Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name: Ian Cunningham
Student ID: x13114425
Supervisor: Dr. Paul Stynes

SECTION 2 Confirmation of Authorship
The acceptance of your work is subject to your signature on the following declaration:
I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: Ian Cunningham
Date: 10/05/2017

National College of Ireland

BSc in Computing

2016/2017

Ian Cunningham

X13114425

Ian.Cunningham@student.ncirl.ie

Scriba-College

Technical Report

National College of Ireland

Table of Contents

Table of Figures

Executive Summary

Each and every day thousands of students attend college and are the owner of an Android device. The way in which students currently interact with their college material to create notes is to either create documents of notes or write out their notes on paper. This project aims to eliminate these tiresome tasks to promote a new way of creating notes for the student's college material by interacting the college material on their Android device and annotating the content via the Scriba stylus to create notes.

The intended audience of the project are students that are the owners of an Android device and an a Scriba stylus. The idea is that students will have access to their college documents and presentation slides by having these documents and slides stored on a remote server. While in class or at home, students may create notes with their Scriba stylus.

The project idea has been proposed and the requirements have been gathered while in consultation with David Craig, CEO Dublin Design Studio LTD. The necessary requirements for the project are documented in the requirements section of this report. The technologies that would best suit the projects requirements have been identified.

This project is necessary in this field as it will improve a student's interaction with their college material by providing a useful mobile app to store unique notes and view at any time and any place. The application will also contain an option to create quizzes based on the recorded notes, which adds a unique feature not seen in other applications in the same field.
The key benefit of this application is that it will increase student engagement and knowledge of college material by providing a fun and intuitive way of recording notes. The application will offer quizzes based on questions created by the user.

1 Introduction

1.1 Background

The stylus product “Scriba” is an award winning new stylus, which is designed by Dublin Design Studio. Scriba is designed around the movements of the hand to make it as comfortable as possible to use. Supporting both left and right handed users, Scriba works with mobile and tablet devices and contains a squeeze motion technology to increase control and offer an innovative experience. Scriba’s body bends to the users every touch and movements are detected allowing the user to access different application functionality.

Students often find it hard to recall important information from class material or leave it to the last minute to gather all their notes together for studying purposes before exams or continuous assessments. This application will allow students to have their notes all stored in one location. Students will record notes during class and these notes will be saved to accompany the specific document. Therefore, when the student takes important notes in class, the student will know when it comes to exam time that these notes are important and can be easily accessed. The student can then read over their notes anywhere and anytime. The application will provide a way for users to increase engagement with the class content, therefore gaining better and more understanding of the class material. The application will also test the students’ knowledge of their college material by providing quiz’s.

It was decided to create an Android application as numerous of other android applications had been developed for the Scriba stylus by Dublin Design Studio. The application will be created using Java and the Android Studio IDE. Data will be stored in a MySQL database. The user’s files will be stored on a remote server and can be accessed via the application.

1.2 Aims

The purpose of this project is to create a functional Android application that provides students access to their college material. The aim is to increase student engagement with college material by using a Scriba stylus to annotate the information in a way that will make it easier for them to recall important sections or clarify their own understanding of the content, that will provide benefits for the student. The application will aim to serve as a study tool for exams and continuous assessments.

The application will allow the user to store and access their college material, which will be files such as word documents, PDF files and PowerPoint presentation files. The application will integrate Bluetooth connectivity so as to connect to the Scriba stylus, which connects to devices via Bluetooth Low Energy. The Scriba will have functions for interacting with the college material including functions such as highlight text, change text colour, underline text, etc. All functions will result in a note been created.

The application will also enable the student to create quizzes based on questions that the user can create themselves. The user can choose to create a quiz for an individual subject. The application will implement a AI personal tutor using Googles API.ai. The application will also provide the student the option of creating mind maps based on the notes. The student will also be able to create and store study plan’s.

1.3 Technologies

1.3.1 Java

The programming language that will be used to implement the Android application will be Java. Java will be used to implement the logic and the functionality on the client side of the application. Java is a primary language for Android development but is not the only language used for Android development.

1.3.2 XML

XML (Extensible Markup Language) is used in every Android application is a standard when developing Android applications. XML is used for creating the user interface in an Android application. The advantage of using XML is that it separates the presentation from the logic, which helps Android applications follow the Model-View-Controller pattern. XML will be used to represent the presentation of my application.

1.3.3 PHP

PHP (PHP: Hypertext Preprocessor) is a server side scripting language. PHP scripts are stored and executed on the server. PHP will be used in the application for controlling user access, sending and receiving data from the database and sending files to be stored on the server. PHP will be responsible for the server side of the application and directly communication with the database.

1.3.4 MySQL

The database chosen to communicate with the application is a MySQL database. MySQL is a relation database management system that provides good functionality with PHP. The database will data such as user's information, file information and stored notes.

1.3.5 JSON

JavaScript Object Notation (JSON) is a lightweight data-interchange format that is human readable and machine readable. JSON will be used in the application for transmitting data between the application and the server.

1.3.6 Android Studio IDE

The Android Studio Integrated Development Environment is the official Android IDE for Android development offered by Google, which is based on IntelliJ IDEA. The IDE comes with the Android SDK built in with the required libraries, debugger and emulator. Real Android devices will be used for testing the application as this is a faster process than testing on the emulator and this will also give me a feel of how the application will look and function on a real Android device. This is the IDE that will be used for the development of the application.

1.3.7 Android Bluetooth LE API

The Android BLE API provides support for Bluetooth Low Energy. This enables Android applications to communicate with BLE devices. This API will be used in the application to provide communication between the application and the Scriba stylus as this is a BLE supported device.

1.3.8 Cloud Convert API

The Cloud Convert API is a RESTful API that provides the ability to convert multiple file types to a HTML file type. A call to the API enables a user’s file to be converted to HTML file and then rendered in an Android WebView.

1.3.9 API.AI

The API.ai is a Conversational User Experience Platform that enables developers to integrate intelligent conversation interfaces within applications. This enabled the application to implement an intelligent personal assistant that will be trained to respond to computer science related questions and will be trained in other areas in the future.

1.3.10 GitHub

GitHub is a web-based version control system were developers store their projects and collaborate. GitHub is a requirement for this project and therefore, will be used for version control in this project as their will be constant updates and changes. Also in the case of anything going wrong there will be a need previous revisions of the project. GitHub will keep record off all changes to a project and store all versions of a project in single repository.

1.4 Structure

1.4.1 System

This chapter will discuss and describe the necessary requirements for the project such as functional and non-functional requirements. This section will discuss the process of each functional requirement in detail. Also documented are the data requirements, the user requirements, environmental requirements and usability requirements.

This chapter will also discuss and present the design and architecture of the application. Mock-ups of the application will be also presented within this section and details about the various methods that will be used for testing the application.

1.4.2 Conclusions

This chapter will describe and discuss in detail the advantages of the project and the disadvantages of the project. This section will also examine the opportunities and limitations of the project.

1.4.3 Further Developments or Research

This chapter will discuss any further developments of the project and how the project could expand. This section will also describe any research undertaken to support the any further developments for the project and detail what type of environments this application can expand into.

1.4.4 References

This chapter will list all references used for the necessary research for the project and all documents referenced with this document.

2 System

2.1 *Requirements*

2.1.1 Functional requirements

This section lists the functional requirements in **ranked order**.

2.1.1.1 Use Case Diagram

The Use Case Diagram provides an overview of all functional requirements.

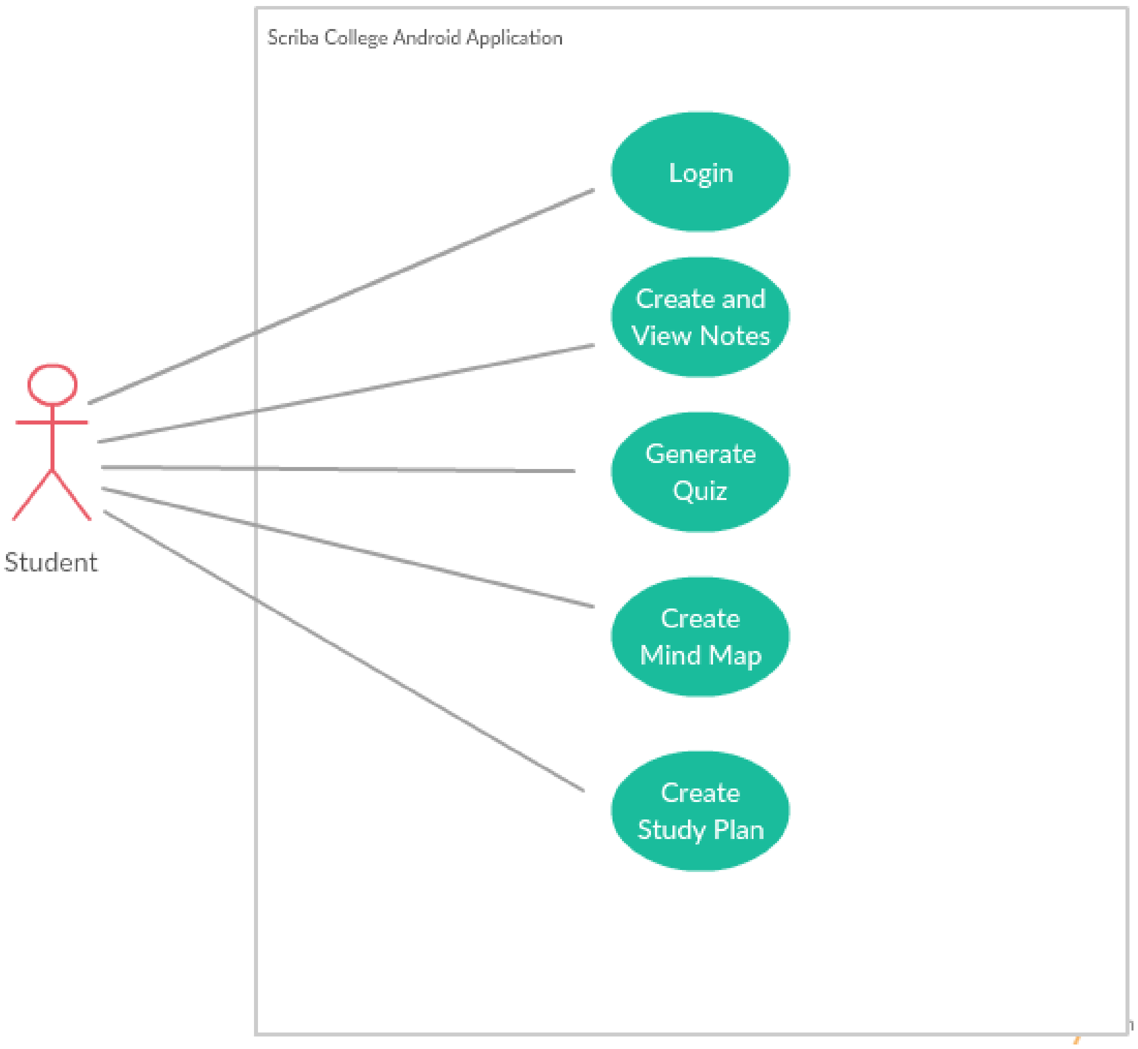


Figure 1: Scriba College Use Case Diagram

2.1.1.2 Requirement 1 <Create and View Notes>

2.1.1.2.1 Description & Priority

This use case describes the how a note is created by uploading a document and annotating the document via the Scriba stylus and then viewed by the user. This use case is the most important in the application as it is the core functionality The priority of this use case is high.

2.1.1.2.2 Use Case

ID

UC02

Scope

The scope of this use case is to create notes that can be later retrieved for viewing by user.

Description

This use case allows a user to create and view a note.

Use Case Diagram

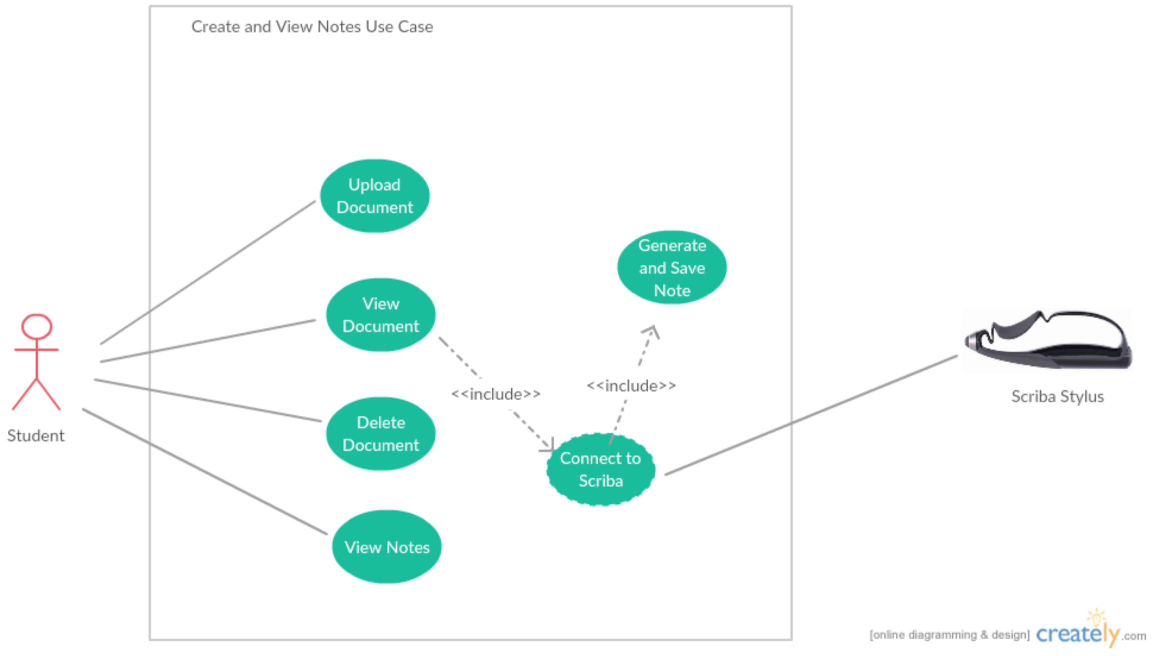


Figure 2: Create and View Notes Use Case

Flow Description

Precondition

The user must have a registered account and be signed into the application.

Activation

This use case starts when a user uploads a document to the application.

Main flow

1. The user presses the select file button and selects a file from device storage
2. The user presses the upload file button and file is sent to server
3. The system responds with a success or fail message (See E1)
4. The user selects the file to view
5. The system presents the selected file
6. The system prompts the user to connect to a Scriba stylus
7. The user connects their Scriba stylus to their device (See A1)
8. The user annotates the content in the file such as highlight, draw, erase depending which mode the Scriba stylus is presently in.
9. The system generates notes on the annotated information and stores these notes in the database.
10. The user views the generated notes.

Alternate flow

A1: <Scriba stylus not connected>

1. The user unsuccessfully connects their Scriba stylus
2. The system prompts the user to reconnect their Scriba stylus.
3. The use case continues at position 7 of the main flow

Exceptional flow

E1: <Server response fail>

4. The system responds with an error message that the file was not uploaded to the server.
5. The user selects a file to upload
6. The use case continues at position 3 of the main flow

Termination

The system presents the user with the main activity screen.

Post condition

The system stores the notes on the database if the use case was successful

2.1.1.3 Requirement 2 <Generate Quiz>

2.1.1.3.1 Description & Priority

This use case describes how a quiz is generated from the saved notes on the database. A quiz can be generated based on user-created questions for different subjects. The priority of this use case is high.

2.1.1.3.2 Use Case

ID

UC03

Scope

The scope of this use case is to create a quiz based on the users created questions.

Description

This use case describes process of generating a quiz.

Use Case Diagram

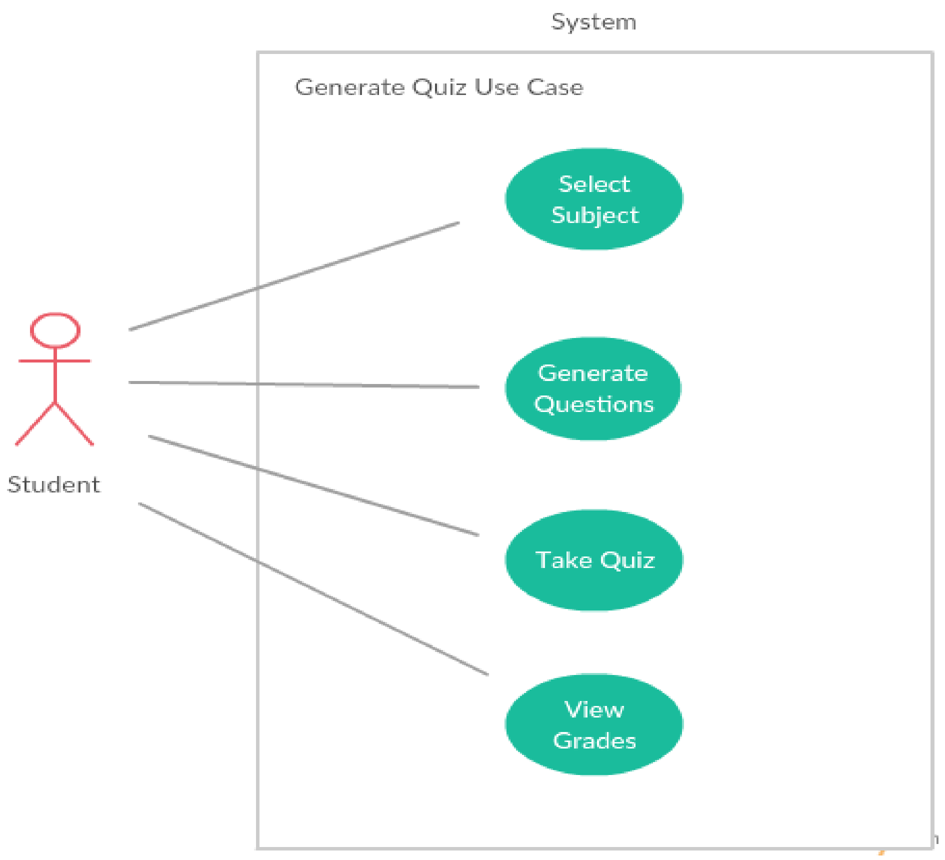


Figure 3: Generate Quiz Use Case

Flow Description

Precondition

There must be existing notes in the database.

Activation

This use case starts when a user selects a subject to generate a quiz on.

Main flow

11. The system identifies the user's selection of subject (See E1)
12. The system returns a quiz based on chosen subject
13. The user answers a serious of questions
14. The system records the users grade (See A1)
15. The user exits the quiz

Alternate flow

A1: <Grade not greater than highest grade>

7. The users grade is not greater than the current stored grade for the current quiz.
8. The grade is not recorded
9. The use case continues at position 15 of the main flow.

Exceptional flow

E1: <Invalid/No subject entered>

10. The user has not entered a valid subject or has left the subject field blank.
11. The system responds with an error message stating that a subject must be entered.
12. The use case continues at position 11 of the main flow.

Termination

The system presents the user with the main activity screen.

Post condition

The system stores the user grade on the database if the use case was successful.

2.1.1.4 Requirement 3 <Login and User Registration>

2.1.1.4.1 Description & Priority

This use case describes how the user is able to create an account for the application by providing a unique username and password that will be stored on the database. When an account has been created, the user shall be able to login into the application. The user will enter their unique credentials that will be compared to credentials stored in the database. Access to application is granted upon successful comparison of user credentials. This use case is essential and the priority of this requirement is high as a user needs to create an account and login to have access to the applications features.

2.1.1.4.2 Use Case

ID

UC01

Scope

The scope of this use case is to create a user account and allow access to the system.

Description

This use case describes the process of a user signing into the system.

Use Case Diagram

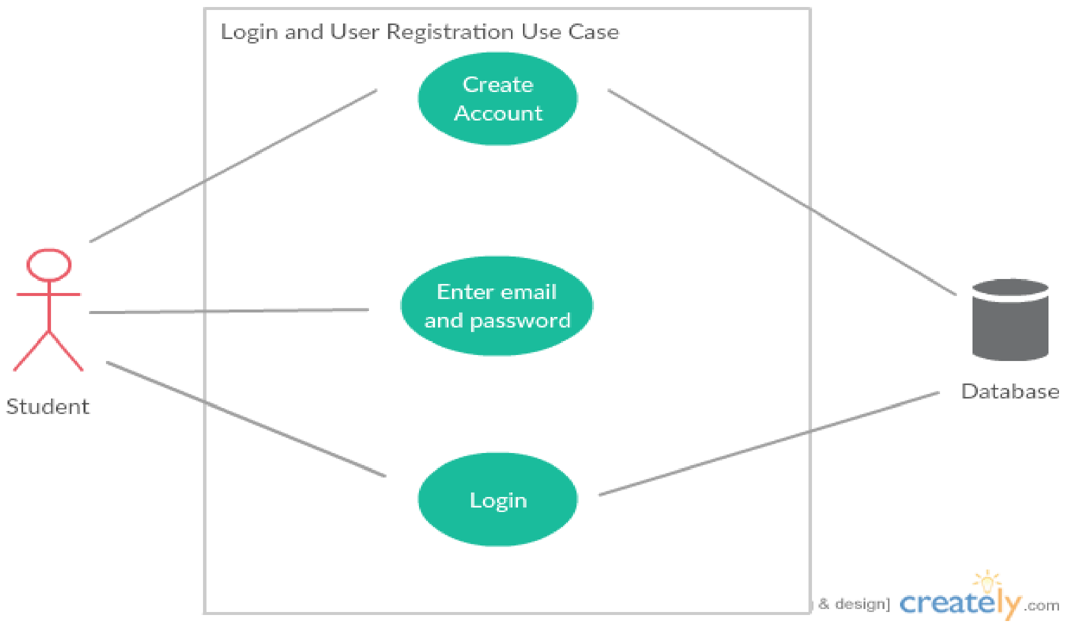


Figure 4: Login and Registration Use Case

Flow Description

Precondition

An internet connection is required to create an account and login.

Activation

This use case starts when a user clicks the login button.

Main flow

16. The system prompts the user to create an account or login
17. A new user will be prompted to enter their credentials for their account
18. The system stores the new user's credentials on the database See E1)
19. The system prompts an existing user is prompted to login
20. The user enters their credentials (See A1)
21. The system checks the users entered credentials against the database for verification.
22. The system grants access to the user and redirects the user to the main activity screen.

Alternate flow

A1: <Wrong user credentials>

13. The user enters the incorrect credentials
14. The system displays a message stating that the entered details are incorrect
15. The use case continues at position 19 of the main flow

Exceptional flow

E1: <User already exists>

16. The user enters existing credentials for an existing account.
17. The system responds with an error message stating that an account already exists with those credentials
18. The use case continues at position 16 of the main flow

Termination

The system presents the user with the main activity screen.

Post condition

The system stores the user's credentials on the database if the use case is successful.

2.1.1.5 Requirement 4 <Create Mind Map>

2.1.1.5.1 Description & Priority

This use case describes how the user can create mind maps, based on the notes stored on the database, for study purposes. A mind map can be created for different subjects, with all their notes or for single notes. This use case is of medium to low priority as this function is an alternative to studying the created notes and taking the quiz's.

2.1.1.5.2 Use Case

ID

UC04

Scope

The scope of this use case is to allow the user to create a mind map based on specific notes.

Description

This use case describes the process of creating a mind map.

Use Case Diagram

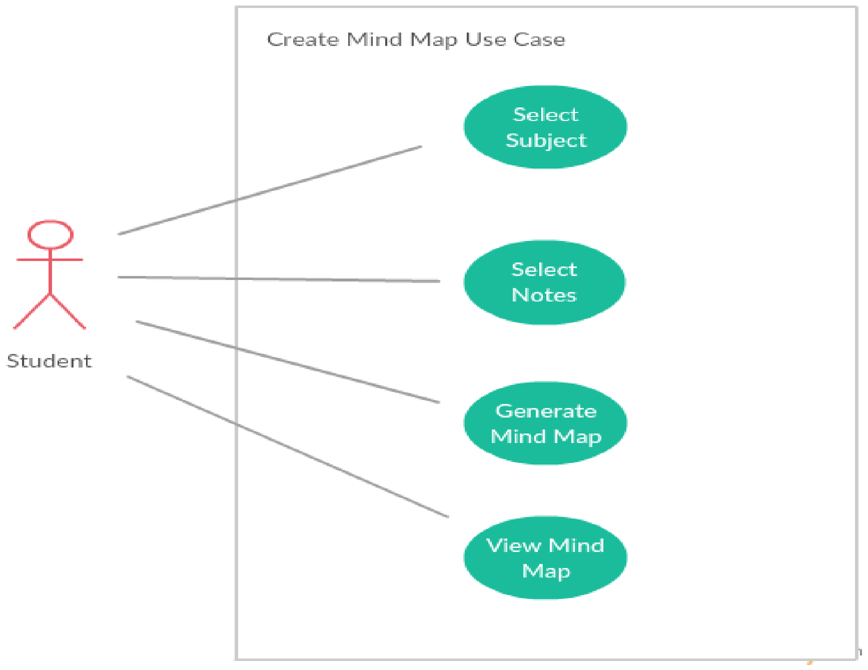


Figure 5: Create Mind Map Use Case

Flow Description

Precondition

The system must have notes stored

Activation

This use case starts when a user clicks the create mind map button.

Main flow

23. The user selects a subject and note(s) for mind map rendering (See E1).
24. The system identifies the user's selection of subject and note(s) (See A1)
25. The system renders the mind map and stores on the server

Alternate flow

A1: <Invalid/No selection of subject or note(s)>

19. The user enters invalid/selects no subject or note(s)

20. The system responds with a warning message that a subject or note(s) must be entered.
21. The use case continues at position 24 of the main flow.

Exceptional flow

E1: <No notes present>

22. The user has no existing notes in the application
23. The system displays an error message stating that no notes are present and that a mind map cannot be created.
24. The use case continues at position 23 of the main flow

Termination

The system presents the user with the Mind Map Activity with a list of created mind maps.

Post condition

The system stores the mind map on the server and stores name and path off the mind map on the database.

2.1.1.6 Requirement 5 <Create Study Plan>

2.1.1.6.1 Description & Priority

This use case describes how the user can create a study plan. A study plan can be created in a way to suit the users time management. This use case is of medium to low priority.

2.1.1.6.2 Use Case

ID

UC05

Scope

The scope of this use case is to allow a user to create a study plan by entering subjects and assigning allocated times to these subjects.

Description

This use case describes the process of creating a study plan.

Use Case Diagram

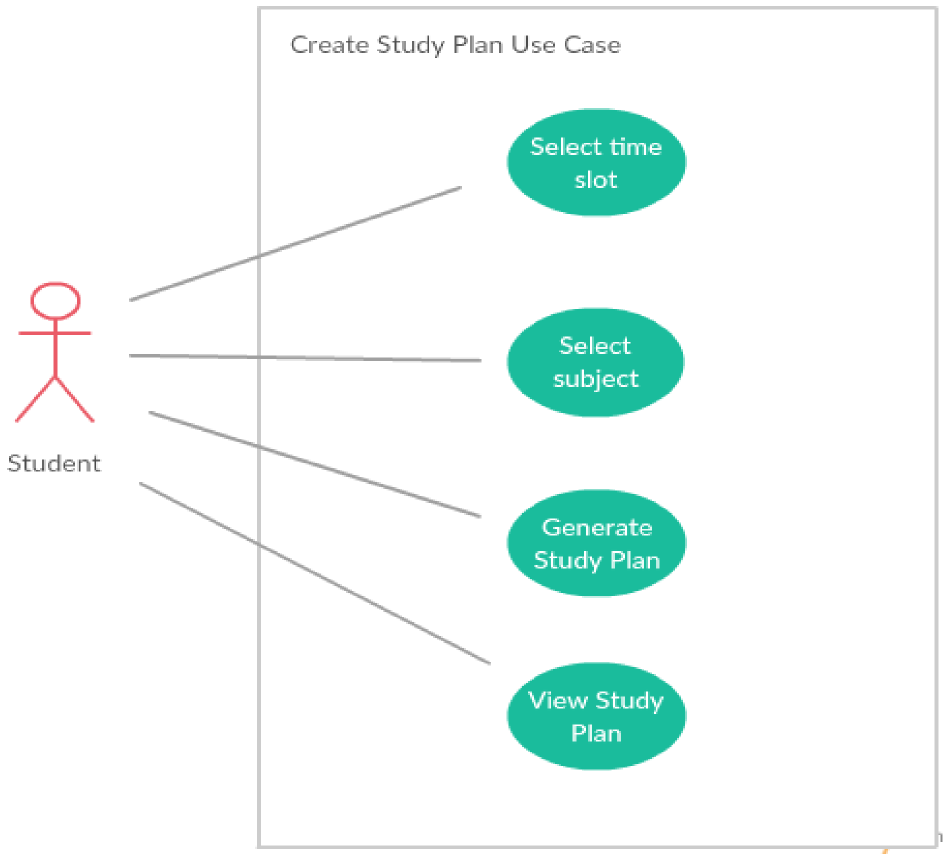


Figure 6: Create Study Plan Use Case

Flow Description

Precondition

The user must be signed into the application.

Activation

This use case starts when a user clicks the create study plan button.

Main flow

26. The system prompts the user to enter details
27. The user enters the subjects and assigns a time slot to each subject (See A1)

28. The system generates a new study plan image (See E1)
29. The user views the study plan.

Alternate flow

A1: <Invalid/No details entered>

22. The user does not enter valid details.
23. The system responds with a warning message stating that no invalid details have been entered.
24. The use case continues at position 26 of the main flow

Exceptional flow

E1: <Image not generated>

25. The system displays a warning message stating that the study plan was not created and to try again.
26. The user re-enters the details
27. The use case continues at position 28 of the main flow

Termination

The system presents the user with the Study Plan Activity providing a list of viewable study plan's.

Post condition

The system stores the study plan image on the server and stores the image name and path on the database, if the use case is successful.

2.1.2 Non-Functional requirements

2.1.2.1 Performance/Response time requirement

The application should be fast and responsive to all device screen sizes. The application shall switch between activities in minimal time. The login process should take no more than 5 seconds and interaction with the Scriba stylus should be performed in a seamless manner.

2.1.2.2 Availability requirement

The application will available at anytime and anywhere, once installed on the user's device. An internet connection is needed for connection to the database for access and retrieval of files, grades, mind maps and study plans. Unfortunately, if the user has no internet connection they will not be able to access the application and their stored content.

2.1.2.3 Recover requirement

The applications files stored on the remote server and the data saved on the hosted database shall be backed-up to multiple locations to prevent loss of data and to keep the application live in the case of a disaster. A set of policies and procedures should put in place for the recovery process.

2.1.2.4 Robustness requirement

The application should be designed in a way that will prevent the app from crashing. The application should not crash due to invalid request but rather respond with an error message to the user.

2.1.2.5 Security requirement

The application will connect to the database which is hosted on a remote server, hence security will be an important factor to consider for the application. The application will encrypt private user information such as the user's password.

2.1.2.6 Reliability requirement

In mobile applications, reliability is an important factor to consider. If the application keeps crashing, this will affect the overall reliability of the application resulting in poor reviews of the application. The system should be able to manage minor issues such as interference with the Bluetooth connection between the Scriba stylus and the application.

2.1.2.7 Maintainability requirement

The applications code should be written in a way that is easy to read and understand. This will provide the application with the ability to be easily maintained and make it easy for applying new updates. The application should be designed in a way that splits the application into separate sections i.e. separation of concerns (SoC).

2.1.2.8 Portability requirement

As the application is an Android application, it will need to be downloaded on an Android smartphone/tablet and will not be downloadable to Windows/Apple devices. The application will also need to installed on devices that support Bluetooth Low Energy (BLE) as the Scriba stylus will connect with the user’s device via BLE. The application should be portable with Android OS 5.0 Lollipop.

2.1.2.9 Extendibility requirement

The application should be designed and implemented in way that will take into consideration future growth. This will provide the application the ability to extend the system and a good design will enable the application to be easily extended with additions such as new functions.

2.1.2.10 Reusability requirement

The applications code should be written in a way that will enable reusability. The application can then be reusable in different environments. For example, the application can be reused as a workplace tool.

2.1.2.11 Resource utilization requirement

The application should utilize the Android Bluetooth API for supporting BLE connections with the Scriba stylus and utilize embedded tools for opening the different types of files within the application resulting in third party file manager applications not been required to open the files.

2.1.3 Data requirements

2.1.3.1 MySQL

A MySQL database will be used to store the user’s information, all uploaded files information and all the user’s notes. The database will be hosted on a remote server.

2.1.3.2 JSON

The data received from the server will be in JSON format will be parsed to plain text within the application to display to the user.

2.1.4 User requirements

The purpose of the Scriba College Android application is to provide students with a way to interact with their college material with a Scriba stylus to aid study and recollection of notes.

The main user requirement of the Scriba College Application is to provide an app that will increase student engagement with college material and serve as a study tool to the user.

The user is required to be the owner of an Android device in order to be able to install the application.

The user’s Android device must be running the operating system Android 4.3 Jelly Bean (API Level 18) in order to run the application.

The user is required to have internet access on their devices in order to create an account and login to the application.

The user is required to be the owner of a Scriba stylus, which will enable the user to annotate their college material.

The system shall allow the user to upload college documents in the form of word files, pdf files and power point files etc. and provides users the option to view their uploaded files.

The app shall allow the user to annotate the college material with the Scriba stylus in a manner that will make it easier for them to recall important sections or clarify their own understanding of the content by saving content that was highlighted or drawn over as note.

Each user shall be able to save notes to accompany the specific document they relate to and have an option for viewing these notes.

The app shall allow the user to create a user account with a unique username and password and the option to login in using their unique user account credentials.

The app shall be easy for the user to navigate and shall provide the user access to functionality quickly through interaction with the Scriba stylus.

Each user shall be able to select functionality within the app i.e. highlight, font color, underline, etc. by using Scriba's squeeze motion technology.

The application shall allow for the creation of quizzes based on the users created questions.

The application shall provide a chatbot in the form of a personal tutor. The chatbot shall respond in timely manner to the user's verbal questions.

The user shall be able to create mind maps based on their notes and create study plans.

2.1.5 Environmental requirements

2.1.5.1 Android Devices

Multiple Android devices will be required for testing purposes. Using multiple devices to test the application will visually show how the application will be presented on different types of Android devices. These devices will be required to be running at least operating system Android 4.3 Jelly Bean (API Level 18) as this the API level where Bluetooth low energy support for Android was first introduced.

2.1.5.2 Android Studio IDE

The Android Studio IDE will be used to create the application using the Java programming language. This IDE also includes the Android SDK built in, which includes the required libraries for developing Android applications. The IDE also provides support for debugging and testing.

2.1.5.3 Hosting Service

A hosting service will be required for storing files on the server and for hosting the database. The hosting service that will be used for the project is X10 Hosting. This will be the hosting service used for the finished product as the development process used XAMPP for a web server solution.

2.1.5.4 Scriba Stylus

A Scriba stylus will be required for testing the communication between the application and the Scriba for monitoring if the correct values are being sent from the Scriba stylus to the application. Also this will be needed mainly for testing the functionality within the app that requires a Scriba, such as the highlight, draw and erase functions.

2.1.6 Usability requirements

2.1.6.1 Efficiency

The application will be effective in accomplishing its task is reasonable time. The application functions will work as expected and in the case of an error, the user will be notified.

2.1.6.2 Intuitiveness

The applications user interface should be intuitive, easy to follow and navigate. All messages displayed to the user should be presented in an understandable manner.

2.1.6.3 Workload

The application should appear user friendly and easy to use rather than being perceived to have a hard to use and frustrating user interface.

2.1.6.4 Scriba Instructions

The application will provide the user with a dialog box of instructions to instruct the user on how to use the Scriba stylus with the application.

2.2 Design and Architecture

2.2.1 Assumptions / Constraints / Standards

2.2.1.1 ASSUMPTIONS

- The user of the application will be the owner of an Android device.
- The Android Studio IDE will be used to develop the application.
- The application will be intuitive.
- The application depends on a login system.

2.2.1.2 CONSTRAINTS

It will be necessary to design and develop the application aimed at a specific Android API. The application will be aimed at minimum Android Jelly Bean (API Level 18). This is necessary as this is the first API to introduce support for Bluetooth Low Energy (BLE) support.

It is necessary for the user to purchase a Scriba stylus device to fully interact with the Scriba College application. The user will also have to be the owner of an Android device to install the application on their devices.

2.2.1.3 STANDARDS

The application will be designed using the Android theme "Material Design". Material design is a new theme that provides new system widgets and advanced animations. The theme allows developers to customize the look and feel of their applications by enabling control over a color palette, which in turn enables the developers set their specific colors to the action bar and the status bar using specified attributes for the theme (Android Developers, 2017).

2.2.2 Architecture Design

This section outlines the system and hardware architecture design of the system that is being built.

2.2.2.1 Logical View

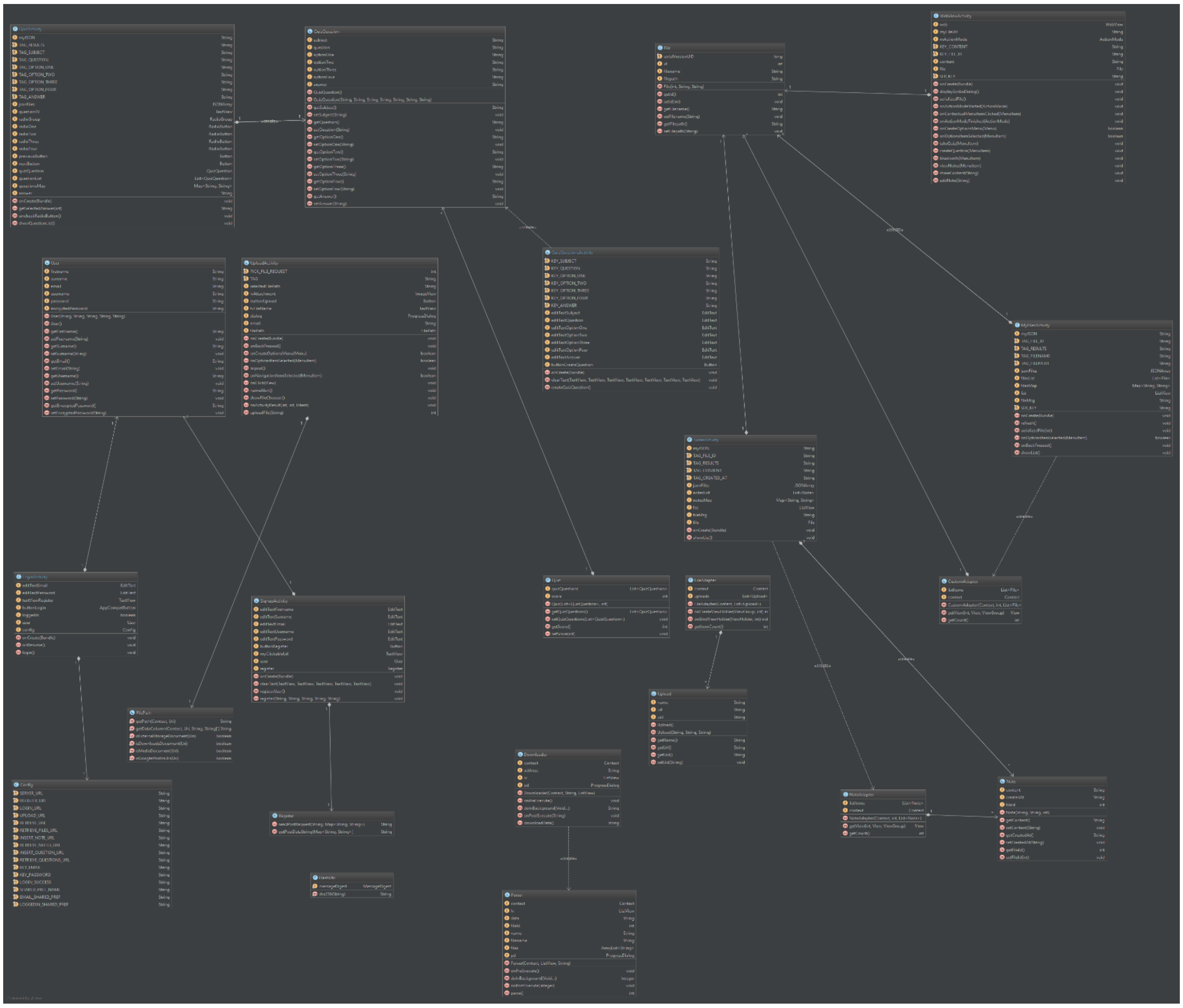


Figure 7: Scriba College class diagram

The above class diagram illustrates the classes that comprise the system. The diagram shows all the required classes necessary to construct the system and exhibits the relationships and dependencies between the classes of the system.

2.2.2.2 Hardware Architecture

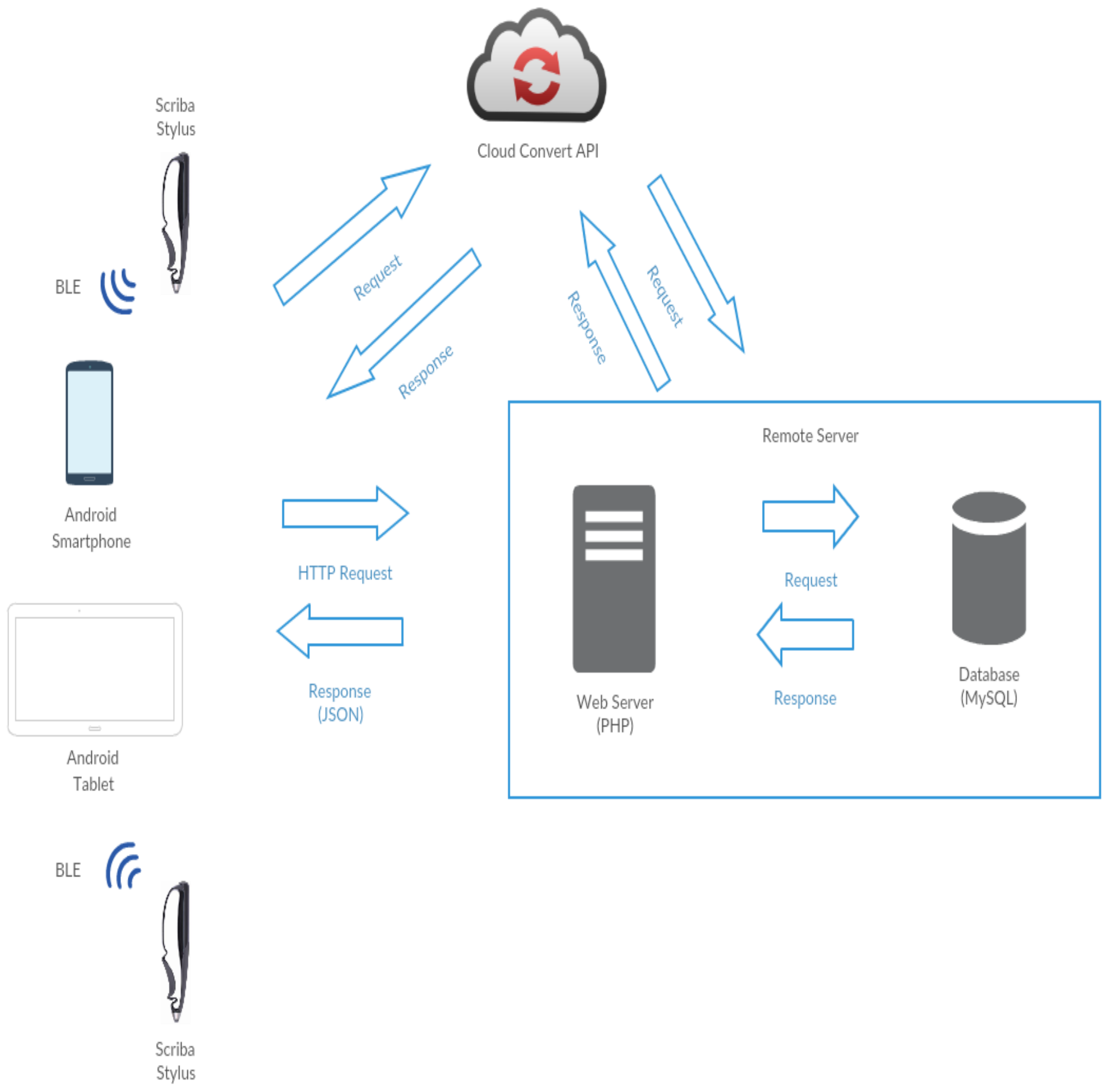


Figure 8: Scriba College hardware architecture diagram

The application will utilize a three-tier architecture between client and server. The Android device (client) will send a HTTP request to the web server. The web server will then run the PHP scripts to connect to the MySQL database and make a request for data. The database will send a response back to the web server with the requested data. The PHP scripts on the web server will convert the received data into JSON and sends this data to the Android device (client). The client receives the data and parses this data into plain text and displays on screen. When viewing a file, the client will send a request to the Cloud Convert API, which in turn will make a request to the web server for the specified file. The web server responds with the requested file and the Cloud Convert API responds with the converted file.

2.2.2.3 Software Architecture

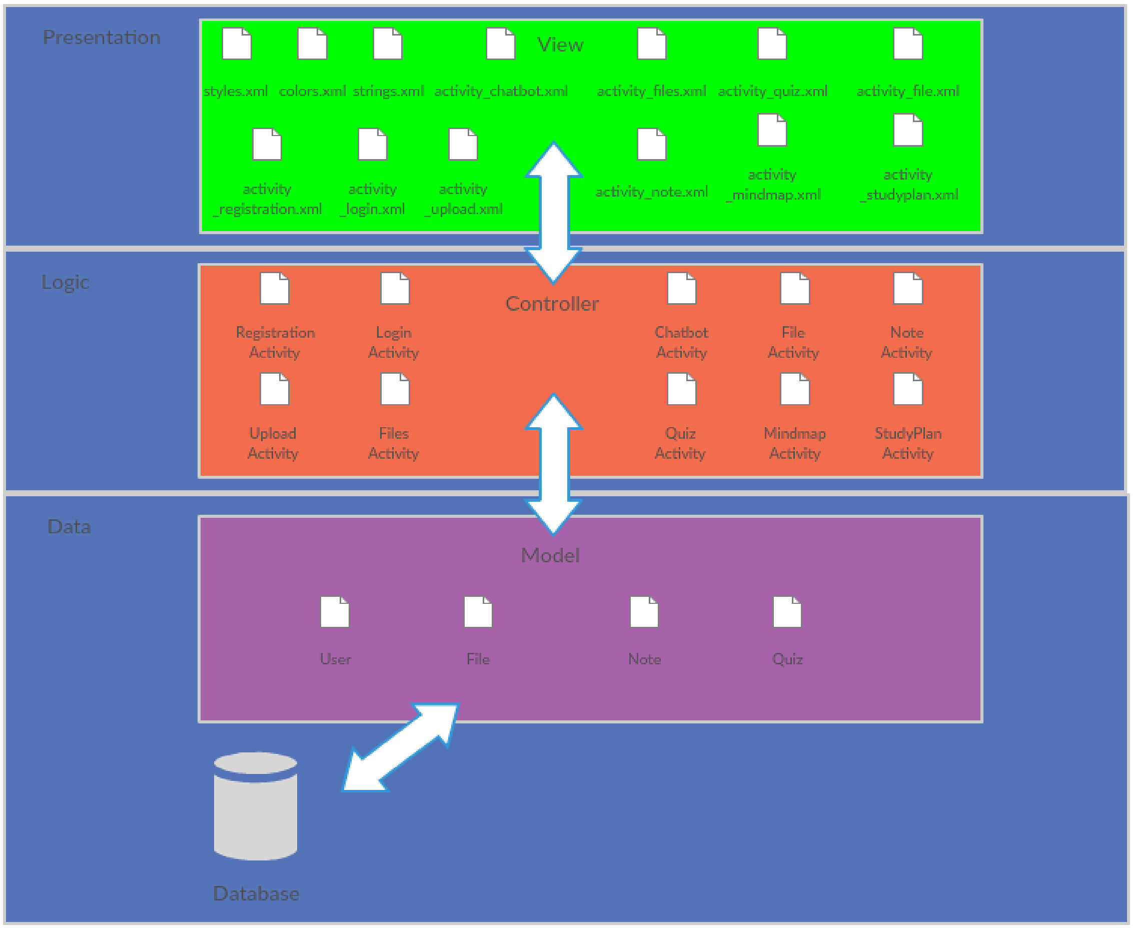


Figure 9: Scriba College software architecture diagram

The above diagram exhibits the software architecture for the Scriba College application. The system will utilize a three-tier architecture (Presentation, Logic and Data) and follow the MVC (Model-View-Controller) architectural design pattern for developing software. The application will be partitioned into three main components, the Model, the View and the Controller. The model is responsible for the data logic and contains object classes to be stored as data, while the view is responsible for the user interface logic and relates to the layout files, which are in XML format. The controller acts as an interface between the model and the view and is responsible for the business logic of the application. The controller relates to the Activity classes of the application that handle the user input from the view, update the database and updates the view accordingly.

2.2.2.4 Security Architecture

The application should uphold security for the system and the applications users by requesting permissions before the application can integrate with the user's device data and system features. The application should request the user to grant permission.

The login and registration system should be designed in a secure manner using the HTTPS (Hypertext Transfer Protocol Secure) protocol, which is a combination of HTTP (Hypertext Transfer Protocol) and SSL (Secure Sockets Layer). Communication between client and server should be encrypted and all user's passwords should be encrypted using the Secure Hash Algorithm (SHA) 256. The uploading and retrieval of files should also make use of the HTTPS protocol.

2.2.2.5 Communication Architecture

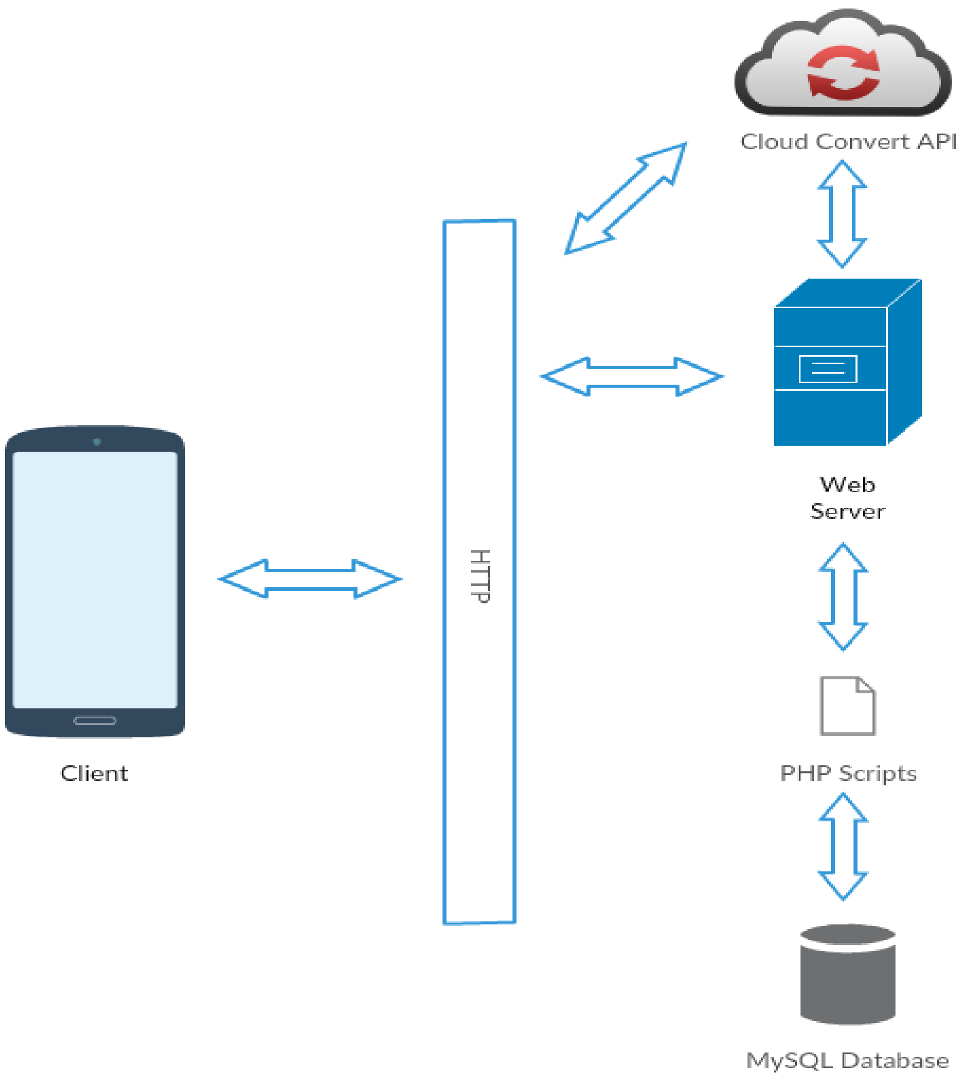


Figure 10: Scriba College Communication Diagram

The client device must establish a connection to the internet first in order to communicate with the web server. The web server will host the PHP scripts that will send and read to and from with the database system. The client will also make a request to the Cloud Convert API to convert a file and the API will respond with a HTML version of the file ready to be viewed by the client device.

2.2.2.6 Performance

The performance of the Scriba College will be dependent on response time, memory and scalability. The function for loading all the files from the server should retrieve all files within five seconds. Also, when uploading a file to the server, the task should take no more than five seconds. Creation of a quiz should take at maximum five seconds. The application should be fast and responsive to all device screen sizes. The application shall switch between activities in minimal time.

The applications database should provide sufficient storage memory to hold all user registration and login details, uploaded file information, all the created notes, questions, quizzes, mind maps and study plans.

The system should be scalable by offering the same functionality and response time as more users are added to the system and the volume in usage increases.

2.2.3 System Design

2.2.3.1 Database Design

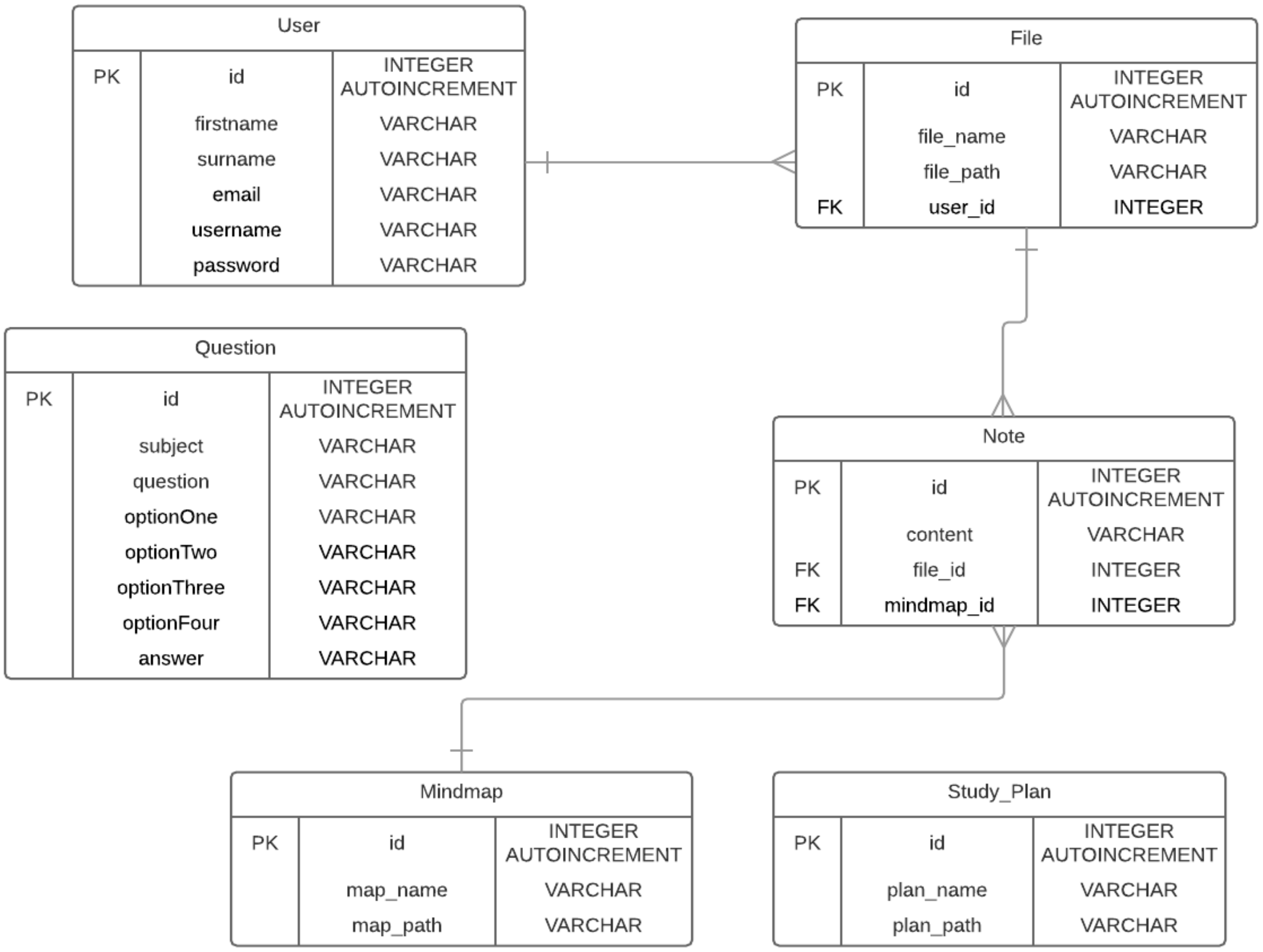


Figure 11: ERD of the Scriba College Database system

The database system for the Scriba College application will consist six table. The seven tables are as follows:

- User
- File
- Note
- Question
- Mind map
- Study Plan

The user table will store all the user's information such as name, email, username, and password. The file table will store information regarding the files uploaded by the user such as the name of the file and path of the file on the server. The relationship between the user table and the file table is a one-to-many relationship, as a user can have many files. The note table will store the content of the notes taken by the user from files. The relationship between the file table and the note table is a one-to-many relationship, as a file can have many notes associated with it. The question table holds all questions created by the user for future quizzes. The mind map table will contain the mind maps image name and the path of the mind map image on the server. The note table and the mind map table will form a many-to-one relationship, as many notes can be contained in a mind map. The study plan table will store information such the study plan's name and the path of the study plan file located on the server.

2.2.3.2 Data Conversions

There is a need for files to be accessible and viewable within the application and not with the assistance from any third-party applications e.g. the native Android PDF Reader.

Conversion of pdf and office files into different file formats (html) to enable viewing of document within Android web view, which also enabled text extraction from the file. The Android device (client) will send a request to the Cloud Convert API with the API key, file extension and URL path to the file as parameters and will respond with a converted file viewable in the Android web view.

The Android device (client) will send a HTTP request to the web server. The web server will then run the PHP scripts to connect to the MySQL database and make a request for data. The database will send a response back to the web server with the requested data. The PHP scripts on the web server will convert the received data into JSON and sends this data to the Android device (client). The client receives the data and parses this data into plain text and displays on screen.

2.2.3.3 Application Program Interfaces

Cloud Convert API – Cloud Convert API provides a RESTful API for converting multiple file types into HTML file formats.

Bluetooth Low Energy API – The Android Bluetooth Low Energy provides communication between Android devices and BLE devices.

Nordic Semiconductor Container application as a library – Contains the Android BLE API and methods for connection with the Scriba stylus.

2.2.3.4 User Interface Design

The application will follow the Android Material Design guidelines. Material design provides a new theme, new widgets and API's for designing Android applications. Documentation regarding Material Design can be found at the following link:

https://material.io/guidelines/

2.2.3.5 Performance

The application should thrive to produce smooth user interface performance. The application should provide user interface responsiveness by optimizing layout hierarchies to cater for every screen size on many different devices. The user interface elements such as buttons should respond immediately to user interaction and carry out the required functions. Dialogs should appear when necessary to inform user of any changes of state or any background tasks. The user interface should also be dynamic and make visible/invisible user interface elements that are applicable or not at the given time. Once an action is performed, user interface elements may appear or disappear.

2.2.3.6 Section 508 Compliance

The application will aim to be Section 508 compliant. The application will make use of text and images with high color contrast. The application should use incorporate complementary colors from the color wheel to design the user interface to accommodate for users with vision impairment's and color deficiencies. The application should also ensure all navigation elements are noticeable to the user.

2.3 Implementation

2.3.1 Upload a file to the server

The *UploadActivity* class contains the logic for uploading the files to the server. The below code shows the implementation of the *uploadFile(final String selectedFilePath)* method, which uploads a file to the server. The method expects a file path of string type to be passed as a parameter. An integer value is returned that represents the server's response code i.e. 200-OK.

```
/*
 * method that uploads file to server
 * @param selectedFilePath
 * @return serverResponseCode
 */
public int uploadFile(final String selectedFilePath){

    int serverResponseCode = 0;

    HttpURLConnection connection;
    DataOutputStream dataOutputStream;
    String lineEnd = "\r\n";
    String twoHyphens = "--";
    String boundary = "*****";


    int bytesRead,bytesAvailable,bufferSize;
    byte[] buffer;
    int maxBufferSize = 1 * 1024 * 1024;
    File selectedFile = new File(selectedFilePath);


    String[] parts = selectedFilePath.split("/");
    final String fileName = parts[parts.length-1];

    if (!selectedFile.isFile()){
        dialog.dismiss();

        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                tvFileName.setText("Source File Doesn't Exist: " +
selectedFilePath);
            }
        });
        return 0;
    }else{
        try{
            FileInputStream fileInputStream = new FileInputStream(selectedFile);
            URL url = new URL(Config.UPLOAD_URL);
            connection = (HttpURLConnection) url.openConnection();
            connection.setDoInput(true);//Allow Inputs
            connection.setDoOutput(true);//Allow Outputs
            connection.setUseCaches(false);//Don't use a cached Copy
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Connection", "Keep-Alive");
            connection.setRequestProperty("ENCTYPE", "multipart/form-data");
            connection.setRequestProperty("Content-Type", "multipart/form-
data;boundary=" + boundary);
            connection.setRequestProperty("uploaded_file",selectedFilePath);

            //creating new dataoutputstream
            dataOutputStream = new
DataOutputStream(connection.getOutputStream());

            //writing bytes to data outputstream
            dataOutputStream.writeBytes(twoHyphens + boundary + lineEnd);
            dataOutputStream.writeBytes("Content-Disposition: form-data;
name=\"uploaded_file\";filename=\""
                    + selectedFilePath + "\"" + lineEnd);

            dataOutputStream.writeBytes(lineEnd);
            //returns no. of bytes present in fileInputStream
            bytesAvailable = fileInputStream.available();
            //selecting the buffer size as minimum of available bytes or 1 MB
            bufferSize = Math.min(bytesAvailable,maxBufferSize);
            //setting the buffer as byte array of size of bufferSize
            buffer = new byte[bufferSize];

            //reads bytes from FileInputStream(from 0th index of buffer to size
of the buffer)
            bytesRead = fileInputStream.read(buffer,0,bufferSize);

            //loop repeats till bytesRead = -1, i.e., no bytes are left to read
            while (bytesRead > 0){
                //write the bytes read from inputstream
                dataOutputStream.write(buffer,0,bufferSize);
                bytesAvailable = fileInputStream.available();
                bufferSize = Math.min(bytesAvailable,maxBufferSize);
                bytesRead = fileInputStream.read(buffer,0,bufferSize);
            }

            dataOutputStream.writeBytes(lineEnd);
            dataOutputStream.writeBytes(twoHyphens + boundary + twoHyphens +
lineEnd);

            serverResponseCode = connection.getResponseCode();
            String serverResponseMessage = connection.getResponseMessage();

            Log.i(TAG, "Server Response is: " + serverResponseMessage + ": " +
serverResponseCode);

            // response code of 200 indicates the server status OK
            if(serverResponseCode == 200){
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        tvFileName.setText("File Upload completed”);
                    }
                });
            }

            //closing the input and output streams
            fileInputStream.close();
            dataOutputStream.flush();
            dataOutputStream.close();



        } catch (FileNotFoundException e) {
            e.printStackTrace();
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(UploadActivity.this,"File Not Found",
Toast.LENGTH_SHORT).show();
                }
            });
        } catch (MalformedURLException e) {
            e.printStackTrace();
            Toast.makeText(UploadActivity.this, "URL error!",
Toast.LENGTH_SHORT).show();

        } catch (IOException e) {
            e.printStackTrace();
            Toast.makeText(UploadActivity.this, "Cannot Read/Write File!",
Toast.LENGTH_SHORT).show();
        }
        dialog.dismiss();
        return serverResponseCode;
    }

}
```

The *uploadFile(final String selectedFilePath)* method is invoked when the upload button is clicked. The method validates that s file has been selected to upload by ensuring the string variable selectedFilePath contains a valid file path. The method requires a variable of type View, which in this case is the attachment icon.

```
@Override
public void onClick(View view) {
    if(view == attachment){

        // click show file chooser on attachment icon click
        showFileChooser();
    }
    if(view == buttonUpload){

        // if a file path is selected on upload button Click, upload file
        if(selectedFilePath != null){
            dialog = ProgressDialog.show(UploadActivity.this, "", "Uploading
File...", true);

            new Thread(new Runnable() {
                @Override
                public void run() {
                    // new thread to handle Http Operations
                    uploadFile(selectedFilePath);
                }
            }).start();
        }else{
            Toast.makeText(UploadActivity.this,"Please choose a File First",
Toast.LENGTH_SHORT).show();
        }
    }
}
```

The *showFileChooser()* method is used to send an intent to display all pickers for files with mime type set by the *setType()* method. This enables the user to pick a file to upload from their Android device.

```
private void showFileChooser() {
    int PICKFILE_RESULT_CODE=1;
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
    intent.setType("*/*");
    startActivityForResult(intent,PICKFILE_RESULT_CODE);
}
```

The below code is the server side PHP script for inserting document information into the File table in the database.

```
<?php
session_start();
$userid = $_SESSION['userid'];

if (is_uploaded_file($_FILES['uploaded_file']['tmp_name'])) {
    $uploads_dir = './uploads/';
    $tmp_name = $_FILES['uploaded_file']['tmp_name'];
    $file_name = $_FILES['uploaded_file']['name'];
    move_uploaded_file($tmp_name, $uploads_dir.$file_name);

    require_once('dbConnect.php');
    mysqli_query($con,"INSERT  INTO  `File`  (filename,filepath,ownerid)  VALUES
('".$file_name."','".$uploads_dir.$file_name."','$userid')") or
       trigger_error($con->error."[ $sql]");
    mysqli_close($con);
} else{
    echo "File not uploaded successfully.";
}
?>
```

2.3.2 Retrieve user files

The *MyFilesActivity* class contains the logic for retrieving files from the server that belong to a specific user. The below code shows the implementation of the *RetrieveJSONData* class, which extends AsyncTask to complete operations in the background, is contained within the *MyFilesActivity* class. This class contains the method *doInBackground(String… params)*, which uses the *HttpURLConnection* class to read all the data specified by the URL. When the operation is complete, the *onPostExecute(String result)* is invoked and assigns the JSON result to a string variable and calls the *showList()* method.

```
class RetrieveJSONData extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... params) {
        InputStream inputStream = null;
        String result = null;
        try {
            URL url = new URL(Config.RETRIEVE_FILES_URL);
            HttpURLConnection con = (HttpURLConnection) url.openConnection();

            inputStream = new BufferedInputStream(con.getInputStream());

            BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream, "UTF-8"), 8);
            StringBuilder sb = new StringBuilder();

            String line = null;
            while ((line = reader.readLine()) != null)
            {
                sb.append(line + "\n");
            }
            result = sb.toString();
        } catch (Exception e) {

        }
        finally {
            try{if(inputStream != null)inputStream.close();}catch(Exception
squish){}
        }
        return result;
    }

    @Override
    protected void onPostExecute(String result){
        myJSON=result;
        showList();
    }
}
```

The below code shows the implementation of the *showList()* method. This method creates a JSON array, which appends the results from the response. The JSON array holds the list of files in JSON format. Each file reference is then extracted to string variables, which in turn is added to a File object and add to an List of type File.

```
protected void showList(){
    try {
        JSONObject jsonObj = new JSONObject(myJSON);
        jsonFiles = jsonObj.getJSONArray(TAG_RESULTS);

        for(int i = 0; i< jsonFiles.length(); i++){

            int id = 0;
            String filename = null;
            String filepath = null;

            try {
                JSONObject c = jsonFiles.getJSONObject(i);
                {
                }
                id = c.getInt("id");
                filename = c.getString(TAG_FILENAME);
                filepath = c.getString(TAG_FILEPATH);

            } catch (JSONException e) {
                e.printStackTrace();
            }

            filesMap = new HashMap<String,String>();

            filesMap.put(TAG_FILE_ID, String.valueOf(id));
            filesMap.put(TAG_FILENAME, filename);
            filesMap.put(TAG_FILEPATH, filepath);

            File file = new File(id, filename, filepath);
            filesList.add(file);
        }

        CustomAdapter adapter = new CustomAdapter(MyFilesActivity.this,
R.layout.files_listview_item, filesList);

        list.setAdapter(adapter);

    } catch (JSONException e) {
        e.printStackTrace();
    }

}
```

The below code is the server side PHP script for retrieving files for a specific user from file table in the database.

```
<?php
 session_start();
 $userid = $_SESSION['userid'];

require_once('dbConnect.php');

$sql = "SELECT id, filename, filepath FROM File WHERE ownerid = '$userid'";
$res = mysqli_query($con,$sql);
$result = array();

while($row = mysqli_fetch_array($res)){
       array_push($result,
       array('id'=>$row[0],
       'filename'=>$row[1],
       'filepath'=>$row[2]
       ));
}
```

```
echo json_encode(array("result"=>$result));

mysqli_close($con);
?>
```

2.3.3 Create a note

The *WebViewActivity* class contains the logic for creating a note. The below code is used to extract the selected text from the Android WebView widget. As the WebView does not support the native Android text selection feature, it was necessary to implement a JavaScript function extract the selected text and return in string format. The *addNote(String noteContent)* method is then invoked and is pass the selected text as a parameter.

```
web.evaluateJavascript("(function(){return window.getSelection().toString()})()"
,new ValueCallback<String>() {
    @Override
    public void onReceiveValue(String selectedText) {
        Log.d("LogName", selectedText);
        content = selectedText;
    }}
);
addNote(content);
```

The below code shows the implementation of the *addNote(String noteContent)* method, which sends a POST request to the specified URL with the required parameters and listens for a server response to inform the user if a note was successfully created or not by displaying a message on screen.

```
private void addNote(String noteContent){

    StringRequest stringRequest = new StringRequest(Request.Method.POST,
Config.INSERT_NOTE_URL,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {

Toast.makeText(WebViewActivity.this,response,Toast.LENGTH_LONG).show();
                }
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
Toast.makeText(WebViewActivity.this,error.toString(),Toast.LENGTH_LONG).show();
                }
            }){

        @Override
        protected Map<String,String> getParams(){
            Map<String,String> params = new HashMap<String, String>();
            params.put(KEY_CONTENT, noteContent);
            params.put(KEY_FILE_ID, String.valueOf(file.getId()));
            return params;
        }

    };

    RequestQueue requestQueue = Volley.newRequestQueue(this);
    requestQueue.add(stringRequest);
}
```

The below code is the server side PHP script for inserting a note into the note table in the database.

```
 <?php

if($_SERVER['REQUEST_METHOD']=='POST'){
 $content= $_POST['content'];
 $fileid= $_POST['fileId'];

 if($content == ''){
 echo 'nothing selected';
 }else{
 require_once('dbConnect.php');

 $sql = "INSERT INTO Note (content, fileid) VALUES('$content', '$fileid')";
 if(mysqli_query($con,$sql)){
 echo 'Note Successfully Created';
 }else{
 echo 'Oops! Something went wrong. Please try again!';
 }
  }
   mysqli_close($con);
}else{
echo 'Error';
}
?>
```

2.3.4 Add a quiz question

The *QuizQuestionActivity* class contains the logic for creating questions for quizzes. The below code shows the implementation of the *createQuestion()* method, which sends a POST request to the specified URL with the required parameters and listens for a server response to inform the user if a question was successfully created or not by displaying a message on screen.

```
private void createQuizQuestion(){

    final String subject = editTextSubject.getText().toString();
    final String question = editTextQuestion.getText().toString();
    final String optionOne = editTextOptionOne.getText().toString();
    final String optionTwo = editTextOptionTwo.getText().toString();
    final String optionThree = editTextOptionThree.getText().toString();
    final String optionFour = editTextOptionFour.getText().toString();
    final String answer = editTextAnswer.getText().toString();

    final QuizQuestion quizQuestion = new QuizQuestion(subject, question,
optionOne, optionTwo, optionThree, optionFour, answer);


    quizQuestion.setSubject(editTextSubject.getText().toString());
    quizQuestion.setQuestion(editTextQuestion.getText().toString());
    quizQuestion.setOptionOne(editTextOptionOne.getText().toString());
    quizQuestion.setOptionTwo(editTextOptionTwo.getText().toString());
    quizQuestion.setOptionThree(editTextOptionThree.getText().toString());
    quizQuestion.setOptionFour(editTextOptionFour.getText().toString());
    quizQuestion.setAnswer(editTextAnswer.getText().toString());

    StringRequest stringRequest = new StringRequest(Request.Method.POST,
Config.INSERT_QUESTION_URL,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {

Toast.makeText(QuizQuestionsActivity.this,response,Toast.LENGTH_LONG).show();
                }
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {

Toast.makeText(QuizQuestionsActivity.this,error.toString(),Toast.LENGTH_LONG).sh
ow();
                }
            }){

        @Override
        protected Map<String,String> getParams(){
            Map<String,String> params = new HashMap<String, String>();
            params.put(KEY_SUBJECT, quizQuestion.getSubject());
            params.put(KEY_QUESTION, quizQuestion.getQuestion());
            params.put(KEY_OPTION_ONE, quizQuestion.getOptionOne());
            params.put(KEY_OPTION_TWO, quizQuestion.getOptionTwo());
            params.put(KEY_OPTION_THREE, quizQuestion.getOptionThree());
            params.put(KEY_OPTION_FOUR, quizQuestion.getOptionFour());
            params.put(KEY_ANSWER, quizQuestion.getAnswer());
```

```
                return params;
            }

        };

        RequestQueue requestQueue = Volley.newRequestQueue(this);
        requestQueue.add(stringRequest);
    }
```

The *QuizQuestion* model class contains the information for structuring a quiz question. This class contains variables such as subject, question, the four options for the user to pick from and the actual answer. Also provided are the getter and setter methods for each variable.

```
**
 * @author Ian Cunningham
 */

public class QuizQuestion {

    private String subject;
    private String question;
    private String optionOne, optionTwo, optionThree, optionFour;
    private String answer;

    public QuizQuestion(String subject, String question, String optionOne,
String optionTwo, String optionThree, String optionFour, String answer) {
        this.subject = subject;
        this.question = question;
        this.optionOne = optionOne;
        this.optionTwo = optionTwo;
        this.optionThree = optionThree;
        this.optionFour = optionFour;
        this.answer = answer;
    }

    public String getSubject() {
        return subject;
    }

    public void setSubject(String subject) {
        this.subject = subject;
    }

    public String getQuestion() {
        return question;
    }

    public void setQuestion(String question) {
        this.question = question;
    }

    public String getOptionOne() {
        return optionOne;
    }

    public void setOptionOne(String optionOne) {
        this.optionOne = optionOne;
    }

    public String getOptionTwo() {
        return optionTwo;
    }

    public void setOptionTwo(String optionTwo) {
        this.optionTwo = optionTwo;
    }

    public String getOptionThree() {
        return optionThree;
    }

    public void setOptionThree(String optionThree) {
        this.optionThree = optionThree;
    }

    public String getOptionFour() {
        return optionFour;
    }

    public void setOptionFour(String optionFour) {
        this.optionFour = optionFour;
    }

    public String getAnswer() {
        return answer;
    }

    public void setAnswer(String answer) {
        this.answer = answer;
    }
}
```

The below code is the server side PHP script for inserting a quiz question into the question table in the database.

```
<?php


if($_SERVER['REQUEST_METHOD']=='POST'){
 $subject= $_POST['subject'];
 $question= $_POST['question'];
 $optionOne= $_POST['optionOne'];
 $optionTwo= $_POST['optionTwo'];
 $optionThree= $_POST['optionThree'];
 $optionFour= $_POST['optionFour'];
 $answer= $_POST['answer'];


 require_once('dbConnect.php');
 $sql = "INSERT INTO Question (subject, question, optionOne, optionTwo, optionThree, optionFour, answer) VALUES('$subject', '$question', '$optionOne', '$optionTwo', '$optionThree', '$optionFour', '$answer')";

 if(mysqli_query($con,$sql)){

 echo 'Question Successfully Created';

 } else{

 echo 'Oops! Something went wrong. Please try again!';

 }

   mysqli_close($con);

} else{

echo 'Error';

}

?>
```

2.3.5 Connect to Scriba stylus

The ability to connect and communicate with the Scriba stylus is made easy by importing the nRF-Toolbox library into the project and adding a reference to the library in the dependencies within the project.

```
public void bluetooth(MenuItem item) {
    Intent intent= new Intent(this, HRSActivity.class);
    startActivity(intent);
}
```

2.3.6 Highlighting text with Scriba

The below code is the JavaScript function that highlights the selected text from a file via the Scriba stylus if in highlight mode. The code retrieves the current selected text by the user, finds the range of the selected text and then highlights the text yellow.

```
document.getElementById(\"page-container\").onclick = function() {\n" +
        "\t\t// Get Selection\n" +
        "    sel = window.getSelection();\n" +
        "    if (sel.rangeCount && sel.getRangeAt) {\n" +
        "        range = sel.getRangeAt(0);\n" +
        "    }\n" +
        "    // Set design mode to on\n" +
        "    document.designMode = \"on\";\n" +
        "    if (range) {\n" +
        "        sel.removeAllRanges();\n" +
        "         sel.addRange(range);\n" +
        "     }\n" +
        "     // Colorize text\n" +
        "     document.execCommand(\"BackColor\", false, \"yellow\");\n" +
        "     // Set design mode to off\n" +
        "     document.designMode = \"off\";\n" +
        "     \n" +
        "     return window.getSelection().toString();\n" +
        "     }
```

2.4 Graphical User Interface (GUI) Layout

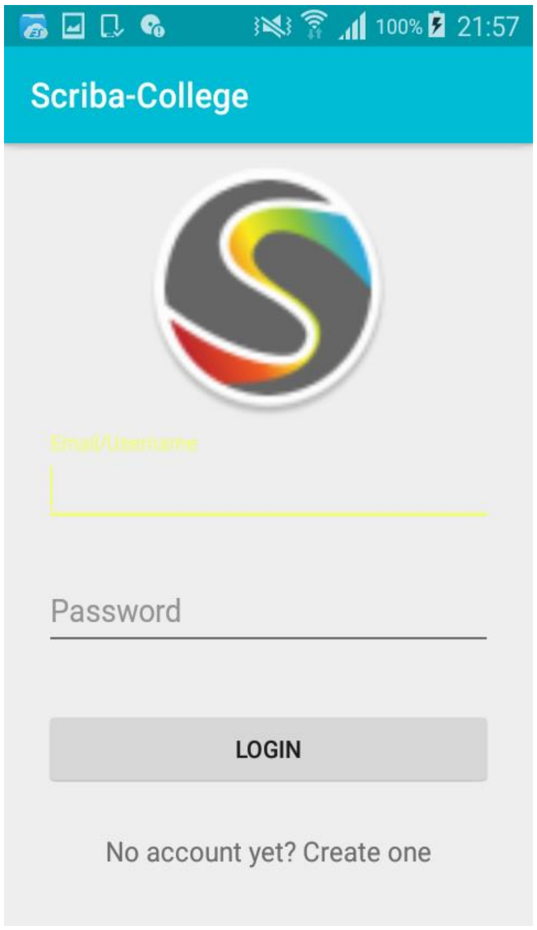


Figure 12: Login Activity

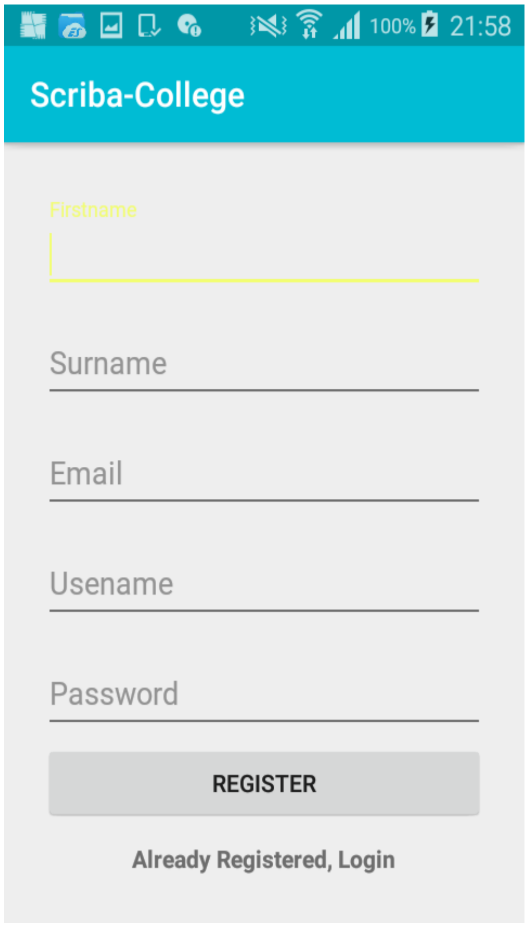


Figure 13: Signup Activity

Figure 12 shows the design of the Login page. The user will enter there email address and password, then press the sign in button. The user will then be redirected to the Upload File page which is shown in Figure 14. Alternatively, the user can create an account if they do not have an existing account. By pressing the join now button, the user will nbe redirected to the registration page shown in Figure 13. This page will require the user to enter their first anme, last name, email address and select a password.

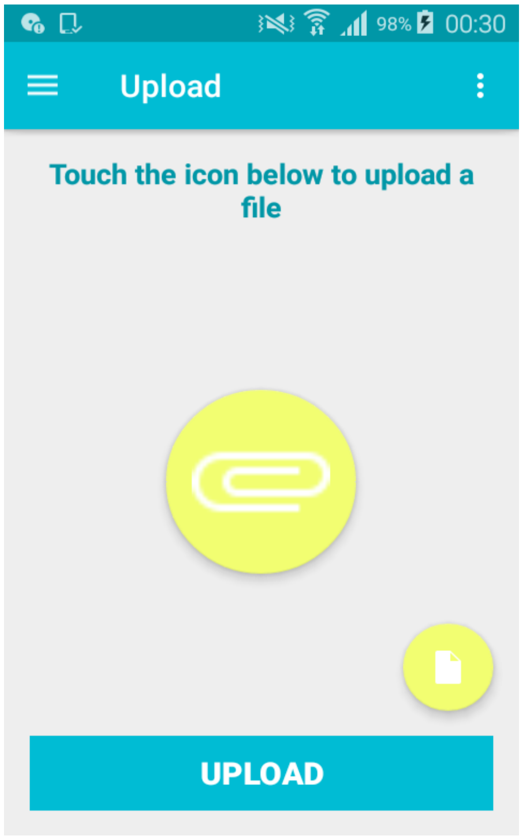


Figure 14: Upload Activity

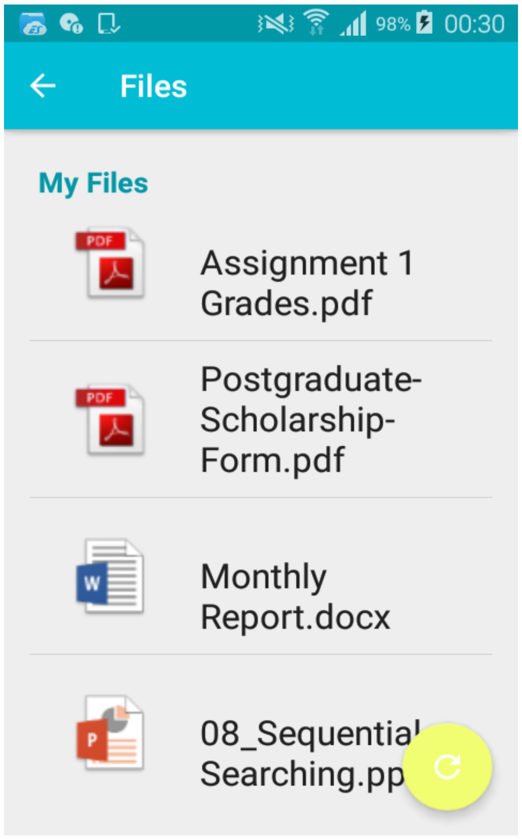
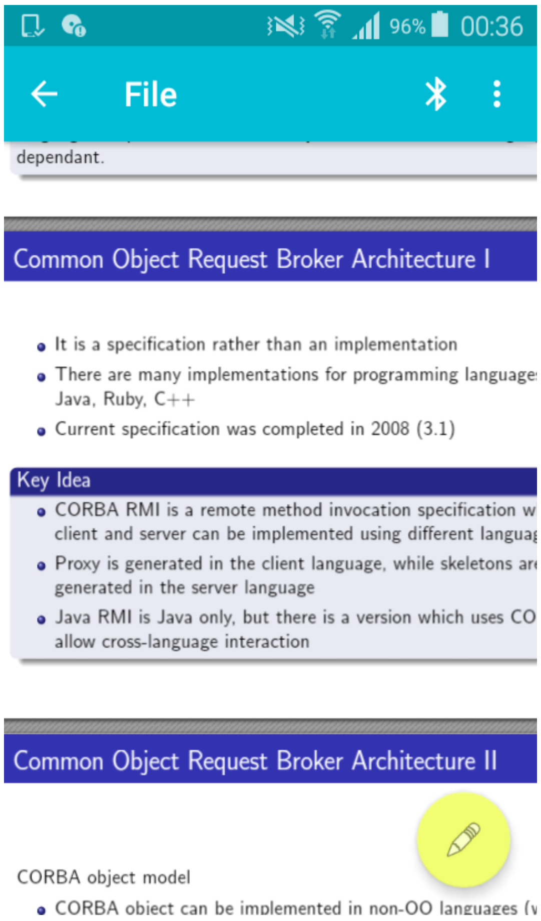


Figure 15: Files Activity

Figure 14 shows the Upload file page the user is redirected to upon successful login. By pressing the button in the middle of the page the user will be able to attach a file to upload. When a chosen file is selected, the user will upload by pressing the button labelled upload. The bu tton at the bottom right of the page will take the user to their uploaded files on the My Files page as seen in Figure 15. The My Files page will list all the users files and the user can view a files by simply pressing on one of the listed files.


96% 00:36

File

dependant.

Common Object Request Broker Architecture I

- It is a specification rather than an implementation
- There are many implementations for programming language
  Java, Ruby, C++
- Current specification was completed in 2008 (3.1)

Key Idea

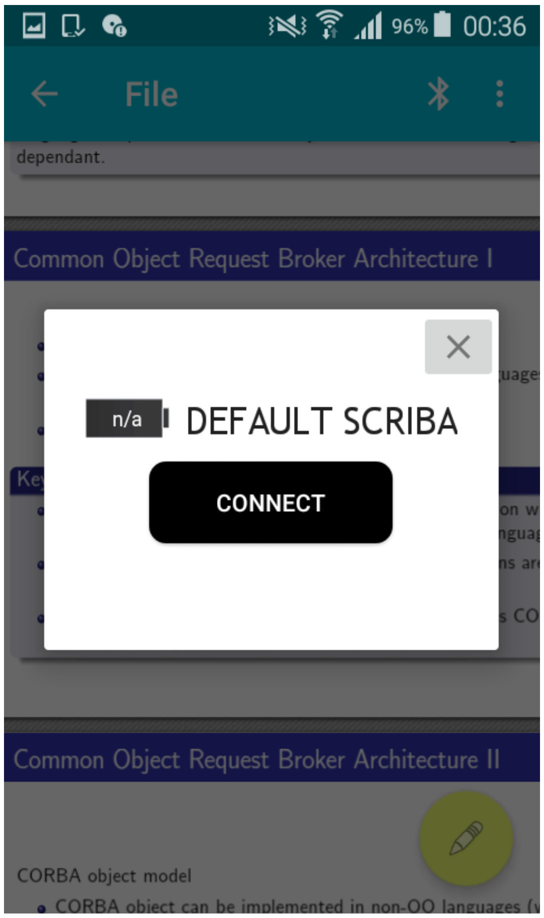
- CORBA RMI is a remote method invocation specification w
  client and server can be implemented using different languag
- Proxy is generated in the client language, while skeletons ar
  generated in the server language
- Java RMI is Java only, but there is a version which uses CO
  allow cross-language interaction

Common Object Request Broker Architecture II

CORBA object model

- CORBA object can be implemented in non-OO languages (v

Figure 16: View File Activity


96% 00:36

File

dependant.

Common Object Request Broker Architecture I

n/a DEFAULT SCRIBA

CONNECT

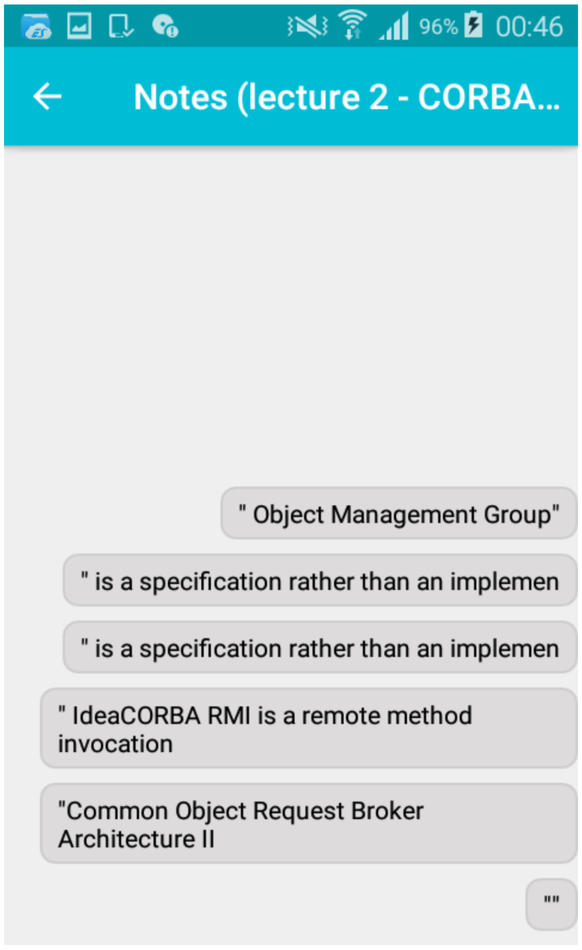
Common Object Request Broker Architecture II

CORBA object model

- CORBA object can be implemented in non-OO languages (v

Figure 17: Connect Scriba Dialog

Figure 16 shows the file that was selected by the user for viewing. By pressing the bluetooth icon on the toolbar, a dialog will prompt the user to connect their Scriba stylus as seen in Figure 17. When a Scriba is connected to the application, the user will annotate the content at their own will by simply drawing over the screen with the Scriba.


Notes (lecture 2 - CORBA...

" Object Management Group"

" is a specification rather than an implemen

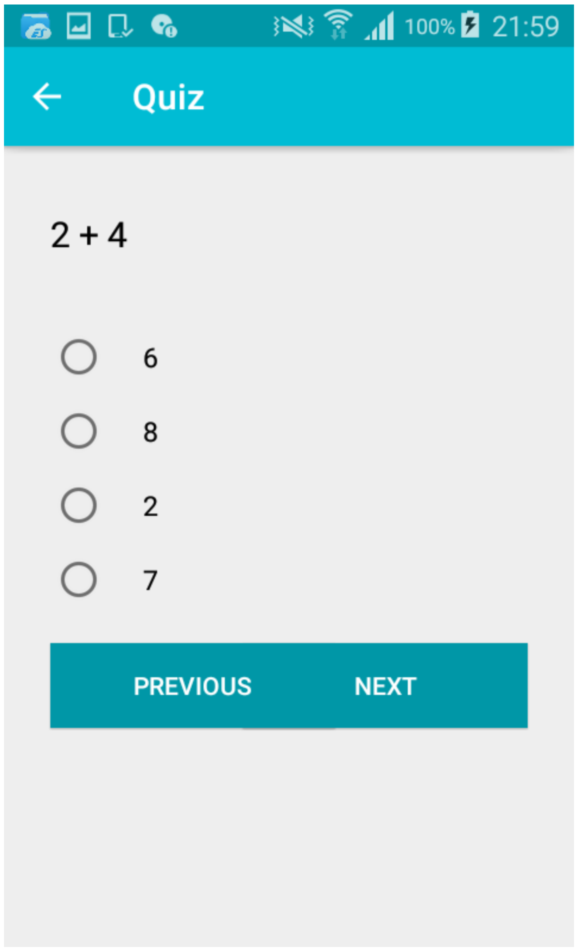
" is a specification rather than an implemen

" IdeaCORBA RMI is a remote method invocation

"Common Object Request Broker Architecture II

""

Figure 18: Notes Activity


Quiz

2 + 4

6

8

2

7

Figure 19: Quiz Activity

Figure 18 shows the notes activity were the notes that the user recorded for the specific file are shown.The user can then select to generate a quiz based on these stored questions and a quiz will be genrated as seen in Figure 19.

2.5 Testing

The Android Studio IDE was designed to make testing simple buy providing a built-in JUnit test framework. With Android, there is two types of unit tests, Local unit tests and Instrumented unit tests. Local unit tests are run on the local Java Virtual

Machine (JVM), while Instrumented tests are run on an Android device or an Android emulator (Energy, 2016).

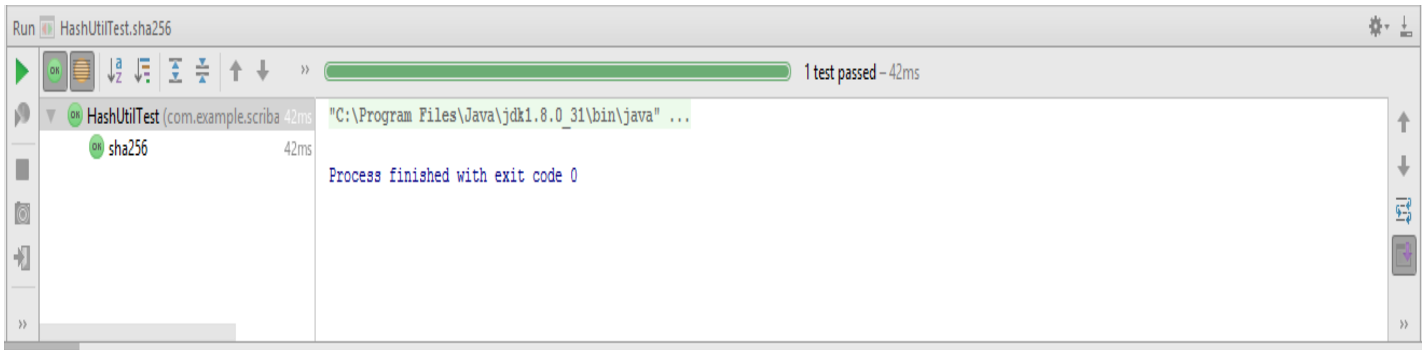
The tests will concentrate on testing the logic of the application, which will consist of many unit tests. Unit tests will be used to establish if the functionality of components is functioning as expected. Both local and instrumented testing will be applied to the application as their will be a need for testing individual functions without Android dependencies (local) and there will also be a need for testing functions that rely on the Android API (instrumented). Integration testing will be used to verify that the application behaves in a way that is expected such as user interactions and user input. Automation testing will also be used to simulate user interactions with application such as the sign up and login functions.

2.5.1 Unit testing

The JUnit Test Framework is a unit testing framework that provides testing functionality for the Java programming language. Provided below are unit test cases for the HashUtil class and the QuizQuestion model class.

2.5.1.1 HashUtilTest

```
/**
 * @author Ian Cunningham
 */
public class HashUtilTest {

    private String expected =
"5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8";
    private String input = "password";
    private String output;

    @Test
    public void sha256() throws Exception {
        output = HashUtil.sha256(input);
        assertEquals(expected, output);
    }
}
```

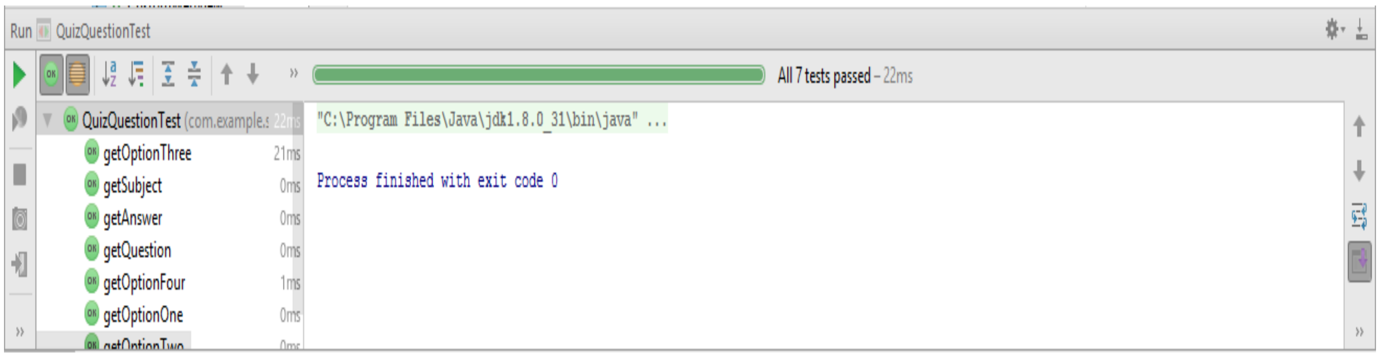


```
"C:\Program Files\Java\jdk1.8.0_31\bin\java" ...

Process finished with exit code 0
```

Figure 20: HashUtil unit test result

2.5.1.2 QuizQuestionTest

```
/**
 * @author Ian Cunningham
 */
public class QuizQuestionTest {

    @Test
    public void getSubject() throws Exception {
        QuizQuestion quizQuestion = new QuizQuestion();
        String expected = "Distributed Systems";
        quizQuestion.setSubject("Distributed Systems");
        String output = quizQuestion.getSubject();
        assertEquals(expected, output);
    }

    @Test
    public void getQuestion() throws Exception {
        QuizQuestion quizQuestion = new QuizQuestion();
        String expected = "Which of the following is a falacie of distributed
computing";
        quizQuestion.setQuestion("Which of the following is a falacie of
distributed computing");
        String output = quizQuestion.getQuestion();
        assertEquals(expected, output);
    }

    @Test
    public void getOptionOne() throws Exception {
        QuizQuestion quizQuestion = new QuizQuestion();
        String expected = "The network is reliable";
        quizQuestion.setOptionOne("The network is reliable");
        String output = quizQuestion.getOptionOne();
        assertEquals(expected, output);
    }

    @Test
    public void getOptionTwo() throws Exception {
        QuizQuestion quizQuestion = new QuizQuestion();
        String expected = "The network is reliable all the time, whatever the
situation";
        quizQuestion.setOptionTwo("The network is reliable all the time,
whatever the situation");
        String output = quizQuestion.getOptionTwo();
        assertEquals(expected, output);
    }

    @Test
    public void getOptionThree() throws Exception {
        QuizQuestion quizQuestion = new QuizQuestion();
        String expected = "Topology always changes";
        quizQuestion.setOptionThree("Topology always changes");
        String output = quizQuestion.getOptionThree();
        assertEquals(expected, output);
    }

    @Test
    public void getOptionFour() throws Exception {
        QuizQuestion quizQuestion = new QuizQuestion();
        String expected = "There is two administrators";
        quizQuestion.setOptionFour("There is two administrators");
        String output = quizQuestion.getOptionFour();
        assertEquals(expected, output);
    }

    @Test
    public void getAnswer() throws Exception {
        QuizQuestion quizQuestion = new QuizQuestion();
        String expected = "The network is reliable";
        quizQuestion.setAnswer("The network is reliable");
        String output = quizQuestion.getAnswer();
        assertEquals(expected, output);
    }
}
```



```
Run  QuizQuestionTest
All 7 tests passed – 22ms

QuizQuestionTest (com.example.  22ms
    getOptionThree   21ms
    getSubject        0ms
    getAnswer         0ms
    getQuestion       0ms
    getOptionFour     1ms
    getOptionOne      0ms
    getOptionTwo      0ms

"C:\Program Files\Java\jdk1.8.0_31\bin\java" ...

Process finished with exit code 0
```

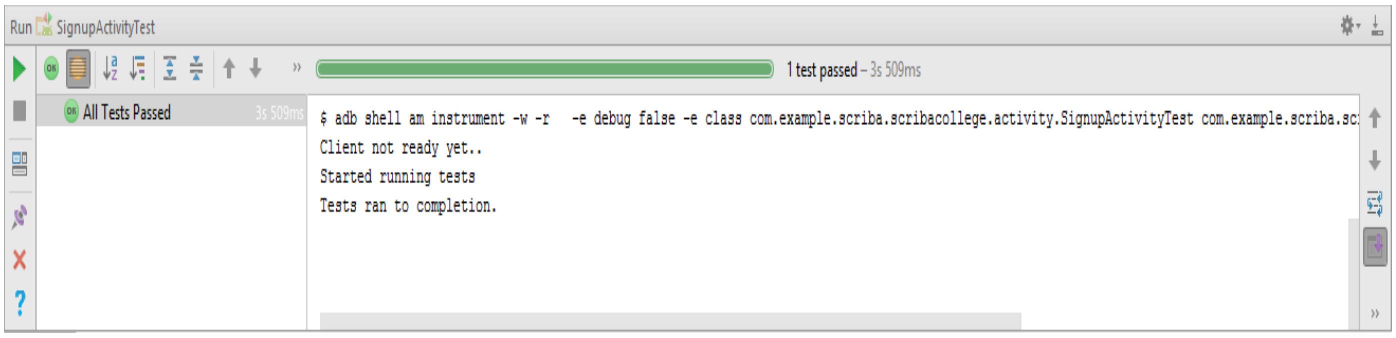
Figure 21: QuizQuestion unit test result

2.5.2 Integration testing

The integration tests were performed using the Expresso testing framework. The Expresso testing framework functions with the AndroidJUnitRunner and provides an instrumentation-based API that provides support for automated UI testing. Presented below are the automated tests for simulating user actions for registering with the application and signing into the application.

2.5.2.1 SignupActivityTest

```
@LargeTest
@RunWith(AndroidJUnit4.class)
public class SignupActivityTest {

    @Rule
    public ActivityTestRule<LoginActivity> mActivityTestRule = new
ActivityTestRule<>(LoginActivity.class);

    @Test
    public void signupActivityTest() {
        pressBack();

        ViewInteraction appCompatTextView = onView(
                allOf(withId(R.id.linkSignup), withText("No account yet? Create
one")));
        appCompatTextView.perform(scrollTo(), click());

        ViewInteraction appCompatEditText = onView(
                withId(R.id.editTextFirstname));
        appCompatEditText.perform(scrollTo(), replaceText("Lee"),
closeSoftKeyboard());

        ViewInteraction appCompatEditText2 = onView(
                withId(R.id.editTextSurname));
        appCompatEditText2.perform(scrollTo(), replaceText("McCarthy"),
closeSoftKeyboard());

        ViewInteraction appCompatEditText3 = onView(
                withId(R.id.editTextEmail));
        appCompatEditText3.perform(scrollTo(), replaceText("lmac@gmail.com"),
closeSoftKeyboard());

        ViewInteraction appCompatEditText4 = onView(
                withId(R.id.editTextUsername));
        appCompatEditText4.perform(scrollTo(), replaceText("lmac"),
closeSoftKeyboard());

        ViewInteraction appCompatEditText5 = onView(
                withId(R.id.editTextPassword));
        appCompatEditText5.perform(scrollTo(), replaceText("password"),
closeSoftKeyboard());

        ViewInteraction appCompatEditText6 = onView(
                allOf(withId(R.id.editTextPassword), withText("password")));
        appCompatEditText6.perform(pressImeActionButton());

        ViewInteraction appCompatButton = onView(
                allOf(withId(R.id.buttonRegister), withText("Register"),
                        withParent(withId(R.id.content_main))));
        appCompatButton.perform(scrollTo(), click());

    }

}
```

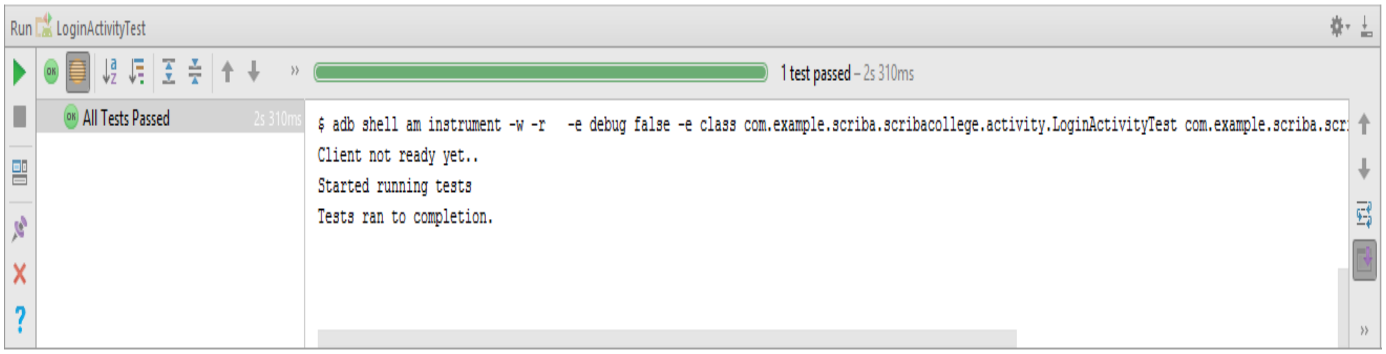


```
$ adb shell am instrument -w -r   -e debug false -e class com.example.scriba.scribacollege.activity.SignupActivityTest com.example.scriba.sc
Client not ready yet..
Started running tests
Tests ran to completion.
```

Figure 22: SignupActivity automated test result

2.5.2.2 LoginActivityTest

```
@LargeTest
@RunWith(AndroidJUnit4.class)
public class LoginActivityTest {

    @Rule
    public ActivityTestRule<LoginActivity> mActivityTestRule = new
ActivityTestRule<>(LoginActivity.class);

    @Test
    public void loginActivityTest() {
        ViewInteraction appCompatEditText = onView(
                withId(R.id.editTextEmail));
        appCompatEditText.perform(scrollTo(), replaceText("lmac@gmail.com"),
closeSoftKeyboard());

        ViewInteraction appCompatEditText2 = onView(
                withId(R.id.editTextPassword));
        appCompatEditText2.perform(scrollTo(), replaceText("password"),
closeSoftKeyboard());

        ViewInteraction appCompatEditText3 = onView(
                allOf(withId(R.id.editTextPassword), withText("password")));
        appCompatEditText3.perform(pressImeActionButton());

        ViewInteraction appCompatButton = onView(
                allOf(withId(R.id.buttonLogin), withText("Login")));
        appCompatButton.perform(scrollTo(), click());

    }
}
```



1 test passed – 2s 310ms

```
$ adb shell am instrument -w -r   -e debug false -e class com.example.scriba.scribacollege.activity.LoginActivityTest com.example.scriba.scr
Client not ready yet..
Started running tests
Tests ran to completion.
```

Figure 23: LoginActivity automated test result

2.6 Customer testing

The application was tested by the Customer David Craig, CEO Dublin Design Studio LTD and feedback was acquired. The application was also tested by a host of people such as fellow students to gain an understanding of the usability of the application, as this application is aimed at students. A set of tasks to be completed during the test was devised and a questionnaire was devised for the testers to complete to give their overall feedback on the application. An email was sent to the participants asking of them to participate in the testing process. A date was scheduled with testers that accepted the invitation to take part in the test. As all participants, would require a Scriba stylus to access all functionality within the application, it was necessary for a Scriba stylus be provided and thus all participants were required to participate in the test at a specific location. The application will also be deployed to the Google Play Store in the future.

2.6.1 Tasks to be completed by tester

- Upload a file to the server
- Open a file from the server
- Create a note
- Create a quiz question

2.6.2 Questionnaire

The questionnaire comprised of ten questions. The questions put forward to all participants during the testing process are as follows:

1. What is your background?

2. What is your gender?

3. Do you engage with smartphone applications regularly?

4. Are you familiar with Android operating system and Android devices in general?

5. What is your preferred learning style?

6. Have you previously used any study tool applications?

7. Did you find that the design of the application was user-friendly?

8. Did you find the application easy to use and navigate?

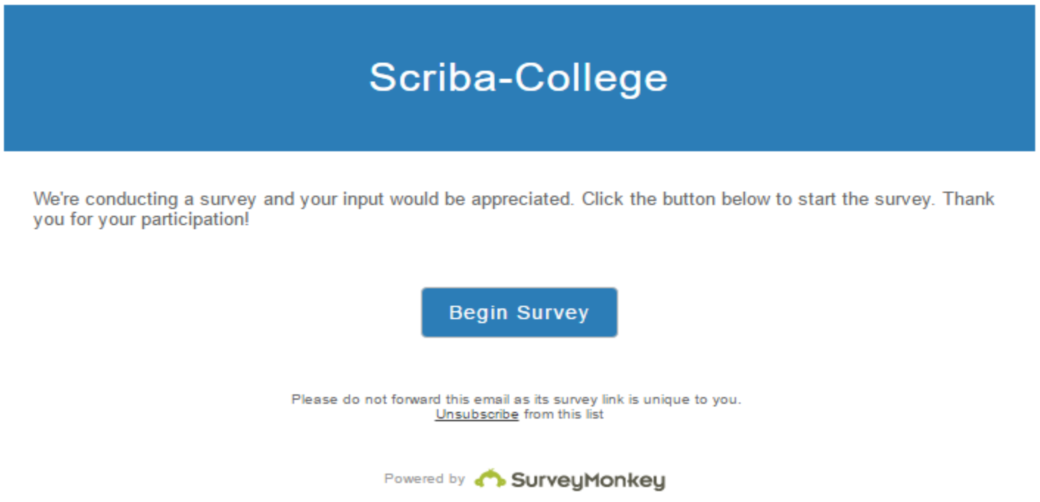
9. Do you have any recommendations for the application?

10. Would you use this application for studying purposes in the future?

2.7 Evaluation

The system was evaluated using several different techniques such as unit testing, integration testing and customer testing. As shown above all unit tests passed. Integration testing allowed to test how different parts of the application works with components of the application such as the MySQL database. The integration tests allowed for the ability to simulate user interactions with the application. All integration tests passed successfully.

An email was sent to 10 students to establish if they would be interested in participating in testing the Scriba-College application. The recipient of the email was informed on the process of the test, which included a set of steps to complete within the application and then a questionnaire to be completed, which focuses on the user’s perception of the application. Unfortunately, out of the ten emails sent, only six responded with an acceptance email. Below is an example email sent to a participant after they have tested the application, which contains a link to the questionnaire.



Scriba-College

We're conducting a survey and your input would be appreciated. Click the button below to start the survey. Thank you for your participation!

Please do not forward this email as its survey link is unique to you.
Unsubscribe from this list

Figure 24: Email containing link to questionnaire

The results of the questionnaires from the six participants were recorded and analysed, which generated a view on the usability of the application. Another email was sent to the test participants to thank them for their participation and to inform them that their feedback is valued. All ten questions were completed by the six participants.

Question One

The results showed that 100% of the respondents were students.

1. What is your background?

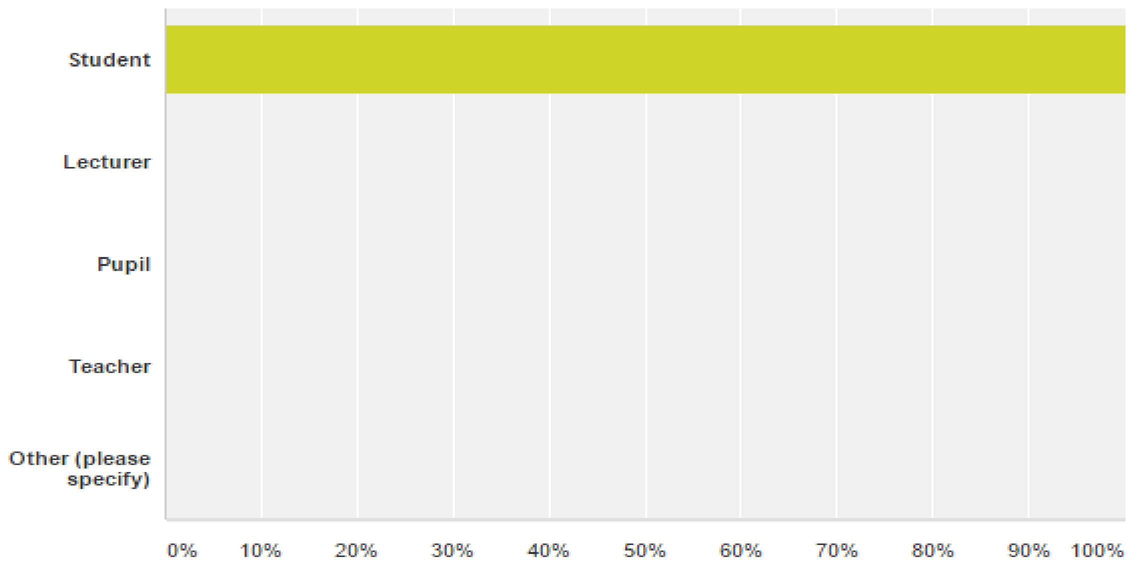


Figure 25: Question one results

Question Two

The results also showed that 66% of the respondents were male and 33% female.

2. What is your gender?

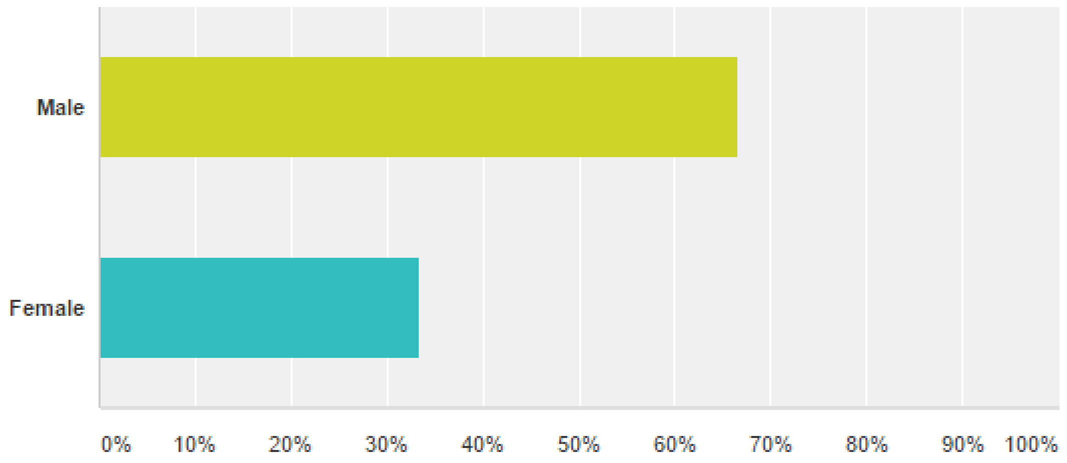


Figure 26: Question two results

Question Three

The results also show that every participant that took the test were aware of smartphone applications and engage with smartphone applications on a regular basis. 100% of the respondents answered yes.

3. Do you engage with smartphone applications regularly?

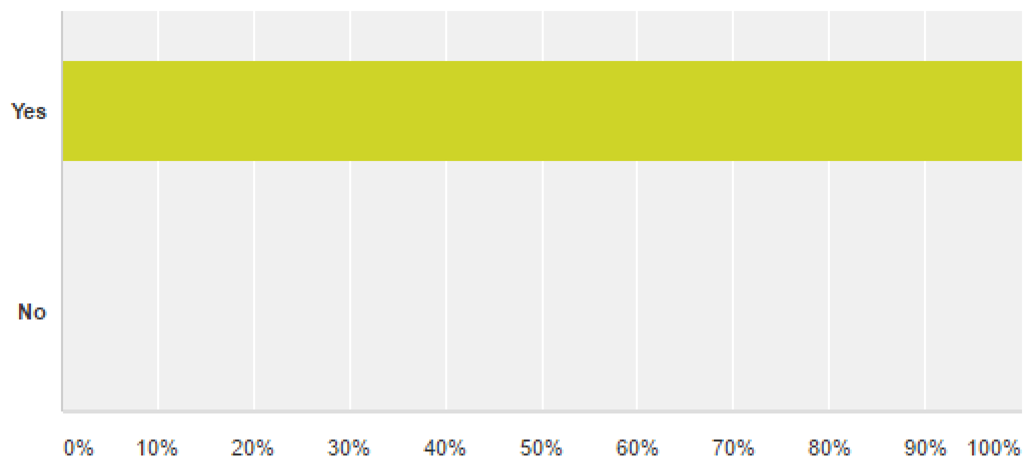


Figure 27: Question three results

Question Four

There was a 66.67% familiarity of the Android operating system. Out of the 6 respondents, four were familiar with Android platform as they currently owned an Android device. The other two respondents, 33.33% of the respondents, were current owners of an iPhone and had always used Apple devices.

4. Are you familiar with Android operating system and Android devices in general?

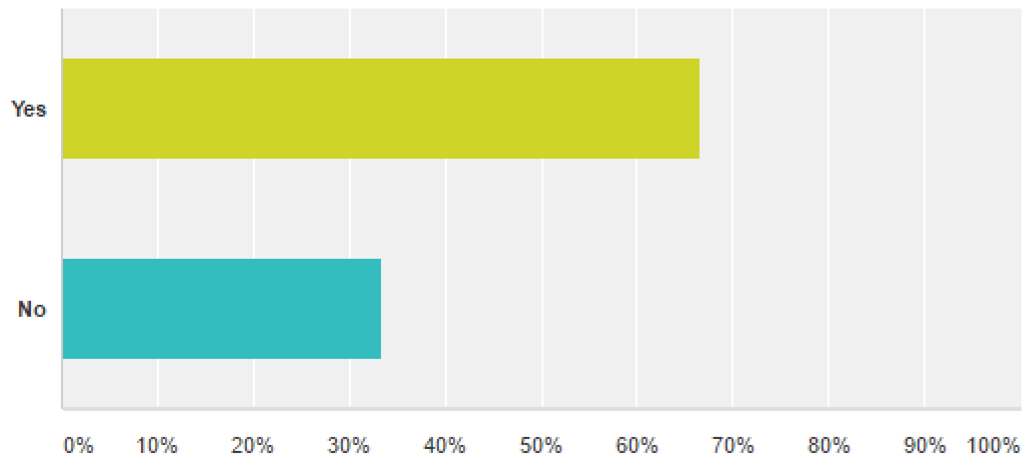


Figure 28: Question four results

Question Five

Out of the six respondents, 33.33% preferred visual learning, 33.33% preferred verbal learning and 33.33% preferred physical learning.

5. What is your preferred learning style?

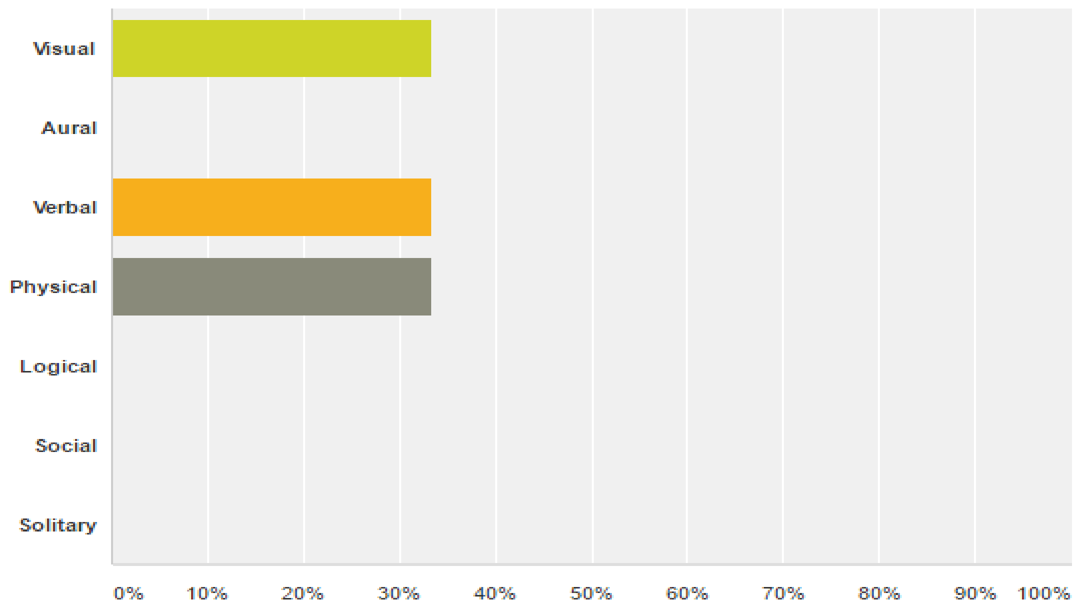


Figure 29: Question five results

Question Six

The results show that 66.67% of the respondents have previously used a current study tool application. One respondent mentioned the iTunes University study tool. 33.33% of the respondents have not previously used a study tool application.

6. Have you previously used any study tool application?

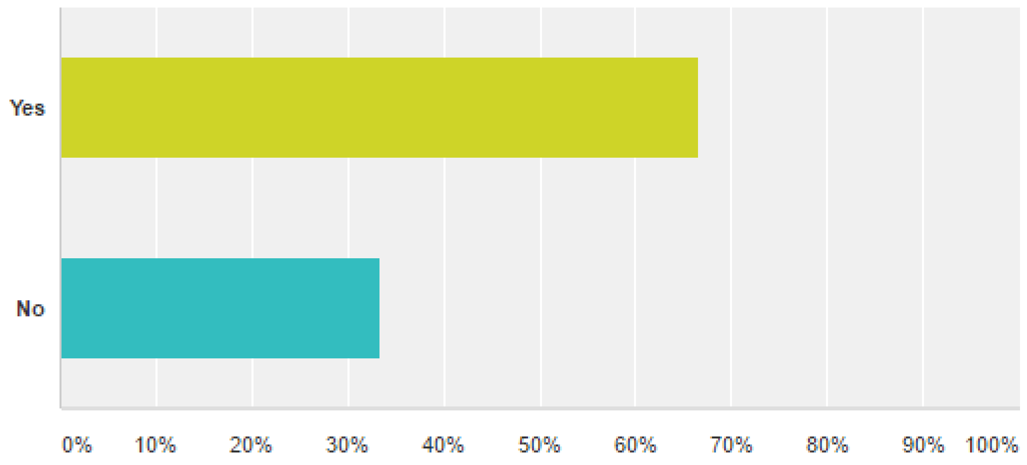


Figure 30: Question six results

Question Seven

Out of all the respondents, 100% found that the design of the application was user-friendly.

7. Did you find that the design of the application was user-friendly?

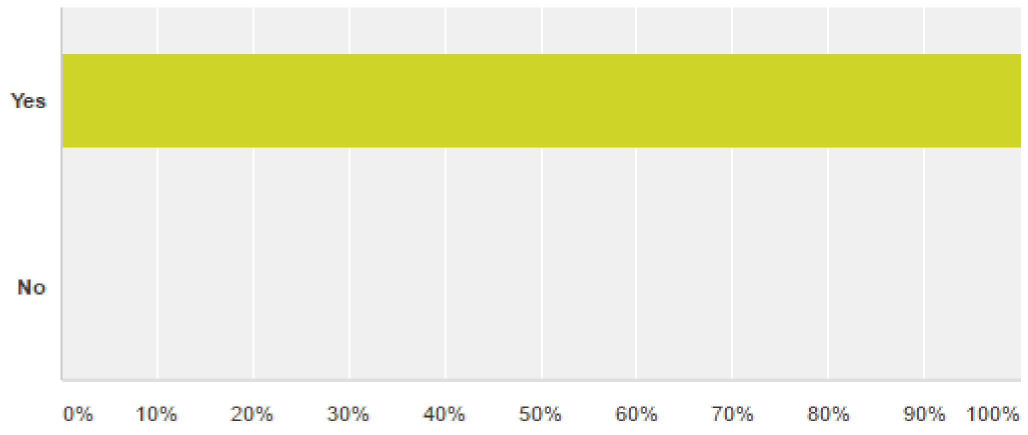


Figure 31: Question seven results

Question Eight

66.67% of the respondents found the application easy-to-use and navigate, while 33.33% did not find the application easy to use and navigate. Reasons mentioned included not being familiar with the Android operating system and Android devices. Also, a respondent mentioned that it wasn't obvious as to where all navigation buttons are located. This will be rectified to eliminate any future confusion when using the application.

8. Did you find the application easy to use and navigate?

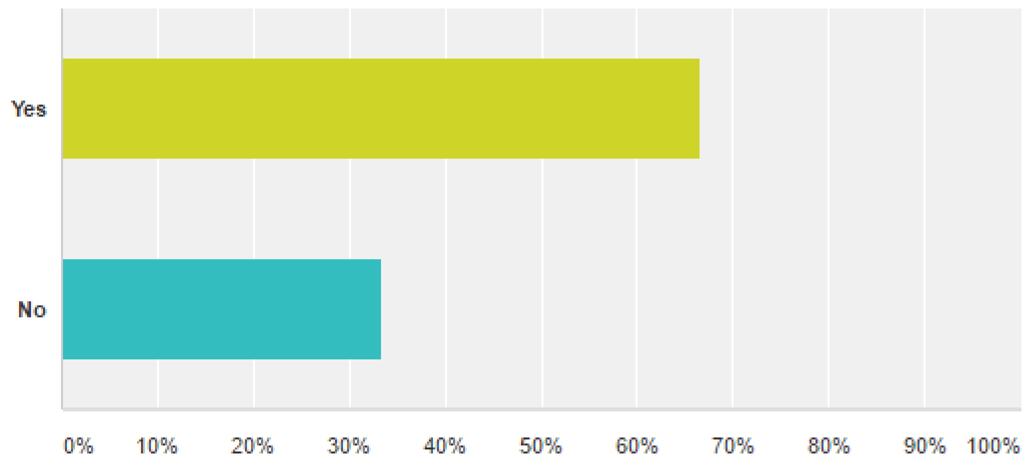


Figure 32: Question eight results

Question Nine

Once again, 66.67% of the respondents gave a recommendation for the application. 33.33% of the respondents did not make a recommendation. Some of the recommendations included be more obvious with the navigation and different colors used. For example, a respondent did not like the yellow on the input boxes

9. Do you have any recommendations for the application?

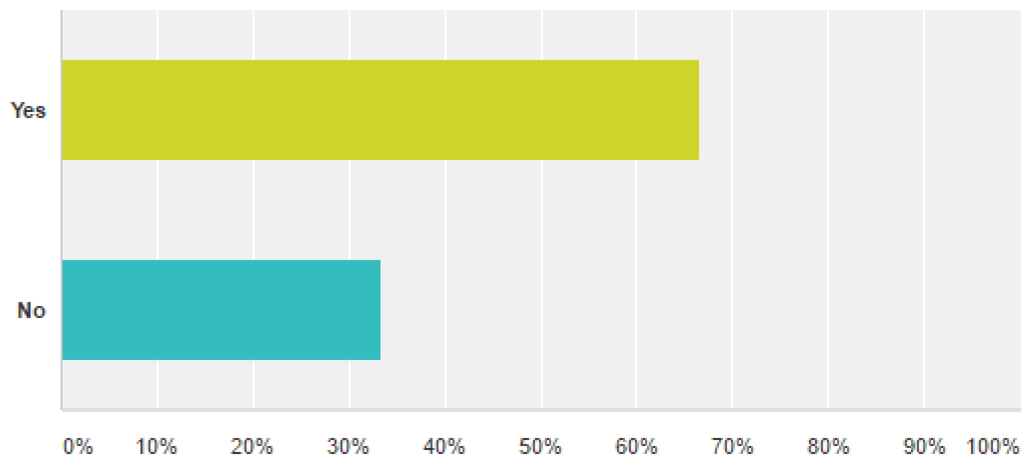


Figure 33: Question nine results

Question Ten

All participants of the questionnaire responded that they would use this application for studying purposes in the future.

10. Would you use this application for studying purposes in the future?

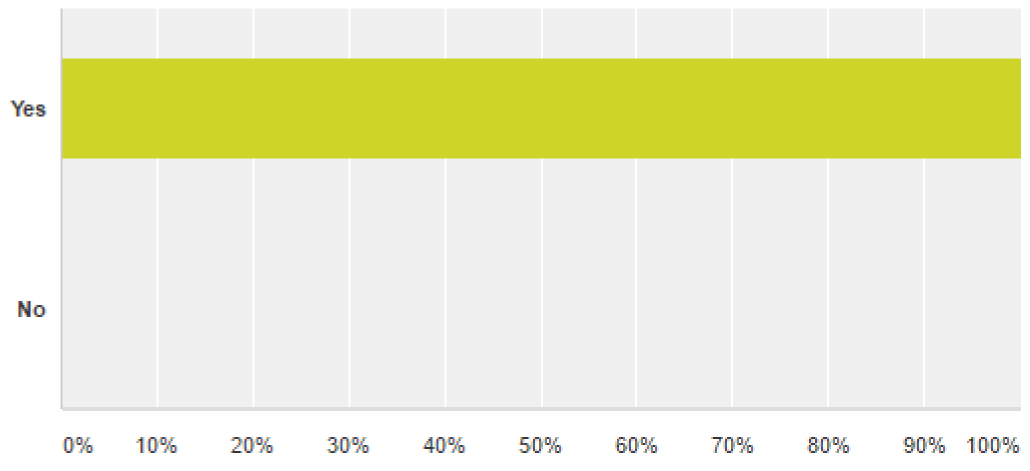


Figure 34: Question ten results

3 Conclusions

The advantages of the Scriba College application are that it will increase student engagement with college material by providing a new and intuitive way of creating and storing college notes. The student's notes will be stored in centralised area with ease of access to the student at anytime and anyplace. The application will enable the student to annotate the college material in a way that that will make it easier for student to recall important sections and clarify their own understanding of the study material. The application will also test the student on their knowledge of their college material by providing quiz's.

The disadvantages of the application include the necessity that the user must own an Android device. This will mean that owners of Apple and Windows phones will not be able to make use of the application. Another disadvantage of the application is that the user is required to purchase a Scriba stylus to enable full functionality from the application.

4 Further development or research

The application over time could evolve into a tool that could be utilized in a workplace and school environment. There are many different types of scenarios where this application could be useful and does not have to be constrained to just a college environment. Also, multi-language support functionality could be implemented to attract interest globally.

The application could also be further expanded in the future by creating a version of the application for the Apple platform and Windows platform, or build a cross-platform application that would run on all three platforms.

5 References

Energy, B. (2016). *Bluetooth Low Energy* | *Android Developers*. [online] Developer.android.com. Available at: https://developer.android.com/guide/topics/connectivity/bluetooth-le.html [Accessed 6 Dec. 2016].

Testing, G. (2016). *Getting Started with Testing* | *Android Developers*. [online] Developer.android.com. Available at: https://developer.android.com/training/testing/start/index.html [Accessed 6 Dec. 2016].

Usabilityfirst.com. (2016). *Usability First - About Usability - Requirements Specification* | *Usability First*. [online] Available at: http://www.usabilityfirst.com/about-usability/requirements-specification/ [Accessed 3 Dec. 2016].

Using the Material Theme | Android Developers. 2017. Using the Material Theme | Android Developers. [ONLINE] Available at: https://developer.android.com/training/material/theme.html. [Accessed 12 February 2017].

6 Appendix

6.1 Project Proposal

Project Proposal

Scriba College

Ian Cunningham

X13114425

Ian.Cunningham@student.ncirl.ie

BSc (Hons) in Computing – BSHCSD4

Software Development

Date : 21/10/2016

Contents

6.1.1 Objectives

The goal of this project is to create a scalable mobile application that runs on the Android platform. The application will aim to serve as study tool for students, which makes accessing and interacting with class content easier. The app will allow the user to annotate the information received in a manner that will make it easier for them to recall important sections or clarify their own understanding of the content. The application will store the student's presentation or study documents in the form of files such as docx, pdf, ppt etc. The app will integrate the Scriba stylus to allow them to quickly interact with the learning content. Depending on what mode the Scriba is in users will be able draw, highlight or erase. When highlight or draw mode the content that is selected will be saved as a note to accompany the presentation. The student will also be able to share their notes with their class mates.

The project is aimed at students using Android devices so that in class they can bring their device and Scriba and record important notes while in class from the class material. The application will provide an original method of interacting with study material that focuses the user's attention on the learning process rather and aids future study and recollection.

Objective 1: Serve as a study tool

Objective 2: Increase student engagement with class content

Objective 3: Enhance users understanding of the class content

Objective 4: Develop a way of accessing class notes on the go

Objective 5: Aid future study and recollection

Objective 6: Encourage student-to-student engagement by sharing notes with class mates

Objective 7: Also, serve as a cloud storage solution for college material.

6.1.2 Background

The stylus product “Scriba” is an award winning new stylus, which is designed by Dublin Design Studio. Scriba is designed around the movements of the hand to make it as comfortable as possible to use. Supporting both left and right handed users, Scriba works with mobile and tablet devices and contains a squeeze motion technology to increase control and offer an innovative experience. Scriba’s body bends to the users every touch and movements are detected allowing the user to access different application functionality.

Students often find it hard to recall important information from class material or leave it to the last minute to gather all their notes together for studying purposes before exams or continuous assessments. This application will allow students to have their notes all stored in one location. Students will record notes during class and these notes will be saved to accompany the specific document. Therefore when the student takes important notes in class, the student will know when it comes to exam time that these notes are important and can be easily accessed. The student can then read over their notes anywhere and anytime. The application will provide a way for users to increase engagement with the class content, therefore gaining better and more understanding of the class material.

I am comfortable in Android Development and have created numerous Android applications in previous projects. I have also created Android applications for a company while on work placement and these applications also integrated functionality with the Scriba stylus.

6.1.3 Technical Approach

In order to have the most effective impact on interacting with the students, an Android mobile application would best suit the needs of the project. I have chosen to develop for Android devices as it used the most widely used operating system in the mobile market. I am experienced in Android Development and have created numerous Android applications for college projects and work placement projects.

Also another important factor that made look in the direction of Android was the matter of cost-effectiveness for the student as Android devices can purchased from as little as €50.

The files that the user uploads to the application will be stored in a file folder on the server and the filename and path to the file on the server will be stored in a MySQL database hosted on a cloud hosting service called “x10 Hosting”. The files will be saved and retrieved from the server using PHP. The file data will be retrieved in JSON format. The JSON string will then be parsed to display the name of each file in an Android List View.

Following a lot of research regarding opening files within my Android application, I have concluded that best solution would be to use an Android Web View for containing the opened files and use Googles free embed tool called GView for displaying the files content. This allows to view all your files in a Google Drive preview. This can be achieved by placing http://docs.google.com/gview?url= in front of the full path URL to the file.

The Implementation is described in detail in Technical Details, although this may change over time.

6.1.4 Special resources required

Bluetooth Low Energy (BLE) API - For this project, the app is required to interact with the Scriba stylus via BLE.

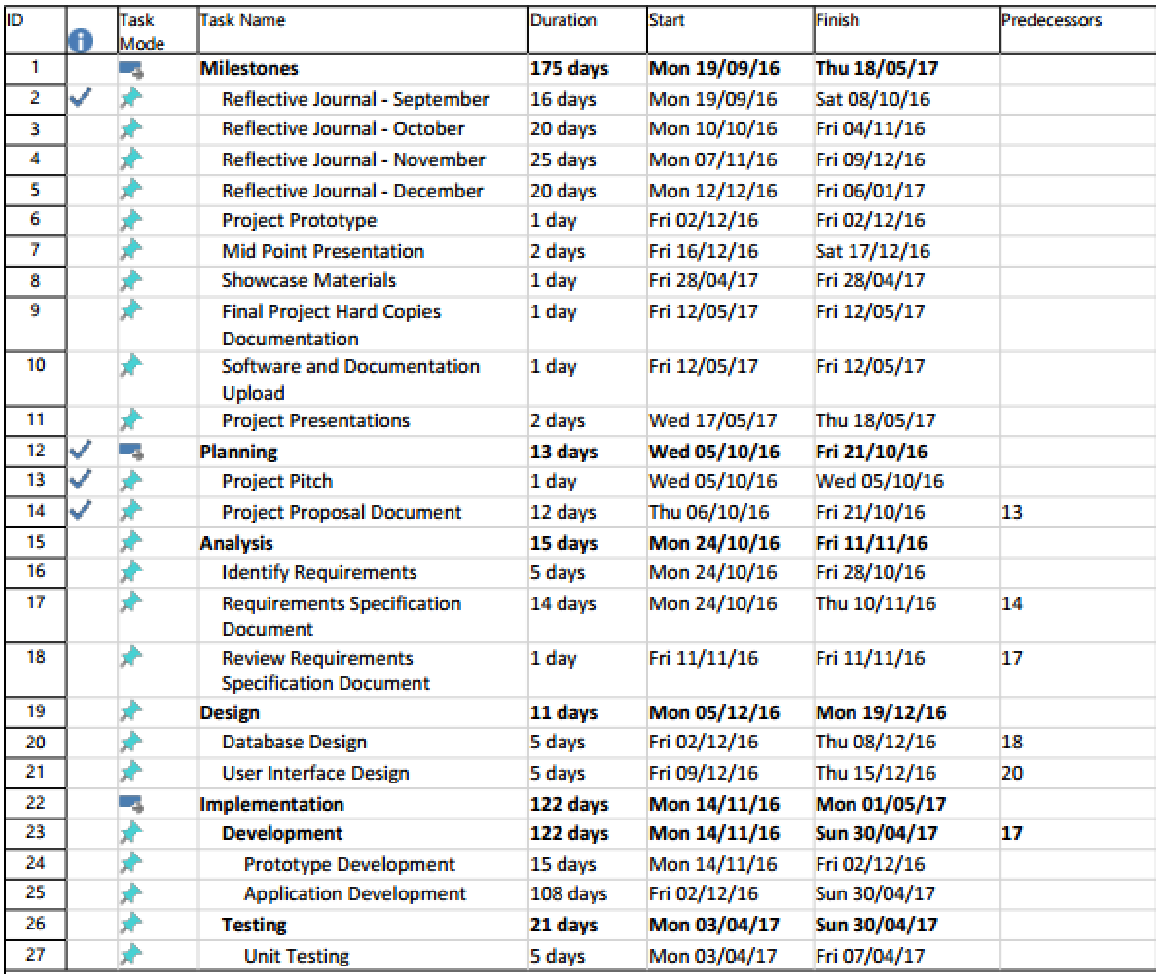
Scriba Stylus – For communicating with the BLE API and coordinating user functions within the application

Android Device – For testing purposes as Android emulator not reliable

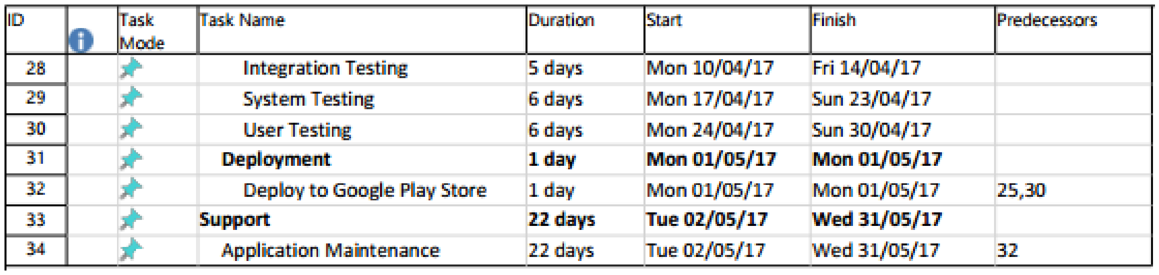
Android Studio – The IDE that will be used for the development of this project will be Android Studio, as I am experienced with this IDE from previous projects.

6.1.5 Project Plan

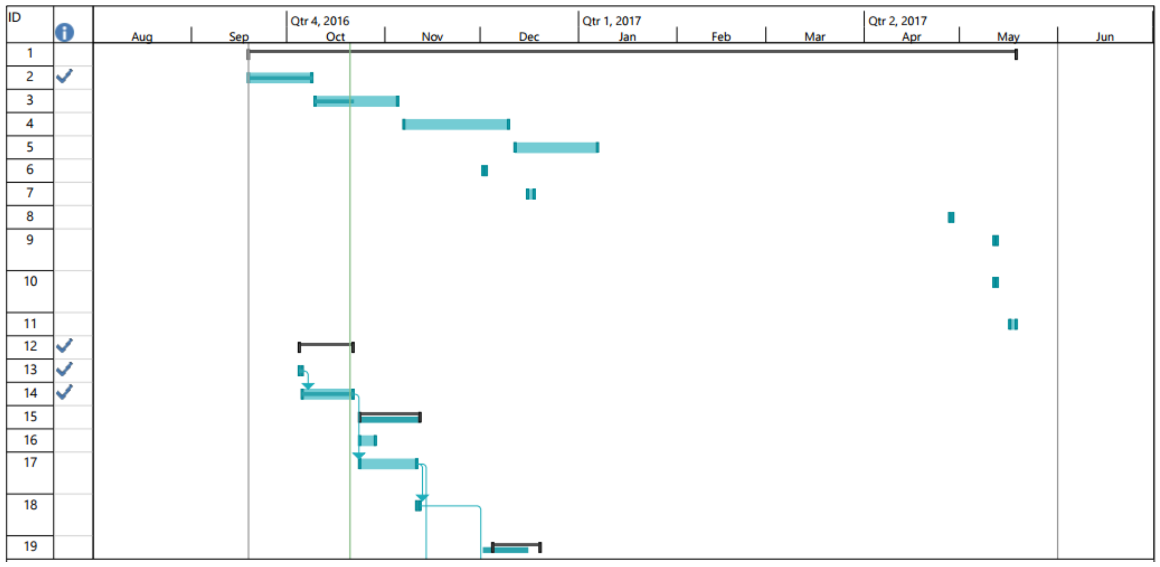
Gantt chart using Microsoft Project with details on implementation steps and timelines

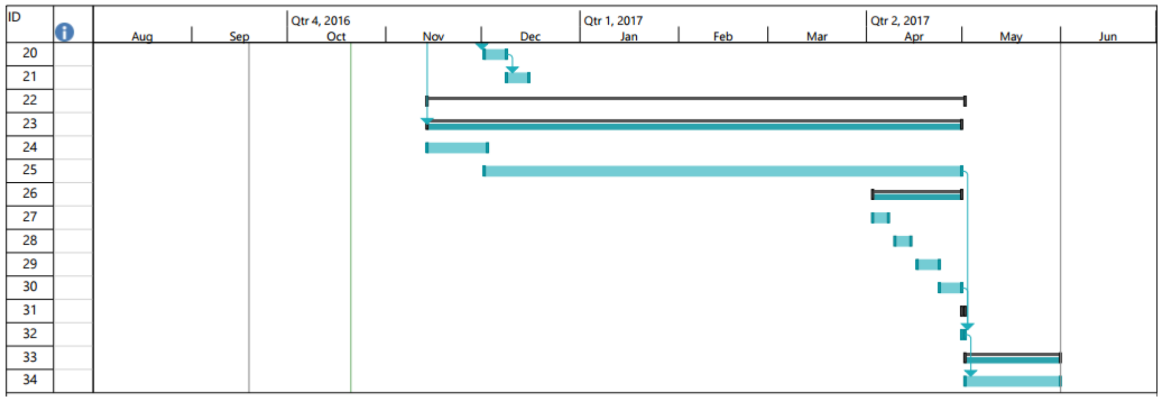


ID	ℹ	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		▬	**Milestones**	**175 days**	**Mon 19/09/16**	**Thu 18/05/17**	
2	✓	📌	Reflective Journal - September	16 days	Mon 19/09/16	Sat 08/10/16	
3		📌	Reflective Journal - October	20 days	Mon 10/10/16	Fri 04/11/16	
4		📌	Reflective Journal - November	25 days	Mon 07/11/16	Fri 09/12/16	
5		📌	Reflective Journal - December	20 days	Mon 12/12/16	Fri 06/01/17	
6		📌	Project Prototype	1 day	Fri 02/12/16	Fri 02/12/16	
7		📌	Mid Point Presentation	2 days	Fri 16/12/16	Sat 17/12/16	
8		📌	Showcase Materials	1 day	Fri 28/04/17	Fri 28/04/17	
9		📌	Final Project Hard Copies Documentation	1 day	Fri 12/05/17	Fri 12/05/17	
10		📌	Software and Documentation Upload	1 day	Fri 12/05/17	Fri 12/05/17	
11		📌	Project Presentations	2 days	Wed 17/05/17	Thu 18/05/17	
12	✓	▬	**Planning**	**13 days**	**Wed 05/10/16**	**Fri 21/10/16**	
13	✓	📌	Project Pitch	1 day	Wed 05/10/16	Wed 05/10/16	
14	✓	📌	Project Proposal Document	12 days	Thu 06/10/16	Fri 21/10/16	13
15		📌	**Analysis**	**15 days**	**Mon 24/10/16**	**Fri 11/11/16**	
16		📌	Identify Requirements	5 days	Mon 24/10/16	Fri 28/10/16	
17		📌	Requirements Specification Document	14 days	Mon 24/10/16	Thu 10/11/16	14
18		📌	Review Requirements Specification Document	1 day	Fri 11/11/16	Fri 11/11/16	17
19		📌	**Design**	**11 days**	**Mon 05/12/16**	**Mon 19/12/16**	
20		📌	Database Design	5 days	Fri 02/12/16	Thu 08/12/16	18
21		📌	User Interface Design	5 days	Fri 09/12/16	Thu 15/12/16	20
22		▬	**Implementation**	**122 days**	**Mon 14/11/16**	**Mon 01/05/17**	
23		📌	**Development**	**122 days**	**Mon 14/11/16**	**Sun 30/04/17**	**17**
24		📌	Prototype Development	15 days	Mon 14/11/16	Fri 02/12/16	
25		📌	Application Development	108 days	Fri 02/12/16	Sun 30/04/17	
26		📌	**Testing**	**21 days**	**Mon 03/04/17**	**Sun 30/04/17**	
27		📌	Unit Testing	5 days	Mon 03/04/17	Fri 07/04/17	



ID	ℹ	Task Mode	Task Name	Duration	Start	Finish	Predecessors
28		📌	Integration Testing	5 days	Mon 10/04/17	Fri 14/04/17	
29		📌	System Testing	6 days	Mon 17/04/17	Sun 23/04/17	
30		📌	User Testing	6 days	Mon 24/04/17	Sun 30/04/17	
31		📌	**Deployment**	**1 day**	**Mon 01/05/17**	**Mon 01/05/17**	
32		📌	Deploy to Google Play Store	1 day	Mon 01/05/17	Mon 01/05/17	25,30
33		📌	**Support**	**22 days**	**Tue 02/05/17**	**Wed 31/05/17**	
34		📌	Application Maintenance	22 days	Tue 02/05/17	Wed 31/05/17	32





6.1.6 Technical Details

Frontend

- Android – Java
- XML – for design and layout
- JSON for representation of the data

Bluetooth Connectivity

- Bluetooth Low Energy (BLE) API for Android

Database Management System

- MySQL

Server Side Language

- PHP

Hosting Server

- X10 Hosting

Libraries

- May configure the Nordic Semiconductor application “nRF Toolbox” into a library to include in the project. This app provides a BLE API for connecting BLE devices. Nordic Semiconductor are the manufacturers of the chipset included in the Scriba stylus.

6.1.7 Evaluation

Android Studio was designed with testing in mind and makes testing simple by providing a JUnit test framework. I will perform unit tests on the functions and methods by using Android’s JUnit test framework.

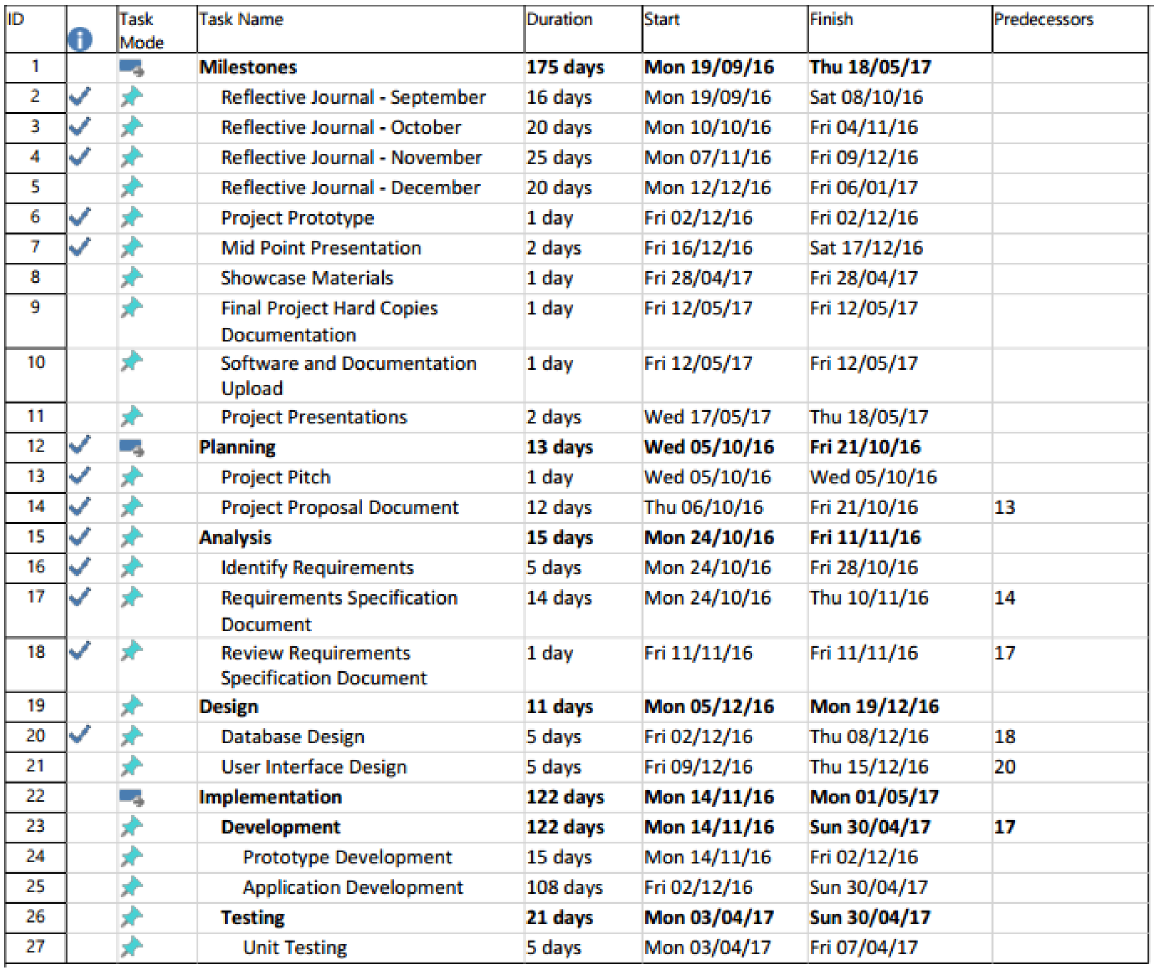
I will perform integration tests to insure different aspects of the application work together correctly, ensure Bluetooth connectivity works and also perform system testing to ensure stable communication between the application and the server, the application and the Scriba stylus and the application as a system.

I will evaluate the system with an end user by testing the application on an Android device and on the Android Emulator provided by Android Studio. I will also deploy the application to the Google Play Store when complete. I will purchase a Google Play Developer Console as this is needed for deploying apps to the Google Play Store. Users can rate the app and leave their reviews of the application.

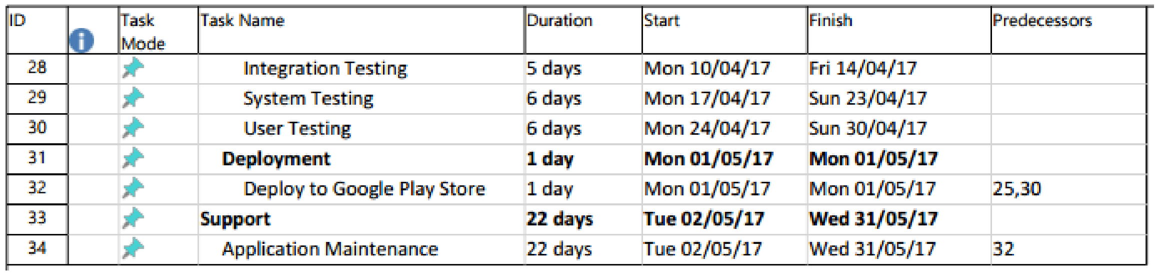
Ian Cunningham 21/10/2016

Signature of student and date

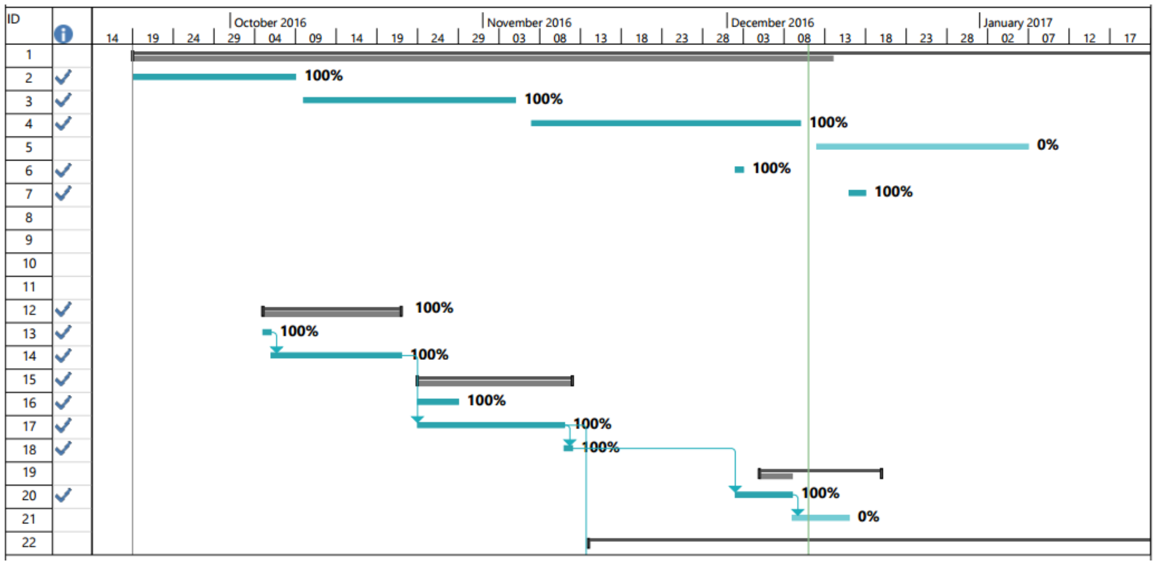
6.2 Project Plan

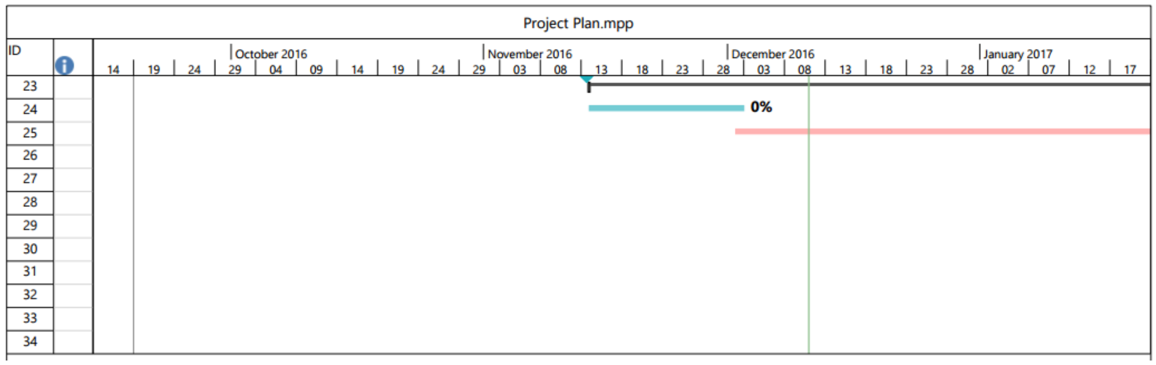


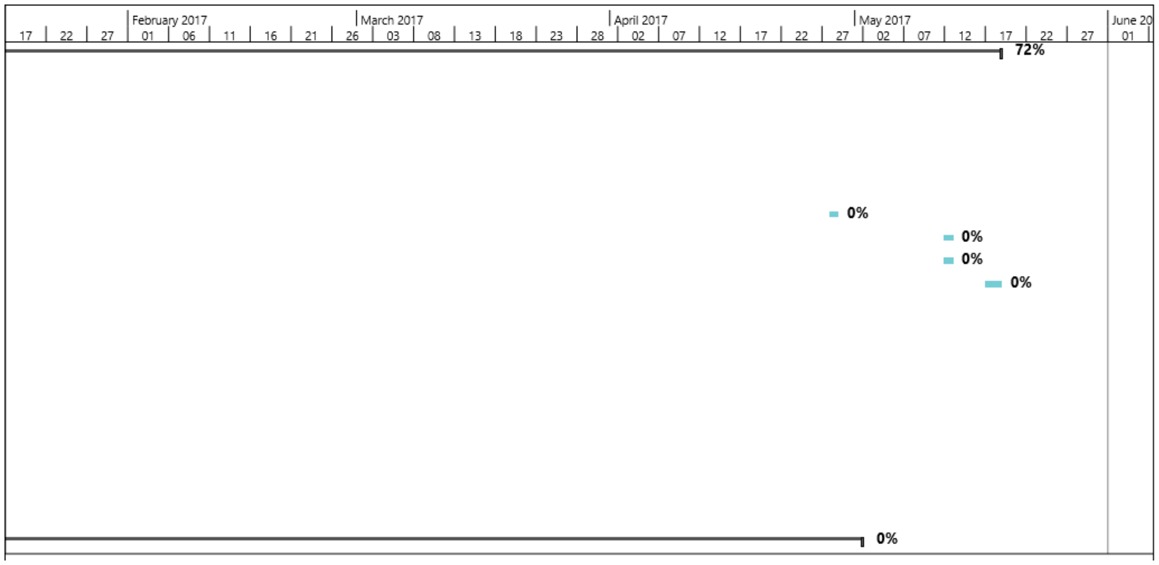
ID	ℹ	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		▬	**Milestones**	**175 days**	**Mon 19/09/16**	**Thu 18/05/17**	
2	✓	📌	Reflective Journal - September	16 days	Mon 19/09/16	Sat 08/10/16	
3	✓	📌	Reflective Journal - October	20 days	Mon 10/10/16	Fri 04/11/16	
4	✓	📌	Reflective Journal - November	25 days	Mon 07/11/16	Fri 09/12/16	
5		📌	Reflective Journal - December	20 days	Mon 12/12/16	Fri 06/01/17	
6	✓	📌	Project Prototype	1 day	Fri 02/12/16	Fri 02/12/16	
7	✓	📌	Mid Point Presentation	2 days	Fri 16/12/16	Sat 17/12/16	
8		📌	Showcase Materials	1 day	Fri 28/04/17	Fri 28/04/17	
9		📌	Final Project Hard Copies Documentation	1 day	Fri 12/05/17	Fri 12/05/17	
10		📌	Software and Documentation Upload	1 day	Fri 12/05/17	Fri 12/05/17	
11		📌	Project Presentations	2 days	Wed 17/05/17	Thu 18/05/17	
12	✓	▬	**Planning**	**13 days**	**Wed 05/10/16**	**Fri 21/10/16**	
13	✓	📌	Project Pitch	1 day	Wed 05/10/16	Wed 05/10/16	
14	✓	📌	Project Proposal Document	12 days	Thu 06/10/16	Fri 21/10/16	13
15	✓	📌	**Analysis**	**15 days**	**Mon 24/10/16**	**Fri 11/11/16**	
16	✓	📌	Identify Requirements	5 days	Mon 24/10/16	Fri 28/10/16	
17	✓	📌	Requirements Specification Document	14 days	Mon 24/10/16	Thu 10/11/16	14
18	✓	📌	Review Requirements Specification Document	1 day	Fri 11/11/16	Fri 11/11/16	17
19		📌	**Design**	**11 days**	**Mon 05/12/16**	**Mon 19/12/16**	
20	✓	📌	Database Design	5 days	Fri 02/12/16	Thu 08/12/16	18
21		📌	User Interface Design	5 days	Fri 09/12/16	Thu 15/12/16	20
22		▬	**Implementation**	**122 days**	**Mon 14/11/16**	**Mon 01/05/17**	
23		📌	**Development**	**122 days**	**Mon 14/11/16**	**Sun 30/04/17**	**17**
24		📌	Prototype Development	15 days	Mon 14/11/16	Fri 02/12/16	
25		📌	Application Development	108 days	Fri 02/12/16	Sun 30/04/17	
26		📌	**Testing**	**21 days**	**Mon 03/04/17**	**Sun 30/04/17**	
27		📌	Unit Testing	5 days	Mon 03/04/17	Fri 07/04/17	

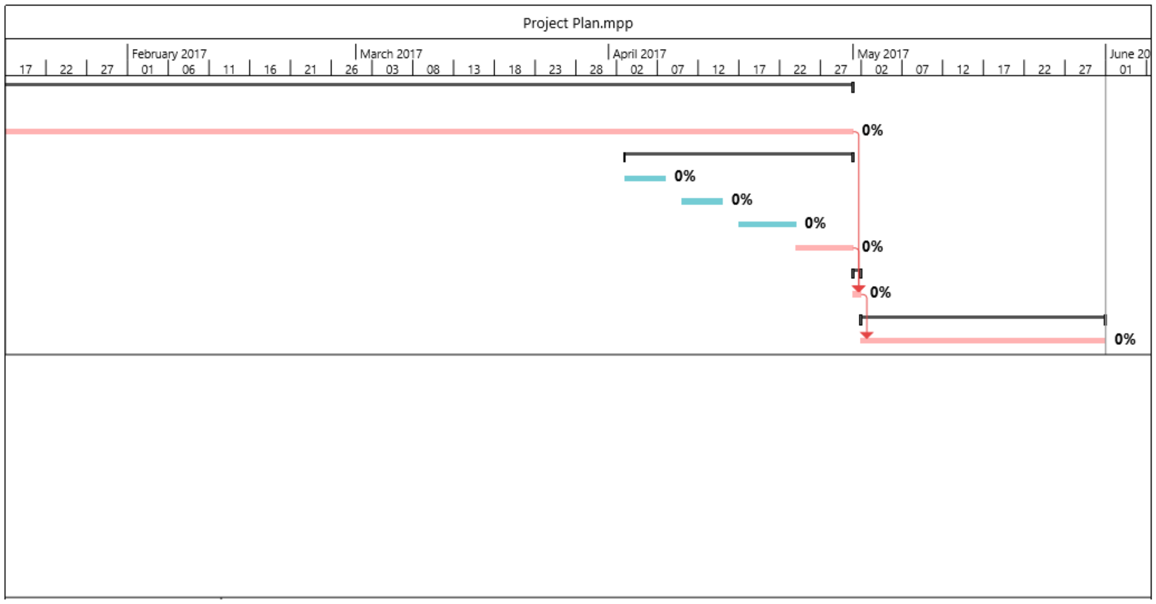


ID	ℹ	Task Mode	Task Name	Duration	Start	Finish	Predecessors
28		📌	Integration Testing	5 days	Mon 10/04/17	Fri 14/04/17	
29		📌	System Testing	6 days	Mon 17/04/17	Sun 23/04/17	
30		📌	User Testing	6 days	Mon 24/04/17	Sun 30/04/17	
31		📌	**Deployment**	**1 day**	**Mon 01/05/17**	**Mon 01/05/17**	
32		📌	Deploy to Google Play Store	1 day	Mon 01/05/17	Mon 01/05/17	25,30
33		📌	**Support**	**22 days**	**Tue 02/05/17**	**Wed 31/05/17**	
34		📌	Application Maintenance	22 days	Tue 02/05/17	Wed 31/05/17	32


ID
October 2016
November 2016
December 2016
January 2017
100%
0%


Project Plan.mpp
ID
October 2016
November 2016
December 2016
January 2017
0%


February 2017
March 2017
April 2017
May 2017
June 20
72%
0%



Project Plan.mpp
February 2017
March 2017
April 2017
May 2017
June 20
17 22 27 01 06 11 16 21 26 03 08 13 18 23 28 02 07 12 17 22 27 02 07 12 17 22 27 01
0%
0%
0%
0%
0%
0%
0%

6.3 Monthly Journals

6.3.1 Reflective Journal (September)

Reflective Journal

Student name: Ian Cunningham

Programme: BSHCSD4 – Software Development Stream

Month: September 2016

My Achievements

After countless time of thinking about my 4th year software project I now think I have an idea suitable for my 4th year project. This month, I have been involved in a lot research in relation to my project idea to iron out any issues that may arise further down the line. From this research, I have concluded that my project is a feasible idea to achieve and I am confident of achieving an efficient end result. My contributions to the project this month included researching the various techniques and technologies that would best suit the needs of my project.

My Reflection

I feel that this project is definitely achievable and should be attempted as it provides functionality seen in disparate applications to create a new application. The project will be an Android application that will serve as a study tool for students. The app is aimed at students using Android devices so that in class they can bring device and Scriba and record important notes from class as over marks to the presentation material. Scriba is a new stylus product and the app will allow the user to annotate the information received in a manner that will make it easier for them to recall important sections or clarify their own understanding of the content. The app will store presentation or study documents received on a server, in the form of files such as docx, pdf, ppt etc. The app will integrate the Scriba stylus to allow them to quickly interact with the learning content. Depending on what mode the

Scriba is in users will be able draw, highlight, etc. When in highlight or draw mode the content that is selected will be saved as a note to accompany the presentation.

I feel this will be challenging as it involves developing an application that combines several different functionalities such as saving documents to a server, displaying documents from a server, drawing/highlighting over the documents, generating notes from the documents, storing the generated data, embedding framework software and Bluetooth connectivity. This project is different to others in the same area as the app will integrate the functionality seen in different kinds of applications to create a new learning tool. The app also takes advantage of a new stylus product which provides the user quick access to functionality of the app, unlike other offerings which require drop down menus and settings.

Intended Changes

Next month, after the project has been pitched to a panel and if the project has been accepted I will immediately get to work on my Project Proposal Document and Requirements Specification Document. I will also aim to do a bit more research in relation to the application and get started on a prototype.

Supervisor Meetings

I am currently awaiting confirmation of whether my project idea has been accepted or rejected and will be notified in the next few days. When the decision is made, I will immediately get in contact with an Academic Supervisor.

Date of Meeting: N/A

Items discussed: N/A

Action Items: N/A

6.3.2 Reflective Journal (October)

Reflective Journal

Student name: Ian Cunningham

Programme: BSHCSD4 – Software Development Stream

Month: October 2016

My Achievements

My project idea was pitched to a panel and was accepted with revisions. I then started working on Project Proposal and this month I have successfully completed my Project Proposal and have also been assigned my project Supervisor. My Supervisor for the project will be Mr Paul Stynes. I have been involved in a bit of research to refresh the mind regarding Requirements Specification documents. I have now begun writing my requirements specification document and the process of gathering the requirements of the project. I have been in consultation with David Craig, CEO of Dublin Design Studio to gather the expected user requirements of the project. My contributions to the project this month included completing and uploading the Project Proposal document, gathering the requirements of the project and getting started on writing the requirements specification document. I have also identified my functional requirements and non-functional requirements, therefore completing my use case diagram.

My Reflection

I feel that it made sense and was helpful to refresh the mind and do a bit of research on Requirement Specification documents. It was especially helpful to refresh the mind about UML diagrams such as class diagrams and use case diagrams. I also found useful information on writing use cases effectively including their main flow, alternate flow and Exceptional flow.

I felt the meeting with David Craig helped iron out any issues or conflicts we had about the project. This meeting also helped to identify the user requirements, functional requirements and nonfunctional requirements for the application.

I feel that creating a project plan and following the Project Plan Gantt Chart included in the Project Proposal is working well and is helping me keep on track with my project deadlines.

Intended Changes

Next month, I will continue to complete the requirements specification document including GUI mock-ups and the system architecture diagram. I will also aim to do research regarding different technologies and API'S available to use for the project. I also hope to get started on a prototype of the project.

Supervisor Meetings

When I was assigned my project Supervisor, I immediately got in contact with Paul Stynes to arrange a meeting. Paul responded with a proposed meeting for two days later which was pleasant as it meant we could start talking about the project early.

Date of Meeting: 24th October 2016

Items discussed:

- Introduction to project/What type of application it will be
- How the project will work/what it will do?
- Gathering requirements

Action Items:

- Identify Requirements
- Create use case diagram

- Continue Requirements Specification document

6.3.3 Reflective Journal (November)

Reflective Journal

Student name: Ian Cunningham

Programme: BSHCSD4 – Software Development Stream

Month: November 2016

My Achievements

This month I have started working on my technical report and have also begun the development of a prototype for the Mid-Point Presentation on the 19th December 2016. I have also continued with previous research for the project, while researching new topics for the project. These new topics include researching ways in which to the implement the functionality of generating a quiz based on the users stored notes using natural language processing. This is a new area to me and will obviously take some time to gain a proper understanding of the area but I will obviously give it my best shot. The topics of interest for research include natural language processing, OWL ontology and an online OWL tutorial called the Pizza tutorial. This month I successfully completed and uploaded the requirements specification document.

My Reflection

I feel that the workload in other modules with Continuous Assessments and projects has slightly held backed the focus on the project as I would of liken more time to focus on this project. I have been successful in getting each deliverable for this project uploaded on time, even with the workload from the other modules.

I feel that if I can get the functionality of generating quizzes based on the users notes working, this would be a fantastic achievement and a very unique feature to my project. Similar applications in this area do not offer this feature.

Intended Changes

Next month, I will continue to complete the Technical Report for upload. I will also continue to develop my prototype and prepare for the Mid-Point Presentation. I also aim to do research regarding the new functions pitched and continue to plan ahead for the project. I will also look into the other functionality ideas for the project such as creating mind maps and study plans.

Supervisor Meetings

This month I had two meetings with my project Supervisor. In the first meeting we mainly discussed about new functionalities that could be added to the project such as the generation of quiz's using natural language processing. We also discussed and I was given advice on applying the finishing touches to the Requirements Specification document.

During the second meeting we mainly discussed the generating quiz functionality and was advised on a few items to research and a tutorial to complete to gain an understanding of the technologies needed to fulfil this functionality.

Date of Meeting: 2nd November 2016

14th November 2016

Items discussed:

- Completing the Requirements Specification
- New functionalities for the project

Action Items:

- Research technologies behind natural language processing
- Start Technical Report and prototype
- Complete online Pizza tutorial for to try gain understanding of what is needed for the generate quiz functionality.

6.3.4 Reflective Journal (December)

Reflective Journal

Student name: Ian Cunningham

Programme: BSHCSD4 – Software Development Stream

Month: December 2016

My Achievements

This month I have completed and uploaded the Technical Report. I have also created a prototype of the project consisting of a few project functions. This was necessary for demonstration purposes for the Mid-Point Presentation. On the 19th December 2016, I presented my project Mid-Pont Presentation and demonstrated the functions of my project prototype to my project supervisor Paul Stynes and Mohammad Iqbal.

This month I also planned to do more research related to the project, but due to other module projects and uploads, was unable to perform as research as I would of liken to. I will now start the development of the final project by extending my current project prototype.

My Reflection

I thought that the Mid-Point Presentation went well and I was pleased with feedback given. I am also pleased with the grade I have received for the Mid-Point

Presentation. I feel that the project is going according to the project plan, which is also contributing to keeping on track with all project related uploads.

Intended Changes

Next month, I will continue to do more research for certain functions for the application. I will also start the development of the final project by creating the database needed for the application and implement the overall design and style of the application. As the project is a mobile application, design is also an important factor to consider and it is important to provide users with a user-friendly and easy to navigate application. I will also start to implement the function of connecting to and interaction with the Scriba stylus within the application as this is the next step in functionality to extend from the current prototype.

Supervisor Meetings

This month there was two meetings scheduled with my project supervisor, but for some reason one was cancelled and then I unfortunately could not make the other meeting. In January, when the exams are finished and the new semester is beginning I will organize a meeting with my supervisor. I will also seek help from my supervisor regarding the function of generating quiz using natural language processing and OWL ontology.

Date of Meeting: 5th December 2016 - Cancelled

12th December 2016 – Not Present

Items discussed:

- Meeting Cancelled
- Not Present

Action Items:

- Research technologies behind natural language processing
- Complete online Pizza tutorial for to try gain understanding of what is needed for the generate quiz functionality
- Create Database for application
- Design Graphical User Interface
- Extend current implemented functions for final project
- Implement documented functions, such as interaction with the Scriba stylus and Create and View Notes

6.3.5 Reflective Journal (January)

Reflective Journal

Student name: Ian Cunningham

Programme: BSHCSD4 – Software Development Stream

Month: January 2017

My Achievements

This month I have been successful in designing the user interface of the system and implementing the database for the application. The database is a MySQL database which will communicate with the application with the assistance of PHP scripts for server side code. The database currently consists of three tables User, File and Note but as the project progresses, there will be a need to add tables to store the generated quiz's, the created mind maps and study plan's. I have also begun the process of writing my project analysis and design document, which will describe and define the necessary information required to effectively provide a description of the architecture and design of the system for the purpose of the software development process.

This month I have also done some research regarding Ontologies and web semantics. I was advised to have a look at the Pizza tutorial by my project supervisor, which was available as a sample project with a downloadable ontology tool called “Protégé”. I have downloaded this tool and am currently investigating the provided sample ontologies.

My Reflection

I feel that the project is going well and this week have a made a few breakthroughs in the project in terms of functionality and file conversion. The tools used in the project such as Google’s GView Embed Tool will not be needed in the project anymore for the purpose of viewing files. When using this tool, it would allow the viewing of a document but did not support Android’s native text selection feature in the way that suffice for the project. I have been successful in finding an alternative solution that supports the sort of functionality that is needed for the project. I have found an API called Cloud Convert API which converts documents into many different file formats. In the project, when the user selects a file to view, a request is sent to the API with parameters specifying the current file type and the file type you want to convert to. In my case, all files will be converted to html files and displayed in an Android Web View.

Intended Changes

Next month, I will continue writing my project analysis and design document and aim to have it completed by the 24th February. I will also create a library for the project to call functions necessary to communicate with the Scriba stylus. The library is an application provided by Nordic Semiconductor, the creators of the Scriba’s SoC (System on a Chip. I will transform this android application into a library that can be used as a dependency in my project. I also aim to start creating my ontology for the Quiz function.

Supervisor Meetings

This month there were no meetings with the supervisor due to exams and Semester 2 beginning on the 23rd January 2017. I have been in touch with my

Supervisor and on the 6th February, we had a meeting to provide feedback from the mid-point presentation and explain exactly where I gained and lost marks. We also discussed how the project was progressing. My supervisor has also arranged a meeting for the 13th February 2017.

Date of Meeting: 6th February 2017

13th February 2017 - Arranged

Items discussed:

- Individual feedback from mid-point presentation
- Ontologies and Web Semantics, the best approach

Action Items:

- Further research on Ontologies and Web Semantics
- Extend Database for application
- Finish Project Analysis and Design document
- Integrate documented functions, such as interaction with the Scriba stylus
- Create Library for the project

6.3.6 Reflective Journal (February)

Reflective Journal

Student name: Ian Cunningham

Programme: BSHCSD4 – Software Development Stream

Month: February 2017

My Achievements

This month I have been successful in uploading the project analysis and design specification. I have also fully implemented the database for the application with all the necessary tables and the relationships between these tables have been configured. The database consists of six tables, User, File, Note, Quiz, Mind map and Study Plan. I have also created a library for the project to provide functions that are necessary for interaction with the Scriba stylus. The library was created using an android application provided by Nordic Semiconductor, the creators of the Scriba's SoC (System on a Chip). The application was transformed into a library using tools provided in the Android Studio IDE. This enabled me to add the library as a dependency in my project.

This month I have also done more research regarding Ontologies and web semantics but not as much as would have liked due to other projects. I was advised to have a look at web sematic dictionaries by my project supervisor, which could help with putting meaning to the students stored notes within application. I have also started researching this topic.

My Reflection

I feel that project is going according-to the plan and I am coming across new options in terms of technologies and platforms for the project. I am currently considering integrating Firebase into the project to provide the database and storage facility. Firebase is a platform mobile and web applications that provides many tools for building high-quality applications. Firebase provides many services such as Authentication, Real-time Database, Storage, Cloud Messaging, Hosting and many more.

Intended Changes

Next month, I aim to implement functionality with the Scriba stylus by using the library created and develop the necessary Scriba functions that will be contained within the application. Such features include highlighting document text which will then be stored automatically as a note and drawing over document content which will give the user the option to store an image of the current page of the document that has been drawn over.

Supervisor Meetings

This month there was weekly meetings with my project supervisor. Each week I provided details about my current standing in relation to the project and was given set goals to reach for each week. New ideas were devised such as using a web dictionary to apply meaning to the text saved as a note. I have also been advised to keep investigating web semantics and ontologies for the generation of quizzes as this would be a unique feature not seen in a study tool application.

Date of Meeting: 6th February 2017

13th February 2017

20th February 2017

27th February 2017

Items discussed:

- Individual feedback from mid-point presentation
- Ontologies and Web Semantics, the best approach
- Progression of project
- Web dictionary's
- Consider implementing functionality with Firebase

Action Items:

- Further research on Ontologies and Web Semantics
- Researching web dictionary's
- Research Firebase for Android
- Implement functionality with the Scriba library for the project
- Develop Scriba in-app functions

6.3.7 Reflective Journal (March)

Reflective Journal

Student name: Ian Cunningham

Programme: BSHCSD4 – Software Development Stream

Month: March 2017

My Achievements

This month I have explored implementing Firebase for Android into my application for authentication, cloud storage and database. This was going well until I encountered a problem with rendering the files stored on Firebase cloud storage. When receiving the URL of the file intended to be viewed there is an issue with displaying the document with the Cloud Convert API needed for converting the files to html files to enable the files to be viewable within an Android WebView widget. I could not find a solution to this problem as most Firebase tutorials provided have only dealt with storing and retrieving images specifically on the Firebase cloud storage and have found it hard to find an example of storing and retrieving documents on Firebase cloud storage. I have therefore reverted to my original storage and database source using XAMPP, MySQL and PHP. XAMPP server will be used during the development process and then deployed to a live hosting service when project is complete.

This month I have started developing the AI chat bot within the application using API.ai. API.ai is a platform for building conversational interfaces for applications. The chat bot will be trained to react to questions from the user, specifically trained to react to questions about computer science. I have also started developing the quiz functionality where the user can create quiz questions for studying purposes. I have also started writing the final technical report for the final project upload.

My Reflection

I feel the project is going well but was slowed down with the attempt to integrate functionality with Firebase cloud storage and database. As mentioned above the issue with displaying the URL of the file from Firebase storage within an Android WebView widget. I feel that the original source for file storage and data storage works better for this project and seems better in performance terms and the rendering of the documents.

Intended Changes

From this point until the final upload of the project I plan to implement all remaining functionalities and carry out my testing of the project. Several testing methods will be carried such as unit testing, integration testing and customer testing. When my final technical report is complete I intend on having the document proof-read by my project supervisor.

Supervisor Meetings

This month there was weekly meetings with my project supervisor. Each week I provided details about my current standing in relation to the project and was given set goals to reach for each week. New ideas were devised such as using a AI chat bot within the application rather than the web semantic quiz generation as there were many issues encountered trying to implement the automatic quiz generation as the area of web semantics was new to me. I have also been advised to get a start on the final technical report for the final project upload.

Date of Meeting: 6th March 2017

13th March 2017

20th March 2017

27th March 2017

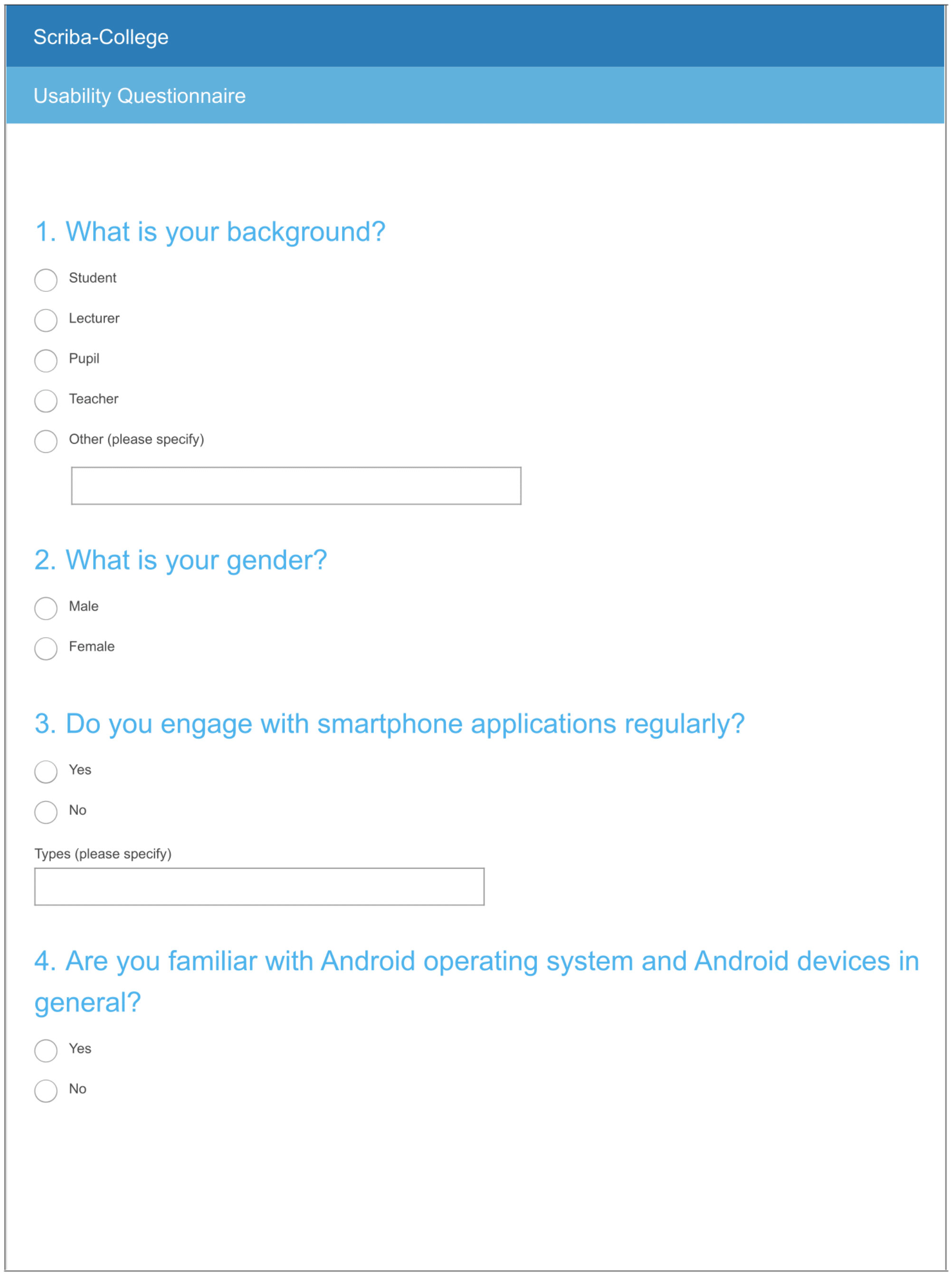
Items discussed:

- Replacing web semantic quiz with AI chat bot embedded within app
- Implement Firebase for Android
- Progression of project
- Reverting to original source of file and data storage
- Getting started on the final technical report

Action Items:

- Implement AI chat bot to embed within application
- Functionality to enable user to create quizzes with questions created by them
- Writing the final technical report
- Develop Scriba in-app functions
- Other functionality such as creating mind maps and study plans

6.4 Other Material Used



Scriba-College

Usability Questionnaire

1. What is your background?

◯ Student

◯ Lecturer

◯ Pupil

◯ Teacher

◯ Other (please specify)

2. What is your gender?

◯ Male

◯ Female

3. Do you engage with smartphone applications regularly?

◯ Yes

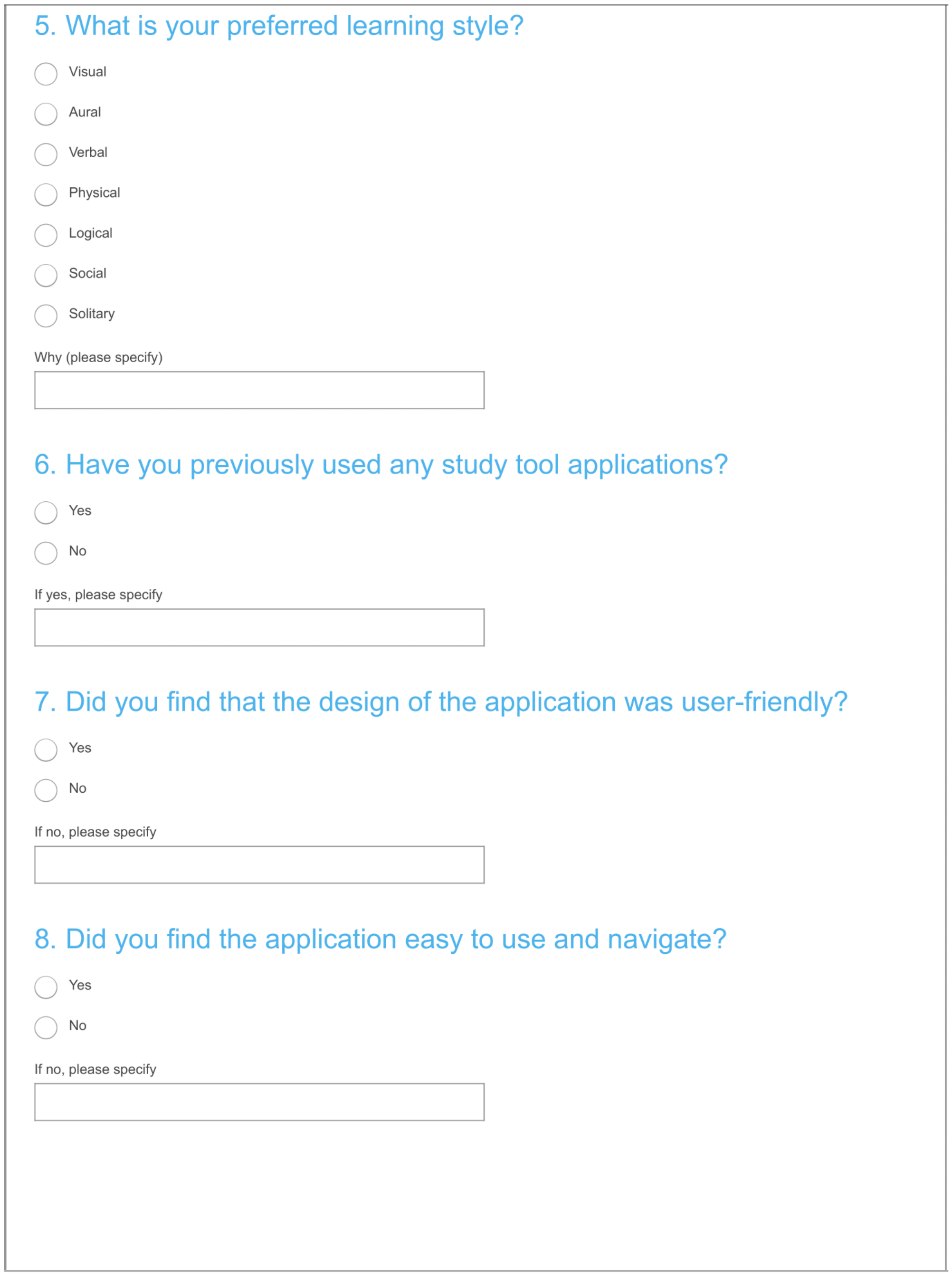
◯ No

Types (please specify)

4. Are you familiar with Android operating system and Android devices in general?

◯ Yes

◯ No



5. What is your preferred learning style?

- Visual
- Aural
- Verbal
- Physical
- Logical
- Social
- Solitary

Why (please specify)

6. Have you previously used any study tool applications?

- Yes
- No

If yes, please specify

7. Did you find that the design of the application was user-friendly?

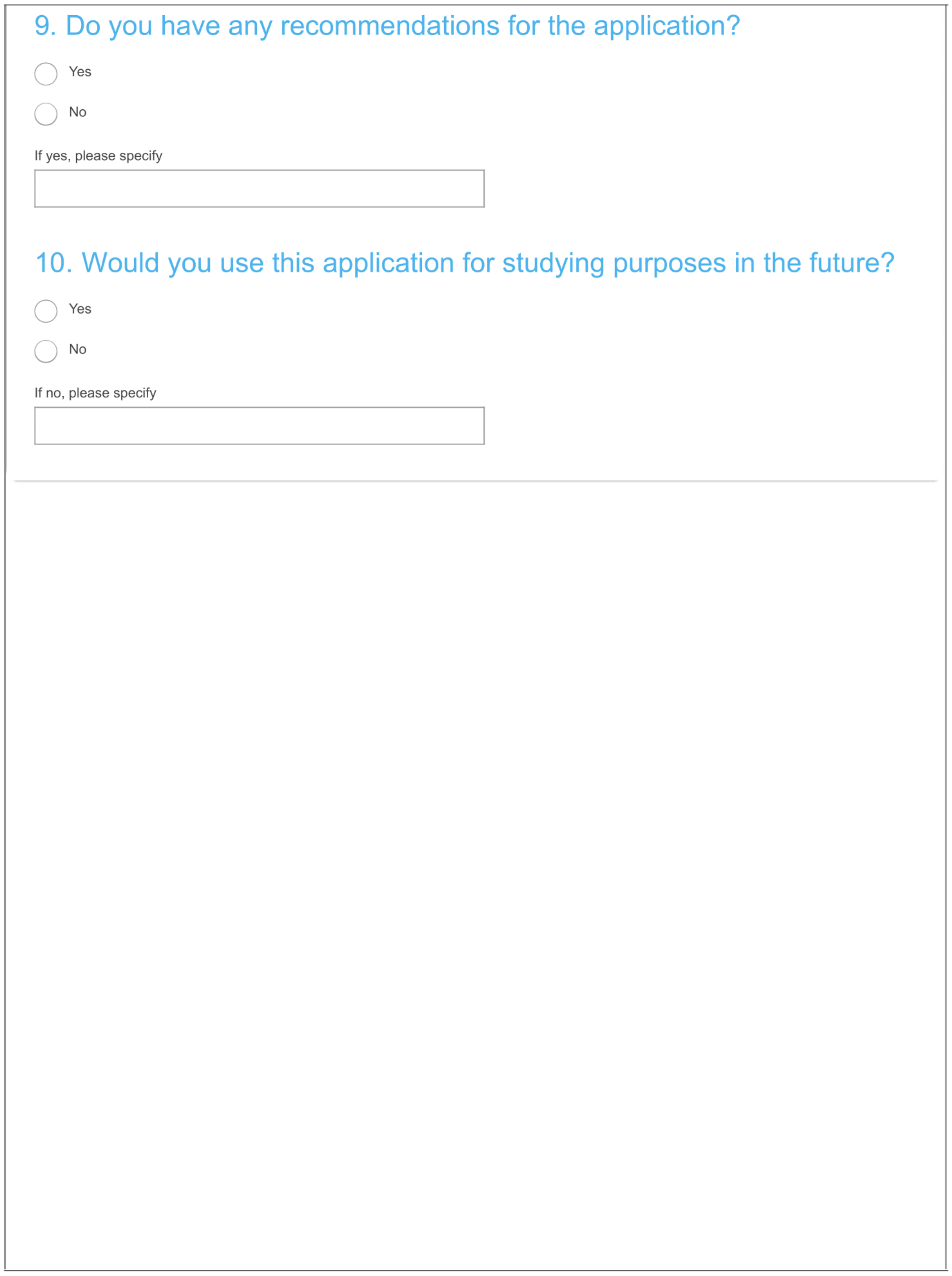
- Yes
- No

If no, please specify

8. Did you find the application easy to use and navigate?

- Yes
- No

If no, please specify



9. Do you have any recommendations for the application?

◯ Yes

◯ No

If yes, please specify

10. Would you use this application for studying purposes in the future?

◯ Yes

◯ No

If no, please specify

6.5 Document References

The following table summarizes the documents referenced within this document.

Document Name and Version	Description	Location
Material Design Specification	*Guidelines for using Material Design within your Android applications*	https://material.io/guidelines/

6.6 Key Terms

The following table provides definitions for terms relevant to this document.

Term	**Definition**
Scriba	*The next generation stylus created by Dublin Design Studio Ltd.*
API	*Application Programming Interface. Cloud Converter API for converting documents to html files for viewing in a web view.*
BLE	*Bluetooth Low Energy that provides communication with the Scriba stylus.*
REST	*Representational State Transfer. A RESTful API enables interoperability and communication between client and server.*
MVC	*Model-View-Controller is a design pattern used in software development.*
HTTPS	*Hypertext Transfer Protocol Secure is a protocol that enables communication over the internet with an encrypted connection.*
HTTP	*Hypertext Transfer Protocol is a protocol that enables communication over the internet.*
SSL	*Secure Sockets Layer is the standard security technology that provides an encrypted connection between client and server.*
SHA	*Secure Hash Algorithm 256 is a cryptographic hash function that generates a unique 256-bit hash.*