

myTaxiService

November 4, 2015

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Purpose . . . . .	4
1.2	Scope . . . . .	4
1.3	Stakeholders . . . . .	5
1.4	Glossary . . . . .	5
<b>2</b>	<b>Overall description</b>	<b>7</b>
2.1	Product Perspectives . . . . .	7
2.2	Product Functionalities . . . . .	7
2.3	User characteristics . . . . .	8
2.4	Constraints and General Assumptions . . . . .	8
2.4.1	Regulatory policies . . . . .	8
2.4.2	Hardware limitations . . . . .	8
2.4.3	Software limitations . . . . .	8
2.4.4	Interfaces to other applications . . . . .	9
2.4.5	Parallel operations . . . . .	9
2.5	Domain Assumptions . . . . .	9
2.5.1	User identification . . . . .	9
2.5.2	Taxi Driver identification . . . . .	9
2.5.3	Taxi zones . . . . .	9
2.5.4	Taxi availability . . . . .	9
2.5.5	Reservation . . . . .	10
2.5.6	Sharing option . . . . .	10
2.5.7	Sharing decline . . . . .	10
2.5.8	GPS correctness . . . . .	10
2.5.9	Starting point correctness . . . . .	10

<b>3</b>	<b>Specific Requirements and Scenarios</b>	<b>11</b>
3.1	Functional Requirements . . . . .	11
3.1.1	Registration and login . . . . .	11
3.1.2	Profile management . . . . .	11
3.1.3	User side taxi request . . . . .	12
3.1.4	User side taxi sharing . . . . .	12
3.1.5	User side taxi reservation . . . . .	12
3.1.6	Taxi driver side notifications . . . . .	13
3.1.7	Queue Management . . . . .	13
3.2	Scenarios . . . . .	14
3.2.1	Registration . . . . .	14
3.2.2	Login . . . . .	14
3.2.3	Taxi request . . . . .	14
3.2.4	Check for taxi status . . . . .	14
3.2.5	Taxi reservation . . . . .	15
3.2.6	Delete reservation . . . . .	15
3.2.7	Share a taxi . . . . .	15
3.2.8	Taxi driver availability . . . . .	15
3.2.9	Taxi driver confirmation . . . . .	15
3.3	Event flow and Scheme Diagrams . . . . .	15
3.3.1	Use Case Diagram . . . . .	16
3.3.2	Class Diagram . . . . .	17
3.3.3	Registration . . . . .	18
3.3.4	Login . . . . .	20
3.3.5	Taxi request . . . . .	22
3.3.6	Check for taxi status . . . . .	24
3.3.7	Taxi reservation . . . . .	26
3.3.8	Delete a reservation . . . . .	28
3.3.9	Share a taxi . . . . .	30
3.3.10	Taxi driver availability . . . . .	32
3.3.11	Taxi driver confirmation . . . . .	33
3.4	Non-functional Requirements . . . . .	33
3.4.1	User friendliness . . . . .	33
3.4.2	Portability . . . . .	34
3.4.3	Maintainability & Reliability . . . . .	34
3.4.4	Availability & Performance . . . . .	34
3.4.5	Security . . . . .	34

<i>CONTENTS</i>	3
<b>4 Alloy Modeling</b>	<b>35</b>
<b>5 Other information</b>	<b>36</b>

# Chapter 1

## Introduction

This section aims to show in general terms the purpose and a brief description of the project, with the integration of a glossary which will be useful for the better understanding of the terms.

### 1.1 Purpose

This is the Requirement Analysis and Specification Document of the project. Its purpose is to give fairly detailed specifications for the system to be. It illustrates a complete declaration for the development system, explain constraints and possible interactions with other external applications. The aim of this document is to be proposed to stakeholders for their approval and to be as a reference for the development team.

### 1.2 Scope

Users with prior registration, through a web or a mobile application, are able to request a taxi, giving the starting position of the ride and the number of passengers. The overall area covered by taxis is divided in smaller zones by the system, which picks the first available taxi in the zone selected by the user or place the user in queue if no taxi is available at the moment. It then sends a notification to the chosen taxi driver through the mobile app. The taxi driver has to decide whether to accept or decline the intervention. Once a taxi driver confirms the user is notified of the needed time for the taxi to arrive and is given a code in order to verify his identity. Users can reserve taxis giving the system the starting and the arrival position

besides the starting time. It is not possible to reserve within the last two hours before the ride. If the request for a reservation is accepted by the system it confirms the user about his ride and places a taxi at the starting position 10 minutes before the meeting time. Users can also enable the taxi sharing option when sending a request. The system, in this case, searches for other users who are willing to share a taxi. If there are users who go in the same direction the system alerts all the users and the taxi that the ride will be shared and calculates the fee for each user depending on the length of their ride.

### 1.3 Stakeholders

The government of the city, the institution which submitted and funded the project for the most part. Its main objective is to reorganize and optimize the city taxi service under the request of the inhabitants that complained about the bad quality of the service. The taxi companies which joined the project and funded the remaining part. Their main goal is to earn the most from through a better management of their service. The inhabitants / future users of the application who wants an easier access to the service and a reduced waiting time in queue for a taxi.

### 1.4 Glossary

This section explain the meaning of frequently used terms in this document.

- MTS: acronym of the application and the full name is “MyTaxiService”;
- Guest: person that needs to register to the system in order to access all of the functionalities;
- User: a person that is registered to the system and can access to the costumer’s services;
- Taxi driver: the person who is logged in as a taxi driver. It is needed to distinguish between normal users (customers) and the taxi drivers.
- Application: with this term it is intended both the web and mobile application. When a different specification for the web and mobile is needed, it will be explicitly said.
- Request: an immediate need of a taxi.
- Reservation: prenotation of a taxi in a postponed time and date.

- Ride: used to identify both requests and reservations.
- Busy: when a taxi driver has accepted a ride and therefore he is not in the queue.
- Available: when a taxi driver is in a queue waiting to accept a ride.

## Chapter 2

# Overall description

### 2.1 Product Perspectives

The application to be implemented is going to be mostly independent and self-contained although it will have to be supported by some external systems. The management of the taxi queues and the allocation of the taxi for each passenger is done automatically by the application. myTaxiService is going to be developed as both a Mobile and Web application and so, in order to correctly work, it will need to be integrated with Internet. Another integration is the use of GPS for the navigation system of each taxi. It is going to be used to gather information about the position of the taxis, so that the system can calculate the optimal route and the number of taxis in each zone.

### 2.2 Product Functionalities

The actors and the relative features for each one that myTaxiServer offers are:

**A1** Guest: guests can:

- register;
- log in.

**A2** User: users can:

- modify personal information;



- request a taxi;
- check taxi status;
- make a reservation;
- dismiss a reservation;
- abilitate sharing option.

**A3** Taxi Driver: taxi drivers can:

- change availability;
- confirm or decline a request.

## 2.3 User characteristics

Users of the application are required to have a basic web knowledge and a device, like a personal computer or a smartphone, to be able to utilize the service and an identification document in order to avoid the registration of multiple accounts by the same person. Taxi drivers aside from the previous specifications and the strict necessity of a smartphone with integrated gps system are registered by the taxi company to avoid any kind of illegal use.

## 2.4 Constraints and General Assumptions

### 2.4.1 Regulatory policies

The system has to follow the laws about the privacy of the data given by the users and the policies regarding cookie storage.

### 2.4.2 Hardware limitations

On the user side the application has to run on a device connected to the internet while on the taxi driver side the device has to be equipped also with a GPS system. The system has to run on a rented or owned server.

### 2.4.3 Software limitations

- If it is run on a mobile device, the application needs to be compatible with the following operating systems: iOS, Android and Windows Phone, covering at least the 90% of the devices.

- The application, if used via the web, has to be compatible with the most important browsers (Chrome, Firefox, Internet Explorer and Safari).

#### **2.4.4 Interfaces to other applications**

API's for maps, GPS management and the navigation system, besides the use of the different application markets are necessary. An e-mailing interface is also needed to send confirmation of the various reservations and requests.

#### **2.4.5 Parallel operations**

MTS needs to support parallel operations from different users and taxi drivers

### **2.5 Domain Assumptions**

All the possible ambiguities and the needed hypotheses are clarified in this section and explained with all their reasons.

#### **2.5.1 User identification**

In order to avoid possible multiple accounts or any kind of abuse of the application, user must register themselves with a valid email address and also must attach a valid identification document. Both of them are unique by definition.

#### **2.5.2 Taxi Driver identification**

The mobile application for drivers isn't available on marketplaces, this owing to prevent from any kind of cyberattack or illegal use. Every taxi company provides the application and a unique identification code, among all companies, to each driver.

#### **2.5.3 Taxi zones**

The city is divided in zones, each one with two square kilometers of extension approximately.

#### **2.5.4 Taxi availability**

When a driver declare his availability to the system, he is in the queue and can answer to a request in every moment. The driver may also not give the disposability and the system doesn't consider him as in a zone queue.

### **2.5.5 Reservation**

The taxi driver chosen by the system for a reservation is always an available taxi .

### **2.5.6 Sharing option**

All shared rides are notified to the driver, who is informed of all destination he has to reach, and also all passengers.

### **2.5.7 Sharing decline**

When a user receives a confirmation of a shared ride, he can't delete the call anymore.

### **2.5.8 GPS correctness**

All the GPS data incoming from different devices are 100% correct.

### **2.5.9 Starting point correctness**

The starting point inserted from a user is always correct.

## Chapter 3

# Specific Requirements and Scenarios

In this chapter there are the main functional requirements with the relative scenarios, event flows and scheme diagrams. The reader can also find the non-functional requirements of the application.

### 3.1 Functional Requirements

In this section are described the main functional requirement.

#### 3.1.1 Registration and login

1. Guests, who are not yet logged in the system, are only able to login or to register.
2. If a guest tries to login with wrong credentials he is requested to retry and sent to the previous page.
3. During the registration guests have to give a valid email address to which will be sent a confirmation email and a valid ID document code to demonstrate their legitimacy. The system checks for the authenticity of the document before accepting the registration.
4. When registering a guest has also to insert a password with at least eight characters.

#### 3.1.2 Profile management

1. Users are able to add and modify their email, password, telephone number, home address.

2. Users can change the ID document code but it has to be verified by the system.
3. If the users modifies the password, the system must ask for the old password in order to verify the authenticity of the user. This to increment the security of the application.
4. Users cannot add more than one home address and ID document code.
5. Users are not able to modify their name and surname.

### 3.1.3 User side taxi request

1. If users give non existing starting or destination addresses the system notifies them to try again and refuses the request.
2. If the user has already a pending request and asks for another one he is notified that the second request will not be forwarded and it is refused.
3. If the request is valid the user is asked to wait for the first taxi that will accept it.
4. When a taxi driver accepts the request, the user is given the taxi ID and a code to show to the driver in order to avoid other passengers to take the incoming taxi.
5. A user is able to see where the taxi is and the expected time for its arrival.

### 3.1.4 User side taxi sharing

1. While sending a request or reserving a taxi the user must have the possibility to share the ride with other users.
2. If the ride will be shared the client is advised from the system with a notification containing the percentage of the owed fee.

### 3.1.5 User side taxi reservation

1. To reserve a taxi the user must provide date and time of the reservation.
2. Date must be greater or equal than the one in which the reservation is submitted.
3. Users cannot reserve in the last two hours before the ride and so those hours must be unavailable.
4. Users must insert starting point and final destination of the ride.
5. If users give non existing starting or destination addresses the system notifies them to try again and refuses the reservation.

6. As the system can compute the average time of the reserved ride, if the user asks for another ride for a time in which he should still be on the first one it is refused and the client notified.
7. Users are notified when the taxi driver accepts the ride.
8. Users are able to cancel the reservation until 10 minutes before the meeting time, not later.

### 3.1.6 Taxi driver side notifications

1. Drivers are told by the system in which zone they are working.
2. When ready, the drivers have to alert the system that they're waiting for a request and are put in the queue of their zone.
3. When a request arrives to the system the chosen taxi driver is given the informations about the ride.
4. A taxi driver has to answer the request within 1 minute otherwise the request passes to the next available taxi and he is moved to the bottom of the queue.
5. Taxi drivers must be informed if a ride is shared or not.
6. If a ride is a reservation the taxi driver must be informed of it.
7. Taxi drivers can only see their own requests and reservations and not the ones of other drivers.
8. Taxi drivers cannot receive notification while busy.
9. When a taxi driver accepts a request he is automatically set to busy and removed from the queue.

### 3.1.7 Queue Management

1. The city is divided in virtual zones.
2. Each zone of the city has an assigned queue of taxis.
3. The system is aware of the positions of each taxi through GPS informations.
4. The system knows the statistic distribution of taxi requests and reservations in the city and analyzes through these data the optimal division of taxis in different zones.

5. When a taxi driver becomes available he is informed by the system in which zone he is assigned.
6. In case a request cannot be accepted by any taxi in the zone the system searches for the nearest taxi in the surrounding zones.
7. Taxis rotate their positions during the week to avoid unfair management and significant difference in rides and payment.

## 3.2 Scenarios

In this section are described some of the most frequent use case of the application, divided in several different scenarios.

### 3.2.1 Registration

A person has just downloaded the application from the app store and he would like to make use of it. He decides to register to it and so he clicks on the “sign up” button. A registration form shows up and he fills it with his personal data. If the information are correct he becomes a new user of the product, otherwise he is redirected to the home page with an error notification.

### 3.2.2 Login

A user of myTaxiService needs a taxi ride. He starts the application and clicks on the “login” button. After doing that he is redirected to the login page and so he fills the form with his e-mail address, the one he used to register to the service, and his password. If the informations are correct he is able to make a taxi request, otherwise an error message will appear.

### 3.2.3 Taxi request

A user wants to request a taxi from the central station. He clicks on the “request” button and fills the form, with the starting and destination point. He then waits for the confirmation. When a taxi driver confirms the request a notification with the taxi ID, a code that has to be shown to the driver and the approximate waiting time is sent to the user.

### 3.2.4 Check for taxi status

A user has just requested a taxi and a driver confirmed the ride. The user wants to know how much time is left until the arrival of the taxi and the position of this one. He then clicks on the “check taxi status” button and selects the request he made.

### 3.2.5 Taxi reservation

A user wants to reserve a taxi for the next day. He clicks on the “reserve” button and then completes the form with the time and date, the starting point and the final destination. The confirmation of the insertion of the ride in the system is sent to the user. The system manages the reservation as a taxi request, allocating a taxi ten minutes before the meeting time. From this point the execution flow equal to a taxi request.

### 3.2.6 Delete reservation

A user has made a reservation for a taxi for the next day. He then decides to delete his reservations. He goes to the home page of the app, clicks on the “check taxi status” then he selects the reservation he made for the next day and deletes it and it is removed from the system.

### 3.2.7 Share a taxi

A first user wants to reach a destination and a second user’s destination is on the road of the first one. Both of them decided to use the possibility of sharing the taxi. The system calculates the road and the fee percentage for each passengers and sends a notification to the first available taxi driver. A confirmation arrives to both the users who shares the ride.

### 3.2.8 Taxi driver availability

A taxi driver has just completed a ride and he is able to receive other requests. He then goes to the availability options page on the application and sets himself as “available”.

### 3.2.9 Taxi driver confirmation

A taxi driver receives a request for a ride. If he can take the request he confirms his availability and a notification is sent to the passenger, otherwise the system tries with the next taxi in the queue and so on.

## 3.3 Event flow and Scheme Diagrams

In this section the reader can find the event flow of the above scenarios are analyzed in detail and are accompanied by the relative schemes.



### 3.3.1 Use Case Diagram

Below is presented the Use Case Diagram which represents all the main use cases of the application.

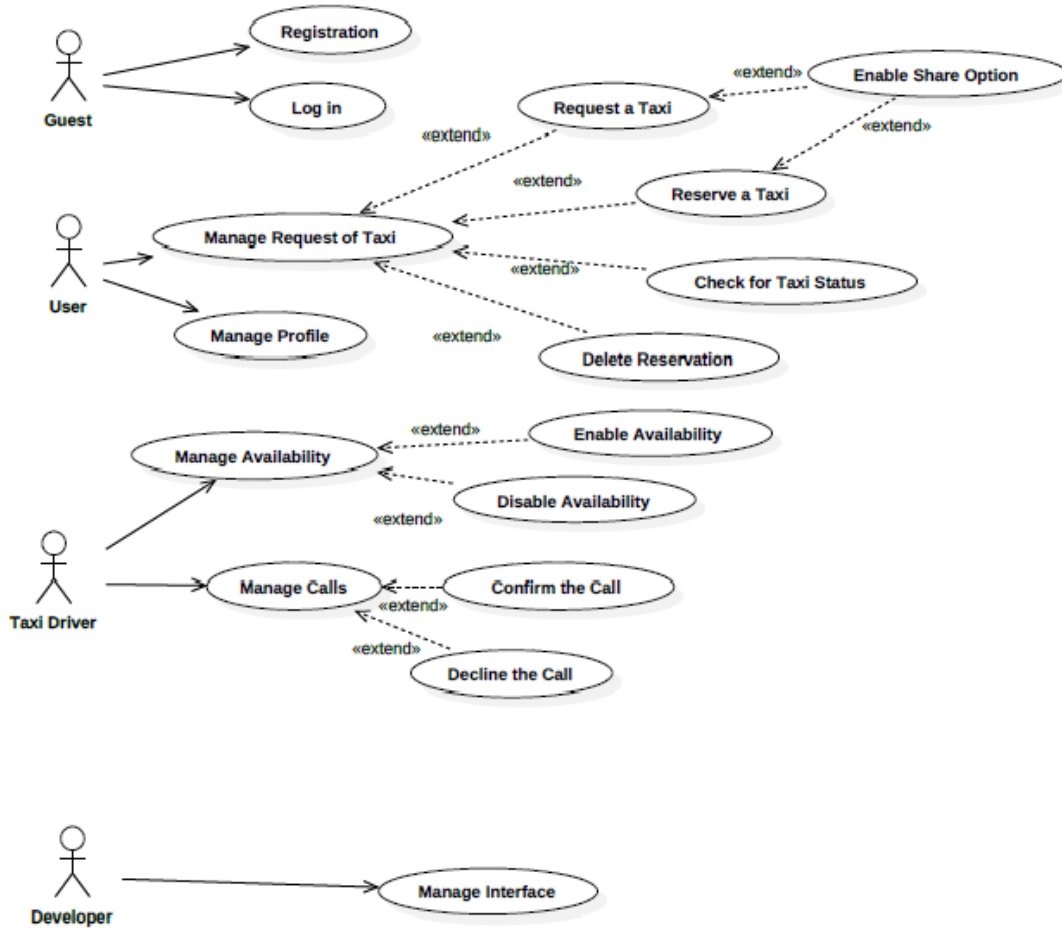


Figure 3.1: Use Case Diagram for myTaxiService

### 3.3.2 Class Diagram

Below is presented the Class Diagram which represents the main classes of the application

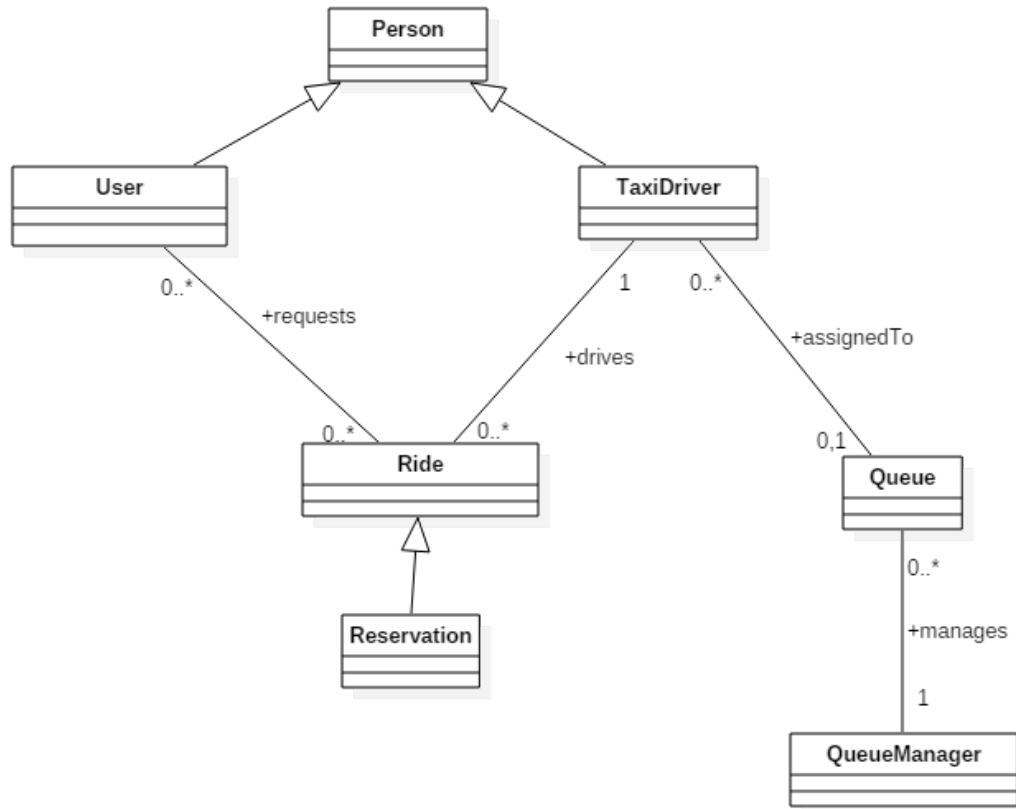


Figure 3.2: Class Diagram for myTaxiService

### 3.3.3 Registration

Actor	Guest
Precondition	the visitor doesn't have an account and he isn't yet registered
Event flow	<ol style="list-style-type: none"><li>1. the guest accesses to the application or to the web page</li><li>2. he then clicks on the "sign up" button</li><li>3. he fills the form with his personal informations, a valid e-mail and ID document</li><li>4. he clicks on "register" button</li><li>5. the system checks the given inputs</li><li>6. the guest is inserted in the system as a registered user</li><li>7. the guest is sent to the home page of the application</li></ol>
Postcondition	the guest becomes a new user of the application and from now on can log in
Exceptions	the e-mail is already present in the system so it is not anymore unique or the given ID document is not valid

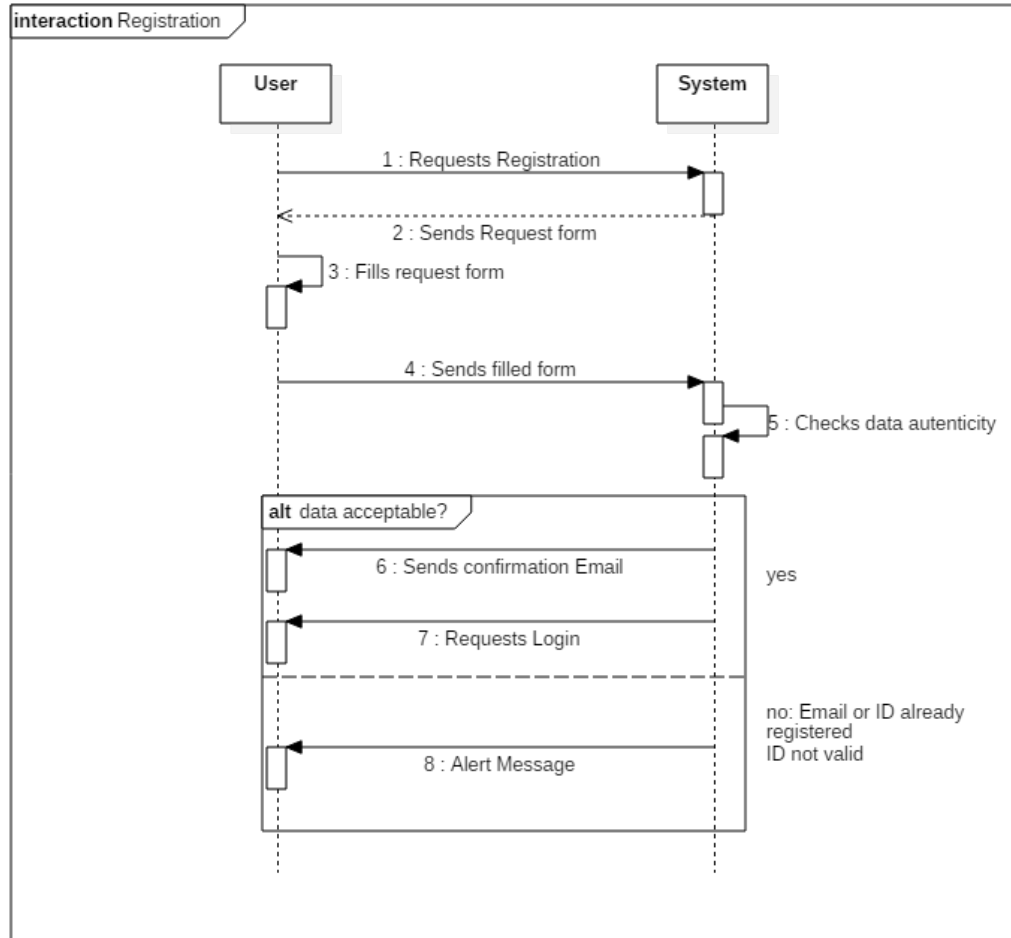


Figure 3.3: Sequence diagram for the registration

**3.3.4 Login**

Actor	guest
Precondition	the guest is registered in the system
Event flow	<ol style="list-style-type: none"><li>1. the guest fills the form, inserting as input his e-mail address and his password</li><li>2. the guest clicks on the “log in” button</li><li>3. the system checks the correctness of the input</li><li>4. the guest becomes a user</li></ol>
Postcondition	the user can access to all the available functionalities
Exceptions	the input password or e-mail are incorrect and the guest cannot log in

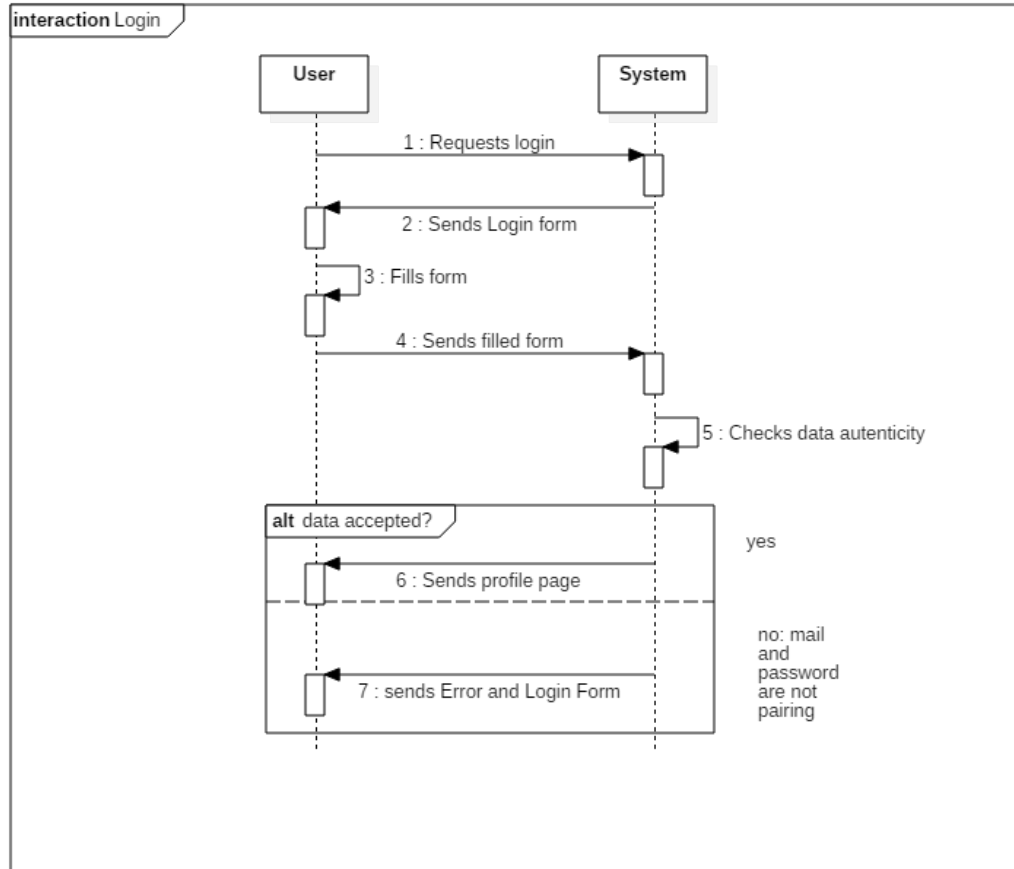


Figure 3.4: Sequence diagram for login

**3.3.5 Taxi request**

Actor	user
Precondition	the user is logged in
Event flow	<ol style="list-style-type: none"><li>1. the user clicks on the “request” button</li><li>2. he then fills the form with the starting point and the final destination</li><li>3. he clicks on the “submit” button</li><li>4. The system analyzes the request and, depending on the zone and the queue of taxis sends a notification to the first available taxi in the queue</li><li>5. When a taxi driver confirms the ride a notification is sent to the user</li></ol>
Postcondition	the user successfully requested the ride and waits for the arrival of the taxi
Exceptions	if the time of the ride coincides with a reservation that the user has previously made the request is refused

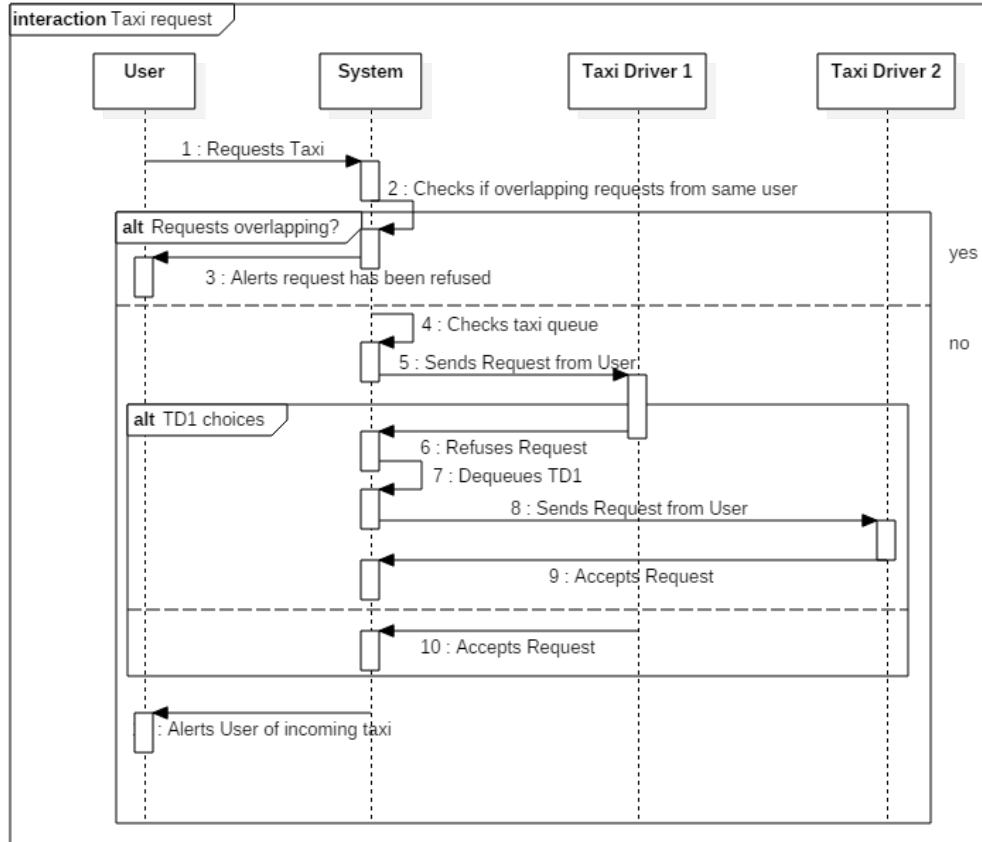


Figure 3.5: Sequence diagram for taxi request



**3.3.6 Check for taxi status**

Actor	user
Precondition	the user has requested a taxi
Event flow	<ol style="list-style-type: none"><li>1. the user clicks on the “check taxi status” on the home page</li><li>2. the system uses the GPS information to calculate the position of the incoming taxi and the expected time</li><li>3. the system shows to the user the information gathered about the incoming taxi</li></ol>
Postcondition	the user knows all the information about the arriving taxi
Exceptions	none

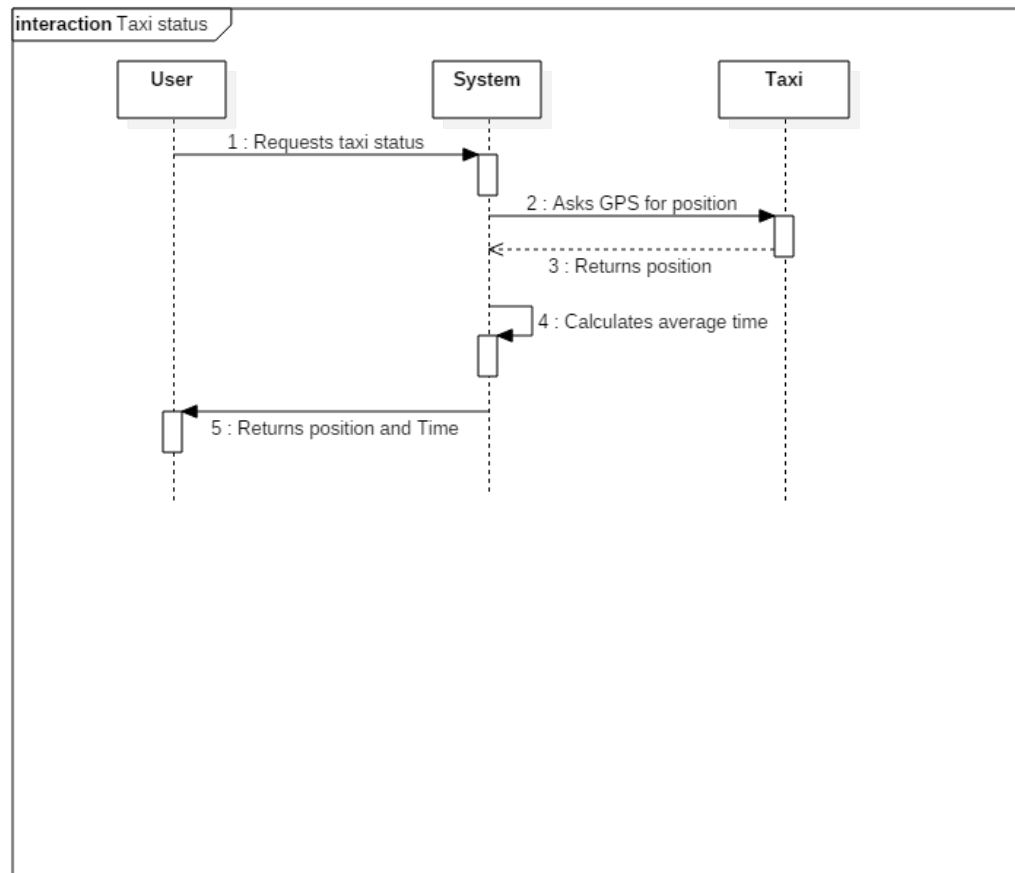


Figure 3.6: Check taxi status sequence diagram

**3.3.7 Taxi reservation**

Actor	user
Precondition	the user is logged in the system
Event flow	<ol style="list-style-type: none"> <li>1. the user clicks on the “reserve” button</li> <li>2. he fills the form with the time, starting point and final destination</li> <li>3. if the information are correct the reservation is inserted in the system</li> <li>4. the system informs the user of the occurred insertion</li> <li>5. the system sends a notification to the first available taxi in the queue 10 minutes before the meeting point</li> <li>6. when a taxi driver confirms the ride a notification with the summary of the reservation is sent to the user</li> </ol>
Postcondition	the user has a new reservation stored in the system
Exceptions	the reservation time coincides with another reservation made in a previous moment

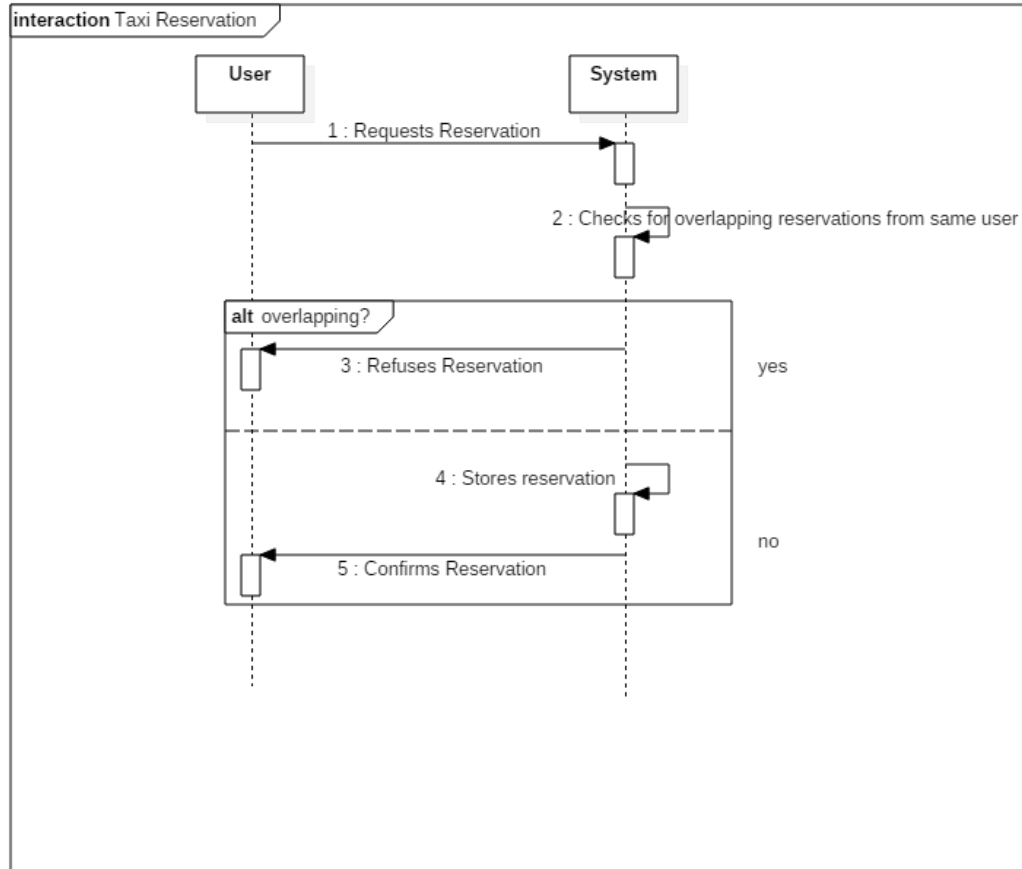


Figure 3.7: Taxi reservation sequence diagram

**3.3.8 Delete a reservation**

Actor	user
Precondition	the user made a reservation
Event flow	<ol style="list-style-type: none"><li>1. the user clicks on the “check reservations” button.</li><li>2. he then selects the reservation to be deleted</li><li>3. the user clicks on the “delete” button</li><li>4. the system removes the reservation</li></ol>
Postcondition	the reservation that the user made is not present anymore in the system
Exceptions	none

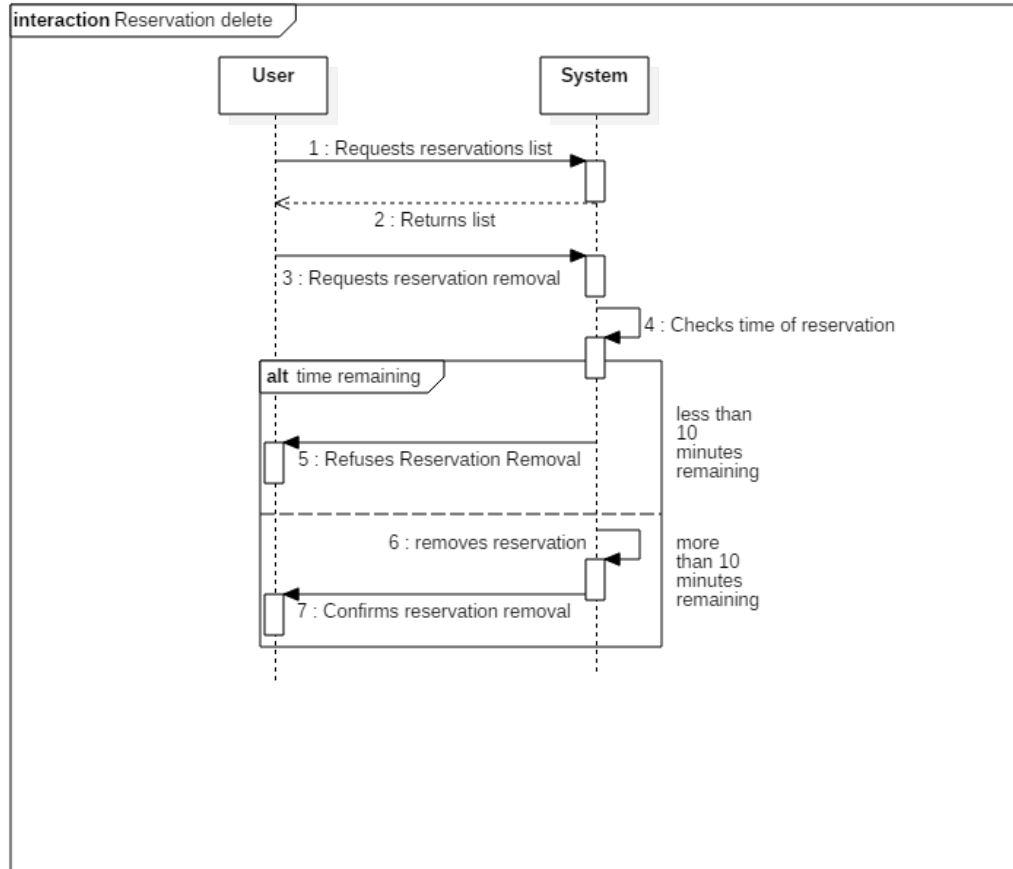


Figure 3.8: Delete a reservation sequence diagram

**3.3.9 Share a taxi**

Actor	user
Precondition	the user is logged in
Event flow	<ol style="list-style-type: none"> <li>1. the user clicks on the “request” button</li> <li>2. he then fills the form with starting point and final destination</li> <li>3. he checks the “share a taxi” option</li> <li>4. the system searches for possible matches with other users</li> <li>5. if a match is found, the system sends a notification to the taxi driver</li> <li>6. when a taxi driver confirms the ride, a confirmation is sent to all the users that share the ride</li> </ol>
Postcondition	the taxi ride is shared among two or more users
Exceptions	there is no user departing from the same zone or going in the same direction. In this case the user cannot share the ride with anyone

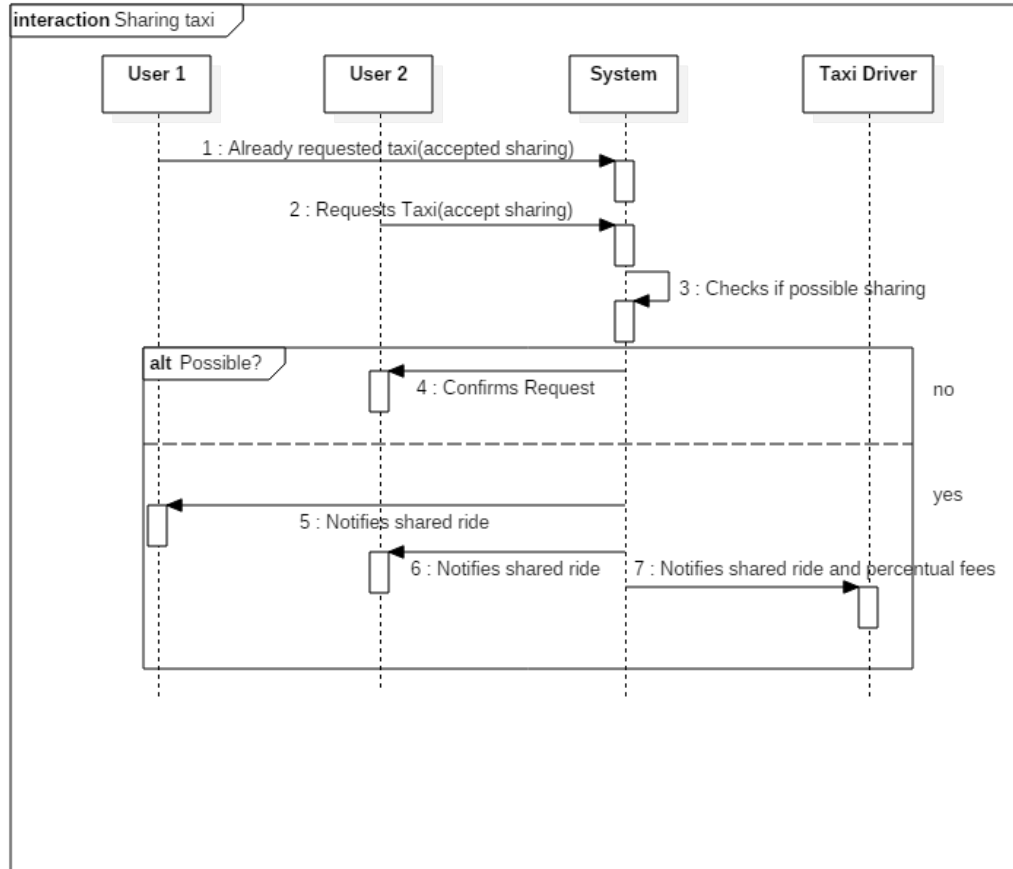


Figure 3.9: Share a taxi sequence diagram



**3.3.10 Taxi driver availability**

Actor	taxi driver
Precondition	he is logged in the system
Event flow	<ol style="list-style-type: none"><li>1. the taxi driver clicks on the “manage availability” button</li><li>2. he is redirected to a page that shows him his availability options</li><li>3. he can choose to set himself as “busy” or “available”</li><li>4. in case the taxi driver switches to “busy” the system removes him from the queue</li><li>5. in case the taxi driver switches to “available” the system adds him to the queue of taxis</li></ol>
Postcondition	if the taxi driver is “busy” the system will not send him request notifications
Exceptions	none

**3.3.11 Taxi driver confirmation**

Actor	taxi driver
Precondition	he is logged in the system and is available
Event flow	<ol style="list-style-type: none"> <li>1. the taxi driver receives a notification for a ride</li> <li>2. He analyzes the allocation request</li> <li>3. He then decides whether to accept or refuse it</li> <li>4. If he accepts it a confirmation is sent to the user</li> <li>5. the system places him out of the queue when the ride is about to begin and sets him as “busy”</li> <li>6. if the ride is refused the system notifies the next taxi in the queue and he is sent in the end of the queue</li> </ol>
Postcondition	the ride is associated with a taxi driver and he is approaching to the meeting point
Exceptions	the taxi driver could not answer in less then one minute, so he is not able to confirm the ride anymore

**3.4 Non-functional Requirements**

This section contains some of the main non functional requirements that the Software to be should fulfil in order to provide a proper quality of the service and guarantee an accurate stream of operations.

**3.4.1 User friendliness**

Due to the wide target of people for whom the application is intended , it should present a very intuitive and effective user interface . Also old aged people and people with a poor knowledge of web or mobile applications should understand how to move inside the application and get a taxi.

### 3.4.2 Portability

The mobile application should be available for all major mobile devices with different kind of hardware and software. Also the web application should be used via all the main browser(e.g. : Chrome,Mozilla) with no kind of restriction of operative system or hardware features.

### 3.4.3 Maintainability & Reliability

The application need high ease of maintenance due to the fact that it has normally to be operative every day of the week , 7/7 days, and every hour, 24/24. Some exceptions could happen, but any necessary maintenance has to be very rare.

### 3.4.4 Availability & Performance

1. The system should be reactive and able to manage at least 1'000 of concurrent and simultaneous operations. Because of this and in order to avoid traffic overloading more than one server will be needed.
2. At least 90% of the operations have to be completed in less than 100 ms.
3. In order to process all the information there is a need to have at least 24 gigabyte of RAM.
4. All the produced data has to be stored, so the servers must have at least 10 terabytes.

### 3.4.5 Security

1. The security of the system is guaranteed in different ways. A user is identified by the tuple email, password, number of id document, whereas a taxi driver is identified by the tuple email,code of company, number of license. No user has to be able to insert data into the database but all the details of the various rides (like the starting and destination point, user, taxi driver...) are stored and used to compute the distribution of the taxis.
2. passwords associated to users must be cripted in order to avoid security failures of the system.

## Chapter 4

# Alloy Modeling

## Chapter 5

### Other information