ECE 651 Project Deliverable 2: Prototype

1. Metadata

1.1 Project Title

UW Carpool Application

1.2 Team Members Information

Name	Quest ID	Student ID
Jiatong He	j246he	20738590
Shuting Lian	s6lian	20718506
Zhilun Chang	z9chang	20727042
Zizheng Jiang	z225jian	20716960

2. Overview

For this project, we are all beginners, so in the early stage we have been learning basics of Android and Kotlin. We learned about the four components of Android and its life cycle; we learned how to find data relationships and build related tables and the database; we also searched and learned some Android APIs on the Internet and how to use them in our project.

At the same time, we also encountered some difficulties in the process of learning. We were not familiar with IDE and some other tools, which had delayed our initial development process. For example, we didn't know the IDE had its own VCS (version control system) tool, which made us spend a lot of time to upload the project on Github. We also didn't know the version of related API and didn't understand the IDE's error message, so we had a lot of problems when building the project. When building the ER diagram, we had some controversies about the relationship and the representation method between data.

After learning materials related to those topics and understanding base contents, the basic architecture of the application was clarified among group members. At the same time, a basic and unified opinion was reached on how to implement this application.

2.1 Architecture Overview

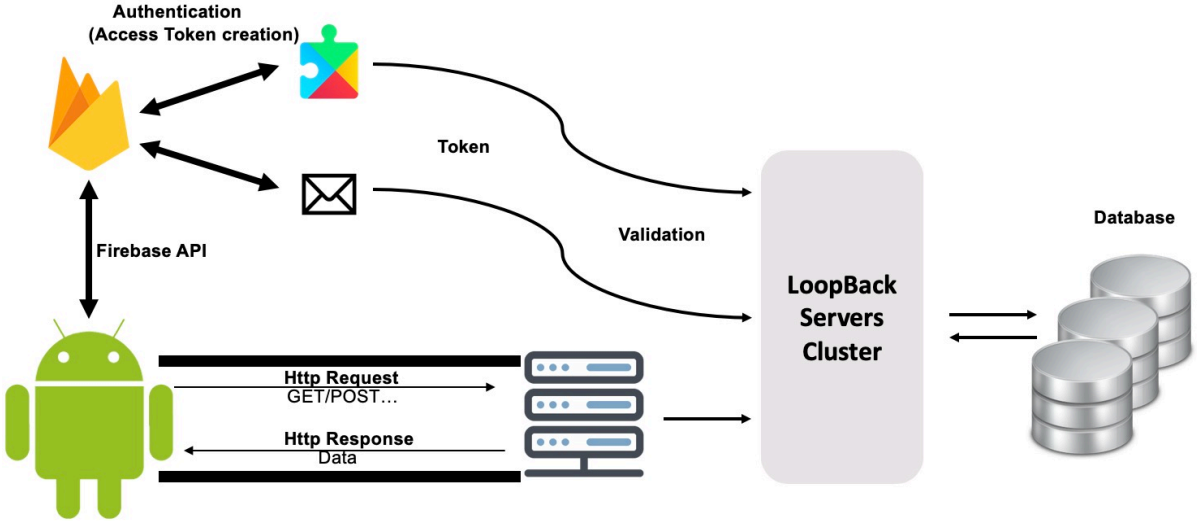

Figure 1. Application Architecture

As we can see from the above figure, we use the Firebase API to implement the login authentication. In order for the users to login to the UW Carpool Application, the application first need to get the user's

authentication credentials. These credentials can be user's email address and password, or OAuth tokens generated by third-party identity provider services (google, twitter and so on). After that, the application can pass these credentials to the Firebase Authentication SDK. The Firebase back-end service then validates these credentials and returns the response to the client. Once the user has successfully logged in, the application can access the user's basic profile information and control user's access to data stored in the application. The application can also verify the identity of the user in our own back-end service using the provided authentication token.

Based on our user scenarios analysis in deliverable 1. When the user is a passenger, our application generates relevant database search statements based on the content s/he entered, uses HTTP Request GET method to retrieve the relevant data and returns to the related activity as a list. When the user is a driver, the application generates a corresponding database insert statement based on the content s/he inputs and then uses the HTTP Request POST method to insert the relevant data into related tables in the database. We may implement the function to modify the number of vacancies in the driver's table in the database after the passenger confirms that they are going to take the specific driver's car.

2.2 Component Diagram

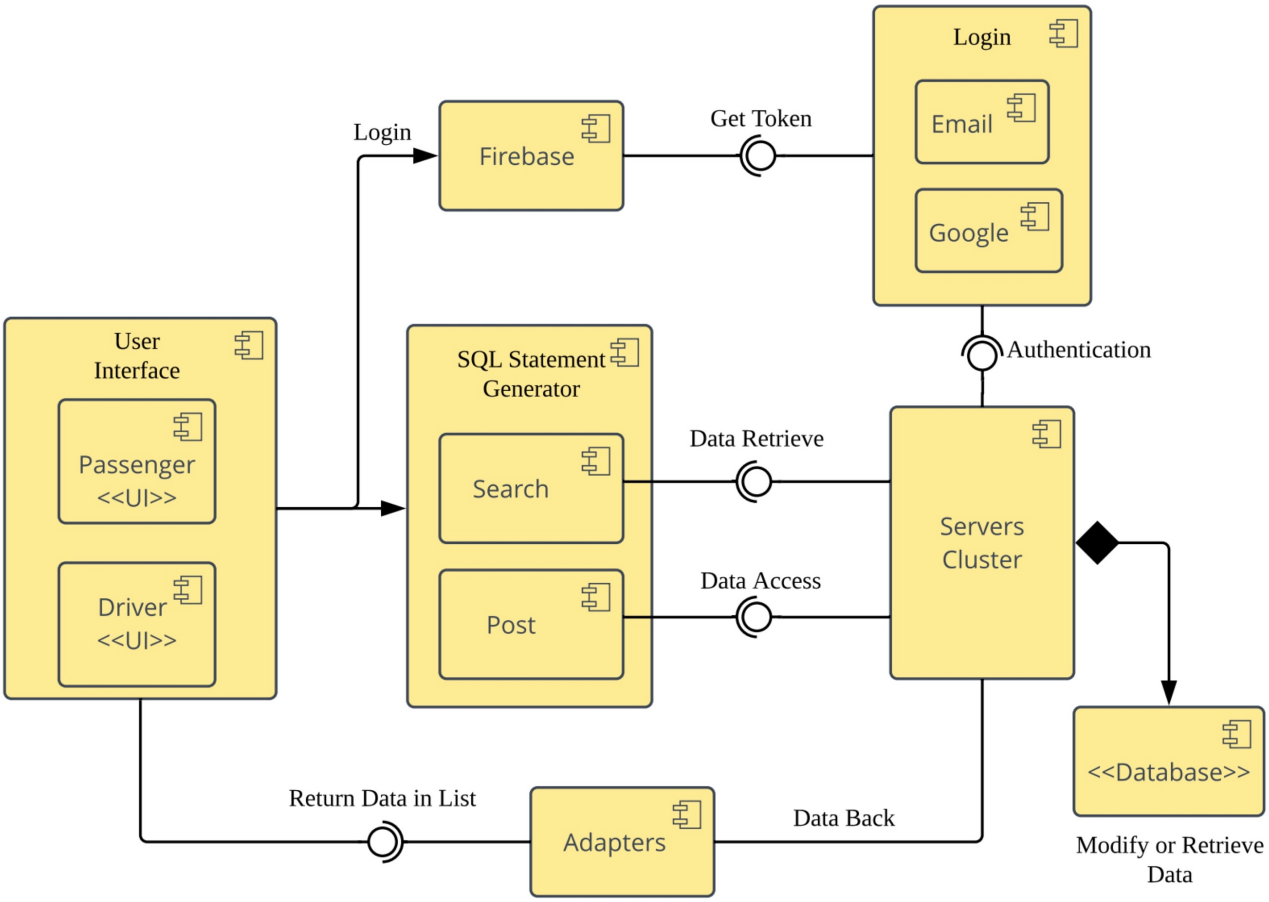

Figure 2. Component Diagram

Our system's component diagram is shown as above Figure 2. The User Interface, Firebase service, Login, SQL Statement Generator and Adapters run in Android environment. The Server Cluster includes Google login authentication server and our own server. The database runs on our own server.

Our target users are divided into two groups – passengers and drivers. The application uses the Firebase service for email login and Google login. When the user logs in by either way, the login component gives the user's basic information to the server, which would be used for the authentication. The Firebase then gets the token returning from the login activities, indicates that the user's identity has been verified.

Users interact with our database by searching available carpool offers and posting new offers. In this part, the application will generate the corresponding SQL statement based on the content entered by the user,

and then use the corresponding method of HTTP to establish connections with the server and modify the data in database.

The adapter is an interface that runs on the Android environment which connects back-end data to the front-end display and it is an important intermediate layer between data and the User Interface (UI). It shows the data in different forms according to the designer's needs.

2.3 Project Progress

At present, we have implemented the basic login activities and built the corresponding architecture for future extension of the application. The following images are the structure of our project and the front-end of current implementation.