



ASHESI UNIVERSITY

NEARBY TEACHER FINDER: A LOCATION-BASED PRIVATE TEACHER RECOMMENDATION SYSTEM FOR PUPILS

APPLIED PROJECT

B.Sc. Computer Science

Tony Chisenga

2021

ASHESI UNIVERSITY

**A LOCATION-BASED PRIVATE TEACHER RECOMMENDATION
SYSTEM FOR PUPILS**

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

Tony Chisenga

2021

DECLARATION

I hereby declare that this applied project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:



.....
Candidate's Name:

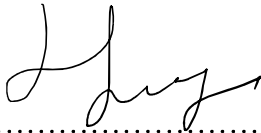
Tony Chisenga

.....
Date:

13/05/2021
.....

I hereby declare that preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of Applied Project laid down by Ashesi University.

Supervisor's Signature:



.....
Supervisor's Name:

Eric Ocran

.....
Date:

14/05/2021
.....

Acknowledgement

I thank God for the strength he has given me to work on this capstone project. All glory and honor to him.

I want to express my sincere gratitude to MasterCard Foundation for their unwavering support and care towards me, especially when our learning was remote.

To my supervisor and all the lecturers who rendered your support to me, I appreciate you. Finally, thanks to all my friends who always gave me words of encouragement when things got tough.

Abstract

This project seeks to address challenges pupils face when looking for a teacher within their location who offers extra lessons in a particular subject. It also aims to address the primary challenge teachers who provide tuitions face, i.e., publicizing themselves to pupils in need of their service. In Zambia, just like other countries, the practice of taking lessons outside the class schedule has gained ground among secondary school pupils. They choose to do this to improve their academic performance, which becomes poor due to inefficiencies in the teaching-learning process in some schools. In this project, the solution is a location-based system that discovers teachers for a particular subject based on the pupil's location. In addition, pupils can view teacher profiles and interact with them.

Table of Contents

| | |
|--|-----|
| DECLARATION | i |
| Acknowledgement | ii |
| Abstract | iii |
| Chapter 1: Introduction | 1 |
| 1.1 Background | 1 |
| 1.2 Related work | 1 |
| 1.3 Proposed solution | 3 |
| Chapter 2: Requirements | 4 |
| 2.1 Overview | 4 |
| 2.2 Use case diagrams | 4 |
| 2.3 Flowcharts | 8 |
| 2.4 Functional requirements | 11 |
| 2.4.1 User requirements | 11 |
| 2.4.2 System requirements | 11 |
| 2.5 Non-functional requirements | 12 |
| Chapter 3: Architecture and design | 13 |
| 3.1 Overview | 13 |
| 3.2 Presentation tier | 14 |
| 3.3 Application tier | 14 |
| 3.4 Data tier | 15 |
| Chapter 4: Implementation | 17 |
| 4.1 Presentation tier | 17 |

| | |
|---|----|
| 4.1.1 Languages | 17 |
| 4.1.1.1 HTML & CSS | 17 |
| 4.1.1.2 JavaScript | 18 |
| 4.1.2 Frameworks | 18 |
| 4.1.2.1 Flutter | 18 |
| 4.2 The application tier..... | 18 |
| 4.2.1 Languages | 18 |
| 4.2.1.1 JavaScript | 18 |
| 4.2.1.2 Dart | 19 |
| 4.3 The data tier | 19 |
| 4.3.1 Firebase Firestore | 19 |
| 4.4 Tools & APIs | 19 |
| 4.4.1 Google Geolocation API | 19 |
| 4.4.2 GPS | 19 |
| 4.5 System components | 20 |
| 4.5.1 Signup and Login | 20 |
| 4.5.2 Finding a teacher near pupil's location | 20 |
| 4.5.3 The administrator | 21 |
| Chapter 5: Testing | 23 |
| 5.1 Development testing | 23 |
| 5.1.1 Unit testing | 23 |
| 5.1.2 Component testing | 26 |
| 5.1.3 System testing | 29 |

| | |
|---|----|
| 5.2 User testing | 29 |
| 5.3 Form testing | 29 |
| Chapter 6: Conclusion, Limitations, and Future works | 31 |
| 6.1 Limitations | 31 |
| 6.2 Future works | 31 |
| 6.3 Conclusion | 32 |
| References: | 33 |
| Appendices | 35 |
| Appendix A: System testing code snippet..... | 35 |
| Appendix B: Login UI..... | 37 |
| Appendix C: Subjects screen..... | 38 |
| Appendix D: Some responses by participants during requirement analysis..... | 40 |

List of tables

| | |
|--|----|
| Table 5.1: Summary of unit testing of the Nearby Teacher Finder..... | 25 |
| Table 5.2: Summary of the component tests conducted and their results..... | 28 |

List of figures

| | |
|---|----|
| Figure 2.1: Use case diagram for the pupil..... | 5 |
| Figure 2.2: Use case diagram for the teachers and system administrator..... | 7 |
| Figure 2.3: Flowchart for the pupils..... | 8 |
| Figure 2.4: Flowchart for the teachers..... | 9 |
| Figure 2.5: Flowchart for the system administrator..... | 10 |
| Figure 3.1: The three-tier client-server architecture..... | 13 |
| Figure 3.2: The classes and methods in the Nearby Teacher Finder..... | 15 |
| Figure 4.1: Code that makes a user an admin..... | 22 |
| Figure 5.1: Unit tests for input validation functions in signup and login..... | 24 |
| Figure 5.2: Results for the unit test of the signup and login input validation..... | 24 |
| Figure 5.7: Form testing to ensure correct details are provided..... | 30 |

Chapter 1: Introduction

1.1 Background

Education is one of the essential tools contributing to building a better nation. Since Independence, the education system in Zambia has undergone many reforms[1]. With the reforms, authorities hope to discover the best ways to instill knowledge in pupils and aim to archive sustainable development goal(SDG) number 4 by the United Nations, i.e., to ensure inclusive and equitable quality education and promote lifelong learning opportunities for all[2]. Despite all the efforts, the academic performance of secondary school pupils continues to be low[3]. For the pupils, one of the contributing factors to this is the large class sizes, especially in public schools, which make it difficult for teachers to handle, resulting in the teaching-learning process being inefficient.

Furthermore, pupils do not have an opportunity to have one-on-one sessions with teachers to ask about concepts that they do not understand in class. These challenges cause poor academic performance for some pupils since they cannot keep up with the entire class. To avoid this, pupils have identified taking extra lessons as a way to improve their performance. However, the main challenge is finding skillful teachers who offer extra lessons and are close to them. Hence the need for a platform to facilitate this process.

1.2 Related work

This section discusses already-existing solutions to the problem in question developed in different parts of the world. These are insightful as they present different approaches to tackling this problem. The common ones are using GPS to locate the user and find the distances between the users using a distance algorithm. Another way is by using machine learning techniques that

perform classification of the profiles or characteristics of both the teachers and pupils and then link them based on similarity.

Warit Taveekarn et al developed a system called FindMyTutor[4]. The application uses location, gender, age, and rating to recommend teachers to a pupil in Thai. The system allows pupils to find a teacher in their area, and it also provides them a messaging feature that enables them to chat on the platform. The main focus of the application is to eliminate the intermediary between the teachers and the pupils in offering tuition. There are recommendation agencies where teachers who want to offer extra lessons have to pay a fee for them to be recommended to students in Thai.

Apart from eliminating the intermediary between pupils and teachers, the application was developed only for devices that run on the Android operating system. Although the developers argue that android is the most commonly used operating system, IOS has also become common. Hence, the need for the application to be developed with a cross-platform language.

In Karachi, Muhammad Saad et al. also developed a similar system called Smart Tutor Finder[5]. The application was built only for devices that run on the Android operating system. Just like FindMyTutor, the main goal of this application was to eliminate a third party in the process of a pupil finding a tutor. Like in Thai, teachers who want to offer extra lessons are supposed to pay a certain amount to an agency that will recommend them to pupils. In recommending teachers, the application only focuses on the location and the rating of the teachers. With this system, the teachers are not recommended as per the subject they teach, which might cause inconveniences for some of the students by just displaying all teachers for different subjects in their location.

Similarly, Dolly Panchal et al. developed the Android Application for finding Tutors using data mining techniques[6]. The system is an android based application as well that was built to link pupils to teachers in Mumbai. One of the objectives of this application is to enhance teacher-pupil relationships. Just like the Smart Tutor Finder, this system does not provide a list of subjects in which a pupil may need a teacher for extra lessons. This system uses a classifier algorithm called the Naïve Bayes to find similar characteristics between the profile of the teacher and that of the pupil. In short, it matches a particular teacher to a pupil in a location-based on similarity identified by the Naïve Bayes.

The three solutions discussed above have contributed heavily to finding a solution to the problem under discussion. Each uses a unique approach to helping pupils find a teacher for extra lessons in their area and interact with them. However, the solutions do not consider the security of the pupils. They do not provide a verification feature that would enable the system administrator to ensure that the person who has registered as a teacher is a genuine teacher.

1.3 Proposed solution

This project involves building the Nearby Teacher Finder. It is a location-based recommendation system(both mobile and web-based application) that discovers teachers who offer private tuition and presents them to secondary school pupils who need the service. The recommendation is based on the location of the pupil. When a pupil signs up or signs in on the system, they are presented with a list of subjects. From the list, a pupil picks a subject in which they need extra lessons. After that, a list of teachers for the chosen subject is displayed, including the teachers' distance from the pupil. To see a teacher's profile or connect with them, the pupil must select them from the list.

Chapter 2: Requirements

2.1 Overview

This process involves establishing the goals, services, and constraints of a system by consulting its users[7]. During this process, two primary users were identified, i.e., pupils searching for a teacher offering extra lessons and the teachers themselves. The requirements for this system were gathered through interactions, i.e., interviewing the identified users to ensure that the solution meets their needs.

After a series of interactions with the users, it was realized that the system needed to have an administrator. It also required to have two parts. The first one is a version for the pupils, allowing them to sign up, find teachers offering a subject in their area, and interact with them. The other version is for the system administrators and teachers. This version would allow the teachers to sign up, set their profiles, choose the subjects they would like to offer lessons for, and interact with pupils. It would also enable the administrator to confirm that everyone who signs up as a teacher is genuine and perform other admin roles.

The following subsections of this chapter discuss the use case diagrams, flowcharts, functional and non-functional requirements.

2.2 Use case diagrams

A use case diagram depicts the interactions between the users of a system and the system itself. Below are the use case diagrams for the Nearby Teacher Finder.

The pupil: The pupil logs in to the system. If they do not have an account, they must sign up and provide some of their credentials. Upon successful log in, they can view various subjects displayed

to them. From the list of subjects displayed, the pupil selects one of them in which they want a teacher for extra lessons. A list of all the teachers near their location is then displayed to them. After this, they can then view all the teachers' profiles and initiate conversations with them using the platform. Figure 2.1 below indicates these interactions.

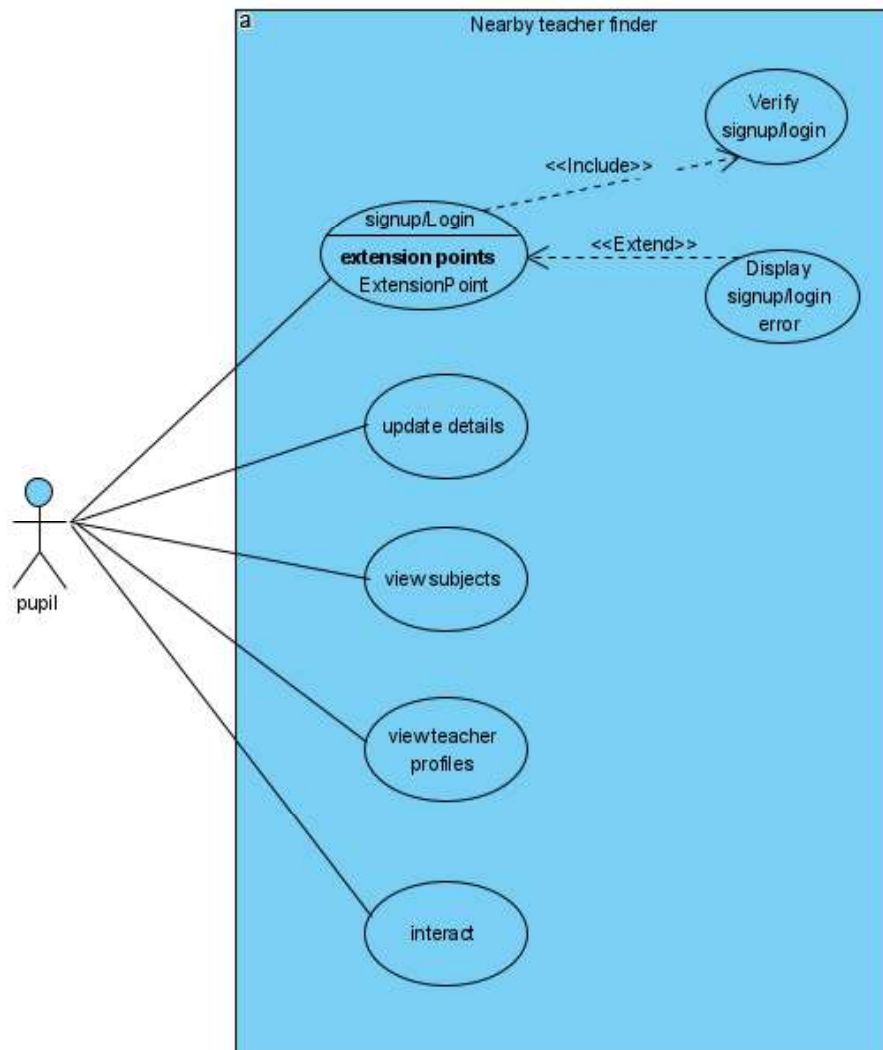


Figure 2.1: Use case diagram for the pupil

The teacher: The teacher logs in to the system. If they do not have an account, they must sign up. After they have signed up, they will be required to provide credentials that will allow for their verification as a genuine teacher by the system administrator. Before the administrator responds to their request, they have no access to any of the functionalities on the system. After being approved, they can update their details, select subjects in which they offer extra lessons, and interact with pupils interested in them.

The system administrator: The admin of the system logs in. Upon successful login, they can check for teachers who have signed up to be offering extra lessons, approve or decline their request. They also have the privilege to add or remove a subject from the system.

The figure below shows the interactions of teachers and system administrators with the system.

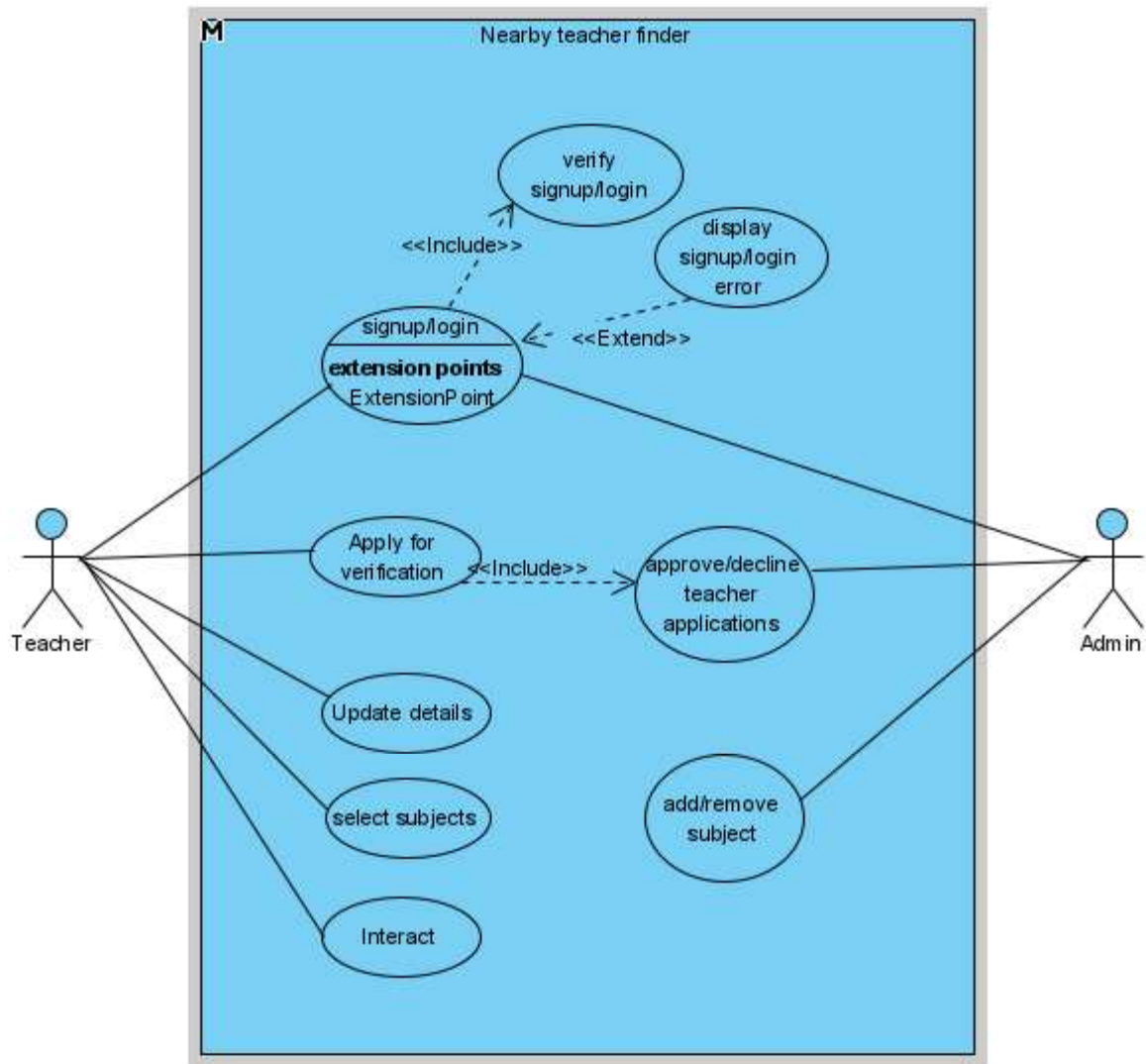


Figure 2.2: Use case diagram for the teachers and system administrator

2.3 Flowcharts

A flowchart is a diagram that shows how data flows in a system and how decisions are made to control events. Below are the flowcharts for the pupils, teacher, and admin described in subsection 2.2 above.

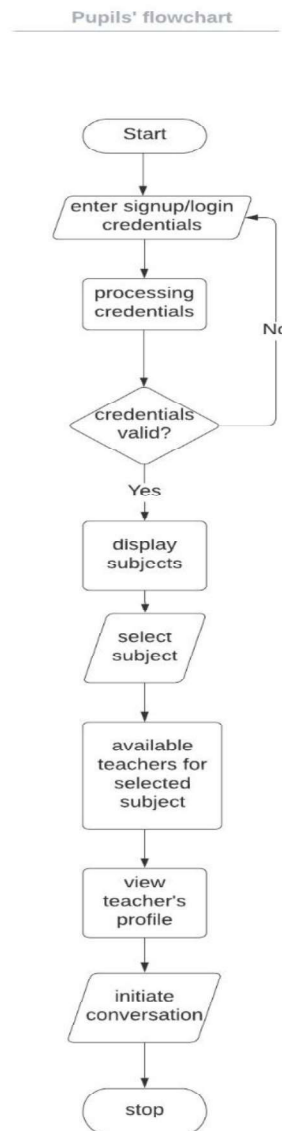


Figure 2.3: Flowchart for the pupils

Teachers' flowchart

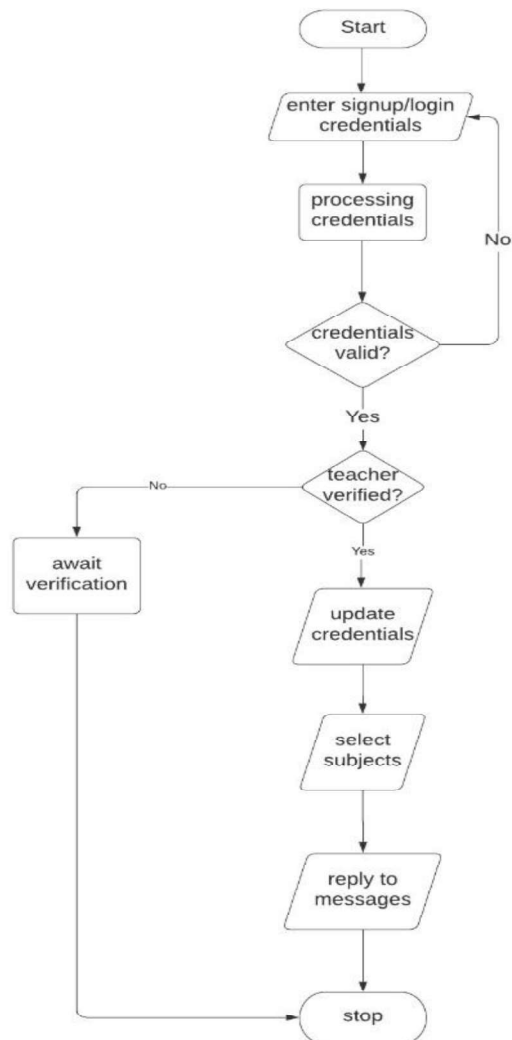


Figure 2.4: Flowchart for the teachers

Admin's flowchart

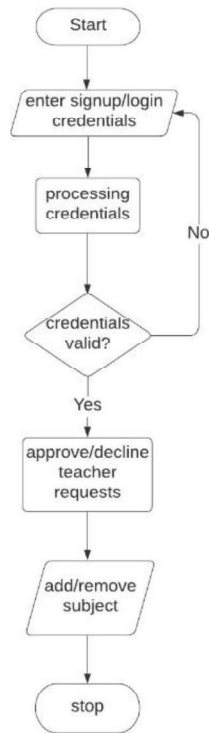


Figure 2.5: Flowchart for the system administrator

2.4 Functional requirements

These describe what the system should do. They are divided into user requirements and system requirements.

2.4.1 User requirements

User requirements describe what the user does with the system. The following are the user requirements for the Nearby Teacher Finder.

- I. The user should be able to sign up and log in to the system.
- II. The user should be able to update their credentials.
- III. The pupils should be able to view subjects in which they are looking for a teacher.
- IV. The pupils should view profiles of teachers who offer extra lessons for a particular subject and are close to them.
- V. The pupils should be able to interact with teachers on the platform.
- VI. The teachers should be able to apply for verification of their authenticity by the system administrator.
- VII. The teachers should be able to select subjects in which they offer lessons.
- VIII. The teachers should be able to reply to messages from pupils.
- IX. The system administrator should be able to add or remove a subject from the system.
- X. The admin should be able to approve or decline requests by teachers.

2.4.2 System requirements

System requirements are the specifications that the system must have to satisfy the user requirements. The following are the system requirements for the Nearby Teacher Finder.

- I. The system should allow users to sign up and log in to it.
- II. The system should allow users to update their credentials.
- III. The system should be able to display subjects with available teachers in the pupil's location.
- IV. The system should allow pupils to view profiles of teachers near their location.
- V. The system should allow teachers and pupils to send each other messages.
- VI. The system should allow teachers to select subjects in which they offer lessons.
- VII. The system should allow teachers to apply for verification.
- VIII. The system should allow the admin to approve or decline requests by teachers.
- IX. The system should allow the admin to add or remove subjects.

2.5 Non-functional requirements

These are requirements that are not directly concerned with the specific services delivered by the system to its users. The following are the non-functional requirements for the Nearby Teacher Finder.

- I. *Security*: the system and the data of users should be protected against attacks.
- II. *Usability*: the system should be easy to use.
- III. *Performance*: the system should be fast in returning the results that the user requests.

Chapter 3: Architecture and Design

3.1 Overview

This chapter discusses the architecture used in the system, i.e., the 3-tier client-server architecture. This architecture organizes an application into three logical and physical computing tiers[8], namely the presentation tier, the application tier, and the data tier. Below is an image that shows this architecture.

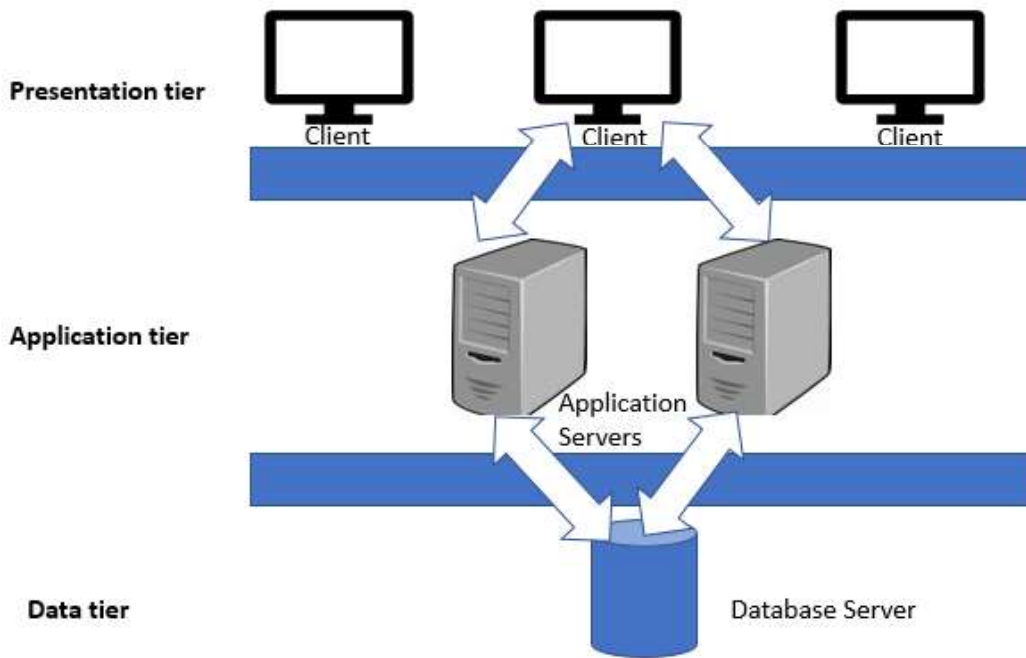


Figure 3.1: The three-tier client-server architecture

3.2 The Presentation Tier

This tier enables users to interact with the application. It also facilitates the display and collection of information from them. For the Nearby Teacher Finder, this layer will allow teachers to view available subjects, select a subject in which they offer lessons, update their credentials and view messages. It will also allow the admin to view all requests by teachers to be approved or declined on the system, add or remove a subject. The pupils will be able to view subjects, view teachers and their profiles, and view text messages sent to them.

The interface is kept simple and consistent, i.e., it maintains the same design of buttons, input fields, and colors on all its pages.

3.3 The application tier

The application tier is responsible for processing data that are collected from the presentation tier. It contains the business logic, i.e., the code that handles communication between the database and the user interface or presentation tier. It also performs deletion and modification of data in the data tier. Below is a class diagram that shows the classes and methods that perform different operations on the data in the Nearby Teacher Finder.

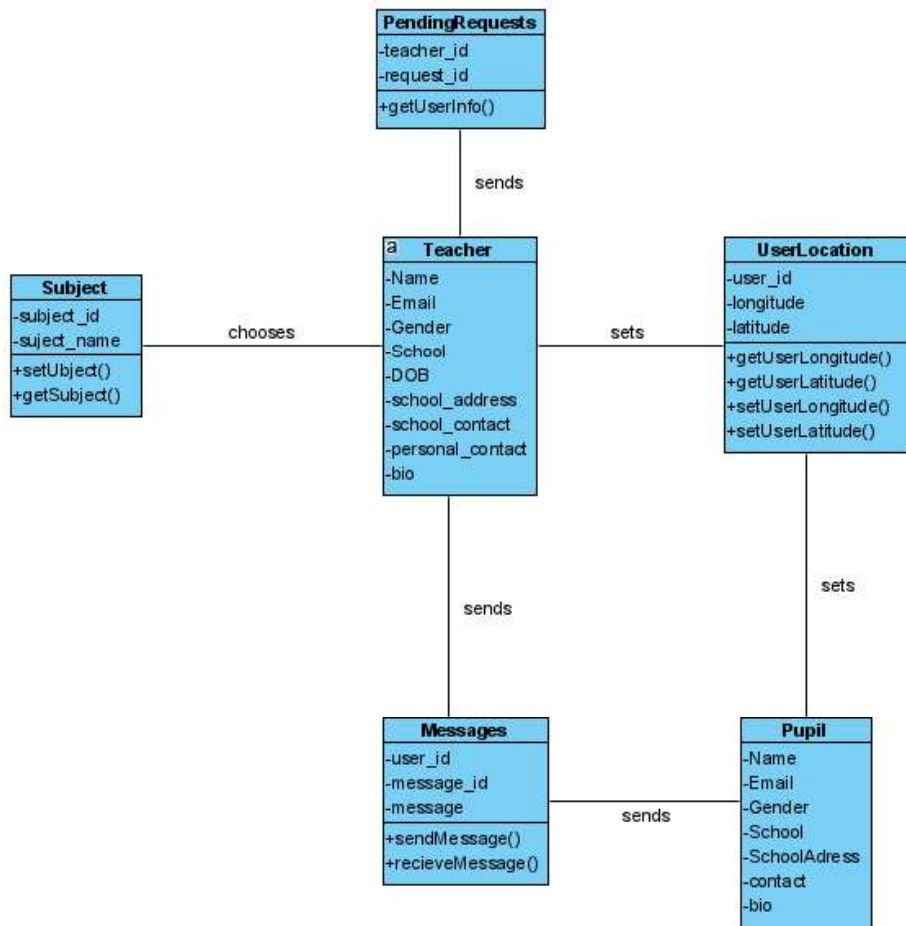


Figure 3.2: The classes and methods in the Nearby Teacher Finder

3.4 The data tier

The data tier is where the information processed by the application is stored and managed. The Nearby Teacher Finder uses a NoSQL database called Firebase Firestore. This means that the storage of data is non-tabular. Firebase Firestore uses collections to store data. The following are the collections that the Nearby Teacher Finder has.

Pupils: This collection stores the information of the pupils on the system, which includes the names, date of birth, email, gender, and school name.

Teachers: This collection stores the information of all the teachers that the admin has approved on the system, which includes the names, date of birth, email, gender, and school name.

Subjects: This collection stores all the subjects that the admin adds to the system.

PendingRequests: Stores the information of all the teachers who just signed up and are awaiting verification by the admin. The admin uses the data stored here to crosscheck with the institution indicated by the teacher if they are genuine. This information includes the teacher name, subject that the teacher teaches, the contact information of the school where they teach, and the teacher's contact information.

Userlocation: This stores information that identifies the user's location, i.e., their longitude and latitude.

Chatroom: This stores conversations between the teachers and the pupils.

Chapter 4: Implementation

As discussed in chapter three, the Nearby Teacher Finder follows the 3-tier client-server architecture. This section discusses the technologies, tools, and APIs used at every layer of the architecture.

4.1 The Presentation Tier

As discussed in chapter 3, the presentation tier is the user interface of the application. The following are the languages and tools used to build this layer.

4.1.1 Languages

4.1.1.1 HTML and CSS

HTML is a markup language that is used to create web pages. It is used to organize and format a document and makes possible the creating and structuring of sections, paragraphs, headings, links, and blockquotes of a web page. With HTML, the browser can display text as elements and load images.

CSS is a language that is used to describe how elements of a markup language like HTML should be presented on a webpage. This is made possible with the use of components such as fonts, colors, and layouts. With CSS, a web application can adapt the presentation of its pages to different devices, such as large screens and small screens.

For the Nearby Teacher Finder, HTML and CSS have been used to create user interface layout and appearance.

4.1.1.2 JavaScript

JavaScript is a programming language used in web development. It is powerful because its elements enable a webpage to be interactive. This programming language can be used for both frontend and backend programming.

In the presentation layer, JavaScript has been used to display errors and other messages to the user.

4.1.2 Frameworks

4.1.2.1 Flutter

Flutter is an open-source UI software development kit that is used to develop mobile and web applications. The presentation layer of the mobile version of Nearby Teacher Finder uses widget components of flutter to create the user interface.

4.2 The Application Tier

This tier uses business logic to perform several operations like processing information collected from the presentation tier and add, delete, or update information in the data tier.

4.2.1 Languages

4.2.1.1 JavaScript

As stated in section 4.1.1.2, JavaScript can be used for server-side programming. For the web version of the Nearby Teacher Finder, the programming language has been used to query the data tier and insert data into it.

4.2.1.2 Dart

Dart is an open-source, object-oriented programming language for developing both web and mobile applications. It is an easy-to-learn language whose programs run faster. The programming language has been used to query and insert data into the data tier for the mobile version of the system since it is used to code flutter applications.

4.3 Data tier

This is where the data processed by the application tier is stored and managed.

4.3.1 Firebase Firestore

Firebase Firestore is a NoSQL database for mobile, web, and server development. Firebase is powerful because it uses real-time listeners to keep data in-sync across applications. It has been used to store all data for the Nearby Teacher Finder.

4.4 Tools and APIs

4.4.1 Google geolocation API

Geolocation is an API used to get a person's location using information about cell towers and WiFis in the person's location. For this system, it is used to get the longitude and latitude points of the users.

4.4.2 GPS

GPS stands for Global Positioning System. Among other things, this service provides users with positioning and navigation. In the system, it is also used to track the location of the pupils using the mobile application.

4.5 System components

This section explains how some system components were implemented.

4.5.1 Signup and Login

The Nearby Teacher Finder requires that its users have an account for them to use it. Using an email and password, the firebase authentication tool enables users to create an account if they do not have one or log in if they already have an account. The signup process is simple for the pupils. Upon providing their email and password, they can access other system functionalities like viewing subjects and checking profiles of teachers near them.

For the teachers, the process is longer. More information is collected during signup. This information enables the administrator of the system to physically crosscheck the details provided by the teacher with the school or institution where they work before approving their request. This ensures that only authentic teachers are registered on the system. Upon approval by the admin, the teacher is granted access to other functionalities of the system.

4.5.2 Finding a teacher near pupil's location

The goal of the system is to help pupils find teachers who offer lessons within their location. Upon login, the application requests permission to access the user's location through the GPS feature on the mobile phone. This will enable the application to identify the location of the user. When the user picks a subject in which they need a teacher for extra lessons, the application retrieves all the teachers for that subject, performs calculations of the distance, and retrieves all the teachers near the pupil's location. The distance is computed using the haversine formula[9]. The formula calculates the distance by taking the longitude and latitude values of two points or locations. The formula is as follows:

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot a \cdot \tan2\left(\sqrt{a}, \sqrt{(1-a)}\right)$$

$$d = R \cdot c$$

Where φ is the latitude, λ is the longitude, R is earth's radius(mean radius = 6,371km)

Note: Angles have to be in radians to pass to trigonometric functions.

4.5.3 The administrator

The system administrator approves or declines requests by teachers to use the system to publicize themselves to pupils. They also have the privilege to add and remove subjects to the system. This is made possible with the use of firebase cloud functions that enable a system to define rules for other users to have certain privileges or access to certain information that every user cannot access. Below is a snippet of the function defined to give other users certain privileges(make them admin).

```
const functions = require("firebase-functions");

// // Create and Deploy Your First Cloud Functions
// // https://firebase.google.com/docs/functions/write-firebase-
// functions
//
// exports.helloWorld = functions.https.onRequest((request, response)
=> {
//   functions.logger.info("Hello logs!", {structuredData: true});
//   response.send("Hello from Firebase!");
// });
const admin = require('firebase-admin');
admin.initializeApp();

exports.addAdminRole = functions.https.onCall((data, context) => {
```

```
//get user and add customer claim(admin)
return admin.auth().getUserByEmail(data.email).then(user => {
  return admin.auth().setCustomUserClaims(user.uid, {
    admin: true
  });
}).then(() => {
  return {
    message: `Success ${data.email} has been made admin`
  }
}).catch(err => {
  return err;
});
});
```

Figure 4.1: Code that makes a user an admin

Chapter 5: Testing

This chapter discusses various tests that have been done for the Nearby Teacher Finder. Testing of the system intends to show that it meets the functional and non-functional requirements. Furthermore, the activity helps discover situations in which the behavior of the software is incorrect, undesirable, or does not conform to its specification[7]. For the Nearby Teacher Finder, two types of testing have been conducted, namely development testing and user testing.

5.1 Development testing

This involves all testing activities of the program that are conducted during the development phase. It is carried out at three levels, namely unit testing, component testing, and system testing.

5.1.1 Unit testing

In unit testing, individual program units are tested. The focus is testing the functionality of objects or methods of a class. Using the flutter_test package for flutter, unit testing was conducted for the methods that ensure that the right inputs are provided during signup or sign-in functionalities. Below are the code snippets and the results.

```
import 'package:csappliedteacherapp/src/screens/authenticate/login.dart';
import 'package:flutter_test/flutter_test.dart';

void main() {
  test("", () {});

  test("empty email returns error string", () {
    var result = EmailFieldValidator.validate('');
    expect(result, "Email can\'t be empty");
  });
}
```

```

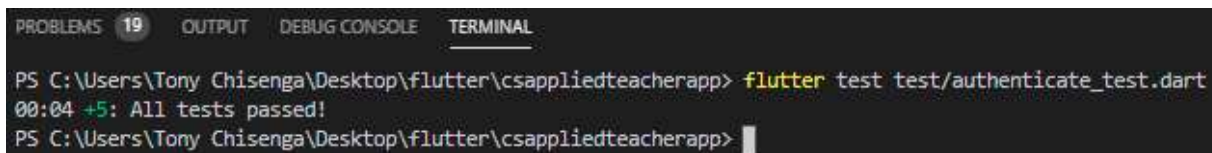
test("non-empty email returns null", () {
  var result = EmailFieldValidator.validate('email');
  expect(result, null);
});

test("empty password returns error string", () {
  var result = PasswordFieldValidator.validate('');
  expect(result, "Password should have more than 8 characters");
});

test("non-empty password returns null", () {
  var result = PasswordFieldValidator.validate('password');
  expect(result, null);
});
}

```

Figure 5.1: Unit tests for input validation functions in signup and login



```

PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\Tony Chisenga\Desktop\flutter\csappliedteacherapp> flutter test test/authenticate_test.dart
00:04 +5: All tests passed!
PS C:\Users\Tony Chisenga\Desktop\flutter\csappliedteacherapp>

```

Figure 5.2: Results for the unit test of the signup and login input validation

The code above shows four unit test cases for the signup and login functionalities. The table below summarizes the test cases and their results.

Table 5.1: Summary of unit testing of the Nearby Teacher Finder

| Test No. | Input | Expected result | Actual result | Comment |
|----------|--------------------------------|--|---|------------------------------------|
| 1 | no input(empty email field) | Empty email returns an error string | Empty email returned an error string “Email cant be empty” | The program unit works as expected |
| 2 | email | null (i.e., no error string) | null (i.e., no error string) | The program unit works as expected |
| 3 | no input(empty password field) | Empty password returns an error string | Empty password returned an error string “password should have more than 8 characters” | The program unit works as expected |
| 4 | password | null (i.e., no error string) | null (i.e., no error string) | The program unit works as expected |

5.1.2 Component testing

In component testing, the focus is testing the interfaces of various components. Since flutter is based on widgets, component testing may be referred to as widget testing in flutter and involves

testing widgets and how the user interface interacts with other widgets. For the Nearby Teacher Finder, the screens that have been tested include the login and signup, screens that display subjects and teachers, and the conversations screen. Below are the snippets of the code and their results.

```
import 'package:csappliedteacherapp/src/screens/home/subject_list.dart';
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

void main() {
  Widget makeTestTableWidget({Widget child}) {
    return MaterialApp(
      home: child,
    );
  }

  testWidgets("", (WidgetTester tester) async {
    //await tester.pumpWidget();
    SubjectsList subjectScreen = SubjectsList();
    await tester.pumpWidget(makeTestTableWidget(child: subjectScreen));
  });
}
```

Figure 5.3: Code snippet for component testing of the subjects screen

```
PS C:\Users\Tony Chisenga\Desktop\flutter\csappliedteacherapp> flutter test test/subjectlist_widget_test.dart
00:09 +1: All tests passed!
PS C:\Users\Tony Chisenga\Desktop\flutter\csappliedteacherapp> █
```

Figure 5.4: Result for component testing of the subjects screen

```
import 'package:csappliedteacherapp/src/screens/home/subject_tutors_list.dart';
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

void main() {
```

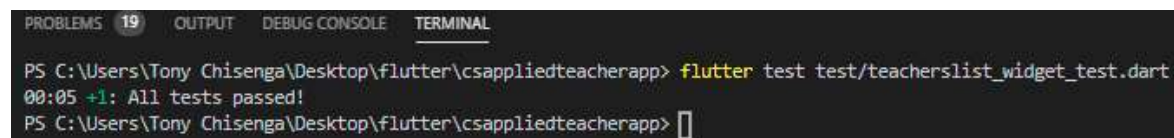
```

Widget makeTestTableWidget({Widget child}) {
  return MaterialApp(
    home: child,
  );
}

testWidgets("", (WidgetTester tester) async {
  //await tester.pumpWidget();
  TeachersList screen = TeachersList();
  await tester.pumpWidget(makeTestTableWidget(child: screen));
});
}

```

Figure 5.5: Code snippet for component testing of the Teachers screen



The screenshot shows an IDE interface with a terminal window. The terminal has tabs for PROBLEMS (19), OUTPUT, DEBUG CONSOLE, and TERMINAL. The terminal text shows a PowerShell prompt at 'C:\Users\Tony Chisenga\Desktop\flutter\csappliedteacherapp' where the command 'flutter test test/teacherslist_widget_test.dart' is executed. The output shows '00:05 +1: All tests passed!' followed by another PowerShell prompt.

```

PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\Tony Chisenga\Desktop\flutter\csappliedteacherapp> flutter test test/teacherslist_widget_test.dart
00:05 +1: All tests passed!
PS C:\Users\Tony Chisenga\Desktop\flutter\csappliedteacherapp> 

```

Figure 5.6: Result for component testing of the teachers' screen

Table 5.2: Summary of the component tests conducted and their results.

| Test No. | Component | Input(s) | Expected result | Actual result | Comment |
|----------|---------------|---------------------------|---------------------------|---------------------------|-----------------------------|
| 1 | Login | Email and password | Login successful | Login successful | Component works as expected |
| 2 | Signup | Email and password | Signup successful | Signup successful | Component works as expected |
| 3 | SubjectList | Subject in the search bar | Display list of subjects | Display list of subjects | Component works as expected |
| 4 | TeachersList | N/A | Display list of teachers | Display list of teachers | Component works as expected |
| 5 | Conversations | message | Send and display messages | Send and display messages | Component works as expected |

5.1.3 System testing

System testing involves integrating all the components of the system and testing the system as a whole. The focus of this activity is to see how components interact with each other. In flutter, this may be referred to as integration testing. See *Appendix A* for the code snippet and the results.

5.2 User testing

This type of testing requires the involvement of the users of the system. Here, the users interact with the system and advise on testing the system. For the Nearby Teacher Finder, seven users (four pupils and three teachers) participated in this process. The participants were able to signup, login, and use other functions of the system. Helpful feedback was gathered, and changes were made to the system. For the mobile application, the users emphasized the need for a search bar that would allow pupils to search for subjects not displayed on the screen. For the web application, the teachers advised that the UI be redesign because the initial one did not look attractive. See appendices B and C for the interfaces before and after user testing.

5.3 Form testing

The forms for the web application were tested to check and ensure that all the users provide the right data. If the user enters wrong input, error messages are shown to them, and they are asked to input the correct data before they proceed. The figure below shows an error displayed when the user enters the wrong credentials during login.

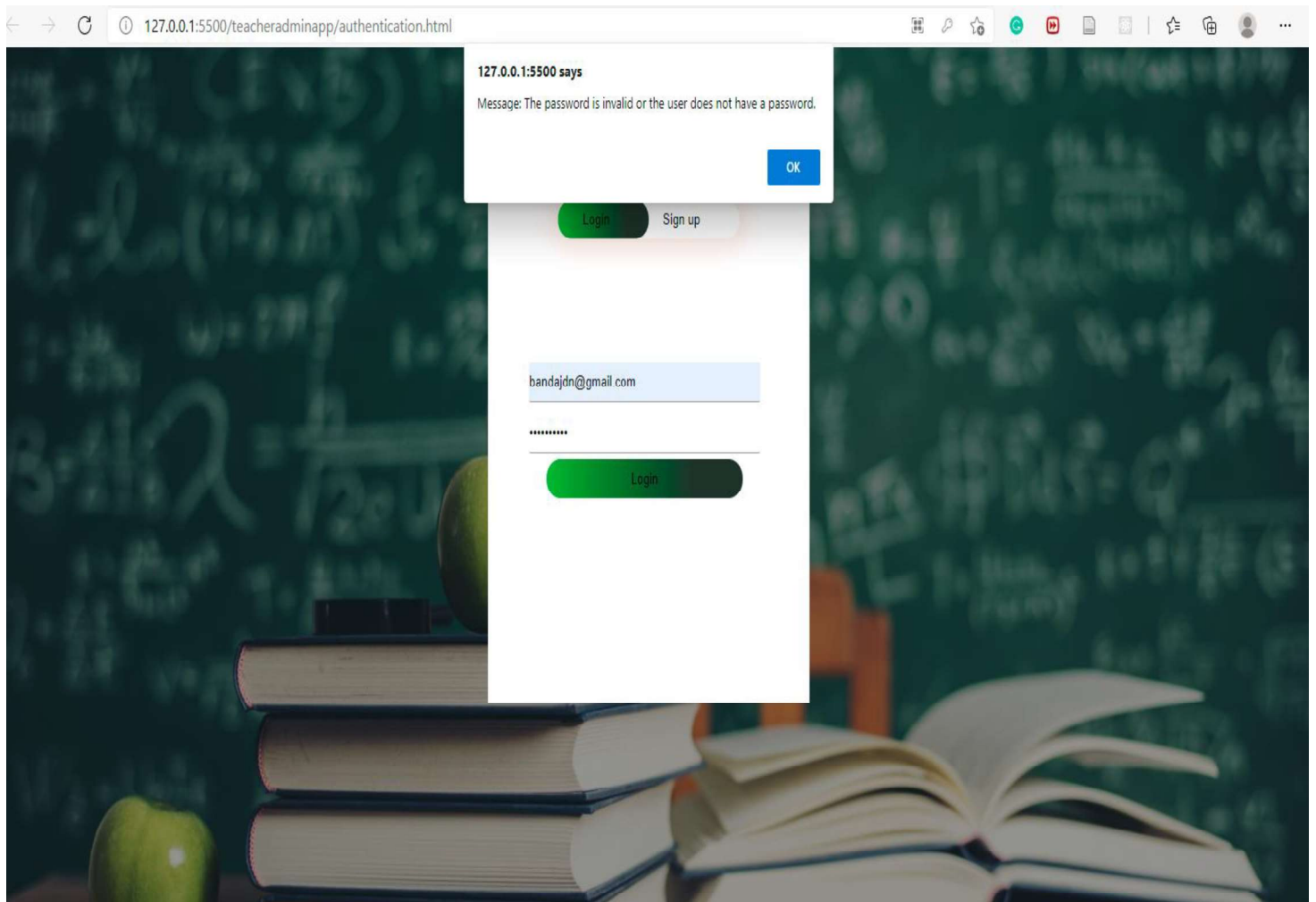


Figure 5.7: Form testing to ensure correct details are provided

Chapter 6: Conclusion, Limitations and Future Works

This project aimed to develop a software that would enable secondary school pupils to find skillful teachers who offer extra lessons within their location. It also aimed to provide a platform that would enable teachers who offer extra lessons to publicize themselves to pupils. With the Nearby Teacher Finder, a pupil can discover teachers in their area in only a few steps. The first step is to sign up or log in to the system, then search for or select a subject in which they want extra lessons, view teachers for that subject and their profiles, and interact with these teachers.

6.1 Limitations

The Nearby Teacher Finder uses the geolocation API to get the location of teachers who sign up on the system. It is important to note that the API does not work in some browsers. Because the system for the teachers is a web application, their location may not be obtained if they use a browser that does not support this API.

Secondly, the system does not allow its users to update their email addresses because the system administrator only uses emails to verify the details of a new user who has just signed up. This means that if a user wants to change the email address on the system, they will have to open another account.

6.2 Future work

This project presents possibilities for future work. Firstly, the limitations mentioned in section 6.1 have to be addressed in future works. In addition, a mobile version of the system for the teachers has to be developed as it would be more convenient for them to use a mobile application on their mobile phones rather than using a web application.

Secondly, more features can be added to the system. One of them is a schedule that would allow teachers to indicate their available times. This would help the pupils to choose a teacher with a plan that is more favorable to them. Another feature would be to allow pupils to rate teachers according to their experience.

Lastly, instead of requesting a teacher to provide details for their verification as a genuine teacher, a feature could be added to the system that would allow the teacher to upload their teacher certificates or license to be used for the teacher verification process.

6.3 Conclusion

This successful project presents how the exhausting process of searching for a teacher for extra lessons can be simplified using technology. If used as intended, it can contribute to enhancing the academic performance of secondary school pupils.

References

- [1] Mubanga Lumpa. 2018. Zambia's educational reforms since Independence. (October 2018). Retrieved April 27, 2021 from <http://www.daily-mail.co.zm/zambias-educational-reforms-since-independence/>
- [2] United Nations. 2018. The 17 Goals | Sustainable Development. (April 2018). Retrieved April 27, 2021 from <https://sdgs.un.org/goals>
- [3] UNICEF. 2018. Education in Zambia, we support the provision of quality education. (August 2018). Retrieved April 27, 2021 from <https://www.unicef.org/zambia/education>
- [4] Warit Taveekarn, Rukpatsorn Latthitham, Nuttawat Kittichareonjit and Vasaka Visoottiviseth. 2014. FindMyTutor: An Android Application for Matching Students and Private Tutors. *2014 Third ICT International Student Project Conference (ICT-ISPC2014)*, (March 2014), 5-8. DOI: 10.1109/ICT-ISPC.2014.6923205
- [5] Muhammad Saad, Farheen Iqbal and Muhammad Qasim Pasta. (2019). Smart Tuition Finder: An Educational App and SDGs. *2019 IEEE International Conference on Engineering, Technology and Education (TALE)*, (Dec. 2019), 6 pages. DOI: 10.1109/TALE48000.2019.9225881
- [6] Dolly Panchal, Mili Sanghavi, ShikhaDevi Pandey and Elizabeth George. (2016). Android Application for finding Tutors using Data Mining techniques. *International Journal of Scientific & Engineering Research* 7, 4 (April 2016), 1118-1123. Retrieved from <https://www.ijser.org/researchpaper/Android-Application-for-finding-Tutors-using-Data-Mining-techniques.pdf>
- [7] Ian Sommerville. 2011. *Software Engineering* (9th. ed.). Pearson, Boston, MA.

[8] IBM Cloud Education. 2020. Three-Tier Architecture. (October 2020). Retrieved April 27, 2021 from <https://www.ibm.com/cloud/learn/three-tier-architecture>

[9] Chris Veness. (2002). Calculate distance, bearing and more between Latitude/Longitude points. (2002). Retrieved April 1, 2021 from <https://www.movable-type.co.uk/scripts/latlong.html>

APPENDICES

Appendix A: system testing code snippet

```
// Imports the Flutter Driver API.
import 'package:flutter_driver/flutter_driver.dart';
import 'package:test/test.dart';

void main() {
  group('Nearby Teacher Finder', () {
    // First, define the Finders and use them to locate widgets from the
    // test suite. Note: the Strings provided to the `byValueKey` method must
    // be the same as the Strings we used for the Keys in step 1.
    final emailFinder = find.byValueKey('userEmail');
    final passwordFinder = find.byValueKey('password');
    final loginButtonFinder = find.byValueKey('Login');
    final subjectsFinder = find.byValueKey('subjectsKey');
    final teachersFinder = find.byValueKey('teachersKey');

    FlutterDriver driver;

    // Connect to the Flutter driver before running any tests.
    setUpAll(() async {
      driver = await FlutterDriver.connect();
    });

    // Close the connection to the driver after the tests have completed.
    tearDownAll(() async {
      if (driver != null) {
        driver.close();
      }
    });

    test('Login', () async {
      await driver.tap(emailFinder);
      await driver.enterText("njwaki@gmail.com");

      await driver.tap(passwordFinder);
      await driver.enterText("njwaki1234");
    });
  });
}
```

```

    await driver.tap(loginButtonFinder);
    await driver.waitFor(find.text("search for subject"));

    // await driver.tap(subjectsFinder);
    // await driver.waitFor(find.text("Available teachers..."));

    // await driver.tap(teachersFinder);
    // await driver.waitFor(find.text("About"));

  });
});
}

```

Code for system testing

```

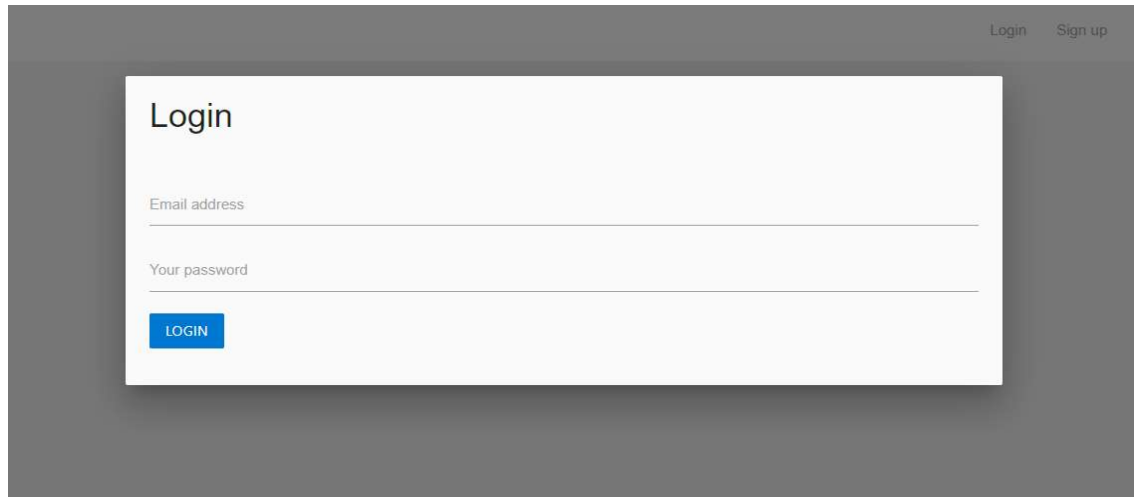
I/zygote (11706): Background concurrent copying GC freed 16614(1169KB) AllocSpace objects, 7(136KB) LOS objects, 50% free, 1794KB/3MB, paused 548us total 340.712ms
I/ProviderInstaller(11706): Installed default security provider GmsCore_OpenSSL
W/System (11706): Ignoring header X-Firebase-Locale because its value was null.
I/ProviderInstaller(11706): Installed default security provider GmsCore_OpenSSL
W/System (11706): Ignoring header X-Firebase-Locale because its value was null.
I/zygote (11706): Background concurrent copying GC freed 20868(1287KB) AllocSpace objects, 4(80KB) LOS objects, 49% free, 2MB/5MB, paused 9.919ms total 755.253ms
I/zygote (11706): Do partial code cache collection, code=59KB, data=49KB
I/zygote (11706): After code cache collection, code=59KB, data=49KB
I/zygote (11706): Increasing code cache capacity to 256KB
D/FirebaseAuth(11706): Notifying id token listeners about user ( rnbwHxSqFQeD1koDrT3ai7SQIDm2 ).
D/FirebaseAuth(11706): Notifying auth state listeners about user ( rnbwHxSqFQeD1koDrT3ai7SQIDm2 ).
W/ServiceFlutterDriver: waitFor message is taking a long time to complete...
00:15 +1: Nearby Teacher Finder (tearDownAll)
00:15 +1: All tests passed!

```

Results for system testing

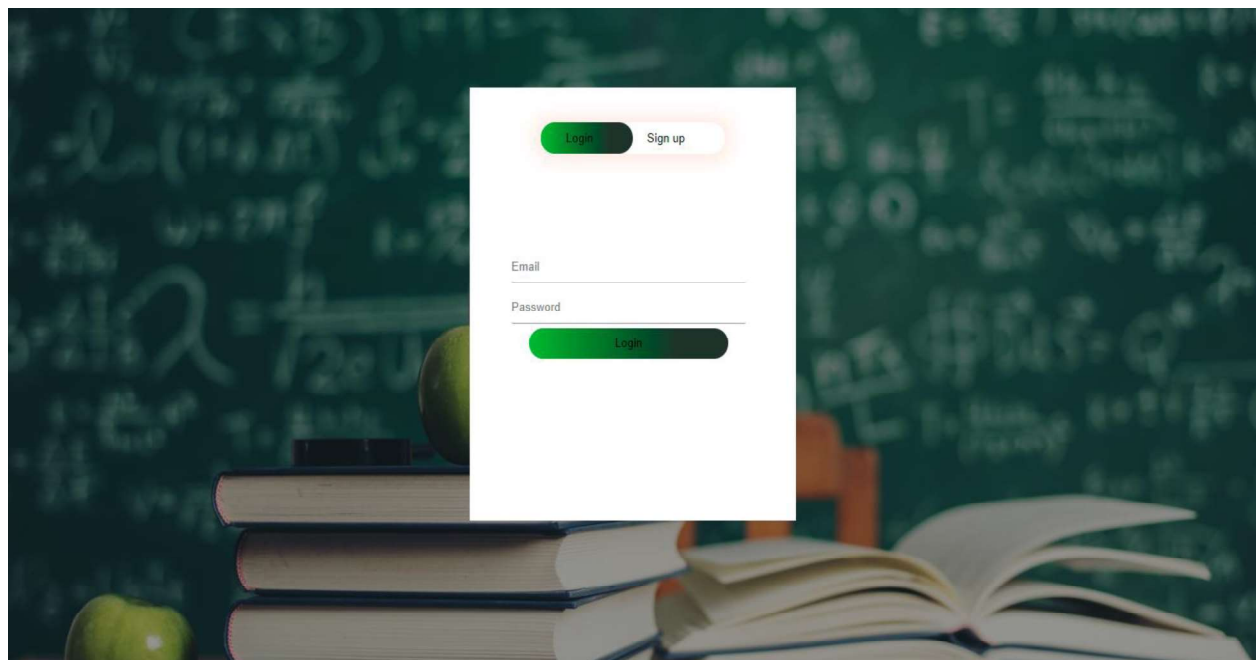
Appendix B: Login UI

Login UI before and after user testing



The image shows a login form on a dark gray background. In the top right corner, there are links for "Login" and "Sign up". The form itself is a white box with the title "Login" at the top. Below the title, there are two input fields: "Email address" and "Your password". At the bottom of the form is a blue button labeled "LOGIN".

Login before user testing

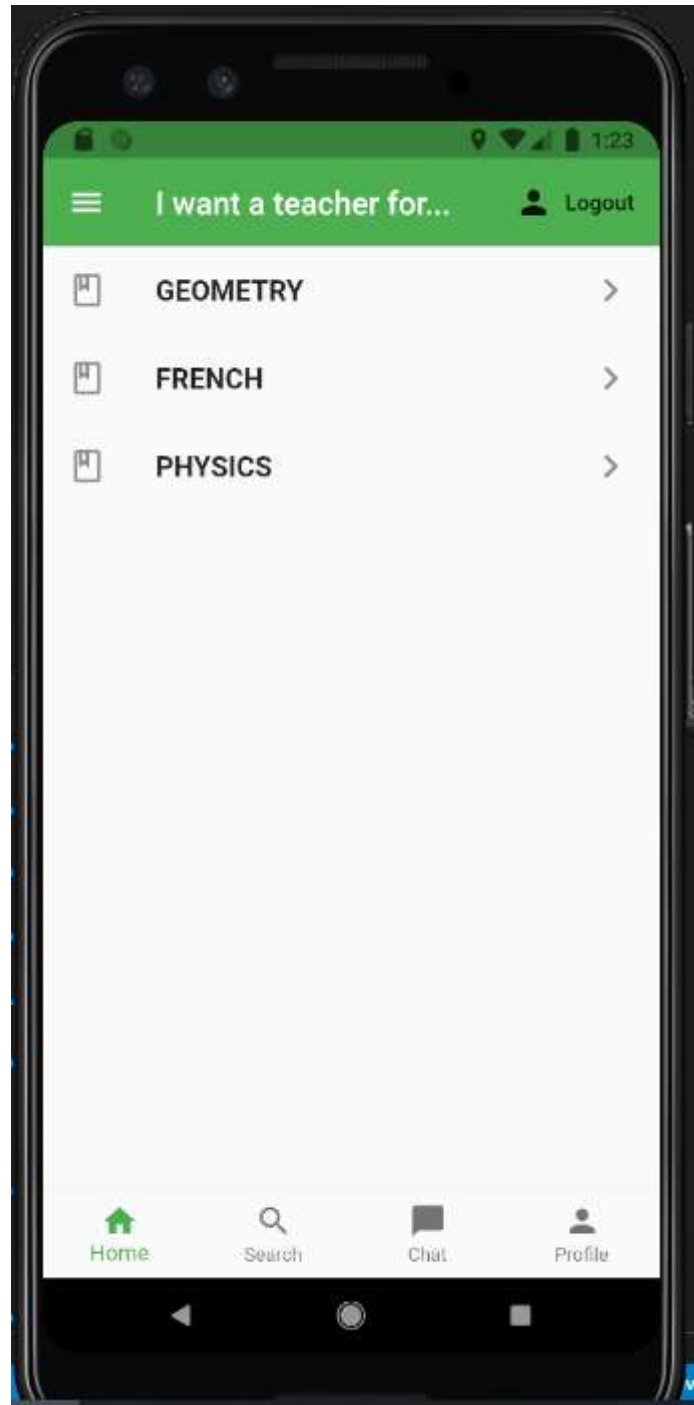


The image shows a login form overlaid on a background of books and a chalkboard with mathematical formulas. The form is a white box with a rounded top. At the top, there are two buttons: "Login" (green) and "Sign up" (white). Below these buttons are two input fields: "Email" and "Password". At the bottom of the form is a green button labeled "Login".

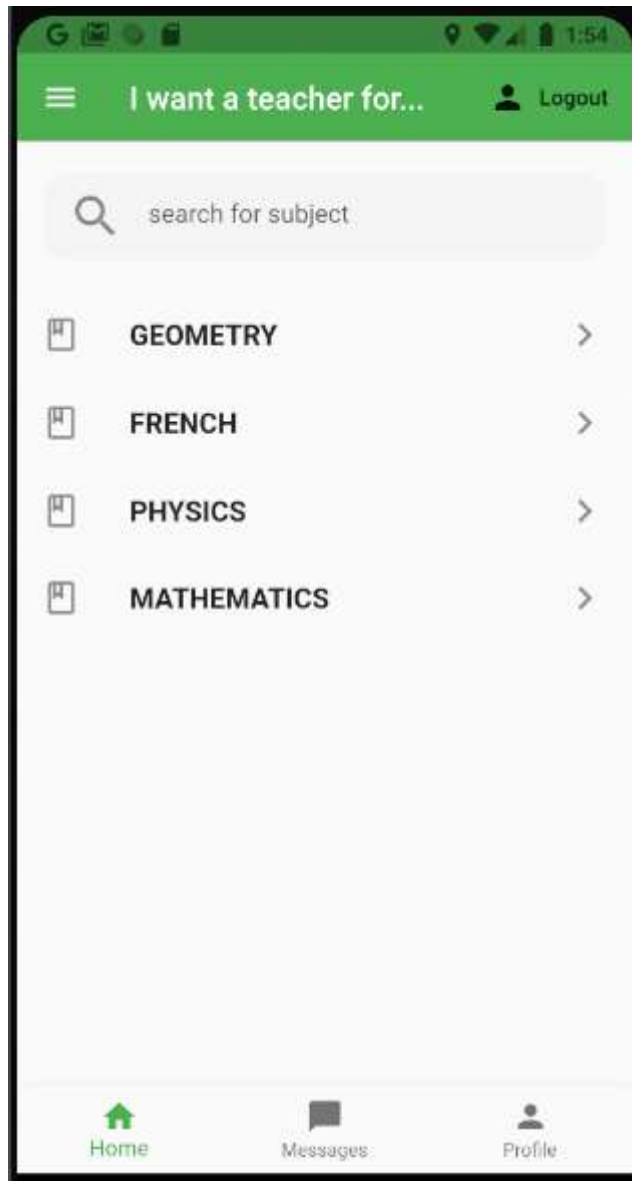
Login UI after user testing

Appendix C: Subjects screen

Subjects screen before and after user testing.

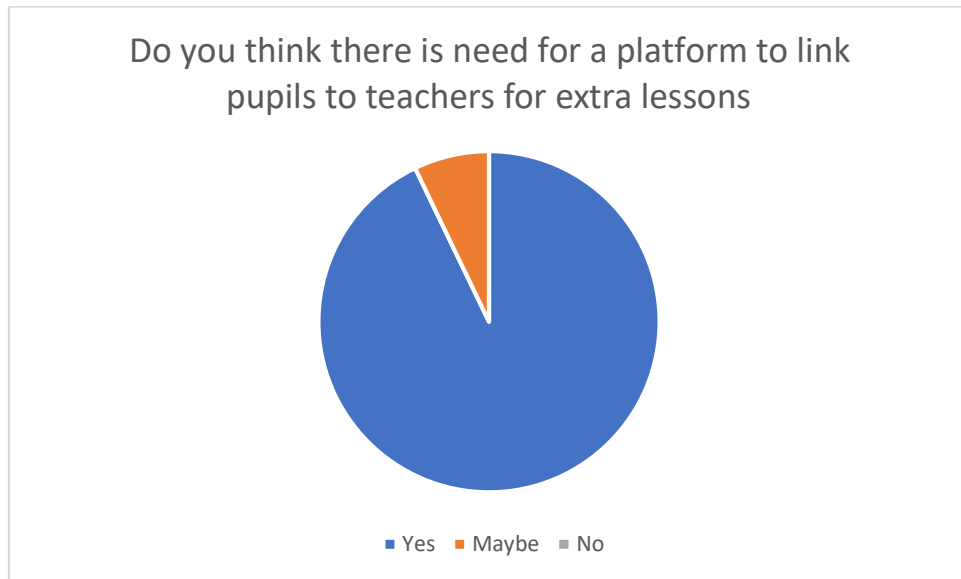


Subjects screen before user testing

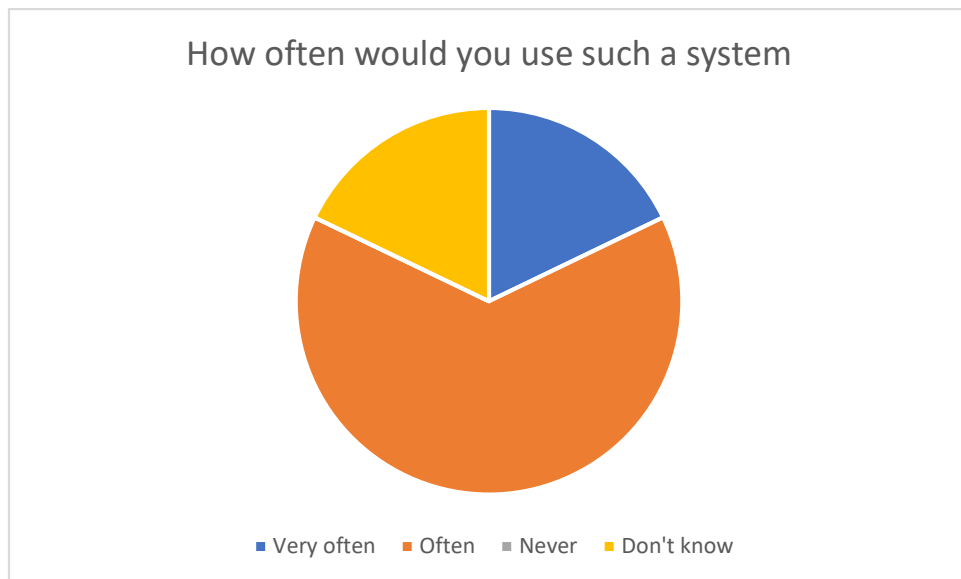


Subjects screen after user testing

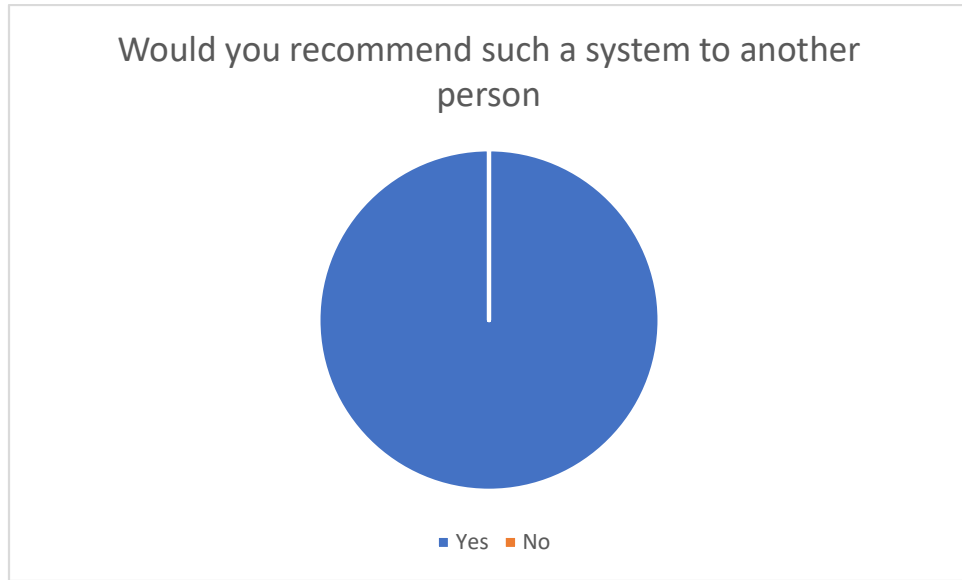
Appendix D: some questions during requirement analysis



A pie chart representing users' thoughts on the needs for the platform



A pie chart representing how frequent users would use the system



A pie chart representing whether or not users can recommend the system to others