Mobile Programing Project
CarPool ASU

Participants

Morcous Wael William **19P8211**

Submitted to

Dr. Haytham Azmi Mahmoud Elmiligi
Eng. Doaa Mohamed Mahmoud Ismail

Introduction

Carpool is a revolutionary rideshare application designed to streamline rideshare services for the Faculty of Engineering Community at Ain Shams University. Developed exclusively for students, Carpool aims to provide a reliable and efficient transportation solution for commuting to and from the AinShams campus, with a specific focus on routes to Abdu-Basha and Abbaseya square.

Key Features

Closed Community Sign-In: Users are required to sign in with an active account ending with @eng.asu.edu.eg to ensure a trusted and closed community environment.

Student-Operated Service: Carpool is operated by students, for students, fostering a community-driven approach to ridesharing.

Limited Destination Points and Ride Times: To regulate the service during this pilot project, Carpool offers rides to only two destination points, Gate 3 and 4, with a fixed start ride time at 7:30 am and a return ride time at 5:30 pm from the Faculty of Engineering campus.

Reservation System for Ride Times

Customers needing a ride at 7:30 am must reserve their seat before 10:00 pm the previous day.
Customers requiring a ride at 5:30 pm must reserve their seat before 1:00 pm on the same day.
Development Technologies
Choose either Native Android app development using Java or Kotlin or a cross-platform approach with Flutter and Android Studio.

App Description/Requirements

App Features

1. **Login Page with Firebase Authentication**

Sign-up option using Firebase Authenticate.
Including a test account with login information for testing.

2. **List of Available Routes:**

Display routes to and from Ainshams campus.
Implement the list using a list view builder.

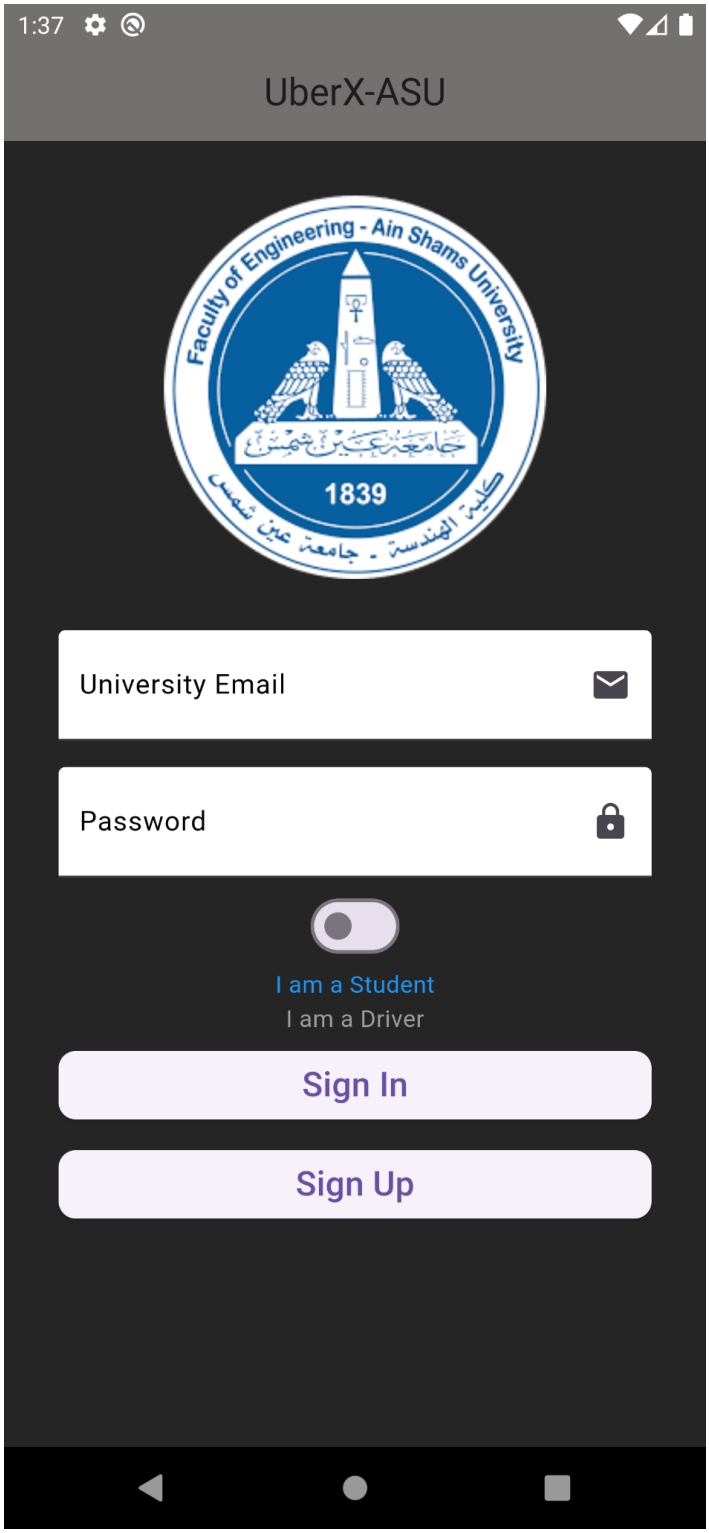
3. **Cart Page for Order Review and Payments.**
4. **Order History with Tracking/Status Page.**
5. **Database Integration:**

Use Firebase real-time database for routes and order status.
SQlite for displaying user profile page
Confirm orders and update status data.
Confirm orders before 11:30 pm for morning rides and before 4:30 pm for afternoon rides.

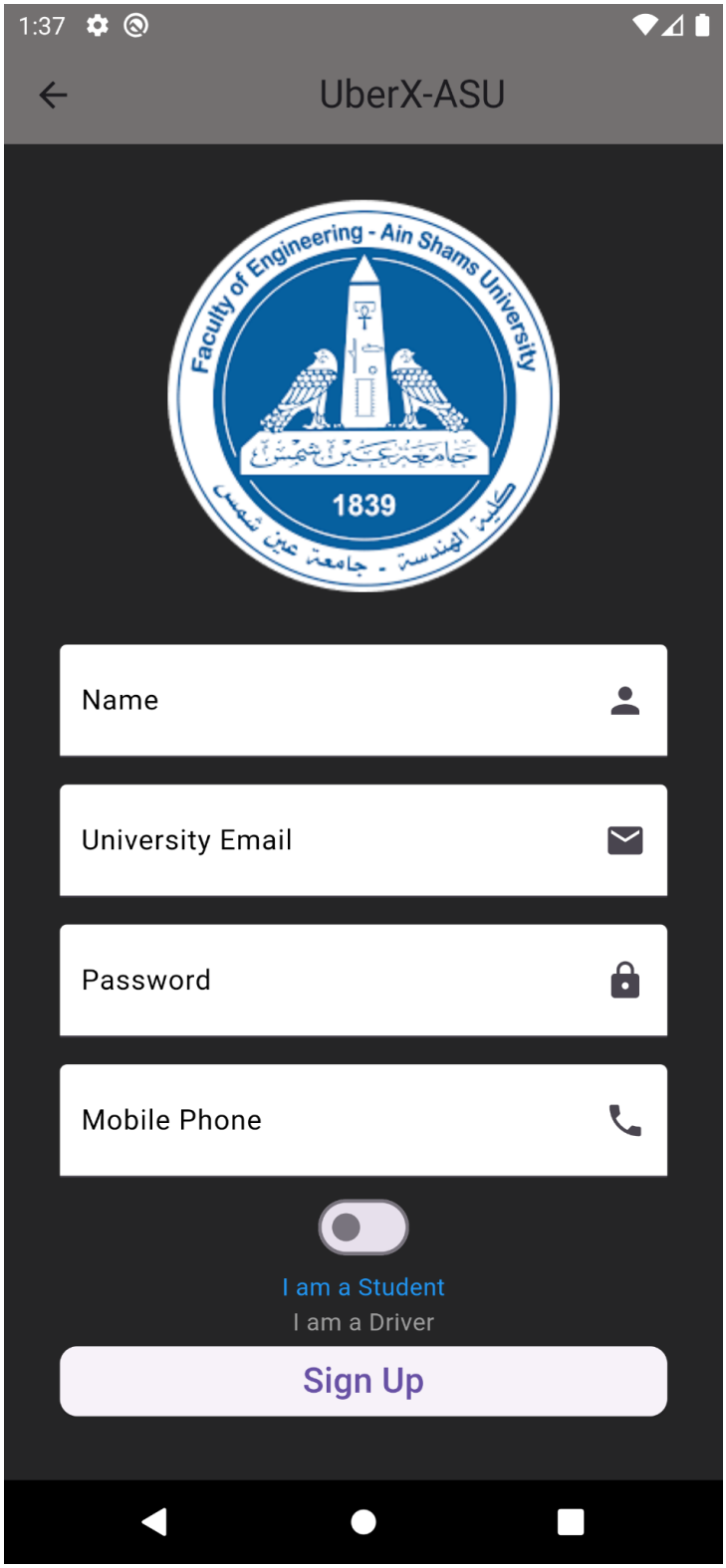
6. **Payment and Order Tracking Page.**
7. **Web Application for Drivers:**
   Includes
   Adding trips
   Showing driver trips
   And accepting / rejecting orders:

UI Design

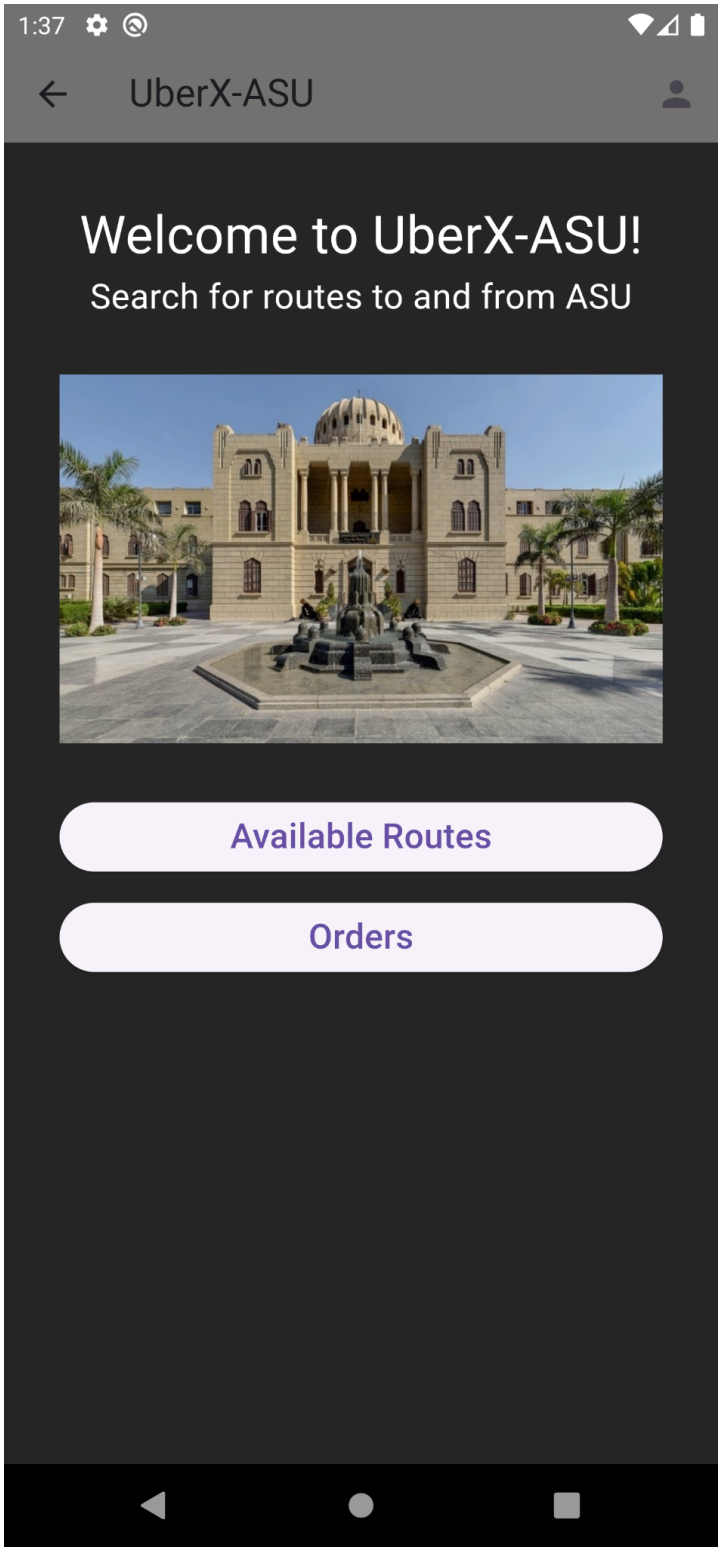
Sign In:



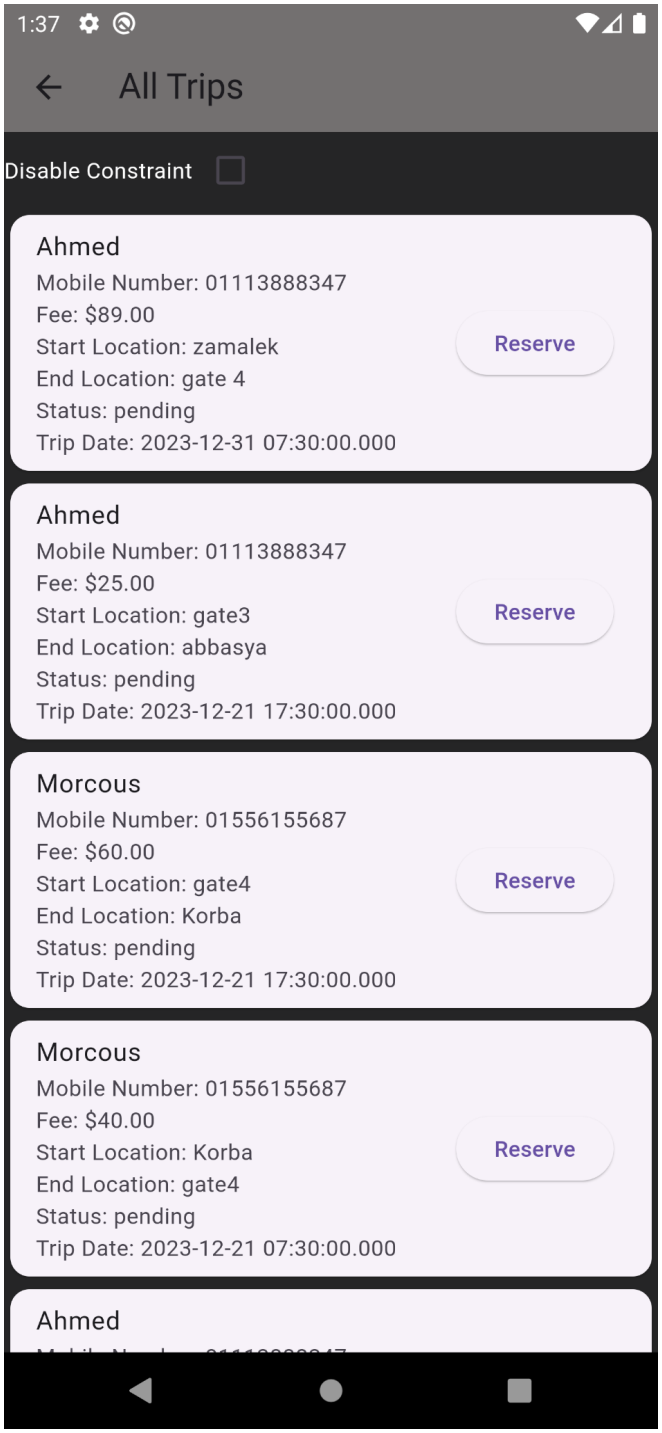
Sign Up:


1:37
UberX-ASU
Faculty of Engineering - Ain Shams University
1839
Name
University Email
Password
Mobile Phone
I am a Student
I am a Driver
Sign Up

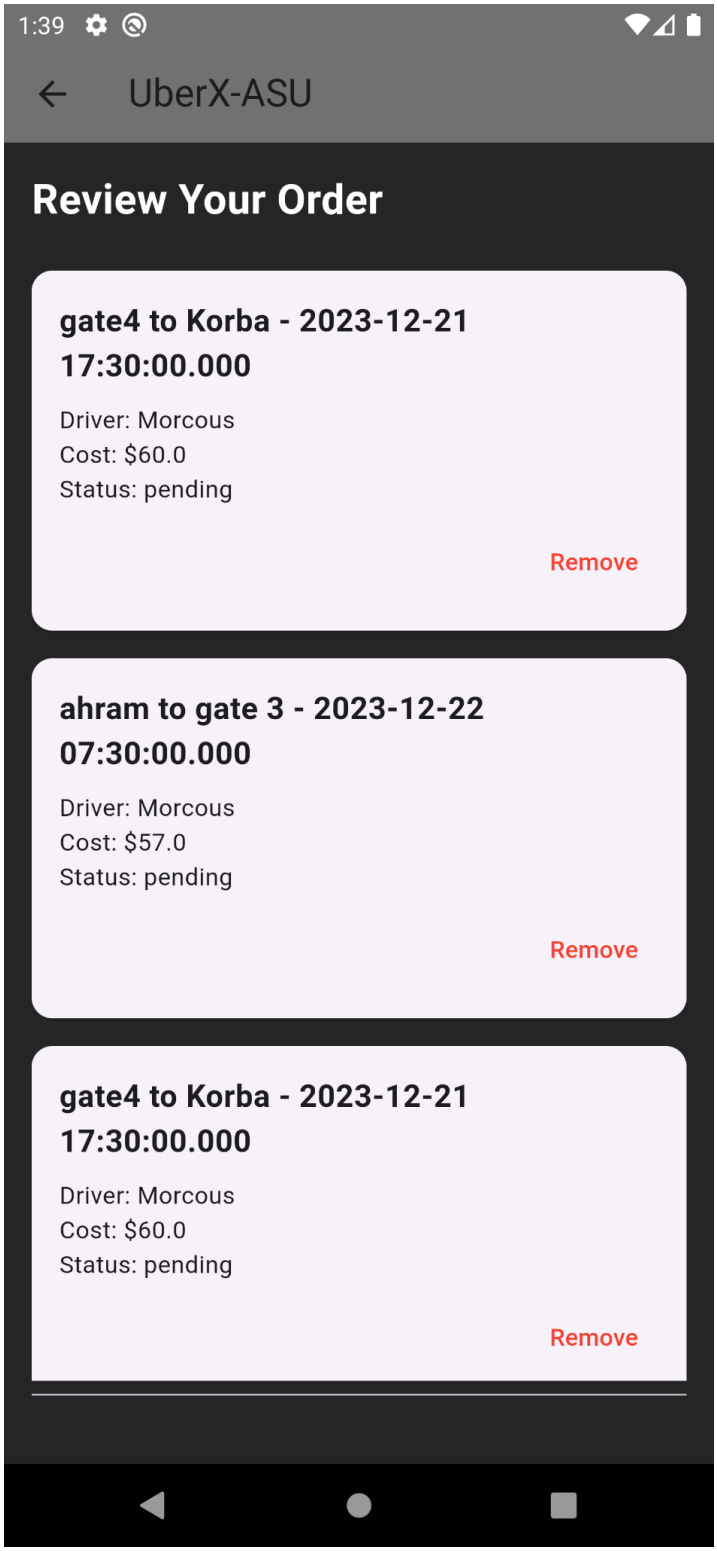
User Home:


1:37
UberX-ASU
Welcome to UberX-ASU!
Search for routes to and from ASU
Available Routes
Orders

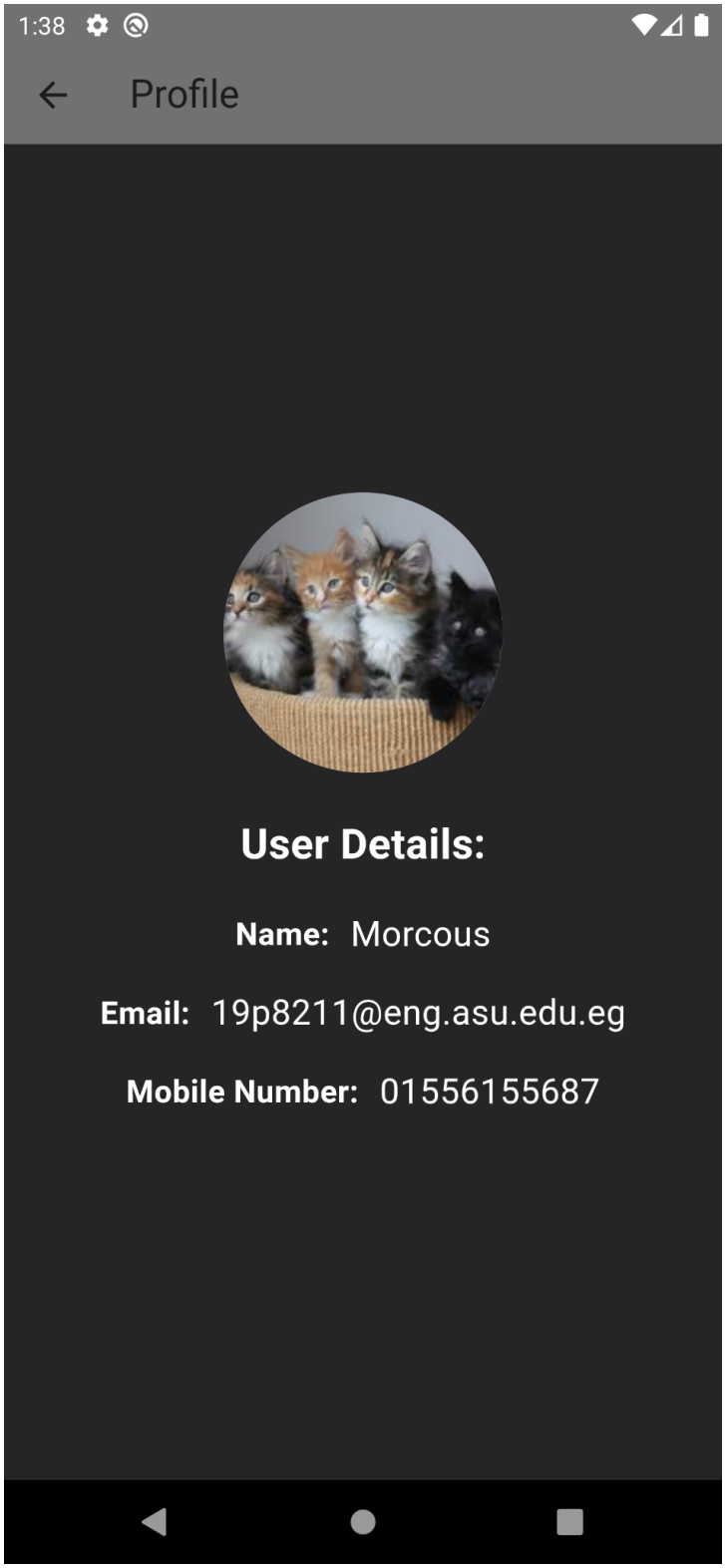
User Trips:


1:37
All Trips
Disable Constraint
Ahmed
Mobile Number: 01113888347
Fee: $89.00
Start Location: zamalek
End Location: gate 4
Status: pending
Trip Date: 2023-12-31 07:30:00.000
Reserve
Ahmed
Mobile Number: 01113888347
Fee: $25.00
Start Location: gate3
End Location: abbasya
Status: pending
Trip Date: 2023-12-21 17:30:00.000
Reserve
Morcous
Mobile Number: 01556155687
Fee: $60.00
Start Location: gate4
End Location: Korba
Status: pending
Trip Date: 2023-12-21 17:30:00.000
Reserve
Morcous
Mobile Number: 01556155687
Fee: $40.00
Start Location: Korba
End Location: gate4
Status: pending
Trip Date: 2023-12-21 07:30:00.000
Reserve
Ahmed

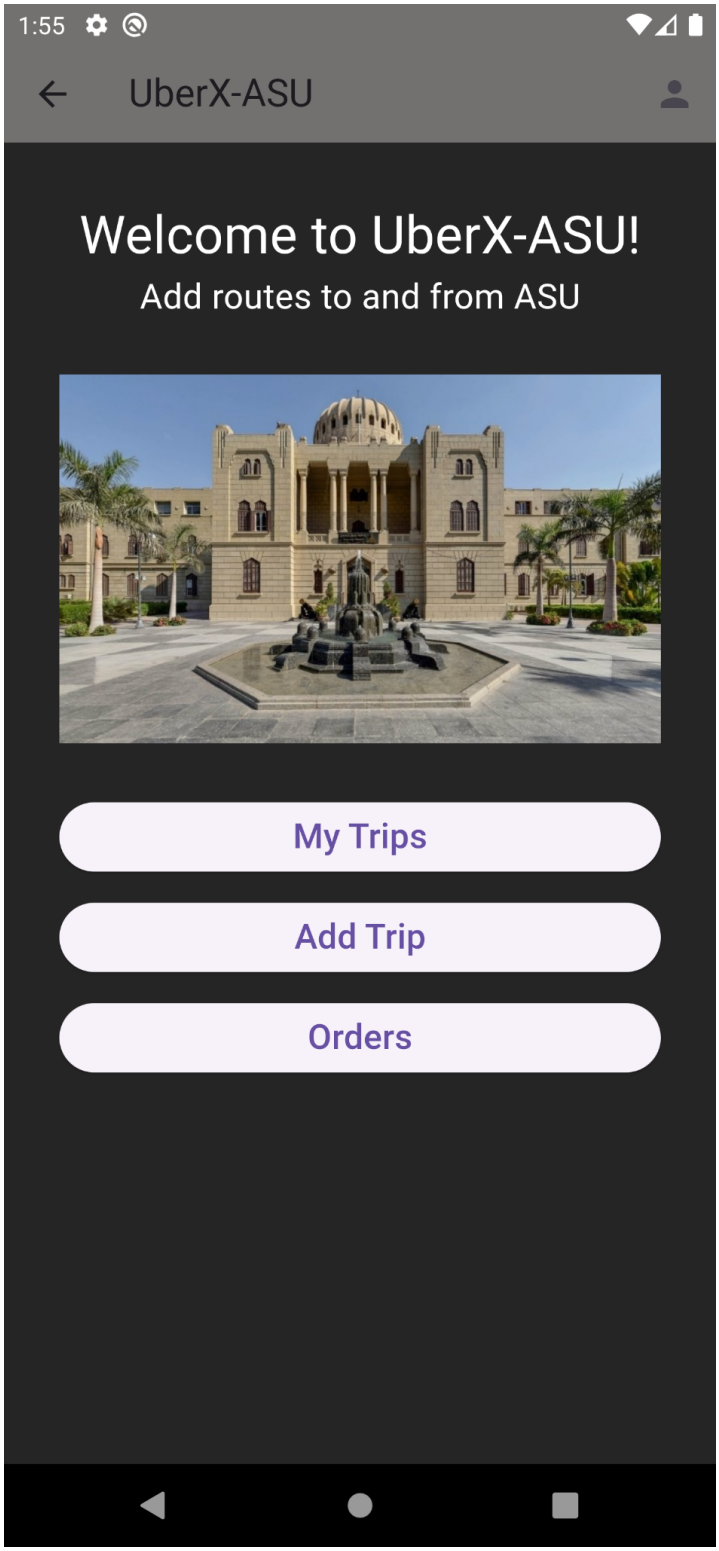
User Orders:


1:39
UberX-ASU
Review Your Order
gate4 to Korba - 2023-12-21 17:30:00.000
Driver: Morcous
Cost: $60.0
Status: pending
Remove
ahram to gate 3 - 2023-12-22 07:30:00.000
Driver: Morcous
Cost: $57.0
Status: pending
Remove
gate4 to Korba - 2023-12-21 17:30:00.000
Driver: Morcous
Cost: $60.0
Status: pending
Remove

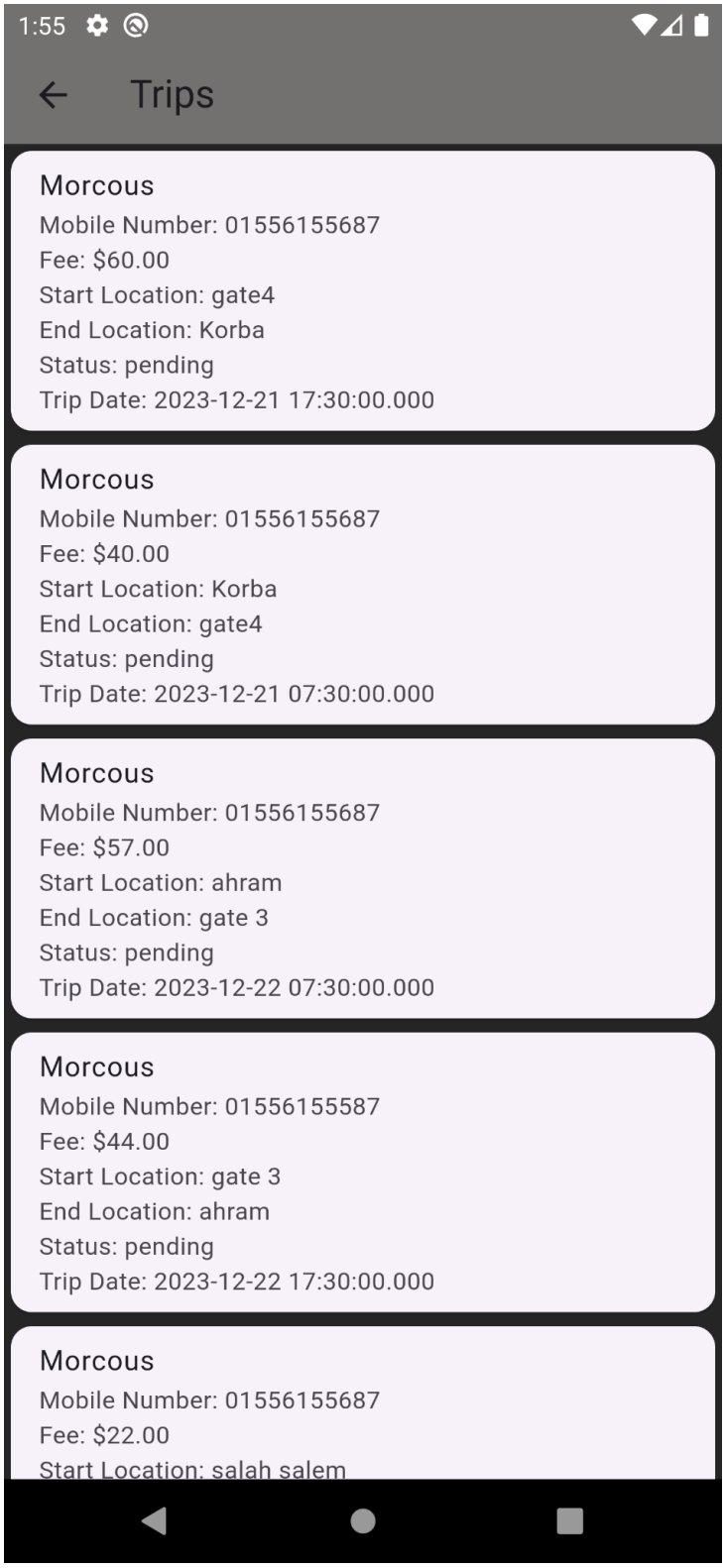
Profile:


1:38
Profile
User Details:
Name: Morcous
Email: 19p8211@eng.asu.edu.eg
Mobile Number: 01556155687

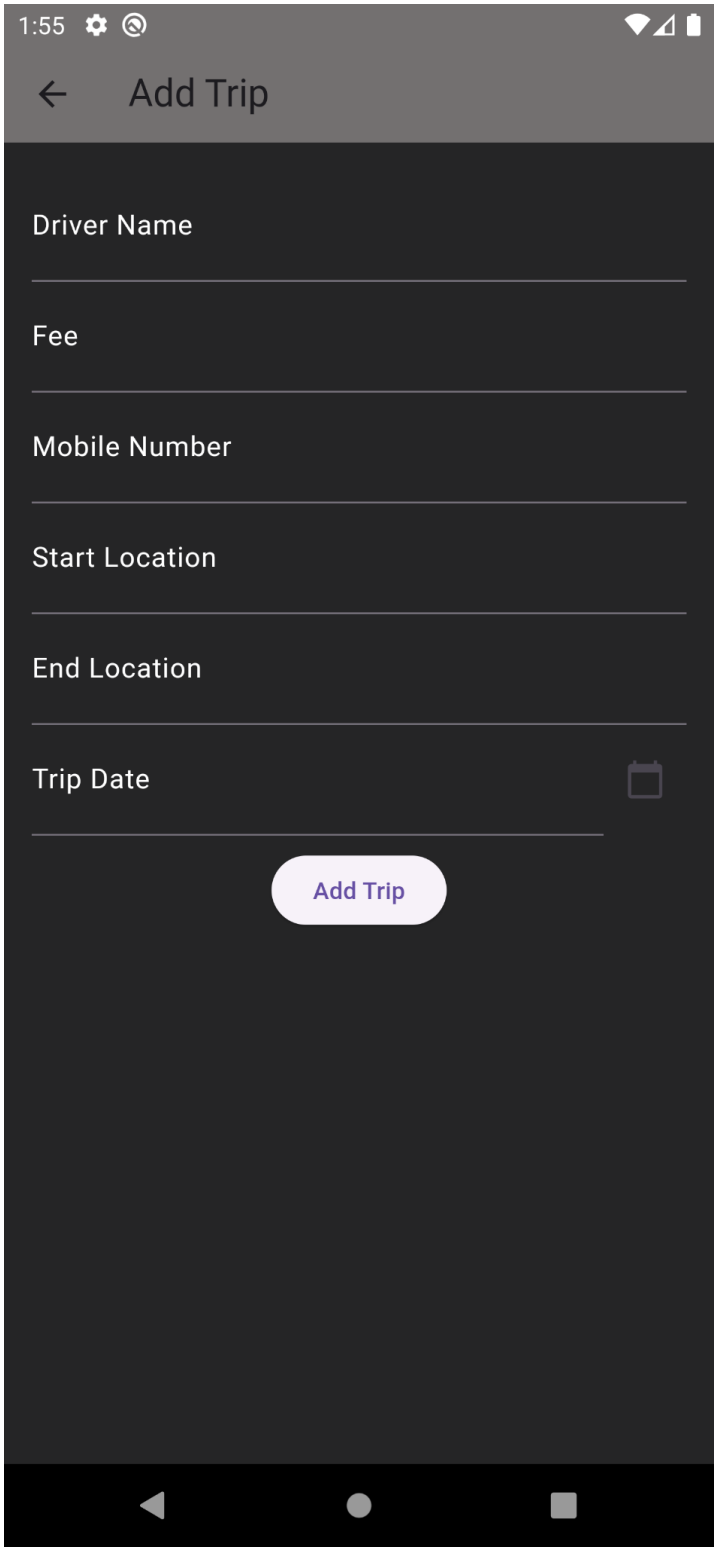
Driver_home:


1:55
UberX-ASU
Welcome to UberX-ASU!
Add routes to and from ASU
My Trips
Add Trip
Orders

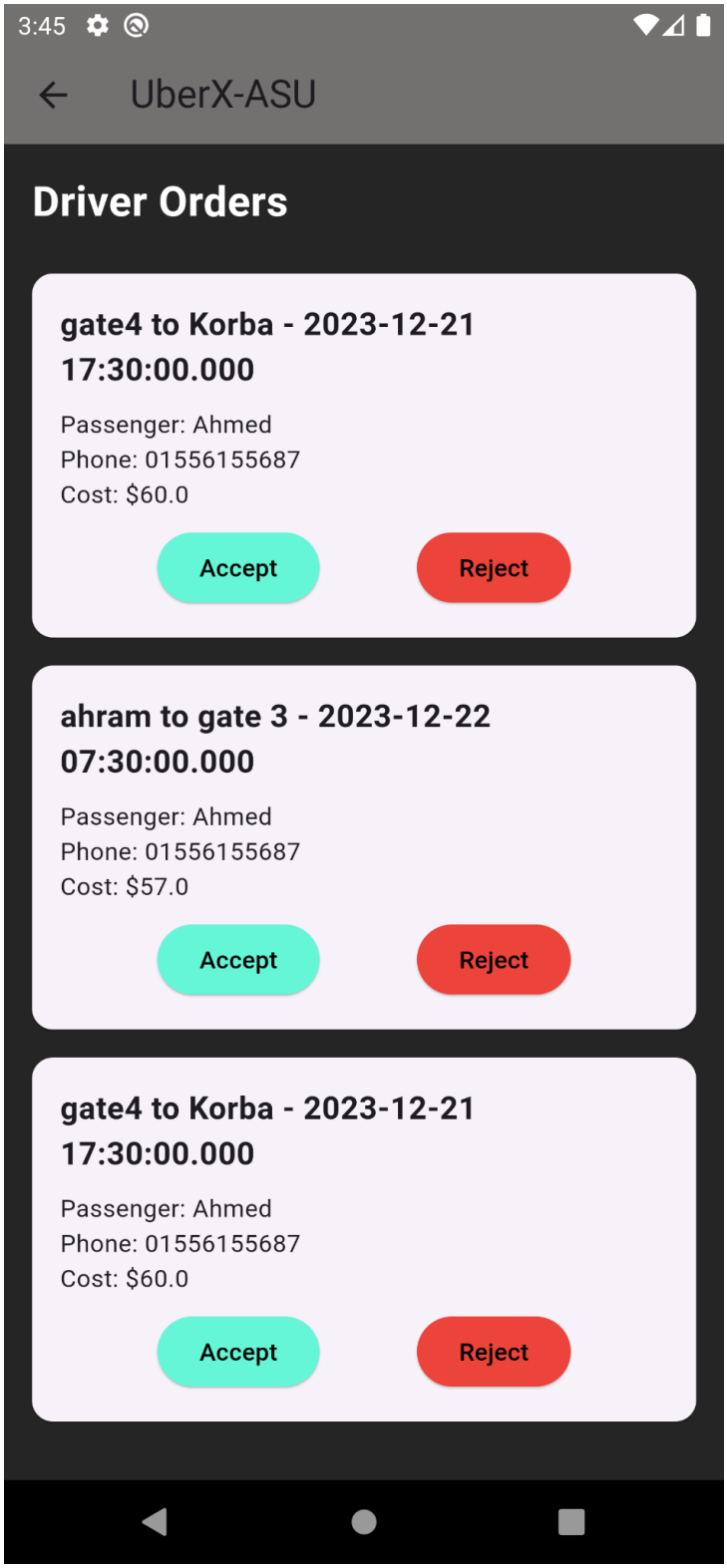
Driver Trips:



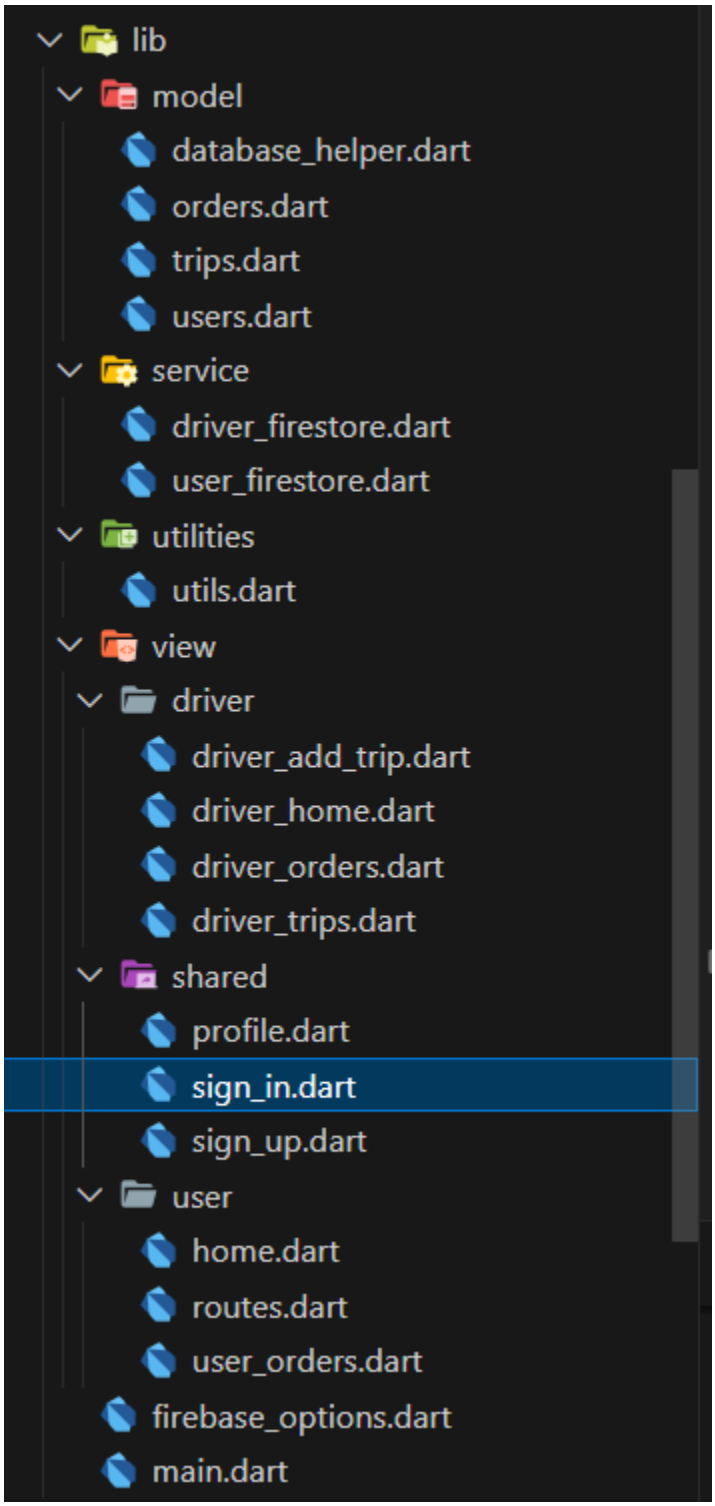
Driver Add Trip:



Driver Orders:



Structure:


lib
model
database_helper.dart
orders.dart
trips.dart
users.dart
service
driver_firestore.dart
user_firestore.dart
utilities
utils.dart
view
driver
driver_add_trip.dart
driver_home.dart
driver_orders.dart
driver_trips.dart
shared
profile.dart
sign_in.dart
sign_up.dart
user
home.dart
routes.dart
user_orders.dart
firebase_options.dart
main.dart

Code:

Database:

Model:

Dbb helper:

```
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';
import 'package:path_provider/path_provider.dart';

class DatabaseHelper {
  static const _databaseName = "MyDatabase.db";
  static const _databaseVersion = 1;

  static const table = 'my_table';

  static const columnName = 'name';
  static const columnEmail = 'email'; // New field: email
  static const columnPhone = 'phone'; // New field: mobile number
  static const columnPassword = 'password'; // New field: password

  late Database _db;

  // this opens the database (and creates it if it doesn't exist)
  Future<void> init() async {
    try {
      final documentsDirectory = await getApplicationDocumentsDirectory();
      final path = join(documentsDirectory.path, _databaseName);
      _db = await openDatabase(
        path,
        version: _databaseVersion,
        onCreate: _onCreate,
      );

      // Check if the table already exists
      if (!await isTableExists(_db, table)) {
        await _onCreate(_db, _databaseVersion);
      }
    } catch (e) {
      print('Error initializing database: $e');
```

```
    }
  }

  // Function to check if a table exists
  Future<bool> isTableExists(Database db, String table) async {
    var res = await db.rawQuery(
        "SELECT name FROM sqlite_master WHERE type='table' AND
name='$table'");
    return res.isNotEmpty;
  }

  // SQL code to create the database table
  Future _onCreate(Database db, int version) async {
    await db.execute('''
          CREATE TABLE $table (
            $columnName TEXT NOT NULL,
            $columnEmail TEXT NOT NULL,  -- New field
            $columnPhone TEXT NOT NULL,  -- New field
            $columnPassword TEXT NOT NULL  -- New field
          )
          ''');
  }

  // Helper methods

  // Inserts a row in the database where each key in the Map is a column
name
  // and the value is the column value. The return value is the id of the
  // inserted row.
  Future<int> insert(Map<String, dynamic> row) async {
    return await _db.insert(table, row);
  }

  // All of the rows are returned as a list of maps, where each map is
  // a key-value list of columns.
  Future<List<Map<String, dynamic>>> queryAllRows() async {
    return await _db.query(table);
  }
```

```
  // All of the methods (insert, query, update, delete) can also be done
using
  // raw SQL commands. This method uses a raw query to give the row count.
  Future<int> queryRowCount() async {
    final results = await _db.rawQuery('SELECT COUNT(*) FROM $table');
    return Sqflite.firstIntValue(results) ?? 0;
  }
}
```

Order.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter_milestone_1/model/trips.dart';

enum OrderStatus { pending, accepted, rejected, none }

class Orders {
  String? orderId;
  String? userId;
  final String userName;
  final String phoneNo;
  final double totalAmount;
  final Trip trip;
  OrderStatus statusOrder;

  Orders({
    required this.trip,
    this.orderId,
    this.userId,
    required this.statusOrder,
    required this.userName,
    required this.phoneNo,
    required this.totalAmount,
  });
  Map<String, dynamic> orderToJson() {
    return {
      'orderId': orderId,
      'userId': userId,
      'userName': userName,
      'phoneNo': phoneNo,
      'totalAmount': totalAmount,
```

```
      'trip': trip.tripToJson(),
      'statusOrder': statusOrder.toString().split('.').last,
    };
  }

  factory Orders.fromSnapshot(DocumentSnapshot<Map<String, dynamic>>
document) {
    if (document.data() != null) {
      final data = document.data()!;
      return Orders(
        orderId: document.id,
        userId: data['userId'],
        userName: data['userName'],
        phoneNo: data['phoneNo'],
        trip: Trip.fromJson(data['trip']),
        statusOrder: data['statusOrder'] == 'pending'
            ? OrderStatus.pending
            : data['statusOrder'] == 'accepted'
                ? OrderStatus.accepted
                : data['statusOrder'] == 'rejected'
                    ? OrderStatus.rejected
                    : OrderStatus.none,
        totalAmount: data['totalAmount'],
      );
    }
    return Orders(
      orderId: document.id,
      userId: '',
      userName: '',
      phoneNo: '',
      trip: Trip(
        driverId: '',
        driverName: '',
        fee: 0,
        mobileNumber: '',
        startLocation: '',
        endLocation: '',
        tripDate: DateTime.now(),
        statusTrip: TripStatus.pending,
      ),
```

```
      statusOrder: OrderStatus.none,
      totalAmount: 0,
    );
  }
}
```

Trip.dart

```
enum TripStatus {
  pending,
  ongoing,
  completed,
}

class Trip {
  final String? driverId;
  final String driverName;

  final String mobileNumber;
  final String startLocation;
  final String endLocation;
  final DateTime tripDate;
  final double fee;
  TripStatus statusTrip = TripStatus.pending;

  Trip({
    this.driverId,
    required this.driverName,
    required this.fee,
    required this.mobileNumber,
    required this.startLocation,
    required this.endLocation,
    required this.tripDate,
    required this.statusTrip,
  });

  Map<String, dynamic> tripToJson() {
    return {
      'driverId': driverId,
      'driverName': driverName,
      'fee': fee,
```

```
      'mobileNumber': mobileNumber,
      'startLocation': startLocation,
      'endLocation': endLocation,
      'tripDate': tripDate.toString(),
      'statusTrip': statusTrip.toString().split('.').last,
    };
  }

  factory Trip.fromJson(Map<String, dynamic> json) => Trip(
        driverId: json['driverId'] ?? '',
        driverName: json['driverName'] ?? '',
        fee: json['fee'] ?? '',
        startLocation: json['startLocation'] ?? '',
        endLocation: json['endLocation'] ?? '',
        tripDate: DateTime.parse(json['tripDate'] ?? ''),
        mobileNumber: json['mobileNumber'] ?? '',
        statusTrip: json['statusTrip'] == 'pending'
            ? TripStatus.pending
            : json['statusTrip'] == 'ongoing'
                ? TripStatus.ongoing
                : TripStatus.completed,
      );
}
```

Users.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';

class UserandDriver {
  final String name;
  final String email;
  final String password;
  final String mobileNumber;

  UserandDriver(
      {required this.name,
      required this.email,
      required this.password,
      required this.mobileNumber});
```

```
  factory UserandDriver.fromSnapshot(DocumentSnapshot snapshot) {
    Map<String, dynamic> data = snapshot.data() as Map<String, dynamic>;
    return UserandDriver(
      name: data['name'],
      email: data['email'],
      password: data['password'],
      mobileNumber: data['mobileNumber'],
    );
  }

  Map<String, dynamic> objectToJson() {
    return {
      'name': name,
      'email': email,
      'password': password,
      'mobileNumber': mobileNumber,
    };
  }
}
```

Services

User_servicefirestore:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter_milestone_1/model/orders.dart';
import 'package:flutter_milestone_1/model/users.dart';
import 'package:flutter_milestone_1/model/trips.dart';
import 'package:flutter_milestone_1/utilities/utils.dart';

class UserFirestoreServices {
  static final FirebaseFirestore firestoreDB = FirebaseFirestore.instance;
  static late String authenticaionID;
  final CollectionReference users = firestoreDB.collection('Users');
  final CollectionReference orders = firestoreDB.collection('Orders');
  final CollectionReference trips = firestoreDB.collection('Trip');
```

```
  static Future<void> initializeID() async {
    authenticaionID = FirebaseAuth.instance.currentUser!.uid;
  }

  static Future<void> addUser(UserandDriver myuser) async {
    try {
      await firestoreDB
          .collection('Users')
          .doc(authenticaionID)
          .set(myuser.objectToJson());
    } on Exception catch (e) {
      print(e.toString());
    }
  }

  static Future<UserandDriver> getUserInfo() async {
    try {
      final documentSnapshot =
          await
firestoreDB.collection('Users').doc(authenticaionID).get();
      if (documentSnapshot.exists) {
        return UserandDriver.fromSnapshot(documentSnapshot);
      } else {
        print('Document does not exist on the database');
        return UserandDriver(
          name: '0',
          email: '0',
          password: '0',
          mobileNumber: '0',
        );
      }
    } on Exception catch (e) {
      print(e.toString());
      return UserandDriver(
        name: '0',
        email: '0',
        password: '0',
        mobileNumber: '0',
      );
```

```
  }
}

static Future<List<Trip>> getAllTrips() async {
  try {
    QuerySnapshot querySnapshot =
        await FirebaseFirestore.instance.collection('Trip').get();

    return Utils.processQuerySnapshot(querySnapshot);
  } catch (e) {
    print('Error fetching available trips: $e');
    return [];
  }
}

static Future<void> addOrder(Orders order) async {
  final jsonOrder = order.orderToJson();
  jsonOrder['userId'] = authenticaionID;
  try {
    await firestoreDB.collection('Orders').add(jsonOrder);
    print('added successfully');
  } on Exception catch (e) {
    print(e.toString());
  }
}

static Future<List<Orders>> getUserOrders() async {
  final List<Orders> orders = [];
  try {
    final documentSnapshots = await firestoreDB
        .collection('Orders')
        .where(
          'userId',
          isEqualTo: authenticaionID,
        )
        .get();
    for (var doc in documentSnapshots.docs) {
      final tripItem = Orders.fromSnapshot(doc);
      orders.add(tripItem);
    }
```

```
      print(orders.length);
    } on Exception catch (e) {
      print(e.toString());
    }
    return orders;
  }
}
```

Driver_firestore:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter_milestone_1/model/orders.dart';
import 'package:flutter_milestone_1/model/trips.dart';
import 'package:flutter_milestone_1/utilities/utils.dart';

class DriverFirestoreServices {
 static final FirebaseFirestore firestoreDB = FirebaseFirestore.instance;
 static late String authenticaionID;
 final CollectionReference users = firestoreDB.collection('Users');
 final CollectionReference orders = firestoreDB.collection('Orders');
 final CollectionReference trips = firestoreDB.collection('Trip');

 static Future<void> initializeID() async {
  authenticaionID = FirebaseAuth.instance.currentUser!.uid;
 }

 static Future<void> addTrip(Trip trip) async {
  try {
   await firestoreDB.collection('Trip').add(trip.tripToJson());
  } on Exception catch (e) {
   print(e.toString());
  }
 }

 static Future<List<Trip>> getTripsForDriver(String driverId) async {
```

```
  try {
    QuerySnapshot querySnapshot = await FirebaseFirestore.instance
        .collection('Trip')
        .where('driverId', isEqualTo: driverId)
        .get();

    return Utils.processQuerySnapshot(querySnapshot);
  } catch (e) {
    print('Error fetching trips: $e');
    return [];
  }
}

static Future<List<Orders>> getDriverOrders() async {
  final List<Orders> orders = [];
  try {
    final documentSnapshots = await firestoreDB.collection('Orders').get();
    for (var firestoreorders in documentSnapshots.docs) {
      if (firestoreorders.data()['trip']['driverId'] == authenticaionID) {
        final tripItem = Orders.fromSnapshot(firestoreorders);
        orders.add(tripItem);
      }
    }
  } on Exception catch (e) {
    print(e.toString());
  }
  return orders;
}

static Future<void> editOrderStatus(
    String orderId, OrderStatus newStatus) async {
  final body = {
    'statusOrder': newStatus.toString().split('.').last,
  };
  try {
    await firestoreDB.collection('Orders').doc(orderId).update(body);
  } on Exception catch (e) {
    print(e.toString());
```

```
  }
 }
}
```

Utils.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter_milestone_1/model/trips.dart';

class Utils {
 static List<Trip> processQuerySnapshot(QuerySnapshot querySnapshot) {
  return querySnapshot.docs.map((doc) {
   Map<String, dynamic> data = doc.data() as Map<String, dynamic>;
   return Trip(
    driverId: data['driverId'],
    driverName: data['driverName'],
    fee: data['fee'],
    mobileNumber: data['mobileNumber'],
    startLocation: data['startLocation'],
    endLocation: data['endLocation'],
    tripDate: DateTime.parse(data['tripDate']),
    statusTrip: TripStatus.values.firstWhere(
     (e) => e.toString() == 'TripStatus.${data['statusTrip']}',
    ),
   );
  }).toList();
 }
}
```

Views:

Drive_home

```dart
import 'package:flutter/material.dart';
import 'package:flutter_milestone_1/model/trips.dart';
import 'package:flutter_milestone_1/service/driver_firestore.dart';

class DriverAddTrip extends StatefulWidget {
  const DriverAddTrip({Key? key}) : super(key: key);

  @override
  State<DriverAddTrip> createState() => _DriverAddTripState();
}

class _DriverAddTripState extends State<DriverAddTrip> {
  final _formKey = GlobalKey<FormState>();
  DateTime? _selectedDate;
  String? driverName;
  double? fee;
  String? mobileNumber;
  String? startLocation;
  String? endLocation;
  DateTime? tripDate;
  TripStatus statusTrip = TripStatus.pending;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Add Trip'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: SingleChildScrollView(
          child: Form(
            key: _formKey,
```

```
        child: Column(
          children: [
            TextFormField(
              decoration: const InputDecoration(
                labelText: 'Driver Name',
                labelStyle: TextStyle(color: Colors.white), // Text color
              ),
              style: const TextStyle(color: Colors.white), // Text color
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter the driver name';
                }
                return null;
              },
              onSaved: (value) {
                driverName = value;
              },
            ),
            TextFormField(
              decoration: const InputDecoration(
                labelText: 'Fee',
                labelStyle: TextStyle(color: Colors.white), // Text color
              ),
              style: const TextStyle(color: Colors.white), // Text color
              keyboardType: TextInputType.number,
              validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter the fee';
                }
                return null;
              },
              onSaved: (value) {
                fee = double.tryParse(value!);
              },
            ),
            TextFormField(
              decoration: const InputDecoration(
                labelText: 'Mobile Number',
```

```
            labelStyle: TextStyle(color: Colors.white), // Text color
          ),
          style: const TextStyle(color: Colors.white), // Text color
          keyboardType: TextInputType.phone,
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Please enter the mobile number';
            }
            return null;
          },
          onSaved: (value) {
            mobileNumber = value;
          },
        ),
        TextFormField(
          decoration: const InputDecoration(
            labelText: 'Start Location',
            labelStyle: TextStyle(color: Colors.white), // Text color
          ),
          style: const TextStyle(color: Colors.white), // Text color
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Please enter the start location';
            }
            return null;
          },
          onSaved: (value) {
            startLocation = value;
          },
        ),
        TextFormField(
          decoration: const InputDecoration(
            labelText: 'End Location',
            labelStyle: TextStyle(color: Colors.white), // Text color
          ),
          style: const TextStyle(color: Colors.white), // Text color
          validator: (value) {
            if (value == null || value.isEmpty) {
```

```
          return 'Please enter the end location';
        }
        return null;
      },
      onSaved: (value) {
        endLocation = value;
      },
    ),
    //Date
    Row(
      children: [
        Expanded(
          child: TextFormField(
            decoration: const InputDecoration(
              labelText: 'Trip Date',
              labelStyle:
                  TextStyle(color: Colors.white), // Text color
            ),
            style:
                const TextStyle(color: Colors.white), // Text color
            readOnly: true,
            controller: TextEditingController(
              text: _selectedDate == null
                  ? ''
                  : _selectedDate!.toString(),
            ),
            onTap: () async {
              final DateTime? pickedDate = await showDatePicker(
                context: context,
                initialDate: DateTime.now(),
                firstDate: DateTime(2015),
                lastDate: DateTime(2050),
              );
              if (pickedDate != null) {
                setState(() {
                  _selectedDate = pickedDate;
                });
              }
```

```
          },
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Please enter the trip date';
            }
            return null;
          },
          onSaved: (value) {
            tripDate = DateTime.parse(value!);
          },
        ),
      ),
      IconButton(
        onPressed: () async {
          final DateTime? pickedDate = await showDatePicker(
            context: context,
            initialDate: DateTime.now(),
            firstDate: DateTime(2015),
            lastDate: DateTime(2050),
          );
          if (pickedDate != null) {
            setState(() {
              _selectedDate = pickedDate;
            });
          }
        },
        icon: const Icon(Icons.calendar_today),
      ),
    ],
  ),
  Padding(
    padding: const EdgeInsets.all(8.0),
    child: ElevatedButton(
      onPressed: () {
        if (_formKey.currentState!.validate()) {
          _formKey.currentState!.save();
          if (["gate 4", "gate4", "gate 3", "gate3"]
              .contains(endLocation!.toLowerCase())) {
```

```
                  _selectedDate =
                      _selectedDate!.copyWith(hour: 7, minute: 30);
                  } else {
                    _selectedDate =
                      _selectedDate!.copyWith(hour: 17, minute: 30);
                  }
                  tripDate = _selectedDate;

                  // Create a new Trip object with the entered values
                  final newTrip = Trip(
                    driverId: DriverFirestoreServices.authenticaionID,
                    driverName: driverName!,
                    fee: fee!,
                    mobileNumber: mobileNumber!,
                    startLocation: startLocation!,
                    endLocation: endLocation!,
                    tripDate: tripDate!,
                    statusTrip: statusTrip,
                  );

                  // add trip to firestore
                  DriverFirestoreServices.addTrip(newTrip);
                  // display a snackbar to confirm that the trip has been added
                  ScaffoldMessenger.of(context).showSnackBar(
                    const SnackBar(
                      content: Text('Trip added successfully'),
                    ),
                  );
                }
              },
              child: const Text('Add Trip'),
            ),
          ),
        ],
      ),
    ),
  ),
),
```

```
  );
```

Driver_add_trip

```
import 'package:flutter/material.dart';
import 'package:flutter_milestone_1/model/trips.dart';
import 'package:flutter_milestone_1/service/driver_firestore.dart';

class DriverAddTrip extends StatefulWidget {
  const DriverAddTrip({Key? key}) : super(key: key);

  @override
  State<DriverAddTrip> createState() => _DriverAddTripState();
}

class _DriverAddTripState extends State<DriverAddTrip> {
  final _formKey = GlobalKey<FormState>();
  DateTime? _selectedDate;
  String? driverName;
  double? fee;
  String? mobileNumber;
  String? startLocation;
  String? endLocation;
  DateTime? tripDate;
  TripStatus statusTrip = TripStatus.pending;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Add Trip'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: SingleChildScrollView(
          child: Form(
            key: _formKey,
            child: Column(
              children: [
```

```
          TextFormField(
            decoration: const InputDecoration(
              labelText: 'Driver Name',
              labelStyle: TextStyle(color: Colors.white), // Text color
            ),
            style: const TextStyle(color: Colors.white), // Text color
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Please enter the driver name';
              }
              return null;
            },
            onSaved: (value) {
              driverName = value;
            },
          ),
          TextFormField(
            decoration: const InputDecoration(
              labelText: 'Fee',
              labelStyle: TextStyle(color: Colors.white), // Text color
            ),
            style: const TextStyle(color: Colors.white), // Text color
            keyboardType: TextInputType.number,
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Please enter the fee';
              }
              return null;
            },
            onSaved: (value) {
              fee = double.tryParse(value!);
            },
          ),
          TextFormField(
            decoration: const InputDecoration(
              labelText: 'Mobile Number',
              labelStyle: TextStyle(color: Colors.white), // Text color
            ),
```

```
    style: const TextStyle(color: Colors.white), // Text color
    keyboardType: TextInputType.phone,
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please enter the mobile number';
      }
      return null;
    },
    onSaved: (value) {
      mobileNumber = value;
    },
  ),
  TextFormField(
    decoration: const InputDecoration(
      labelText: 'Start Location',
      labelStyle: TextStyle(color: Colors.white), // Text color
    ),
    style: const TextStyle(color: Colors.white), // Text color
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please enter the start location';
      }
      return null;
    },
    onSaved: (value) {
      startLocation = value;
    },
  ),
  TextFormField(
    decoration: const InputDecoration(
      labelText: 'End Location',
      labelStyle: TextStyle(color: Colors.white), // Text color
    ),
    style: const TextStyle(color: Colors.white), // Text color
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please enter the end location';
      }
```

```
            return null;
          },
          onSaved: (value) {
            endLocation = value;
          },
        ),
        //Date
        Row(
          children: [
            Expanded(
              child: TextFormField(
                decoration: const InputDecoration(
                  labelText: 'Trip Date',
                  labelStyle:
                      TextStyle(color: Colors.white), // Text color
                ),
                style:
                    const TextStyle(color: Colors.white), // Text color
                readOnly: true,
                controller: TextEditingController(
                  text: _selectedDate == null
                      ? ''
                      : _selectedDate!.toString(),
                ),
                onTap: () async {
                  final DateTime? pickedDate = await showDatePicker(
                    context: context,
                    initialDate: DateTime.now(),
                    firstDate: DateTime(2015),
                    lastDate: DateTime(2050),
                  );
                  if (pickedDate != null) {
                    setState(() {
                      _selectedDate = pickedDate;
                    });
                  }
                },
                validator: (value) {
```

```
            if (value == null || value.isEmpty) {
              return 'Please enter the trip date';
            }
            return null;
          },
          onSaved: (value) {
            tripDate = DateTime.parse(value!);
          },
        ),
      ),
      IconButton(
        onPressed: () async {
          final DateTime? pickedDate = await showDatePicker(
            context: context,
            initialDate: DateTime.now(),
            firstDate: DateTime(2015),
            lastDate: DateTime(2050),
          );
          if (pickedDate != null) {
            setState(() {
              _selectedDate = pickedDate;
            });
          }
        },
        icon: const Icon(Icons.calendar_today),
      ),
    ],
  ),
  Padding(
    padding: const EdgeInsets.all(8.0),
    child: ElevatedButton(
      onPressed: () {
        if (_formKey.currentState!.validate()) {
          _formKey.currentState!.save();
          if (["gate 4", "gate4", "gate 3", "gate3"]
              .contains(endLocation!.toLowerCase())) {
            _selectedDate =
                _selectedDate!.copyWith(hour: 7, minute: 30);
```

```
                    } else {
                      _selectedDate =
                          _selectedDate!.copyWith(hour: 17, minute: 30);
                    }
                    tripDate = _selectedDate;

                    // Create a new Trip object with the entered values
                    final newTrip = Trip(
                      driverId: DriverFirestoreServices.authenticaionID,
                      driverName: driverName!,
                      fee: fee!,
                      mobileNumber: mobileNumber!,
                      startLocation: startLocation!,
                      endLocation: endLocation!,
                      tripDate: tripDate!,
                      statusTrip: statusTrip,
                    );

                    // add trip to firestore
                    DriverFirestoreServices.addTrip(newTrip);
                    // display a snackbar to confirm that the trip has been added
                    ScaffoldMessenger.of(context).showSnackBar(
                      const SnackBar(
                        content: Text('Trip added successfully'),
                      ),
                    );
                  }
                },
                child: const Text('Add Trip'),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}
```

```
}
```

Driver_order

```
import 'package:flutter/material.dart';
import 'package:flutter_milestone_1/model/orders.dart';
import 'package:flutter_milestone_1/service/driver_firestore.dart';

class DriverOrders extends StatefulWidget {
  const DriverOrders({Key? key}) : super(key: key);

  @override
  State<DriverOrders> createState() => _DriverOrdersState();
}

class _DriverOrdersState extends State<DriverOrders> {
  List<Orders> driverOrders = [];

  @override
  void initState() {
    super.initState();
    loadDriverOrders();
  }

  Future<void> loadDriverOrders() async {
    try {
      List<Orders> orders = await DriverFirestoreServices.getDriverOrders();

      setState(() {
        driverOrders = orders;
        print(orders);
      });
    } catch (e) {
      print('Error loading driver orders: $e');
    }
  }
```

```
Future<void> acceptOrder(String orderId) async {
  try {
    await DriverFirestoreServices.editOrderStatus(
        orderId, OrderStatus.accepted);
    // Refresh the list of driver orders after accepting an order
    loadDriverOrders();
  } catch (e) {
    print('Error accepting order: $e');
  }
}

Future<void> rejectOrder(String orderId) async {
  try {
    await DriverFirestoreServices.editOrderStatus(
        orderId, OrderStatus.rejected);
    // Refresh the list of driver orders after rejecting an order
    loadDriverOrders();
  } catch (e) {
    print('Error rejecting order: $e');
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text("UberX-ASU")),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          const Text(
            "Driver Orders",
            style: TextStyle(
              fontSize: 24,
              fontWeight: FontWeight.bold,
              color: Colors.black,
            ),
```

```
          ),
          const SizedBox(height: 16.0),
          Expanded(
            child: ListView.builder(
              itemCount: driverOrders.length,
              itemBuilder: (context, index) {
                return buildOrderItem(driverOrders[index]);
              },
            ),
          ),
        ],
      ),
    ),
  );
}

Widget buildOrderItem(Orders order) {
  return Card(
    margin: const EdgeInsets.symmetric(vertical: 8.0),
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            "${order.trip.startLocation} to ${order.trip.endLocation} - ${order.trip.tripDate}",
            style: const TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
          ),
          const SizedBox(height: 8.0),
          Text("Passenger: ${order.userName}"),
          Text("Phone: ${order.phoneNo}"),
          Text("Cost: \$${order.totalAmount}"),
          const SizedBox(height: 8.0),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              ElevatedButton(
                onPressed: () => acceptOrder(order.orderId!),
```

```
              child: const Text("Accept"),
            ),
            ElevatedButton(
              onPressed: () => rejectOrder(order.orderId!),
              style: ButtonStyle(
                backgroundColor: MaterialStateProperty.all(Colors.red),
              ),
              child: const Text("Reject"),
            ),
          ],
        ),
      ],
    ),
  ),
 );
 }
}
```

Driver_trips

```
import 'package:flutter/material.dart';
import 'package:flutter_milestone_1/model/trips.dart';
import 'package:flutter_milestone_1/service/driver_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

class DriverTrips extends StatefulWidget {
 const DriverTrips({Key? key}) : super(key: key);

 @override
 State<DriverTrips> createState() => _DriverTripsState();
}

class _DriverTripsState extends State<DriverTrips> {
 List<Trip> trips = [];

 @override
```

```
void initState() {
  super.initState();
  loadTripsData();
}

Future<void> loadTripsData() async {
  try {
    String authenticationId = FirebaseAuth.instance.currentUser!.uid;
    List<Trip> driverTrips =
        await DriverFirestoreServices.getTripsForDriver(authenticationId);

    setState(() {
      trips = driverTrips;
    });
  } catch (e) {
    print('Error loading trips: $e');
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Trips'),
    ),
    body: ListView.builder(
      itemCount: trips.length,
      itemBuilder: (context, index) {
        final trip = trips[index];
        return Card(
          child: ListTile(
            title: Text(trip.driverName),
            subtitle: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text('Mobile Number: ${trip.mobileNumber}'),
                Text('Fee: \$${trip.fee.toStringAsFixed(2)}'),
                Text('Start Location: ${trip.startLocation}'),
```

```
                Text('End Location: ${trip.endLocation}'),
                Text('Status: ${trip.statusTrip.toString().split('.').last}'),
                Text('Trip Date: ${trip.tripDate.toString()}'),
              ],
            ),
            onTap: () {
              // Handle trip selection
            },
          ),
        );
      },
    ),
  );
 }
}
```

Profile:

```
import 'package:flutter/material.dart';
import 'package:flutter_milestone_1/model/users.dart';
import 'package:flutter_milestone_1/service/user_firestore.dart';

class ProfileScreen extends StatelessWidget {
 ProfileScreen({required Key key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Profile'),
    ),
    body: FutureBuilder<UserandDriver>(
      future: UserFirestoreServices.getUserInfo(),
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return Center(child: CircularProgressIndicator());
```

```
        } else if (snapshot.hasError) {
          return Center(child: Text('Error: ${snapshot.error}'));
        } else {
          UserandDriver userData = snapshot.data!;
          return buildProfileWidget(userData);
        }
      },
    ),
  );
}

Widget buildProfileWidget(UserandDriver userData) {
  return Center(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        const CircleAvatar(
          radius: 80,
          backgroundImage: NetworkImage('https://placekitten.com/200/200'),
        ),
        const SizedBox(height: 24),
        const Text(
          'User Details:',
          style: TextStyle(
            fontSize: 24,
            fontWeight: FontWeight.bold,
            color: Colors.white,
          ),
        ),
        const SizedBox(height: 12),
        buildDetailRow('Name', userData.name),
        buildDetailRow('Email', userData.email),
        buildDetailRow('Mobile Number', userData.mobileNumber),
      ],
    ),
  );
}
```

```
Widget buildDetailRow(String label, String value) {
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 8),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Text(
          '$label:',
          style: const TextStyle(
            color: Colors.white,
            fontWeight: FontWeight.bold,
            fontSize: 18,
          ),
        ),
        const SizedBox(width: 12),
        Text(
          value,
          style: const TextStyle(
            color: Colors.white,
            fontSize: 20,
          ),
        ),
      ],
    ),
  );
 }
}
```

Sign in

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter_milestone_1/service/driver_firestore.dart';
import 'package:flutter_milestone_1/service/user_firestore.dart';
```

```dart
class SignIn extends StatefulWidget {
  const SignIn({Key? key}) : super(key: key);

  @override
  State<SignIn> createState() => _SignInState();
}

class _SignInState extends State<SignIn> {
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  var _isDriver = false;
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Center(child: Text("UberX-ASU")),
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(32.0),
          child: Form(
            key: _formKey,
            child: Center(
              child: Column(
                children: [
                  Image.asset('assets/images/asu_circular.png'),
                  const SizedBox(height: 30),
                  TextFormField(
                    controller: _emailController,
                    decoration: const InputDecoration(
                      suffixIcon: Icon(Icons.email),
                      hintText: 'Enter Your University Email',
                      labelText: 'University Email',
                      labelStyle: TextStyle(color: Colors.black),
                      fillColor: Colors.white,
                      filled: true,
```

```
  ),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your email';
    } else if (!value.contains('@eng.asu.edu.eg')) {
      return 'Invalid email format';
    }
    return null;
  },
),
const SizedBox(height: 16.0),
TextFormField(
  controller: _passwordController,
  decoration: const InputDecoration(
    suffixIcon: Icon(Icons.lock),
    hintText: 'Enter Password',
    labelText: 'Password',
    labelStyle: TextStyle(color: Colors.black),
    fillColor: Colors.white,
    filled: true,
  ),
  obscureText: true,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter your password';
    } else if (value.length < 6 || value.length > 20) {
      return 'Password must be between 6 and 20 characters';
    }
    return null;
  },
),
const SizedBox(height: 5),
Switch(
  value: _isDriver,
  onChanged: (value) {
    setState(() {
      _isDriver = value;
    });
```

```
  },
),
Column(
  children: [
    Text("I am a Student",
        style: TextStyle(
            color: _isDriver ? Colors.grey : Colors.blue)),
    Text("I am a Driver",
        style: TextStyle(
            color: _isDriver ? Colors.blue : Colors.grey)),
  ],
),
const SizedBox(height: 5),
ElevatedButton(
  onPressed: () {
    String email = _emailController.text.trim();
    String password = _passwordController.text.trim();
    if (_formKey.currentState?.validate() ?? false) {
      FirebaseAuth.instance
          .signInWithEmailAndPassword(
        email: (_isDriver) ? '$email.driver' : email,
        password: password,
      )
          .then((userCredential) async {
        if (_isDriver) {
          await DriverFirestoreServices.initializeID();
          if (!context.mounted) return;
          Navigator.pushNamed(context, 'driverHome');
        } else {
          await UserFirestoreServices.initializeID();
          if (!context.mounted) return;
          Navigator.pushNamed(context, 'home');
        }
      }).catchError((error) {
        // Authentication failed, print error message
        print(error);
      });
    }
```

```
              },
              style: ElevatedButton.styleFrom(
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(10)),
                minimumSize: const Size(double.infinity, 40),
              ),
              child:
                  const Text("Sign In", style: TextStyle(fontSize: 20)),
            ),
            const SizedBox(height: 10),
            ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, 'signUp');
              },
              style: ElevatedButton.styleFrom(
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(10)),
                minimumSize: const Size(double.infinity, 40),
              ),
              child:
                  const Text("Sign Up", style: TextStyle(fontSize: 20)),
            ),
          ],
        ),
      ),
    ),
  );
 }
}
```

Sign up

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
```

```dart
import 'package:flutter_milestone_1/model/database_helper.dart';
import 'package:flutter_milestone_1/model/users.dart';
import 'package:flutter_milestone_1/service/user_firestore.dart';

class SignUp extends StatefulWidget {
  const SignUp({Key? key}) : super(key: key);

  @override
  State<SignUp> createState() => _SignUpState();
}

class _SignUpState extends State<SignUp> {
  final _nameController = TextEditingController();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  final _mobilePhoneController = TextEditingController();
  var _isDriver = false;
  final _formKey = GlobalKey<FormState>();
  final dbHelper = DatabaseHelper();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Center(child: Text("UberX-ASU")),
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(32.0),
          child: Form(
            key: _formKey,
            child: Center(
              child: Column(
                children: [
                  Image.asset('assets/images/asu_circular.png'),
                  const SizedBox(height: 30),
                  TextFormField(
                    controller: _nameController,
                    decoration: const InputDecoration(
                      suffixIcon: Icon(Icons.person),
```

```
                    hintText: 'Enter Your Name',
                    labelText: 'Name',
                    labelStyle: TextStyle(color: Colors.black),
                    fillColor: Colors.white,
                    filled: true,
                  ),
                  validator: (value) {
                    if (value == null || value.isEmpty) {
                      return 'Please enter your name';
                    }
                    return null;
                  },
                ),
                const SizedBox(height: 16.0),
                TextFormField(
                  controller: _emailController,
                  decoration: const InputDecoration(
                    suffixIcon: Icon(Icons.email),
                    hintText: 'Enter Your University Email',
                    labelText: 'University Email',
                    labelStyle: TextStyle(color: Colors.black),
                    fillColor: Colors.white,
                    filled: true,
                  ),
                  validator: (value) {
                    if (value == null || value.isEmpty) {
                      return 'Please enter your email';
                    } else if (!value.contains('@eng.asu.edu.eg')) {
                      return 'Invalid email format';
                    }
                    return null;
                  },
                ),
                const SizedBox(height: 16.0),
                TextFormField(
                  controller: _passwordController,
                  decoration: const InputDecoration(
                    suffixIcon: Icon(Icons.lock),
                    hintText: 'Enter Password',
                    labelText: 'Password',
```

```
                    labelStyle: TextStyle(color: Colors.black),
                    fillColor: Colors.white,
                    filled: true,
                  ),
                  obscureText: true,
                  validator: (value) {
                    if (value == null || value.isEmpty) {
                      return 'Please enter your password';
                    } else if (value.length < 6 || value.length > 20) {
                      return 'Password must be between 6 and 20
characters';
                    }
                    return null;
                  },
                ),
                const SizedBox(height: 16.0),
                TextFormField(
                  controller: _mobilePhoneController,
                  decoration: const InputDecoration(
                    suffixIcon: Icon(Icons.phone),
                    hintText: 'Enter Your Mobile Phone',
                    labelText: 'Mobile Phone',
                    labelStyle: TextStyle(color: Colors.black),
                    fillColor: Colors.white,
                    filled: true,
                  ),
                  validator: (value) {
                    if (value == null || value.isEmpty) {
                      return 'Please enter your mobile phone';
                    } else if (!RegExp(r'^[0-9]{11}$').hasMatch(value))
{
                      return 'Invalid mobile phone format';
                    }
                    return null;
                  },
                ),
                const SizedBox(height: 5),
                Switch(
                  value: _isDriver,
                  onChanged: (value) {
```

```
                    setState(() {
                      _isDriver = value;
                    });
                  },
                ),
                Column(
                  children: [
                    Text("I am a Student",
                        style: TextStyle(
                            color: _isDriver ? Colors.grey :
Colors.blue)),
                    Text("I am a Driver",
                        style: TextStyle(
                            color: _isDriver ? Colors.blue :
Colors.grey)),
                  ],
                ),
                ElevatedButton(
                  onPressed: () async {
                    _insert();
                    String name = _nameController.text.trim();
                    String email = _emailController.text.trim();
                    String password = _passwordController.text.trim();
                    String mobilePhone =
_mobilePhoneController.text.trim();
                    if (_formKey.currentState!.validate()) {
                      if (email.endsWith('@eng.asu.edu.eg')) {
                        FirebaseAuth.instance
                            .createUserWithEmailAndPassword(
                              email: (_isDriver) ? '$email.driver' :
email,
                              password: password,
                            )
                            .then((value) => {
                                 UserFirestoreServices.initializeID(),

UserFirestoreServices.addUser(UserandDriver(
                                      name: name,
                                      email: email,
                                      password: password,
```

```
                                    mobileNumber: mobilePhone,
                                  )),
                                });
                            if (_isDriver) {
                              Navigator.pushNamed(context, 'driverHome');
                            } else {
                              Navigator.pushNamed(context, 'home');
                            }
                          }
                        },
                        style: ElevatedButton.styleFrom(
                          shape: RoundedRectangleBorder(
                              borderRadius: BorderRadius.circular(10)),
                          minimumSize: const Size(double.infinity, 40),
                        ),
                        child:
                            const Text("Sign Up", style: TextStyle(fontSize:
20)),
                      ),
                    ],
                  ),
                ),
              ),
            ),
          ),
        );
  }

  void _insert() async {
    // row to insert
    Map<String, dynamic> row = {
      DatabaseHelper.columnName: _nameController.text.trim(),
      DatabaseHelper.columnEmail: _emailController.text.trim(),
      DatabaseHelper.columnPhone: _mobilePhoneController.text.trim(),
      DatabaseHelper.columnPassword: _passwordController.text.trim(),
    };
    final id = await dbHelper.insert(row);
    debugPrint('inserted row id: $id');
  }
```

```
}
```

Home

```
import 'package:flutter/material.dart';

class Home extends StatefulWidget {
  const Home({super.key});

  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("UberX-ASU"), actions: <Widget>[
        IconButton(
          icon: const Icon(Icons.person),
          tooltip: 'Profile',
          onPressed: () {
            Navigator.pushNamed(context, 'profile');
          },
        ),
      ]),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(32.0),
          child: Center(
            child: Column(
              children: [
                const Text("Welcome to UberX-ASU!",
                    style: TextStyle(
                      fontSize: 30,
                      color: Colors.white,
                    )),
```

```
            const Text(
              "Search for routes to and from ASU",
              style: TextStyle(fontSize: 20, color: Colors.white),
            ),
            const SizedBox(height: 30),
            // get image from assets
            Image.asset('assets/images/asu_building.png'),
            const SizedBox(height: 30),
            ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, 'route');
              },
              style: ElevatedButton.styleFrom(
                minimumSize: const Size(
                    double.infinity, 40), // Set minimum button size
              ),
              child: const Text("Available Routes",
                  style: TextStyle(fontSize: 20)),
            ),
            const SizedBox(height: 10),
            ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, 'userOrders');
              },
              style: ElevatedButton.styleFrom(
                minimumSize: const Size(
                    double.infinity, 40), // Set minimum button size
              ),
              child: const Text("Orders", style: TextStyle(fontSize: 20)),
            ),
          ],
        ),
      ),
    ),
  ));
 }
}
```

Routes

```
import 'package:flutter/material.dart';
import 'package:flutter_credit_card/flutter_credit_card.dart';
import 'package:flutter_milestone_1/model/orders.dart';
import 'package:flutter_milestone_1/model/trips.dart';
import 'package:flutter_milestone_1/model/users.dart';
import 'package:flutter_milestone_1/service/user_firestore.dart';

class Routes extends StatefulWidget {
  const Routes({Key? key}) : super(key: key);

  @override
  State<Routes> createState() => _RoutesState();
}

class _RoutesState extends State<Routes> {
  List<Trip> allTrips = [];
  TextEditingController cardNumberController = TextEditingController();
  TextEditingController expiryDateController = TextEditingController();
  TextEditingController cardHolderNameController = TextEditingController();
  TextEditingController cvvCodeController = TextEditingController();
  bool disableconstraint = false;
  UserandDriver user = UserandDriver(
    name: 'Ahmed',
    email: '',
    password: '',
    mobileNumber: '',
  );

  @override
  void initState() {
    super.initState();
    loadAllTripsData();
  }
```

```dart
Future<void> loadAllTripsData() async {
  try {
    List<Trip> trips = await UserFirestoreServices.getAllTrips();

    setState(() {
      allTrips = trips;
    });
  } catch (e) {
    print('Error loading all trips: $e');
  }
}

bool _canReserve(Trip trip) {
  final Duration difference = trip.tripDate.difference(DateTime.now());
  if (disableconstraint) {
    return true;
  }
  if (trip.endLocation == "gate 3" ||
      trip.endLocation == "gate 4" ||
      trip.endLocation == "gate3" ||
      trip.endLocation == "gate4") {
    if (difference.inHours > 10 ||
        difference.inHours == 9 && difference.inMinutes.remainder(60) > 30) {
      return true;
    }
  } else {
    if (difference.inHours > 5 ||
        (difference.inHours == 4 &&
            difference.inMinutes.remainder(60) > 30)) {
      return true;
    }
  }
  return false;
}

Future<void> _reserveTrip(Trip trip) async {
  Orders order = Orders(
    trip: trip,
```

```
    statusOrder: OrderStatus.pending,

    phoneNo: trip.mobileNumber,
    userName: user.name,
    totalAmount: trip.fee,

    // You need to set appropriate values based on your data model

    // Set other properties based on your data model and the trip
    // For example, you might use trip.driverName, trip.mobileNumber, etc.
  );

  // Add the order to Firestore
  await UserFirestoreServices.addOrder(order);

  // Create an Orders object from the Trip
}

void _showPaymentSheet(Trip trip) {
  showModalBottomSheet(
    isScrollControlled: true, // Set to true for a full-height bottom sheet
    context: context,
    builder: (BuildContext context) {
      return SingleChildScrollView(
        child: Container(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            mainAxisSize: MainAxisSize.min,
            children: [
              CreditCardWidget(
                cardNumber: cardNumberController.text,
                expiryDate: expiryDateController.text,
                cardHolderName: cardHolderNameController.text,
                cvvCode: cvvCodeController.text,
                showBackView: false,
                onCreditCardWidgetChange: (CreditCardBrand) {},
              ),
```

```
            TextFormField(
              controller: cardNumberController,
              decoration: const InputDecoration(labelText: 'Card Number'),
              keyboardType: TextInputType.number,
            ),
            TextFormField(
              controller: expiryDateController,
              decoration: const InputDecoration(labelText: 'Expiry Date'),
              keyboardType: TextInputType.number,
            ),
            TextFormField(
              controller: cardHolderNameController,
              decoration:
                  const InputDecoration(labelText: 'Cardholder Name'),
            ),
            TextFormField(
              controller: cvvCodeController,
              decoration: const InputDecoration(labelText: 'CVV'),
              keyboardType: TextInputType.number,
            ),
            ElevatedButton(
              onPressed: () {
                // Assuming payment is successful, reserve the trip
                _reserveTrip(trip);
                Navigator.pop(context); // Close the bottom sheet
              },
              child: const Text('Pay and Reserve'),
            ),
          ],
        ),
      ),
    );
  },
);
}

@override
Widget build(BuildContext context) {
```

```
return Scaffold(
  appBar: AppBar(
    title: const Text('All Trips'),
  ),
  body: Column(children: [
    Row(
      children: [
        const Text('Disable Constraint',
            style: TextStyle(color: Colors.white)),
        Checkbox(
            value: disableconstraint,
            onChanged: (value) {
              setState(() {
                disableconstraint = value!;
              });
            }),
      ],
    ),
    Expanded(
      child: ListView.builder(
        itemCount: allTrips.length,
        itemBuilder: (context, index) {
          final trip = allTrips[index];
          return Card(
            child: ListTile(
              title: Text(trip.driverName),
              subtitle: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text('Mobile Number: ${trip.mobileNumber}'),
                  Text('Fee: \$${trip.fee.toStringAsFixed(2)}'),
                  Text('Start Location: ${trip.startLocation}'),
                  Text('End Location: ${trip.endLocation}'),
                  Text(
                      'Status: ${trip.statusTrip.toString().split('.').last}'),
                  Text('Trip Date: ${trip.tripDate.toString()}'),
                ],
              ),
```

```
              trailing: ElevatedButton(
                onPressed: () {
                  if (!_canReserve(trip)) {
                    ScaffoldMessenger.of(context).showSnackBar(
                      const SnackBar(
                        content: Text(
                            'You cannot reserve this trip. Please select another trip.'),
                      ),
                    );
                    return;
                  }
                  // Handle the button press to show payment sheet
                  _showPaymentSheet(trip);
                },
                child: const Text('Reserve'),
              ),
              onTap: () {
                // Handle trip selection
              },
            ),
          );
        },
      ),
    ),
  ]),
 );
 }
}
```

User orders:

```
import 'package:flutter/material.dart';
import 'package:flutter_milestone_1/model/orders.dart';
import 'package:flutter_milestone_1/service/driver_firestore.dart';

class UserOrders extends StatefulWidget {
  const UserOrders({Key? key}) : super(key: key);
```

```
  @override
  State<UserOrders> createState() => _UserOrdersState();
}

class _UserOrdersState extends State<UserOrders> {
  List<Orders> driverOrders = [];
  @override
  void initState() {
    super.initState();
    loadDriverOrders();
  }

  Future<void> loadDriverOrders() async {
    try {
      List<Orders> orders = await DriverFirestoreServices.getDriverOrders();

      print("Driver Orders: $orders"); // Print orders to console

      setState(() {
        driverOrders = orders;
      });
    } catch (e) {
      print('Error loading driver orders: $e');
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("UberX-ASU"),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            // Header
```

```
          const Text(
            "Review Your Order",
            style: TextStyle(
                fontSize: 24,
                fontWeight: FontWeight.bold,
                color: Colors.white),
          ),
          const SizedBox(height: 16.0),
          // List of Selected Rides
          Expanded(
            child: ListView.builder(
              itemCount: driverOrders.length,
              itemBuilder: (context, index) {
                return buildOrderItem(driverOrders[index]);
              },
            ),
          ),
          // Total Cost
          const Divider(),
          const SizedBox(height: 16.0),
        ],
      ),
    ),
  );
}
Widget buildOrderItem(Orders order) {
  return Card(
    margin: const EdgeInsets.symmetric(vertical: 8.0),
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            "${order.trip.startLocation} to ${order.trip.endLocation} - ${order.trip.tripDate}",
            style: const TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
          ),
          const SizedBox(height: 8.0),
```

```
          Text("Driver: ${order.trip.driverName}"),
          Text("Cost: \$${order.totalAmount}"),
          Text("Status: ${order.statusOrder.toString().split('.').last}"),
          const SizedBox(height: 8.0),
          Align(
            alignment: Alignment.centerRight,
            child: TextButton(
              onPressed: () {
                // Implement your logic to remove the order
                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(
                    content: Text("Order removed from cart"),
                  ),
                );
              },
              style: ButtonStyle(
                foregroundColor: MaterialStateProperty.all(Colors.red),
              ),
              child: const Text("Remove"),
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

Firebase options

```
// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars, avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
```

```dart
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
///   options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for web - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        return ios;
      case TargetPlatform.macOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for macos - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.windows:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for windows - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      default:
        throw UnsupportedError(
```

```
        'DefaultFirebaseOptions are not supported for this platform.',
      );
  }
 }

 static const FirebaseOptions android = FirebaseOptions(
  apiKey: 'AIzaSyDdp2vtBleqUw35N-eNzo0awied3eNVi4g',
  appId: '1:703297432412:android:5c16d2f30082a11ae9ef99',
  messagingSenderId: '703297432412',
  projectId: 'final-project-fe701',
  storageBucket: 'final-project-fe701.appspot.com',
 );

 static const FirebaseOptions ios = FirebaseOptions(
  apiKey: 'AIzaSyAPqjV12Wke_7VTrflrvwIG0RTBp5a6o0U',
  appId: '1:703297432412:ios:3a4dd60a0c9f69bde9ef99',
  messagingSenderId: '703297432412',
  projectId: 'final-project-fe701',
  storageBucket: 'final-project-fe701.appspot.com',
  iosBundleId: 'com.example.flutterMilestone1',
 );
}
```

Main dart

```
// ignore_for_file: non_constant_identifier_names

import 'package:flutter/material.dart';
import 'package:flutter_milestone_1/model/database_helper.dart';
import 'package:flutter_milestone_1/view/driver/driver_add_trip.dart';
import 'package:flutter_milestone_1/view/driver/driver_home.dart';
import 'package:flutter_milestone_1/view/driver/driver_orders.dart';
import 'package:flutter_milestone_1/view/driver/driver_trips.dart';
import 'package:flutter_milestone_1/view/shared/profile.dart';
import 'package:flutter_milestone_1/view/user/home.dart';
import 'package:flutter_milestone_1/view/user/routes.dart';
import 'package:flutter_milestone_1/view/shared/sign_in.dart';
import 'package:flutter_milestone_1/view/user/user_orders.dart';
```

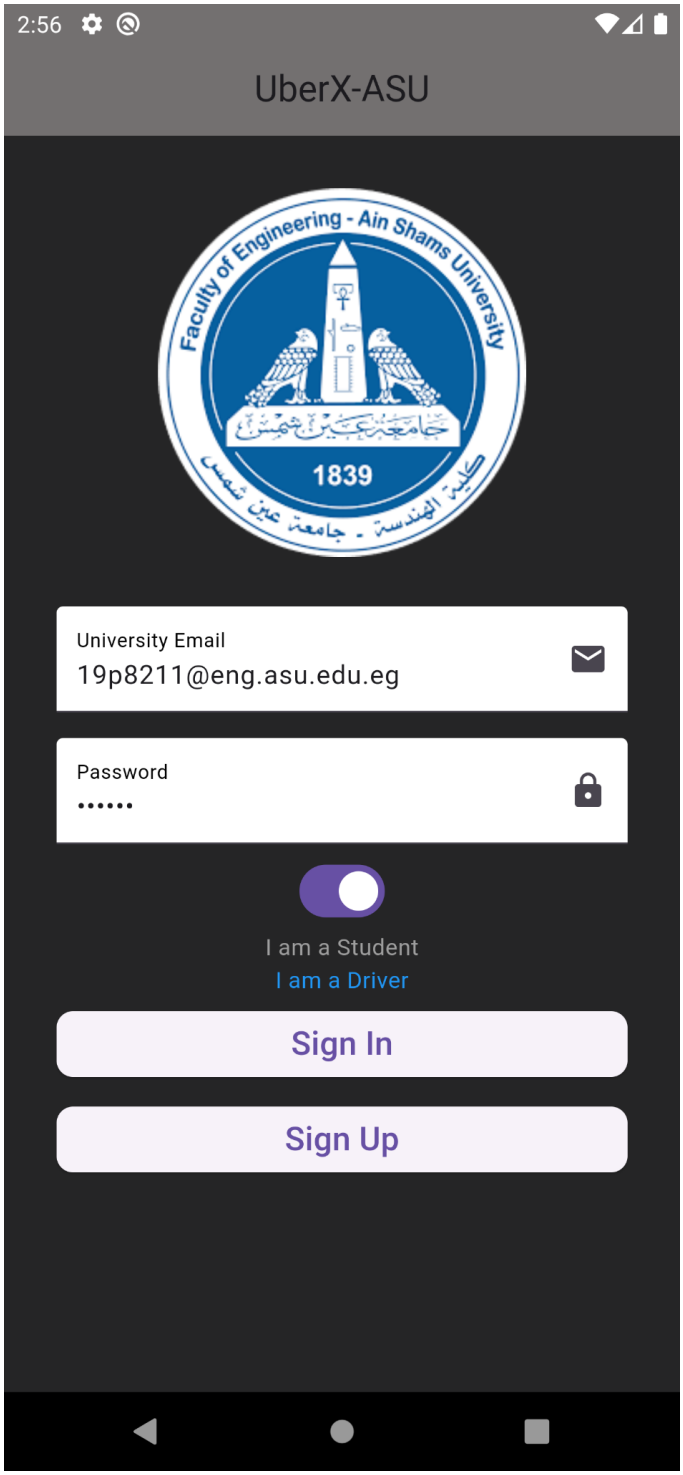
```dart
import 'package:flutter_milestone_1/view/shared/sign_up.dart';
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';

final dbHelper = DatabaseHelper();

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await dbHelper.init(); // Ensure that init is called first
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(const MyApp()); //MaterialApp
}

final ThemeData myTheme = ThemeData(
  primaryColor: const Color(0xFF252526),
  scaffoldBackgroundColor: const Color(0xFF252526),
  appBarTheme: const AppBarTheme(
    color: Color(0xff737070),
  ),
);

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: myTheme,
      debugShowCheckedModeBanner: false,
      home: const SignIn(),
      routes: {
        'signIN': (context) => const SignIn(),
        'home': (context) => const Home(),
        'route': (context) => const Routes(),
        'userOrders': (context) => const UserOrders(),
        'signUp': (context) => const SignUp(),
```

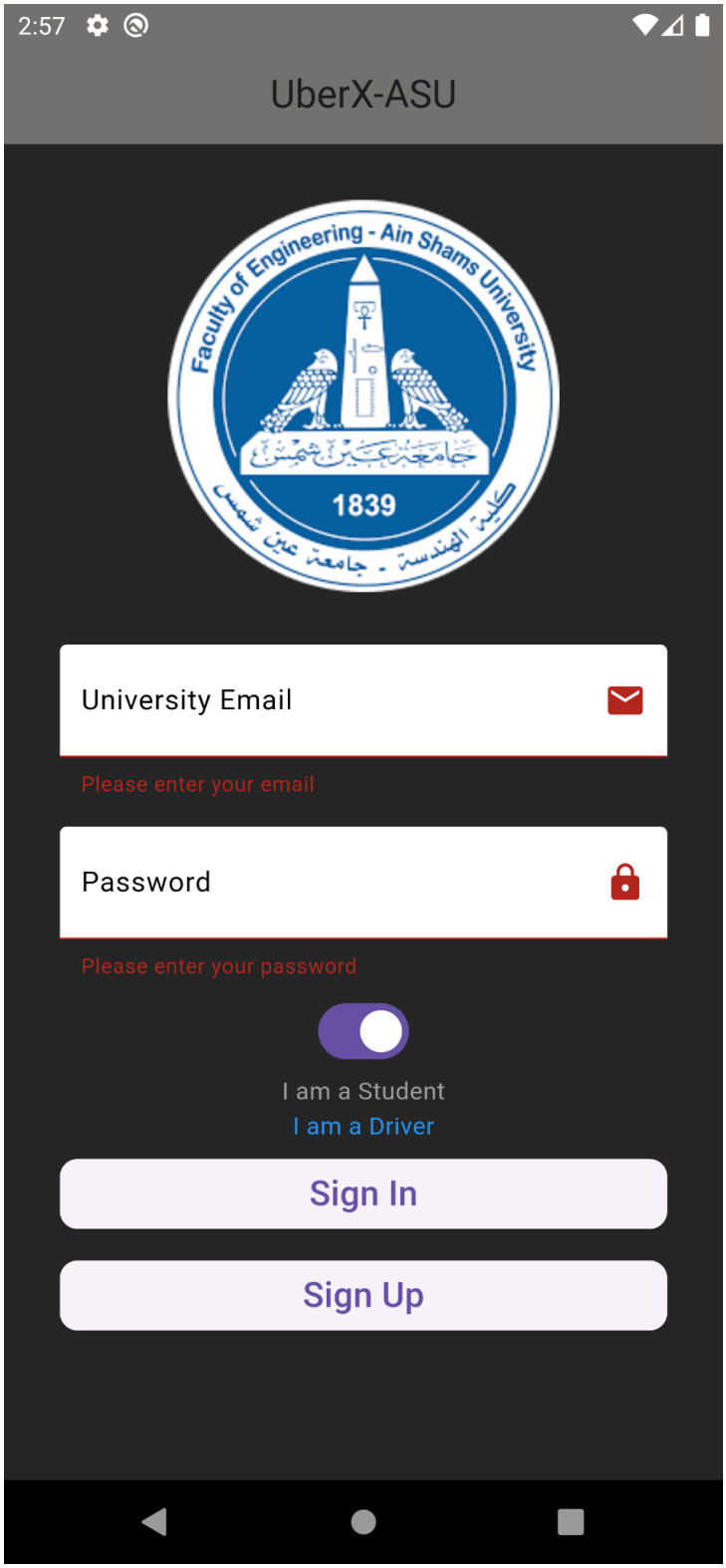
```
      'driverTrips': (context) => const DriverTrips(),
      'driverHome': (context) => const DriverHome(),
      'driveraddTrip': (context) => const DriverAddTrip(),
      'driverOrders': (context) => const DriverOrders(),
      'profile': (context) => ProfileScreen(key: UniqueKey()),
    },
  );
}
```

Test Cases

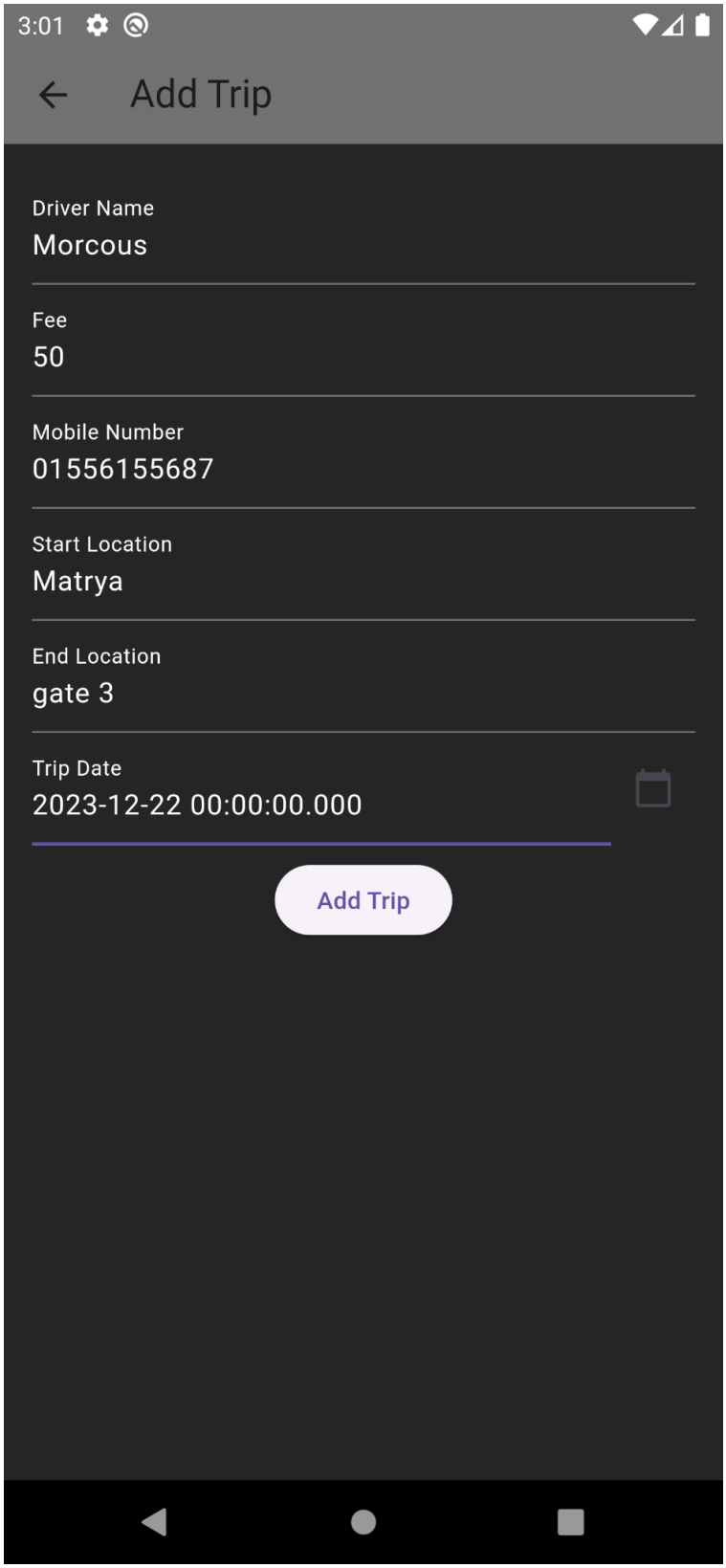
1)Sign in with
Name: Morcous
Email: 19p8211@eng.asu.edu.eg
Password:123456

Mobile Number: 01556155687

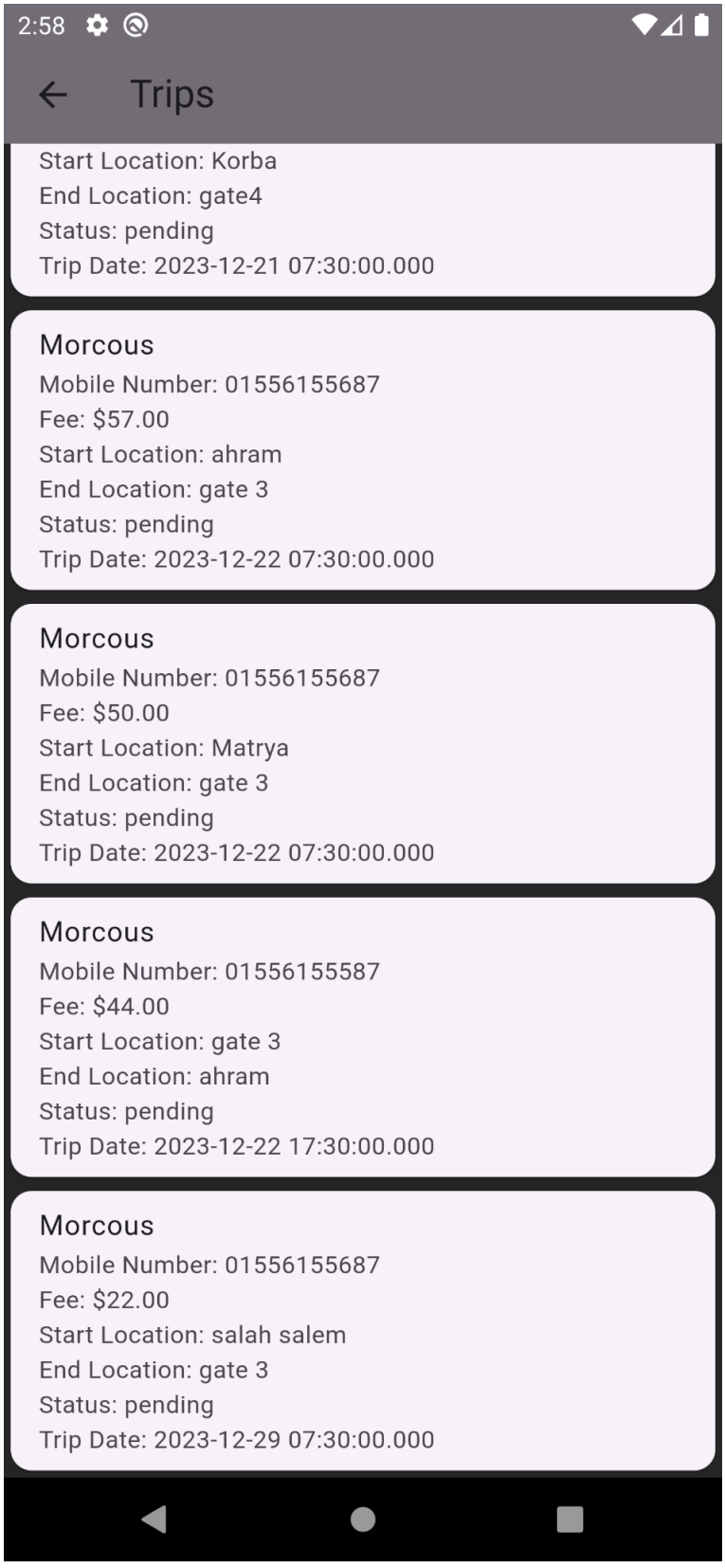



2:57
UberX-ASU
Faculty of Engineering - Ain Shams University
1839
University Email
Please enter your email
Password
Please enter your password
I am a Student
I am a Driver
Sign In
Sign Up

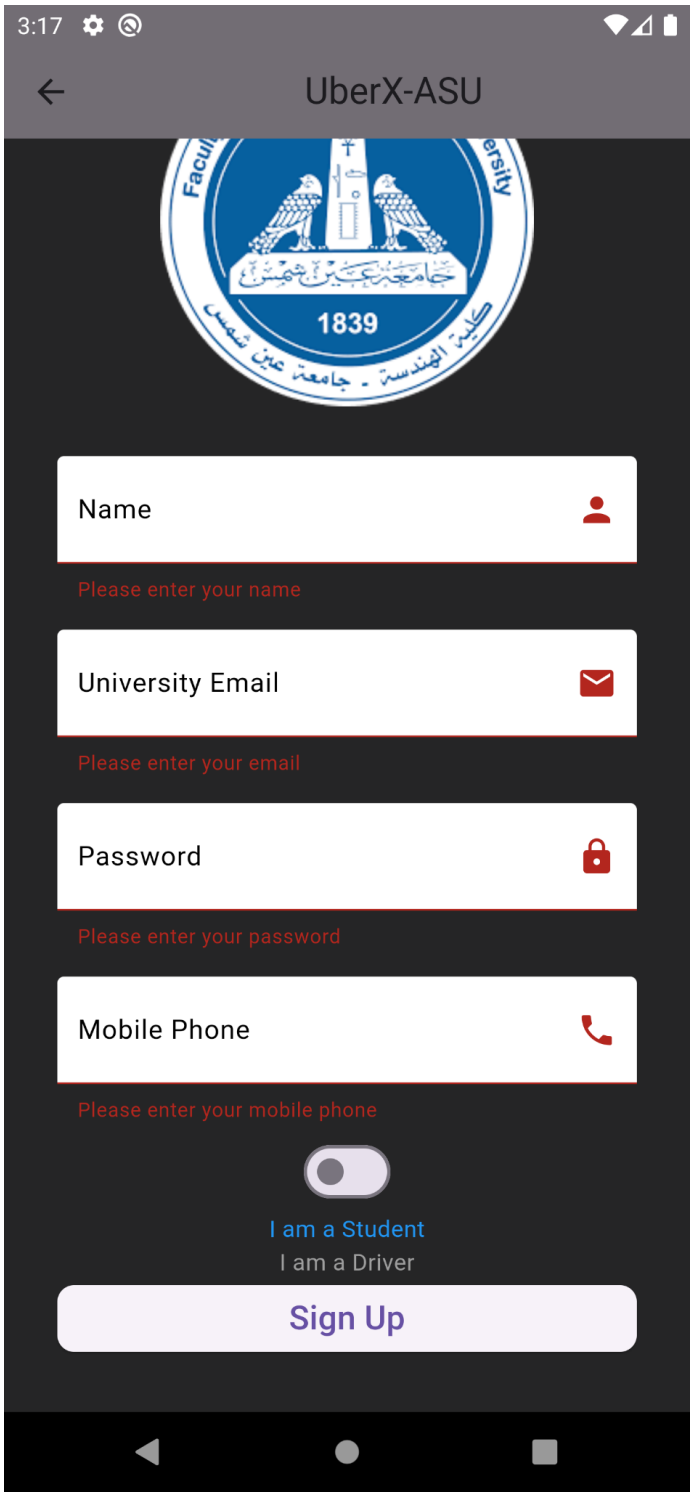
2) add trip


3:01
Add Trip
Driver Name
Morcous
Fee
50
Mobile Number
01556155687
Start Location
Matrya
End Location
gate 3
Trip Date
2023-12-22 00:00:00.000
Add Trip

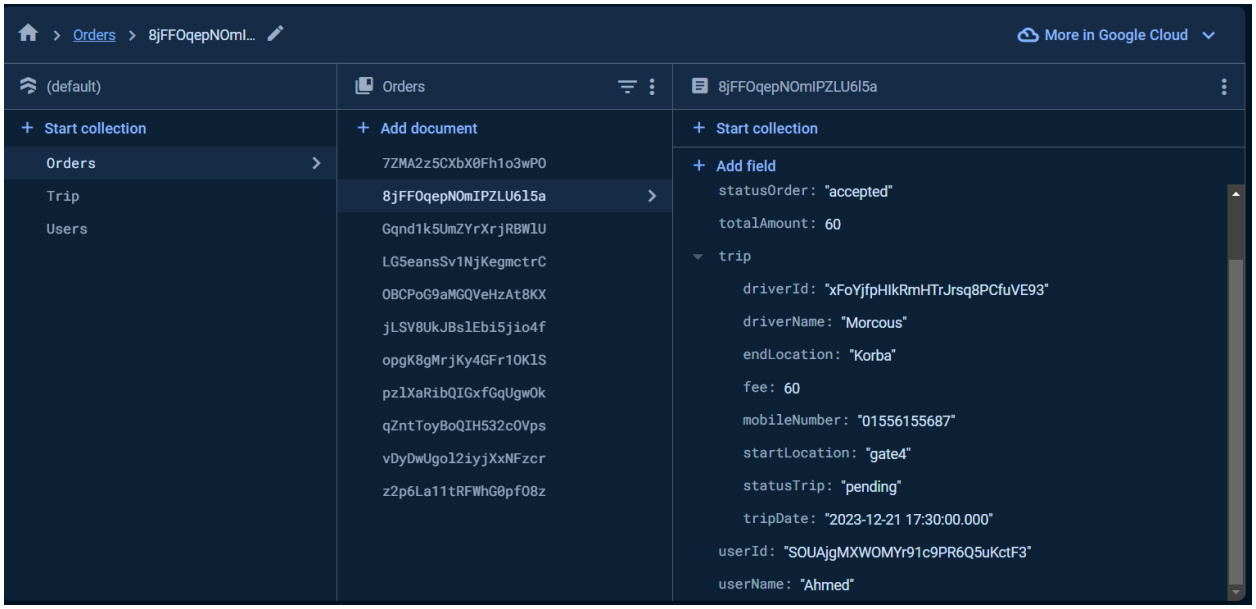
3)see my trips

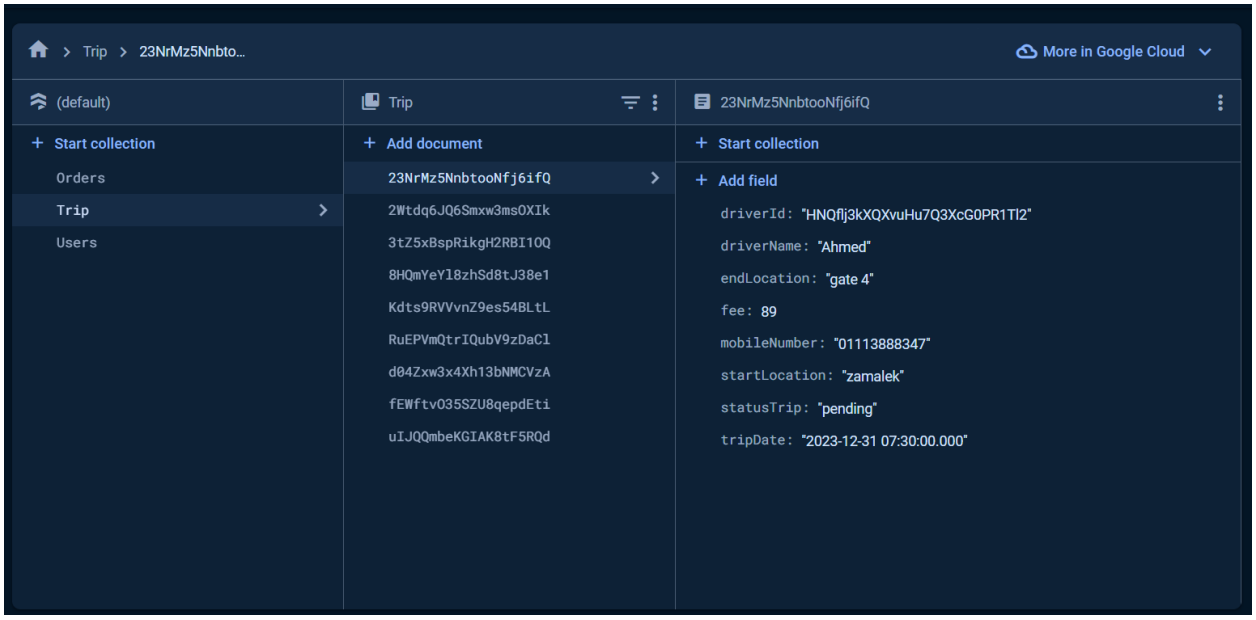


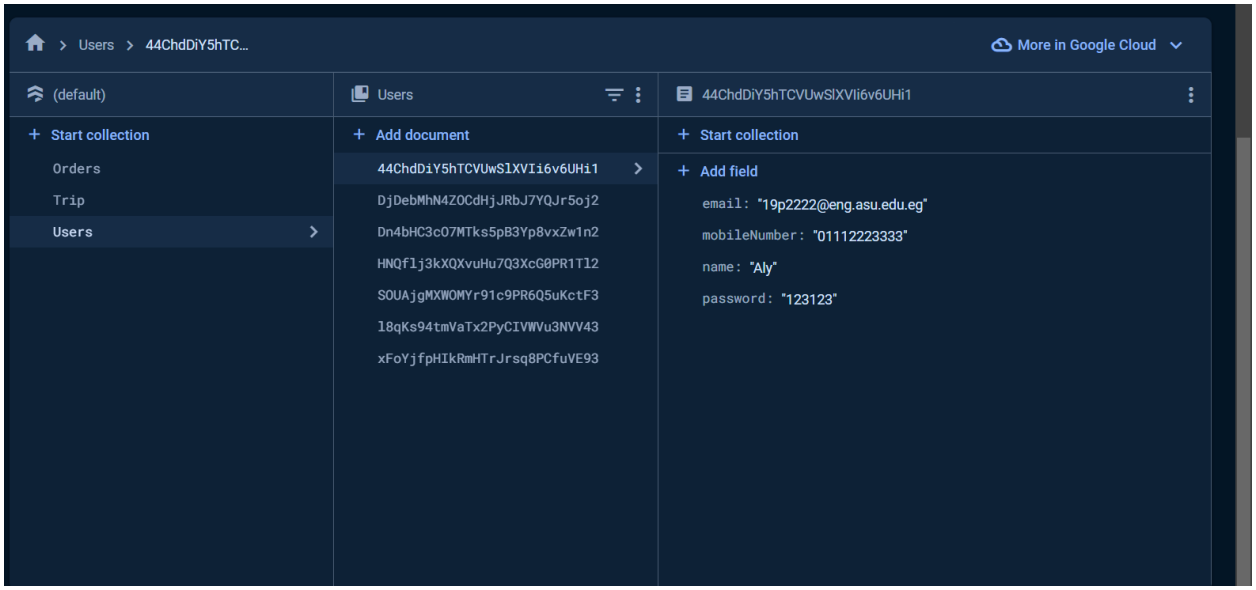
Scenario 2:
Testing signup



Database screenshots






Users
44ChdDiY5hTC...
More in Google Cloud
(default)
Start collection
Orders
Trip
Users
Users
Add document
44ChdDiY5hTCVUwSlXVIi6v6UHi1
DjDebMhN4ZOCdHjJRbJ7YQJr5oj2
Dn4bHC3cO7MTks5pB3Yp8vxZw1n2
HNQflj3kXQXvuHu7Q3XcG0PR1Tl2
SOUAjgMXWOMYr91c9PR6Q5uKctF3
l8qKs94tmVaTx2PyCIVWVu3NVV43
xFoYjfpHIkRmHTrJrsq8PCfuVE93
44ChdDiY5hTCVUwSlXVIi6v6UHi1
Start collection
Add field
email: "19p2222@eng.asu.edu.eg"
mobileNumber: "01112223333"
name: "Aly"
password: "123123"