POLITECNICO DI MILANO

Software Engineering II

myTaxiService

RASD

Requirements Analysis and Specification Document

Author:

Hamidreza Hanafi

Navid Heidari

November 6th 2015

Table of Contents

1. INTRODUCTION

1.1 PURPOSE

This document represents the Requirement Analysis and Specification Document (RASD). The main purpose of this document is to describe the system in terms of functional and non-functional requirements, show the constraints and the limit of the software and simulate the typical use cases that will occur after the development. This document is intended to all developers and programmers who have to implement the requirements, to system analyst who want to integrate other system with this one, and could be used as a contractual basis between the customer and the developer.

1.2 DESCRIPTION OF THE GIVEN PROBLEM

The “myTaxiService” is the name of an application that will be created to help taxi services in city. In particular, we want to simplify the access of passengers to the service and guarantee a fair management of taxi queues for taxi drivers.

Passengers can request a taxi either through the web or the mobile application. The system answers to the request by informing the passenger about the code of the incoming taxi and the waiting time. Taxi drivers use the mobile application to inform the system about their availability and to confirm that they are going to take care of a certain request. The system guarantees a fair management of taxi queues. In particular, the city is divided in taxi zones (approximately 2 km2 each). Each zone is associated to a queue of taxis. The system automatically computes the distribution of taxis in the various zones based on the GPS information it receives

from each taxi. When a taxi is available, its identifier is stored in the queue of taxis in the corresponding zone.

When a request arrives from a certain zone, the system forwards it to the first taxi queuing in that zone. If the taxi confirms, then the system will send a confirmation to the passenger. If not, then the system will forward the request to the second in the queue and will, at the same time, move the first taxi in the last position in the queue.

A user also can reserve a taxi by specifying the origin and the destination of the ride. The reservation has to occur at least two hours before the ride. In this case, the system confirms the reservation to the user and allocates a taxi to the request 10 minutes before the meeting time with the user.

1.3 GOALS

When the system (myTaxiService) will be developed, it would provide some features:

- Registration for unregistered users (Sign Up for passengers or drivers)
- Allow users to Sign In
- Sending service request by passenger
- Receiving the respond of the request by passenger from the system
- Allow passenger to reserve a service (at least two hours before the ride)
- Allow driver to declare his/her availability
- Responding to a request from the system by driver
- The system will manage the queue of taxis of each zone with respect to their availability and their responds to the requests

1.4 ASSUMPTIONS

In some cases we have to decide about some non clear points and we have to have some assumption here:

- If a passenger sends a request service, he cannot cancel it and has to pay the service price.
- Passenger can reserve taxi from at least 2 hours before the ride.
- Passenger can cancel the reservation at least 11 minutes before the ride (before allocation any taxi to that reservation), otherwise, the passenger cannot cancel the reservation.
- When a taxi driver on top of the list does not confirm a request, the system moves the driver to the last position of the queue.
- For use of the system, all users have to register at the first step. Because for being able to send a request by passenger or respond to a request by driver, they have to be members before. Therefore for the starting point, all the users have to Sign up.
- The system answers to the request by informing the passenger about the code of the incoming taxi and the waiting time. This waiting time will be calculated by the system with the taxis GPS and the passengers address.
- When a driver receives a service request from the system, he/she has 2 minutes to accept it, otherwise the system move the driver to the last position of the queue and send the request to the next driver in the queue (now on the top of the queue).

1.5 PROPOSED SYSTEM

The myTaxiService system is proposed as a web platform and mobile application towards a better providing of taxi services.

Passengers can manage their profiles and ask for a taxi at the moment or make a taxi reservation at least 2 hours before ride. The drivers also can manage their profiles and declare their availabilities in a certain zone of city. They also have the possibility of responding to the request receiving from the system.

Besides the abilities of such users, the system will manage these services and optimize the queue of taxi drivers in each zone. The system also receives reservation requests from passengers, and 2 hours before ride, allocates a taxi to that reservation.

1.6 IDENTIFYING STAKEHOLDERS

The main stakeholder of the project is the professor who defined this project for the course Software Engineering II. We think during this project we should learn better the steps of producing an application and also use those experiences for the next projects. At the other hand we are supposed to deliver each phase of the project before its deadline time.

2. ACTORS IDENTIFYING

- **Unregistered User:** a person who can Sign up. Indeed for using the myTaxiService system, the Unregister User has to Sign up and fill the form of registration and then become a Passenger or Driver for using the system.
- **Passenger:** one of the main actors of this project is Passenger. Passenger is a person who already registered and can send request for a taxi or make reservation in the future. He/she can also modify the reservation before allocating a taxi to that reservation by system.

- **Driver:** the other main actor of the system is Driver. The driver is a registered user who has a taxi. He/she can declare if he/she is available to work and also can confirm or reject the requests of services.
- **Admin:** the admin can manage the passengers and drivers profiles (ban users) and also modify (add, update or delete) the zones of the city.

3. REQUIREMENTS

1. **Registration for unregistered users (Sign Up for passengers or drivers)**
    - The system shows a form for registration
2. **Allow users to Sign In**
    - The system shows Sign In form and allows users to Sign In
3. **Sending service request by passenger**
    - Passenger must be signed in
    - System shows the form of service request to passenger
4. **Receiving the respond of the request by passenger from the system**
    - Passenger must be signed in and already sent his request
    - The system must receive a confirm respond from driver
    - The system will show the confirmation to the passenger
5. **Allow passenger to reserve a service**
    - Passenger is already signed in and selects reservation
    - The system shows the reservation form
6. **Allow driver to declare his/her availability**
    - Driver must be signed in
    - The system shows the profile form to driver and driver can modify his/her availability

7. **Allow driver to declare his/her availability**
    - Driver must be signed in
    - The system shows the passengers request to the driver and he/she can confirm the request
8. **The system will manage the queue of taxis**
    - The system must have the drivers availability
    - The system also checks their confirmations for the requests and updates the queue

3.1 FUNCTIONAL REQUIREMENTS

Now we can specify some functional requirements for the actors of MeteoCal:

- **Unregistered User:**
    - Sign up
- **Passenger**
    - Log in
    - Send service requests
    - Receive responds for service requests
    - Make reservation
    - Modify or delete reservation at least 11 minutes before the ride
    - Being noticed about the code of the incoming taxi and the waiting time
- **Driver**
    - Log in
    - Set his/her availability
    - Receives requests from system
    - Respond to the requests

- **Admin**
    - Log in
    - Manage users profiles
    - Manage city zones

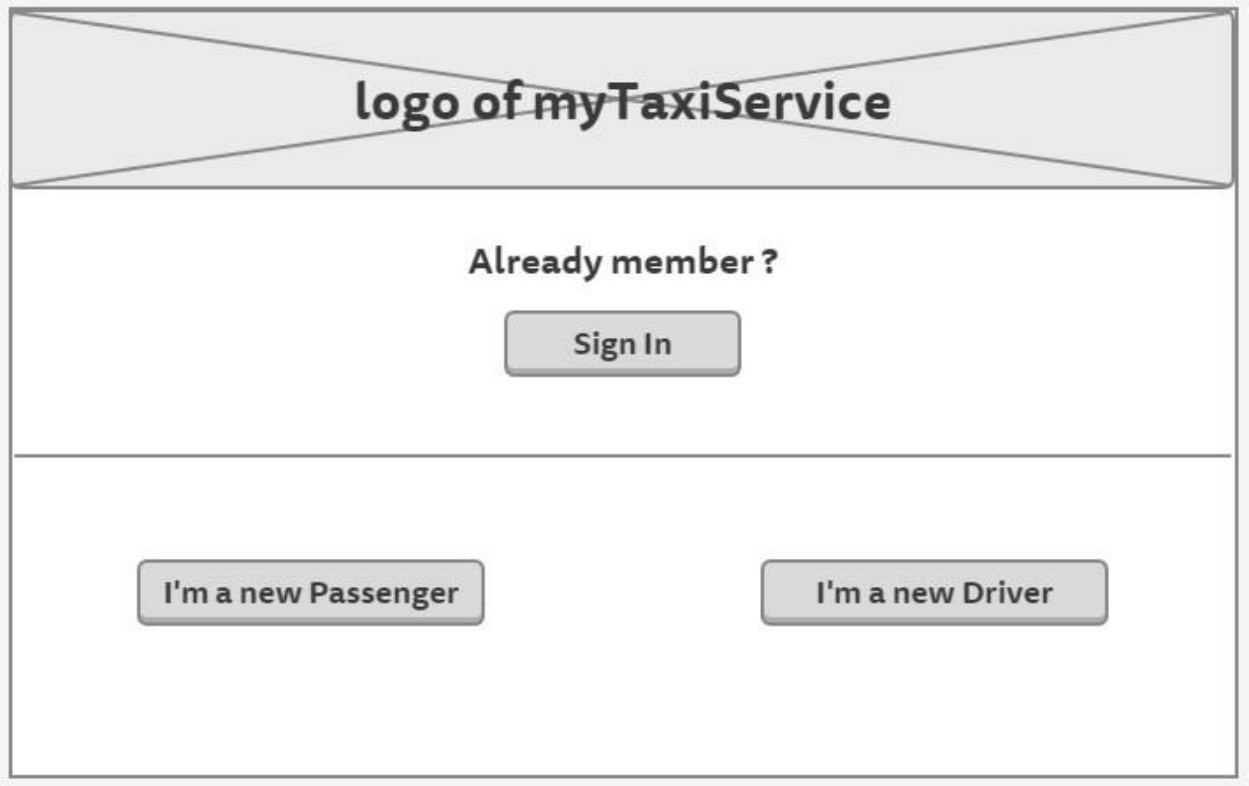
3.2 NON FUNCTIONAL REQUIREMENTS

3.2.1 User Interface

The interface of the application will be used via web or mobile app. In the first page of the application, the user can sign up or sign in to the system. When the user logged in as a passenger, he/she can see the profile page. By filling the parts of this page, can send a request to the system for that moment or make a reservation for at least 2 hours later. On the other hand, if a user signs in as a driver, he/she in this profile page can manage the availability and respond the requests to the system.

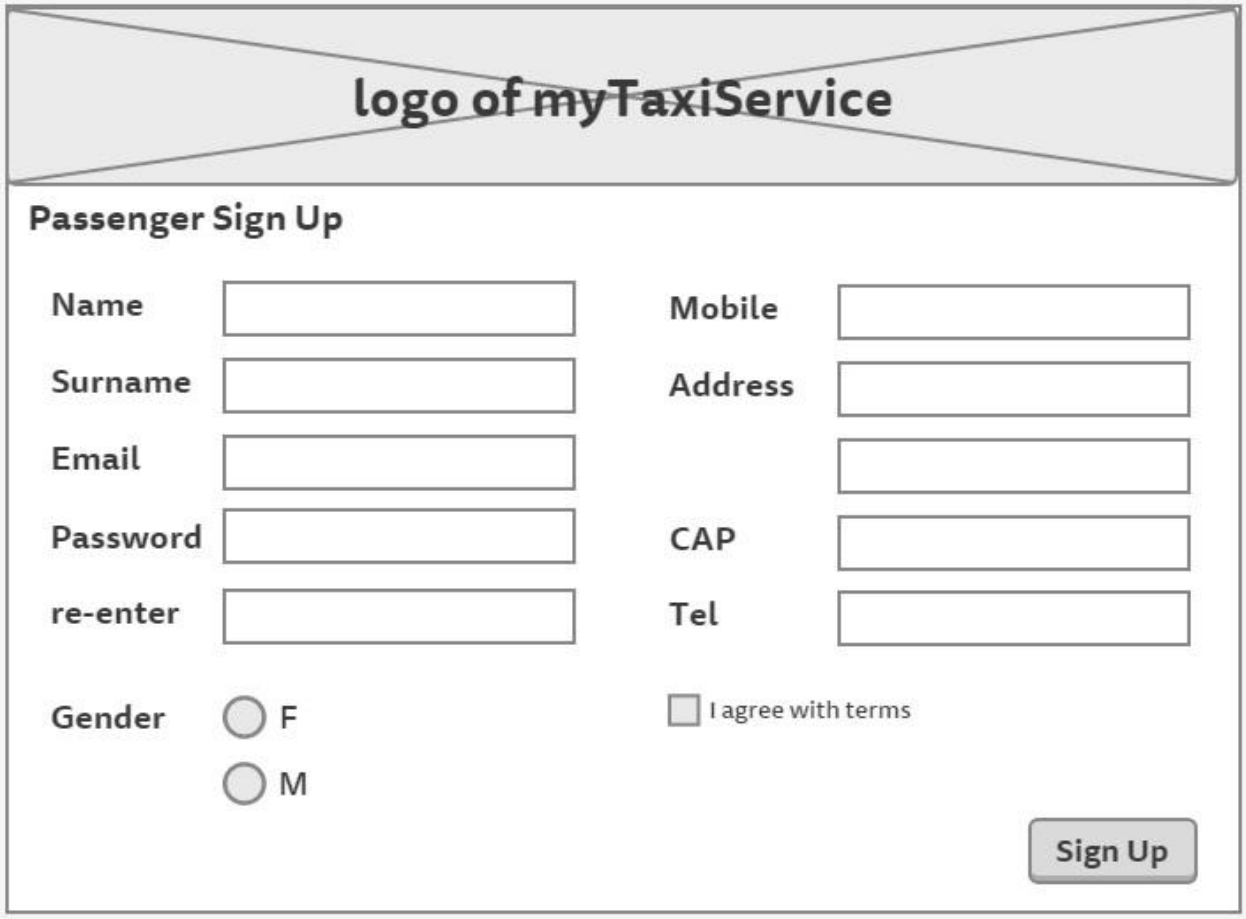
It is better to have a platform for the application that all the users can use it simply and easily. So we will try to create an application that seems familiar to use for all users.

We can see below the sketch of some pages of the application.

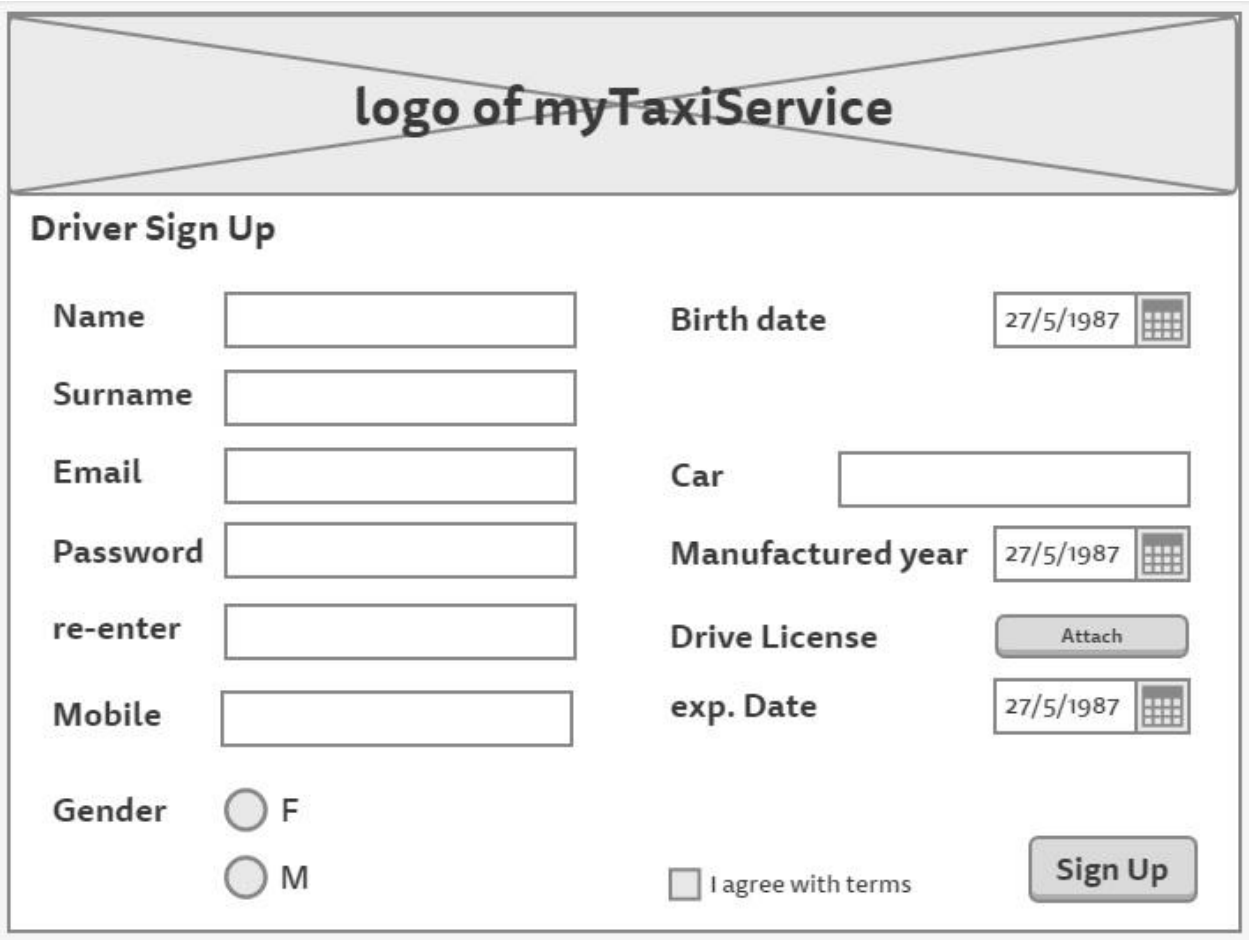


If user already registered can fill the username and password parts and log in to the system.

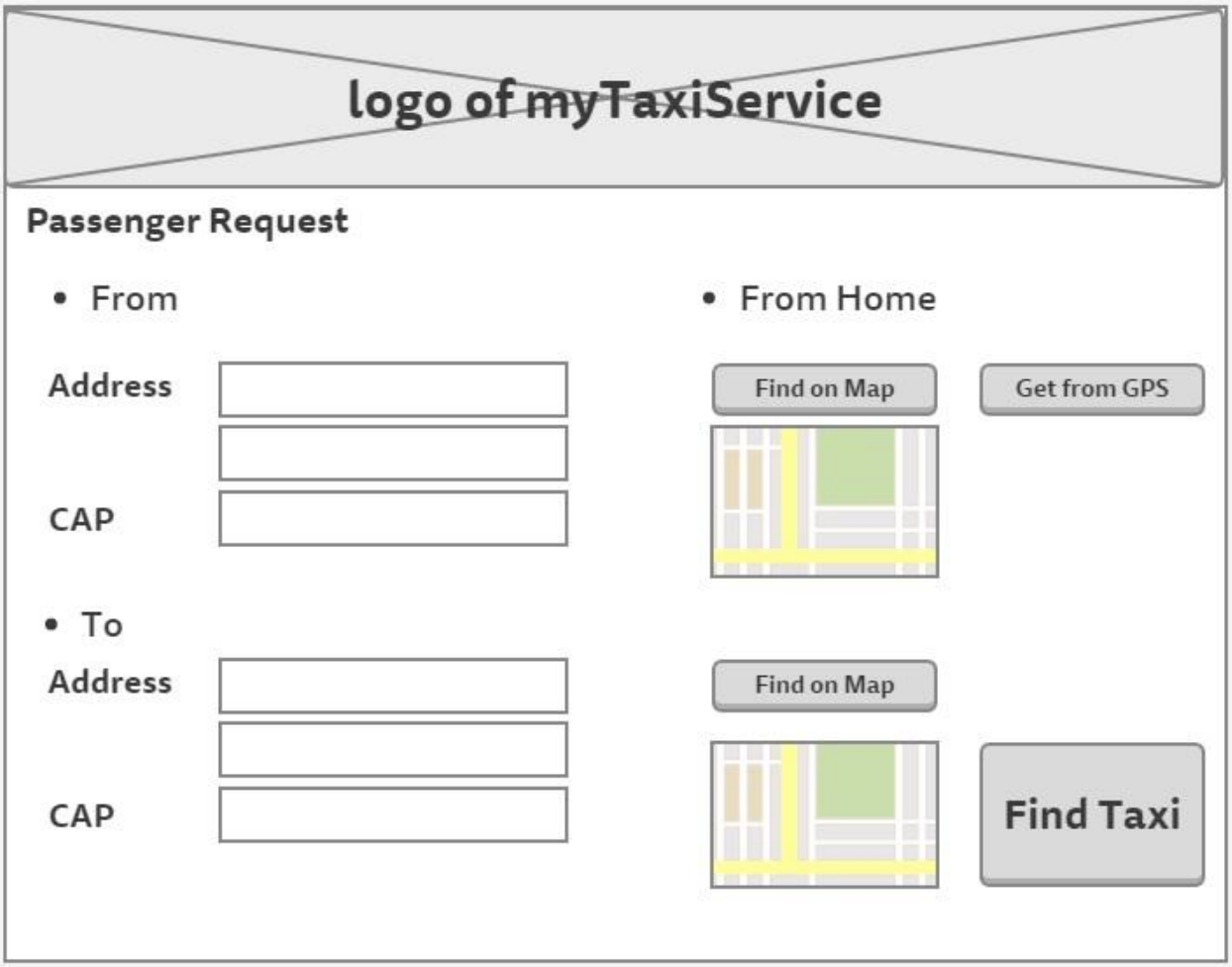
If user wants to use the app as a passenger, has to sign up here.



And if user wants to use the app as a driver, has to sign up here.



When a passenger signs into the system, he/she can send a request by filling the parts of this page.



3.2.2 Documentation

For each step of the project we will have to provide documents before their deadline date. The steps will be:

- **RASD:** Requirement Analysis and Specification Document, The RASD contains the description of the scenarios, the use cases that describe them, and the models describing requirements and specification.
- **Design Document (DD):** it contains a functional description of the system and other materials to provide a good and complete view of the system.
- **Installation Manual:** a guide for the user to install myTaxiService.
- **User Manual:** a guide to use myTaxiService.
- **Testing Manual:** a report of testing other projects of the other groups.

3.2.3 Architectural considerations

The platform that will be used in this project is J2EE with a database. An Internet connection will be need for using this application. Software Interfaces :

- A Database Management System (DBMS):

  – Name: MySQL.

  – Version: 5.6.21

  – Source: http://www.mysql.it/
- Java Virtual Machine (JVM)

  – Name: JEE

  – Version: 7

  – Source: http://www.oracle.com/technetwork/java/javaee/tech/index.html
- Application server:

  – Name: Glassfish.

  – Version: 4.1.

  – Source: https://glassfish.java.net/

- Operating System (OS)

  – Application must be able to run on any OS which supports JVM and DBMS specified before.

- Android Software Development Kit (SDK)

  – Name: Android SDK

  – Version: API level 19

  – Source: https://developer.android.com/sdk/index.html

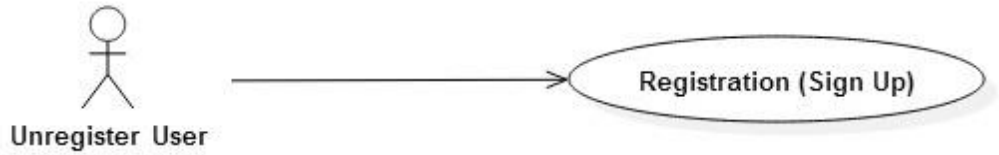
4. SCENARIOS IDENTIFYING

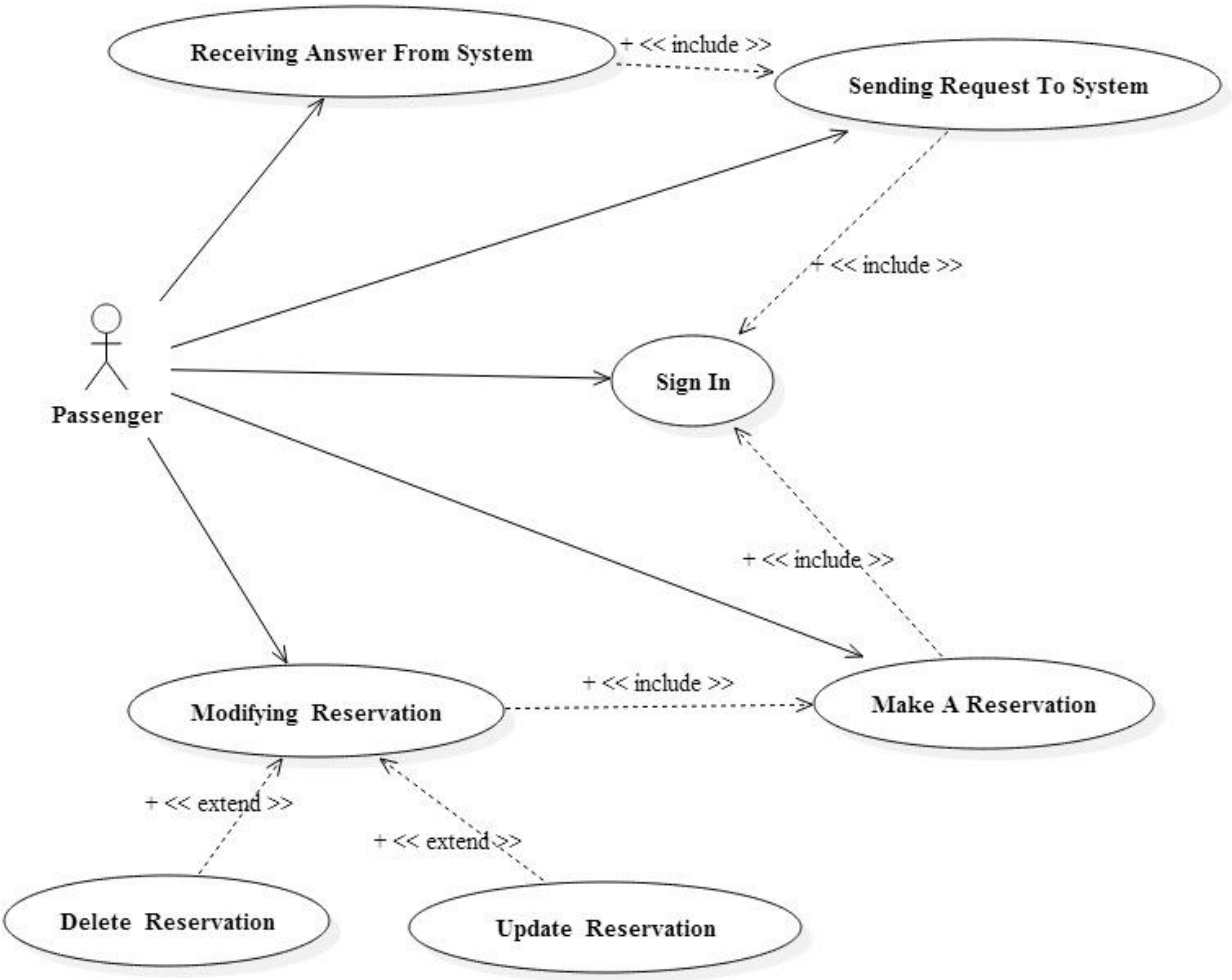
Here we have some possible scenarios:

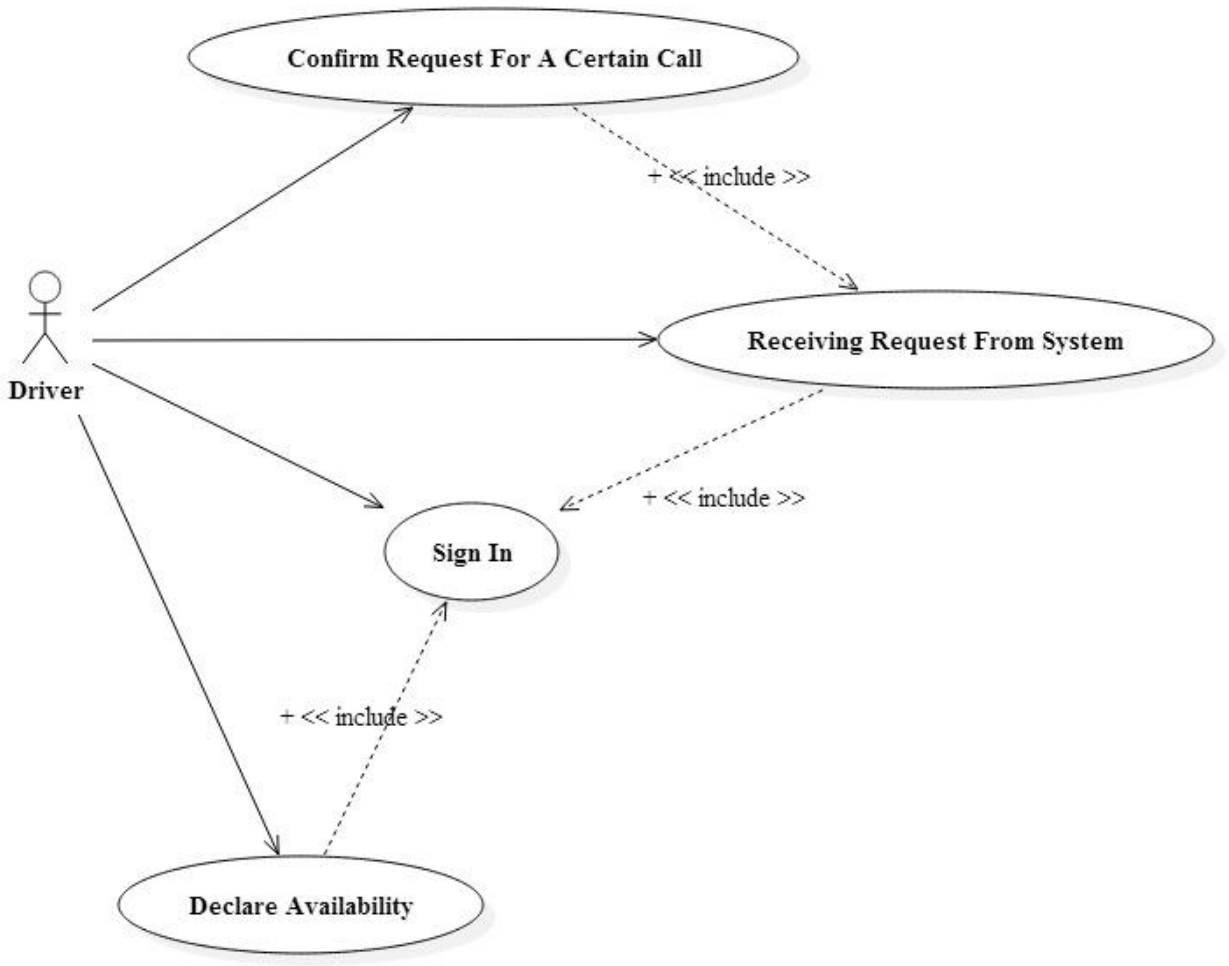
- Last week, Paul introduced a new application to his friend Denis. Paul said that he just found this amazing app on the Internet that is called myTaxiService. Denis found this app will be very useful for him because he always uses taxi. So he decides to use this application. Then he goes to the website and will see that he has to sign up if he wants to use this app. So he fills the form of the registration and will have his own username and password and therefore his own profile.
- Denis wants to go to a job meeting and did not want use his own car. He decides to call for a taxi. He remembers that he already has myTaxiService! Then he goes to his profile and sends a request to the system for a taxi. After two minutes, the system informs him that in 5 minutes he will receive a taxi with this code: 20151006.
- Andrea is a driver who uses the myTaxiService application for his work. But today, after 3 hours of working, he does not feels he is sick and needs to go home and rest. So he turns off his availability via his mobile app and goes home.
- Paul lives in Manchester. Tomorrow morning, he has to go to Milan to visit a friend and needs a taxi at 5:00 AM. He signs in to his profile on myTaxiService application and makes a reservation. At 4:50 he will receives a message from system about the details of the incoming taxi.

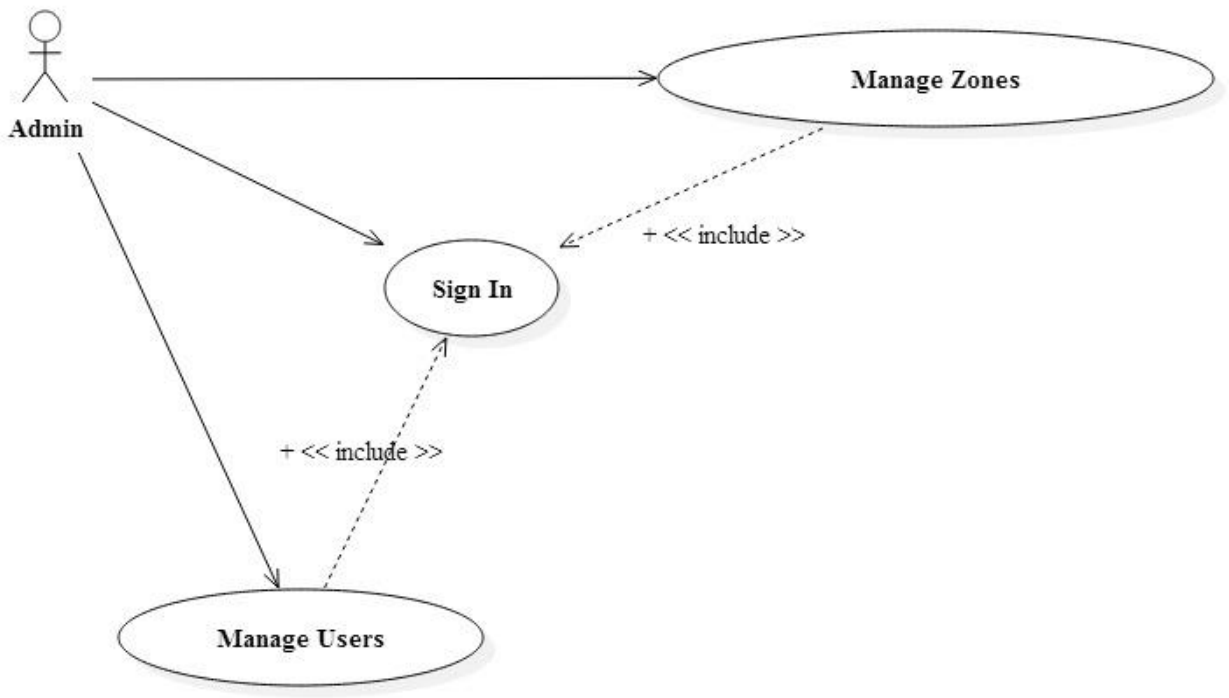
5. UML MODELS

5.1 USE CASE DIAGRAMS






Confirm Request For A Certain Call
+ << include >>
Driver
Receiving Request From System
+ << include >>
Sign In
+ << include >>
Declare Availability


Admin
Manage Zones
+ << include >>
Sign In
+ << include >>
Manage Users

5.2 USE CASES DESCRIPTIONS

Name	Sign In
Actor	User (Passenger or Driver)
Entry Conditions	The user has signed up successfully to the system
Flow of Events	• The user is in the Sign In page • The system shows the Sign in form • The user types the username and password and clicks the button • The system shows the profile page
Exit Conditions	There is not any exit conditions
Exceptions	The username and/or password is wrong , the system shows an error massage

Name	Send Request
Actor	Passenger
Entry Conditions	The Passenger is registered and signed in
Flow of Events	• The passenger is in his/her profile • The system shows the Passenger Request form • The passenger fill the form parts like origin and destination address • After filling the parts, the passenger click on the “Find Taxi” button to send his/her request to the system
Exit Conditions	The request will be sent to the system
Exceptions	The user did not fill all mandatory parts of the event , then the system will show an error massage

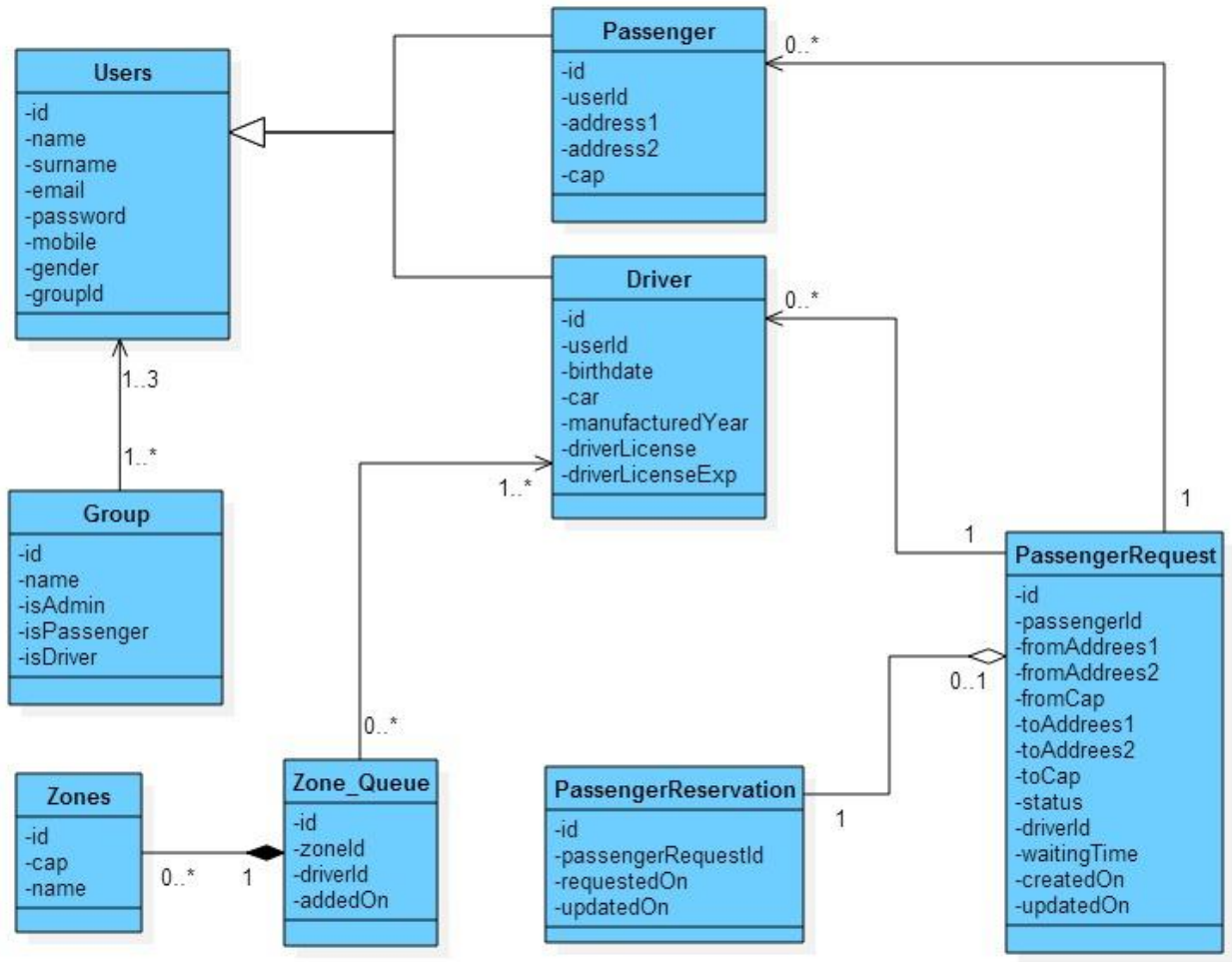
Name	Respond to Request
Actor	Driver
Entry Conditions	The driver is signed in and received a request from system
Flow of Events	• The driver is in his/her profile • The driver receives a new service request • The system shows the request massage • The driver can see the request details and then clicks on "Accept" or "Decline" button
Exit Conditions	If the driver accepts this request, the system will add this driver to the PassengerRequest and store it
Exceptions	The driver has 2 min to respond, otherwise the request will be assumed as "Decline"

Name	Modifying Reservation (Update)
Actor	Passenger
Entry Conditions	The user is signed in to the system and the reservation has been created before
Flow of Events	• The goes to the reservation part of the profile • The system shows the reservation details • The passenger click on a reservation • The passenger click on the “Edit” button • The system shows a form in which the passenger can change the previous date/time of the reservation • The passenger changes the date/time of the reservation and then click on “Save” button
Exit Conditions	The reservation on the profile will be updated and the changes will be store in database
Exceptions	• It is not possible to modify a reservation after allocating a taxi to that by the system (less than 10 min to reservation time). • Some part of the event details still remains empty , the system shows an error massage

Name	Modifying Reservation (delete)
Actor	Passenger
Entry Conditions	The user is signed in to the system and the reservation has been created before
Flow of Events	• The goes to the reservation part of the profile • The system shows the reservation details • The passenger click on a reservation • The passenger click on the “Delete” button • The system asks if he/she wants to delete it • The passenger selects “Yes” button and the reservation will be cancelled
Exit Conditions	The reservation from the profile will be deleted and the changes will be store in database
Exceptions	• It is not possible to delete a reservation after allocating a taxi to that by the system (less than 10 min to reservation time).

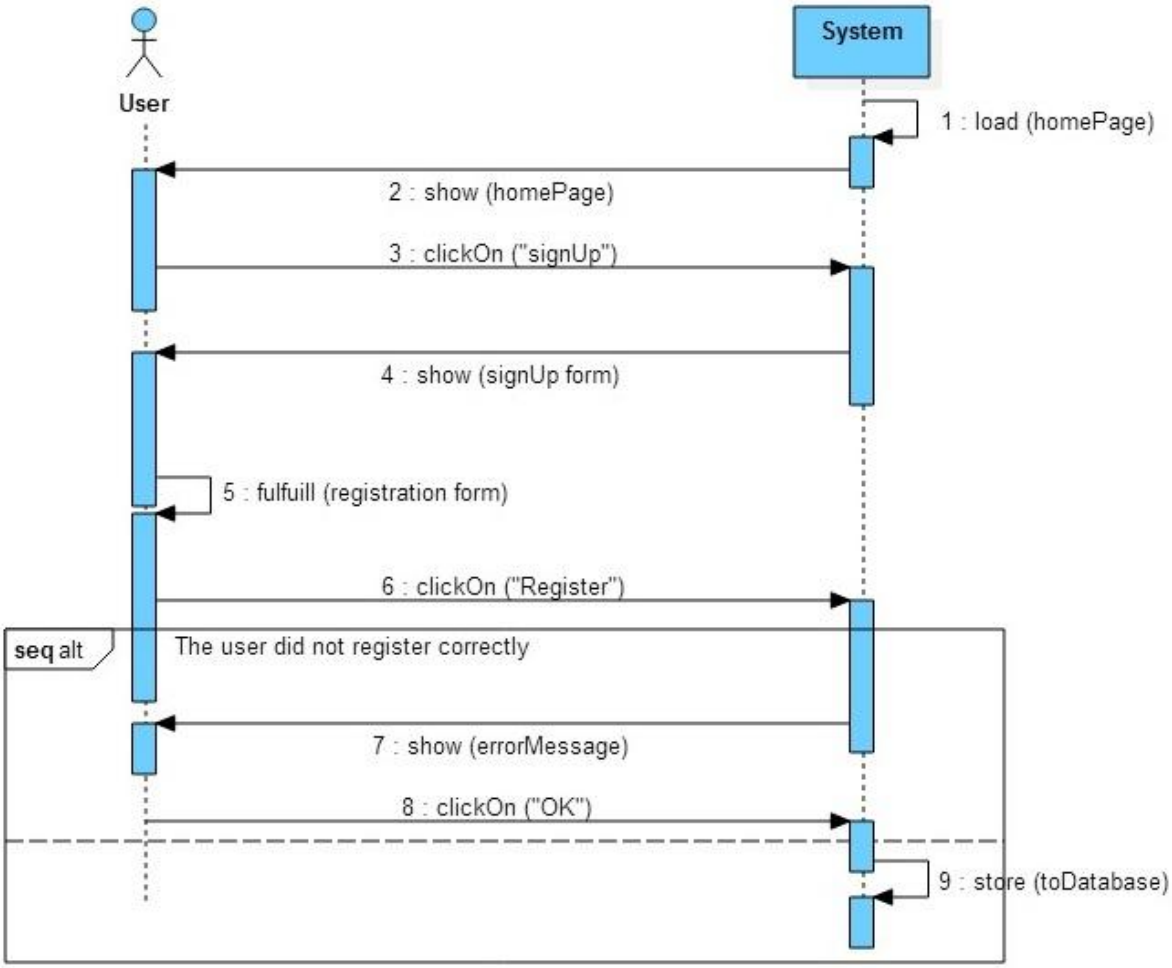
Name	Declaration of Availability
Actor	Driver
Entry Conditions	The driver is signed in
Flow of Events	• The driver is in his/her profile • The system shows the profile page in which the driver can modify his availability • The driver can declare his/her availability by selecting the "Available" or "Not Available" to the system
Exit Conditions	There is not any exit conditions
Exceptions	No exceptions

5.3 CLASS DIAGRAM

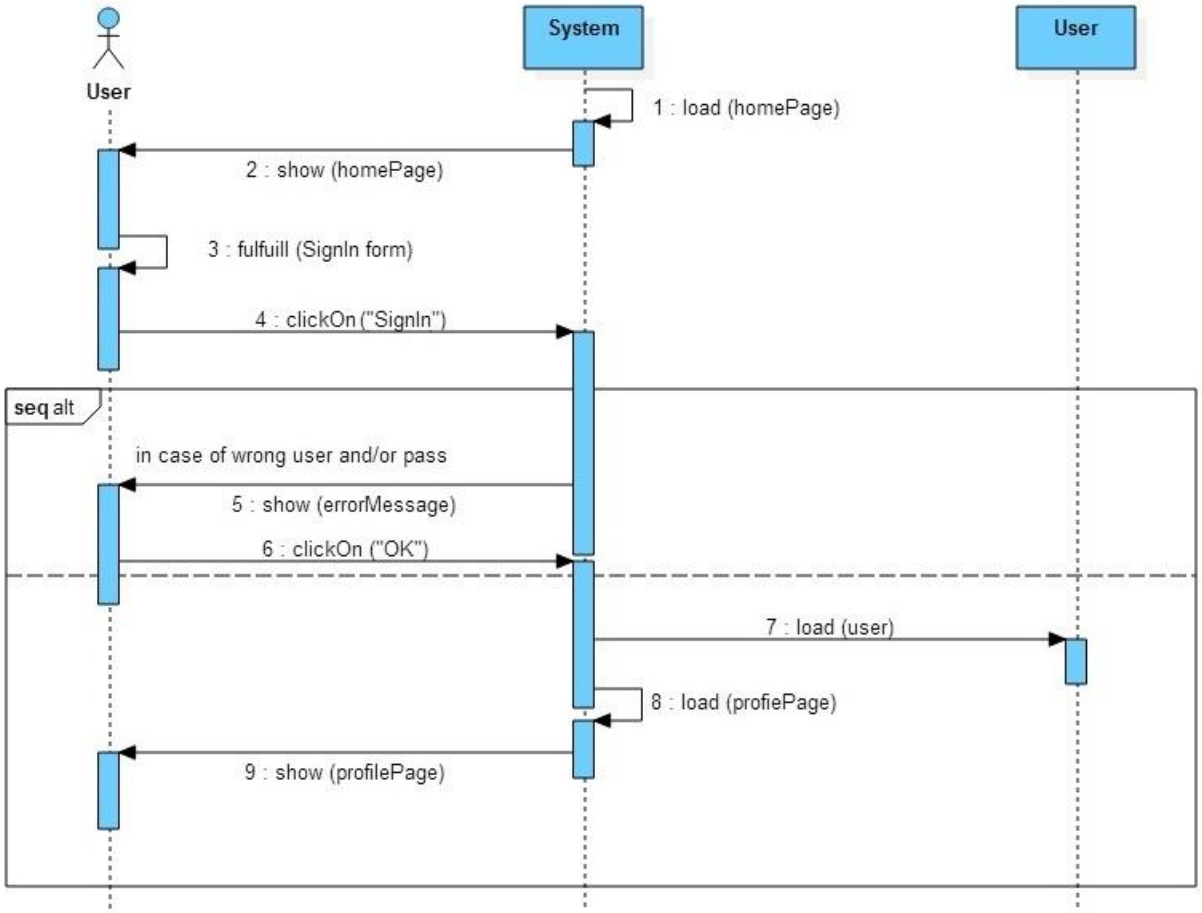


5.4 SEQUENCE DIAGRAMS

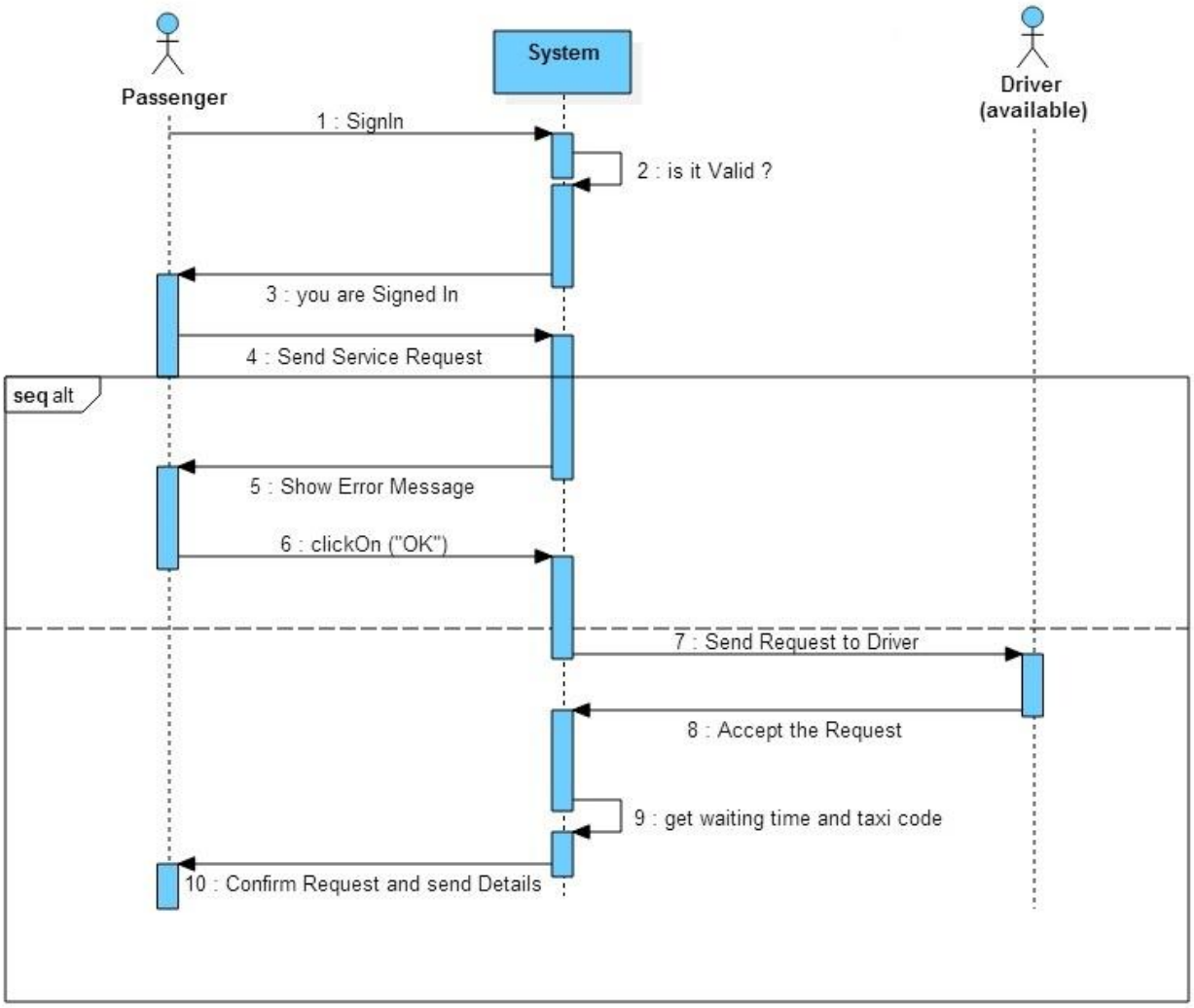
5.4.1 Sign Up



5.4.2 Sign In

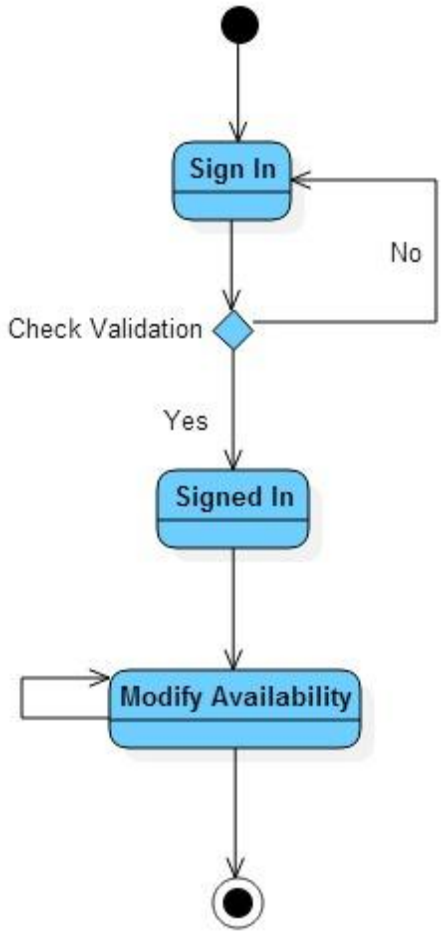

User
System
User
1 : load (homePage)
2 : show (homePage)
3 : fulfuill (SignIn form)
4 : clickOn ("SignIn")
seq alt
in case of wrong user and/or pass
5 : show (errorMessage)
6 : clickOn ("OK")
7 : load (user)
8 : load (profiePage)
9 : show (profilePage)

5.4.3 SEND AND RECEIVE REQUEST

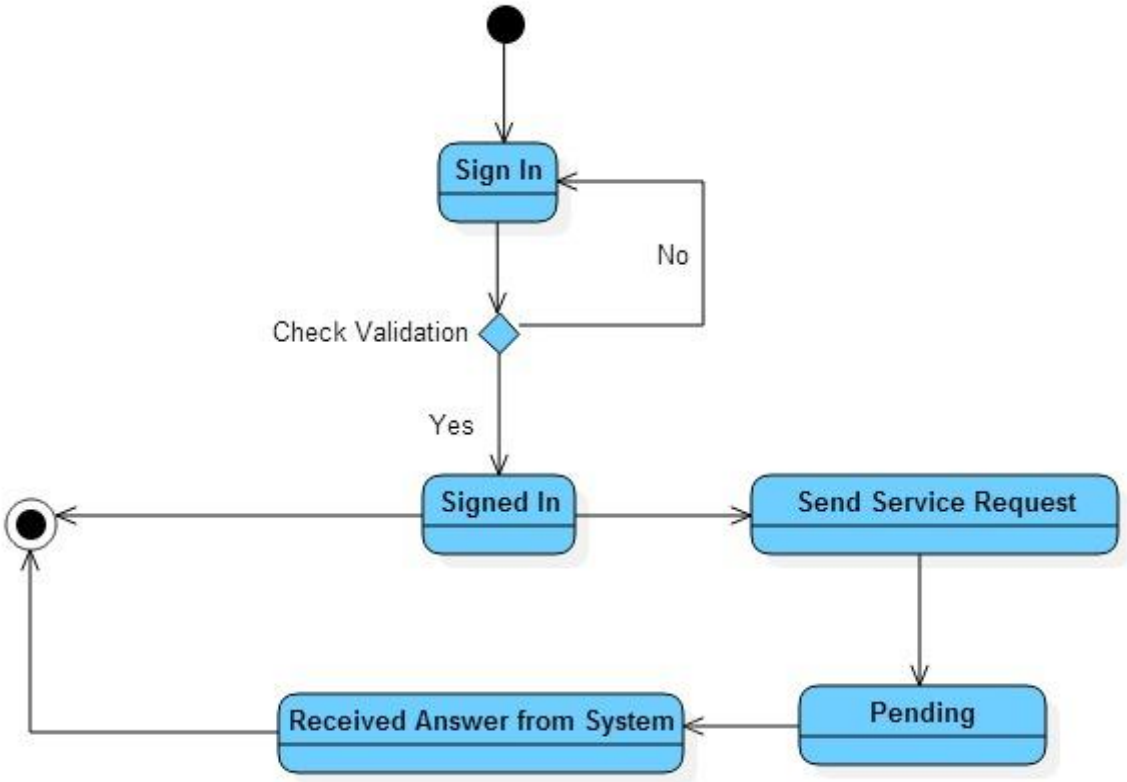


5.5 STATE CHART DIAGRAMS

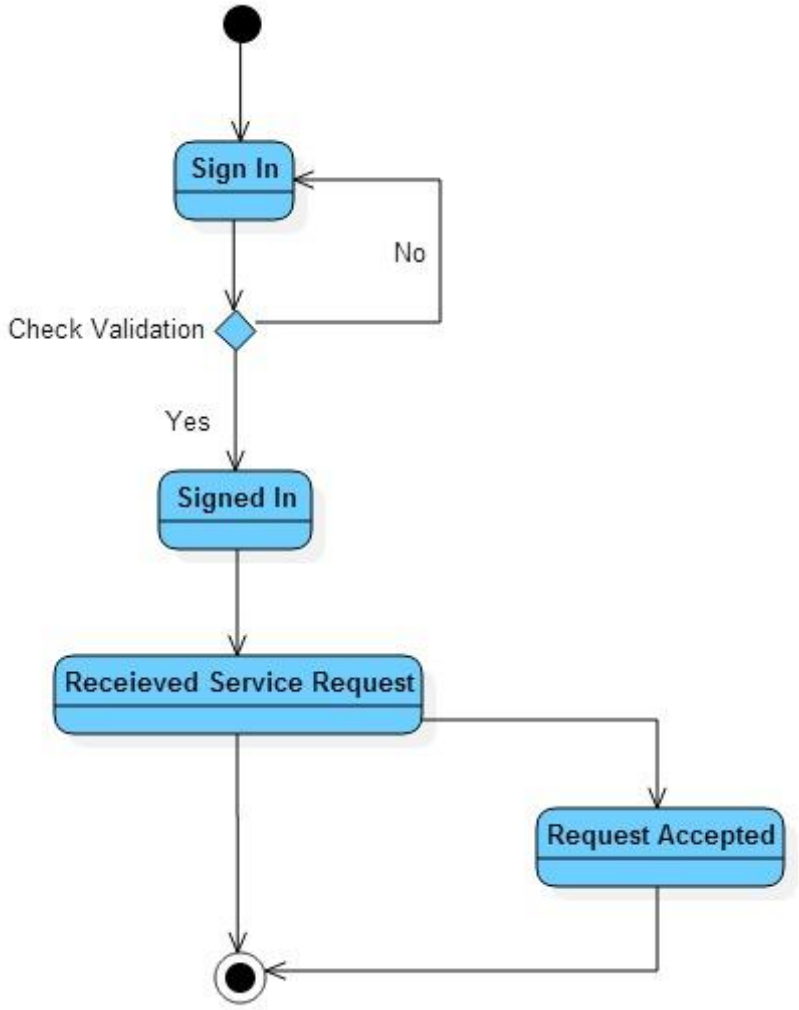
5.5.1 DRIVER AVAILABILITY



5.5.2 PASSENGER REQUEST



5.5.3 DRIVER RESPOND TO REQUEST



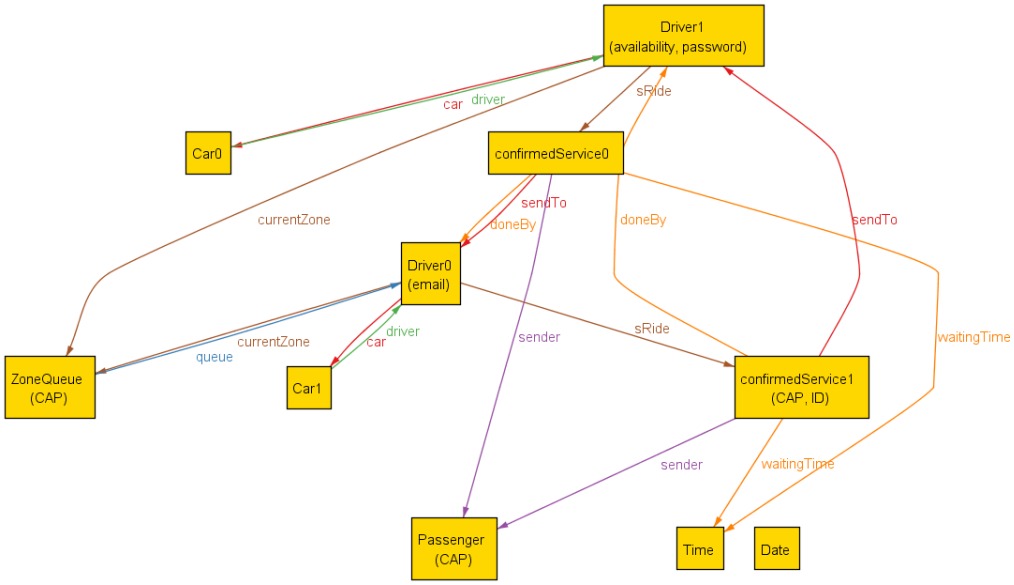
6. ALLOY MODELING

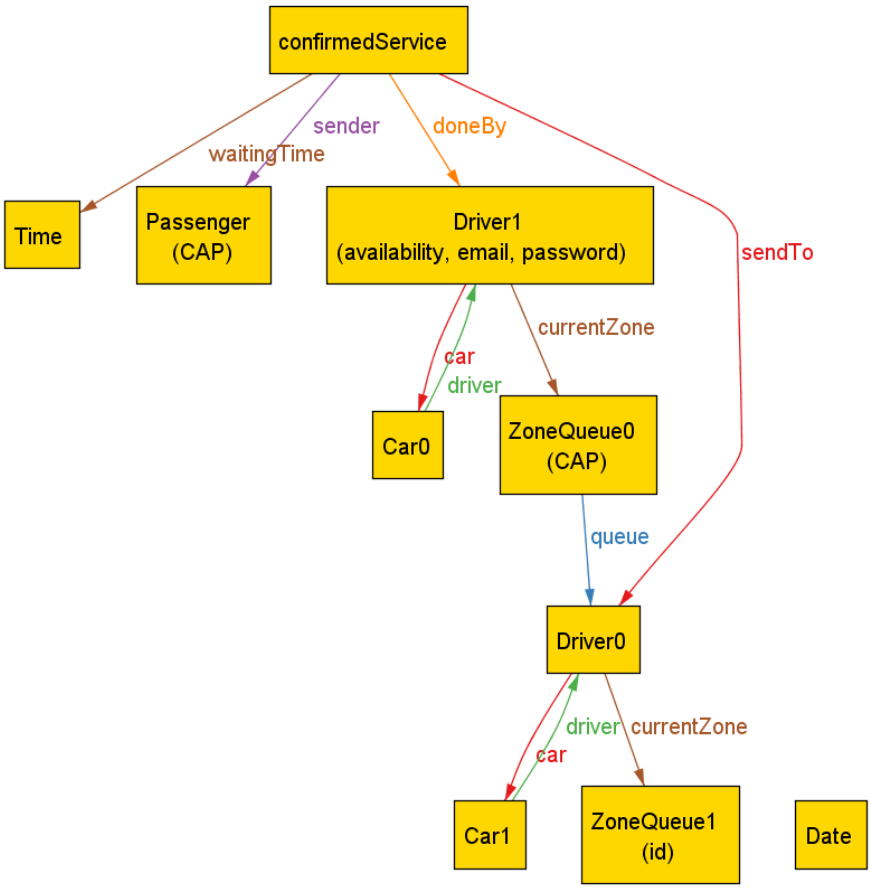
```
Executing "Run show"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   3005 vars. 273 primary vars. 4446 clauses. 424ms.
   Instance found. Predicate is consistent. 162ms.
```

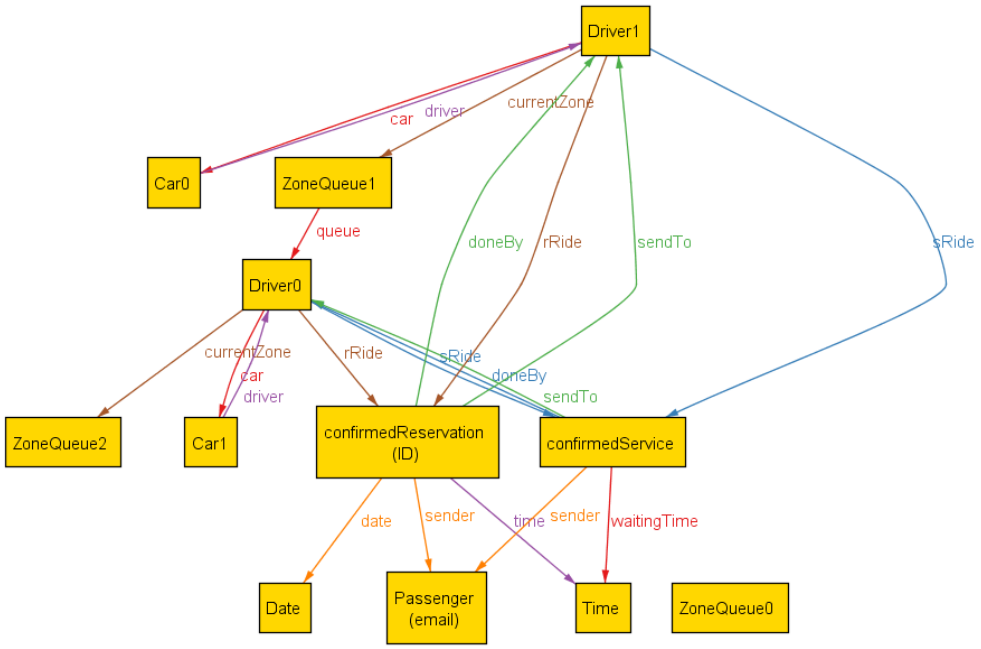
```
abstract sig User{
name: lone Txt,
surname: lone Txt,
email: one Txt,
password: one Txt
}


sig Passenger extends User{
address: one Txt,
CAP: one Integer
}


sig Driver extends User{
car: one Car,
currentZone: one Zone,
availability: one Integer,
sRide: set confirmedService,
rRide: set confirmedReservation
}

sig Car{
driver: one Driver
}

abstract sig Zone{
id: one Integer,
CAP: one Integer,
name: one Txt
}

sig ZoneQueue extends Zone{
queue: set Driver,
// topOfTheList: lone Driver
}
```

```
abstract sig Request{
ID: one Integer,
sender: one Passenger,
fromAddress: one Txt,
CAP: one Integer,
toAddress: lone Txt,
sendTo: some Driver
}

sig confirmedService extends Request{
doneBy: one Driver,
waitingTime: one Time
}

sig confirmedReservation extends Request{
doneBy: one Driver,
date: one Date,
time: one Time
}

sig Txt{}
sig Integer{}
sig Date{}
sig Time{}


//////////////////////////////////////////////////////////////
/////////    <<          FACTS              >>    //////////
//////////////////////////////////////////////////////////////

// Only the Driver who received the Service request, can confirm that Service
fact rightDriverForService{
all cS:confirmedService, r:Request, d:Driver |
cS = d.sRide implies cS.doneBy = r.sendTo
&&
cS.doneBy = r.sendTo implies cS = d.sRide
}
```

```
// Only the Driver who received the Reservation request, can confirm that Reservation
fact rightDriverForReservation{
all cR:confirmedReservation, r:Request, d:Driver |
cR = d.rRide implies cR.doneBy = r.sendTo
&&
cR.doneBy = r.sendTo implies cR = d.rRide
}

// no more than ONE driver for each Service
fact onlyOneDriverPerService{
all cS:confirmedService |
#cS.doneBy=1
}

// no more than ONE driver for each Reservation
fact onlyOneDriverPerReservation{
all cR:confirmedReservation |
#cR.doneBy=1
}


// it is not possible to create two different profiles with the same email address
fact uniqueEmail{
all u1, u2 : User | u1.email = u2.email implies u1=u2
}


//Driver and Passenger are in the same zone for Reservation or an immidiate service
fact sameZoneDriverAndPassenger {
all r:Request,  cS:confirmedService,  cR:confirmedReservation |
r.CAP = ((cS.doneBy).currentZone).CAP || r.CAP = ((cR.doneBy).currentZone).CAP
}

// There is a symmetric relation between Driver and its Car
fact symmDriverCar{
all c:Car, d:Driver | (c in d.car implies d in c.driver) && (d in c.driver implies c in d.car)
}

pred show{}
run show
```


Driver1
(availability, password)
car
driver
sRide
Car0
confirmedService0
currentZone
sendTo
doneBy
doneBy
sendTo
Driver0
(email)
sRide
waitingTime
sender
driver
car
currentZone
queue
ZoneQueue
(CAP)
Car1
confirmedService1
(CAP, ID)
sender
waitingTime
Passenger
(CAP)
Time
Date


confirmedService
sender
doneBy
waitingTime
Time
Passenger
(CAP)
Driver1
(availability, email, password)
sendTo
currentZone
car
driver
Car0
ZoneQueue0
(CAP)
queue
Driver0
driver
currentZone
car
Car1
ZoneQueue1
(id)
Date


Driver1
car
driver
currentZone
Car0
ZoneQueue1
queue
doneBy
rRide
sendTo
sRide
Driver0
currentZone
car
driver
rRide
sRide
doneBy
sendTo
ZoneQueue2
Car1
confirmedReservation
(ID)
confirmedService
date
sender
time
sender
waitingTime
Date
Passenger
(email)
Time
ZoneQueue0

7. USED TOOLS

The tools we used to create this document:

- Microsoft Office Word 2007: to redact and to format this document;
- GoMockingBird Platform: to create the UI sketches;
- StarUML: to create all diagrams.
- Alloy 4.2, to show the model consistency