Design

For this project I have 4 pieces that need to come together, I need a map on which to display Portsmouth and all the users, the rest of the front-end of the app, the database, and finally the back-end server to link the database to the front-end.

For the map I will use the OpenLayers JavaScript library as I have a lot of experience with it using it for 2 previous projects. It is also free and allows me to both display a map, which will be an OpenStreetMap map, as well as overlay markers and shapes onto said map. Also since this is a JavaScript library it is compatible with any framework through the use of a WebView.

For the rest of the front-end of the app, I will use Flutter as it allows me to develop both for iOS and Android whilst writing just one set of code. Flutter also has a very large set of libraries developed for it that will allow me to easily and efficiently extend the functionality of the app. Finally, I have experience using Flutter to develop an App as I have previously used it to create my app University Bus Portsmouth.

The server will be hosted on AWS as I have experience using this tool and it allows me to efficiently create a database using Amazon RDS and create AWS lambda methods to add additional features. AWS also has an email service that I can utilise to send the login codes.

The database will be a MySQL database as this is the type that I am most familiar with and it has the capabilities that I require.

I have split the rest of the design of this project into 3 distinct sections: the UI and visual design; the systems architectural design; and finally the database design. Whilst these are presented as separate, the design was not done separately, instead an iterative approach was taken to the design when adding each feature. A part of the UI would be designed to display the feature to the user, the architectural design was then updated to support this feature, and finally the database was updated as necessary to complete the design of the feature from front end to back.

UI and Visual Design

Since this is an app for the University of Portsmouth, the aim is to use the colours of the university, which are primarily a dark purple and blue.

To create initial designs for the UI I will use Flutter Viz as it is a graphical tool meant for building UI to use for Flutter projects. Due to this the designs used in the project itself should be similar, but will have to be recreated out of Flutter code.

Login Screen

The login screen needs to allow users to enter their email and then enter the code sent to their email. So first I will display a text box and text field allowing users to enter their email. Then only once that is done I will display a box for the code to be entered. The background will be coloured in purple.

A computer screen shot of a email box

Description automatically generated A screenshot of a computer

Description automatically generated

Main Screen

The main screen will be divided into three sections, the app bar at the top coloured in purple, the navigation bar at the bottom, and the content of the current screen in the middle.

A black rectangle with white dots

Description automatically generated

In this case since the current screen is the home screen which contains a Map at the centre which cannot be replicated in Flutter Viz, this Map has been left blank. The map displayed will be an Open Street Map with markers and shapes overlaid.

Connections

The connections screen should allow users to manage who has permission to see them, and who they have permission to see. To allow this, the connections screen will first display a list of all the users that this user either has permission to see, or that has permission to see this user. I will also supply buttons next to each user in the list so the given user can manage whether they should have permission or not.

I will also have a button at the top that takes the user to a screen that will let them request permission to locate other users. This screen will then allow users to search for other users and provide a button to request permission from them.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Profile

The profile screen should allow users to enter their name, faculty, school, and if applicable, course. It should also allow users to select which graduation zones they want to be seen in.

A screenshot of a cell phone

Description automatically generated

Architectural Design

The architecture can be split into 2 parts, the front-end and back-end. The back-end will be created out of a set of AWS lambda methods that will retrieve data from the database and set data into the database. All of these lambda methods, except the login methods, will be authenticated with a JWT(json web token) that will be generated by the server when the user logs in. The use of JWTs will allow all requests sent to the server to be associated with the user that sent the request.

The front-end design will be based around the different screens as there will not be a significant part of the front end that is not directly working with the UI. The UI will be built by embedding screens(widgets) within other screens(widgets), this is standard for Flutter. However, for this project, all functionality will be to its relevant screen, and if its needed by multiple screens, then it is attached to the closest parent screen to both of the child screens.

A diagram of a company

Description automatically generated

Database

The database is a MySQL database hosted on AWS using the RDS. AWS was chosen for this project as they have a very reliable database service that I have used before whilst also allowing for back-end processing to be written and deployed efficiently using AWS lambda.

I used https://app.diagrams.net/ to create an entity relationship diagram for my database. There are four parts of this schema all linked by the user table, the location permission part, the course part, the graduation zones part, and finally the user table itself.

A screenshot of a computer

Description automatically generated

User Table

This table stores everything directly related to a user, which includes data such as their name and email address, but also their course, their last known latitude and longitude along with the time when that location was set, as well as their login code. When each user first enters their email address a unique user id is generated for them and that is set as the primary key in the database so that in the future if support for changing your email address is desired then it will be more feasible to add.

The has\_logged\_in row and the email\_verified are separate as has\_logged\_in is used to direct the app to take the user to their profile if they have never edited it before and so remains false until the profile has been edited, whilst email\_verified is false until the user correctly enters the code which will log them in for the first time.

Location Permissions

The location\_permission table is used to give permission to see other users. When a user, let’s say User A, first asks for permission from another user, User B, a new entry gets added to the location\_permission table where from\_user is the user id of User A, to\_user is the user id of user B, and permission\_granted is set to Requested. Then when user B accepts the request, the permission\_granted row is changed to Granted. This now allows for User A to see the location of User B, but not the other way around. If User B wants to see User A then they must request permission separately, at which point a new entry will be created in the location\_permission table where from\_user is the user id of User B, to\_user is the user id of user A, and permission\_granted is set to Requested.

This approach of creating separate entries in the location\_permission table for when User A requests permission from User B and from the other way around allows for location permissions to be managed separately, as in it allows User A to see User B but not the other way around, whilst still keeping an efficient and easy to understand database design.

Course

The faculty, school and course tables do not store any information about the users themselves. What these tables do store is the academic structure of the university, all the courses on offer, linked to which school they are in, and linking each school to which faculty they are in. This then allows the user table to efficiently store the faculty, school and course of the user by simply linking to these tables.

The advantage of this is that when a course, school or even faculty is removed from the University of Portsmouth, its entry into its table can simply be deleted and its presence in the entries of the users who have added it to their profile and easily be deleted as well by cascade.

Graduation Zones

The actual geojsons for the graduation zones are stored in the graduation\_zones\_text table whilst the label for the zone is stored in the graduation\_zones table. These are separate due the fact that the geojsons are too large to be indexed in a MySQL database, and so the solution to this problem is to store the zone id and label in a separate table that can then be indexed for fast retrieval. These zones are then linked to users using the user\_zones table.

When a user saves to their profile that they want to appear in a given zone, then an entry in the user\_zones table is created with both the users id and the zones id. When a user then saves that they no longer want to appear in a zone, that entry is then deleted. This allows for quick retrieval of which zones a user wants to appear in, not getting slowed down by the lack of indexing in the graduation\_zones\_text table.