



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

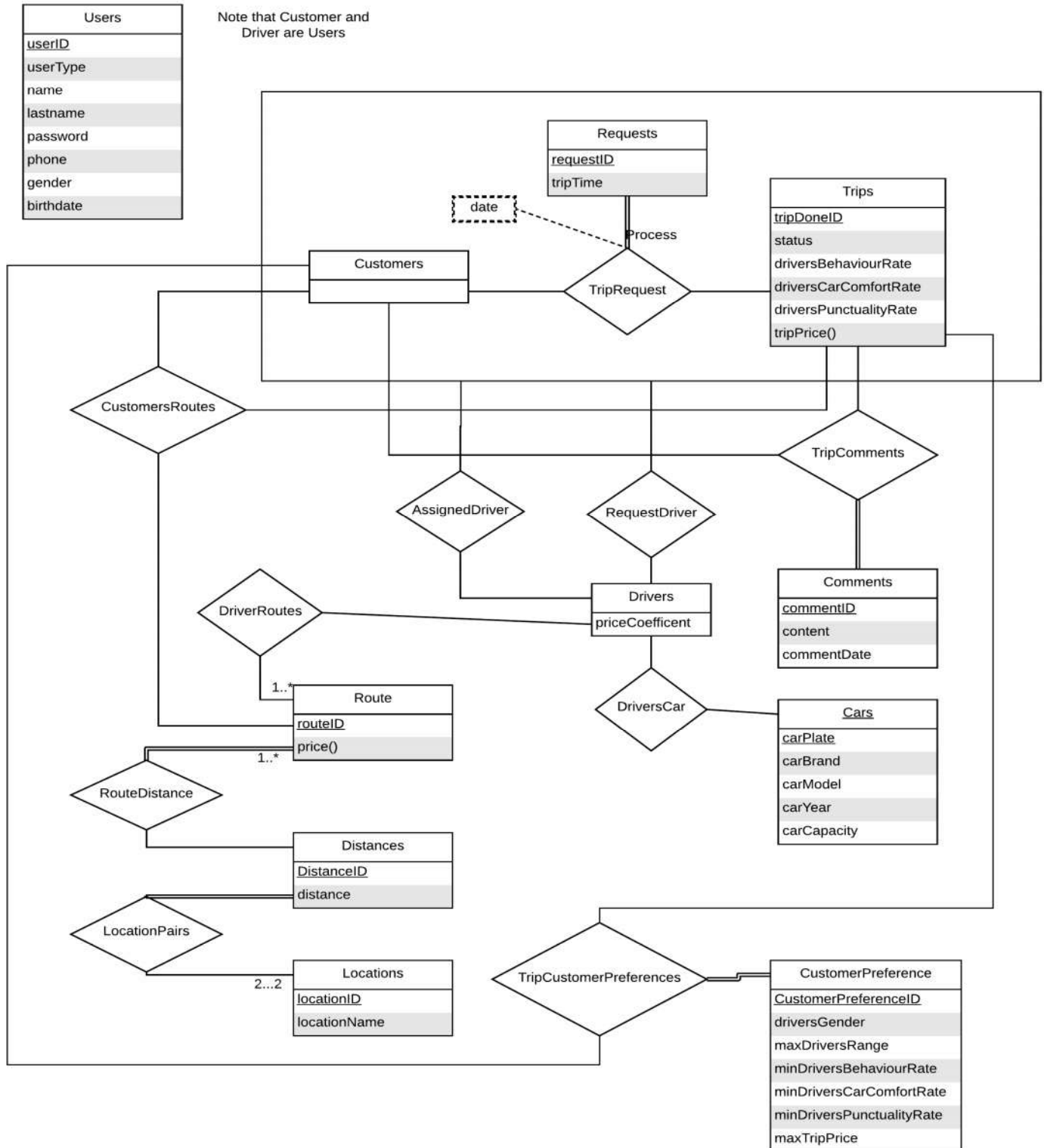
21200509 Zeynep Delal Mutlu Sec-1

Table Of Contents

1. Revised E/R Model	4
2. Relation Schemas	5
2.2 Customers	5
2.4 Requests	5
2.5 Trips	5
2.6 Cars	6
2.7 Route	6
2.8 Comments	6
2.9 CustomerPreference	6
2.10 Distances	6
2.11 Locations	7
2.13 TripRequest	7
2.14 TripConstruct	7
2.15 CustomerRoutes	7
2.16 DriverRoutes	7
2.17 TripComments	8
2.18 DriversCar	8
2.19 TripDriverPreferences	8
2.20 LocationPairs	8
3. Functional Dependencies and Normalization of Tables	8
4. Functional Components	9
4.1. Use Cases / Scenarios	9
4.2 Algorithms	14
4.3 Data Structures	14
5. User Interface Design and Corresponding SQL Statements	14
5.1 SignUp	14
5.2 Login	17
5.3 PassengerHome	17
5.4 PassengerProfile	19
5.5 PassengerSettings	20
5.6 DriverHome	21
5.7 DriverProfile	22
5.8 DriverSettings	23
6. Advanced Database Components	23
6.1 Views	23
6.1.1 Seeing all comments irrespective of the trips for a driver	23
6.1.2 Showing only time, date and locations of past trips	24

6.1.3 zdfgh	24
6.1.4 dfvgzv	24
6.2 Stored Procedures	24
6.2.1 Rating for Users and Drivers	24
6.2.2 Trip Time	24
6.2.3 Trip Price	24
6.3 Reports	24
6.3.1 aesf	24
6.3.2 aesf	24
6.4 Triggers	24
6.5 Constraints	25
7. Implementation Plan	25

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 TripConstruct

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:

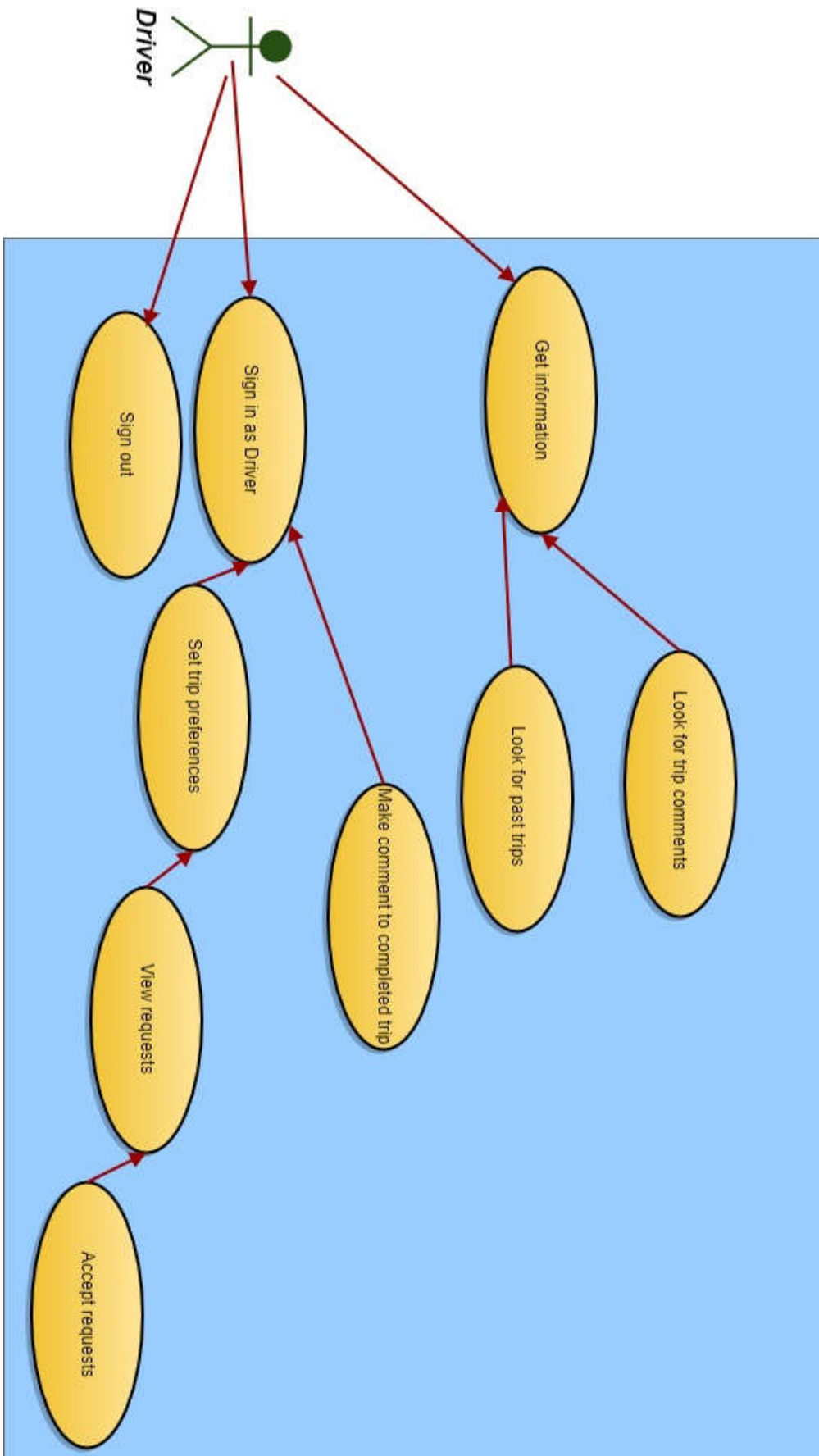
3. Functional Dependencies and Normalization of Tables

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



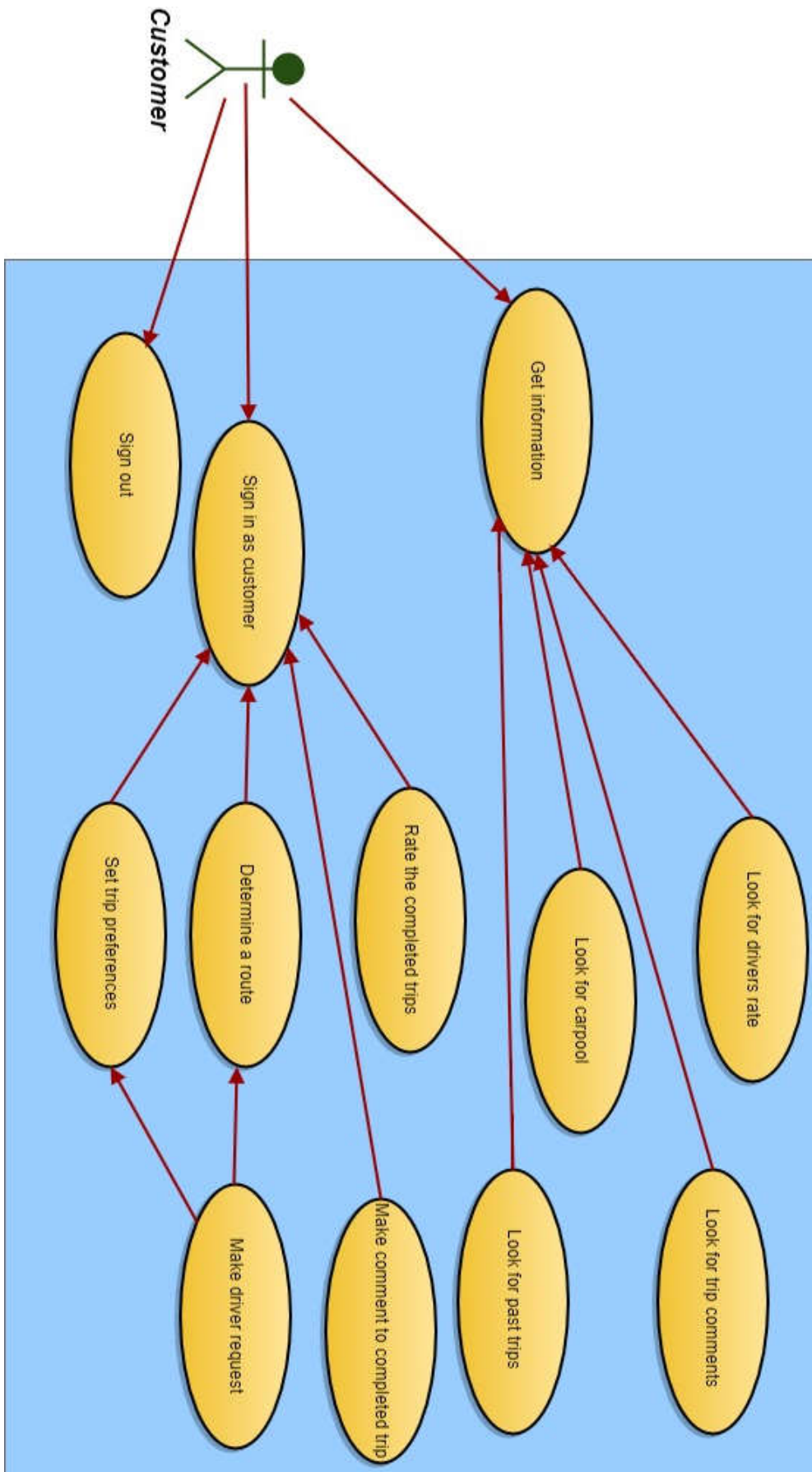
Users who are signed in as Drivers will be able to:

- 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.

The screenshot shows a web browser window titled "A Web Page" with the URL "https://https://hughugsy.github.io/BFSP/SignUp". The page header is "ZORN Private Taxi". On the left is a black silhouette of a person. The main form contains the following fields:

- Name*: Zeynep
- Lastname*: Mutlu
- Password*: ****
- Birth Date*: 14 / 2 / 1993 (with a calendar icon)
- Phone*: +(90) 555 55 55
- Gender*: Female (dropdown menu)

Below these fields are two tabs: "Passenger" and "Driver". The "Driver" tab is selected, revealing a section titled "Your Car" with the following fields:

- Plate*: 06-CCC-40
- Brand*: volkswagen
- Model*: Polo 1.4
- Year*: 2012
- Capacity*: 3
- Your price coefficient: 1.0 (with a dropdown arrow) and a question mark icon

There is an "Add another car" button next to the price coefficient field. At the bottom right of the form is a "Sign Up" button.

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)

The screenshot shows a web browser window titled "A Web Page" with the URL "https://https://hughugygithubio/BFSP/SignUp". The page header is "ZORN Private Taxi". The main form is for user registration. It includes a profile icon placeholder and the following fields: Name* (Zeynep), Lastname* (Mutlu), Password* (****), Birth Date* (14 / 2 / 1993), Phone* (+90) 555 55 55, and Gender* (Female). Below these fields are two tabs: "Passenger" and "Driver". The "Driver" tab is selected, revealing a section titled "Preferences for a driver in your trips". This section contains five fields: Gender (any), Range (km) (20 max), Behaviour (4.5 min), Car Comfort (3.5 min), and Punctuality (4.0 min). A "Sign Up" button is located at the bottom right of the form.

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



A Web Page

https://https://hughugsy.github.io/BFSP/SignIn

ZORN Private Taxi

User ID: 170001

Password: ****

Sign Up Sign In

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

The screenshot shows a web browser window titled "A Web Page" with the URL "https://https://z-e-r-0.github.io/ZORN/PassengerHome". The page header is "ZORN Private Taxi" with user, settings, and share icons. The main content area is titled "Request for a trip". On the left, there are input fields for "Starting Point" (Bilkent), "Destination" (Locations dropdown showing Kizilay and Botikent), "Trip Date" (19 / 11 / 2017), "Trip Time" (15 and 30), and "Price (TL)" (20 max). A "Search" button is at the bottom left. On the right, a message says "You can cahange your preferences for drivers in setting". Below this is a table titled "Choose divers to send trip request" with columns: Name, Rate, Estimated Price, and Select. The table lists four drivers: Giacomo Gullizzoni (Rate 4.0, Price 15 TL, checked), Marco Botton (Rate 3.7, Price 15 TL, checked), Mariah Maclachlan (Rate 4.1, Price 16.5 TL, checked), and Valerie Liberty (Rate 3.1, Price 15 TL, unchecked). A "Request" button is at the bottom right.

Name	Rate	Estimated Price	Select
Giacomo Gullizzoni	4.0	15 TL	<input checked="" type="checkbox"/>
Marco Botton	3.7	15 TL	<input checked="" type="checkbox"/>
Mariah Maclachlan	4.1	16.5 TL	<input checked="" type="checkbox"/>
Valerie Liberty	3.1	15 TL	<input type="checkbox"/>

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

The screenshot shows a web browser window titled "A Web Page" with the URL <https://https://hughugsy.github.io/BFSP/PassengerProfile>. The page header is "ZORN Private Taxi" with navigation icons for user, settings, and a right arrow. The main content area displays the user profile for "Rıdvan Çelik, 24" with a placeholder icon. Below the profile, there are two sections: "Trips" and "Preferences for Driver".

Trips

Bilkent - SiNcAn, 15:30, 19 / 11 / 2017
Bilkent - Batıkent, 17:00, 1 / 11 / 2017
Kırşehir - AŞTİ, 10:00, 3 / 10 / 2017
Bilkent - Kırşehir, 19:00, 24 / 9 / 2017

Preferences for Driver

Gender	<input type="text" value="any"/>	
Range (km)	<input type="text" value="20"/>	max
Behaviour	<input type="text" value="4.5"/>	min
Car Comfort	<input type="text" value="3.5"/>	min
Punctuality	<input type="text" value="4.0"/>	min

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

A Web Page

https://https://hughugsy.github.io/BFSP/PassengerSetting

ZORN Private Taxi

Rıdvan Çelik, 24
Your ID: 170001

Change Password

Previous

New Password

Phone

Preferences for Driver

Gender

Range (km) max

Behaviour min

Car Comfort min

Punctuality min





Inputs:

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


SQL Statements:

5.6 DriverHome




A Web Page



https://https://z-e-r-0.github.io/ZORN/DriverHome



ZORN Private Taxi



If you want to pick up multiple passengers

-> their route and time should be the same


or

-> their time should be differ at least double estimated time of first trip

Trip request from passanger

Available quota: 2

19 / 11 / 2017



Name	Time	Pick up	Drop	Estimated travel time	Select
Rıdvan Çelik	15.00	Bilkent	Kızılay	30 min	<input checked="" type="radio"/>
Nergis Ünal	16.30	Bilkent	SiNcAn	1 h	<input type="radio"/>
Orhun Kar	20.00	Kızılay	Batıkent	30 min	<input type="radio"/>
Zeynep Mutlu	22.15	Kızılay	SiNcAn	1 h 15 min	<input type="radio"/>

Refresh

Approve

Inputs: @

Process:


SQL Statements:

5.7 DriverProfile

A Web Page

https://https://hughugsy.github.io/BFSP/DriverProfile

ZORN Private Taxi



Dr. Driver, 33

RATE	4.9	Plate	<input type="text" value="06-CCC-40"/>
BEHAVIOUR	5.0	Brand	<input type="text" value="volkswagen"/>
CAR COMFORT	4.9	Model	<input type="text" value="Polo 1.4"/>
PUCTUALITY	4.8	Year	<input type="text" value="2012"/>
		Capacity	<input type="text" value="3"/>
		Your price coeffient	<input type="text" value="1.0"/> ?

Trips

Inputs:

Process:

SQL Statements:

5.8 DriverSettings

A Web Page

https://https://hughugsy.github.io/BFSP/DriverSetting

ZORN Private Taxi

Dr. Driver, 33
Your ID: 170002

Change Password

Previous	****
New Password	****
Phone	+(90) 555 55 55
Plate	06-CCC-40
Brand	volkswagen
Model	Polo 1.4
Year	2012
Capacity	3
Your price coefficient	10 ?

Add another car save

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 be 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
 - DrivePreferenceID
- 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.