Guuber

Design Document

Lunwen He: lunwenh, lunwenh@andrew.cmu.edu
Yanning Liu: yanningl, yanningl@andrew.cmu.edu
Ziming Wang: zimingw, zimingw@andrew.cmu.edu

1 Purpose of this document.

This is the design document of Guuber. In section 2, we will first describe our system's architecture which is CS(client-server) based. In section 3, our detailed code structure and class diagrams of client and server are discussed in detail. In section 4, a global view of our intent classes among all activities is provided. In section 5, we will discuss our exception handling. In section 6, we will talk about our database schema about both server and client. In section 7, we will show our requirement analysis of screen design. In the last section, screen flows for each use case are provided.

2 System architecture

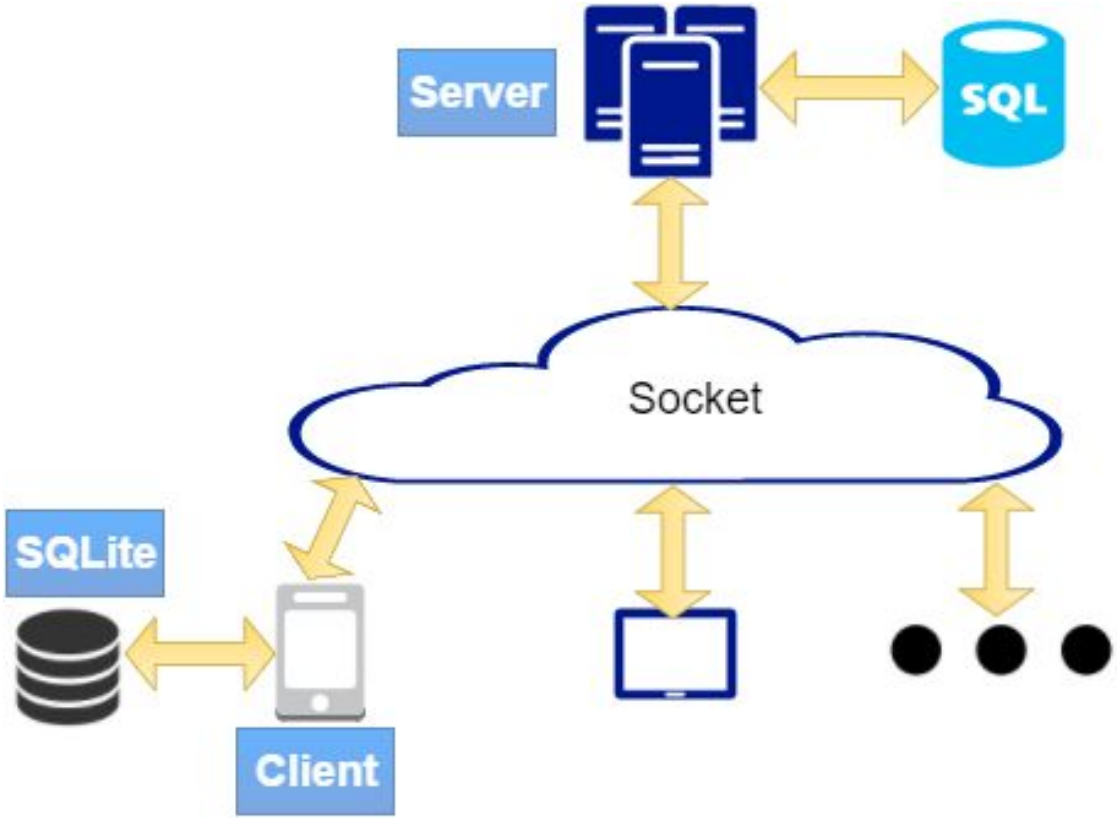


Figure 1.

Figure 1 is a general view of our system architecture which is based on client-server. We have different kinds of clients like cell phones, tablets, etc. Each client can talk to remote server through socket. We have database both in client and server. Message history and transactions are stored in client. User information are stored in server. There are several reasons for this choice. The most important reason is to reduce the workload of server. Server is mainly response for validating user's information and communication between users. For example, when a new user signs up in our system, server needs to validate the information submitted

from client. Another example is that when two users are trying to communicate, server need to transfer messages between these two clients. The second reason is that only user information are used globally. Message history and transactions are user specific and they do not need to be hold in server. In client and server, we use SQLite and MySQL as database respectively. The details about database schema will be discussed in section 6.

3 Code structure and class diagram

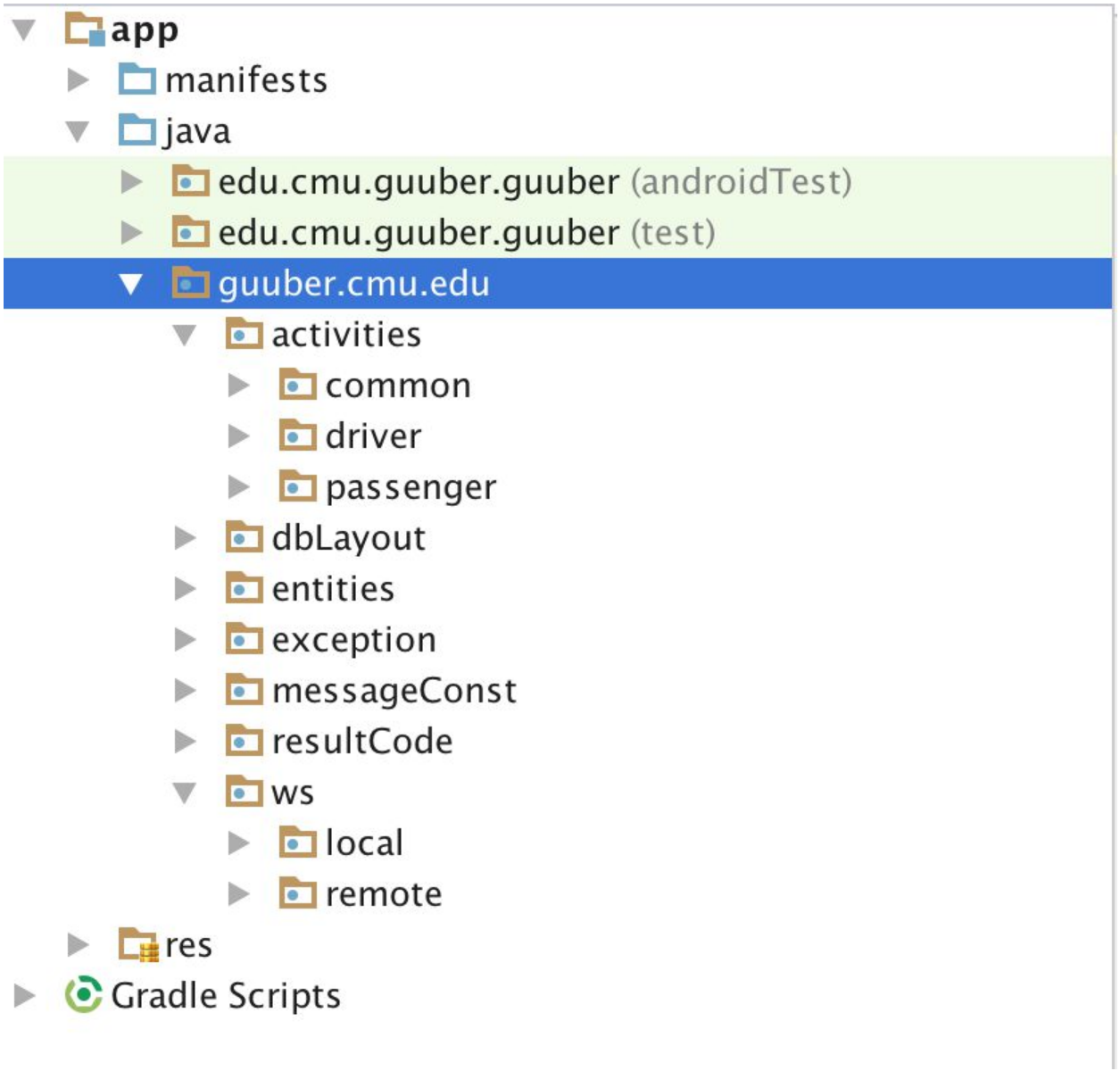


Figure 2. Client code structure

Figure 2 shows the code structure of client. In package activities, we define all activities which are responsible for interacting with users. Under that, we divide all activities into three categories: common, driver and passenger. In common package, we define all common

activities which can be shared between driver and passenger. In driver and passenger package, we have activities specific for driver and passenger respectively. Package dbLayout holds all classes related to client database models, which defines the schema of database tables, and database controllers, which provide CRUD operations to client database. Package entities contains all entities that are used to store or transfer information between client and server like user, message, transaction. In package exception, we create our customized exception handling and logging. For example, when a user tries to sign in with invalid username and password, it will create a SinInException. The detail of exception will be talked in the last section. In package ws, we have our web service related interfaces, like remote server connection, Google Map API, which are further divided into remote and local. Under remote package, we have GuuberService which is a background service connecting with our server through socket, sending and receiving messages to/from server. Under local package, we have all handlers which interacts with local database services and provide CRUD operations to all activities. In package messageConst, we defines all operations of background GuuberService, activities who need GuuberSevice to send/receive messages and all message types server and client can handle.

Following are the detailed class diagram for client.

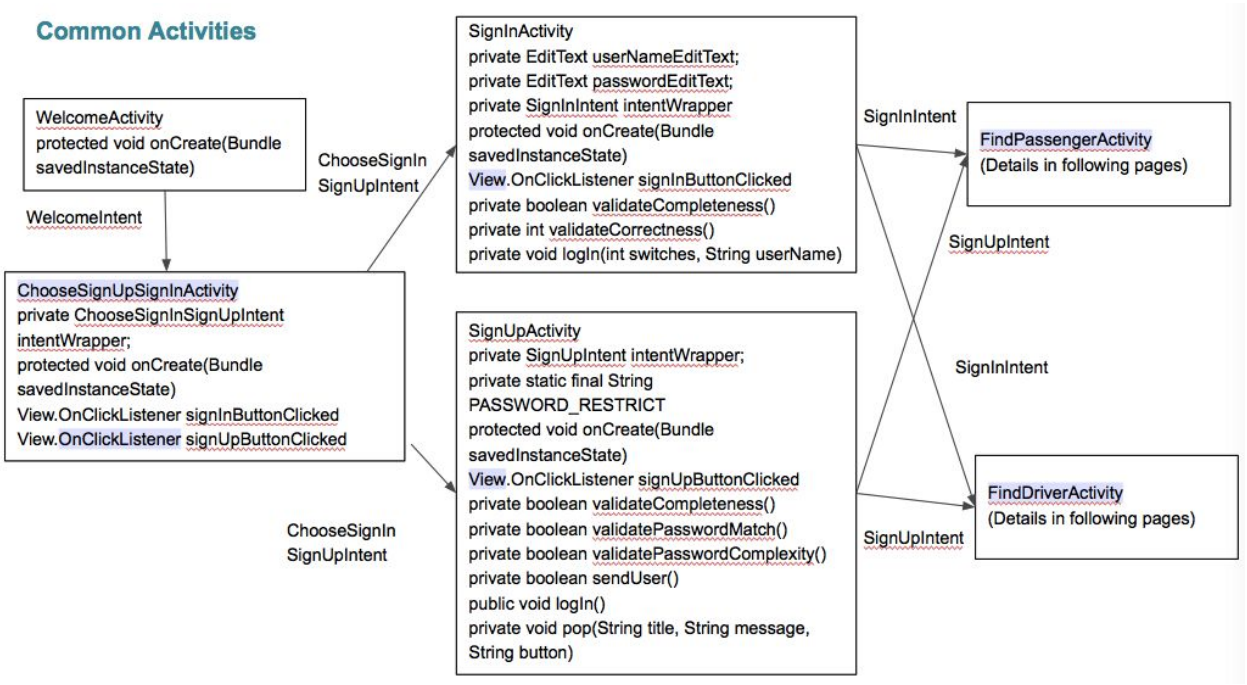


Figure 3. Common activities

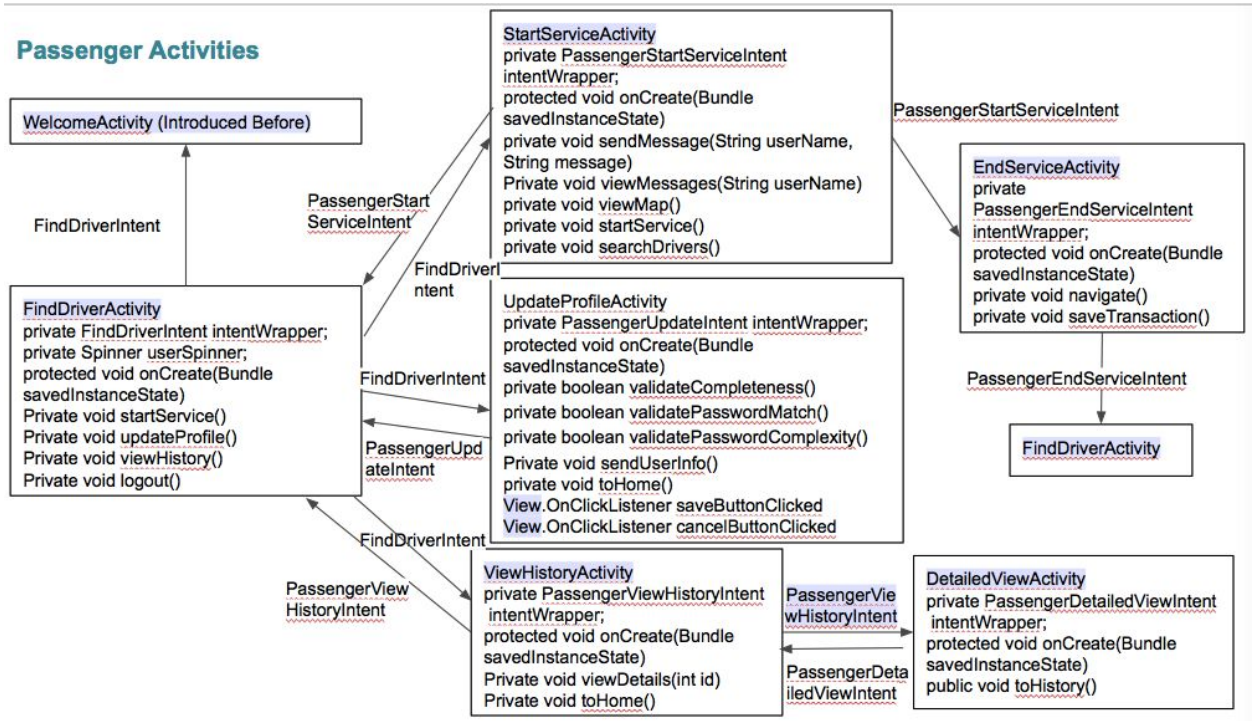


Figure 4. Passenger activities

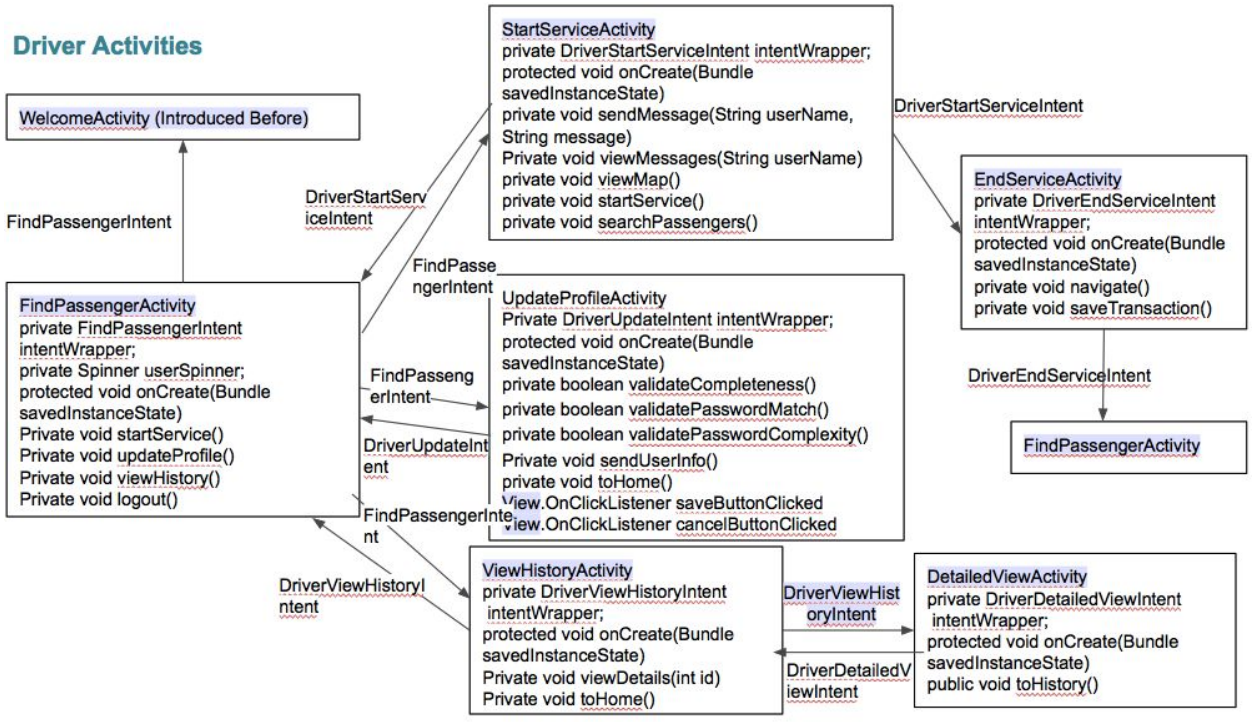


Figure 5. Driver activities

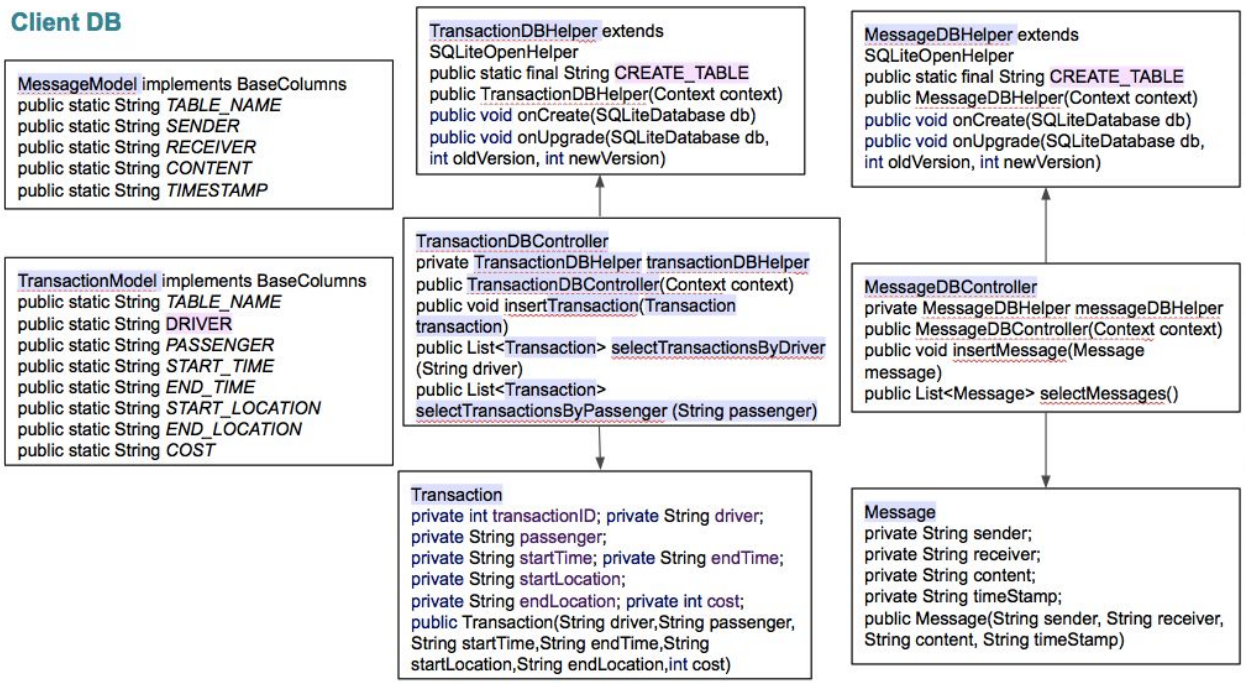


Figure 6. Client DB

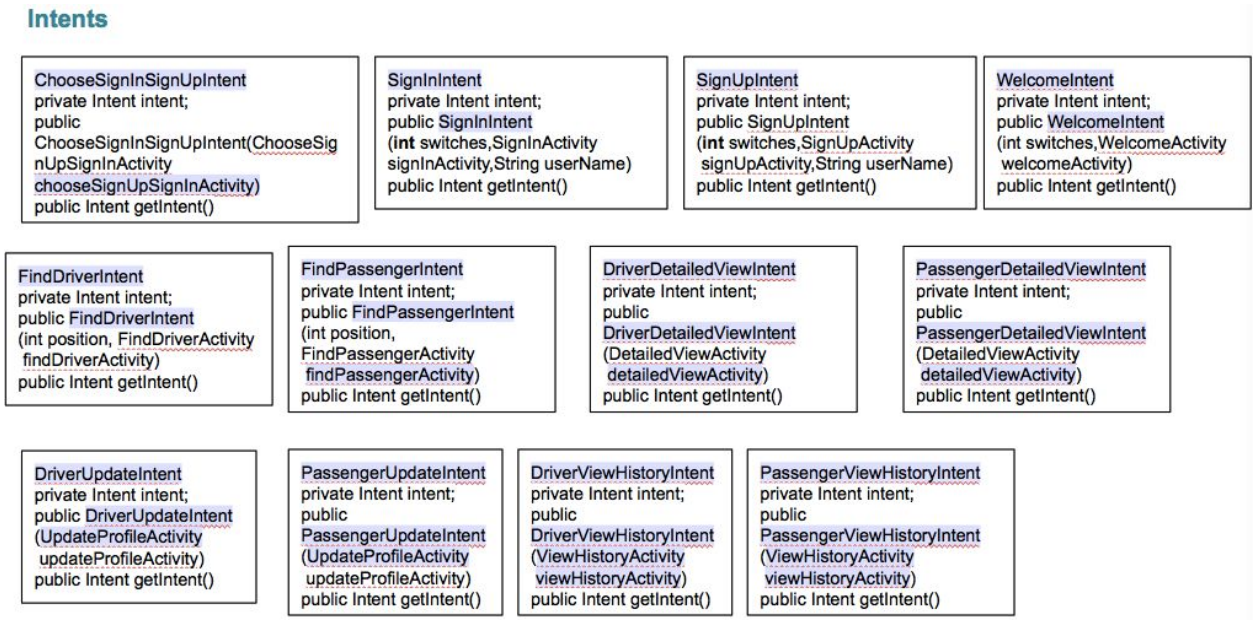


Figure 7. Intents

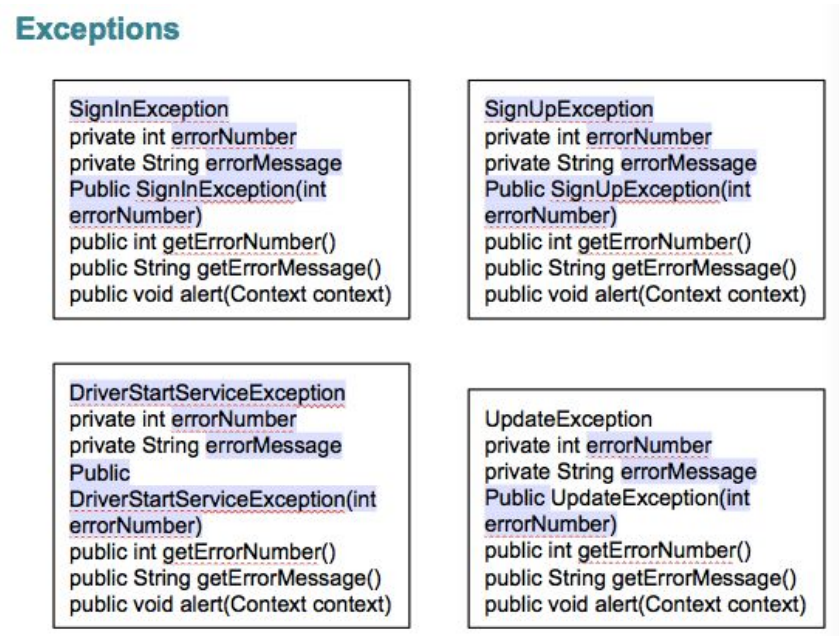


Figure 8. Exceptions

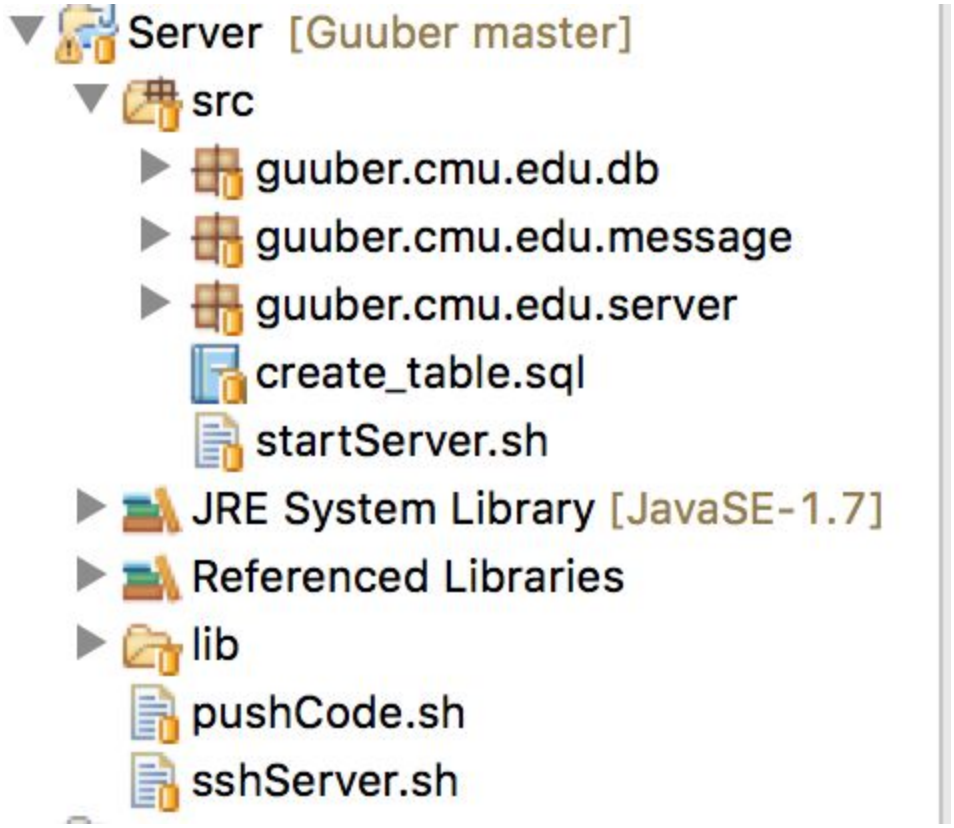


Figure 9. Server code structure

Figure 9 shows the code structure in server. There are three packages in server. Package db contains the definition and controller classes for server database. Package server holds classes responsible for interacting with client. Package message contains all message types server and client can handle.

Following is the class diagram for server.

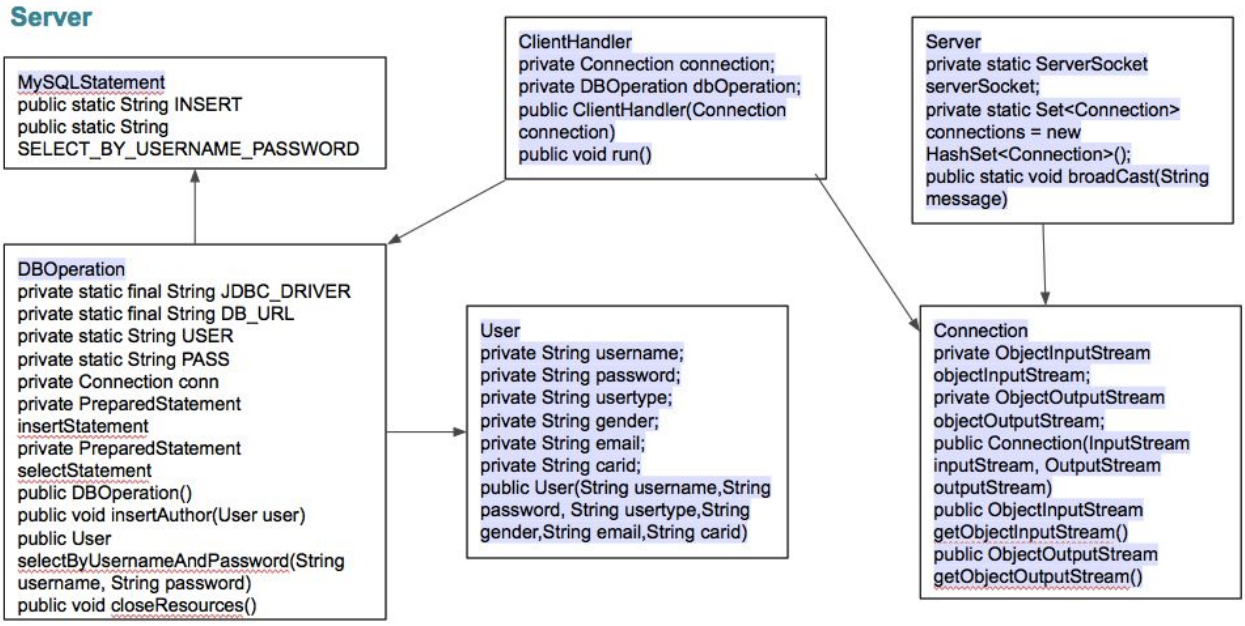


Figure 10. Server

4 Global view of intent

The following is a global view of our intent classes between different activities. Each arrow represents an intent which starts another activity. It also describes all possible page flows in our application. For more detailed information, please refer to our activity classes.

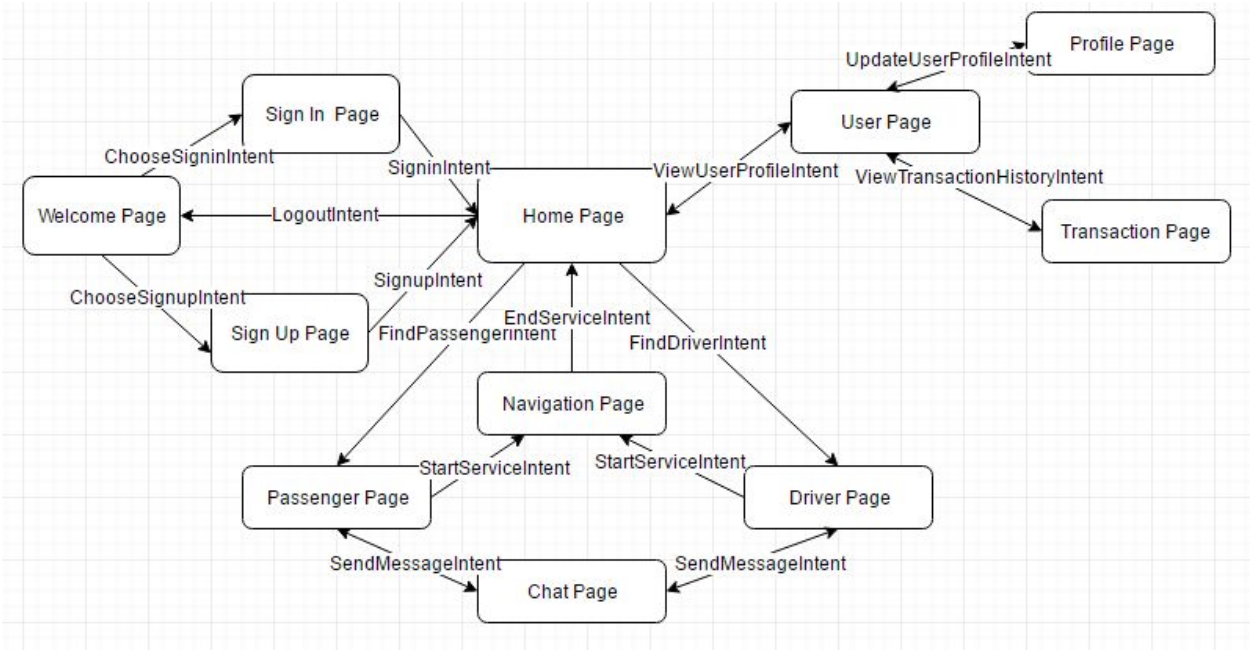


Figure 10. Global view of intent

5 Exception Handling

5.1SignInException

When sign in, exception will be thrown when a. User doesn't fill in a valid username b; user doesn't fill in a valid password; c. username and password are not a valid pair in database. Fix method is to raise an alert message to ask user to enter username and password again or choose sign up.

5.2 SignUpException

When sign up, exception will be thrown when a. User doesn't provide a valid username; b. Password doesn't meet the requirements (length, character types); c. password and retype don't match; d. user doesn't select gender; e. User doesn't select a user type; f. User doesn't provide a valid email address; g. User doesn't provide a valid car ID (for drivers); h. Username already exist in database. Fix method is to ask user to complete all the information and submit until pass all requirements.

5.3 UpdateException

When user updates profile, exception will be thrown when a. User doesn't provide a valid

username; b. Password doesn't meet the requirements (length, character types); c. password and retype don't match; d. User doesn't provide a valid email address; e. User doesn't provide a valid car ID (for drivers); f. Update on server failed. Fix method is to ask user to correct the corresponding field as indicated by alert.

5.4 Driver Start Service Exception

When a driver tries to start the service (journey), exception will be thrown when a. The driver has not selected a passenger; b. passenger hasn't enter a destination. Fix method is to select a passenger and ask the passenger to enter a destination.

6 Database schema

6.1 Server Database Schema

In server side, we store user information. The following is the schema of user_table.

Schema of user info table(user_table)

ID, username, password, salt, gender, usertype, email, carID

ID: the unique ID of user generated by database

Username: the name of user, it's unique among all users

Password: encrypted password

Salt: salt related for encrypting password

Gender: the gender of user

userType: the type of user, possible values are driver and passenger

Email: the email of user

carID: the car ID if user is driver

6.2 Client Database Schema

In client, we store messages and transactions. The following are schemas for message_table and transaction_table.

Schema of user message table(message_table)

ID, sender, receiver, timestamp, content

ID: the unique ID of message generated by database

Sender: the sender of message

Receiver: the receiver of message

Timestamp: the timestamp of message

Content: the content of message

Schema of transaction details table(transaction_table)

ID, driver, passenger, startTime, endTime, startLocation, endLocation, cost

ID: the unique ID of transaction generated by database

Driver: the driver name of driver involved in transaction

Passenger: the passenger name of passenger involved in transaction

startTime: the start time of transaction

endTime: the end time of transaction

startLocation: the start location of transaction

endLocation: the end location of transaction

Cost: the cost of transaction, the default value is 0 if there is no cost

ER model of database:

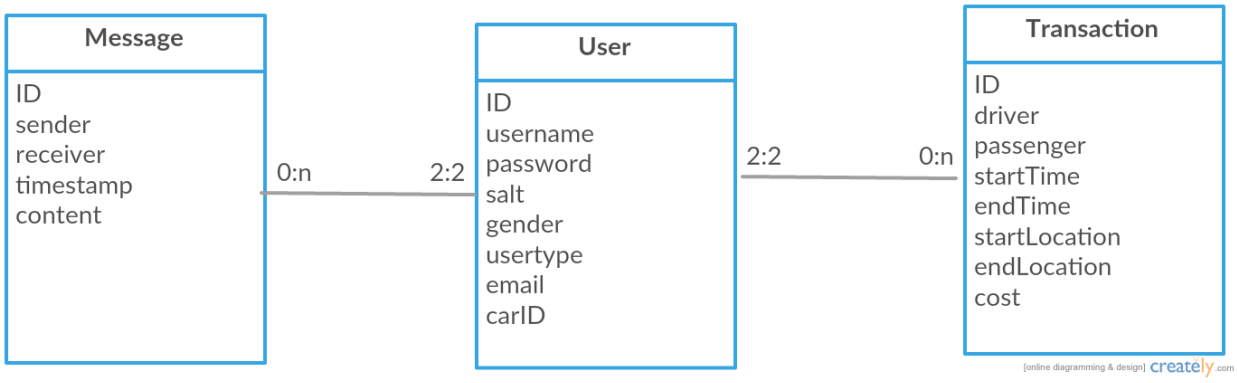


Figure 11. ER model of database

A user may involved in none or more messages.

A message can only have two users involved, one is driver, another is passenger.

A user may involved in none or more transactions.

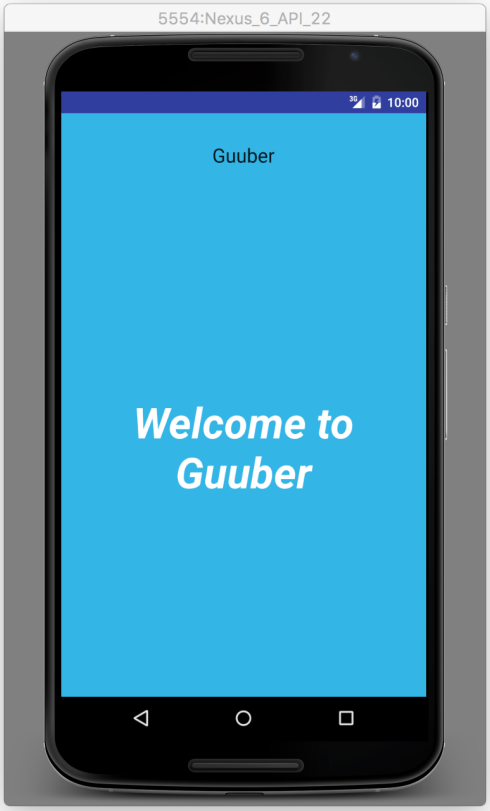
A transaction can only have two users involved, one is driver, another is passenger.

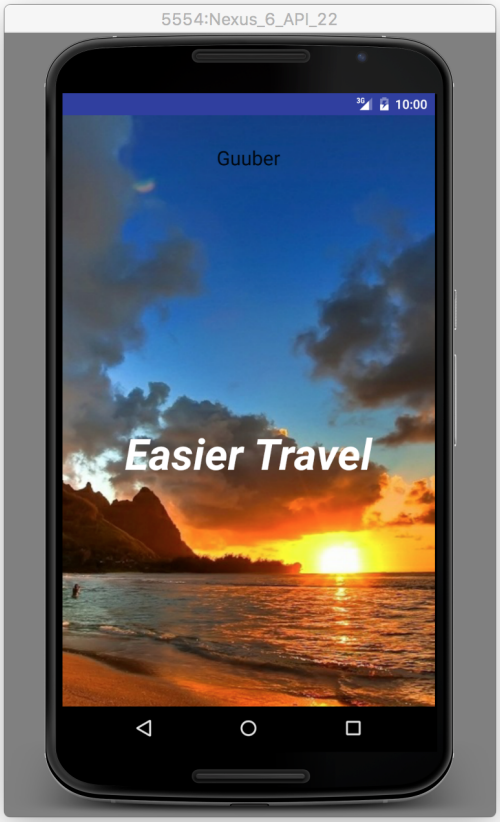
7 Screen design

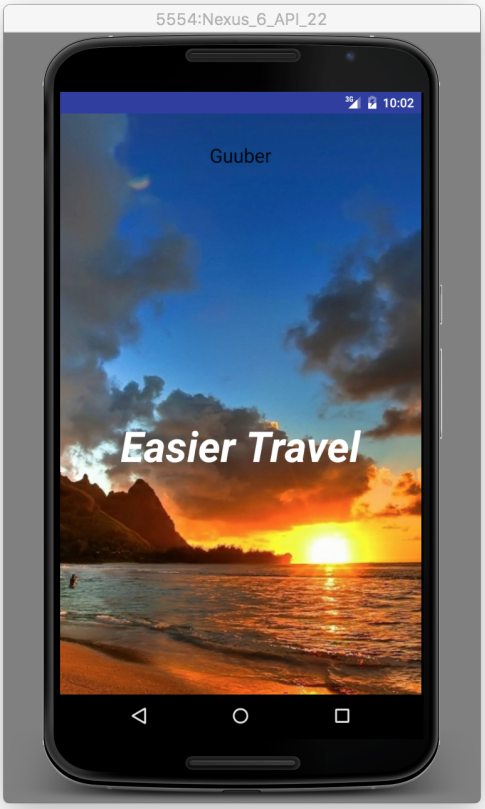
Following is our screen layout design for each screen described in our UI mockup in requirement document. We divide our screen into three categories: common screens, driver screens and passenger screens. Common screens are screens shared by both drivers and passengers like welcome page, sign-up page and sign-in page. Driver screens and passenger screens are specific to drivers and passengers respectively. For more details, please refer to our layout files.

7.1 Common screens

7.1.1 Welcome page

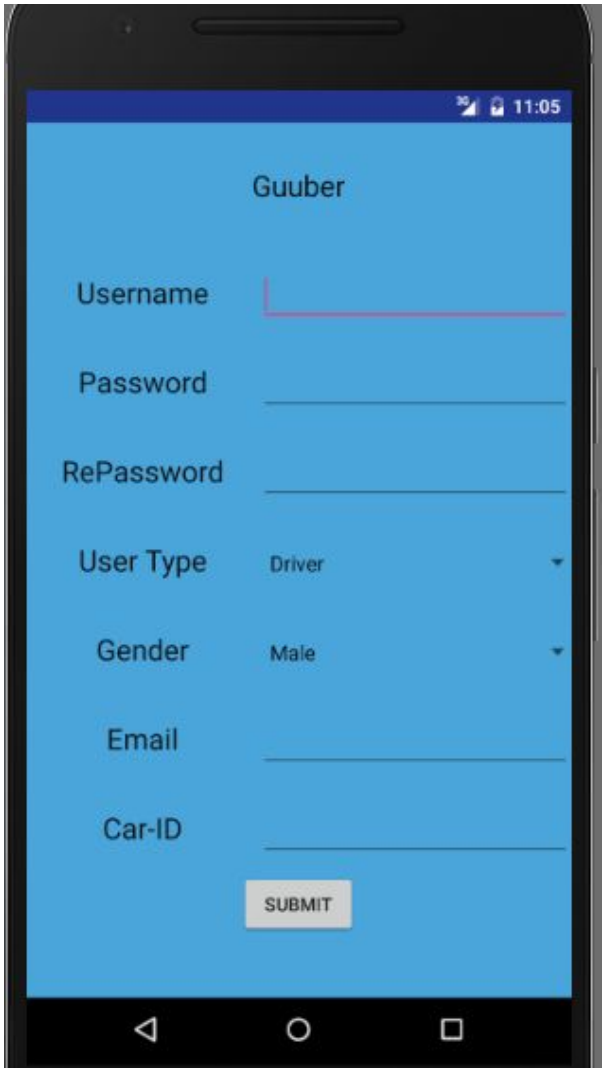




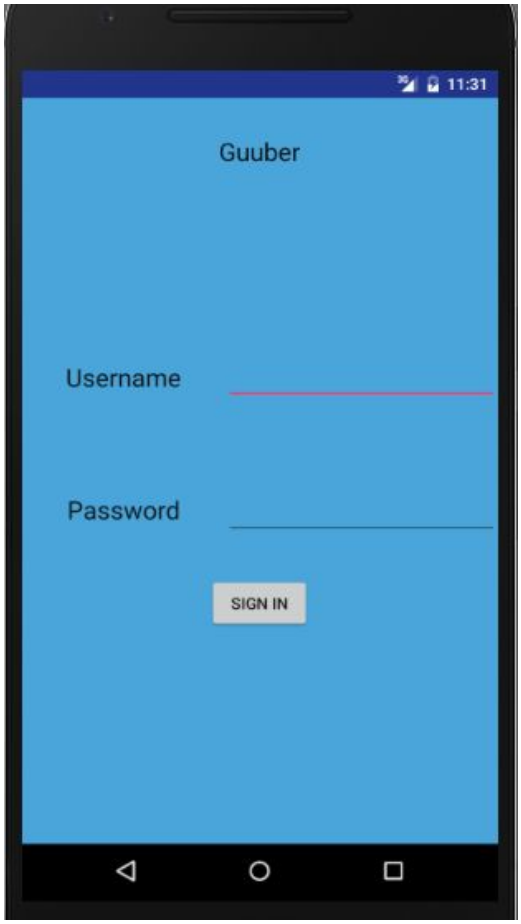

5554:Nexus_6_API_22
10:02
Guuber
Easier Travel


5554:Nexus_6_API_22
10:01
Cheaper Travel

7.1.2 Sign-up page

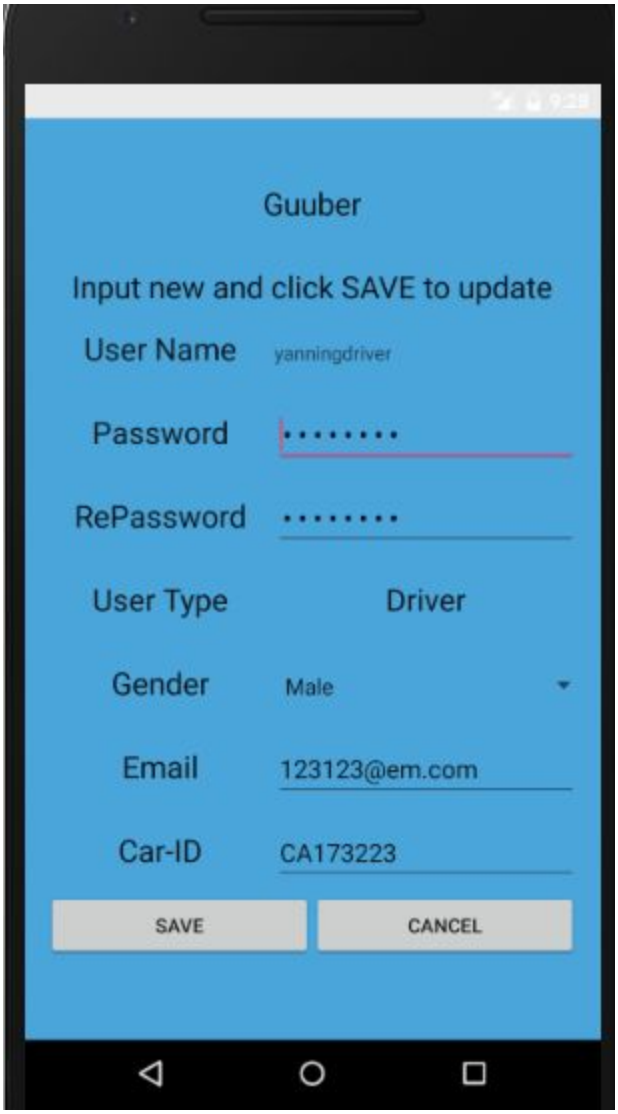


7.1.3 Sign-in page

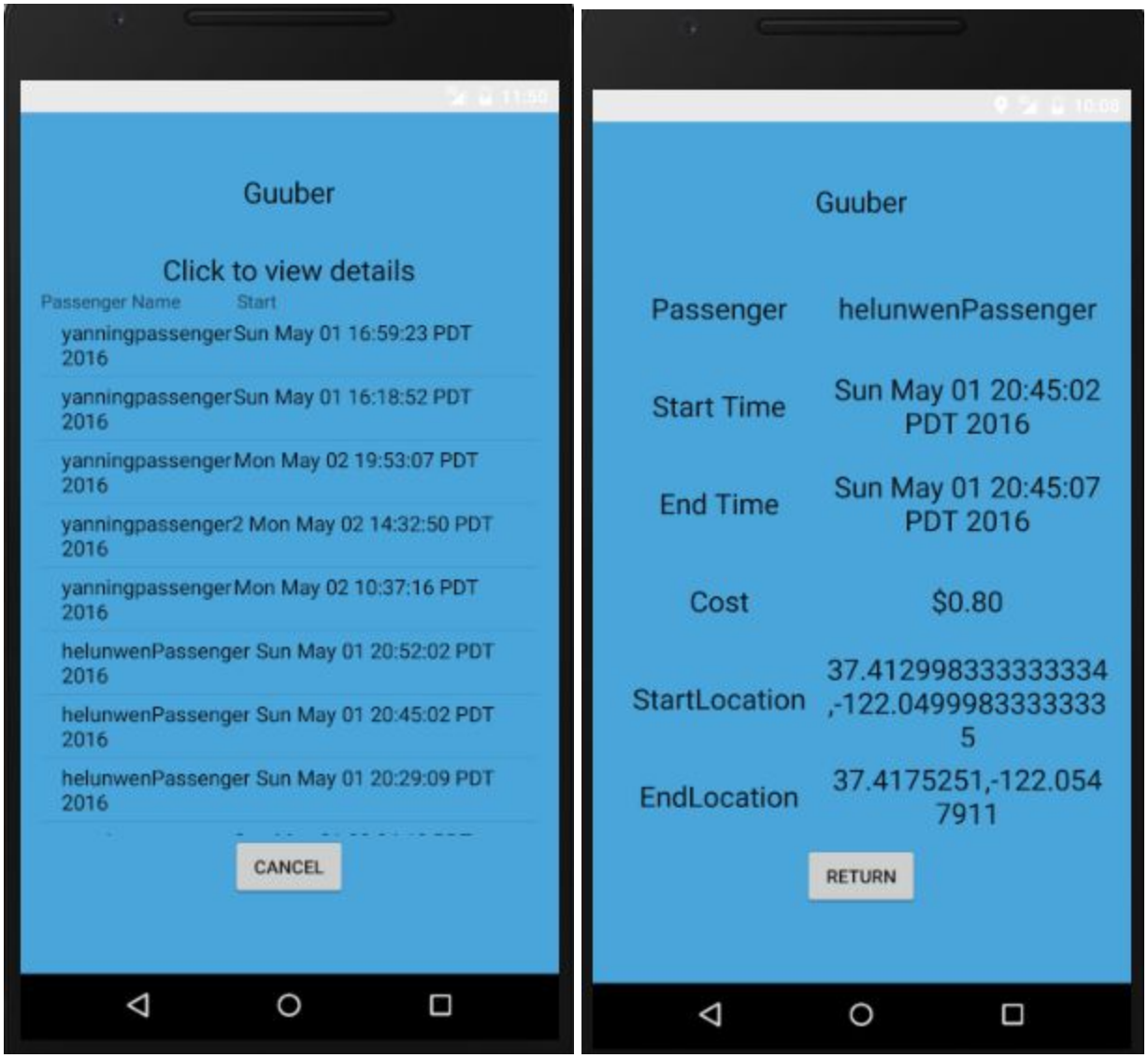


7.2 Driver screens

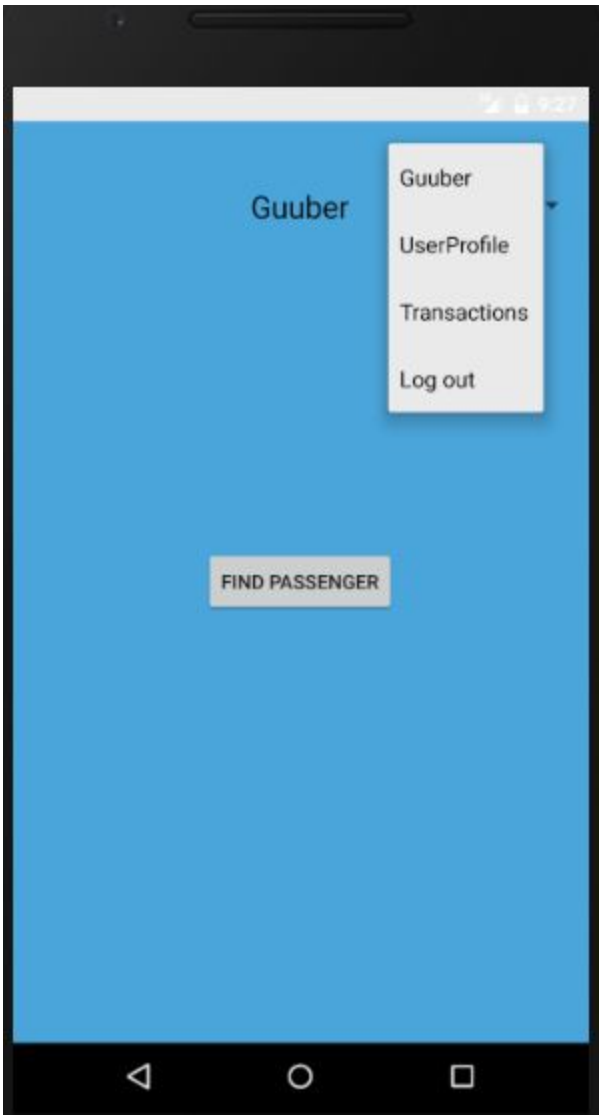
7.2.1 Drivers update profile page



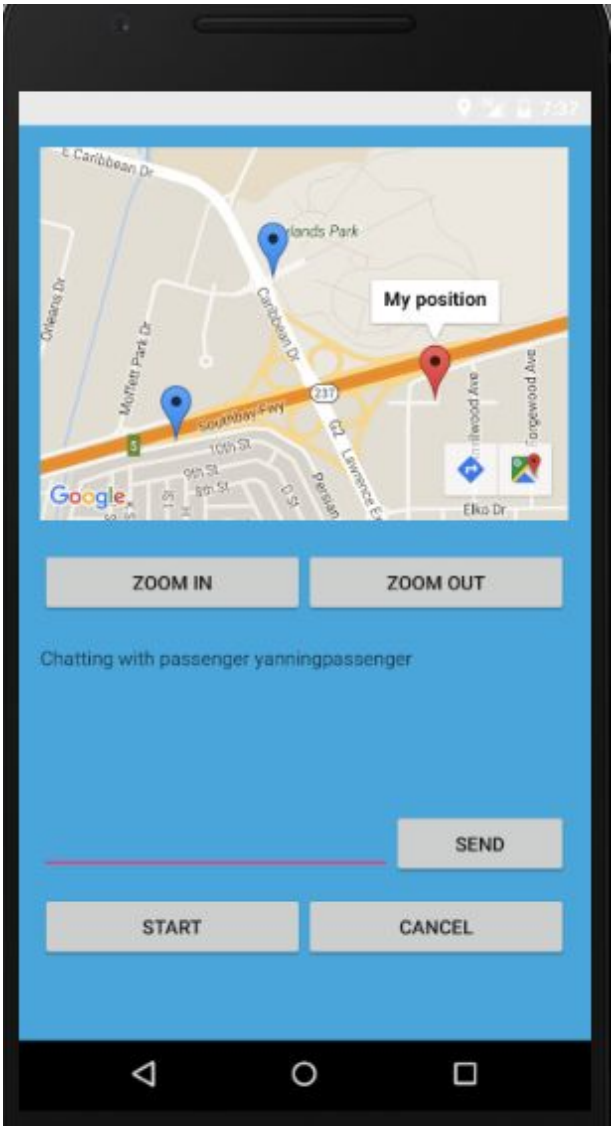
7.2.2 Drivers view transaction history page



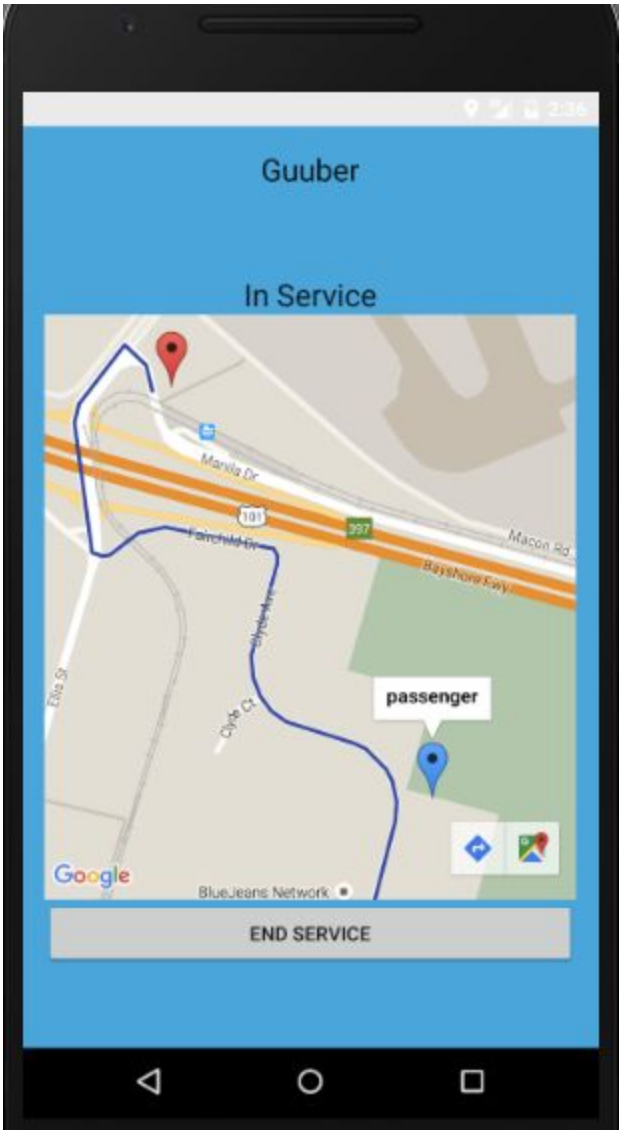
7.2.3 Drivers find passenger page



7.2.4 Drivers start service pages

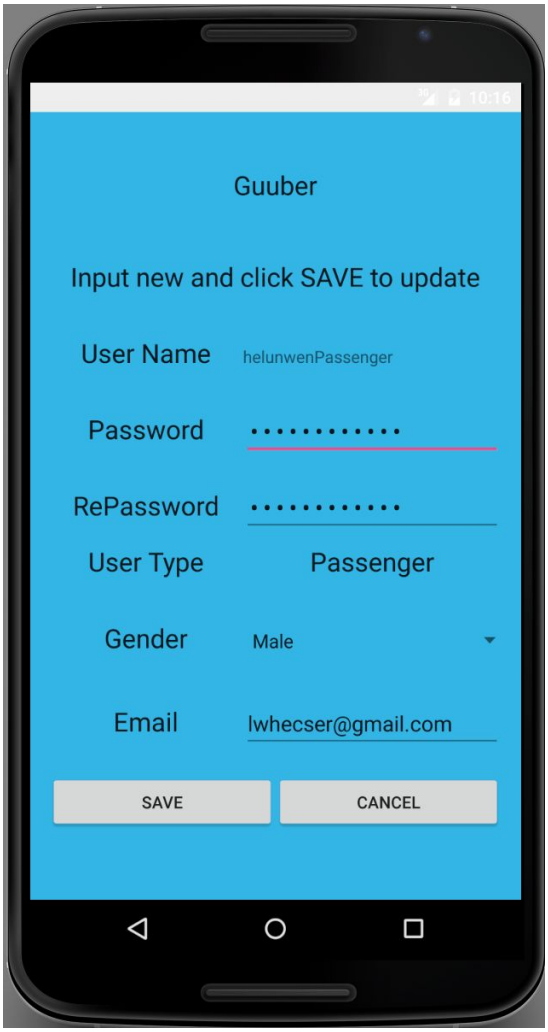


7.2.5 Drivers end service pages

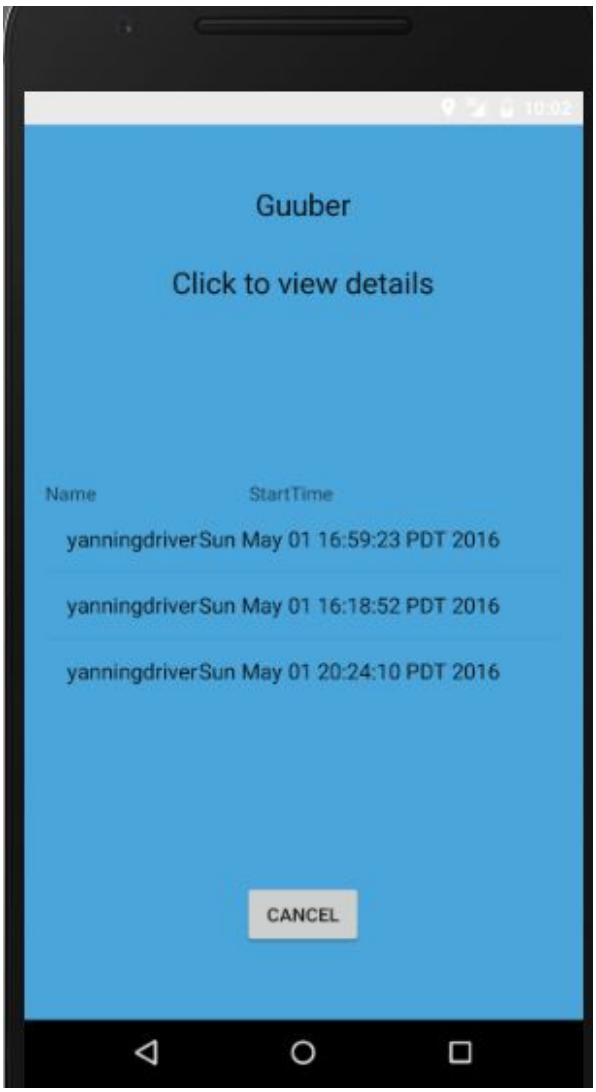


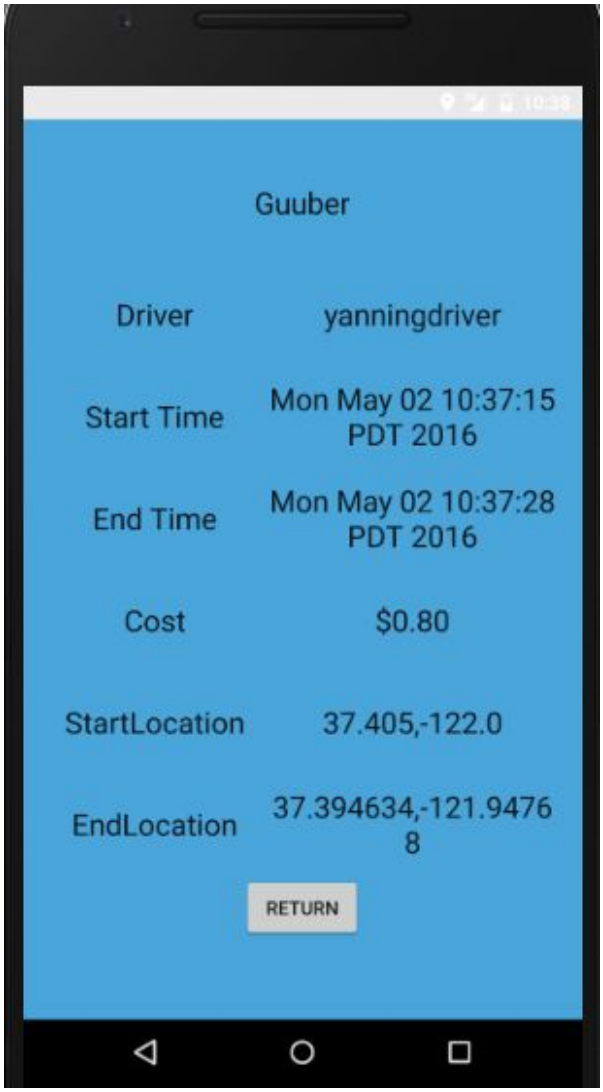
7.3 Passenger screens

7.3.1 Passengers update profile pages



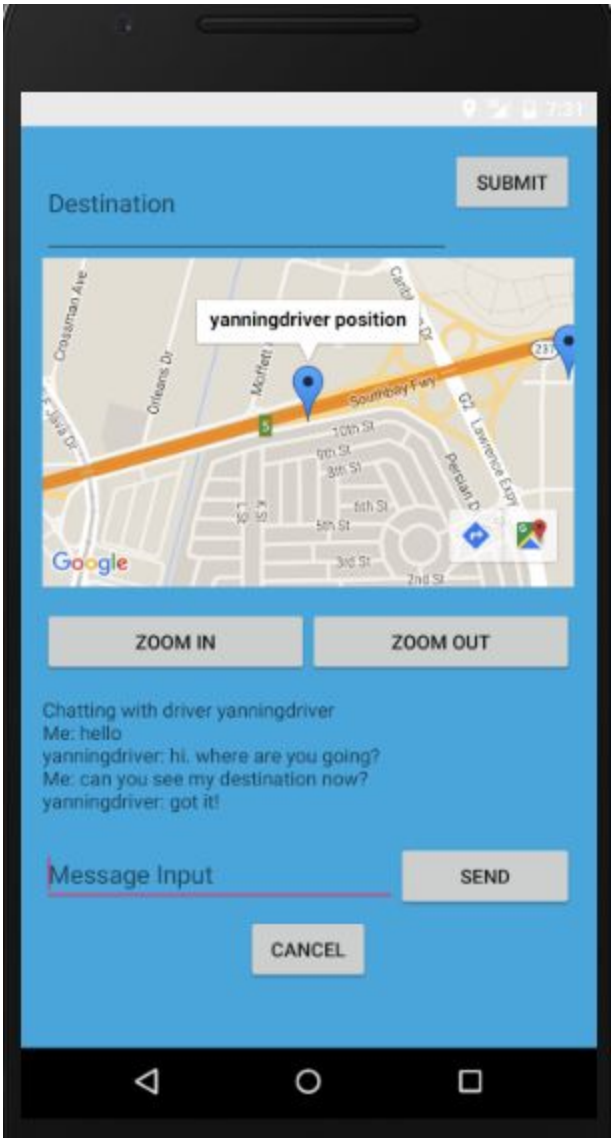
7.3.2 Passengers view transaction history pages



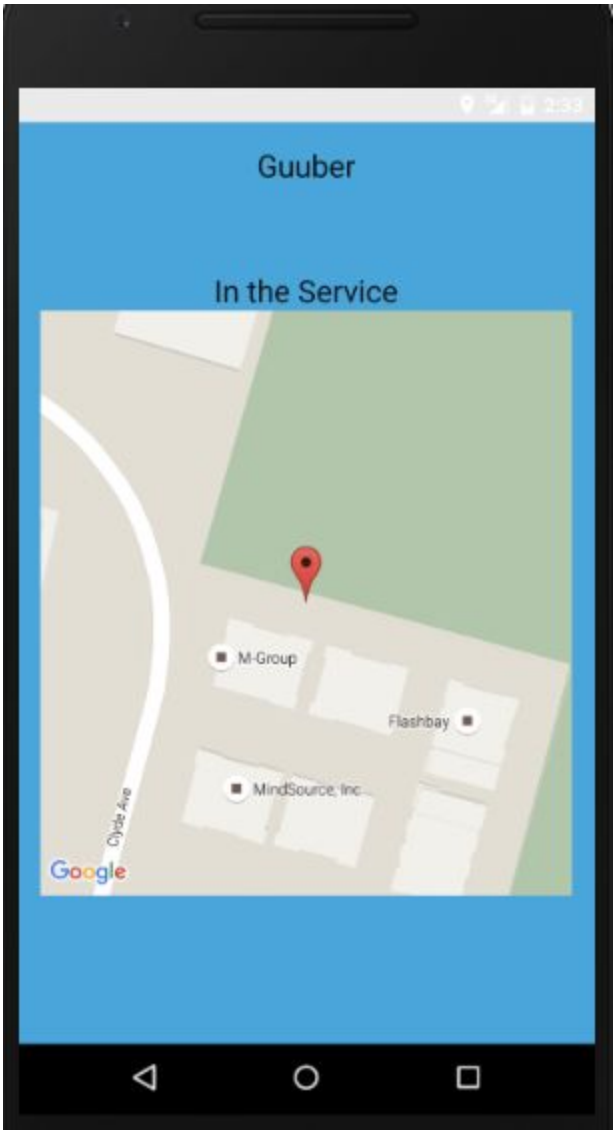


7.3.2 Passengers find driver pages

7.3.3 Passengers start service pages

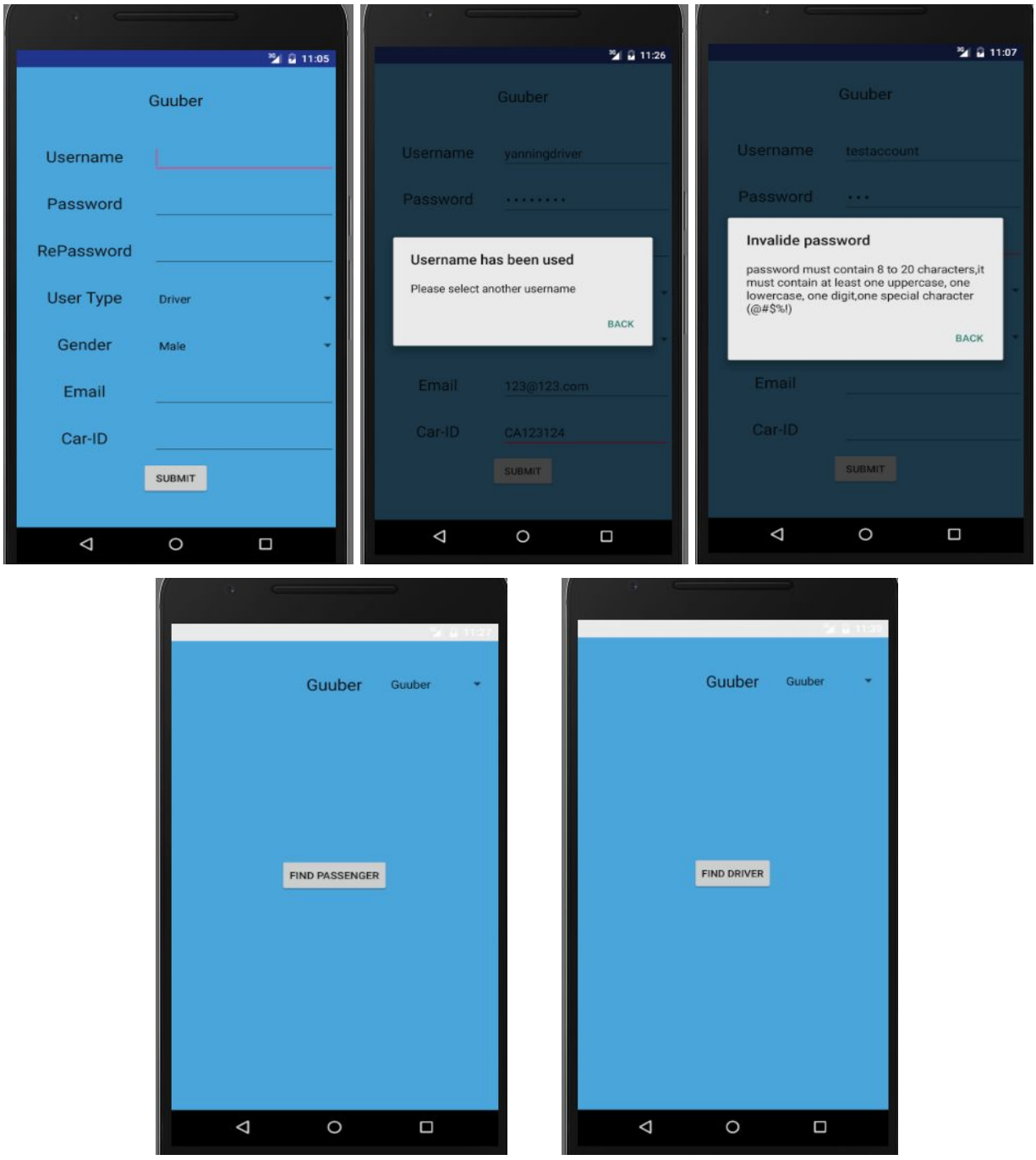


7.3.4 Passengers end service pages

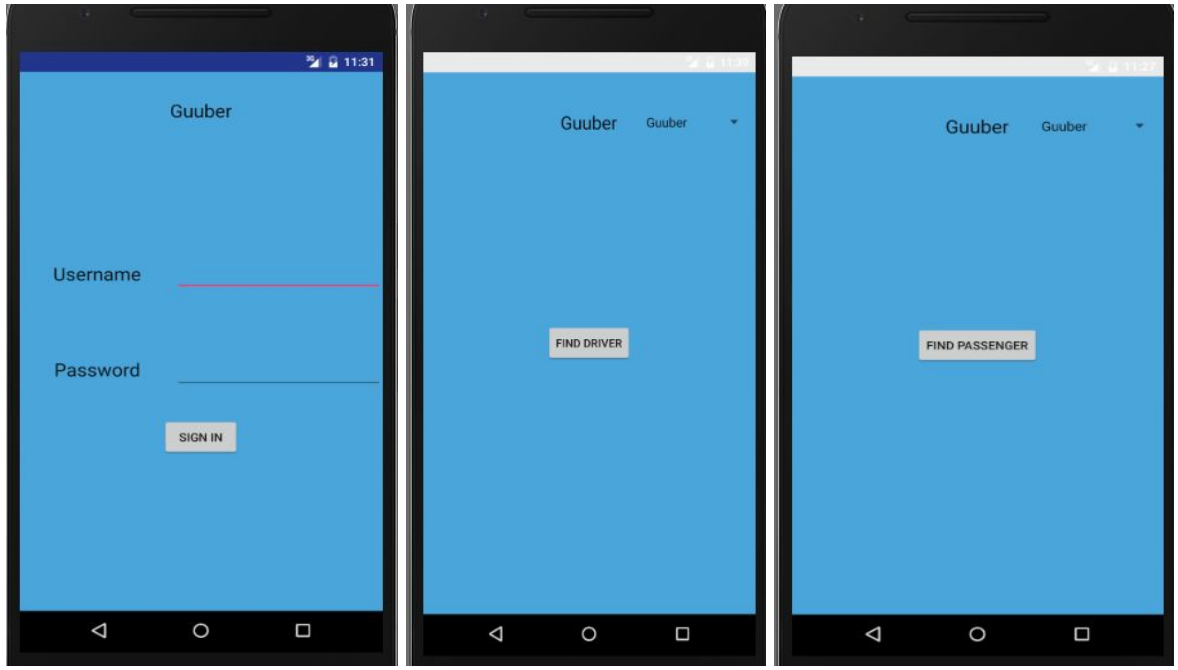


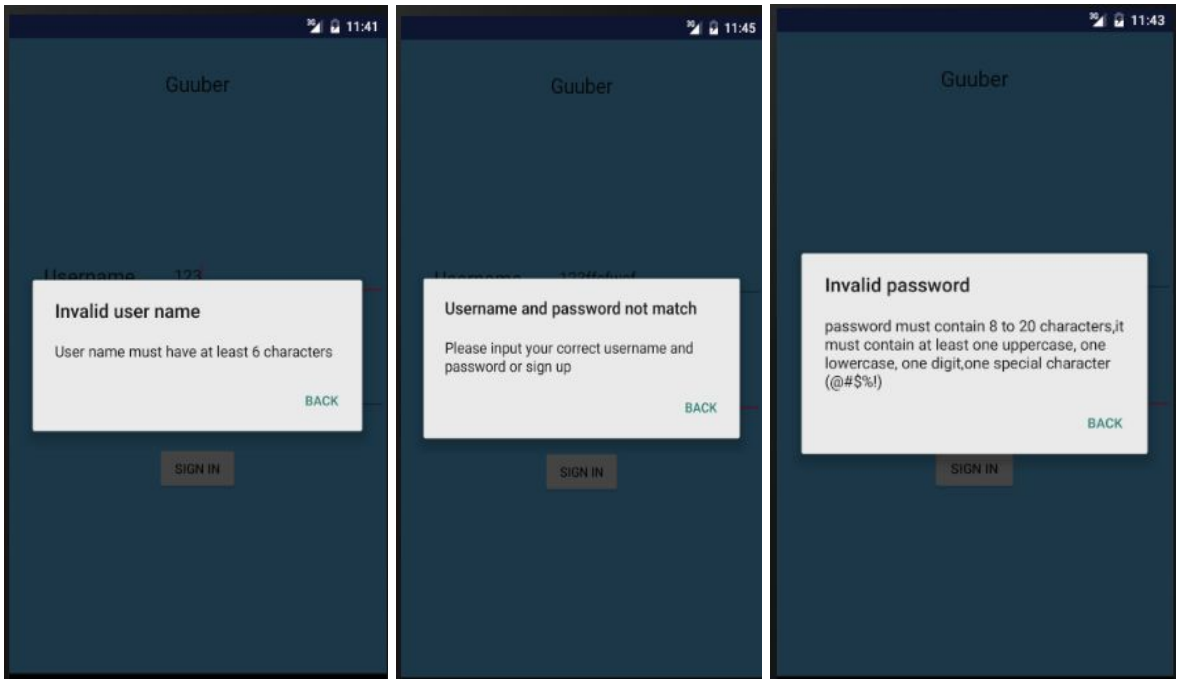
8 Screen flow of each use case

8.1 Sign-Up

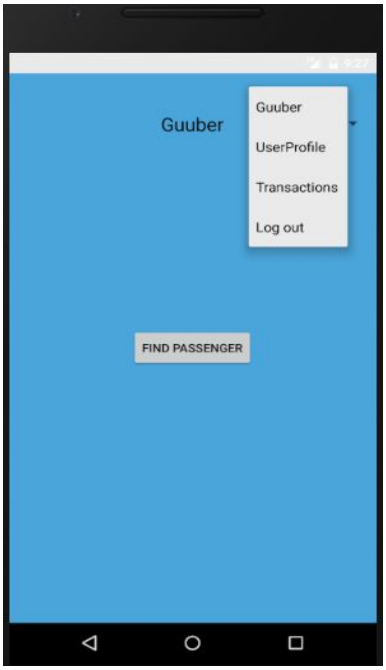


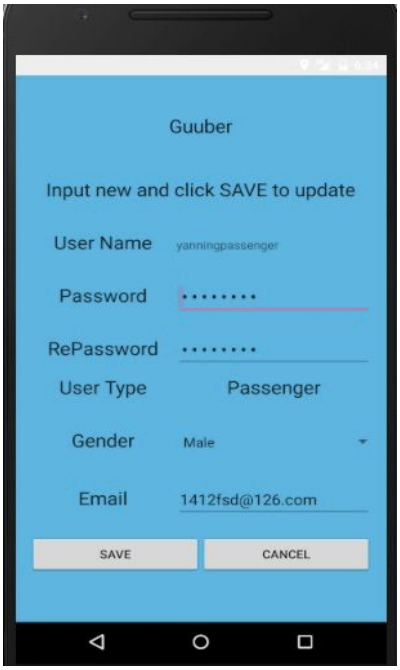
8.2 Sign-In

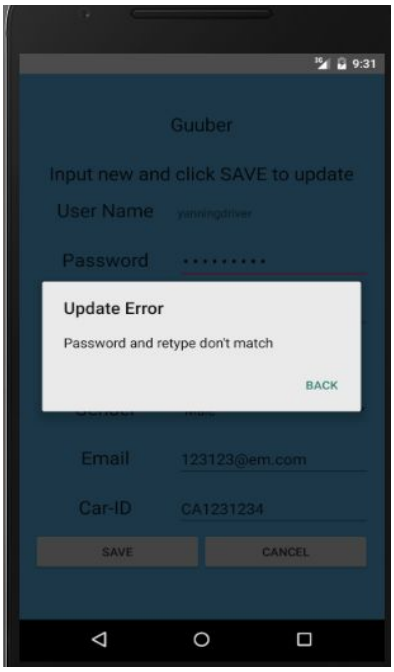

11:31
Guuber
Username
Password
SIGN IN
Guuber
Guuber
FIND DRIVER
Guuber
Guuber
FIND PASSENGER

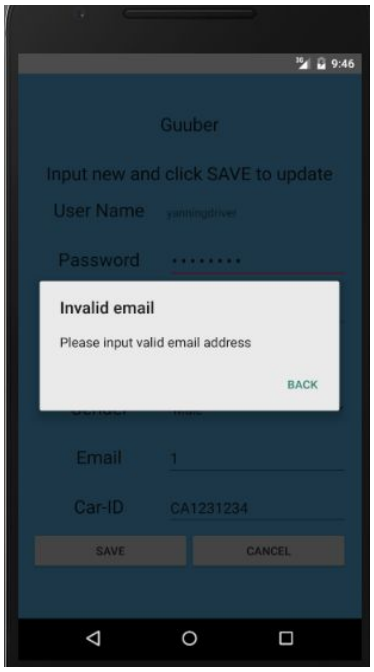

11:41
Guuber
Invalid user name
User name must have at least 6 characters
BACK
SIGN IN
11:45
Guuber
Username and password not match
Please input your correct username and password or sign up
BACK
SIGN IN
11:43
Guuber
Invalid password
password must contain 8 to 20 characters,it must contain at least one uppercase, one lowercase, one digit,one special character (@#$%!)
BACK
SIGN IN

8.3 Update Profile

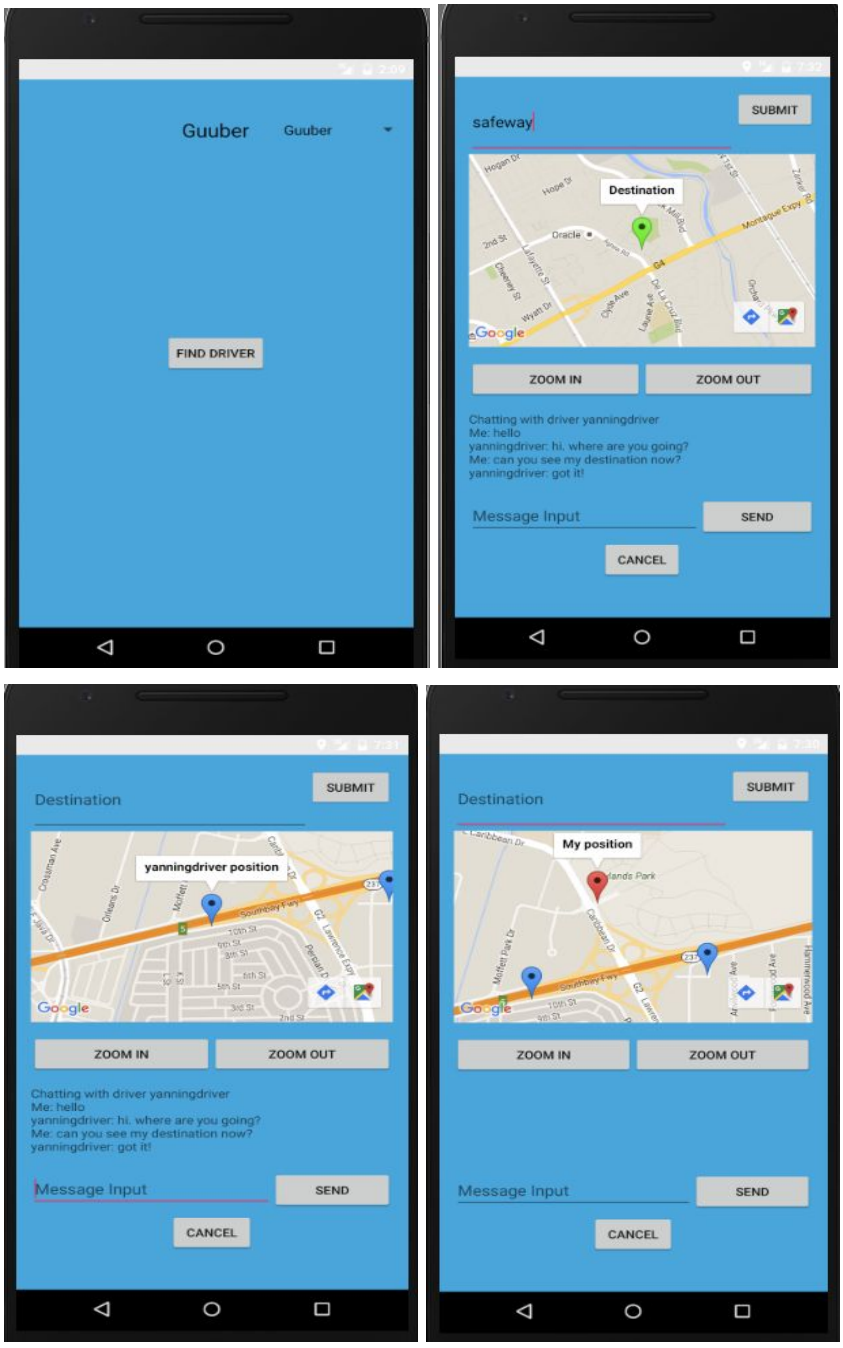




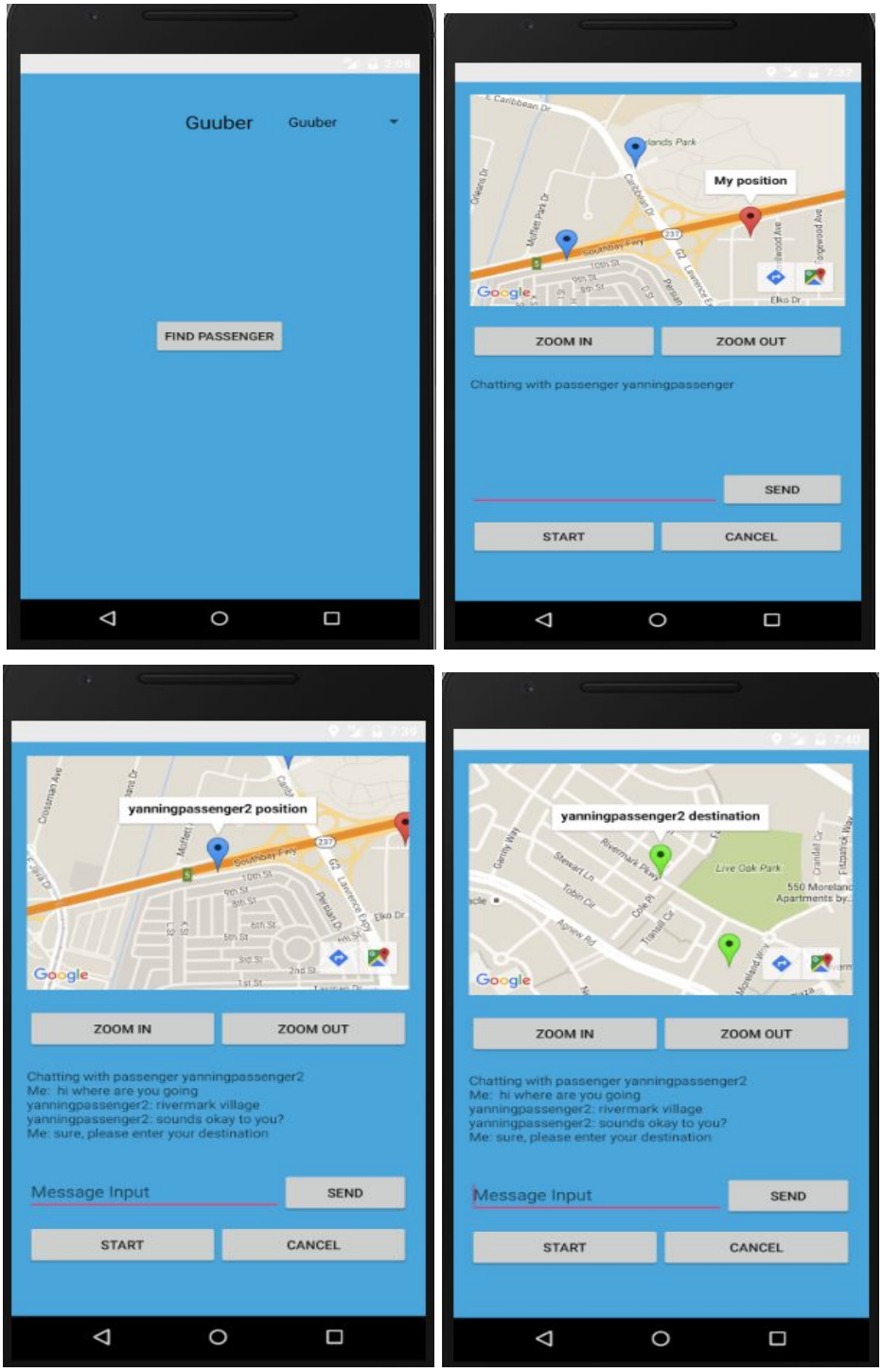




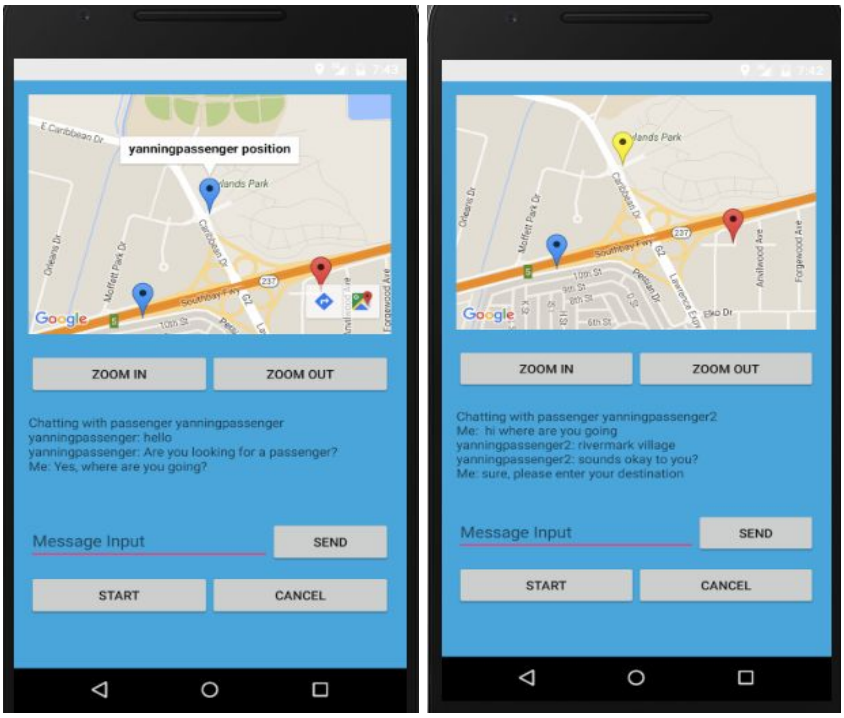
8.4 Find Driver



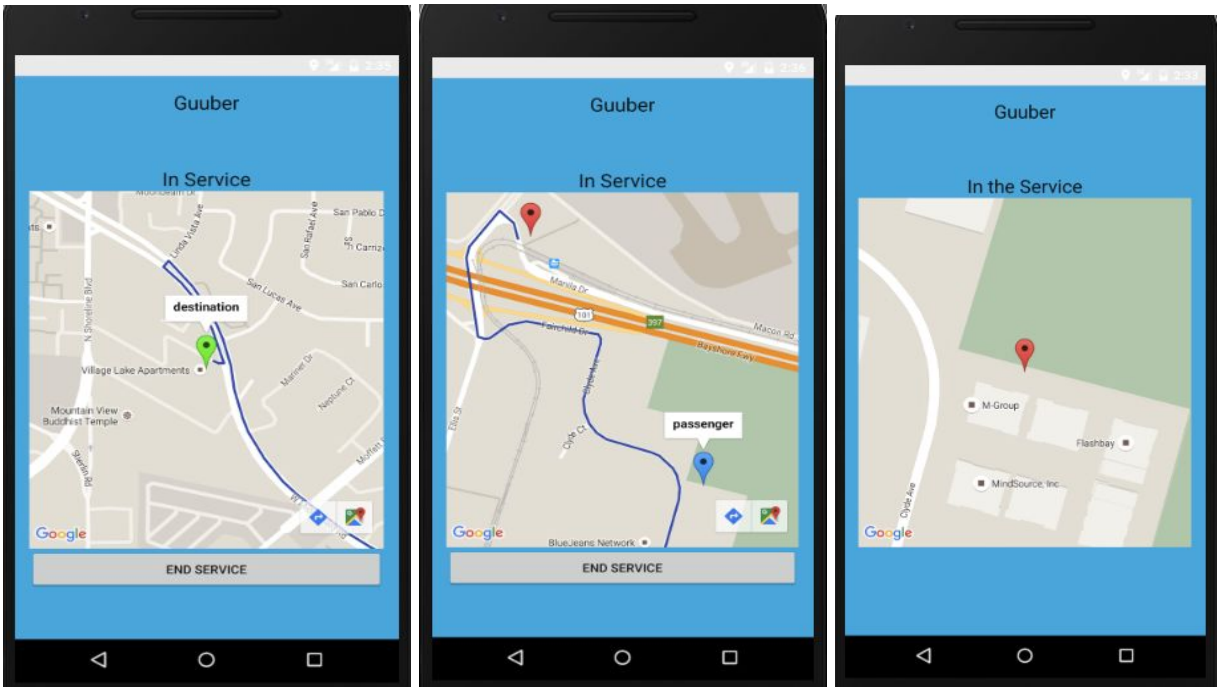
8.5 Find Passenger



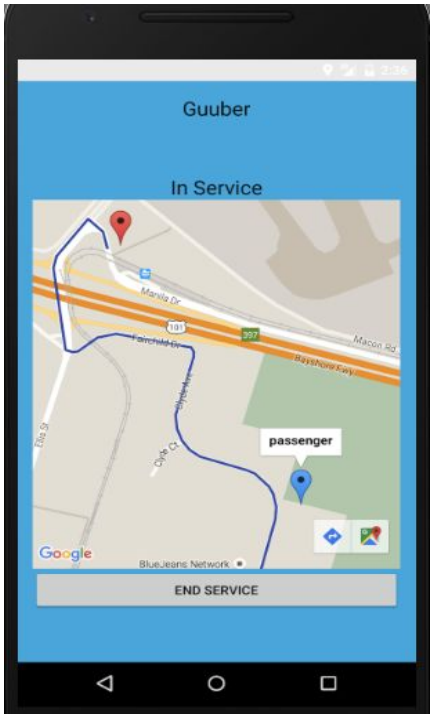
8.5 Send Message

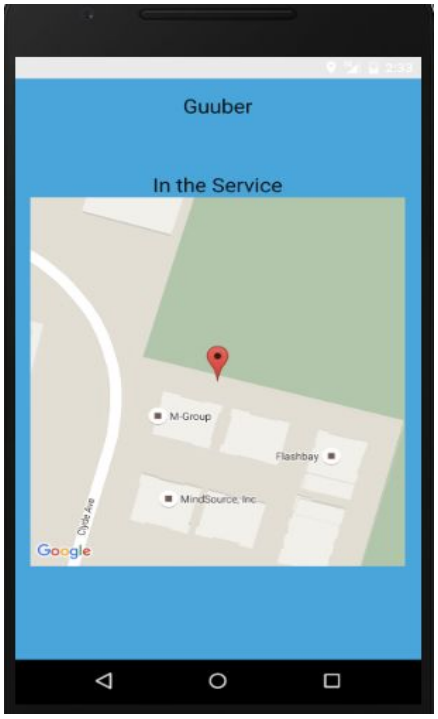


8.6 Start service



8.7 End service





8.8 View transaction history