

My Taxi Service

REQUIREMENTS ANALYSIS AND SPECIFICATION DOCUMENT



Julián David Gallego García

Politecnico di Milano November 22, 2015





CONTENTS

1. Introduction	- 4 -
1.1. Purpose	- 4 -
1.2. Scope	- 4 -
1.3. Definitions, acronyms, abbreviations Reference documents	- 5 -
1.3.1. Definitions	- 5 -
1.3.2. Acronyms	- 5 -
1.3.3. Abbreviations	- 5 -
1.3.4. Reference documents.....	- 5 -
1.4. Overview	- 6 -
2. Overall Description	- 6 -
2.1. Product perspective.....	- 6 -
2.2. Product functions	- 6 -
2.3. User characteristics	- 7 -
2.4. Constraints.....	- 7 -
2.4.1. Regulatory Policies.....	- 7 -
2.4.2. Hardware.....	- 7 -
2.4.3. Software	- 7 -
2.4.4. Interfaces to other applications	- 8 -
2.5. Assumptions and Dependencies	- 8 -
3. Specific Requirements.....	- 9 -
3.1. Software Interface Requirements	- 9 -
3.1.1. User Interfaces.....	- 9 -
3.2. Functional Requirements	- 17 -
3.3. Scenarios	- 17 -
3.4. UML Models.....	- 19 -
3.4.1. Use Case Diagram	- 19 -
3.4.2. Class Diagram	- 29 -
3.5. Software System Attributes.....	- 29 -
3.5.1. Reliability	- 29 -
3.5.2. Availability	- 29 -
3.5.3. Security	- 29 -
3.5.4. Maintainability	- 30 -
3.5.5. Usability.....	- 30 -
4. Alloy	- 30 -
4.1. Alloy Code	- 30 -
4.2. Results.....	- 33 -
4.3. MetaModel	- 34 -
5. Software used.....	- 34 -



1. Introduction

1.1. Purpose

This document is the Requirement Analysis and Specification Document (RASD) for the My Taxi Service project. This document has the purpose to present all the requirements and specifications for the design and development of the software, identifying the user groups and its different system requirements to build the best system for each of them.

1.2. Scope

The goal of this project is to optimize the taxi service in a large city through a web application and a mobile app. These applications allow the passengers to request a taxi in an easy way and the taxi drivers to have a fair management of taxi queues.

For the purpose of this application, the city is divided into taxi zones approximately 2 km², each is associated to a queue of taxis to complete all the requests of each zone. The application informs the passenger about the code of the incoming taxi and the waiting time, meanwhile the taxi drivers can inform their actual availability and confirm if they are going to take care of a certain call.

The specific goals of this application are:

- G1** Allow the possible clients to register.
- G2** Allow the taxi drivers to register.
- G3** Allow the users to log in.
- G4** Allow the taxi driver to change his availability.
- G5** Allow the possible passenger to request a taxi.
- G6** Allow the taxi driver to accept or decline a service request.
- G7** Send confirmation code and waiting time to the passenger.
- G8** Allow the users to modify their personal information.
- G9** Allow easy integration with future services.



1.3. Definitions, acronyms, abbreviations Reference documents

1.3.1. Definitions

Application: A computer program designed for a specific task or use.

Queue: In programming:

- a. A sequence of stored data or programs awaiting processing.
- b. A data structure from which the first item that can be retrieved is the one stored earliest.

Request: The act or an instance of asking for something.

Requirement: something that is needed or that must be done. Something that is necessary for something else to happen or be done.

System: A group of interacting, interrelated, or interdependent elements forming a complex whole, especially:

- a. A network of structures and channels, as for communication, travel, or distribution.
- b. A network of related computer software, hardware, and data transmission devices.

User: a person or thing that uses something.

1.3.2. Acronyms

GPS: Global Positioning System.

UML: Unified Modelling Language

IEEE: Institute of Electrical and Electronics Engineers

1.3.3. Abbreviations

App: Application.

1.3.4. Reference documents

IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.

Specification document: Software Engineering 2 Project, AA 2015-2016 Assignments 1-2.

IEEE Std 1233, guide for developing system requirements specifications.



1.4. Overview

The document is organized in:

- **Section 1:** brief description of the idea, goals and objective of the project.
- **Section 2:** Overall description of the application, understanding of the what is going to have the service, the possible actions that can be done by the users and what is suppose to happen to the service work his best.
- **Section 3:** Specific Requirements for the the user interface and functionalities for the users, in this part is listed all functional requirements, possible scenarios and use cases. UML diagrams are presented in this section to allow a better understanding of the functionality of the system.
- **Section 4:** Alloy Modeling, which contains the alloy model which allows to verify the consistency of the proposed solution model.

2. Overall Description

2.1. Product perspective

The product will consist in two main interfaces, one for the drivers and one for the passengers, for the passengers the product will be a mobile app and web application and for the drivers as well but different from the passengers, functionalities will be designed for each type of user, the product only will need an Internet connection and GPS to work. The application for the drivers will interact with the passengers' application to the system works.

2.2. Product functions

As said before the product will have functions for the drivers and for the users and there are:

Common functions:

- Register.
- Log in.
- Modify account/ Delete account.

Drivers functions:

- Allows to change status.
- Allows to accept or decline a call.
- Cancel service.
-

Passengers functions:

- Allows to request a Taxi.
- Cancel a request in the first 2 minutes.
- Allows the passenger to see the status of his request.



2.3. User characteristics

The future users of the applications are the people from the city with access to internet, basic computer skills and don't need technical expertise. The taxi driver first of all will have to fulfill all the requirements to be a taxi driver in the city also will be necessary go to some training about the application to achieve a good effectiveness with the service.

2.4. Constraints

2.4.1. Regulatory Policies

The application will be under the regulatory policies about the privacy of the passengers which will require the customers to accept during the registration the personal data treatment document.

The taxi drivers when they go to the registration office have to sign a paper module that has an apart about the veracity of the documents the driver submit as a professional taxi driver by the city and another about the personal data management.

The system must ask permission to acquire web cookies.

2.4.2. Hardware

The devices must have internet connection to run properly the application. The customers, passengers and taxi drivers, must have the access to Internet in order to use the application, and also for the use of the GPS option, they must have GPS in the device, but it is not mandatory.

2.4.3. Software

My Taxi Service must support parallel operations from different type of users on different platforms.

For the user to use the service on the web application, the System shall be Windows 7 or greater, Mac OS X 10.8 Mountain Lion or greater or GNU/Linux Ubuntu. The System must be able to run a web browser to run the application on the web as Firefox, Microsoft Edge, Google Chrome, Opera or Safari.

For the user to use the service on the mobile app, the System shall be Android 4.0 or greater or IOS 7 or greater.

The database shall be stored on a Windows or Linux server using Apache and MySQL.



The web application and the mobile application must be developed having in mind the capacity of the database.

2.4.4. Interfaces to other applications

For the moment there is not an actual interaction with other external applications, but the interface structure should be developed to enable further implementation of services and other applications.

2.5. Assumptions and Dependencies

- A user only has one account.
- The user can easily download the app.
- The users must be logged in the app in order to interact with it.
- A user can only request one taxi at a time.
- The system will send a message to the passenger immediately the driver accepts the request.
- Taxi driver's status:
 - Available, willing to receive a request.
 - Unavailable, occupied in a request or can not attend one.
- The taxi driver status must be available for being considered in the queue.
- If there is not a response from the taxi driver within 1 minute, it will assume refusal of the service.
- If at the meeting point the passenger doesn't show within 5 minutes, the driver can change his status to available and the service is assumed finished.
- If in the zone of the passenger there is not available taxi the system will search the near taxi available to complete the request.
- If there is not a taxi near enough to complete a request, the system will send a message to the passenger telling him to try again later.
- The system will check from time to time updates for the app automatically.



3. Specific Requirements

3.1. Software Interface Requirements

The interfaces have to be simple and intuitive. The users should be able to easily understand the basic functions without previous knowledge of the system. The web application must be accessible from the most used browser. The mobile app must be able to run on IOS, Android, Windows Phone.

3.1.1. User Interfaces

At the following section it will see the different interfaces of the Web application and Mobile Application for the users, the drivers and the passenger, it will give a better understanding of the system for the final users.

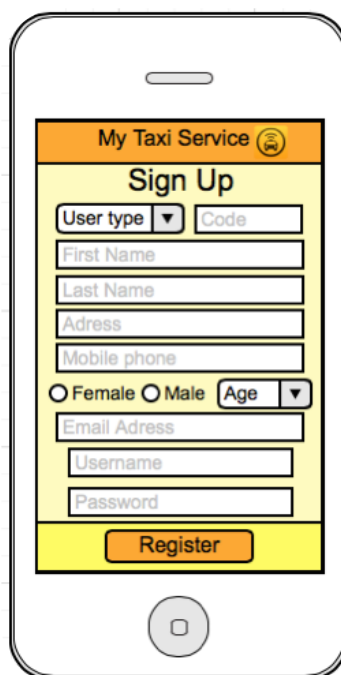
3.1.1.1. Sign up users Interfaces.

The Figure 1 and Figure 2 shows the interfaces for the passengers and the driver to sign up in the different ways that has the system.



The web application sign up interface is displayed in a browser window with the URL `www.MyTaxiService.it/sign_Up`. The header is orange and contains the text "My Taxi Service" and a taxi icon. The main content area has a yellow background. On the left, there is a large "Sign Up" text and a circular icon with a taxi. On the right, there is a registration form with the following fields: "User type" (dropdown), "Code" (text), "First Name" (text), "Last Name" (text), "Email Address" (text), "Mobile phone" (text), "Gender" (radio buttons for "Male" and "Female"), "Age" (dropdown), "Address" (text), "Username" (text), and "Password" (text). A "Register" button is at the bottom.

Figure 1. Sign up Web application



The mobile application sign up interface is shown on a smartphone screen. The header is orange and contains the text "My Taxi Service" and a taxi icon. The main content area has a yellow background. The form fields are arranged vertically: "User type" (dropdown), "Code" (text), "First Name" (text), "Last Name" (text), "Address" (text), "Mobile phone" (text), "Gender" (radio buttons for "Female" and "Male"), "Age" (dropdown), "Email Address" (text), "Username" (text), and "Password" (text). A "Register" button is at the bottom.

Figure 2. Sign up Mobile Application

3.1.1.2. Log in users' interfaces.

The Figure 3 and Figure 4 shows the log in on the Web application and Mobile application.

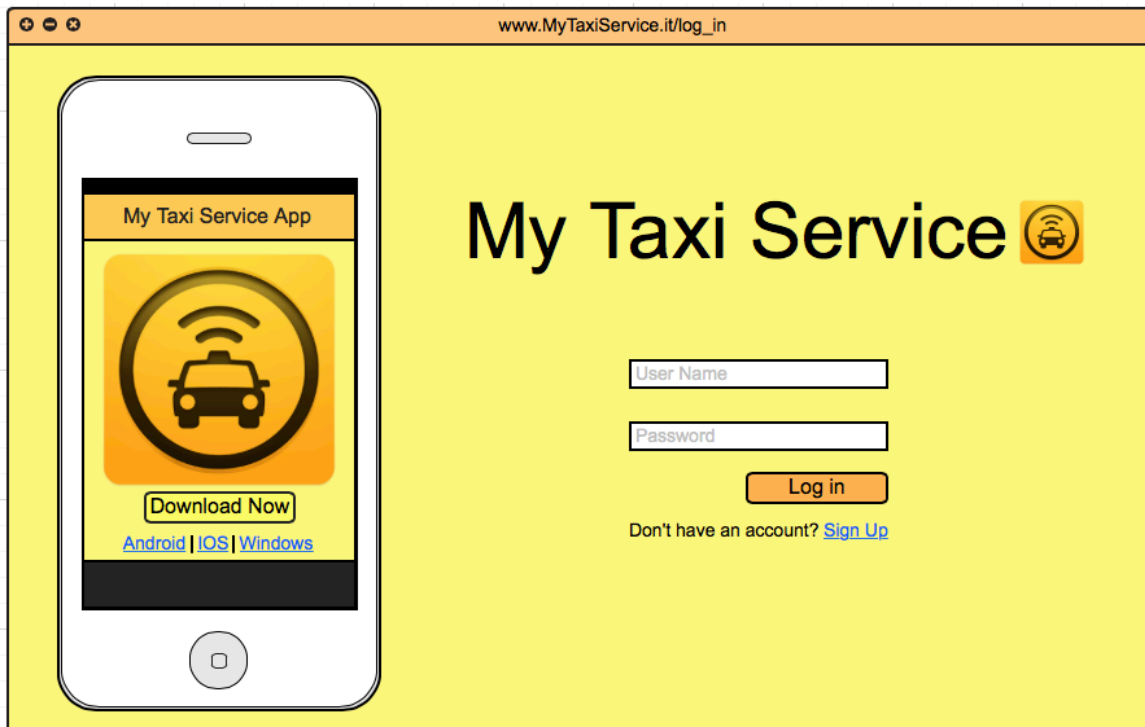


Figure 3. Log in Web Application

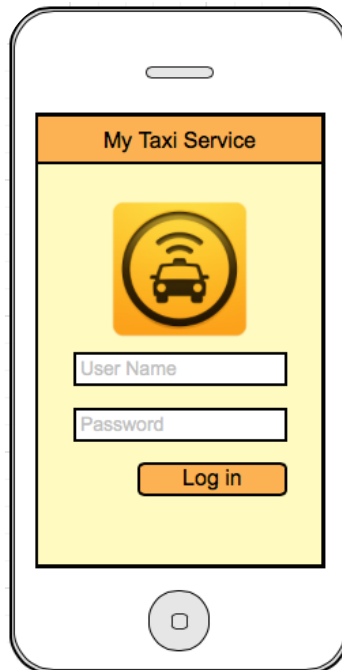


Figure 4. Log in Mobile Application

3.1.1.3. Request a taxi.

The Figure 5 and Figure 6 it sees the interfaces for request the taxi.

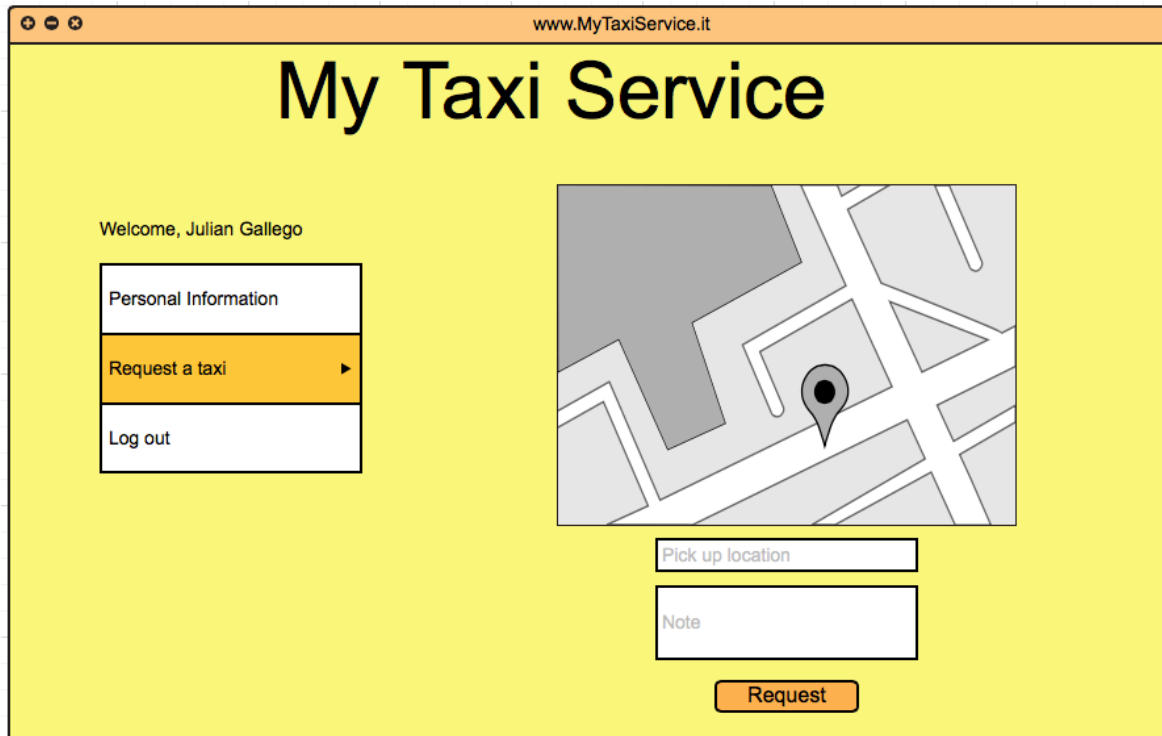


Figure 5. Taxi request web application.

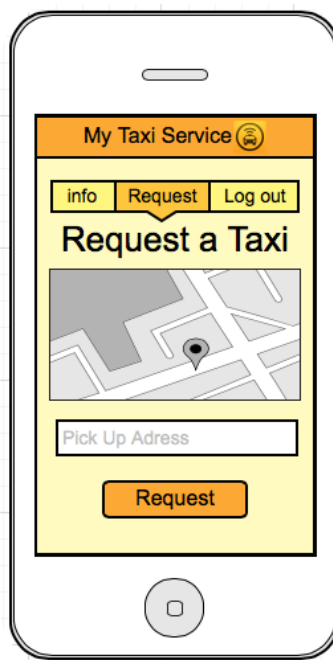


Figure 6. Taxi request mobile application.

3.1.1.4. Interface After Request was accepted

After the request is made and a taxi driver accepted to do the ride, is displayed the waiting time and ride code.

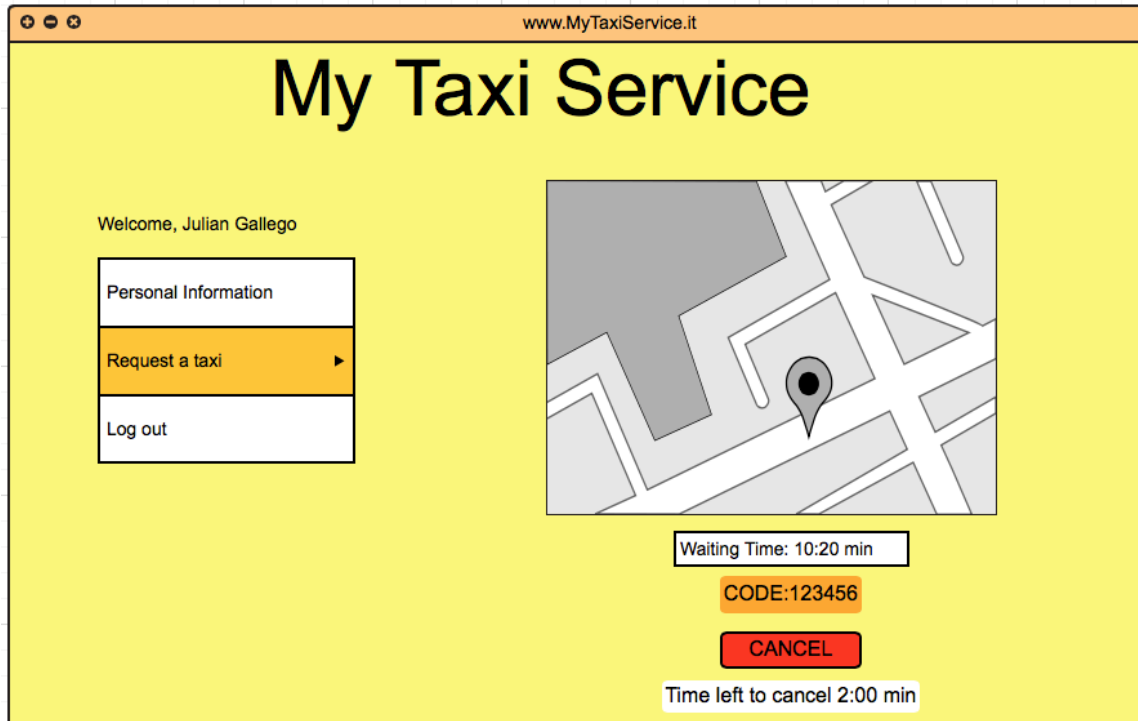


Figure 7. After taxi request mobile application.

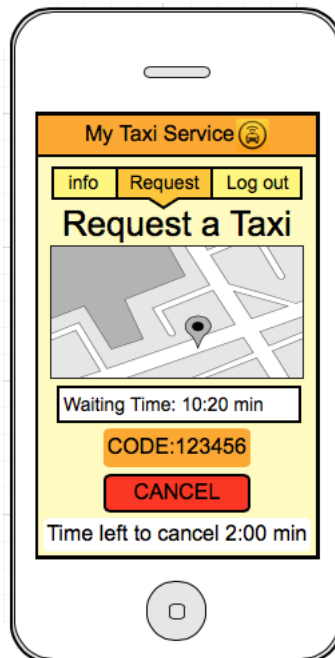


Figure 8. After Taxi request mobile application.

3.1.1.5. Waiting Request interface

When the driver Access to his account and select search passenger this is the way they can see it if he accesses from the web browser.

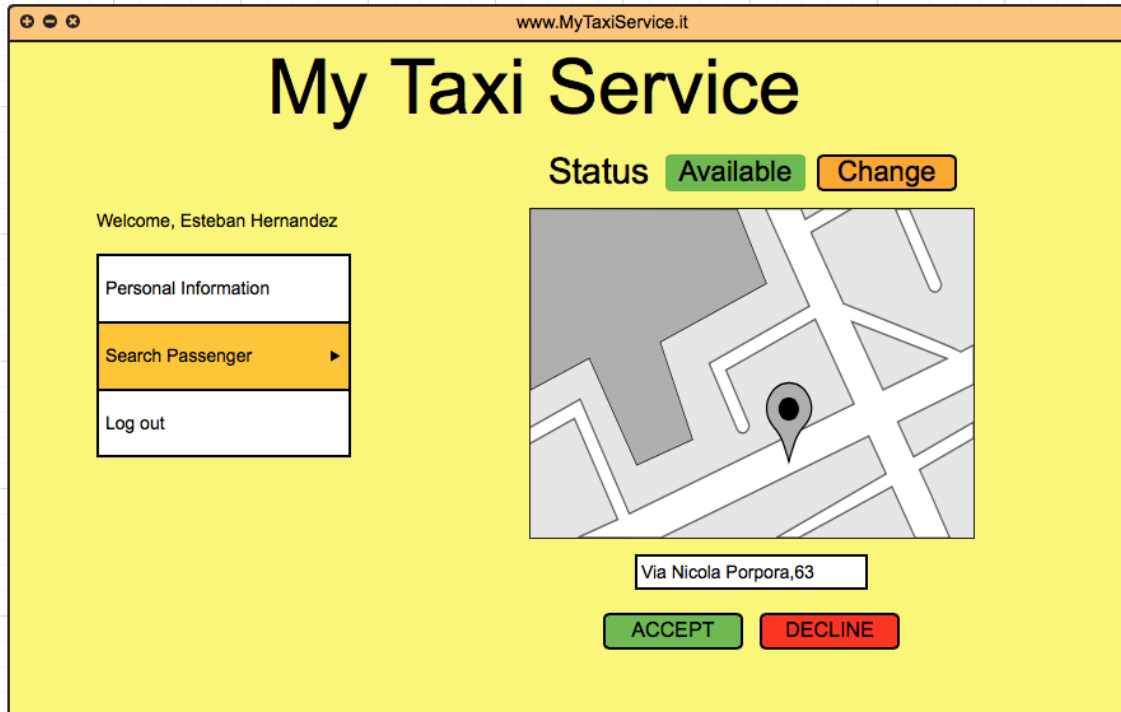


Figure 9. Passenger search web application.

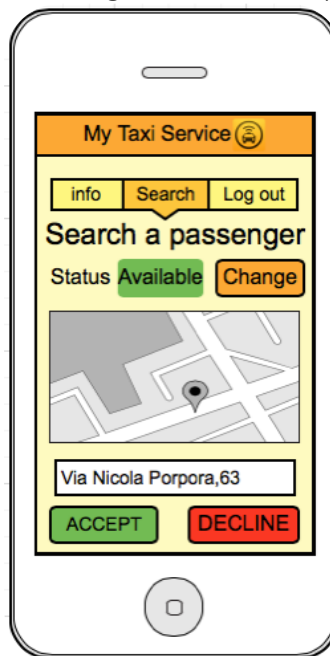


Figure 10. Passenger search mobile application.

3.1.1.6. Interface After Request accepted

When the driver accepts a ride the following interface is displayed till he confirms the arrival, validating the code of the passenger with the one he has. In the interface appears the option cancel, in case something happened to the driver or car and can't provide the service.

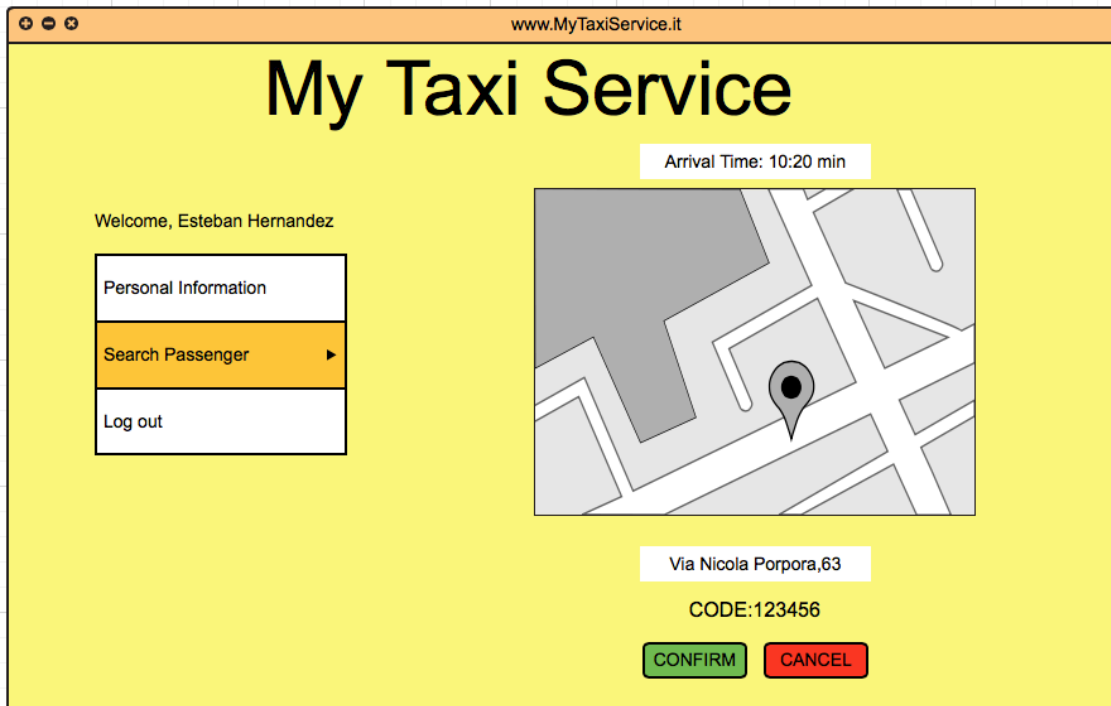


Figure 11. After Request accepted web application.

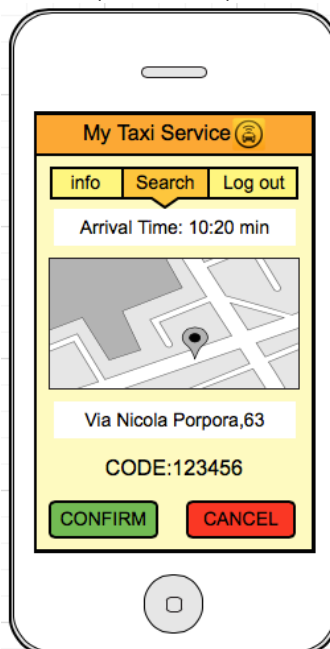


Figure 12. After Request accepted mobile application.

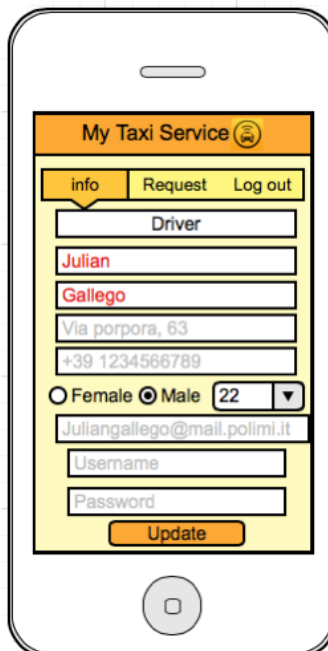
3.1.1.7. Manage information

The following is the interface of the driver to manage the personal information, the one for the passenger is almost the same with 2 differences one is that he can change the name and last name, the second one is a change on the menu of Search passenger for Request a taxi.



The screenshot shows a web browser window with the URL `www.MyTaxiService.it/sign_Up`. The page has an orange header with the text "My Taxi Service" and a car icon. On the right, it says "Welcome, Julian Gallego" with a "Log out" link. The main content area is yellow and titled "Personal Information". On the left, there is a sidebar menu with three items: "Personal Information" (highlighted with an orange background and a right-pointing arrow), "Search Passenger", and "Log out". The main form contains several input fields: "Driver" (a label), "Julian" (first name), "Gallego" (last name), "Via porpora, 63" (address), "Juliandavid.gallego@mail.polimi.it" (email), "+39 1234566789" (phone number), gender selection with "Male" selected, an age dropdown set to "22", "Username", and "Password". An "Update" button is at the bottom. A large circular icon with a car and signal waves is on the right.

Figure 13. Manage information web application.



The screenshot shows a mobile application interface on a smartphone. The header is orange with "My Taxi Service" and a car icon. Below the header is a navigation bar with three tabs: "info" (selected), "Request", and "Log out". The main form is yellow and contains the same fields as the web version: "Driver", "Julian", "Gallego", "Via porpora, 63", "Juliandavid.gallego@mail.polimi.it", "+39 1234566789", gender selection with "Male" selected, an age dropdown set to "22", "Username", and "Password". An "Update" button is at the bottom. The interface is designed for a small screen with a single column of elements.

Figure 14. Manage information mobile application.



3.2. Functional Requirements

The functional requirements are divided by the class of user that going to interact with the system, this way:

- **Non user:**
 - Sign up.
- **Passenger:**
 - Log in.
 - Manage his personal information.
 - Call a taxi.
 - Cancel service.
- **Taxi driver:**
 - Log in.
 - Manage his personal information.
 - Change his status (free or busy for a ride).
 - Cancel service.

3.3. Scenarios

- **Scenario 1:** Esteban just arrived to Milan and there is no one in the arrival who can get him to his hotel. He remembers the My Taxi Service app from the last time he came to the city and does the log in using his account he created the last time. He accesses to request a taxi and allows the app to access to his position by the GPS of the phone, then he confirms the request and wait to someone to pick him up. In few minutes Esteban is in the taxi going to his hotel to rest in his room.
- **Scenario 2:** Sebastian is a taxi driver and he received a call for a ride, he looked the position of the possible passenger and accepted the call, then his status changed to unavailable and now he can see in his phone the best way to go to the meeting point, the time left and the ride code. When he arrived to the point, he confirms the code of the ride with the passenger, when he finished the ride, he changed the status into available and waited for the next call.
- **Scenario 3:** Felipe is a taxi driver and he must be at the Duomo to pick up Camilo. During the ride to the Duomo the car had a problem and stopped working. Felipe immediately tried to repair it but there wasn't anything he can do. He sent a notification canceling the ride, the system automatically searched another driver to Camilo.
- **Scenario 4:** Gaia saw an advertisement talking about My Taxi Service App so she decided to download it and used it because she was in a lonely place and needed to go fast home. After she download the app, she registered himself into the application and in few minutes she was taking the taxi to go home.



- **Scenario 5:** Alex is a driver of a taxi and use My Taxi Service app, he just ended his work shift, so before going home he selected the "Change" function from the Search passenger page to set his status to unavailable. The system removed Alex from the queue he was in. The next day, Alex opened the app and used the "Change" option again, the system changed his status from Unavailable to Available, meanwhile the system traced Alex's position via GPS, found the zone where he was, allocated him at the bottom of the queue associated to his zone, and a few minutes later, Alex was doing his first ride of the day.
- **Scenario 6:** Luis was going to an important meeting and was waiting the train that can take him there. The trains were delayed that day, so Luis had to requested a taxi outside the Station because he wasn't going to make it by train, but he didn't leave the gate trying to catch the train. 1:30 minute later the train arrived to the station. Luis is forced to cancel his request, so he accessed the app with his phone and cancel it. The system checked if the cancellation has been done in time (no more then 2' from the acceptance of the request), and confirmed it. The system also notified the driver and put him in the 1 place of the queue and searched another request in the zone for him.

3.4. UML Models

3.4.1. Use Case Diagram

Now its present the Use Case Diagram for the whole application derived from the functionalities of the system. In the next sections its explain more details about the Use Cases and the flow of events associated to them.

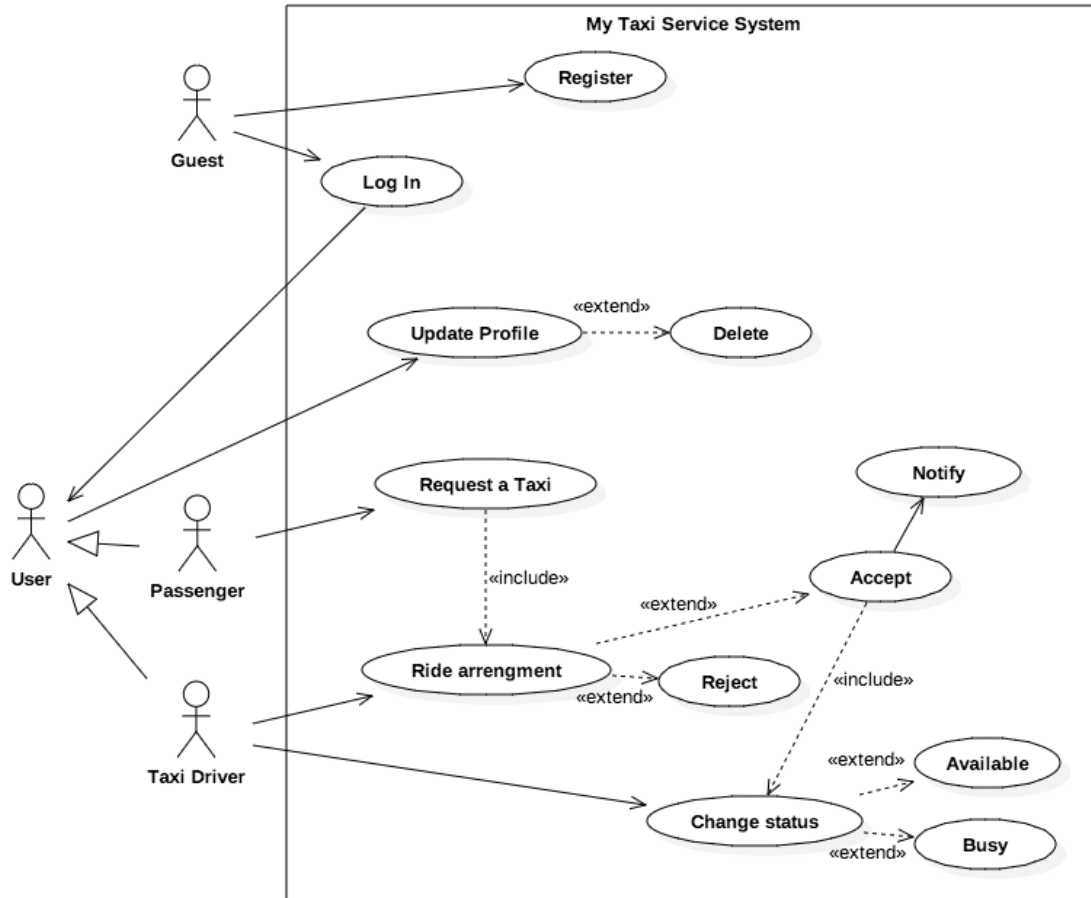


Figure 15. Overview Use Cases

3.4.1.1. Users registration

Name	Sign Up passenger
Actors	Passenger(visitor)
Goal	G1
Entry Condition	The use case starts when the visitor access in the home page of the site to the sign up option.
Event Flow	1. The process starts when the visitor clicks on the button Sign up.



	<p>2. The side brings the form for user registration.</p> <p>3. The visitor fills in all mandatory fields.</p> <p>4. The system will verify the input data. If everything is fine, the system will send a validation email.</p> <p>5. Now the visitor access to the link sent in the email that activate the account.</p> <p>6. The system creates the new user and displays a success message on the page.</p>
Output Condition	When the system creates to the visitor an account the use case terminated.
Exception	<p>-Visitor is already a user.</p> <p>-Visitor chooses a username already possessed by an other user.</p> <p>-Visitor inserts an e-mail that is already associated with another user.</p> <p>-The databases have a problem.</p> <p>In all this cases the system displays an error message and the visitor is redirected to the registration form.</p>

On the driver registration for security reasons the taxi drivers have to go to an authorized office to obtain a security code, the code will be delivered after the person in charge verify all the information about the driver availability to provide the service in the city, the code will give access to register as a taxi driver for our system.

Name	Driver Registration
Actors	Taxi Driver(visitor)
Goal	G2
Entry Condition	The use case starts when the visitor access in the home page of the site to the sign up option.
Event Flow	<p>1. The process starts when the visitor clicks on the button Sign up.</p> <p>2. The side brings the form for user registration.</p> <p>3. The visitor fills in all mandatory fields using the code that was given to the them in the office.</p>

	<p>4. The system will verify the input data. If everything is fine and the code doesn't have used before, the system will send a validation email.</p> <p>5. Now the visitor access to the link sent in the email that activate the account.</p> <p>6. The system creates the new user and displays a success message on the page.</p>
Output Condition	When the driver access to his main page the use case terminated.
Exception	<p>-Visitor is already a user.</p> <p>-Visitor chooses a username already possessed by an other user.</p> <p>-Visitor inserts an e-mail that is already associated with another user.</p> <p>-The databases have a problem.</p> <p>In all this cases the system displays an error message and the visitor is redirected to the registration form.</p>

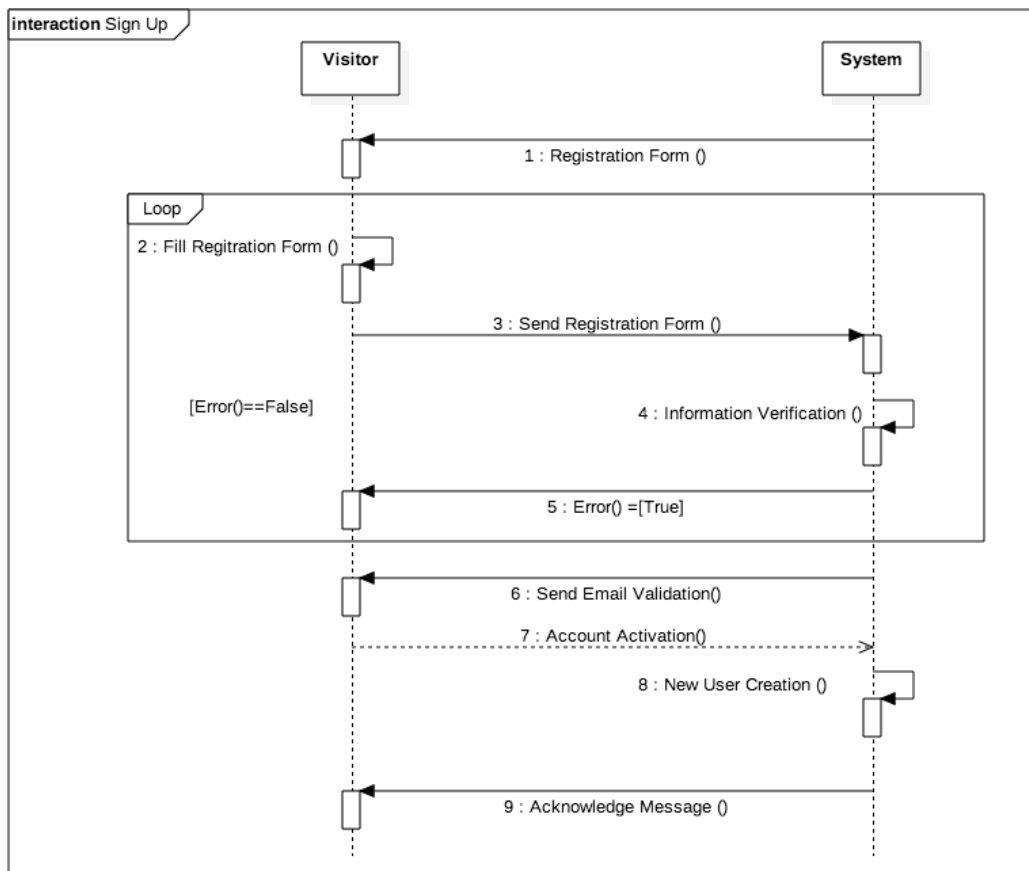


Figure 16. Sign up sequence diagram.

3.4.1.2. Log in

Name	Log in
Actors	Passenger and Taxi Driver
Goal	G3
Entry Condition	The use case starts when the passenger or taxi driver access the main page of the site.
Event Flow	<ol style="list-style-type: none"> 1. The process starts when the visitor fills the mandatory fields (username and password) in the home page. 2. The system will verify the input data. If everything is fine, the system will display the main page for the user (passenger or driver). 3. Now the user has access to his services.
Output Condition	When the passenger access to his main page the use case terminated.
Exception	<p>-Visitor don't remember his password or username. -Visitor (driver) inserts a username that is already being used in other device.</p> <p>In all this cases the system displays an error message and the visitor is redirected to the home page</p>

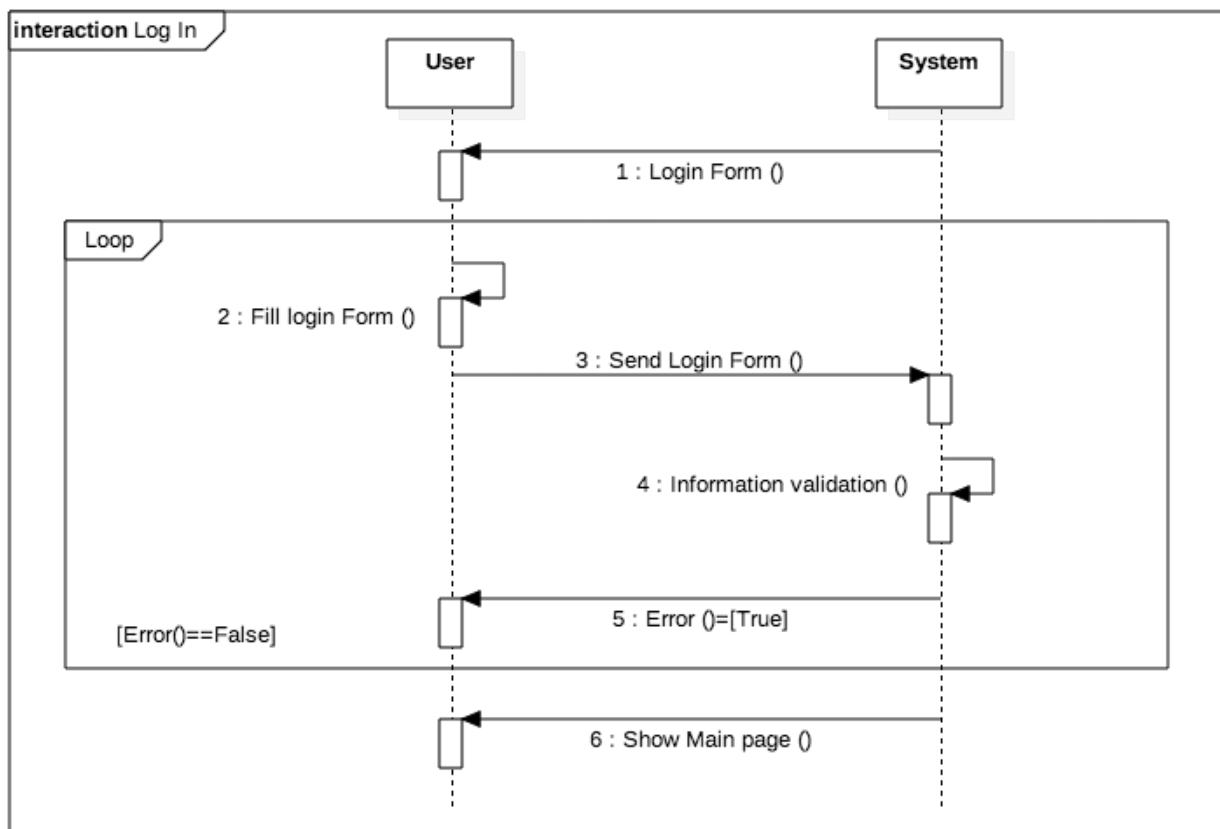


Figure 17. Login sequence diagram.



3.4.1.3. Manage personal information

Name	Manage personal information
Actors	Passenger and Taxi Driver(user)
Goal	G8
Entry Condition	The use case starts when the passenger access in the menu, personal information.
Event Flow	<ol style="list-style-type: none">1. The user on the home page selects on the menu the personal information option, then the process starts.2. User change the fields he wants to change.3. User clicks on save button to upload the changes to the system.4. The system will ask the user to enter his password to confirm the identity of the user.5. The application will verify the input data. If everything is fine, the system will change the data.6. The system will show an acknowledge message. <p>Note: The driver can't change information related to his name and last name, the other fields are available to be changed.</p>
Output Condition	When the user see again the information page with the new information the use case terminated.
Exception	-User don't remember his password.

In the following page is showing the sequence diagram for this use case.

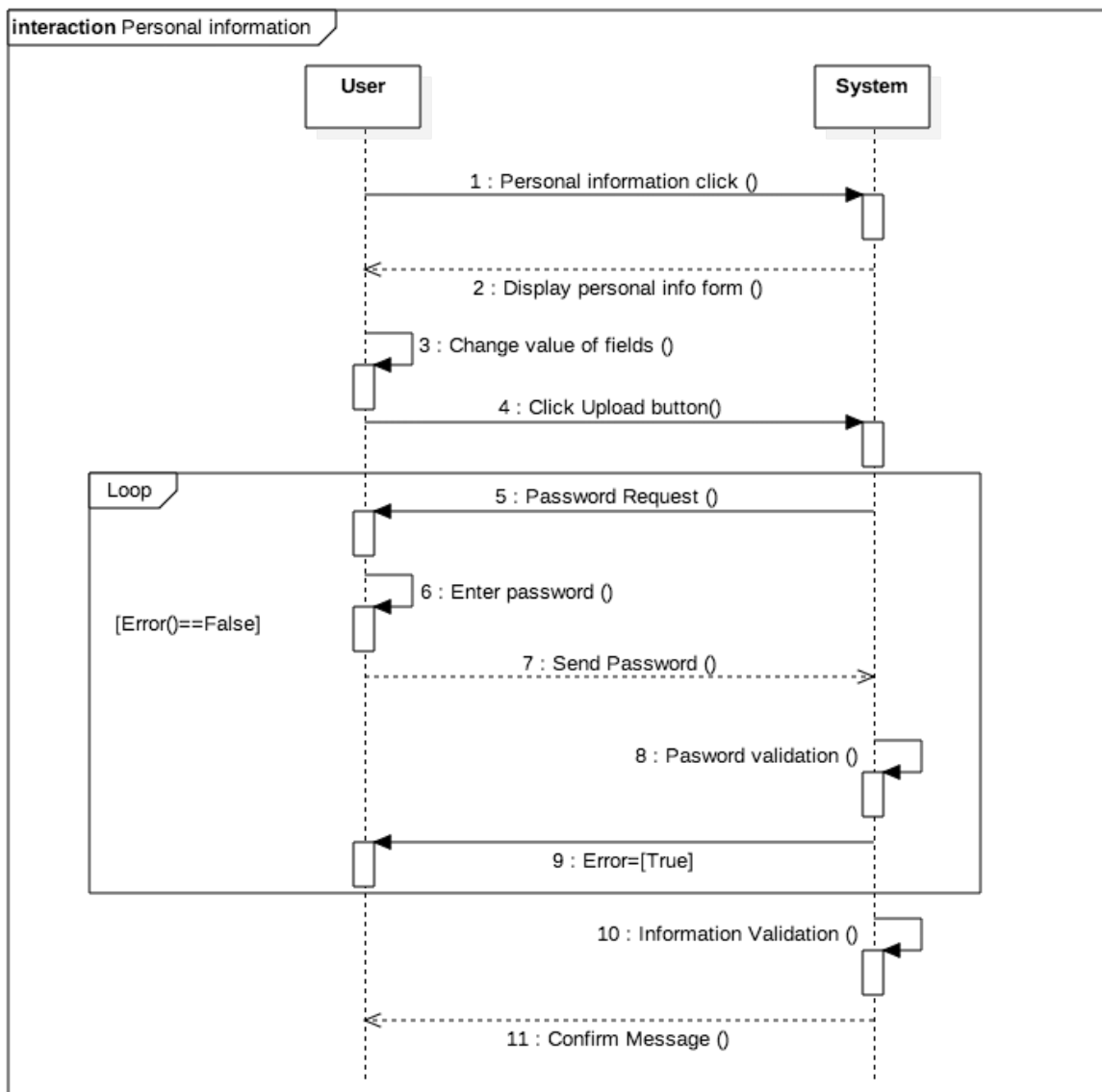


Figure 18. Manage information sequence diagram.

3.4.1.4. Change driver status

Name	Change Status
Actors	Taxi Driver
Goal	G4
Entry Condition	The use case starts when he press change status on the search request page.
Event Flow	<p>1. The driver on the search request page, clicks on the button change availability, then the process starts.</p> <p>2. The system will toggle the present state of the driver. Depending</p>

	on the new status of the driver, he will be added or removed from the queue.
	3. The system will update the change in the page.
Output Condition	When the status in the page is changed the use case terminated.
Exception	-The driver is taking a passenger to a destination.

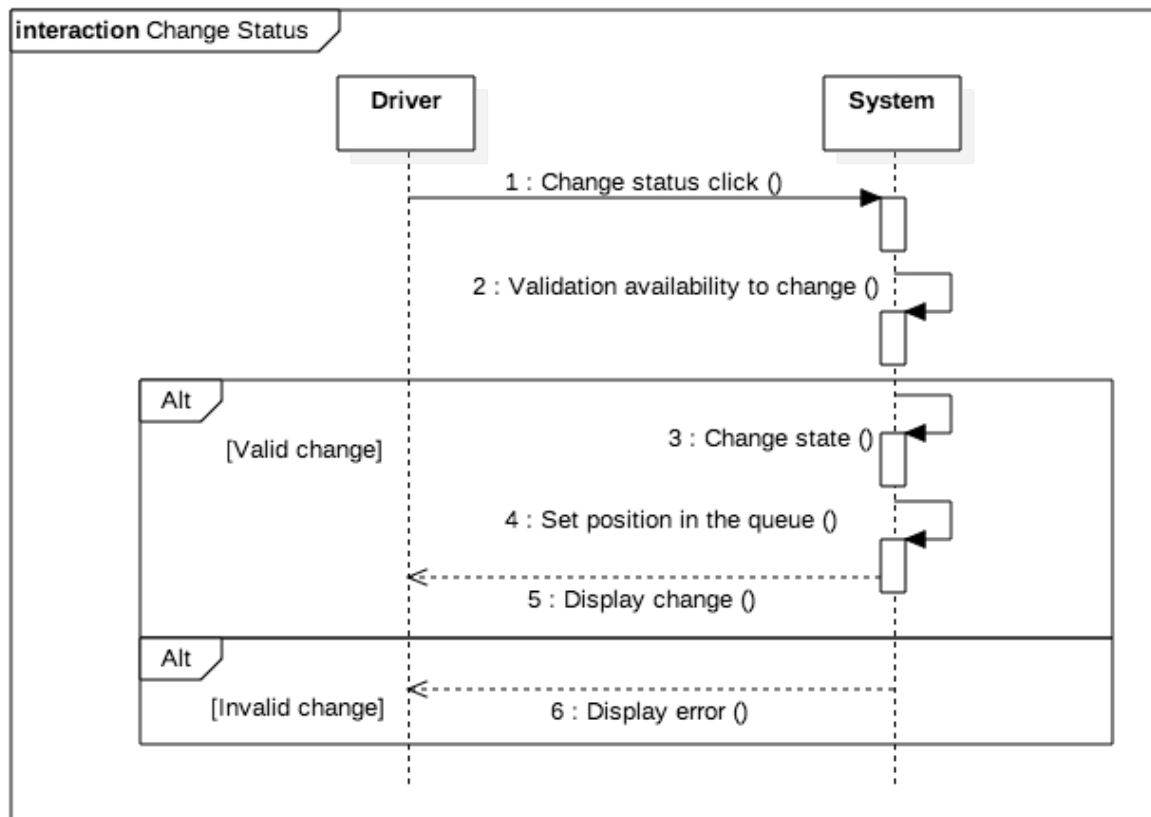


Figure 19. Change status sequence diagram.

3.4.1.5. Request a Taxi

Name	Request a Taxi
Actors	Passenger and Taxi Driver
Goal	G5,G6,G7
Entry Condition	The use case starts when the passenger request a taxi
Event Flow	<p>1. The passenger on the main page selects on the menu the request a Taxi option, then the process starts.</p> <p>2. The passenger fills in all mandatory fields.</p> <p>3. The passenger clicks on request button to confirm the request.</p>

	<p>4. The application will verify the input data. If everything is fine, the system will search for an available taxi nearby in the zone.</p> <p>5. When an available taxi driver takes care of the request, a code and waiting time will be confirmed to the passenger.</p>
Output Condition	When the passenger aboard the taxi the use case terminated.
Exception	-The passenger already has asked for a taxi.

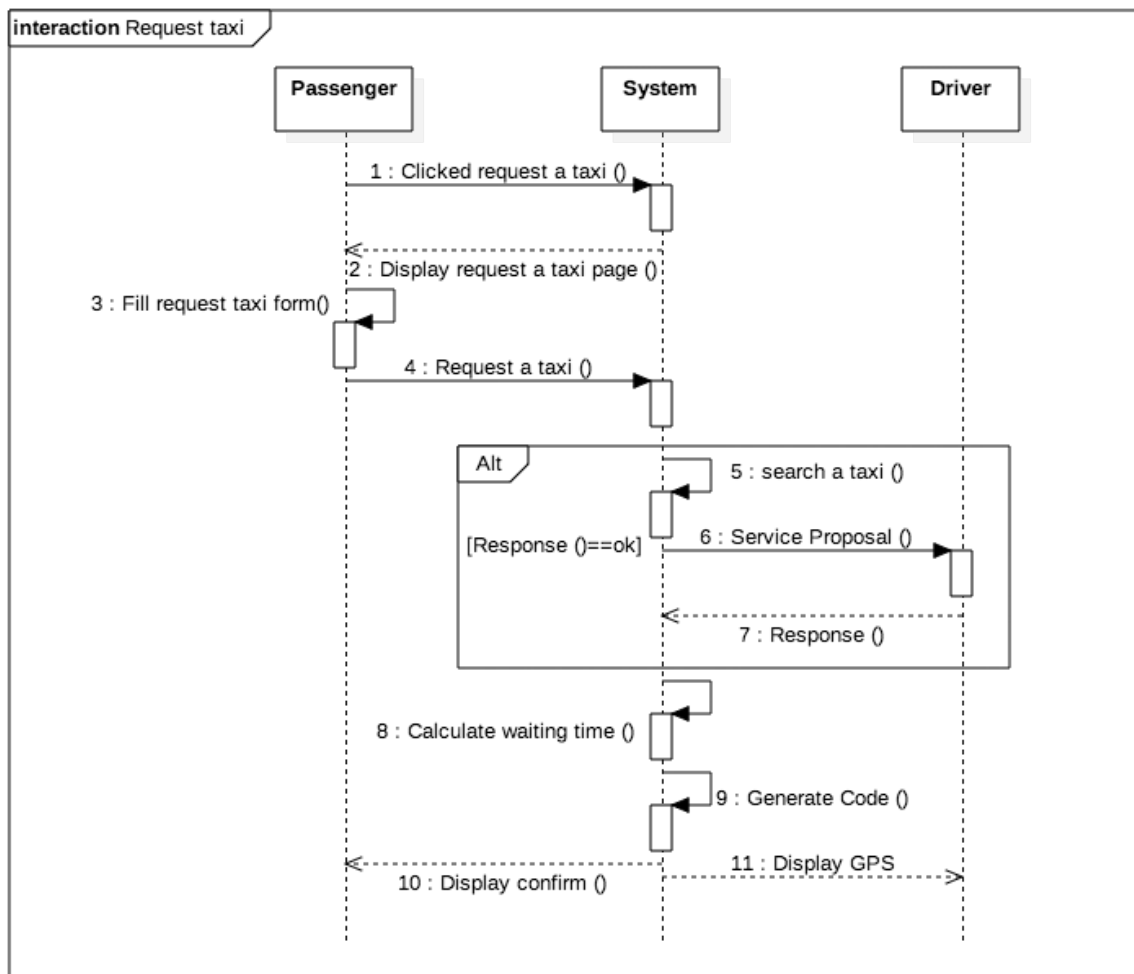


Figure 20. Request a taxi sequence Diagram.

3.4.1.6. Searching passenger

Name	Search a passenger
Actors	Passenger and Taxi Driver
Goal	G5,G6,G7

Entry Condition	The use case starts when the driver change his availability to Available.
Event Flow	<ol style="list-style-type: none"> 1. The driver on the home page selects on the menu the search request option, then the process starts. 2. The system will update the driver's status and then add it to the final of the queue. 3. Then when the drivers get the first place on the queue, the system will search for a passenger in his zone. 4. When a passenger is found, it will be displayed in the driver's screen the position and time to the place. 5. The driver then has to choose to accept it or decline it. 6. If the driver accepts the request the system will display a code in the interface. If he declines it the driver will be send to the finish of the queue.
Output Condition	When the driver accepts or declines a request the use case terminated.
Exception	-There is not passengers asking for a ride.

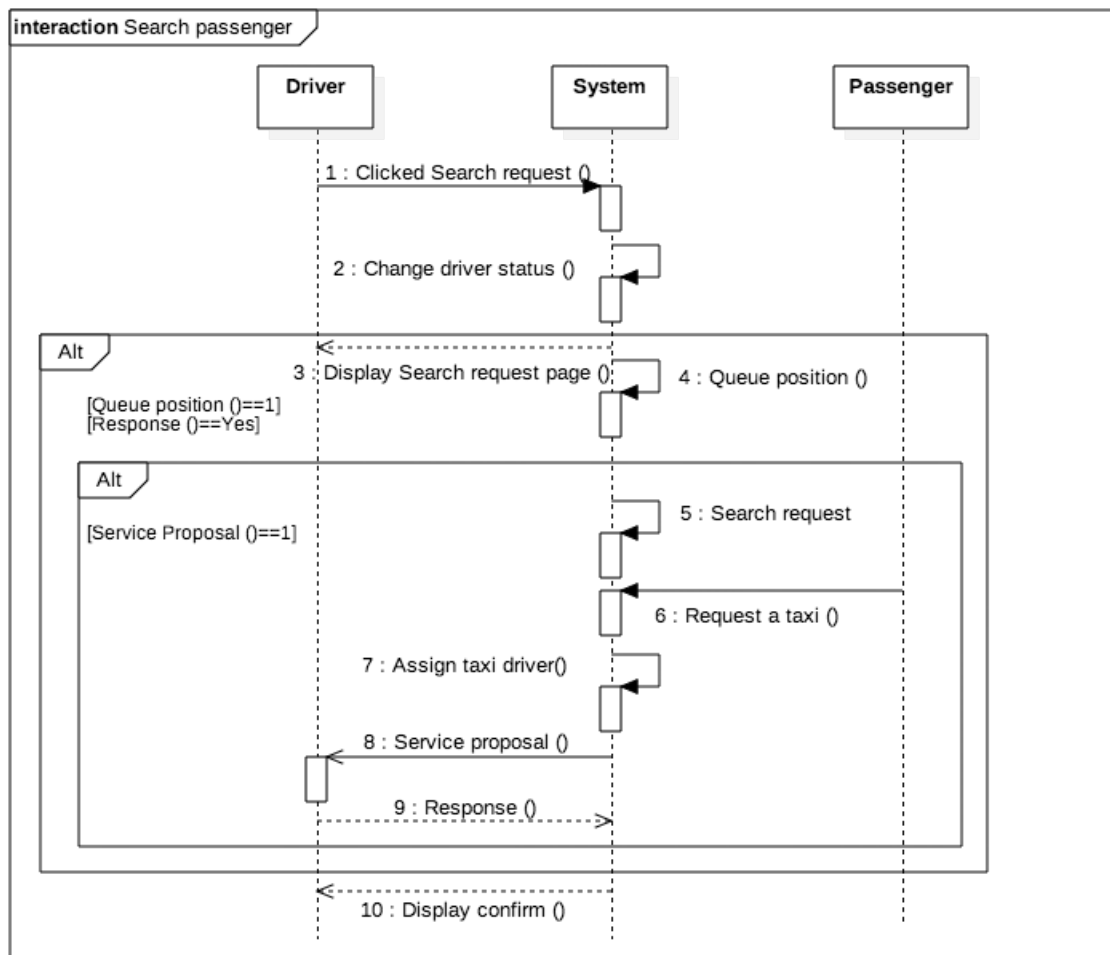


Figure 21. Search a passenger sequence diagram.

3.4.1.7. Notification about ride

Name	Notification about ride
Actors	Passenger and Taxi Driver
Goal	G7
Entry Condition	The use case starts when the the taxi driver accepts the request.
Event Flow	<p>1. The system will display the code and the waiting time in the request page for the passenger.</p> <p>2. When the taxi arrives and confirm the arrival, checking the code from the passenger, the system will remove the waiting time and code from the passenger interface.</p>
Output Condition	When the system remove the waiting time and code the use case terminated.
Exception	-The GPS of the driver suddenly gets off

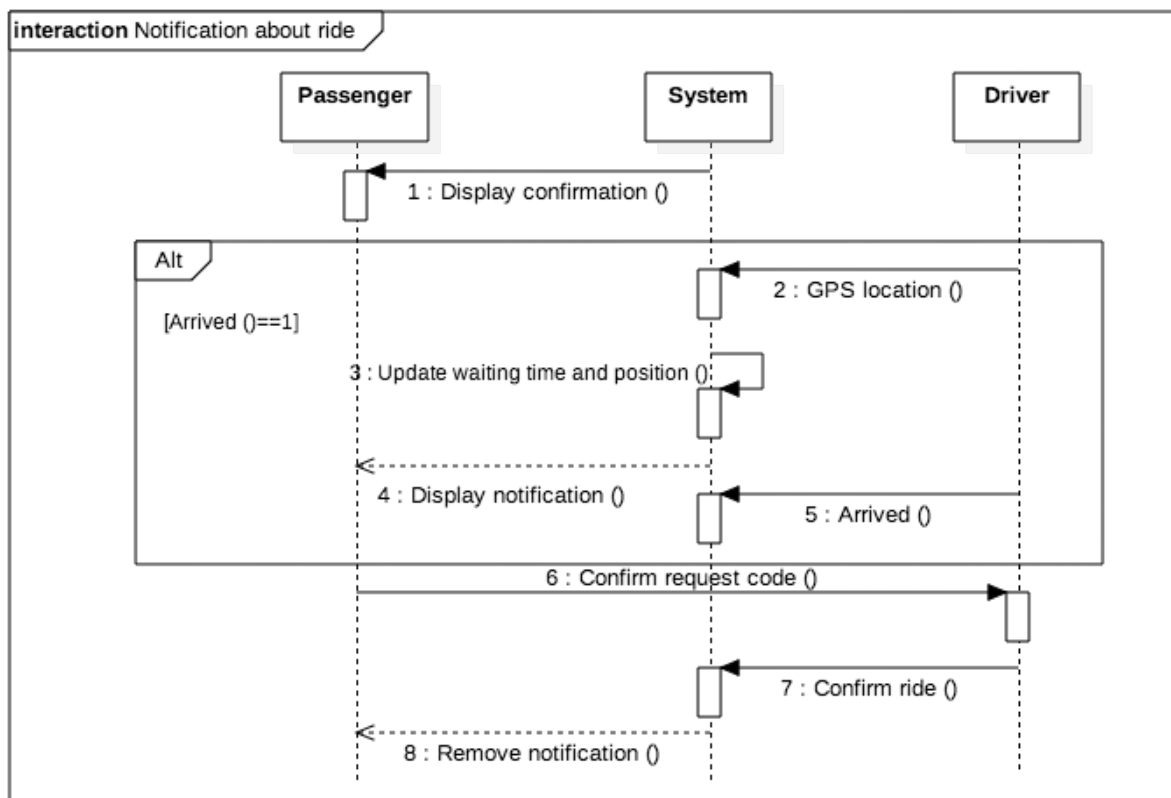


Figure 22. Notification about the ride sequence diagram.

3.4.2. Class Diagram

On the following figure can be appreciated the Class diagram of the system.

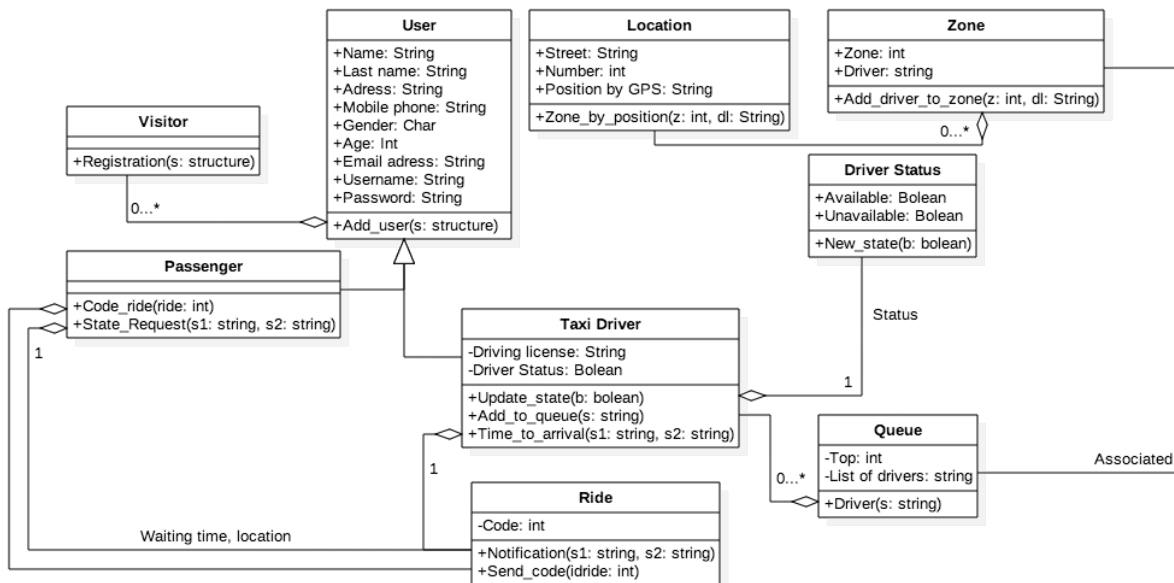


Figure 23. Class diagram of the system.

3.5. Software System Attributes

3.5.1. Reliability

The system will be developed with redundancy, periodical backups, and all the functions must pass several test to be implemented in the program, to avoid algorithm problems as possible.

3.5.2. Availability

My taxi service will be accessible anytime. All the service will be hosted in a cloud platform primary, but is also will have dedicated servers that will act as a backup plan in case of the cloud platform goes down, they will download the information often from the cloud and can start working at any time they would be need it, the system when starts send a message to the users and restart the services as soon as possible.

3.5.3. Security

The app is password-protected, there is a method to authenticate the user identity and specially the driver's identity. The system will guarantee the safety of the user's data according with the agreement accepted at the registration.



3.5.4. Maintainability

The system will be very maintainable thanks to a modular structure. The application will be documented in a way that will truly help the future developers, explaining how the application works and how it has been developed, in addition, the code will be periodically examined and improved, with a proper documentation.

3.5.5. Usability

The application is meant to be as easy and intuitive to use as possible. The drivers have capacitation for improve their performance with the app. All the users can provide feedbacks to improve the app.

4. Alloy

4.1. Alloy Code

module MyTaxiService

/ DATA DEFINITIONS*/*

```
abstract sig User {
    name : one String ,
    lastname : one String ,
    address : one String ,
    mobilephone : one String ,
    Gender: one String,
    Age: one String,
    email : one String ,
    username : one String ,
    password : one String,
    userID: one String,
}

sig Visitor extends User{}

sig Passenger extends User {
    Ride_code: one String ,
    position_request: one position,
}
```



```
sig Taxi_Driver extends User {  
    current_position : one position ,  
    driving_license : one String ,  
    status : one Status_Driver,  
    Code_drive: one String,  
    Allow:one String  
}
```

```
enum Status_Driver{Available,Unavailable}
```

```
sig position{  
    latitude: one String,  
    longitue: one String,  
}
```

```
sig city{  
    zones:set zone,  
}
```

```
sig zone{  
    Num: one String,  
    area: one position,  
    queue: one Queue  
}
```

```
sig Queue{  
    queue_Num: one String,  
    Drivers: set Taxi_Driver,  
    Passngers: set Passenger,  
    Next: one Taxi_Driver  
}
```

```
//FACTS
```

```
fact Users{  
    //Unique users for each email,username and user id  
    no disj u1,u2:User | u1.email = u2.email  
    no disj u1,u2:User | u1.username = u2.username  
    no disj u1,u2:User | u1.userID = u2.userID  
}
```

```
fact Drivers{  
    //there is only one driver for each license and Allow code  
    no disj d1,d2:Taxi_Driver | d1.driving_license= d2.driving_license
```



```
no disj d1,d2:Taxi_Driver | d1.Allow = d2.Allow
no disj d1,d2:User | d1.Allow = d2.Allow
}

fact Single_taxi{
    //only one taxi will take the request
    no disj p1,p2:Passenger, r:Taxi_Driver | r.Code_drive=p1.Ride_code and
r.Code_drive=p2.Ride_code and p1.userID=p2.userID
}

fact not_empty{
    //there must be at least a zone
    #zone>1
}

fact Zone_queue{
    //each zone has a different queue
    all z1:zone, z2:zone | z1.queue=z2.queue implies z1=z2
}

fact Taxi_OneQueue{
    //a driver only can be in one Queue at time
    no t:Taxi_Driver, q1,q2:Queue | (t in q1.Drivers) and (t in q2.Drivers) and(q1 != q2)
}

//ASSERTIONS

assert single_users{
    //there is only one user for each id
    no disj u1,u2:User | u1.userID = u2.userID
}
check single_users for 10

assert ONLYQueueOrZone{
    //There is only one queue to one zone,
    no disj z1,z2: zone | z1.queue = z2.queue
}
check ONLYQueueOrZone

assert Single_taxi{
    //only one taxi will take the request
    no disj p1,p2:Passenger, r:Taxi_Driver | r.Code_drive=p1.Ride_code and
r.Code_drive=p2.Ride_code implies (p1.userID=p2.userID)
```




```
}  
check Single_taxi for 10  
  
assert addNewUser{  
    all u,u1,u2: User | (u not in u1) and addNewUser[u, u1,u2] implies (u in u2)  
}  
check addNewUser for 7  
  
//PREDICATES  
  
pred show(){  
    #city=1  
    #zone>5  
    #Queue>6  
    #Taxi_Driver>10  
    #Passenger>10  
}  
run show  
  
pred addNewUser(u,u1,u2:User){  
    u.userID not in u1.userID implies u.userID=u2.userID  
}  
run addNewUser for 10
```

4.2. Results

In the following Figure it is shown the results of the model in alloy of the system.

```
commands were executed. The results are:  
#1: No counterexample found. single_users may be valid.  
#2: No counterexample found. ONLYQueueOrZone may be valid.  
#3: No counterexample found. Single_taxi may be valid.  
#4: No counterexample found. addNewUser may be valid.  
#5: No instance found. show may be inconsistent.  
#6: No instance found. addNewUser may be inconsistent.
```

Figure 24. result of the model in Alloy Analyzer.

