



POLITECNICO MILANO 1863

Computer Science and Engineering

SOFTWARE ENGINEERING 2

A.A. 2015 / 2016

MyTaxiService

RASD

**Requirements Analysis and Specification
Document**

November 6th, 2015

REFERENCE PROFESSOR:

Mirandola Raffaella

AUTHORS:

Polidori Lucia

Regondi Chiara

Table of contents

1. Introduction	4
1.1 RASD: definition and main features	4
1.2 Requirement Engineering: introduction and definition	4
1.3 MyTaxiService: description of the given problem	5
1.4 Scope	5
1.5 Glossary	5
1.6 Stakeholders identification	6
1.7 Reference documents	7
1.8 Overview	7
2. Overall Description	8
2.1 Goals	8
2.2 Domain properties	8
2.3 Assumptions	9
2.4 The system we propose: current context and future system	9
3. Actors Identification	10
4. Requirements	11
4.1 Functional requirements	13
4.2 Non-functional requirements	13
4.2.1 User interface	13
4.2.2 Documentation	18
4.2.3 Quality of Service attributes	18
4.2.4 Architecture	18
4.3 Pseudo - Requirements (constraints)	19
5. Scenarios Identification	20
6. UML Modeling	26
6.1 Use Cases: identification and description	26
6.1.1 Sign up	27
6.1.2 Log in	28
6.1.3 Recover the password	28
6.1.4 Request a taxi	29
6.1.5 Reserve a taxi	30
6.1.6 Update personal information	31

6.1.7	Visualize the reservations.....	31
6.1.8	Delete a reservation.....	32
6.1.9	Receive a request confirmation notification	32
6.1.10	Receive a reservation confirmation notification.....	33
6.1.11	Send notifications about taxi's availability.....	33
6.1.12	Receive a notification of a request or a reservation.....	33
6.1.13	Send notifications about accepted requests	34
6.2	Class Diagram	35
6.3	Sequence Diagrams	36
6.3.1	Sign up	36
6.3.2	Log in	37
6.3.3	Recover the password	38
6.3.4	Request a taxi	39
6.3.5	Reserve a taxi	40
6.3.6	Update personal information (modify).....	41
6.3.7	Visualize the reservations.....	42
6.3.8	Delete a reservation.....	43
6.3.9	Receive a request confirmation notification	44
6.3.10	Receive a reservation confirmation notification.....	44
6.3.11	Send notifications about taxi's availability.....	45
6.3.12	Receive a notification of a request or a reservation.....	45
6.3.13	Send notifications about accepted requests	46
7.	Alloy Modeling	47
7.1	Introduction to Alloy.....	47
7.2	Alloy code.....	49
7.3	Result of analyzer.....	53
7.4	Generated world	54
8.	Used Tools	55
9.	Working Hours	55

1 Introduction

1.1 RASD: definition and main features

The Requirements Analysis and Specification Document (RASD) is a document containing the description of the requirements and specifications analysis phase of a software development process. In this document, we are going to illustrate our web application, called “myTaxiService”.

The publication is produced in strict compliance with the standard IEEE 830-1998 (Recommended Practice for Software Requirements Specifications).

We write this document in order to help you understand the requirements of the system, explaining both the application domain and the system that will be realized.

This document represents the basis for the project planning and it's useful to estimate its duration and its cost.

It is also used to carry out test activities, verification and validation and to monitor requirements changes and, consequently, the evolution of the software. Therefore we must not think of it as a static document, but as a document that evolves and improves until it reaches its final form.

1.2 Requirement Engineering: introduction and definition

The Requirement Engineering is the process of discovering the degree to which the software meets the purpose for which it was intended, by identifying stakeholders and their needs and by documenting these needs in a form that is available for analysis, communication and implementation. So, in other words, the Requirement Engineering is a process whose goal is to identify and communicate the purpose and the requirements of a software system.

In this document, we will follow the Jackson and Zave approach, “The World and The Machine”, in order to carry out our reasoning process and identify goals, domain assumption and requirements of our application.

According to this approach, the system that has to be developed (software-to-be) is the “machine”, while the portion of the real world which is affected by the machine is the “world”.

So, Jackson and Zave distinguish between “world phenomena”, which represent all those phenomena (events) occurring in the world and that the machine cannot observed, and “machine phenomena”, which represent all those phenomena entirely located inside the machine.

However the intersection between this two sets of phenomena is not empty: there are some “shared phenomena”, that the world shares with the machine. The Requirement Engineering must be able to identify the shared phenomena, that represent the interface between the world and the machine, identifying all the features the system should have in order to produce benefits for the external world.

Analyzing this sets of phenomena, we can define the goals of our system, the properties of the application domain and the requirements.

1.3 MyTaxiService: description of the given problem

We will design the so-called “myTaxiService”, an online application for requesting or reserving a taxi in a large city.

The system has to allow the registration of new users and allow them to request or reserve a taxi, specifying the origin and the destination of the ride.

The system will answer to the request by informing the passenger about the code of the incoming taxi and the waiting time, while it will answer to a reservation confirming it through a notification.

The system has to guarantee a fair management of taxi queues: it automatically computes the distribution of taxis in the various zones based on the GPS information it receives from each taxi.

In this way, when a request arrives from a certain zone, the system has to forward it to the first taxi queuing in that zone. If the taxi confirms, the system will send a notification to the passenger, otherwise the system has to forward the request to the second in the queue.

The system will receive the information about the availability through a mobile application used by the taxi drivers to communicate with the system itself.

1.4 Scope

The software-to-be will be developed as a web based application that will let the user request of reserve a taxi for a specific ride, with notifications about the availability and the code of the incoming taxi, in addition to the waiting time.

1.5 Glossary

In this section we define some words that we will use in the documentation of the project.

Application: a software made to implement one or more services/activities.

Application domain: set of phenomena and observable properties, typical of the environment in which the software is expected to produce its effects.

Authentication: the act of inserting username and password correctly, in order to complete the log in and access the system.

Code of a taxi: it's the identification number of a particular taxi. Every taxi has a different code.

E-mail notification: a notification sent as an e-mail to a user.

Error message: a notification displayed on the screen used in order to inform the user of an error.

Goal: aim/purpose/utility that has to be reached through the software.

Guest: an unregistered user, or, more in generally, someone that is visualizing the site but has not done the log in yet.

Notification: a notification is something, such as a highlighted box in the system or a specific e-mail received, through which notice is given.

Password: a string that only the registered user knows and that is necessary for the authentication.

Personal profile page: the registered user's private page, where he can see his notifications, choose whether to request or reserve a taxi and click on specific links in order to visualize his personal information and the list of all his reservations.

Registered user: an user that is already registered to the system and is allowed to do the log in.

Request: the act of call a taxi through the web application. After a request, a taxi will be allocated and will arrived at the required location as soon as possible.

Requirement: a requirement express a desire of the costumer concerning the application. Every requirement has to be satisfied in order to reach all the goals.

Reservation: the act of book a taxi from a point A to a point B, on day D, at the time X.

Taxi-driver: the person who drives the taxi and use the application that is installed on board.

User: see "Registered user".

Username: a string, chosen by the registered user, to identify himself and that is necessary, together with his password, for the authentication.

Web application: an application accessible through the web using a network (such as an intranet or the Internet).

1.6 Stakeholders identification

Our "financial" stakeholder is the professor who assigned us this project.

We have to deliver this project to the professor within the end of January 2016, showing that we have understood the requirements analysis and design phases of the software development process.

The government of city X "virtually" gave us this project in order to simplify the access of passengers to the taxi service and guarantee an efficient management of taxi queues.

The government needs to have a product that works fine and respects all the specification given to us.

We think that our typical users can be:

- occasional users that sometimes can use this application to request a taxi when they need it;
- habitual users that need regularly a taxi and that can exploit the registration in order to reserve a taxi for a specific ride.

However, in order to guarantee a better management of the application, the registration is required for every user: a guest can only navigate through the homepage or decide to sign up. In order to make a request or a reservation both occasional and habitual users should sign up and create an account.

1.7 Reference documents

We wrote this document following the structure and the subdivision into paragraphs of the standard IEEE-STD-830-1993 and drawing from old documents of previous projects.

1.8 Overview

This document is structured into nine main parts:

- Section 1: Introduction
this section contains an explanation of the document, a description of the problem that will be faced and some other general information about the software.
- Section 2: Overall Description
this section gives information about the goals and the domain properties of the application, in addition to a brief description of the actual system and the software-to-be.
- Section 3: Actor Identification
this section gives information about all the actors involved and that will be using our software.
- Section 4: Requirements
this section contains all the specific requirements of the software (functional, non-functional, constraints).
- Section 5: Scenarios Identification
this section contains the description of some of the typical scenarios that can be identified.
- Section 6: UML Modeling
in this section the main use cases are identified; moreover, in order to better understand all the functionalities of the software, UML diagrams are here provided, modeling all the specific requirements listed before.
- Section 7: Alloy Modeling
this section contains a simple introduction to the Alloy notation and the alloy modeling of the previous requirements.
- Section 8: Used Tools
in this section, all the tools we have used in order to develop this document are listed.
- Section 9: Working Hours
this sections contains the result of our effort, quantified in the number of hours we have needed in order to develop this document.

2 Overall Description

2.1 Goals

For our system it's possible to identify the following minimum goals:

- [G1]: for every request, a taxi should arrive in the correct zone of the city;
- [G2]: for every reservation, a taxi should arrive in the correct zone at the time established;
- [G3]: all registered users must be able to call a taxi, since the time of registration;
- [G4]: all registered users must be able to book a taxi from point A to point B, at the time X and for a specific date D;
- [G5]: all registered users must be able to receive a notification containing the code and the waiting time when they require a taxi;
- [G6]: all registered users must be able to receive a notification including the confirmation of the reservation when they book a taxi;
- [G7]: all unregistered users have to be able to register to the application;
- [G8]: all taxi drivers must be able to notify the system with their availability;
- [G9]: all taxi drivers must be able to notify the system saying that he's busy because he's taking care of another request.

2.2 Domain properties

We suppose that the following conditions hold in the world we analyze:

- for each call, the system requires some information: the location, the number of passengers (from 1 to 4) and the name of the person who does the request;
- for each reservation, the system requires some information: the origin and the destination of the ride, the time and the date for the reservation, the number of passengers (from 1 to 4) and the name of the person who does the reservation;
- when a reservation is done, the system allocated a taxi 10 minutes before the meeting time with the passenger;
- every reservation has to be done at least 2 hours before the ride;
- when a taxi is mobilized, it will reach the site of the request as quickly as possible;
- accurate taxi locations are known by the GPS signal of each taxi;
- the city is divided into zones and each zone is associated with a taxi queue;
Each taxi is queued in his area according to the position obtained by the GPS;
- when a taxi is busy, it is placed at the bottom of his queue;
- availability of taxis are known thanks to a mobile app owned by each driver.

2.3 Assumptions

There are few points that could result not clear in the specification document, so we will have to assume some facts:

- ◆ only users who have already registered to the application can exploit the request and reservation functionalities;
- ◆ a registered user cannot cancel the immediate request for a taxi;
- ◆ a registered user can cancel a reservation up to 30 minutes before the reservation time;
- ◆ when a request from a certain area is received, the system checks the taxi queue of that area, and sends a request to the first taxi in the list. If the taxi confirm its availability, then the system sends the confirmation to the passenger, otherwise the taxi is moved at the end of the queue;
- ◆ there must be always a taxi in every queue;
- ◆ when the taxi driver accepts a call, he changes his status from "available" to "occupied";
- ◆ when a taxi is allocated in order to take care of a reservation, the taxi driver changes his status to "occupied";
- ◆ when the taxi driver finishes a race, he changes his status from "occupied" to "available";
- ◆ the app used by the drivers is installed on board, on a mobile data terminal placed in each taxi;
- ◆ each driver has his own taxi, so he's identified by the code of his taxi;
- ◆ if a passenger doesn't show up at the meeting with the taxi he requested or reserved, he will receive an e-mail notification with a penalty fee that he has to pay within 2 months;
- ◆ if a passenger doesn't pay the taxi after the ride, he will receive an e-mail notification with a penalty fee that he has to pay within 2 months.

2.4 The system we propose: current context and future system

MyTaxiService is a completely new software. Currently, there are no applications of this kind, so our project is not an evolution of something already existing, but it's something new, built on the specific requests of our customer.

Currently, the taxi service of this city can be accessed only by the telephone and the management of the taxi queues in the different zones of the city turns out to be quite complex with the growth of the requests.

We propose a web based platform that allows a registered user to call a taxi or reserve one for a specific ride at a particular time. The system will confirm both the requests and the reservations by sending a notification to the passenger; in case of a taxi request, the passenger will also receive the code of the incoming taxi and the expected waiting time.

The system allow you to delete only the reservations and not the immediate requests.

Together with the web application, our system will provide a mobile app for the taxi drivers, so that they can communicate their availability or their tasks to the system, in order to allow a better management of the taxi queues and of all the requests or reservations received.

3 Actors Identification

GUEST :

he can visit the homepage and register into the system, creating a personal account, in order to become a registered user.

REGISTERED USER :

after successful login, he can access to his personal profile, modify his personal information and immediately request or reserve a taxi filling out the dedicated forms.

He can receive notifications about the incoming taxi, visualize his past and current reservations and eventually decide to delete one of the current ones.

TAXI DRIVER :

he is already pre-registered into the mobile application installed on his taxi and can use it to answer to specific requests forwarded by the system, communicate his availability and eventual calls he's taking care of.

4 Requirements

According to the model of The World and The Machine presented by Jackson and Zave, assuming that all the domain properties and the assumptions we have defined in the previous chapter hold in the world, we have identified the requirements that satisfy the goals described earlier in the document.

These requirements are the main features characterizing our system:

♦ **Registration**

- the system should provide a “sign up” functionality;
- when a new registration is done, the system should create a new profile in the passengers database and allow the new user to log in;
- the system should store user information including name, surname, e-mail, mobile phone number, credit card, username and password.

♦ **Log In**

- the system should provide a “log in” functionality to access the application;
- the system should grant the access to the application only after a user types a correct username and password;
- when the log in is corrected, the system should show the user his personal page;
- the system should show an error message whenever the authentication is wrong.

♦ **Insert, edit, delete personal information**

- the system has to allow the user to insert new personal data inside his profile information;
- the system has to allow the user to modify existing personal information;
- the system has to allow the user to delete existing personal information.

♦ **Request a taxi**

- the system should provide a “request” functionality, allowing a passenger to request a taxi from 6:00 a.m. to 9:00 p.m.;
- the system has to forward each request to the first taxi situated in the requested zone, in order to check its availability, and then mobilize the first available taxi in the queue of that zone;
- the system should move a taxi to the bottom of the queue if not available for one of the current requests;

♦ **Reserve a taxi**

- the system should provide a “reserve” functionality in order to allow a passenger to reserve a taxi (from 6:00 a.m. to 9:00 p.m.) for a specific ride;
- when a reservation is received at least 2 hours before the ride, the system confirms it to the passenger;
- when a reservation is not done at least 2 hours in advanced, the system should show an error message;
- the system should allocate for every reservation one of the available taxis 10 minutes before the established meeting time;

- the system should provide a function that allows a user to visualize the list of reservations he has done.

♦ **Drivers communication**

- the system shall be able to receive communication and information about availability from each taxi driver;
- the system should store all the information about each taxi's availability;
- when a request from the system is received, a taxi driver should communicate its availability to the system itself;
- when a taxi driver decides to take care of certain calls, he should inform the system.

♦ **Localization**

- the system has to be able to accurately localize the distribution of the taxis in each zone using a GPS locator.

♦ **Notifications**

- the system has to notify a user, confirming his request, when a taxi is available;
- after confirming a request, the system shall communicate to the passenger the code of the incoming taxi and the estimated waiting time;
- the system should confirm a successful reservation sending a notification to the passenger.

♦ **Payments**

- the system should check the payments of all the passengers every month;
- the system should send an e-mail notification to inform all those users who doesn't pay the taxi requested or reserved, or who doesn't show up, that the payment has not been accomplished: they will have to pay a penalty fee within 2 months.
- the system should send an e-mail to all those users who has already paid, confirming the payments.

♦ **Other services development**

- the software has to be developed in a way that can allow future extensions: the system should offer an API interface that allow future developers to extend the application's functionalities, implementing new services.

4.1 Functional requirements

After defining in the previous paragraph the main features of MyTaxiService, we can now identify some functional requirements for each actor involved:

- ★ Guest: he can
 - sign up

- ★ Registered user: he can
 - log in
 - modify his profile information
 - request a taxi filling out the related form
 - reserve a taxi filling out the related form
 - receive a taxi request confirmation
 - receive information about the code of the incoming taxi and the waiting time
 - receive a notification confirming a reservation
 - visualize all his previous and current reservations

- ★ Taxi driver: he can
 - send information about his availability
 - send information about the calls he takes care of
 - receive information about the requests and the reservations he has to satisfy

4.2 Non-functional requirements

The non-functional requirements describe all those aspects of the system not directly related to its behavior and its functionalities.

4.2.1 User interface

The interface of our passengers application is thought to be used via web, while the interface for the taxi drivers is offered through a mobile application installed on board on a data terminal.

We want the web application to be a minimal and user friendly interface, so that every passenger can be able to use it: buttons to come back to the homepage or to the personal profile page and to logout are always visible.

Here are now presented some mockups that represent our idea of the structure of the most important application pages.

Homepage

The mockup below shows the homepage of MyTaxiService: here users can decide to log in, if they're already registered, or sign up, in order to create a new account.

The mockup shows a web browser window with the title 'My Taxi Service' and the URL 'http://mytaxiservice.com'. The page features a black and white checkered border at the top. The main heading is 'My Taxi Service' with a taxi icon. Below the heading is a 'MENU' button and a language selector set to 'English'. The central text reads 'YOUR RIDE... WHENEVER , WHEREVER' followed by the text 'Request or reserve your taxi from 6:00 a.m. to 9:00 p.m.'. There are two buttons: 'Log In' and 'Sign Up NOW!'. At the bottom, there is a link to 'Info'.

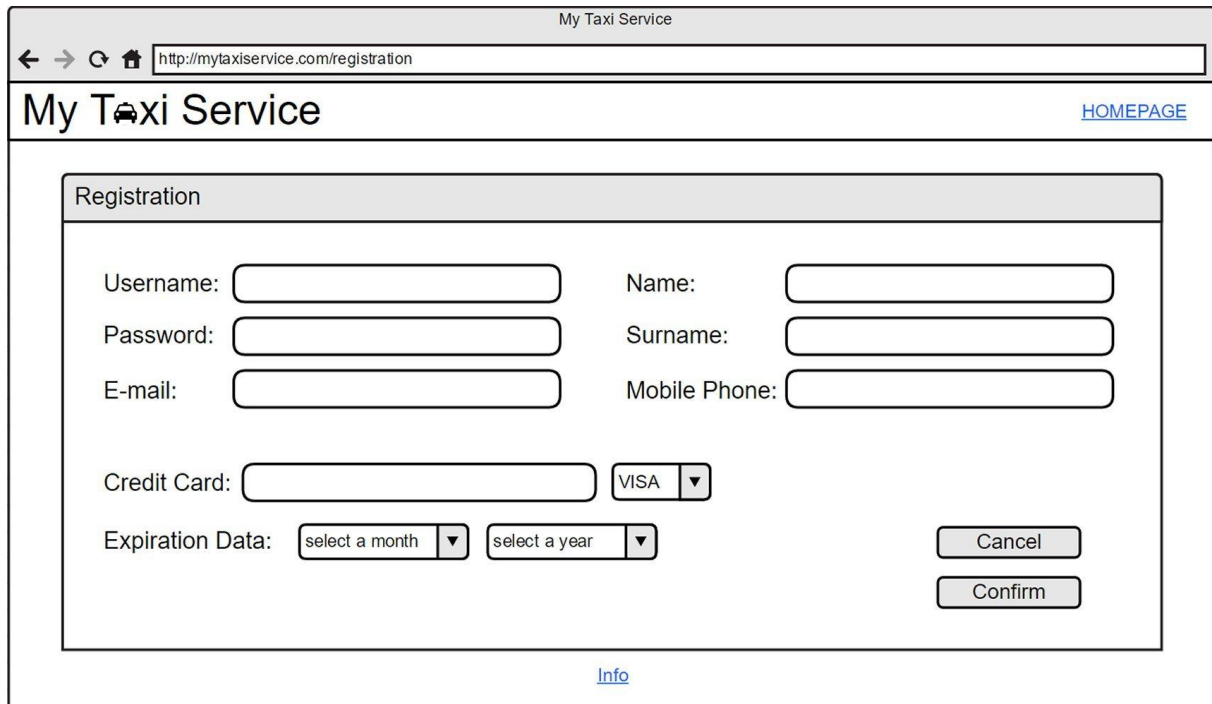
Log In

The mockup below shows the log in page: here users can access the system filling out the form with their credentials.

The mockup shows a web browser window with the title 'My Taxi Service' and the URL 'http://mytaxiservice.com/login'. The page features a black and white checkered border at the top. The main heading is 'My Taxi Service' with a taxi icon. Below the heading are links to 'HOMEPAGE' and 'Sign Up'. The central form is titled 'Log in' and contains fields for 'Username:' and 'Password:'. There is a link 'I forgot my password' below the password field. A 'Confirm' button is at the bottom of the form. At the bottom of the page, there is a link to 'Info'.

Registration

The mockup below shows the registration form page: a guest has to fill out the form in order to become a registered user and create his personal account.

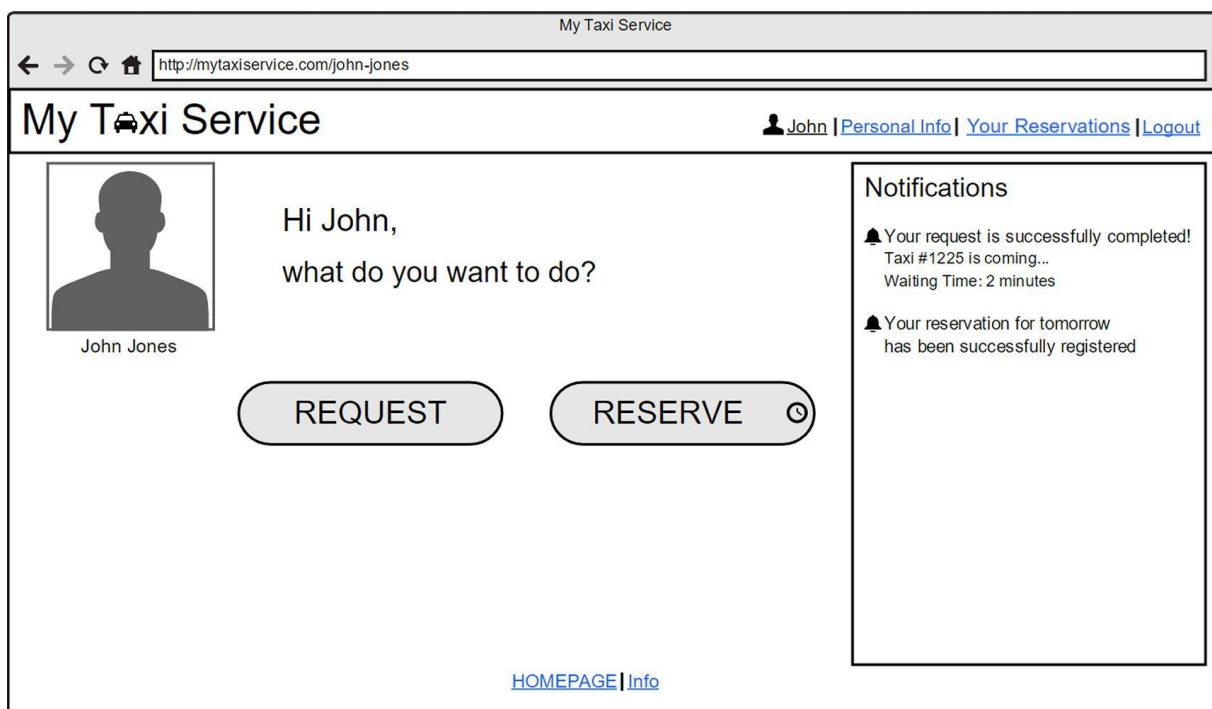


The mockup shows a web browser window titled "My Taxi Service" with the URL "http://mytaxiservice.com/registration". The page header includes the site name and a "HOMEPAGE" link. The main content area is titled "Registration" and contains a form with the following fields: Username, Password, E-mail, Name, Surname, Mobile Phone, Credit Card (with a dropdown menu showing "VISA"), and Expiration Date (with dropdowns for "select a month" and "select a year"). There are "Cancel" and "Confirm" buttons at the bottom right of the form. An "Info" link is located below the form.

Personal Profile Page

The mockup below shows the profile page of a registered user: here he can decide to request or reserve a taxi and see the notifications send to him by the system.

Eventually he can also skip to the list of his reservations or to his personal information page, clicking on the specific button.



The mockup shows a web browser window titled "My Taxi Service" with the URL "http://mytaxiservice.com/john-jones". The page header includes the site name and navigation links: "John", "Personal Info", "Your Reservations", and "Logout". The main content area is divided into three sections: a user profile section on the left with a silhouette icon and the name "John Jones"; a central section with the text "Hi John, what do you want to do?" and two buttons labeled "REQUEST" and "RESERVE"; and a "Notifications" section on the right. The notifications section contains two messages: "Your request is successfully completed! Taxi #1225 is coming... Waiting Time: 2 minutes" and "Your reservation for tomorrow has been successfully registered". At the bottom of the page, there are links for "HOMEPAGE" and "Info".

Taxi Request

The mockup below shows the request form page: the user can fill the form in order to send an immediate request for a taxi.

The mockup shows a web browser window titled "My Taxi Service" with the URL "http://mytaxiservice.com/john-jones/request". The page header includes the site name "My Taxi Service" and navigation links: "John", "Personal Info", "Your Reservations", and "Logout". The main heading is "Request you taxi now!". Below it is a form titled "Request" with the following fields: "Name" (text input), "Surname" (text input), "Number of passengers" (dropdown menu with "1" selected), and "Location" (text input with a location pin icon). A "Send" button is at the bottom of the form. At the bottom of the page are links for "HOMEPAGE" and "Info".

Taxi Reservation

The mockup below shows the reservation form page: the user can fill the form in order to reserve a taxi for a specific ride.

The mockup shows a web browser window titled "My Taxi Service" with the URL "http://mytaxiservice.com/john-jones/reservation". The page header includes the site name "My Taxi Service" and navigation links: "John", "Personal Info", "Your Reservations", and "Logout". The main heading is "Reservation". Below it is a form titled "Reservation" with the following fields: "Name" (text input), "Surname" (text input), "Date" (three dropdown menus for "day", "month", and "year"), "Time" (two dropdown menus for "hour" and "min"), "Number of passengers" (dropdown menu with "1" selected), "Origin" (text input with a location pin icon), and "Destination" (text input with a location pin icon). A "Send" button is at the bottom of the form. At the bottom of the page are links for "HOMEPAGE" and "Info".

Personal Information

The mockup below shows the profile information of a specific user: the user can simply visualize his information, or, eventually, decide to modify it.

The mockup shows a web browser window with the title 'My Taxi Service'. The address bar displays 'http://mytaxiservice.com/john-jones/information'. The page header includes the service name and navigation links: 'John | Personal Info | Your Reservations | Logout'. The main content area displays the user's profile information in a form-like structure:

- Name: John
- Surname: Jones
- E-mail: john.jones@gmail.com [Edit]
- Mobile Phone: +393391234567 [Edit]
- Credit Card: VISA, ****1111 | Expiration Date: 01/18 | [Edit] [Delete] [Add]
- Username: Jhonny46 [Edit]
- Password: ***** [Edit]

A 'Save changes' button is located to the right of the form. At the bottom, there are links for 'HOMEPAGE' and 'Info'.

Taxi Driver App Interface

The mockup below shows the interface of each taxi application installed on board: the taxi driver can communicate his availability through the dedicated buttons, read the notifications received by the system and accept a specific request.

The mockup shows a tablet interface for a taxi driver. The top status bar displays 'Taxi #1225' and the time '1:31 PM'. The main interface includes:

- Two buttons: 'Available' and 'Occupied'.
- A 'GPS' section with a map showing the current location.
- A 'Notifications' section with a list of incoming requests. The first notification is 'NEW REQUEST: Leonardo Da Vinci Square' with an 'Accept' button and a checkmark.

4.2.2 Documentation

We decide to produce and release the following documents in order to better explain our project and all the development phases:

- ✓ *RASD (Requirements Analysis and Specification Document)*, in order to understand better and explain efficiently our project, defining all the goals, domain properties and assumptions, describing all the main features of the systems and modeling all the requirements and specifications through specific notations.
- ✓ *DD (Design Document)*, in order to define the real structure, the architecture and eventual design constraints of our application.
- ✓ *Function Points Document*, in order to assess the effort and the cost required for the development of our own application

4.2.3 Quality of Service attributes

- **Usability:** the web application is addressed to different varieties of people, with different ages and different skills and level of knowledge concerning informatics.
The web application has to be user-friendly, minimal and quite intuitive, in order to be used without particular problems by every kind of user. In order to achieve this goal, not too much functionalities have to be developed and the user interface should be simple and intuitive.
- **Performance:** the application has to be able to manage a high number of users connected at the same time and more than one request or reservation done simultaneously.
The response time (confirmation and communication about the details of the incoming taxi) to a specific request has to be reasonable, within at most 3 minutes since the request.
- **Accessibility:** the application is always on; however, request and reservation services should be available every day, only from 6:00 a.m. to 9:00 p.m.;
- **Reliability:** the system should be reliable and it has to be able not to lose passengers data contained in the database, even in case of breakdowns or damages. The system should do a daily backup of all the available data in order to grant reliability.
- **Security:** every user can access only to the functionalities he is allowed to. For this reason an authentication mechanism exploiting access credentials is needed.

4.2.4 Architecture

According to the requirements specified above, as for the architecture, the system will need one or more dedicated servers and a database to store all the user's information.

Eventually, another persistent and stable memory system will be added in order to allow the backup of all the data and increase the system's reliability.

An internet connection is required to access the application.

4.3 Pseudo-requirements (constraints)

The so-called “pseudo-requirements” simply represents constraints on the system imposed by the client or the environment in which the system operates.

MyTaxiService doesn't have particular constraints concerning interfaces or specific operations for the management of the system.

However it is subjected to constraints concerning the delivery and the tools needed in order to use the software.

As for the tools, it's necessary for the user to have an internet connection and to use a web browser that allows the navigation. Also the taxi drivers has to have installed on board a mobile data terminal which allows the installation of the mobile app they should use to communicate with the system.

As for the delivery, the project and all the related documents have to be completed within a fixed period of time.

Here are the deadlines we have to respect:

- October 14th, 2015: *Group Registration*
- November 6th, 2015: *Requirements Analysis and Specification Document (RASD)*
- December 4th, 2015: *Design Document*
- January 30th, 2016: *Function Points*

5 Scenarios Identification

We describe here some possible scenarios that can happen: we will use these scenarios in the following chapter, in order to model our system using UML diagrams.

Name	Registration
Participating Actor	Guest: Alice
Flow of Events	<p>Alice has heard about a new service: "myTaxiService". So, she decides to sign up, because she thinks that this service can be useful for her and that she will probably use it the next week. She accesses the site and she clicks on the "Sign Up NOW!" button. She fills out the form with the information needed for the registration (name, surname, e-mail, mobile phone number, username and password) and she clicks on "Confirm". However she forgot to insert the number of her credit card, so an error message is displayed and Alice is brought back to the page where the registration form is shown. Alice insert all the data again and then she clicks on "Confirm". The system confirms her registration and redirects Alice on her personal profile page.</p> <p>Now she wants to see the list of her reservations, so she clicks on "Your Reservations" at the top of the page: the list is obviously empty because she has just registered to the system.</p>

Name	Password recovery
Participating Actor	Registered user: Claire
Flow of Events	<p>Claire wants to check her taxi reservation for tomorrow, but she is a bit careless and she forgot her password. She accesses the site and she clicks on "Log In". She tries two different passwords but both the times an error message is displayed. So, she clicks on "I forgot my password" and then a window appears. She clicks on "Send a new password" and the system sends her a new password by e-mail. After two minutes she receives the mail with the new password. She tries the log in again: she inserts her username and the new password received and confirms: now she is correctly authenticated.</p>

Name	Request of a taxi
Participating Actor	Registered user: Claire
Flow of Events	<p>Claire needs a taxi to arrive on time to a meeting on the other side of the city, at Bond Street 11. So, she decides to use the web application “myTaxiService”, which she has already registered to some months ago. She accesses the site and clicks on the button “Log In”.</p> <p>She fills out the form with her username and her password and she clicks “Confirm”.</p> <p>After the authentication is completed, she is redirected on her personal profile page, where she can decide to visualize her personal information or the list of all her previous reservations clicking on the appropriated links at the top of the page.</p> <p>Now she clicks on the button “REQUEST” in order to complete her request: she is redirect on a page where the request form is shown. She inserts her name, surname and the other information needed for the request: the number of passengers and her actual position (location). She clicks on “Send” and she waits.</p> <p>After one minute she receives both an e-mail notification and a notification on her personal page with the code of the incoming taxi and the waiting time (5 minutes).</p> <p>After five minutes the taxi arrives and she can reach her meeting on time.</p> <p>At the end of the ride she pays using the credit card added to her profile: after a minute she receives an e-mail confirming her payment.</p>

Name	Reservation of a taxi
Participating Actor	Registered user: Lucy
Flow of Events	<p>Lucy discovers that she has to take a plane on November 12th at 11:00 a.m., but nobody can drive her to the airport.</p> <p>She decides to reserve a taxi that could take her to the airport on time: she accesses the site “myTaxiService”, where she has already registered to some days before, but she has never reserved or requested a taxi yet. In the homepage she clicks on the button “Log In” and she fills out the form writing her username and her password, then she clicks on “Confirm”. The authentication is correct so she is redirected to her personal profile page.</p> <p>She clicks on the button “RESERVE” and fills out the reservation form displayed: she inserts her name (Lucy), surname (Smith), the date (November 12th, 2015), the time of the reservation (8:00 A.M.), the number of passengers (1), the origin (Bond Street,13) and the destination (Airport, St. James Street) of the ride. At the end she clicks on “Confirm”. After 2 minutes she receives both an e-mail notification and a notification on her personal page confirming her reservation.</p> <p>Now she wants to see the list of her reservations, so she clicks on “Your Reservations” at the top of the page: she has only one reservation in the list, the one that she has just done, because this is the first time she reserves a taxi since she has registered to the application.</p>

Name	Reservation of a taxi out of time
Participating Actor	Registered user: John
Flow of Events	<p>John wants to reserve a taxi in order to be sure to arrive on time at the conference that will start in an hour and a half in Old Street 5.</p> <p>He accesses the web page “myTaxiService” and he clicks on the button “Log In”. He authenticates himself and at this point he is redirected to his personal page.</p> <p>He clicks on “RESERVE” and then fills out the reservation form that is shown, writing the information needed: name, surname, the date, the time of the reservation, the number of passengers, the origin and the destination of the ride. Then he clicks on “Confirm”.</p> <p>But in this moment an error message appears. In fact, John forgot that the reservation has to be done at least two hours before the reservation time. So, he comes back to his personal page and decides to call a taxi.</p> <p>He clicks on “REQUEST” and he inserts the information needed for the request: name, surname, number of passengers and his location. At the end he click on “Send”. After two minutes he receives both an e-mail notification and a notification on her personal page with the code of the incoming taxi and the waiting time (8 minutes).</p>

Name	Not allowed reservation
Participating Actor	Registered user: Luke
Flow of Events	<p>Luke wants to reserve a taxi to join his friends tonight, Saturday, October 31st, in a new pub in Ludgate Hill. He decides to reserve it with the application “myTaxiService”, but he forgot that the service isn’t available during the night.</p> <p>He accesses the site and clicks on “Log In”. He inserts username and password and he is redirected on his personal profile page.</p> <p>He clicks on “RESERVE” and fills out the form with the information needed for the reservation: name (Luke), surname (Dames), date (October 31st, 2015), time (10:30 p.m.), number of passengers (1), the origin (St. Jones Square) and the destination (Ludgate Hill) of the ride. Then he clicks on “Send”. However an error message is displayed, because the reservation service is available only until 9:00 p.m.</p> <p>Luke decides to use underground.</p>

Name	Modifying personal information: mobile phone number
Participating Actor	Registered user: Mary
Flow of Events	<p>Mary changed her telephone number two days ago, so she wants to update it in her profile on “myTaxiService”. She accesses the homepage and she clicks on “Log In”. She authenticates correctly and she is redirected to her personal profile page. She clicks on “Personal Info” on the top of the page and she is redirected to the page where she can visualize her personal information. She click on the “Edit” button corresponding to the mobile phone number and she modifies the existing number, inserting the new one. The she clicks on “Save changes” in order to store the update.</p>

Name	Cancellation of a reservation
Participating Actor	Registered user: Lucy
Flow of Events	<p>A week before, the conference of November 12th is cancelled. So Lucy has to delete the taxi reservation she has made the previous month. She accesses the site and clicks on “Log In”. She inserts her credentials and then she confirms. Her personal profile page is shown: she clicks on “Your Reservation” to visualize the list of all her reservations. She clicks on the “Delete” button corresponding to the reservation she wants to cancel. The system displays a message to ask her to confirm is operation: Lucy clicks on “Yes”.</p> <p>So, a message that informs Lucy that her reservation has been successfully deleted is displayed. Lucy has now canceled her reservation and is redirected to her reservations list page.</p> <p>After a minute she receive both an e-mail notification and a notification in the box on her personal page, confirming the cancellation of the November 12th reservation.</p>

Name	Cancellation of a reservation out of time
Participating Actor	Registered user: Alice
Flow of Events	<p>Some days ago, Alice reserved a taxi using the application “myTaxiService” for today at 2:00 p.m. to go to work.</p> <p>At 1:45 p.m. Alice decides that she doesn’t want that taxi anymore, because she wants to lose weight and she chooses to go on foot. So, she accesses the site and she clicks on “Log In”. She authenticates and she is redirected to her personal profile page. She clicks on “Your Reservations” to visualize her reservations; then she clicks on the button “Delete” corresponding to the reservation she wants to cancel. However an error message is displayed, because she can’t delete a reservation less than 30 minutes before the reservation time. Alice will take the taxi.</p>

Name	Taxi driver notifications
Participating Actor	Taxi driver: Bob
Flow of Events	<p>Bob is starting his daily work, so he gets in his taxi.</p> <p>He wants to communicate that he is available through the application installed on board. So, he clicks on the button “Available” and he waits for the first requests. After ten minutes he receives a notification: a taxi is requested in Savile Road 43, not far from his position. Bob clicks on “Accept”, to communicate to the system that he’s going to take care of the incoming request, and then changes his status on “Occupied”, clicking on the appropriate button. He arrives at the place of the meeting and he drives the passenger to his destination in Rotten Road.</p> <p>Now that he has finished his race, he’s ready to accept other requests, so he clicks on the “Available” button to communicate it to the system.</p>

6 UML Modeling

6.1 Use Cases: identification and description

In this paragraph we are going to identify and describe the most important use cases of our project.

So, based on the scenarios defined in the previous chapter, we can derive some significant use cases:

- Sign up
- Log in
- Recover the password
- Request a taxi
- Reserve a taxi
- Update personal information
- Visualize the reservations
- Delete a reservation
- Receive a request confirmation notification
- Receive a reservation confirmation notification
- Send notifications about taxi's availability
- Receive a notification of a request or a reservation
- Send notifications about accepted requests

6.1.1 Sign up

Name	<i>Sign up</i>
Actors	Guests
Entry conditions	None.
Flow of events	<ul style="list-style-type: none">▪ The guest enters the website.▪ The guest clicks on the “Sign Up NOW!” button.▪ The guest fills out the registration form, writing all the requested information, all mandatory:<ul style="list-style-type: none">- Username and password for his account- E-mail- Name- Surname- Mobile phone number- Credit card number and type- Credit card expiration date (month and year)▪ The guest clicks of the “Confirm” button.▪ The system stores the new data in the passengers database▪ The system displays a confirmation message, informing the new user that his registration has been successfully completed.▪ The system shows the new user his personal profile.
Exit conditions	The registration is successfully completed and the new user enters his personal area.
Exceptions	<ul style="list-style-type: none">★ <u>The guest is already a registered user.</u> If this exception occurs, the system displays the error message “ERROR: you’re already registered” and the application goes back to the homepage.★ <u>The username the guest inserts is already existing.</u> If this exception occurs, the system displays the error message “ERROR: username already chosen” and the application goes back to the page where the registration form is shown.★ <u>The guest doesn’t fill all the mandatory fields in the registration form.</u> If this exception occurs, the system displays the error message “ERROR: all the mandatory fields has to be filled” and the application goes back to the page where the registration form is shown.

6.1.2 Log in

Name	<i>Log in</i>
Actors	Registered users
Entry conditions	The user is registered to the system.
Flow of events	<ul style="list-style-type: none">▪ The user enters the website.▪ The user clicks on the “Log In” button.▪ The user fills out the login form with his username and password.▪ The user clicks of the “Confirm” button.▪ The system shows the user his personal profile.
Exit conditions	The user has accessed to his personal area.
Exceptions	<ul style="list-style-type: none">★ <u>The credentials inserted by the user are incorrect.</u> If this exception occurs, the system displays the error message “ERROR: wrong credentials” on the screen and the application goes back to the login page.

6.1.3 Recover the password

Name	<i>Recover the password</i>
Actors	Registered users
Entry conditions	The user is on the log in page.
Flow of events	<ul style="list-style-type: none">▪ The user clicks on “I forgot my password” and he’s redirected to a new page for the recovery.▪ The user clicks on “Send a new password”.▪ The user is back on the log in page.▪ The user receives an e-mail with the new password.
Exit conditions	The user is back on the log in page and has a new incoming e-mail.
Exceptions	No exceptions.

6.1.4 Request a taxi

Name	<i>Request a taxi</i>
Actors	Registered users
Entry conditions	The user is already logged into the application and he's on his profile page.
Flow of events	<ul style="list-style-type: none">▪ The user clicks the "REQUEST" button in his personal profile page.▪ The user fills out the request form with the information required:<ul style="list-style-type: none">- Name- Surname- Number of passengers (from 1 to 4)- Location.▪ The user clicks on the "Send" button in order to complete his request.▪ The system brings the user back to his personal profile page.
Exit conditions	The user is back on his profile page.
Exceptions	<ul style="list-style-type: none">★ <u>The guest doesn't fill all the mandatory fields in the request form.</u> If this exception occurs, the system displays the error message "ERROR: all the mandatory fields has to be filled" and the application goes back to the page where the request form is shown.★ <u>The request happens when the service is not available (after 9:00 p.m. or before 6:00 a.m.).</u> If this exception occurs, the system displays the error message "ERROR: you're out of time! The system is not available" and the application goes back to the profile page.

6.1.5 Reserve a taxi

Name	Reserve a taxi
Actors	Registered users
Entry conditions	The user is already logged into the application and he's on his profile page.
Flow of events	<ul style="list-style-type: none">▪ The user clicks the "RESERVE" button in his personal profile page.▪ The user fills out the reservation form with the information required:<ul style="list-style-type: none">- Name- Surname- Date of the reservation- Time of the reservation- Number of passengers (from 1 to 4)- Location: origin and destination of the ride▪ The user clicks on the "Send" button in order to complete his reservation.▪ The system brings the user back to his personal profile page
Exit conditions	The user is back on his profile page.
Exceptions	<ul style="list-style-type: none">★ <u>The guest doesn't fill all the mandatory fields in the reservation form.</u> If this exception occurs, the system displays the error message "ERROR: all the mandatory fields has to be filled" and the application goes back to the page where the reservation form is shown.★ <u>The reservation isn't done at least two hours in advanced.</u> If this exception occurs, the system displays the error message "ERROR: the reservation cannot be forwarded! A reservation must be done at least two hours in advance" and the application goes back to the profile page.★ <u>The reservation happens when the service is not available (after 9:00 p.m. or before 6:00 a.m.).</u> If this exception occurs, the system displays the error message "ERROR: you're out of time! The system is not available" and the application goes back to the profile page.

6.1.6 Update personal information

Name	<i>Update information</i>
Actors	Registered users
Entry conditions	The user is already logged into the application and he's on his profile page.
Flow of events	<ul style="list-style-type: none">▪ The user clicks the "Personal Info" link in his personal profile page.▪ The user visualizes his personal information.▪ The user clicks on the "Edit" (or "Add" or "Delete") button corresponding to the information he wants to update.▪ The user modifies the data in the specific input forms.▪ The user click on the "Save changes" button in order to confirm the updates.▪ The system brings the user back to his personal profile page.
Exit conditions	The user is back on his profile page.
Exceptions	<ul style="list-style-type: none">★ <u>The user inserts a new e-mail, a new username or a new password which are not valid.</u> If this exception occurs, the system displays the error message "ERROR: not valid information" and the application goes back to the page where the personal information is shown.★ <u>The user deletes all the credit cards inserted.</u> If this exception occurs, the system displays the error message "ERROR: you must insert at least a credit card" and the application goes back to the page where the personal information is shown.

6.1.7 Visualize the reservations

Name	<i>Visualize reservations</i>
Actors	Registered users
Entry conditions	The user is already logged into the application and he's on his profile page.
Flow of events	<ul style="list-style-type: none">▪ The user clicks the "Your Reservations" link in his personal profile page.▪ The user visualizes his reservations.
Exit conditions	The user is on his reservations list page.
Exceptions	No exceptions.

6.1.8 Delete a reservation

Name	<i>Delete a reservation</i>
Actors	Registered users
Entry conditions	The user is already logged into the application and he's visualizing the list of his reservations.
Flow of events	<ul style="list-style-type: none">▪ To every reservation corresponds a button that allows the user to cancel it, so the user clicks the "Delete" button corresponding to the reservation he wants to cancel.▪ The system displays a message to ask the user to confirm is operation▪ The user clicks on "Yes".▪ The system displays a confirmation message, informing the new user that his reservation has been successfully deleted.▪ The system brings the user back to his reservations list page.▪ The user receives both an e-mail notification and a notification in the box on his personal page, confirming the cancellation.
Exit conditions	The user is on his reservations list page.
Exceptions	<ul style="list-style-type: none">★ <u>The user decides to cancel a reservation when the time is over (less than 30 minutes before the reservation).</u> If this exception occurs, the system displays the error message "ERROR: you're out of time. This reservation cannot be deleted" and the application goes back to the page where all the user's reservations are shown.★ <u>The user clicks "No" when the system asks to confirm the delete operation.</u> If this exception occurs, the application goes back to the page where all the user's reservations are shown.

6.1.9 Receive a request confirmation notification

Name	<i>Receive a request confirmation notification</i>
Actors	Registered users
Entry conditions	The user is already logged into the application and he has just requested a taxi.
Flow of events	<ul style="list-style-type: none">▪ The system sends an e-mail and shows a notification in the user's notifications box, confirming the request and communicating the information (code and waiting time) of the incoming taxi.
Exit conditions	The user has a new incoming e-mail and a new notification in his notifications box.
Exceptions	No exceptions.

6.1.10 Receive a reservation confirmation notification

Name	<i>Receive a reservation confirmation notification</i>
Actors	Registered users
Entry conditions	The user is already logged into the application and he has just successfully completed a reservation.
Flow of events	<ul style="list-style-type: none">▪ The system sends an e-mail and shows a notification in the user's notifications box, confirming the reservation.
Exit conditions	The user has a new incoming e-mail and a new notification in his notifications box.
Exceptions	No exceptions.

6.1.11 Send notifications about taxi's availability

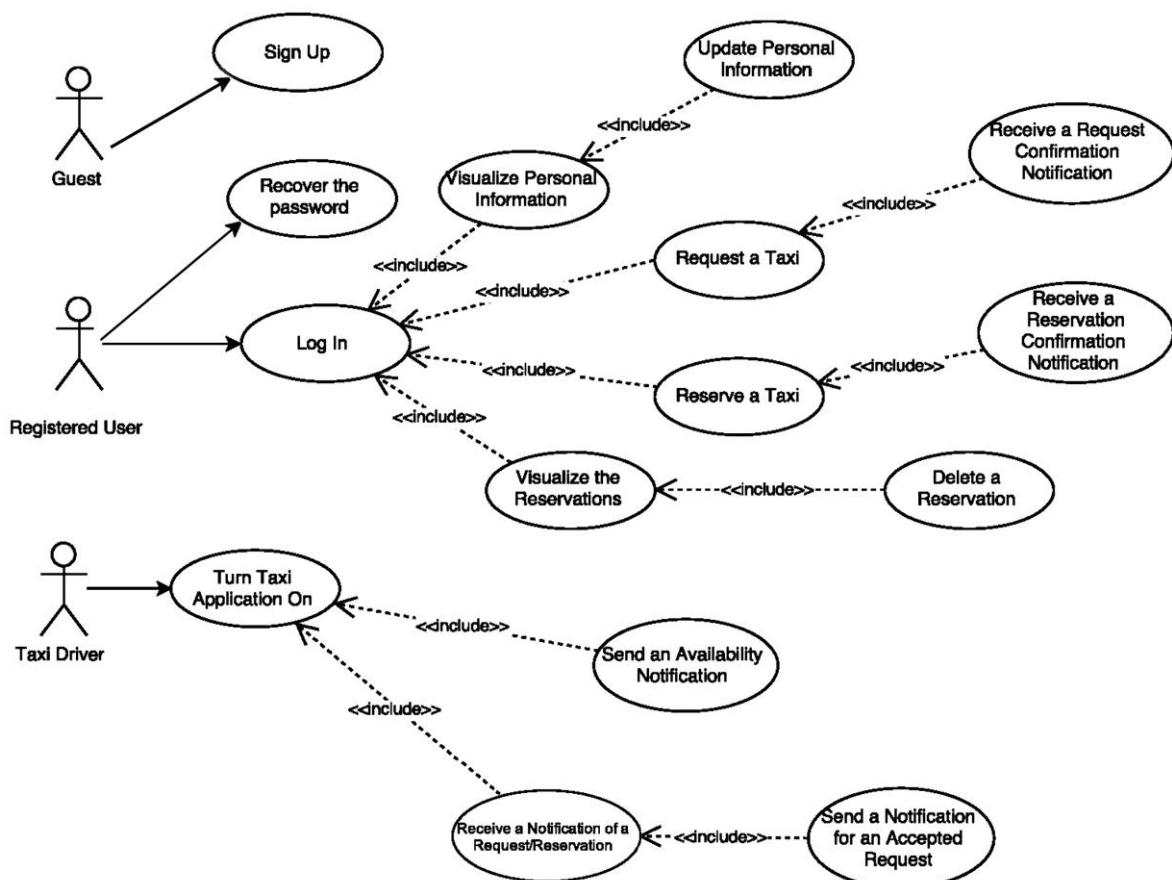
Name	<i>Send an availability notification</i>
Actors	Taxi drivers
Entry conditions	The taxi application is on (the taxi is automatically logged into the system).
Flow of events	<ul style="list-style-type: none">▪ The taxi driver clicks on the "Available" or on the "Occupied" button to communicate to the system whether he's available to accept a request or not.
Exit conditions	The taxi status is now "Available" or "Occupied".
Exceptions	No exceptions.

6.1.12 Receive a notification of a request or a reservation

Name	<i>Receive a request or reservation notification</i>
Actors	Taxi drivers
Entry conditions	The taxi application is on (the taxi is automatically logged into the system).
Flow of events	<ul style="list-style-type: none">▪ The system shows a notification in the notifications box of a specific taxi communicating an incoming request or reservation, specifying the corresponding location.
Exit conditions	The taxi driver has a new notification in his notifications box.
Exceptions	No exceptions.

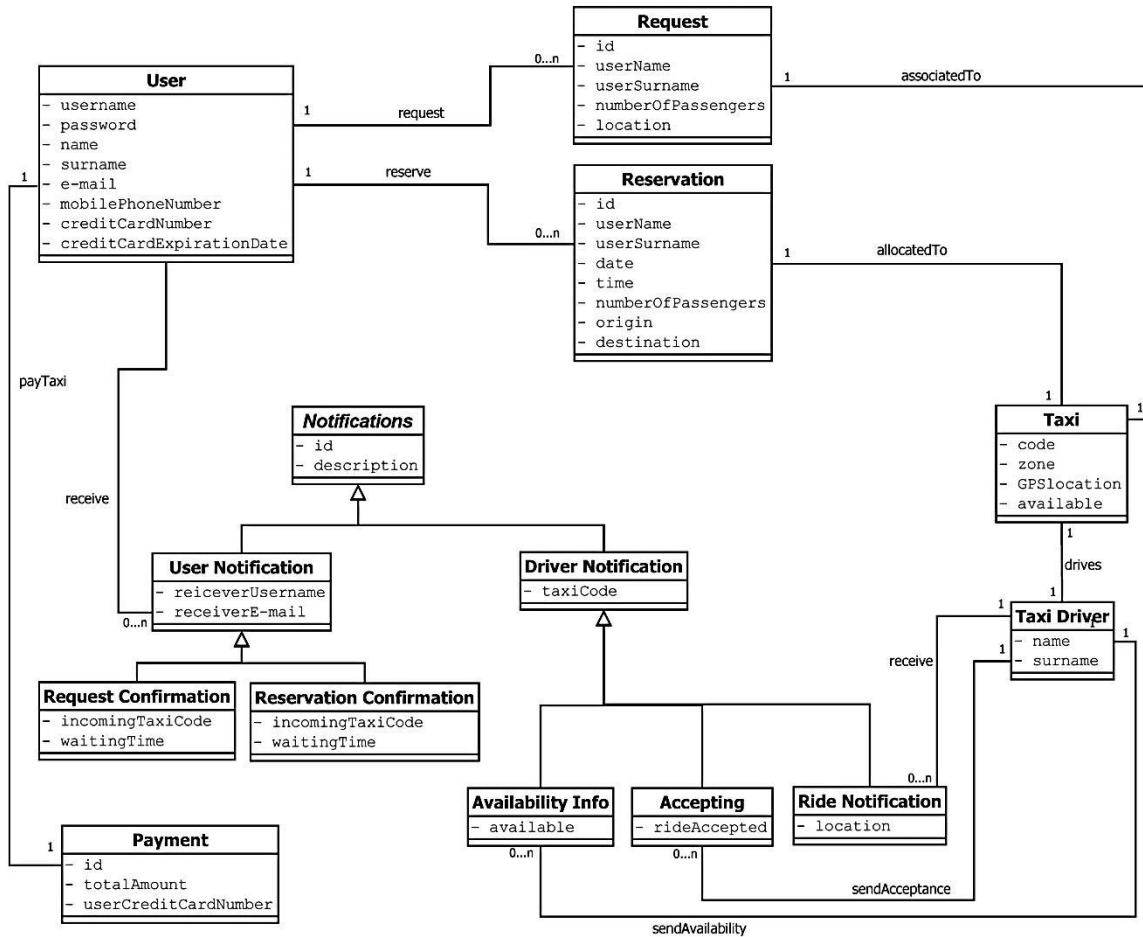
6.1.13 Send notifications about accepted request

Name	Send a notification for an accepted request
Actors	Taxi drivers
Entry conditions	The taxi application is on (the taxi is automatically logged into the system) and the driver has just received a new request or reservation.
Flow of events	<ul style="list-style-type: none"> ▪ The taxi driver clicks on the “Accept” button next to the notification of the request that he’s going to take care of. ▪ The taxi driver clicks on “Occupied”.
Exit conditions	The taxi status is now “occupied”.
Exceptions	<p>★ <u>The taxi driver doesn’t accept the new request or the new reservation within a minute.</u></p> <p>If this exception occurs, the system marks the taxi as “Occupied” and delete the request notification, forwarding it to the next one.</p>



6.2 Class Diagram

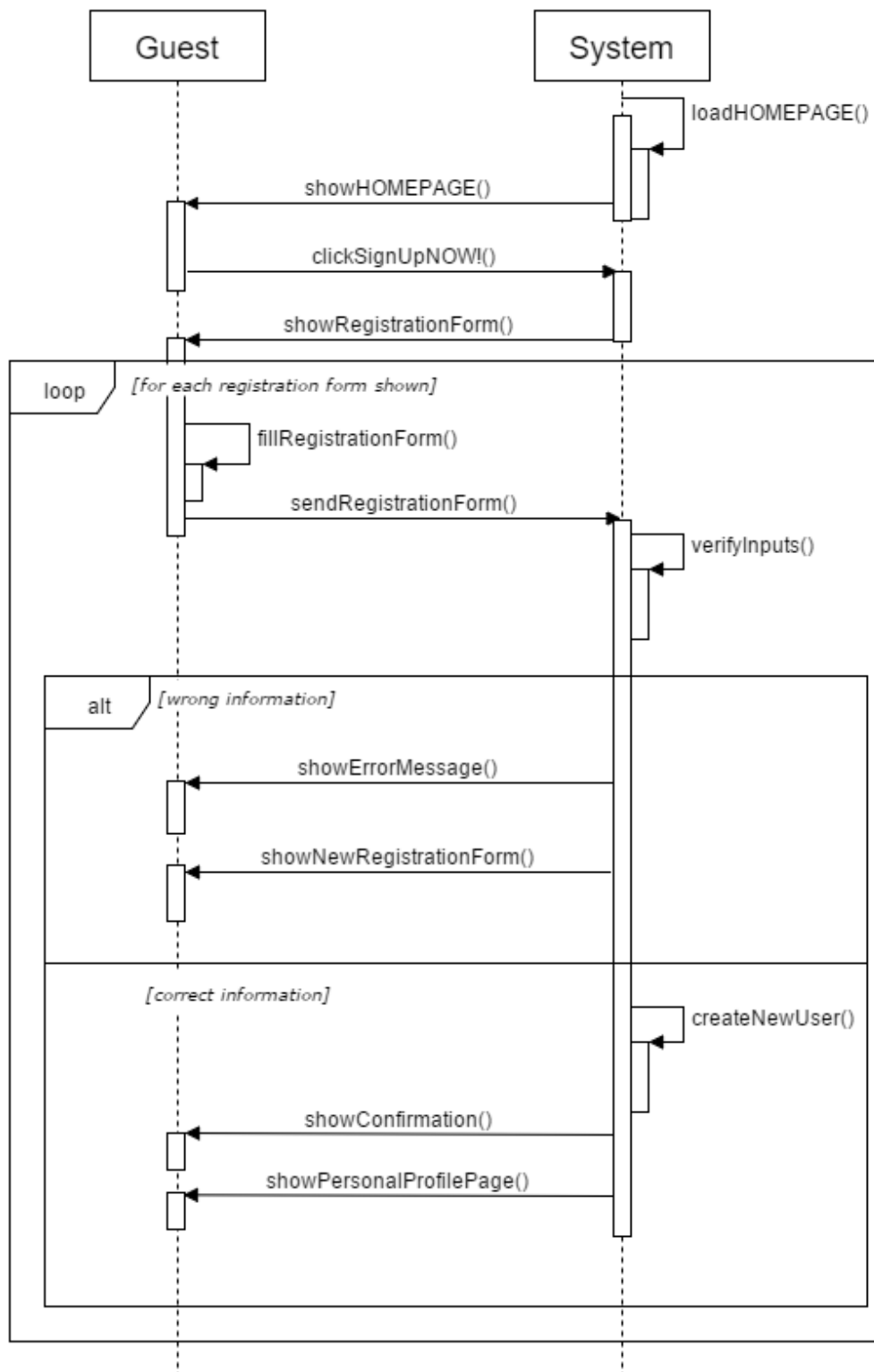
Now that we have identified and described our use cases, we can draw a general and simplified class diagram of our system:



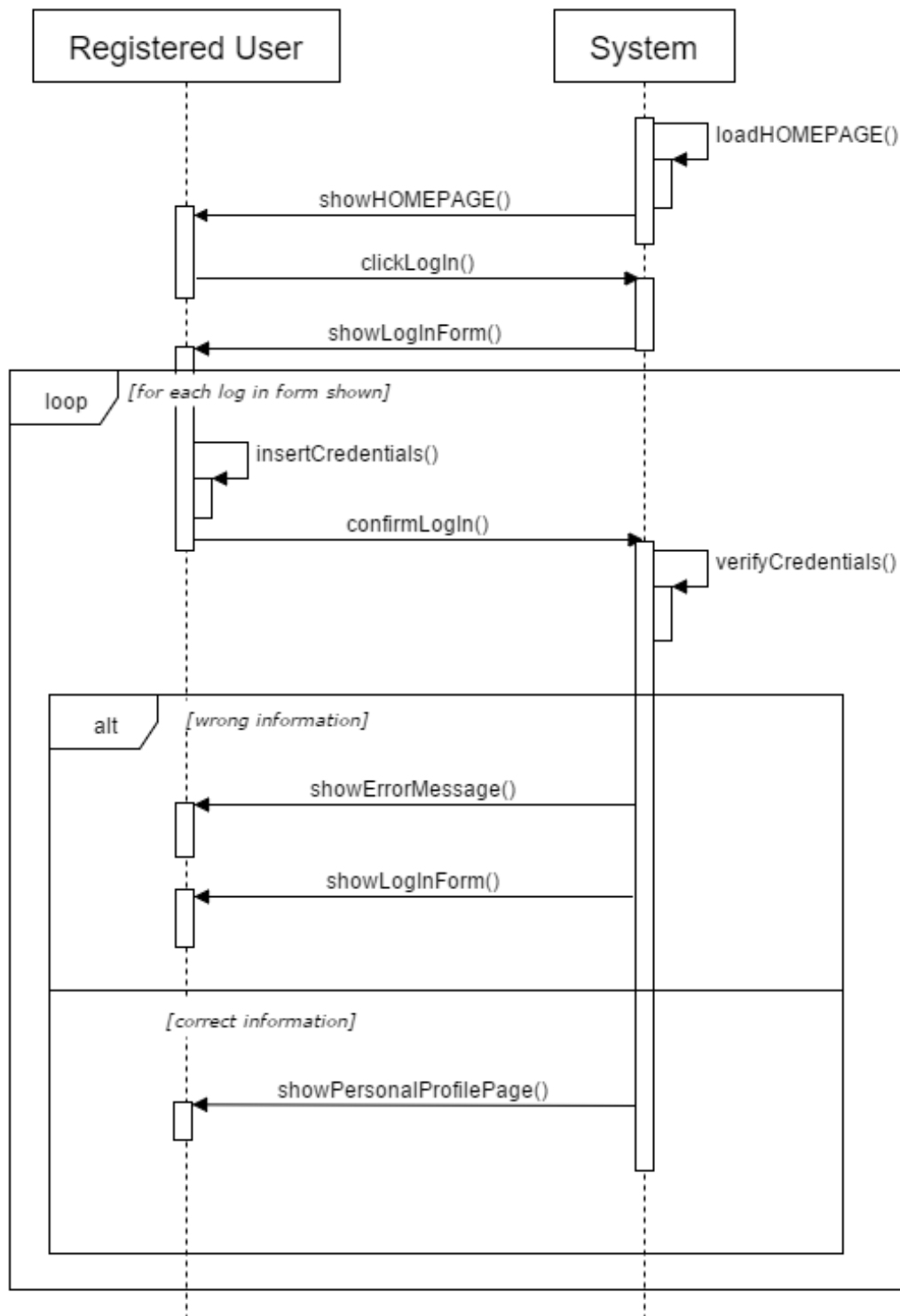
6.3 Sequence Diagrams

In this paragraph we provide the sequence diagrams associated to the use cases defined in the previous paragraphs: these are interaction diagrams that allow us to model the interactions between objects. So, starting from a specific scenario or from the flow of events defined in a specific use case, we model the interactions, the messages and the events that occurs in a specific situation.

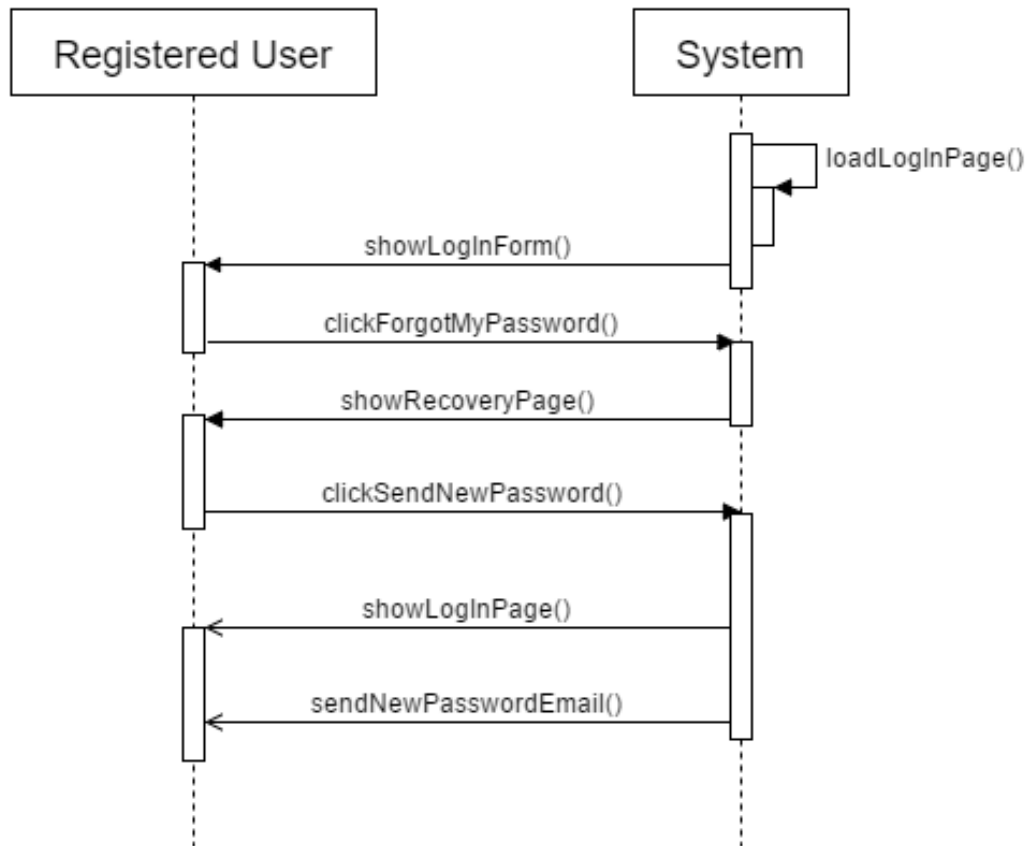
6.3.1 Sign up



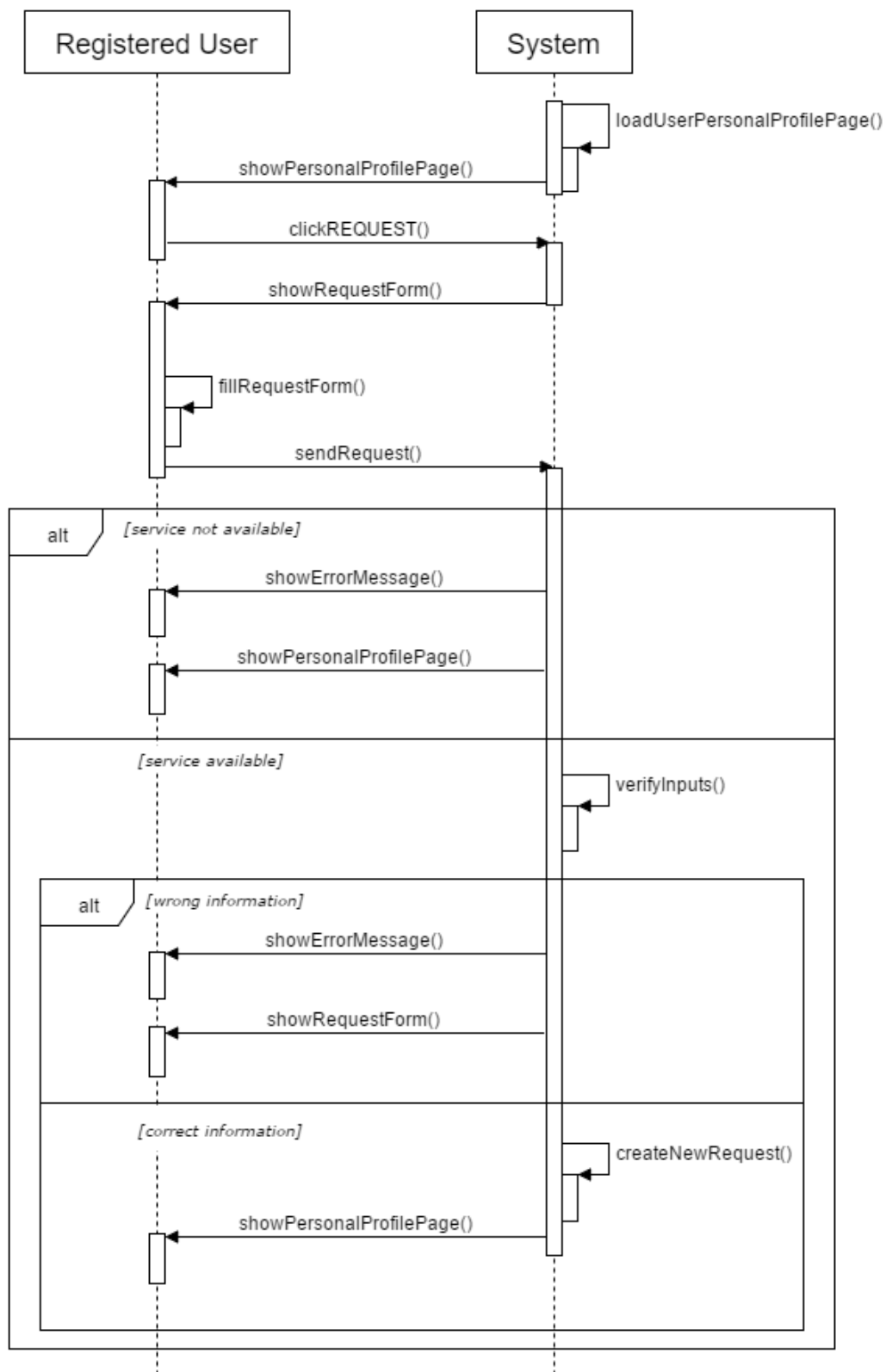
6.3.2 Log in



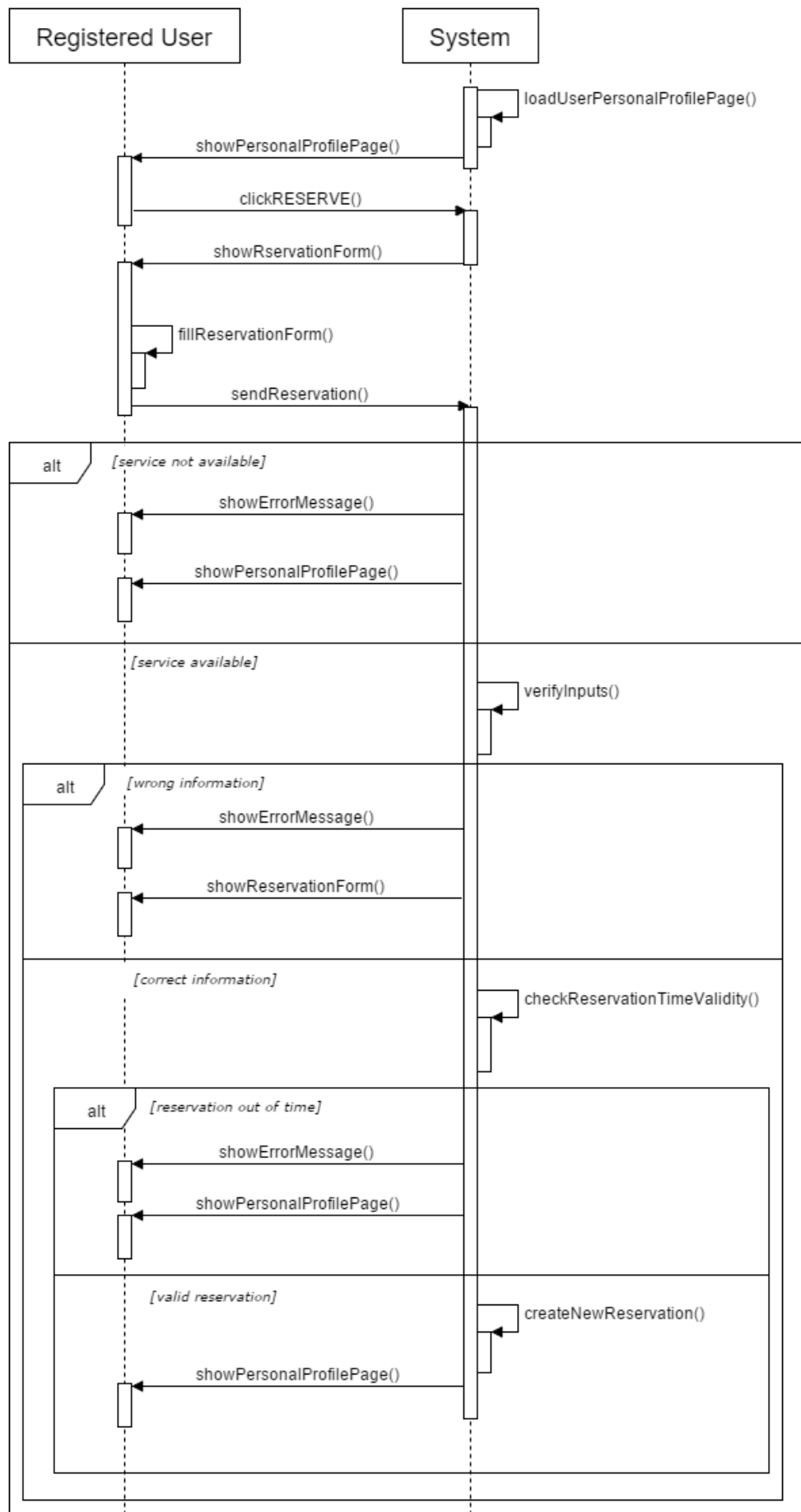
6.3.3 Recover the password



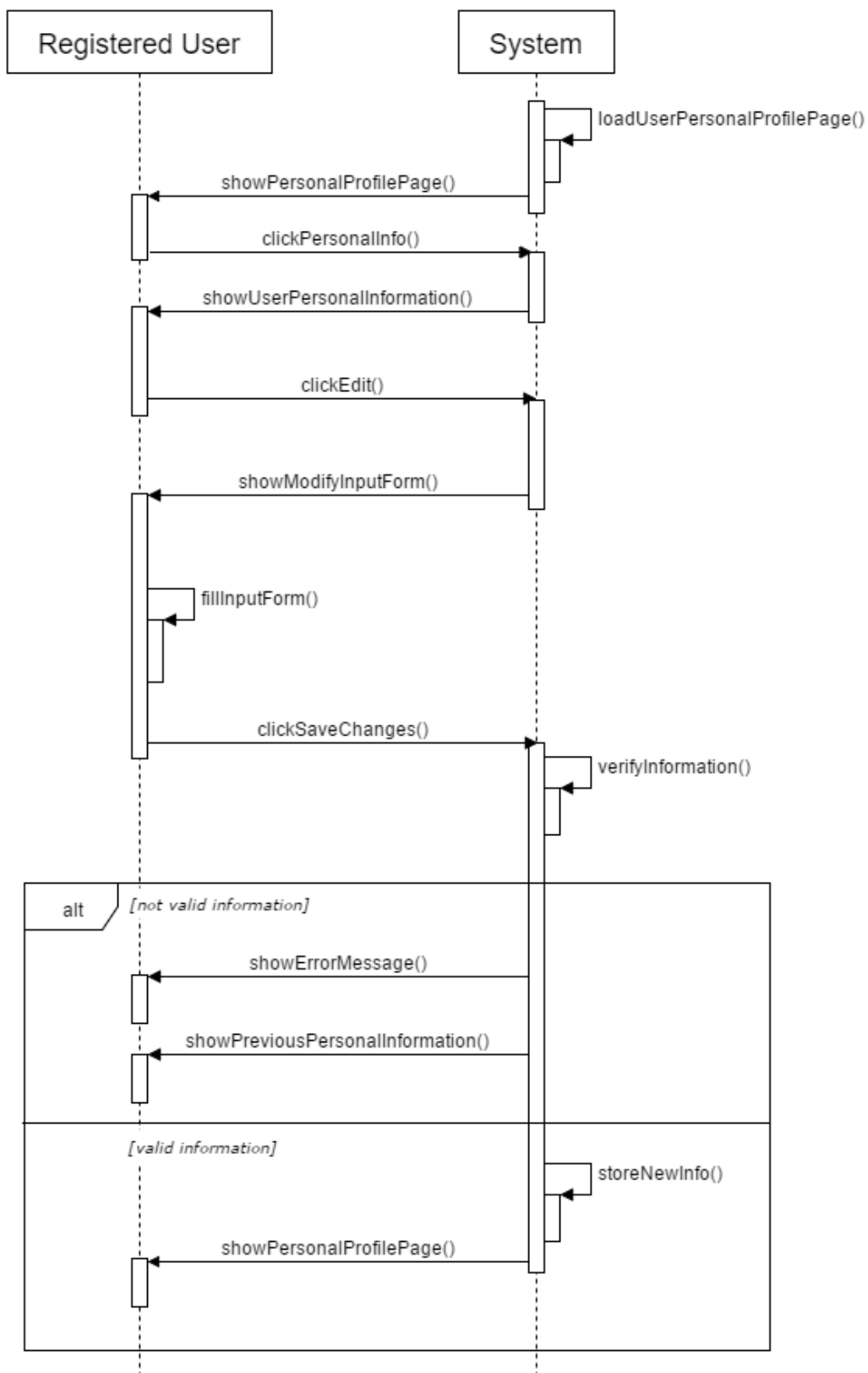
6.3.4 Request a taxi



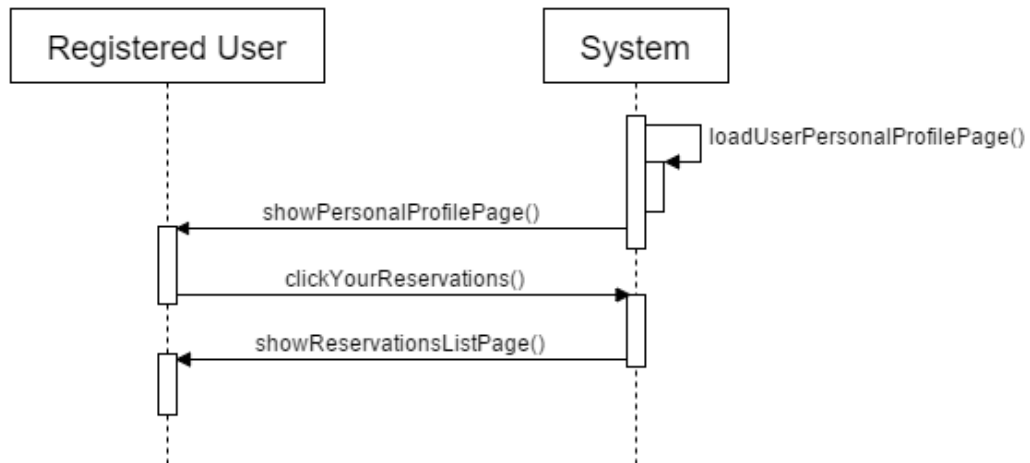
6.3.5 Reserve a taxi



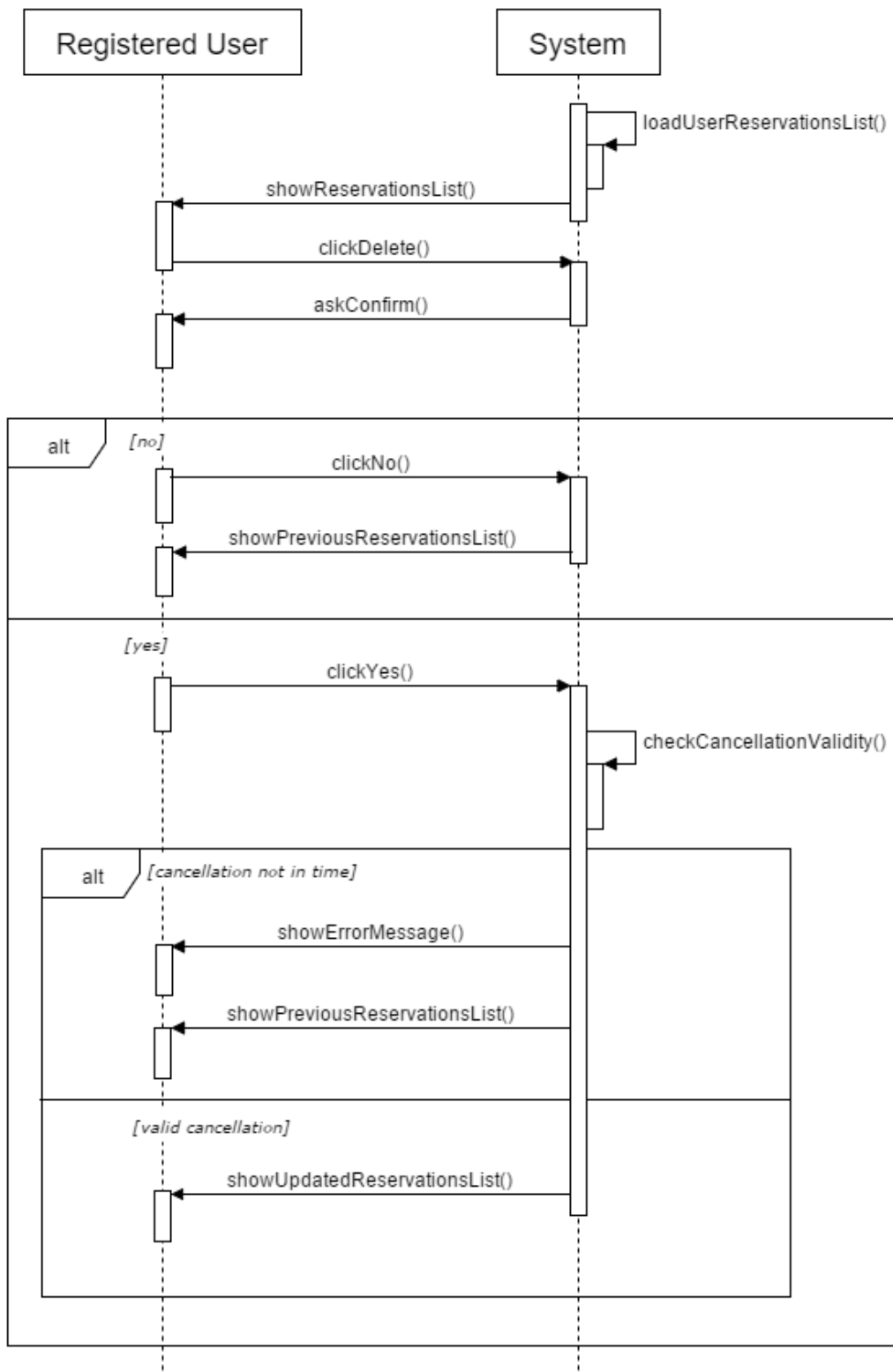
6.3.6 Update personal information (modify)



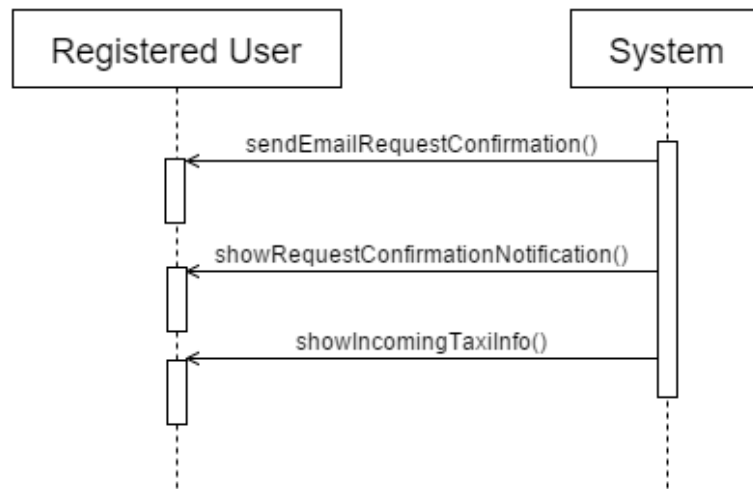
6.3.7 Visualize the reservations



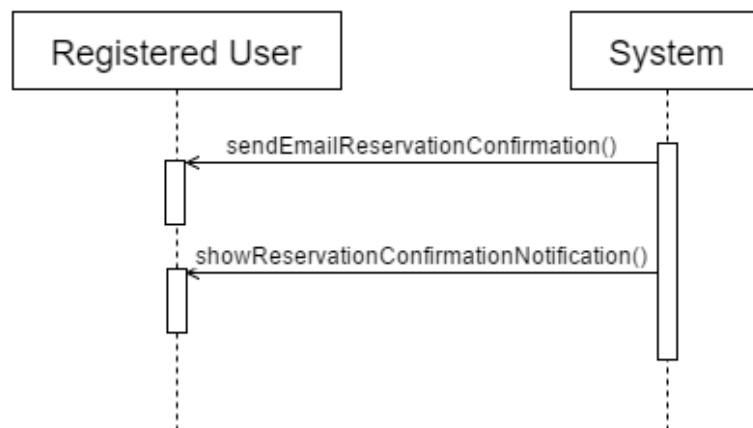
6.3.8 Delete a reservation



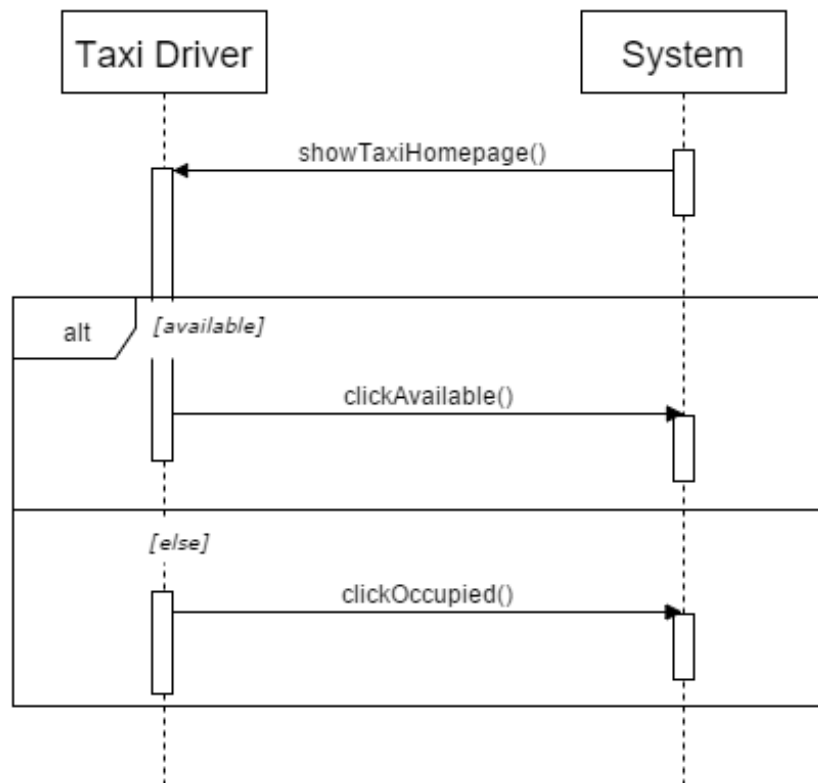
6.3.9 *Receive a request confirmation notification*



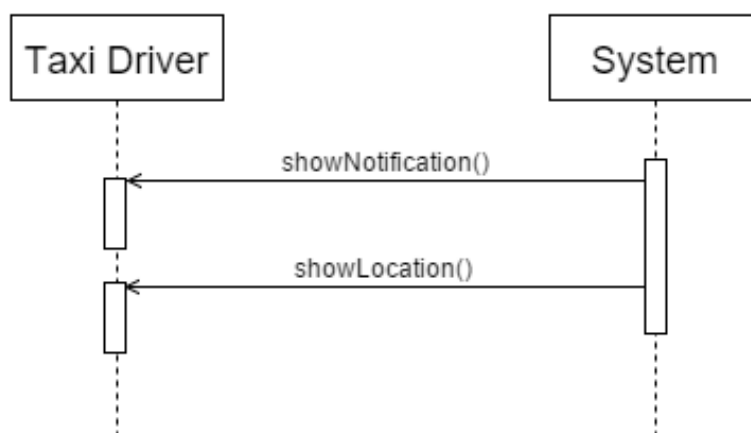
6.3.10 *Receive a reservation confirmation notification*



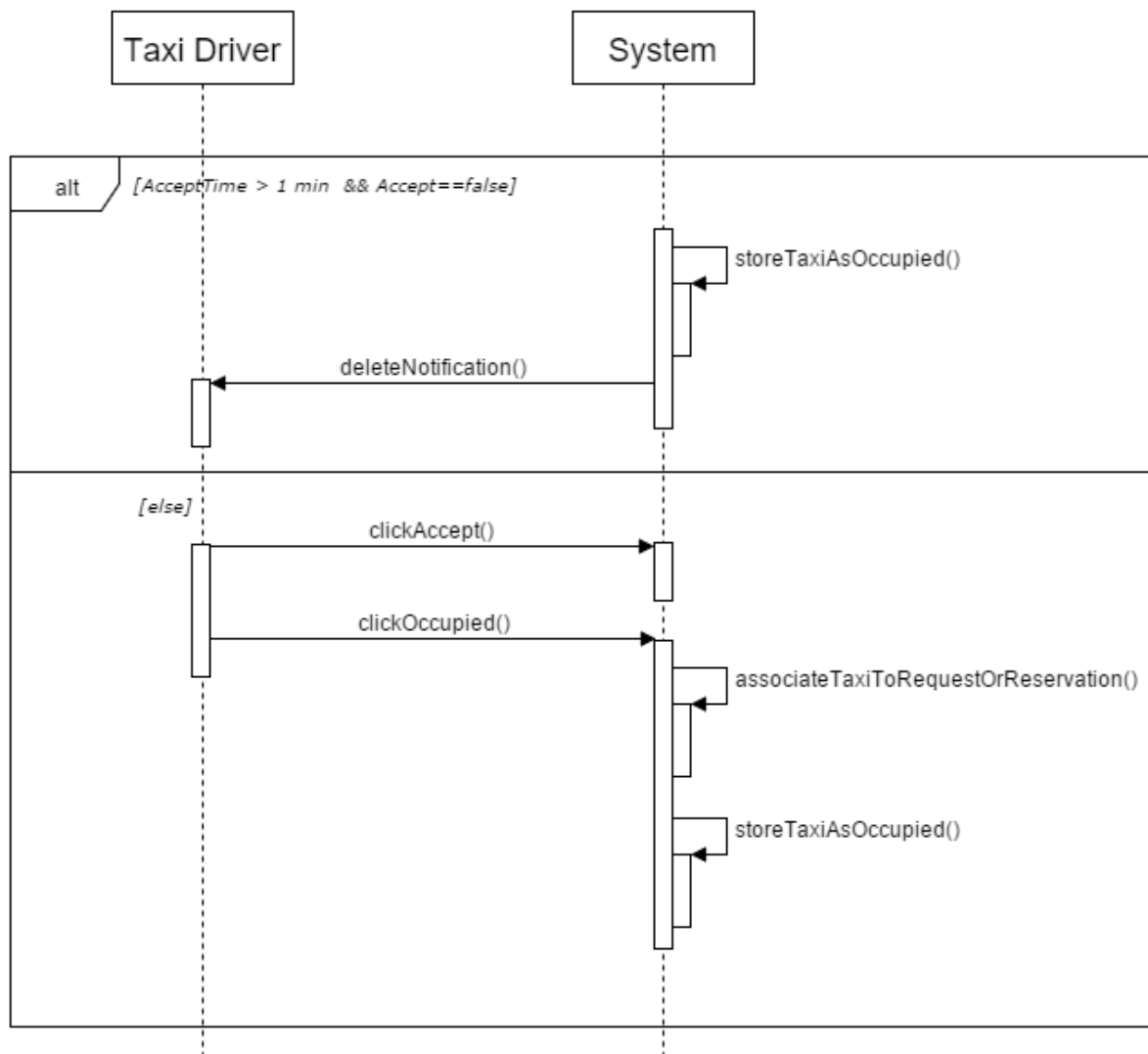
6.3.11 Send notifications about taxi's availability



6.3.12 Receive a notification of a request or a reservation



6.3.13 Send notifications about accepted request



7 Alloy Modeling

7.1 Introduction to alloy

Alloy is a formal language used to define models of systems and software: its aim is describing the structural and behavioral properties of a software and all its specifications. It uses first order logic predicates and relational calculus: the basic concept of Alloy is the one of “relation”, so, even the relational algebra is one of the basis of this notation.

Furthermore, it has a powerful and fast automatic tool to simulate and verify the validity of the specific properties defined.

In order to verify some specifications and properties of our system, we used the Alloy language and the Alloy tool (Alloy Analyzer 4.2).

We made a very small model that represents the most important dynamics and some actors and entities of our system:

- User
- Taxi
- Need (Reservation or Request)
- TaxiDriver
- UserNotification
- TaxiDriverNotification
- Payment

From a purely logical point of view, it was decided to reason about the following basic logical relationships:

User = ((U1), (U2), (U3), ...)

All the users are different from each other.

Taxi = ((T1), (T2), (T3), ...)

All the taxis are different from each other.

Need = ((N1), (N2), (N3), ...)

A need can be a reservation or a request. All needs are different from each other.

Reservation = ((RS1), (RS2), (RS3),...)

All the reservations are different from each other.

Request = ((RQ1), (RQ2), (RQ3),...)

All the requests are different from each other.

TaxiDriver = ((D1), (D2), (D3), ...)

All the taxi-drivers are different from each other.

UserNotification = ((UN1), (UN2), (UN3), ...)

The user notification is the notification that the system sends to the user.

All the user-notifications are different from each other.

TaxiDriverNotification = ((DN1), (DN2), (DN3), ...)

The taxi-driver notification is the notification that the system sends to the taxi-driver.

All the taxi-driver notifications are different from each other.

Payment = ((P1), (P2), (P3), ...)

All payments are different from each other.

The following two relations link the need to the user:

reservationOwnerConservation = ((RS1,U2), (RS2,U3), (RS3,U5), ...)

Every reservation can be related to only one user.

requestOwnerConservation = ((RQ1, U1), (RQ2, U1), (RQ3, U2), ...)

Every request can be related to only one user.

userNotificationNeedConservation = ((UN1, N1), (UN4, N3), (UN5, N9), ...)

Every notification can be related to only one need (reservation or request).

userNotificationNeed = ((U1,UN1), (U2, UN4), (U3, UN5), ...)

Every user-notification can be related to only one user.

The objective is to create the following ternary relation:

userNotification-user-need = ((UN1,U1,N1), (UN4,U2,N3), (UN5,U3,N9),...)

taxi-TaxiDriver = ((T1,D1),(T3,D2),(T4,D5),...)

Every taxi has an only taxi-driver.

needTaxi = ((N1, T2), (N3, T4), (N5, T5), ...)

Every taxi can be related to only one need (reservation or request).

driverNotificationNeedConservation = ((DN1,N1), (DN2,N3), (DN3,N5), ...)

Every driver-notification can be related to only one taxi-driver.

driverNotificationNeed = ((D1,DN1), (D2,DN2), (D3,DN3), ...)

Every driver-notification can be related to only one taxi-driver.

The objective is to create the following ternary relation:

driverNotification-driver-need = ((D1,DN1, N1), (D2,DN2, N5), (D3,DN3, N5), ...)

$\text{paymentNeedConservation} = ((P1, N1), (P2, N3), (P3, N5), \dots)$

Every payment can be related to only one need (reservation or request).

$\text{paymentUser} = ((P1, U1), (P2, U3), (P3, U7), \dots)$

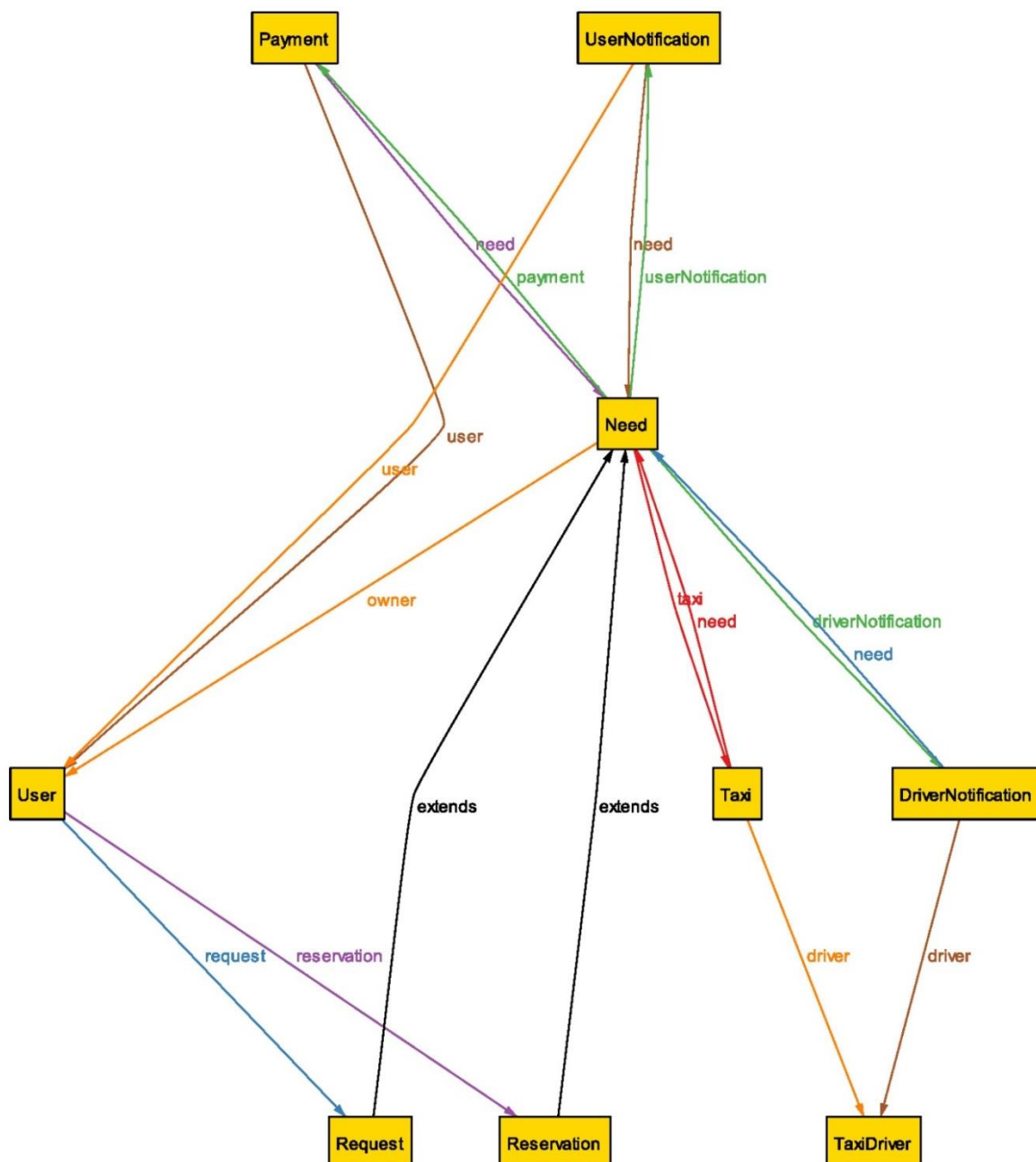
Every payment can be related to only one user.

The objective is to create the following ternary relation:

$\text{payment-user-need} = ((P1, U1, N1), (P2, U3, N3), (P3, U7, N5), \dots)$

7.2 Alloy code

Here the metamodel we generate through the Alloy Analyzer:



We now report the code we wrote and, in the next paragraph, some examples of World generated by Alloy Analyzer, in order to prove that our model is consistent,

```

/*****MODEL*****/
sig User{
    request: set Request,
    reservation: set Reservation
}

abstract sig Need{
    taxi: one Taxi,
    owner: one User,
    userNotification: one UserNotification,
    driverNotification: one DriverNotification,
    payment: one Payment
}

sig Request extends Need{}

sig Reservation extends Need{}

sig Taxi{
    need: one Need,
    driver: one TaxiDriver
}

sig TaxiDriver{
}{
    // one taxi has exactly one driver
    one t: Taxi | this = t.driver
}

sig UserNotification {
    need: one Need,
    user: one User
}

sig DriverNotification {
    need: one Need,
    driver: one TaxiDriver
}

sig Payment{
    need: one Need,
    user: one User
}
```

```

// one reservation has exactly one user
fact reservationOwnerConservation {
    all r: Reservation, u: User | r in u.reservation iff r.owner = u
}

// one request has exactly one user
fact requestOwnerConservation{
    all r: Request, u: User | r in u.request iff r.owner = u
}

// one request or reservation has exactly one taxi
fact needTaxi{
    all t: Taxi, n: Need | t in n.taxi iff t.need = n
}

// for each request or reservation a userNotification is sent to the user
fact userNotificationNeedConservation{
    all u: UserNotification, n: Need |
        u in n.userNotification iff u.need = n
}

fact userNotificationNeed {
    all n: UserNotification | n.user in n.need.owner
}

// for each request or reservation a driverNotification is sent
// to the taxi driver
fact driverNotificationNeedConservation{
    all d: DriverNotification, n: Need |
        d in n.driverNotification iff d.need = n
}

fact driverNotificationNeed {
    all n: DriverNotification | n.driver in n.need.taxi.driver
}

// for each request or reservation a payment is request to the user
fact paymentNeedConservation{
    all p: Payment, n: Need | p in n.payment iff p.need = n
}

fact paymentNeed {
    all p: Payment | p.user in p.need.owner
}

```

```

/*****ASSERTIONS*****/

assert oneTaxiOneTaxiDriver{
    all d: TaxiDriver | one t: Taxi | t.driver = d
}
check oneTaxiOneTaxiDriver

assert oneNeedOneTaxi{
    all n: Need | one t: Taxi | t.need = n
}
check oneNeedOneTaxi

assert oneNeedOnePayment{
    all n: Need | one p: Payment | p.need = n
}
check oneNeedOnePayment

assert oneNeedOneUserNotification{
    all n: Need | one u: UserNotification | u.need = n
}
check oneNeedOneUserNotification

assert oneNeedOneDriverNotification{
    all n: Need | one d: DriverNotification | d.need = n
}
check oneNeedOneDriverNotification

assert notificationConnectedAtLeastOneNeed{
    no n: UserNotification | n not in Need.userNotification
}
check notificationConnectedAtLeastOneNeed

assert driverNotificationConnectedAtLeastOneNeed{
    no d: DriverNotification | d not in Need.driverNotification
}
check driverNotificationConnectedAtLeastOneNeed

/*****SHOW*****/

pred show() {
}
run show for 3 but exactly 3 User, exactly 2 Reservation, exactly 1 Request

```

7.3 Result of analyzer

We report here an example of the result given by the Analyzer:

Executing "Check oneTaxiOneTaxiDriver"

Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1912 vars. 162 primary vars. 3297 clauses. 360ms.
No counterexample found. Assertion may be valid. 62ms.

Executing "Check oneNeedOneTaxi"

Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1912 vars. 162 primary vars. 3297 clauses. 296ms.
No counterexample found. Assertion may be valid. 47ms.

Executing "Check oneNeedOnePayment"

Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1912 vars. 162 primary vars. 3297 clauses. 359ms.
No counterexample found. Assertion may be valid. 47ms.

Executing "Check oneNeedOneUserNotification"

Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1912 vars. 162 primary vars. 3297 clauses. 281ms.
No counterexample found. Assertion may be valid. 47ms.

Executing "Check oneNeedOneDriverNotification"

Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1912 vars. 162 primary vars. 3297 clauses. 234ms.
No counterexample found. Assertion may be valid. 31ms.

Executing "Check notificationConnectedAtLeastOneNeed"

Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1893 vars. 162 primary vars. 3216 clauses. 179ms.
No counterexample found. Assertion may be valid. 16ms.

Executing "Check driverNotificationConnectedAtLeastOneNeed"

Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1893 vars. 162 primary vars. 3216 clauses. 194ms.
No counterexample found. Assertion may be valid. 172ms.

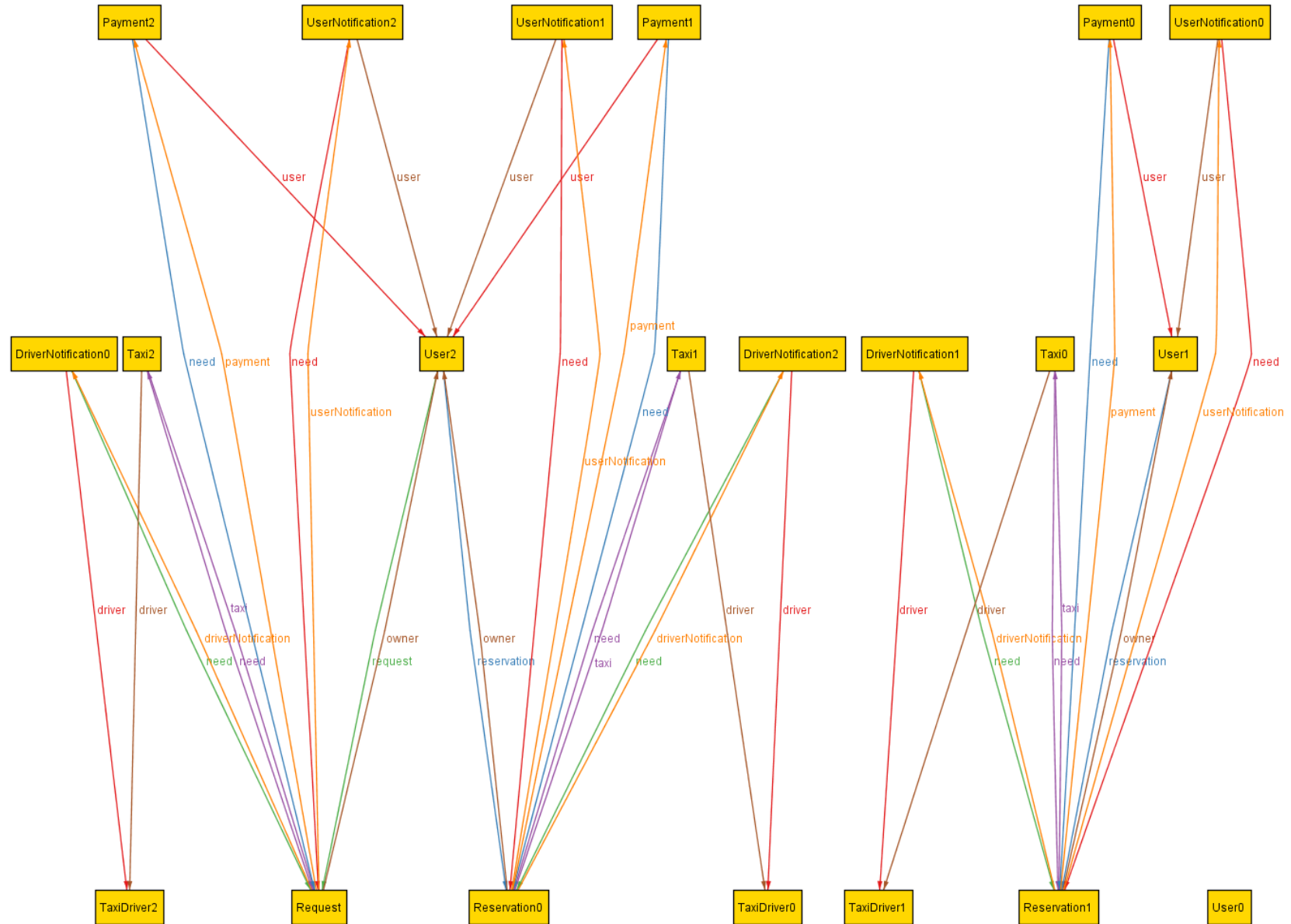
Executing "Run show for 3 but exactly 3 User, exactly 2 Reservation, exactly 1 Request"

Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1459 vars. 141 primary vars. 2519 clauses. 141ms.
Instance found. Predicate is consistent. 109ms.

8 commands were executed. The results are:

- #1: No counterexample found. oneTaxiOneTaxiDriver may be valid.
- #2: No counterexample found. oneNeedOneTaxi may be valid.
- #3: No counterexample found. oneNeedOnePayment may be valid.
- #4: No counterexample found. oneNeedOneUserNotification may be valid.
- #5: No counterexample found. oneNeedOneDriverNotification may be valid.
- #6: No counterexample found. notificationConnectedAtLeastOneNeed may be valid.
- #7: No counterexample found. driverNotificationConnectedAtLeastOneNeed may be valid.
- #8: **Instance found.** show is consistent.

7.4 Generated world



8 Used Tools

In this paragraph we are going to list all the tools we have used to create this document:

- *Microsoft Office Word 2010* : to redact and format this document
- *Moqups* : to create sketches
- *Draw.io* : to create UML Use Case Diagrams and Sequence Diagrams
- *Dia* : to create UML Class Diagram
- *Alloy Analyzer 4.2* : to prove the consistency of our model

9 Working Hours

The document has been developed trying to work together most of the time. It's important to underline that no section has been exclusively developed by a single member of the group: time has been divided rationally in such a way that each member of the group checks and reviews the work done by the other (at least 2/3 hours).

As for the impossibility to work always together, we decided to divide the tasks: Polidori mostly worked on the Overall Description, the Scenarios Identification and the Alloy Modeling, while Regondi focused on the Actors Identification, the Requirements section and the UML Modeling.

The Introduction was produced jointly.

Our project plan was characterized by a daily meeting of at least 3/4 hours to check the whole project and the new compositions produced in the meantime.

We started the redaction of the document as soon as possible, in order to finish some days before the deadline and have the possibility to review everything carefully and correct possible mistakes.

Here a full table containing all the information about our working hours:

ASSIGNMENT	DEADLINE	START DATE	END DATE	DAYS	HOURS		
					<i>Polidori</i>	<i>Regondi</i>	<i>TOT</i>
RASD	06 / 11 / 2015	18 / 10 / 2015	05 / 11 / 2015	18	48	48	96
1. <i>Introduction</i>		18 / 10 / 2015	25 / 10 / 2015	8	5	5	10
2. <i>Overall Description</i>		21 / 10 / 2015	24 / 10 / 2015	4	5	3	8
3. <i>Actors Identification</i>		21 / 10 / 2015	23 / 10 / 2015	3	1	2	3
4. <i>Requirements</i>		21 / 10 / 2015	25 / 10 / 2015	5	4	9	13
5. <i>Scenarios Identification</i>		24 / 10 / 2015	28 / 10 / 2015	5	6	3	9
6. <i>UML Modeling</i>		24 / 10 / 2015	31 / 10 / 2015	8	4	16	20
7. <i>Alloy Modeling</i>		28 / 10 / 2015	03 / 11 / 2015	7	17	4	21
<i>Review (of the whole document)</i>		03 / 11 / 2015	05 / 11 / 2015	3	6	6	12