



# POLITECNICO MILANO 1863

Software Engineering 2 Project

## **MyTaxiService**

PART 1:

Requirements Analysis  
and  
Specification Document

Valeria Deciano 858479  
Fabio Calabretta 852717

Version 2: 1/12/2015

## SOMMARY

<b>1. INTRODUCTION.....</b>	<b>3</b>
<b>1.1 Purpose.....</b>	<b>3</b>
<b>1.2 Description of the given problem .....</b>	<b>3</b>
<b>1.3 Actual System .....</b>	<b>3</b>
<b>1.4 Actors .....</b>	<b>4</b>
<b>1.5 Identification of stakeholders .....</b>	<b>4</b>
<b>1.6 Goals .....</b>	<b>4</b>
<b>1.7 Definitions and Acronyms .....</b>	<b>4</b>
<b>1.7.1 Definitions .....</b>	<b>4</b>
<b>1.7.2 Acronyms.....</b>	<b>5</b>
<b>2. OVERALL DESCRIPTION .....</b>	<b>6</b>
<b>2.1 Product Prospective.....</b>	<b>6</b>
<b>2.2 User Characteristics .....</b>	<b>6</b>
<b>2.3 Constraints .....</b>	<b>6</b>
<b>2.4 Assumption .....</b>	<b>7</b>
<b>2.5 Domain Properties .....</b>	<b>7</b>
<b>2.6 Future possible implementation.....</b>	<b>8</b>
<b>3. REQUIREMENTS .....</b>	<b>9</b>
<b>3.1 Functional Requirements .....</b>	<b>10</b>
<b>3.2 Non Functional Requirements .....</b>	<b>11</b>
<b>3.2.1 User Interfaces .....</b>	<b>11</b>
<b>3.2.2 Documentation.....</b>	<b>26</b>
<b>4. SCENARIOS .....</b>	<b>27</b>
<b>5. UML MODELS.....</b>	<b>28</b>
<b>5.1 Use Case Diagrams.....</b>	<b>28</b>
<b>5.2 Use Cases Description .....</b>	<b>31</b>
<b>5.3 Class Diagram.....</b>	<b>42</b>
<b>5.4 Sequence Diagrams.....</b>	<b>43</b>
<b>5.5 Activity Diagram.....</b>	<b>54</b>
<b>5.6 State Chart Diagrams.....</b>	<b>55</b>
<b>6. ALLOY.....</b>	<b>56</b>

# 1. INTRODUCTION

**This chapter contains same basically information about the software and a description of the document**

## 1.1 Purpose

With this document we want to analyse the system concerned, studying functional and non-functional requirements, use cases, scenarios and consider the limits that the system may present. This document is addressed not only to the client who requests the service, but also experts in the field such as programmers and developers.

## 1.2 Description of the given problem

What it was required by the customer is a new computerized system called "myTaxiService", to manage taxis reservation and optimize this service. The system provides two types of interfaces for the passenger: web and mobile interface. Instead, for taxi drivers there is only a mobile interface through which requests are received. There is also another type of interface to enable the development of new functionalities of the system. The city is divided into zones of approximately 2km<sup>2</sup> to which is associated a taxi queue. This queue is formed by adding a taxi as soon as it becomes available. All this is done automatically by the system also thanks to the GPS localization of the vehicles.

The basic operation is the following: received a request from the client, this is passed to the first available taxi of the queue relating to the zone from which the request comes. The taxi driver can accept the request and, in this case, the information about the taxi reserved are sent to the client, or, if there's no answer within a specific time frame, the request is passed to the next taxi available and, at the same time, the first taxi moves in the last position in the queue. The other possibility that has the customer is to reserve a taxi at least two hours before, specifying the departure and the destination of the route.

In this case the system sends a confirmation, but it assigns the taxi only ten minutes before the meeting time with the user.

## 1.3 Actual System

Nowadays the demand for a taxi is made exclusively through the telephone network by calling a specific number, while the taxi assignment is done manually by an operator with the help of a computer terminal on which you can view in real time the available taxis on a city map. When the reservation is made, it will still be the operator to contact the taxi driver chosen to communicate the address where the customer is located in order to pick him up.

## 1.4 Actors

The actors are:

- USERS: which require the taxi service, making a reservation (specifying the departure, destination and schedule time) or an immediate request (specifying the address).  
Then, they are divided into:
  - registered users: they can do both a reservation and an immediate request
  - non-registered users: they can only make an immediate request
- TAXI DRIVERS: that offer the transport service to the user
- DEVELOPERS: that change or add functionality to the system, whenever it is necessary.

## 1.5 Identification of stakeholders

The main stakeholder of this project is the government organization of the city that requested this system with the objective of optimizing as much as possible the management of the taxi service.

## 1.6 Goals

The objectives of the system are:

- It must be possible to choose the service type to which you are interested: "reservation" or "immediate request".
- Allow to require a taxi by inserting the address that corresponds to the meeting point with the user
- Allow to reserve a transport specifying departure, destination and schedule time. The latter will only be accepted if it is at least two hours after the current time.
- Allow to view and confirm the information about the taxi.
- It must provide to the taxi driver the possibility to accept requests.
- It must be possible to notify the availability state of a taxi
- Must provide a programmatic interface for programmers to allow developing the system

## 1.7 Definitions and Acronyms

With the purpose to minimize the ambiguity writing the document, the meanings of some words used are explained below.

### 1.7.1 Definitions

**User:** is a generic client that uses the taxi service.

**Unregistered user:** is a user who is not signed-up into the system but, however, he can ask for a taxi, but he cannot make reservations.

**Registered user:** a user signed-up into the system, which it is associated some information such as username, password, name, surname, email, phone number and credit card.

**Authenticated user:** is a user who has logged into the system

**Non-authenticated user:** is a user who may or may not be registered in the system but is not logged

**Queue:** is the list in which each taxi is inserted as soon as it becomes available. The queue is relative to a specific area.

**Zone:** is a city area of approximately 2 km<sup>2</sup> in which the taxis are located. For each of them is linked a taxi queue.

**Reservation:** is only used for requests that have a departure, a destination and a schedule time.

**Request:** is used to indicate the services required in real time.

**Route:** is the service made available to the user.

### 1.7.2 Acronyms

**RASD:** Requirements Analysis and Specification Document

**DD:** Design Document

## **2. OVERALL DESCRIPTION**

**This chapter contains general information about the software and an analysis of constraints and assumptions**

### **2.1 Product Prospective**

The new system must be able to interact and to integrate at best with the existing system based on the telephone network. The system also has an internal interface for developers to allow the development of additional functionalities in the system.

### **2.2 User Characteristics**

The user that we expect to use our service is a person who want an easy and intuitive way to reserve a taxi in any place at any time.

This user must be able to use a web browser and have access to internet.

### **2.3 Constraints**

#### **2.3.1 Regulatory policies**

The system uses “cookie” to track user activity and to improve his experience with the service. The system asks the policy agreement only the first time, after which, the answer will be stored in the system. Moreover, personal data stored in the system are used in privacy and security and they will not be disclosed to third parties.

#### **2.3.2 Hardware limitations**

MyTaxiService does not require significant hardware limitations.

#### **2.3.3 Interfaces to other applications**

MyTaxiService must not interact with other applications.

#### **2.3.4 Parallel operation**

MyTaxiService must support parallel operations from different users.

#### **2.3.5 Documents related**

- Requirements and Analysis Specification Document (RASD)
- Design Document (DD)
- User’s Manual
- Testing report

### 2.3.6 Properties

The system must be:

- Scalable: it must be able to develop new features;
- Reliable: it must be guaranteed to work correctly;
- Available: it must be accessible at any time;
- Safe: it must ensure the privacy of personal information;
- Efficient: must use the resources with the least waste of time and effort;
- Usable: it must be easy to use;
- Navigable: the pages of the site/app must be well connected to facilitate the transition from page to page;
- Accessibility: it must be possible to use the system from any browser;
- Readable: the information must be displayed clearly;
- Maintainable: it must be easily maintainable and upgradeable.

## 2.4 Assumption

1. Considering that is not specified if the user must be authenticated and registered in the system we decided to distinguish between two types of users: non-authenticated user who can access into the system and can send a request, and authenticated user, who has all the functionalities of an non-authenticated user with the addition of being able to make (and cancel) a reservation. It can associate a credit card to arrange payment of the service in telematics.
2. Cancellation may be made only in case of the reservation (then only by authenticated users) until ten minutes before the time indicated in this, before it is assigned a taxi.
3. We assume that the a taxi driver has only a minute to respond to a call
4. A user can insert the number of people for whom the service is required, but the system will request more taxis in case they are necessary. The request must in any case be one.

## 2.5 Domain Properties

1. We assume that a request is made for real need and thus there's no reason should be made a cancellation of it.
2. It can't be requested by the same user more than one service at the same time but can be made requests and/or reservations at different times.
3. If an available taxi driver changes zone is put down in the queue of the area where it is located and removed from the queue of the previous area.
4. If it is sent a transport request to an available taxi driver then, in the minute he has to respond, they aren't sent to him other requests.

5. If there's a request or a reservation in a zone, but inside there aren't available taxis, therefore the system uses the queue of adjacent zones.

## **2.6 Future possible implementation**

A possible development for the system is to make available the taxi sharing for users who wish to share the route and saving the cost of travel.



### 3. REQUIREMENTS

**This chapter analyses all the requirements of the system**

We have determined the following requirements:

**1. Registration of a person in the system**

The system must provide a sign up functionality.

**2. Login into the system**

The system must provide a sign in functionality to recognise a registered user.

**3. Request a taxi service**

The system must provide to user the functionalities to insert information about its location and the number of people for whom the service is required.

**4. Reservation of a taxi**

The system must be able to provide to the authenticated user the possibility to make a reservation. Thus it must provide the functionalities to insert the starting and arrival point of the itinerary and the time schedule (which is only accepted if it is at least two hours later than the reservation time).

**5. Display the confirmation and information about taxi**

The system must allow to user to view this information and it must be able to supply within a short time.

**6. Cancellation of a reservation**

The system must allow to the authenticated user to delete one or more reservations respecting the time constraint (at least 10 minutes before the reservation).

**7. View the requests for services**

The system needs to provide only to the taxi driver the possibility to view the requests for transport and it has to provide the possibility to choose to accept the requests.

**8. Communication of taxi availability**

The system shall allow the driver to communicate when he's available for a new service.

**9. Travel payment**

If the user is authenticated and this expresses the desire to pay through the system (if he has registered his credit card), the taxi driver must have the functionalities to request payment.

**10. Add new functionalities of the system**

The programmer can add new functionalities to the system

**11. Modify functionalities of the system**

The programmer can modify a functionality of the system

### 3.1 Functional Requirements

After general requirements, we analyse now those inherent to each actor. In this regard, we distinguish different types of users because they have different capabilities to full availability.

Unregistered user:

- sign up in the system

Non-authenticated user:

- make a request (and obtain the information)

Authenticated user:

- login
- make a request (and obtain the information)
- make a reservation (and display information)
- show reservations and related information

Taxi driver:

- login
- accept the request for transport
- ask payment
- communicate availability

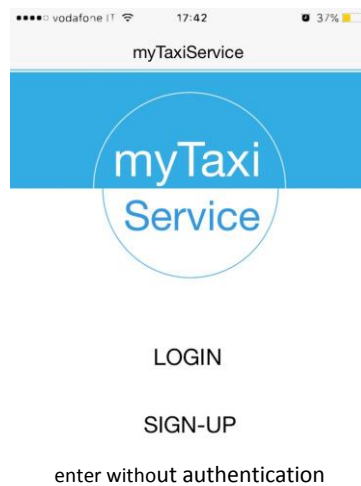
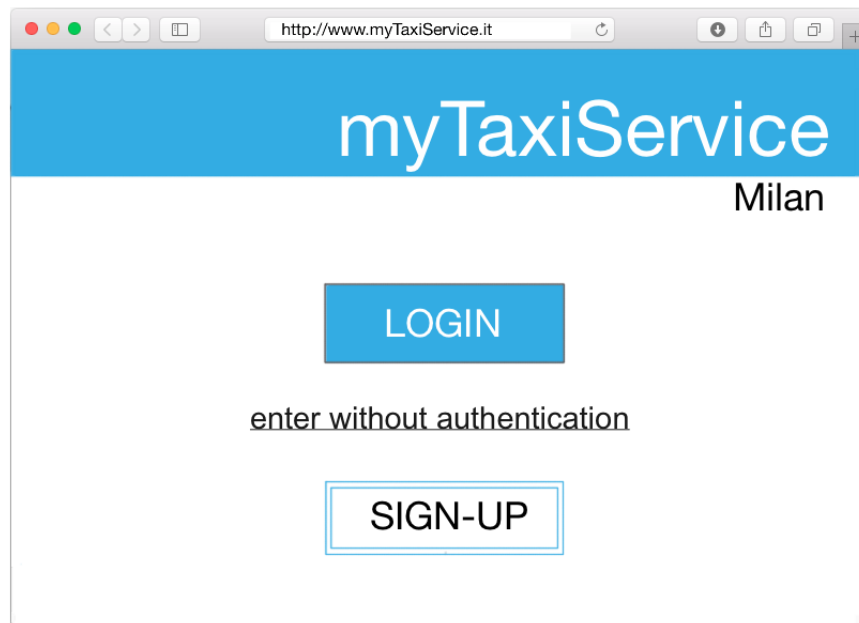
Programmer

- program new functionalities
- modify functionality

## 3.2 Non Functional Requirements

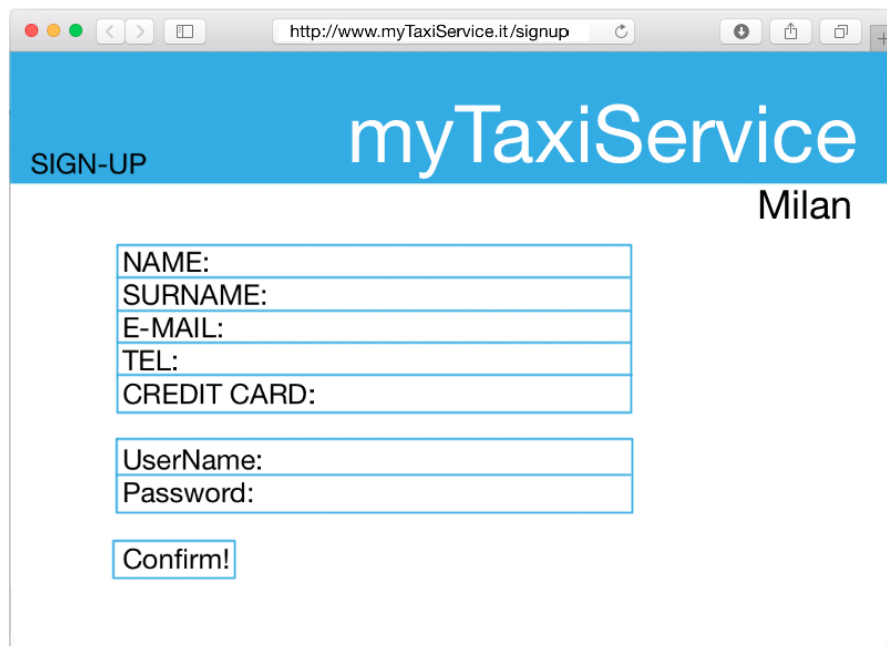
### 3.2.1 User Interfaces

The application interface of myTaxiService is designed for both web and mobile. For each of them, it is provided the Home Page:



Thus you have three options: do log in, for the authenticated user, do a sign-up, for unregistered user, and enter without registration for every type of user.

If you choose to sign up into the system, then will be available this type of page:

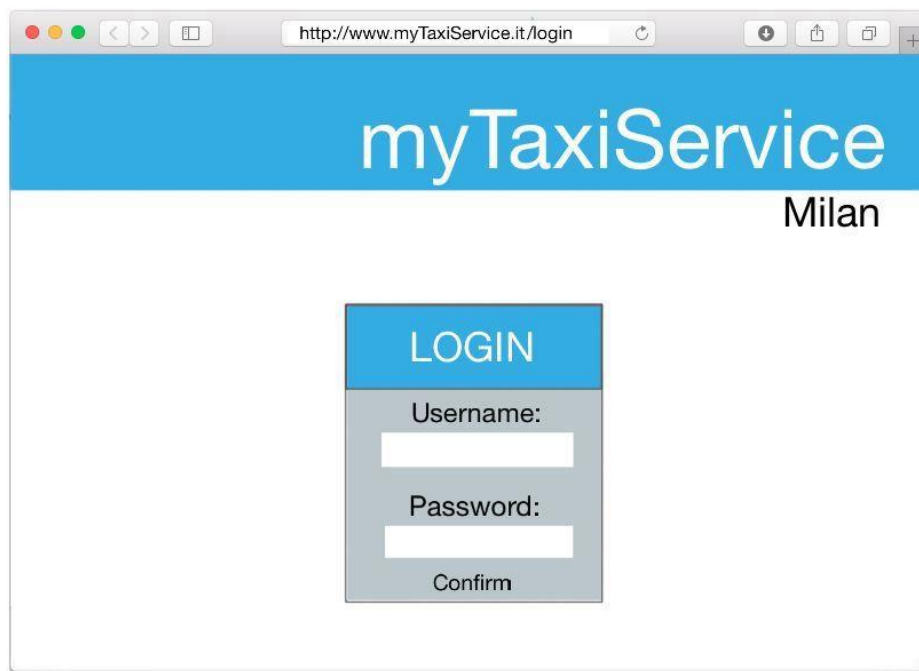


A screenshot of a web browser showing the sign-up page for myTaxiService. The browser's address bar displays "http://www.myTaxiService.it/signup". The page has a blue header with the text "myTaxiService" and "SIGN-UP" on the left, and "Milan" on the right. Below the header, there are several input fields: "NAME:", "SURNAME:", "E-MAIL:", "TEL:", "CREDIT CARD:", "UserName:", and "Password:". At the bottom, there is a "Confirm!" button.



A screenshot of a mobile phone displaying the myTaxiService sign-up page. The status bar at the top shows "vodafone IT", "17:42", and "37%". The page has a blue header with the myTaxiService logo. Below the header, the text "SIGN-UP" is centered. There are seven input fields labeled "name", "surname", "tel", "e-mail", "credit card", "username", and "password". At the bottom, there is a "Confirm!" button.

If you choose to log-in into the system, then you will see a screen like this:

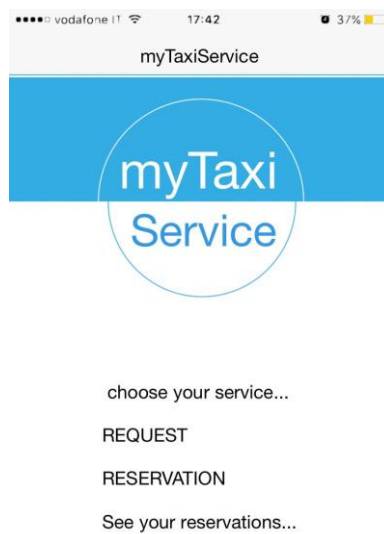
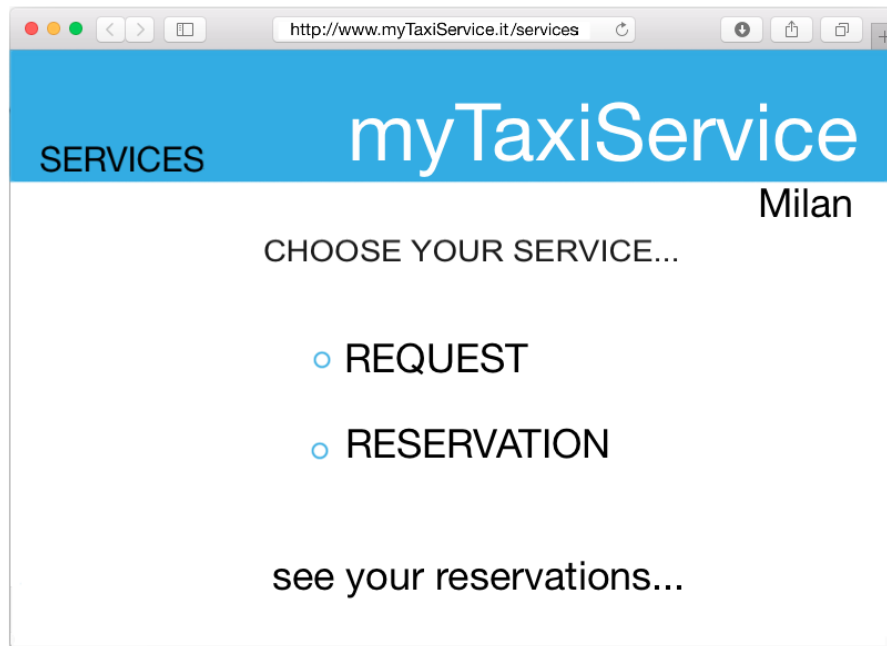


A screenshot of a web browser window displaying the login page for myTaxiService. The browser's address bar shows the URL `http://www.myTaxiService.it/login`. The page has a blue header with the text "myTaxiService" and "Milan" to its right. In the center of the page is a login form with a blue header labeled "LOGIN". Below this header, the form contains labels for "Username:" and "Password:" followed by white input fields. At the bottom of the form is a button labeled "Confirm".

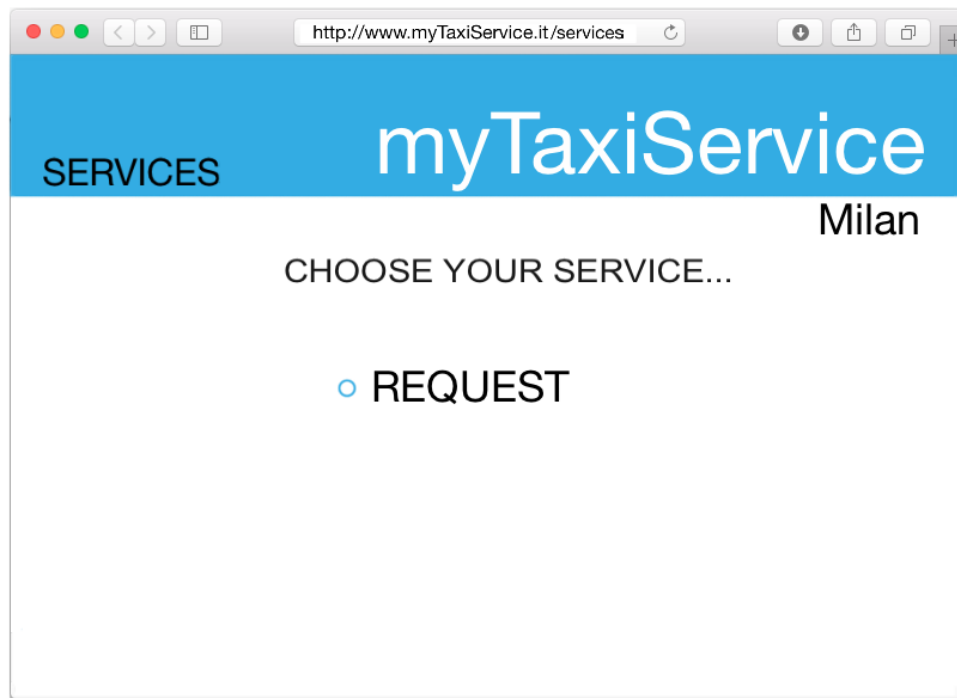


A screenshot of a mobile phone screen displaying the myTaxiService login page. The status bar at the top shows "vodafone IT", the time "17:42", and a battery level of "37%". The page features a blue header with the "myTaxiService" logo, which consists of the text "myTaxi" above "Service" inside a white circle. Below the header is the word "LOGIN". The login form is a blue rectangle containing labels for "username" and "password" next to white input fields. At the bottom of the form is a button labeled "confirm!".

After logging, the authenticated user will have three functions: make a reservation, make a request or see reservations made:



However, if you aren't an authenticated user (and you choose "enter without authentication") then you will have a "non-authenticated user Homepage" only with the chance to make a request.



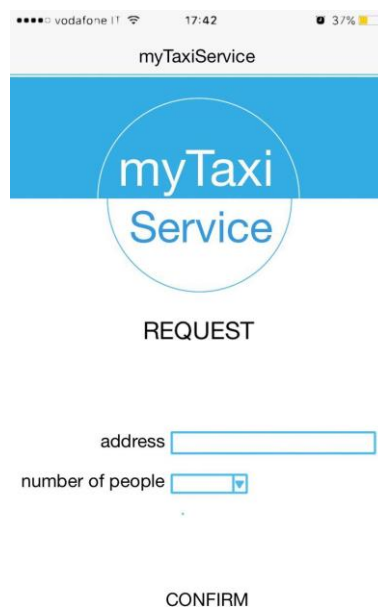
choose your service...

REQUEST

If you choose to make a request, then you see this screen:



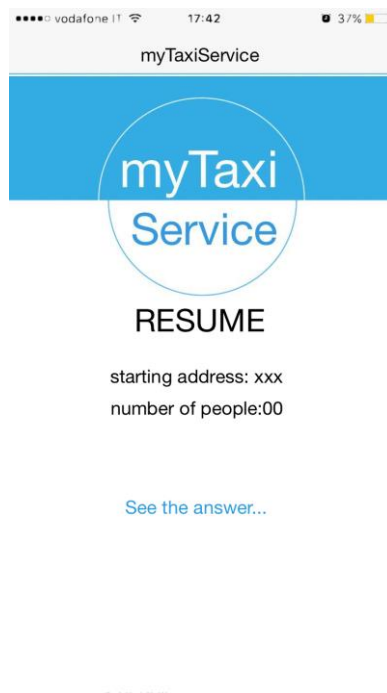
A screenshot of a web browser window showing the 'myTaxiService' website. The browser's address bar displays 'http://www.myTaxiService.it/request'. The page has a blue header with the word 'REQUEST' on the left and 'myTaxiService' in the center. On the right side of the header, the word 'Milan' is displayed. Below the header, the text 'Do you want to take a taxi?' is centered. Underneath this text are two input fields: a text box labeled 'address:' and a dropdown menu labeled 'number of people:'. At the bottom of the page, the word 'CONFIRM' is centered.



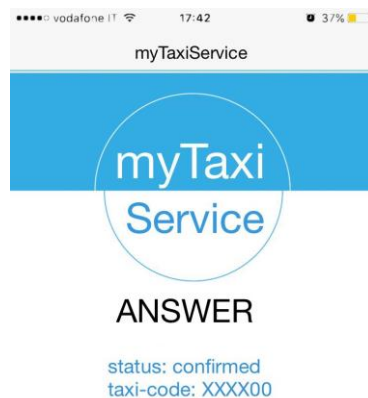
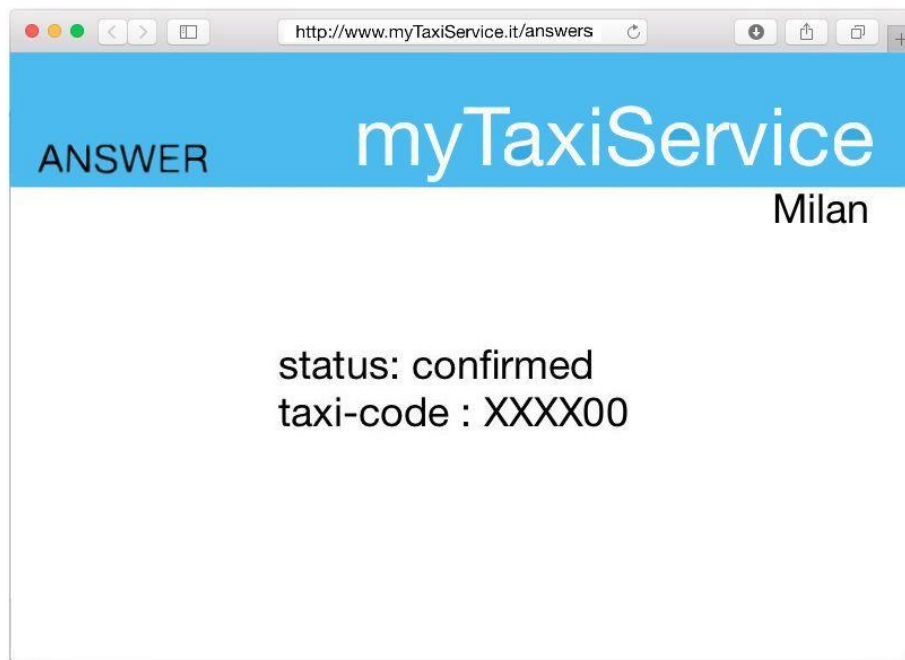
A screenshot of a mobile phone screen displaying the 'myTaxiService' website. The status bar at the top shows 'vodafone IT', the time '17:42', and a battery level of '37%'. The website's header features the 'myTaxiService' logo, which consists of the text 'myTaxi' above 'Service' inside a circular graphic. Below the logo, the word 'REQUEST' is centered. Further down, there are two input fields: a text box labeled 'address' and a dropdown menu labeled 'number of people'. At the bottom of the screen, the word 'CONFIRM' is centered.



Following the confirmation of the request, it will be shown the summary of the application and, from which, you can see the answer.



Clicking “see the answer” you can display the taxi-code and the status of the request, or reservation. In fact also when we do a reservation and confirming it, it is possible to see the answer, in the same way.



Another possibility for the authenticated user is to do a reservation, inserting not only the number of people and the starting address, but also arrival address and the time.

http://www.myTaxiService.it/reservation

RESERVATION myTaxiService

Do you want to reserve a taxi? Milan

starting address:

arrival address:

time:

number of people:

CONFIRM

vodafone IT 17:42 37%

myTaxiService

myTaxi Service

RESERVATION

starting address

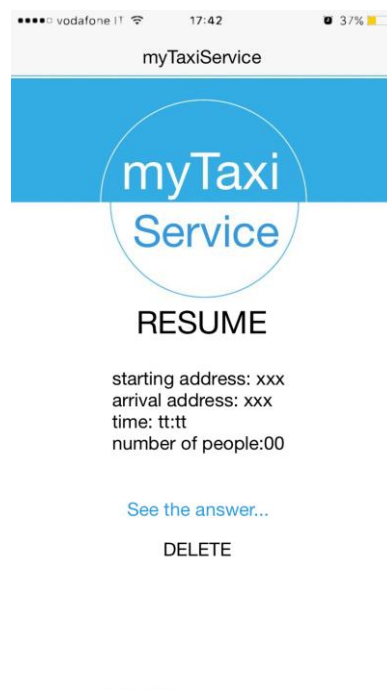
arrival address

number of people

time

CONFIRM

Confirming reservation is displayed the resume of reservation and through it, it is possible to see the answer of the reservation, but also possible to delete by clicking “delete”. It is possible to delete a reservation only if the taxi has not yet been assigned (there are more than ten minutes to the time of reservation).

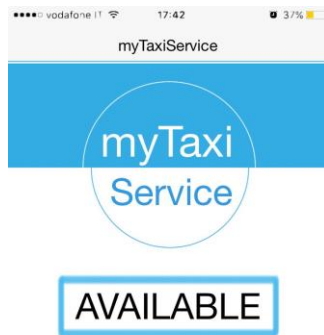


The authenticated user can also view his reservation if there are. Choosing “see your reservations”, in fact the user displays a list of reservation and can see the resume of each one. Through the resume the user can see the answer or delete the reservation



sometimes the system shows error message, such as if the username and/or password are not correct, or if the address doesn't exist. This behaviour of the system are cleared later.

For the taxi driver exists only a mobile application. He shows the same Homepage of the other user, but, after log-in, the system recognizes the taxi driver and shows him a screen, through which he can communicate his availability .



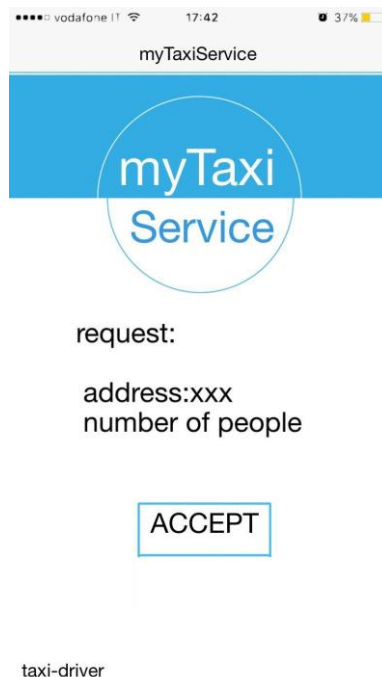
taxi-driver

If the taxi is available, then he can receive a request. But in the waiting period he sees this screen:



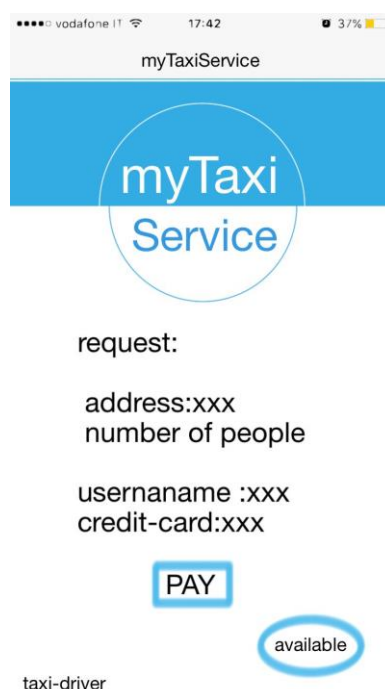
taxi-driver

Only in the moment that the request arrives, further a notified, the taxi driver page is updated and he can accept the request. If he doesn't accept within the time, the request is automatically reject and the system shows the waiting page.

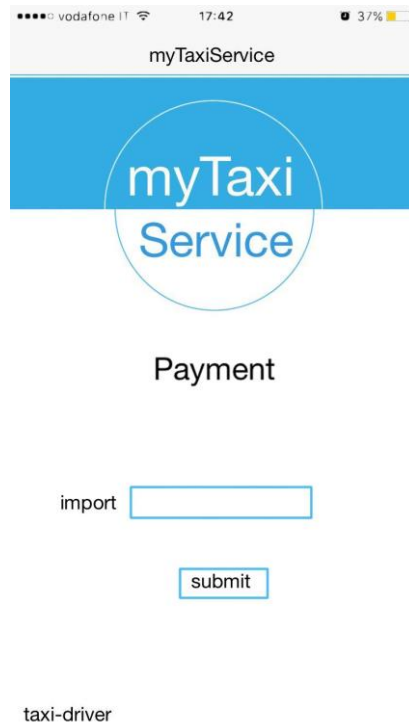


If he accepts the request, the system shows the information of the itinerary. As long as the driver does not make available or will not make the request for payment will remain on this page.

If the client wants to pay automatically (if he is an authenticated user and he has registered a credit card) the taxi driver can request a payment clicking the button "pay"



If he notifies its availability then returns to the waiting state, otherwise, if it requires payment must insert the 'amount and later return to the initial screen where notify, when he wants, his availability status..



The screenshot shows a mobile app interface for 'myTaxiService'. At the top, the status bar indicates 'vodafone IT', signal strength, time '17:42', and battery level '37%'. The app header is 'myTaxiService'. Below it is a blue banner with the 'myTaxi Service' logo. The main screen is titled 'Payment'. There is an 'import' label next to a text input field, and a 'submit' button below it. The text 'taxi-driver' is visible at the bottom of the screen.

There is also a programmable interface for the programmer. In fact, he displays the homepage and, after login, he is recognized by the system that provides him the Programmer Homepage



The screenshot shows a web browser window with the URL 'http://www.myTaxiService.it/ProgrammerHP'. The page has a blue header with 'PROGRAMMER myTaxiService' and 'Milan' on the right. The main content area contains two radio button options: 'add functionality' and 'modify functionality'.

Now he can choose to add a new functionality or modify an existing one.



In the first case, a programmable interface is provided him and he can submit the functionality code.



http://www.myTaxiService.it/AddF

PROGRAMMER myTaxiService

Milan

code...

Submit...

Otherwise, if he chooses to modify functionality, he sees a list of them and select the functionality wants to improve.



http://www.myTaxiService.it/listFun

PROGRAMMER myTaxiService

Milan

- Functionality1
- Functionality2
- ...

So, the functionality code is showed in a programmable interface where he can submit his modifications.



When he finishes to modify functionality, or to add the functionality, the system shows him his homepage.

### 3.2.2 Documentation

Will be realized the following documents;

- RASD: Requirement Analysis and Specification Document that analyses the given problem considering the requirements, the goals, the scenarios, the use cases of the application and hypnotizing the behaviour of the system when this is not completely clear
- DD: Design Document, to define accurately the structure of the system
- Instruction manual: to explain to users how to use the system
- Testing Report: which contains the test results

## 4. SCENARIOS

### **This chapter analyses same possible scenarios of the system**

Fabio would like to organize a disco night with his group of friends this weekend. He knows very well that at a certain time the metro closes and, assuming that the party most likely will end late, he thinks that the best thing to do is to take a taxi for the return. Being 8 people he's concerned that, left the disco, there may be problems or significant delays, reserving the two taxis via mobile phone just after the disco. Therefore researching on Google, he sees the web page of MyTaxiService where is possible to do reservations specifying starting address, arrival address, schedule time and number of people.

Fabio is very happy because that's what he wanted. He does immediately the registration, he reserves the taxi, and finally he downloads the mobile application of MyTaxiService recommended on the website, so as to have everything just a click away!

Valeria is just come out of work and urgently needs a taxi to arrive in time to a drink with friends. It is a regular user of MyTaxiService, therefore she takes its smartphone and using the app immediately makes a request for a taxi. The system processes the Valeria's request and the first taxi in the queue, related to the area where Valeria is situated, it is driven by Gianni, who receives an instant notification on his smartphone to approve the request.

Gianni, however, is on break and he's eating a sandwich; so he doesn't notice the request, which is forwarded to the second driver in the queue, while Gianni's vehicle is located by the system at the bottom of it. Andrea is the second taxi driver who has just received the notification, and he goes immediately to pick up his client. Valeria is very happy and thinks she will continue to use MyTaxiService.

Filippo is a university student and he decided to return at home this weekend. He is usually to reserve a taxi to go to the station, to not make too much effort in carrying the heavy suitcase. With a few taps on the phone he does a reservation using the app. Missing only fifteen minutes, but Filippo is very late and he is not yet ready to go out. He thinks he's not able to be on time to take the taxi. No problem! With the app, Filippo can cancel his reservation until 10 minutes before from the time schedule, without any cost. Now he can makes another reservation and finish getting dressed.

Andrea and Francesca are planning to go to the cinema tonight. Thus they reserved a taxi on MyTaxiService at 08:30 p.m. In the evening, two friends, Giorgio and Martina decided to go with them. Giorgio proposed to take his car, but they don't remember to cancel the reservation. Therefore the taxi driver goes to pick-up his clients but they don't arrive. He waits some minutes and after he decides to leave. And he communicates to the system that he's now again available.

Tomorrow Giulio and Elisa leave for the holidays. They have the plane at 7:00 a.m. and they know that public transport in the early morning pass infrequently. Thus Giulio decides to make a reservation for 5:00 a.m. with the laptop on MyTaxiService.it, right after completing the registration form. After booking, he realizes that the time of the reservation, including the check-in and the travel from home to the airport, is too late and they could not be in time to take the plane. So he deletes the reservation an he makes a new one at 4:00 a.m

## 5. UML MODELS

**In this chapter are presented same UML diagrams that explain the behaviour of the system**

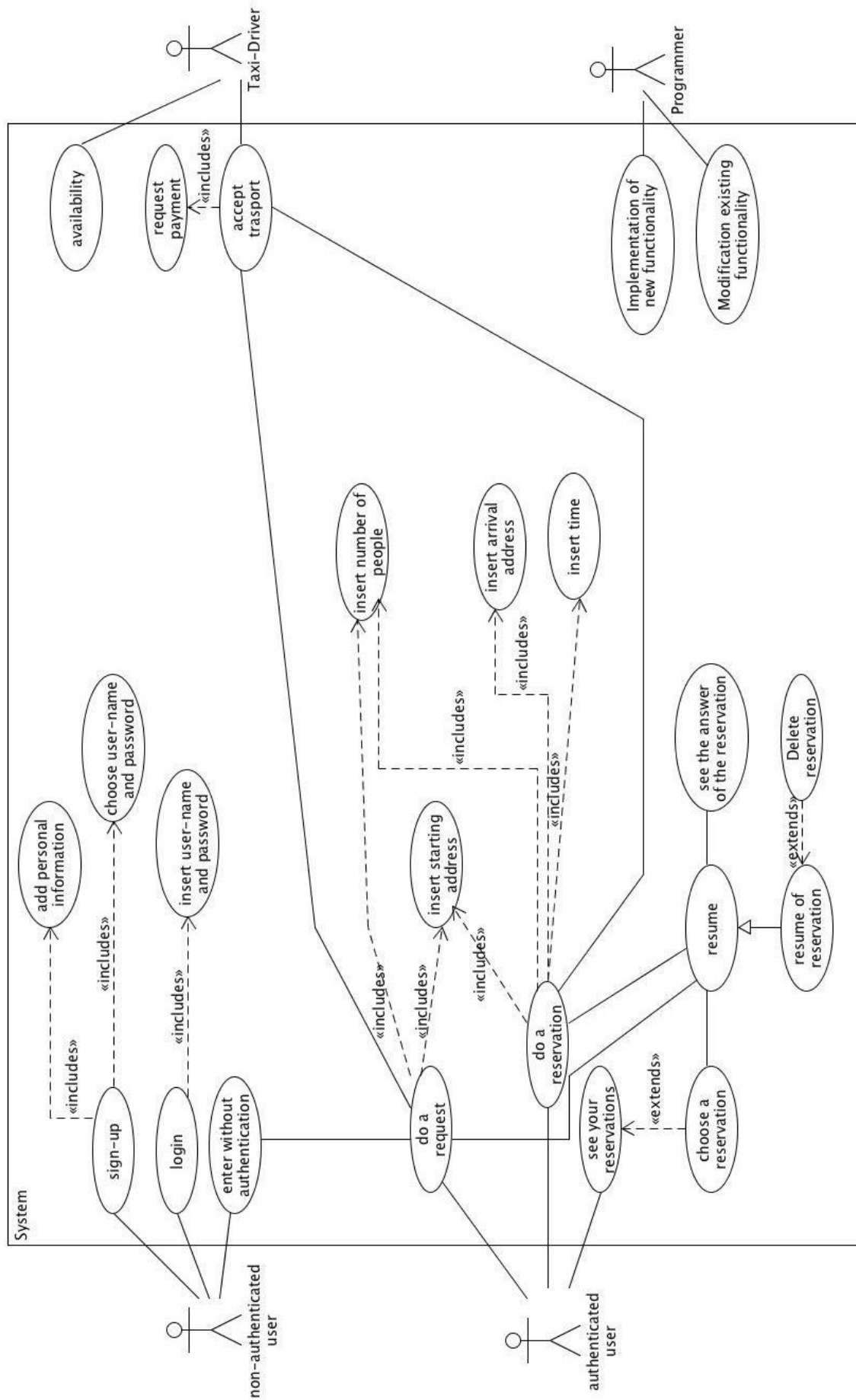
### 5.1 Use Case Diagrams

Analysing scenarios in the previous chapter, we can identify different use cases:

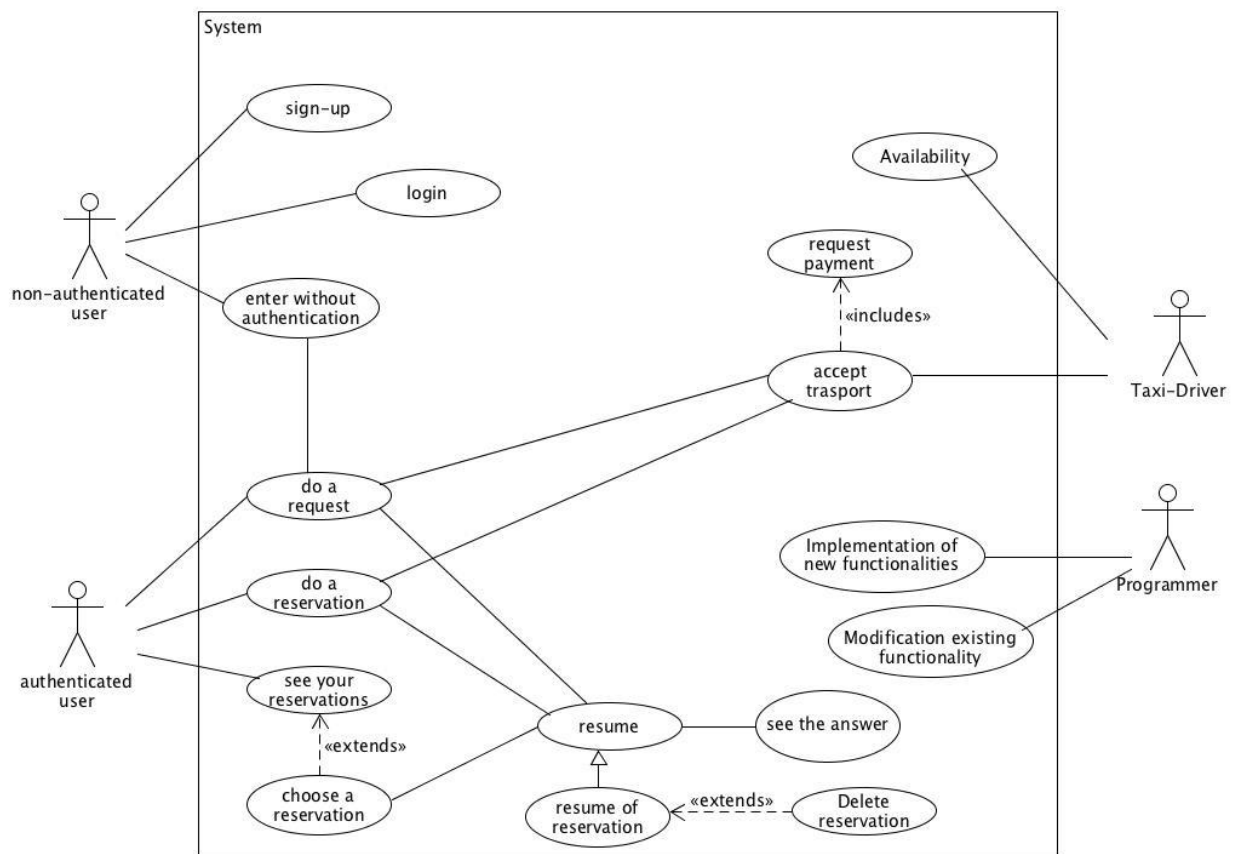
- Sign-up;
- Login;
- Enter without authentication;
- Do a request;
- Do a reservation;
- See your reservations;
- Resume;
- See the answer
- Availability
- Accept transport;
- Request payment;
- Implementation of new functionality;
- Modification of existing functionality;

In the next page is presented the complete use case diagram.

In this diagram are presented some use cases that extend and explain the behaviour of the system but are “included” in other them



So the use case diagram, only with the essential use cases is:



This is the diagram that we are going to analyse in the following part

## 5.2 Use Cases Description

In this part of the document are presented all use cases and each one is described accurately.

We refine the use case “Sign-up”:

Name:	Sign-up
Actors:	Non-Registered user
Entry Conditions:	The user isn't registered to the system
Flow of event:	<ul style="list-style-type: none"><li>- The non-authenticated user enter in the website</li><li>- The user clicks the “SIGN-UP” button</li><li>- The user fills the form where he have to write:<ul style="list-style-type: none"><li>Name</li><li>Surname</li><li>Telephone number (optionally)</li><li>E-mail (optionally)</li><li>Username</li><li>Password</li><li>Credit card code (optionally)</li></ul></li><li>- He clicks the button “confirm”</li><li>- The system shows him the page with the possibility to choose : reservation, request, see your reservations.</li></ul>
Exit Condition:	Registration is done successfully
Exceptions:	<p>Exceptions are caused when:</p> <ul style="list-style-type: none"><li>- One of the not optional field isn't filled</li><li>- The user-name is already chosen by another user</li></ul>

We refine the use case “Log in”:

Name:	Login
Actors:	Non-Authenticated user
Entry Conditions:	The non-authenticated user is already sign-up in the system
Flow of event	<ul style="list-style-type: none"><li>- The user enters in the system</li><li>- He chooses the button “LOG IN”</li><li>- The users submits the user name and password, and clicks “confirm”</li></ul>
Exit Condition:	The system shows to the user a page with the possibility to chose: reservation, request, see your reservations
Exceptions:	The password and/or the user name are incorrectly inserted or one of them isn’t inserted



We refine the use case “Enter without authentication”:

Name:	Enter without authentication
Actors:	Non-Authenticated user
Entry Conditions:	The user isn't authenticated to the system
Flow of event	<ul style="list-style-type: none"><li>- The user enters in the system</li><li>- The user chooses to continue to navigate the website without authentication and clicks the button “enter without authentication”</li></ul>
Exit Condition:	The system shows the Non-Authenticated Home page, where is only possible to do a request

We refine the use case “Do a request”:

Name:	Do a request
Actors:	Non-authenticated user and authenticated user
Entry Conditions:	All the users can show this page
Flow of event	<p>The non authenticated user can do only a request, infect, when he chooses “ enter without authentication” the system shows the Non-Authenticated Home page, where is only possible to do a request</p> <p>The authenticated user, after the login can chooses the service. If he selects “do a request” the system shows this page.</p> <p>The user must to insert the starting address and the number of people</p> <p>The user clicks “confirm”</p>
Exit Condition:	The system shows a page with the resume of the request
Exceptions:	The system could raise an exception if the insert address is not valid or if aren’t inserted some fields

We refine the use case “Do a reservation”:

Name:	Do a reservation
Actors:	Authenticated user
Entry Conditions:	The user must be authenticated in the system
Flow of event	<p>An authenticated user, after the log in, can choose to do a reservation, to do a request or see your reservations. If he chooses to do a reservation clicks the button “do a reservation”.</p> <p>The user must to insert the starting address, the arrival address, the time and the number of people</p> <p>The user clicks “confirm”</p>
Exit Condition:	The system shows a page with the resume of the request
Exceptions:	<p>The system could raise an exception if:</p> <ul style="list-style-type: none"><li>- the insert addresses are not valid</li><li>- the gap of time between the moment he does the reservation and the time of the reservation is less than 2 hours</li><li>- if aren't inserted some fields</li></ul>

We refine the use case “See your reservations”:

Name:	See your reservations
Actors:	Authenticated user
Entry Conditions:	The user must be authenticated in the system
Flow of event	An authenticated user, after the log in, can choose to do a reservation, a request or see your reservations. If he clicks the button “see your reservations” is visible the list of all future reservation, if there are.
Exit Condition:	Clicking the button “resume” is possible to see the resume of chosen reservation.
Exceptions:	The system could raise an exception if there aren’t future reservations

We refine the use case “Resume”:

Name:	Resume
Actors:	Authenticated user and Non-Authenticated user
Entry Conditions:	The user can use this service following a reservation or a request, and also if he choice to see one of his reservations.
Flow of event	Clicking “resume” the system displays the information of the request/reservation
Exit Condition:	Clicking the button “see the answer” is possible to visualize the state and the taxi code, if the taxi is already assigned. In case of a resume of reservation, if the taxi is not already assigned is possible to do a cancellation clicking the button “delete”.
Exceptions:	If the user want to delete a reservations but the taxi is already assigned the system send an error message

We refine the use case “See the answer”:

Name:	See the answer
Actors:	Authenticated user and Non-Authenticated user
Entry Conditions:	The user can use this service following only if he has already done a reservation or a request
Flow of event	He have to click the button “answer” In this page is possible to see the state of reservation and the taxi-code, if it is assigned
Exit Condition:	To return at the home page of an authenticated user clicks on the symbol of “myTaxiService”.

We refine the use case “Availability”:

Name:	Availability
Actors:	Taxi-Driver
Entry Conditions:	The user have to be taxi-driver
Flow of event	After log in, the system recognizes the taxi driver and gives him the possibility to change his state in available
Exit Condition:	Clicking the button “Available” the system is waiting a request of service

We refine the use case “Accept transport”:

Name:	Accept transport
Actors:	Taxi-Driver
Entry Conditions:	The taxi-driver communicates his state of availability and he is waiting for a request of transport.
Flow of event	When a user does a request or a reservation the system sends this to the first taxi driver in the queue. He can respond to the request of service or not, in this case the request is reject after a period of time.
Exit Condition:	Clicking “ACCEPT” the taxi driver accepts the request of transport and displays the details of itinerary

We refine the use case “Request payment”:

Name:	Request payment
Actors:	Taxi-Driver
Entry Conditions:	If a taxi driver accepts a request, and if a client is a registered user with associated a credit card.
Flow of event	He can request a payment, clicking on “payment” and inserting the amount to pay
Exit Condition:	Clicking “submit”, the payment is done and the taxi driver can communicate, if he wants, his availability
Exceptions:	The system could raise an exception if the payment is not accepted

We refine the use case “Implementation of new functionality”:

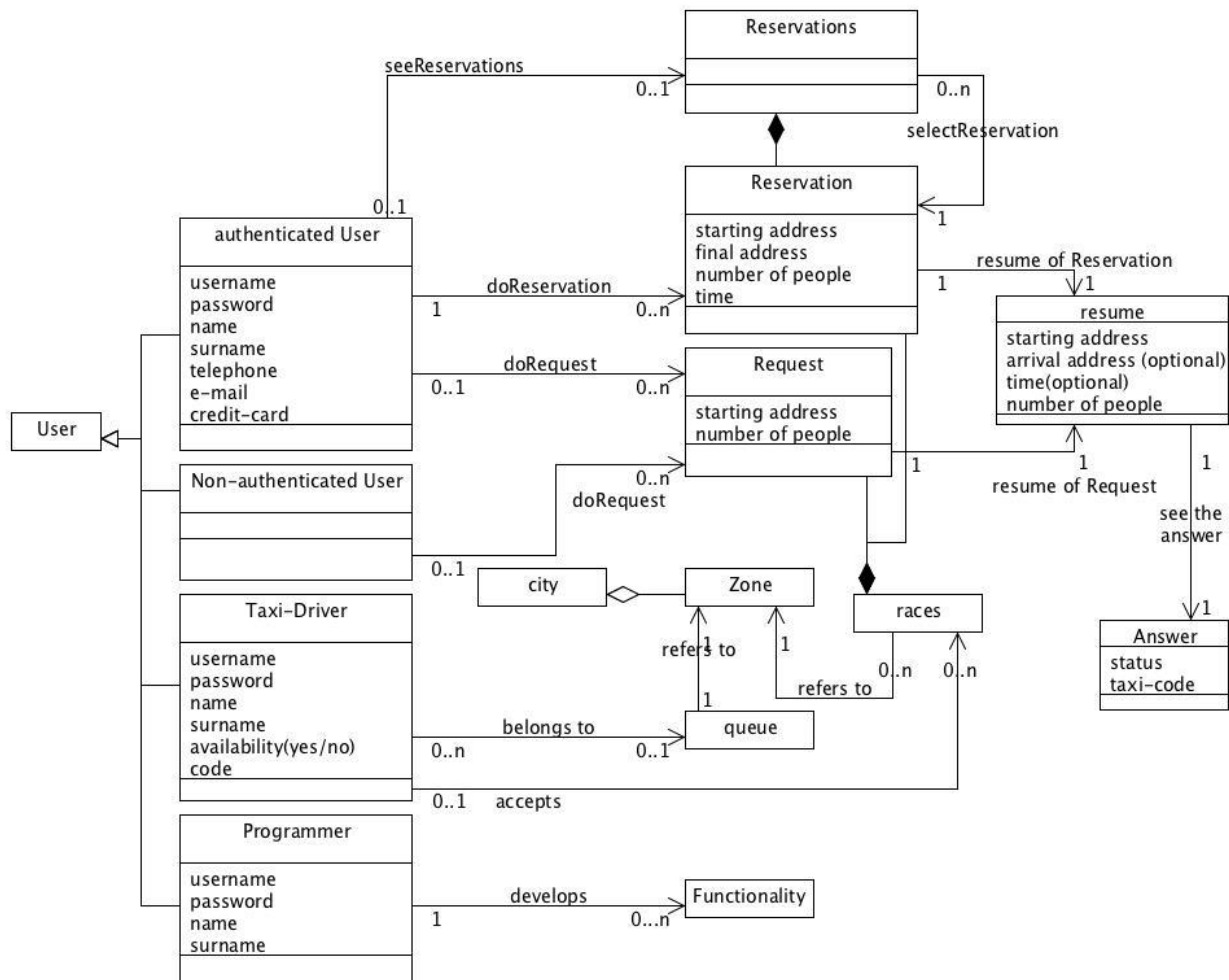
Name:	Implementation of new functionality
Actors:	Programmer
Entry Conditions:	The user must be a programmer
Flow of event	The programmer has a programmable interface to add a new functionalities to the system



We refine the use case “Modification of existing functionality”:

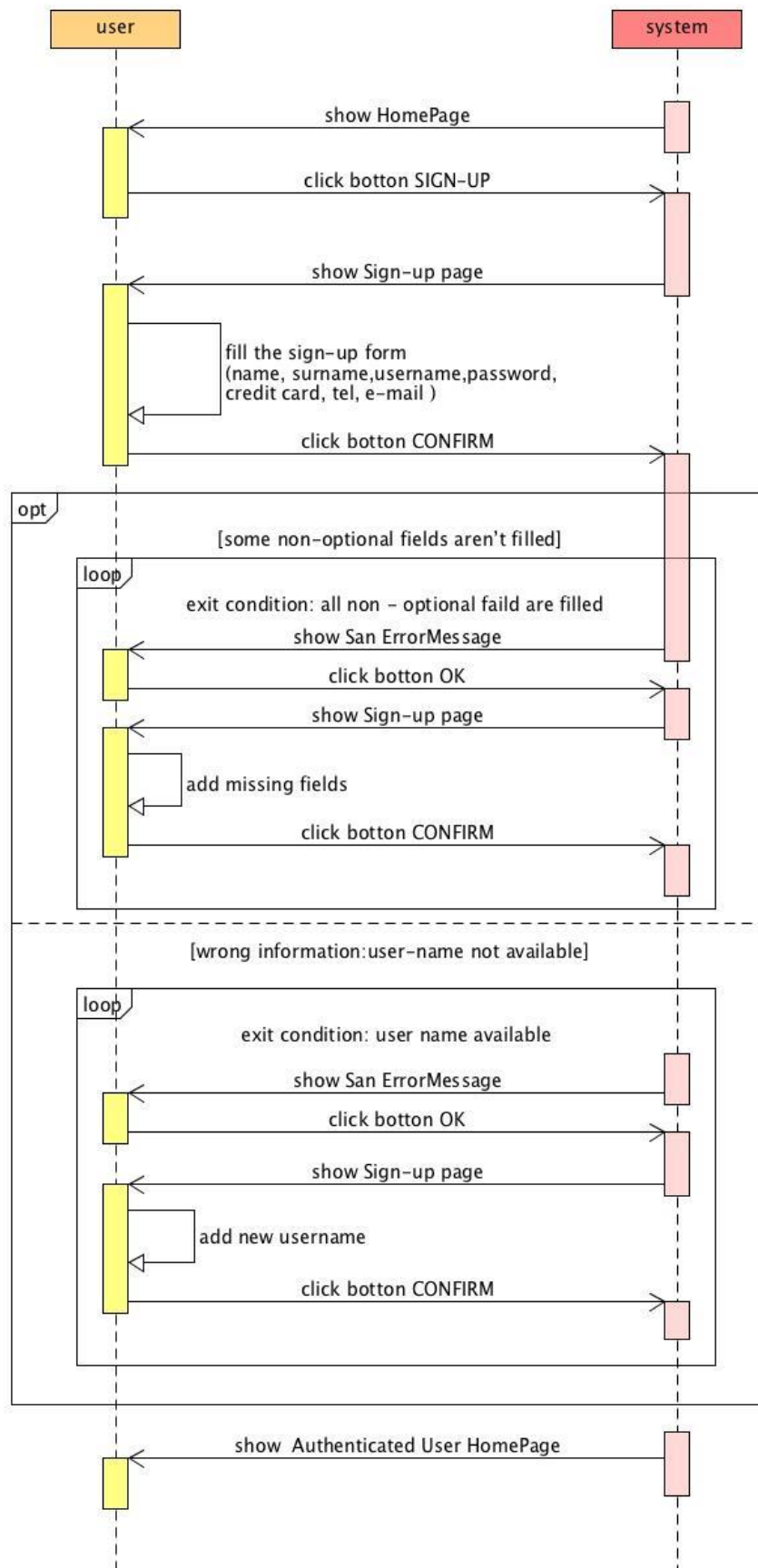
Name:	Implementation of new functionality
Actors:	Programmer
Entry Conditions:	The user must be a programmer and at least one functionality must be in the system
Flow of event	The programmer chooses in a list the functionality he wants modify and he has a programmable interface to submit this modification

### 5.3 Class Diagram

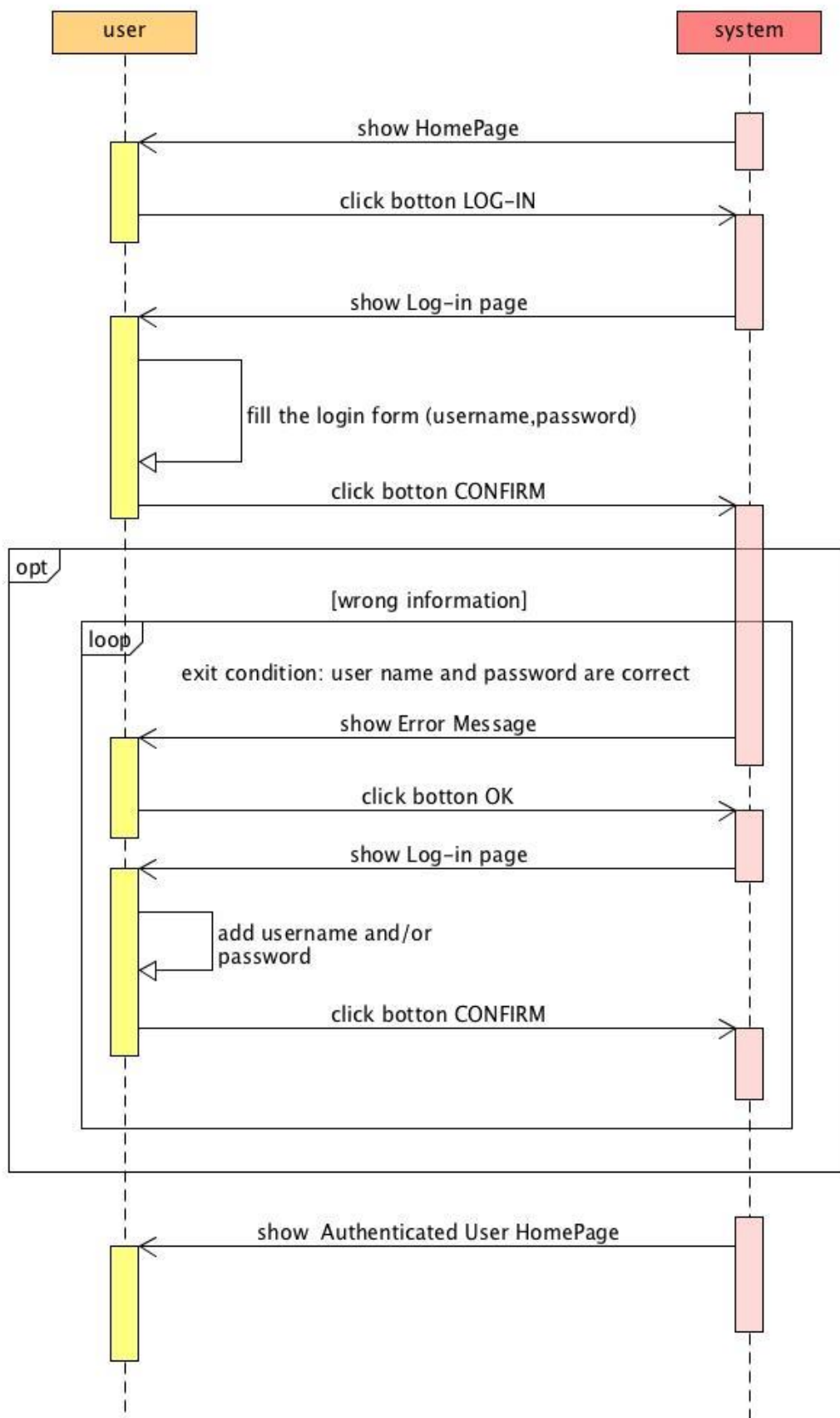


## 5.4 Sequence Diagrams

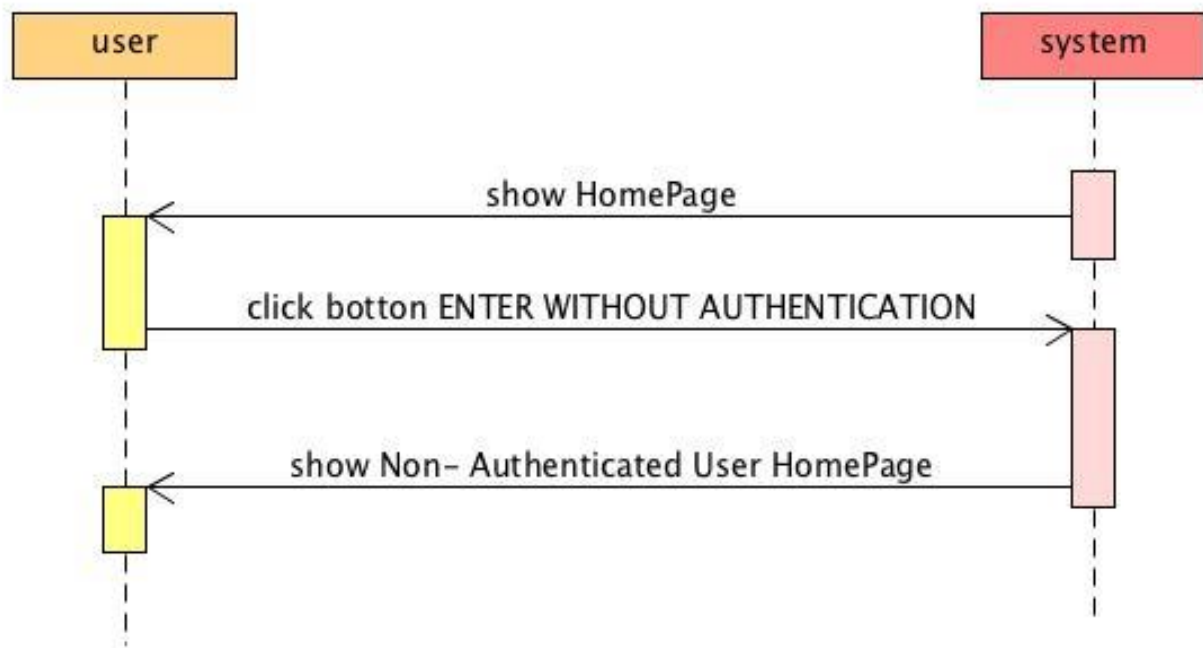
- Sign-up



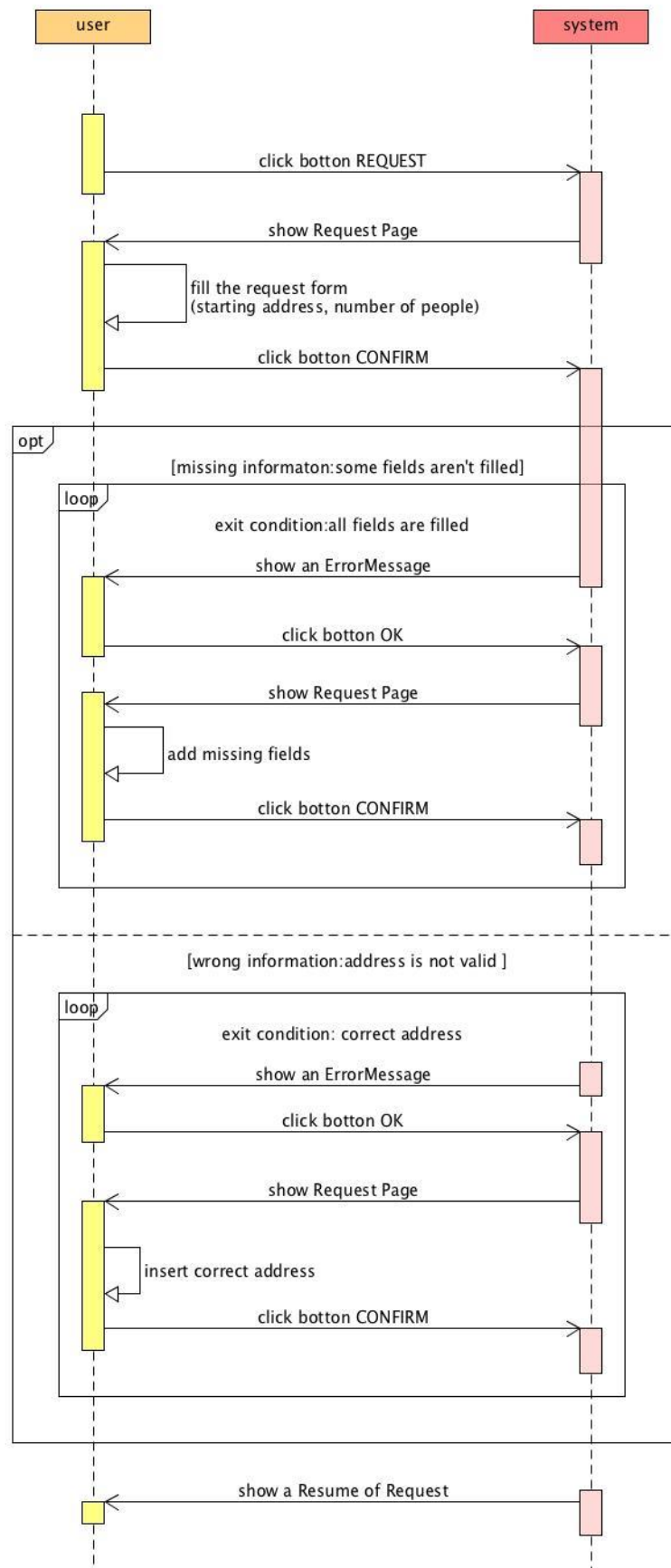
- Login



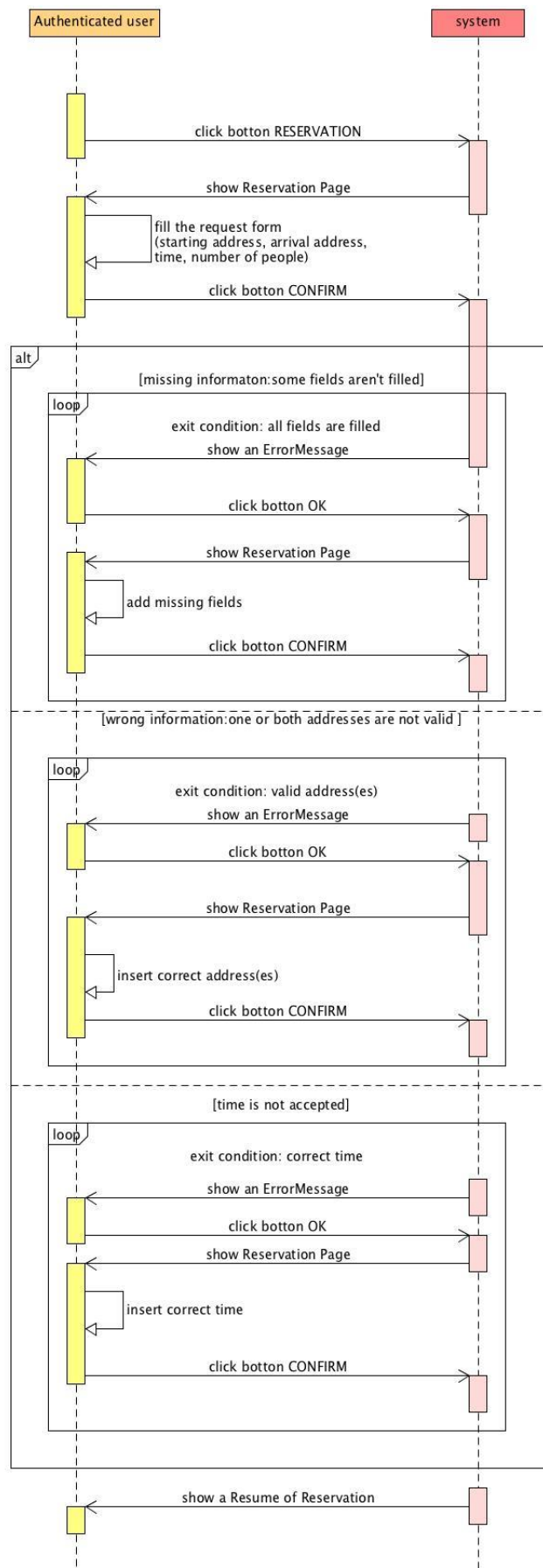
- Enter without authentication



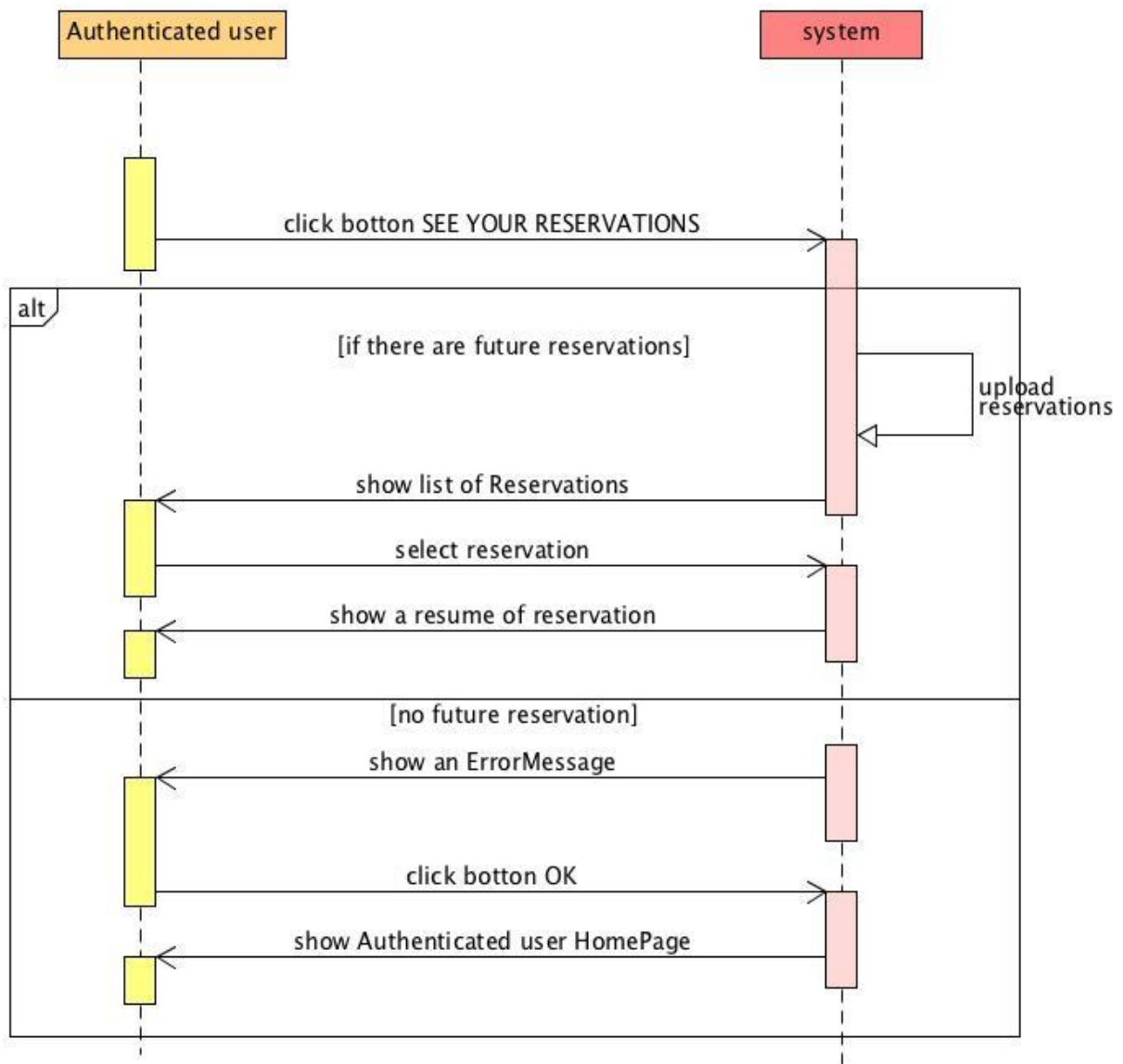
- Do a request



- Do a reservation

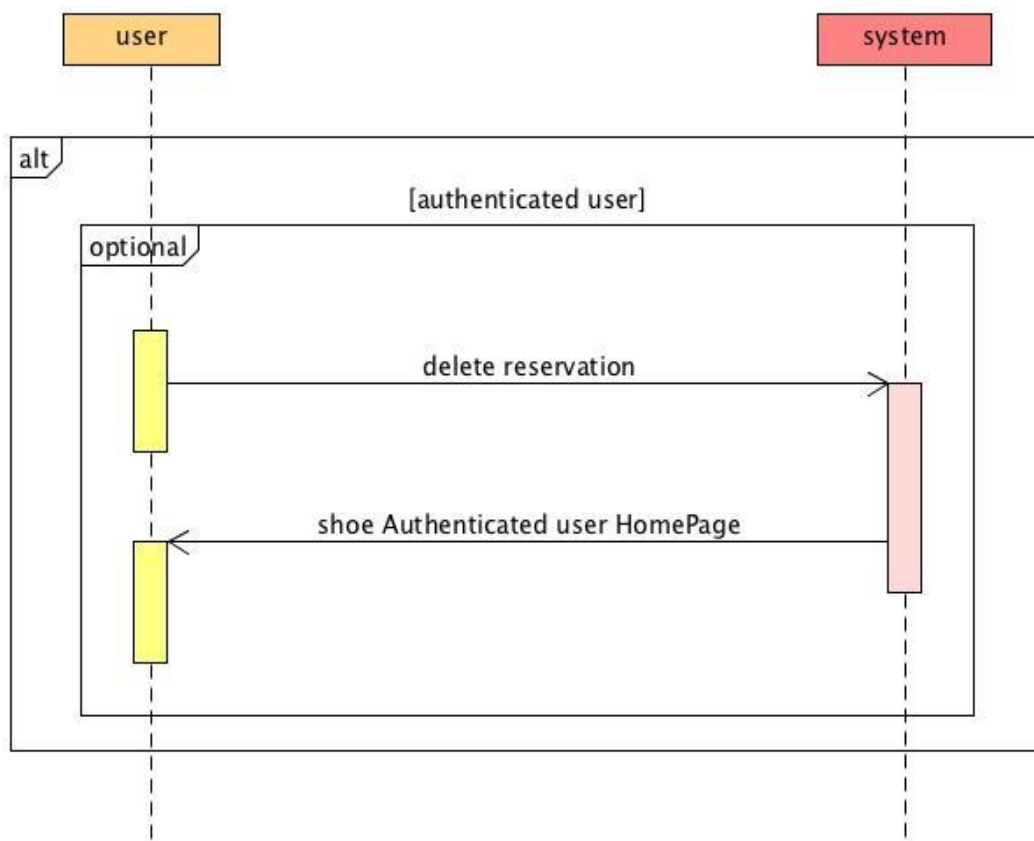


- See your reservations

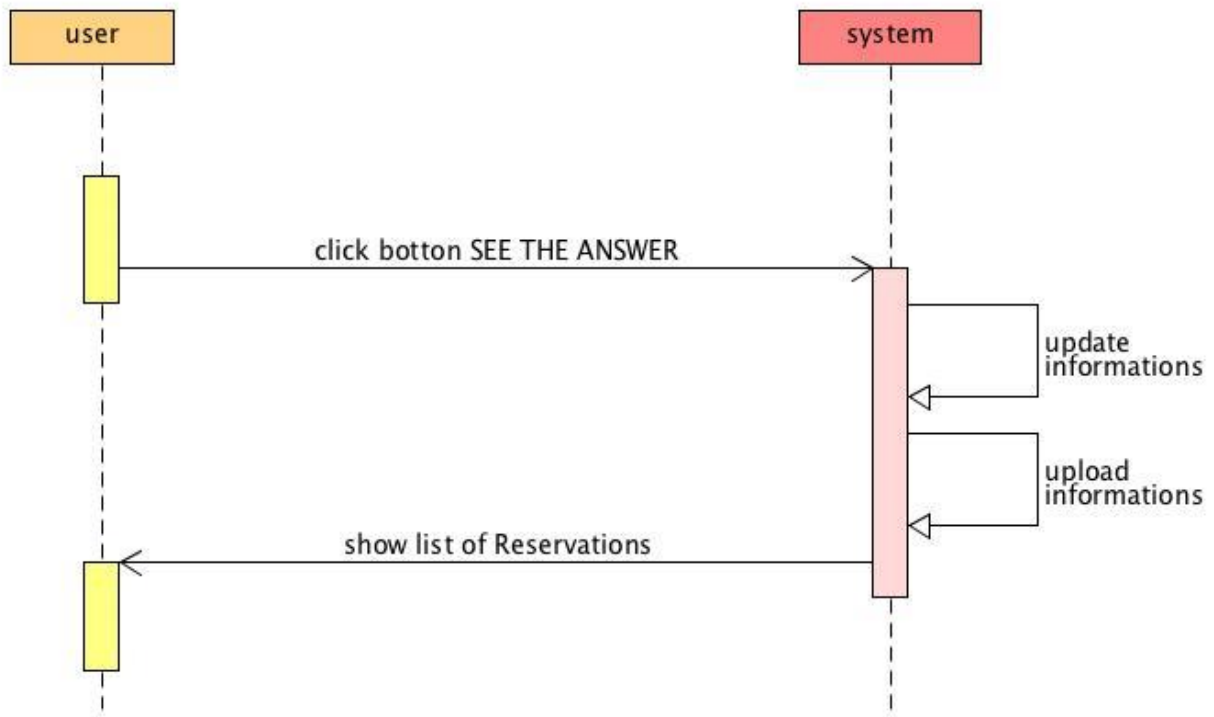




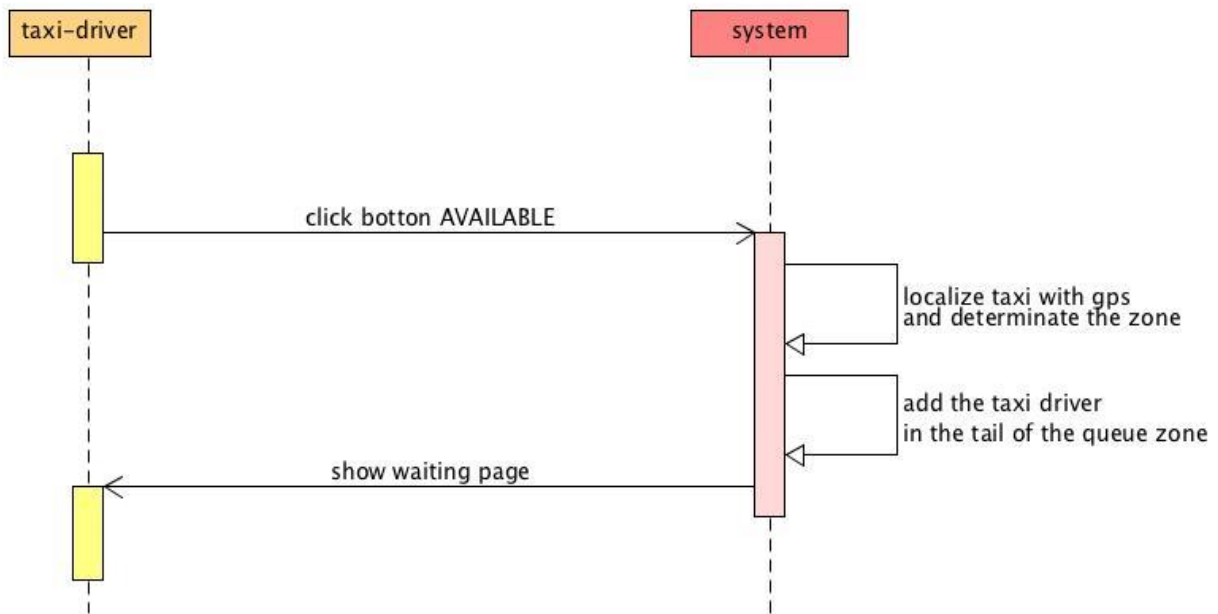
- Resume



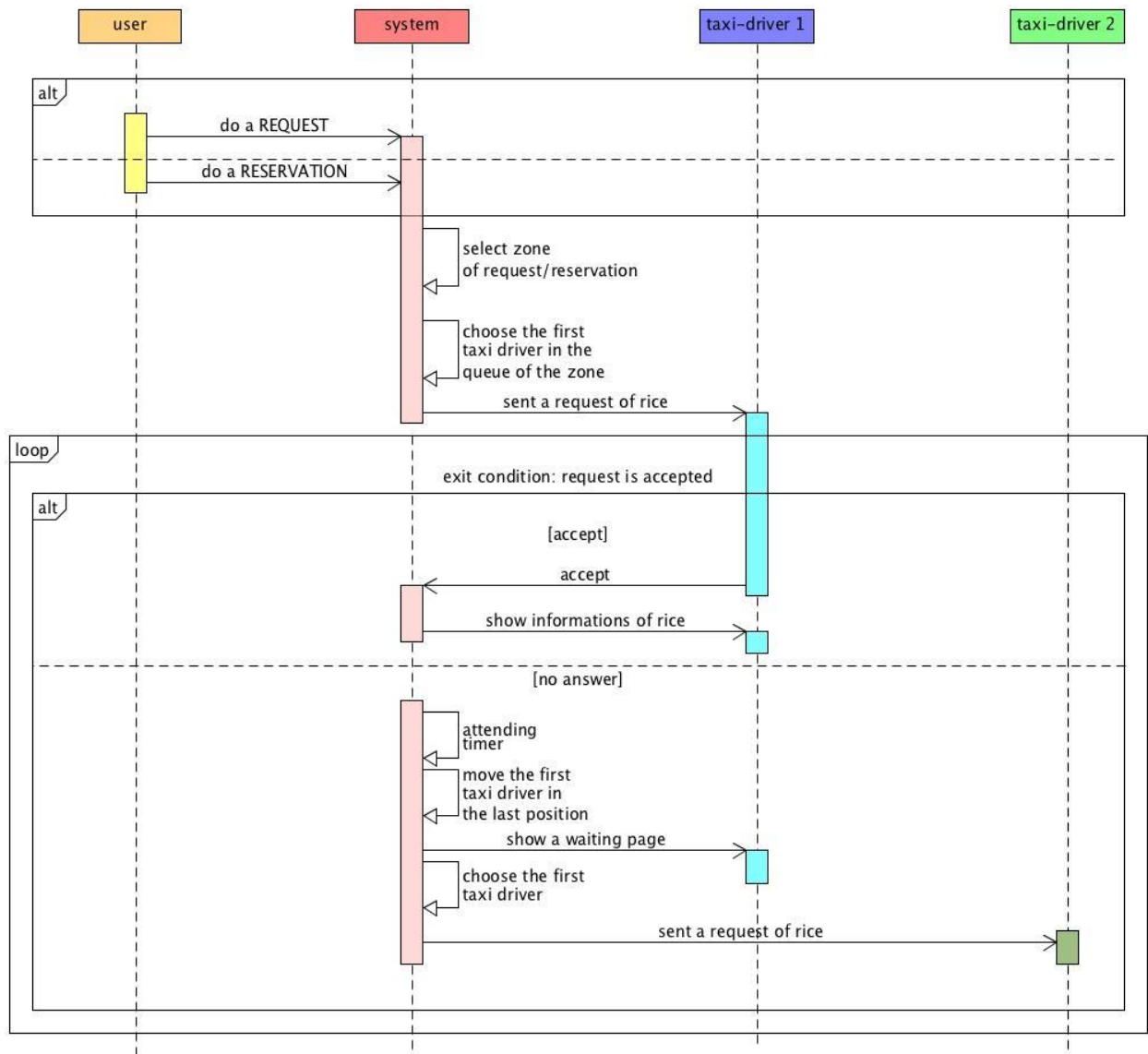
- See the answer



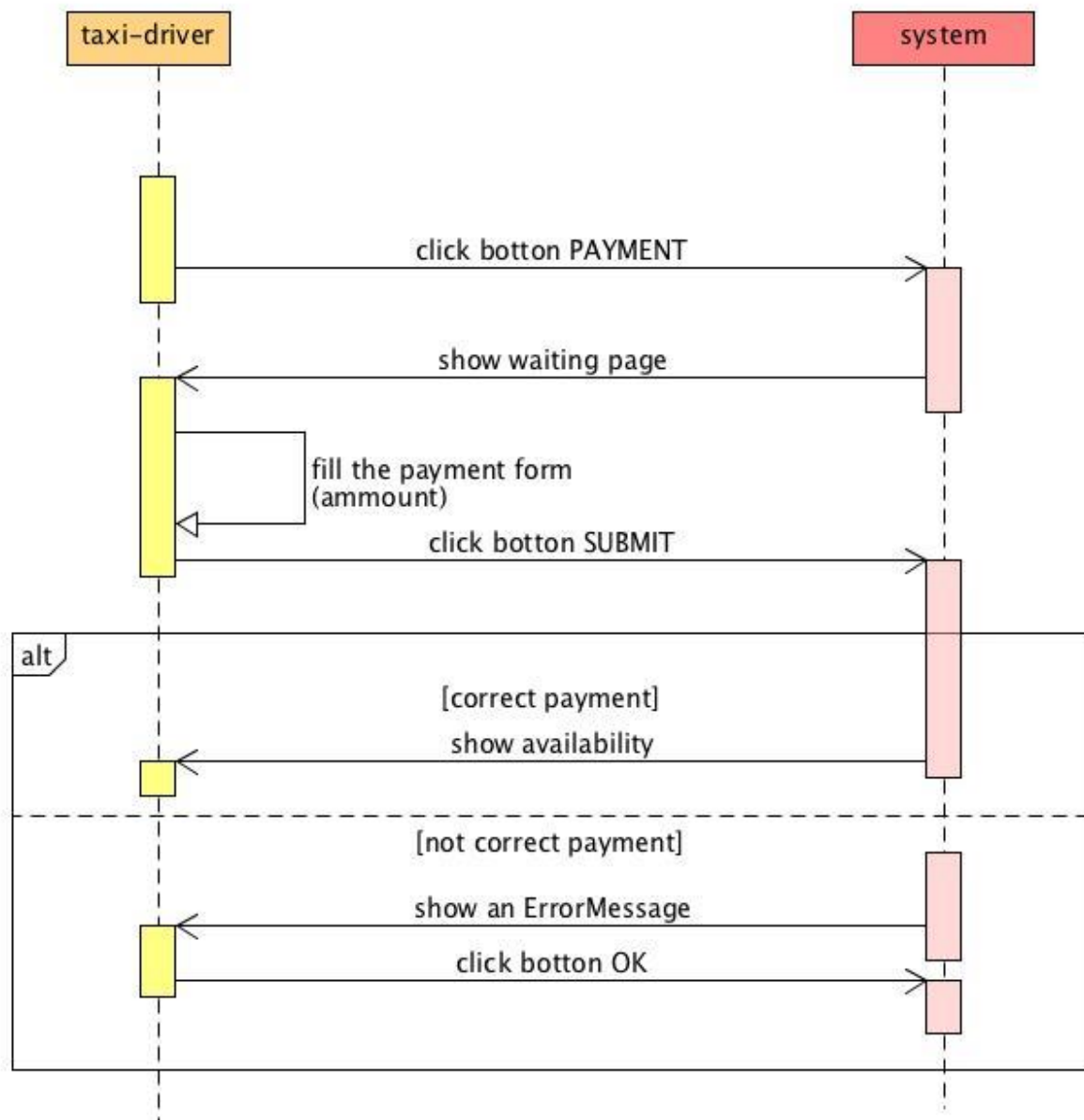
- Availability



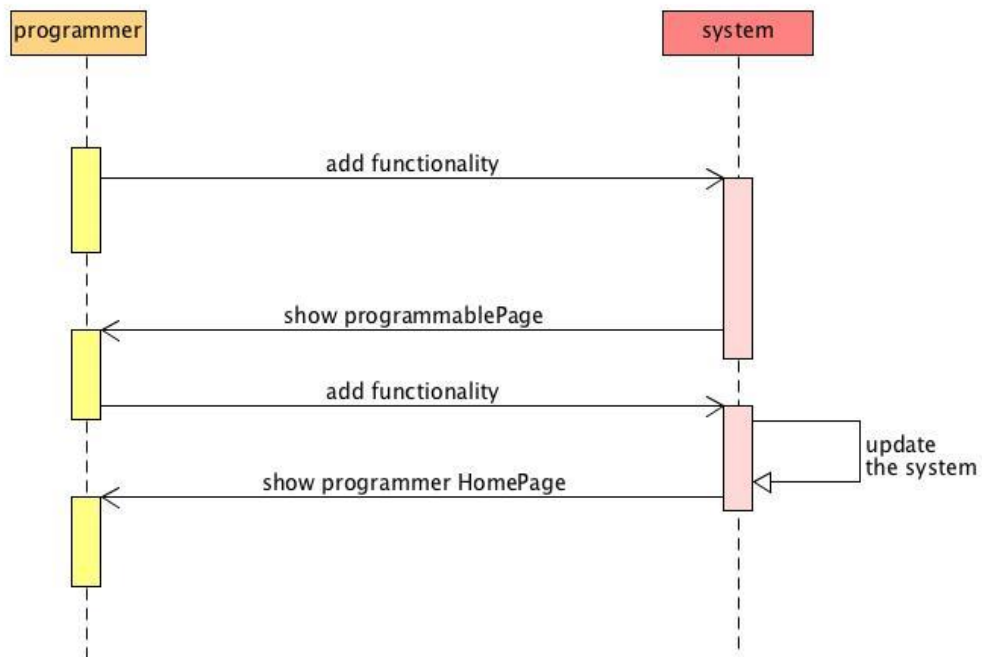
- Accept transport



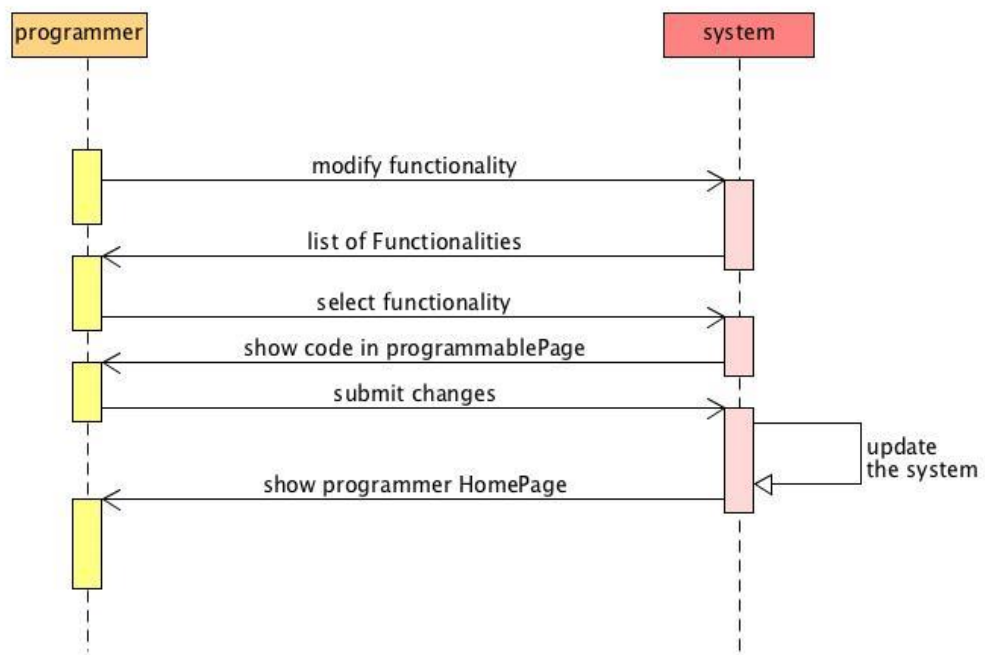
- Request payment



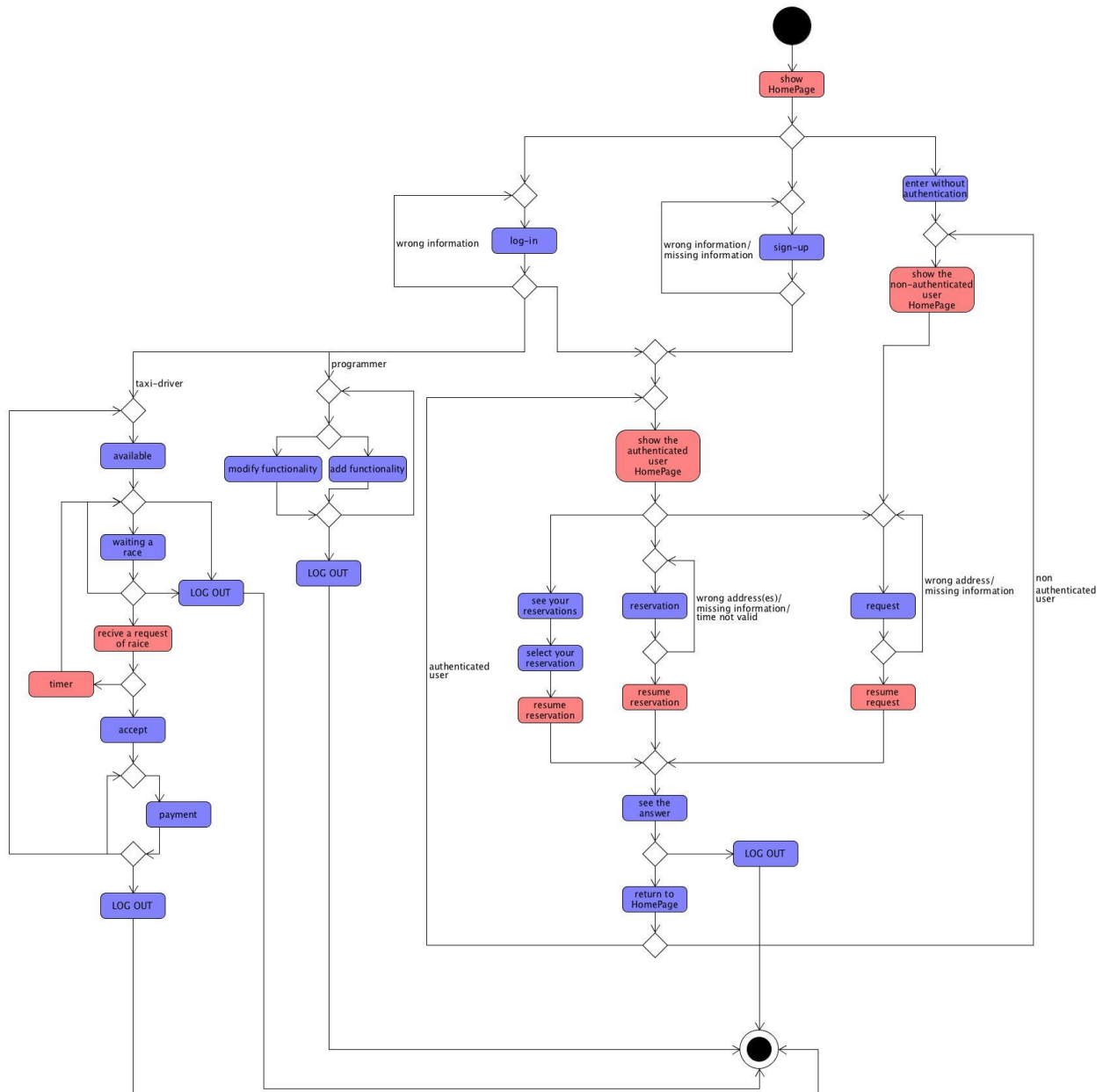
- Implementation of new functionality



- Modify existing functionality



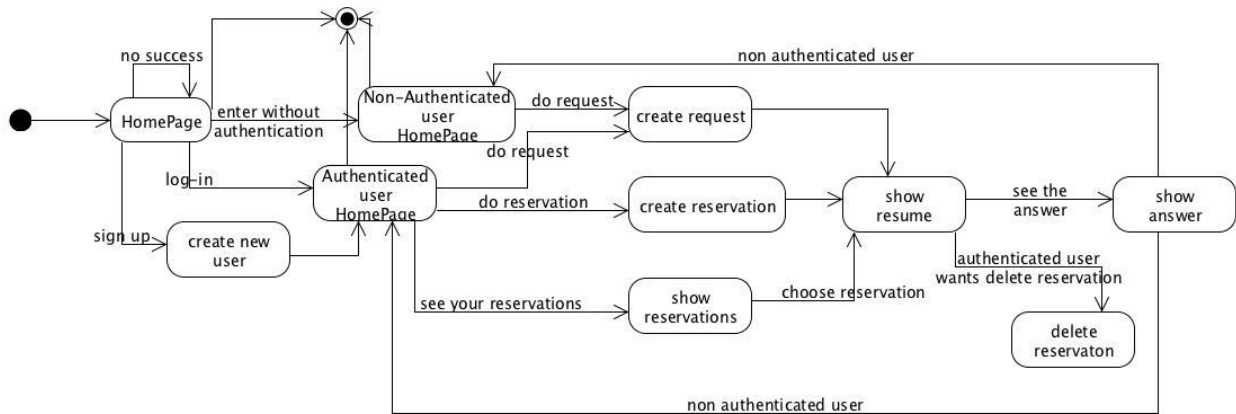
## 5.5 Activity Diagram



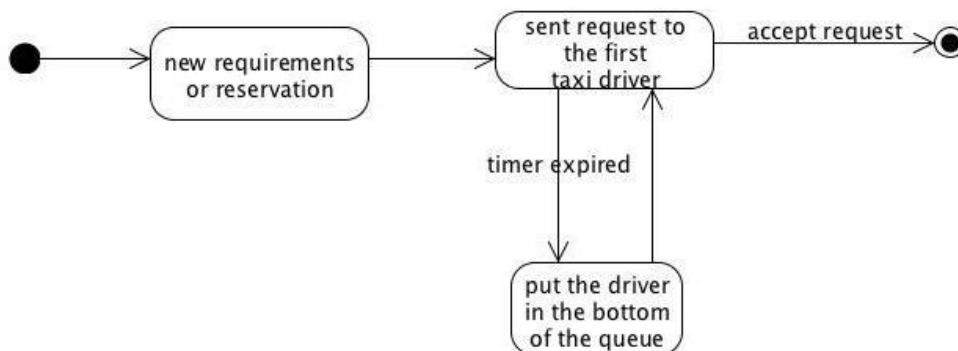
The red activity belong to the system, the blue one to the users

## 5.6 State Chart Diagrams

The following state machine diagram would give a global idea about the way to do a reservation or a request.



The following state machine diagram would give a global idea about the interaction with taxi driver when the system recives a request or reservation.



## 6. ALLOY

### 6.1 Alloy code

```

/*****
/ Photograph of the system made at a specified instant t
/ *****/

//general user: represents both registered and non registered user
//he can have made a request or not and he's in a specified zone
abstract sig User {
    request: lone Request,
    isInZoneU: one Zone
}
sig NonRegisteredUser extends User {}

//a registered user can do one or more reservations of a taxi
sig RegisteredUser extends User {
    reservation: some Reservation
}
//a non registered user must have underway at least a request to "live" in the system
fact {
    no n: NonRegisteredUser | #n.request < 1
}
fact ReverseRelation {
    request = ~byUserReq
    reservation = ~byUserRes
}
// who drives a taxi
sig TaxiDriver {
    drive: one Taxi,
    myState: one TaxiDriverState
}
//taxi drivers have 2 kind of possible states: available or busy
abstract sig TaxiDriverState{
    state: one TaxiDriver
}
fact {
    state = ~myState
}
sig Busy extends TaxiDriverState{}

sig Available extends TaxiDriverState{}

//if a taxi driver has a request from a user, this implies that he's "busy" and can't serve another user
//if there are no requests, all of the taxi drivers are "available"
fact{
    all td: TaxiDriver | (#Request > 0 and #td.drive.isRequestedBy > 0) iff td.myState in Busy
    #Request = 0 implies TaxiDriver.myState = Available
}

//signature of a taxi car
sig Taxi {
    isRequestedBy: lone Request,
    driveBy: one TaxiDriver,
    isInZoneT: one Zone
}
fact {
    taxi = ~isRequestedBy
    drive = ~driveBy
    isInZoneT = ~taxiQueue
}
//particular area in which the city is divided
sig Zone {
    taxiQueue: some Taxi
}
}
```



```

//in all the zones there must always be at least a taxi available (the parameter can change)
fact{
  all z: Zone | !((z.taxiQueue.driveBy.myState & Available) in none)
}

//a request in real time, it combines a user to one or more taxis
sig Request{
  byUserReq: one User,
  taxi: some Taxi,
  startingPoint: one Address
}

//the user and the taxi of a request must come from the same area
fact {
  all x: Request | x.byUserReq.isInZoneU = x.taxi.isInZoneT and x.taxi.isInZoneT = x.startingPoint.inZone
  and x.byUserReq.isInZoneU = x.startingPoint.inZone
}

//booking of a taxi by a registered user at a specified time, from a particular address to another one
sig Reservation{
  byUserRes: one RegisteredUser,
  atTime: Time,
  fromAddr: one Address,
  toAddr: one Address
}

//if a user has, for instance, 2 reservation, they can't be at the same time
//the departure and the arrival of a reservation can't be equal
fact {
  all r1: Reservation, r2: Reservation | ((r1!=r2) and (r1.byUserRes = r2.byUserRes)) implies (r1.atTime != r2.atTime)
  all r: Reservation | r.fromAddr != r.toAddr
}

//departure or arrival of a reservation
sig Address{
  inZone: one Zone,
  depRequest: set Request,
  arrival: set Reservation,
  departure: set Reservation
}
fact{
  depRequest = ~startingPoint
  fromAddr = ~departure
  toAddr = ~arrival
}

//the time schedule of a reservation
sig Time{
  scheduleOfThe: some Reservation
}
fact{
  scheduleOfThe = ~atTime
}

```

```

//assertions
assert UserTypeDisjoint{
  RegisteredUser & NonRegisteredUser = none
}

check UserTypeDisjoint

assert NoRequestIfAvailable{
  all t: TaxiDriver | (t.myState in Available) implies not(Request in t.drive.isRequestedBy)
}

check NoRequestIfAvailable

assert NoReservationForUnregisteredUser{
  all u: NonRegisteredUser, r: Reservation | not (u -> r in request)
}

check NoReservationForUnregisteredUser

assert NoReservationInCommon{
  all r1: RegisteredUser, r2: RegisteredUser | r1.reservation & r2.reservation = none
}

check NoReservationInCommon

assert NoRequestInCommon{
  all r1: User, r2: User | r1.request & r2.request = none
}

check NoRequestInCommon

pred show {}

run show

```

## 6.2 Alloy worlds

On the following pages they will be shown some “alloy world-graph” projected on some real-life examples.

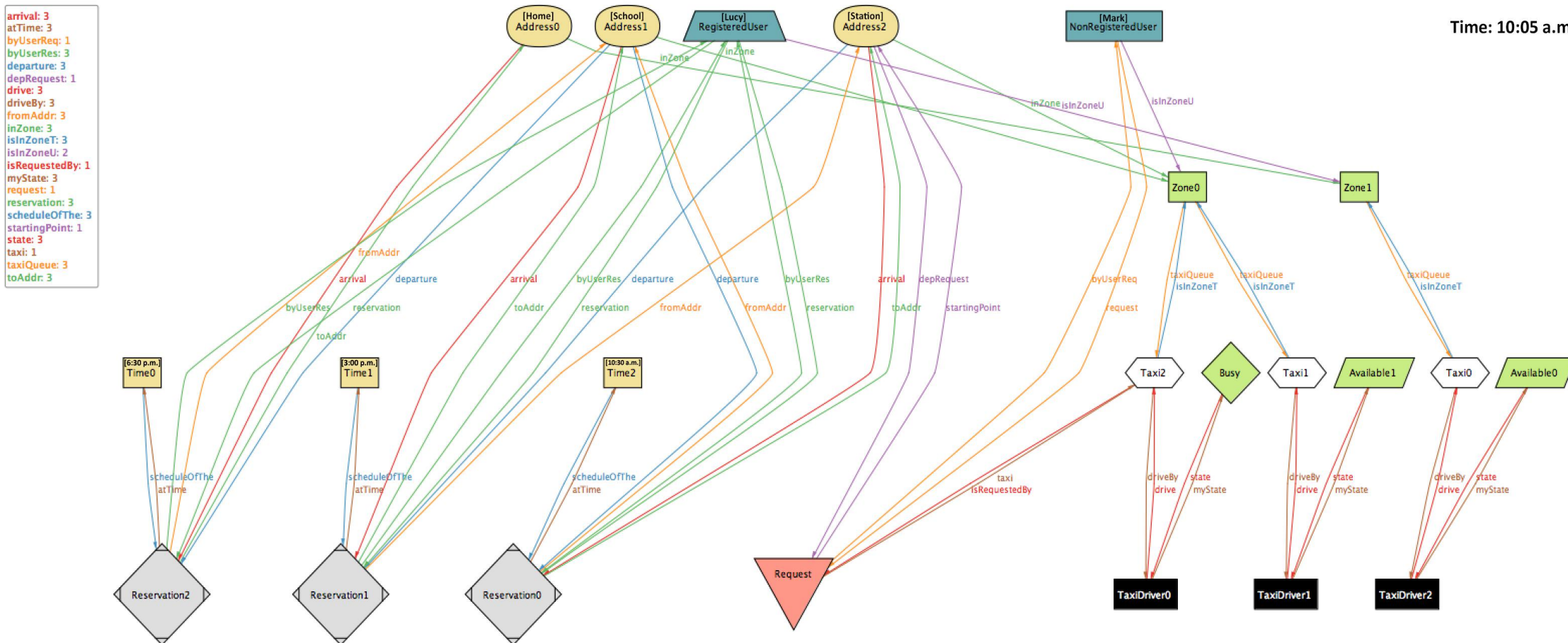
### Example 1

It's 10:05 a.m. Lucy is a professor and now she's at school and she's just finished his lecture. However, Lucy teaches also in another department in a nearby town, and so she has to go to the train station to get the train, which leads her to the other school. In the afternoon she has to return to the first building and finally she moves from school to home. She's subscribed to "MyTaxiService" and she has 3 reservations associated with her profile today:

1. 10:30 a.m. | school -> station
2. 03:00 p.m. | station -> school
3. 06:30 p.m. | school -> home

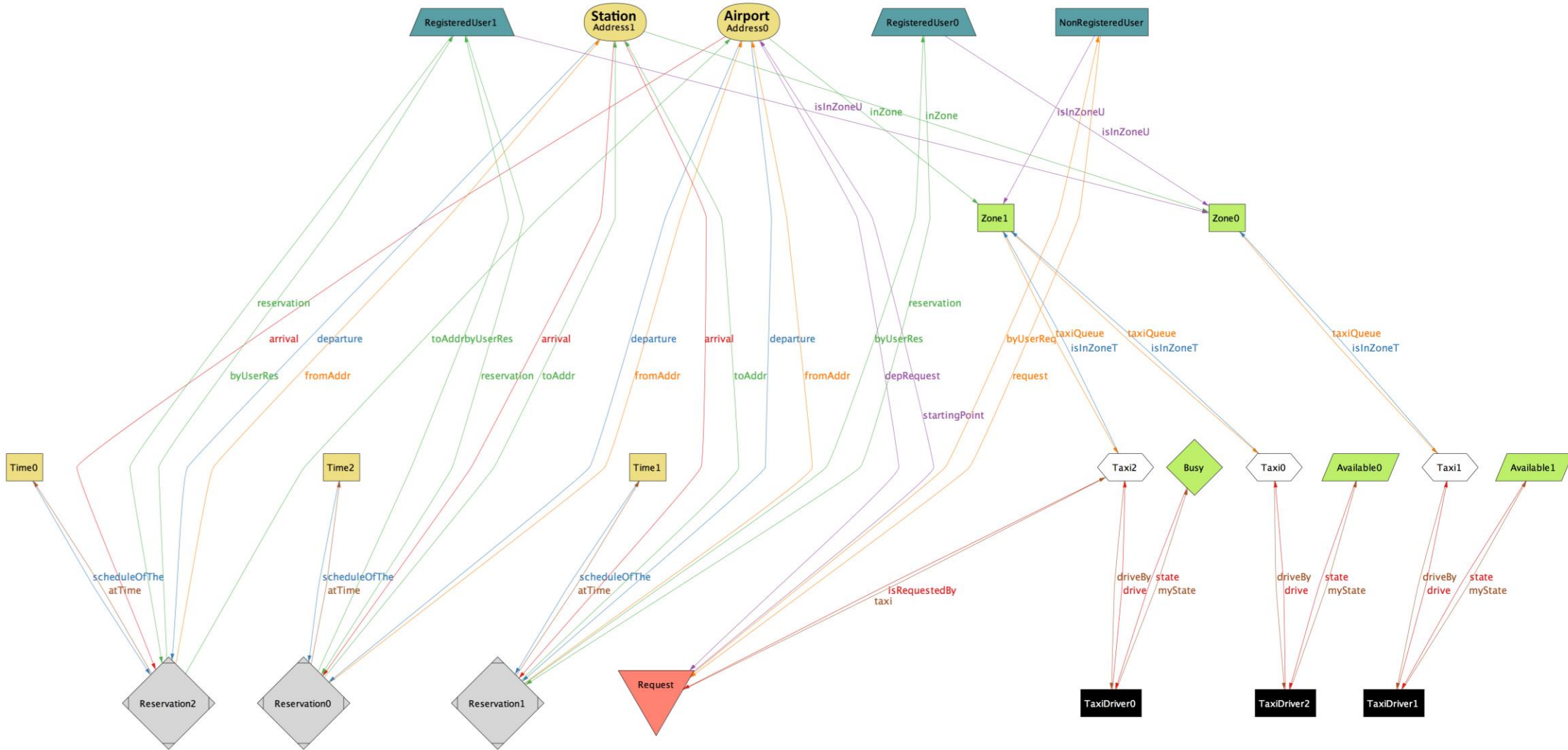
Mark, an important manager, is just arrived at the train station and he has to go quickly to the office. In order to move swiftly, he's requested a taxi with the app, that he had downloaded during the travel.

Time: 10:05 a.m.



## Example 2

Another example in which users move from the airport to the train station and vice versa



Version2:

In this second version we added the functionality "Modify Functionality" for the programmer, so we modified some diagrams.