Student Support Android System

by

Rebecca Fan

This report is submitted in partial fulfilment of the requirements of the Degree in Electronic and Communications Engineering (DT008) of the Technological University Dublin.

April 20th, 2020

Supervisor: Lejla Rovcanin

School of Electrical and Electronic Engineering

DECLARATION

I, the undersigned, declare that this report is entirely my own written work, except where otherwise accredited, and that it has not been submitted for a degree or other award to any other university or institution.

Signed: Rebecca Fan

Date: 20 Apr 2020

Table of Contents

Table of Figures:

1.0 Introduction

In this era of technology, it is estimated that the number of smartphone users in the world is approximate 3.5 billion in 2020[1]. All smartphones are filled with mobile applications. This report will show the progress of developing an android app. This android application was made with students in mind as students are known have a busy and hectic lifestyle and college may be very overwhelming. TUD students need to have access to their timetable and to Brightspace for the assignment and module details. To make the life of a student a little easier the Student Support System app will have many of these features in one app. This app was made using a software called Android studio using the programming language Java. This report will outline the process from research, design and implementation to get to the final product.

As a TUD student, building an android app for students from a student's point of view, creates an ideal app for TUD students by knowing their needs and wants. Learning how to develop android app, many skills were learnt and by having skills in application development will broaden even more career opportunities as an Electronics and Communications Engineer.

2.0 Projective Objective

The project objectives are to create an android app that is student centred to organise their time and to include the features a student needs in just one app.

The following were the initial objectives of the app:

- Attendance monitoring
  - Attendance monitoring will by tracked by either using Wi-Fi or GPS.

- Notifications
  - Notifications with information of the next upcoming class will be notified to the students 15 minutes before their next class.

- Timetable synchronisation
  - The timetable will be synchronised to the TUD timetable.

- Connection to VLE Assignments
  - Brightspace will be connected to the app so the student can access course content and view assignments.

However, these features were quite unachievable for the short time span that was given for this project. Many of these features needed developer access or access to the TUD server, which was not granted for this app, therefore compromises were made.

The following features are included in the final app:

- Login Authentication
  - Login to the app using the students TUD Microsoft email and password.

- Contact page
  - It will contain contact information of the developers of the app.

- Sleep Tracker

  - A sleep tracker to time the hours the user would have slept.

- A navigation bar
  - To access the home page, contact, sleep tracker pages and a sign out button.

- Sign-out Button
  - A button on the navigation bar that will sign out the user.

3.0 Application Development

3.1 Background Research and Tools selection

Mobile applications are software application developed specifically for use on small, wireless computing devices, such as smartphones and tablets, rather than desktop or laptop computers [2]. Mobile applications are made to provide users with similar services to those accessible on PCs (Personal Computers) but on a more portable smaller device [3]. These mobile apps can be purchased from the Google Play Store for android users or the App Store for IOS Apple users.

3.1.1 WordPress

WordPress is a popular open source software used to create websites or blogs. It contains many templates that are customisable and simple to use. For this app, a blog was made using WordPress to document progress made to the app. Weekly logs are made to document the personal progress made to the app.

3.1.2 Udacity Online Courses

The most essential skill to have when developing an android app is knowledge of a programming language. The most preferred languages to create Android apps are Java and Kotlin, the developer needs to be familiar with the core concept of at least on of these programming languages. To learn the basis of developing the UI (User Interface) or the front-end of the app using the language Java [4]. Udacity is a website that provided free online course that was useful to get started on Application Development. The course named: ‘Android Basics: User Interface’ [5] it showed the basis of Java and Android Studio which is the Integrated Development Environment (IDE) used for this project. To design the user interface in Android, XML is used. XML stands for eXtensible Markup Language, a markup language system designed for the processing formatting and presenting of text [6].

3.1.3 Android Studio

An Integrated Development Environment (IDE), is a software for building applications that combines common developer tools into a single graphical user interface (GUI) by consisting editing source code, building executables, and debugging [7].

Android Studio is a popular and the recommended IDE for android app development. It offers tools custom-tailored for Android developers. To integrate XML layouts with the business logic in Android Studio it uses Gradle, which is a build toolkit, to automate and manage build processes [4]. Android Studio includes project and code templates such as a navigation drawer which was used in this project. It also has an emulator which installs and starts your apps fast and allows you to prototype and test your app on various Android devices [8].

3.1.4 Git/GitHub

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. This allows for changes in files while previous working copies are safe and secure.

Git is a free and open-source software used for version control in this project. GitHub is the online platform that stores files and to have an online backup of those files [9]. It provides as a progress log and version control for this project as it shows when a "commit" was made which means changes you have made to files that will be saved as one version. Version control is essential as changes in code are made these changes can be tedious to find, using Git, it can look back and indicate and compare where changes have been made.

3.2 Application Features Research

3.2.1 Brightspace

Brightspace is the virtual learning environment for TUD. This is where students have access to their course work provided by lecturers, students also submit their assignments on Brightspace. Brightspace has a developer platform that includes APIs, libraries and tools that developers use to integrate their own projects. To include Brightspace into the project, the developer must apply as a developer to D2L (Desire2Learn) the company that made Brightspace and acquire an App ID-key pair [10]. To register and have this project approved to have access to their API's would've taken too long for this short timespan this project had.

If this project had access to the Brightspace API the end users would have had access to their courses, the course content, enrolments and grades. Brightspace API's were plentiful, there was many options that could have been incorporated such as notifications for submissions and announcements made by lecturers. The feature of Brightspace was unfortunately not incorporated for the Student Support System but Brightspace have a mobile app of their own called Pulse that TUD students are using for their coursework.

3.2.2 Timetable and Notifications

TUD has a designated timetabling app [11], it was used as inspiration for the timetabling feature for the Student Support System app as it had class information also notifications which would prompt the user 15 minutes before their next class with information such as room number for their next class. After some research the TUD timetable app was not made by TUD but an external company, therefore getting the framework or the code was not possible. TUD has a website for timetabling also but accessing all the course timetabling information was denied as access to TUD servers was not granted.

3.2.3 Attendance

Grades usually correlates to attendance, the less a student attends class the more effect it will have on their grades. To add an attendance feature to the app would have allowed the students to keep track of their attendance and to allow students to sign into class easier for the lecturers as the do not have to waste time on rollcalls.

The following were the options that to include attendance monitoring to the project:

- GPS
  - For this option, the idea would be for the students to monitor their own attendance by using GPS, the GPS feature would locate if the student is in the correct classroom and

time. GPS feature on the smartphone's needed to be accessed but a flaw that was made apparent was the precision of the location. GPS would not to accurate enough to know what floor or room the user is in.

- QR code
  - All smartphones have cameras, to utilise this for attendance monitoring students could use their camera to scan a QR code to confirm their attendance. At the start of each class the lecturer may provide a QR code generated specifically for that class and allow the students to scan this code to sign into that class.

- NFC (Near Field Communication)
  - NFC is a short-range wireless connectivity standard that allows for an to transfer information between two devices instantly [12]. It is significantly faster than using QR codes. RFID (Radio Frequency Identification) scanners, are made up of integrated circuits containing a tiny antenna for transferring information [13]. The information is made up of radio waves which is converted into a more usable form of data. By pairing these scanners with the NFC feature on a smartphone. Students would simply be able to tap their phone into a scanner to sign themselves into class.

The above options required lengthy research, with no prior knowledge on app development and major features that use GPS, or the sensors in the phone, the attendance monitoring feature was not achievable as this was not the sole feature of the Student Support System app. With the time limit of the project it was irresponsible to focus on this major feature that would consume all the time for this project.

The following features are included in the final version of the app. These features were more achievable with the time span and the with no prior knowledge of app development or with the programming language Java.

3.2.4 Authentication

Authentication was the major feature for Student Support System app. While researching authentication, the first option that was used was Node.js, an open source server environment used to create the backend for authentication. Node.js properties were:

- Generate dynamic page content,
- Create, open, read, write, delete, and close files on the server
- Add, delete, modify data in your database [14].

This authentication method was using JSON Web Token (JWT). JWT is a self-contained way for securely transmitting information between parties as a JSON object. To use this for authentication, once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token [15].

MongoDB was used as the database, it is a document database, which means it stores data in JSON-like documents [16]. Using MongoDB, it can show and verify if authentication is working and show the list of users registered. Mongoose is a MongoDB object modelling tool designed for Node.js [17], it's used to create models and to connect to the express server. Express is a web application framework in Node.js to create a web server. A node.bcryp.js is used to hash passwords, leaving passwords are not unhashed would exposes credentials of the user [18].

This route for authentication was difficult as there were many steps such as building a sever and creating a database and it was all written in JavaScript. Issues occurred as learning 2 different programming languages was redundant as the main program used for this project was Android Studio and it uses Java.

The more efficient option for authentication which was to use Firebase. Firebase is a comprehensive mobile development platform that has many tools and functionalities like authentication, analytics, and databases [19]. Firebase Authentication made building authentication systems a lot easier than with Node.js as Firebase take care of the difficulties like connecting to a server and database. To allow sign in using the students own TUD email address, which is are linked to Microsoft, Firebase SDK had a tutorial to add Firebase to an Android Studio project and how add Microsoft authentication into the project to handle the entire sign-in flow [20].

3.2.5 Sleep Tracker

On average, most college students get 6 - 6.9 hours of sleep per night [21], students are notoriously known to be sleep-deprived due to an overload of activities. Sleep is vital for normal motor and cognitive function. To allow students to track their sleep hours a sleep tracker was added to this app. In Android there is a Chronometer class [22] that implements a simple timer. Using this Chronometer, it allows the students to press a button to start the timer of when they fall asleep and to stop the timer for when they wake up.

3.2.6 Navigation Bar and Contact page

A navigation bar was added to allow easy access to the different features/pages of the app. In Android Studio there is already a prebuilt template of a default navigation bar. Using one of the options of the navigation bar, a contact page was made to include the contact information of the developers of the Student Support System App.

3.3 Application Design

A prototype the first example of design intent for a project. Prototyping allows designers to present their designs and allow to use them to shape the final look of the project. Low-fidelity (lo-fi) prototyping is a quick and easy way to translate high-level design concepts into feasible designs [23]. Lo-fi protypes are made to test their functionality, in this project a few sketches were made. The follow are the sketches made very early on of the project for the Login page, a page to add items, the Home page and the sleep tracker:

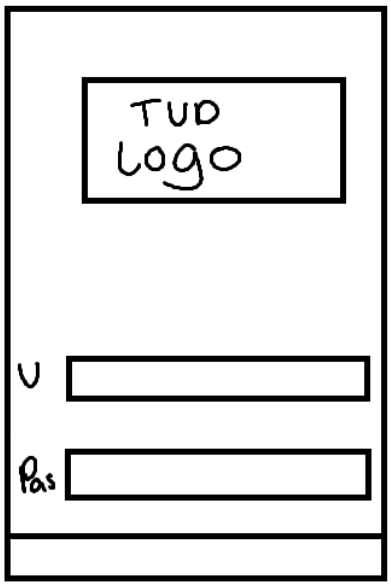


Figure 1 – Login Page

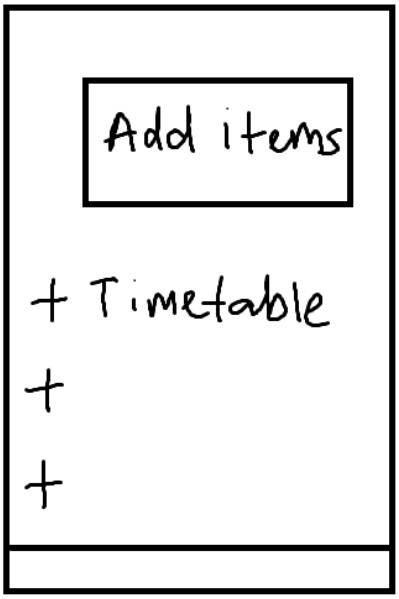


Figure 2 – Add Items

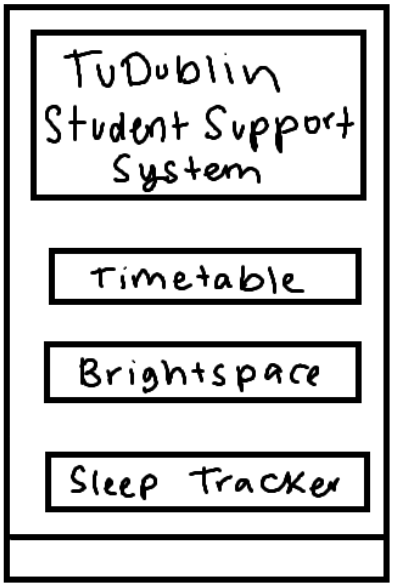


Figure 3 – Homepage

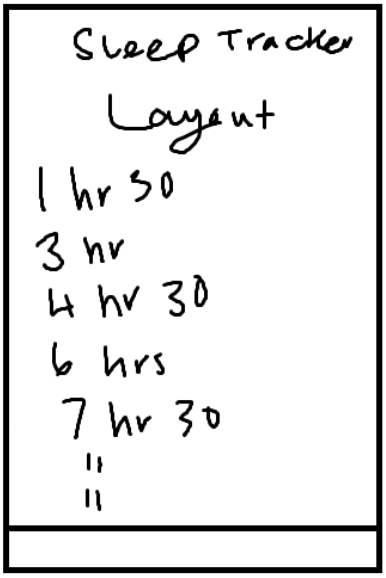


Figure 4 – Sleep Tracker Layout

More low fidelity prototypes were made for the final version of the Student Support System app this includes a login page, a homepage and the sleep tracker. They determine the final layout and design with this in mind, writing code for the pages were made easier.

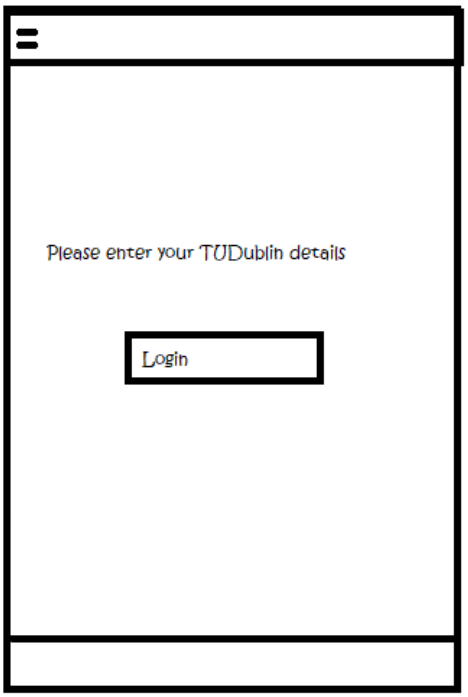


Figure 5 – Login Page Final

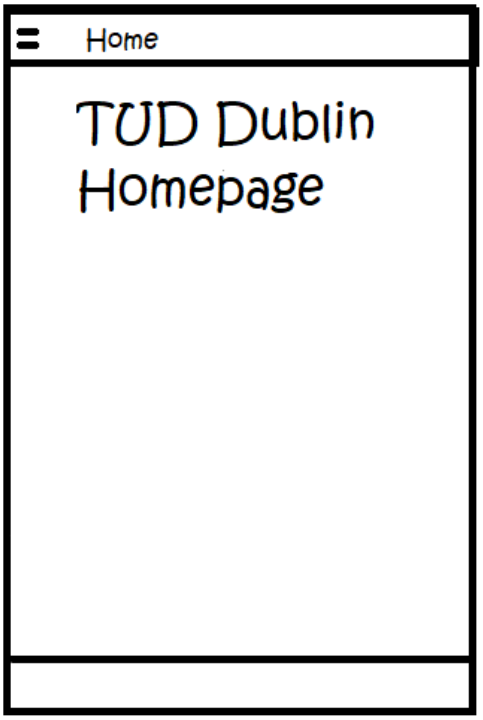


Figure 6 – TUD Dublin Homepage Final

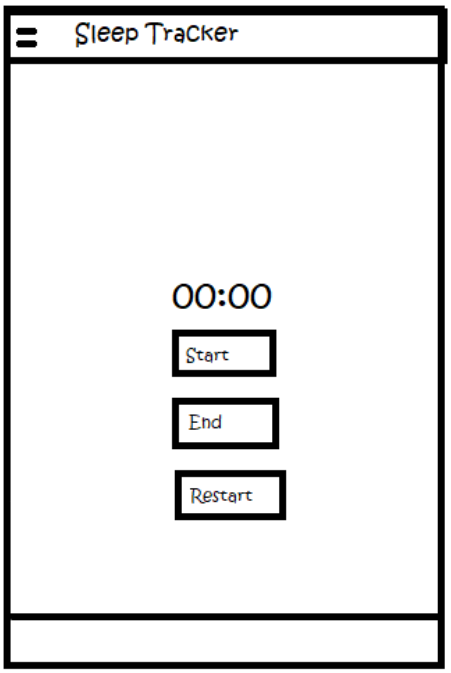


Figure 7 – Sleep Tracker Page

3.4 Application Implementation Testing

The next section shows implementation testing which is the process of putting the researched plan into effect and to test each section that is implemented.

3.4.1 Login Implementation

This was the first step made when developing Student Support System app. As mentioned above this was firstly implemented using Node.js for the backend of authentication. It was written in Visual Studio Code which is a source-code editor developed by Microsoft. Node.js must be installed to the laptop. Using a tutorial on YouTube called "Building a Node.js API Authentication with JTW Tutorial" [24], the steps in the video was followed. It was not an informative video as it did not explain the uses and reasons for many of it's the steps. It was quite difficult to keep up with the narrator because of that. It was also written in JavaScript.

The better and easier option for this project was to use Firebase as it was written in Java, the same language used for the front end in Android Studio as there was a step by step detailed guide to authenticate using Microsoft on Android Studio.

3.4.1.1 Add Firebase to app and Android Studio

Register the Student Support System App with Firebase and add Firebase Android configuration file to your app by downloading google-services.json file and add it to the app directory of the Student Support System App in Android Studio. To enable Firebase products the google-services plugin must be added into the Gradle files of the app and add Firebase SDKs by adding the dependences into the Gradle. Sync app the ensure that the dependencies are compatible. These steps are outlines by Firebase's documentation "Add Firebase to your Android project" [25]

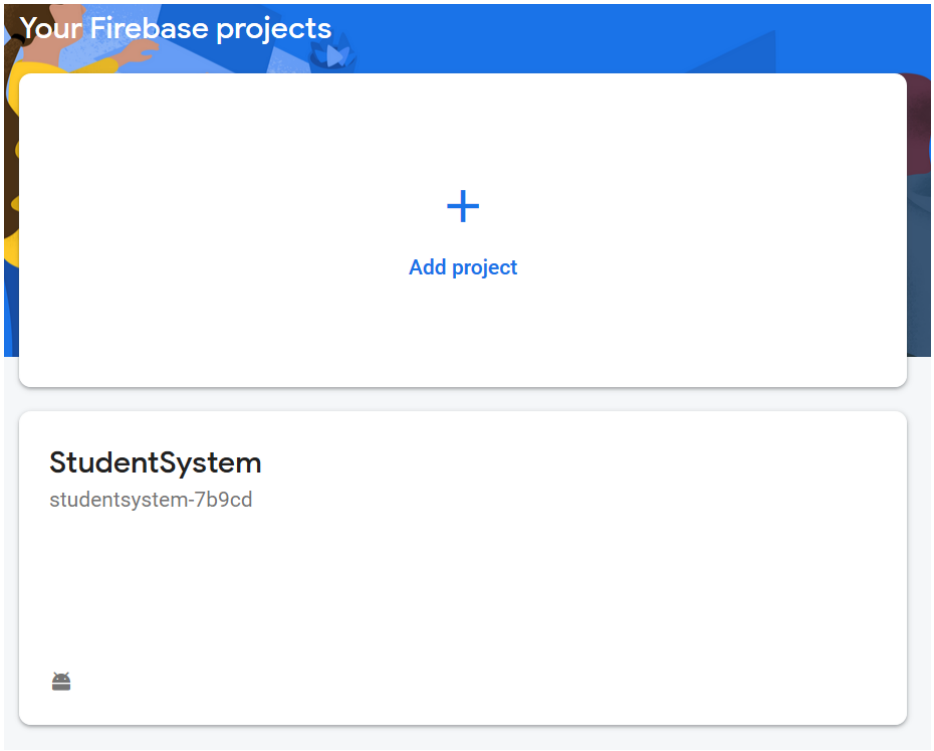


Figure 8 - Register Student Support System App to Firebase

3.4.1.2 Enable Microsoft as a sign-in provider

In the Firebase console in the Auth section enable the Microsoft provider as a Sign in method. Add the Client ID and Client Secret from that provider's developer console to the provider configuration (google-services.json file).

```
"oauth_client": [
  {
    "client_id": "421976284058-k2hctk8frvc58grlnvnjule1o7vvo19u.apps.googleusercontent.com",
    "client_type": 1,
    "android_info": {
      "package_name": "com.example.studentsystem",
      "certificate_hash": "50e99d30cfe00cc276447ee25a57dd856739d9ae"
    }
```

Figure 9 – Enable Microsoft provider

3.4.1.3 Sign-in flow with Firebase SDK

The steps outlined by Firebase in “Authenticate Using Microsoft on Android” [26] were followed as it provides the extensive code to allow authorize access to Azure Active Directory (Azure AD) which is Microsoft's cloud-based identity and access management service [27]. Azure AD uses OAuth 2.0 as their protocol for authorisation. OAuth is a standard that apps can use to provide client applications with “secure delegated access” [28]

3.4.1.4 Login UI

The UI for the Login page was made with help from an article on Medium.com [29]. It has a step to step guide on how to make the front end of a Login and Registration page of the app it also includes font. Some components used for the final Login Page:

1. Fragment
   - A Fragment is a piece of an application's user interface or behaviour that can be placed in an Activity [30].
2. Button
   - A button consists of text or an icon or both with an action assigned to it and that action will occur when the button is touched [31].
3. TextView
   - A user interface element that is used to set and display text [32].

This page contains two TextViews and a Button. A title and a message to alert the user to Log in with their TUD Microsoft account. The Login button brings the user to a redirected page to Microsoft to login.

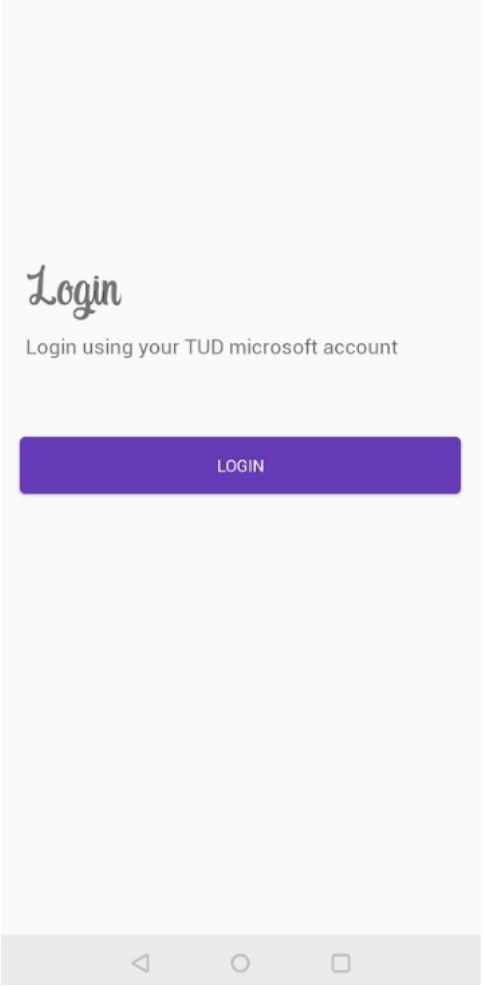


Figure 10 – Login Page

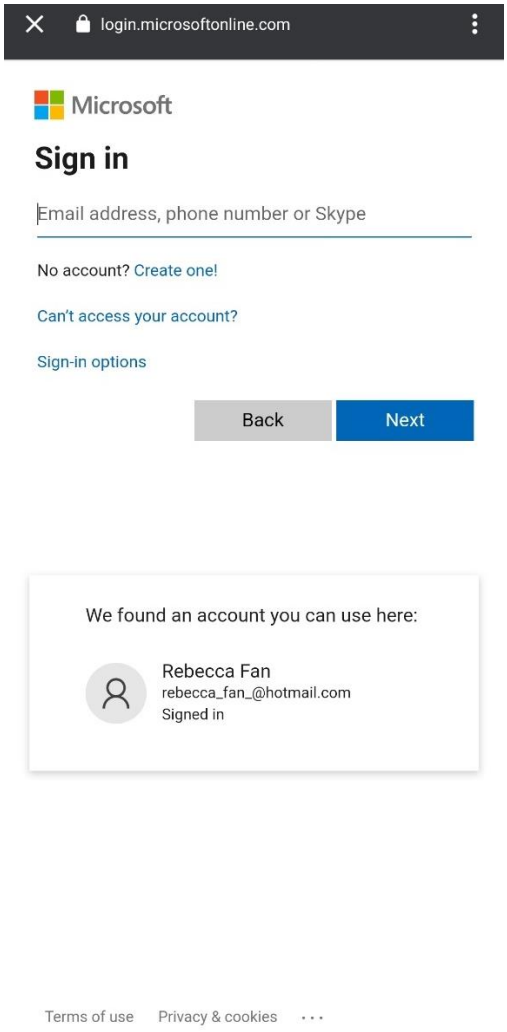


Figure 11 - Microsoft Login Page

The redirected page will open to a Microsoft Sign in page. If a user has been signed in before their account will be prompted as an account that is useable without having to go through the sign in process every time the app is opened.

This app was made for TUD students, so it was ideal to log into the app using their TUD Microsoft account, after testing this feature an issue had become apparent when a TUD account tries to log in. Since their account is run by TUD, it needed admin approval to allow their account to be logged into the Student Support System App.

Therefore, for this app only personal Microsoft accounts can be used to sign in.

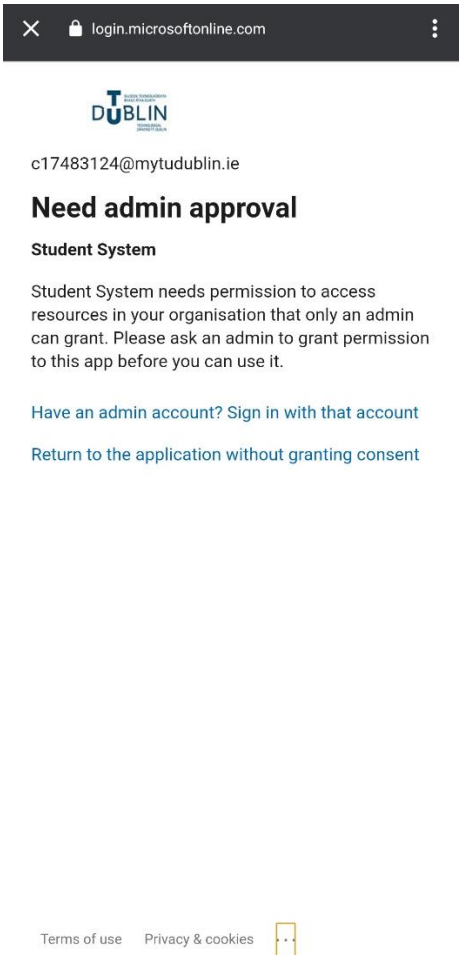


Figure 12 - Admin approval to use TUD account

3.4.2 Home Page

A toast which is a small pop up message will pop up to indicate if login was successful. If login is successful, the page will be redirected to the home page of the app. This page has an image of the TUD logo and a message that explains the purpose of the app.

The toast message in implemented by the following:

```
Toast.makeText(getActivity(), "Hi " + authResult.getUser().getDisplayName() + "
\nSuccessfully logged in with microsoft !!", Toast.LENGTH_LONG).show();
```

This gets the details from authentication go get the users name to display in the toast message.

To add the TUD logo and text to explain the app, the following code in implemented in the home fragment:

```
<ImageView
    android:id="@+id/TUDLogo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:scaleType="centerInside"
    app:srcCompat="@drawable/tud" />

<TextView
    android:id="@+id/home_desc"
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:text="This app is made by 2 TUD students with the goal of helping student
life at college easier."
    />

<TextView
    android:id="@+id/home_desc2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:text="To have features like timetable, BrightSpace, attendance monitoring
and sleep tracking in one app."
    />
```

With that code implemented, the UI for the home page will look like the following:

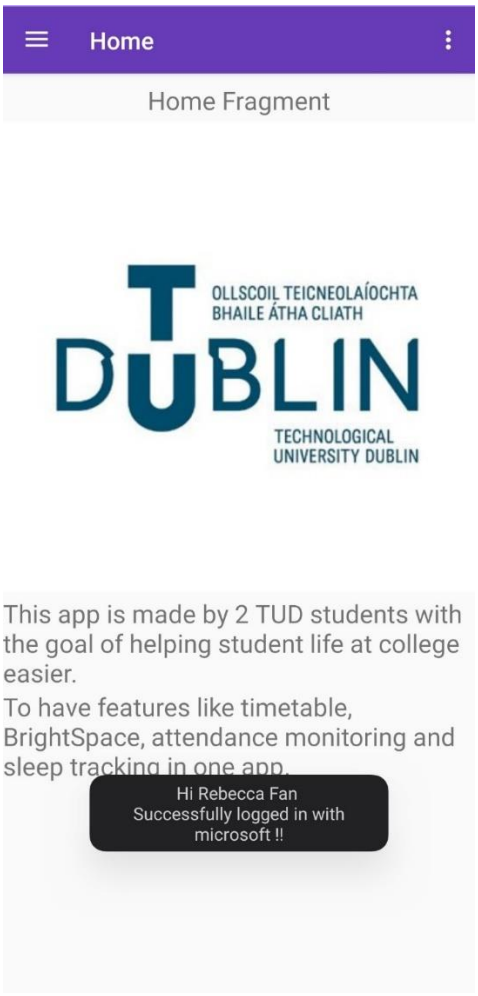


Figure 13 - Home Page

3.4.3 Navigation Bar

Android Studio has an option when creating a new project, there are many templates to choose from, for the Student Support System the Navigation Drawer Activity template was chosen to allow the user to interchange to the features easily.

With this template chosen Android Studio will create a default template with default fragment pages and UI that was changed to the features for this app. At the top of the navigation bar it has an image, name and email section and has five options Home, Gallery, Slideshow, Tools, Share and Send. These lead to empty fragment pages.

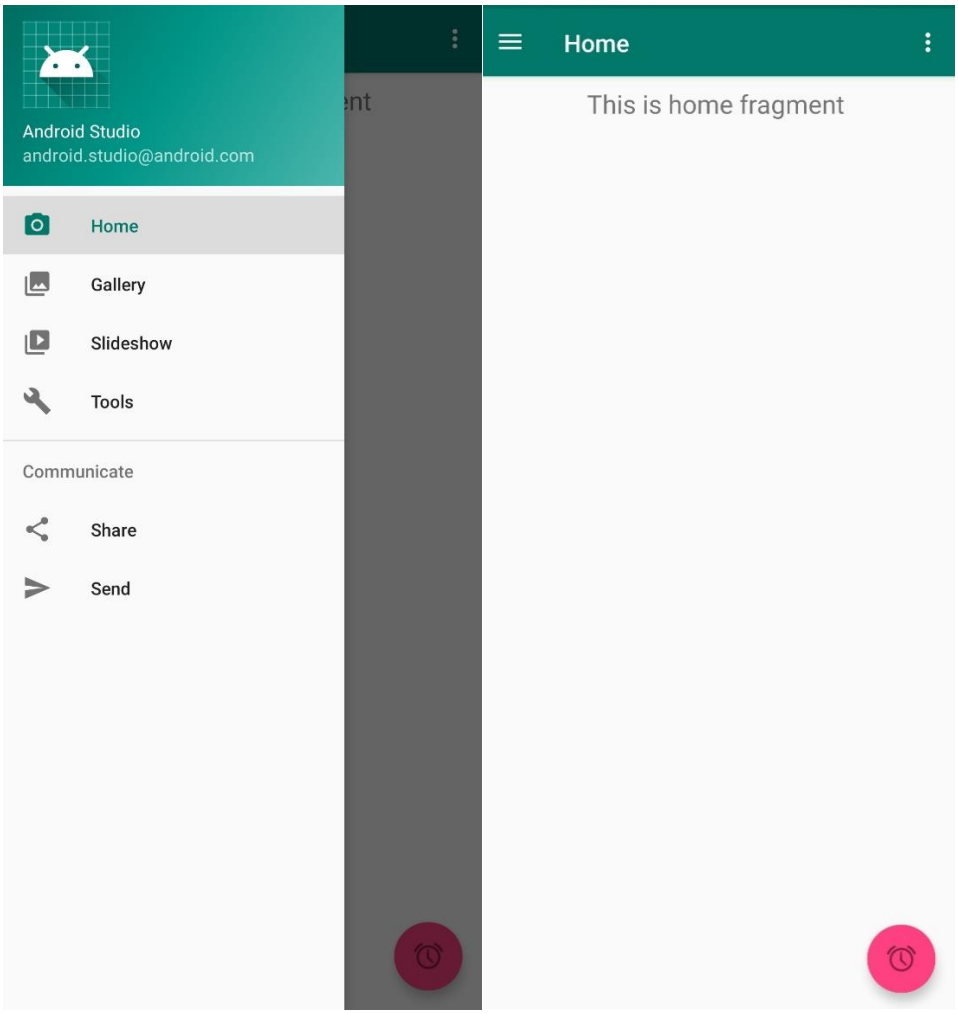


Figure 14 - Default Navigation Bar and Fragment

When the user successfully signs in, their name and email address will be displayed at the top of the navigation bar using the Firebase to get the signed in users name and email. This is implemented by the following code:

In HomeActivity.java:

```
FirebaseUser = FirebaseAuth.getInstance().getCurrentUser();
TextView username = headerView.findViewById(R.id.username);
username.setText(firebaseUser.getDisplayName());
TextView email = headerView.findViewById(R.id.email);
email.setText(firebaseUser.getEmail());
```

In nav_header_home.xml:

```
<TextView
    android:id="@+id/username"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingTop="@dimen/nav_header_vertical_spacing"
```

```
    android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

<TextView
    android:id="@+id/email"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="@style/TextAppearance.AppCompat.Body2" />
```

Unnecessary pages were removed and renamed to include, Home, Sleep Tracker, Contact and Sign Out. The icons were changed to be relevant to the options by downloading icon images on the web and adding it into the drawable folder on android studio where it could be accessed.

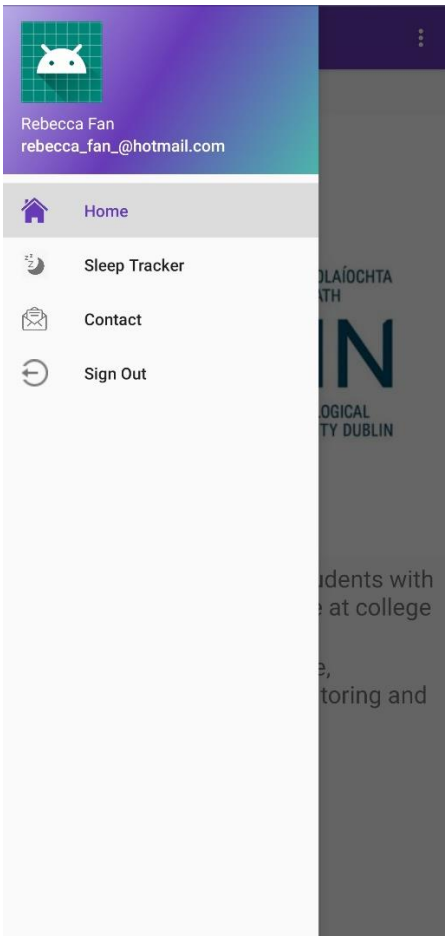


Figure 15 - Final Navigation Bar

To implement that the following code was used:

In activity_home_drawer.xml

```
<item
    android:id="@+id/nav_home"
    android:icon="@drawable/home"
    android:title="@string/menu_home" />
<item
    android:id="@+id/nav_sleeptracker"
    android:icon="@drawable/sleep"
    android:title="Sleep Tracker" />
<item
```

```
    android:id="@+id/nav_contact"
    android:icon="@drawable/email_icon"
    android:title="Contact" />
<item
    android:id="@+id/nav_signout"
    android:icon="@drawable/signout"
    android:title="Sign Out" />
```

3.4.4 Sleep Tracker

This feature was implemented by Vivien Enriquez. Using the Chronometer Class which makes a simple timer. The sleep tracker page contains and three buttons and a text display. The first button starts the timer for when the user sleep, the second button is to end the timer for when the user wakes up and finally third button that resets the timer. The text displays how long the user has been sleeping.

The UI is implemented using the following code:

In fragment_sleeptracker.xml:

```
<Chronometer
    android:id="@+id/chronometer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30sp"/>
<Button
    android:id="@+id/chronoStart"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Start Sleep" />

<Button
    android:id="@+id/chronoEnd"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="End Sleep"
    android:onClick="endChronometer"/>

<Button
    android:id="@+id/chronoReset"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Reset Sleep"
    android:onClick="resetChronometer"/>
```

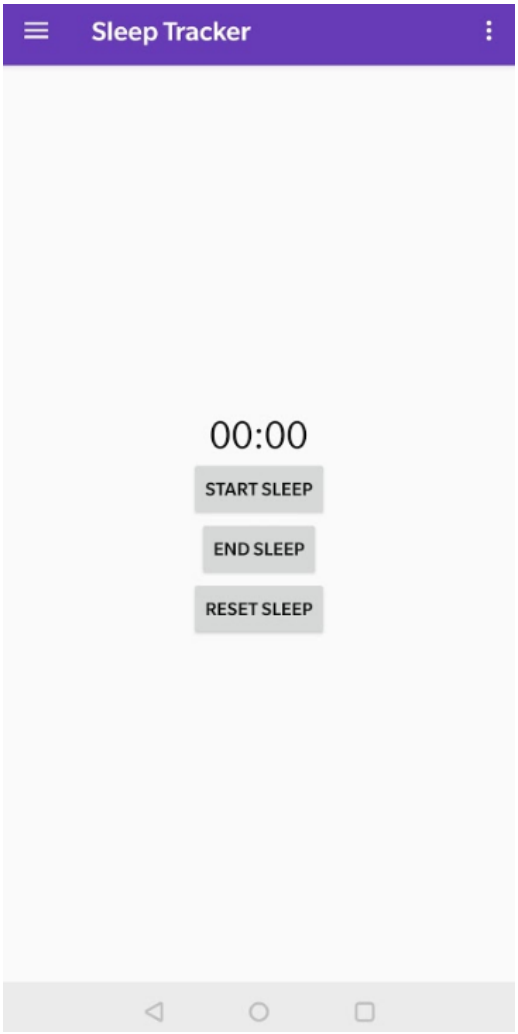


Figure 16 - Sleep Tracker Page

The following code is the action linked to the start, stop and restart buttons:

In SleepTrackerFragment.java:

```
public void startChronometer(View v) {
    if (!running) {
        chronometer.setBase(SystemClock.elapsedRealtime() - pauseOffset);
        chronometer.start();
        running = true;
    }
}
public void endChronometer(View v) {
    if (running) {
        chronometer.stop();
        pauseOffset = SystemClock. elapsedRealtime() - chronometer.getBase();
        running = false;
    }
}
public void resetChronometer(View v) {
    chronometer.setBase(SystemClock.elapsedRealtime());
    pauseOffset = 0;

}
```

3.4.5 Contact Page

A contact page was added to provide information on the developers of the app. It includes their name and a contactable email address. The following is the UI for the contact page:

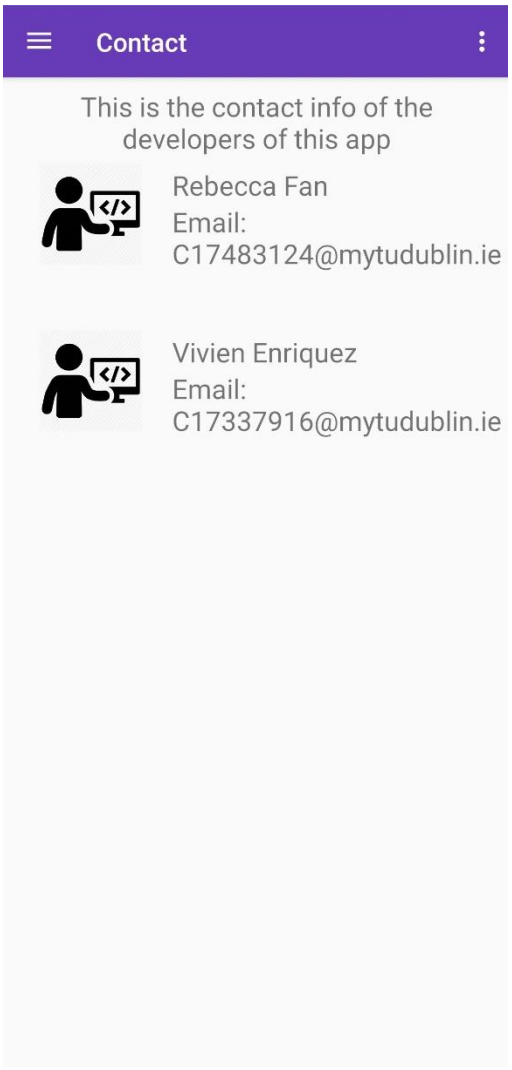


Figure 17 - Contact Page

To implement the icon and text for the developers information, the following code is used:

```
<LinearLayout
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:gravity="center">

        <ImageView
            android:id="@+id/dev_icon"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:adjustViewBounds="true"
            android:scaleType="fitCenter"
            android:layout_marginLeft="5dp"
            android:src="@drawable/developer_icon">
```

```
            </ImageView>

        </LinearLayout>

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="2"
            android:gravity="left"
            android:orientation="vertical">
    <TextView
        android:id="@+id/name_rf"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_marginTop="8dp"
        android:text="Rebecca Fan" />

    <TextView
        android:id="@+id/email_rf"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="Email: C17483124@mytudublin.ie"
        />

    </LinearLayout>
```

3.4.6 Sign Out Button

To allow the user to sign out of the Student Support App the last option on the navigation bar will be used at a sign out button. When it is pressed it will sign out the user and bring them back to the Login page.

In fragment_signout.xml:

```
<Button
    android:id="@+id/btn_SignOut"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sign Out" />
```

To assign the sign out action to the button the following code is implemented:

In SignOut_fragment.java:

```
Button btn_SignOut = root.findViewById(R.id.btn_SignOut);
btn_SignOut.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FirebaseAuth.getInstance().signOut();
        Intent myIntent = new Intent(getActivity(),
                MainActivity.class);
        startActivity(myIntent);
```

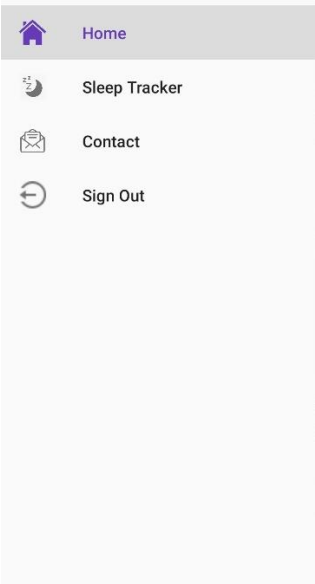


Figure 18 - Sign out button on the navigation bar

3.5 Issues and Future Work

The main issues that was faced for this app was the inaccessibility to TUD servers. In section 3.2 of this report, it outlined the possibilities of including the original objectives of the Brightspace feature and timetabling if the access to the TUD servers was granted. Even with authentication functioning in the app, students are unable to sign into the app with their TUD accounts without TUD admin's approval which was the main focus of the app. If more time was allocated the attendance monitoring feature may have been possible as well, as it was difficult to research all the features and learn how to implement them in the short allocated time for this project without prior knowledge and intermediate skills on developing android applications or programming with Java.

4.0 Conclusion

The final version of the Student Support System App, outcomes includes a functioning android app with Authentication with a navigation bar that has access to the home page, the sleep tracker feature, contact page and to the sign out button. Learning how to develop an android app has helped to develop many skills such as:

Learning the programming language Java: Java was created in 1995 and has become a very popular language. It is used for many kinds of applications for mobile, desktop and the web. There are many used for Java even for games and database connection [33]. Java is one of the easier programming languages and it works on many different platforms like Mac and Windows. Learning many different types of programming languages will broaden career opportunities for a software engineer.

Basis to Application Development: By using the Android Studio IDE it made it easy to learn how to develop apps as it has default templates of UI and code to be able to understand where to add code and the design into the project. Learning how to use Android Studio as it has many functions to build and compile the project as well as an emulator to emulate many types of phone and simulate the old and new versions of Android OS. Learning how to version control is very helpful for projects not just to document and save the files but to allow to compare changes that were made.

Appendix

Project Timeline:

Week | Date	**Aim**	**Comment**
Week 1 | 27/01/2020	Research on features	Researched on authentication and Brightspace API and attendance monitoring.
Week 2 | 03/02/2020	Learn Udacity Online Courses	Get familiar with Android Studio and writing code in java
Week 3 | 10/02/2020	Progress Report Due	The project progress was due and continued with Udacity course.
Week 4 | 17/02/2020	Authentication by Node.js	Researched on Node.js and followed a YouTube video on authentication using Node.js
Week 5 | 24/02/2020	Authentication by Node.js	Following the YouTube Tutorial
Week 6 | 02/03/2020	UI for Login Page	Design and make the UI for login page
Week 7 | 09/03/2020	Make the navigation bar	Using Android Studio, add navigation bar to the page opened after login is successful

Week 8 | 16/03/2020	Research on Firebase and add Firebase to Android Studio	Research on Firebases platform and how to include it into Android Studio
Week 9 | 23/03/2020	Implement Firebase Authentication and add user info to the nav bar	Implemented authentication by Microsoft using Firebase
Week 10 | 30/03/2020	Change the UI of Login page to incorporate authentication with Microsoft	Still implementing authentication and change the design of the UI for the app.
Week 11 | 06/04/2020	Create pages for contact and add information to home page. Start on final report.	Changed the default fragments the features relevant for this app and add information
Week 12 | 13/04/2020	Combined with sleep tracker and cleaned up the code and changed the icons to the nav bar. Build on final report.	Combined the sleep tracker given by Vivien and added different icons and names on the nav bar.
Week 13 | 19/04/2020	Final Project Report Due.	Worked on project report until due date.

Project logs from WordPress:

The following are the logs written on WordPress, accessible using the following link:

https://studentsystem.home.blog/

Week 1:

Overview of first meeting, to make a blog on WordPress. Attempt online courses on Udacity on how to develop android apps. Research on features outlined in the objectives

Android Basics: User Interface

https://www.udacity.com/course/android-basics-user-interface–ud834

Developing Android Apps

https://www.udacity.com/course/new-android-fundamentals–ud851

Week 2

Basics of Android Studio was learnt from the Udacity courses, a birthday card app was made to use ImageView, TextView and LinearLayout.

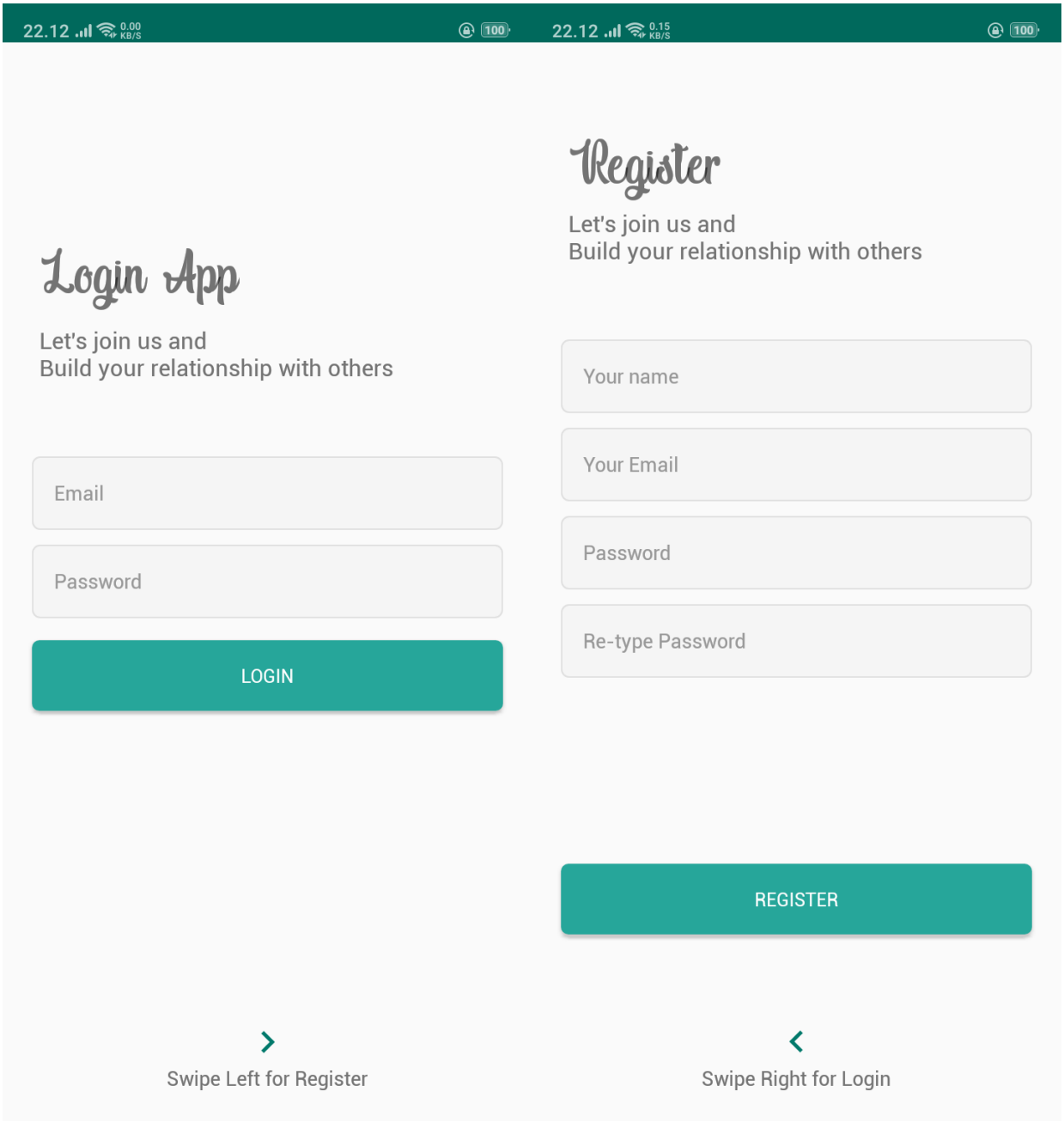
Week 3

Progress report is due, add findings and assumed project timeline. Build a cohesive report.

Week 4-6

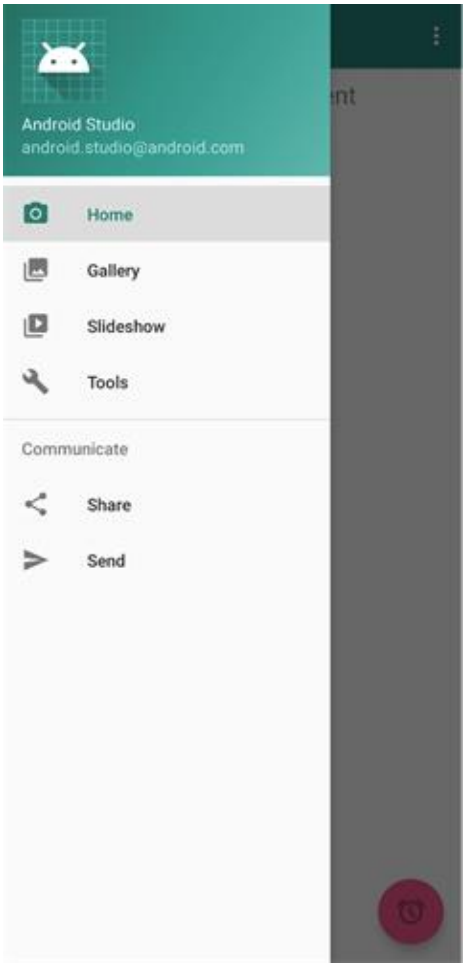
Learning Java by YouTube Videos. Did research on authentication. Recommended the following video that runs through the backend of authentication using Node.js and an express server with MongoDb as the database.

Designed the following user interface for the Login page and register page was made by following the website:



Week 7

When login is successful it will open to the home page where there is also a navigation bar. The navigation bar is available from android studio, so it is prebuilt as a default template.

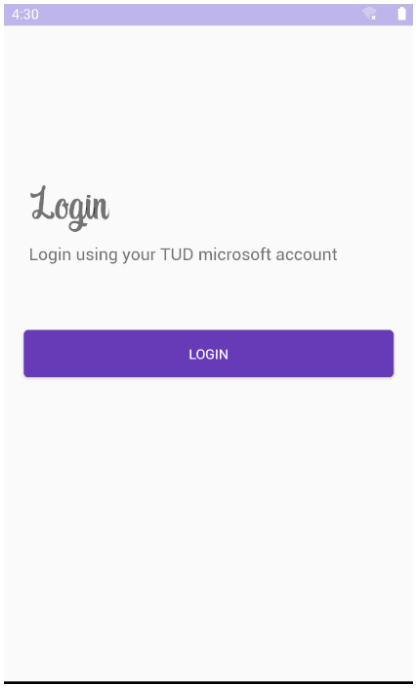


Week 8/9

Researched on authentication with Microsoft using Firebase as it doesn't require to build a server and database. Using Firebase Authentication resolves many things and it is much easier to implement compared to Node.js

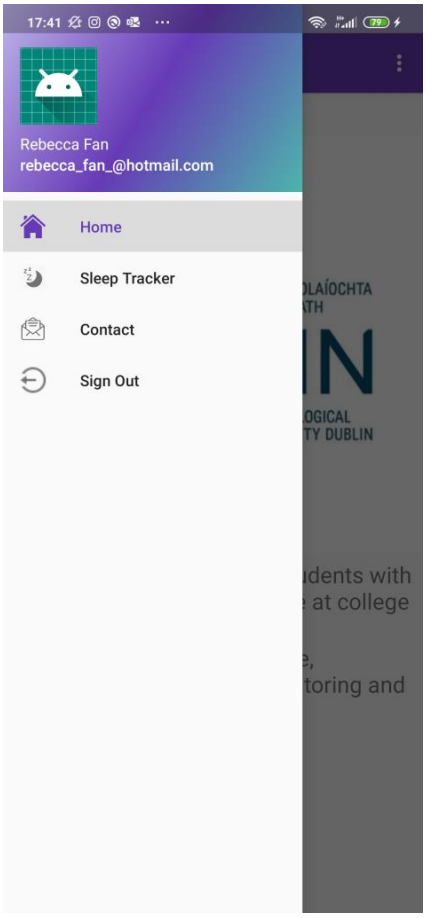
https://firebase.google.com/docs/auth/android/microsoft-oauth

There is no need to have edit text boxes for name, email, etc and for register as it will be replaced by a login button where it will redirect to sign in with Microsoft



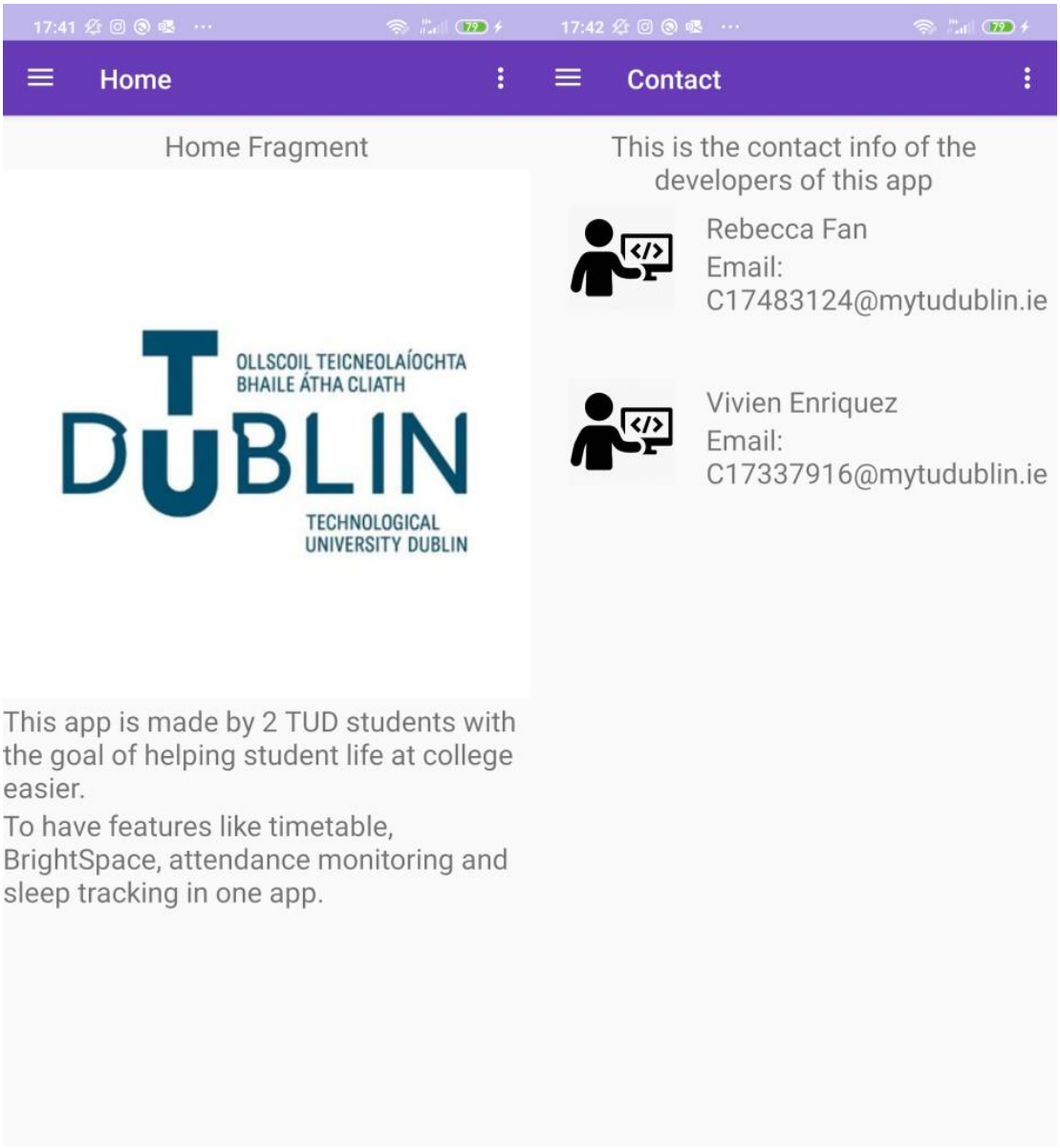
Week 10

Tidy up the navigation bar and added the Name and email on the app drawer when logged in using getname and getemail from the logged in users info. Changed the primary colors and changed the nav drawer icons.



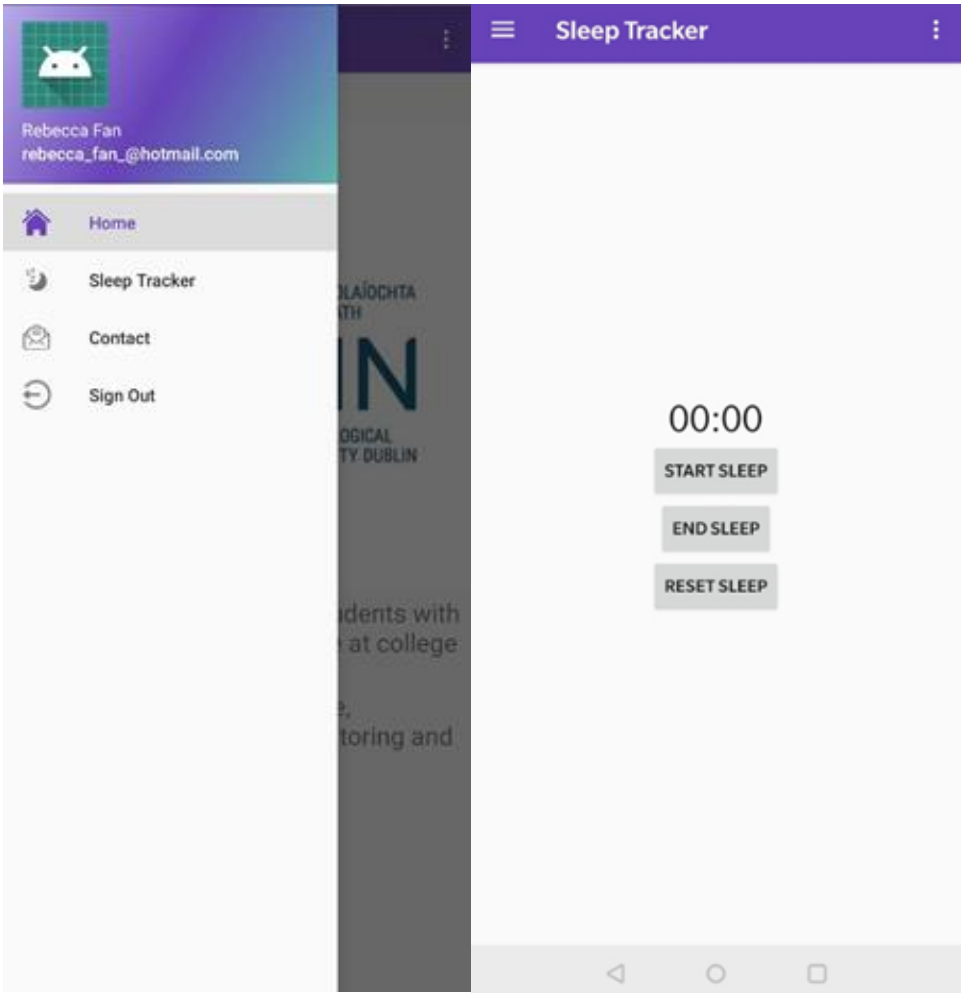
Week 11

Added content info fragment and home info in home fragment.



Week 12

Combined with sleep tracker to produce the final app with all the features.



GitHub

This shows the regularity of committing code to GitHub and may be used as logs for this project:

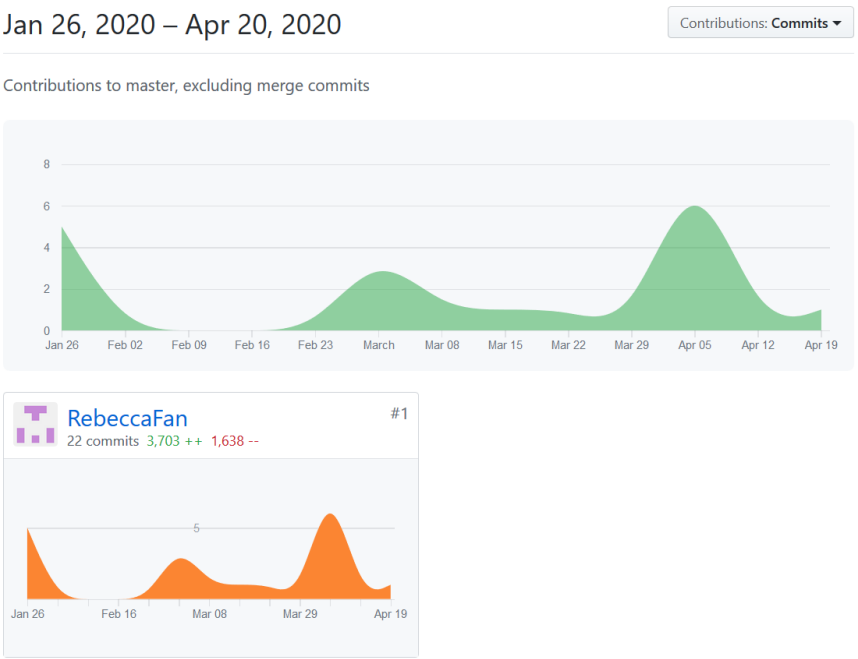


Figure 19 - Graph of committing to GitHub

In the gap of Feb 02-Feb 16 the Node.js version was made, using another repo, the following is the log for the app version with Node.js:

Figure 20 - GitHub commits, Node.js version

The dates for work committed is available on GitHub as well all the code for Student Support System App.

https://github.com/RebeccaFan/StudentSystem

Code for Node.js authentication:

In index.js

```
const express = require("express");
const dotenv = require("dotenv");
const mongoose = require("mongoose");

const app = express();
dotenv.config();
const PORT = process.env.PORT || 8080;

//Import Routes
const authRoute = require("./routes/auth");
const postRoute = require("./routes/posts");
//Connect to DB
mongoose.connect(
  process.env.DB_CONNECT,
  { useNewUrlParser: true, useUnifiedTopology: true },
  () => console.log("Connected to DB!!")
);

//Middleware
app.use(express.json());

//Route Middlewares
```

```
app.use("/api/user", authRoute);
app.use("/api/posts", postRoute);

app.listen(PORT, () => {
  console.log("server up and running");
});
```

In app.js

```
const express = require("express");
const app = express();
const mongoClient = require("mongodb").MongoClient;

const url = "mongodb://localhost:27017";

app.use(express.json());

mongoClient.connect(url, (err, db) => {
  if (err) {
    console.log("Error while connecting mongo client");
  } else {
    const myDb = db.db("myDb");
    const collection = myDb.collection("myTable");

    app.post("/signup", (req, res) => {
      const newUser = {
        name: req.body.name,
        email: req.body.email,
        password: req.body.password
      };

      const query = { email: newUser.email };

      collection.findOne(query, (err, result) => {
        if (result == null) {
          collection.insertOne(newUser, (err, result) => {
            res.status(200).send();
          });
        } else {
          res.status(400).send();
        }
      });
    });

    app.post("/login", (req, res) => {
      const query = {
        email: req.body.email,
        password: req.body.password
      };
```

```
      collection.findOne(query, (err, result) => {
        if (result != null) {
          const objToSend = {
            name: result.name,
            email: result.email
          };

          res.status(200).send(JSON.stringify(objToSend));
        } else {
          res.status(404).send();
        }
      });
    });
  }
});

app.listen(3000, () => {
  console.log("Listening on port 3000...");
});
```

In auth.js

```
const router = require("express").Router();
const User = require("../model/User");
const jwt = require("jsonwebtoken");
const bcrypt = require("bcryptjs");
const { registerValidation, loginValidation } = require("../validation");

router.post("/register", async (req, res) => {
  //validate the data
  const { error } = registerValidation(req.body);
  if (error) return res.status(400).send({ error: error.details[0].message });

  //check if user is in the database
  const emailExist = await User.findOne({ email: req.body.email });
  if (emailExist)
    return res.status(400).send({ error: "Email already exists" });

  //Hash Password
  const salt = await bcrypt.genSalt(10);
  const hashedPassword = await bcrypt.hash(req.body.password, salt);

  //Create a new user
  const user = new User({
    name: req.body.name,
    email: req.body.email,
    password: hashedPassword
  });
  try {
```

```
    const savedUser = await user.save();
    res.status(200).send({ user: user._id });
  } catch (err) {
    res.status(400).send({ error: err });
  }
});

//LogIn
router.post("/login", async (req, res) => {
  const { error } = loginValidation(req.body);
  if (error) return res.status(400).send({ error: error.details[0].message });

  //check if email already exists
  const user = await User.findOne({ email: req.body.email });
  if (!user) return res.status(400).send({ error: "Email does not exist" });
  //Check the password
  const validPass = await bcrypt.compare(req.body.password, user.password);
  if (!validPass) return res.status(400).send({ error: "Invaild Password" });

  //Create token and assign
  const token = jwt.sign({ _id: user._id }, process.env.TOKEN_SECRET);
  res
    .status(200)
    .header("auth-token", token)
    .send({ "auth-token": token });
});
module.exports = router;
```

In posts.js

```
const router = require("express").Router();
const verify = require("./verifytoken");

router.get("/", verify, (req, res) => {
  //   res.json({
  //     posts: {
  //       title: "First Post",
  //       description:  “stuff"
  //     }
  //   });
  res.send({ user: req.user });
});

module.exports = router;
```

In verifytoken.js

```
const jwt = require("jsonwebtoken");

module.exports = function(req, res, next) {
```

```
  const token = req.header("auth-token");
  if (!token) return res.status(401).send("Access denied");

  try {
    const verified = jwt.verify(token, process.env.TOKEN_SECRET);
    req.user = verified;
    next();
  } catch (err) {
    res.status(400).send("Invalid Token");
  }
};
```

In .env

```
DB_CONNECT = mongodb+srv://User1:User@cluster0-
vtvcf.mongodb.net/test?retryWrites=true&w=majority
TOKEN_SECRET = asdfghjkewertyu
```

In package.json

```
{
  "name": "login_backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "dev": "nodemon index.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@hapi/joi": "^17.1.0",
    "bcryptjs": "^2.4.3",
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "jsonwebtoken": "^8.5.1",
    "mongoose": "^5.8.11",
    "nodemon": "^2.0.2"
  }
}
```

Code for the final app:

In AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.studentsystem">

    <application
```

```
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme.NoActionBar">
        <activity
            android:name=".HomeActivity"
            android:label="@string/title_activity_home"
            android:theme="@style/AppTheme.NoActionBar.NoActionBar"></activity>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

In HomeFragment.java:

```
package com.example.studentsystem.ui.home;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProviders;

import com.example.studentsystem.R;

public class HomeFragment extends Fragment {

    private HomeViewModel homeViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        homeViewModel =
                ViewModelProviders.of(this).get(HomeViewModel.class);
        View root = inflater.inflate(R.layout.fragment_home, container, false);
        final TextView textView = root.findViewById(R.id.text_home);
        homeViewModel.getText().observe(this, new Observer<String>() {
            @Override
            public void onChanged(@Nullable String s) {
                textView.setText(s);
            }
        });
        return root;
    }
}
```

In HomeViewModel.java:

```
package com.example.studentsystem.ui.home;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

public class HomeViewModel extends ViewModel {

    private MutableLiveData<String> mText;

    public HomeViewModel() {
        mText = new MutableLiveData<>();
        mText.setValue("Home Fragment");
    }

    public LiveData<String> getText() {
        return mText;
    }
}
```

In SleepTrackerFragment.java:

```
package com.example.studentsystem.ui.sleeptracker;

import android.os.Bundle;
import android.os.SystemClock;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Chronometer;
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProviders;

import com.example.studentsystem.R;

public class SleepTrackerFragment extends Fragment {
    private Chronometer chronometer;
    private boolean running;
    private long pauseOffset;

    private SleepTrackerViewModel sleepTrackerViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        sleepTrackerViewModel =
                ViewModelProviders.of(this).get(SleepTrackerViewModel.class);
        View root = inflater.inflate(R.layout.fragment_sleeptracker, container, false);
        sleepTrackerViewModel.getText().observe(this, new Observer<String>() {
                @Override
         public void onChanged(@Nullable String s) {
         }
                });
```

```
        chronometer = root.findViewById(R.id.chronometer);
        chronometer.setFormat ("Sleep Timer: %s");

        Button chronoStart = root.findViewById(R.id.chronoStart);
        chronoStart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startChronometer(v);
            }
        });
        Button chronoEnd= root.findViewById(R.id.chronoEnd);
        chronoEnd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                endChronometer(v);
            }
        });
        Button chronoReset= root.findViewById(R.id.chronoReset);
        chronoReset.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                resetChronometer(v);
            }
        });
        return root;
    }

    public void startChronometer(View v) {
        if (!running) {
            chronometer.setBase(SystemClock.elapsedRealtime() - pauseOffset);
            chronometer.start();
            running = true;
        }
    }
    public void endChronometer(View v) {
        if (running) {
            chronometer.stop();
            pauseOffset = SystemClock. elapsedRealtime() - chronometer.getBase();
            running = false;
        }
    }
    public void resetChronometer(View v) {
        chronometer.setBase(SystemClock.elapsedRealtime());
        pauseOffset = 0;

    }
}
```

In SleepTrackerViewModel.java:

```
package com.example.studentsystem.ui.sleeptracker;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

public class SleepTrackerViewModel extends ViewModel {
```

```
    private MutableLiveData<String> mText;

    public SleepTrackerViewModel() {
        mText = new MutableLiveData<>();
        mText.setValue("This is gallery fragment");
    }

    public LiveData<String> getText() {
        return mText;
    }
}
```

In ContactFragment.java:

```
package com.example.studentsystem.ui.contact;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProviders;

import com.example.studentsystem.R;

public class ContactFragment extends Fragment {

    private ContactViewModel contactViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        contactViewModel =
                ViewModelProviders.of(this).get(ContactViewModel.class);
        View root = inflater.inflate(R.layout.fragment_contact, container, false);
        final TextView textView = root.findViewById(R.id.text_contact);
        contactViewModel.getText().observe(this, new Observer<String>() {
            @Override
            public void onChanged(@Nullable String s) {
                textView.setText(s);
            }
        });
        return root;
    }
}
```

In ContactViewModel.java:

```
package com.example.studentsystem.ui.contact;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

public class ContactViewModel extends ViewModel {
```

```
    private MutableLiveData<String> mText;

    public ContactViewModel() {
        mText = new MutableLiveData<>();
        mText.setValue("This is the contact info of the developers of this app");
    }

    public LiveData<String> getText() {
        return mText;
    }
}
```

In SignOutFragment.java:

```
package com.example.studentsystem.ui.signout;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import androidx.annotation.Nullable;
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProviders;

import com.example.studentsystem.MainActivity;
import com.example.studentsystem.R;
import com.google.firebase.auth.FirebaseAuth;

public class SignOutFragment extends Fragment {


    private SignOutViewModel signOutViewModel;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        signOutViewModel =
                ViewModelProviders.of(this).get(SignOutViewModel.class);
        View root = inflater.inflate(R.layout.fragment_signout, container, false);
        final TextView textView = root.findViewById(R.id.text_tools);
        signOutViewModel.getText().observe(this, new Observer<String>() {
            @Override
            public void onChanged(@Nullable String s) {
                textView.setText(s);
            }
        });
        Button btn_SignOut = root.findViewById(R.id.btn_SignOut);
        btn_SignOut.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FirebaseAuth.getInstance().signOut();
                Intent myIntent = new Intent(getActivity(),
                        MainActivity.class);
                startActivity(myIntent);
```

```
            }
        });
        return root;



    }

}
```

In SignOutViewModel.java:

```
package com.example.studentsystem.ui.signout;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

public class SignOutViewModel extends ViewModel {

    private MutableLiveData<String> mText;

    public SignOutViewModel() {
        mText = new MutableLiveData<>();
        mText.setValue("This is tools fragment");
    }

    public LiveData<String> getText() {
        return mText;
    }
}
```

In HomeActivity.java:

```
package com.example.studentsystem;

import android.content.Intent;
import android.os.Bundle;

import android.view.MenuItem;
import android.view.View;

import androidx.annotation.NonNull;
import androidx.core.view.GravityCompat;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;

import com.google.android.material.navigation.NavigationView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import androidx.drawerlayout.widget.DrawerLayout;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.view.Menu;
import android.widget.TextView;
```

```
public class HomeActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener{

    private AppBarConfiguration mAppBarConfiguration;
    DrawerLayout drawer;
    NavController navController;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_sleeptracker, R.id.nav_contact)
                .setDrawerLayout(drawer)
                .build();
        navController = Navigation.findNavController(this, R.id.nav_host_fragment);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);
        navigationView.setNavigationItemSelectedListener(this);

        View headerView = navigationView.getHeaderView(0);

        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        TextView username = headerView.findViewById(R.id.username);
        username.setText(firebaseUser.getDisplayName());
        TextView email = headerView.findViewById(R.id.email);
        email.setText(firebaseUser.getEmail());

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.home, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        // Handle navigation view item clicks here.
        switch (item.getItemId()) {
            case R.id.nav_signout: {
                FirebaseAuth.getInstance().signOut();
                Intent myIntent = new Intent(HomeActivity.this,
                        MainActivity.class);
```

```
                startActivity(myIntent);
                break;
            }
            case R.id.nav_sleeptracker: {
                navController.navigate(R.id.nav_sleeptracker);
                break;
            }
            case R.id.nav_contact: {
                navController.navigate(R.id.nav_contact);
                break;
            }
            case R.id.nav_home: {
                navController.navigate(R.id.nav_home);
                break;
            }
        }
        //close navigation drawer
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }

}
```

In LoginFragment.java:

```
package com.example.studentsystem;


import android.content.Intent;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.OAuthProvider;

import java.util.ArrayList;
import java.util.List;


/**
 * A simple {@link Fragment} subclass.
 */
public class LoginFragment extends Fragment {

    FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();

    public LoginFragment() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_login, container, false);
        Button button = (Button) view.findViewById(R.id.btn_login);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                // https://firebase.google.com/docs/auth/android/microsoft-
oauth#before_you_begin

                //FirebaseAuth.getInstance().signOut();
                OAuthProvider.Builder provider =
OAuthProvider.newBuilder("microsoft.com");
                // Force re-consent.
                provider.addCustomParameter("prompt", "select_account");
```

```
                List<String> scopes =
                        new ArrayList<String>() {
                            {
                                add("mail.read");
                                add("calendars.read");
                            }
                        };
                provider.setScopes(scopes);

                Task<AuthResult> pendingResultTask =
firebaseAuth.getPendingAuthResult();
                if (pendingResultTask != null) {
                    // There's something already here! Finish the sign-in for your user.
                    pendingResultTask
                            .addOnSuccessListener(
                                    new OnSuccessListener<AuthResult>() {
                                        @Override
                                        public void onSuccess(AuthResult authResult) {
                                            // User is signed in.
                                            // IdP data available in
                                            //
authResult.getAdditionalUserInfo().getProfile().
                                            // The OAuth access token can also be
retrieved:
                                            //
authResult.getCredential().getAccessToken().
                                            // The OAuth ID token can also be retrieved:
                                            // authResult.getCredential().getIdToken().
                                            Toast.makeText(getActivity(), "Hi " +
authResult.getUser().getDisplayName() + " \nSuccessfully logged in with microsoft !!",
Toast.LENGTH_LONG).show();
                                            Intent myIntent = new Intent(getActivity(),
                                                    HomeActivity.class);
                                            startActivity(myIntent);
                                        }
                                    })
                            .addOnFailureListener(
                                    new OnFailureListener() {
                                        @Override
                                        public void onFailure(@NonNull Exception e) {
                                            // Handle failure.
                                            Log.d("Failed", "onFailure: ");
                                            Toast.makeText(getActivity(), "Failed to
login with microsoft", Toast.LENGTH_LONG).show();
                                        }
                                    });
                } else {
                    // There's no pending result so you need to start the sign-in flow.
                    // See below.
                    firebaseAuth
                            .startActivityForSignInWithProvider(/* activity= */
getActivity(), provider.build())
                            .addOnSuccessListener(
                                    new OnSuccessListener<AuthResult>() {
                                        @Override
                                        public void onSuccess(AuthResult authResult) {
                                            Toast.makeText(getActivity(), "Hi " +
authResult.getUser().getDisplayName() + " \nSuccessfully logged in with microsoft !!",
Toast.LENGTH_LONG).show();
```

```
                                        Intent myIntent = new Intent(getActivity(),
                                                HomeActivity.class);
                                        startActivity(myIntent);
                                    }
                                })
                            .addOnFailureListener(
                                    new OnFailureListener() {
                                        @Override
                                        public void onFailure(@NonNull Exception e) {
                                            // Handle failure.
                                            Toast.makeText(getActivity(), "Failed to
login with microsoft", Toast.LENGTH_LONG).show();
                                        }
                                    });
                }

            }

        });
        // Inflate the layout for this fragment
        return view;
    }

}
```

In MainActivity.java:

```
package com.example.studentsystem;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentPagerAdapter;
import androidx.viewpager.widget.ViewPager;

import android.os.Bundle;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ViewPager viewPager = findViewById(R.id.viewPager);

        AuthenticationPagerAdapter pagerAdapter = new
AuthenticationPagerAdapter(getSupportFragmentManager());
        pagerAdapter.addFragmet(new LoginFragment());
        viewPager.setAdapter(pagerAdapter);
    }
}
class AuthenticationPagerAdapter extends FragmentPagerAdapter {
    private ArrayList<Fragment> fragmentList = new ArrayList<>();

    public AuthenticationPagerAdapter(FragmentManager fm) {
```

```
        super(fm);
    }

    @Override
    public Fragment getItem(int i) {
        return fragmentList.get(i);
    }

    @Override
    public int getCount() {
        return fragmentList.size();
    }

    void addFragmet(Fragment fragment) {
        fragmentList.add(fragment);
    }

}
```

In activity_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_home"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />


    <com.google.android.material.navigation.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_home"
        app:menu="@menu/activity_home_drawer" />

</androidx.drawerlayout.widget.DrawerLayout>
```

In activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.viewpager.widget.ViewPager
        android:id="@+id/viewPager"
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</RelativeLayout>
```

In app_bar_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomeActivity">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.NoActionBar.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.NoActionBar.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_home" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

In fragment_contact.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/text_contact"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:gravity="center"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


    <LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:weightSum="3"
        android:paddingBottom="40dp">


        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center">

            <ImageView
                android:id="@+id/dev_icon"
                android:layout_width="80dp"
                android:layout_height="80dp"
                android:adjustViewBounds="true"
                android:scaleType="fitCenter"
                android:layout_marginLeft="5dp"
                android:src="@drawable/developer_icon">

            </ImageView>

        </LinearLayout>

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="2"
            android:gravity="left"
            android:orientation="vertical">
    <TextView
        android:id="@+id/name_rf"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_marginTop="8dp"
        android:text="Rebecca Fan" />

    <TextView
        android:id="@+id/email_rf"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="Email: C17483124@mytudublin.ie"
        />

    </LinearLayout>
</LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:weightSum="3">

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
```

```
            android:layout_weight="1"
            android:gravity="center">

            <ImageView
                android:id="@+id/dev_icon2"
                android:layout_width="80dp"
                android:layout_height="80dp"
                android:adjustViewBounds="true"
                android:scaleType="fitCenter"
                android:layout_marginLeft="5dp"
                android:src="@drawable/developer_icon">

            </ImageView>

        </LinearLayout>

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="2"
            android:gravity="left"
            android:orientation="vertical">

            <TextView
                android:id="@+id/name_ve"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:text="Vivien Enriquez"
                android:textSize="20sp" />

            <TextView
                android:id="@+id/me_text_view"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Email: C17337916@mytudublin.ie"
                android:textSize="20sp" />

        </LinearLayout>
    </LinearLayout>
    </LinearLayout>
```

In fragment_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >

    <TextView
        android:id="@+id/text_home"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
```

```
        android:textAlignment="center"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


    <ImageView
        android:id="@+id/TUDLogo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerInside"
        app:srcCompat="@drawable/tud" />

    <TextView
        android:id="@+id/home_desc"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="This app is made by 2 TUD students with the goal of helping
student life at college easier."
        />

    <TextView
        android:id="@+id/home_desc2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="To have features like timetable, BrightSpace, attendance
monitoring and sleep tracking in one app."
        />

</LinearLayout>
```

In fragment_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".LoginFragment">

    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/app_name"
        android:textSize="50sp"
        android:fontFamily="@font/indigo_daisy"
        android:layout_marginStart="25dp"
        android:layout_marginBottom="10dp"
        android:layout_above="@id/tv_subtitle"/>

    <TextView
        android:id="@+id/tv_subtitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/tag"
        android:textSize="17sp"
```

```
        android:fontFamily="@font/roboto_regular"
        android:layout_marginStart="25dp"
        android:layout_marginBottom="50dp"
        android:layout_above="@id/btn_login"/>


    <Button
        android:id="@+id/btn_login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:background="@drawable/btn_custom"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:layout_marginTop="15dp"
        android:fontFamily="@font/roboto_regular"
        android:textColor="@android:color/white"
        android:text="@string/login"/>


</RelativeLayout>
```

In fragment_signout.xml

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

    <TextView
        android:id="@+id/text_tools"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:textAlignment="center"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:gravity="center_horizontal" />

    <Button
        android:id="@+id/btn_SignOut"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sign Out" />




</RelativeLayout>
```

In fragment_sleeptracker.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    >

    <Chronometer
        android:id="@+id/chronometer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"/>
    <Button
        android:id="@+id/chronoStart"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start Sleep" />

    <Button
        android:id="@+id/chronoEnd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="End Sleep"
        android:onClick="endChronometer"/>

    <Button
        android:id="@+id/chronoReset"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Reset Sleep"
        android:onClick="resetChronometer"/>



</LinearLayout>
```

In nav_header_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="@dimen/nav_header_height"
    android:background="@drawable/side_nav_bar"
    android:gravity="bottom"
    android:orientation="vertical"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:theme="@style/ThemeOverlay.AppCompat.Dark">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/nav_header_desc"
```

```
        android:paddingTop="@dimen/nav_header_vertical_spacing"
        app:srcCompat="@mipmap/ic_launcher_round" />

    <TextView
        android:id="@+id/username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="@dimen/nav_header_vertical_spacing"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

    <TextView
        android:id="@+id/email"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="@style/TextAppearance.AppCompat.Body2" />


</LinearLayout>
```

In activity_home_drawer.xml

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

    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_home"
            android:icon="@drawable/home"
            android:title="@string/menu_home" />
        <item
            android:id="@+id/nav_sleeptracker"
            android:icon="@drawable/sleep"
            android:title="Sleep Tracker" />
        <item
            android:id="@+id/nav_contact"
            android:icon="@drawable/email_icon"
            android:title="Contact" />
        <item
            android:id="@+id/nav_signout"
            android:icon="@drawable/signout"
            android:title="Sign Out" />
    </group>



</menu>
```

In mobile_navigation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mobile_navigation"
    app:startDestination="@+id/nav_home">

    <fragment
        android:id="@+id/nav_home"
        android:name="com.example.studentsystem.ui.home.HomeFragment"
```

```
        android:label="@string/menu_home"
        tools:layout="@layout/fragment_home" />

    <fragment
        android:id="@+id/nav_sleeptracker"
        android:name="com.example.studentsystem.ui.sleeptracker.SleepTrackerFragment"
        android:label="@string/menu_sleeptracker"
        tools:layout="@layout/fragment_sleeptracker" />

    <fragment
        android:id="@+id/nav_contact"
        android:name="com.example.studentsystem.ui.contact.ContactFragment"
        android:label="@string/menu_contact"
        tools:layout="@layout/fragment_contact" />

    <fragment
        android:id="@+id/nav_signout"
        android:name="com.example.studentsystem.ui.signout.SignOutFragment"
        android:label="@string/menu_signout"
        tools:layout="@layout/fragment_signout" />

</navigation>
```

In colors.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#673AB7</color>
    <color name="colorPrimaryDark">#B2A598E4</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

In dimens.xml

```
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
    <dimen name="nav_header_vertical_spacing">8dp</dimen>
    <dimen name="nav_header_height">176dp</dimen>
    <dimen name="fab_margin">16dp</dimen>
</resources>
```

In strings.xml

```
<resources>
    <string name="app_name">Login</string>

    <string name="hello_blank_fragment">Hello blank fragment</string>
    <string name="login">Login</string>
    <string name="password">Password</string>
    <string name="e_mail">Email</string>
    <string name="tag">Login using your TUD microsoft account</string>
    <string name="title_activity_home">HomeActivity</string>
    <string name="navigation_drawer_open">Open navigation drawer</string>
    <string name="navigation_drawer_close">Close navigation drawer</string>
    <string name="nav_header_title">Android Studio</string>
    <string name="nav_header_subtitle">android.studio@android.com</string>
    <string name="nav_header_desc">Navigation header</string>
    <string name="action_settings">Settings</string>

    <string name="menu_home">Home</string>
```

```
    <string name="menu_sleeptracker">Sleep Tracker</string>
    <string name="menu_contact">Contact</string>
    <string name="menu_signout">Sign Out</string>
    <string name="menu_share">Share</string>
    <string name="menu_send">Send</string>



</resources>
```

In values.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="default_web_client_id" translatable="false">421976284058-
pmkrd8t44b8lfuk45cq6h1v04ptvjm08.apps.googleusercontent.com</string>
    <string name="firebase_database_url" translatable="false">https://studentsystem-
7b9cd.firebaseio.com</string>
    <string name="gcm_defaultSenderId" translatable="false">421976284058</string>
    <string name="google_api_key"
translatable="false">AIzaSyDzMjKnifOIlVgOrsZ_Mlc6WoIhFrnnQUk</string>
    <string name="google_app_id"
translatable="false">1:421976284058:android:ece383b3228de641a4b823</string>
    <string name="google_crash_reporting_api_key"
translatable="false">AIzaSyDzMjKnifOIlVgOrsZ_Mlc6WoIhFrnnQUk</string>
    <string name="google_storage_bucket" translatable="false">studentsystem-
7b9cd.appspot.com</string>
    <string name="project_id" translatable="false">studentsystem-7b9cd</string>
</resources>
```

In build.Gradle(StudentSystem)

```
// Top-level build file where you can add configuration options common to all sub-
projects/modules.

buildscript {
    repositories {
        google()
        jcenter()

    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.6.2'
        classpath 'com.google.gms:google-services:4.3.3'

    }
}

allprojects {
    repositories {
        google()
        jcenter()

    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

In build.Gradle(app)

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.2"
    defaultConfig {
        applicationId "com.example.studentsystem"
        minSdkVersion 16
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'com.google.android.material:material:1.0.0'
    implementation 'androidx.navigation:navigation-fragment:2.0.0'
    implementation 'androidx.navigation:navigation-ui:2.0.0'
    implementation 'androidx.lifecycle:lifecycle-extensions:2.0.0'
    implementation 'com.google.firebase:firebase-analytics:17.2.2'
    implementation 'com.google.firebase:firebase-auth:19.2.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
}
```

References

[1]"How Many People Have Smartphones in 2020 | Oberlo", *Oberlo*. [Online]. Available: https://ie.oberlo.com/statistics/how-many-people-have-smartphones. [Accessed: 19- Apr- 2020]

[2]M. Rouse, "What is mobile app? - Definition from WhatIs.com", *WhatIs.com*, 2013. [Online]. Available: https://whatis.techtarget.com/definition/mobile-app. [Accessed: 19- Apr- 2020]

[3]"What is a Mobile Application? - Definition from Techopedia", *Techopedia.com*, 2018. [Online]. Available: https://www.techopedia.com/definition/2953/mobile-application-mobile-app. [Accessed: 19- Apr- 2020]

[4]S. Siddhartha, "Android mobile web developer essential skills", *Pluralsight.com*, 2019. [Online]. Available: https://www.pluralsight.com/blog/software-development/10-skills-for-android-devs-. [Accessed: 19- Apr- 2020]

[5]"Android Basics: User Interface | Udacity", *Udacity.com*. [Online]. Available: https://www.udacity.com/course/android-basics-user-interface--ud834. [Accessed: 19- Apr- 2020]

[6]V. Beal, "What is Markup Language? Webopedia Definition", *Webopedia.com*. [Online]. Available: https://www.webopedia.com/TERM/M/markup_language.html. [Accessed: 19- Apr- 2020]

[7]"What is an IDE?", *Redhat.com*. [Online]. Available: https://www.redhat.com/en/topics/middleware/what-is-ide. [Accessed: 19- Apr- 2020]

[8]"What is the use of Android studio?", *Quora*. [Online]. Available: https://www.quora.com/What-is-the-use-of-Android-studio. [Accessed: 19- Apr- 2020]

[9]"Intro to Github for version control", *Ourcodingclub.github.io*. [Online]. Available: https://ourcodingclub.github.io/tutorials/git/. [Accessed: 19- Apr- 2020]

[10]"Welcome — Developer Platform (April 2020)", *Docs.valence.desire2learn.com*. [Online]. Available: https://docs.valence.desire2learn.com/. [Accessed: 19- Apr- 2020]

[11]"TUD Timetabling (DIT)", *Play.google.com*, 2020. [Online]. Available: https://play.google.com/store/apps/details?id=com.baz.oli.dittimetable&hl=en_US. [Accessed: 19- Apr- 2020]

[12]K. Tang, "How we designed an NFC system to track event attendance at our hackathon", *Medium*, 2019. [Online]. Available: https://medium.com/vandyhacks/how-we-designed-an-nfc-system-to-track-event-attendance-at-our-hackathon-dfa2cb49480d. [Accessed: 19- Apr- 2020]

[13]"What is a Radio Frequency Identification Reader (RFID Reader)? - Definition from Techopedia", *Techopedia.com*. [Online]. Available: https://www.techopedia.com/definition/26992/radio-frequency-identification-reader-rfid-reader. [Accessed: 19- Apr- 2020]

[14]"Node.js Introduction", *W3schools.com*. [Online]. Available: https://www.w3schools.com/nodejs/nodejs_intro.asp. [Accessed: 19- Apr- 2020]

[15]"Get Started with JSON Web Tokens - Auth0", *Auth0*. [Online]. Available: https://auth0.com/learn/json-web-tokens/. [Accessed: 19- Apr- 2020]

[16]"The most popular database for modern apps", *MongoDB*. [Online]. Available: https://www.mongodb.com. [Accessed: 19- Apr- 2020]

[17]B. Batbold, "Working with MongoDB (Mongoose) on NodeJS", *Medium*, 2019. [Online]. Available: https://medium.com/@bilguun132/working-with-mongodb-mongoose-on-nodejs-2317e0fe2290. [Accessed: 19- Apr- 2020]

[18]B. Batbold, "Handling Authentication and Authorization with Node", *Medium*, 2019. [Online]. Available: https://medium.com/quick-code/handling-authentication-and-authorization-with-node-7f9548fedde8. [Accessed: 19- Apr- 2020]

[19]"Firebase", *Firebase*. [Online]. Available: https://firebase.google.com/. [Accessed: 19- Apr- 2020]

[20]"Authenticate Using Microsoft on Android | Firebase", *Firebase*. [Online]. Available: https://firebase.google.com/docs/auth/android/microsoft-oauth#handle_the_sign-in_flow_with_the_firebase_sdk. [Accessed: 19- Apr- 2020]

[21]"University Health Center | Managing Stress | Sleep | University Health Center", *Uhs.uga.edu*. [Online]. Available: https://www.uhs.uga.edu/sleep. [Accessed: 19- Apr- 2020]

[22] "Chronometer", *Developer.android.com*. [Online]. Available: [Online]. Available: https://developer.android.com/reference/android/widget/Chronometer. [Accessed: 19- Apr- 2020]

[23]"Prototyping 101: The Difference between Low-Fidelity and High-Fidelity Prototypes and When to Use Each | Adobe Blog", *Adobe Blog*. [Online]. Available: https://theblog.adobe.com/prototyping-difference-low-fidelity-high-fidelity-prototypes-use/. [Accessed: 19- Apr- 2020]

[24]D. Ed, "Build A Node.js API Authentication With JWT Tutorial", YouTube, 2019. [Online]. Available: https://www.youtube.com/watch?v=2jqok-WgelI. [Accessed: 09- Feb- 2020]

[25]"Add Firebase to your Android project", *Firebase*. [Online]. Available: https://firebase.google.com/docs/android/setup#prerequisites. [Accessed: 19- Apr- 2020]

[26]"Authenticate Using Microsoft on Android | Firebase", *Firebase*. [Online]. Available: https://firebase.google.com/docs/auth/android/microsoft-oauth#before_you_begin. [Accessed: 19- Apr- 2020]

[27]"What is Azure Active Directory? - Azure Active Directory", *Docs.microsoft.com*. [Online]. Available: https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis. [Accessed: 19- Apr- 2020]

[28]"What the Heck is OAuth?", *Okta Developer*. [Online]. Available: https://developer.okta.com/blog/2017/06/21/what-the-heck-is-oauth. [Accessed: 19- Apr- 2020]

[29]M. Jalaludin, "How to Create User Interface Login & Register with Android Studio", *Medium*, 2018. [Online]. Available: https://medium.com/muhamadjalaludin/how-to-create-user-interface-login-register-with-android-studio-34da847b05b2. [Accessed: 19- Apr- 2020]

[30]S. Kantamani, "What Is a Target Fragment?", *Medium*, 2020. [Online]. Available: https://medium.com/better-programming/what-is-target-fragment-da0e7c7f345c. [Accessed: 19- Apr- 2020]

[31]"Button", *Developer.android.com*. [Online]. Available: https://developer.android.com/guide/topics/ui/controls/button. [Accessed: 19- Apr- 2020]

[32]"Android TextView with Examples - Tutlane", *Tutlane.com*. [Online]. Available: https://www.tutlane.com/tutorial/android/android-textview-with-examples. [Accessed: 19- Apr- 2020]

[33]"Java Introduction", *W3schools.com*. [Online]. Available: https://www.w3schools.com/java/java_intro.asp. [Accessed: 19- Apr- 2020]