COMP6239 Mobile App Development Assignment

Yiu Ting Lai*, Jun Zhou†
Email: *ytl1u18@soton.ac.uk, †jz10n18@soton.ac.uk

1. Introduction

This is a technical report describes our development of the assignment of COMP6239. Our development works begin by conducting a requirement analysis to the scenario written on the course sheet. Secondly, designs of screen pages and back-end data infrastructure are done in parallel. Implementation and software testing works are followed by next. Finally, the back-end systems are published to AWS for public accessible, and the app is submitted to Handin system.

2. System Analysis

A. Functional Requirements

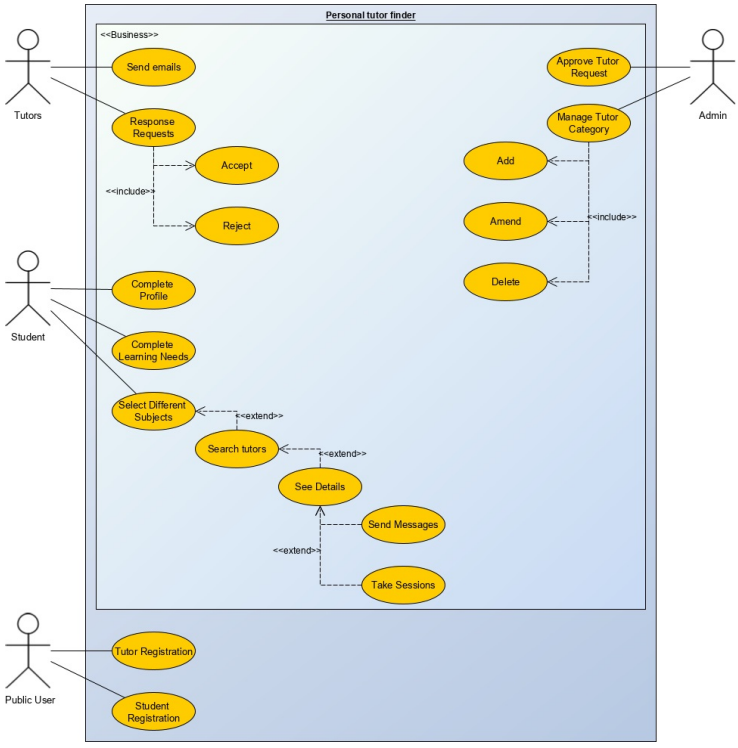

Fig. 1: The UML Use Case Diagram

Based on the scenario specified in the coursework sheet, functional requirements are extracted and listed using an UML Uses Case diagram. There are 4 actors inside the application, including students, tutors, administrators, and public users. Public users are able to register as either students or tutors. Students can search, acquire details, send private messages, and book to tutors based on their selected subjects. Administrators have privileges to create and remove a particular subject, and can approve tutors from registrations. Finally, an approved tutors can receive messages, take booking requests and confirm to appointments. Figure 10 describes the functionality of different actors inside the personal tutor finder app.

In this project, we also implemented some advanced features to improve our UX. Student can search tutors based on a postcode, and messages are sent in Real-time and the receiver could be notified immediately. A cloud-based back-end system is acting as the backbone and supporting the functionality of the app.

B. Non-Functional Requirements

By the coursework sheet, handling of navigation buttons and error reporting would be necessary. Additionally, the appropriate uses of colours should be easily readable and not be messing and confusing. Good uses of complementary colours could give a contrast between the foreground and background [1]. The details of our achievements could be found in section 6.

C. Assumptions

The provided scenario on the coursework sheet did not specify the number of subjects that a tutor can supervised. In our submitted project, we assume that each tutor can supervised only a single subject in the system.

3. Software Architecture

The submitted prototype comes with 2 different parts, the front-end and back-end. The front-end provides a number of graphical user interfaces which are actually the screen pages, while the back-end

consists of a HTTP web server, a Socket message server, and a relational database.

A. *Front-End*

The front-end is actually implemented as the mobile app, it has totally 17 screen pages available for the 4 actors specified in section 2. Navigation between screen pages are concerned. Figure 2 describes the relationships and function calls across multiple screen pages.

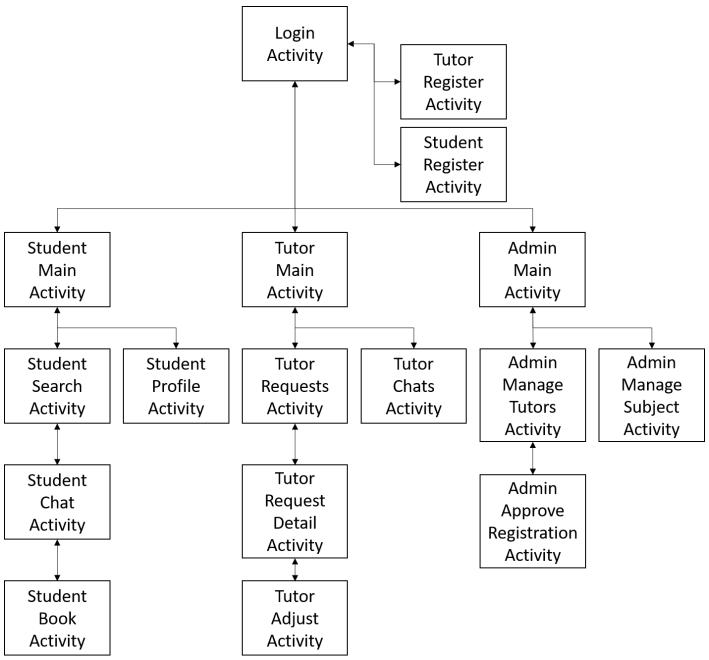

Fig. 2: The flow diagram of Android Activity

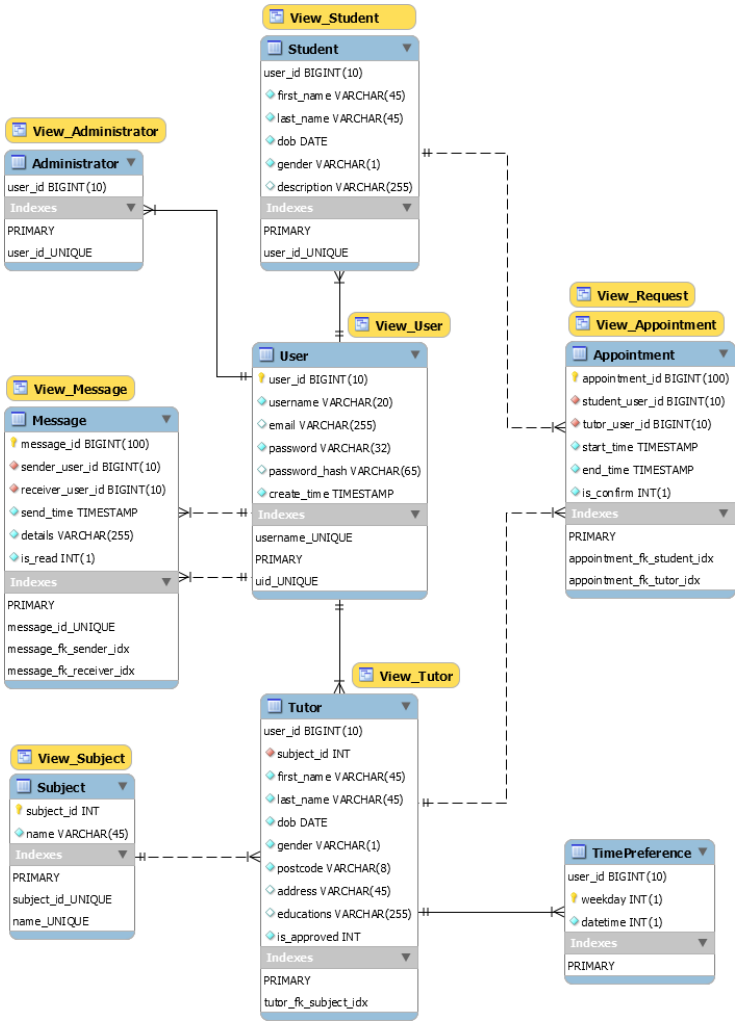

Fig. 3: The Entity Relationship Diagram of Database

B. *Back-End*

Our back-end database is designed using a relational-based structure, where a table represents an entity, an entity is the blueprint of its instances, and the lines between entities describe their relationships. Figure **??** is an entity relationship diagram, which illustrates the infrastructures of our data.

Apart from the database, there would be some web APIs which are acting as the bridge between the mobile app and the database. The web APIs consist of around 20 interfaces for receiving HTTP requests, including the types of GET, POST and DELETE. The front-end send a HTTP requests for data mediation rather than directly accessing the database. Our available web APIs are denoted in figure 4.

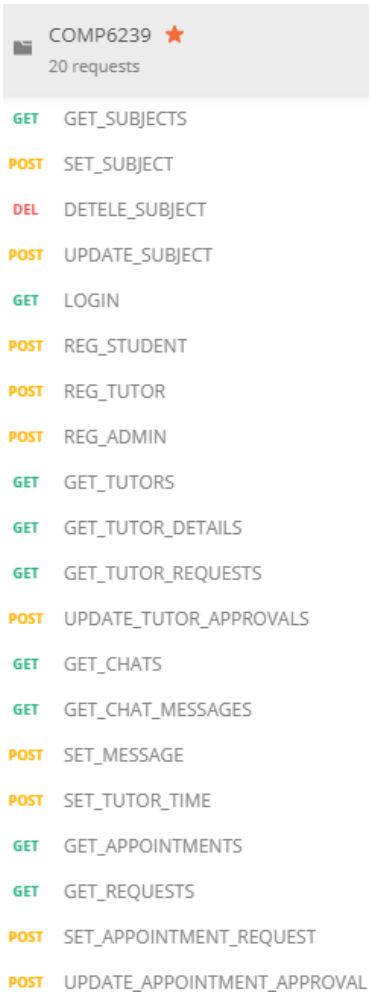

Fig. 4: The Back-End Web APIs

4. Technologies and Techniques

A. Software Development Model

Since the scenario is fixed and no amendments were issued through emails, the requirements will not be changed during the development stage. The Waterfall model is used in our development of out prototype. This coursework begins with requirement analysis, followed by system design and implementation. Integration and Testing is the next stages. Finally, the prototype is submitted through the Handin system. [2]

B. Integrated Development Environment

As mentioned in the lectures, our mobile would be targeting on Android devices. Android Studio is powered by IntelliJ, which is also the official IDE for developing a native mobile application with the completed library support on Android [3].

C. Back-end Database

The back-end database is designed using MySQL Workbench, which provides the features of forward engineering by modelling the infrastructure using an entity relationship diagram, and it converts the model into SQL scripts for database creation [4]. It visualises the data structures, and the front-end and back-end API section can start the development works before the actual creation of the back-end database. It accelerates our process of development as well.

D. Real-time Message Notification

To achieve the Real-time communication between tutors and students, we considered to use TCP socket, which is a network communication mechanism enabling sending and receiving information between devices. It is actually an interface in network communication. The skeleton of our code is modified from an online tutorial in our prototype [5], where both the client and server side are written in Java.

The two sides of socket communication are server and client. The server need to be assigned port while clients need IP address and port of server. In the implementation of real-time chat in our application, the socket runs on the server side and continuously listening the special predefined port, while the socket client would be in motion when application users trigger the chatActivity page in the application.

E. Interface of Web APIs

Before we finalized the actual interfaces between front-end and back-end, modeling of the back-end API would be necessary. Throughout this project, we use SwaggerHub to design our API. It provides a platform for us to work collaboratively.

F. Software Testing

By ensuring the correct information is transferred to, save, and returned from the back-end to front-end, testing of the APIs would be vital. Postman is an integrated tools for testing the communication between front-end and back-end, it simplifies the workloads for repetitive inputs and button clicks on the front-end [6].

5. Environment of Hosting

Our back-end systems are hosted using the Amazon AWS, which used a EC2 virtual machine for the web APIs and notification server, and a RDS for hosting a MySQL database. A GitHub url is also provided for publishing the our latest version of our project. Table I shows our hosting environments and its version of platform.

Hosting	Access URL	Port	Platforms
AWS EC2	http://35.178.209.191:80/COMP6239/server/api/	HTTP (80)	PHP 8.0, hosting the Web APIs
AWS EC2	35.178.209.191:9999	9999	Socket for Real-time notifications and messages
AWS RDS	steedzh.cb5bdohx0yuo.eu-west-2.rds.amazonaws.com	3306	MySQL 5.6, hosting the database of the app
GitHub	https://github.com/SteedZH/Android	-	Project repository

TABLE I: Permissionless v.s. Permissioned blockchain

6. UI and UX Design

The implementation of our screen pages follows to the Android Design Principles [7], which is defined by the Android User Experience Team.

A. Simplify My Life

1) Always showing where you are: Our app gives the user confident to know their pathway of using the app. For example, the search for tutors on studentSearchActivity can return from the details page of a tutor to their previous selected subject page but not losing their chosen subject, or even return back to the parent student main page. Figure 5 describes the good implementation of back navigation.

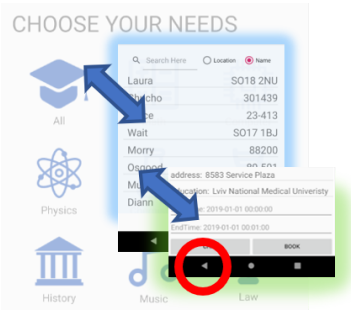

Fig. 5: The home screen of student to search by subjects

2) Pictures are faster than words: By using icons for choosing subjects to search tutors, student can easily get the ideas and it draws an attention to them. Figure denotes the

Fig. 6: The home screen of student to search by subjects

3) Users have final say: Just in case when the user mistakenly hit the back navigation button on their main screen page, a confirm dialog would be appeared to double check their decisions. A user can undo by clicking the cancel button, rather than logout and returning to the login page directly after the login or back navigation button is clicked. Figure 7 shows the confirmation dialog box for logout.

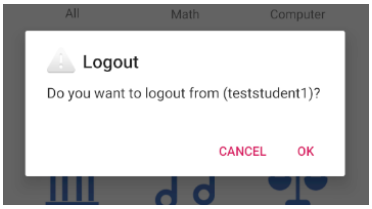

Fig. 7: The confirm dialog of logout

4) Appropriate Input Methods: For inputting information with format of limitations or particular choices of values, there should be a regularised input tool rather than typing plain texts. Considering the input of date-time, a plain input texts could lead to the confuse of formatting for end-user. Also they might be providing the wrong date by typos or careless mistakes. Having a regularised date-time picker could improve the UX, where it provides a human readable interface for inputting the date and time. Figure **??** demonstrates our data-time picker when a student is making an appointment request. The code is modified from an online tutorial and credited to Anupam's works [8].

Fig. 8: The date-time picker modified from Anupam's code [8]

5) Only show what you need when you need it: As mentioned in section 2, there are 4 different actors inside the app. Students, Tutors, and Administrators will have their own roles and permissions to some dedicated features, while they are also not available for Public Users. Showing the corresponding functions at the appropriate screen would be our most concern. For example, an Administrator can manage the list of subjects that are available for tutoring, but a tutor cannot create a new or remove to an existing subject. From figure 9, the feature of managing subjects should be available only in the main page of Administrator, where it should not be showing in other actor's menu.

Fig. 9: The Differences Between an Admin and Student Menu

B. Make Users Feel Amazing

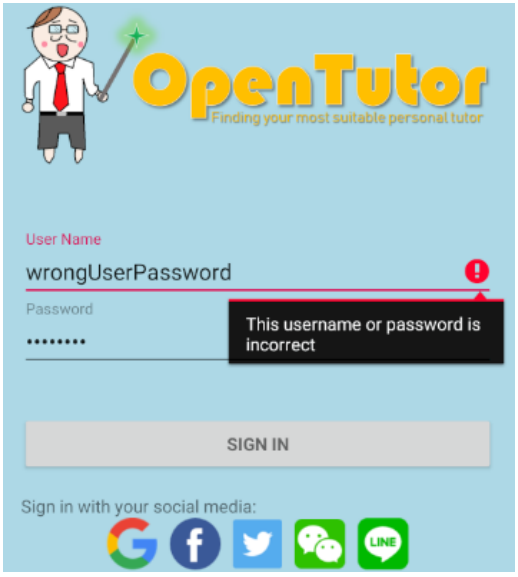

Fig. 10: The error notify in our app

1) Prompt to make corrections:

REFERENCES

[1] M. Patkar, "Learn the basics of color theory to know what looks good," lifehacker.com, 7 2014. [Online]. Available: https://lifehacker.com/learn-the-basics-of-color-theory-to-know-what-looks-goo-1608972072

[2] S.Balaji and D. M. S. Murugaiyan, "Waterfall vs. v-model vs. agile: A comparative study on sdlc," in *International Journal of Information Technology and Business Management Vol.2 No. 1*, Jun 2012. [Online]. Available: http://jitbm.com/Volume2No1/waterfall.pdf

[3] "Android studio." [Online]. Available: https://developer.android.com/studio

[4] "Mysql workbench - enhanced data migration." [Online]. Available: https://www.mysql.com/products/workbench/

[5] xueyuediana, "Transfer of notifications between server and android applications. [translated from original language in chinese]," CSDN, Sep 2015. [Online]. Available: https://blog.csdn.net/xueyuediana/article/details/48181819

[6] "Postman — api development environment." [Online]. Available: https://www.getpostman.com/

[7] "Android design principles," Android User Experience Team. [Online]. Available: http://www.androiddocs.com/design/get-started/principles.html

[8] A. Chugh, "Android date time picker dialog," JournalDev. [Online]. Available: https://www.journaldev.com/9976/android-date-time-picker-dialog