Super Tutor

Final Year Project Document

Emile Bronkhorst [W1429295]

Mentor: Alexander Bolotov

Mobile learning with elements of Automation

App Name:	Super Tutor
Project keywords:	Automation, Mobile Learning
Platform:	Android Devices
Cross-platform integration:	YES (Core structures)
Mentor:	Alexander Bolotov

Abstract

Super tutor is a mobile application that will enable users to discover their learning style through a quick 10 question quiz (developed using the most popular Educational psychology methods). Users will then be able to browse a catalogue of topics uploaded to the application by Content Managers and have direct access to a personalised, tailor made course (dependent on learning preference, Super Tutor will organise the topic contents suited to your learning style). This document will detail how the problem was discovered, how the problem currently is in existence, what the possible solution is and the various other aspects of the problem domain. In addition, it will cover the various aspects of the Software Development Lifecycle (Requirements elicitation, Design, Implementation, Testing) in depth and detail which direction was taken for this particular project.

Acknowledgements

I would like to take this time to given special thanks to Dr. Alexander Bolotov who has given a lot of his time to ensure that the project has been driven in the right direction and to a very high standard. I thoroughly appreciate all the time and effort you have put into this project and guiding it in the correct direction and couldn't have been more pleased with having you as a mentor to oversee this project.

Apache License

Libraries used within the project:

- JSON simple (Cross platform integration)
- Apache Commons HTTP Client (File uploading)
- Google Cloud Messaging server (gcms) – GCM Push notifications
- Java Mail API (Server sided emails)

Table of Contents

Chapter 1: Project overview & Problem statement

1.1 Problem statement

Interactive learning is becoming a large area of interest in the technological world today. Developers are consistently looking for new ways to let Artificial intelligence take over every aspect of your life, although there have been many prestigious developments in various fields, the field of Education has yet to see an 'Intelligent application'.

Super Tutor is a mobile learning tool which aims to provide you with a more 'personal' experience when it comes to education. Every individual has a learning preference, some learn better through seeing graphs, hearing audio, watching videos etc. Super Tutor attempts to take all these learning styles and combine them into an application that can tailor content to your preference.

1.2 Project overview

What does this mean? Our Artificial Intelligence algorithm will take into consideration your learning style when you browse the many topics available on the platform. Once you download a topic, the AI will attempt to 're-organise' the contents of the topic to match how you learn best. Here's an example, lets assume you wanted to study the basics of Object orientated programming, instead of reading boring text. If you are of Visual preference, the AI will organise the Graphs, Diagrams and Images first with the appropriate explanations for each following. This way you get to see things in action before even getting to the textual representation.

Project components

The software will be split into 3 portions:

- A web server (to pass information between the various platforms, store user data and manage content)
- Desktop application (used by Content managers to upload/manage topics available)
- Mobile application – The platform in which users will have access to the content available on the web server as well as house the Artificial intelligence algorithm

1.2 Project objectives

Super tutor will aim to provide users with a rich, interactive user experience engaging the user at every point of call. An intuitive user interface can cater for these objectives however satisfying them will require consideration of the following components:

- Maintainability – The platform must be maintainable from a central location (i.e. the web server)
- Performance – Due to the nature of the application, the AI must perform efficiently
- Interactivity – The application must consist of an engaging task in each activity within the application

Other aspects to take into consideration during development:

- User engagement – The application must be able to leave users wanting to use it more with every use
- Extendibility – Ensure that the application is extendible with minimal user interaction required (i.e. no need for update downloads)
- Guided dialogs – Activities should state the purpose intended before the user interacts with the appropriate activity

1.3 Definitions

- API – Application Programming Interface
- ST – Super Tutor
- AI – Artificial intelligence
- CM – Content Manager
- UI – User Interface
- SRS – Software Requirements Specification
- DDS – Design Document Specification

Chapter 2: Software Methodology

2.1 Prologue

Software methodologies have been developing over the centuries with the earliest models starting as early as 1950. It was in 1970 that the first software methodology became popular (Waterfall model) and since this decade, more have arisen. Software methodologies have changed the way that software has developed over the years providing a clearer management and planning structure throughout its various phases. This section will look at the many software methodologies available and detail some of their advantages and disadvantages. Finally, it will look at which methodology has been chosen for development of Super Tutor and the reasoning behind this decision.

Sub section 2: Developmental Methodologies

2.2.1 Waterfall model

The Waterfall model was developed in 1970 by Winston W. Royce and was the first developmental methodology to arise. This model compared to its rivals is one of the easiest to implement and use and consists of a project review at the end of each phase to determine whether the project should be discarded or development should be continued.

The Waterfall method has been proven to be a solid methodology when the project requirements are well known, technology is understood and the project size is not that large. Consequently, the greater the size of the project the more things can go wrong.

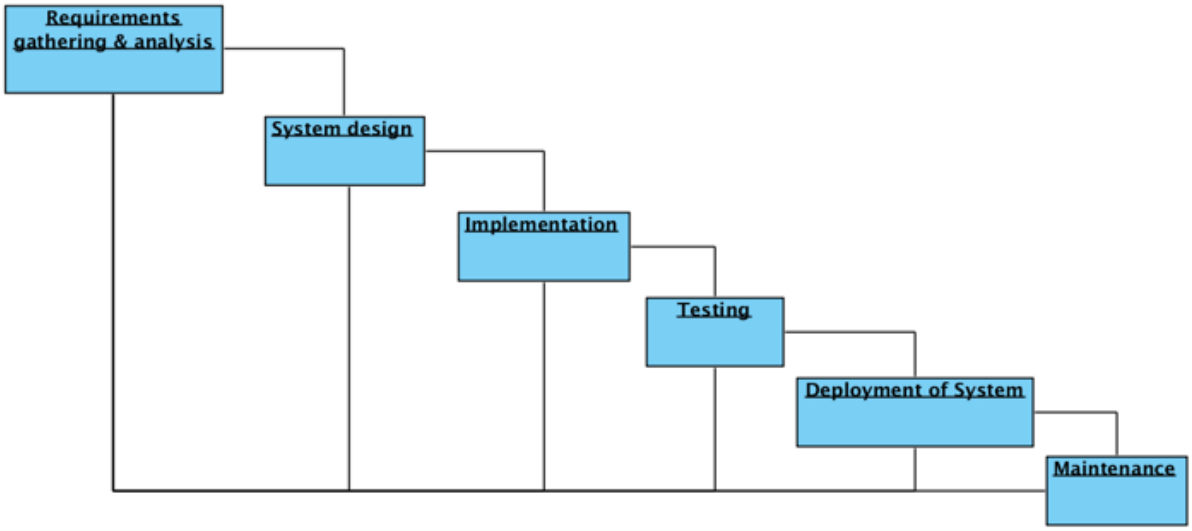


Figure 1.1 Waterfall model development phases

Fig. 1.1 shows the Waterfall development phases. At the end of each respective phase there is a project review to determine whether the project development should continue, or terminate and be re-assessed from the Requirements gathering phase (stage 1). This brings rise to some issues such as increasing the development time and thus giving rise to the issue of not meeting project deadlines.

Advantages of the Waterfall model:

- Simple and easy to use
- Phases are completed in sequence (they do not execute at the same time)
- Project reviews at the end of each phase
- Works well for small projects where requirements are very clearly

Disadvantages of Waterfall model:

- High amounts of risk
- If the project hits the testing phase, it is difficult to go back and change a feature that was not well defined in the requirements analysis
- Not suitable for object orientated projects (very lengthy projects)
- Not suitable for projects where the requirements may evolve

2.2.2 Incremental development (Rapid application development)

Incremental development is a software methodology that works on the premise of the software being split into smaller “portions” or increments. These portions are then pushed through the Software development life cycle (from development through to testing) till the increment is ready to be deployed.

This methodology allows for software to be developed ‘asynchronously’, meaning that although the application is being split into several portions and distributed, these portions are being worked on simultaneously meaning that usable features are produced early on in the software life cycle.

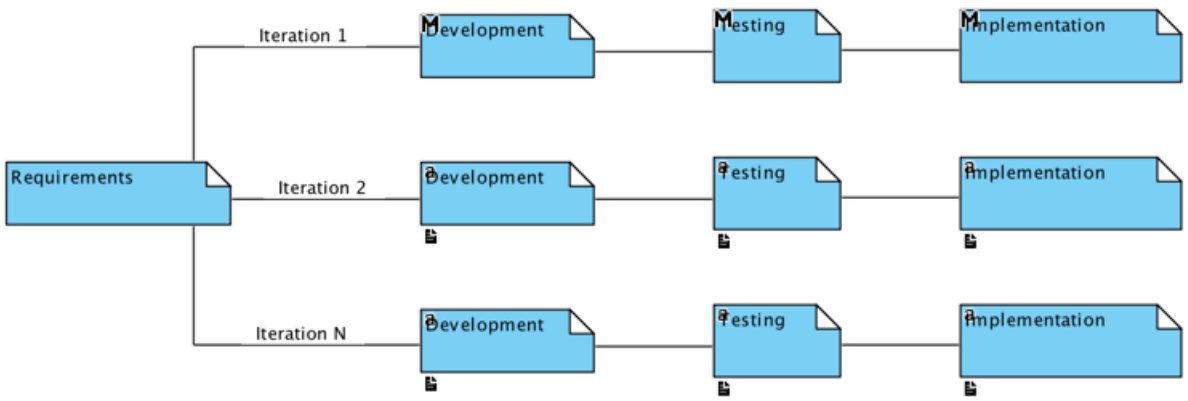


Figure 1.2 Incremental development iteration process

Fig. 1.2 shows the iteration process followed during incremental development. Each increment is given a specific requirement analysis to follow. Once the requirements have been understood, the development phase begins. Once the development of the increment is completed, the increment is tested against the requirements. If the testing phase has been passed, the increment is deployed to intermediate version of the overall system until the system has been fully completed.

Advantages of incremental development:

- Asynchronous development of increments
- Working software produced early in the development life cycle
- Risks are identified and managed during each iteration
- Works well when the requirements are constantly evolving

Disadvantages of incremental development:

- Not suitable for small projects
- More management of the project is required
- Defining increments will be dependent on knowledge of the complete system

2.2.3 Spiral model

Spiral development, proposed by Barry Boehm in 1986 combines some key aspects of the Waterfall model and some aspects of the rapid prototyping models. The spiral model is an incremental, risk orientated life cycle model which comprises of 4 stages; requirements analysis, identify and resolve risks, development and testing, planning next iteration.

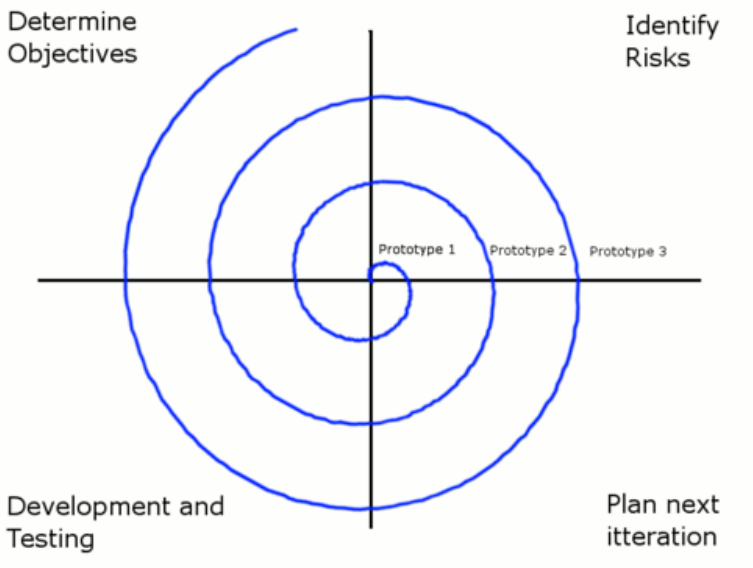


Figure 1.3 Spiral model development phases

Fig 1.3 shows the development phases of the Spiral model. Due to the iterative nature of the process, adding/changing requirements is an easy process as well as identifying and managing any risks involved. This type of development also produces test versions of the software early in the development phases as opposed to that of its predecessor the Waterfall model.

Advantages of Spiral development:

- Software produced early in life cycle
- Risk reduction during each phase
- Requirements can be added/changed easily
- Useful when the project size is fairly large

Disadvantages of Spiral development:

- Cost may be large due to the risk assessment (expertise required)
- Risk of not meeting budget or project deadline
- More documentation due to the intermediate phases

2.3 Chosen methodology

Having explored the methodologies as previously mentioned, developing Super Tutor will require the following:

- Builds early within the software life cycle
- Manageable risk assessment with each build
- Time effective model (to meet project deadline)
- Allows for changing requirements
- Builds being developed simultaneously

Due to the requirements aforementioned, from the developmental methodologies discussed above it would be best fitting to adopt Iterative, incremental development (Rapid application development) for the purpose of this project. This will allow the project to be completed on time, have the risks assessed with each increment, allow for multiple builds to be developed simultaneously and cater for evolving requirements.

Chapter 3: Software lifecycle

3.1 Prologue

The Software Development Life Cycle (SDLC) is a process adopted in industry to produce high quality, scalable applications. The SDLC has 4 main stages; Requirements analysis & gathering, Design phase, Implementation phase, Testing. Each phase within the cycle contributes to the production of high quality, reliable software which can be deployed upon completion. This section will discuss the various phases of the SDLC and how each of them will be used in conjunction with this project.

3.2 Requirements elicitation

Software is only as good as its purpose in the market. Requirements elicitation (analysis) is the most vital part of the SDLC due to the application features/functionality being discovered at this phase.

Developers will interface with users to make a list of features that should be included in the first builds of the application. This list can be gathered through a series of methods all valid as analysis techniques e.g. Questionnaires, Interviews, Paper prototyping, Focus groups etc. Once the list has been compiled and understood, the Requirements can be clearly defined and placed within the Software Requirement Specification (SRS) document. This document is then used to develop the application as a reference to what the software should entail.

Super Tutor will incorporate a variety of analysis techniques which include:

- Questionnaire
- Paper prototyping
- Existing products analysis

These methods will allow for stable requirements to be documented in the SRS document. The aforementioned methods will also allow for requirements to change as the project develops.

3.3 Design phase

During the design phase the product architecture is decided. Developers will decide on a system best suited for the requirements proposed in the Software Requirements Specification (SRS) document and present these to the stakeholders. Due to the various issues which may arise during the various phases of the SDLC, several design proposals are included in this section with the stakeholders selecting the most cost effective, risk minimising methodology.

Once the design methodology has been chosen and the system architectures have been decided upon, the software moves to the 'Implementation phase in the SDLC where the implementation phase begins.

Due to Super Tutor being developed for Android/Desktop applications. It will highlight the following architectures:

- Mobile operating systems
- Desktop programming languages
- Database architecture
- Application prototypes (both mobile & desktop)

These points of interest will allow for stakeholders to make a suitable decision as to what architecture to use for each portion of the application as well as assess the advantages/disadvantages of using each type of architecture.

3.4 Implementation phase

The implementation phase holds the creation of the product based on the Design Document Specification (DDS). During this phase, developers begin to implement the features stated in the SRS and depending on what software methodology was chosen, contribute to the deployment of the application (when all phases have been completed).

The programming language that the application is written in is dependent on what the Design document specification states (this is usually restricted to what the organisation has available).

Due to Super Tutor being run on multiple platforms (mobile & desktop), cross-platform integration will need to be taken into consideration when developing the database (back-end server).

Things to consider:

- Implementation code for Server
- Implementation code for Mobile application
- Implementation code for Content managers (Desktop application)

3.5 Testing phase

The testing phase can be conducted in two ways; the developer tests the build against criteria specified in the DDS, test users (alpha/beta testers) can be used to gather feedback of what the application lacks, does well and which features requires improvement. After a testing strategy has been decided the results are passed to the Evaluation phase of the SDLC.

For the purpose of this project, due to the time constraints imposed by the project deadline Super Tutor will adopt testing done by the developer against a set of testing criteria defined later in this document.

Sub section 3.6: Project Management

3.6.1 Management strategy

To ensure that the project has an effective management strategy, Super Tutor will ensure to cater for the following:

- Sufficient/Maximum user involvement
- Realistic project goals/deadlines
- Well defined Software Requirements Specification Document
- Well managed risks (being assessed with each increment)
- Use of modern technologies (for efficiency and extendibility)
- Required resources clearly defined

Following the criteria specified above, the project should run as efficiently as possible. Deadlines/targets will be met on time with little/no overlap of tasks. This management strategy also takes into consideration evolving requirements in which case it would not hinder the progress of the software.

3.6.2 Management schedule

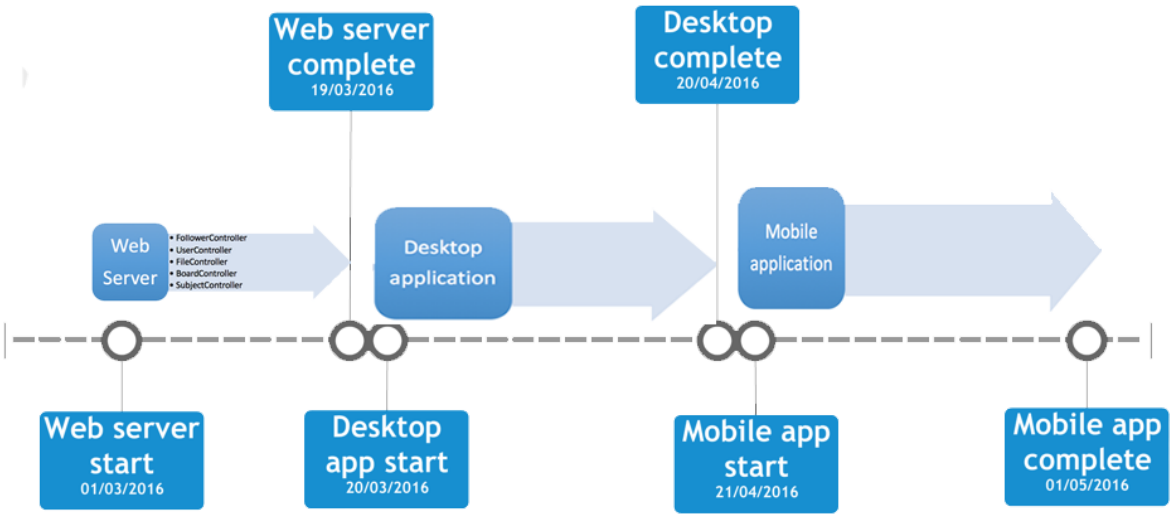


Figure 1.4 Management schedule timeline

Fig. 1.4 shows the proposed schedule along with start/completion dates for each phase of the project. The management strategy can be adapted should the need arise to extend the deadlines.

3.6.3 Reasoning for schedule

Due to the nature of the application, Super Tutor requires a stable web server which is the core of the framework. The need for a stable database within this type of application is vital and hence why 19 days has been designated to this portion.

Next the desktop application (for Content managers) will require the functionality of uploading/downloading content to and from the servers as well as grabbing analytics etc. from the servers. Approximate time required is 30 days to ensure that all features are working correctly to the fullest extent.

Finally, the mobile application will re-use components from the Web server and desktop application however will have a difference in terms of the questionnaires, test activities and topic restructuring (machine learning algorithm) which would require the most time out of the allocated 10 days.

Chapter 4: Requirements Analysis

4.1 Prologue

This section will detail the various sections involved in the Requirement analysis phase. First, market research on existing products will be conducted to gather what questions should be prevalent in the questionnaire. Shortly after, users will be presented with questionnaires to gather knowledge of their expectations. Paper prototypes will then be distributed to allow users to see a visual representation of their requirements before moving onto UML diagrams and the various areas this covers. Finally, this section will document the Functional/Non-functional requirements which have been gathered from the Requirements analysis phase (this will be placed in the Software Requirements Specification document).

Sub section 4.2: Requirements Elicitation

4.2.1 Market research (Existing products)

Market research is vital in designing a good questionnaire as it allows people to determine what features are already available in existing systems and what features those systems lack which brings rise to features for new applications. Below is a list of existing products for the 'already same category' application but offered as a web based solution rather than a mobile application:

- Code academy
- Lynda
- W3Schools

These products have a distinct layout for their testing/revising. The user is first prompted to read their "topic information" which shortly after then taking a test to solidify their knowledge. From these websites we can depict the following questions; average test length, colour schemes associated with education, registration questions, styling of application. These questions will allow a questionnaire with suitable base questions to be designed from which we can administer the survey and record the results for a prototype to be designed.

SITE	TESTING	REVIEW/LEARN	INDIVIDUAL STYLE	REGISTRATION	OFFLINE MODE	FOLLOW USERS/SOCIAL
LYNDA	✓	✓	X	✓	X	X
W3SCHOOLS	✓	✓	LIMITED	✓	X	X
CODE ACADEMY	✓	✓	X	✓	X	X

The table above shows a list of features present in the existing products, and features which are lacking.

4.2.2 Requirements gathering (Questionnaire)

The questionnaire consists of 15 questions. The sample size for this questionnaire will be 10 individuals from varying backgrounds. The results are documented and provided in the appendix as evidence of the questionnaire being conducted. These questions cover the following aspects of the application (***see Appendix A***):

- User registration (i.e. fields they are willing to commit during registration)
- User profile preferences (i.e. social aspects desired or not)
- Look and feel usability (e.g. Flat vs Material design, Colour schemes)
- Information about the tests being administered (i.e. how many questions etc.)
- User suggestions (if not covered by set questions)

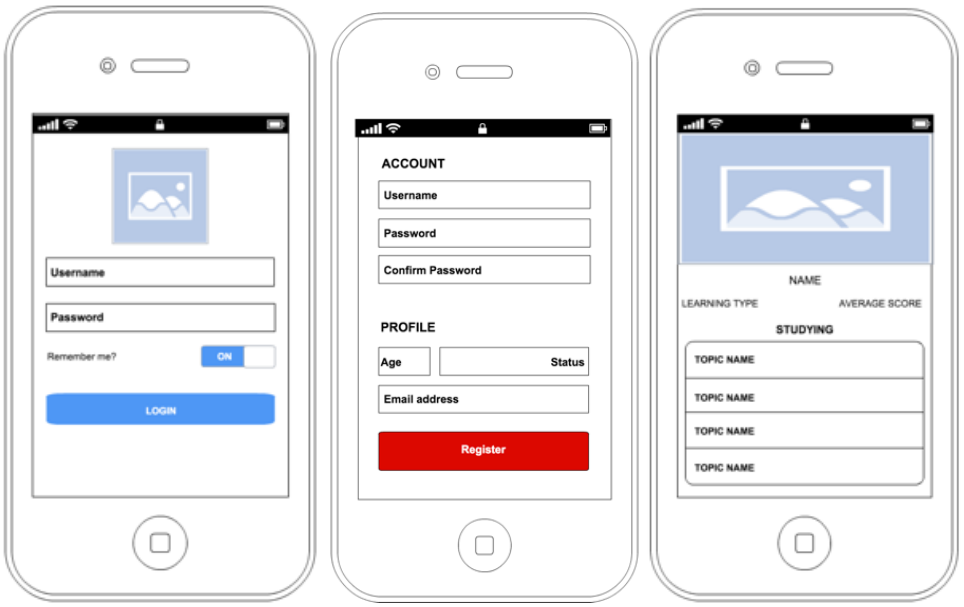
The questionnaire can be viewed at the following URL (http://goo.gl/forms/MdshkxdsyN)

Questionnaire Results:

The results from the questionnaire have been very clear in what users want as it pertains to the functionality of the application. User registration is well determined (in terms of what data they are comfortable sharing when users register to the service), the look and feel of the application is also clearly outlined with the styles which are requested, length of tests and question statistics have been documented (allowing a well rounded, engaging and interactive test) and any suggestions which users would like to contribute have been put forward.

4.2.3 Paper prototypes

Users often do not know what they want during the requirements phase without seeing an implementation of the product. A visual aspect always gives some comfort as to the type of application they will be dealing with, what they can expect from interactions, the features it will offer and the basic flow of events within the application. A throw away prototype will be best in this case to give the users (based on their feedback) to server as a visual representation of their requirements being implemented. From this they can then further narrow down the removals/additions to the application.



Note: These prototypes are used for feedback on the placing/feel of elements. They in ***no way*** reflect the final product.

Sub section: 4.3 Requirements Modelling (UML Diagrams)

4.3.1 Context diagram

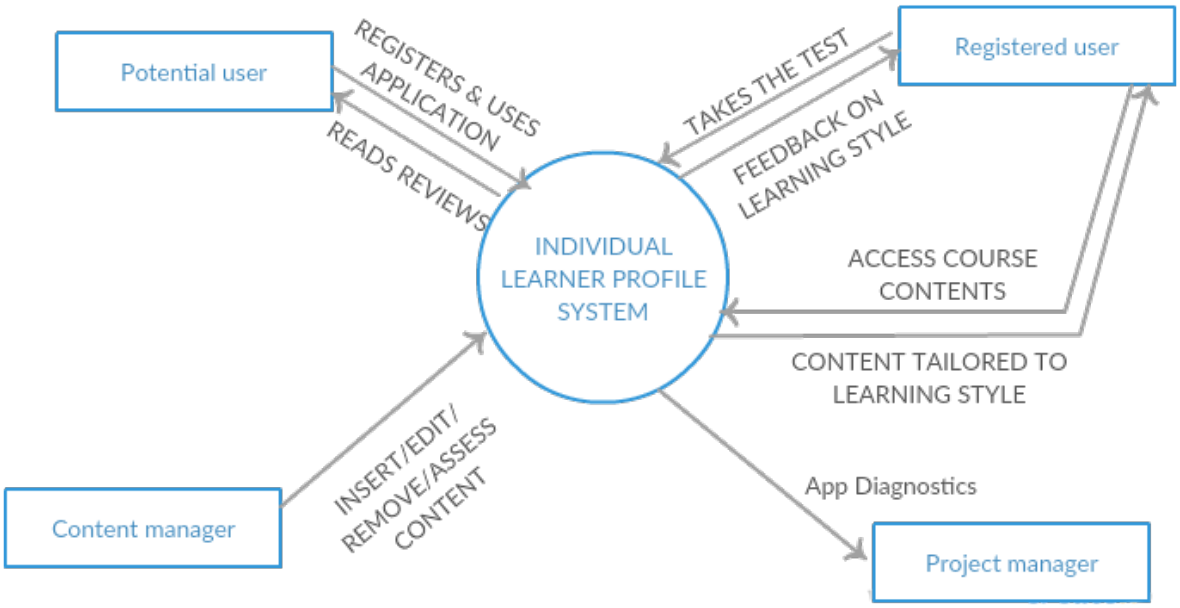


Figure 1.5 Interaction between entities and system

Fig. 1.5 shows the various entities of the system and how each of them interact with the environment encapsulated within. This provides us with a clear direction as to which use cases will be required and how they are implemented with their specifications.

4.3.2 Use case diagram

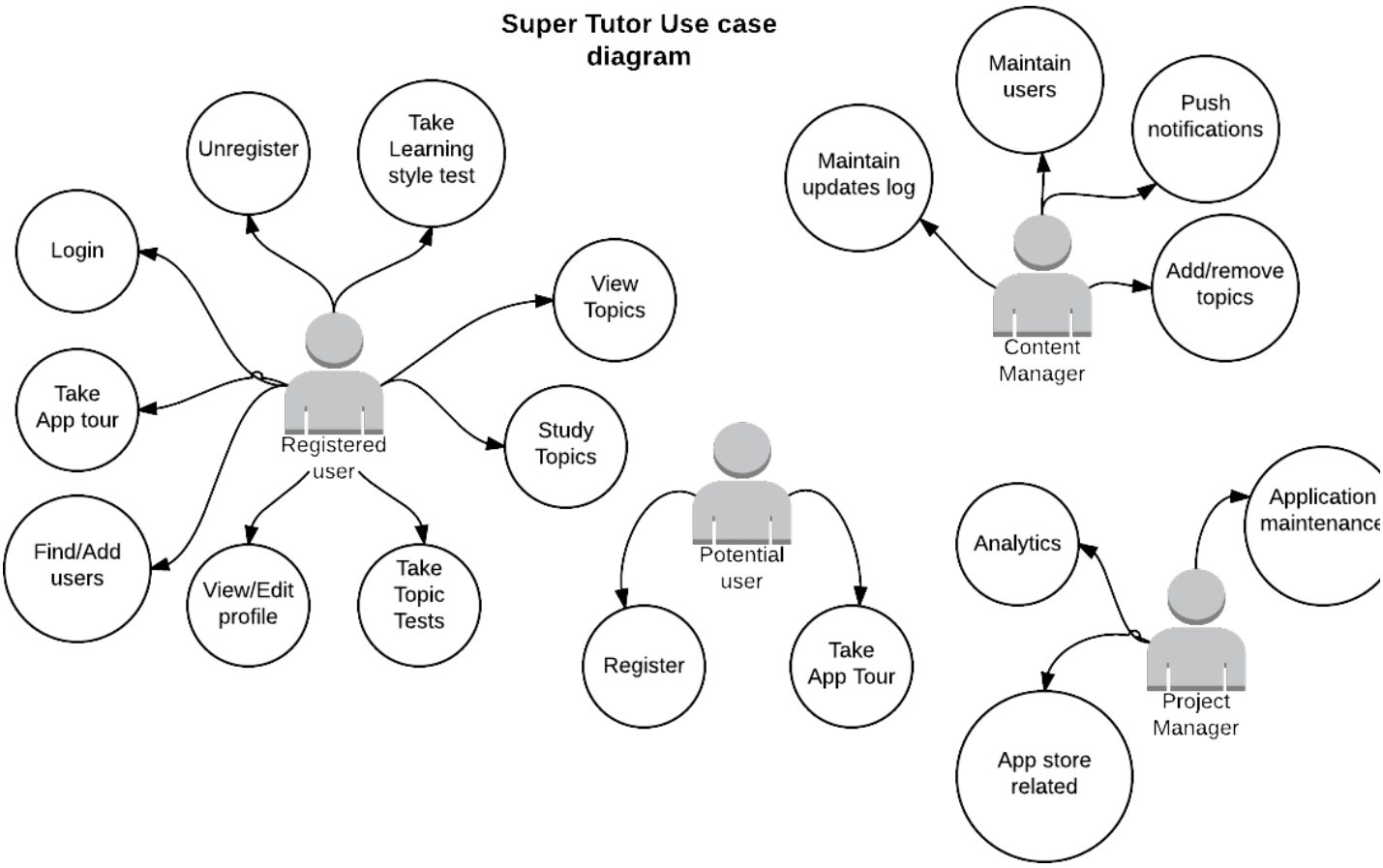


Figure 1.6 Super Tutor use case diagram

Fig. 1.6 shows the use case diagram. This diagram displays the various users of the systems in the environment for which they exist and the decisions they are allowed to make/take when interacting with the application.

4.3.3 Use case specifications

Due to the number use cases within this project, this sub-section will document the key use case specifications.

Use-case: App tour

Description: Allows the user to view a 'Walkthrough' of the applications features with an image and some caption corresponding to the image

Actors: Unregistered User, Registered User

Preconditions: Must have the application downloaded on their smartphone

Flow of Events

Basic Flow:

1. User opens the application
2. System detects user is unregistered (verifying through local DB)
3. System presents the application tour to the user
4. User reads and scrolls through all the available tour pages
5. System detects user is at the end of the tour pages and terminates

Alternate Flow:

1. User selects the "App tour" option in the settings/more/help page
2. Application detects user is already registered
3. The system hides the "Register" and "Login" buttons respectively
4. The user scrolls through the available pages in the tour
5. The system detects the user is at the end of the pages and App tour terminates and returns user to the appropriate page

Exception: *N/A*

Use-case: Register

Description: Allows the user to register for an account to use the application

Actor: Potential User

Pre-conditions: User clicks the “Register” button on the App tour page/Start page

Flow of Events

Basic Flow:

1. System detects user is attempting to register and connects to the server
2. System presents user with registration form
3. User fills in the required fields within the form
4. User clicks the “Create account” button
5. System loads users form and validates the fields
6. System parses the information to the Database (online server)
7. User is presented with a message that their account has been created successfully

Alternate Flow:

1. System fails to connect to the database (due to internet connection issues etc.)
2. User is presented with error message “Unable to register at this time, please try again later”
3. System terminates the registration screen

Exception:

Step 1: System fails to connect to the database, in which case the application will present and error to the user “Unable to register at this time, please try again later”, and the registration screen will be terminated

Step 5: User enters invalid characters in the fields, the system will notify the user of the error

Step 5: Failed validation will require user to enter the correct information into the fields

Step 6: User attempts to register an account with an already existing username, database will return to the application that the username already exists in which case an error is presented to the user prompting them to change their username

Use-case: Login

Description: This allows registered users to log into the application to access its features

Actor: Registered user

Pre-conditions: The user must have already registered for an account. The user must also have clicked the “Login” button or be at the login page at time of this specification

Flow of Events

Basic Flow:

1. System detects that user would like to log in
2. User fills out the login form and clicks “Login”
3. System detects users request and validates fields
4. System presents the user with a loading interface
5. System connects to database and parses information
6. The system re-directs the user to the ‘Main dashboard’ of the application

Alternate Flow:

1. User has entered invalid information
2. The system will present the user with an error “Invalid username/password, try again”

Exceptions:

Step 5: The login server is unavailable, in which case the user will be presented with an error message “Unable to log in at this time, try again later.”

Use-case: Take Learning Style Test

Description: Presents the user with a test in which will determine what style of learning best suits the user

Actor: Registered user

Pre-conditions: The user must have clicked on the 'Take the test' button

Flow of Events

Basic Flow:

1. System detects the user wants to be tested
2. System presents the user with the important information about the test i.e. Time limit etc.
3. User clicks "Start test"
4. System detects users input, presents the user with the test
5. Once the user has cycled through all the questions, user clicks "Finish test"
6. System detects user is at the end of the test and evaluates the users' responses to questions
7. System calculates the users learning style and redirects the user to the results screen
8. System parses the information to the database
9. User acknowledges their learning style

Alternate Flow:

1. User wishes to cancel the test after starting it
2. User clicks the "Exit" button
3. System detects that user wants to stop the testing phase
4. The user is presented with a message "Your test has been terminated, you are free to re-test at anytime"

5. The user acknowledges the notification and is presented with the ‘Main Dashboard’

Exceptions:

Step 8: The system is unable to connect to the results database for that user, in which case it will save the result in offline mode and try again at a later stage

Rare exception: During the test, if the user’s smartphone/application crashes due to performance/memory issues on the device, the test will need to be restarted

Use-case: View/Add topics

Description: Users can view the list of topics/subjects that the application allows them to study

Actor: Registered user

Pre-conditions: User has clicked on “Add subject” button in the main dashboard

Flow of Events

Basic Flow:

1. System detects user wants to add a topic, presents them with the topics screen
2. System connects to the database
3. System downloads all available topic information
4. User scrolls through the list of topics to see what’s on offer (topics already added will be removed from the list)
5. User selects the topic/subject to view more information
6. System detects users input
7. System presents the user with the topic/subject information

Alternate Flow:

1. After the user deciding they want to add the topic by clicking “Download Topic” button

2. System detects users input
3. System presents a loading screen
4. System connects to the database and downloads the topic information
5. System adds the topic to the users' library
6. System presents a "Topic successfully added" message to the user

Exceptions:

Step 3/4- (*alternative flow***):** The system cannot connect to the module database, presenting the user with an error message "Unable to download topic/subject, try again later"

Step 4: Users' network connection terminates during the download process, the user is presented with an error message

Use-case: View/Edit Profile

Description: The user is able to edit the information they submitted upon registration; name, institution of study, privacy settings,

Actor: Registered user

Pre-conditions: The user clicks the 'Settings' cog on their profile page

Flow of Events

Basic Flow:

1. The system detects the users request to edit profile
2. System populates the fields to be edited with currently known information
3. System presents the fields to the user (available to edit)
4. User edits the values of the fields ready to be updated
5. User clicks "Update profile"
6. System detects users input

7. System presents user with loading dialogue
8. System validates information and connects to the database
9. System updates the information on the database
10. System presents success dialogue to user “Successfully updated”
11. System redirects user to their Profile Page with the updated information (also stored locally)

Alternate Flow:

1. User has entered invalid information in the fields to be updated
2. System provides error to the user
3. User updates the fields with correct information

Exceptions:

Step 8: User has entered incorrect information into the fields in which case an error message will be presented to them “The information entered is in the correct format, please verify”

Step 8: The system fails to connect to the database to update the users’ information, an error message is presented to the user “Unable to update profile at this time, your information will be synced later automatically”. The system will then redirect the user to the Profile page

Use-case: Push notifications

Description: Users will receive a notification on their device with the message which has been administered

Actors: Content Manager (CM)

Pre-conditions: The content manager must be logged into the admin panel

Flow of Events

Pre-conditional Flow:

1. CM enters information into the web panel login
2. System detects input and validates information
3. System presents the web panel (if correct information entered)

Basic Flow:

1. System detects the CM's request to push a notification
2. System presents the manager with a 'Notification message' field
3. CM enters the message to be pushed to devices
4. CM clicks "Send notification"
5. System presents the CM with a loading screen
6. System connects to the Notification API and distributes the message

Alternate Flow:

1. CM enters the incorrect details
2. System presents error message to the CM "Incorrect details entered, try again"
3. System redirects user to the login page

Exceptions:

Step 3: The CM has entered invalid characters in the notification message, the system will present an error message to the CM "Invalid characters in message, try again"

Step 6: The notification API is unavailable for android devices (unlikely)

Use-case: Add/Remove topics

Description: The content manager can add/remove topics from the system (whether it be downloadable or hard coded)

Actor: Content Manager (CM)

Pre-conditions: The CM needs to have the information for the topic being added/removed in hand

Flow of Events

Basic Flow:

1. System detects CM's request to add content
2. System presents the topic addition/removal screen
3. CM adds the content desired following the steps presented by the system
4. System detects CM is at the end of the addition process
5. System adds content to the database allowing download

Alternate Flow:

1. System detects invalid content being added
2. System presents the CM with an error message
3. System terminates job

Exceptions: N/A

4.3.4 Domain model

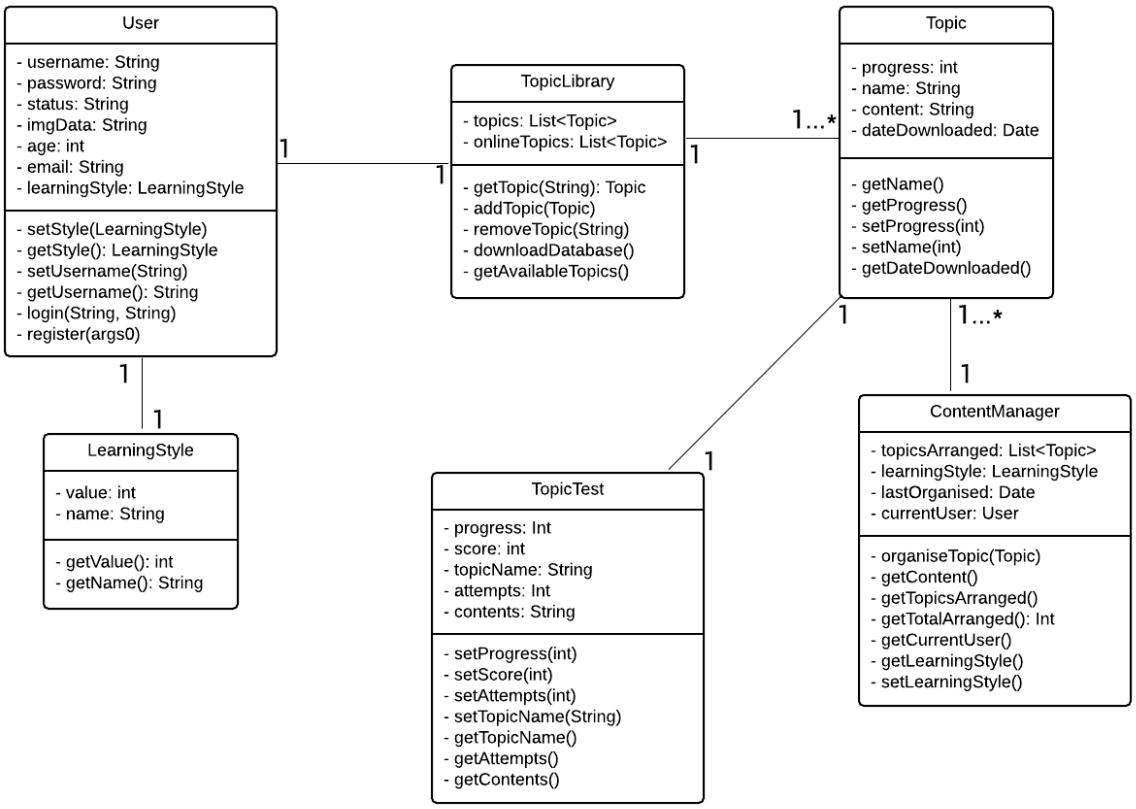


Figure 1.7 Domain model including multiplicities

Fig. 1.7 shows the Domain model proposed for Super Tutor. This model provides an idea of the data and behaviours which have arisen from the problem domain. This model may change as the project evolves.

4.3.5 Class Diagram

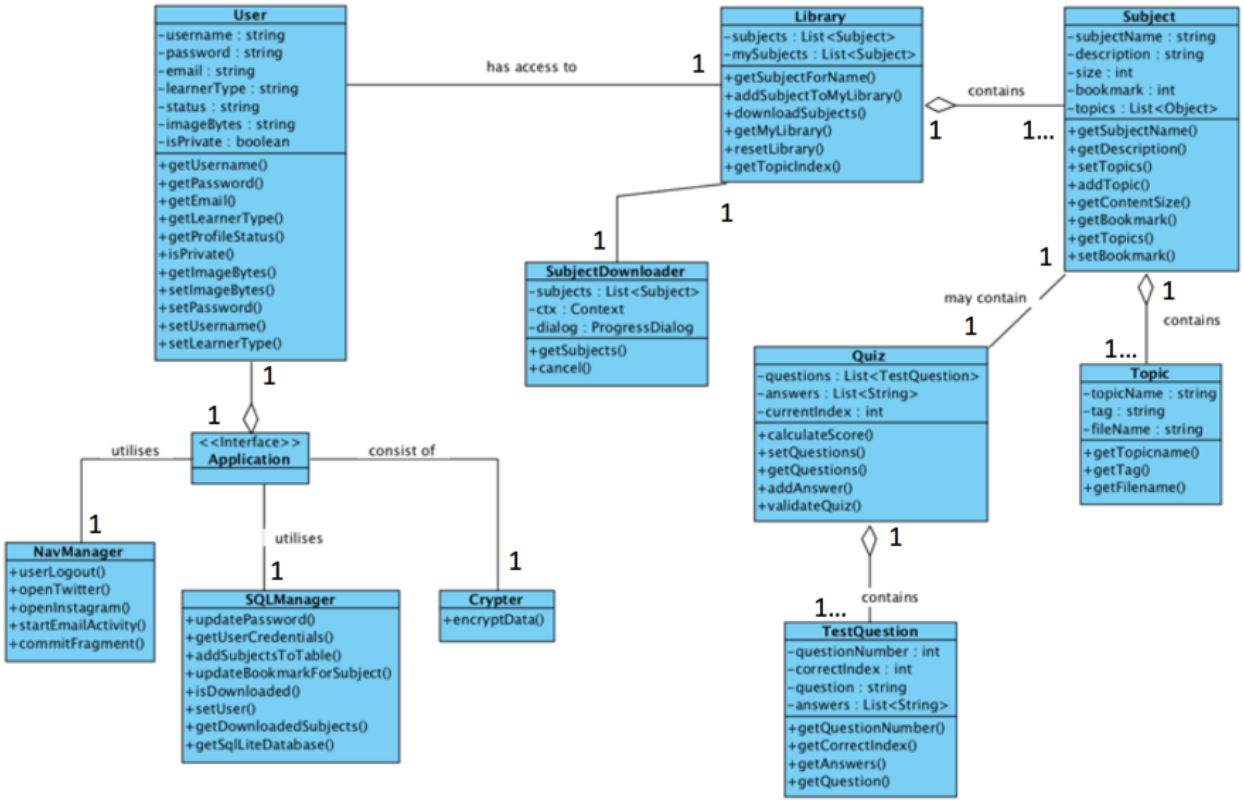


Figure 1.8 Class diagram with relationships to key classes

Fig 1.8 shows the relationships between key classes within the application. This will provide structure during the development phase and will allow for quick and easy implementation for the developer.

4.3.6 Sequence diagram

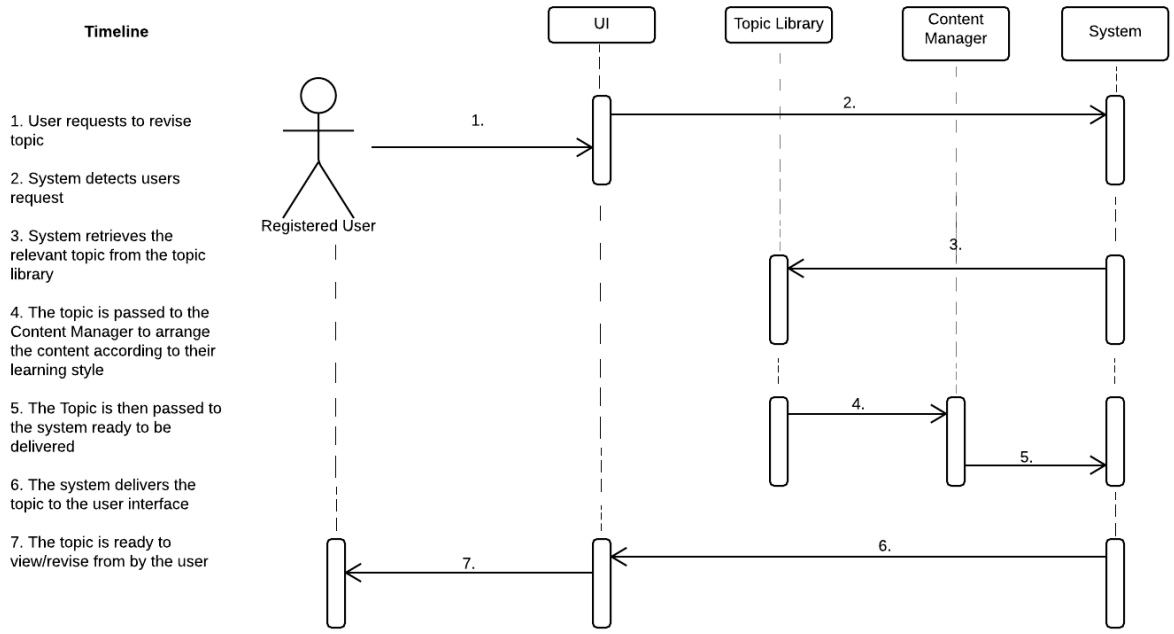


Figure 1.9 Sequence diagram for topic view

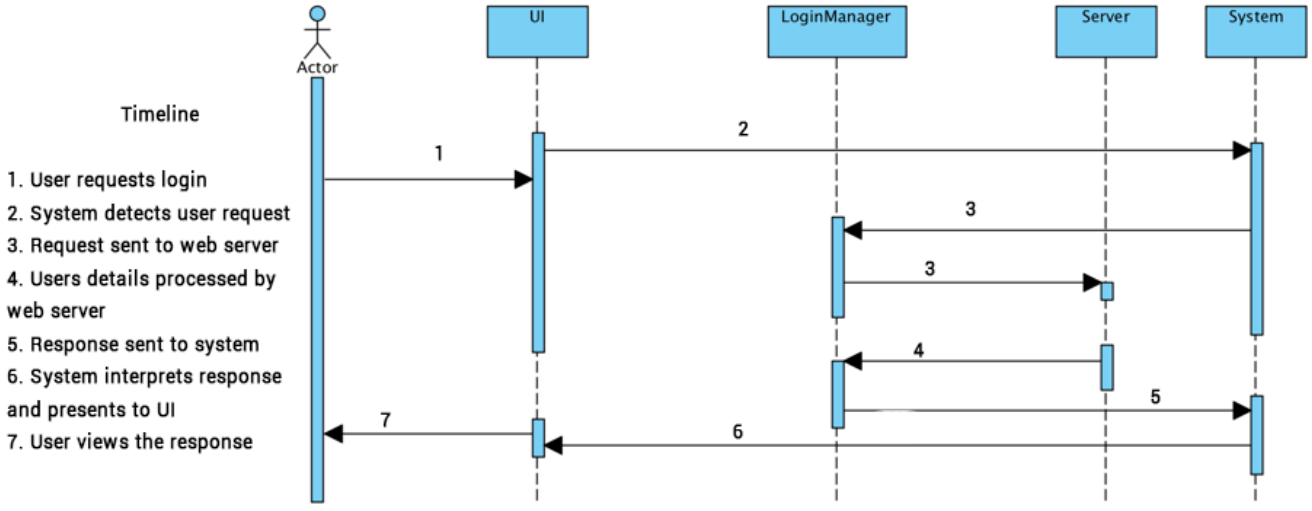


Figure 2.0 Sequence diagram for login

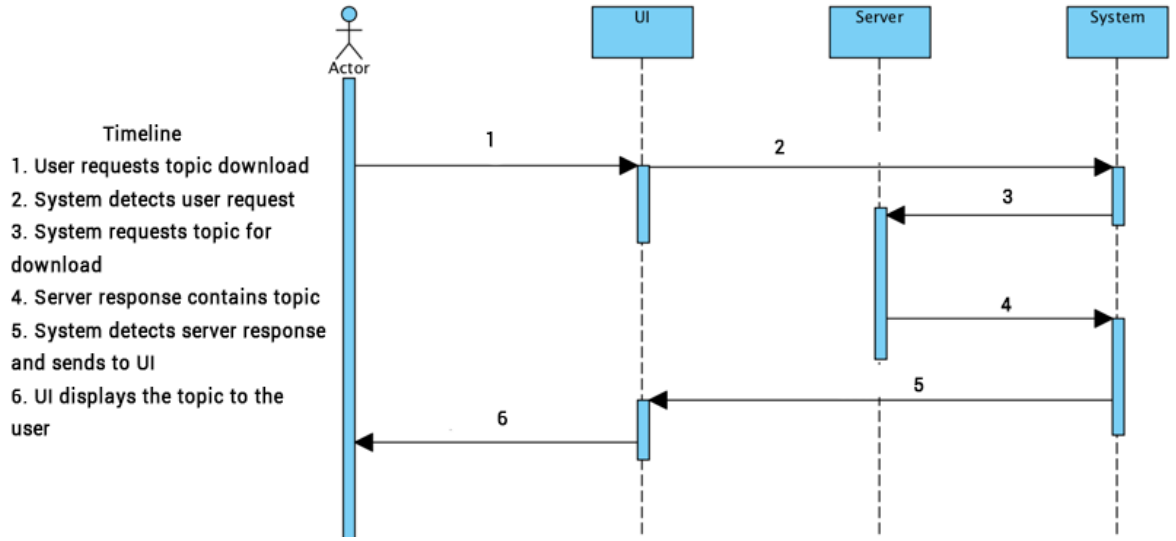


Figure 2.1 Sequence diagram for downloading the topic library

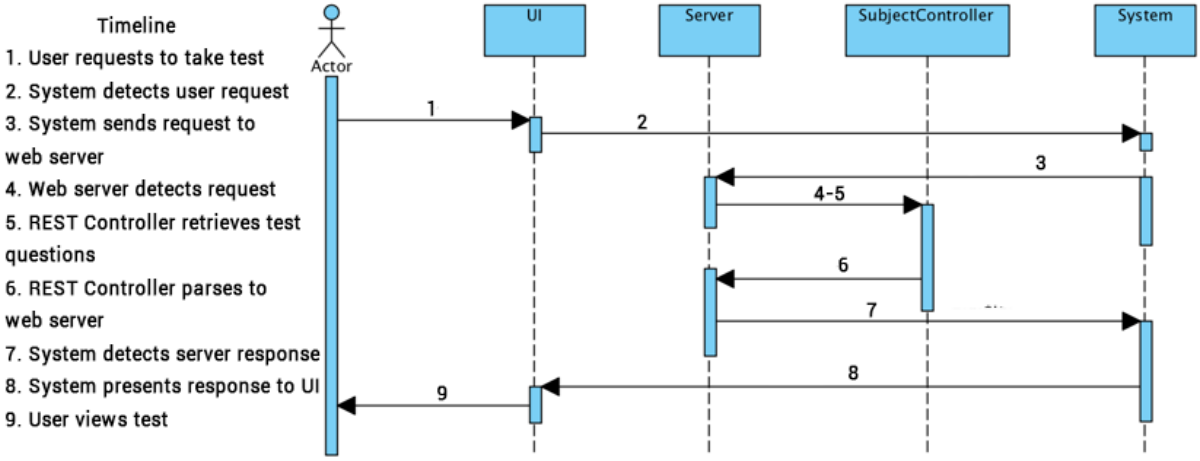


Figure 2.2 Sequence diagram for downloading the topic test

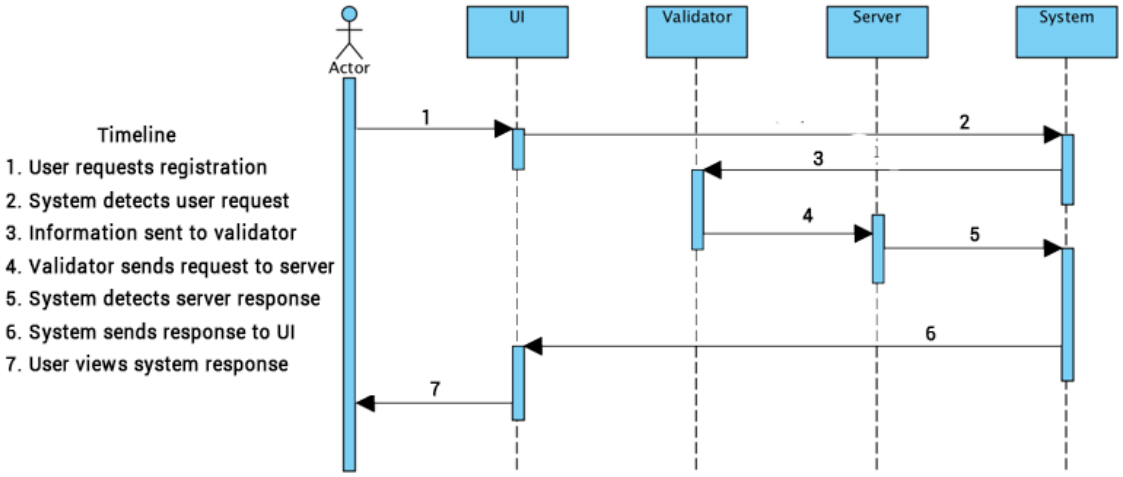


Figure 2.3 Sequence diagram for registration

Sub section 4.4: Functional/Non-functional Requirements

4.4.1 Functional requirements

1. **Splash/Loading Screen (E)**

1.1 Users should be presented with a splash/loading screen upon entering the application

1.2 The application should establish a connection to the server at this point notifying the user of the success (redirect to application- if auto log in enabled and asynchronously retrieve data) or fail (popup message stating the issue, with redirect to the application main page- local mode)

1.3 Query local database for any stored information i.e. subjects stored locally/user information

2. Application Tour/Start-up screen (E)

Note: *This requirement will only execute if the user is a 'first-time user' of the application or if entered through the 'App tour' feature within the application.*

2.1: Users should be able to view the contents of the application via screenshots scrollable on this screen (this will act as an 'App tour' for the application) **(D)**

2.1.1: Users should be shown the core features of the application (through images) to grasp a better understanding of what's on offer/intrigue them to use it

2.1.2: Users should also be presented with an appropriate caption for the image being displayed (see **requirement 2.1.1**) to annotate the feature

2.2: Upon closing the Application tour, users should be returned to the Login page (if first time user) or if accessed through the settings menu, back to the main dashboard.

3. User Registration (E)

3.1: Users should be able to register for accounts which will require the following information; name, D.O.B, current status, email address, username, password

3.2: The email address provided must be a valid format. The application will check the email address and notify the user if its in the incorrect format

3.3: Passwords must consist of a capital letter and number (for added security). The application will validate this field and notify the user of any errors with the format provided

3.4: Users should be able to select a 'Current Status' from a drop-down menu within the registration process. This is to grasp what type of users are using the application

3.5: The registration form should be a single page with minimal information. Where there is certain formatting/requirements, this should be clearly stated.

4. User Login (E)

4.1: Users should be able to log into the application using the credentials specified at registration (refer to **Requirement 2**)

4.2: Once the user has entered their login details, the login button should become interactable

4.2.1: Validation checks for the username should be in place (i.e. correct amount of characters, no invalid characters etc.)

4.2.2: Validation checks for the password should be in place (i.e. does not contain any invalid characters)

4.3: Users should be provided an option to automatically log in (next time the application starts- providing they have entered details correctly)

4.4: Users should have a facility in which if they forget their password, they may request a new one via a 'Forgot password' interface/screen

4.5: Users should have access to the registration screen via a link on this page (for any users who navigated to this page accidentally without registering)

4.6: Once the user clicks "Login", the application should validate the fields entered

4.6.1: The application should check that the username and password are correct/matching database information for the associated account

4.6.2: If the username/password is incorrect, the application should provide feedback to the user notifying them of the issue (not specifically which field as this opens up more security risks)

4.6.3: The data should be retrieved/sent asynchronously so that it does not hinder the functioning of the application

4.6.4: Users should be presented with a "Logging in, please wait…" message to alert them that there is an activity taking place

5. Main Dashboard (E)

5.1: Users should be presented with the topics they are currently studying via the application upon first visit to this page

5.1.1: If the user is not currently subscribed to any of the topics available to them, they should be provided with a message prompting them to subscribe to one of the available topics

5.1.2: Users should be able to tap the topic they are studying and continue from where they last left off on that topic (if not completed)

5.2: Users should be able to access various areas of the application via the Navigation drawer found in the Application bar (***this requirement will be prevalent throughout the applications main pages***)

5.3: Users should be able to see what type of learner they are (had they have taken the test). If they have not yet taken the test

5.3.1: If users have not already taken the test to determine what type of learner they are, there will be a message on screen prompting them to do so

6. Application Library (E)

6.1: Users should have access to the application library (where the topics are hosted).

6.1.2: If the topic has already been downloaded, it should present a "Downloaded" tag to the user.

6.2: Users should be presented with a 'Download dialog' when they have selected the topic they wish to download.

6.2.1: The 'Download dialog' should display the content size and provide an option to cancel if the user chooses to do so

6.3: If the user chooses to download the topic, they should be presented with a progress bar and the status of their download.

6.4: Once a topic has been fully downloaded, users should have the topic available in their personal library.

7. Topic Testing page (E)

7.1: Users should be able to take tests on the topics they have completed revision for

7.2: The tests should consist of a variety of questions based on the topic they have studied i.e. If the user has studied basics of Object orientated programming, then it covers no more than what has been taught in this section.

7.3: Users should be aware of their progress throughout the test so that they know how far in they are

8. Personality Test (E)

8.1: Users will be able to interact with the test in a variety of ways

8.1.1: The test should take into account various aspects of interaction i.e. multiple choice, videos, memory recall, fill in the blanks etc.

8.2: Users should be aware of their progress during the test. This will be presented via a progress bar throughout the test displaying their completion percentage

8.3: Users should be able to repeat the test at any point if they wish to re-assess their learning style

8.4: Once the test has been completed the users learning style will be given as feedback

8.4.1: A results page will notify the user of what type of learner they are

8.4.2: The users learning style should be saved locally on their profile so that the application can re-arrange the content accordingly

9. Individual Profile Page (D)

The individual profile page will be useful for adding a social element to the application. This is where users can share their learning style, scores on tests, information on topics they are studying etc.

9.1: Users should be able to make their profile public/private (if public, then users on the application can find them through the registered users directory)

9.2: Users should be able to customise their profile page **(L)**

9.2.1: Users should be able to change their profile image, name, status

9.2.2: Users should be able to change the background colour of their profile page (where their profile information is displayed)

9.2.3: Users should be able to access their friends list and view their profiles by clicking on their names

9.2.4: Users should be able to search for users via the search icon located in the navigation bar of the application on the profile page

4.4.2 Non-functional requirements

1. **Asynchronous loading (E)**

   If any data is being retrieved from the server (i.e. data related to the user's profile, gathering new topics, updating results on the server etc.), it should be done asynchronously so that it doesn't hinder the functionality of the application

2. **Progress bar updates (D)**

   Progress bars used throughout the application must be updated in real time, this is to make the user aware of their progress throughout that specific task in real time

3. **Security (E)**

   All data stored within the server must be securely encrypted, this is to ensure that users information isn't compromised. This will be done in 2 phases, one using SSL encryption to the server and the data being encrypted in MD5 hashes

4. **Privacy (E)**

   In the case of the profile page, users should be able to privatise their profile, removing it from the application directory. This is essential to the users who do not want their information being displayed in the directory

5. **Network availability (D)**

   If the users network is not available, they should be prompted. Once they have dismissed the prompt the user should be directed to the dashboard in offline mode. This will allow users to continue revising the topics (if previously downloaded and stored on the device locally) and then later sync their data back to the server once their network becomes available

6. **Extendibility (L)**

   The applications topic/subjects that are able to be studied should be easy extensible, this is to allow for dynamic additions of future topics/subjects to the application. This should all be done via the server so that no new application updates (in some cases) will have to take place. This feature will allow for easy maintenance and updates

7. **Stability (E)**

   The application should never crash and should be thoroughly tested through all the possible situations. Test scenarios will be documented within the testing phase and rigorously tested using 3 different testing techniques for maximum efficiency.

8. **Usability (E)**

   The application should have a well thought out user interface. All the main features of the application should be clearly visible to the user and have a clear sense of direction in which way the application is meant to be used. If users are unaware of

how to use the application, there will be a feature which allows them to see 'how its done'

9. **Response time (E)**

The application should have a very fast response time (high efficiency). This is to hinder users from waiting prolonged periods of time to use the application features. This step will require two phases; a highly stable server with good specifications i.e. processor, network uplink etc. and also a well written application with data structures for maximum efficiency on the front end

4.5 Project research

Every individual has a different/unique style of learning. Gathering data to identify how you learn best can be a very taxing process, although we do not understand how every individual learns entirely, many theories have arisen in the field of Educational Psychology. This section will discuss some of the various methods which have been researched (most popular methods) to provide an effective and accurate way of determining your learning preference.

4.5.1 Walter Burke Barbe et al – Learning modalities

Walter Burke Barbe et al (WB) proposed the theory of learning modalities, better known as the VAK (Visual, Auditory, Kinaesthetic) learning styles. The premise of their research was based on the idea that every individual will undergo a change of learning style as they age throughout their life. In the late 1970's after

conducting research on a group of individuals, WB concluded that people vary in their strengths with their results showing that from the sample; 30% were visual learners, 30% mixed, 25% auditory and 15% kinaesthetic. In addition to this data they concluded that auditory learning took place in your primary years, visual & kinaesthetic learning during your teen/secondary years and finally visual & auditory in your adulthood years.

7.3.1 Neil Fleming's VARK Model

Neil Fleming (NF) proposed the VARK (Visual, Auditory, Read/Write, Kinaesthetic) model in 1987, which became a popular model once well established in 2006. NF's model was derived from previous neuro-linguistic programming models and adds a new category, 'Read/Write' preference. Rather than giving you feedback on one specific preference, NF's model provides you with scores on all 4 preferences. This model not only is accurate based on the series of questions in each category, but also allows the individual to explore the various preferences (once scores are obtained) to cater for any discrepancies in the reliability.

Chosen method of assessing personalities

Although there are many studies conducted on which research provides the most accurate results, for the purpose of the scope of this application, it would be best to use the VARK model proposed by Neil Fleming (2006) as it also takes into consideration the 'Read/Write' preference (better suited for the purposes of studying new content- dependent on what preference).

Developing a quiz that will satisfy these needs will need to take this into account with each scenario consisting of the following:

- A question which can be split into 4 sub-categories
- 4 Answers (1 of each category Visual, Auditory, Kinaesthetic, Read/Write – answer code hidden from users)

Each 'scored' question will be inserted into an algorithm which will determine how the user is best suited for each learning preference. After calculating the best style for the user, the user will be informed.

4.6 Security

Due to the nature of the information being gathered from users, Super Tutor will require a high level of security to ensure that the information stored on the server is not compromised. This section will highlight the various methods of security encryption that are possible and the chosen method of encryption for the purpose of this application.

4.6.1 MD5 Encryption

MD5 encryption algorithm is a cryptographic hash function which produces a 128-bit encrypted string. MD5 was developed in 1991 by Ronald Rivest to replace its predecessor the MD4 cryptography. This encryption method is said to be a one-way function, once the information is encrypted it cannot be decrypted (unless brute forcing). Although this standard is said to be highly secure, there have been several decryptions of MD5 algorithms during the Flame malware attack in 2012.

4.6.2 SHA-1

SHA-1 or Secure Hash Algorithm 1 was developed in 1995 by the NSA (National Security Agency) in the United States. This type of cryptography produces a message digest of 160 bits in length which is rendered as a 40 digit long hexadecimal string. At the time this algorithm was announced, it was thought to be the most secure algorithm until the year of 2005 when it was reportedly attacked several times and decrypted. Prior to these events, the NSA were consistently developing its successors (SHA-2 through to SHA-512) through the years.

4.6.3 SHA-2

SHA-2 or Secure Hash Algorithm 2 are a set of cryptographic functions which consists of 6 hash functions all containing a digest. These digests are as follows:

- 224 bits
- 256 bits
- 384 bits
- 512 bits

These cryptographic standards were developed by the NSA in 2001 and are said today to be one of the most secure algorithms in existence. Although attempts have succeeded in breaking the encryption, it is secure enough to last 46 iterations before being decrypted.

4.6.4 Chosen Security cryptography

Due to the strength of the SHA-2/256 algorithm (46 iterations), Super Tutor will benefit from this largely as the computing power required to run 46 iterations of brute force to decrypt this algorithm is significant and provides for the least likely room for a successful attack.

However, dealing with user information is a highly sensitive topic thus brining rise to the second level of encryption that Super Tutor will use. SSL (Secure Sockets Layer) is a secure security standard for establishing an encrypted connection between the browser and the web server. The protocol works as shown below:

- Browser generates a session key using the websites public key
- Website generates a session key using its private key
- If the keys matched, then the browser is granted access, if not, the connection is refused

The cryptographic keys are generated during the application process for the SSL protocol. This will ensure that all information being sent over to the server is highly secure and provides enough safety for users' information.

Sub section 4.7: Ethical issues

4.7.1 Intellectual property

Due to the requirements and the need for cross-platform integration (Java JSON support – JSON Simple, GCM Push notifications library, Apache HTTP Client), Super tutor will utilise some external libraries (all available under the Apache 2.0 License – *Acknowledgements*) which are property of their respective owners and will grant them credit where credit is due.

4.7.2 Privacy

Users' right to privacy (discussed in *4.6 Security*) is a major concern when it comes to ethics. Due to this factor, Super Tutor will attempt to minimise the risk the system as it pertains to user privacy and allow for users to be at ease when using the application and its products. Users also have the right to control what information is held on them, in which case they have the ability to remove this information should they wish for us to no longer be in possession of their details. In addition, users should be free from surveillance, as Super Tutor does not require any picture/video of the user, this has been dealt with automatically.

Super Tutor will never divulge users' information to any of its employees/developers. In the case that the individual requires access to an individuals personal file, the individual will be notified of the intent to make them aware of what the data is going to be used for. Super Tutor will not pass any information on to third parties unless permission from the user has been granted.

4.7.3 Malicious intent

Users' should have the right to use software without the thought of malicious intent. Although gathering data may not seem directly as a "malicious" act, if the data were to be compromised then the attacker may use the data maliciously. It is due to this that encryption and security must be of paramount important during this project (discussed in *4.6 Security*).

Chapter 5: Design phase

5.1 Prologue

Object orientated design allows programmers to firstly plan and visualise their code before writing any form of implementation. This allows for more efficient planning, designing and a faster implementation process. It also shows the interactions of the various objects within the system being used to solve the problem domain (on a conceptual level).

Methodology

The Booch method [1] covers a variety of OO methods including the analysis and design phases. Due to the document already containing the analysis phase (see index page for reference), this section will be on the design phase. Mapping the problem statement and transforming it into a Domain model proves easy using this method; nouns will become classes, adjectives will become variables/attributes and verbs will become the functions in the diagram (see problem statement).

[1] https://www.slac.stanford.edu/BFROOT/www/doc/workbook_kiwi/coding/booch/method.html

Object orientated programming consists of a programming model which maps a set of data to set of objects and shows how the two interact concurrently. This programming technique has become widely used in the 21st century with *almost* all programming languages adopting and encouraging people to use the Object Orientated design. Super Tutor will be based around this practice and thus it is appropriate to use the Object Orientated Design model (such as Boochs' model) to demonstrate the various underpinnings of such a project.

Sub section 5.2: Database/Server back end

5.2.1 Introduction

Super Tutor will need to cater for several things which can be used across various platforms. For the purpose of uploading content (through the content manager), it will require the ability to upload content to the server, downloading the content to the mobile/desktop application and also store the information on various users/subjects added to the system.

Whilst Super Tutor is written for Android, the web server should be accessible across any platform once the application is ready to be scaled (across several platforms). Spring is run within a JVM based system and allows for flexibility and scalability as well as ease of use and portability. As the application evolves and more features/functionality is added, the server back end will need to evolve accordingly to support the various changes. Flexibility is essential when it comes to updating/maintaining a scalable application thus making it the better candidate amongst the other competitors (e.g. PHP & MySQL etc.).

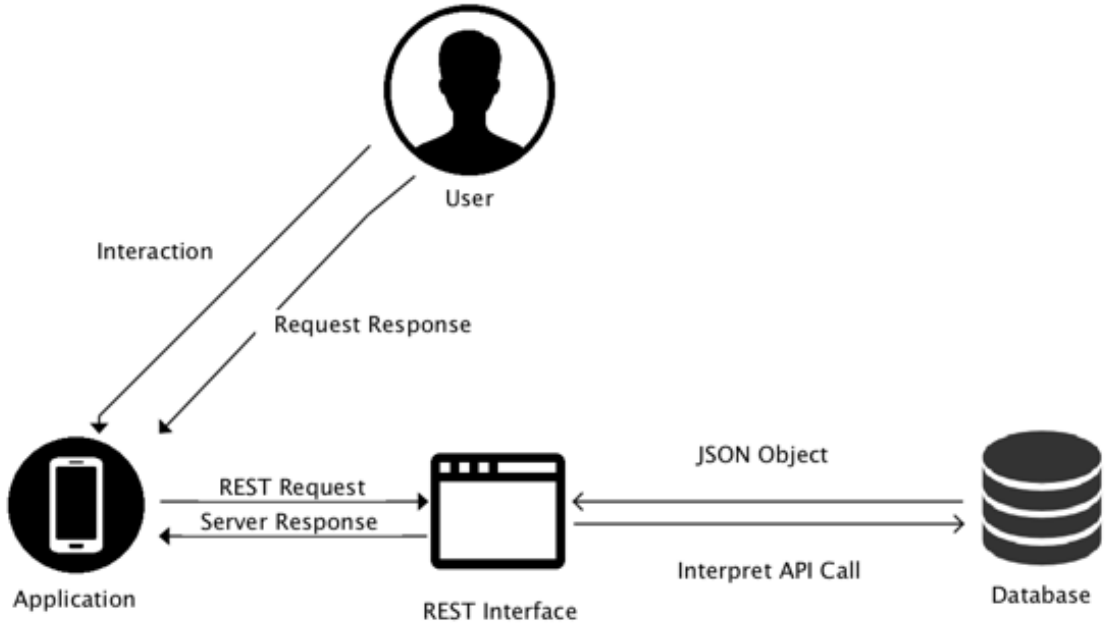
Spring MVC framework provides a model-view controller back end server which can be easily integrated into any application via web requests. The web requests are handled by a third party library (integrated into Spring MVC) called Apache Tomcat. This powerful server sided application can be used to develop an extremely powerful REST (Representational State Transfer) API which can be used to access various objects/data stored within the application by making simple URL requests and reading the data from the application in JSON format.

5.2.2 Database architecture

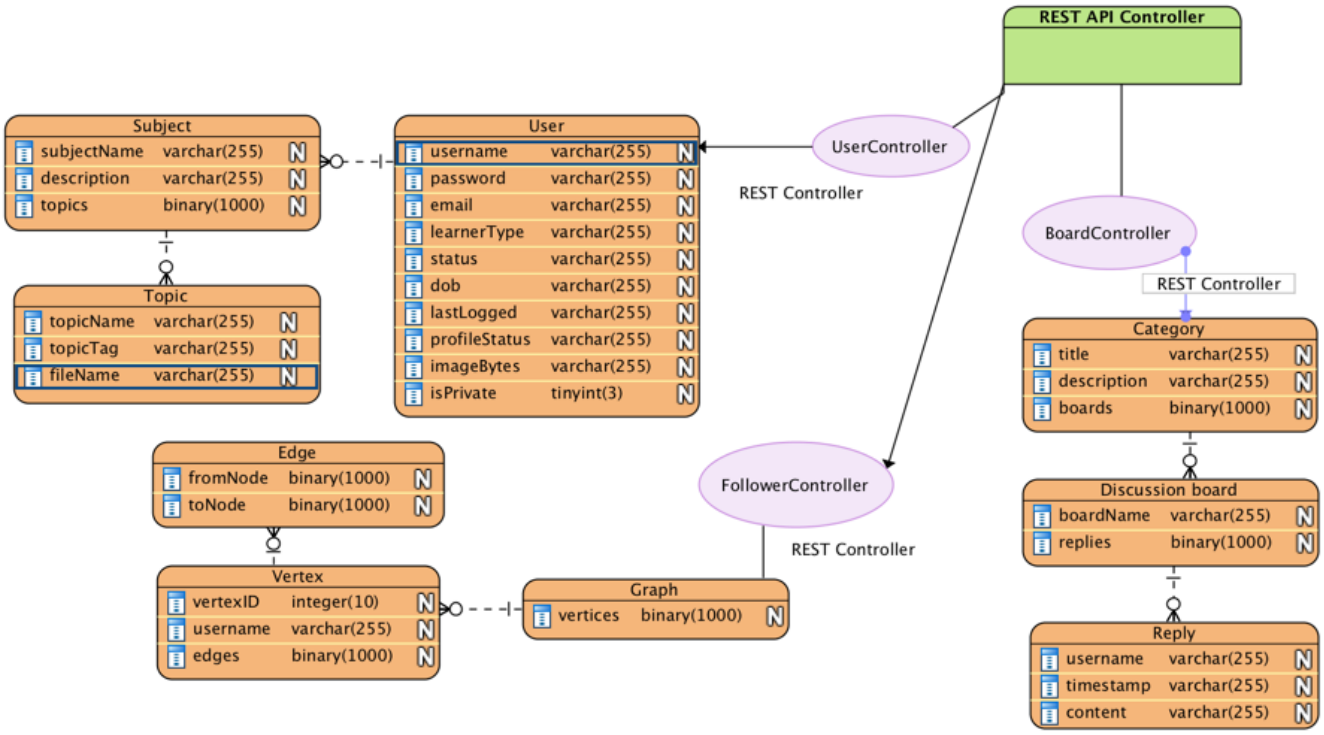
The typical architecture for the database will consist of the following interactions/flow:

1. User makes a request via the application
2. The application connects to the REST API (creates call)
3. The REST API communicates the request to the Database which retrieves the appropriate information
4. The database returns this information to the REST interface (browser output stream)
5. Application interprets the response and presents it to the user appropriately

REST API Call Request Diagram



REST API – Entity Relationship Diagram



REST API Controller

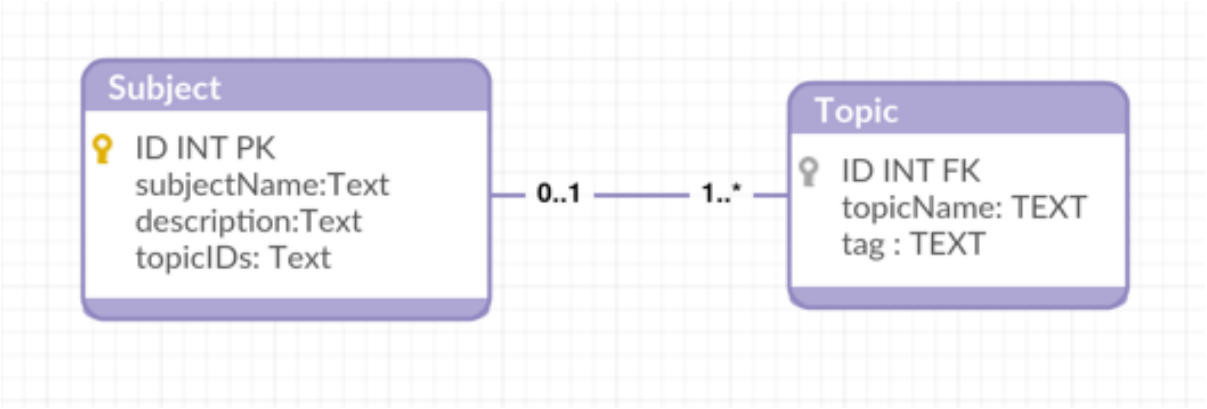
The REST (Representational State Transfer) API will act as an intermediate between the database and the response object (the output stream). The REST Controllers can be called through the API via direct URL calls.

REST Controller

The REST Controller objects will act as the containers for communication with the REST API Controller. Calls will be made to the API Controller, passed onto the REST Controller objects appropriate to the request and these will return the appropriate responses.

5.2.3 SQLite Database

SQLite is a Database standard which is used in the Android platform. Stemming from a transactional SQL database engine, this will provide a useful method of storing information required for offline use when there is no network connectivity available.



This schema will be used to store the information in the SQLite Database

Sub section 5.3: Application Design

5.3.1 System architectures

There are several architectures in which this application can be developed and deployed. This section will explore each of these architectures in depth, giving more information about the advantages/disadvantages of the architectures being described, finally ending with which architecture will be best suited for the purpose of this application and how it will benefit by using the selected architecture.

The architectures being described in this section will be as follows:

- Native applications
- Web applications
- Hybrid applications

5.3.2 Native mobile applications

Native applications are software applications that can be accessed on a mobile device, typically hosted on an app-store/market of some sort and downloaded directly to the user's device. These types of applications can make use of the features available on the platform it is being hosted on e.g. GPS location etc.

Advantages:

- Hosted on the users mobile locally
- Better for offline use when there is no internet connection available
- Able to utilise all of the mobiles features, e.g. GPS location, Camera etc.

- Provides a better UI experience more specific to what the user is usually use to using
- Greater efficiency/speeds vs Web & Hybrid applications

Disadvantages:

- System is not cross compatible e.g. for iOS, Desktop etc.
- User base is restricted to the platform its run on
- High cost involved in development

5.3.3 Web applications

Web applications are those which are hosted on web servers and accessed through the device's browser. These are particularly cost efficient as you do not need to develop the same application for the various mobile platforms. It works on the premise that it is "write once run everywhere" similar to that of Java. Although this is very cost efficient there are some other features which are restricted from these type of applications e.g. the need for internet connectivity. Web applications are typically written in HTML5 which has given rise to some powerful API that allows for giving the "Native application" look and feel.

Advantages:

- Cross platform integration is easy
- Hosted online (more discoverable to new users)
- Native application look and feel
- Maintenance is fast and no new downloads required for users to access new features

Disadvantages:

- Relies solely on internet connectivity
- Cannot utilise the full arsenal of features available on the device
- Offline features are limited and not reliable
- User interface is one that users are not already use to

5.3.4 Hybrid applications

Hybrid applications are essentially web applications written to function within a “wrapper” for the mobile platform. This wrapper gives access to the devices full list of features e.g. accessing contact book, GPS location etc. Hybrid applications are websites run within a browser through the platforms native application. This type of architecture is easy to implement, requires little to no cross platform changes and is cost effective compared to that of a native application.

Advantages:

- Make use of device features fully e.g. Camera etc.
- Runs within a native application wrapper
- Cost effective
- Cross-platform integration is simple and easy to implement

Disadvantages:

- Still relies on internet connection for content access

- Speed is not as fast as native applications

5.3.5 Desktop applications

Super Tutor will also require a Desktop application in which the Tutor/Content Managers can upload content and view various other information from. This section will detail the programming languages available to use on Desktop platforms and which language has been chosen for the development of the Desktop application. The two languages being detailed are cross-platform compatible (easier to implement across various operating systems).

5.2.2.1 C++

C++ is a developing Object orientated language which is available across several platforms e.g. MacOS, Windows, Linux etc. Although they require different implementations, cross platform integrating is a possibility.

Advantages:

- Rich and extensive libraries
- Stronger type checking
- Type safe linking

Disadvantages:

- Less efficient garbage collection
- As the complexity increases, resource usages do
- Not fully an object orientated language

5.2.2.2 Java

Java is an Object orientated programming language developed by Oracle. One of the main advantages of Java is that it is a “write once run everywhere” language. The code is executed inside a Java Virtual Machine which will convert the Java code into a machine readable format and handle the execution.

Advantages:

- Object orientated
- Wide variety of extensions/libraries
- Efficient garbage collection (providing for better run times)
- Multi-threading
- Synchronisation

Disadvantages:

- No pre/post conditions
- No user interface builder
- No built in browsing support

5.3.6 System architecture suited for Super Tutor

Although Super Tutor is run from a back-end web server, its most powerful feature is providing an Offline mode for users to access content when there is no internet connection available. In addition to this, it must cater for an “already familiar” UI experience so that

users find it quick and easy to navigate around the application and use features that they are already accustomed to using in everyday life. Some of the other important features are:

- Speed and efficiency (users must be able to access content in a timely manner with limited resources being used)
- Security (users must have their information securely stored with limited information being uploaded to the web)
- Must have access to data/features on the user's phone for increased interactivity

For the reasons highlighted above, Super Tutor will be developed as a Native mobile application. Although this type of application is not cross-platform, this can also be seen as an advantage in the sense that users are more likely to repeatedly use applications which are developed on a platform familiar to them. For this reason, it can be seen as an advantage rather than a disadvantage for Native applications.

In addition, although the development cost for this type of application is far more significant than its alternatives, the features which this architecture gives rise to is more useful to the user than the cost of development. It has been proven in industry that a good application will always require a lot of investment and developmental sacrifice. If the application is developed to a standard that is unbeatable the costs of development will be covered from other areas such as advertising, product promotions etc. once there is a large enough user base.

In terms of the Desktop application, Super Tutor will be developed using the Java platform. Connection to the back-end web server must be asynchronous so that it does not hinder the functioning of the foreground application. For this reason, multi-threading will play a huge role in accomplishing this function. In addition, the UI thread must completely run independently to the application thread (networking etc.) and this is where Object orientated programming will play a vital role in developing such a system.

Sub section 5.4: Operating Systems

5.4.1 Introduction to android Operating systems

Android provides various versions of the Android operating systems which are compatible on devices which meet specific requirements. These operating systems all have different features and and functionalities with some being better than others and more efficient than others in some cases. This section will highlight the various operating systems being looked at and which operating systems Super Tutor will cater for.

5.4.2 Android Jelly Bean

Android Jelly Bean is one of the earlier versions of Android released in the year 2012. This version of the operating system had all the predecessor features along with some performance improvements as well as improvements to the overall look and feel of the operating system. In 2013, at a Google I/O developer conference there were several more game changing features released in this version of the operating system. These changes include:

- Screen savers
- Multi-user support (for when logging into tablets)
- Lock-screen widgets
- Quick access to the settings interface

5.4.3 Android Lollipop

Android Lollipop now the most used Android operating system was announced in 2014 was developed using old system architectures (Android Kit Kat) with some new improvements. This game changing update brought rise to the now popular methodology “Material design” for mobile applications. With a completely re-designed user interface and set of tools for developers to use, the applications became more responsive, efficient and more appealing to the user demographics.

Other features announced in this build:

- Battery consumption improvements
- Re-designed notification system (using cards)
- 5000 new API’s for developers to use
- Support for new processor architectures

5.4.4 Android Marshmallow

Android Marshmallow was announced at a recent Google I/O Developer conference in 2015. Although there are many changes to the performance/architecture of the new

operating system, it is still fairly new and requires users with older versions of phones to buy a new device which supports this type of architecture.

This operating system has many features and benefits but due to the current life span it has been available for and due to the amount of users who are still using the Kit Kat platform, it would not be worth using until there is a higher rate of users using this platform.

5.4.5 Chosen Operating System for Super Tutor

Although all of the operating systems aforementioned have their individual benefits, due to the popularity of the Android Lollipop platform and the various features introduced with this build, Super Tutor will support this at a target level with a minimum level of API 21 (Kit Kat). The Lollipop build will allow for a better user experience due to the Material Design language introduced in this build, along with the 5000 new API calls that can be utilised within this build.

New devices running Android Marshmallow will most certainly still be able to use the full list of features available in Super Tutor, however newly introduced features such as the Fingerprint unlock feature brought forth in Marshmallow will not be available to use with the Super Tutor application. This does not in any way limit the performance improvements brought about in the Marshmallow build, however does limit the features that Super Tutor has access to as mentioned above.

Sub section 5.5: Application Prototyping

5.5.1 Introduction

Before developing an application, there are a few things which need to be documented/mapped out first. Prototypes allow for applications to be given a clear sense of direction as to where the application is going, what the application is going to look like (even before any implementation has began) and the features associated with each node/screen within the application. This allows for developers to visualise where the elements of the UI will be placed on the canvas and focus solely on the functionality/implementation of the underlying code (aspects such as efficiency, concurrency etc.).

This section will detail the various screens (further referred to as Nodes) of the application and show the relative locations and positioning of all the components. The prototypes presented here will be detailed fully as it pertains to their functionality as well as if there are any expected changes.

5.5.2 Mobile Prototype table

Prototype name	Approx. Features	Networking	Associated Nodes
Splash screen	1	YES	App tour, Login screen
App tour	2	NO	Login screen
Login screen	2	YES	Registration screen, Dashboard
Registration screen	3	YES	Dashboard, Login screen
Dashboard	3	YES	Navigation drawer
Discussion board	4	YES	Navigation drawer
Personality quiz	3	YES	Navigation drawer

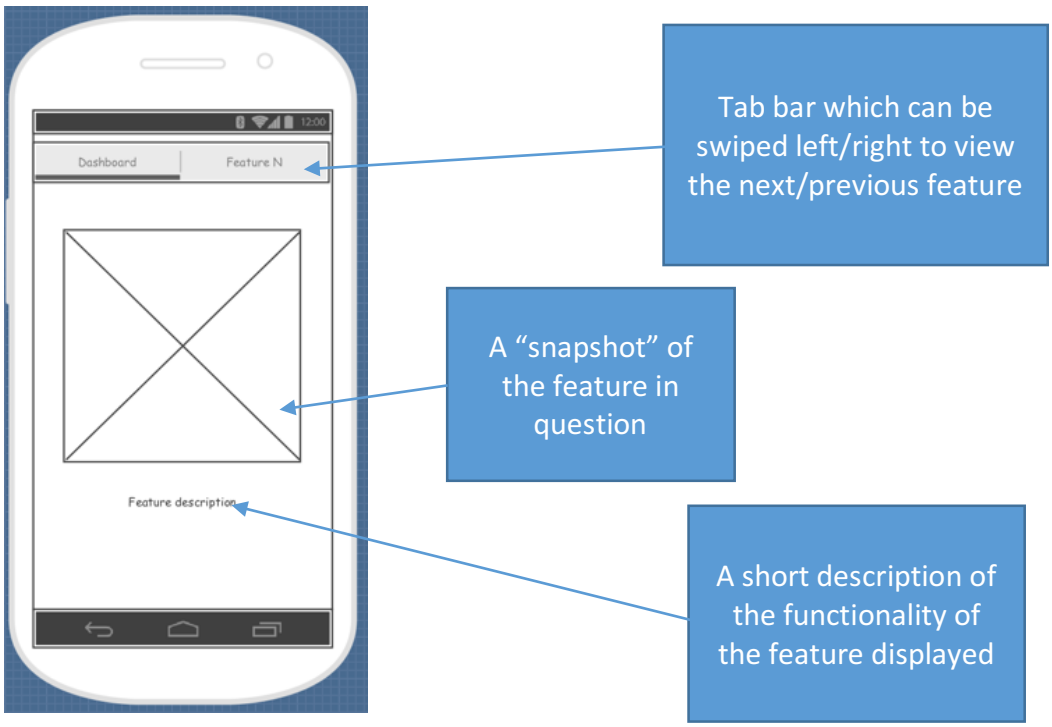
Library	2	YES	Navigation drawer, Dashboard
Profile page	6	YES	Connections, Add friend
Settings node	3	YES	App tour, Change password
Change password	1	YES	N/A

5.5.3 Mobile Prototypes

Splash screen

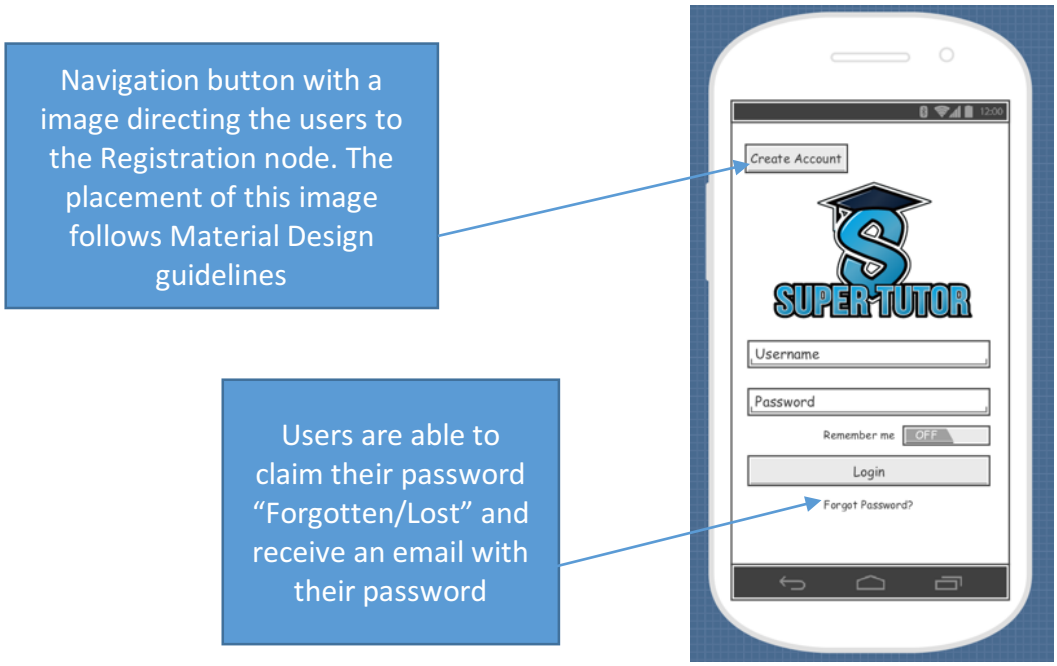
The Splash node will be responsible for parsing data from the local database to the application context (when available). If none is available, determine whether the user is a first time user, in which the App tour node will be displayed or direct them to the Login node if there is no data stored.

App Tour



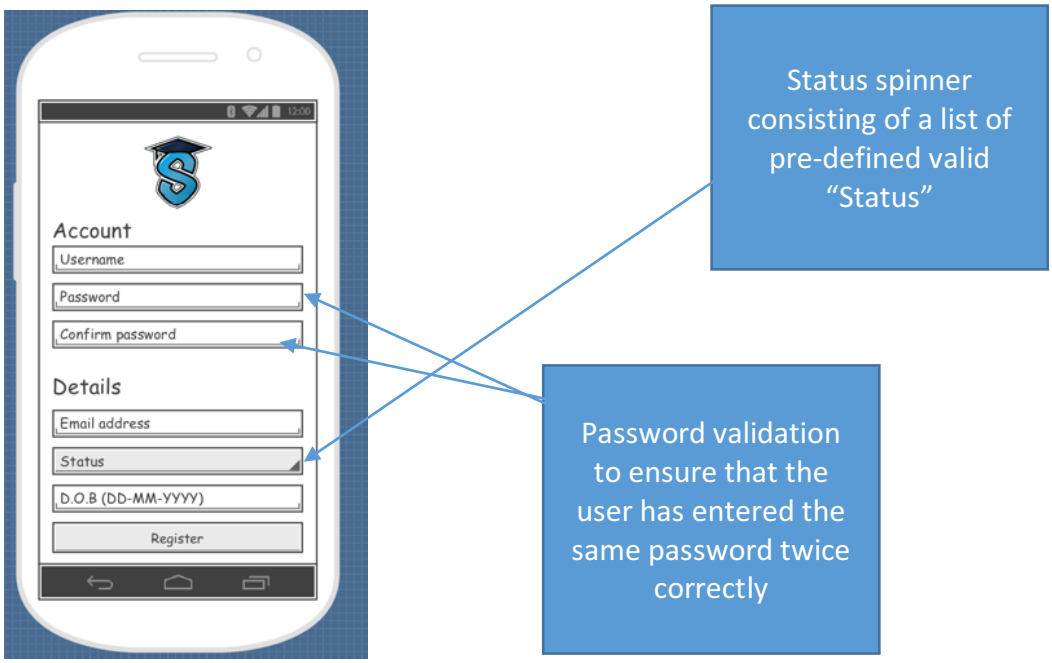
Users will be able to navigate through the various features of the application displayed in this Node. This will provide an insight of what features the application has to offer, which may intrigue a user to register for an account.

Login screen



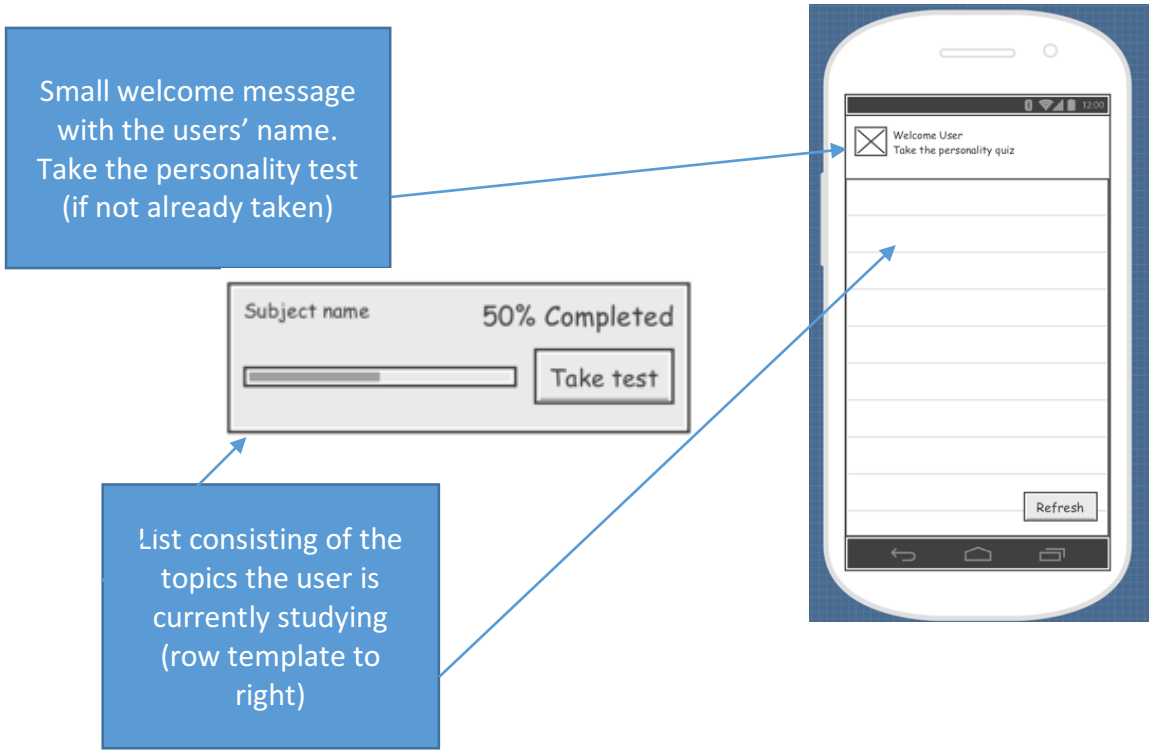
The login activity will allow users to access the main features of the application (after successful login). Users will also be able to register for an account or claim their password lost/forgotten.

Registration node



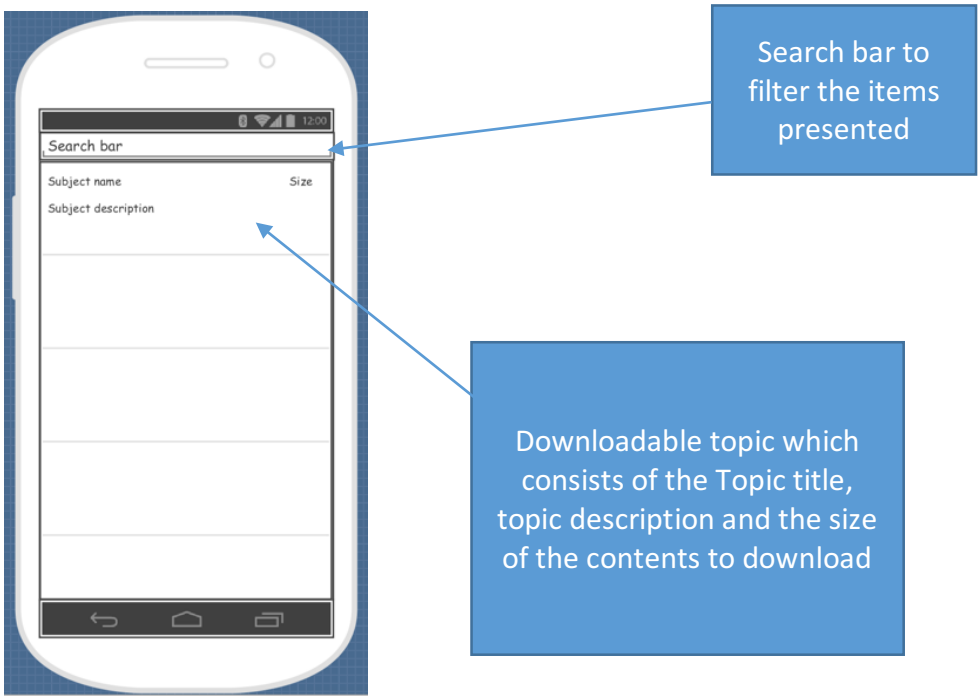
The registration screen allows users to register for an account (if they do not already have one). The features of this node consist of field validation and registration notifications (i.e. success/fail etc.).

Dashboard



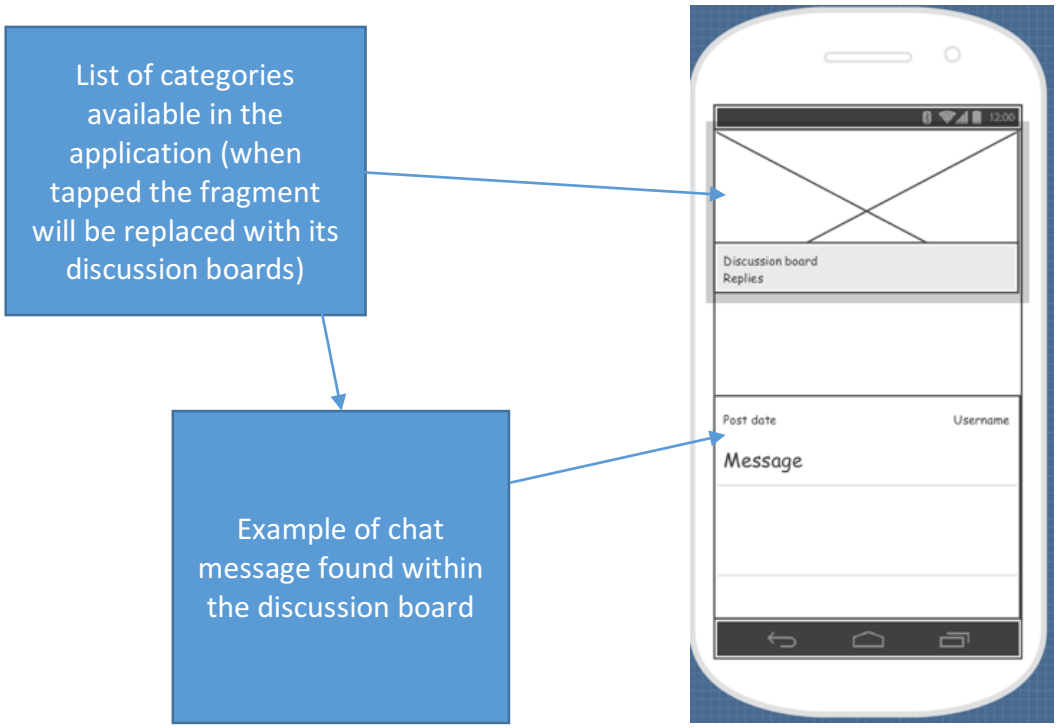
The Dashboard node will present the users with the topics that they are currently studying. The user has the option to revise, test themselves, or take the personality test again.

Library node



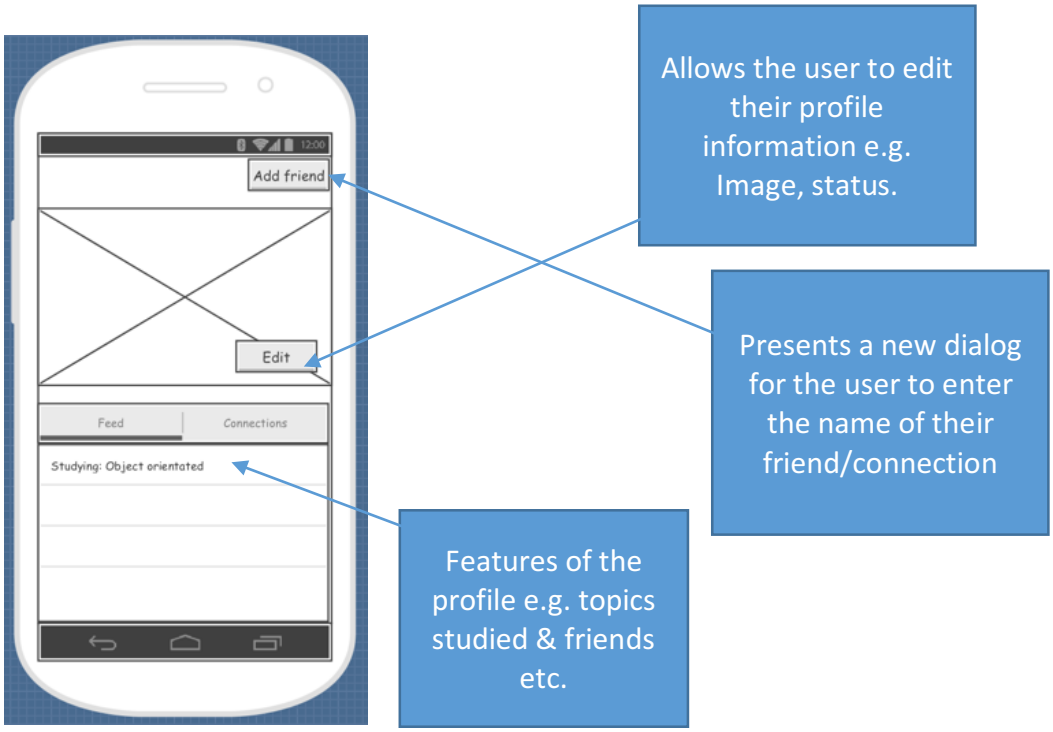
The Library node will house the topics available to download via the application. The rows will display the size, subject title and description. Users can tap the subject row to download.

Discussion board



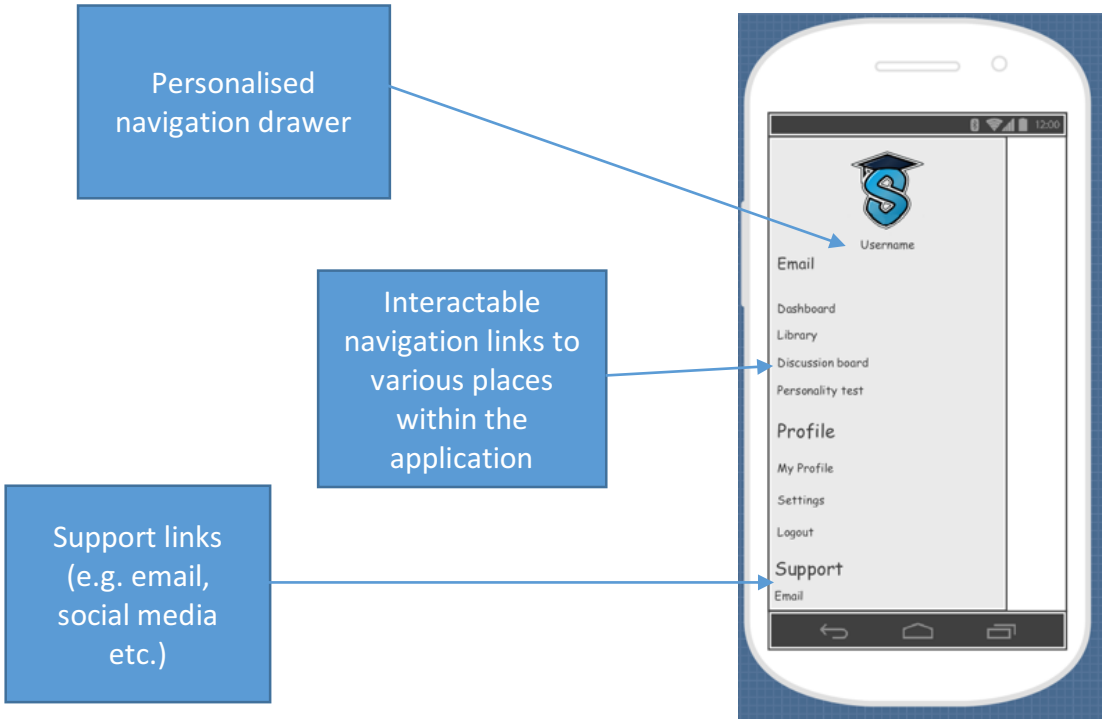
The Discussion board container will consist of fragments which will allow the user to end up in a “chat-room” style discussion board.

Profile node



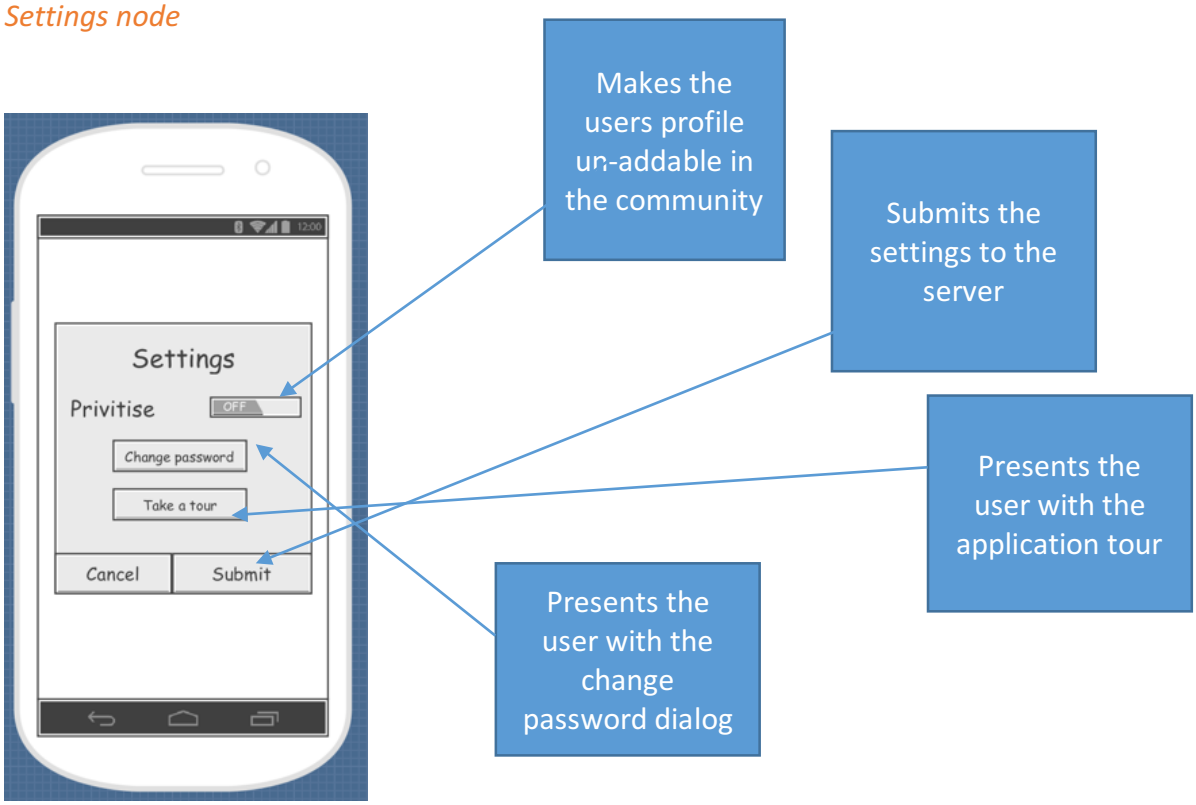
The Profile page will be specific to the users' credentials. They will be able to change their information as it pertains to display picture, status and add friends/connections as they want. Any connections to the web server are done asynchronously.

Navigation drawer



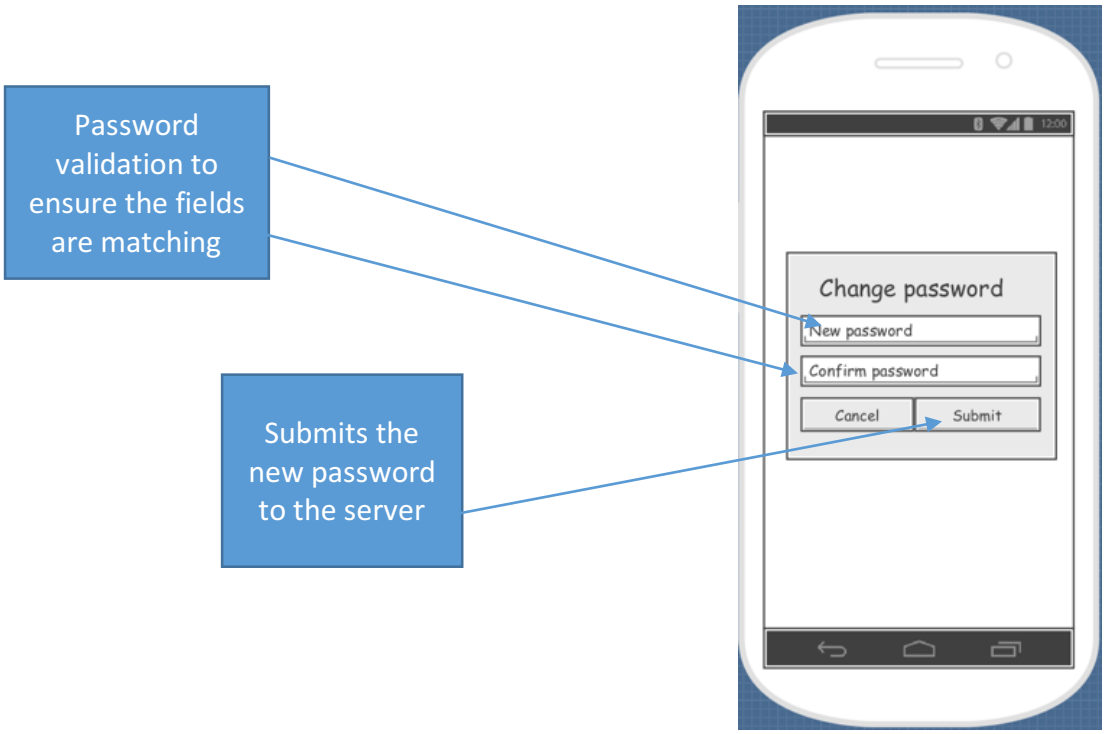
The navigation drawer will provide access to the various features of the application. Users will be able to navigate using this.

Settings node



The Settings node will allow users to customise various aspects of their application. This will include changing their password, privatising their profile and taking the app tour again.

Change password



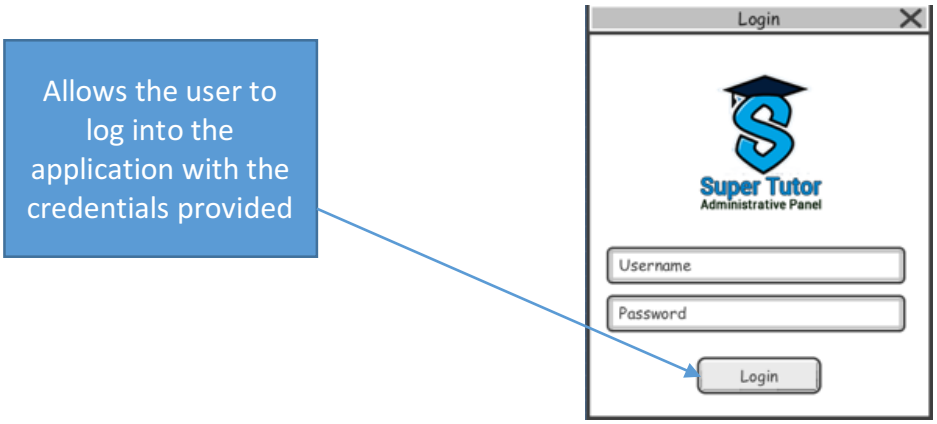
This node will allow users to change the password to their account. All networking is done asynchronously.

5.5.4 Desktop Prototype table

Node name	Approx. features	Networking	Related nodes
Login screen	2	YES	Dashboard
Dashboard	5	YES	Help menus, Login screen, Topic manager
Topic Manager	2	YES	Dashboard
Help dialogs	6	NO	Dashboard

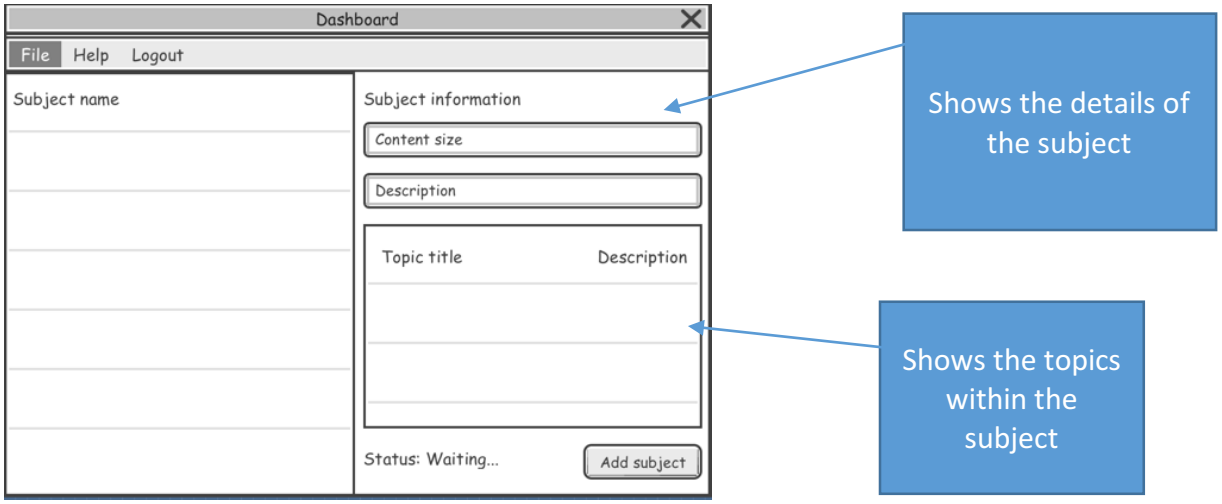
5.5.5 Desktop Prototypes

Login screen



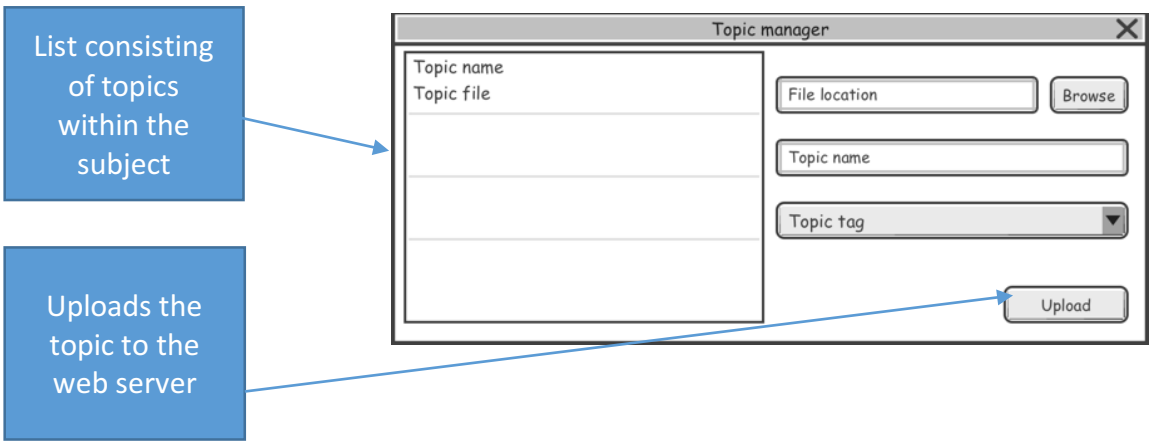
The login screen will give the Content manager/Tutor access to the main Dashboard of the application. All network connections are asynchronous and do not hinder the use of the UI.

Dashboard node



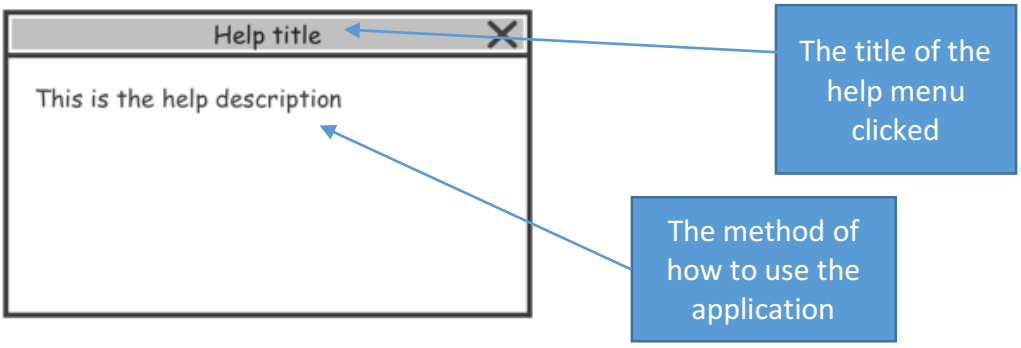
Allows the user to view the various subjects and upload new ones with topics. The subjects can also have various functions applied with a right-click context menu.

Topic manager



Allows the user to upload topics and remove them. The file location will display the location of the file user has selected to be uploaded.

Help dialog



Help dialog will consist of the various help messages across the application. This will consist of the following dialogs; change font, how to upload, how to delete topics/subjects etc.

Chapter 6: Implementation

Sub section 6.1 Algorithmic design

6.1.1 Prologue

Super Tutor will require a AI (Artificial intelligence Algorithm) which will consider factors such as how the user learns best and the topics which need to be organised. The first part of the algorithm will require the AI to determine what type of learner the user is. For this, a formula needs to be devised in which the AI can 'adapt' over time as more users are subscribed to the service. This section will detail the background research done into Artificial intelligence and Machine learning, document the formulae's used and demonstrate the AI in action through means of activity/state diagrams.

6.1.2 Background research

Artificial intelligence is becoming an increasingly popular area of research in the field of Computer science. Humans are consistently looking for ways to replace human representatives with that of machines, e.g. Automated checkout in super markets, chat bots in chat applications which connects the user to the respective product and many other usages.

With an ever evolving market technology has to keep up to date with the change and for this reason it is fitting that Super Tutor uses an Artificial intelligence to make way in the field of personalised learning.

The first point of call would be to design an algorithm which determines which personality the user is with the second requiring an algorithm to organise the application's content in a way that suits the users learning preference. The next section will discuss the steps taken into development of the algorithms and the appropriate state diagrams.

6.1.3 Algorithm development & state diagram

Firstly, tackling the issue of determining what learning preference the user is, this portion will be about the development of this algorithm and how the decisions were made. Below are assumptions we can already draw from the personality questionnaire devised earlier on in this documentation. The questionnaire is set in a way that will give us 4 outputs, the score for visual, auditory, kinaesthetic and read/write respectively. Using these scores, we can develop a wrapper to place them in with their respective opCodes (Operation codes i.e. R, V, A, K).

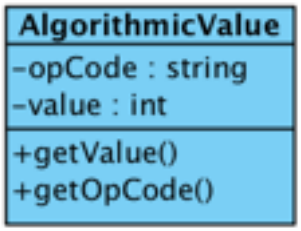


Figure 2.4 AlgorithmicValue wrapper

Fig. 2.4 shows the wrapper which will store the values required for the calculations along with their opCodes (*detailed see above*). From this we can develop a comparator to sort the values with the highest order first. Necessary for future developments of the application, an algorithmic constant multiplier has been introduced into the equation:

AlgorithmicValue = Ak x v

Where Ak is an algorithmic constant multiplier, v is the value retrieved from the questionnaire

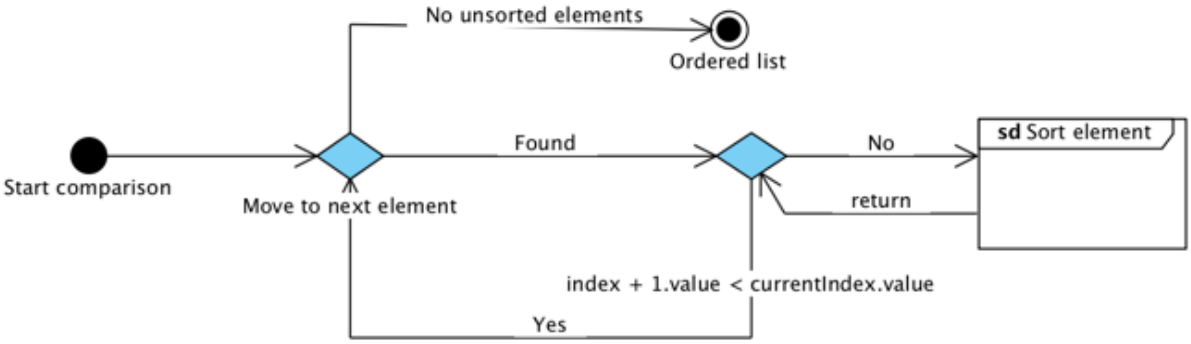


Figure 2.5 Comparator Algorithm

Fig. 2.5 demonstrates the underpinnings of the comparator algorithm. This will allow for faster development of the overall AI, and allows for clear direction of how the comparator needs to function.

After retrieving the data from the comparator, the value which is present in index 0 will be the greatest score out of the values retrieved from the users score. Next, retrieve the opCodes associated with that value from the AlgorithmicValue wrapper, and determine what style the user is.

Next, we will need to determine an algorithm for arranging the topics based on the users learning style.

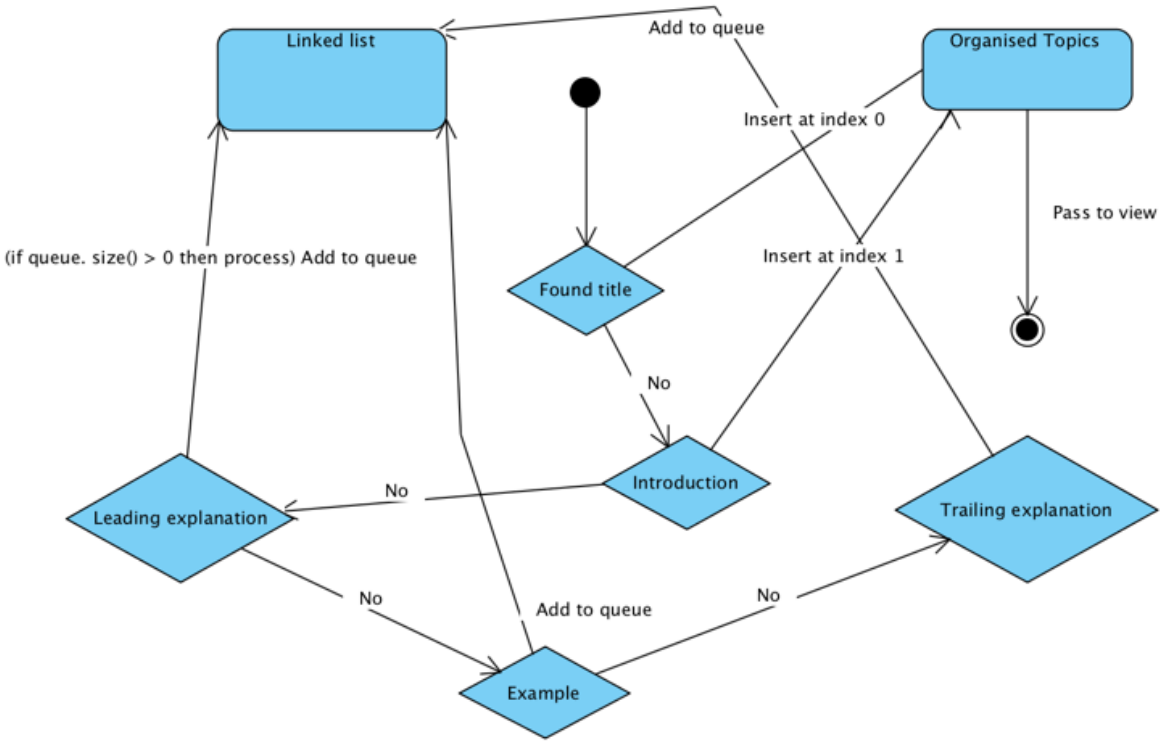


Figure 2.6 Algorithm design of Content Manager

Fig. 2.6 is a state diagram of a possible implementation of the Content Manager algorithm. The algorithm will consist of a Queue, and a list of organised content. Iterating through the list of topics, it will determine which path to follow dependent on the “Tag” field from the Topic wrapper. The explanation is as follows:

- Topic titles are always first
- Introductions follow after the topic title
- Leading explanation – An explanation of a diagram which follows
- Trailing explanation – An explanation of a diagram which was previously seen
- Example – A diagram demonstrating the topic being discussed

Topics uploaded to Super Tutor will follow this structure. We can assume that topics uploaded will already have the appropriate format and will only require the Content Manager to place the order of the topic.

Finally, to give the application a sense of community as well as allow users to 'compete' with one another, a follower base is required where users can follow other members registered to the application (if profile is not private) and allow them to view their statistics.

Undirected graphs are the best algorithmic approach to achieve this functionality. The graph will consist of 3 portions:

- A Graph (which will consist of the edges, vertices)
- An Edge
- A Vertex

System design for undirected graph:

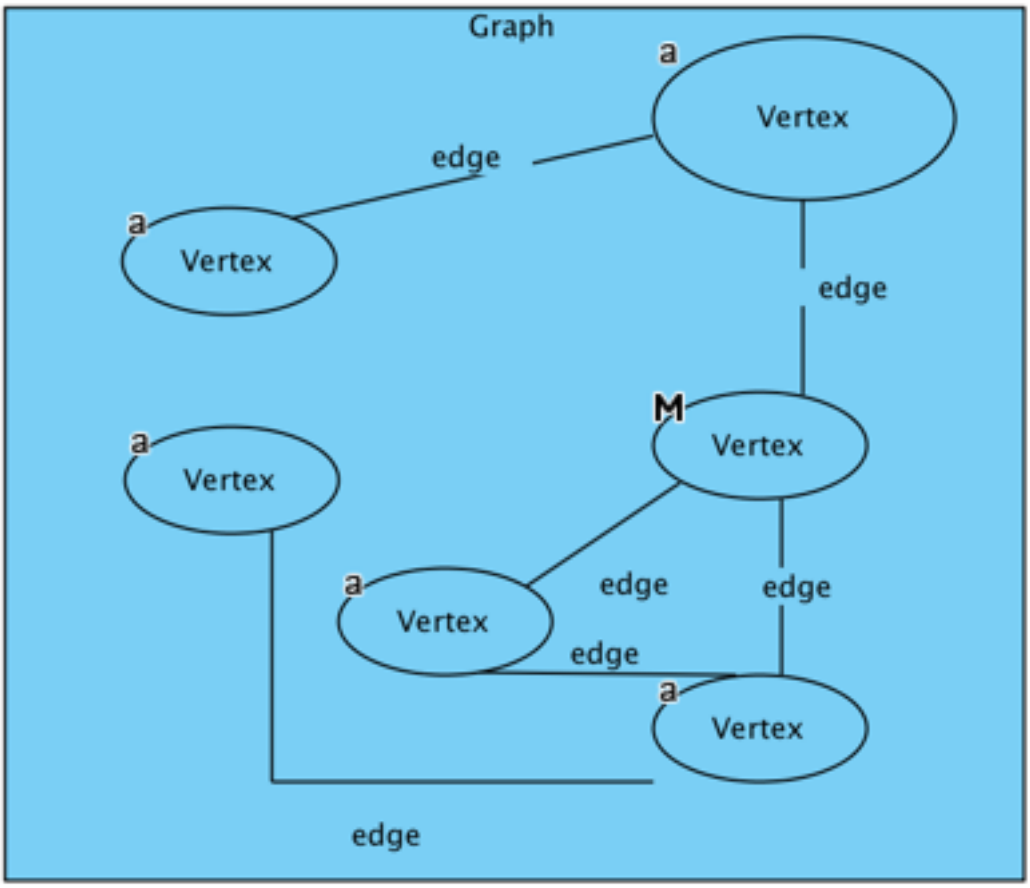


Figure 2.7 Undirected graph algorithm

Fig. 2.7 shows the design of the undirected graph algorithm. Each Vertex will represent a user, each Edge will be the connection between two users and the entire system maps to form a Graph. The Graph will contain the methods relevant to connecting two Vertices with an Edge and retrieving information from its connections i.e. User X is following User Y.

Sub section 6.2 Key features

6.2.1 Prologue

This section will document some of the key features implemented across the Super Tutor platform. The decisions for the implementation will be documented thoroughly starting with the web server, moving onto the desktop application and then the mobile application (order of development).

6.2.2 Web server implementation

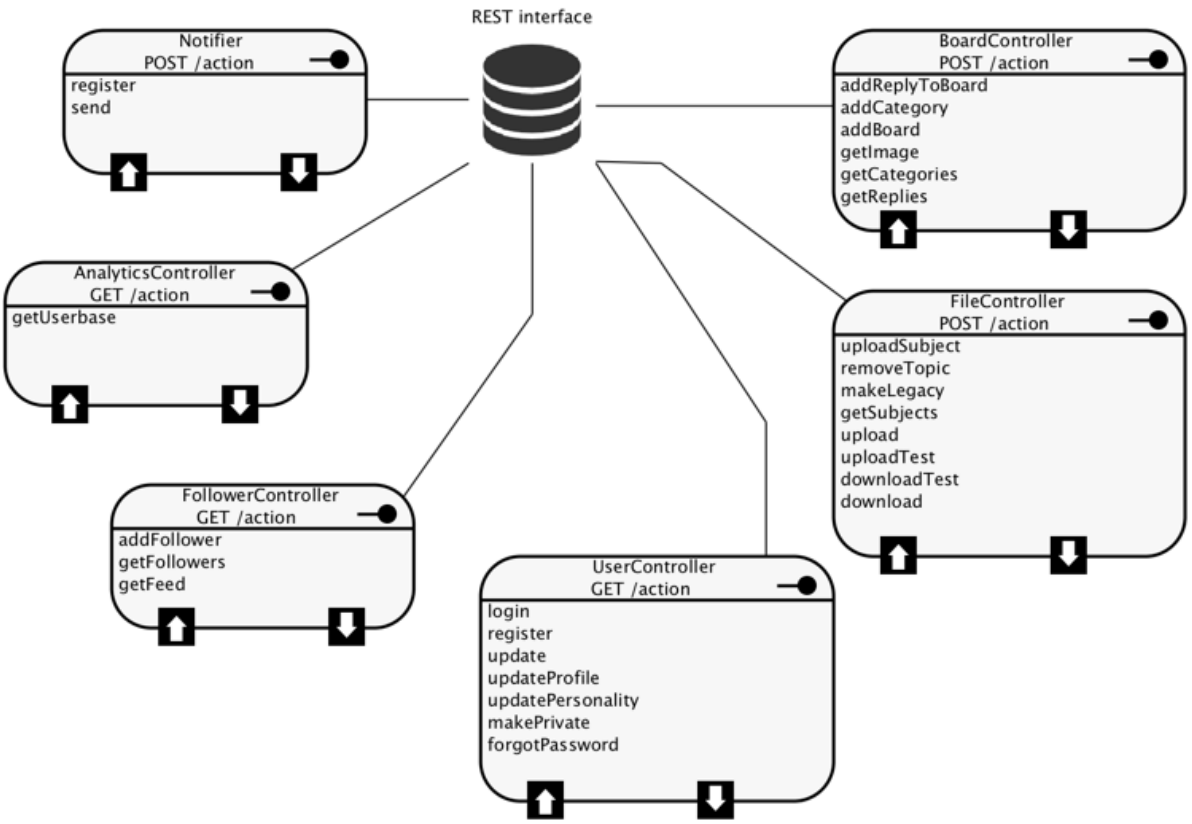


Figure 2.8 REST Interface Architecture

Fig. 2.8 shows the architecture which will be used for the implementation of the database. The REST Interface will direct the requests to the appropriate REST controller in which the request will be dealt with.

Preliminary references:

- Response 200 – Successfully performed operation
- Response 201 – Failed to perform operation
- Response 202 – Operation cannot be completed due to existing entry

Utilities class

This class will be used to grab various methods which will assist across the various classes. These methods will retrieve the information using Predicate technology introduced in Java 8.

```
public static boolean listHasUser(List<User> objects, String predicate){
    return objects.stream().filter(user -> user != null && user.getUsername().equalsIgnoreCase(predicate)).findFirst().isPres
}

public static User getUserForUsername(List<User> objects, String username){
    return objects.stream().filter(user -> user != null && user.getUsername().equalsIgnoreCase(username)).findFirst().get();
}

public static boolean listHasUserForEmail(List<User> list, String email){
    return list.stream().filter(user -> user != null && user.getEmail().contains(email)).findFirst().isPresent();
}

public static User getUserForEmail(List<User> list, String email){
    return list.stream().filter(user -> user != null && user.getEmail().contains(email)).findFirst().get();
}

public static boolean listHasSubject(List<Subject> subjects, String name){
    return subjects.stream().filter(subject -> subject != null && subject.getSubjectName().equals(name)).findFirst().isPresen
}

public static Subject getSubjectForName(List<Subject> subjects, String name){
    return subjects.stream().filter(subject -> subject != null && subject.getSubjectName().equals(name)).findFirst().get();
}
```

The methods shown above highlight some of the methods contained within the class. These all have static modifiers so that they are visible throughout the application without the need for creating a new object of this class every time.

Each REST controller is defined by the @RestController annotation, followed by a REST API mapping with the @RequestMapping annotation:

```
@RestController
@RequestMapping("/users")
public class UserController {
```

The REST interface for the UserController will be http://localhost:8080/users/

The "localhost" portion of the URL will be replaced with the servers IP address, the "8080" portion is the default port for Spring MVC to listen for requests on.

/users/register/{username}/{password}/{email}/{status}/{dob}

```
@RequestMapping(value="register/{username}/{password}/{email}/{status}/{dob}", produces = "application/json")
public String registerUser(@PathVariable String username, @PathVariable String password,
        @PathVariable String email, @PathVariable String status, @PathVariable String dob) throws IOException{

    User user = new User(username, password, email, "Not set", status, dob);

    if(!Utils.listHasUser(users, username)){
        if(users.add(user)){
            fileUtils.writeUserObject(user);
            graph.addVertex(new Vertex(graph.getVertices().size(), username));
            fileUtils.writeGraph(graph);

            return printer.printObject(new Response(200));
        }
    } else {
        return printer.printObject(new Response(202));
    }

    return printer.printObject(new Response(201));
}
```

Order of operation:

1. Check if the list contains the user with the username specified (uses Utilities class)

2. Add the user to the list if not present
3. Map the user to the Graph (allows followers)
4. Write the user object to the database local store (essential for crash handling)

/users/login/{username}/{password}

```
@RequestMapping(value="login/{username}/{password}", produces = "application/json")
public String getUser(@PathVariable String username, @PathVariable String password) throws JsonProcessingException{
    String output = "";

    User user = null;

    if(Utilities.listHasUser(users, username)){
        user = Utilities.getUserForUsername(users, username);
        if(user != null){
            String cryptedPass = Crypter.encryptPass(user.getPassword());
            if(cryptedPass.equalsIgnoreCase(password)){
                output = printer.printObject(user);
                user.setLastLogged(Utilities.getTimestamp());
            }
        }
    }

    if(user == null){
        output = printer.printObject(new Response(201));
    }

    return output;
}
```

Order of operation:

- Check the list has the username
- Get the users profile
- Encrypt the password of the retrieved profile and compare to password passed
- If the passwords match, return the user object and write last logged in time

Adding a connection within the graph between two Users

```
public void addConnection(String username, Vertex vertex){
    if(containsUsername(username)){
        Vertex storedVertex = getVertice(username);
        if(storedVertex != null){
            storedVertex.addConnection(vertex);
        }
    } else {
        Vertex newUser = new Vertex(vertices.size(), username);
        newUser.addConnection(vertex);
        vertices.add(newUser);
    }
}

public boolean containsUsername(String username){
    return vertices.stream().filter(user -> user != null && user.getUsername().equalsIgnoreCase(username)).findAny().isPresen
}

public Vertex getVertice(String username){
    return vertices.stream().filter(user -> user != null && user.getUsername().equalsIgnoreCase(username)).findFirst().get();
}
```

If the Graph already has the user's vertex, it will retrieve the vertex and add a connection to that vertex. If it does not contain a vertex for the user, it will create a new vertex, add a connection to the desired vertex and store it within the graph.

/files/download/{username}/{subject}/{filename}

```
@RequestMapping(value="download/{username}/{subject}/{filename}")
public void downloadFile(HttpServletResponse response, @PathVariable String username,
        @PathVariable String subject, @PathVariable String filename) throws JsonProcessingException{

    try {
        File subjectDir = fileUtils.getSubjectDirectory(subject);
        if(subjectDir != null && subjectDir.exists()){
            File downloadFile = new File(subjectDir + "/" +  filename + ".png");

            response.setContentLength((int)downloadFile.length());
            response.setContentType("image/png");

            FileInputStream ins = new FileInputStream(downloadFile);
            OutputStream outStream = response.getOutputStream();

            byte[] buffer = new byte[1024];
            int bytesRead = -1;
            while ((bytesRead = ins.read(buffer)) != -1) {
                outStream.write(buffer, 0, bytesRead);
            }

            response.flushBuffer();

            if(Utilities.listHasUser(UserController.users, username)){
                User user = Utilities.getUserForUsername(UserController.users, username);
                if(user != null){
                    user.addSubject(subject);
                    fileUtils.writeUserObject(user);
                }
            }
        }
    }catch(Exception ex){}
}
```

The above portion of code is the request code for the downloading of the topic files. First the directory is checked for any contents, if the contents are available it is written to a

FileInputStream which will be copied through to the response OutputStream. This will require the response to be flushed to the interface so that the data can be read by the client.

```
public static List<Topic> organiseTopics(String learnerType, Subject subject){
    List<Topic> topics = new ArrayList<>();

    if(validForArranging(learnerType)){
        LinkedList<Topic> queue = new LinkedList<>();
        for(Topic topic: subject.getTopics()){
            processTopic(topics, queue, topic);
        }
        if(queue.size() > 0){
            processQueue(topics, queue);
        }

        return topics;
    }

    return subject.getTopics();
}
```

The machine algorithm shown above is the implementation of the algorithm used for content arranging, below the processQueue function deals with processing.

```
private static void processQueue(List<Topic> topics, List<Topic> source){
    for(Topic topic: source){
        topics.add(topic);
    }
    source.clear();
}

private static void processTopic(List<Topic> topics, List<Topic> queue, Topic topic){
    int index = 0;

    switch(topic.getTag()){
        case "Title":
            topics.add(0, topic);
            break;
        case "Introduction":
            topics.add(1, topic);
            break;
        case "Leading explanation":
            queue.add(topic);
            index = queue.indexOf(topic);
            break;
        case "Example":
            queue.add(topic);
            if(index != -1){
                insertElementAtIndex(queue, index, topic);
            }
            break;
        case "Trailing explanation":
            queue.add(topic);
            if(queue.size() > 0){
                index = -1;
                processQueue(topics, queue);
            }
            break;
    }
}
```

```
public static String encryptData(String data){
    try {
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] passBytes = data.getBytes();
        md.reset();
        byte[] digestedValues = md.digest(passBytes);
        StringBuilder buffer = new StringBuilder();

        for(byte b: digestedValues){
            buffer.append(Integer.toHexString(0xff & b));
        }

        return buffer.toString();
    }catch(NoSuchAlgorithmException ex){}
    return null;
}
```

The screenshot above shows how the SHA-256 algorithm digest has been implemented. This method was used to encrypt all the sensitive information relating to the user.

```
/**
 * Created by EmileBronkhorst on 18/04/16.
 * Copyright 2016 Envision Tech LLC
 */
public class AppLibrary {

    private static AppLibrary instance;
    private static Context ctx;
    private List<Subject> subjects;
    private List<Subject> myLibrary;

    private SubjectDownloader downloader;

    private AppLibrary(){
        this.subjects = new ArrayList<>();
        this.myLibrary = new ArrayList<>();
    }
```

The above screenshot shows how the ApplicationLibrary was implemented. Only 1 instance of the library is allowed per session in which a method named downloadSubjects was called to retrieve the topics

```
@Override
protected List<Subject> doInBackground(Void... params) {

    try {
        String tempUrl = String.format(Network.NETWORK_URL, "files/getSubjects");
        URL url = new URL(tempUrl);

        HttpURLConnection conn = Network.getConnection(url);
        if(conn != null && conn.getResponseCode() == Network.RESPONSE_OK){
            String jsonData = Network.getPageData(conn.getInputStream());
            JSONArray subjectArray = new JSONArray(jsonData);

            if(subjectArray != null){
                for(int i = 0; i < subjectArray.length(); i++){
                    JSONObject subjectData = subjectArray.getJSONObject(i);
                    if(subjectData != null){
                        String subjectName = subjectData.getString("subjectName");
                        String description = subjectData.getString("description");
                        long contentSize = subjectData.getLong("contentSize");

                        Subject subject = new Subject(subjectName, description, contentSize);

                        JSONArray topicArray = subjectData.getJSONArray("topics");
                        if(topicArray != null){
                            for(int j = 0; j < topicArray.length(); j++){
                                JSONObject topicData = topicArray.getJSONObject(j);
                                if(topicData != null){
                                    String topicName = topicData.getString("topicName");
                                    String topicDescription = topicData.getString("topicTag");
                                    String fileName = topicData.getString("fileName");

                                    Topic tempTopic = new Topic(topicName, topicDescription, fileName);
                                    subject.addTopic(tempTopic);
                                }
                            }
                        }
                        this.subjects.add(subject);
                    }
                }
            }
        }
    }catch(Exception ex){}

    return subjects;
```

This portion of code demonstrates the downloading of the topics from the web application. First a connection to the server is established, in which page contents are parsed into a JSON array. The array is then iterated for entries of Subjects first (then topics within each subject iteration). Before beginning a new subject iteration, the contents of the constructed subject is added to a list and returned as a result at the end of the processing.

Sub section 6.3 User documentation

Super Tutor User Manual

Table of Contents

Chapter 1: Prologue

Super Tutor aims to cater for your educational needs by providing a service of personalisation tailored to your studying. Our advanced content management algorithms will allow you to revise the topics you wish in a way that best suits your learning preference. Whilst providing a tool for personalised learning we firmly believe in the ideology of individuals studying more efficiently when you are working with your peers, Super Tutor has its own community with which you can interact, keep track of or even shy away from the other members registered to the application.

Ever wanted to study topics which aren't offered at your institution? Well now you can, Super Tutor houses a large content library in which you can browse and explore the topics that appeal to you. If there is a topic that hasn't been added, simply make a request and our Content Managers will have a look into adding this. If the content is added to the application after you have downloaded it there is no need to update your application, you will receive a notification when new content is added and it will automatically be available to you for download… Yes, it's that simple!

What are you waiting for? Have a look at the various features that Super Tutor has to offer and indulge as you enter the world of personalised learning.

Chapter 2: System requirements

In order for Super Tutor to work to maximum efficiency and bring you a stable, reliable and speedy performance, there are a few things you need to ensure you have:

Android Kit Kat 4.4+

Your android device should be at a minimum Android Kit Kat (version 4.4 or newer). This is to ensure that Super Tutor is being viewed in the way that it was intended, with high quality graphics, saleable images, and a rich and intuitive user experience.

Required Permissions:

In order to get the most out of your experience, Super Tutor will require you to allow it to use the following permissions on your Android Device. Don't worry, any information used by the application is encrypted and securely stored on our database. We will never pass your information onto third parties.

- INTERNET
- Read/Write to device storage
- Access your Network state
- Google account information
-

Security

Security can be a touchy subject for many new users who are sceptical about the safety of their information. At Super Tutor we want you to use the application as comfortably as possible being reassured at every possible opportunity that your sensitive data is being stored securely.

Our application uses two types of encryption:

- The data sent from the application is encrypted using an algorithm developed by the National Security Agency (NSA – America)
- The data is sent over a secured connection encrypted by Secure Socket Layer (SSL) to ensure that the encrypted data is further encrypted

We firmly believe that using these two encryption types will give you a sense of data protection and we will guarantee that your information will ***never*** *be passed onto third parties.*

Chapter 3: Getting started

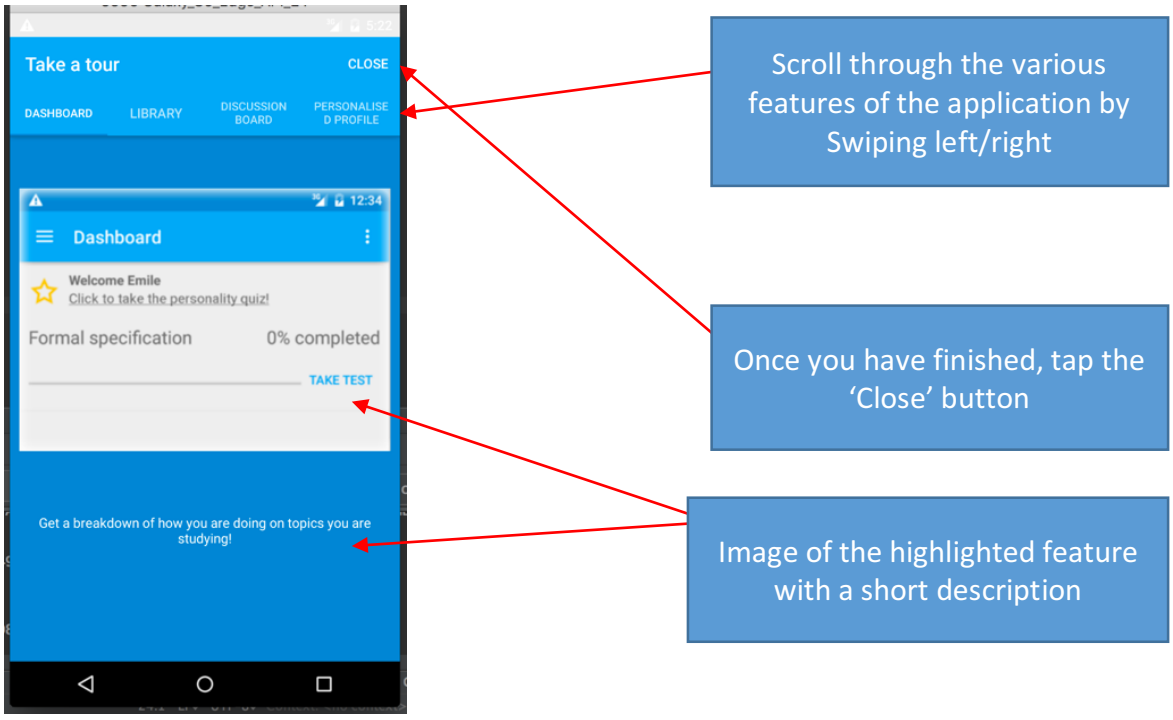
Before you can start using this wonderful tool you will need a few things:

- Active internet connection to download the application
- Register for an account

Just before we move to the next step, ensure you have downloaded the application by visiting the Google Play store and search for “Super Tutor”. Once you have downloaded the application, this next part of the manual will take you through the registration process and detail any other information which may be relevant to your concerns.

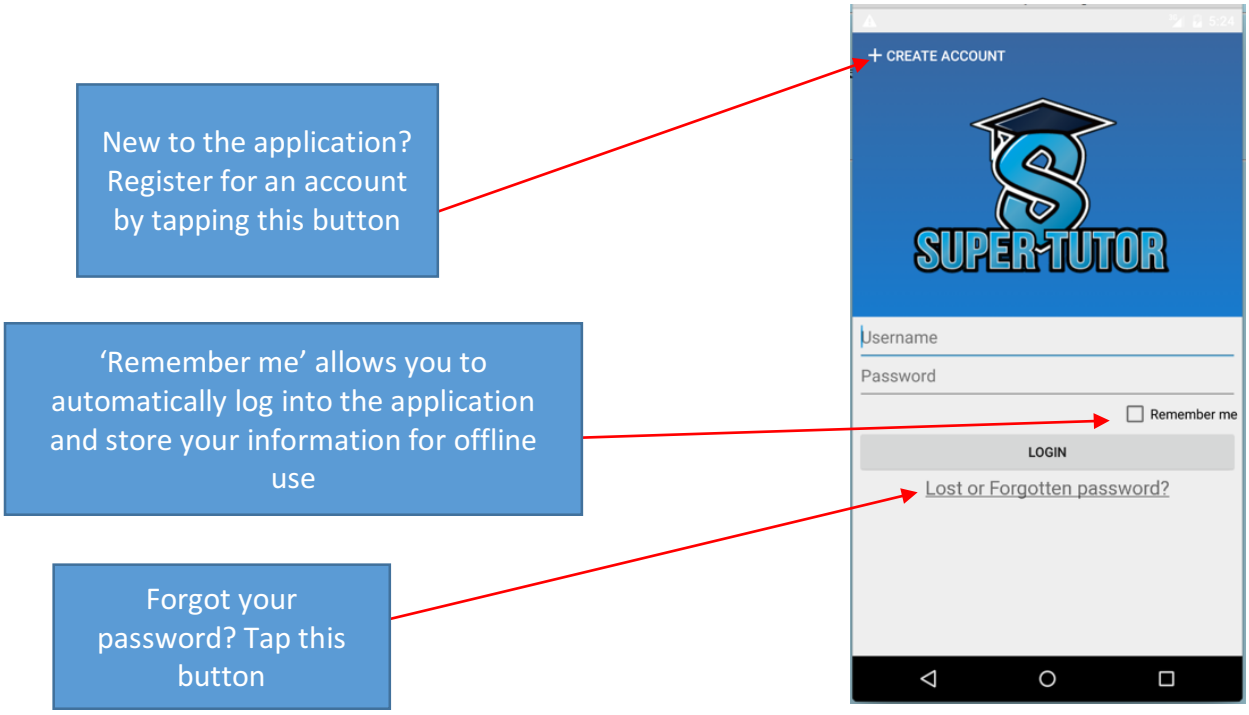
Step 1 – Application tour

First you will be greeted with the application tour, this allows you to browse the various features within the application without registering for an account. If you think that the application will be of benefit to you then continue further!



Step 2 – Login screen

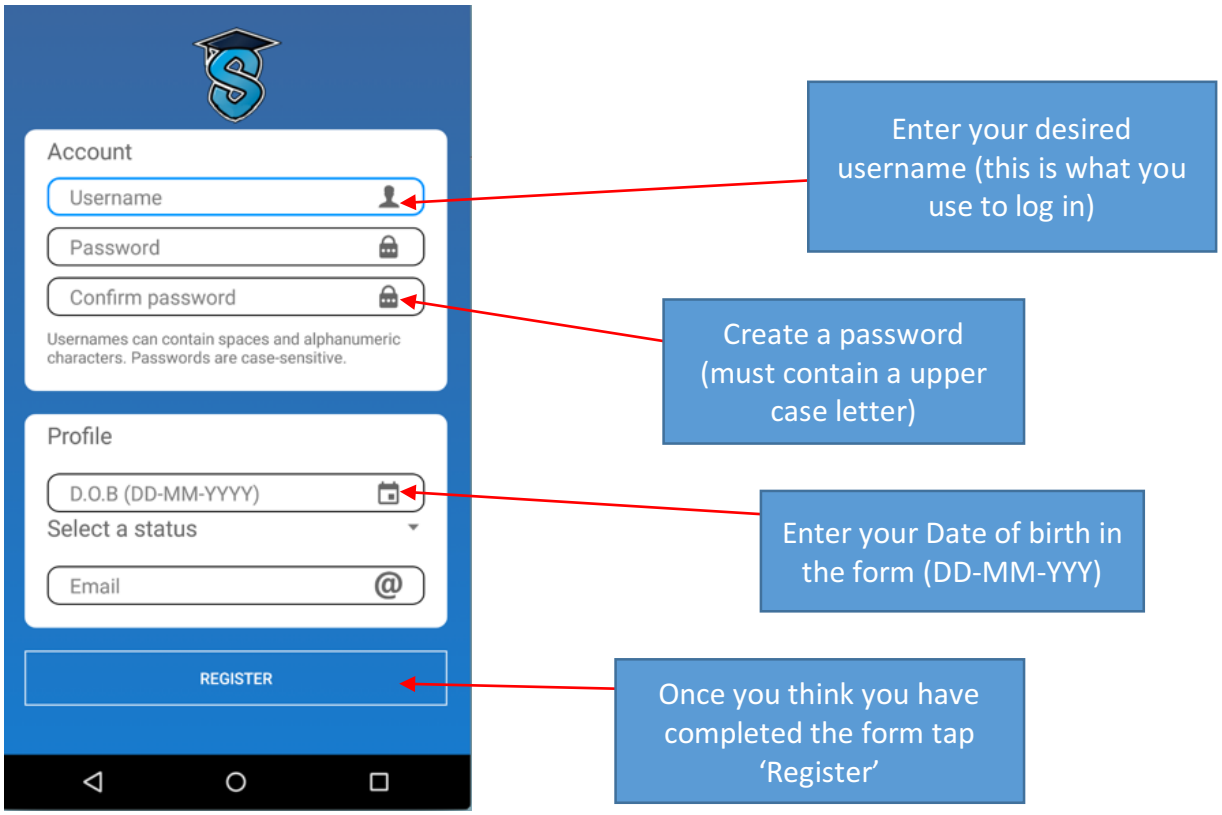
After clicking the ‘Close’ button on the navigation bar in the Application tour screen, you will be presented with our Material design login screen:



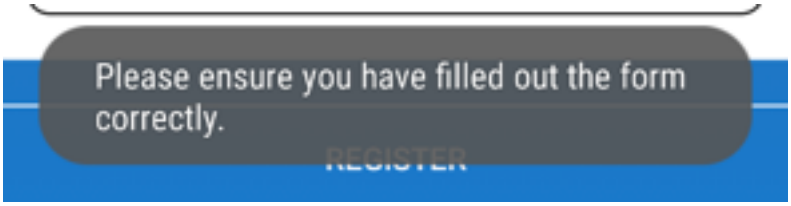
Note: If you are a returning user, enter your details and hit the ‘Login’ button. You can then skip this step!

First time users will require an account; this is the ‘Create account’ button shown above. Tap this and proceed to Step 3.

Step 3: Registration



If for any reason there is information which is incorrectly entered or you have not followed the specification required, you will be presented with a message asking you to fill out the form again following the criteria provided.



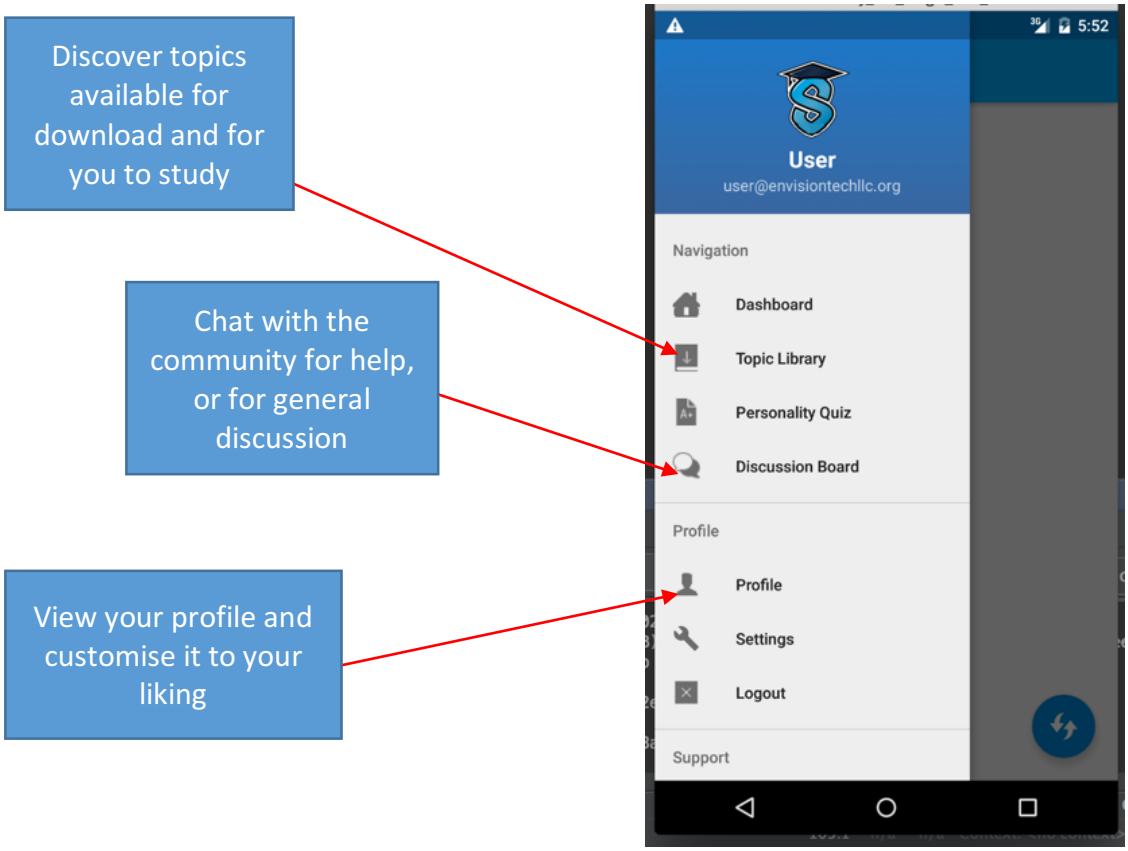
Once you have registered for an account and the application has accepted your registration, proceed to 'How to use the application' section of this manual.

Chapter 4: How to use the application

4.1 Online usage

Super tutor is developed in a way that will provide you content when you have an active network connection, but also deliver you offline content when you don't (providing you have selected the 'Remember me' option).

The online features are highlighted below:

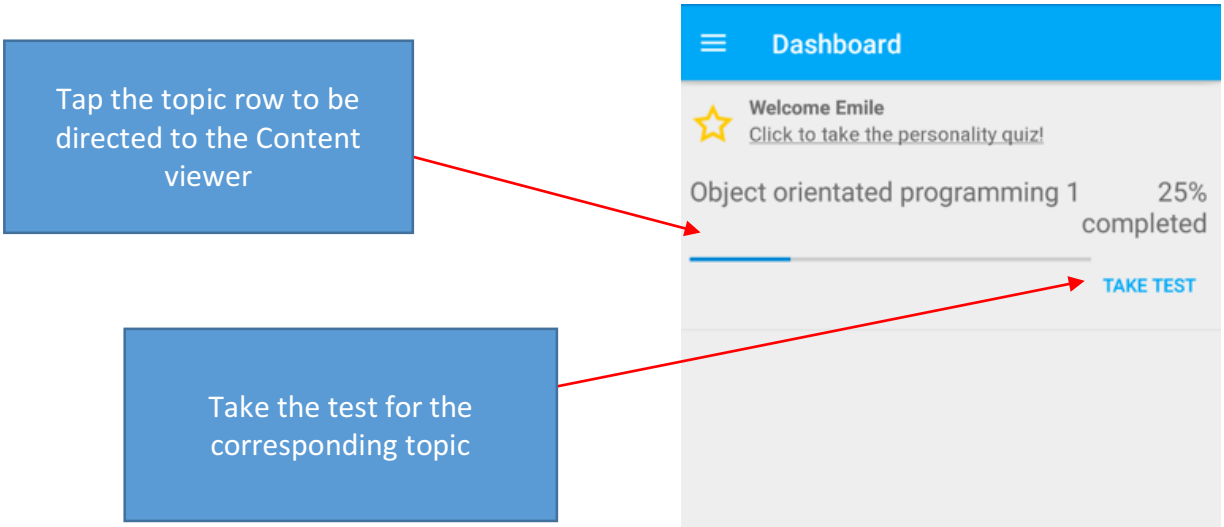


4.1.1 Downloading a topic

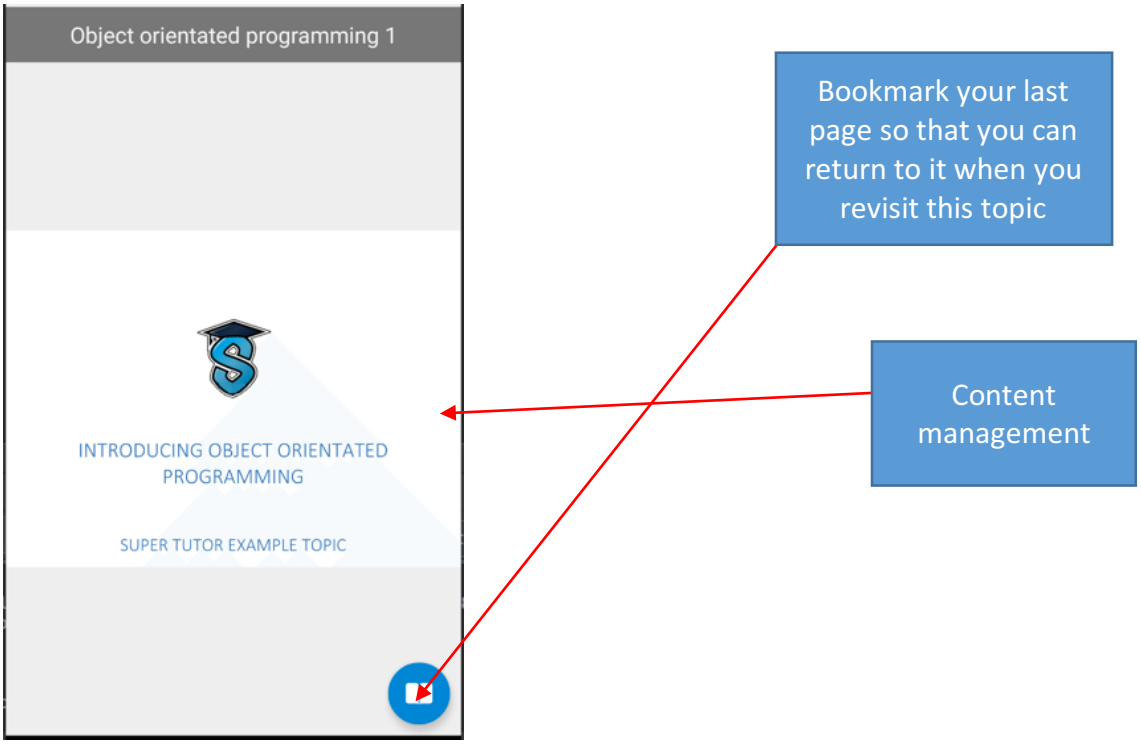
When downloading a new topic visit the Application library, once you have been presented with the topics available for download, ‘Tap’ the one which appeals to you and you will be presented with a ‘Download’ dialog.

Once you are presented with the dialog to download the subject, you can further confirm your decision to download the contents. If you select ‘Cancel’ you will return to the topic library, if you select ‘Download’ you will be presented with the progress of your download.

4.1.2 Dashboard



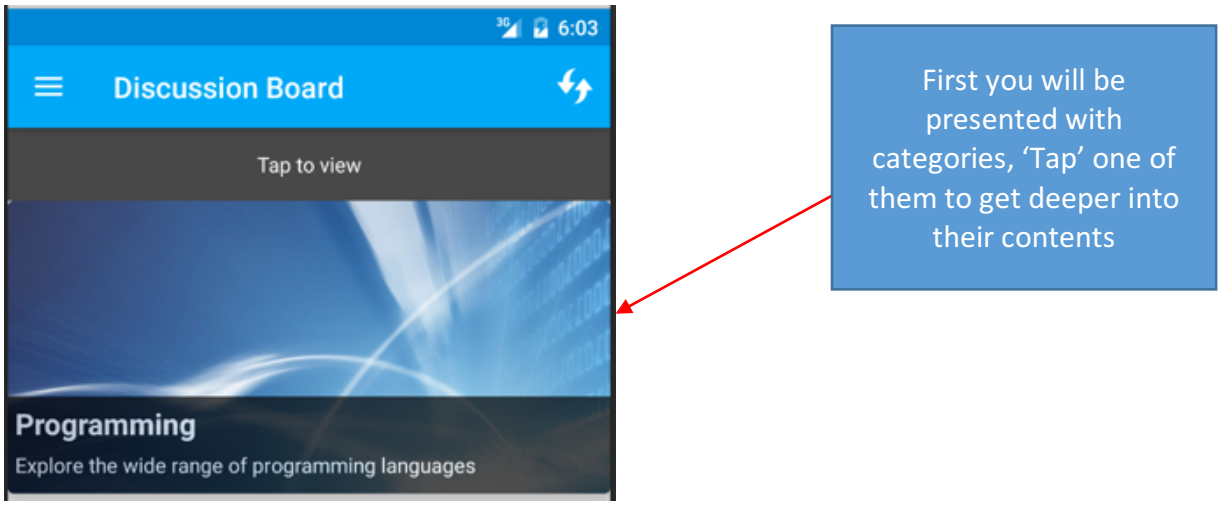
Once you tap the topic row to begin studying a subject you will be presented with the content viewer.



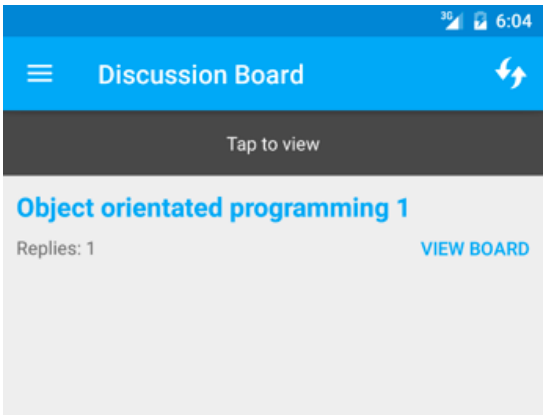
Our unique Artificial intelligence algorithm will take the contents of the topic you selected, re-order it in a way that suits your learning style and present it to you in this view so you can learn the way you were born to learn!

4.1.3 Discussion board

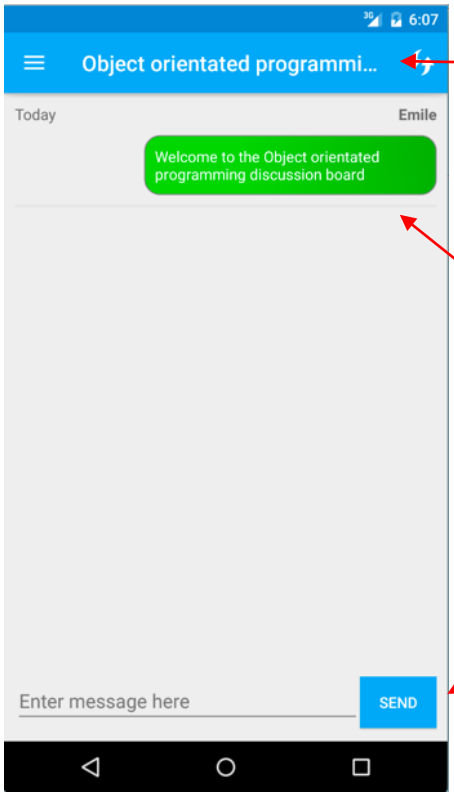
The Discussion board allows you to communicate with other registered users in a 'Chat style' board.



Next, you will see a list of boards available within that category to converse it. Click ‘View board’ on the board of your choice



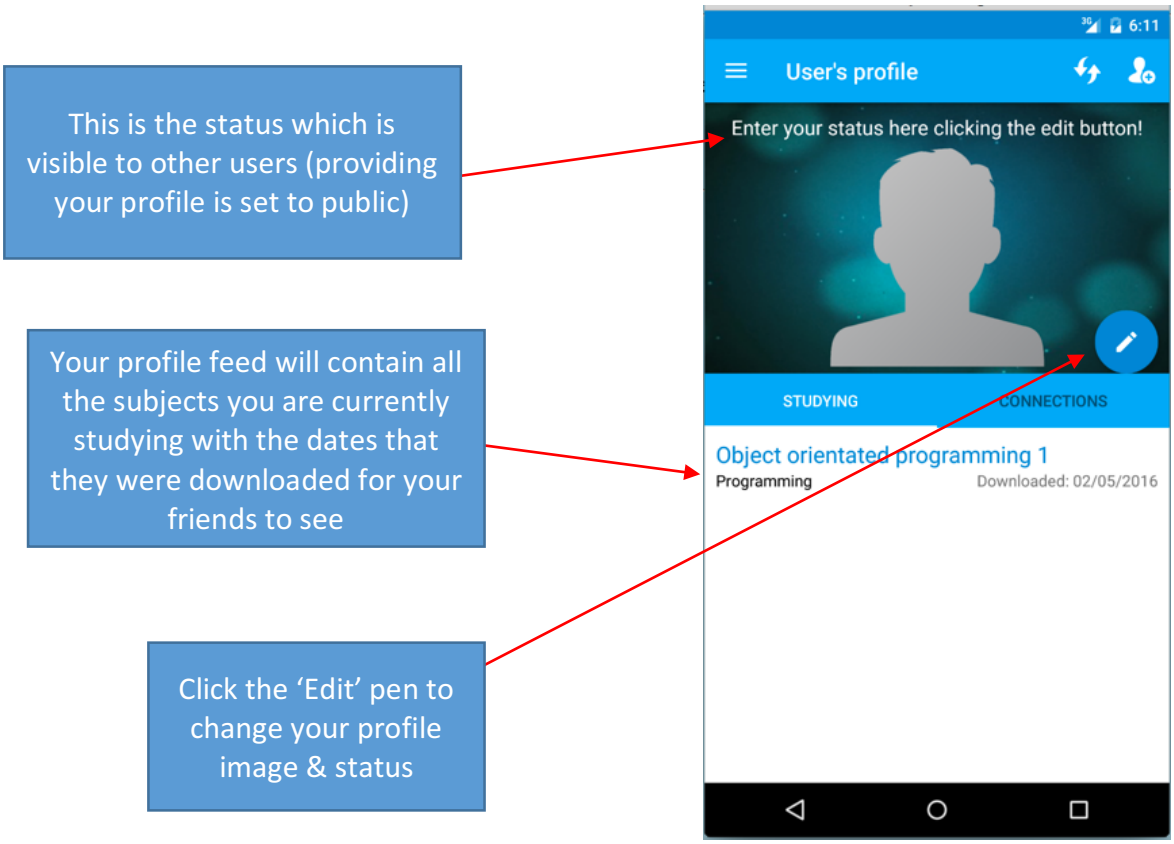
This will bring you to a chat-based application in which you can send/receive messages from users



As you can see this is the discussion board for ‘Object orientated programming’

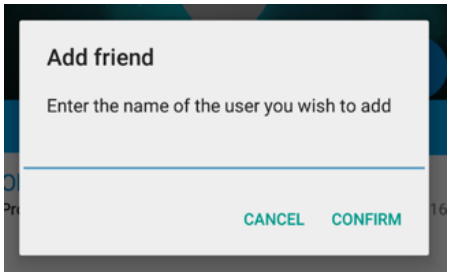
Enter your message here and tap the ‘Send’ button. It will automatically be sent to the server and add your message to the board contents!

4.1.4 Customising your profile

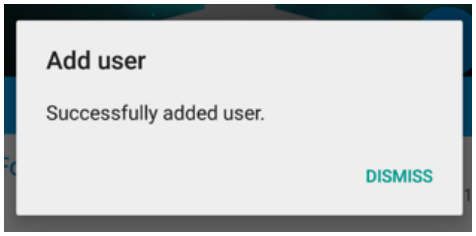


4.1.5 Adding friends

Adding users is simple, tap the  icon to present you the dialog for entering a friends name.

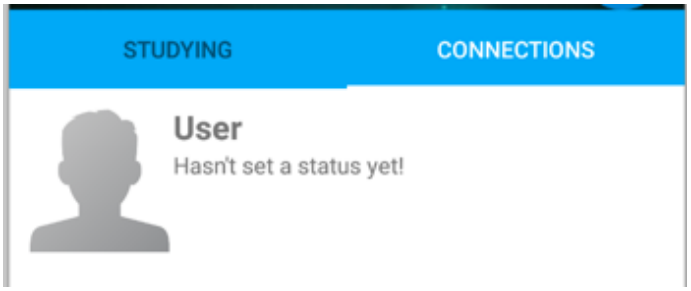


Enter the name of the user you wish to add and the application will give you a response!

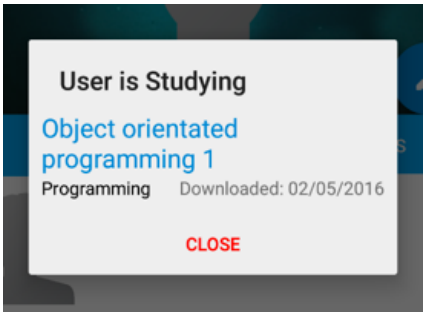


4.1.6 Connections

Want to see what your friends are getting up to? Swipe right to the ‘Connections’ tab in your Profile feed after you have added some friends



Tap the users profile name and you can see what they have been studying!



4.2 Offline usage

In order to access the ‘Offline’ feature of Super Tutor, you must select the “Remember me” option at the Login screen. This will allow you to access the following features offline:

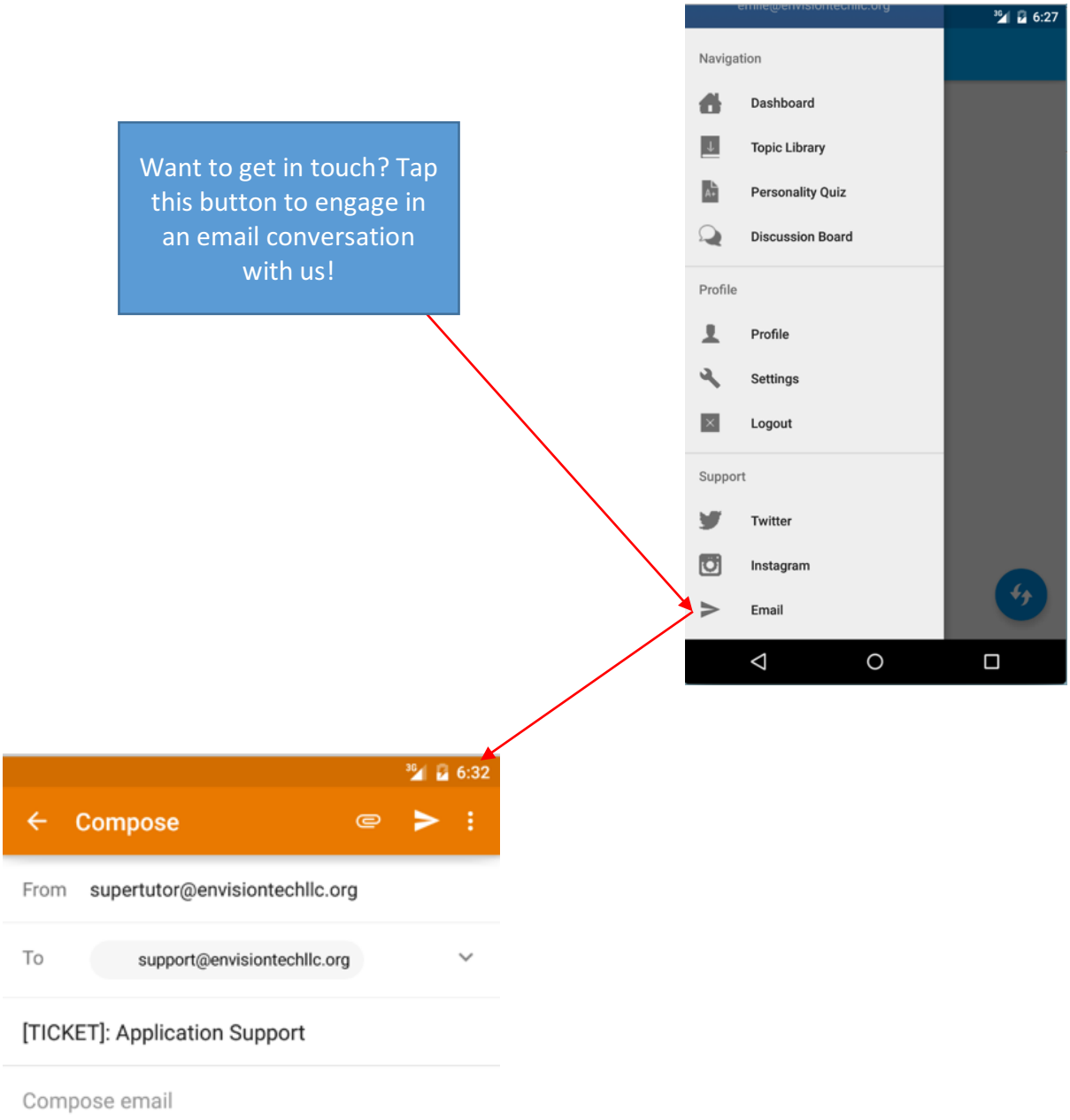
- Downloaded topics
- Personality quiz
- App tour
- Settings menu [partial offline usage]
- Support features e.g. Email, Twitter link, Instagram link

If a feature in offline mode is not available at the time i.e. changing password, it will automatically sync this new data once the application has been made available online again.

Chapter 5: Support

5.1 Email

Want to get in touch with us regarding a question you are having? Not a problem at all, via the navigation drawer within your Super tutor application you can contact us directly with your query:



5.1 Social Media

If emailing us is isn't exciting enough for you, then get in touch via social media. We have a variety of Social media accounts including:

- Twitter
- Instagram

If you still aren't satisfied with the support given in those 3 methods, then visit our developers website www.envisiontechllc.org and drop us a comment there!

Terms & Conditions

Super Tutor uses the following libraries:

- Apache Commons HTTP Client
- Google Cloud Messaging Server (GCM)
- JSON Simple

Super Tutor is in no way affiliated or owns any of these services. The libraries belong to their respective owners and is used in conjunction with the licences for which they are registered underneath (Apache License 2.0)

Super Tutor and its contents (par the aforementioned libraries) were developed solely by Emile Bronkhorst and any re-use of its contents or code base is strictly prohibited.

Super Tutor Desktop User Manual

Table of Contents

Chapter 1: System requirements

Super Tutor Administrative panel requires the following essential system components:

- Java 8 or later
- Stable internet connection

If you do not have the aforementioned requirement of Java, it can be downloaded from www.java.com (Owned by Oracle).

Optional system requirements (maximum performance) these features are not compulsory

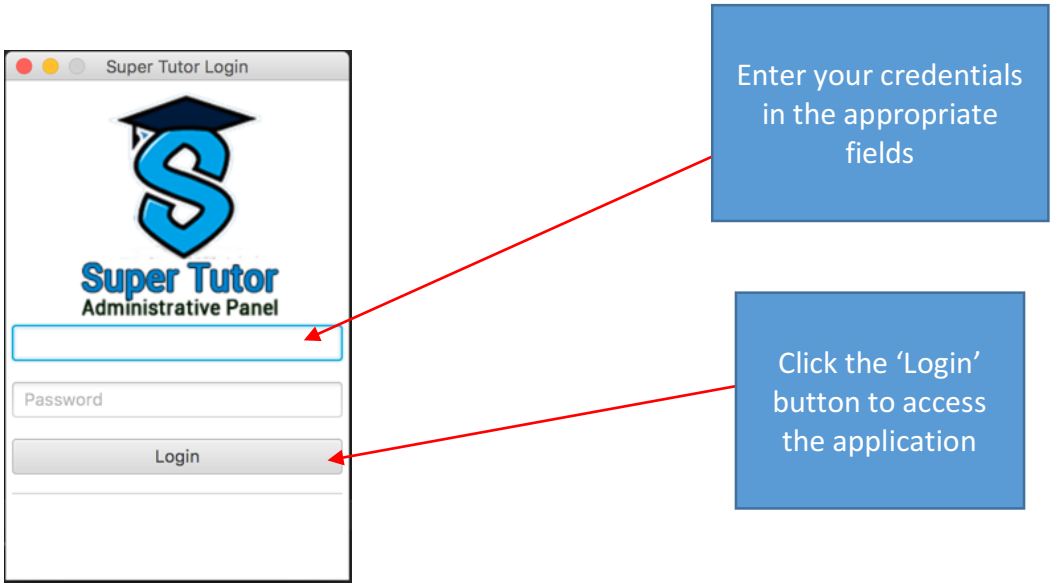
but will provide you with a better user experience when operating this application:

- Core 2 Duo processor
- 2GB dedicated DDR3 RAM
- 10-100MB's Network card

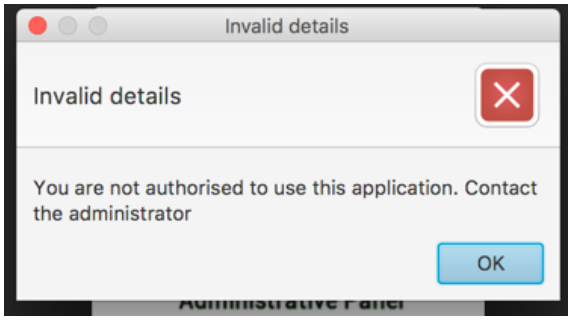
Chapter 2: Getting started

2.1 Login

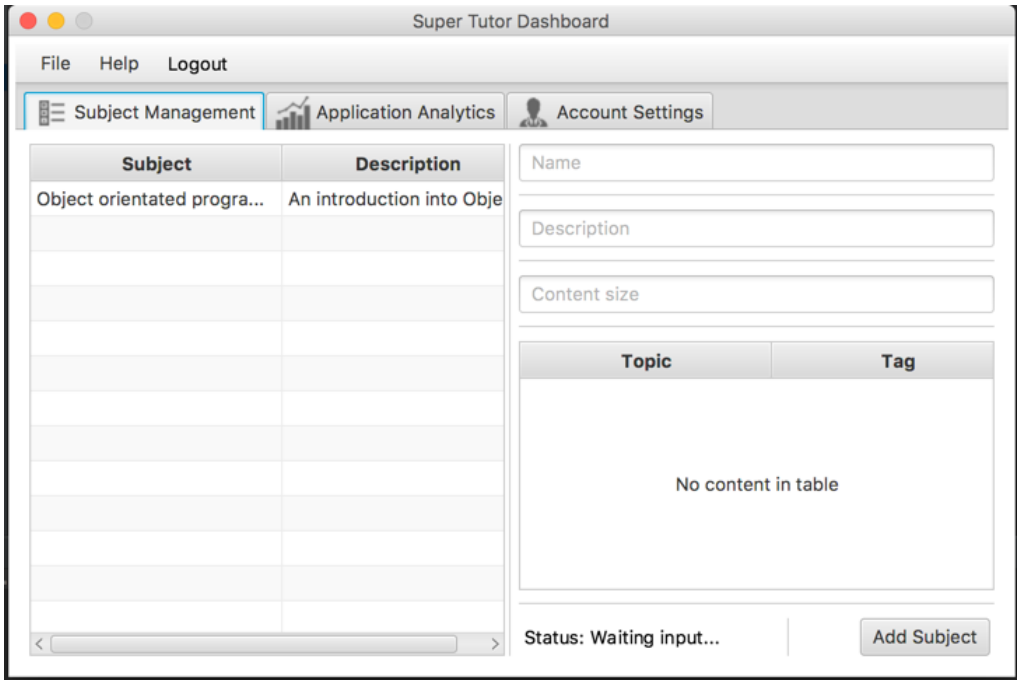
In order to use the applications features, you are required to log into the application with an account that has 'Content Manager' status. To obtain an account with this status level, contact your administrator.



If you are not able to use the application, you will be presented with the following message in which you should contact your System administrator:



2.2 Dashboard

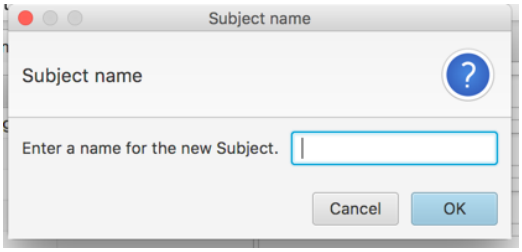


The main panel of the application is called the 'Dashboard'. This container will serve as a parent to several other aspects of the application; Subject management, Application analytics, Account settings. You will also be able to perform superior features to the individual tabs such as Sending notifications.

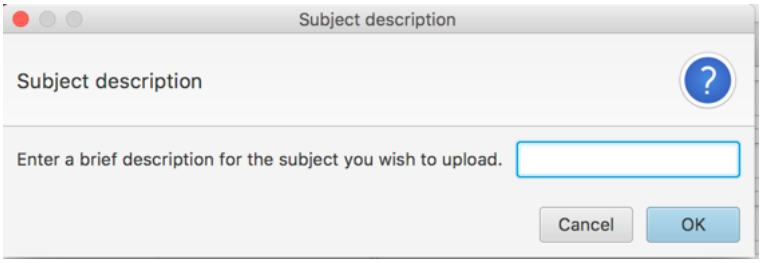
The next section will take you through the process of uploading a subject.

2.2.1 Uploading a subject

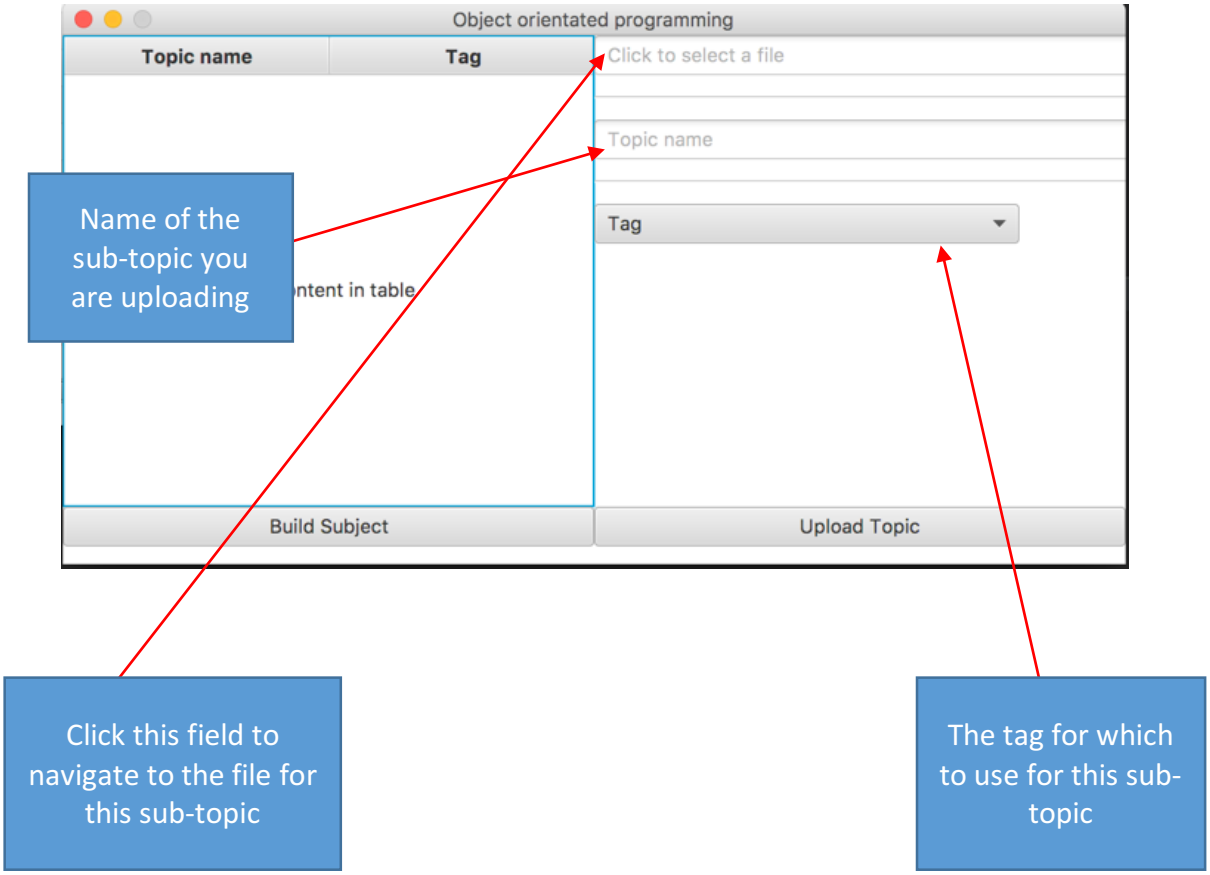
Once you have clicked the 'Add Subject' button you will be presented with a dialog to enter the name of the subject you wish to add:



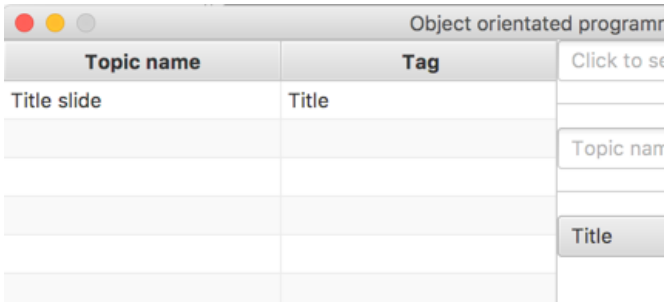
Once you have entered the name for the subject click ‘Ok’. For the purpose of this manual we will be uploading contents for a subject titled ‘Object orientated programming 1’. Once you have selected the ‘Ok’ button, you will be presented with a dialog prompting you for a description:



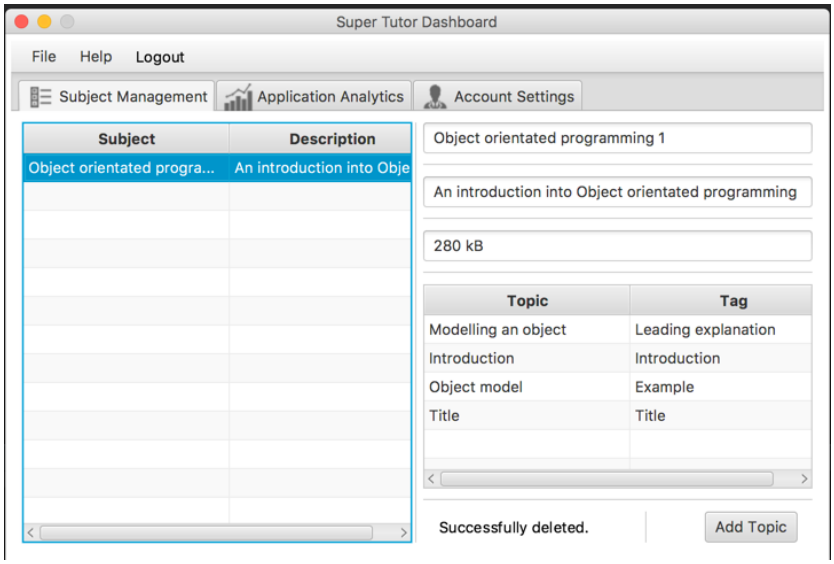
Enter the description for your subject and click ‘OK’ which will take you to the Topic manager for your newly created subject.



Once you have filled out the appropriate information (validation will double check you have), click the 'Upload Topic' button and your submission will be sent to the servers:



Continue to upload the topic contents until you have reached the end of your topic contents. Click 'Build Subject' to finalise your submission, in which you will be taken back to the Dashboard with your newly created topic.



Once at the Dashboard your newly created subject will be visible, in which you can click the entry in the table to add more topics, or view information about the contents of this subject.

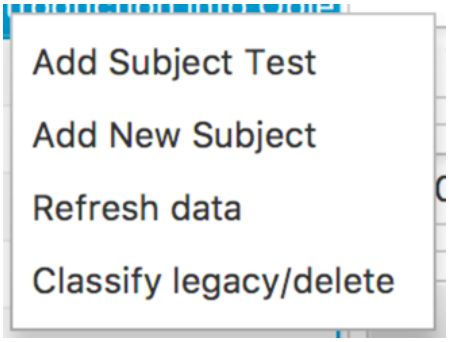


Figure 1.1 – Context menu

Fig. 1.1 shows the Context menu which is accessible by 'Right clicking' on the subject table.

As shown above, there are several extra options which have the following functionality:

- Add a Subject test for your selected subject
- Add a new subject to the database
- Refresh your subject entries

- Classify legacy/delete – This options moves the entire subject to the legacy folder which will render the subject unavailable to application users (this can be undone by System administrators)

2.2.2 Application Analytics

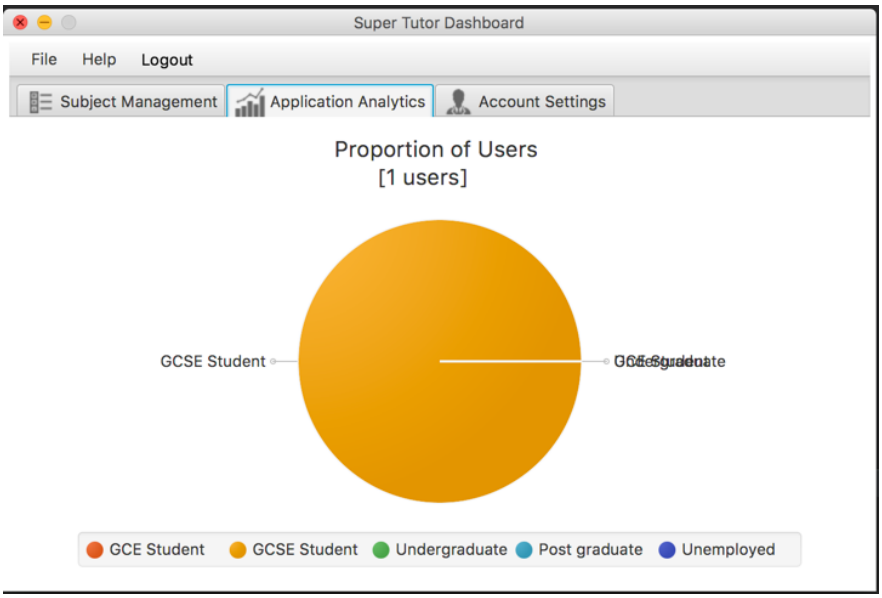
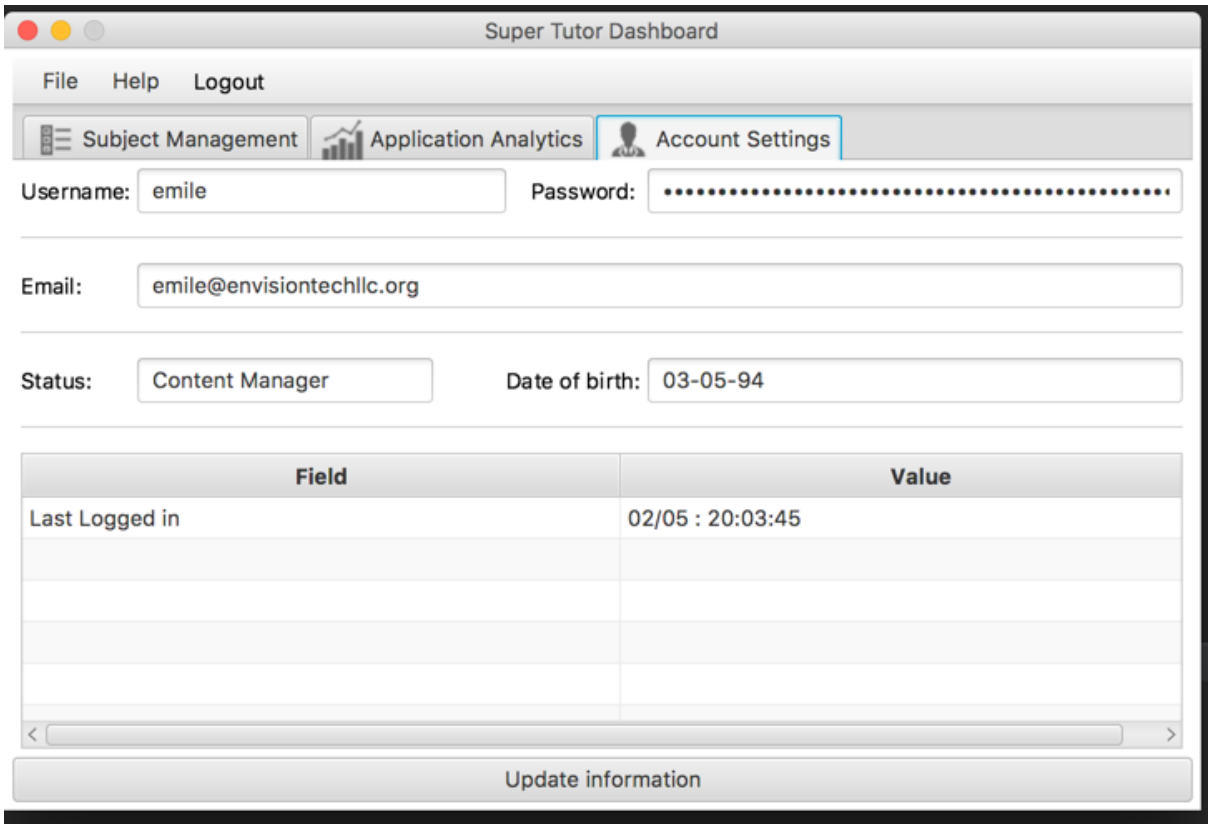


Figure 1.2 – Portion of users registered on the application

Fig. 1.2 Shows the portion of users who are currently registered within the application. In this case, there is only 1 user registered with a status of 'GCSE Student'

Super Tutor allows you to keep tracking of the portion of users using the application as well as the varying types of users. This can be later used in other aspects of the application to develop a greater product.

2.2.3 Account settings



The account settings page will allow you to view various information held on your account.

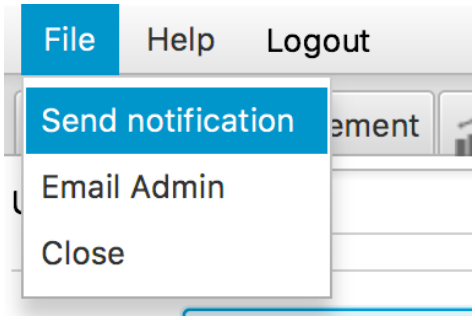
If you wish to at any time update your information, change the values in the fields and click the ‘Update information’ button.

Values that are modifiable:

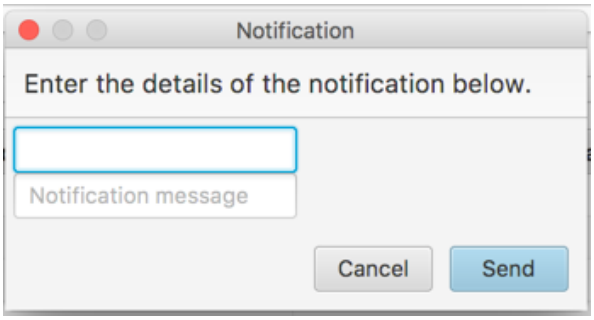
- Email address registered
- Password

As the application evolves various new statistics on users will be provided. This is where the ‘Field, Value’ table comes into play allowing for ease of new additions.

2.2.4 Push Notifications



The ‘File’ menu in the Desktop application is where the option for Push notifications can be found. Click ‘Send notification’ and you will be presented with the following dialog:



Field 1: The title of the notification you wish to send

Field 2: The message you wish the notification to contain

Once you have filled out the appropriate fields, click the ‘Send’ button and the notification will be sent to users registered on the application. A sample has been tested below by using the following parameters:

- Title: “New content added”
- Message: “Object orientated programming has been added to the library”

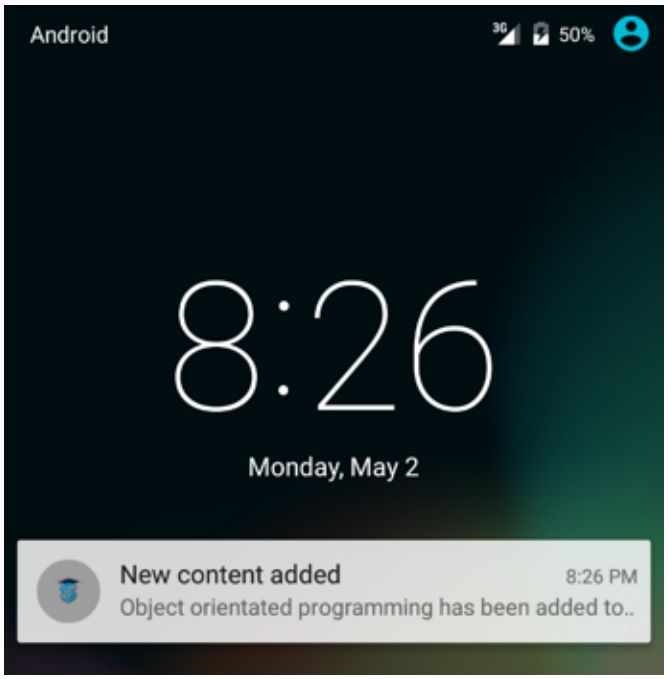
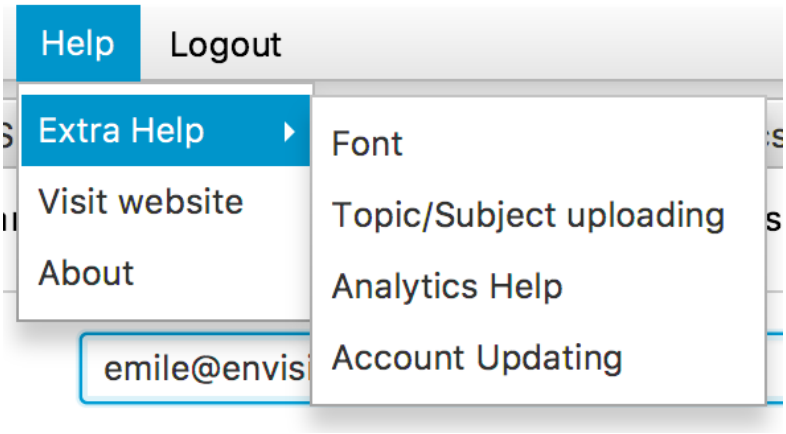


Figure 1.3 – Example of notification on users’ device

Chapter 4: Support

Super Tutor provides a variety of different methods showing you how to use the various different features on the application.



All aspects covered in this documentation is available through the 'Help' menu. If you have any further issues, please email your System administrator.

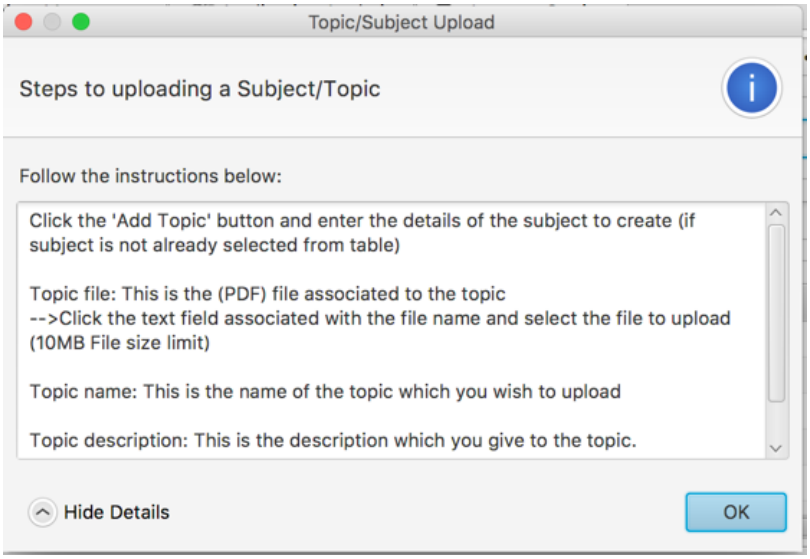


Figure 1.4 – Example of a Help dialog

Terms & Conditions

Super Tutor uses the following libraries:

- Apache Commons HTTP Client
- Google Cloud Messaging Server (GCM)
- JSON Simple

Super Tutor is in no way affiliated or owns any of these services. The libraries belong to their respective owners and is used in conjunction with the licences for which they are registered underneath (Apache License 2.0)

Super Tutor and its contents (par the aforementioned libraries) were developed solely by Emile Bronkhorst and any re-use of its contents or code base is strictly prohibited.

Chapter 7: Testing

7.1 Prologue

Based on the requirements highlighted in the SRS, the mobile application can be split into several components testing. These components will all be provided requirements for which to follow and test against (testing criteria). This portion of the document will highlight the testing plan devised as well as the results of the tests.

7.2 Testing plan

Fig. 2.9 Below shows an approximation of what the testing plan will be split into. Each increment will be detailed further with a diagram, explanation and test method.

Increment name	Approx. Requirements	Related increment
Splash screen	2	App tour, Login
App tour	3	Login
Login	9	Registration,
Registration	5	Dashboard
Dashboard	5	Navigation drawer
Library	3	Navigation drawer
Personality quiz	4	Navigation drawer
Profile	3	Navigation drawer
Settings dialog	3	Navigation drawer
Support	3	Navigation drawer
Navigation drawer	4	Dashboard, Library, Personality quiz, Profile, Settings dialog, support

Figure 2.9 Testing plan (increments devised)

The following increments will be tested against White box testing methods. This is due to the nature of the increments not having any UI interaction with the user and requiring internal functions to be tested.

7.2.1 Splash screen

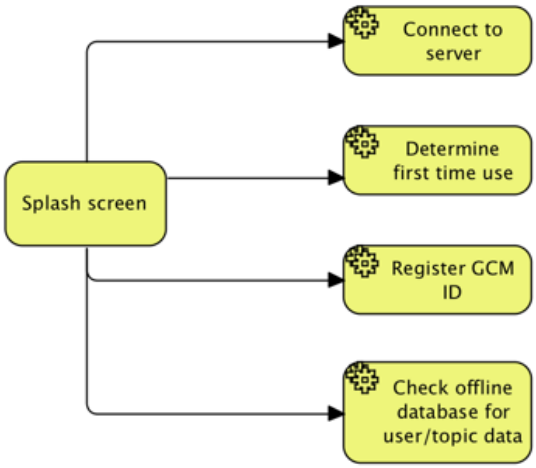


Figure 3.0 Splash screen increment

Fig. 3.0 shows which requirements will be packaged with this increment. Requirements 1.1 -> 1.3 addressed

7.2.2 App Tour

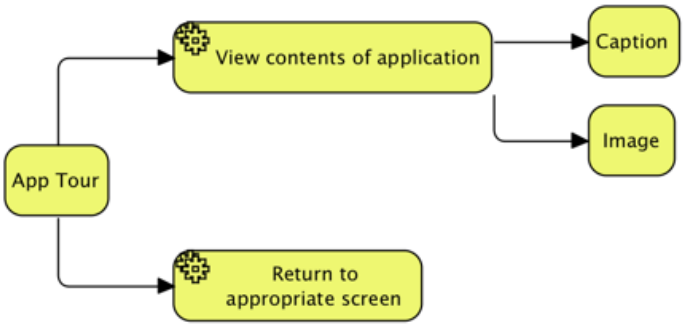


Figure 3.1 App tour

Fig. 3.1 shows the requirements package in the app tour increment. Requirements 2.1 -> 2.2 addressed.

The increments used in the next portion of the Testing plan will all incorporate Black box testing due to the nature of interactions with the User Interface. Refer to the Software Requirements Specification for referral to requirements.

7.2.3 Login screen

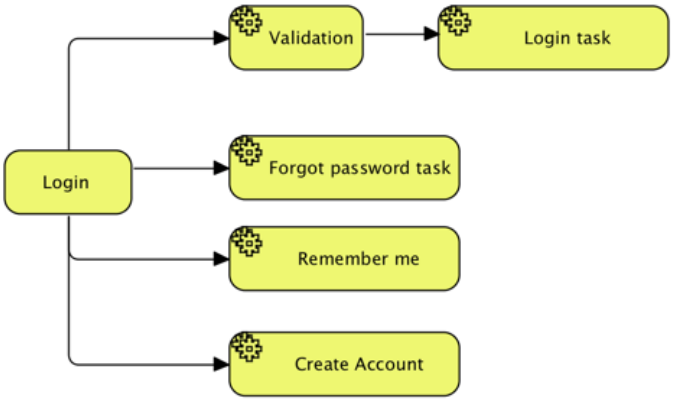


Figure 3.2 Login increment

Fig. 3.2 shows the requirements package with the Login increment. Requirements 4-4 -> 4.6 have been addressed.

7.2.4 Registration screen

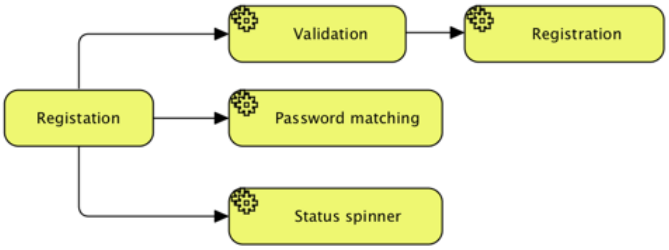


Figure 3.2 Registration increment

Fig. 3.2 shows the requirements packaged with the Registration increment. Requirements 3.1 -> 3.3 have been addressed.

7.2.5 Dashboard

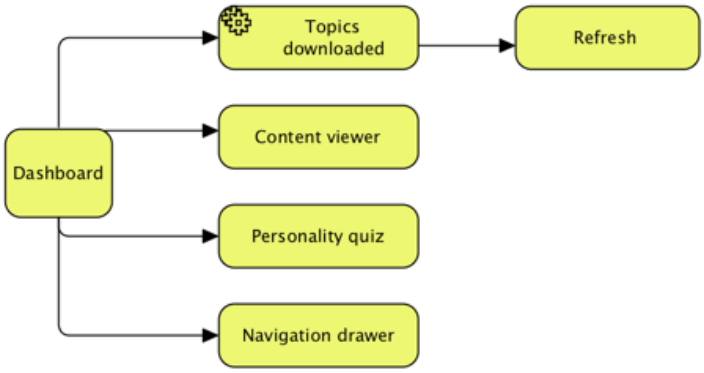


Figure 3.3 Dashboard increment

Fig. 3.3 shows the Dashboard increment, in this increment requirements 5.1 -> 5.5 have been included in addition to a new requirement 'Refresh'.

7.2.6 Application Library

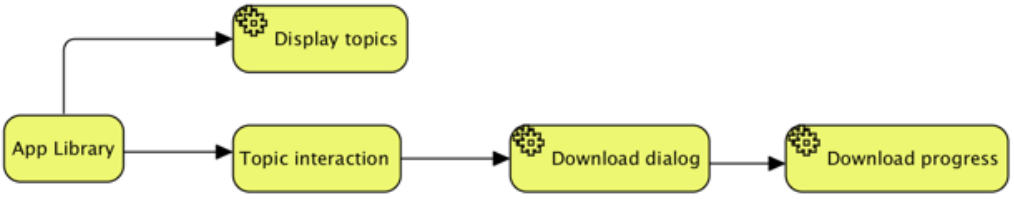


Figure 3.4 Application library increment

Fig. 3.4 shows the application library increment and the packaged requirements. In this increment, Requirements 6.1 -> 6.4 have been addressed.

7.2.7 Topic test

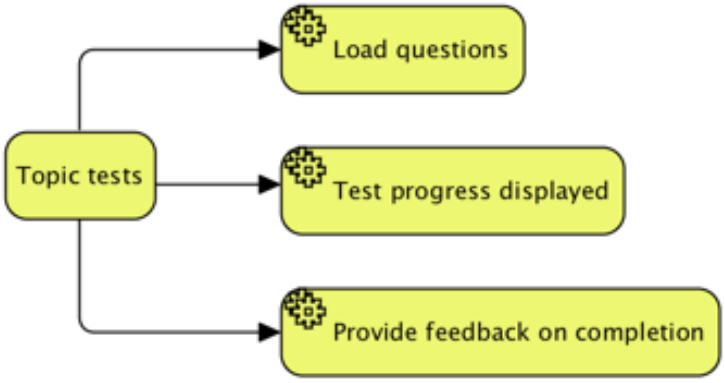


Figure 3.5 Topic test increment

Fig 3.5 shows the requirements packaged with the topic testing. Requirements 7.1 -> 7.3 have been addressed.

7.2.8 Profile page

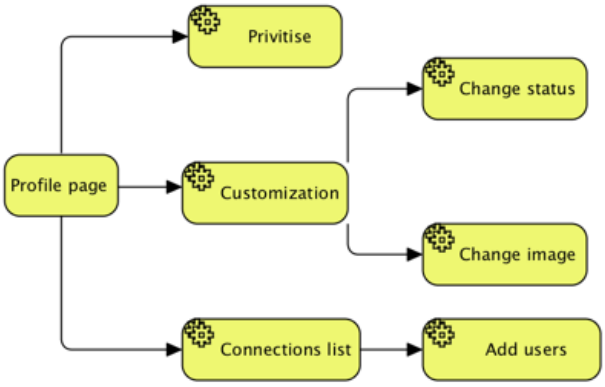


Figure 3.6 Profile increment

Fig. 3.6 shows the Profile increment with its requirements. Requirements 9.1 -> 9.2 have been addressed.

7.2.9 Personality quiz

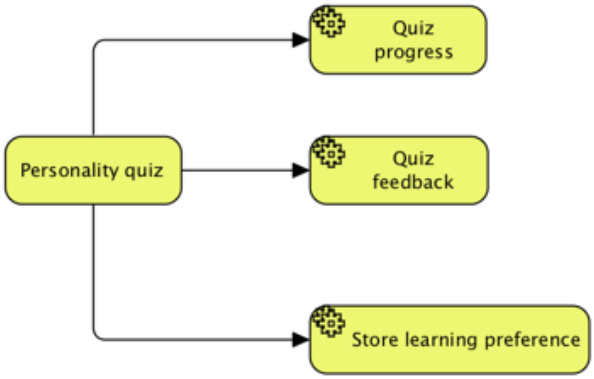


Figure 3.7 Personality quiz increment

Fig. 3.7 shows the requirements packaged with the Personality quiz increment.

Requirements 8.1 -> 8.4 have been addressed.

7.3 Test cases & Results

The first section this section will be exploring is White box testing, in accordance with those increments which require white box testing techniques (refer to Testing plan)

7.3.1 Splash screen increment

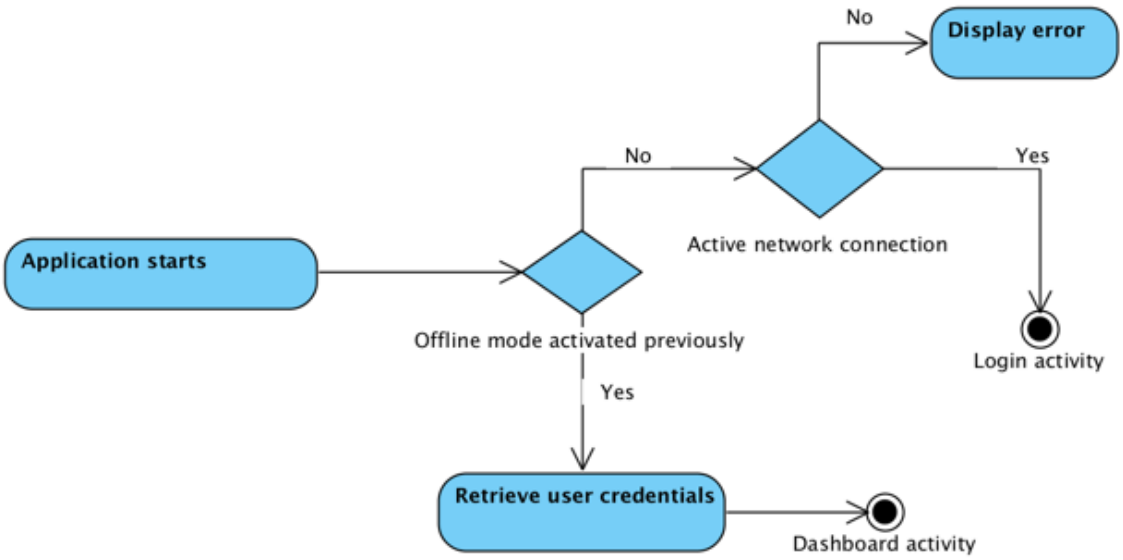
Assumptions made during this increment:

- User has requested the application to run

```
private void checkDatabase(){
    appData = AppContext.getContext();
    dbManager = new SQLManager(this);
    new GCMRegistration(this).execute();

    timer = new CountDownTimer(1100, 1000) {
        @Override
        public void onTick(long millisUntilFinished) {

        }

        @Override
        public void onFinish() {
            List<Subject> list = dbManager.getDownloadedSubjects();
            if(list != null && list.size() > 0){
                AppLibrary library = AppLibrary.getInstance();
                library.addSubjectsToMyLibrary(list.toArray(new Subject[list.size()]));
            }
            HashMap<String, String> userCredentials = dbManager.getUserCredentials();
            if(userCredentials != null){
                User tUser = new User(userCredentials.get("Username"), userCredentials.get("Password"),
                        userCredentials.get("Email"), userCredentials.get("LearnerType"));
                tUser.setStatus(userCredentials.get("Status"));
                tUser.setImageBytes(userCredentials.get("Image"));

                AppContext.getContext().setCurrentUser(tUser);
                Intent dashBoardIntent = new Intent(SplashScreen.this, SuperTutor.class);
                startActivity(dashBoardIntent);
                finish();
            } else {
                if(Utilities.checkConnectivity(SplashScreen.this)){
                    initApplicationContents();
                    return;
                } else {
                    AlertDialog dialog = Utilities.createDialog(SplashScreen.this, "Error", "Unable to connect to server, check your internet conn
                    dialog.show();
                }
            }
        }
    }.start();
}
```

Appendix R.

Control flow for Appendix R:



7.3.2 App tour increment

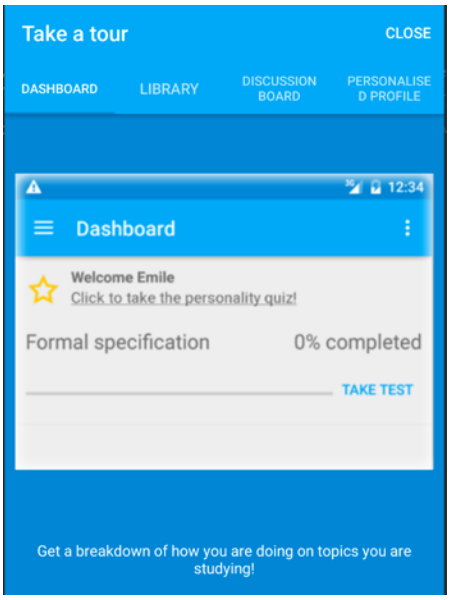
```
private void addFragments(TourPager pager){
    for(XTour tour: XTour.values()){
        pager.addFragment(tour.getTitle(), tour.getDescription(), tour.getImage(), tour.getFragment());
    }
}
```

The code above will iterate through the tour title, tour description, tour resource image reference and the generic fragment template:

```
public enum XTour {

    DASHBOARD("Dashboard", "Get a breakdown of how you are doing on topics you are studying!", R.drawable.dashboard_tour, new GenericTourFragment()),
    LIBRARY("Library", "Browse our large collection of subjects available for studying. When new content is added it will automatically be available he
    DISCUSSION_BOARD("Discussion Board", "Need help on a topic you are revising? Pop into the discussion board and get help from the community!", R.dra
    PROFILE("Personalised profile", "Want to chat with a friend or build your educational portfolio? Now you can with the easy to use Profile page!", R
    ;


    private String title, description;
    private int resourceID;
    private Fragment fragment;

    XTour(String title, String description, int resourceID, Fragment fragment){
        this.title = title;
        this.description = description;
        this.resourceID = resourceID;
        this.fragment = fragment;
    }

    public String getTitle() { return this.title; }

    public String getDescription() { return this.description; }

    public int getImage() { return this.resourceID; }

    public Fragment getFragment() { return this.fragment; }
}
```

Appendix S

Appendix S proof:



Appendix R2

Generic tour fragment used for the appendices above:

```
public class GenericTourFragment extends Fragment {

    private AppCompatActivity activity;

    private Bundle args;
    private TextView textDescription;
    private ImageView imageIcon;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle savedInstance){
        return inflater.inflate(R.layout.generic_tour_template, parent, false);
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);

        activity = (AppCompatActivity)getActivity();
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) { initFragment(view); }

    private void initFragment(View view){
        args = getArguments();
        if(args != null){
            textDescription = (TextView)view.findViewById(R.id.appTour_description);
            imageIcon = (ImageView)view.findViewById(R.id.appTour_image);

            textDescription.setText(args.getString("description"));
            imageIcon.setImageDrawable(getResources().getDrawable(args.getInt("image"), activity.getTheme()));
        }
    }
```

Appendix R3

This section will cover the increments sorted under the Black box testing phase (refer to the Testing plan for identification of which increments)

7.3.3 Login increment

Assumptions made during this phase:

- Sufficient network coverage for connection to server
- The user has never visited the application before, or does not have the 'Remember me' button selected on previous login

ID	Module	Input	Expected Result	Actual Result	Pass/Fail
1	Forgot password	emile@envisiontechllc.org	Email with account password	Email with account password	Pass-Appendix C
2	Create account	Tapped the 'Create account' button	Registration screen	Registration screen	Pass
3	Remember me	Check the 'Remember me' box	Activates offline mode	Created the offline database	Pass
4	Validation	Username: User Password: password	Incorrect login details	Incorrect login details	Pass-Appendix D
4	Validation	Username: User Password: user	Login task	Login task	Pass
5	Login task	N\A	Dashboard	Dashboard	Pass

7.3.4 Registration increment

Assumptions to be made for this increment:

- User has not already registered for an account
- User has an active Network connection
- User selected the 'Create Account' button in the Login screen

ID	Module	Input	Expected Result	Actual Result	Pass/Fail
6	Password matching	Password: temporary Confirm: arbitrary	Passwords do not match	Passwords do not match	Pass – Appendix E
6	Password matching	Password: temporary Confirm: temporary	Passwords match	Passwords match	Pass – Appendix F
7	Status spinner	'Select a status' clicked	Shows valid types	Shows valid types	Pass – Appendix G
8	Validation	Username: User Password: user Confirm: user D.O.B: 03-05-94 Status: GCSE Student Email: user@envisiontechllc.org	Dashboard	Dashboard	Pass – Appendix H
8	Validation	Incomplete form	"Incomplete form, ensure form is filled out correctly"	"Incomplete form, ensure form is filled out correctly"	Pass – Appendix I
8	Validation	Username: User Password: user Confirm: user D.O.B: 03-05-94 Status: GCSE Student Email: user.envisiontechllc.org	"Invalid email address"	"Invalid email address"	Pass – Appendix J

7.3.5 Dashboard increment

Assumptions to be made for this increment:

- User has registered for an account
- User has logged in (if not through registration)
- User has an active Network connection (if no local storage found)
- User has already downloaded a topic from the library

ID	Module	Input	Expected Result	Actual Result	Pass/Fail
9	Personality quiz	'Click to take the personality quiz' tapped	Personality quiz	Personality quiz	Pass
10	Navigation drawer	Tapped the Navigation icon	Navigation drawer shown	Navigation drawer shown	Pass
11	Topic display	N/A	Topics listed	Topics listed	Pass – Appendix K
12	Refresh	Refresh button clicked	"Successfully refreshed list"	"Successfully refreshed list"	Pass – Appendix L
13	Content viewer	Desired topic of study clicked	Content viewer	Content viewer	Pass – Appendix M

7.3.6 Application Library increment

Assumptions made during this increment:

- Users have an active Network connection
- Users have registered for an account
- Users have not downloaded all the topics
- Users have Navigated to the 'Library' navigation item

ID	Module	Input	Expected Result	Actual Result	Pass/Fail
14	Display topics	N/A	List of topics available	List of topics available	Pass – Appendix N
15	Topic interaction	Selected a topic to download	Download dialog	Download dialog	Pass – Appendix N1
16	Download dialog	Clicked ‘Cancel’	Returns to Topic library	Returned to Topic library	Pass
17	Download dialog	Clicked ‘Download’	Progress bar	Progress bar	Pass – Appendix N2

7.3.7 Topic test increment

Assumptions made during this increment:

- User has an active Network connection
- User has registered for an account
- User has downloaded a topic
- User has selected the ‘Take test’ option on the Dashboard

ID	Module	Input	Expected Result	Actual Result	Pass/Fail
18	Load questions	N\A	Test questions	Test questions	Pass – Appendix O2
19	Test progress displayed	N\A	Test progress visible (highlighted tab)	Test progress visible (highlighted tab)	Pass – Appendix O2
20	Provide feedback	Completed test	Dialog with score	Dialog with score	Pass – Appendix O3
21	Provide feedback	Incomplete test	Error dialog – incomplete test	Error dialog – incomplete test	Pass – Appendix O4

7.3.8 Profile increment

Assumptions made during this increment:

- Users have an active Network connection
- Users have registered for an account
- Users have logged into the application
- Users have selected the ‘My Profile’ item in the navigation menu

ID	Module	Input	Expected Result	Actual Result	Pass/Fail
22	Privatise	Username: User Settings dialog clicked, ‘Privatise profile’ selected	Does not appear in users directory	Does not appear in users directory	Pass – Appendix P0/P1/P2
23	Change status	“This is just an example status”, confirm clicked	Status stored	Status stored	Pass – Appendix P3
24	Change image	Selected an image on device	Profile image set to selected image	Profile image set to selected image	Pass – Appendix P4
25	Connections list	 selected	Add friend dialog	Add friend dialog	Pass – Appendix P5

7.3.9 Personality Quiz increment

Assumptions made during this increment:

- Users have registered for an account
- Users have logged into the application
- Users have selected the ‘Personality quiz’ navigation item

ID	Module	Input	Expected Result	Actual Result	Pass/Fail
26	Quiz progress	N/A	Shows the progress of the quiz	Shows the progress of the quiz	Pass – Appendix Q
27	Quiz feedback	Completed quiz	Dialog with results	Dialog with results	Pass – Appendix Q1
28	Store learner preference	Selected ‘Got it’ from results dialog	Dashboard	Dashboard	Pass – Appendix Q2

Chapter 8: Evaluation & Conclusion

Prologue

This section will discuss the Evaluation & Conclusion portion of the project which will involve discussion whether the Requirements stated in the Software Requirement Specification Document were all met, the Time targets for the project and how they were handled in situations which resulted in not meeting them as well as any future directions this project could pursue.

Sub section 8.1 Evaluation

8.1.1 Software Evaluation

This sub-section will highlight the requirements proposed in the SRS document and how they were/were not met. In the event that the requirements were not met, it will discuss a possible implementation of the requirement and briefly discuss why the requirement was not able to be met.

This portion will document the Functional requirements as proposed in the SRS document and how each of them were reflected in this project. Any references to individual Requirements should refer back to the SRS document.

Functional Requirements

R1 (Splash/Loading screen): Requirements 1.1 -> 1.3 were fully met. The application consists of a Splash screen which provides the following functionality:

- Query local database for offline information
- Check for network availability
- Automatically log the user in if offline data was obtained

R2 (Application Tour/Start-up screen): Requirements 2.1 -> 2.2 were fully met. The App tour consists of various images highlighting the important features of the application along with a brief description. Users are able to scroll through the various images & annotations by performing swipe gestures (left/right). Users are also able to close the App tour to return to the appropriate activity:

- If first time user – return to the Login screen
- Return to Dashboard (if app tour accessed via Settings dialog)

R3 (User Registration): Requirements 3.1 -> 3.5 have been fully met. The validator ensures that users enter a valid email, confirm that the confirmation password matches the password entered, confirms that password consists of a capital letter and the username is not already taken.

R4 (User login): Requirements 4.1 -> 4.6 have been fully met. R4.2 has a slight modification in the sense that the 'Login' button remains always interactable. This was thought to be a more realistic approach to interaction then to make the button interactable after details have been entered. The login process will connect to the login REST interface, verify the username & password supplied are matching that of the database (if incorrect feedback is provided), the user is displayed with a message of the activity taking place and all connections are accomplished asynchronously.

R5 (Main Dashboard): Requirements 5.1 -> 5.3 have been accomplished fully. The Dashboard will consist of a list of topics downloaded by the user. If none have been downloaded the user will be presented with a prompt to download some topics and add them to their collection. The Dashboard has a 'Refresh' feature to ensure that if the users connectivity is lacking, they are able to refresh their items. Users are also able to 'tap' the topics they wish to study to be presented with the Content viewer and have their learning preference clearly visible within this section of the application.

R6 (Application Library): Requirements 6.1 -> 6.4 have been fully met. Users are presented with a library containing all of the contents available on the web server. The user will then have the option to download a topics contents by selecting the relevant topic from the list. They will then be presented with a 'Download dialog' in which they can choose to 'Download' the topic or 'Cancel' their decision. Should they choose 'Cancel' they will be returned to the Topic library, however if they select 'Download', the application will display the Download progress of the topic.

R7 (Topic testing): Requirements 7.1 -> 7.3 have been fully met. Users are presented with the option to 'Take test' for the topics downloaded to their personal library. Once this option is selected, the user will be presented with a series of questions to complete. The questions are based on content relevant ONLY to the section studied.

R8 (Personality test): Requirements 8.1 -> 8.4 have been met to the best extent possible within the project time frame. The personality test consists of a series of multiple choice questions. Although these are sufficient enough for the purpose of this project, should future versions of the application arise, it would be better to have more interactivity in this portion of the application i.e. A test consisting of videos, images, audio etc. For the purpose of this project however, the multiple choice questions will suffice.

R9 (Profile page): Requirements 9.1 -> 9.2 have been met fully. Users are able to customise their profile providing a status and profile image which are both synced to the web server for use on other platforms. Users have the option to 'Privatise' their profile, making them appear as hidden to the community directory. Connections can be viewed through the profile page in the profile feed, if a user wishes to see what subjects their 'Connection' is studying, simply tap the Connection name and a dialog will present more information.

Non-functional requirements

Req. Code	Requirement	Completed
NF-1	Asynchronous loading	✓
NF-2	Progress bar updates	✓
NF-3	Security (SHA-256 Encryption)	✓
NF-4	Privacy (Privatise profile)	✓
NF-5	Network availability	✓
NF-6	Extendibility	✓
NF-7	Stability	✓
NF-8	Usability	✓
NF-9	Response time	✓

NF-3 (Security): Super Tutor uses the SHA-256 encryption digest algorithm. This provides for secure data transfer between the client and the web server.

NF-6 (Extendibility): Although not immediately clear how this has been achieved (discussed in more detail in Conclusion section), the framework has been developed in a way which will allow the platform to expand without the need for users to update/download anything new.

NF-9 (Response time): The application has been tested on several platforms (Kit Kat -> Marshmallow, Windows, Mac, Linux) and all applications run smoothly without hindrance. The UI presents responsive interactions and prompts dialogs where appropriate.

Overall majority of the Requirements specified in the SRS document have been met in full. The project has been completed to a sufficient standard fit for the objectives set forth in the preliminary phases.

8.1.2 Project evaluation

This section will highlight the under-pinning's of the project throughout its various phases, what went wrong, what went according to plan, what should have been done better, results on a questionnaire that was distributed to 10 users who tested the application and finally whether or not the Task schedule was met.

Strong agree (1)	Agree (2)	Somewhat (3)	Disagree (4)	Strongly Disagree (5)

Self-evaluation

These questions have been answered using the Likert scale above. This is for the sole purpose of judging the performance of the project

1. The requirements set out were met to a very high standard: **1**
2. Regular meetings were kept with project mentor on progress: **2**
3. The project was well thought out with a clearly defined plan: **1**
4. The testing plan was developed carefully: 2
5. The project was given ample time for each phase: **3**
6. Completion time of project was on schedule: **2**

From the self-evaluation form we can conclude that the project went according to plan for the most part. Goals and objectives were clearly highlighted and a clear testing plan was developed. With regards to the completion time of the project, aspects of the project could have been completed in a more professional manner, keeping track of the tasks that were left to do using a Task manager of some sort would have proven useful during this portion.

What went well?

Aspects of the project that were developed to a high standard with little-no trouble in between:

- Web server – With a large knowledge around architecture, development of the server was quick and easy with no difficulty during implementation
- Software Requirement specification was developed to a high standard with a clear understanding of the requirements from the onset
- Design principles were covered in detail with sufficient background research being conducted on all topics relevant
- Requirements analysis & gathering was completed to a sufficient standard to produce a working prototype. This was particularly due to the understanding gathered during this phase

What could have been done better?

The following are aspects of the project which could have taken a better direction:

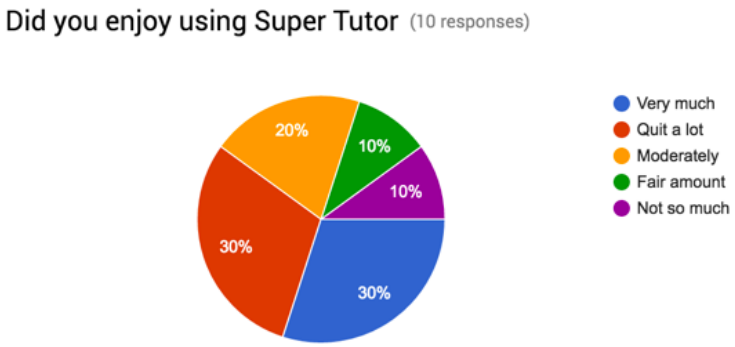
- Algorithm development – Although the currently implemented algorithm (Content management) is sufficient, it would be better to explore the realms of Machine Learning Algorithms to increase the reliability of this current AI and take into consideration factors such as categories of topics studied, time taken to study topics etc.

- Project timeline – Similar to the aforementioned point, the project could have been given a better schedule when it came to the development of the AI. This took longer than expected due to the amount of background research that was required into the topic before a sufficient enough algorithm could be developed.

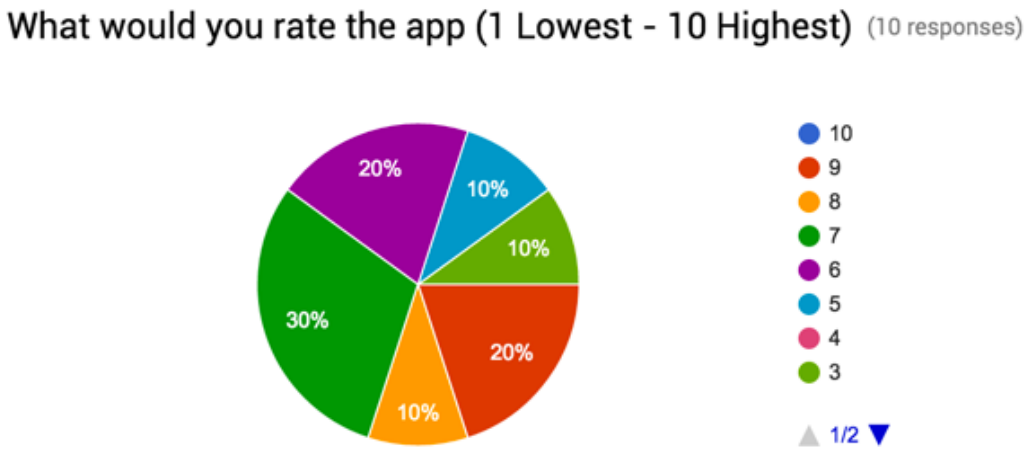
Super Tutor User feedback

In an attempt to gain actual user feedback from students with various different learning styles, Super tutor was distributed amongst level 4, 5 and 6 students. The application was tested and the users were shortly after given a short online questionnaire to fill out detailing their experience with the application (*Full results can be seen in Appendix T*).

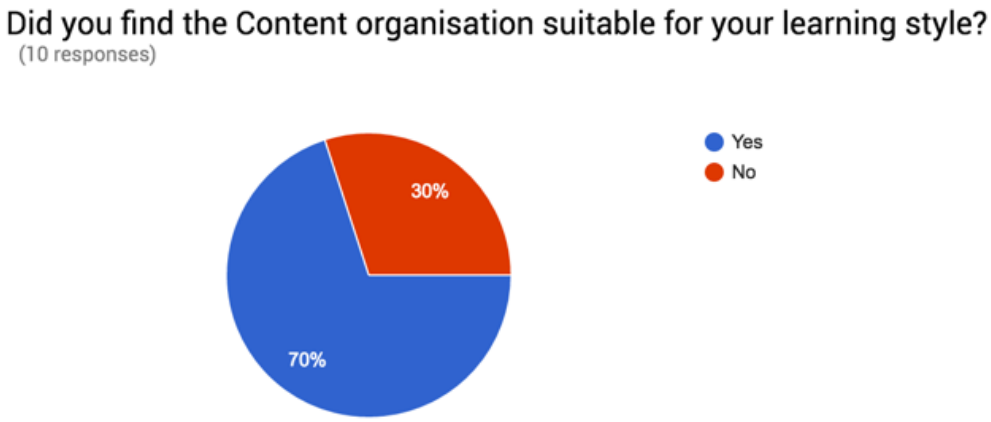
Sample breakdown (10 students); 3 at level 4, 4 at level 5, 3 at level 6



As you can see from the results put forth, the overall consensus was a pleasant result. From the graph we can determine that majority of users enjoyed interacting with the application and its various features. Users were also asked to 'Rate' the application for which the average Rating was **6.7**:



With regards to the AI integrated within Super Tutor, majority of users found the Content management algorithm sufficient for their learning preference:



Based on the results obtained from the after-use questionnaire, the responses from the users who have interacted with the application seem promising. Super Tutor is far from being perfect, but the reaction from stakeholders have left a positive notion on the direction of the project should it continue to be developed further.

8.2 Extendibility

The application developed serves as a Prototype for the requirements discussed in this document. The following sub-section will discuss how the application could possibly be further extended and how the framework exists for this extension to take place. Extension points will be depicted with the abbreviation E (where E stands for Extension Point)

E1: Content available – Super Tutor already has two topics which are available on the platform (Object orientated programming basics, Introduction to Formal Specification). As the application grows so will the demand for more topics (also true for user growth) in which a richer library of topics available will be required. The application would do better if there was a larger variety of topics available for users to download in converting potential users to registered users

E2: AI development – The current Artificial intelligence implemented in Super Tutor is fit for its purpose, however it would be of greater benefit to the users to have an algorithm which can 'Adapt' to their individual usage statistic e.g. If a user learns better through visual means, compile a list of website links based on the topic being studied of all diagrams, videos related to the topic. This will most certainly be a very difficult task to accomplish (through Machine Learning) but if done correctly, could change the educational tech industry

E3: iOS application – Due to the well devised design plan put forth in the Design phase, throughout the course of this project the framework for the iOS application has already been implemented. The web server is platform-independent meaning that it wouldn't matter what platform the server is being queried from, the only issue to be dealt with would

be parsing the data retrieved from the server on that platform. Wrappers used within the web server can be re-used for the iOS counterpart implementation

E4: Greater interaction between users – Currently users are only able to see one another's studied topics. A future implementation of the 'Connections' feature would be to allow users to see each others topics, test results, and challenge them to take a test that they may have scored highly in and attempt to beat it. The framework for such technology has already been put in place by developing a wrapper called 'ProfileFeed' which is interpreted by the Android application and the corresponding fields are displayed in the users feed. Future extensions can expand on this wrapper allowing for a greater variety of interaction between users

8.3 Project comments

Super tutor has been an incredible application to work on both from a developer stand point and to see how users interact with products in various different ways. I have learnt a significant amount from both developing this project and through the background research which was required to develop the project.

Some of the Software skills which were expanded on:

- Vast addition of Android knowledge (AsyncTask, Activity lifecycle, SQLite Database)
- Introduction of Spring MVC DBMS (Thoroughly fluent in REST API development)
- Java security (Knowledge of the encryption algorithms and their history)
- Java 8 (Expanded on prior knowledge in terms of progress binding etc.)

Some Project skills expanded on:

- Requirements Gather & Analysis (Defining clear requirements)
- Design phase (appropriate background research, design decisions)
- Software methodologies (in depth knowledge as well as knowledge of new technologies which have arisen from old methodologies i.e. Agile/Scrum development)

This has been a highly successful project for myself and the things I have learned during its development. I hope to develop this platform to a greater level of interaction and complexity in the near future and attempt to build on the points which were put forward as extensions. Artificial intelligence is an ever growing topic in the tech industry today and it has sparked a new interest for me into learning more about automating the various aspects in our daily lives.

After all, technology of today is the future of tomorrow!

References

Barbe, W.B. (1981) *What we know about Modality strengths*. Available at: http://www.ascd.org/ASCD/pdf/journals/ed_lead/el_198102_barbe.pdf (Accessed: 20 March 2016).

Administrator, B. (2012) *Learning styles - Psykologi*. Available at: https://hjernebark.atlassian.net/wiki/display/PSYK/Learning+styles (Accessed: 21 March 2016).

Wilson, V.A. (1998) *Learning how they learn*. Available at: http://files.eric.ed.gov/fulltext/ED427017.pdf (Accessed: 21 March 2016).

Norman, N. (1998) *Mobile: Native Apps, web Apps, and hybrid Apps*. Available at: https://www.nngroup.com/articles/mobile-native-apps/ (Accessed: 2 April 2016).

MobDevApp (2015) *Understanding the different types of mobile applications*. Available at: http://www.mobdevapp.com/understanding-the-different-types-of-mobile-applications/ (Accessed: 2 April 2016).

Social hunt (2013) *Type of mobile Apps - native App, hybrid App, web applications*. Available at: http://www.socialhunt.net/blog/types-of-mobile-app/ (Accessed: 2 April 2016).

Wikipedia (2016) 'Android version history', in *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Android_version_history (Accessed: 10 April 2016).

118, A. (2015) *What's the most popular version of Android?* Available at: http://www.alphr.com/smartphones/1000324/whats-the-most-popular-version-of-android (Accessed: 10 April 2016).

Oracle (1995) *What is an object? (the Java™ tutorials > learning the java language > object-oriented programming concepts)*. Available at: https://docs.oracle.com/javase/tutorial/java/concepts/object.html (Accessed: 15 April 2016).

Jain, K., Srivastava, T., Jain, A. and Saraswat, M. (2016) 'Machine learning basics for a newbie',*Business Analytics*, 28 April. Available at: http://www.analyticsvidhya.com/blog/2015/06/machine-learning-basics/ (Accessed: 20 April 2016).

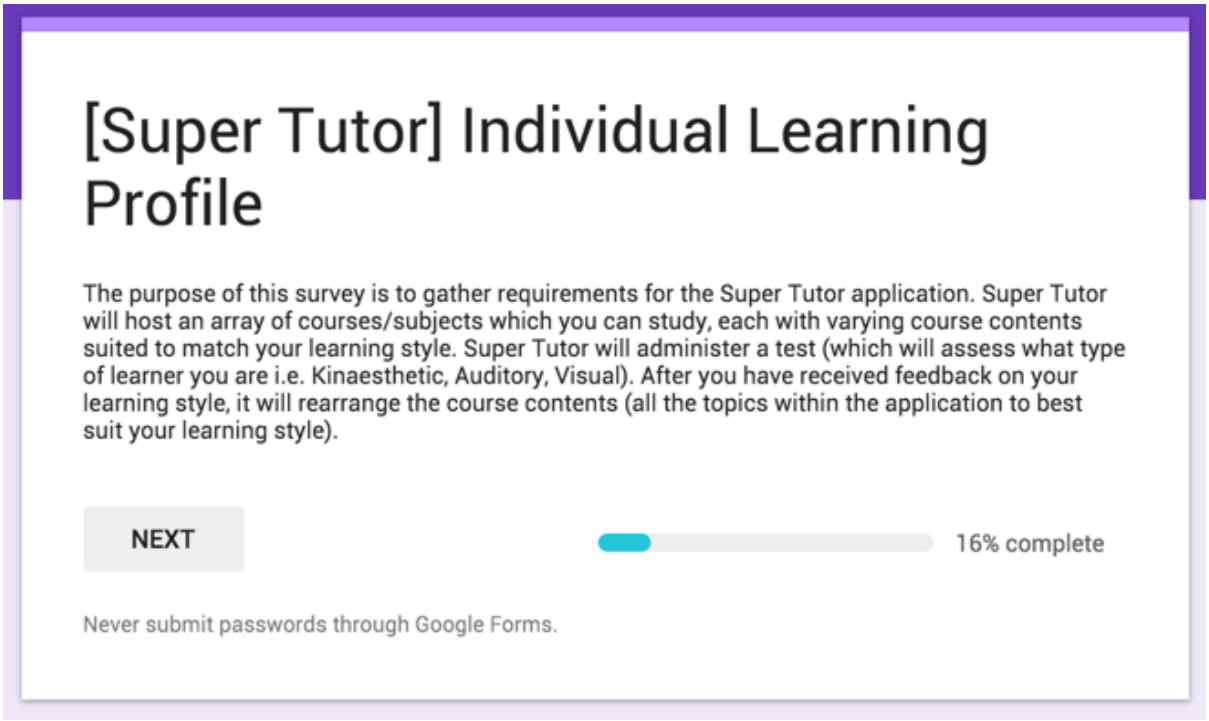
McCrea, N. (2010) *An introduction to machine learning theory and its applications: A visual Tutorial with examples*. Available at: https://www.toptal.com/machine-learning/machine-learning-theory-an-introductory-primer (Accessed: 20 April 2016).

2014, P. (2002) *Undirected graphs*. Available at: http://algs4.cs.princeton.edu/41graph/ (Accessed: 24 April 2016).

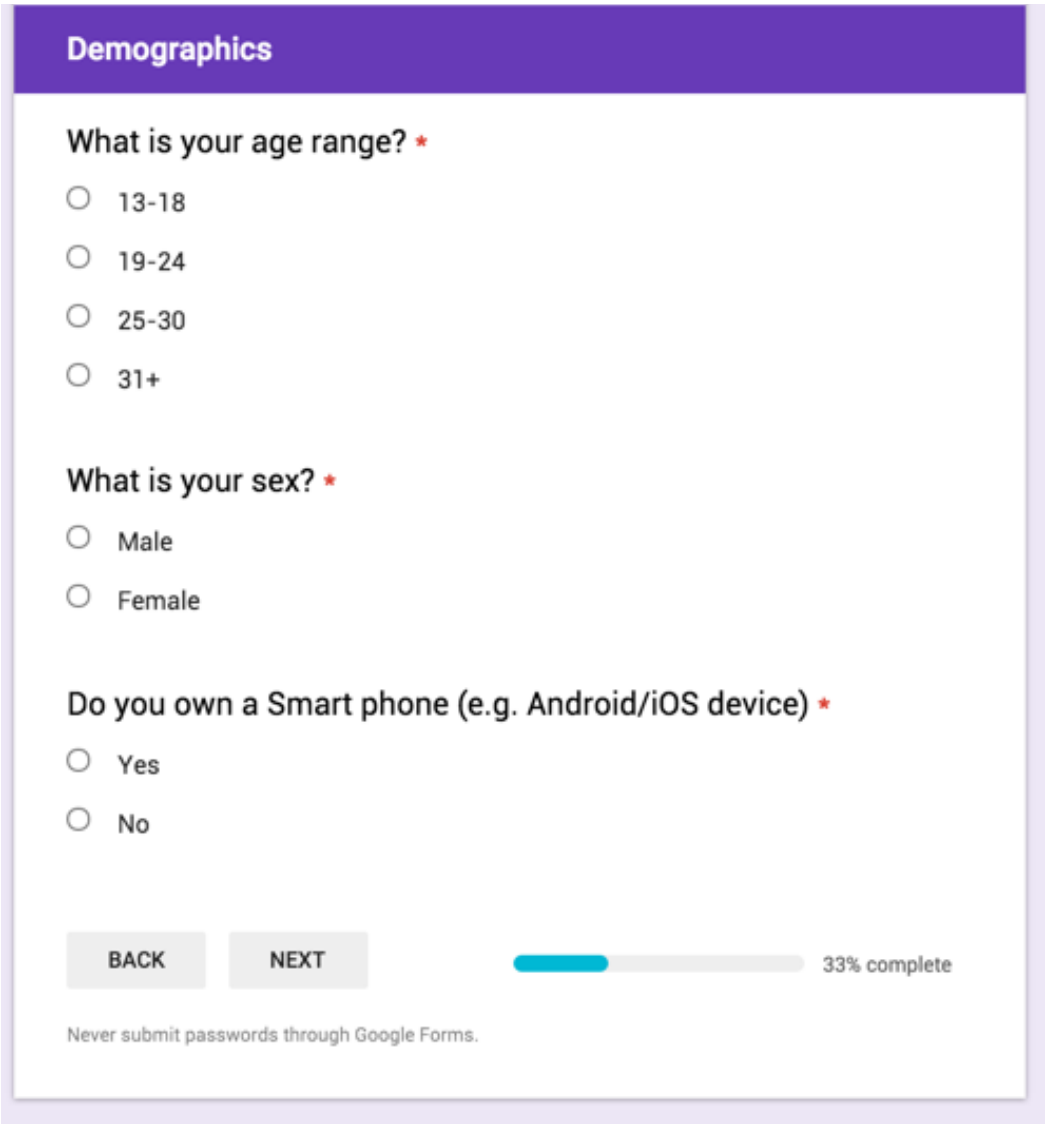
Appendix

Appendix A – *The Questionnaire*

Section 1: The introduction



Section 2: Demographics gathering


Demographics
What is your age range? *
13-18
19-24
25-30
31+
What is your sex? *
Male
Female
Do you own a Smart phone (e.g. Android/iOS device) *
Yes
No
BACK
NEXT
33% complete
Never submit passwords through Google Forms.

Registration & Testing

Do you think there is space for this application in the market? *
Read the introduction located under the title page of this document.

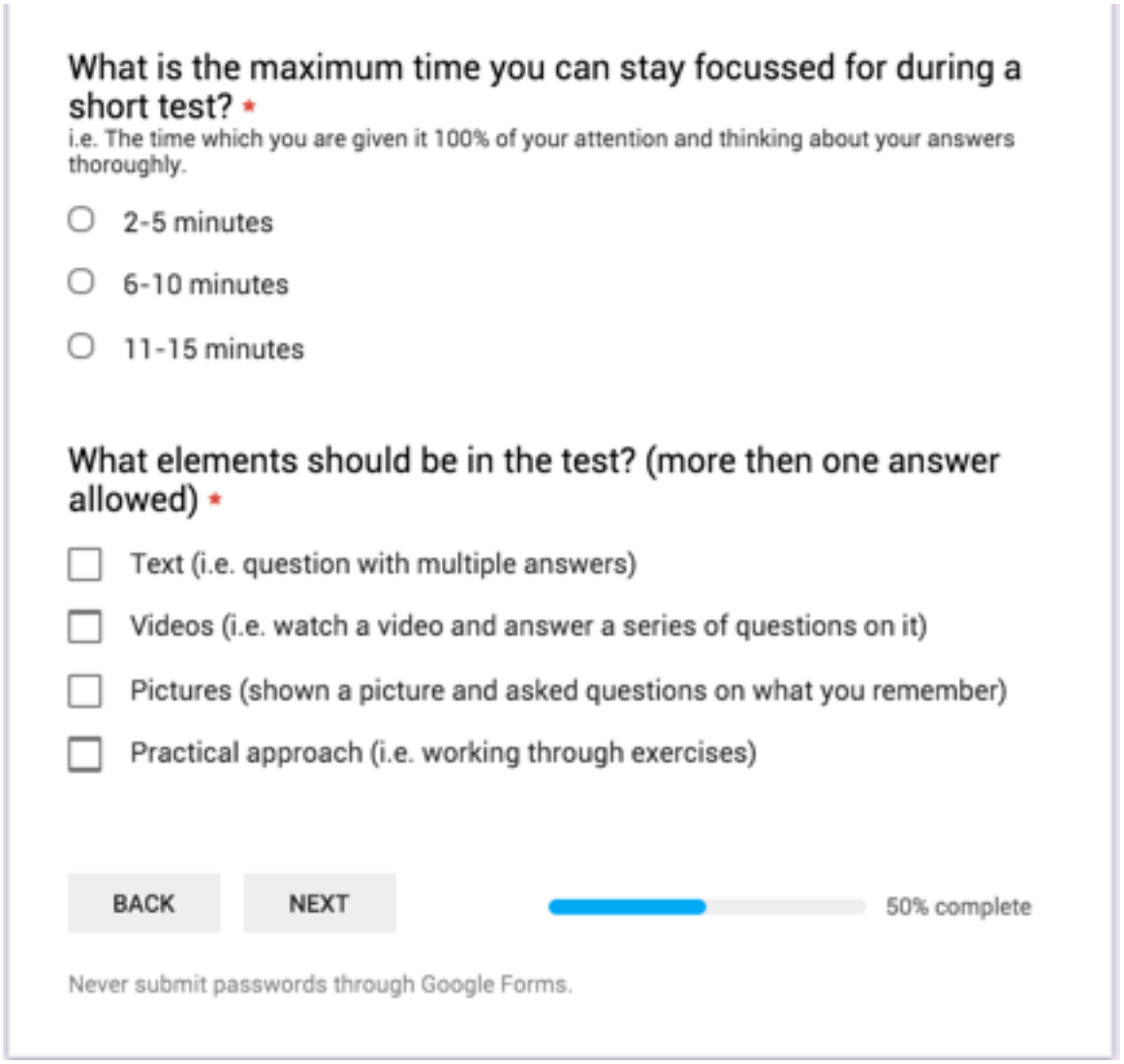
- ○ Yes
- ○ No

Select the information you would be open to share at registration (multiple answers allowed) *
Super tutor will require registration to use, please select the information you would be open to share at registration (this data will be securely stored and NOT shared with any third parties).

- ☐ First name
- ☐ Email Address
- ☐ Institution of study (if applicable)
- ☐ Age
- ☐ Surname
- ☐ Status (i.e. Student/Employed etc.)

Would you like a customisable profile (social element)? *
i.e. Tracks tests you have done, test scores, topics you are studying etc. so friends can keep up to date with what you are doing.

- ○ Yes
- ○ No


What is the maximum time you can stay focussed for during a short test? *
i.e. The time which you are given it 100% of your attention and thinking about your answers thoroughly.
2-5 minutes
6-10 minutes
11-15 minutes
What elements should be in the test? (more then one answer allowed) *
Text (i.e. question with multiple answers)
Videos (i.e. watch a video and answer a series of questions on it)
Pictures (shown a picture and asked questions on what you remember)
Practical approach (i.e. working through exercises)
BACK
NEXT
50% complete
Never submit passwords through Google Forms.

Section 4: App colour scheme, look and feel

Colour scheme, Look & Feel

This section will ask you questions about the colour scheme of the application and usability aspects such as the Look and Feel (how the application looks and interactions with the elements) of the application.

What colours do you associate with education? *

Your answer

Once you click a button, should it appear as a loading or present a new dialogue for loading? *

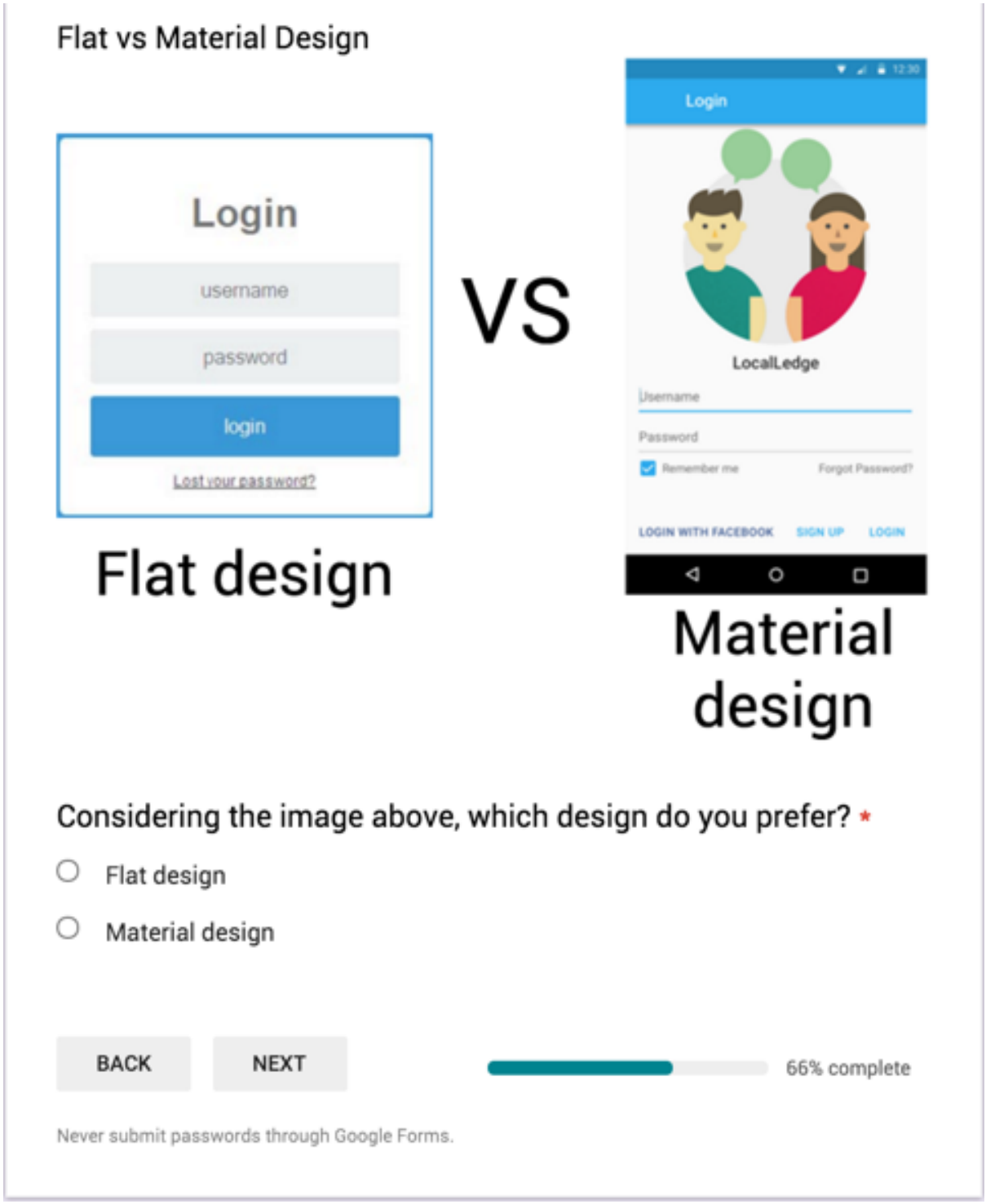
- ○ Load in the same button space
- ○ Presented as new dialogue

Flat vs Depth buttons

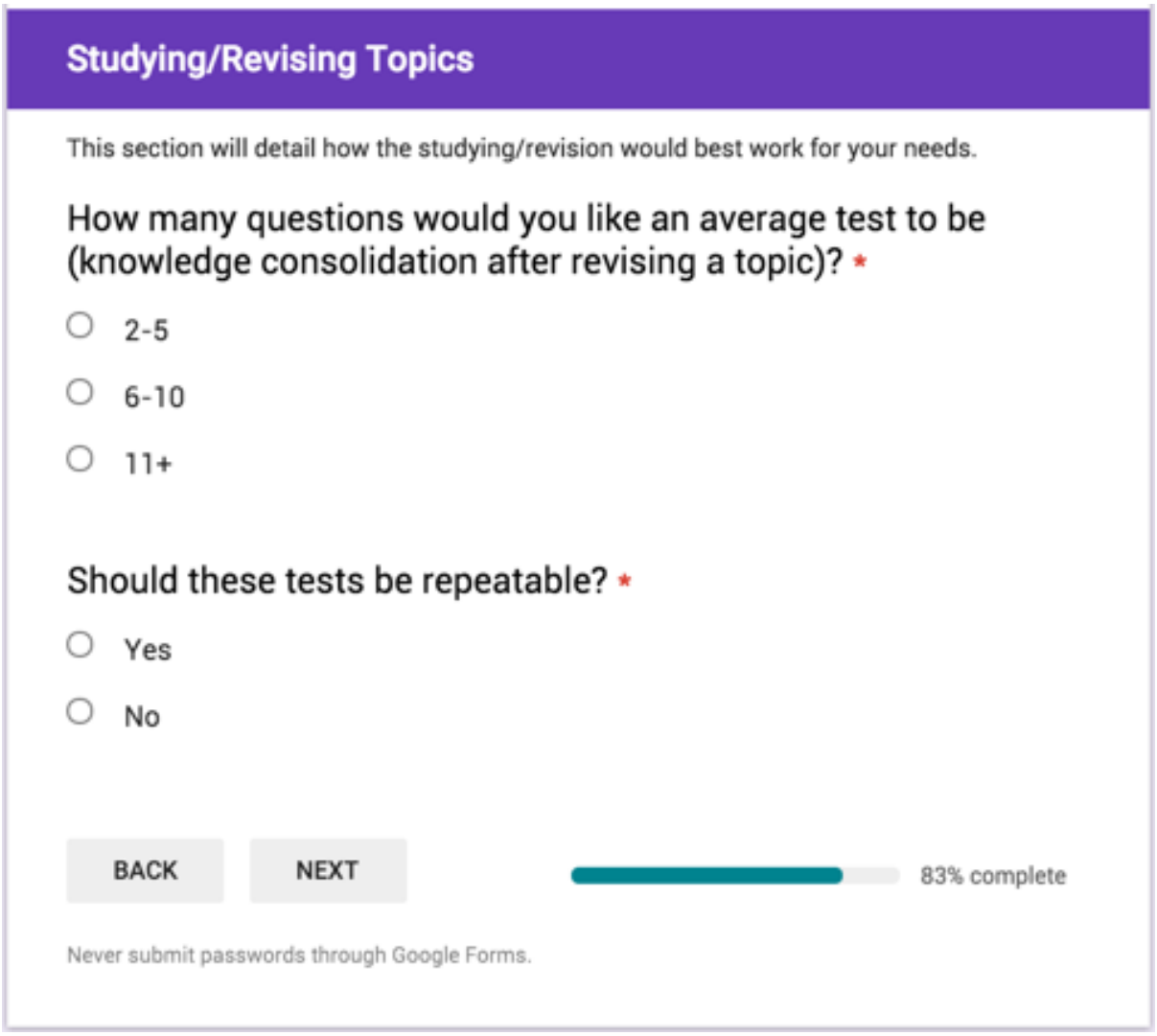
Flat Button Depth Button

Looking at the image above, which button style do you prefer? *

- ○ Flat buttons
- ○ Buttons with styling


Flat vs Material Design
Login
username
password
login
Lost your password?
VS
Login
LocalLedge
Username
Password
Remember me
Forgot Password?
LOGIN WITH FACEBOOK
SIGN UP
LOGIN
Flat design
Material design
Considering the image above, which design do you prefer? *
Flat design
Material design
BACK
NEXT
66% complete
Never submit passwords through Google Forms.

Section 5: Studying/Revising Topics



Studying/Revising Topics

This section will detail how the studying/revision would best work for your needs.

How many questions would you like an average test to be (knowledge consolidation after revising a topic)? *

- 2-5
- 6-10
- 11+

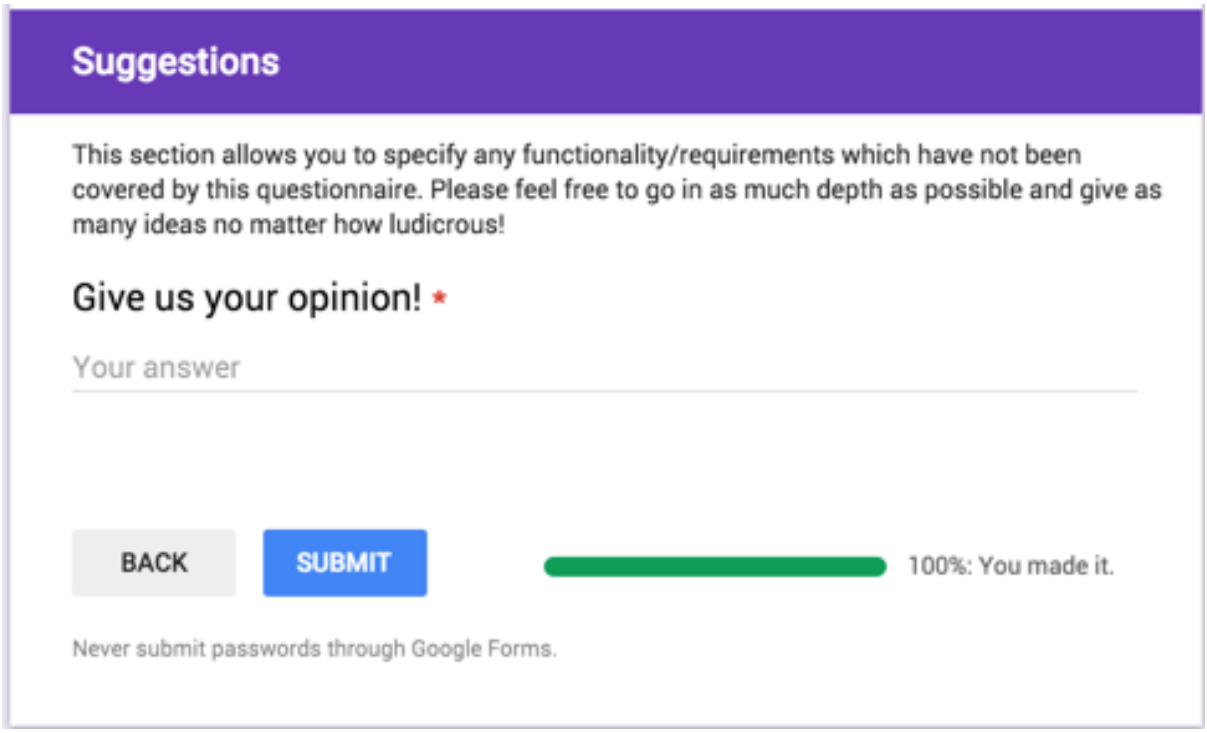
Should these tests be repeatable? *

- Yes
- No

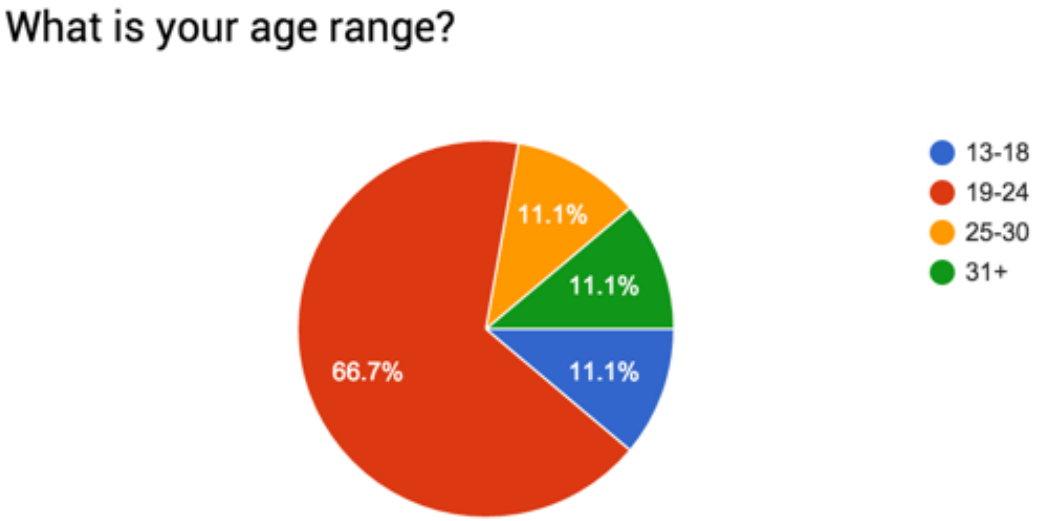
BACK NEXT 83% complete

Never submit passwords through Google Forms.

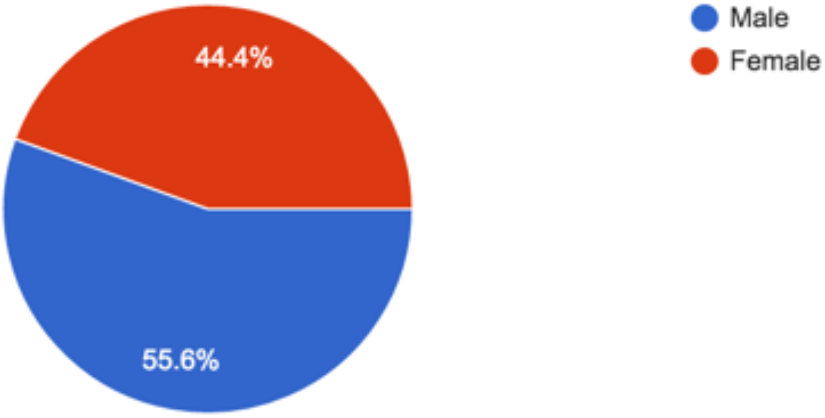
Section 6: Suggestions- Users can mention anything not included



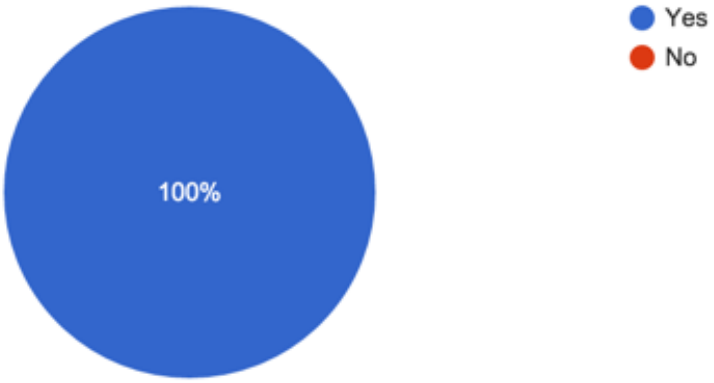
Appendix B- Questionnaire Results



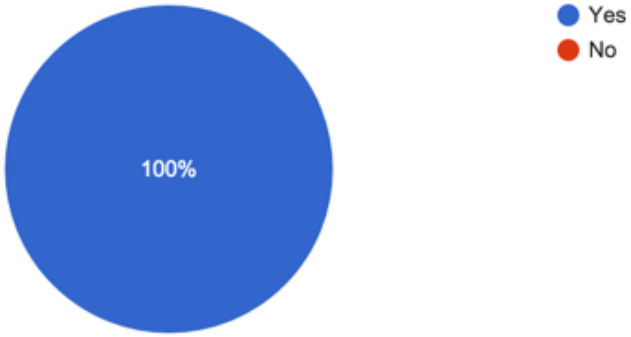
What is your sex?



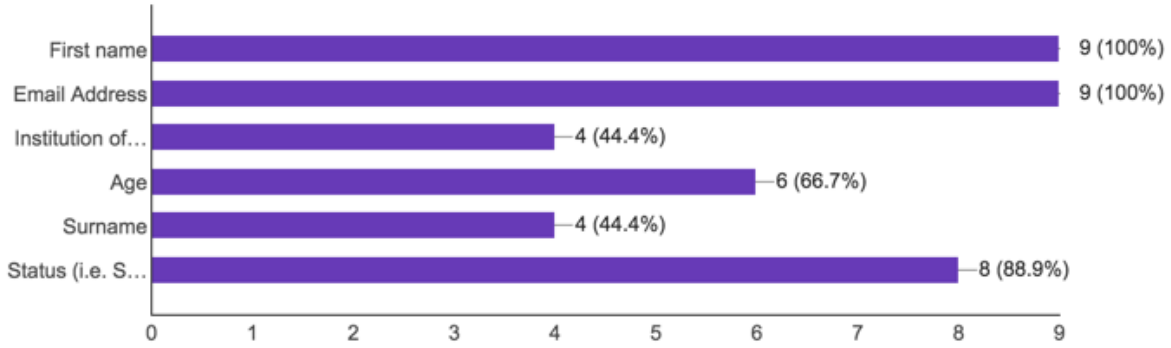
Do you own a Smart phone (e.g. Android/iOS device)



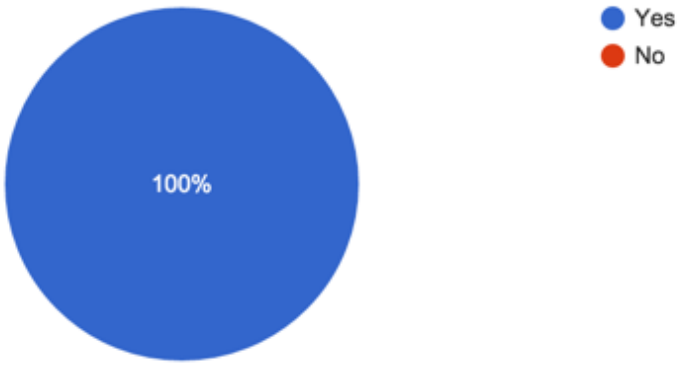
Do you think there is space for this application in the market?



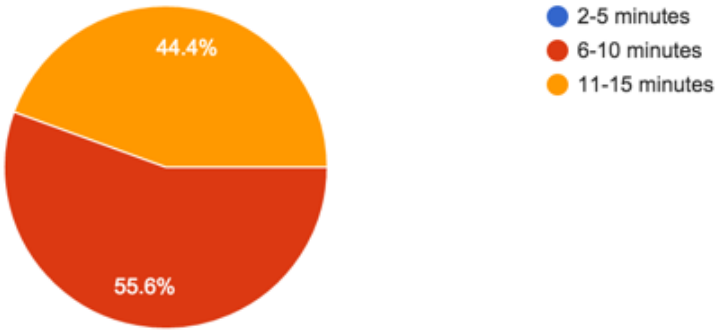
Select the information you would be open to share at registration (multiple answers allowed)



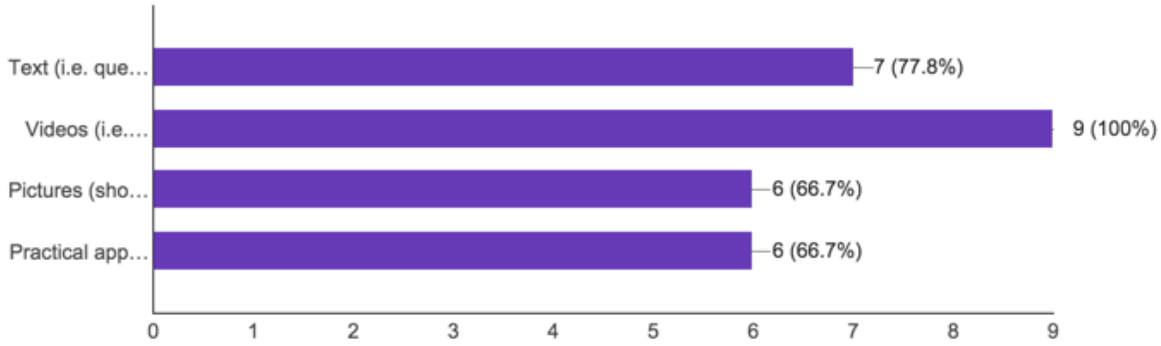
Would you like a customisable profile (social element)?



What is the maximum time you can stay focussed for during a short test?



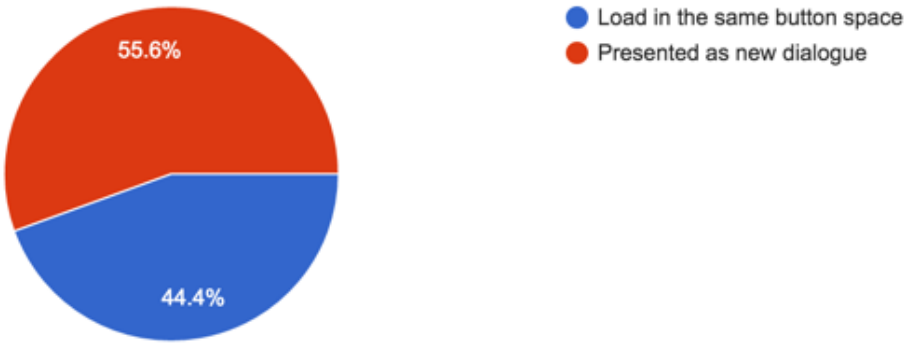
What elements should be in the test? (more then one answer allowed)



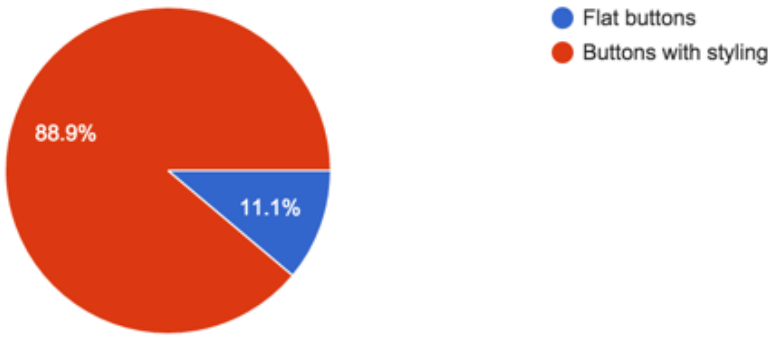
What colours do you associate with education?

White Blue Yellow
black
Blue
Blue, Green
blue
blue
bright colours
Green and Red
Grey

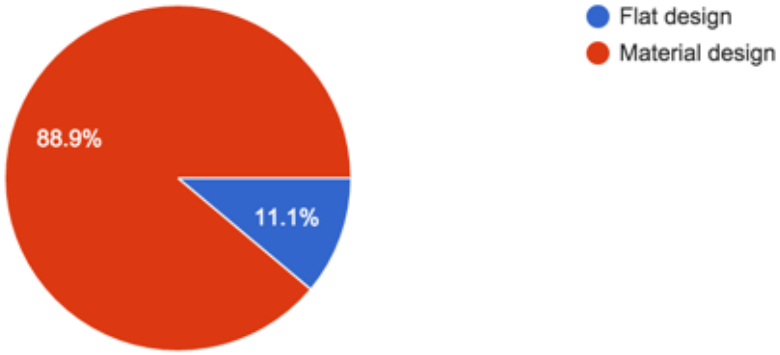
Once you click a button, should it appear as a loading or present a new dialogue for loading?



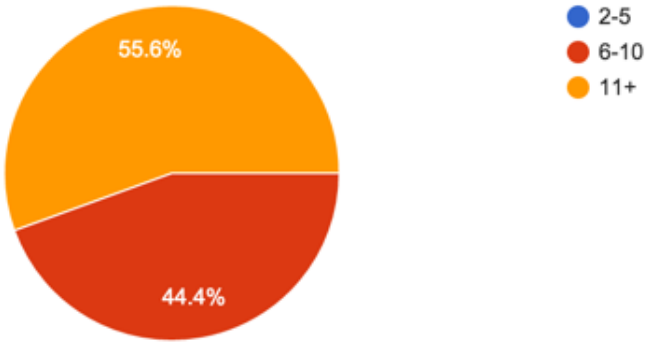
Looking at the image above, which button style do you prefer?

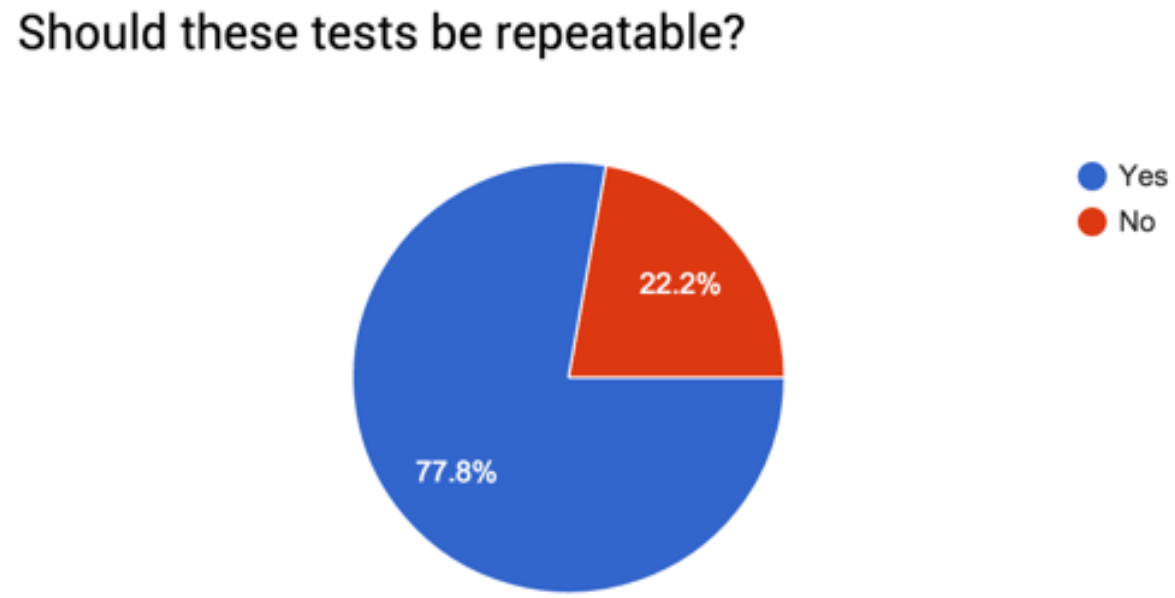


Considering the image above, which design do you prefer?



How many questions would you like an average test to be (knowledge consolidation after revising a topic)?





Give us your opinion!

Give more than 20 questions in a test.
Maybe you can link the test results and the student profile on the external supporting website.
Give limited amount of time depending on the amount of questions just like a real test.
Does super tutor cover all the subjects?
Great idea
Should be for children as well under 11
Calendar/Planner option for you to manage your studies
N/a
Colourful work is a good idea and connecting with universities will be a great help.

Appendix C

noreply@envisiontechllc.org
To: emile@envisiontechllc.org
[Super Tutor]: Looks like you forgot your password!

Yesterday at 20:44

N

Dear User

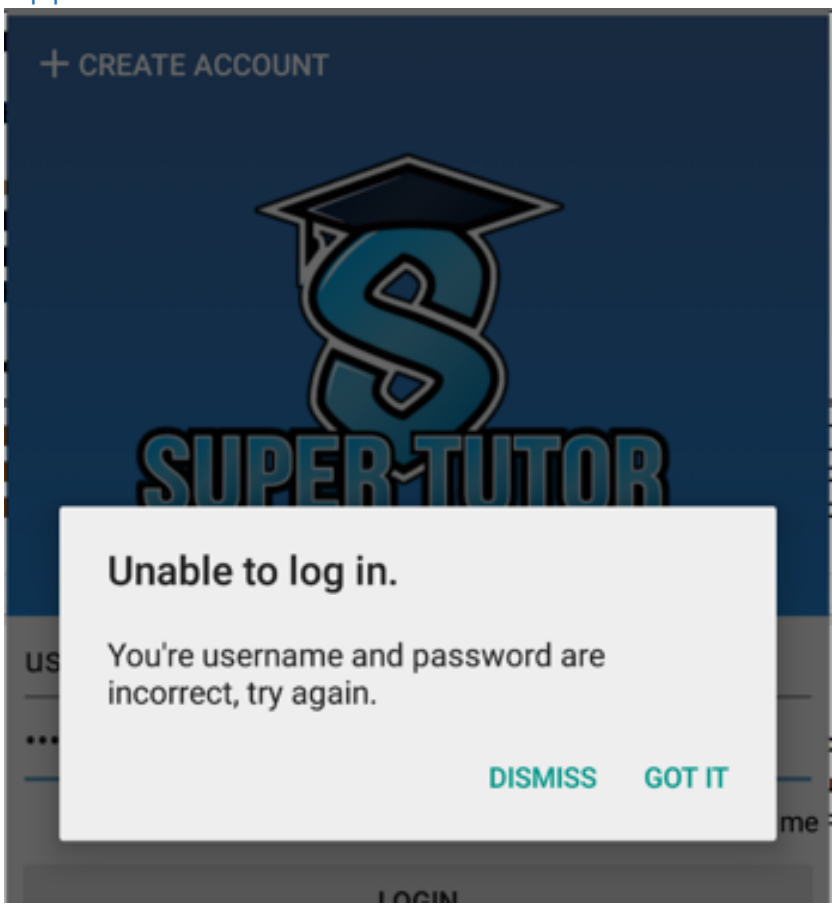
This email is a confirmation of your password recovery. Your password last known is:

admin

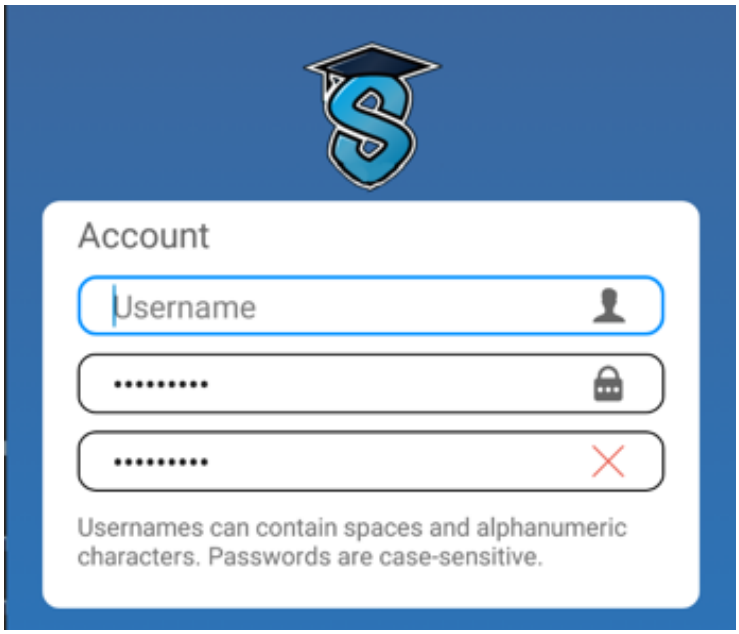
Kind regards,

Super Tutor Support Team

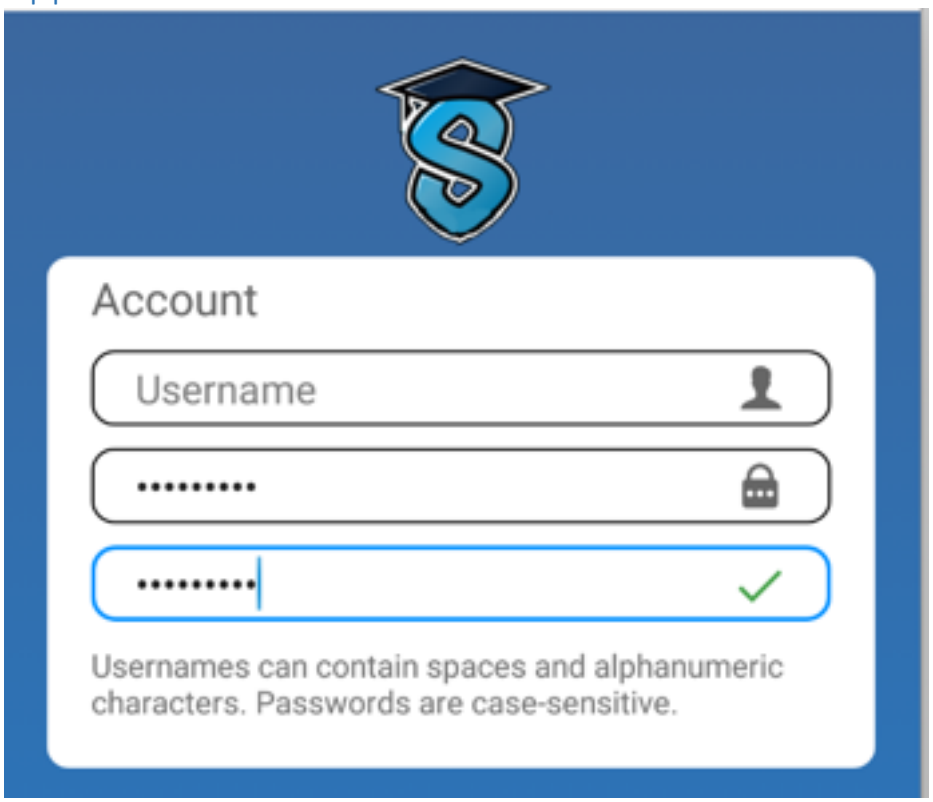
Appendix D



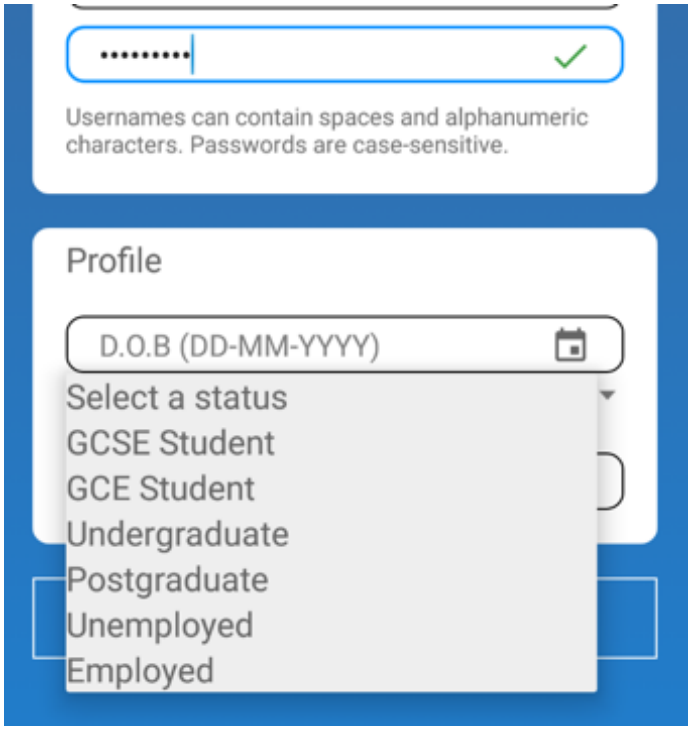
Appendix E



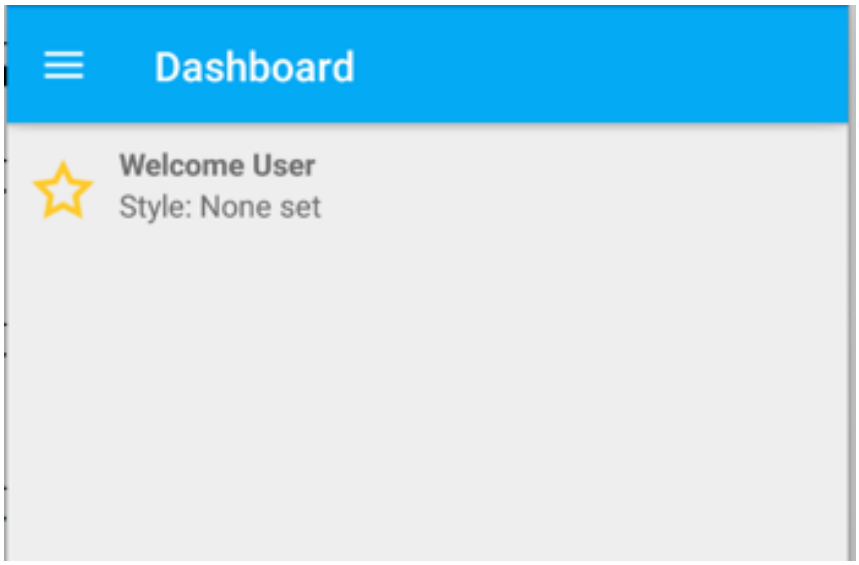
Appendix F



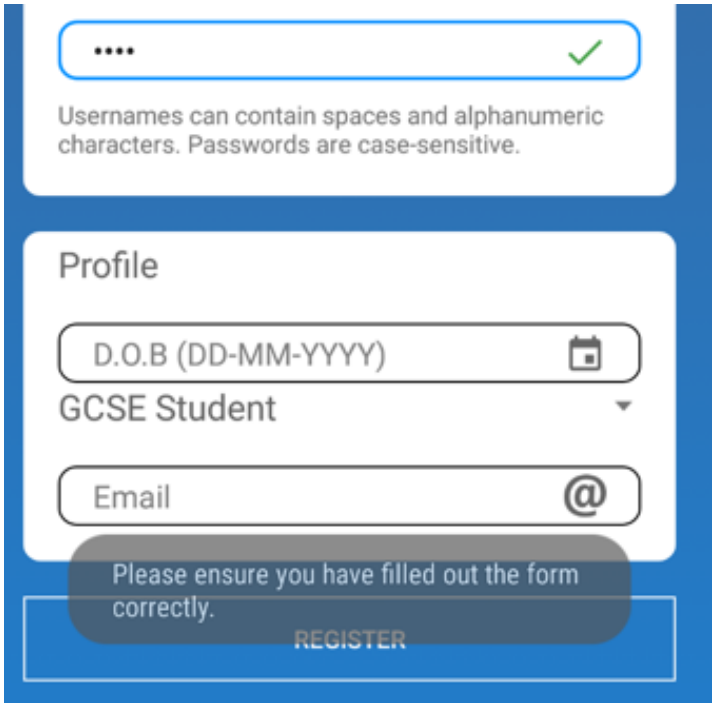
Appendix G



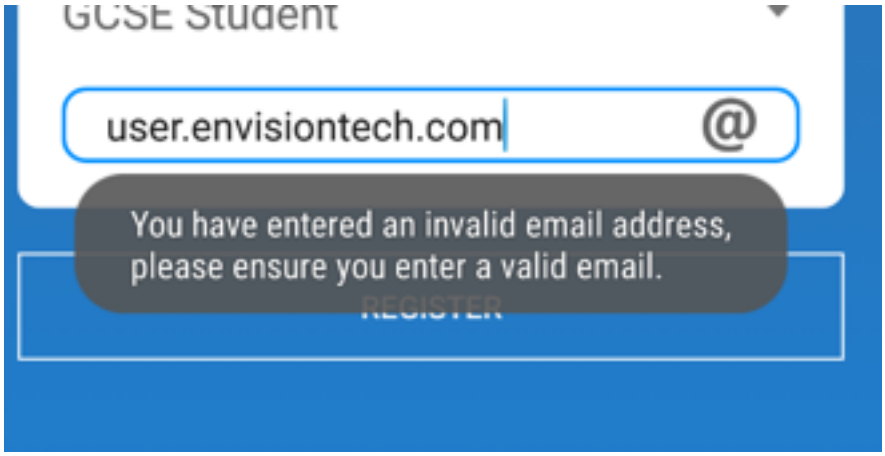
Appendix H



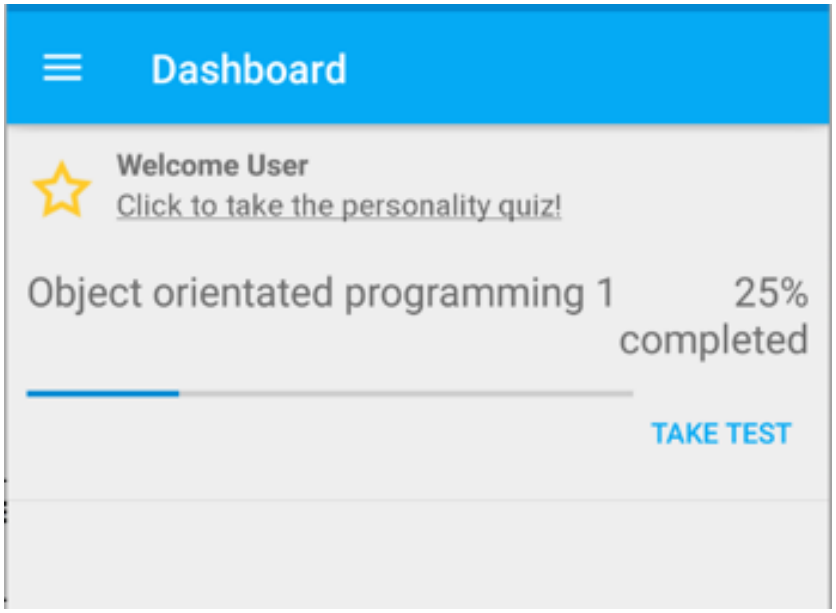
Appendix I



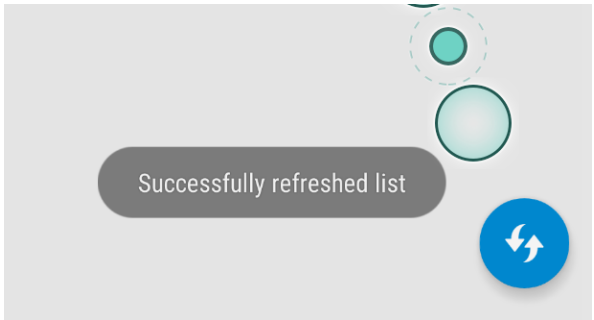
Appendix J



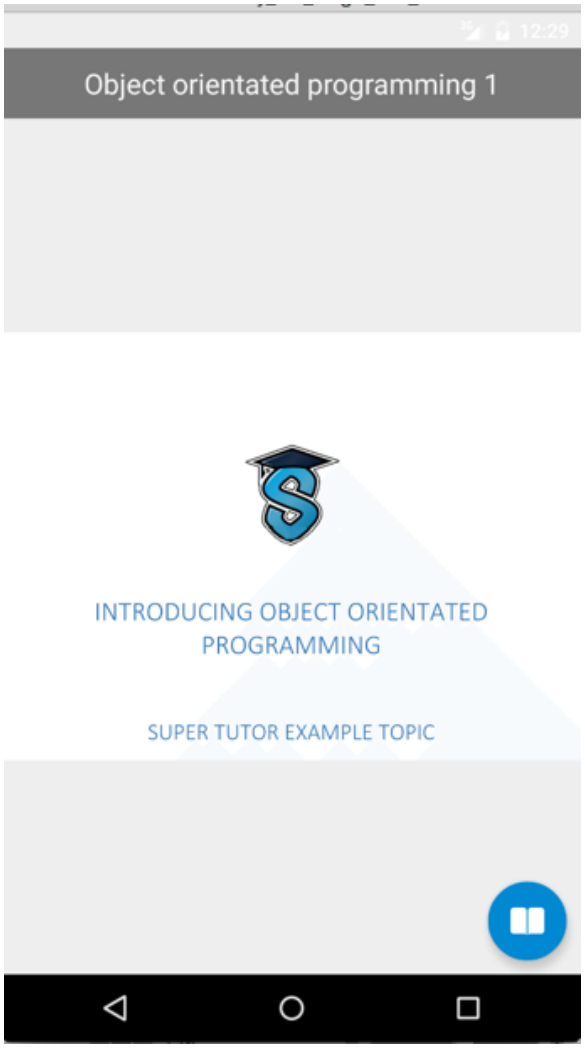
Appendix K



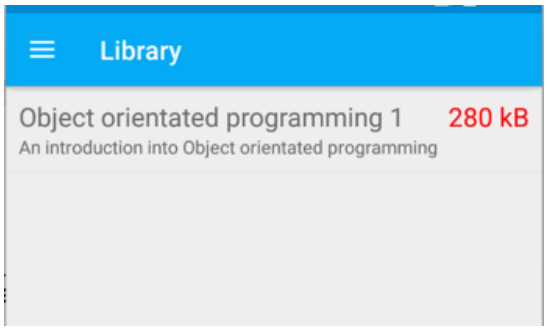
Appendix L



Appendix M

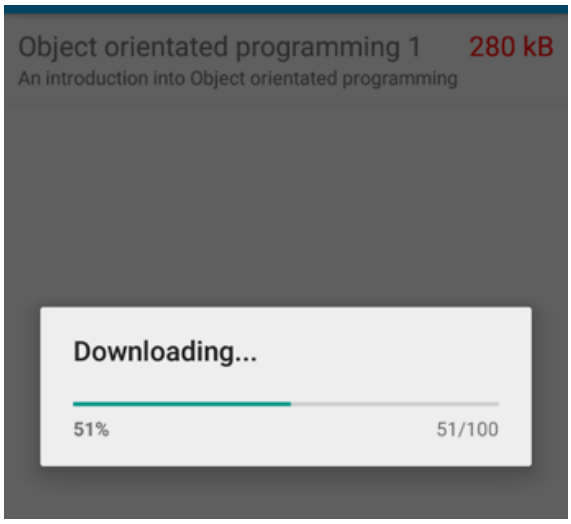


Appendix N

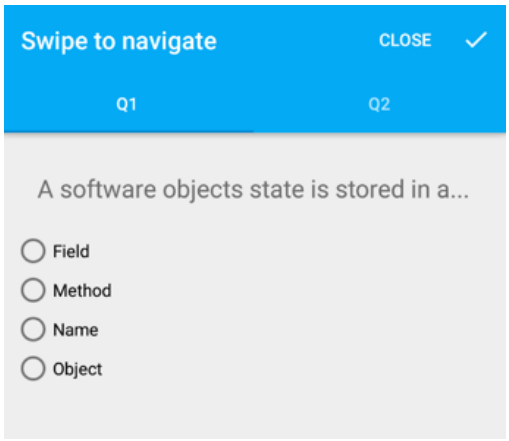


Appendix N1

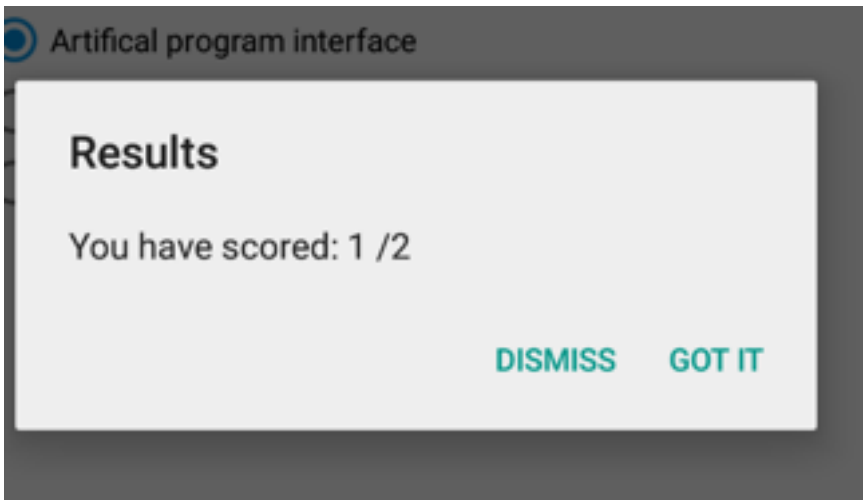
Appendix N2



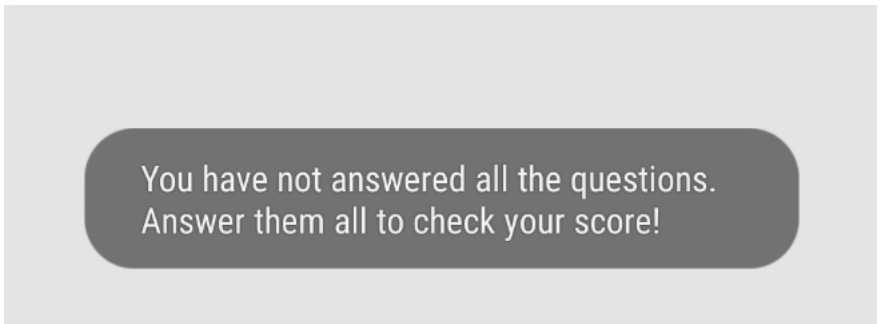
Appendix O2



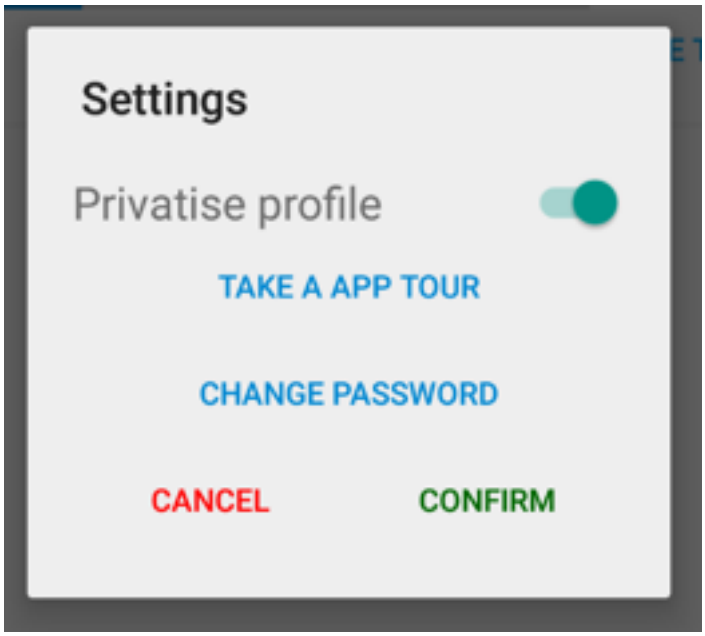
Appendix O3



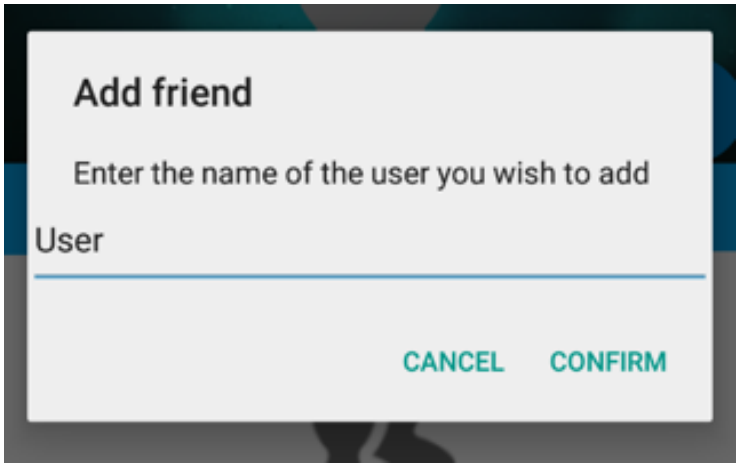
Appendix O4



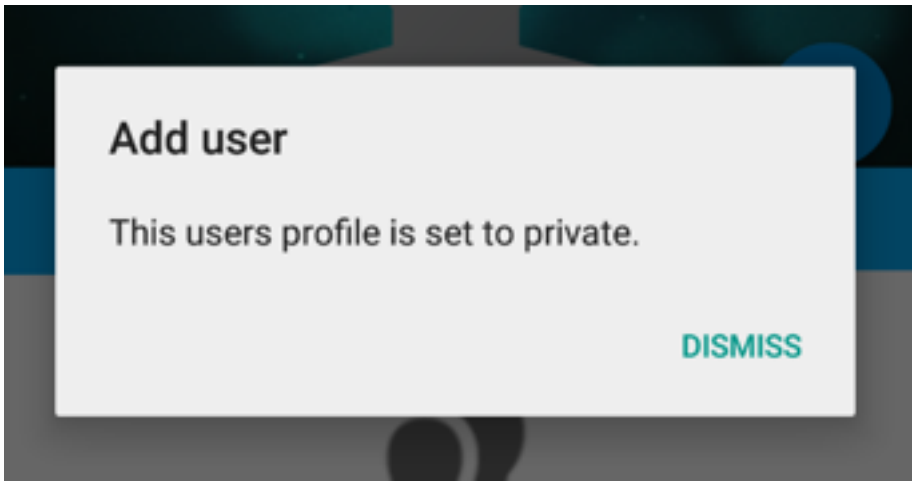
Appendix P0



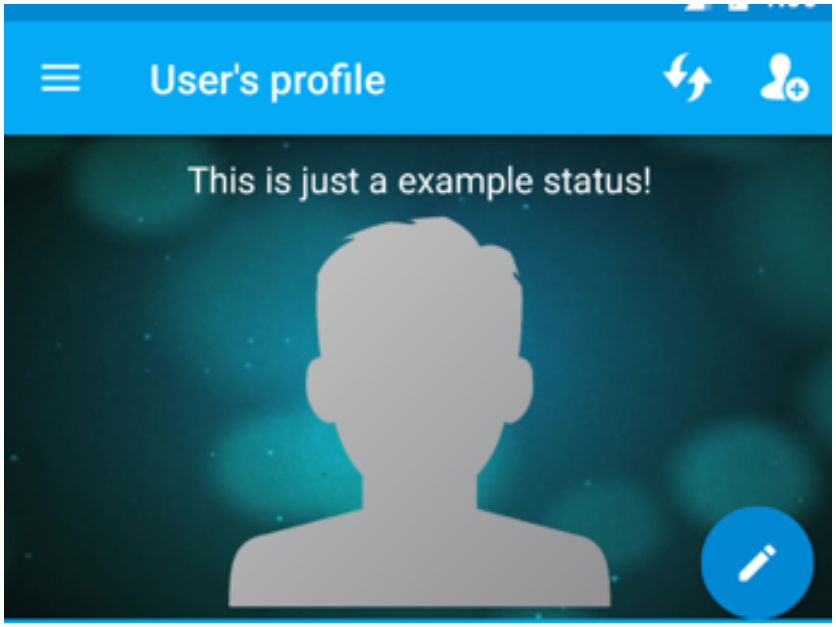
Appendix P1



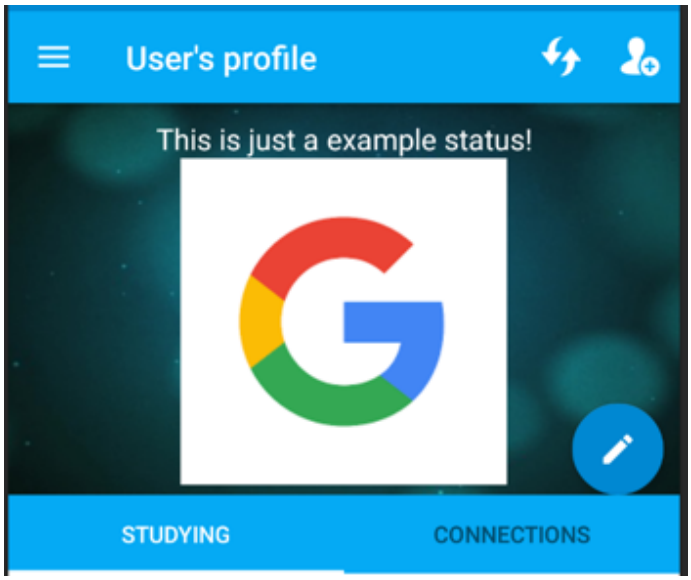
Appendix P2



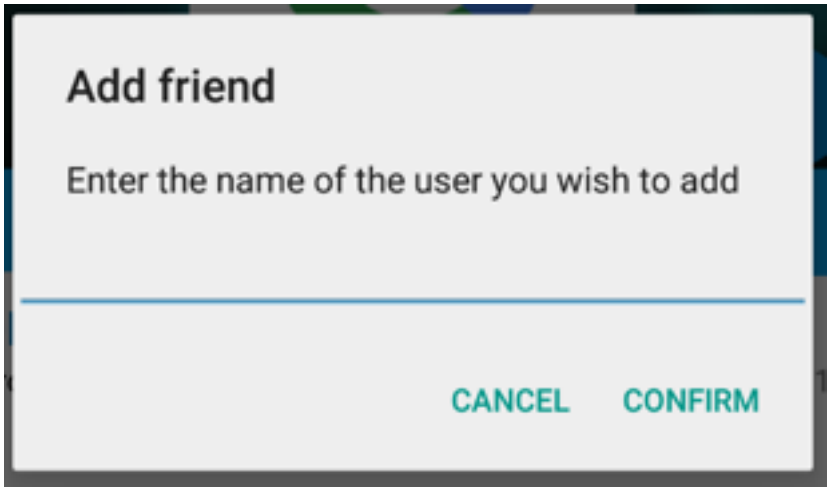
Appendix P3



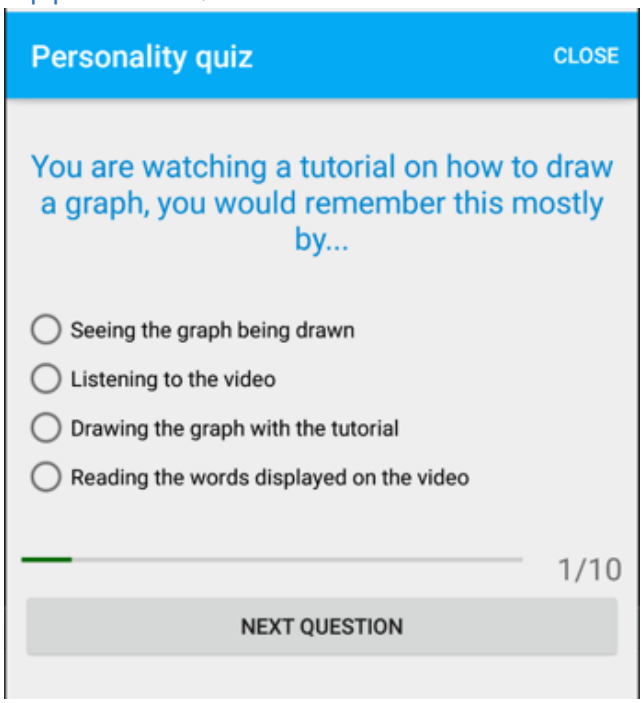
Appendix P4



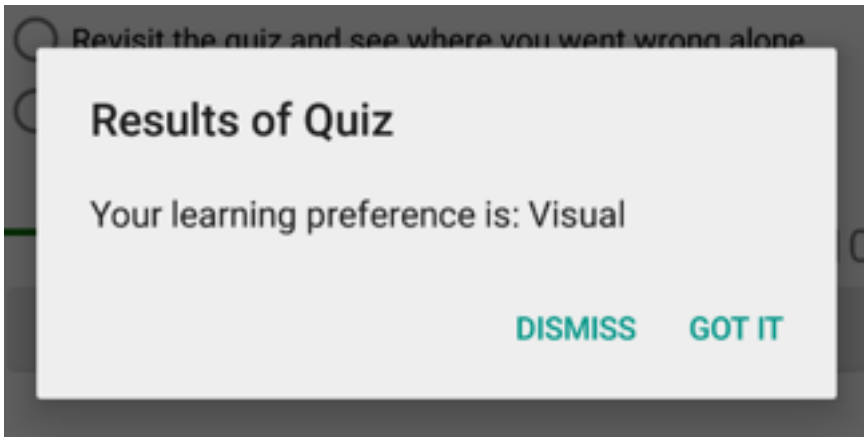
Appendix P5



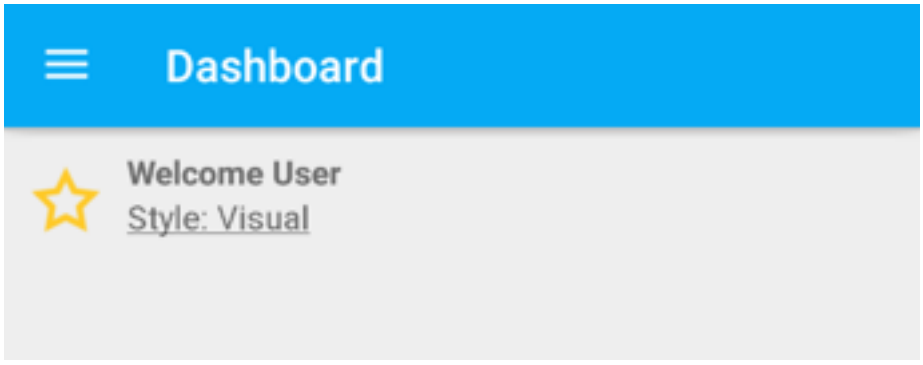
Appendix Q



Appendix Q1



Appendix Q2



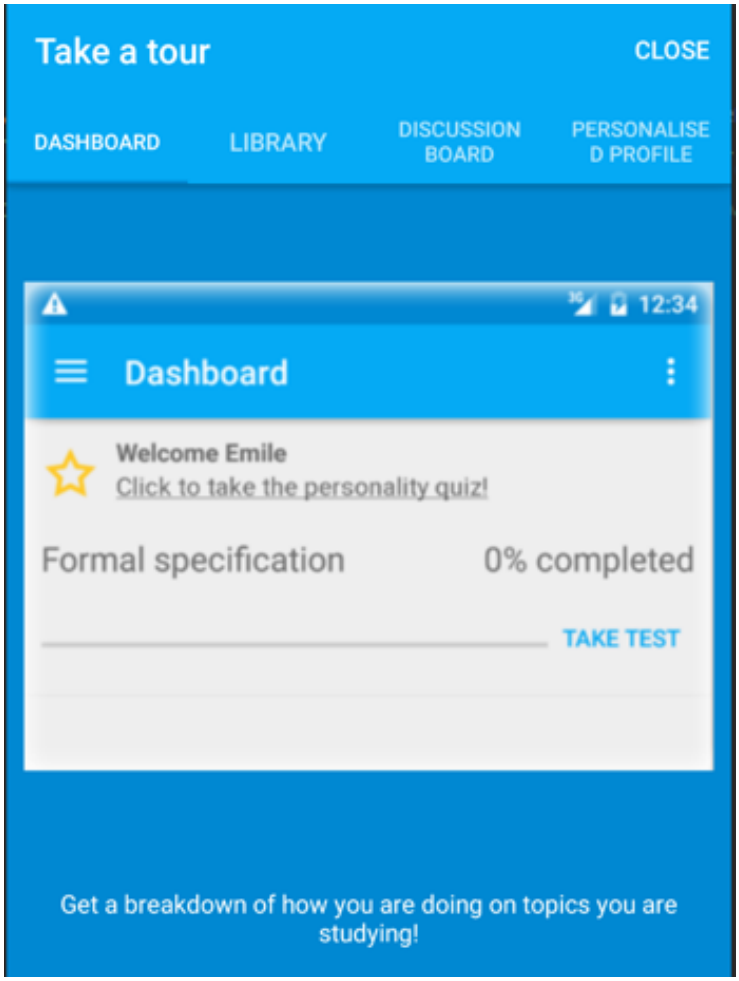
Appendix R

```
private void checkDatabase(){
    appData = AppContext.getContext();
    dbManager = new SQLManager(this);
    new GCMRegistration(this).execute();

    timer = new CountDownTimer(1100, 1000) {
        @Override
        public void onTick(long millisUntilFinished) {

        }

        @Override
        public void onFinish() {
            List<Subject> list = dbManager.getDownloadedSubjects();
            if(list != null && list.size() > 0){
                AppLibrary library = AppLibrary.getInstance();
                library.addSubjectsToMyLibrary(list.toArray(new Subject[list.size()]));
            }
            HashMap<String, String> userCredentials = dbManager.getUserCredentials();
            if(userCredentials != null){
                User tUser = new User(userCredentials.get("Username"), userCredentials.get("Password"),
                        userCredentials.get("Email"), userCredentials.get("LearnerType"));
                tUser.setStatus(userCredentials.get("Status"));
                tUser.setImageBytes(userCredentials.get("Image"));

                AppContext.getContext().setCurrentUser(tUser);
                Intent dashBoardIntent = new Intent(SplashScreen.this, SuperTutor.class);
                startActivity(dashBoardIntent);
                finish();
            } else {
                if(Utilities.checkConnectivity(SplashScreen.this)){
                    initApplicationContents();
                    return;
                } else {
                    AlertDialog dialog = Utilities.createDialog(SplashScreen.this, "Error", "Unable to connect to server, check your internet conne
                    dialog.show();
                }
            }
        }
    }.start();
}
```

Appendix R3

```
public class GenericTourFragment extends Fragment {

    private AppCompatActivity activity;

    private Bundle args;
    private TextView textDescription;
    private ImageView imageIcon;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle savedInstance){
        return inflater.inflate(R.layout.generic_tour_template, parent, false);
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);

        activity = (AppCompatActivity)getActivity();
    }

    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) { initFragment(view); }

    private void initFragment(View view){
        args = getArguments();
        if(args != null){
            textDescription = (TextView)view.findViewById(R.id.appTour_description);
            imageIcon = (ImageView)view.findViewById(R.id.appTour_image);

            textDescription.setText(args.getString("description"));
            imageIcon.setImageDrawable(getResources().getDrawable(args.getInt("image"), activity.getTheme()));
        }
    }
```

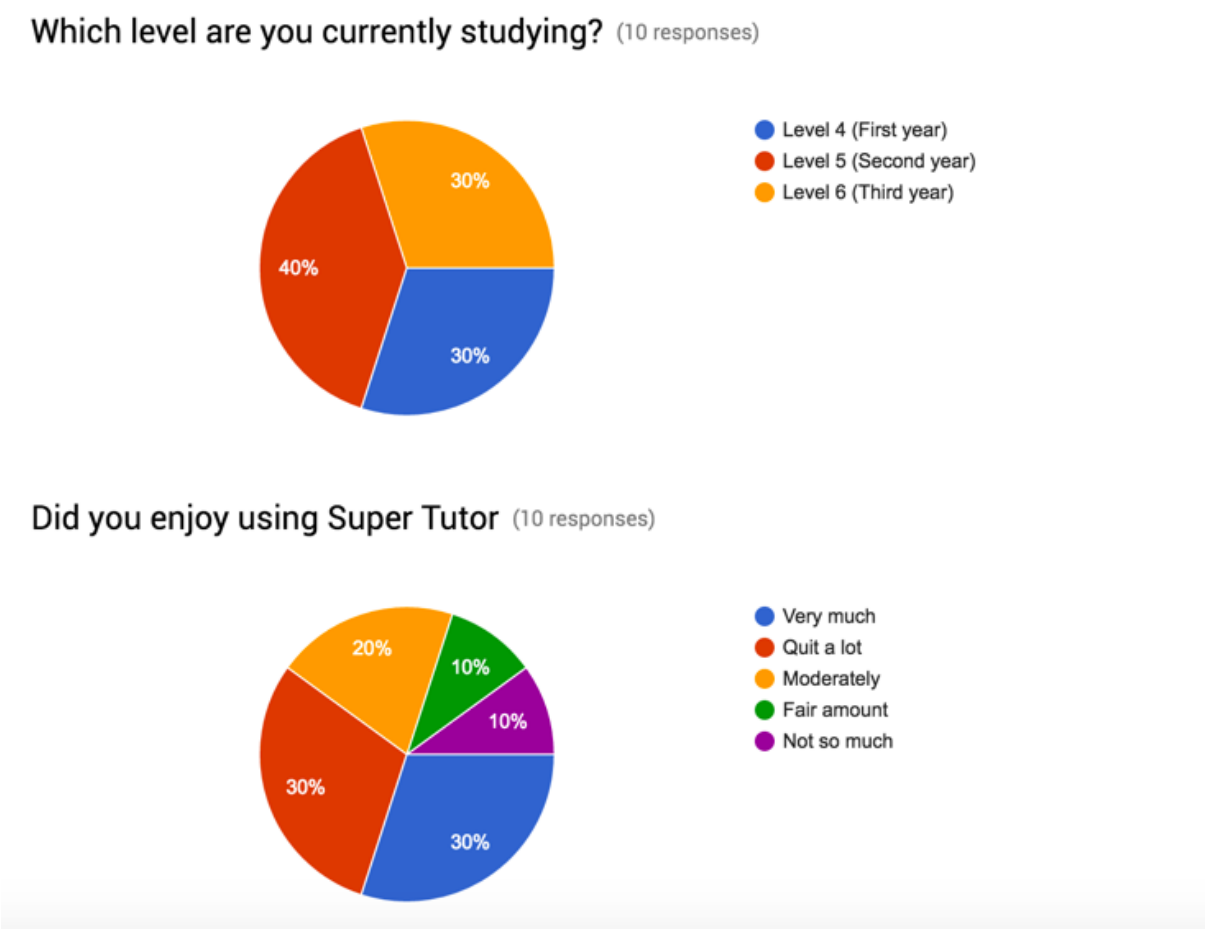
Appendix R2



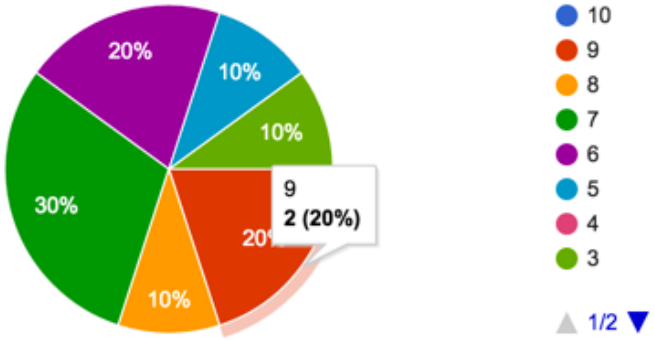
Appendix S

```
public enum XTour {

    DASHBOARD("Dashboard", "Get a breakdown of how you are doing on topics you are studying!", R.drawable.dashboard_tour, new GenericTourFragment()),
    LIBRARY("Library", "Browse our large collection of subjects available for studying. When new content is added it will automatically be available he
    DISCUSSION_BOARD("Discussion Board", "Need help on a topic you are revising? Pop into the discussion board and get help from the community!", R.draw
    PROFILE("Personalised profile", "Want to chat with a friend or build your educational portfolio? Now you can with the easy to use Profile page!", R
    ;


    private String title, description;
    private int resourceID;
    private Fragment fragment;

    XTour(String title, String description, int resourceID, Fragment fragment){
        this.title = title;
        this.description = description;
        this.resourceID = resourceID;
        this.fragment = fragment;
    }

    public String getTitle() { return this.title; }

    public String getDescription() { return this.description; }

    public int getImage() { return this.resourceID; }

    public Fragment getFragment() { return this.fragment; }
}
```

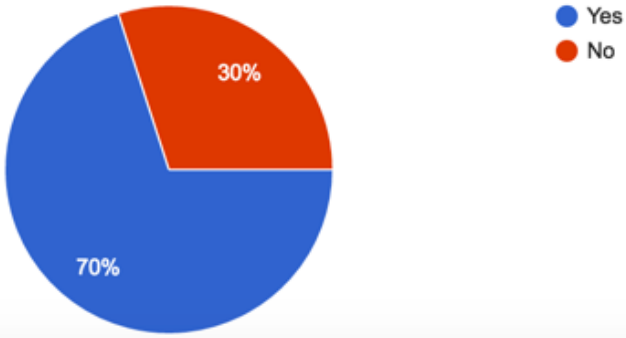
Appendix T


Which level are you currently studying? (10 responses)
30%
40%
30%
Level 4 (First year)
Level 5 (Second year)
Level 6 (Third year)
Did you enjoy using Super Tutor (10 responses)
20%
10%
10%
30%
30%
Very much
Quit a lot
Moderately
Fair amount
Not so much

What would you rate the app (1 Lowest - 10 Highest) (10 responses)



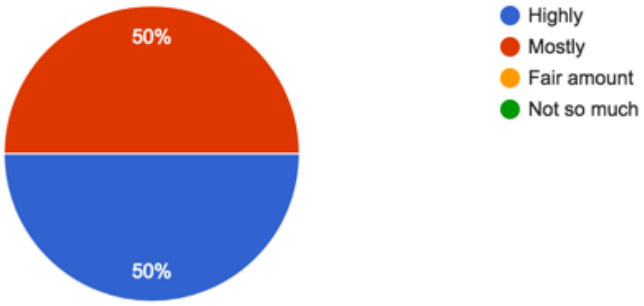
Did you find the Content organisation suitable for your learning style?
(10 responses)



A brief explanation of your choice above (optional) (6 responses)

Particularly liked how the diagrams appeared before the explanations.
I learn best through audio, the app did not have any audio for its content
Content was well organised, use of diagrams was intriguing
Didn't notive a difference
Completely forgot about some elements in OOP, this jogged my memory :)
Noticed a small difference, nothing too big

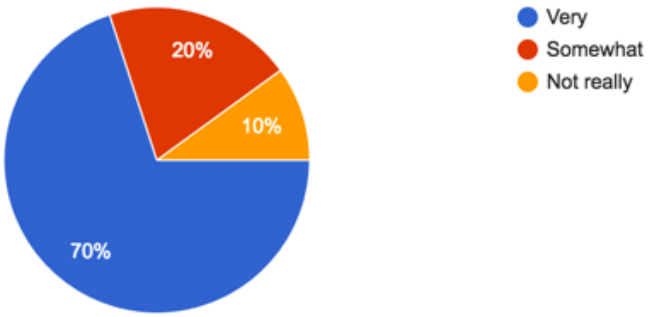
Were the Topic tests relevant to the topic? (10 responses)



What would you add to make the tests more interesting (optional) (5 responses)

Add different elements like fill in the blanks
more interactivity
Add a time limit to the tests/time element
Time them
Some were quite challenging haha

Was the addition of Profiles engaging? (10 responses)



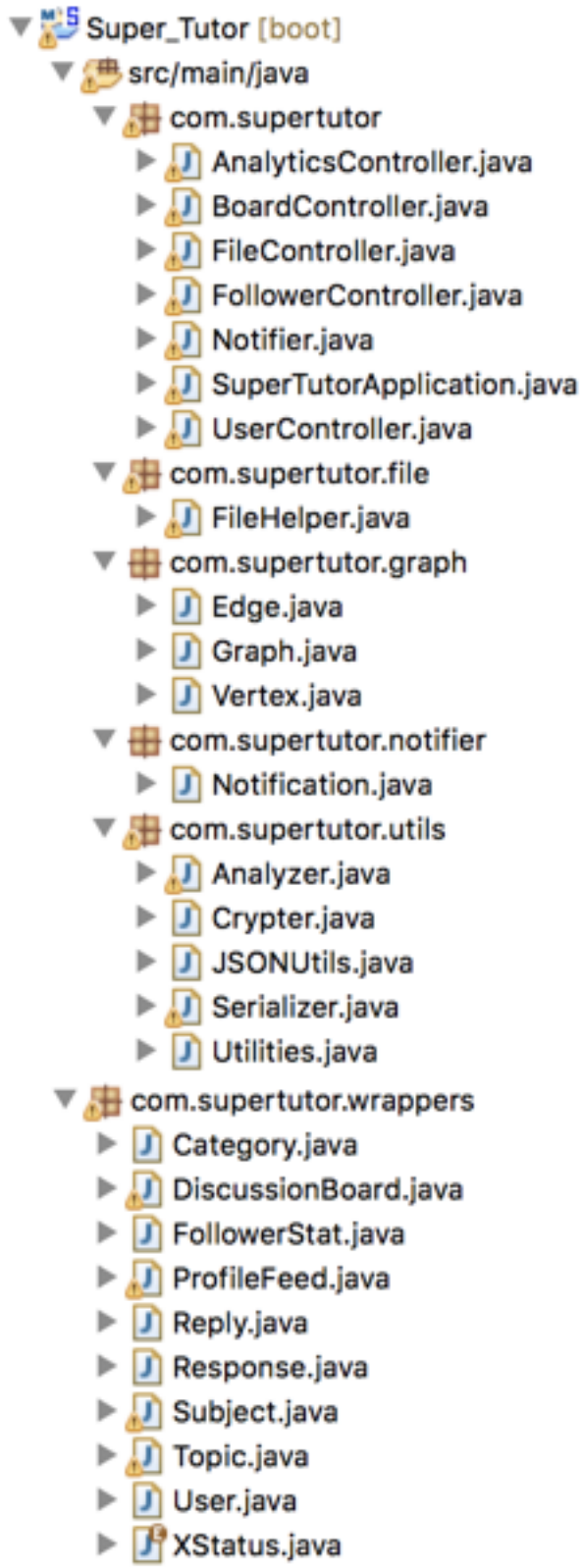
Brief comment about what you enjoyed about this feature (optional)
(4 responses)

added a nice social element
Liked the layout of the profiles
Like the direction this is going, more features would be nice
Liked the layout of profiles should pursue this feature further

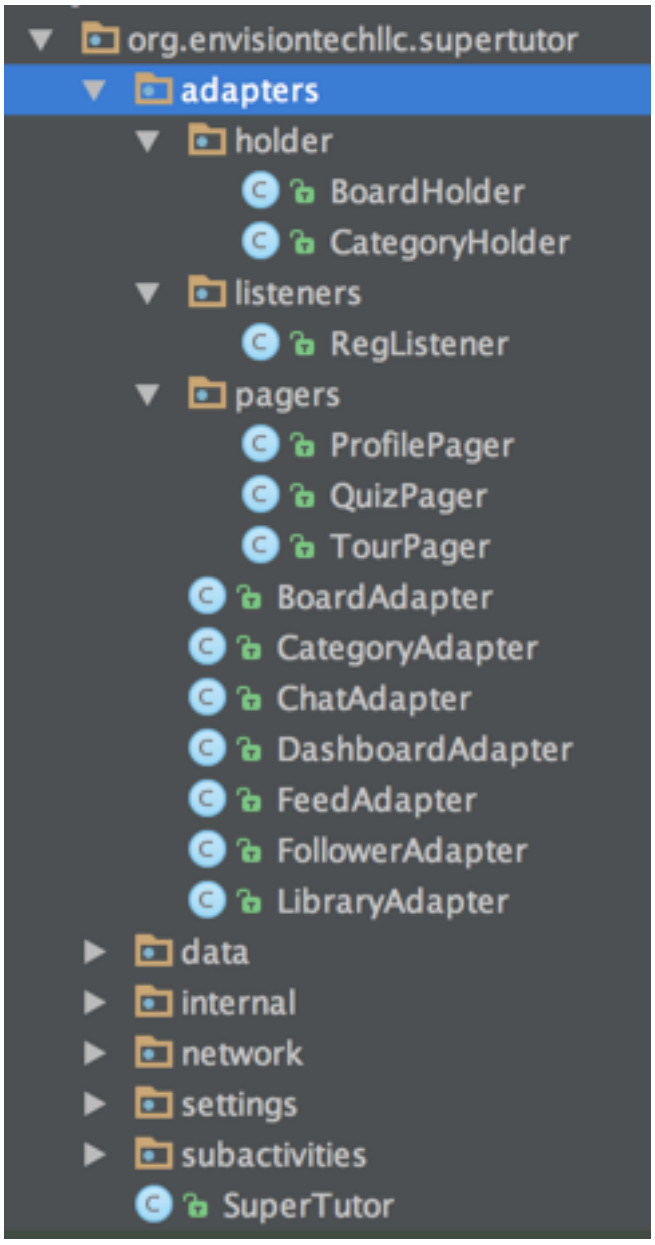
Finally, any other comments (4 responses)

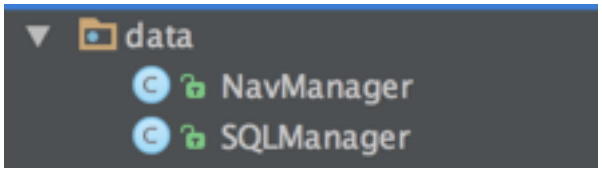
App was easy to use, engaging and has some nice features
very good idea, has a lot of potential
Loved the discussion boards style very nice to have a chat with people
Great app, friendly developer when tested, takes ideas onboard and turns them into features

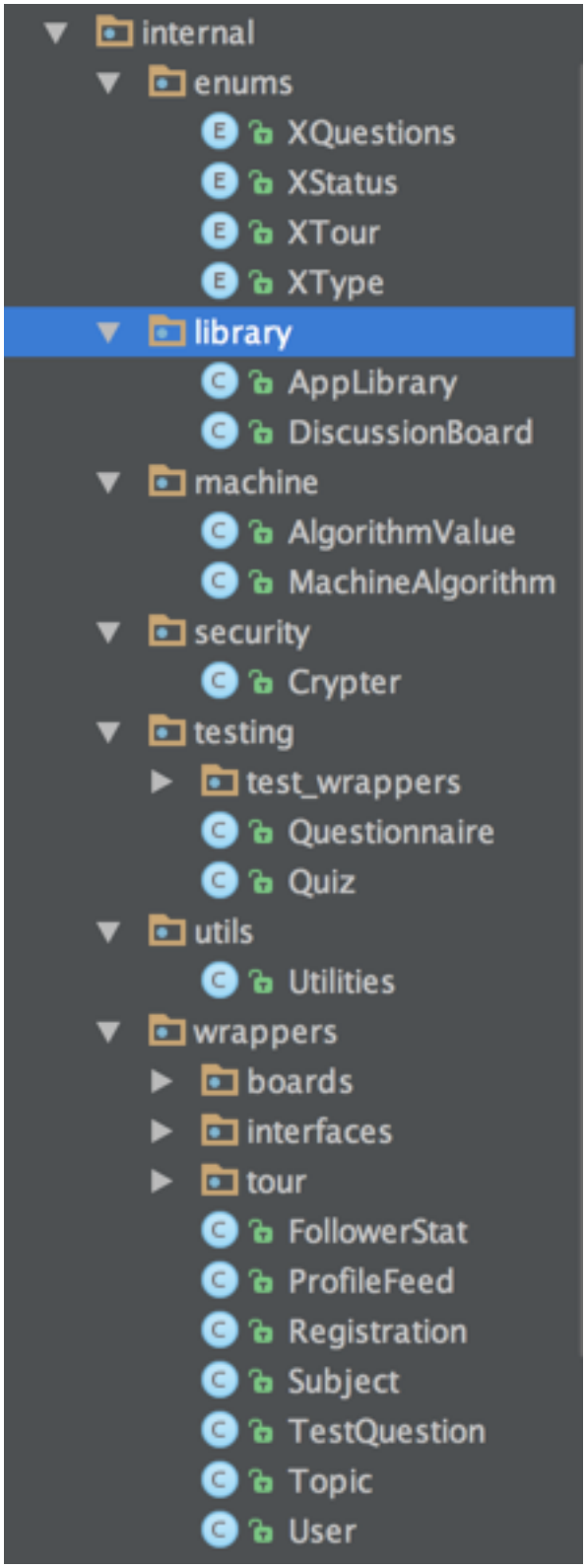
Appendix U – Web server file architecture

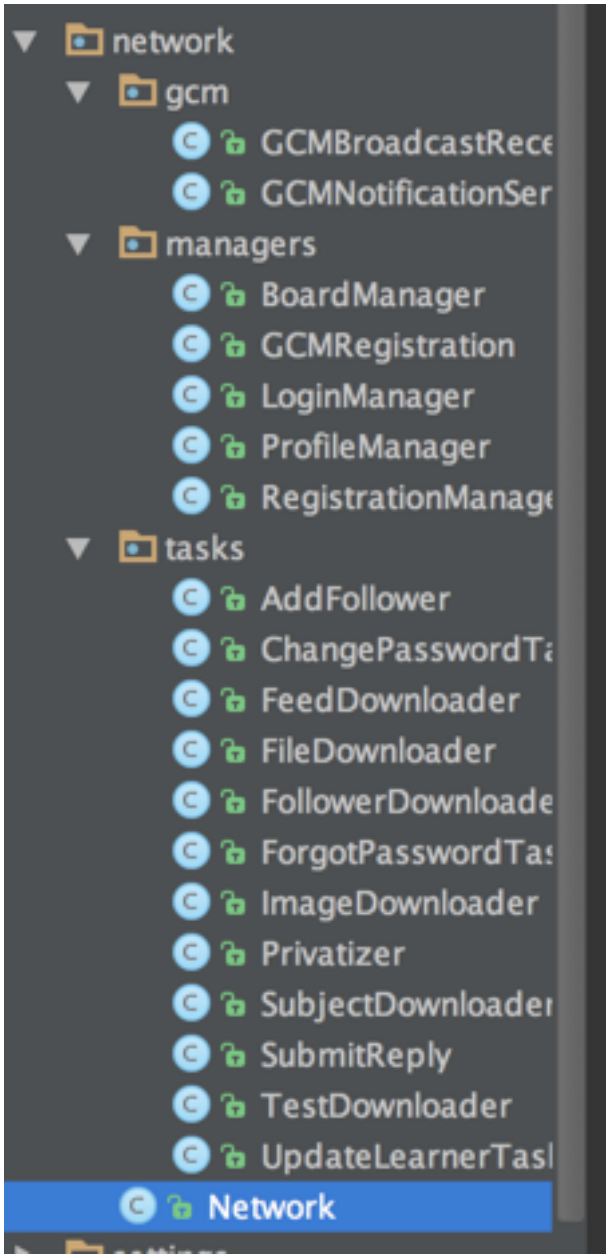


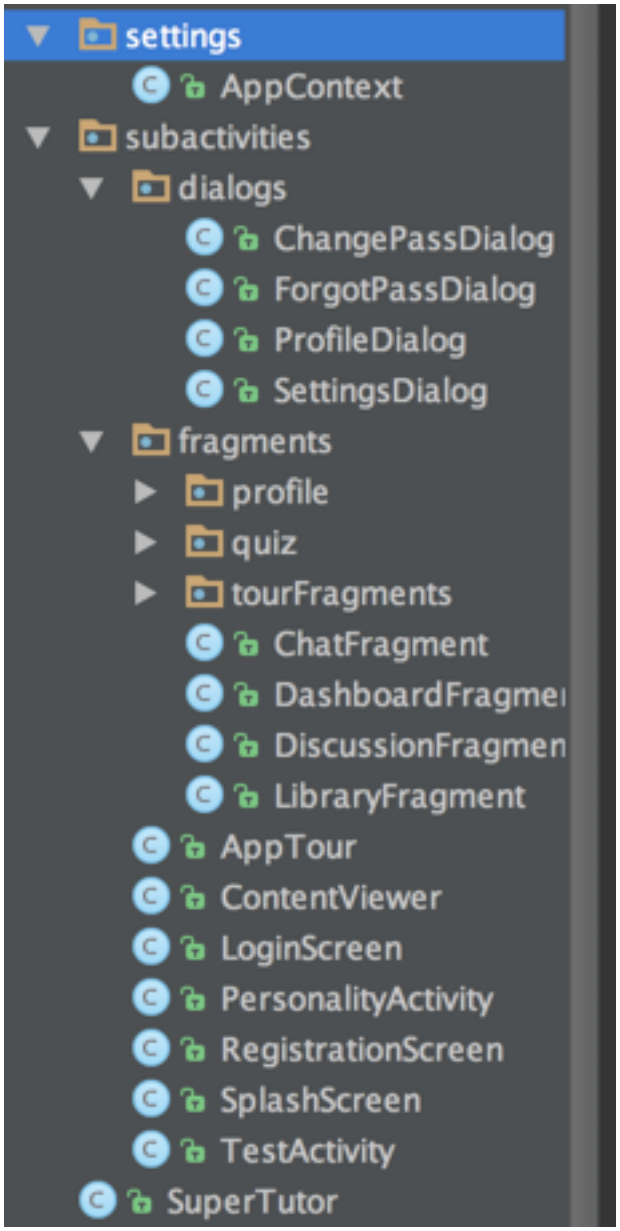
Appendix V – App File Architecture






internal
enums
XQuestions
XStatus
XTour
XType
library
AppLibrary
DiscussionBoard
machine
AlgorithmValue
MachineAlgorithm
security
Crypter
testing
test_wrappers
Questionnaire
Quiz
utils
Utilities
wrappers
boards
interfaces
tour
FollowerStat
ProfileFeed
Registration
Subject
TestQuestion
Topic
User


network
gcm
GCMBroadcastRece
GCMNotificationSer
managers
BoardManager
GCMRegistration
LoginManager
ProfileManager
RegistrationManage
tasks
AddFollower
ChangePasswordTa
FeedDownloader
FileDownloader
FollowerDownloade
ForgotPasswordTas
ImageDownloader
Privatizer
SubjectDownloader
SubmitReply
TestDownloader
UpdateLearnerTasl
Network


settings
AppContext
subactivities
dialogs
ChangePassDialog
ForgotPassDialog
ProfileDialog
SettingsDialog
fragments
profile
quiz
tourFragments
ChatFragment
DashboardFragme
DiscussionFragmen
LibraryFragment
AppTour
ContentViewer
LoginScreen
PersonalityActivity
RegistrationScreen
SplashScreen
TestActivity
SuperTutor