

MyTaxiService

Pozzi Matteo
Scandroglio Stefano

1 Introduction

1.1 Purpose

This document represents the Design Document (DD).

The goal here is to describe more in details the structure of the myTaxiService application by providing a deeper description of how the system will be structured, both in hardware and software terms.

We will also provide a more detailed description on how users will interact with the system and how the system will behave during its usage.

1.2 Scope

The goal of the project is to develop the myTaxiService application, both in terms of mobile application and web application. Once developed, the application will allow users to request a taxi to pick them up as soon as possible, or to reserve a taxi for the future, providing information about the pick up place and the destination of the trip.

The reservation of a taxi has to occur at least two hours before the desired pick up time and may be cancelled or modified up to one hour before the requested time.

The systems will manage different queues of taxis, each one associated to a certain zone of the city, by forwarding requests that come from that zone to the first taxi in the queue of that zone.

Taxi drivers will also use this application to either accept or refuse requests forwarded to them. In case the taxi driver refuses the request, he/she will be moved at the end of the queue.

1.3 Reference documents

To better understand this document we recommend referring to the Requirements Analysis and Specification Document (RASD) we have already presented.

1.4 Document structure

We will first provide a general overview on how the system can be divided in smaller subsystems describing, for each one of those, the provided interfaces and how they interact with each other.

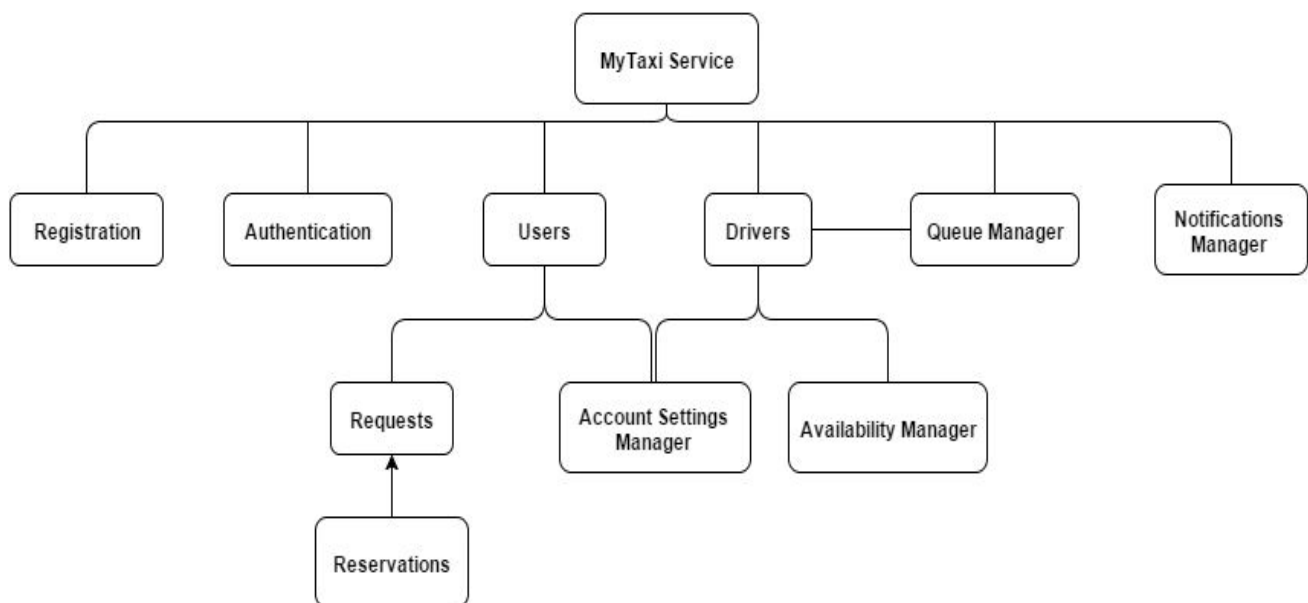
We will then provide an high level description of the components our system will have both in terms of software and hardware.

We will also provide a possible structure for the database through an ER Diagram and then we will describe how the interaction with users will occur through UX Diagrams, BCE Diagrams and Sequence Diagrams.

2 Architectural Design

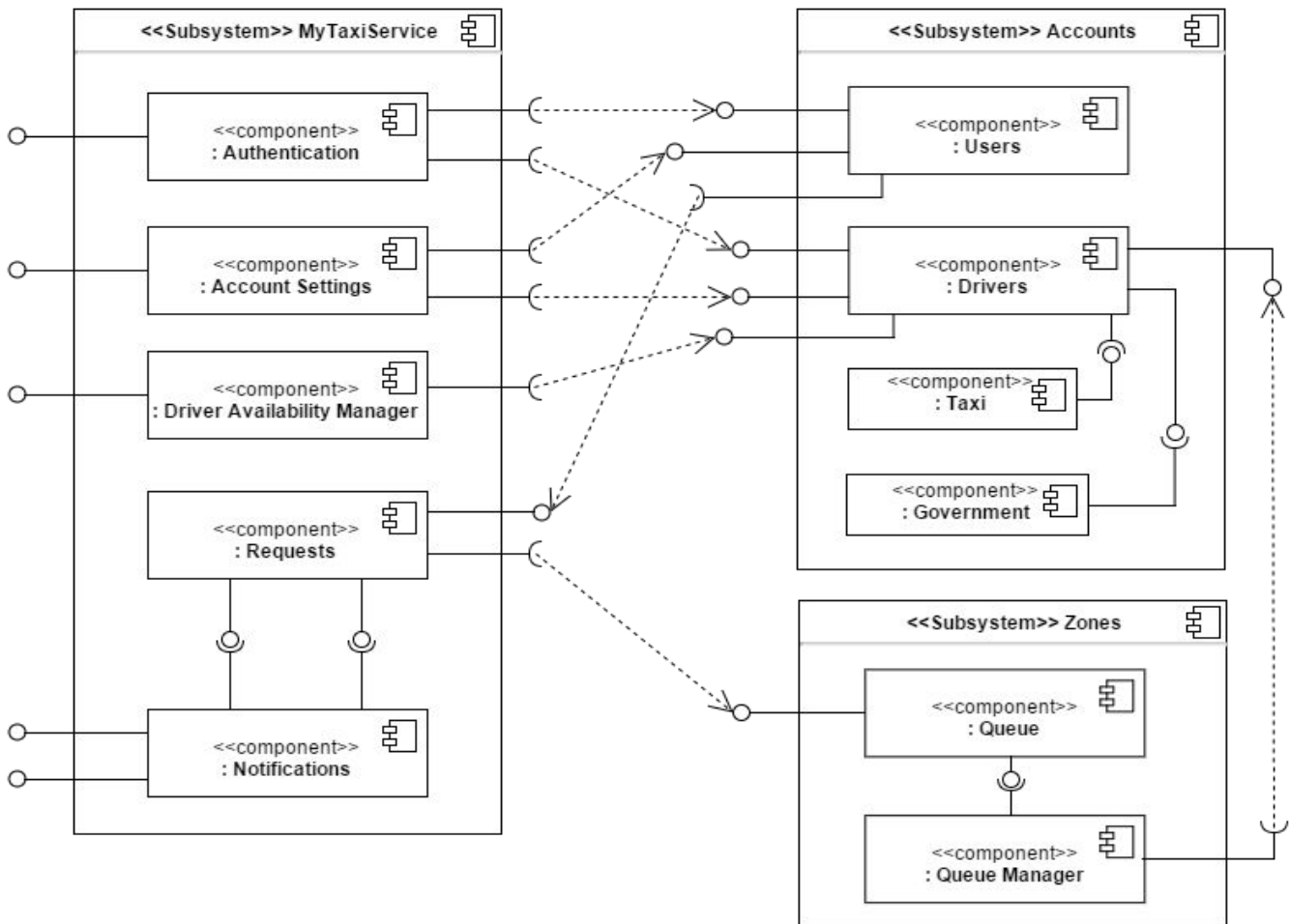
2.1 Overview

In order to analyze our application architecture, we have identified the main components of the system. They are showed in the following diagram:



2.2 Component Diagram

In this section we describe the main high-level components of the application and the interaction between them.



Now we provide a brief description of all the components in the diagram:

We have decided to divide the User and the Driver, despite a Driver is also a User, to better show the different actions made by the two components in the system.

Subsystem MyTaxiService:

- **Authentication:** it's the component that provide to the User or to the Driver the possibility to log-in to the application;
- **Account Settings:** through this component, User/Driver can edit their account settings (name, surname, email and password);

- **Requests:** this component is managed by the User, which is the only one able to make a request. It associates to the request the Driver on top of the Queue (Zone) and it exchanges informations with the Notification component;
- **Notifications:** this component notifies to the user or to the driver the informations related to the Requests component.

Subsystem Accounts:

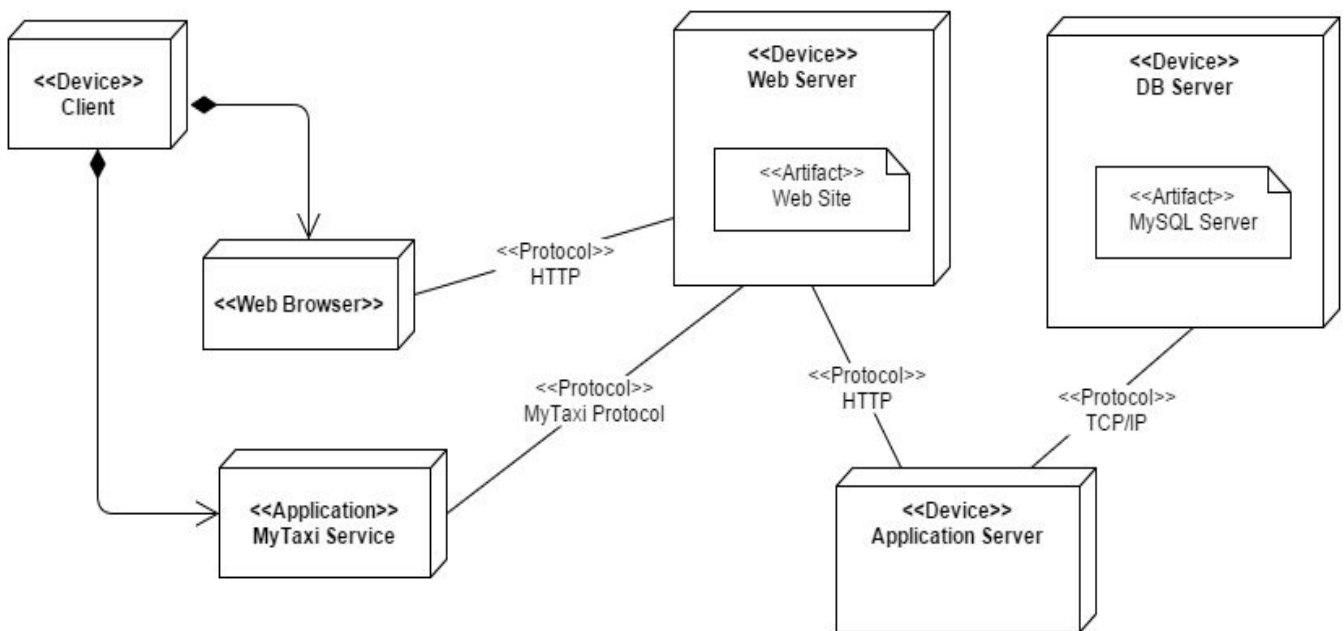
- **Users:** it's the component that manages the users logged in on the application;
- **Drivers:** this component manages all the drivers logged in on the application and all their relations with the connected components;
- **Taxi:** it's the component related to the driver's taxi, which has settings like the plate;
- **Government:** the component that manages and creates new Drivers.

Subsystem Zones:

- **Queue:** component which contains all the Drivers in the related zone. It's a FIFO structure: the first driver entered in the zone is on top of the queue, the last one is on the bottom;
- **Queue Manager:** this component manages the FIFO queue by adding new drivers entered in the zone or by removing drivers that have left the zone and both drivers refusing a request and drivers accepting it.

2.2 Deployment Diagram

In this part of the document, we give an overview of the architecture schematized in the following picture:

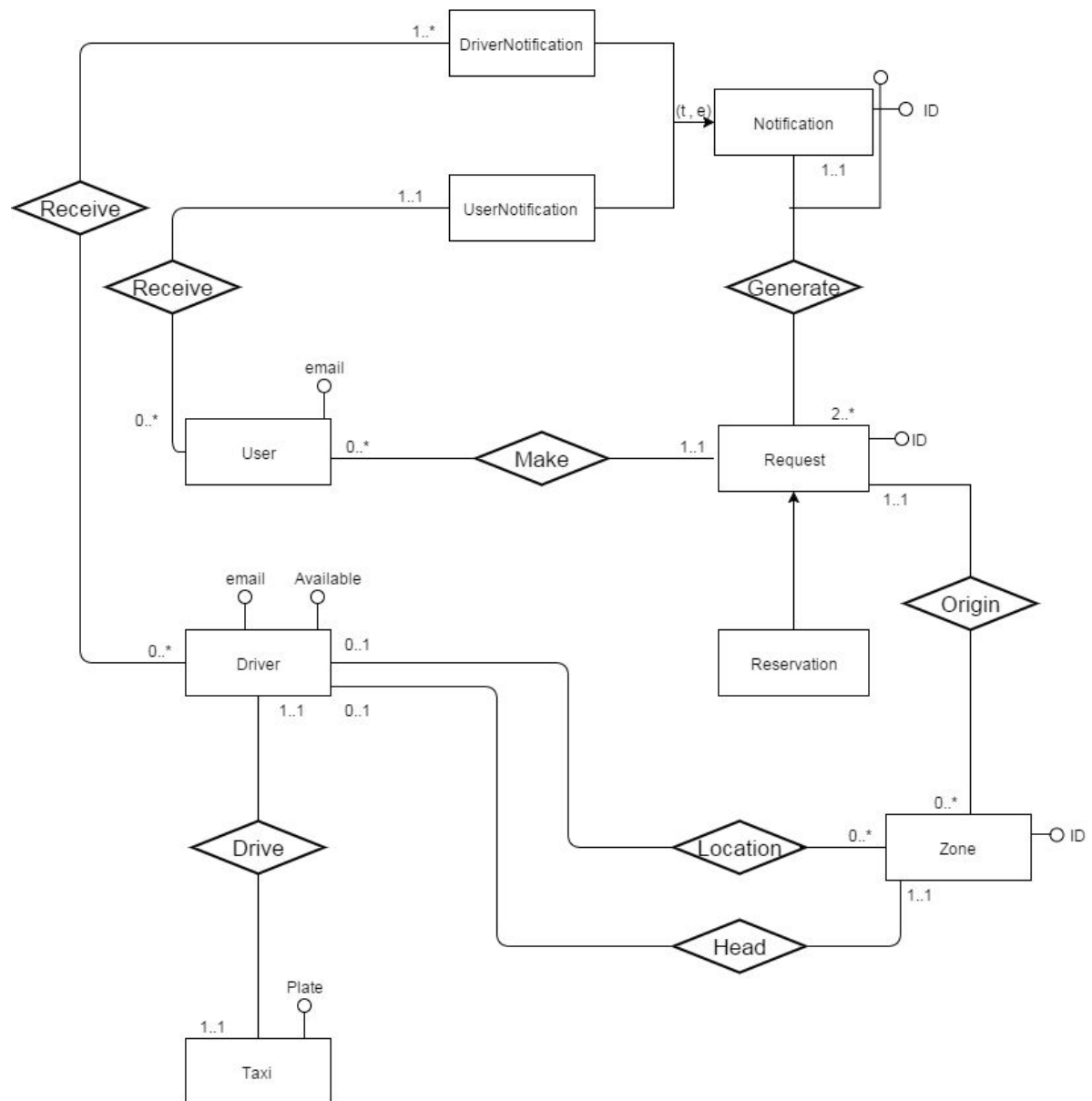


This diagram can be mapped on 3 tiers, which are:

- **Client Tier:** it contains the Client node and also the Web Browser and Application associated to it;
- **Server Tier:** it contains the two servers: Web Server and Application Server, which are connected through HTTP protocol;
- **IES Tier:** this tier contains the DB Server in which there are all the data stored in the database.

2.3 Data Management

To store the data needed for our application we will use a relational database. Here we describe its structure through a ER Diagram.



To keep everything clear and simple in this diagram we have decided to show only the basic structure of entities, without stressing on their various attributes which will be detailed in the following of the document.

That schema will result in the following tables:

- User(email , password , name , surname)
- Driver(email , *taxi* , password , name , surname , available)
- Taxi(plate , model , year)
- Zone(ID , *headOfQueue* , coordinate)
- Request(ID , *user* , *zone* , pickUpLocation , accepted)
- Reservation(ID , *user* , *zone* , pickUpLocation , destination , pickUpTime)
- UserNotification(ID , *user* , *request* , *taxiCode* , ETA)
- DriverNotification(ID , *driver* , *request* , *zone* , status)

**Notation: Primary Key , Foreign Key , Generic Attribute*

With such a translation we have identified the following foreign key links:

- **Driver:**
 - Driver.taxi > Taxi.plate
- **Taxi:**
 - Taxi.zone > Zone.ID
- **Zone:**
 - Zone.headOfQueue > Driver.email
- **Request:**
 - Request.user > User.email
 - Request.zone > Zone.ID
- **Reservation:**
 - Reservation.user > User.email
 - Reservation.zone > Zone.ID
- **UserNotification:**
 - UserNotification.user > User.email

- UserNotification.request > Request.ID
- UserNotification.taxiCode > Taxi.plate
- **DriverNotification:**
 - DriverNotification.driver > Driver.email
 - DriverNotification.request > Request.ID
 - DriverNotification.zone > Zone.ID

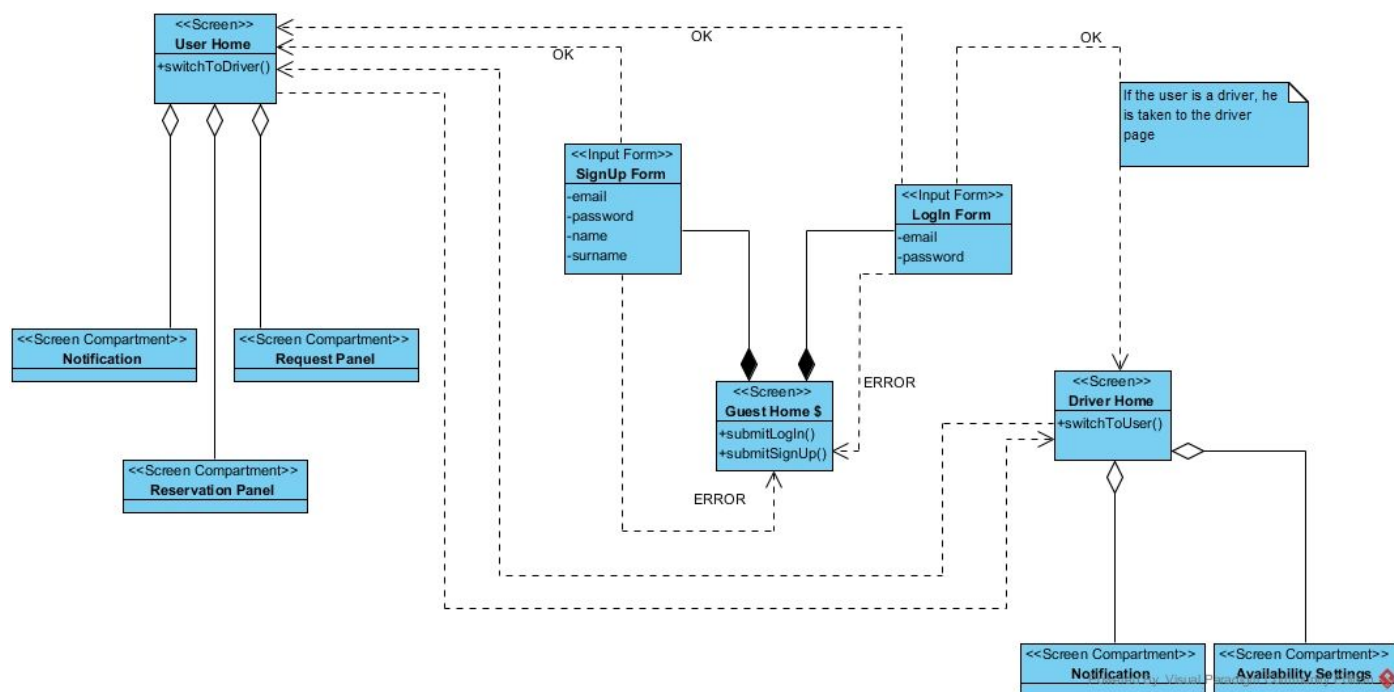
2.4 User Experience

In this section of the document we will present how users and drivers can interact with the application at presentation layer.

Such a description will be given through User Experience Diagrams that are composed of three main paradigms:

- <<screen>>: represent web pages
- <<screen compartment>>: represent portion of web pages that implement a particular function
- <<input form>>: represent part of a web page that contains text field that can be filled by users

2.4.1 General Overview



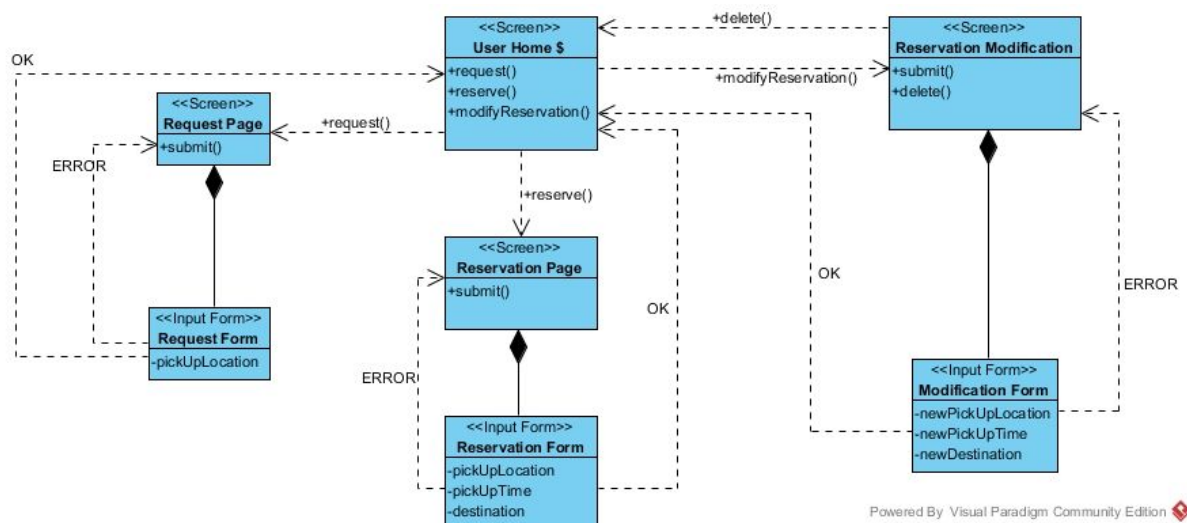
From this diagram is possible to see how the sign up and login functionalities will be provided to both users and drivers.

You can see that, after connecting to the service, users will be able to sign up or login, while drivers will only be able to login.

After the login phase drivers will be taken to their driver page from which they will be able to see incoming request and either accept or refuse it; they will also be able to change their availability settings or switch to their user profile.

On the other hand users, once they've logged in the system, will be able to see incoming notification, make new requests or reservations; they will also be able to switch to their driver account if they own one.

2.4.2 Request and Reservation

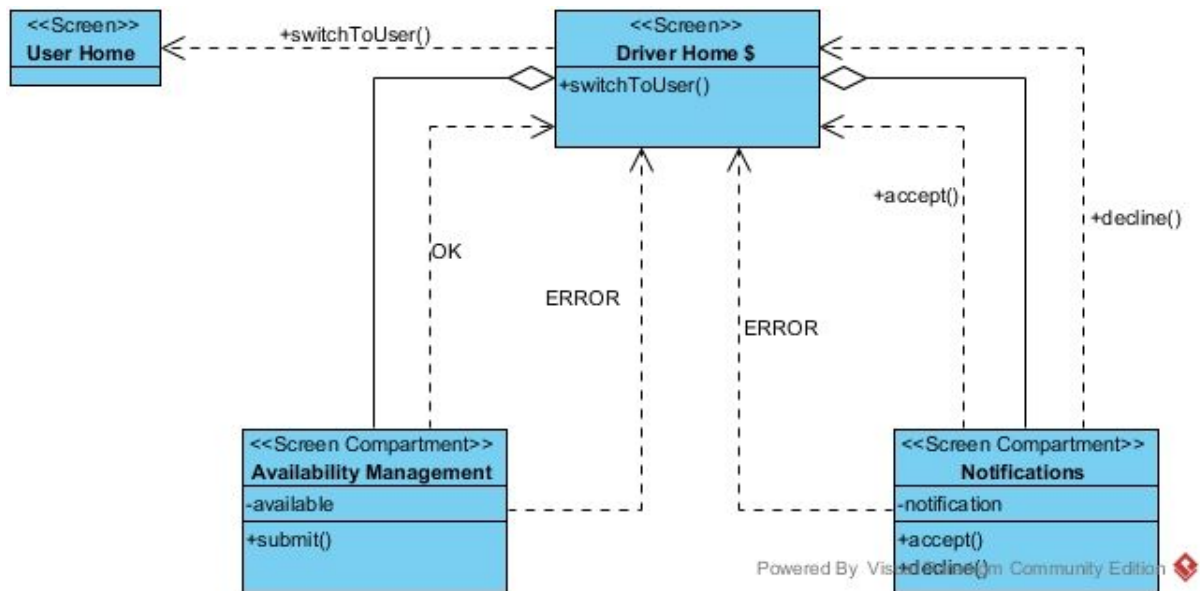


In this diagram we show how users can create a new request, a new reservation or edit an already existing reservation.

You can see that we've identified three pages:

- **Request Page:** is the page from which users will be able to enter the details that are required to make a request for a taxi.
- **Reservation Page:** is the page from which users will be able to enter the details that are required to make a reservation for a taxi.
- **Reservation Modification:** is the page from which users will be able to either modify an existing reservation or to delete it.

2.4.3 Driver Management



You can see that, from his home page, a driver is able to modify his availability settings (from available to unavailable and viceversa), and, if he has an incoming request he can either accept or refuse it.

From his home page he is also able to switch to his user account (see the previous diagram to understand how the interaction within a user page is deployed).

2.5 BCE

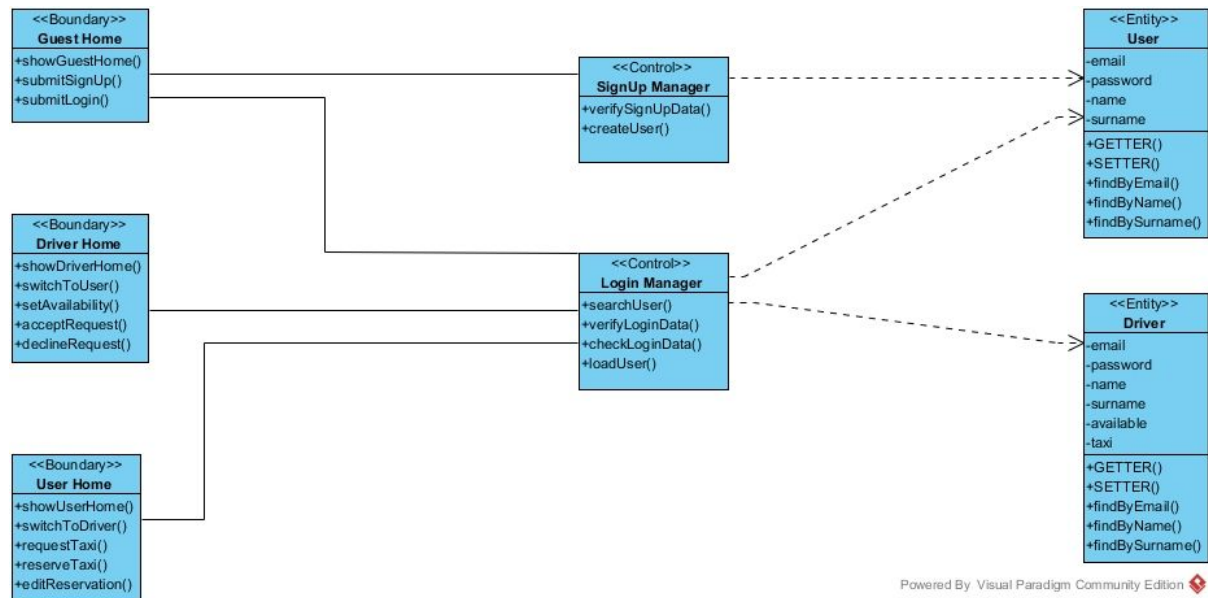
In this section of the document we provide an overview of the system, focusing on components and describing how they interact with each other. This description will be given using the Boundary-Control-Entity pattern, in particular:

- **<<boundary>>** represents components that interact directly with the user and generally they gather **<<screen>>** from UX diagrams.
- **<<control>>** represent parts of the system containing the logic of the application. In particular they allow the communication between components and check if everything works correctly.
- **<<entity>>** represent modules in charge of accessing the data stored in the database.

We had to split the systems in 4 diagrams to make everything readable and to focus only on few aspects at a time.

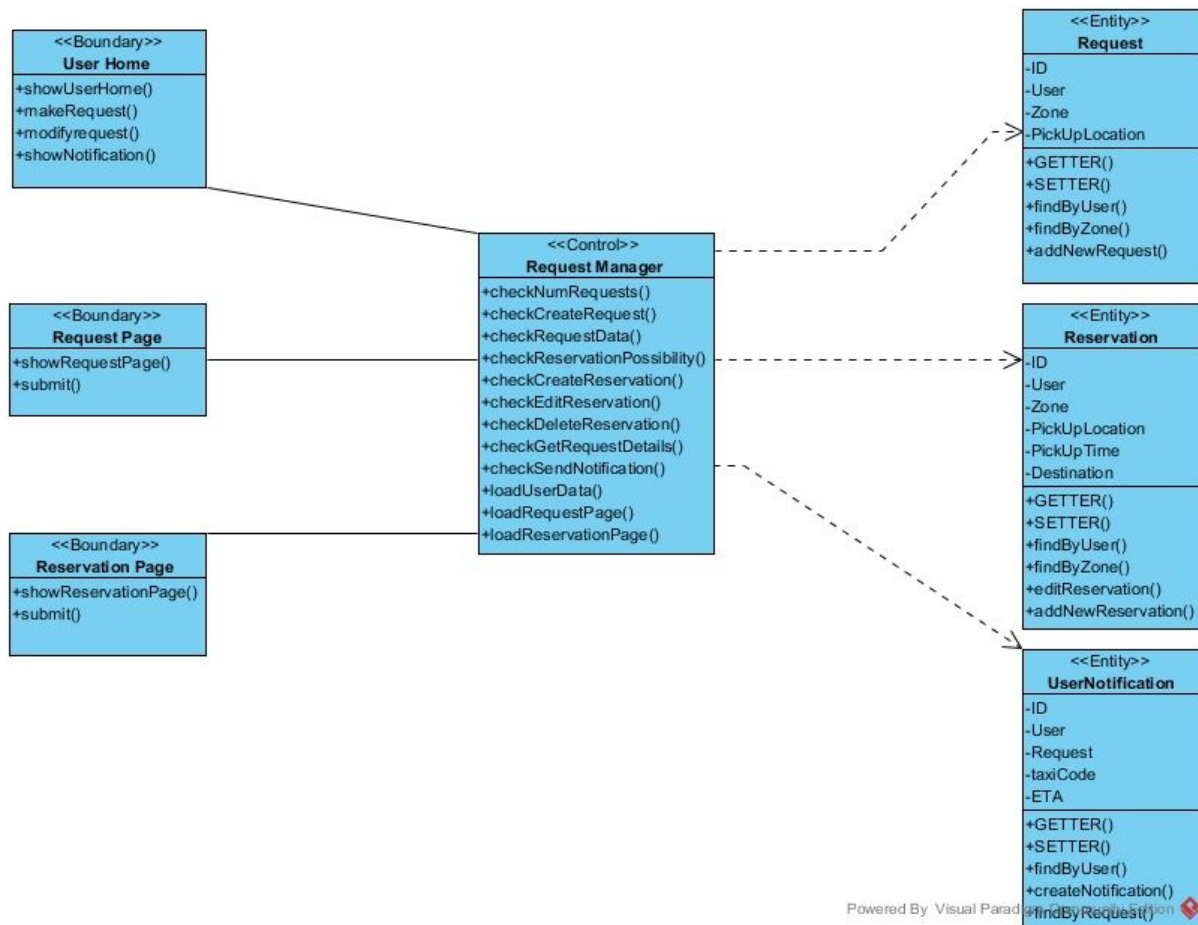
For each diagram we will also provide a brief description of the <<control>> modules as the <<boundary>> and <<entity>> modules have already been described in previous sections (respectively in UX and ER diagrams).

2.5.1 Login and registration



- **SignUp Manager:**
 - is in charge of checking if the data input by a guest is valid and consistent and, in such a case, add a new user to the database.
- **Login Manager:**
 - is in charge of verifying the correctness of data input by a user during the login phase and, if the test is passed it redirects the user to his home page.

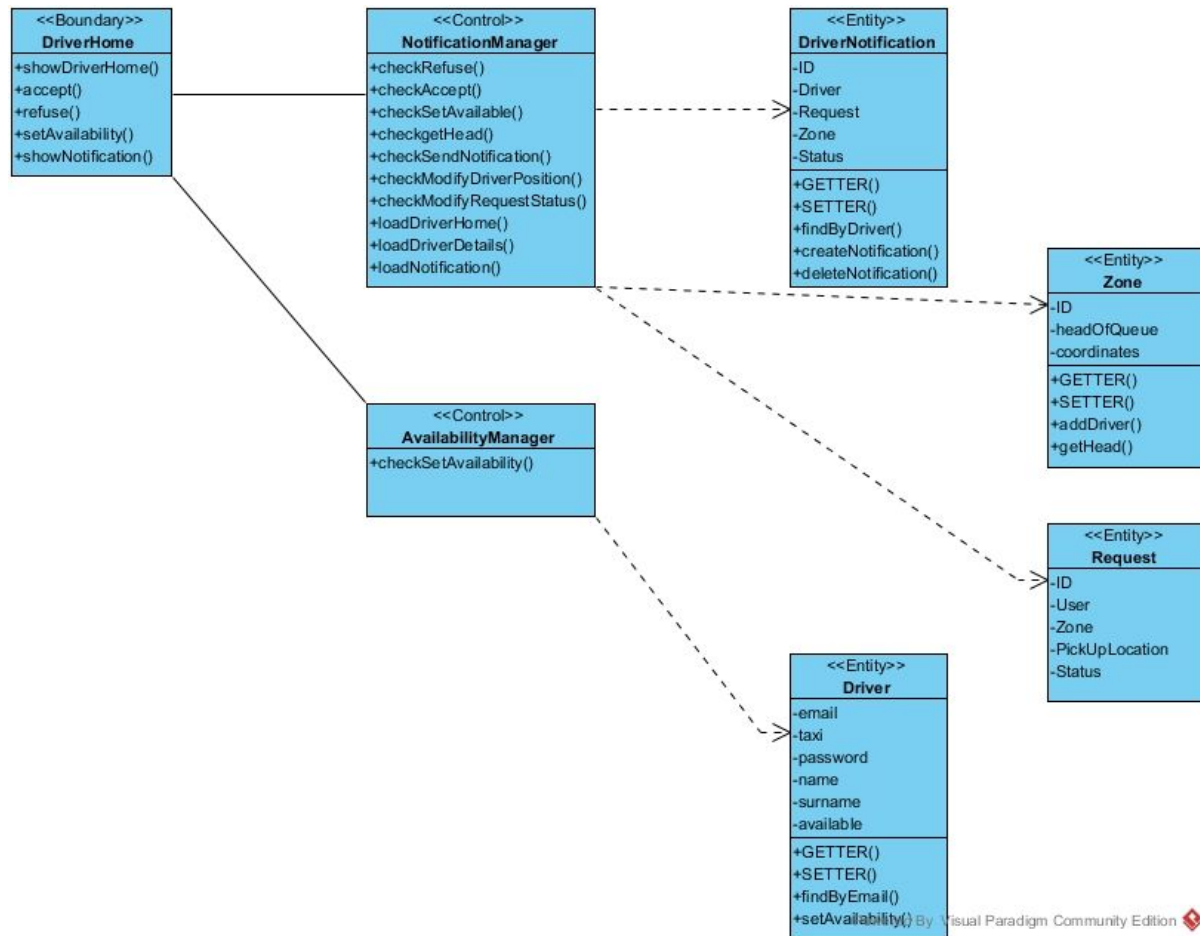
2.5.2 Request and Reservation



- **Request Manager:**

- checks if a user has the rights to request or reserve a taxi.
- if the test is passed, checks for the validity of the data input by the user and, if everything is correct, creates a new request/reservation in the database.
- checks for the validity of data user input to modify the details of an already existing reservation.
- sends notifications to users informing them about the details of a request/reservation they have made.

2.5.3 Driver



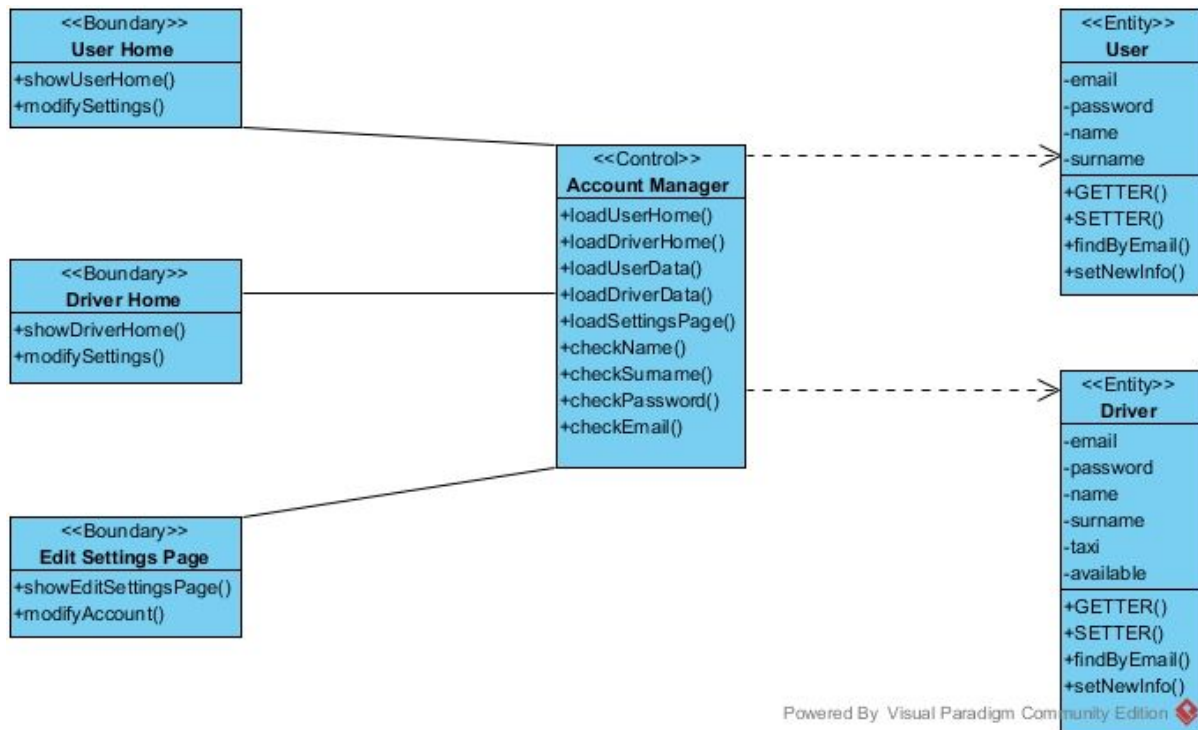
- **Notification Manager:**

- sends notifications about new requests to the driver who is the head of the queue in which the request originated.
- checks if such a notification has been accepted or refused by a driver and modifies the queue of the zone accordingly.

- **Availability Manager:**

- checks if a modification of the available status of a driver is valid and consistent with what is stored in the database.

2.5.4 Profile Manager



- **Account Manager:**

- checks for the validity and consistency of the data input by drivers and users when modifying their profile and then modifies the desired informations.

2.6 Sequence Diagrams

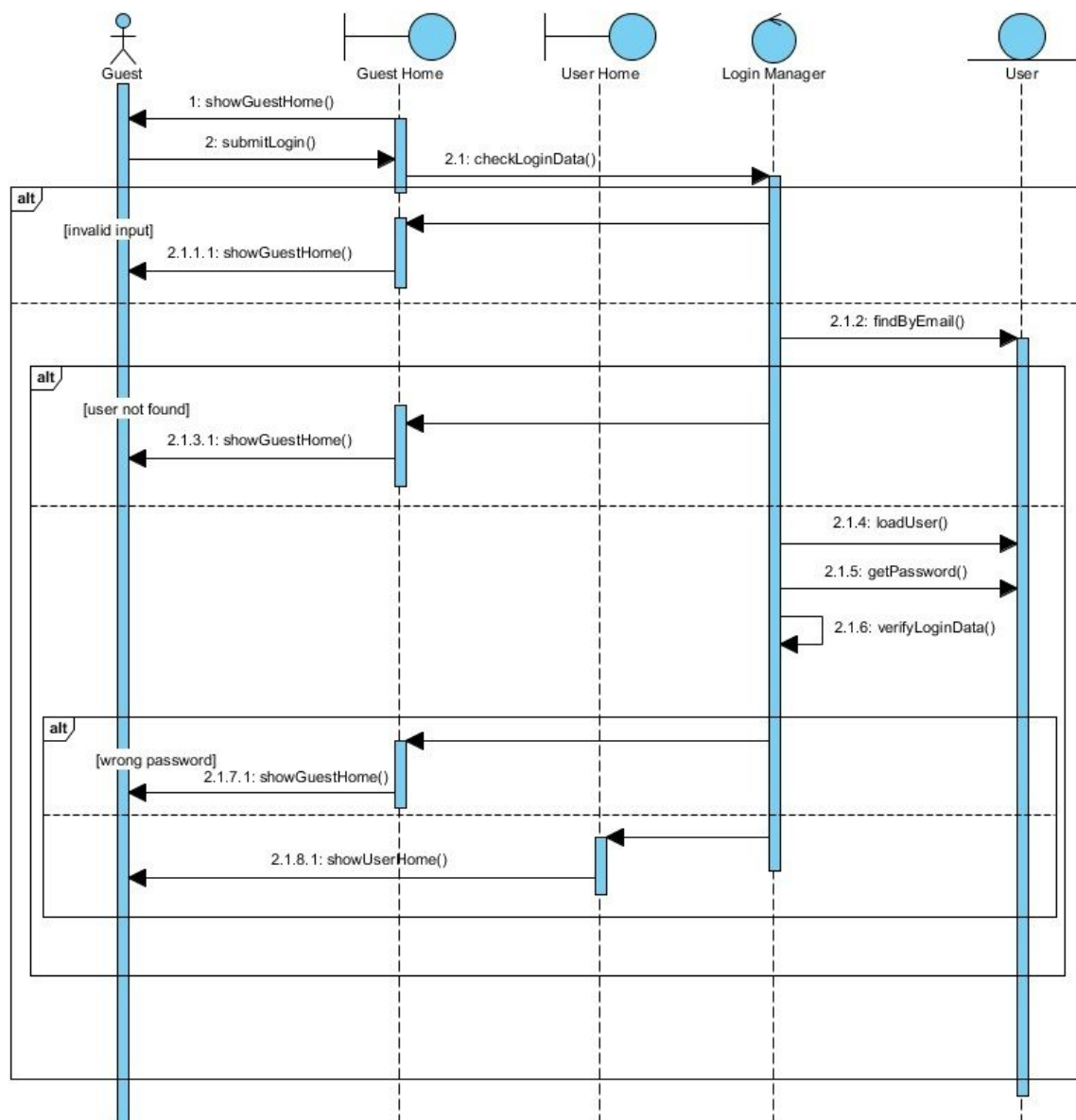
We provide now some Sequence Diagrams in order to describe how some of the main functionalities of the system are provided.

We use boundaries, controls, entities and methods described in BCE Diagrams in the previous section.

2.6.1 Login

Actor: Guest

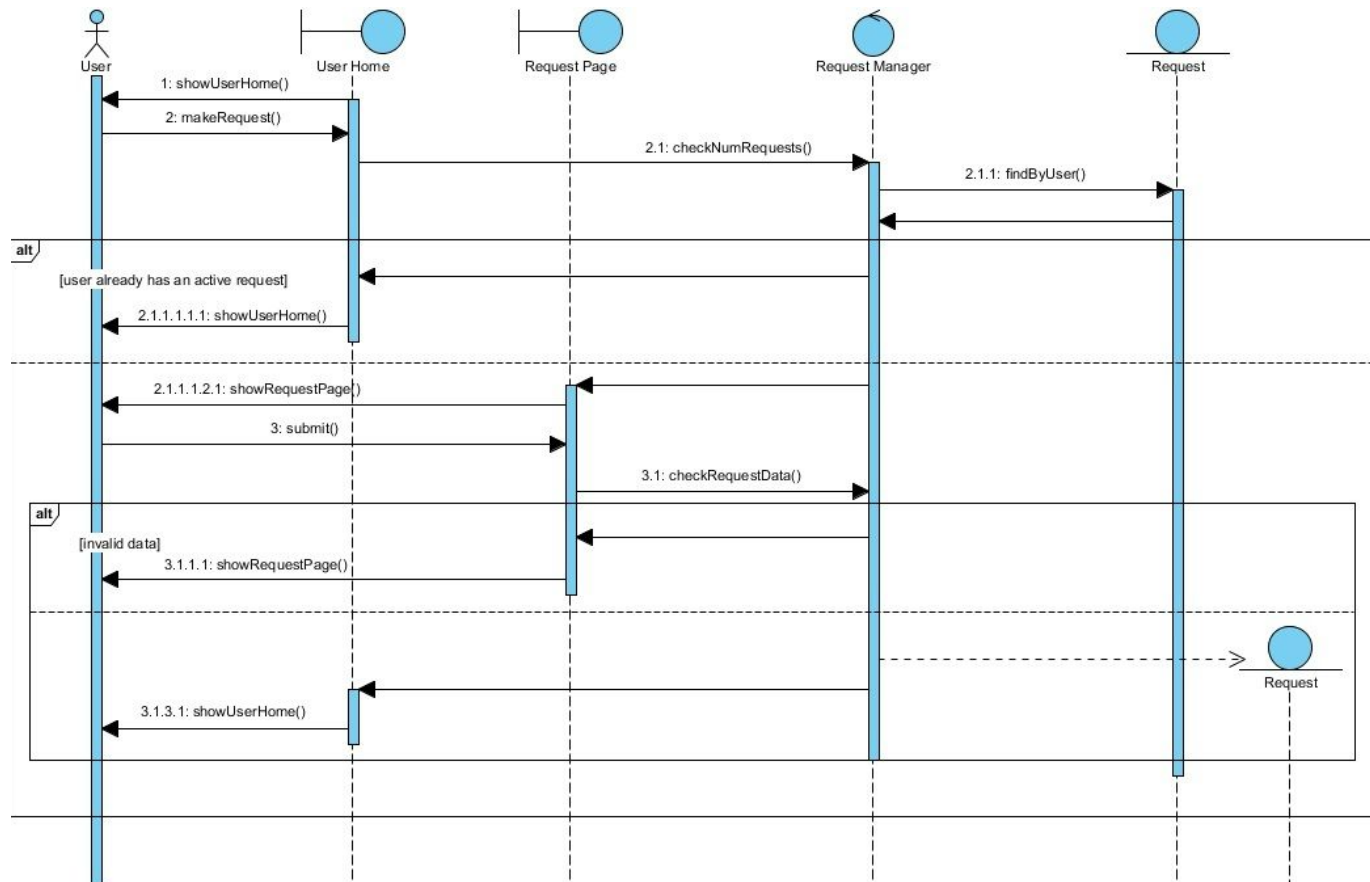
Flow: Guest accesses the application (both mobile and web) and logs in the system by inputting his credentials.



2.6.2 Request

Actor: User

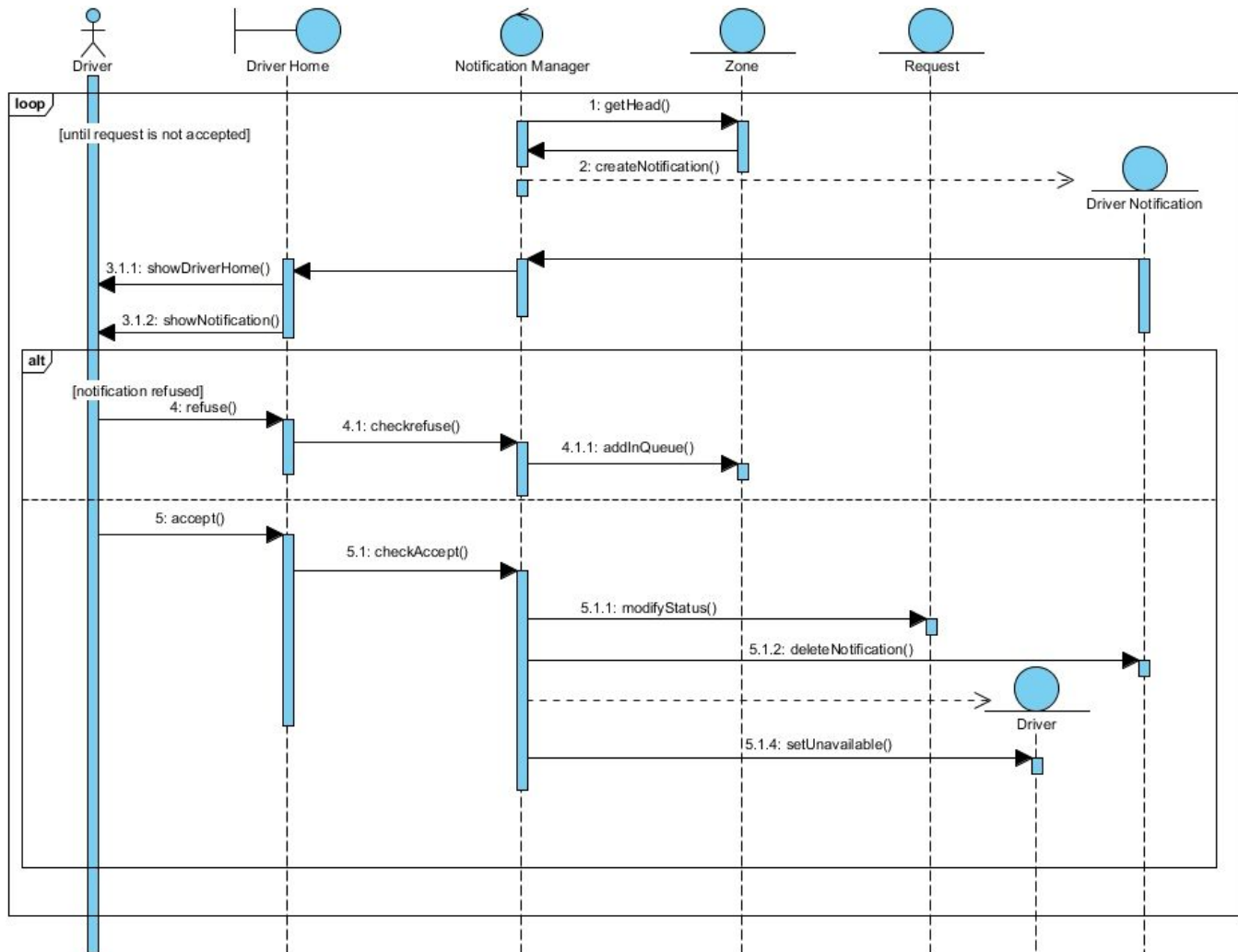
Flow: User makes a new request for a taxi. The system checks if that user hasn't any already active request and, if that's the case, creates a new request in the database with the informations specified by the user.



2.6.3 Driver

Actor: Driver

Flow: When a new request is created by a user, the system sends a notification to the driver which is the head of the queue of the zone from which the request originated. That driver can either accept or refuse that request.

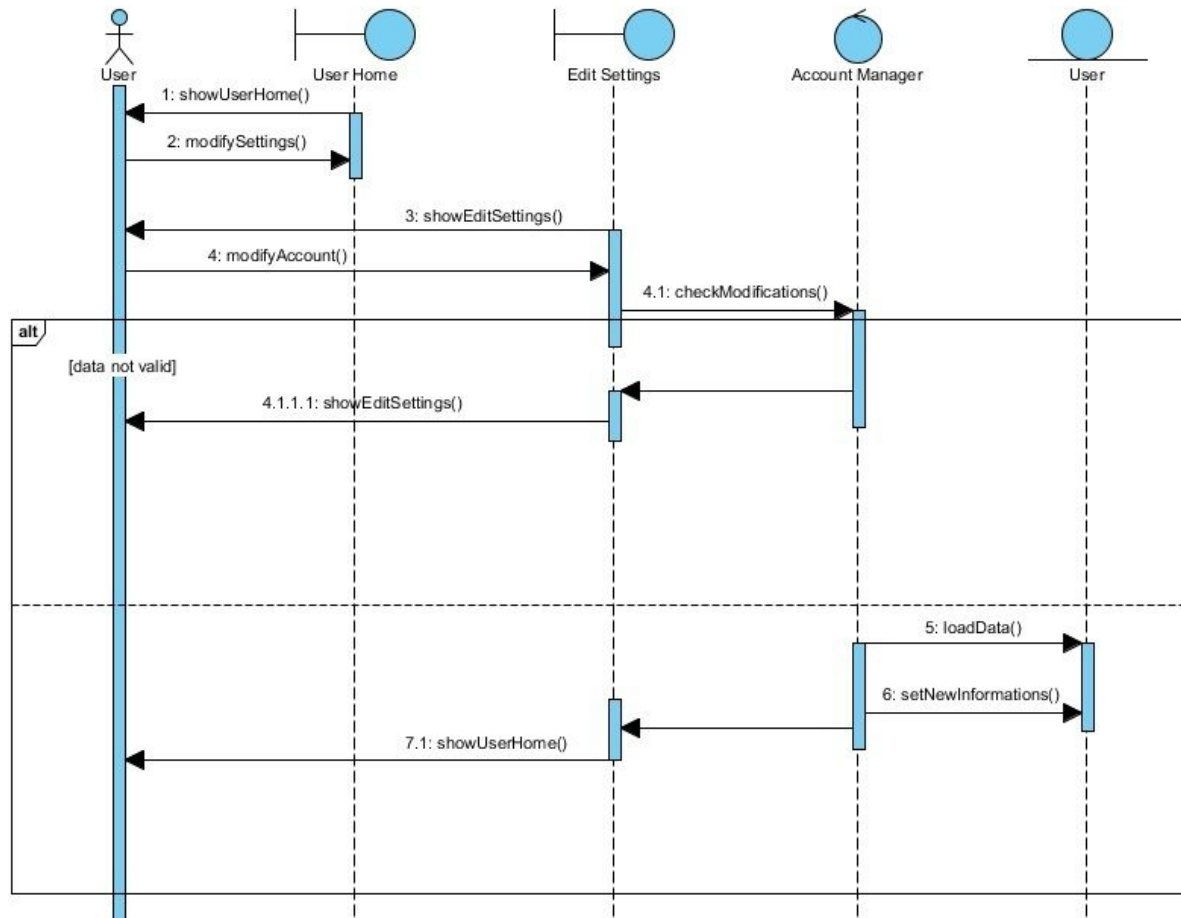


See the algorithm section for the pseudocode of getHead() and addInQueue() functions.

2.6.3 Account Managing

Actor: User

Flow: User accesses the Modify settings page and modifies his informations (name, surname, email, password). Then, the system checks wheter those modifications are valid and, if they are, updates the database.



3 Algorithm Design

We provide now a description of the most important algorithm in our application. This algorithm will be in charge of managing the different queues of taxis, sending notifications to drivers and updating their position inside each queue. The description will be at high level of abstraction but, it will be detailed enough to properly understand how the algorithm works.

```
void forwardNotification(Notification n){
    while(n.request.accepted==no){
        z=n.request.zone;
        d=z.getHead();
        if(d!=null){
            sendNotification(d);
            if(d accepted request){
                n.request.accepted=yes;
                d.setAvailable(no);
            } else{
                z.addInQueue(d); //already removed by getHead()
            }
        } else{
            print("Sorry! No drivers available.");
        }
    }
}
```

Now we will provide the two main functions in *forwardNotification()*, which are *getHead()* and *addInQueue()*:

```
driver getHead(){
    if(queue.size>1 && queue.head!=null){
        tmp=head;
        head=head.next;
        return tmp;
    } elseif(queue.size==1){
        return head;
    } else{
        return null;
    }
}
```

```
void addInQueue(Driver d){
    if(head==null){
        head=d;
        head.next=null;
    } else{
        tmp=head;
        while(tmp.next!=null){
            tmp=tmp.next;
        }
        tmp.next=d;
        d.next=null;
    }
}
```

4 User Interfaces Design

In this section we will provide additional mockups for User Interfaces with respect to those presented in the RASD document.

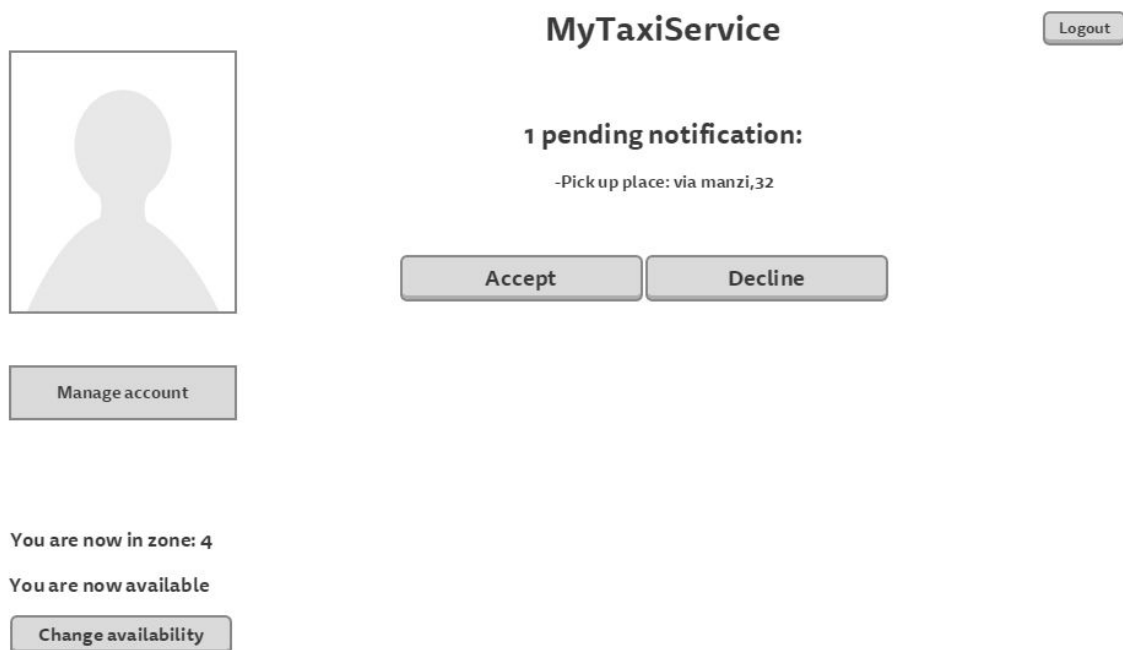
We will cover most of the functionalities provided by the application and, for each interface, we will provide both the mobile version and the web version.

4.1 Home Page

Here we present mockups for the home page of both user and driver.



User Web home page



Driver Web home page

My Taxi Service

Notification:

- Taxi Code: 9430

- ETA: 10 min

New Request

New Reservation

Edit Reservation

Account Settings

Switch to driver

Logout

User mobile home page

My Taxi Service

Informations:

- You are available

- You are in zone 7

Pending Request:

- Pick Up Location: via Manzi,32

A

R

Switch to user

Set availability


Account Settings

Logout

Driver mobile home page

4.2 Taxi Request Page

Here we show the page that allows user to request a new taxi.



Modify/delete reservation

Manage account

MyTaxiService

Logout

Request details: Pick Up Location

Input Desired Pick Up Location

Submit

Web request page

My Taxi Service

New Request

-Pick Up Location:


Desired Pick Up Location

Submit

Mobile request page

4.3 Taxi Reservation Page

Here we show the page that allows user to reserve a new taxi.



Modify/delete reservation

Manage account

MyTaxiService

Reservation details: Pick Up Location

Input Desired Pick Up Location

Pick Up Time

Input Desired Pick Up Time

Destination

Input Destination

Submit

Logout

Web reservation page

My Taxi Service

New Reservation

-Pick Up Location:

Desired Pick Up Location

-Pick Up Time

Desired Pick Up Time

-Destination


Desired Destination

Submit

Mobile reservation page

4.4 Account modification page

From this page users and drivers can modify their account informations.



Modify/delete reservation

Manage account

MyTaxiService

Logout

Account Details: Email: mariorossi@mail.com

Input new email

Password:

New Password

Confirm Password

Name: Mario

New Name

Surname: Rossi

New Surname

Submit

Web account modification page

My Taxi Service

Email: mariorossi@mail.com

New email

Password:

New Password

Confirm Password

Name: Mario

New Name

Surname: Rossi

New Surname

Mobile account modification page

5 Requirements traceability

Now we provide a mapping between the requirements defined in the RASD and the components described in the previous sections of this document.

For an accurate description of each component refer to the Component Diagram provided in section 4.2.

- Registration and login > Authentication component
- Request and reservation management > Requests component
- Queue management > Queue Manager component
- Driver availability management > Driver Availability Manager
- Driver request and reservation management > Requests component and Notification component
- Driver account distribution > Government component
- Notifications > Notification component
- Account managing > Account Settings Manager

6 Conclusion

To redact this document we used the following tools:

- Google Docs (<https://docs.google.com/>) to redact and format the document
- Mockingbird (<https://gomockingbird.com>) to create mockups of UI
- Star UML (<http://staruml.io/>) to design Use Case diagrams
- Visual Paradigm (<http://www.visual-paradigm.com/>) to design Sequence and Class diagrams

To create this document we spent approximately 20 hours each.