BILKENT UNIVERSITY

FALL 2017 - CS 353

TERM PROJECT DESIGN REPORT

Private Taxi Database Management System

GROUP 12

21301027 Nergiz Ünal Sec-1

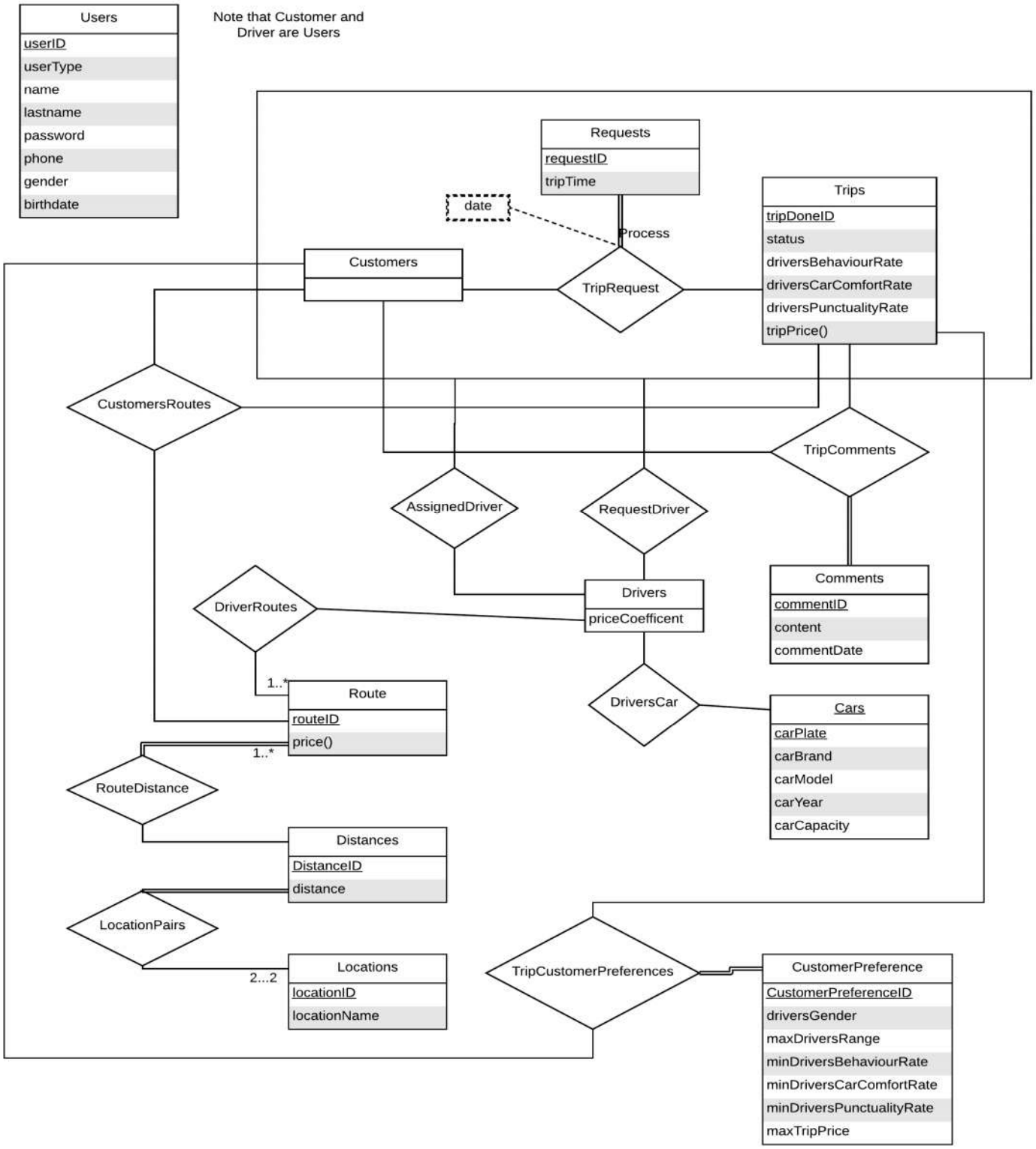
21200992 Rıdvan Çelik Sec-2

21401058 Orhun Kar Sec-2

21200509 Zeynep Delal Mutlu Sec-1

Table Of Contents

1. Revised E/R Model



2. Relation Schemas

2.1 Users

Relational Model:

Users(userID,usertype, name, lastname, password, phone, gender, birthdate)

Functional Dependencies:

Candidate Keys:

Normal Form:

Table Definition:

2.2 Customers

Relational Model:

Customers(userID)

Functional Dependencies:

Candidate Keys:

Normal Form:

Table Definition:

2.3 Drivers

Relational Model:

Drivers(userID)

Functional Dependencies:

Candidate Keys:

Normal Form:

Table Definition:

2.4 Requests

Relational Model:

Requests(requestID,tripTime)

Functional Dependencies:

Candidate Keys:

Normal Form:

Table Definition:

2.5 Trips

Relational Model:

Trips(tripDoneID, status, driversBehaviourRate,driversCarComfortRate, driversPunctualityRate, tripPrice())

Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.6 Cars

Relational Model:

Cars(carPlate,carBrand,carModel,carYear,carCapacity)

Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.7 Route

Relational Model:

Route(routeID, price())

Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.8 Comments

Relational Model:

Comments(commentID, content, commentDate)

Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.9 Distances

Relational Model:

Distances(DistanceID,distance)

Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.10 Locations

Relational Model:

Locations(locationID, locationName)

Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.11 TripRequest

Relational Model:

TripRequest(userID,requestID,tripDoneID,date)

Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.12 TripConsruct

Relational Model:

Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.13 CustomerRoutes

Relational Model:

CustomerRoutes(tripDoneID,userID,routeID)

Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.14 DriverRoutes

Relational Model:

DriverRoutes(userID,routeID)

Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.15 TripComments

Relational Model:

TripComments(commentID,userID,tripDoneID)

Functional Dependencies:
Candidate Keys:

Normal Form:
Table Definition:

2.16 DriversCar

Relational Model:

DriversCar(userID,

Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.17 TripDriverPreferences

Relational Model:
Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

2.18 LocationPairs

Relational Model:
Functional Dependencies:
Candidate Keys:
Normal Form:
Table Definition:

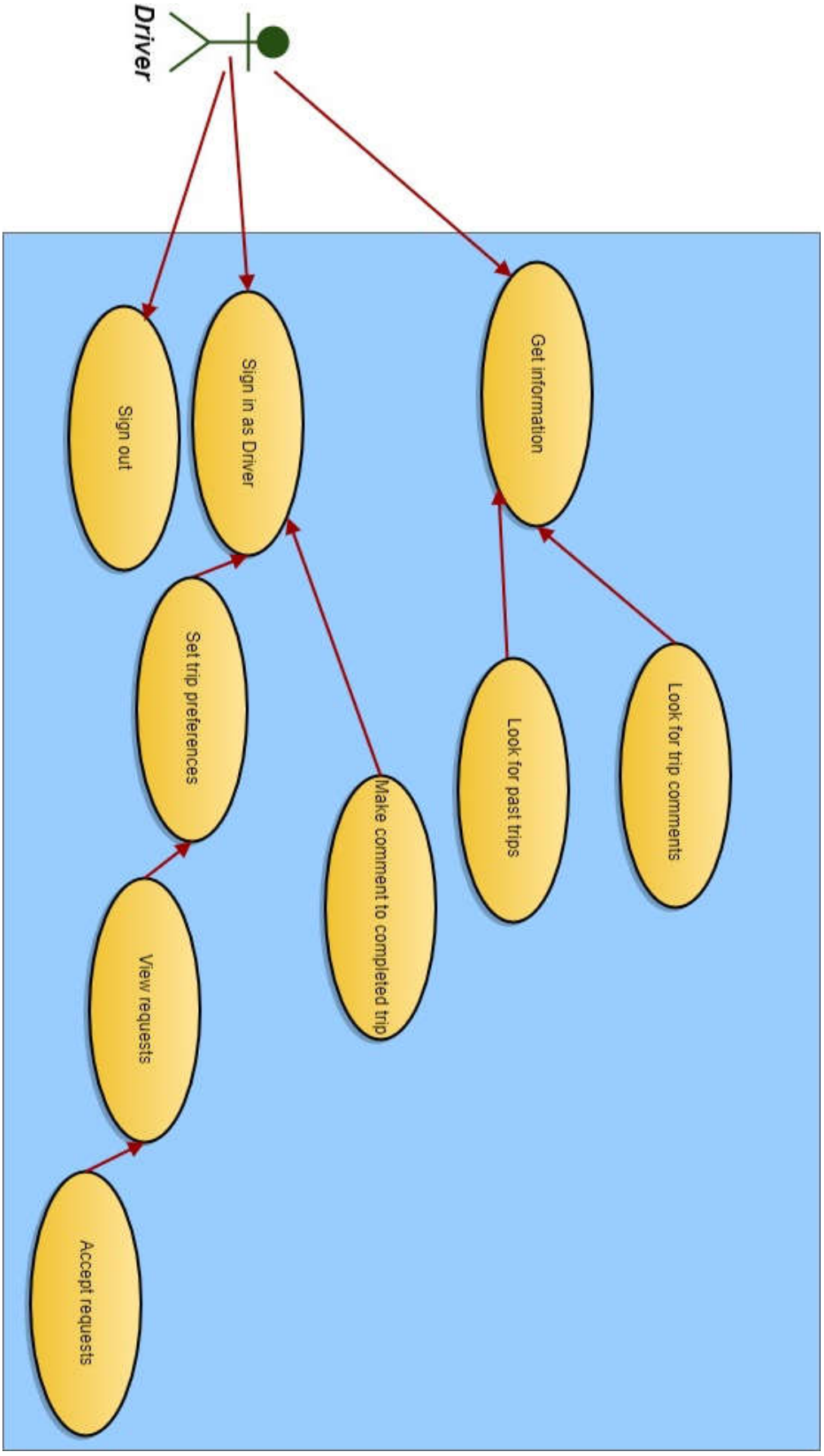
3. Functional Dependencies and Normalization of Tables

Since our database is not much complex, we will not need to decompose any table. In addition functional dependencies have mentioned in in section

4. Functional Components

4.1. Use Cases / Scenarios

4.1.1) Driver


Driver
Sign out
Sign in as Driver
Get information
Set trip preferences
Look for past trips
Look for trip comments
View requests
Make comment to completed trip
Accept requests

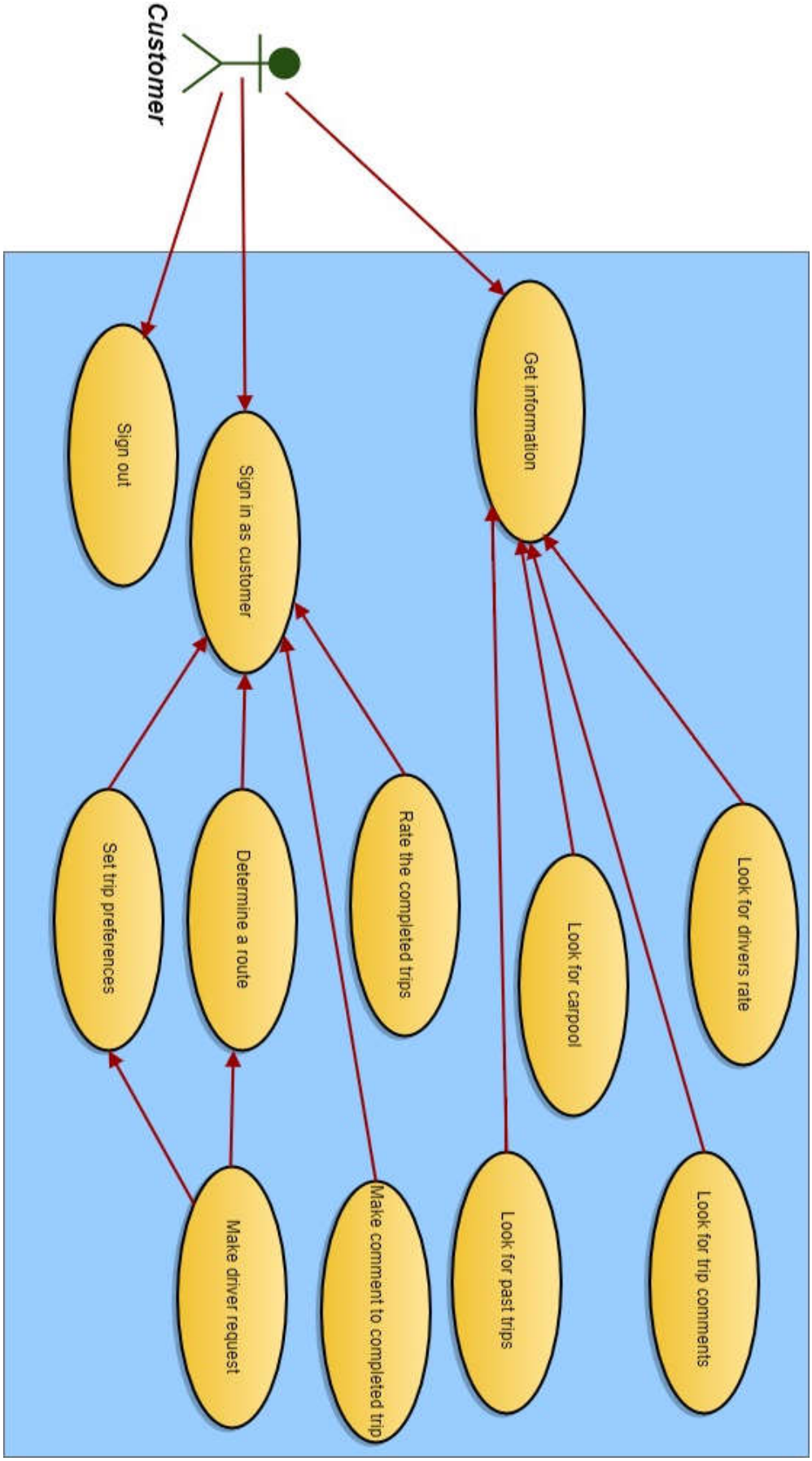
Users who are signed in as Drivers will be able to:

- Change their passwords
- View all completed past trips of his/her own
- View all comments about his/her completed past trips
- Set trip preferences and driver will receive requests according to those preferences set earlier by driver
- See all requests appropriate for driver's preferences and accept these requests
- Add/remove new cars to his/her carpool
- Increase or decrease his price coefficient

Further explanation for specific use cases:

- ➔ To successfully create a trip, driver needs to accept one or more requests from customers
- ➔ As a preference driver must determine his/her stops beforehand to see related requests
- ➔ Driver can pick up multiple customers to his car up to his/her car's limit
- ➔ Driver can change his/her preferences for a trip

4.1.2) Customer



Users who are signed in as Customers will be able to:

- View all of their completed past trips
- View registered drivers and driver rates given by driver’s customers
- See all available cars in the carpool
- See all comments in completed past trips
- Rate their completed past trips via giving points out of five to some categories

In order to make a trip customer will follow these steps:

1. Set trip preferences for their upcoming travel
2. Choose a pick-up and a drop-off point
3. Make a request
4. Wait for a driver to accept the request

Further explanation to specific use cases:

➔ Customer can change his/her preferences

➔ Customer can

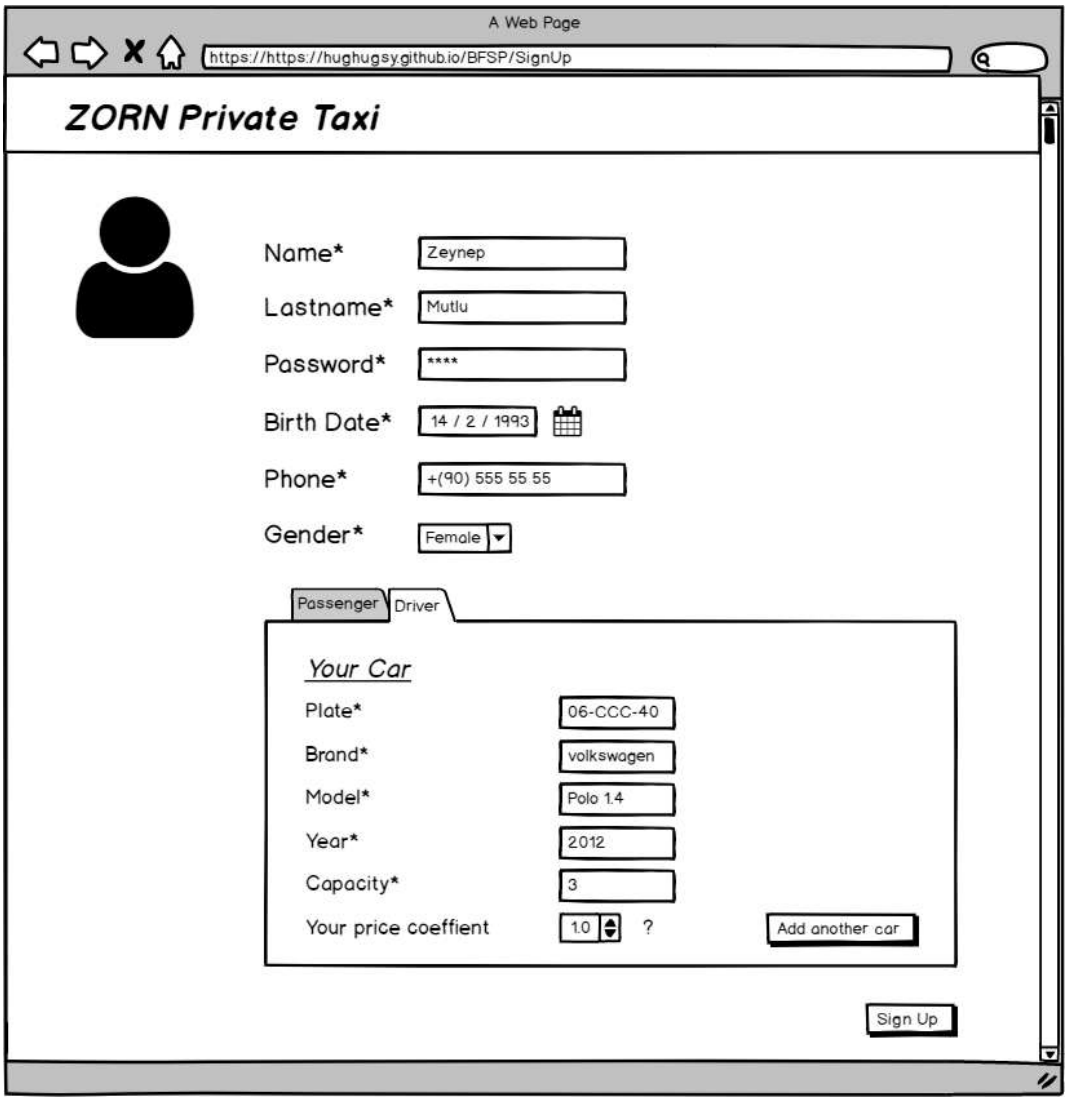
4.2 Algorithms

4.3 Data Structures

5. User Interface Design and Corresponding SQL Statements

5.1 SignUp

In this project, there are two types of user which are ‘passenger’ and ‘driver’. For each type of user, there is two different *SignUp* pages.



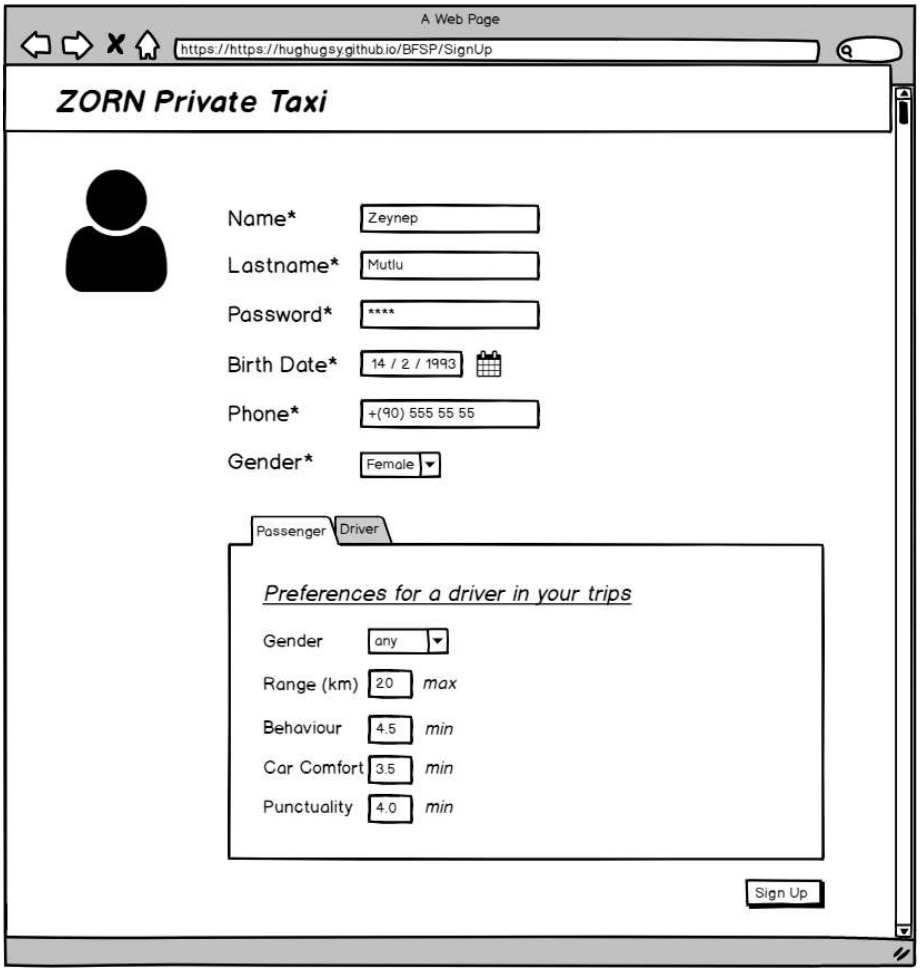
Process: The driver signs up to the system by entering the user information.

➢ **Inserting the driver into the *Users* table:**

Inputs: @userName, @lastName, @password, @phone, @gender, @birthdate, @userType
SQL Statement:
INSERT INTO Users
VALUES (@userID, @userName, @lastName, @password, @phone, @gender, @birthdate, @userType)

➢ **Inserting the car Information to *Car* table:**

Inputs: @carPlate, @carBrand, @carModel, @carYear, @carCapacity
SQL Statement:
INSERT INTO Car
VALUES (@carPlate, @carBrand, @carModel, @carYear, @carCapacity)



Process: The passenger signs up to the system by entering the user information.

➢ **Inserting the passenger into the *Users* table:**

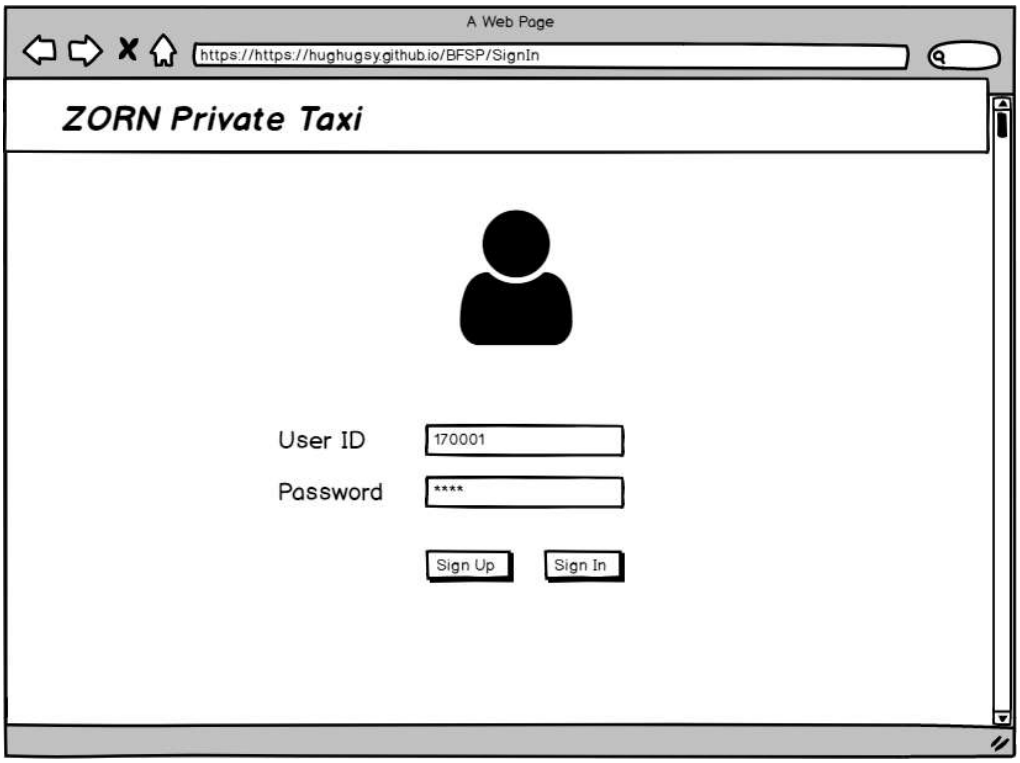
Inputs: @userName, @lastName, @password, @phone, @gender, @birthdate, @userType
SQL Statement:
INSERT INTO Users

VALUES (@userID, @userName, @lastName, @password, @phone, @gender, @birthdate, @userType)

- **Inserting the passenger preferences into *CustomerPreference* table:**

Inputs: @driversGender, @maxDriversRange, @minDriverBehaviourRate, @minDriversPunctualityRate
SQL Statement:
INSERT INTO CustomerPreferences
VALUES (@driversGender, @maxDriversRange, @minDriverBehaviourRate, @minDriversPunctualityRate)

5.2 Login



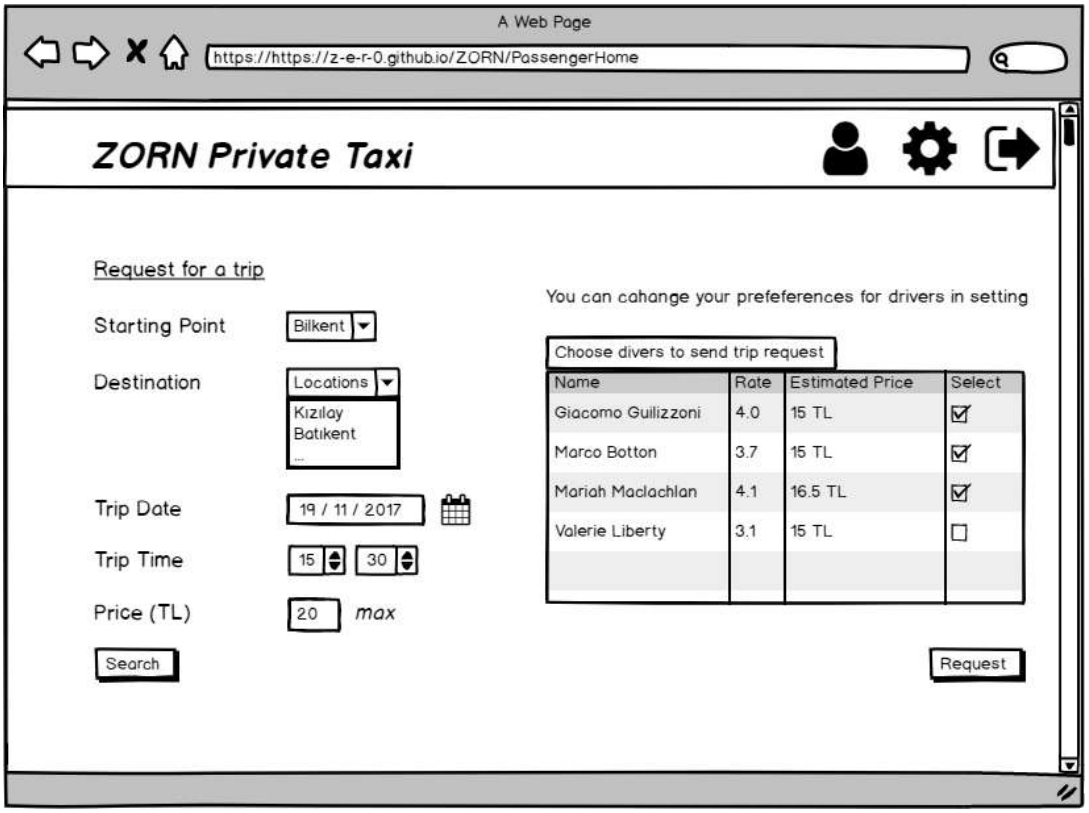
Process: The user enters the userID and the password to login to the system.

Inputs: @userID, @password

- **Login to the application:**

SQL Statements:
SELECT userID, password
FROM Users
WHERE userID = @userID AND password = @password

5.3 PassengerHome



Process: The passenger is searching for the appropriate trip.

- **Request for a trip:**

Inputs: @startingPoint, @destination, @tripDate, @tripTimeSelection
SQL Statement:

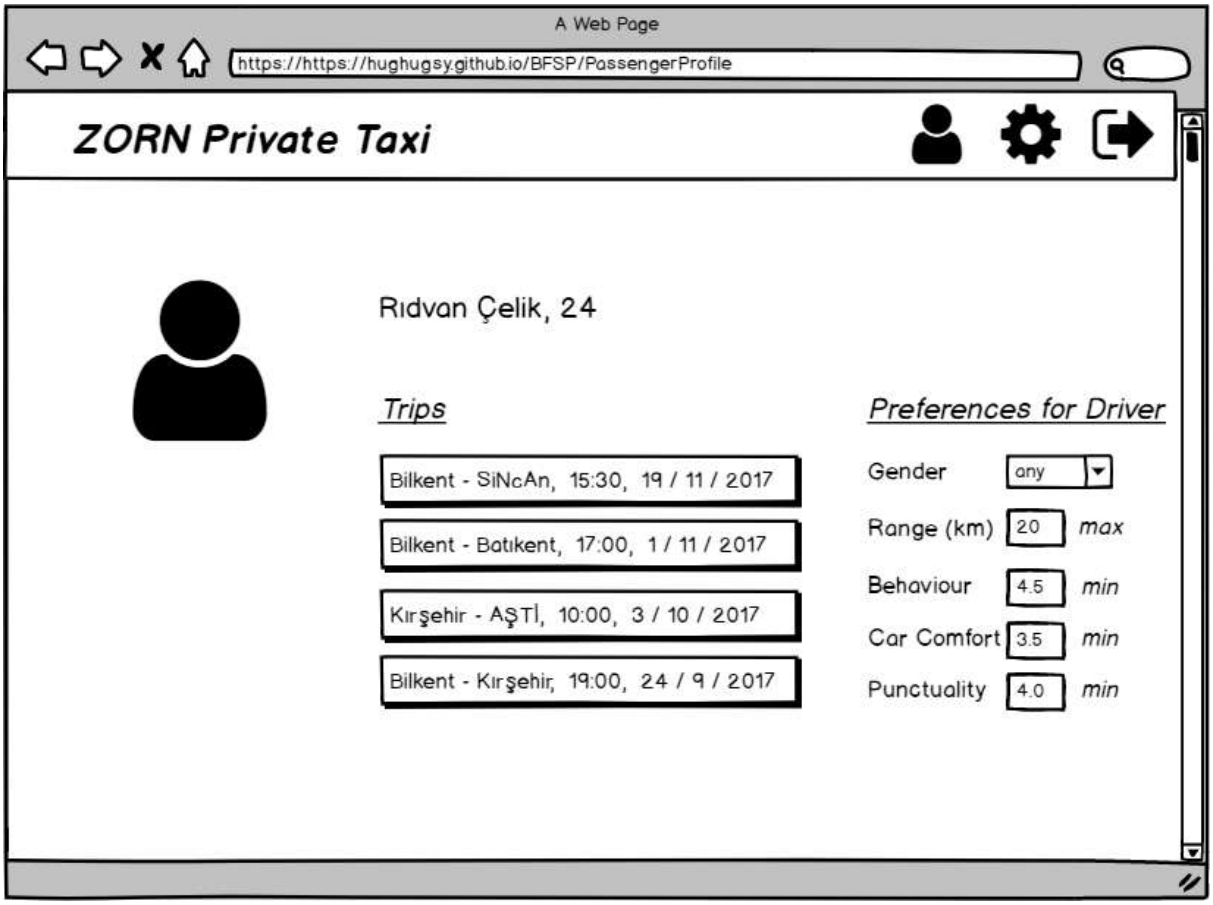
WITH startlocation (locID) as(
SELECT Locations.LocationID
FROM Locations
Where LocationName = “Bilkent”)

SELECT userName, rate, tripPrice
FROM Users|X|Trips|X|Requests|X|CustomersRoutes|X|RouteDistance|X|LocationPairs
WHERE startlocation.locID = LocationPairs.LocationID1

- **Creating the table of appropriate trips:**

Inputs: @driversGender, @maxDriversRange, @minDriverBehaviourRate, @minDriversPunctualityRate
SQL Statement:
INSERT INTO CustomerPreferences
VALUES (@driversGender, @maxDriversRange, @minDriverBehaviourRate, @minDriversPunctualityRate)

5.4 PassengerProfile



Inputs: @startingPoint, @destination, @tripDate, @tripTimeSelection

Process: The PassengerProfile screen shows the information about logged in user. On this screen, following information about user is going to be placed;

Process: The passenger is searching for the appropriate trip.

- **Request for a trip:**

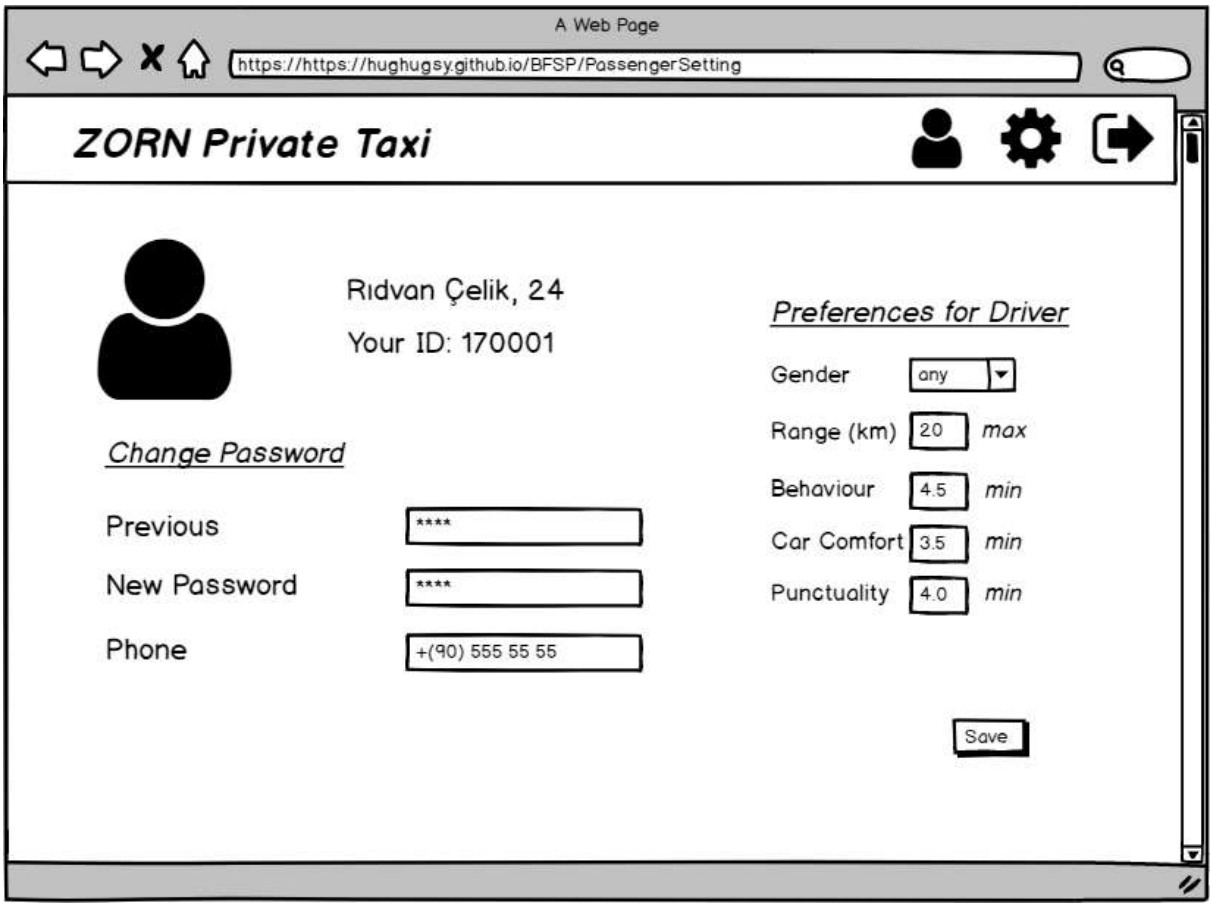
Inputs: @userName, @lastName, @password, @phone, @gender, @birthdate, @userType
SQL Statement:
INSERT INTO Users
VALUES (@userID, @userName, @lastName, @password, @phone, @gender, @birthdate, @userType)

- **Creating the table of appropriate trips:**

Inputs: @driversGender, @maxDriversRange, @minDriverBehaviourRate, @minDriversPunctualityRate
SQL Statement:

INSERT INTO CustomerPreferences
VALUES (@driversGender, @maxDriversRange, @minDriverBehaviourRate,
@minDriversPunctualityRate)

5.5 PassengerSettings

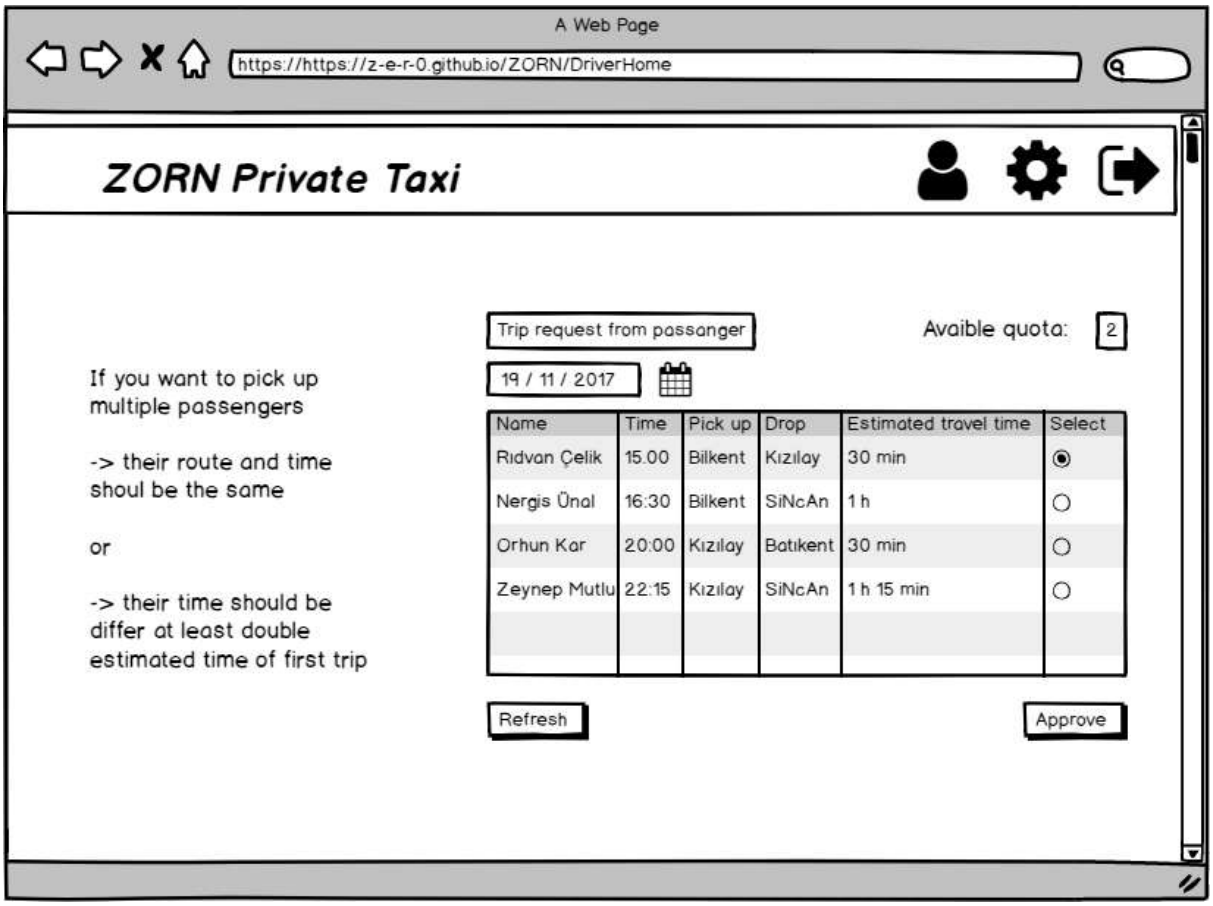


Inputs:

Process: On this page, it is possible to change the information about user.

SQL Statements:

5.6 DriverHome

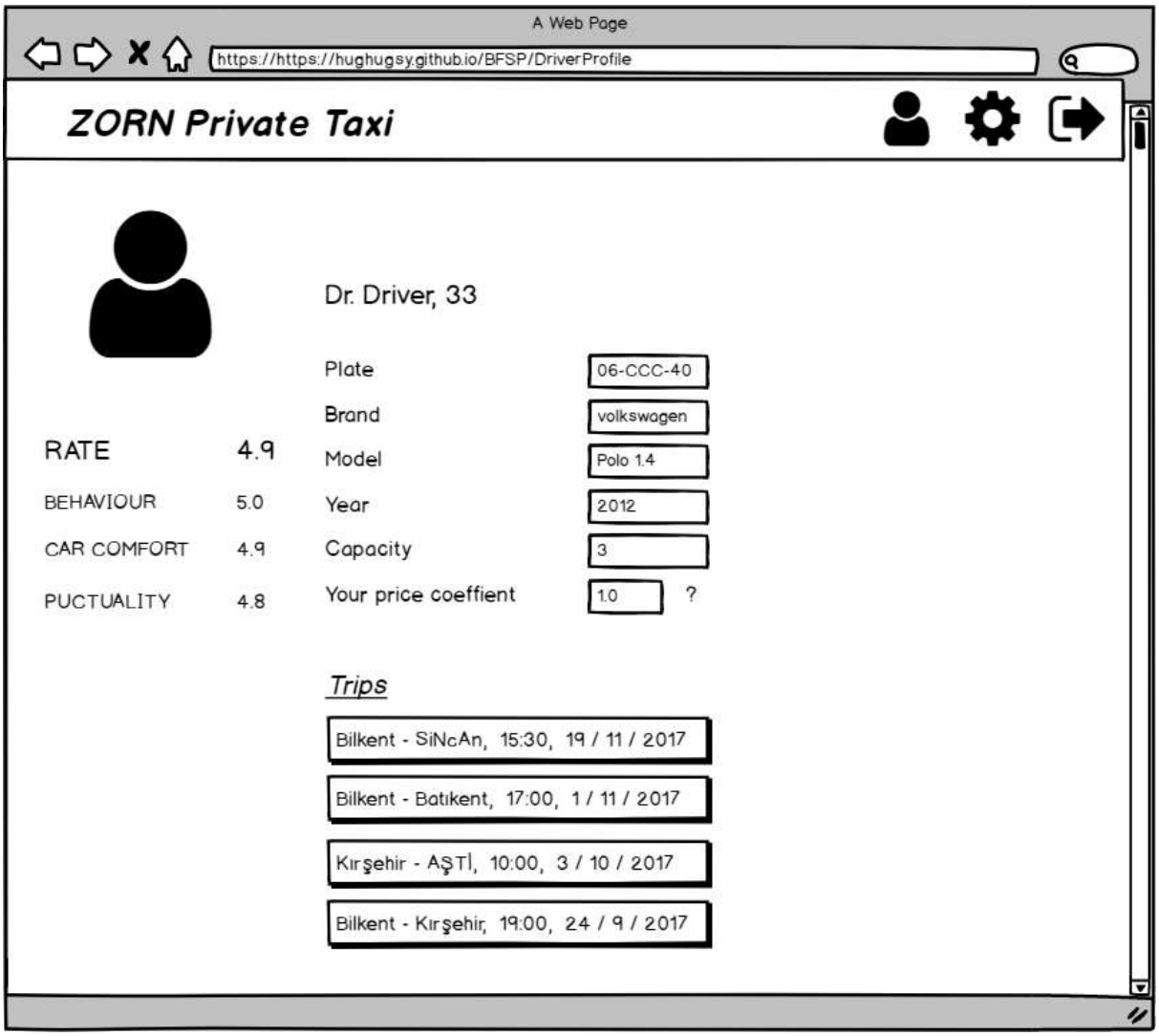


Inputs: @

Process:

SQL Statements:

5.7 DriverProfile

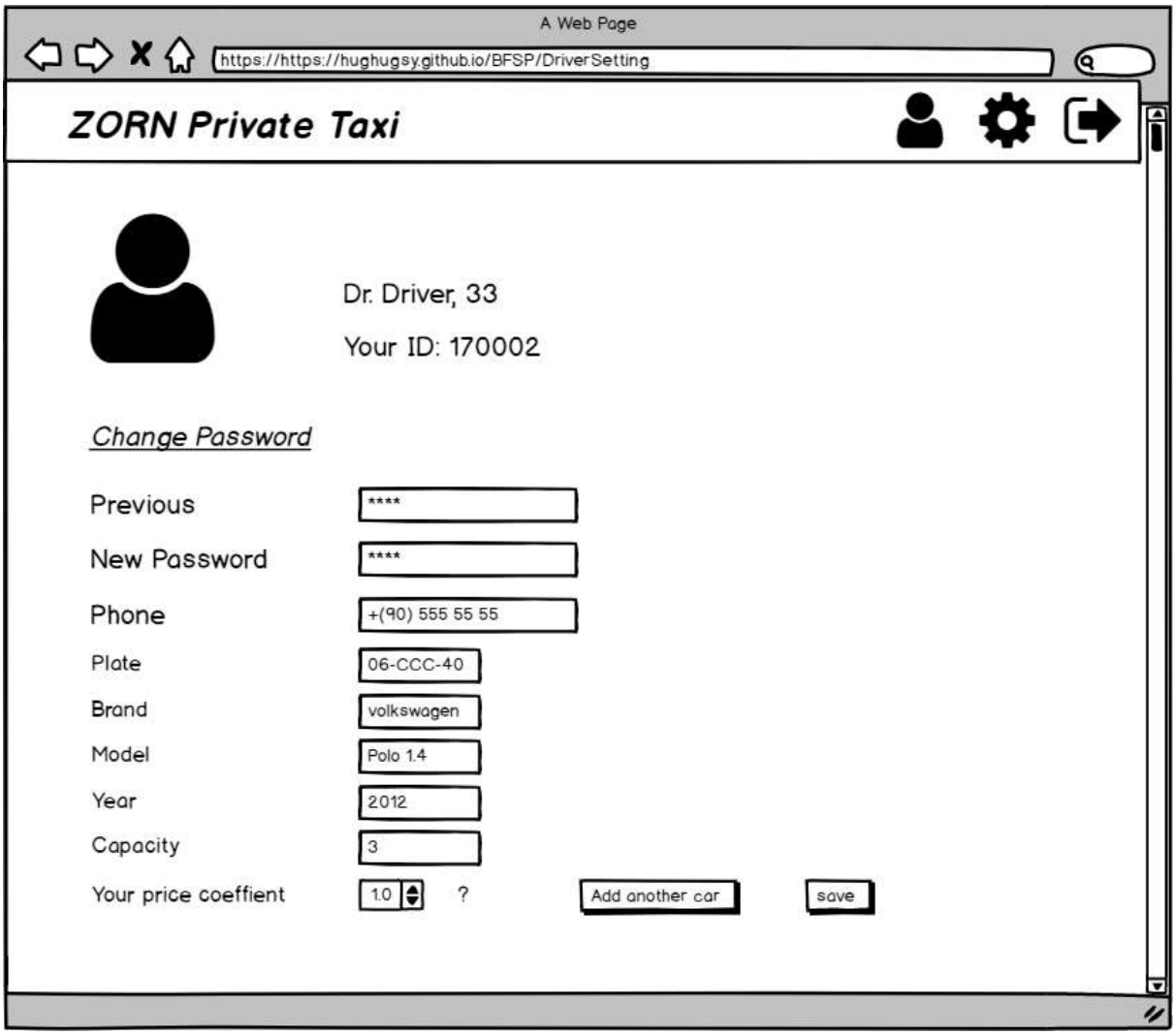


Inputs:

Process:

SQL Statements:

5.8 DriverSettings



Inputs:

Process:

SQL Statements:

6. Advanced Database Components

6.1 Views

6.1.1 Seeing all comments irrespective of the trips for a driver

6.1.2 Showing only time, date and locations of past trips

6.1.3

6.1.4

6.2 Stored Procedures

6.2.1 Rating for Users and Drivers

6.2.2 Trip Time

6.2.3 Trip Price

6.3 Reports

6.3.1

6.3.2

6.4 Triggers

- When a user signs up a system, the user ID is going to be created automatically.
- The rate of each driver is going to be calculated after each rating by the passengers.
- The suggested driver table on the PassengerHome page is going to created after a passenger trip request.

6.5 Constraints

- The system cannot be used without logging in.
- There are some specific ID numbers which are used for identifiers;
    - userID
    - requestID
    - distanceID
    - tripID
    - routeID
    - carPlate
    - futureTripID
    - locationID
    - commentID
    - CustomerPreferenceID
    - DrivePereferenceID

- If any driver wants to add more than one customer to her/his trip, he/she has to add them one by one. After adding each customer request, all related tables are going to be updated.
- A passenger can send a trip request to more than one drivers for the same trip. After first driver's approval, the other requests will be dropped by the system.
- Trip route has to contain at least one location pair.
- Each location pair is made of two locations which are start and end points.
- Each trip price is calculated by just using route distance and driver's price coefficient.
- It is forbidden to exceed the passenger quota for each car.
- A comment to a trip can only be made by a passenger who is attended to that trip

7. Implementation Plan

We are planing to use SQL database server and Visual Studio 2017 ASP.NET. For the application logic and the interface, HTML will be going to be used.