MIDDLE EAST TECHNICAL UNIVERSITY
NORTHERN CYPRUS CAMPUS

Computer Engineering Program

CNG 495 FALL 2022-2023

TERM PROJECT Progress Report I

Team Members : Hamzeh Abu Ali 2419471, (Ahmed Jaber 2470490)

Project Name : Sharek

Contents

List of Figures

Chapter 1

Proposal Template

1.1 Project Brief

Our application aims to connect people around our village by providing an online taxi service platform. This application will allow users to order a taxi through a navigation system that shows available taxi drivers around the users. Users can register as taxi drivers or as normal customer users. Taxi drivers can be drivers with taxi cars, or students who own cars and are willing to provide a taxi service. The application will also provide the users the functionality of sharing a ride with other users. They can open a “share a ride” form, and other interested users can see that form and participate in the ride. Taxi drivers have the option of taking those requests or not. After ordering a ride, users will have the option of calling the taxi driver or contacting them WhatsApp.

We aim to help our fellow students and village residents have a better transportation experience by connecting them to available taxi drivers and students that are willing to provide a service. As transportation is a critical issue in students lives, we hope that this application can ease our lives through providing a platform to connect us all.

1.2 Cloud Delivery Models

Since the application will be a navigation based system, we will use a SAAS cloud delivery model.

1.3 Client Application Tools

Flutter framework will be used to write a mobile application that will be deployed on IOS and Android. Android Studio will be used as a development tool, and Dart as a programming language tool.

1.4 Cloud Service Provider

IBM Cloud Service provider will be used due to the ease of Flutter based applications deployment and its free membership use.

1.5 Project Work Distribution

We are planning to have an even work distribution for both the client and server applications. However, to facilitate a formal work plan, team member Hamzeh will be a lead on the Cloud Server application, and team member Ahmed will be a lead on the Client Mobile Application.

1.6 Software Used

1.6.1 Flutter

Flutter is a UI framework for creating native mobile apps. It allows developers to build a mobile app using a single codebase while helping quickly build iOS and Android apps. The Flutter framework is a UI toolkit that helps developers create high-performance, high-fidelity applications for iOS, Android, Linux, Mac, Windows, etc. At its core, it lets developers build an application UI using widgets that have different states and react to specific events. Flutter framework, powered by Dart (programming language), enables maintaining a single codebase for deployment across the platforms mentioned above. This translates to a smooth, responsive application UI across different screen types, aspect ratios, and orientations [1].

1.6.2 Android Studio

Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools. To support application development within the Android operating system, Android Studio uses a Gradle-based build system, emulator, code templates, and Github integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules. Android Studio uses an Instant Push feature to push code and resource changes to a running application. A code editor assists the developer with writing code and offering code completion, refraction, and analysis. Applications built in Android Studio are then compiled into the APK format for submission to the Google Play Store [2].

1.6.3 IBM Cloudant

A fully managed, distributed database optimized for heavy workloads and fast-growing web and mobile apps, IBM Cloudant is available as an IBM Cloud® service with a 99.99% SLA [3].

1.7 Application Diagrams

1.7.1 Activity Diagram

Below is an Activity Diagram that demonstrates the workflow, procedural logic, and business process of the application.

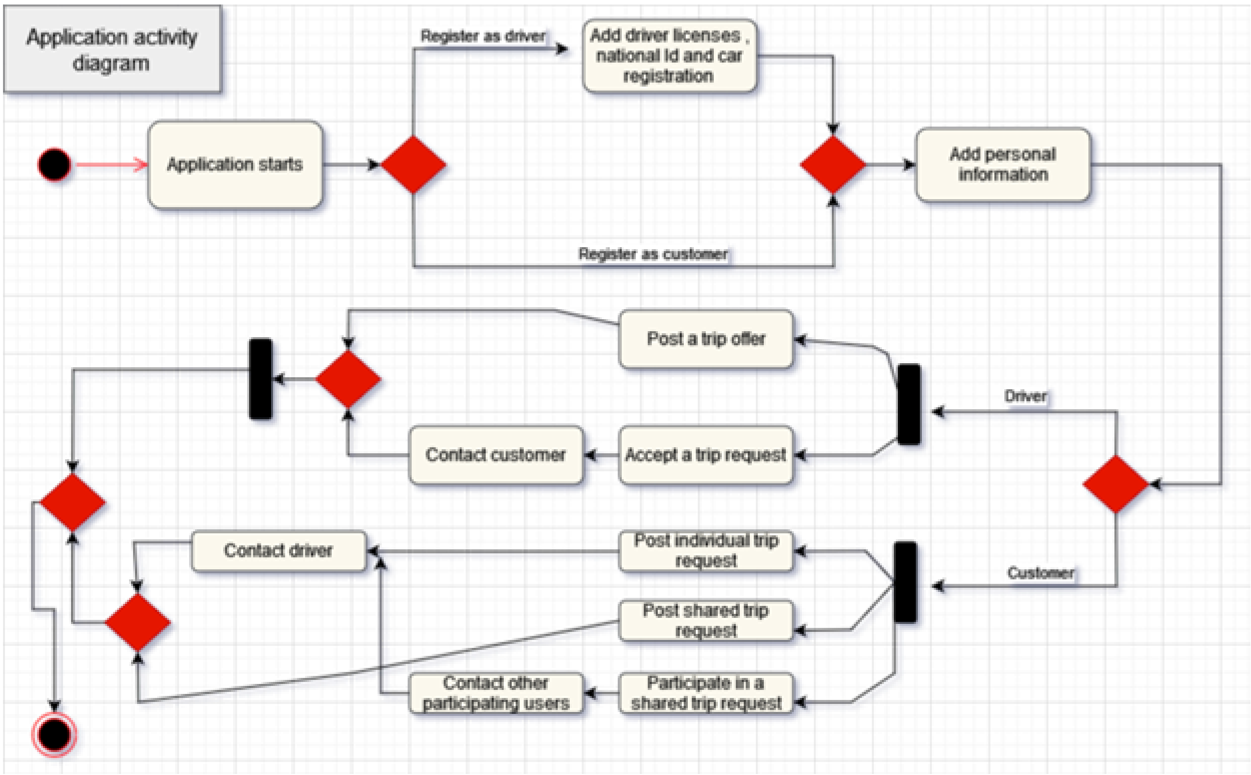


Figure 1.1: Activity Diagram

1.7.2 Sequence Diagrams

Below are some Sequence Diagram that shows three of the main operations done in the application, Registering, Ordering a Taxi, and Sharing a Ride.

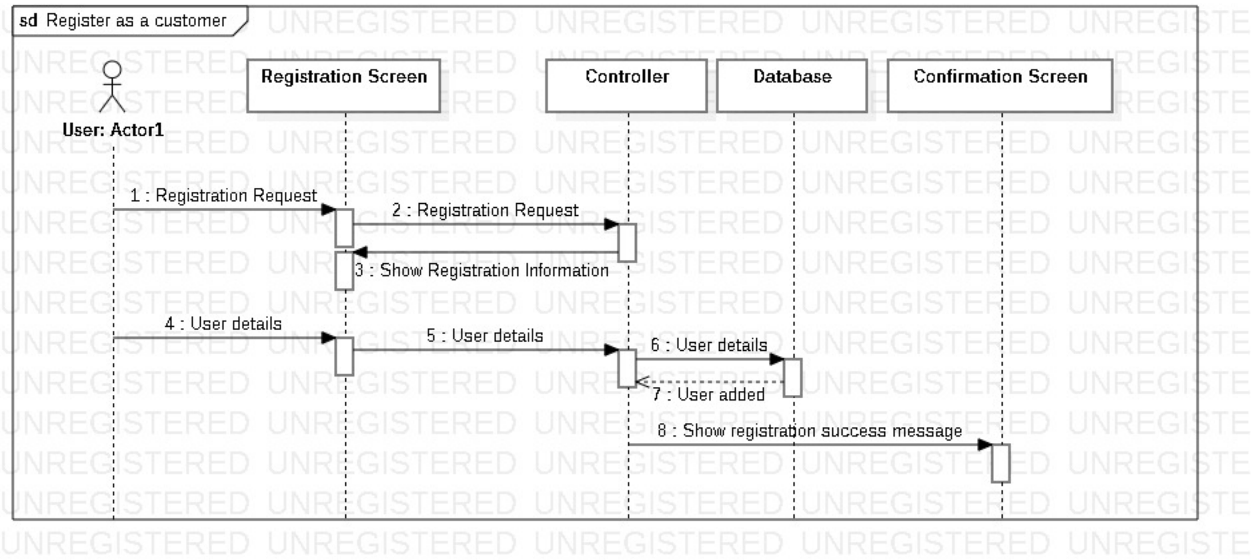


Figure 1.2: Registration Sequence Diagram

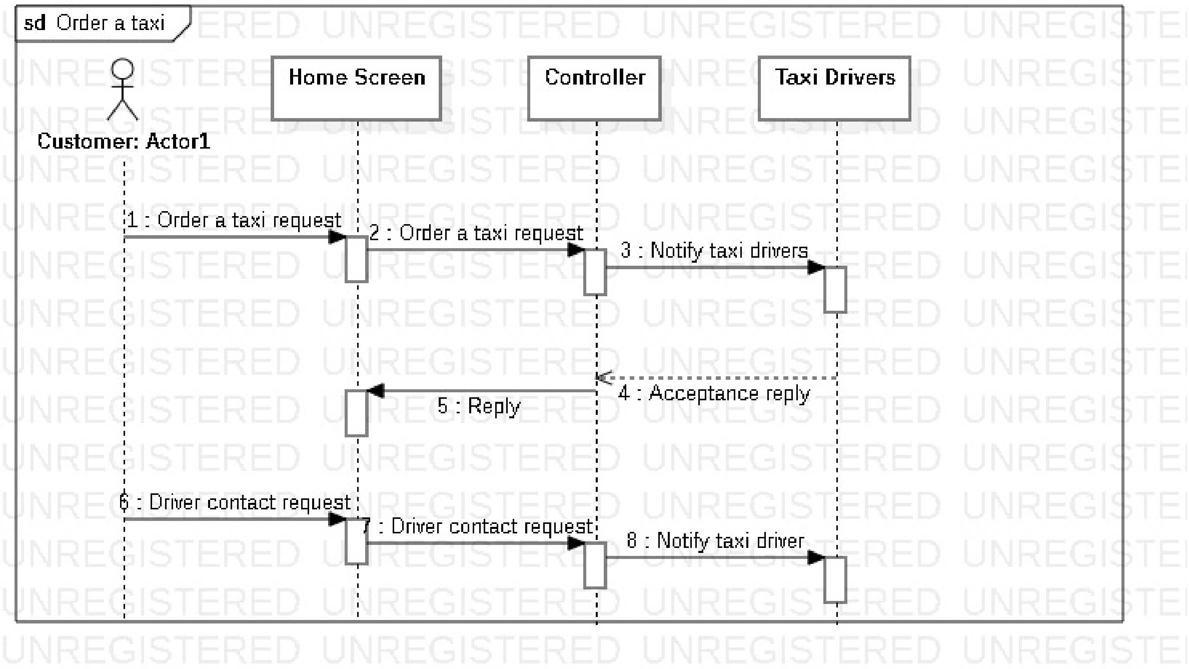


Figure 1.3: Order a Taxi Sequence Diagram

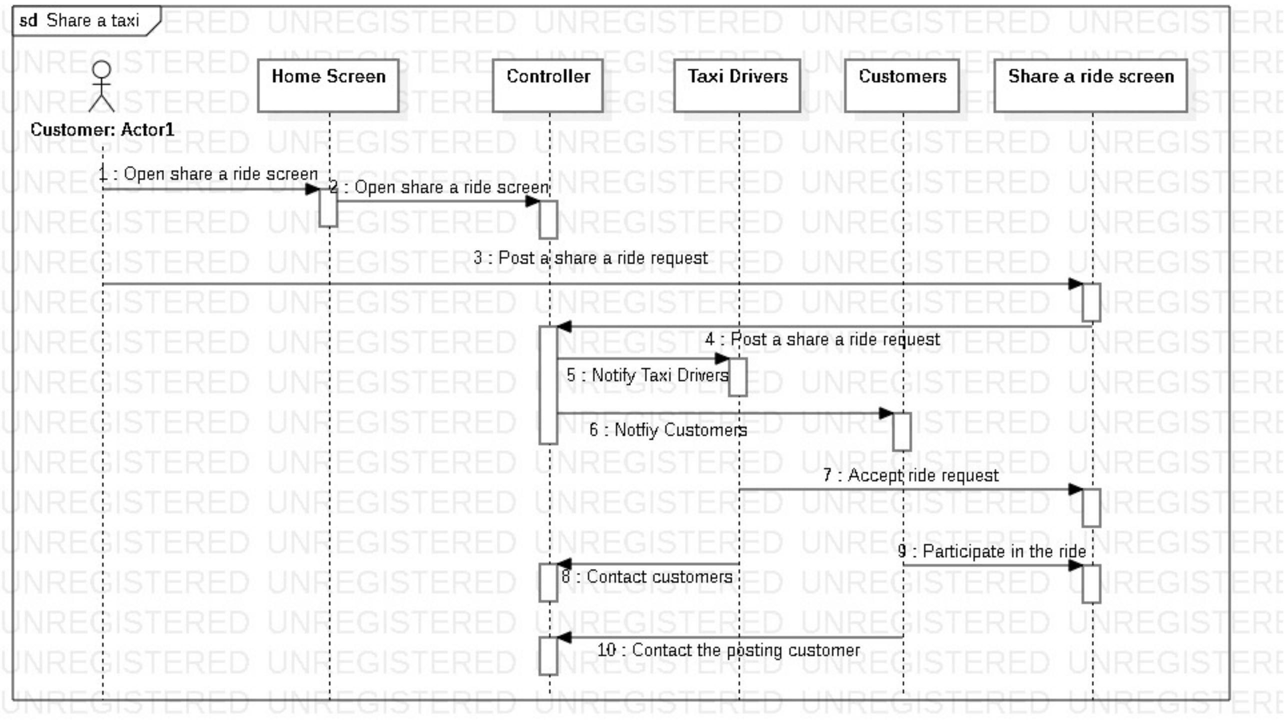


Figure 1.4: Share a Taxi Sequence Diagram

Chapter 2

Introduction

2.1 Project Proposal Cloud Service Modification

The project cloud database was determined to be IBM Cloudant in the project proposal period; however, as we saw that writing in Flutter using Dart had an eased integration with Fire Base [4] -A Google Cloud Service for Mobile Applications-, we decided to go on and deploy our application data there.

Chapter 3

Milestones achieved

3.1 Firebase

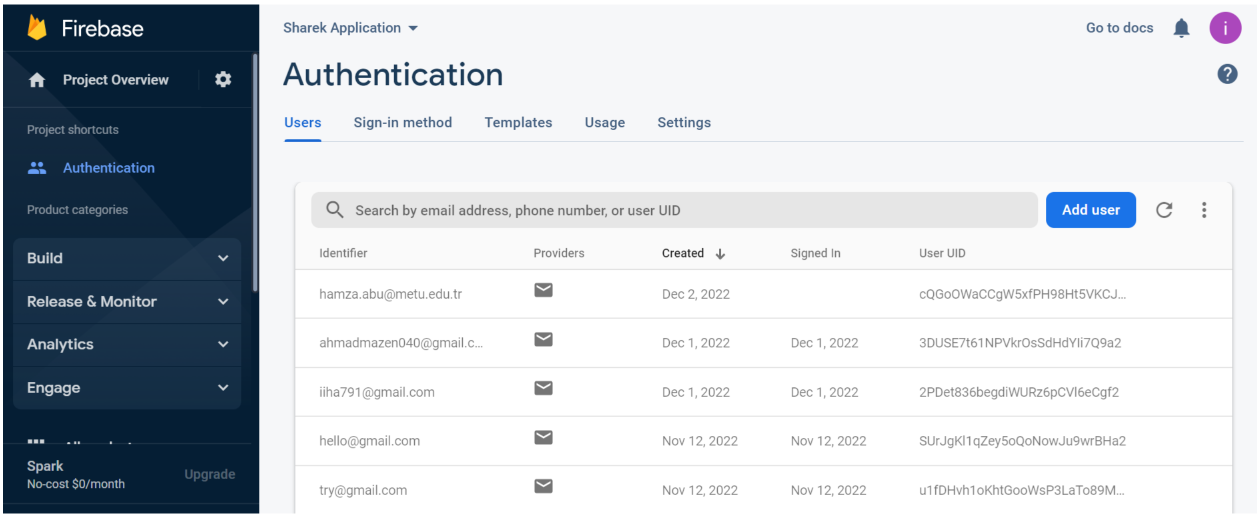
3.1.1 (October 31 - November 6)

Firebase is a backend platform for building Web, Android and IOS applications. It offers real-time database, different APIs, multiple authentication types and a hosting platform [6].

Firebase has the following features:

- It is a real-time database. Changes in the application code are reflected directly in the cloud database.
- Authentication is available on the parameter chosen, in our case, the user password.
- A dashboard is available for ease of use and configuration; see Figure 3.1.

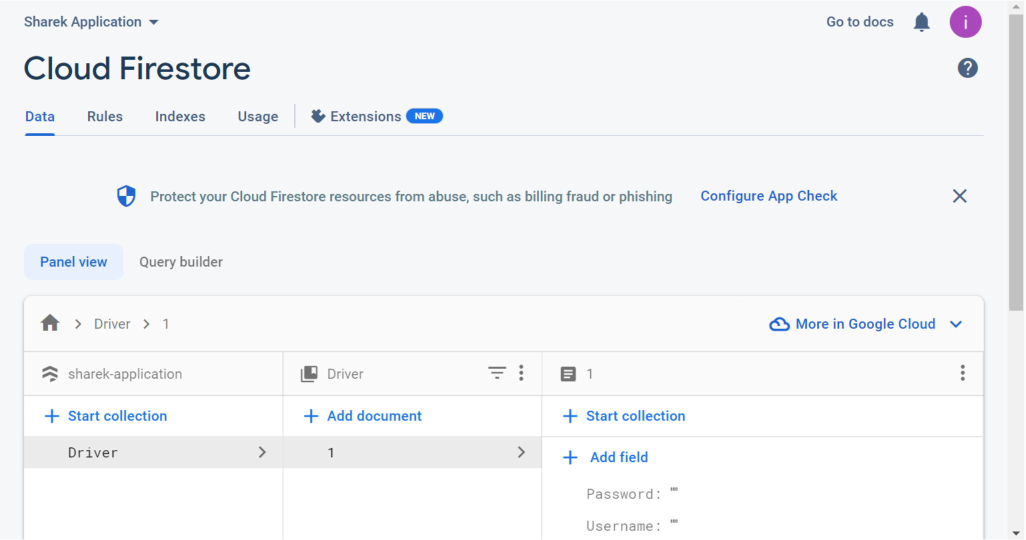
In Figure 3.1, the table of current users and their related information is displayed.



Identifier	Providers	Created ↓	Signed In	User UID
hamza.abu@metu.edu.tr	✉	Dec 2, 2022		cQGoOWaCCgW5xfPH98Ht5VKCJ...
ahmadmazen040@gmail.c...	✉	Dec 1, 2022	Dec 1, 2022	3DUSE7t61NPVkrOsSdHdYIi7Q9a2
iiha791@gmail.com	✉	Dec 1, 2022	Dec 1, 2022	2PDet836begdiWURz6pCVl6eCgf2
hello@gmail.com	✉	Nov 12, 2022	Nov 12, 2022	SUrJgKl1qZey5oQoNowJu9wrBHa2
try@gmail.com	✉	Nov 12, 2022	Nov 12, 2022	u1fDHvh1oKhtGooWsP3LaTo89M...

Figure 3.1: Firebase Dashboard

Using Firebase, we could also write our own queries using Firebase's NoSQL cloud database Firestore.



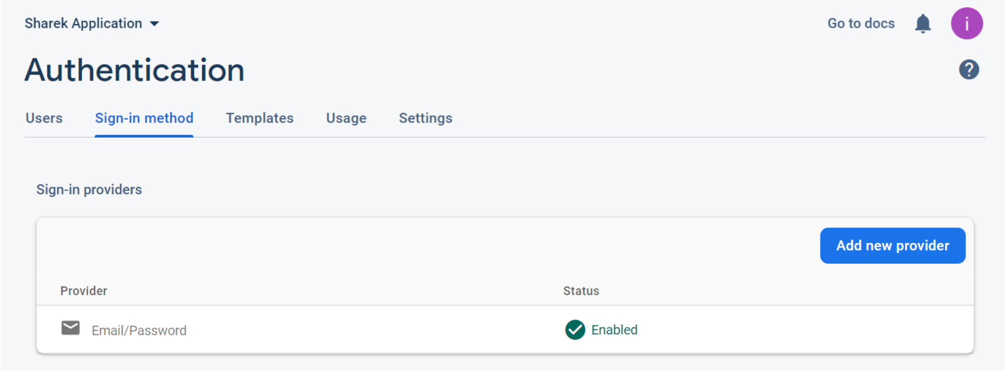
Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing

Password: ""

Username: ""

Figure 3.2: Firestore Database

In our application, authentication is done based on user e-mail address and password. This can be set in the authentication tab on the dashboard.



Provider	Status
Email/Password	✓ Enabled

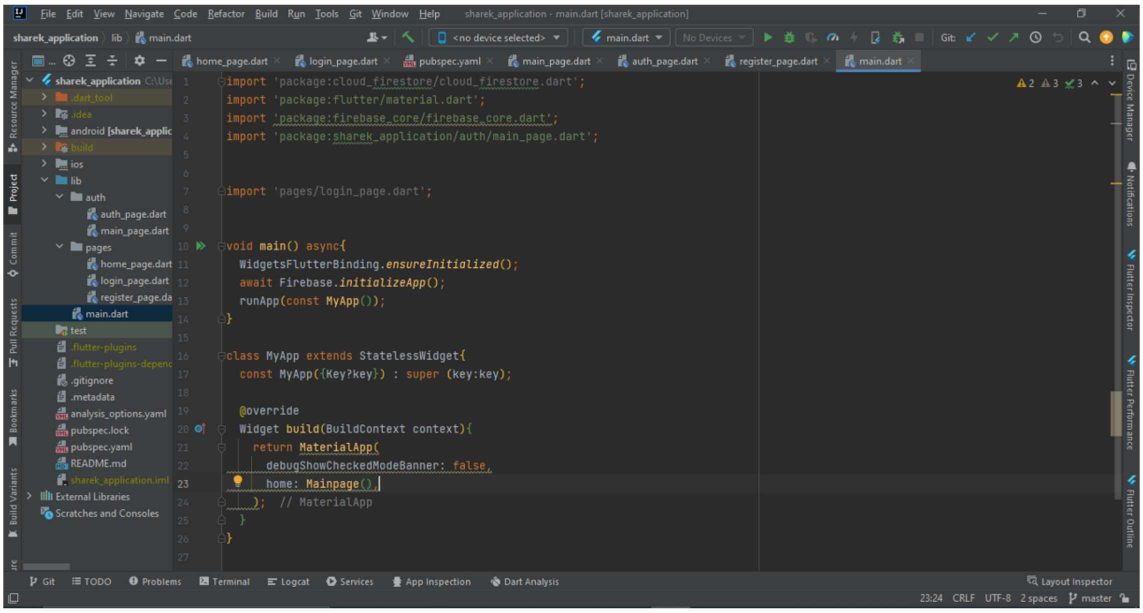
Figure 3.3: User Authentication

3.2 Application Authentication

3.2.1 (November 7 - November 13)

In the figures below, I demonstrate the initialization and authentication of the Firebase connection with our Application program. Intellij was used as an IDE, and the programming language was Dart. The figures also show the parts I assisted in in the user interfaces. The cloning URL for the GitHub repository is present in [5].

In Figure 3.4, necessary imports are called for establishing Firebase connection in lines 1-4. In line 12, Firebase initalizeApp method is invoked at the beginning of the program.

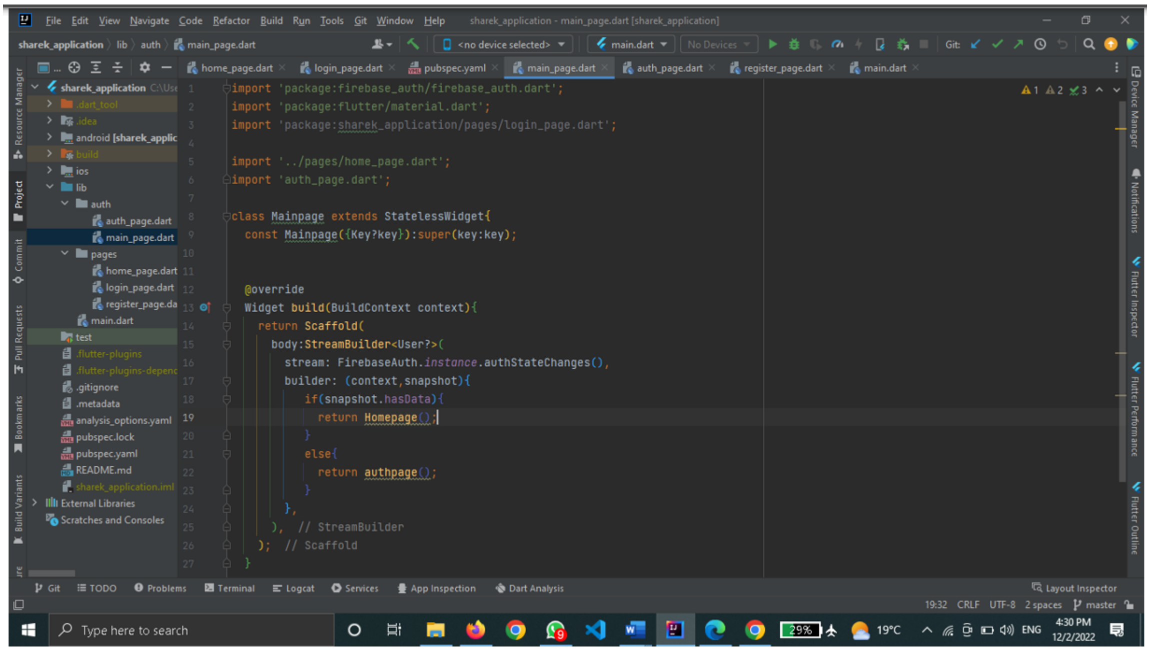


```
1  import 'package:cloud_firestore/cloud_firestore.dart';
2  import 'package:flutter/material.dart';
3  import 'package:firebase_core/firebase_core.dart';
4  import 'package:sharek_application/auth/main_page.dart';
5
6
7  import 'pages/login_page.dart';
8
9
10 void main() async{
11   WidgetsFlutterBinding.ensureInitialized();
12   await Firebase.initializeApp();
13   runApp(const MyApp());
14 }
15
16 class MyApp extends StatelessWidget{
17   const MyApp({Key?key}) : super (key:key);
18
19   @override
20   Widget build(BuildContext context){
21     return MaterialApp(
22       debugShowCheckedModeBanner: false,
23       home: Mainpage(),
24     ); // MaterialApp
25   }
26 }
27
```

Figure 3.4: Instantiating Firebase

3.2.2 (November 14 - November 20)

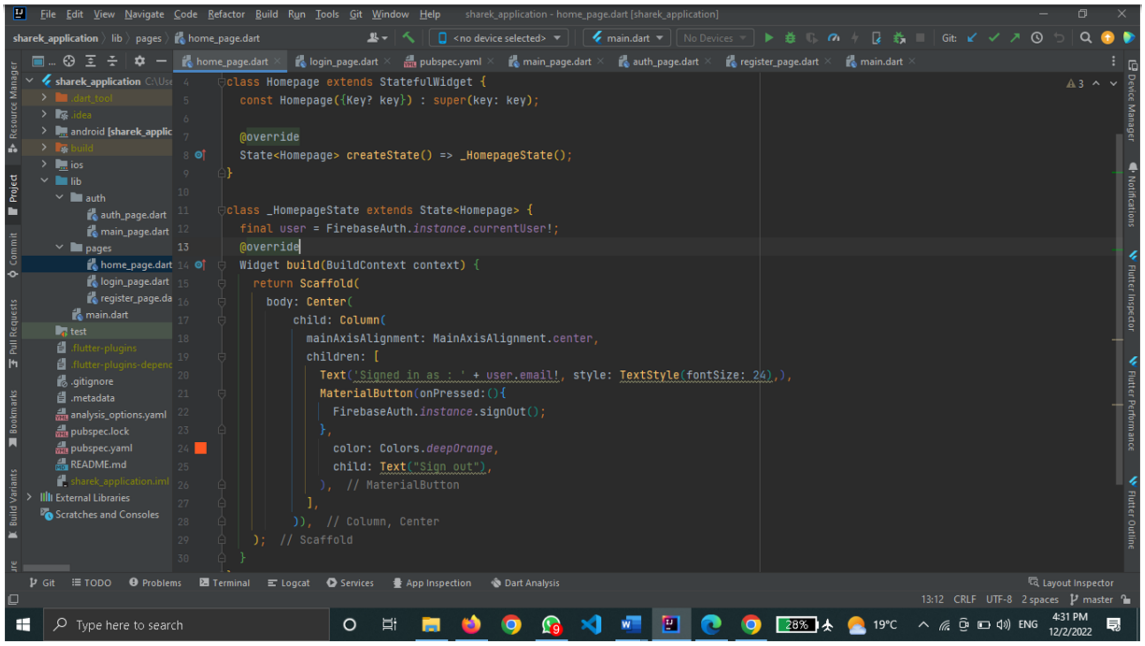
In Figure 3.5, line 16, we begin listening to any authentication requests from the client side by invoking the authStateChanges method.



```
1  import 'package:firebase_auth/firebase_auth.dart';
2  import 'package:flutter/material.dart';
3  import 'package:sharek_application/pages/login_page.dart';
4
5  import '../pages/home_page.dart';
6  import 'auth_page.dart';
7
8  class Mainpage extends StatelessWidget{
9    const Mainpage({Key?key}):super(key:key);
10
11
12   @override
13   Widget build(BuildContext context){
14     return Scaffold(
15       body:StreamBuilder<User?>(
16         stream: FirebaseAuth.instance.authStateChanges(),
17         builder: (context,snapshot){
18           if(snapshot.hasData){
19             return Homepage();
20           }
21           else{
22             return authpage();
23           }
24         },
25       ), // StreamBuilder
26     ); // Scaffold
27   }
```

Figure 3.5: Listening to the Stream for Authentication

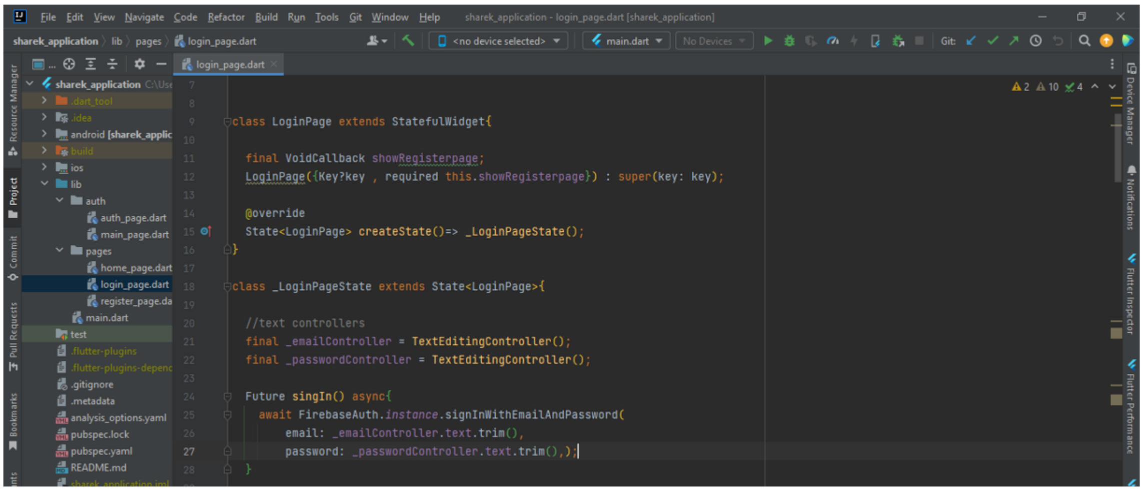
In Figure 3.6, line 12, we fetch the current user of the program from the cloud database for using their attributes in the client-side display.



```
4   class Homepage extends StatefulWidget {
5     const Homepage({Key? key}) : super(key: key);
6
7     @override
8     State<Homepage> createState() => _HomepageState();
9   }
10
11  class _HomepageState extends State<Homepage> {
12    final user = FirebaseAuth.instance.currentUser!;
13    @override
14    Widget build(BuildContext context) {
15      return Scaffold(
16        body: Center(
17            child: Column(
18              mainAxisAlignment: MainAxisAlignment.center,
19              children: [
20                Text('Signed in as : ' + user.email!, style: TextStyle(fontSize: 24),),
21                MaterialButton(onPressed:(){
22                  FirebaseAuth.instance.signOut();
23                },
24                  color: Colors.deepOrange,
25                  child: Text("Sign out"),
26                ), // MaterialButton
27              ],
28            )), // Column, Center
29      ); // Scaffold
30    }
```

Figure 3.6: Data Retrieval

In Figure 3.7, lines 26-27, we prepare the data entered by the user by trimming the excess parts and fitting it into the shape accepted; the controller then handles the entered data and authenticates it with the database. This is related to the login page.

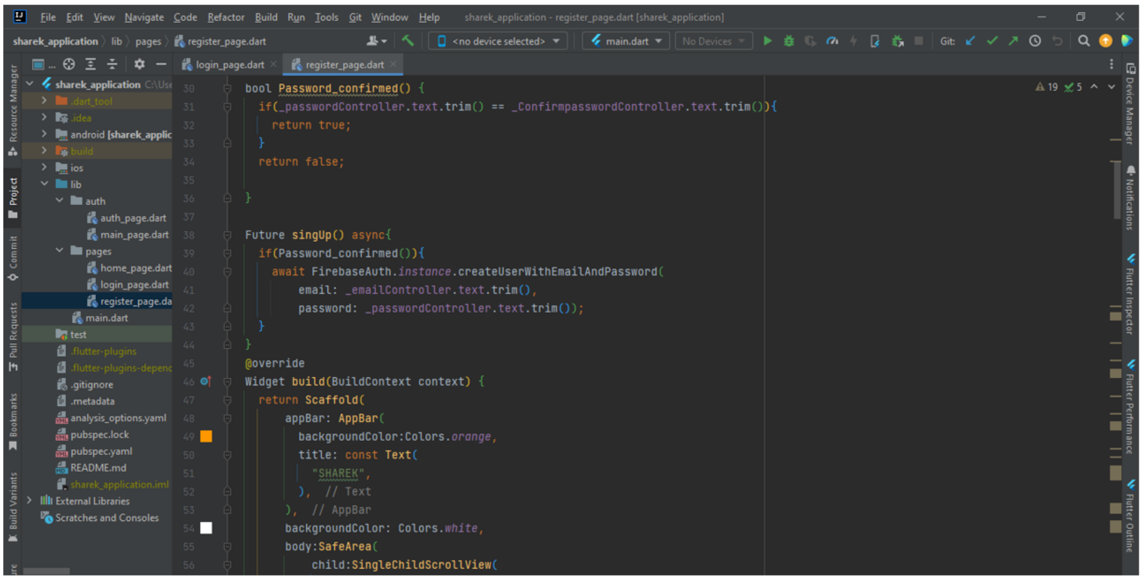


```
7
8
9   class LoginPage extends StatefulWidget{
10
11    final VoidCallback showRegisterpage;
12    LoginPage({Key?key , required this.showRegisterpage}) : super(key: key);
13
14    @override
15    State<LoginPage> createState()=> _LoginPageState();
16  }
17
18  class _LoginPageState extends State<LoginPage>{
19
20    //text controllers
21    final _emailController = TextEditingController();
22    final _passwordController = TextEditingController();
23
24    Future singIn() async{
25      await FirebaseAuth.instance.signInWithEmailAndPassword(
26          email: _emailController.text.trim(),
27          password: _passwordController.text.trim(),);
28    }
```

Figure 3.7: User Login Authentication

3.2.3 (November 21 - November 27)

In Figure 3.8, lines 26-27, we prepare the data entered by the user by trimming the excess parts and fitting it into the shape accepted; the controller then handles the entered data and authenticates it with the database. This is related to the registration page.



```
30  bool Password_confirmed() {
31    if(_passwordController.text.trim() == _ConfirmpasswordController.text.trim()){
32      return true;
33    }
34    return false;
35
36  }
37
38  Future singUp() async{
39    if(Password_confirmed()){
40      await FirebaseAuth.instance.createUserWithEmailAndPassword(
41          email: _emailController.text.trim(),
42          password: _passwordController.text.trim());
43    }
44  }
45  @override
46  Widget build(BuildContext context) {
47    return Scaffold(
48        appBar: AppBar(
49          backgroundColor:Colors.orange,
50          title: const Text(
51            "SHAREK",
52          ),  // Text
53        ),  // AppBar
54        backgroundColor: Colors.white,
55        body:SafeArea(
56            child:SingleChildScrollView(
```

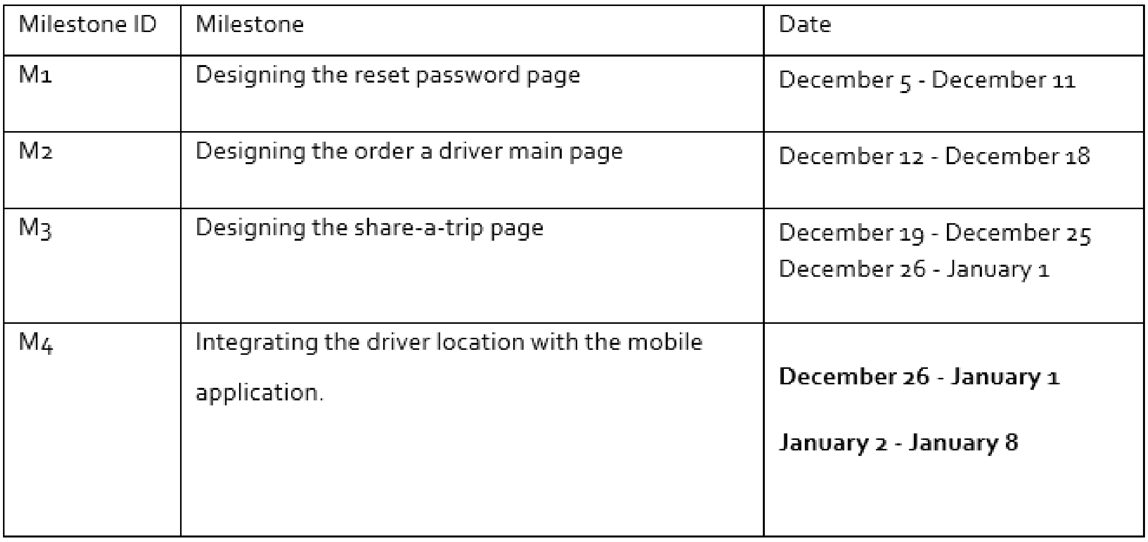
Figure 3.8: User Register Authentication

Chapter 4

Milestones remained

4.1 Milestone Table

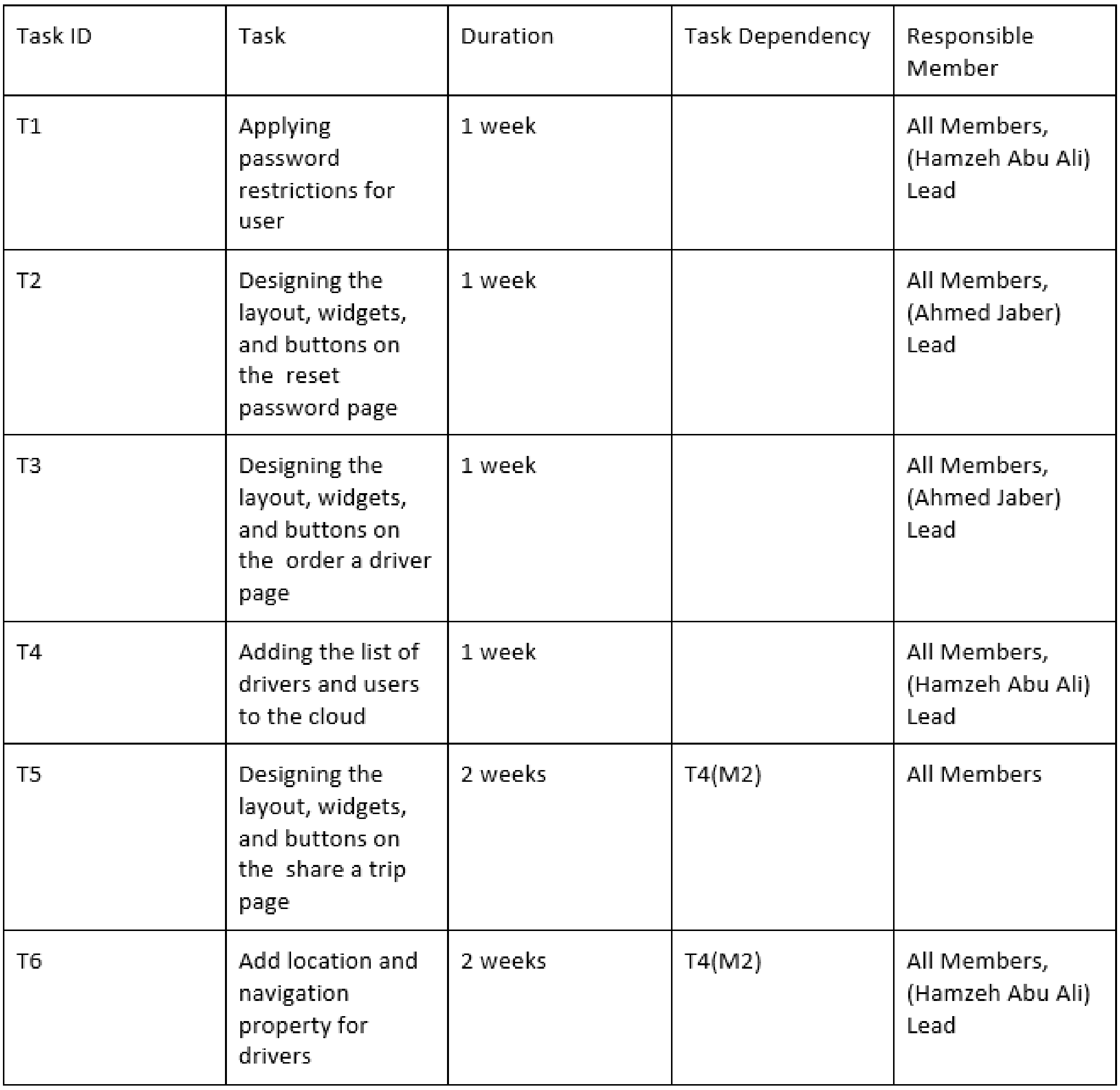
Figure 4.1 displays the milestones remaining for the team to complete the project, along with the dates expected for completeness. Figure 4.2 displays the tasks required to be completed in each milestone, along with the responsible members.



Milestone ID	Milestone	Date
M1	Designing the reset password page	December 5 - December 11
M2	Designing the order a driver main page	December 12 - December 18
M3	Designing the share-a-trip page	December 19 - December 25 December 26 - January 1
M4	Integrating the driver location with the mobile application.	**December 26 - January 1** **January 2 - January 8**

Figure 4.1: Milestone Table

4.2 Tasks Table



Task ID	Task	Duration	Task Dependency	Responsible Member
T1	Applying password restrictions for user	1 week		All Members, (Hamzeh Abu Ali) Lead
T2	Designing the layout, widgets, and buttons on the reset password page	1 week		All Members, (Ahmed Jaber) Lead
T3	Designing the layout, widgets, and buttons on the order a driver page	1 week		All Members, (Ahmed Jaber) Lead
T4	Adding the list of drivers and users to the cloud	1 week		All Members, (Hamzeh Abu Ali) Lead
T5	Designing the layout, widgets, and buttons on the share a trip page	2 weeks	T4(M2)	All Members
T6	Add location and navigation property for drivers	2 weeks	T4(M2)	All Members, (Hamzeh Abu Ali) Lead

Figure 4.2: Tasks Table

Chapter 5

References

1. https://www.accelq.com/blog/flutter-framework/
2. https://www.techtarget.com/searchmobilecomputing/definition/Android-Studio
3. https://www.ibm.com/cloud/cloudant
4. https://firebase.google.com/
5. https://github.com/Codingaway20/sharek_application.git
6. https://www.tutorialspoint.com/firebase/index.htm