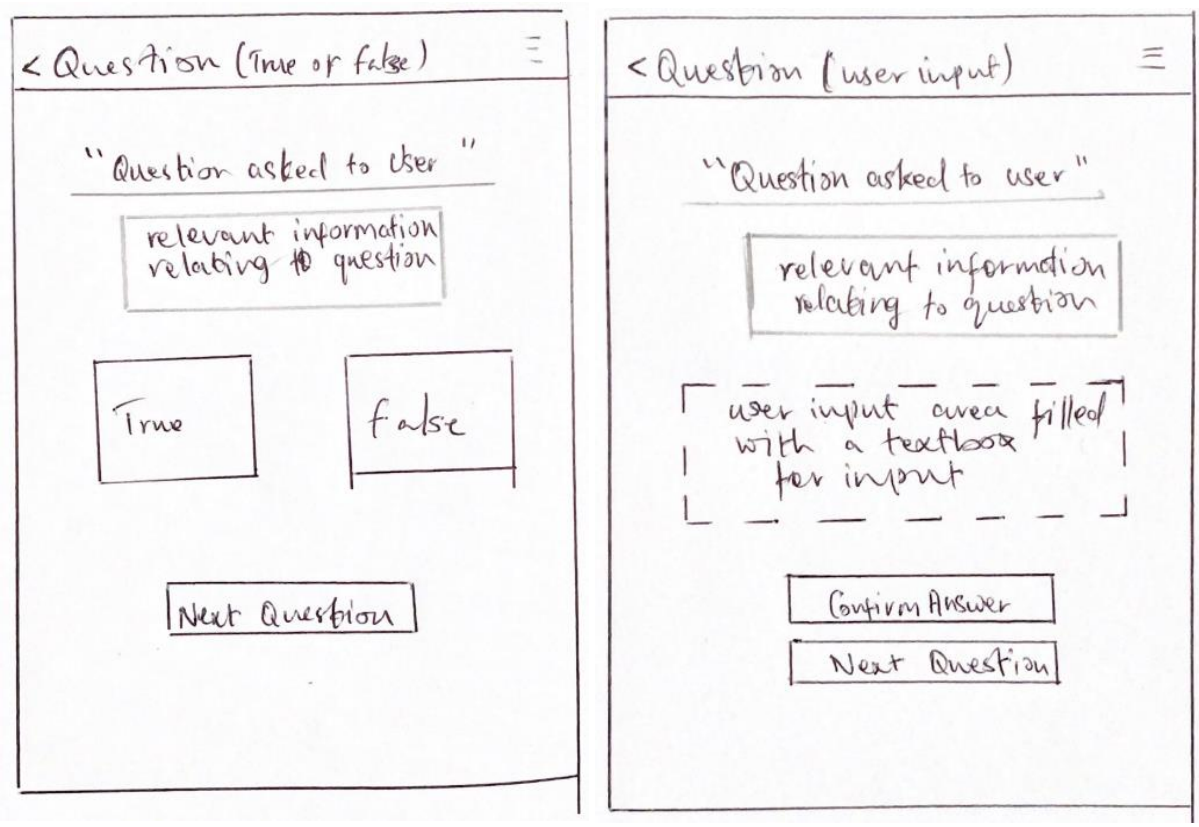


Figure 5.5: Question Activity Page Designs

Figure 5.5 shows the two different question pages that the user will be directed to depending on the topic they have chosen from choose activity page (Figure 5.4). On the left is the page for questions that only require a true or false answer. On the right is the page for question that require the user to input values for the question. The user input question page has a confirm button which will confirm whether the inputted answer is correct or not. Both pages contain a next question button which when clicked will create a new question for the user and replace the text on the page.

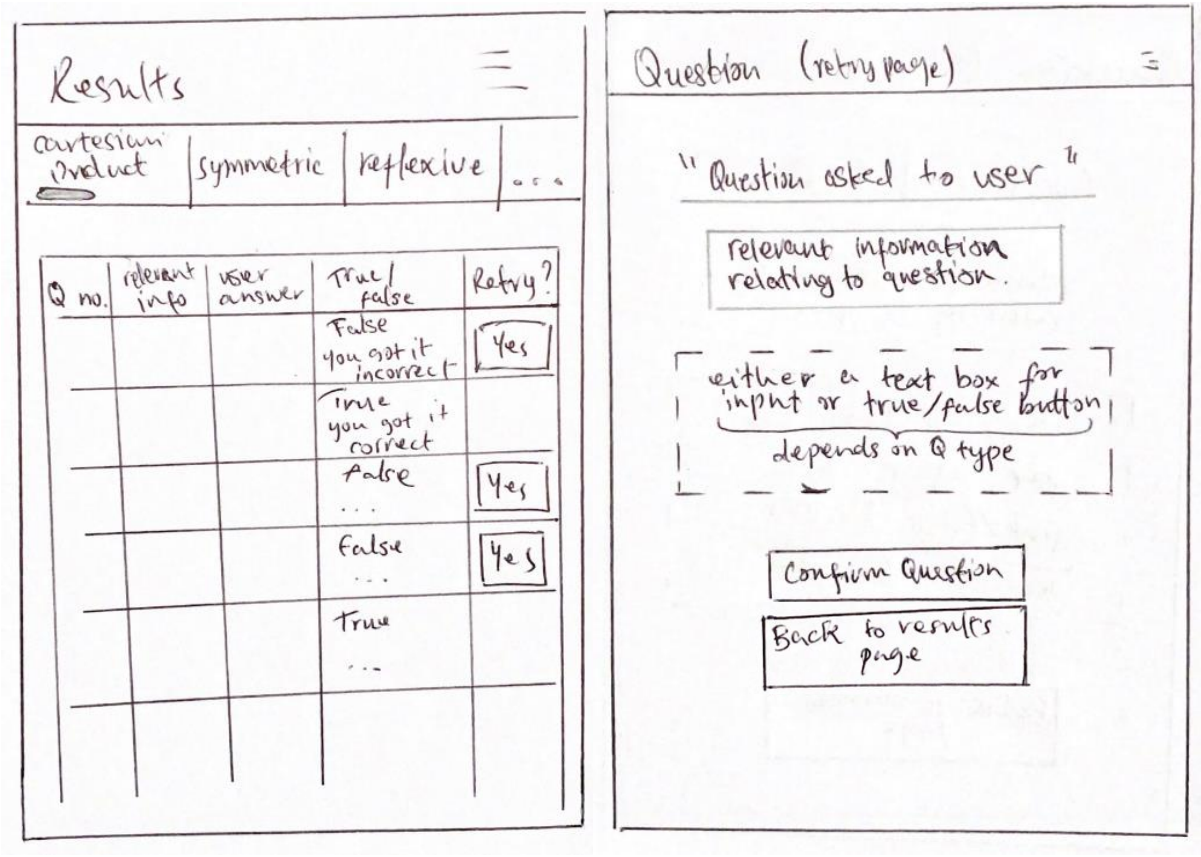


Figure 5.6: Results Activity Page Design and Retry Question Activity Page Design

Figure 5.6 shows results page on the left and the question page on the right for retries. The user will be directed to the results page when they have clicked on the corresponding button in the choose activity page (Figure 5.4). At the top of the results page, there a will be a vertical scroll bar for the different topic areas and below that is a table that will retrieve the questions that user has completed along with retry button at the end depending on whether they got the question correct or not. When the retry button is pressed, the user will be directed to the retry question page, which would display the question that use want to retry. It is based on the design from Figure 5.5 and at the bottom there is a button to redirect the use back to the results page.

Across all the pages, they should have the ability to either logout or go to either the home page or the results page.

5.4 Database Design

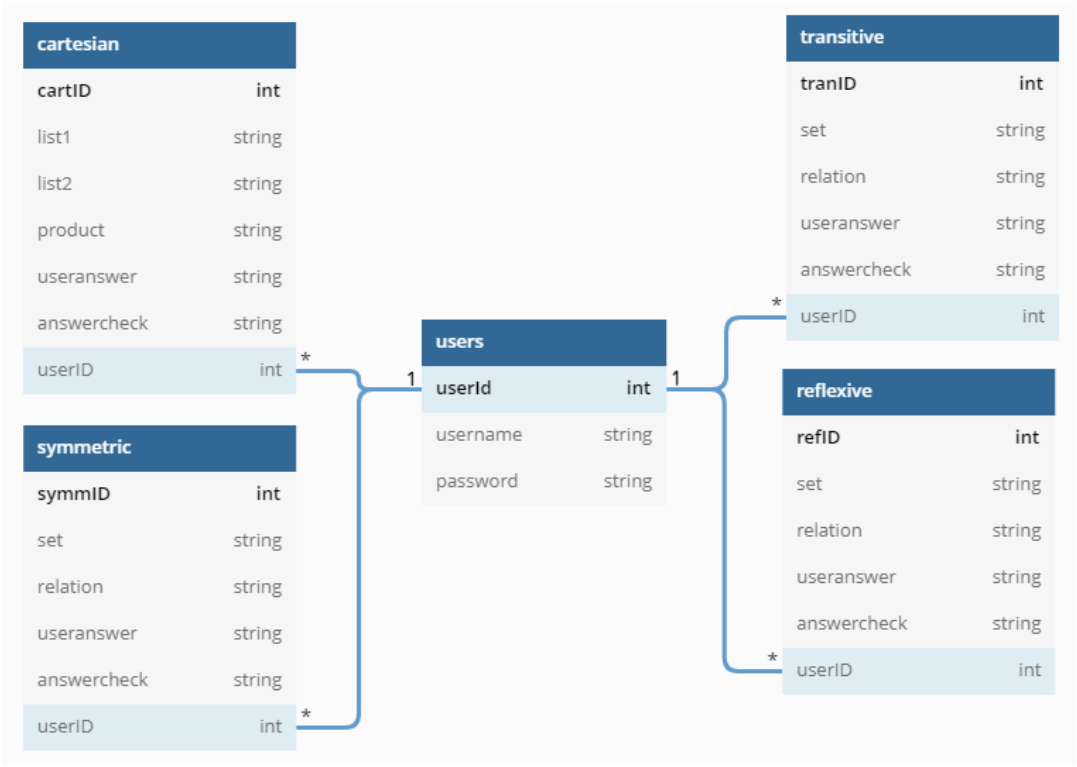


Figure 5.7: Database Entity Relationship Diagram

For the application, a local database will be used with SQLite. That means the database will only be stored on the user's device rather than on a database server. The database of the application will store the information of the user along with a hashed password for better security and the four table; cartesian, symmetric, transitive, reflexive, will each store the question that have been answered by the user along with the userID for ease of access using foreign keys.

5.5 Summary

This chapter has presented the decision designs for the base system architecture and UI elements. The designs outlined in this chapter will be used in the implementation as a foundation on how to design and develop the application.

6. Implementation

This chapter presents the implementation of the application and will outline each iteration of the code This application was developed for Android using Android Studio Bumblebee Version: 2021.1.1. The application was coded in Java for the classes and XML for the resource files and the database implemented was the SQLite.

6.1 Iteration 1

This first iteration focuses on creating the initial activity pages to satisfy the bare minimum requirements. The pages themselves will be based on the UI designs discussed in chapter 5.3.

6.1.1 Implementation

6.1.1.1 Navigation & Creation of Question Pages

Within Android Studios, activities can when first creating the application. Each activity is created with a corresponding Java class and XML resource file. The simplest choice for navigation was through the use of buttons. These were added in the XML file.

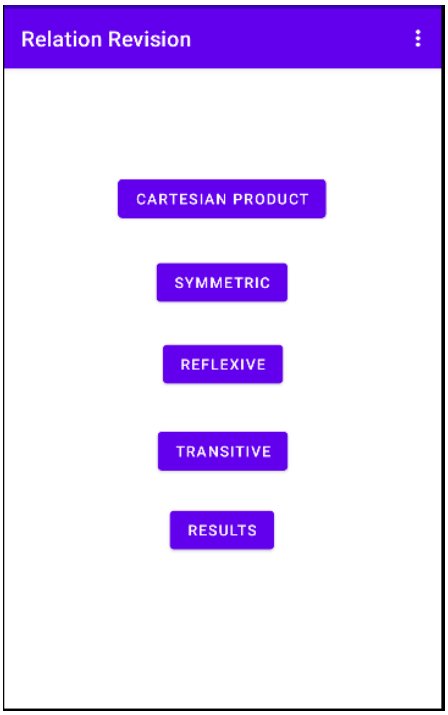


Figure 6.1: Home Page Layout

For the button to work, within the Java file, each button was initialised and on click listeners were set. Within each onClickListener, an Intent was created that will take in the current class, and the class that is required for the user to go to.

```
cartProd.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent buttonClick = new Intent( packageContext: choosePageActivity.this, cartesianProductActivity.class);
        startActivity(buttonClick);
    }
});

sym.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent buttonClick = new Intent( packageContext: choosePageActivity.this, symmetricQuestionActivity.class);
        startActivity(buttonClick);
    }
});

ref.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent buttonClick = new Intent( packageContext: choosePageActivity.this, reflexiveQuestionActivity.class);
        startActivity(buttonClick);
    }
});

tran.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent buttonClick = new Intent( packageContext: choosePageActivity.this, transitiveQuestionActivity.class);
        startActivity(buttonClick);
    }
});
```

Figure 6.2: OnClickListeners for Buttons on Home Page

Figure 6.3 represents the layout of the page for the questions regarding the cartesian products. In the top right corner, a button was created for the user when they require help, this button will also be present in the other layouts for questions (Figures 6.3, 6.4, 6.5). Below that is where the question asked, with the two default TextViews being a place holder for the sets that will be created. In the middle is a EditText box for the user input. Then finally are the two buttons, one for submission and the other for the next question.

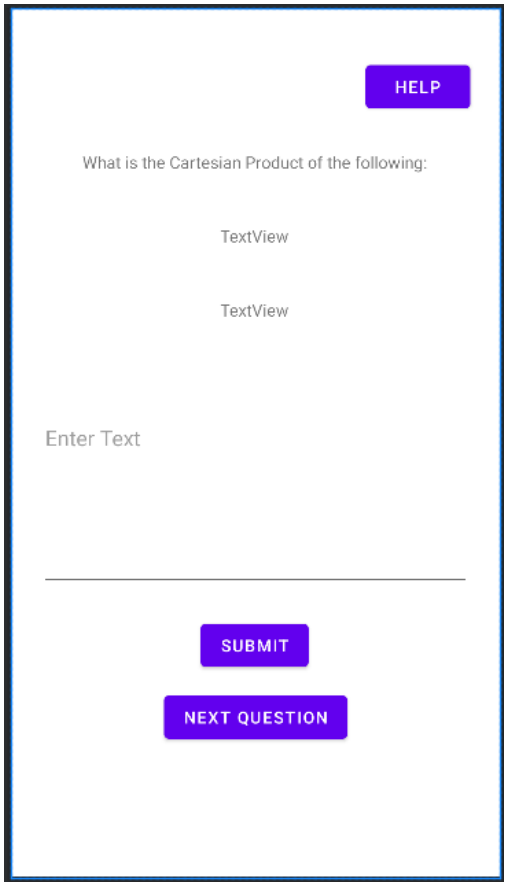


Figure 6.3: Cartesian Question Page Layout

Figure 6.4 represents the page layout for the questions regarding reflexive relations. The plan for the reflexive questions is to ask the user whether the relation they got is reflexive or not and if it is they click true, if not they click false. This done due to the nature of reflexive relations as relations are reflexive if they come back on themselves.

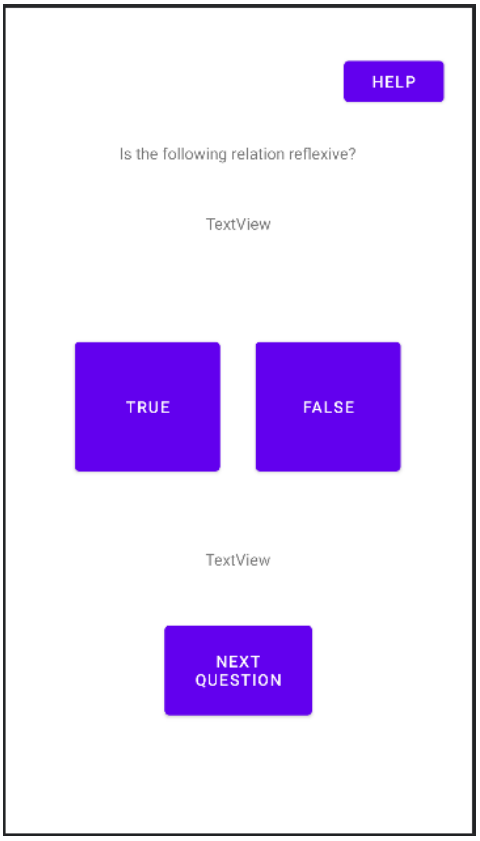


Figure 6.4: Reflexive Question Page Layout

Figure 6.5 represents the page layout for the questions regarding symmetric relations. The user will be asked whether the relation is symmetric or not. If it is then the user clicks true and enables the EditText boxes and submit button, allowing them to input their answer. The TextView below the submit button will give the user a message whether they got it right or not.

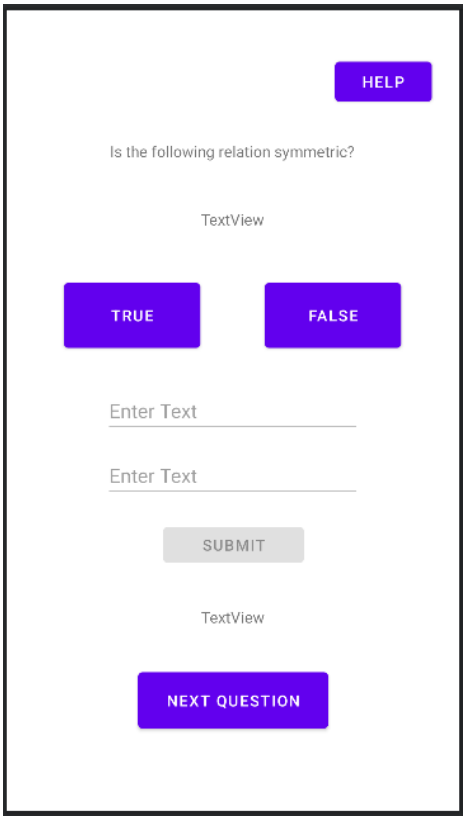


Figure 6.5: Symmetric Question Page Layout

Figure 6.6 represents the page layout for the questions regarding transitive relations. The user will be asked to input the values needed for the question or not. The TextView below the submit button will give the user a message whether they got it right or not.

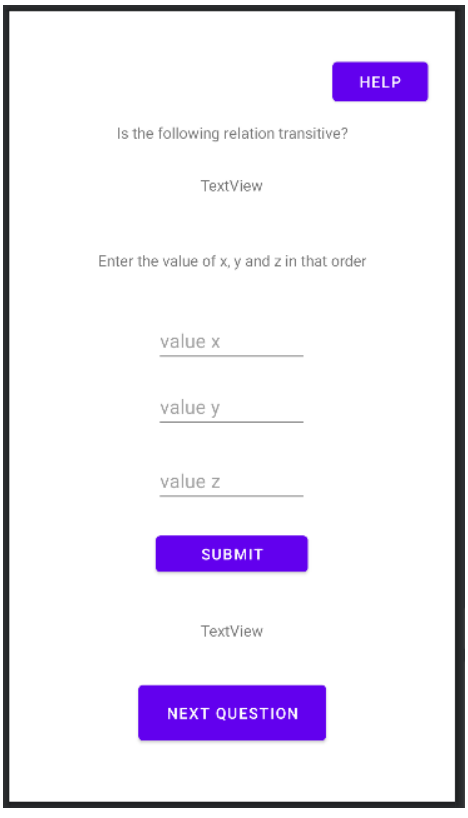


Figure 6.6: Transitive Question Page Layout

Figure 6.7 displays the showMessage function within the cartesianProductActivity class. This will create an alert dialog to show messages to the user. This function will be present throughout the application.

```
public void showMessage(String message) {
    AlertDialog dialog = new AlertDialog.Builder( context: cartesianProductActivity.this)
            .setMessage(message)
            .setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    dialogInterface.dismiss();
                }
            }).create();
    dialog.show();
}
```

Figure 6.7: showMessage Function

```
helpButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        showMessage("If A and B are sets, we call A × B the Cartesian product of A and B:\n" +
                "\n" +
                "A × B = {(a, b) | a ∈ A and b ∈ B}");
    }
});
```

```
helpButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        showMessage("Let R be a binary relation on a set A.\n" +
                "R is reflexive if and only if (x, x) ∈ R for all x ∈ A.");
    }
});
```

```
helpButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        showMessage("Let R be a binary relation on a set A.\n" +
                "R is symmetric if and only if for all x, y ∈ A if (x, y) ∈ R then " +
                "(y, x) ∈ R");
    }
});
```

```
helpButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        showMessage("Let R be a binary relation on a set A.\n" +
                "R is transitive if and only if for all x, y, z ∈ A if (x, y) ∈ R and " +
                "(y, z) ∈ R then (x, z) ∈ R");
    }
});
```

Figure 6.8: Help Button OnClickListener for each Question Page

6.1.1.2 Creation of Action Bar

The action bar is created for their use within each of the activity page. Which action bar is used will depend on the activity page itself.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/itemLogoutMain"
        android:title="Logout"
        android:onClick="logoutIntent"
        app:showAsAction="never"/>

</menu>
```

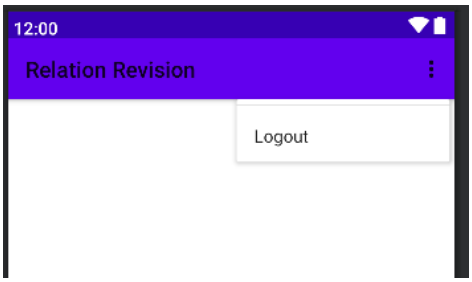


Figure 6.9: XML Code and Layout for Logout Action Bar

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/itemHomeL"
        android:title="Home"
        android:onClick="homeIntent"
        app:showAsAction="never"/>


    <item android:id="@+id/itemLogoutHomeL"
        android:title="Logout"
        android:onClick="logoutIntent"
        app:showAsAction="never"/>

</menu>
```

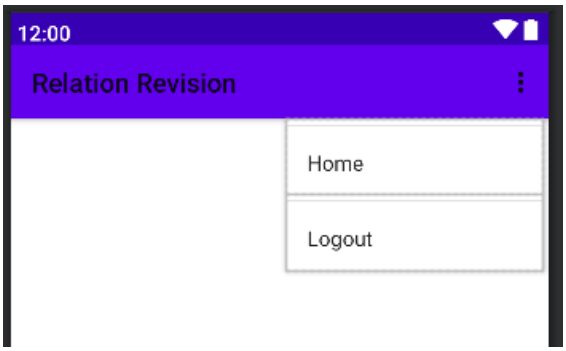


Figure 6.10: XML Code and Layout for Home & Logout Action Bar

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/itemResults"
        android:title="Results"
        android:onClick="resultIntent"
        app:showAsAction="never"/>

    <item android:id="@+id/itemLogout"
        android:title="Logout"
        android:onClick="logoutIntent"
        app:showAsAction="never"/>

</menu>
```

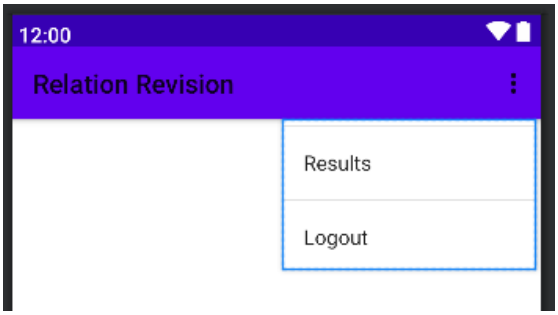


Figure 6.11: XML Code and Layout for Results & Logout Action Bar

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/itemHome"
        android:title="Home"
        android:onClick="homeIntent"
        app:showAsAction="never"/>

    <item android:id="@+id/itemResultsHome"
        android:title="Results"
        android:onClick="resultIntent"
        app:showAsAction="never"/>

    <item android:id="@+id/itemLogoutHome"
        android:title="Logout"
        android:onClick="logoutIntent"
        app:showAsAction="never"/>

</menu>
```

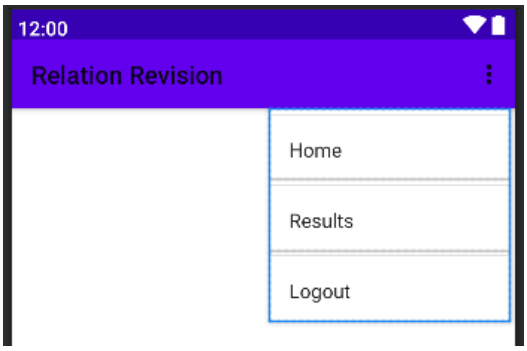


Figure 6.12: XML Code and Layout for Home, Results & Logout Action Bar

Figure 6.13 displays the function of the action bar that will be used throughout the application. The onCreateOptionsMenu is used to display the action bar on the pages. The Intent functions are used for navigational purposes.

```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_home_result_logout, menu);
    return true;
}

public Boolean logoutIntent(MenuItem view) {
    Intent intent = new Intent( packageContext: cartesianProductActivity.this, loginActivity.class);
    startActivity(intent);
    return true;
}

public Boolean resultIntent(MenuItem view) {
    Intent intent = new Intent( packageContext: cartesianProductActivity.this, cartesianResultsActivity.class);
    intent.putExtra(loginActivity.USERID, userID);
    startActivity(intent);
    return true;
}

public Boolean homeIntent(MenuItem view) {
    Intent intent = new Intent( packageContext: cartesianProductActivity.this, choosePageActivity.class);
    intent.putExtra(loginActivity.USERID, userID);
    startActivity(intent);
    return true;
}
```

Figure 6.13: Functions for Action Bar – Image in context to cartesianProductActivity Class

6.1.1.3 Cartesian Class & Implementation

Cartesian product works by combining the first value of the set with the rest of the values within the set. Firstly, two sets would have to be created (Figure 6.14) then those sets would be displayed to the user.

```
// create a the first list (character list) by creating a random sized array
public ArrayList<Character> list1() {
    Random rand = new Random();
    int low = 3;
    int high = 5;
    int result = rand.nextInt( bound: high - low) + low;

    list1 = new ArrayList<>(result);
    for (int i = 0; i <= result; i++) {
        list1.add((char)(rand.nextInt( bound: 26) + 'a')); // add random letter from a-z
    }

    return list1;
}
// creates the second list (integer list)
public ArrayList<Integer> list2() {
    Random rand = new Random();
    int low = 3;
    int high = 5;
    int result = rand.nextInt( bound: high - low) + low;

    list2 = new ArrayList<>(result);

    for (int i = 0; i <= result; i++) {
        list2.add(rand.nextInt( bound: 100));
    }

    return list2;
}
```

Figure 6.14: Set Creation Functions

The function present in Figure 6.15 works by taking in the two randomly created sets and create their respective cartesian products sets, add them to the array list and then returns it so that it can be used for answer checking

```
// create a list of cartesian product using the two list from the parameter
public ArrayList<String> listProduct(ArrayList<Character> setA, ArrayList<Integer> setB) {
    ArrayList<String> product = new ArrayList<~>();

    if (setA.size() == 0) { return product; }
    if (setB.size() == 0) { return product; }

    for (Character i: setA) {
        for (Integer j: setB) {
            String cProd = "(" + i + "," + j + ")";
            product.add(cProd);
        }
    }

    return product;
}
```

Figure 6.15: Cartesian Product create Function

The answerCheck function in Figure 6.16 is used to check how many of the inputted answers the user gave is correct. Each time the answer is correct the correctCount variable increase by one and the incorrectCount variable works in the same principle. Both counts are then added to an array list and returned.

```java
// checks how many answers the user gave was correct
public ArrayList<Integer> answerCheck(ArrayList<String> answer, String userAns) {
    int correctCount = 0;
    int incorrectCount = 0;

    for (int i = 0; i <= answer.size()-1; i++) {
        if (userAns.contains(answer.get(i))) { correctCount += 1; } //true
        else { incorrectCount += 1; } //false
    }

    ArrayList<Integer> count = new ArrayList<>();
    count.add(correctCount);
    count.add(incorrectCount);

    return count;
}
```

Figure 6.16: answerCheck Function

```java
submitButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String uInput = editText.getText().toString();
        ArrayList<Integer> check = cart.answerCheck(cart.listProduct(cL, iL), uInput);

        int correct = check.get(0);
        int incorrect = check.get(1);
        int total = correct + incorrect;

        if (uInput.isEmpty()) { //editText is empty
            showMessage("Please enter your answer");
        }
        else if (correct > 0 && incorrect == 0) { // all of them are correct
            showMessage("You got " + correct + " out of " + total);
        }
        else if (correct >= 0 && incorrect > 0) { // some of them are correct
            showMessage("You got " + correct + " out of " + total);
        }
        else { // none of them are correct
            showMessage("You got none correct");
        }
    }
});
```

Figure 6.17: onClickListener for the Submit Button

```
// create a new question when button is clicked
nextQuestion.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        editText.setText("");
        cL = cart.list1();
        iL = cart.list2();
        String cLStr = "L = " + cL.toString();
        String iLStr = "M = " + iL.toString();
        list1View.setText(cLStr);
        list2View.setText(iLStr);
    }
});
```

Figure 6.18: onClickListener for the Next Question Button

```
trueClick.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (rel.isSymmetric(listSort)) {
            editText1.setEnabled(true);
            editText2.setEnabled(true);
            confirmButton.setEnabled(true);
        }
        else {
            message.setText("It Seems You Answer is Incorrect. Please Try Again");
        }
    }
});
```

Figure 6.19: onClickListener for True Button in the symmetricQuestionActivity Class

6.1.2 Testing

Test Description	Result
Navigation: Home Page to Cartesian Product Question Page	Success
Navigation: Home Page to Reflexive Question Page	Success
Navigation: Home Page to Symmetric Question Page	Success
Navigation: Home Page to Transitive Question Page	Success
Navigation: Home Page to Results Page	Failure
Navigation Action Bar: Home Page to Login Page (for logout)	Failure
Navigation Action Bar: Cartesian Product Question Page to Login Page (for logout)	Failure
Navigation Action Bar: Cartesian Product Question Page to Results Page	Success
Navigation Action Bar: Cartesian Product Question Page to Home Page	Success
Display information relating to cartesian product question	Success
Input answer in EditText box	Success
Verify user input answer with the generated answer	Success
Display how many the user got right when clicked confirm	Success
Replace the current question with new question when clicked Next Question button	Success

Cartesian Question Page: Display help message when help button is clicked	Success
Symmetric Question Page: Display help message when help button is clicked	Success
Transitive Question Page: Display help message when help button is clicked	Success
Reflexive Question Page: Display help message when help button is clicked	Success
Symmetric Question Page: When true button is clicked, the EditText box and submit button is enabled	Success

Table 6.1: Iteration 1 Testing Table

6.1.3 Review

This iteration implemented the main navigational feature for the application as well as the class for creating the data for the cartesianProductActivity page. The failures that were observed during the testing were the navigational features as that requires the presence of the specific page which have not been made yet.

6.2 Iteration 2

6.2.1 Implementation

6.2.1.1 Relations Class & Implementation

The three main topic within relation are symmetric relations, reflexive relations, and transitive relations. The relations class with be used for both creating the set and relation as well as the function used to verify whether the relation is symmetric, reflexive or transitive.

The function in figure 6.20 is used through out all three relation question activity classes as it is needed to create the set that relation is going to be based. The function works by randomly generating a size, determined as either 4 or 5 otherwise the relation could possibly become too large.

```
public ArrayList<Integer> createSet() {
    rand = new Random();
    int res = rand.nextInt( bound: 2);

    ArrayList<Integer> set = new ArrayList<>();
    if (res == 1) {
        for (int i = 0; i < 4; i++) {
            set.add(i + 1);
        }
    }
    else {
        for (int i = 0; i < 5; i++) {
            set.add(i + 1);
        }
    }
    return set;
}
```

Figure 6.20: createSet Function

Within the class a function exists to create all possible relations between the indexes from the set that is taken from the parameter (Figure 6.21), that relation set is returned and in turn used in another function to create a relation set that will be used for the question. The function would randomly

generate an index number and copy it from inputted relation set to its own which is then returned and used to display to the user (Figure 6.22).

```
// create all possible relations between the indexes
public ArrayList<ArrayList<Integer>> relData2d(ArrayList<Integer> set) {
    ArrayList<ArrayList<Integer>> data2d = new ArrayList<~>();
    int counter = 0;
    for (Integer i: set) {
        for (Integer j: set) {
            data2d.add(new ArrayList<Integer>()); // create new list for each new relation
            data2d.get(counter).add(i);
            data2d.get(counter).add(j);
            counter = counter + 1;
        }
    }
```

Figure 6.21: relData2d Function

```
// create a relation set by randomly choosing those from parameter
public ArrayList<ArrayList<Integer>> relListCreate(ArrayList<ArrayList<Integer>> set) {
    ArrayList<ArrayList<Integer>> rel2d = new ArrayList<>();
    rand = new Random();
    int setSize = set.size();
    int size = rand.nextInt(setSize);
    if (setSize == 16) {
        for (int i = 0; i < size; i++) {
            int randSelect = rand.nextInt(setSize);
            ArrayList<Integer> temp = set.get(randSelect);
            if (!rel2d.contains(temp)) {
                rel2d.add(temp);
            }
        }
    }
    else {
        for (int i = 0; i < size; i++) {
            int randSelect = rand.nextInt(setSize);
            ArrayList<Integer> temp = set.get(randSelect);
            if (!rel2d.contains(temp)) {
                rel2d.add(temp);
            }
        }
    }

    return rel2d;
}
```

Figure 6.22: relListCreate function

The isSymmetric function would be used when the user has inputted the values when answering the question and clicked the confirm button. The function would take in the relation that was given to the user and check through each pair to see whether there is a symmetric relation or not, if there is then the function would return true otherwise it would return false. Refer to Figure 6.19 for use.

The user inputs would be stored and put into a temporary array list and then check if the value is in the relation, if it is then the answer would be correct otherwise it would be false (Figure 6.24)

```java
public boolean isSymmetric(ArrayList<ArrayList<Integer>> set) {
    ArrayList<ArrayList<Integer>> temp = new ArrayList<>();
    temp.add(new ArrayList<>());
    temp.add(new ArrayList<>());
    for (ArrayList<Integer> i: set) {
        if (temp.get(0).isEmpty()) {
            temp.get(0).add( index: 0, i.get(0));
            temp.get(0).add( index: 1, i.get(1));
            temp.get(1).add( index: 0, i.get(1));
            temp.get(1).add( index: 1, i.get(0));
        }
        else {
            temp.get(0).set(0, i.get(0));
            temp.get(0).set(1, i.get(1));
            temp.get(1).set(0, i.get(1));
            temp.get(1).set(1, i.get(0));
        }

        if (set.contains(temp.get(0)) && set.contains(temp.get(1))) {
            return true;
        }
    }
    return false;
}
```

Figure 6.23: isSymmetric Function

```java
confirmButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Integer v1 = Integer.parseInt(editText1.getText().toString());
        Integer v2 = Integer.parseInt(editText2.getText().toString());

        ArrayList<ArrayList<Integer>> temp = new ArrayList<>();
        temp.add(new ArrayList<>());
        temp.add(new ArrayList<>());
        temp.get(0).add(v1);
        temp.get(0).add(v2);
        temp.get(1).add(v2);
        temp.get(1).add(v1);

        if (listSort.contains(temp.get(0)) && listSort.contains(temp.get(1))) {
            message.setText("Well Done! Your Answer is Correct");

        }
        else {
            message.setText("It Seems You Answer is Incorrect. Please Try Again");
        }
    }
});
```

Figure 6.24: OnClickListener for Confirm Button in symmetricQuestionActivity Class

The isReflexive function checks to see if the relation that is given to the user a reflexive relation or not. It works by creating an array list with all possible reflexive relation values then checks to see if the parameter (the relation set give to user) contains one of the reflexive relation values.

```java
public boolean isReflexive(ArrayList<ArrayList<Integer>> data) {
    ArrayList<ArrayList<Integer>> refList = new ArrayList<>();
    refList.add(new ArrayList<>());
    refList.add(new ArrayList<>());
    refList.add(new ArrayList<>());
    refList.add(new ArrayList<>());
    refList.add(new ArrayList<>());

    refList.get(0).add(1);
    refList.get(0).add(1);
    refList.get(1).add(2);
    refList.get(1).add(2);
    refList.get(2).add(3);
    refList.get(2).add(3);
    refList.get(3).add(4);
    refList.get(3).add(4);
    refList.get(4).add(5);
    refList.get(4).add(5);

    for (ArrayList<Integer> i: refList) {
        if (data.contains(i)) {
            return true;
        }
    }
    return false;
}
```

Figure 6.25: isReflexive Function

The isTransitive function takes in two parameters. It would loop through the generated relation list that is given to the user and check to see if and only if the first two user input relation values are in the set, then it checks to see if the last value is in the relation set, only then would it return true otherwise everything the function would return false.

```java
public boolean isTransitive(ArrayList<ArrayList<Integer>> data, ArrayList<ArrayList<Integer>> uInput) {
    for (ArrayList<Integer> i: data) {
        if (data.contains(uInput.get(0)) && data.contains(uInput.get(1))) {
            if (data.contains(uInput.get(2))) {
                return true;
            }
        }
    }
    return false;
}
```

Figure 6.26: isTransitive Function

```
confirmButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Integer v1 = Integer.parseInt(x.getText().toString());
        Integer v2 = Integer.parseInt(y.getText().toString());
        Integer v3 = Integer.parseInt(z.getText().toString());

        if (v1 == null || v2 == null || v3 == null) {
            showMessage("Please enter the values");
        }
        else {
            ArrayList<ArrayList<Integer>> temp = new ArrayList<>();
            temp.add(new ArrayList<>());
            temp.add(new ArrayList<>());
            temp.add(new ArrayList<>());

            temp.get(0).add(v1);
            temp.get(0).add(v2);
            temp.get(1).add(v2);
            temp.get(1).add(v3);
            temp.get(2).add(v1);
            temp.get(2).add(v3);

            if (rel.isTransitive(listSort, temp)) {
                message.setText("Well Done! Your Answer is Correct");
            }
            else {
                message.setText("It Seems You Answer is Incorrect. Please Try Again");
            }
        }
    }
```

Figure 6.27: OnClickListener for Confirm Button in transitiveQuestionActivity Class

6.2.1.2 Creation of Database

It has been decided, in Chapter 5.4, that SQLite would be used for the database as it allows for local storage of data, which can also allow for offline usage.

```
public class DBHelper extends SQLiteOpenHelper {

    public DBHelper(Context context) { super(context, name: "FYP_926975.db", factory: null, version: 1); }
    // create database
    @Override
    public void onCreate(SQLiteDatabase myDB) {
        myDB.execSQL("create Table users(userid INTEGER primary key AUTOINCREMENT," +
                                        "username TEXT," +
                                        "password TEXT)");
        myDB.execSQL("create Table cartesian(cartid INTEGER primary key AUTOINCREMENT," +
                                            "list1 TEXT," +
                                            "list2 TEXT," +
                                            "product TEXT," +
                                            "useranswer TEXT," +
                                            "answercheck TEXT," +
                                            "userID INTEGER," +
                                            "foreign key (userID) REFERENCES user(userid))");
        myDB.execSQL("create Table symmetric(symmid INTEGER primary key AUTOINCREMENT," +
                                            "setlist TEXT," +
                                            "relation TEXT," +
                                            "useranswer TEXT," +
                                            "answercheck TEXT," +
                                            "userID INTEGER," +
                                            "foreign key (userID) REFERENCES user(userid))");
        myDB.execSQL("create Table reflexive(refid INTEGER primary key AUTOINCREMENT," +
                                            "setlist TEXT," +
                                            "relation TEXT," +
                                            "useranswer TEXT," +
                                            "answercheck TEXT," +
                                            "userID INTEGER," +
                                            "foreign key (userID) REFERENCES user(userid))");
        myDB.execSQL("create Table transitive(tranid INTEGER primary key AUTOINCREMENT," +
                                            "setlist TEXT," +
                                            "relation TEXT," +
                                            "useranswer TEXT," +
                                            "answercheck TEXT," +
                                            "userID INTEGER," +
                                            "foreign key (userID) REFERENCES user(userid))");
    }
```

Figure 6.28: DBHelper Class Constructor & onCreate Method

```
// drop database if exists
@Override
public void onUpgrade(SQLiteDatabase myDB, int oldVersion, int newVersion) {
    myDB.execSQL("drop Table if exists users");
    myDB.execSQL("drop Table if exists cartesian");
    myDB.execSQL("drop Table if exists symmetric");
    myDB.execSQL("drop Table if exists reflexive");
    myDB.execSQL("drop Table if exists transitive");
}
```

Figure 6.29: onUpgrade Method

```
// check data
public Boolean checkUsername(String username) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: "Select * from users where username = ?", new String[] {username});
    if (cursor.getCount() > 0){
        return true;
    }
    else {
        return false;
    }
}

public Boolean checkPassword(String password) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: "Select * from users where password = ?", new String[] {password});
    if (cursor.getCount() > 0) {
        return true;
    }
    else {
        return false;
    }
}
```

Figure 6.30: checkUsername & checkPassword Methods

```java
public Boolean checkCartQuestion(Integer id, Integer uID) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: "Select * from cartesian where cartid = ? and userID = " + uID, new String[] {id.toString()});
    if (cursor.getCount() > 0) {
        return true;
    }
    else {
        return false;
    }
}

public boolean checkTranQuestion(Integer id, Integer uID) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: "Select * from transitive where tranid = ? and userID = " + uID, new String[] {id.toString()});
    if (cursor.getCount() > 0) {
        return true;
    }
    else {
        return false;
    }
}

public boolean checkSymQuestion(Integer id, Integer uID) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: "Select * from symmetric where symmid = ? and userID = " + uID, new String[] {id.toString()});
    if (cursor.getCount() > 0) {
        return true;
    }
    else {
        return false;
    }
```

Figure 6.31: checkCart Question, checkTranQuestion, checkSymQuestion Methods

```java
// insert data
public Boolean insertUserData(String username, String password) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues userList = new ContentValues();
    userList.put("username", username);
    userList.put("password", password);
    myDB.insert( table: "users",  nullColumnHack: null, userList);
    return true;
}

public Boolean insertCartesianData(String list1, String list2, String product, String userAnswer, String answerCheck, Integer id) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues contents = new ContentValues();
    contents.put("list1", list1);
    contents.put("list2", list2);
    contents.put("product", product);
    contents.put("useranswer", userAnswer);
    contents.put("answercheck", answerCheck);
    contents.put("userID", id);
    myDB.insert( table: "cartesian",  nullColumnHack: null, contents);
    return true;
}
```

Figure 6.32: insertUserData & insertCartesianData Methods

```
public Boolean insertSymmetricData(String list, String relation, String userAnswer, String answerCheck, Integer id) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues contents = new ContentValues();
    contents.put("setlist", list);
    contents.put("relation", relation);
    contents.put("useranswer", userAnswer);
    contents.put("answercheck", answerCheck);
    contents.put("userID", id);
    myDB.insert( table: "symmetric",  nullColumnHack: null, contents);
    return true;
}

public Boolean insertReflexiveData(String list, String relation, String userAnswer, String answerCheck, Integer id) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues contents = new ContentValues();
    contents.put("setlist", list);
    contents.put("relation", relation);
    contents.put("useranswer", userAnswer);
    contents.put("answercheck", answerCheck);
    contents.put("userID", id);
    myDB.insert( table: "reflexive",  nullColumnHack: null, contents);
    return true;
}
```

Figure 6.32: insertSymmetricData & insertReflexiveData Methods

```
public Boolean insertTransitiveData(String list, String relation, String userAnswer, String answerCheck, Integer id) {
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues contents = new ContentValues();
    contents.put("setlist", list);
    contents.put("relation", relation);
    contents.put("useranswer", userAnswer);
    contents.put("answercheck", answerCheck);
    contents.put("userID", id);
    myDB.insert( table: "transitive",  nullColumnHack: null, contents);
    return true;
}
```

Figure 6.33: insertTransitiveData Method

```
// get data
public Cursor getCartesiandata(Integer input) {
    SQLiteDatabase DB = this.getWritableDatabase();
    Cursor cursor  = DB.rawQuery( sql: "Select * from cartesian where userid = " + input,  selectionArgs: null);
    return cursor;
}

public Cursor getUserid(String text) {
    SQLiteDatabase DB = this.getWritableDatabase();
    Cursor cursor = DB.rawQuery( sql: "Select * from users where username = ?", new String[] {text});
    return cursor;
}

public Cursor getReflexivedata(Integer input) {
    SQLiteDatabase DB = this.getWritableDatabase();
    Cursor cursor  = DB.rawQuery( sql: "Select * from reflexive where userid = " + input,  selectionArgs: null);
    return cursor;
}

public Cursor getSymmetricData(Integer input) {
    SQLiteDatabase DB = this.getWritableDatabase();
    Cursor cursor  = DB.rawQuery( sql: "Select * from symmetric where userid = " + input,  selectionArgs: null);
    return cursor;
}

public Cursor getTransitiveData(Integer input) {
    SQLiteDatabase DB = this.getWritableDatabase();
    Cursor cursor  = DB.rawQuery( sql: "Select * from transitive where userid = " + input,  selectionArgs: null);
    return cursor;
}
```

Figure 6.34: getCartesiandata, getUserid, getReflexivedata, getSymmetricData & getTransitiveData Methods

```
public Cursor getSpecificCartdata(Integer input, Integer user) {
    SQLiteDatabase DB = this.getWritableDatabase();
    Cursor cursor = DB.rawQuery( sql: "Select * from cartesian where cartid = " + input + " and userid = " + user,  selectionArgs: null);
    return cursor;
}

public Cursor getSpecificTransdata(Integer input, Integer user) {
    SQLiteDatabase DB = this.getWritableDatabase();
    Cursor cursor = DB.rawQuery( sql: "Select * from transitive where tranid = " + input + " and userid = " + user,  selectionArgs: null);
    return cursor;
}

public Cursor getSpecificSymdata(Integer input, Integer user) {
    SQLiteDatabase DB = this.getWritableDatabase();
    Cursor cursor = DB.rawQuery( sql: "Select * from symmetric where symmid = " + input + " and userid = " + user,  selectionArgs: null);
    return cursor;
}
```

Figure 6.35: getSpecificCartdata, getSpecificTransdata & getSpecificSymdata Methods

6.2.1.3 Creation of Login Page

When the application first starts, the user will first be shown a page to log in. If the user does not already possess an account, then they prompted to create one.

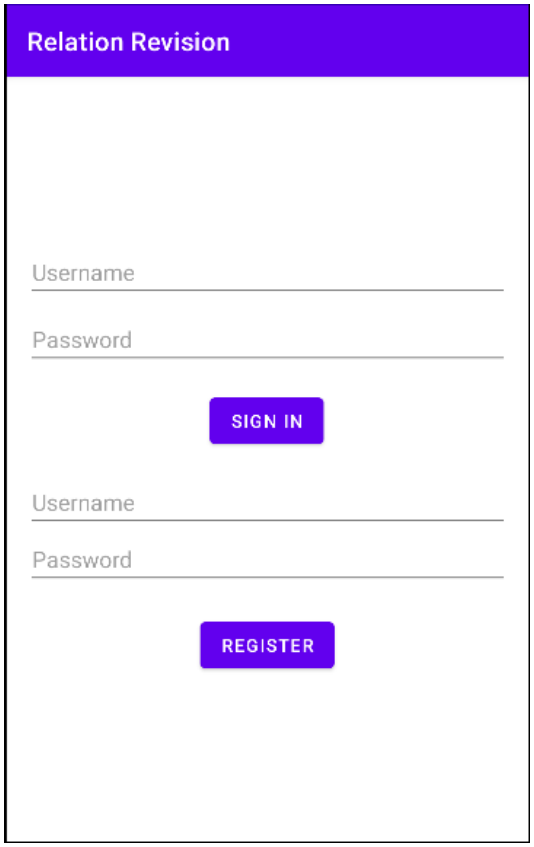


Figure 6.36: Login / Register Page

```
public static final String USERID = "USERID"; // used for passing variable through intent
```

Figure 6.37: Variable for passing the user id.

When the login button is clicked, both the username and password are stored in their respective variables. The password is also then hashed for encryption, using MD5 (Figure 6.42). Both username and password variable will be checked to see if either of the variables are null, if the password is incorrect or if the user does not exist. If any of these checks turn out to be true, then the corresponding message would be displayed to the user instructing them on what to do next. User can also log out using the action bar (Chapter 6.1.1.2)

If both the username and hashed password is correct, then the user would be logged in and directed to the home page through the use of intents. The intent would also pass the user id of the account, taken from the database to the home page. The intents can also be seen being used in Figure 6.13.

```
loginButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String uN = loginUsername.getText().toString();
        String pW = loginPassword.getText().toString();
        String hashedPW = md5(pW); // encrypted password

        if (uN.isEmpty() || pW.isEmpty()) { //empty editText
            showMessage("Please enter your username and/or password.");
        } // both username and password are correct
        else if (db.checkUsername(uN) == true && db.checkPassword(hashedPW) == true) {
            Intent intent = new Intent( packageContext: loginActivity.this, choosePageActivity.class);

            String uID = "";
            Cursor user = db.getUserid(uN);
            while (user.moveToNext()) {
                uID = user.getString( i: 0);
            }
            Integer userID = Integer.parseInt(uID);

            intent.putExtra(USERID, userID);
            startActivity(intent);
        }
        else if (db.checkUsername(uN) == false) { // username is incorrect
            showMessage("Cannot find username. Please try again or register a new account.");
        } // username is correct but password is not
        else if (db.checkUsername(uN) == true && db.checkPassword(hashedPW) == false) {
            showMessage("Password is incorrect. Please try again.");
        }
    }
});
```

Figure 6.38: OnClickListener for Login Button

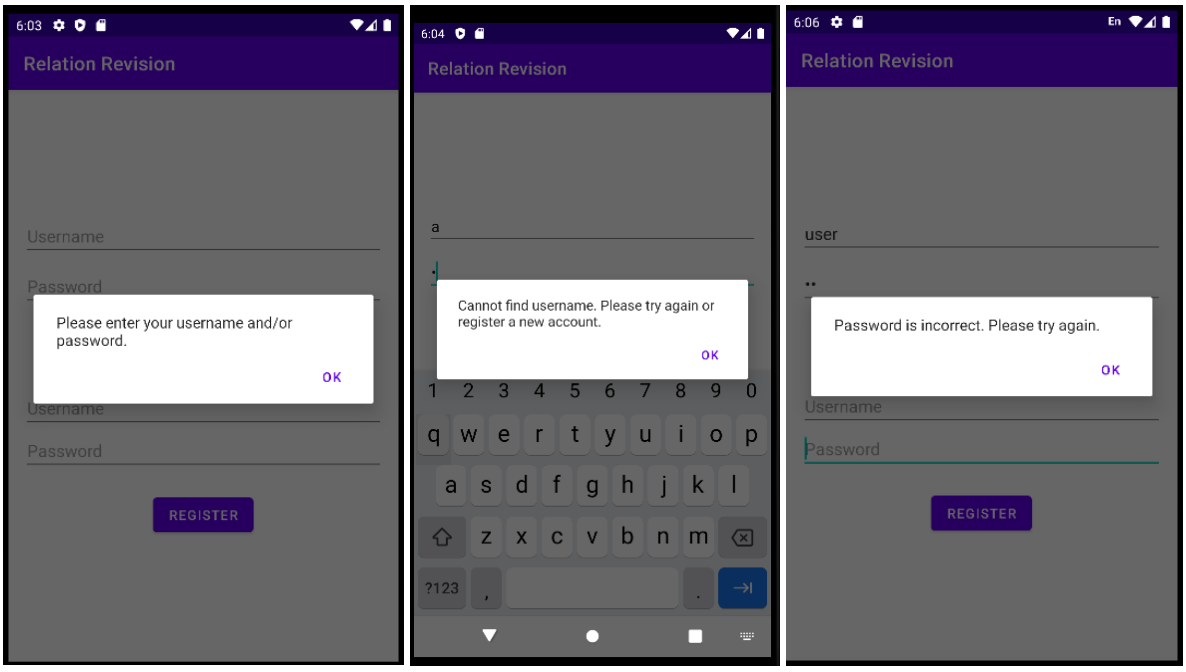


Figure 6.39: Login Error Messages

If the user does not have account, they can make one by using the registration boxes (Figure 6.36). If both entry fields are empty, then a message will be displayed to the user asking them to enter a username and/or password. If the username they entered already exists, then the user will be asked to enter a different one. Lastly if the username doesn't exist then both the username and hashed password would be added to the database.

```
registerButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String uN = registerUsername.getText().toString();
        String pW = registerPassword.getText().toString();
        String hashedPW = md5(pW);

        if (uN.isEmpty() || pW.isEmpty()) { // empty editText
            showMessage("Please enter a username and/or password.");
        }
        else if (db.checkUsername(uN) == true) { // checks if user already exists
            showMessage("Username already exists. Please try again.");
        }
        else { // otherwise add the user to database
            if (db.insertUserData(uN, hashedPW)) {
                showMessage("You are now registered");
                registerUsername.setText(""); // clears the editText boxes
                registerPassword.setText("");
            }
        }
    }
});
```

Figure: 6.40: OnClickListener for Register Button

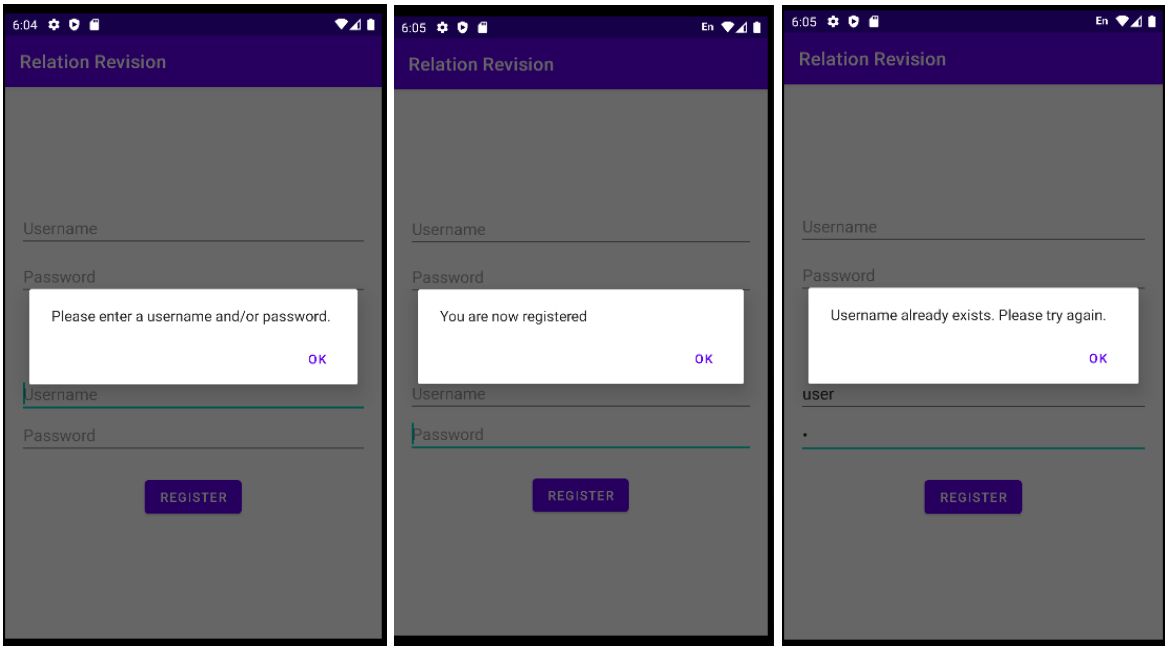


Figure 6.41: Register Messages

```java
// used for encryption of password
public static String md5(final String s) {
    final String MD5 = "MD5";
    try {
        // Create MD5 Hash
        MessageDigest digest = java.security.MessageDigest.getInstance(MD5);
        digest.update(s.getBytes());
        byte messageDigest[] = digest.digest();

        // Create Hex String
        StringBuilder hexString = new StringBuilder();
        for (byte aMessageDigest : messageDigest) {
            String h = Integer.toHexString( i: 0xFF & aMessageDigest);
            while (h.length() < 2)
                h = "0" + h;
            hexString.append(h);
        }
        return hexString.toString();

    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    return "";
}
```

Figure 6.42: MD5 Hashing Function

6.2.1.4 Database Implementation Throughout the Quiz Pages

Now that the database has been added, the details regarding the questions can be added to the database.

```java
submitButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String uInput = editText.getText().toString();
        ArrayList<Integer> check = cart.answerCheck(cart.listProduct(cL, iL), uInput);

        int correct = check.get(0);
        int incorrect = check.get(1);
        int total = correct + incorrect;

        if (uInput.isEmpty()) { //editText is empty
            showMessage("Please enter your answer");
        }
        else if (correct > 0 && incorrect == 0) { // all of them are correct
            showMessage("You got " + correct + " out of " + total);
            db.insertCartesianData(cL.toString(), iL.toString(), cart.listProduct(cL, iL).toString(), uInput,  answerCheck: "True", userID);
        }
        else if (correct >= 0 && incorrect > 0) { // some of them are correct
            showMessage("You got " + correct + " out of " + total);
            db.insertCartesianData(cL.toString(), iL.toString(), cart.listProduct(cL, iL).toString(), uInput,  answerCheck: "False", userID);
        }
        else { // none of them are correct
            showMessage("You got none correct");
            db.insertCartesianData(cL.toString(), iL.toString(), cart.listProduct(cL, iL).toString(), uInput,  answerCheck: "False", userID);
        }
    }
});
```

Figure 6.43: OnClickListener for Submit Button in cartesianProductActivity Class with Database Integration

```
trueClick.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (rel.isReflexive(listSort) == true) { // answer is correct
            message.setText("Well Done! Your Answer is Correct");
            db.insertReflexiveData(set.toString(), listSort.toString(),  userAnswer: "true",  answerCheck: "true", userID);
        }
        else { // answer is incorrect
            message.setText("It Seems You Answer is Incorrect. Please Try Again");
            db.insertReflexiveData(set.toString(), listSort.toString(),  userAnswer: "true",  answerCheck: "false", userID);

        }
    }
});

falseClick.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (rel.isReflexive(listSort) == false) { // answer is correct
            message.setText("Well Done! Your Answer is Correct");
            db.insertReflexiveData(set.toString(), listSort.toString(),  userAnswer: "false",  answerCheck: "false", userID);
        }
        else { // answer is incorrect
            message.setText("It Seems You Answer is Incorrect. Please Try Again");
            db.insertReflexiveData(set.toString(), listSort.toString(),  userAnswer: "false",  answerCheck: "true", userID);

        }
    }
});
```

Figure 6.44: OnClickListener for True & False Buttons in reflexiveQuestionActivity Class with Database Integration

```
trueClick.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (rel.isSymmetric(listSort)) {
            editText1.setEnabled(true);
            editText2.setEnabled(true);
            confirmButton.setEnabled(true);
        }
        else {
            message.setText("It Seems You Answer is Incorrect. Please Try Again");
            db.insertSymmetricData(set.toString(), listSort.toString(), userAnswer: "true", answerCheck: "false", userID);
        }
    }
});

falseClick.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        editText1.setEnabled(false);
        editText2.setEnabled(false);
        confirmButton.setEnabled(false);
        if (listSort.isEmpty() || rel.isSymmetric(listSort) == false) {
            message.setText("Well Done! Your Answer is Correct");
            db.insertSymmetricData(set.toString(), listSort.toString(), userAnswer: "false", answerCheck: "false", userID);
        }
        else if (rel.isSymmetric(listSort) == true) {
            message.setText("It Seems You Answer is Incorrect. Please Try Again");
            db.insertSymmetricData(set.toString(), listSort.toString(), userAnswer: "false", answerCheck: "true", userID);
        }
    }
});
```

```
confirmButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Integer v1 = Integer.parseInt(editText1.getText().toString());
        Integer v2 = Integer.parseInt(editText2.getText().toString());

        ArrayList<ArrayList<Integer>> temp = new ArrayList<>();
        temp.add(new ArrayList<>());
        temp.add(new ArrayList<>());
        temp.get(0).add(v1);
        temp.get(0).add(v2);
        temp.get(1).add(v2);
        temp.get(1).add(v1);

        if (listSort.contains(temp.get(0)) && listSort.contains(temp.get(1))) {
            message.setText("Well Done! Your Answer is Correct");
            db.insertSymmetricData(set.toString(), listSort.toString(), userAnswer: "true", answerCheck: "true", userID);
        }
        else {
            message.setText("It Seems You Answer is Incorrect. Please Try Again");
            db.insertSymmetricData(set.toString(), listSort.toString(), userAnswer: "true", answerCheck: "false", userID);
        }
    }
});
```

Figure 6.45: OnClickListener for True, False & Confirm Buttons in symmetricQuestionActivity Class with Database Integration

```
confirmButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Integer v1 = Integer.parseInt(x.getText().toString());
        Integer v2 = Integer.parseInt(y.getText().toString());
        Integer v3 = Integer.parseInt(z.getText().toString());

        if (v1 == null || v2 == null || v3 == null) {
            showMessage("Please enter the values");
        }
        else {
            ArrayList<ArrayList<Integer>> temp = new ArrayList<>();
            temp.add(new ArrayList<>());
            temp.add(new ArrayList<>());
            temp.add(new ArrayList<>());

            temp.get(0).add(v1);
            temp.get(0).add(v2);
            temp.get(1).add(v2);
            temp.get(1).add(v3);
            temp.get(2).add(v1);
            temp.get(2).add(v3);

            if (rel.isTransitive(listSort, temp)) {
                message.setText("Well Done! Your Answer is Correct");
                db.insertTransitiveData(set.toString(), listSort.toString(), temp.toString(), answerCheck: "true", userID);
            }
            else {
                message.setText("It Seems You Answer is Incorrect. Please Try Again");
                db.insertTransitiveData(set.toString(), listSort.toString(), temp.toString(), answerCheck: "false", userID);
            }
```

Figure 6.46: OnClickListener for Confirm Button in transitiveQuestionActivity Class with Database Integration

6.2.2 Testing

Test Description	Result
Login whilst leaving text box empty, should give error message	Success
Login with a correct username but wrong password, should giver error message	Success
Login with incorrect username, should give error message	Success
Register whilst leaving text box empty, should give error message	Success
Register with an existing username, should give error message	Success
Login with correct username and password, direct user to home page	Success
Register with username that does not exist and password, should give a message	Success
Retrieve user id from database and pass it to the next activity	Success
List created when using createList function	Success
Relations displayed to user in the question pages	Success
New question given when the Next Question button is pressed	Success
Logout	Success
Reflexive Question Page: When true button is clicked, either a message shows that the answer was right or wrong	Success
Reflexive Question Page: When false button is clicked, either a message shows that the answer was right or wrong	Success
Symmetric Question Page: When inputting correct values, and clicking confirm, a message should show saying the user got it correct	Success
Symmetric Question Page: When inputting the same values in each, and clicking confirm, a message should show saying the user got it incorrect	Failure
Symmetric Question Page: When inputting different incorrect values, and clicking confirm, a message should show saying the user got it incorrect	Success
Symmetric Question Page: When inputting a correct value and incorrect value, and clicking confirm, a message should show saying the user got it incorrect	Success

Symmetric Question Page: When true button is clicked, either a message shows that the answer was right or wrong	Success
Symmetric Question Page: When false button is clicked, either a message shows that the answer was right or wrong	Success
Symmetric Question Page: When the user submits an answer with empty EditText boxes, an error message should display	Failure
Transitive Question Page: When user input values are inputted in the correct boxes and the values are correct, a message should show saying the user got it correct	Success
Transitive Question Page: When user input values are inputted in the correct boxes and the values are incorrect, a message should show saying the user got it correct	Success
Transitive Question Page: When user input values are inputted in the incorrect boxes and the values are correct, a message should show saying the user got it incorrect	Success
Transitive Question Page: When the user submits an answer with empty EditText boxes, an error message should display	Failure

Table 6.2: Iteration 2 Test Table

6.2.3 Fix

Test Description - *Symmetric Question Page: When inputting the same values in each, and clicking confirm, a message should show saying the user got it incorrect*

The issue was fixed by adding an IF statement checking if two values from 'temp' where the same.

```
public boolean isSymmetric(ArrayList<ArrayList<Integer>> set) {
    ArrayList<ArrayList<Integer>> temp = new ArrayList<>();
    temp.add(new ArrayList<>());
    temp.add(new ArrayList<>());
    for (ArrayList<Integer> i: set) {
        if (temp.get(0).isEmpty()) {
            temp.get(0).add( index: 0, i.get(0));
            temp.get(0).add( index: 1, i.get(1));
            temp.get(1).add( index: 0, i.get(1));
            temp.get(1).add( index: 1, i.get(0));
        }
        else {
            temp.get(0).set(0, i.get(0));
            temp.get(0).set(1, i.get(1));
            temp.get(1).set(0, i.get(1));
            temp.get(1).set(1, i.get(0));
        }

        if (set.contains(temp.get(0)) && set.contains(temp.get(1))) {
            if (temp.get(0) == temp.get(1)) {
                return false; // if [x,y], x == y
            }
            else {
                return true;
            }
        }
    }
    return false;
}
```

Figure 6.47: isSymmetric Function in Relations Class – Fixed issue

Test Description - *Symmetric Question Page: When the user submits an answer with empty EditText boxes, an error message should display*

The issue was fixed by adding an IF statement checking if the two stored values were empty, which mean the text boxes were not used before submission

```
confirmButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String v1Str = editText1.getText().toString();
        String v2Str = editText2.getText().toString();

        if (v1Str.isEmpty() || v2Str.isEmpty()) {
            showMessage("Please enter your answer");
        }
        else {
            Integer v1 = Integer.parseInt(v1Str);
            Integer v2 = Integer.parseInt(v2Str);

            ArrayList<ArrayList<Integer>> temp = new ArrayList<>();
            temp.add(new ArrayList<>());
            temp.add(new ArrayList<>());
```

Figure 6.48: OnClickListener for Confirm Button in symmetricQuestionActivity Class

Test Description - *Transitive Question Page: When the user submits an answer with empty EditText boxes, an error message should display*

The issue was fixed by checking if the get the values and putting them into string variable and then checking if they were empty.

```
confirmButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String xStr = x.getText().toString();
        String yStr = y.getText().toString();
        String zStr = z.getText().toString();

        if (xStr.isEmpty() || yStr.isEmpty() || zStr.isEmpty()) {
            showMessage("Please enter the values");
        }
        else {
            Integer v1 = Integer.parseInt(xStr);
            Integer v2 = Integer.parseInt(yStr);
            Integer v3 = Integer.parseInt(zStr);

            ArrayList<ArrayList<Integer>> temp = new ArrayList<>();
            temp.add(new ArrayList<>());
            temp.add(new ArrayList<>());
            temp.add(new ArrayList<>());

            temp.get(0).add(v1);
            temp.get(0).add(v2);
            temp.get(1).add(v2);
            temp.get(1).add(v3);
            temp.get(2).add(v1);
```

Figure 6.49: OnClickListener for Confirm Button in transitiveQuestionActivity Class

6.2.4 Review

This iteration implemented the relation class which was used for the question pages for symmetric, reflexive, and transitive relation questions pages. The database and login page were also implemented allowing for users to store their data. After doing the test, failure that were observed were easily fixed by adding in IF statement for comparisons.

6.3 Iteration 3

6.3.1 Implementation

6.3.1.1 Results Page Navigation

The Results page navigation is pretty simple as it is similar to how the home page function with simple button that would use OnClickListeners and Intents to direct the user the respective pages.

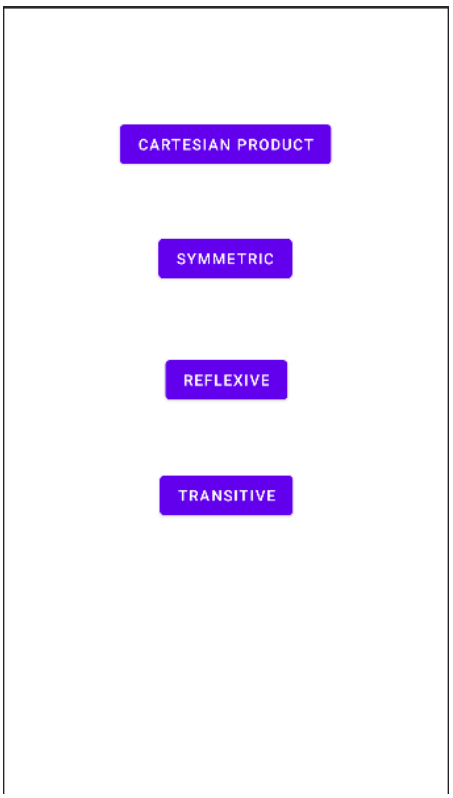


Figure 6.50: Choose Result Layout Page

```
Button cart = (Button) findViewById(R.id.cartesianResultButton);
Button trans = (Button) findViewById(R.id.transitiveResultButton);
Button sym = (Button) findViewById(R.id.symmetricResultButton);
Button ref = (Button) findViewById(R.id.reflexiveResultButton);

userID = getIntent().getIntExtra(loginActivity.USERID,  defaultValue: 0);

cart.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent buttonClick = new Intent( packageContext: chooseResultsActivity.this, cartesianResultsActivity.class);
        buttonClick.putExtra(loginActivity.USERID, userID);
        startActivity(buttonClick);
    }
});

ref.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent buttonClick = new Intent( packageContext: chooseResultsActivity.this, reflexiveResultsActivity.class);
        buttonClick.putExtra(loginActivity.USERID, userID);
        startActivity(buttonClick);
    }
});
```

```
sym.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent buttonClick = new Intent( packageContext: chooseResultsActivity.this, symmetricResultsActivity.class);
        buttonClick.putExtra(loginActivity.USERID, userID);
        startActivity(buttonClick);
    }
});

trans.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent buttonClick = new Intent( packageContext: chooseResultsActivity.this, transitiveResultsActivity.class);
        buttonClick.putExtra(loginActivity.USERID, userID);
        startActivity(buttonClick);
    }
});
```

Figure 6.51: OnClickListeners for Button on Choose Results Page Layout

Initially It was planned to use tables for how the results would be viewed by the user however, it was found that RecyclerView were easy to use for the user. Adapter classes were used for each view results page; cartesian product, reflexive, symmetric, transitive view results pages.

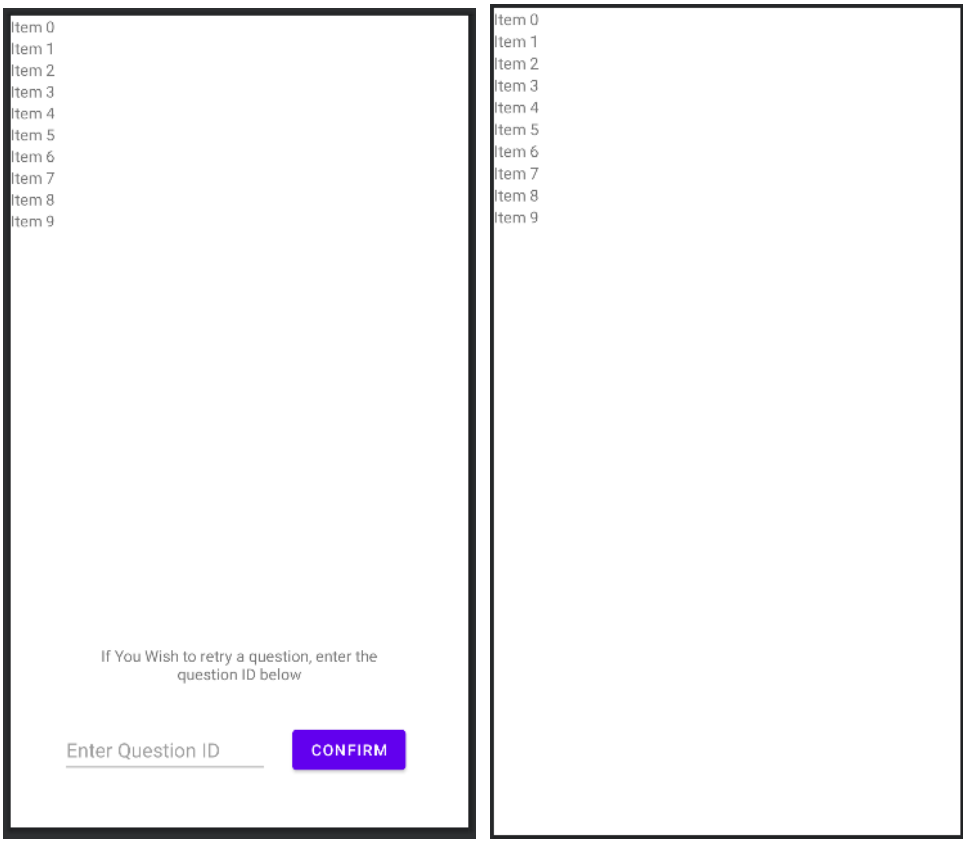


Figure 6.52: Results View Layout Variation 1 (Left) & Variation 2 (Right)

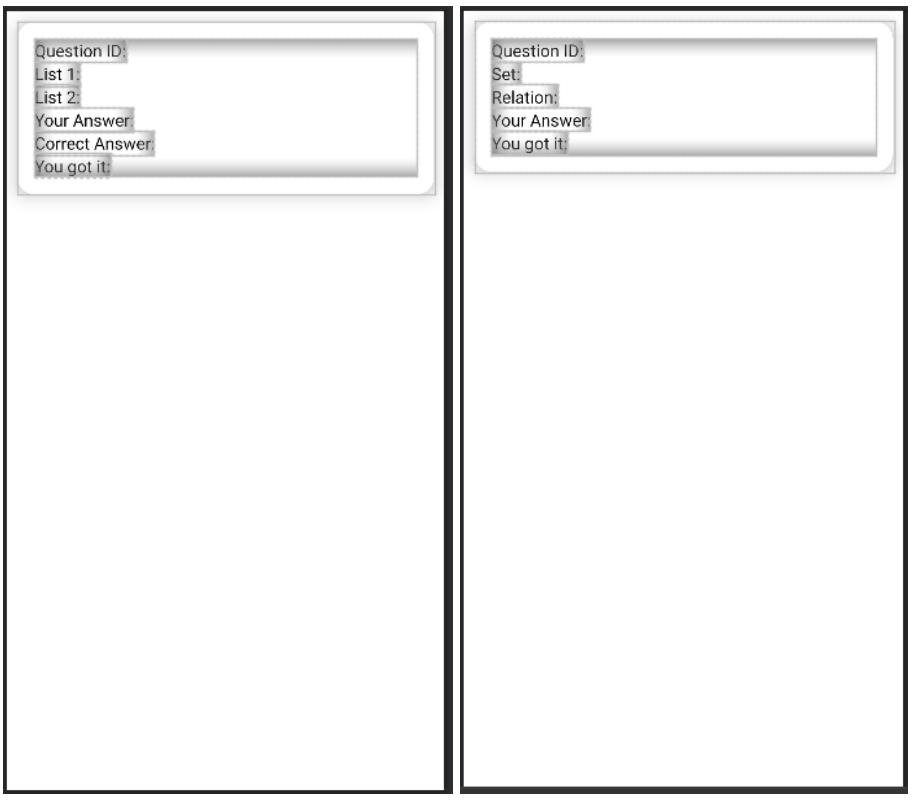


Figure 6.53: CardView Layout for RecyclerView (Left – Cartesian Product, Right – Relations)

The adapter classes contain a constructor for the class, an OnCreateViewHolder which is called by the RecyclerView to display the data, an OnBindViewHolder which set the text to the TextViews in

Figure 6.53, a getItemCount which get the size of the array list, method for initialising the XML objects.

```
public class cartesianAdapter extends RecyclerView.Adapter<cartesianAdapter.MyViewHolder> {
    private Context context;
    private ArrayList questionNum, list1, list2, userAnswer, correctAnswer, ifCorrect;

    public cartesianAdapter(Context context, ArrayList questionNum, ArrayList list1, ArrayList list2, ArrayList userA
        this.context = context;
        this.questionNum = questionNum;
        this.list1 = list1;
        this.list2 = list2;
        this.userAnswer = userAnswer;
        this.correctAnswer = correctAnswer;
        this.ifCorrect = ifCorrect;
    }

    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.cartesian_result_list, parent, attachToRoot: false);
        return new MyViewHolder(view);
    }
```

```
@Override
public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
    holder.questionNum.setText(String.valueOf(questionNum.get(position)));
    holder.list1.setText(String.valueOf(list1.get(position)));
    holder.list2.setText(String.valueOf(list2.get(position)));
    holder.userAnswer.setText(String.valueOf(userAnswer.get(position)));
    holder.correctAnswer.setText(String.valueOf(correctAnswer.get(position)));
    holder.ifCorrect.setText(String.valueOf(ifCorrect.get(position)));
}

@Override
public int getItemCount() { return list1.size(); }

public class MyViewHolder extends RecyclerView.ViewHolder {
    TextView questionNum, list1, list2, userAnswer, correctAnswer, ifCorrect;

    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
        questionNum = itemView.findViewById(R.id.questionIDinput);
        list1 = itemView.findViewById(R.id.textlist1);
        list2 = itemView.findViewById(R.id.textlist2);
        userAnswer = itemView.findViewById(R.id.textuseranswer);
        correctAnswer = itemView.findViewById(R.id.textcorrectanswer);
        ifCorrect = itemView.findViewById(R.id.textcheck);
    }
}
```

Figure 6.54: cartesianAdapter Class for the RecyclerView

```java
public class symmetricAdapter extends RecyclerView.Adapter<symmetricAdapter.MyViewHolder> {
    private Context context;
    private ArrayList questionNum, set, relation, userAnswer, ifCorrect;

    public symmetricAdapter(Context context, ArrayList questionNum, ArrayList set, ArrayList relation, ArrayList userAnswer, ArrayList ifC
        this.context = context;
        this.questionNum = questionNum;
        this.set = set;
        this.relation = relation;
        this.userAnswer = userAnswer;
        this.ifCorrect = ifCorrect;
    }

    @NonNull
    @Override
    public symmetricAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.symmetric_result_list, parent, attachToRoot: false);
        return new symmetricAdapter.MyViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull symmetricAdapter.MyViewHolder holder, int position) {
        holder.questionNum.setText(String.valueOf(questionNum.get(position)));
        holder.set.setText(String.valueOf(set.get(position)));
        holder.relation.setText(String.valueOf(relation.get(position)));
        holder.userAnswer.setText(String.valueOf(userAnswer.get(position)));
        holder.ifCorrect.setText(String.valueOf(ifCorrect.get(position)));
    }
@Override
public int getItemCount() { return set.size(); }

public class MyViewHolder extends RecyclerView.ViewHolder {
    TextView questionNum, set, relation, userAnswer, ifCorrect;

    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
        questionNum = itemView.findViewById(R.id.qIDSym);
        set = itemView.findViewById(R.id.setSymmetric);
        relation = itemView.findViewById(R.id.relationSymmetric);
        userAnswer = itemView.findViewById(R.id.useranswerSymmetric);
        ifCorrect = itemView.findViewById(R.id.checkSymmetric);
    }
}
```

Figure 6.55: symmetricAdapter Class for the RecyclerView

```java
public class reflexiveAdapter extends RecyclerView.Adapter<reflexiveAdapter.MyViewHolder> {
    private Context context;
    private ArrayList questionNum, set, relation, userAnswer, ifCorrect;

    public reflexiveAdapter(Context context, ArrayList questionNum, ArrayList set, ArrayList relation, ArrayList userAnswer, ArrayList ifCorr
        this.context = context;
        this.questionNum = questionNum;
        this.set = set;
        this.relation = relation;
        this.userAnswer = userAnswer;
        this.ifCorrect = ifCorrect;
    }

    @NonNull
    @Override
    public reflexiveAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.reflexive_result_list, parent, attachToRoot: false);
        return new reflexiveAdapter.MyViewHolder(view);
    }
```

```java
@Override
public void onBindViewHolder(@NonNull reflexiveAdapter.MyViewHolder holder, int position) {
    holder.questionNum.setText(String.valueOf(questionNum.get(position)));
    holder.set.setText(String.valueOf(set.get(position)));
    holder.relation.setText(String.valueOf(relation.get(position)));
    holder.userAnswer.setText(String.valueOf(userAnswer.get(position)));
    holder.ifCorrect.setText(String.valueOf(ifCorrect.get(position)));
}

@Override
public int getItemCount() { return set.size(); }

public class MyViewHolder extends RecyclerView.ViewHolder {
    TextView questionNum, set, relation, userAnswer, ifCorrect;

    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
        questionNum = itemView.findViewById(R.id.questionIDinputReflexive);
        set = itemView.findViewById(R.id.setReflexive);
        relation = itemView.findViewById(R.id.relationReflexive);
        userAnswer = itemView.findViewById(R.id.useranswerReflexive);
        ifCorrect = itemView.findViewById(R.id.checkReflexive);
    }
}
```

Figure 6.56: reflexiveAdapter Class for the RecyclerView

```java
public class transitiveAdapter extends RecyclerView.Adapter<transitiveAdapter.MyViewHolder> {
    private Context context;
    private ArrayList questionNum, set, relation, userAnswer, ifCorrect;

    public transitiveAdapter(Context context, ArrayList questionNum, ArrayList set, ArrayList relation, ArrayList userAnswer, ArrayList if
        this.context = context;
        this.questionNum = questionNum;
        this.set = set;
        this.relation = relation;
        this.userAnswer = userAnswer;
        this.ifCorrect = ifCorrect;
    }

    @NonNull
    @Override
    public transitiveAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.transitive_result_list, parent, attachToRoot: false);
        return new transitiveAdapter.MyViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull transitiveAdapter.MyViewHolder holder, int position) {
        holder.questionNum.setText(String.valueOf(questionNum.get(position)));
        holder.set.setText(String.valueOf(set.get(position)));
        holder.relation.setText(String.valueOf(relation.get(position)));
        holder.userAnswer.setText(String.valueOf(userAnswer.get(position)));
        holder.ifCorrect.setText(String.valueOf(ifCorrect.get(position)));
    }
```

```java
@Override
public int getItemCount() { return set.size(); }

public class MyViewHolder extends RecyclerView.ViewHolder {
    TextView questionNum, set, relation, userAnswer, ifCorrect;

    public MyViewHolder(@NonNull View itemView) {
        super(itemView);
        questionNum = itemView.findViewById(R.id.questionIDinputTransitive);
        set = itemView.findViewById(R.id.setTransitive);
        relation = itemView.findViewById(R.id.relationTransitive);
        userAnswer = itemView.findViewById(R.id.useranswerTransitive);
        ifCorrect = itemView.findViewById(R.id.checkTransitive);
    }
}
```

Figure 6.57: transitiveAdapter Class for the RecyclerView

6.3.1.2 View Results Pages

Within each of the view result activities, there exists a constructor for the RecyclerView.

```java
recyclerView = findViewById(R.id.recyclerview);
cartesianAdapter = new cartesianAdapter( context: this, questionNum, list1, list2, userAnswer, correctAnswer, ifCorrect);
recyclerView.setAdapter(cartesianAdapter);
recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
```

```java
recyclerView = findViewById(R.id.recyclerviewRef);
reflexiveAdapter = new reflexiveAdapter( context: this, this.questionID, set, relation, userAnswer, ifCorrect);
recyclerView.setAdapter(reflexiveAdapter);
recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
```

```
recyclerView = findViewById(R.id.recyclerviewSym);
symmetricAdapter = new symmetricAdapter( context: this, this.questionID, set, relation, userAnswer, ifCorrect);
recyclerView.setAdapter(symmetricAdapter);
recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
```

```
recyclerView = findViewById(R.id.recyclerviewTran);
transitiveAdpater = new transitiveAdapter( context: this, this.questionID, set, relation, userAnswer, ifCorrect);
recyclerView.setAdapter(transitiveAdpater);
recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
```

Figure 6.58: Code used for adaptars in the respective view result classes

Each of the classes also contain a displayData function used to pull data from the database and display it to the user.

```
// used to display data in the recyclerView
private void displaydata(Integer input) {
    Cursor cursor = DB.getCartesiandata(input);
    if (cursor.getCount() == 0) {
        Toast.makeText( context: cartesianResultsActivity.this,  text: "No Entry Exists", Toast.LENGTH_SHORT).show();
    }
    else {
        while(cursor.moveToNext()) {
            questionNum.add(cursor.getString( i: 0));
            list1.add(cursor.getString( i: 1));
            list2.add(cursor.getString( i: 2));
            userAnswer.add(cursor.getString( i: 4));
            correctAnswer.add(cursor.getString( i: 3));
            if (cursor.getString( i: 5).equals("True")) {
                ifCorrect.add("Right");
            }
            else {
                ifCorrect.add("Wrong");
            }
        }
    }
}
```

```
private void displaydata(Integer input) {
    Cursor cursor = DB.getReflexivedata(input);
    if (cursor.getCount() == 0) {
        Toast.makeText( context: reflexiveResultsActivity.this,  text: "No Entry Exists", Toast.LENGTH_SHORT).show();
    }
    else {
        while(cursor.moveToNext()) {
            questionID.add(cursor.getString( i: 0));
            set.add(cursor.getString( i: 1));
            relation.add(cursor.getString( i: 2));
            userAnswer.add(cursor.getString( i: 3));
            if ((cursor.getString( i: 3).equals("true") && cursor.getString( i: 4).equals("true")) ||
                    (cursor.getString( i: 3).equals("false") && cursor.getString( i: 4).equals("false"))) {
                ifCorrect.add("Right");
            }
            else {
                ifCorrect.add("Wrong");
            }
        }
    }
}
```

```java
private void displaydata(Integer input) {
    Cursor cursor = DB.getSymmetricData(input);
    if (cursor.getCount() == 0) {
        Toast.makeText( context: symmetricResultsActivity.this,  text: "No Entry Exists", Toast.LENGTH_SHORT).show();
    }
    else {
        while(cursor.moveToNext()) {
            questionID.add(cursor.getString( i: 0));
            set.add(cursor.getString( i: 1));
            relation.add(cursor.getString( i: 2));
            userAnswer.add(cursor.getString( i: 3));
            if ((cursor.getString( i: 3).equals("true") && cursor.getString( i: 4).equals("true")) ||
                    (cursor.getString( i: 3).equals("false") && cursor.getString( i: 4).equals("false"))) {
                ifCorrect.add("Right");
            }
            else {
                ifCorrect.add("Wrong");
            }
        }
    }
}
```

```java
private void displaydata(Integer input) {
    Cursor cursor = DB.getTransitiveData(input);
    if (cursor.getCount() == 0) {
        Toast.makeText( context: transitiveResultsActivity.this,  text: "No Entry Exists", Toast.LENGTH_SHORT).show();
    }
    else {
        while(cursor.moveToNext()) {
            questionID.add(cursor.getString( i: 0));
            set.add(cursor.getString( i: 1));
            relation.add(cursor.getString( i: 2));
            userAnswer.add(cursor.getString( i: 3));
            if (cursor.getString( i: 4).equals("true")) {
                ifCorrect.add("Right");
            }
            else {
                ifCorrect.add("Wrong");
            }
        }
    }
}
```

Figure 6.69: displaydata function in their respective view result classes

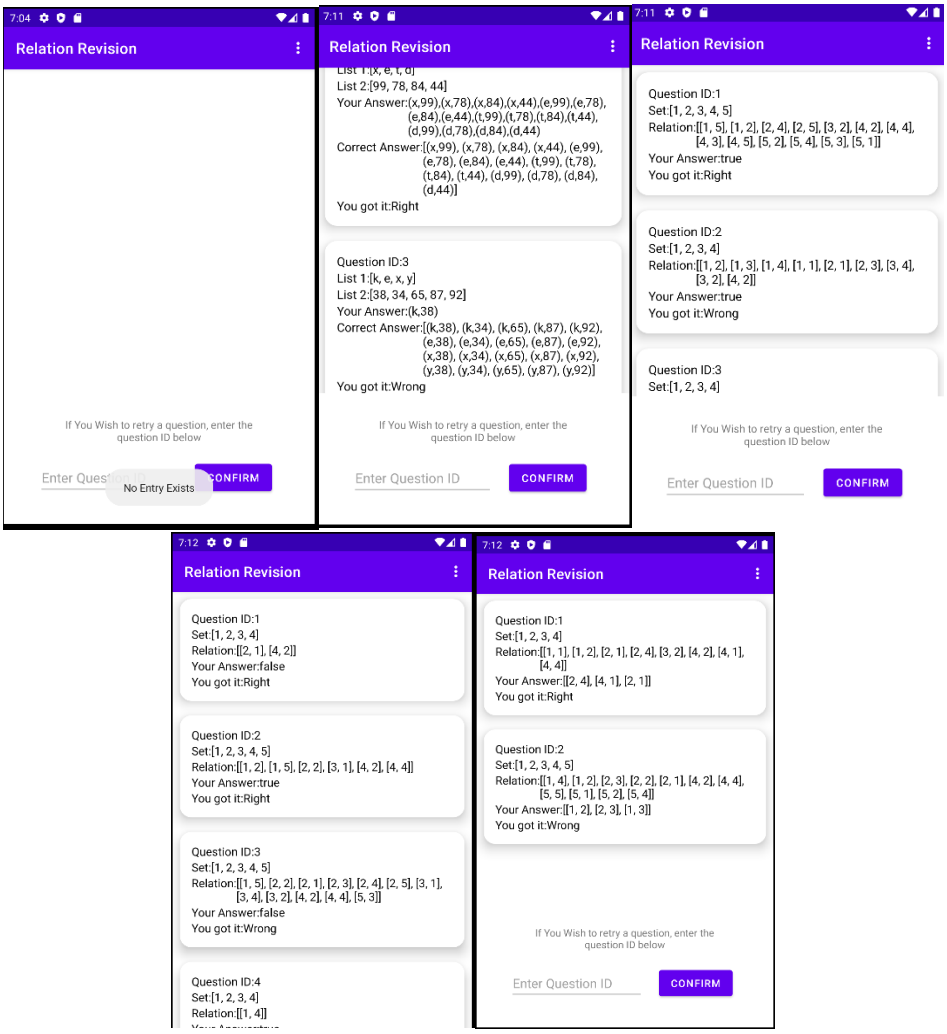


Figure 6.70: Top – Left to Right (Empty Results Page, Cartesian Product Result Page, Symmetric Results Page) Bottom – Left to Right (Reflexive Results Page, Transitive Results Page)

6.3.1.3 Retry Question Activities

When the user wishes to retry a previously failed question, they can find the question id in the results page and input at the bottom, as seen in Figure 5.70.

When the confirm button is clicked then the question id will be checked against the database to see if the question exists or not. If it does, then the user will be directed to the retry page for the specified question.

```
retry.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Integer questionNum = Integer.parseInt(questionID.getText().toString());
        if (DB.checkCartQuestion(questionNum, userID) == true) {
            Intent intent = new Intent( packageContext: cartesianResultsActivity.this, cartesianRetryActivity.class);
            intent.putExtra( name: "question", questionNum);
            intent.putExtra(loginActivity.USERID, userID);
            startActivity(intent);
        }
        else {
            showMessage("The question does not exist. Please choose and existing question.");
        }

    }
});
```

Figure 6.71: OnClickListener for Retry Button in cartesianResultsActivity Class

```
retry.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Integer questionNum = Integer.parseInt(questionID.getText().toString());
        if (DB.checkSymQuestion(questionNum, userID) == true) {
            Intent intent = new Intent( packageContext: symmetricResultsActivity.this, symmetricRetryActivity.class);
            intent.putExtra( name: "question", questionNum);
            intent.putExtra(loginActivity.USERID, userID);
            startActivity(intent);
        }
        else {
            showMessage("The question does not exist. Please choose and existing question.");
        }
    }
});
```

Figure 6.71: OnClickListener for Retry Button in symmetricResultsActivity Class

```
retry.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Integer questionNum = Integer.parseInt(questionID.getText().toString());
        if (DB.checkTranQuestion(questionNum, userID) == true) {
            Intent intent = new Intent( packageContext: transitiveResultsActivity.this, transitiveRetryActivity.class);
            intent.putExtra( name: "question", questionNum);
            intent.putExtra(loginActivity.USERID, userID);
            startActivity(intent);
        }
        else {
            showMessage("The question does not exist. Please choose and existing question.");
        }
    }
});
```

Figure 6.71: OnClickListener for Retry Button in transitiveResultsActivity Class

The question retry layouts look pretty similar to the layouts from 6.1.1.1 with the only difference being in how the user inputs the data and instead of a next button, there is a back button.

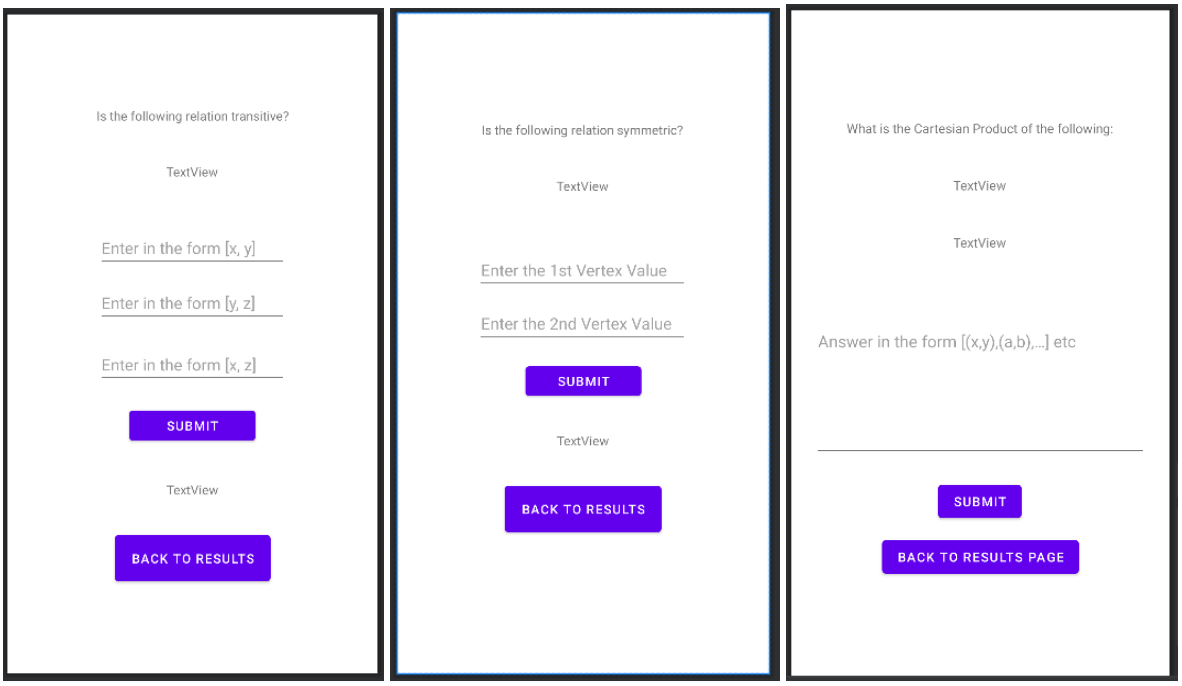


Figure 6.72: Retry Question Pages, Left to Right (Transitive, Symmetric, Cartesian Product)

The question is displayed when the user has submitted the id, as previously stated. That id is passed to the question retry activity classes through the of intent and then used the class from DBHelper to identify the question, which is then displayed to the user.

```
userID = getIntent().getIntExtra(loginActivity.USERID, defaultValue: 0);
Integer questionNo = getIntent().getIntExtra( name: "question", defaultValue: 0);

Cursor cursor = db.getSpecificTransdata(questionNo, userID);
while (cursor.moveToNext()) {
    set = cursor.getString( i: 1);
    relation = cursor.getString( i: 2);
}
setView.setText(relation);
```

Figure 6.73: Code used to pull transitive relation question data from database which is used in setText

```
userID = getIntent().getIntExtra(loginActivity.USERID, defaultValue: 0);;
Integer questionNo = getIntent().getIntExtra( name: "question", defaultValue: 0);

Cursor cursor = db.getSpecificSymdata(questionNo, userID);
while (cursor.moveToNext()) {
    set = cursor.getString( i: 1);
    relation = cursor.getString( i: 2);
}
setView.setText(relation);
```

Figure 6.73: Code used to pull symmetric relation question data from database which is used in setText

```
Integer userID = getIntent().getIntExtra(loginActivity.USERID, defaultValue: 0);;
Integer questionNo = getIntent().getIntExtra( name: "question",  defaultValue: 0);

Cursor cursor = db.getSpecificCartdata(questionNo, userID);
while (cursor.moveToNext()) {
    cL = cursor.getString( i: 1);
    iL = cursor.getString( i: 2);
    product = cursor.getString( i: 3);
}
list1View.setText(cL);
list2View.setText(iL);
```

Figure 6.73: Code used to pull cartesian product question data from database which is used in setText

6.3.2 Testing

Test Description	Result
Navigation: Choose Results Page to Cartesian Product Results Page	Success
Navigation: Choose Results Page to Symmetric Results Page	Success
Navigation: Choose Results Page to Reflexive Results Page	Success
Navigation: Choose Results Page to Transitive Results Page	Success
Navigation: Cartesian Product Results Page to Cartesian Product Retry Page	Success
Navigation: Cartesian Product Retry Page to Cartesian Product Results Page	Success
Navigation: Symmetric Results Page to Symmetric Retry Page	Success
Navigation: Symmetric Retry Page to Symmetric Results Page	Success
Navigation: Transitive Results Page to Transitive Retry Page	Success
Navigation: Transitive Retry Page to Transitive Results Page	Success
Cartesian Product Results Page: If there are no records in the database, a pop shows saying there are no entries	Success
Cartesian Product Results Page: Input question id for question is incorrect, direct user to retry page	Success
Cartesian Product Results Page: Input a question id outside of range, error message is shown to user	Success
Cartesian Product Results Page: Input question id for question is correct, message should be displayed to user	Failure
Cartesian Product Results Page: no question id is inputted when confirm button is clicked, error message show be displayed to user	Failure
Symmetric Results Page: If there are no records in the database, a pop shows saying there are no entries	Success
Symmetric Results Page: Input a question id outside of range, error message is shown to user	Success
Symmetric Results Page: Input question id for question is incorrect, direct user to retry page	Success
Symmetric Results Page: Input question id for question is correct, message should be displayed to user	Failure
Symmetric Results Page: no question id is inputted when confirm button is clicked, error message show be displayed to user	Failure
Transitive Results Page: If there are no records in the database, a pop shows saying there are no entries	Success
Transitive Results Page: Input a question id outside of range, error message is shown to user	Success

Transitive Results Page: Input question id for question is incorrect, direct user to retry page	Success
Transitive Results Page: Input question id for question is correct, message should be displayed to user	Failure
Transitive Results Page: no question id is inputted when confirm button is clicked, error message show be displayed to user	Failure
Cartesian Product Retry Page: When inputting correct values, and clicking confirm, a message should show saying the user got it correct	Success
Cartesian Product Retry Page: empty text box as submission, error message is given to user	Failure
Symmetric Retry Page: When inputting correct values, and clicking confirm, a message should show saying the user got it correct	Success
Symmetric Retry Page: empty text box as submission, error message is given to user	Failure
Transitive Retry Page: When inputting correct values, and clicking confirm, a message should show saying the user got it correct	Success
Transitive Retry Page: empty text box as submission, error message is given to user	Failure

Table 6.3: Iteration 3 Test Table

6.3.3 Review

This iteration implemented, the results and retry activity pages where the user could navigate to the result page for each of the topic questions they have answered. They can view the question and see which ones they have gotten right or wrong. They can also choose the incorrect questions and try them again to see if they can get a better result. The failures that have been observed through the test show that it mainly related verification with the database or with user inputs.

7. Evaluation

7.1 Evaluation Against Objectives

The main aim of this project was to create a mobile application that can assist students that are struggling with the topic of math relations. The aim has been relatively met, as the user can login and access questions relating to the topics with math relations. The artefact does lack features such as a variety of question along with questions that range in difficulty. The objectives were all met, with research providing a background of student performances during the pandemic along with their performance with the increased use of mobile learning applications. Research was also done existing applications on whether they provide support student with individual topics along with how quiz app function. A methodology was also determined that was followed through. Must have requirements were also met during the implementation chapter, with testing to verifying the success of the implementation.

7.2 Evaluation Against Requirements

Functional Requirements			
ID	**Requirements**	**Priority**	**Evaluation**
1	Login Page	Must Have	The final application contains a login page where the user can login to their account
2	Registration	Must Have	The final application gives the user the ability to register a new account.
3	User Friendly UI	Must Have	The application was design and implemented with a simple UI

4	Database	Must Have	A local database was created using SQLite which can store user details and question information
5	Results Display	Must Have	A results page was implemented for each topic area
6	Question Display	Must Have	Each topic area was given a separate question page tailored for its type of question
7	User Friendly Question UI	Should Have	The question UI has been kept in line with the rest of the application, therefore allowing it to be user friendly.
8	Question Verification	Must Have	Each and every question is automatically verified when submitting the user's answer
9	Question database	Must Have	All questions that have been answered by the user where right or wrong is stored in the database
10	User Profile	Could Have	This was not implemented due to time constraints
11	Creation of new questions	Must Have	Each time the user requests a new question, it is then displayed to the user on the same page
12	Request help feature	Should Have	A simple help feature was implemented however a more complicated one could be done in the future
13	Retry question display	Must Have	The question the user has requested to retry is displayed in its own page
14	Question variety	Should Have	The question variety among the topic all stay the same.
15	Quick Access	Should Have	The application can be quickly accessed along with the content through a simple login
16	Password recovery	Should Have	Due to time constraints this was not implemented
17	Password encryption	Must Have	The password is always encrypted using MD5 hashing

Table 7.1: Evaluation Against Functional Requirements

Non-Functional Requirements			
ID	Requirements	Priority	Completed?
1	Application should only work on touch screen devices	Must Have	Completed
2	Compatible with Android devices	Should Have	Completed
3	System Performance is not affected	Should Have	Completed
4	Appropriate GUI	Must Have	Completed
5	Responsiveness	Should Have	Completed
6	Availability	Must Have	Completed

Table 7.2: Evaluation Against Non-Functional Requirements

7.3 Evaluation Against Methodology

The methodology, agile methodology adapted with inspiration from XP, chosen for this project allowed there to be an artefact working consistently throughout the project allowing it to meet the requirements set for this project. meeting base requirements through the period of development. The

agile nature of the methodology allowed for unexpected deviations and error throughout the project. Requirements were ordered based on priority and the higher priority requirements were fulfilled during the early iterations. The methodology allowed for the design stage to be present in each iteration if it was required. At the end of each iteration follows a documented table test containing success and failure of said test, and a review allowing for future reference for future iterations.

8. Conclusion

This project aimed to produce an Android application that could help and support students that were struggling in their understanding of math relations, including cartesian products, reflexive relations, symmetric relations, and transitive relations. The produced artefact achieved this aim to a suitable degree as it currently has the ability to test the user's knowledge of the math relation topics.

By following the chosen methodology, it allowed for an effective collection of requirements, designing, development, implementation, and testing. The final artefact passed and met all high priority requirements that we set.

The produced application was mostly usable with only a few bugs present relating to either user inputs or database verification. Future development of the application would add more functionality, more topic area, a larger variety of questions and difficulties.

Appendix

References

Alqahtani, M., & Mohammad, H. (2015). Mobile Applications' Impact on Student Performance and Satisfaction. *Turkish Online Journal of Educational Technology - TOJET*, *14*(4), 102–112. https://eric.ed.gov/?id=EJ1077662

Balaji, S., & Sundararajan, M. (2012). International Journal of Information Technology and Business Management WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC. *International Journal of Information Technology and Business Management*, *2*(1). https://mediaweb.saintleo.edu/Courses/COM430/M2Readings/WATEERFALLVs%20V-MODEL%20Vs%20AGILE%20A%20COMPARATIVE%20STUDY%20ON%20SDLC.pdf

Bernard, R. M., Borokhovski, E., Schmid, R. F., Tamim, R. M., & Abrami, P. C. (2014). A meta-analysis of blended learning and technology use in higher education: from the general to the applied. *Journal of Computing in Higher Education*, *26*(1), 87–122. https://doi.org/10.1007/s12528-013-9077-3

Hamdan, K., & Ben-Chaban, Y. (2013). An Interactive Mobile Learning Method to Measure Students performance. *12th World Conference on Mobile and Contextual Learning (MLearn 2013)*. https://doi.org/10.5339/qproc.2013.mlearn.26

Iglesias-Pradas, S., Hernández-García, Á., Chaparro-Peláez, J., & Luis Prieto, J. (2021). Emergency Remote Teaching and Students' Academic Performance in Higher Education during the COVID-19 Pandemic: A Case Study. *Computers in Human Behavior*, 106713. https://doi.org/10.1016/j.chb.2021.106713

Lim, K. Y. T., & Wang, J. Y. Z. (2005). Collaborative handheld gaming in education. *Educational Media International*, *42*(4), 351–359. https://doi.org/10.1080/09523980500237765

López-Pérez, M. V., Pérez-López, M. C., & Rodríguez-Ariza, L. (2011). Blended learning in higher education: Students' perceptions and their relation to outcomes. *Computers & Education*, *56*(3), 818–826. https://doi.org/10.1016/j.compedu.2010.10.023

Mccormick, M. (2012). *Waterfall vs. Agile Methodology*. http://www.mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf

Means, B., Toyama, Y., Murphy, R., & Baki, M. (2013). The Effectiveness of Online and Blended Learning: A Meta-Analysis of the Empirical Literature. *Teachers College Record: The Voice of Scholarship in Education*, *115*(3), 1–47. https://doi.org/10.1177/016146811311500307

Powell-Morse, A. (2017, November 2). *Extreme Programming: What Is It And How Do You Use It?* Airbrake Blog. https://airbrake.io/blog/sdlc/extreme-programming

Roberson, J. H., & Hagevik, R. A. (2008). Cell Phones for Education. In *ERIC* (Vol. 11). https://eric.ed.gov/?id=ED518598

Shachar, M., & Neumann, Y. (2003). Differences Between Traditional and Distance Education Academic Performances: A Meta-Analytic Approach. *The International Review of Research in Open and Distributed Learning*, *4*(2). https://doi.org/10.19173/irrodl.v4i2.153

Urtel, M. G. (2008). Assessing academic performance between traditional and distance education course formats. *Journal of Educational Technology & Society*, *11*(1), 322–330. https://www.jstor.org/stable/pdf/jeductechsoci.11.1.322.pdf?casa_token=-lP2hVTaOGsAAAAA:OWVoS_7RKriFUB0VpQ6Wjxacv12-

XynTeOHHWHTobnWpsHKWqFuMvRRiPx1384Ni6GeBYSvARg0OsLyQwrVR-vWwXTJ2Uye4h0pYZrgXsKxD-bL4pKg

Vo, H. M., Zhu, C., & Diep, N. A. (2017). The effect of blended learning on student performance at course-level in higher education: A meta-analysis. *Studies in Educational Evaluation*, *53*, 17–28. https://doi.org/10.1016/j.stueduc.2017.01.002

Zhao, Y., Lei, J., Yan, B., Lai, C., & Tan, H. S. (2005). What Makes the Difference? A Practical Analysis of Research on the Effectiveness of Distance Education. *Teachers College Record*, *107*(8), 1836–1884. https://doi.org/10.1111/j.1467-9620.2005.00544.x

https://play.google.com/store/apps/details?id=com.microsoft.math&hl=en_GB&gl=US

https://play.google.com/store/apps/details?id=com.quizlet.quizletandroid&hl=en_GB&gl=US

https://play.google.com/store/apps/details?id=org.khanacademy.android&hl=en_GB&gl=US

School of Computing
Project Initiation Document

Sonny Pritesh

Mobile Application Specialised in Discrete Mathematics for Computer Science Students

Engineering Project

v 2020-09

1. Basic details

Student name:	Sonny Pritesh
Draft project title:	Mobile Application Specialised in Discrete Mathematics for Computer Science Students
Course:	MEng Computer Science
Project supervisor:	Janka Chlebikova
Client organisation:	N/A
Client contact name:	N/A

2. Degree suitability

The project will be a mobile app for university student in specific computer scientists to aid them with their math revision. During my time in university, I have learned the necessary skill to create a mobile application and I have learned the level mathematics needed for the application.

3. Outline of the project environment and problem to be solved

After talking with student at my university, I found out that a lot of them struggled with the math topics that was taught to them especially during pandemic period when all studies were decided to be done online. Therefore, I decided that to help aid their studies I would make a mobile application.

4. Project aim and objectives

I aim to create a working mobile application allowing students to review their work and hone their math skills, specifically their discrete math skills
Objectives needed to reach my aim:

- Research for any existing application with either the same or similar purpose to mine
- Conduct a survey with student who would be interested with the application
- If needed, I would have to learn new skills, e.g. a new programming language, or relearn skill, e.g. my math knowledge

5. Project deliverables

I deliver a mobile application for an Android operating system. I will also be planning a project report which would include, results of the research for the application, mock designs for the application, implementation and testing details.

6. Project constraints

- Time constraints
- Lack of programming knowledge
- Lack of human participants for research and testing
- Lack of mathematical knowledge

7. Project approach

I will start off by researching the different applications that achieve a similar purpose to mine and will gather the necessary requirements by conducting a questionnaire for students. Once that is done, I plan creating mock designs for the application and then start programming and testing the application.

8. Literature review plan

I will be initially looking at:

- Existing applications already in the market, e.g. Khan Academy
- Similar looking papers that delve into the area of math required for the level of knowledge needed for this application, e.g.
    - Cécile Ouvrier-Buffet, Antoine Meyer, Simon Modeste. (2018). Discrete mathematics at university level. Interfacing mathematics, computer science and arithmetic [University of Agder]. INDRUM Network. https://hal.archives-ouvertes.fr/hal-01849537/document

9. Facilities and resources

I will mainly be using my personal laptop and the school computers when needed.

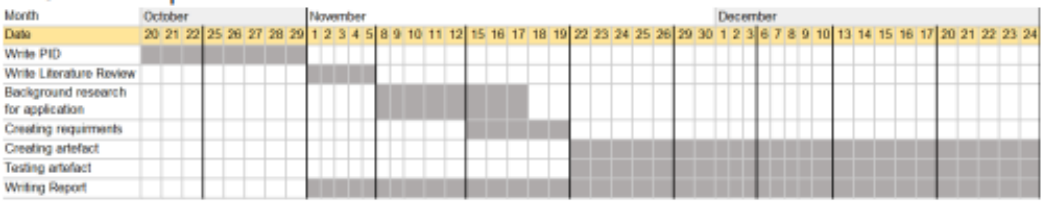
10. Log of risks

Description	Impact	Likelihood	Mitigation	First indicator
COVID-19 outbreak means I cannot get into a lab for usability testing	*Severe*	*Likely*	*Get in while I can, prioritise lab tasks in time.* *Make an alternate test plan that does*	*University informs that lab closure is likely*

			not need the lab.	
Losing my laptop	Severe	Unlikely	Make sure my work is backed up in the cloud or on a separate hard disk	
Corruption of file	Severe	Unlikely	Have multiple copies of said file ready to be used	When the file doesn't open or work as expected
Software failure	Severe	Possibly	Make sure the software is updated to the latest version	Software doesn't open. Software causes file to not open due to software constraints
Time	Severe	Possible	Make sure everything is planned out in advance	When tasks take longer than planned

11. Project plan

1. Write a literature review
2. Conduct research
3. Create artefact
4. Test the artefact
5. Improve upon artefact
6. Write report



The above Gantt chart is rough guideline for me and will be constantly updated throughout the course of the project.

12. Legal, ethical, professional, social issues (mandatory)

I will not be taking any personal data and will be using human participants that mainly go to university. The participant may also decide if they wish to a part of the survey that will be conducted and/or the testing phase of the application.

I am not planning on using a user registration system however if I do then I would make sure that all data related to it would be encrypted.

Ethics Review Certificate

Certificate of Ethics Review

Project title: Mobile Application Specialised in Mathematics for Computer Science Students

Name:	Sonny Pritesh	**User ID**:	926975	**Application date**:	27/10/2021 19:52:19	**ER Number**:	TETHIC-2021-101531

You must download your referral certificate, print a copy and keep it as a record of this review.

The FEC representative(s) for the **School of Computing** is/are **Philip Scott, Matthew Dennis**

It is your responsibility to follow the University Code of Practice on Ethical Standards and any Department/School or professional guidelines in the conduct of your study including relevant guidelines regarding health and safety of researchers including the following:

- University Policy
- Safety on Geological Fieldwork

It is also your responsibility to follow University guidance on Data Protection Policy:

- General guidance for all data protection issues
- University Data Protection Policy

Which school/department do you belong to?: **School of Computing**
What is your primary role at the University?: **Undergraduate Student**
What is the name of the member of staff who is responsible for supervising your project?: **Janka Chlebikova**
Is the study likely to involve human subjects (observation) or participants?: Yes
Will peoples` involvement be limited to just responding to questionnaires or surveys, or providing structured feedback during software prototyping?: Yes
Will the study involve National Health Service patients or staff?: No
Do human participants/subjects take part in studies without their knowledge/consent at the time, or will deception of any sort be involved? (e.g. covert observation of people, especially if in a non-public place): No
Will you collect or analyse personally identifiable information about anyone or monitor their communications or on-line activities without their explicit consent?: No
Does the study involve participants who are unable to give informed consent or in are in a dependent position (e.g. children, people with learning disabilities, unconscious patients, Portsmouth University students)?: Yes
Are drugs, placebos or other substances (e.g. food substances, vitamins) to be administered to the study participants?: No
Will blood or tissue samples be obtained from participants?: No
Is pain or more than mild discomfort likely to result from the study?: No
Could the study induce psychological stress or anxiety in participants or third parties?: No
Will the study involve prolonged or repetitive testing?: No
Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?: No
Are there risks of significant damage to physical and/or ecological environmental features?: No
Are there risks of significant damage to features of historical or cultural heritage (e.g. impacts of study techniques, taking of samples)?: No
Does the project involve animals in any way?: No
Could the research outputs potentially be harmful to third parties?: No
Could your research/artefact be adapted and be misused?: No
Does your project or project deliverable have any security implications?: No

I confirm that I have considered the implications for data collection and use, taking into consideration legal requirements (UK GDPR, Data Protection Act 2018 etc)
I confirm that I have considered the impact of this work and and taken any reasonable action to mitigate potential misuse of the project outputs

I confirm that I will act ethically and honestly throughout this project

Supervisor Review

As supervisor, I will ensure that this work will be conducted in an ethical manner in line with the University Ethics Policy.

Supervisor's signature: Date:

Faculty Ethics Committee Review

Faculty Ethics Committee Member's signature: Date: