Politecnico di Milano
A.A. 2015-2016
Software Engineering 2: “myTaxiService”

Requirements Analysis and Specifications Document

Luiza Bentivoglio, Michele Cantarutti

6 November 2015

Summary

1. Introduction

1.1 Description of the given problem

The aim of this project is to improve the management of the taxi service in a large city, mostly by simplifying the access of passengers to the service and by simplifying the access of passengers to the service and by guaranteeing a fair management of taxi queues.
To further ensure the latter, passengers can also make a reservation by making a request at least 2 hours earlier and by specifying the origin and destination of the ride. To adapt to all situations, the system allows passengers to make requests either through a web application or a mobile app.
The requests will be passed on to the taxi drivers by the system and they will answer through an app.
The taxi drivers will be asked whether they want to accept a call only if they have previously notified the system about their availability.

1.2 Goals

We thought about the possible behaviors of those who will interact with the system and what myTaxiService should provide them, so we planned to give our system these features:

- Users should be able to:
  - Sign up into the system;
  - Log into the system;
  - Request a taxi;
  - Cancel a request;
  - Reserve a taxi;
  - View the reservations;
  - Modify a reservation.
  - Cancel a reservation.

- Taxi drivers should be able to:
  - Log into the system;
  - Inform the system about their availability;
  - Accept a call from the system;
  - Refuse a call from the system;
  - Inform the system they can no longer take care of a call they'd previously accepted.

1.3 Domain Properties

We suppose that these conditions hold in the analyzed world:

• GPS localization is used to know where the taxis are;
• Taxi drivers must be inside the city;
• Taxi drivers should be able to usethe necessary technology to communicate with the system.
• Users should be able to usethe necessary technology to communicate with the system.
• There always is at least one available taxi driver who will accept a request.

1.4 Glossary

Here are the definitions of some words that will be used in our documents.

• USER: by user we mean a person who is already registered in the system. This means they have a profile, through which they can make requests and reservations, and manage them.

• GUEST: a guest is a person who hasn't signed up yet.
Unlike registered users, they cannot make requests and reservations, but they can still access the app, either to sign up or to read about the available features they will be able to access once they register.

• TAXI DRIVER: by taxi driver we mean a person who will interact with the system with the purpose of serving users. Taxi drivers can inform the system about their availability and they can confirm that they are going to take care of a call that the system is trying to assign to them.

• QUEUE: by queue we mean a queue of available taxis in a zone.

• ZONE: by zone we mean an area of the city. Each zone is approximately $2km^2$and is associated with a single queue of taxis.

• REQUEST: a request can be made by a registered user when they want a taxi right away. Once a user has made a request, a taxi will come pick them up shortly after, at the location specified by the user.

• RESERVATION: a reservation can be made by a registered user when they want to book a taxi at a certain time. If a user wants to make a reservation, they will have to provide additional details, namely, the origin and destination of the ride, along with the meeting time. Reservations must be made at least two hours before the desired meeting time.

1.5 Assumptions

There are a few points that are not very clear in the specification document, so we will have to assume some facts.
We assume that:

• The user can cancel requests and reservations.

• The user can modify reservations. If a user wants to modify a reservation, they must notify the system at least 10 minutes before the meeting time, otherwise the reservation will be cancelled.

• If a taxi driver informs the system that theyare available and then they decide to move from a zone to another, then the system will remove their taxi from the queue of the previous zone and move it to the last position of the queue of the new zone.

• Once a call has been forwarded to anavailable taxi driver, a timer starts. If the taxi driver doesn't confirm by the time the timer runs out, the system assumes the taxi driver is no longer available, it moves their taxi to the last position in the queue and it forwards the request to the next available driver in the queue.

• If a taxi driver accepts a call, the system will automatically set the taxi driver as unavailable and therefore remove them from the queue.

• If a taxi driver accepts a call and has a mishap before meeting up with the client, they will notify the system they can no longer take care of that call.

• Usernames and passwords are given to the taxi drivers by the manager of the system (which is the government in our case).

1.6 Proposed system

Our system will follow a client-server pattern. The server will generate dynamic pages for the clients, access data and directly manage queues and requests, so as to always find an optimal solution, by minimizing the waiting time of passengers.
There will be two distinct versions of the application on the client side.
One version will only be available to taxi drivers and it will strictly be a mobile application.It is necessary that taxi drivers can use it to interact with the system, to let it know whether they're available or not, and to let it know if they intend to take care of a call that has been forwarded to them.
The other version will only be available to passengers, registered users, to be precise. This version will be both accessible through a mobile application and a web application.

1.7 Identifying stakeholders

We can identify three main stakeholders. First and foremost, the government, who certainly have great interests in the success of this project, since they assigned us this task.In fact, by granting an improved service, the project will improve an aspect of the life in the city, that is transportation. Passengers are those who will actively enjoy this service, therefore we can assume they also belong to the category of the stakeholders without doubt. Lastly, taxi drivers, who will interact with the system because they will actually work for it and, as a consequence, it's beneficial to them that the system, that is the outcome of this project, turns out to be as smooth as possible.

2. Actors identifying

The actors of our system are basically three:

- Guest: a guest is someone who hasn't signed up yet. A guest can either sign up or read about the possible features they will be able to enjoy once they sign up.

- User: a user is someone who has already signed up and therefore can use the service to book a taxi, either through a request or through a reservation.

- Taxi Driver: a taxi driver is an employee of our service who gets paid for driving customers to their desired destinations upon request. Taxi drivers

actively interact with the system by exchanging information about their availability.

3. Requirements

We determined the following requirements according to the Jackson and Zave analysis. Assuming that the domain properties, which are described above, hold for our purposes, we wrote the requirements in order to satisfy the goals.

➢ Users are be able to:

- Sign up into the system
  - The system has to provide a sign up functionality.

- Log into the system
  - The system has to provide a log in functionality.

- Request a taxi
  - The system has to provide a function to allow a user to make a request inside the city.
  - The system shows the user his/her position, that has been determined through gps localization.
  - The system forwards a request that comes from a certain zone to the first taxi queuing in that zone.

- Cancel a request
  - The system has to provide a function to allow a user to cancel a request.
  - The system has to provide a function to show the taxi driver that the call has been cancelled.

- Reserve a taxi
  - The system has to provide a function to allow a user to make a reservation inside the city.
  - The system forwards a reservation to the first taxi queuing in the zone of the origin of the ride, 10 minutes before the meeting time with the user.

- View the reservations
  - The system has to provide a function that allows the user to see their reservations.

- Modifya reservation
  - The system has to provide a function that allows the user to modify a reservation.
  - The system rejects the modification, if it's done less than two hours before the ride.

- Cancel a reservation
  - The system has to provide a function to allow a user to cancel a reservation.
  - The system has to provide a function to show the taxi driver that the call has been cancelled, in case the taxi driver has already been mobilized.

➢ Taxi drivers can:

- Log into the system
  - The system has to provide a log in functionality.

- Inform the system about their availability;
  - The system has to provide a function to store the identifier of the available taxi in the queue of taxis in the corresponding zone.
  - The system has to remove the taxi from the queue if the taxi driver changes his/her status to unavailable.

- Accept a call from the system;
  - The systemhas to remove the taxi from the queue.
  - The system has toset the taxi driver as unavailable.
  - The system has toestimate the waiting time using the information provided by the GPS.
  - The system has to inform the user about the code of the incoming taxi and the waiting time.

- Refuse a call from the system;
  - The system forwards the request to the second taxi driver in the queue.

– The system moves the first taxi in the last position in the queue.

• Inform the system they can no longer take care of a call they'd previously accepted;
  – The system forwards the request to the second taxi in the queue.
  – The system automatically sets the taxi driver as unavailable.
  – The system removes the taxi from the queue.

3.1 Functional requirements

Now that we have defined the main feature of mytaxiService, we can find some functional requirements concerning each defined actor:

➢ Guest: he can
• Sign up.

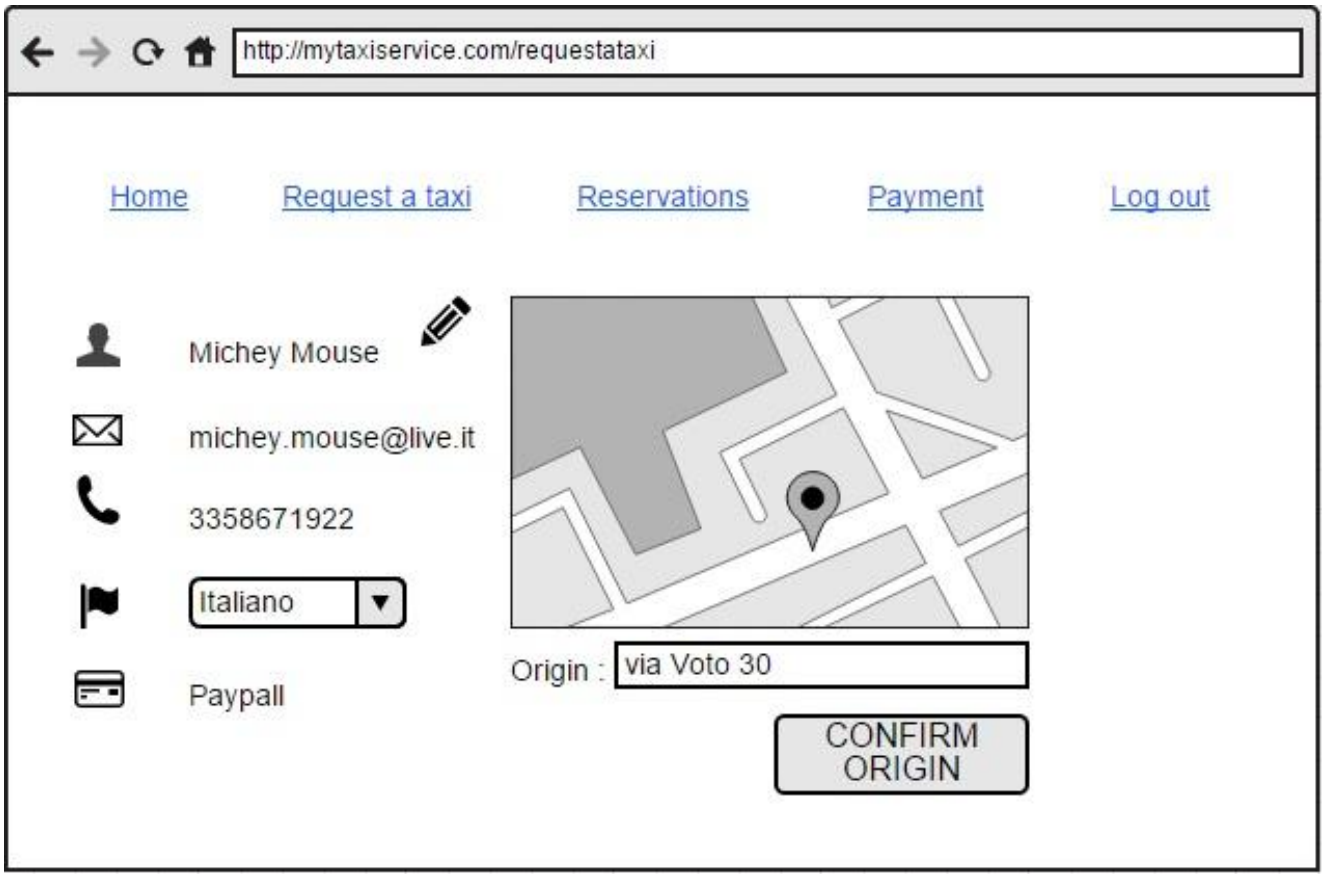
➢ User: he can
• Log in;
• Modify his profile information;
• Request a taxi;
• Cancel a request;
• Reserve a taxi;
• View the reservations;
• Modify a reservation.
• Cancel a reservation.

➢ Taxi Driver: he can
• Log in;
• Inform the system about their availability;
• Accept a call from the system;
• Refuse a call from the system;
• Inform the system they can no longer take care of a call they'd previously accepted.

3.2 Non-functional requirements

3.2.1 User interface

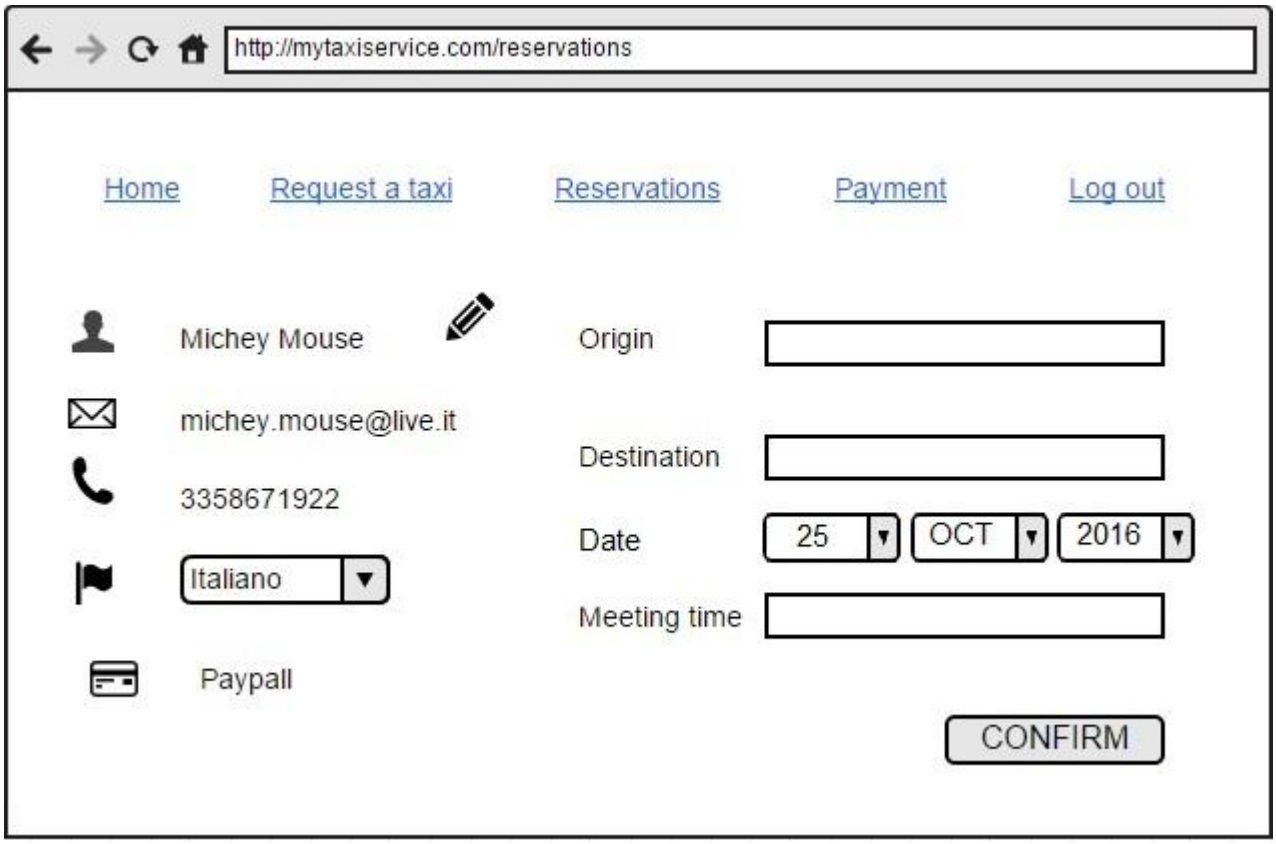
The interface of our application can be used both on a computer and a mobile phone. This sketch is what the user sees when on the Home Page. The user can log out, make a simple request, go on the reservations page (to either create a new one or view an old one), go on the payment page (which is an additional service that allows the user to pay through Paypal). Lastly, the user can confirm his position, which the system manages to estimate thanks to the GPS on the user's phone, to make things faster and smoother for the user.



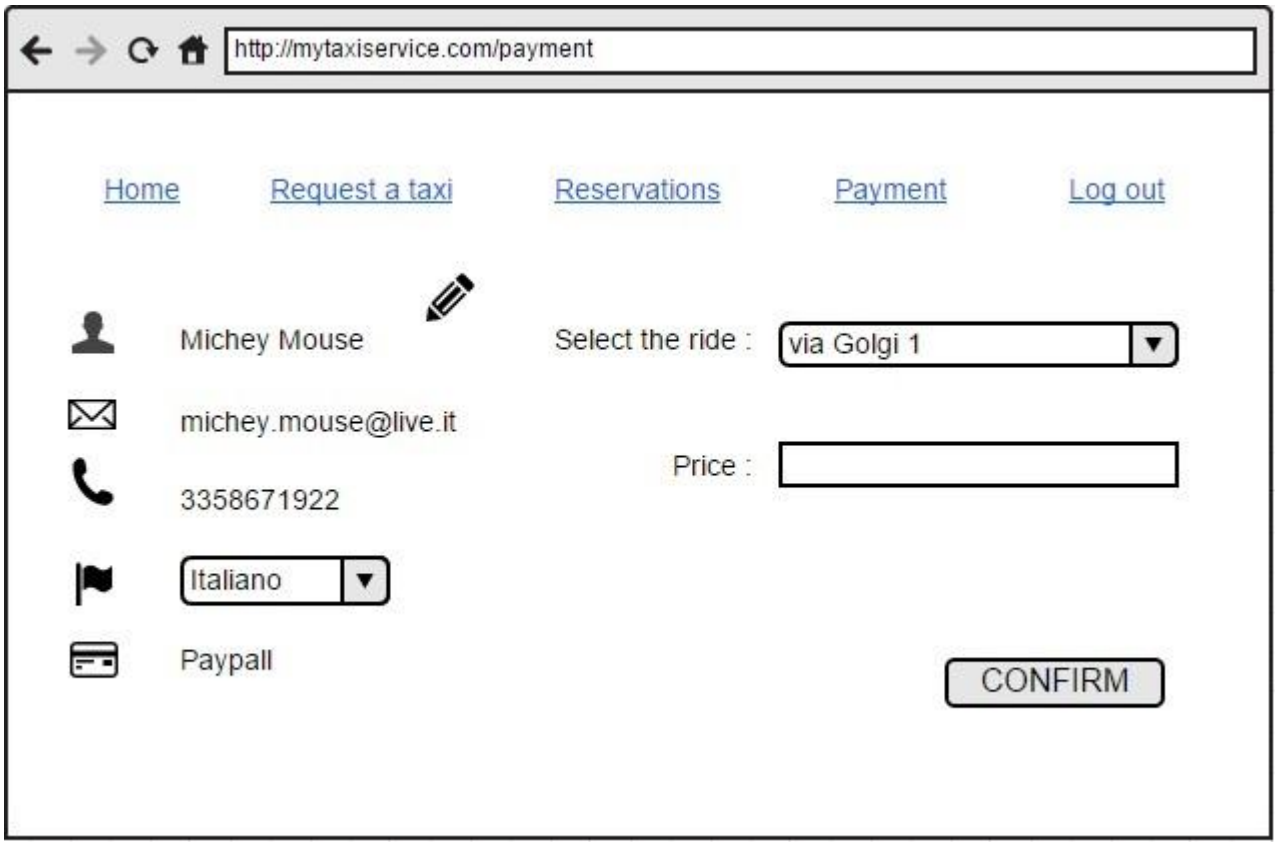

Michey Mouse
michey.mouse@live.it
3516478262
Paypall
Request a taxi
Reservations
Payment
Log out


myTaxiService
via Voto 30
>
CONFIRM ORIGIN

This is the sketch of what the user sees when he goes on the reservations page and opts for making a new reservation. The system asks the user to write in the address of the origin and of the destination, to select the date and time of the meeting. In the end, the user can confirm the operation.



This is the sketch of what the user sees when he clicks Payment. This is an additional service offered by our application which allows the user to pay through PayPal. When on this page, the user can select which ride he/she wants to pay for.



These sketches show 3 screens the taxi driver can see. The one on the left shows what the taxi driver sees, when he/shewants to choose whether he/she is available or not. The one in the middle shows what the taxi driver sees when he/she is asked if he/she wants to accept a call. The one on the right shows what the taxi driver sees when his/her call has been cancelled.

3.2.2 Documentation

We will draft these documents to best organize our work:

- **Project Plan**: to define tasks and to show our organization and the timing of this process.
- **RASD**: Requirement Analysis and Specification Document, to well-understand the given problem and to analyze in a detailed way which our goals are and how to reach them defining requirements and specification.
- **DD**: Design Document, to define the real structure of our web application and its tiers.
- **Installation manual**: explains how to deploy the web site.
- **User manual**: explains how to use the main functionalities of the web application and the mobile application to the user.
- **Taxi driver manual**: explains how to use the main functionalities of the mobile application to the taxi driver.
- **Testing report**: which contains the results of the testing activity performed on a system developed by another group.
- **Project reporting**: which is the result of some analysis done on the project activity.

4. Scenarios identifying

Here are some possible scenarios of the system:

• Bob needs to go to work but his car broke down the previous evening, therefore he decides to go by taxi. After having breakfast, he gets on his computer and logs in on his myTaxiService account to request a taxi, by filling a form where he opts for a simple request. Since he's making a simple request he only has to fill in a form saying the address where he lives. The request is sent to the system, which forwards it to the first available taxi driver in the zone where Bob lives. The taxi driver accepts the call and, as a consequence, Bob receives a confirmation of the request which contains the code of the incoming taxi and the waiting time. Some minutes later, a taxi shows up, Bob notices the code of the taxi is the one he's waiting for, so gets on it and he tells the driver the address of his workplace. The taxi driver takes Bob to work on time and eventually notifies the system that he is available again.

• Alice is on a business trip, she will arrive in the city by train and she will have to stay at a hotel overnight. The train ride will be 4 hours long and she will arrive at the station late at night. Because she knows she will be tired, she wants to make sure she will find a taxi waiting for her right outside the station. Therefore, as soon as she gets on the train, she uses her phone to log in on her myTaxiService account to make a reservation. She fills in all the necessary fields, setting the station as the origin of the ride, the hotel as the destination, and specifying she wants the meeting time to be shortly after the train gets to the station. Around 4 hours later, just before getting off the train, the system forwards the reservation to the first available taxi driver near the station and Alice gets a notification which tells her the code of the taxi that will pick her up. She meets the taxi driver just outside the station and he drives her to the hotel. Right after this, the taxi driver is ready to accept requests again, therefore he notifies the system he’s available.

• On a weekday Carl and his girlfriend are both stressed and tired from work and they agree on spending some relaxing time together during the weekend, therefore Carl suggests to go for a picnic on the beach this Saturday. Carl doesn't want to go to the beach by car because he knows finding a car park will stress him even more, so he goes on myTaxiService and makes a reservation for a taxi on Saturday morning. Friday night, right before going to sleep, Carl watches the weather forecast on TV, which warns of an incoming heavy storm that is going to make the weather rainy all through the weekend. Knowing that he will have to cancel his plans, Carl goes on myTaxiService application and cancels the reservation he had made a few days earlier.

• John needs to go to the mall by taxi, so he requests a taxi on myTaxiService. The system checks the queue associated to the taxi zone where John lives and finds the first available cab in the queue, which is number 358. Mark, the taxi driver driving cab number 358, accepts the call and John receives a confirmation from the system which says a cab will be there to pick him up in 10 minutes. A couple of minutes later, Mark hears a noise coming from his cab and he notices that he got a flat tyre. Because of this, he decides he should give up the call and let the system assign the call to another cab, so he notifies the system of what's happened. The system assigns the call to the next available taxi driver and moves Mark to the last position in the queue. After doing this, the system also notifies John that there will be a delay of about 10 minutes; moreover John receives the code of the new cab that is coming to pick him up. Eventually, the new cab arrives and takes John to the mall.

• Sarah wants to go to the cinema with her friend but she cannot drive. She calls her friend asking him for a lift but he says he will walk to the cinema and therefore he won't be able to come pick her up. Sarah decides to request a taxi through myTaxiService and, a minute later, she gets a confirmation saying a taxi will come in 15 minutes. As she is getting ready to go out, she receives a call from her friend, who says he has changed his mind and he decided to come pick her up instead. Sarah thanks him and goes on her myTaxiService account to cancel the request she'd previously made. As a consequence, the system sets the taxi driver that had been mobilized as available and notifies him that he no longer needs to go to the meeting place because the client has cancelled the request.

• Rocco, a taxi driver, has just driven a client to his destination and notifies the system that he is available again. Sometime later, he notices he's low on petrol and he decides to go to a petrol station. The systemreceives a new call,which comes from the zone where Rocco is, and thus it decides to forward the call to him. Rocco realises he won't make it in time and so he doesn't answer the call. As soon as the timer runs out, the system understands that Rocco cannot accept the call, therefore it moves him to the last position of the queue and forwards the call to the next available taxi driver.

• Artyom is a boy who lives in Kazakhstan and he has a great passion for taxis. He often installs apps and games about taxis, but unfortunately he can barely speak English, so one day he accidentally downloads the myTaxiService app mistaking it for a game and ends up signing up for it. He proceeds to select random features and eventually makes a reservation without realising the app is actually not a game. When prompted to choose the address of the origin of the ride, he writes in the address of his own house in Kazakhstan. The system knows the address is not inside the city and therefore rejects the operation. Artyom, puzzled as to what just happened, closes the app and decides to never open it again.

• David's dog doesn't feel too well this morning so he decides to take it to the vet, but the vet nearby has closed down so he needs to go to another vet far away from where he lives. He looks up the address of the veton Google to make a reservation for the afternoon. Unfortunately, David has lost his glasses and he misreads the address as via Topo Gigio, whereas it's actually via Tortini.Later on, after making the reservation with the wrong address, he realises that the name of the street actually sounds strange to him, and therefore he decides to double check it on Google. In fact, he finds out he had got the name wrong and he proceeds to modify the reservation by changing the address of the destination of the ride to via Tortini. The modification is allowed and accepted by the system because it's made two hours before the ride. A couple of hours later, a taxi comes pick up David and takes him to the vet.

• Mary and Jane want to go to a club tonight and they decide to meet there at 11pm. Mary intends to go there by taxi, so she goes on her myTaxiService account and makes a reservation for tonight. Several hours later, when Mary gets out of the shower, she calls Jane to chat with her to talk about what they will wear, but they end up fighting about what to wear because they find out they're going to wear the same dress. Mary gets so angry that she cancels the meeting with Jane and completely forgets it's 11pm already and the taxi is waiting for her. As soon as she realizes it, she goes on her myTaxiService account and cancels the reservation. The system notifies the taxi driver that the reservation has been cancelled and the taxi driver sets his own status as available.

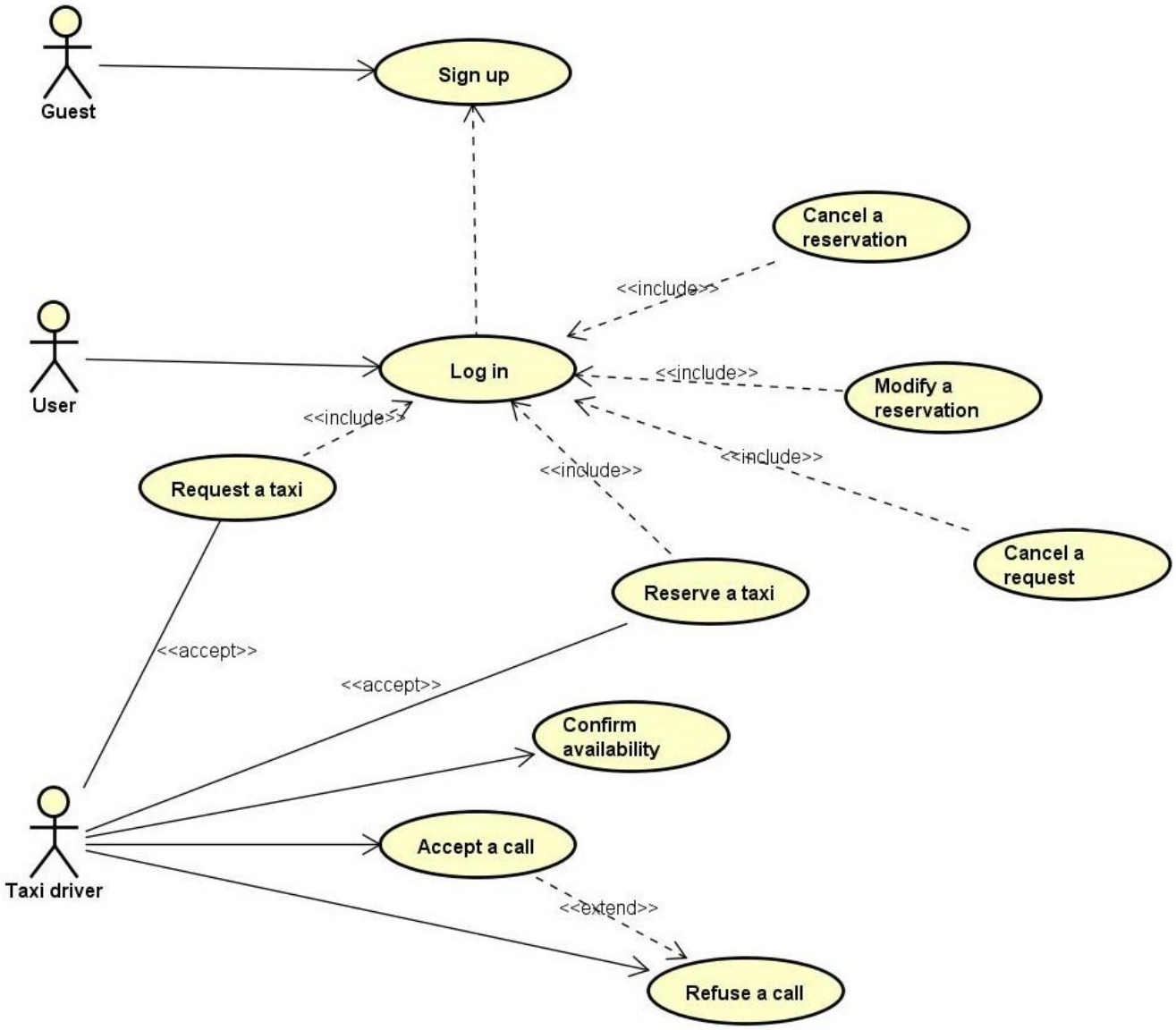
5. UML Models

5.1 Use Case Diagram

We can derive some use cases from the scenarios identified in the previous paragraph:

- Sign up;
- Log in;
- Request a taxi;
- Cancel a request;
- Reserve a taxi;
- Modify a reservation;
- Cancel a reservation;
- Inform about the availability.

Here is a Representation of the Use Case Diagram:



5.2 Use Case Description

We want to describe the use cases that we derived from the scenariosin a detailed way and to try to define them all.
It is important to understand that all the references to “pages”, “buttons” or “input forms” are only hypothesis to make the situation as clear as possible so as to help the reader to draw a visual picture in his/her mind of what we plan to do, but the real structures will be well defined in the Design Document.

5.2.1 Sign up

Name	Sign up.
Actors	Guest.
Entry Conditions	The guest isn’t registered to myTaxiService.
Flow of events	• The guest goes on the website or the mobile app. • The guest clicks/touches the “SIGN UP” button. • The guest fills in the form where he has to write: -Name -Surname -Email -Password -Telephone number • The guest clicks/touches the “DONE” button. • The system shows him his personal page.
Exit conditions	Registrationsuccessfullydone.
Exceptions	An exception can be caused if the format of the e-mail or the telephone number is not correct or the guest inserts an already existing e-mail/telephone number or if some fields aren’t filled.

5.2.2 Log in

Name	Log in.
Actors	User.
Entry Conditions	User has successfully signed up on the system.
Flow of events	• The user goes on the web site/mobile app. • The user fills in the text fields in the home page with the e-mail and password. • The user clicks/touches the “LOG IN” button.
Exit conditions	The system shows the user the “Request taxi” page.
Exceptions	The password and/or e-mail inserted by the user are wrong.

5.2.3 Request a taxi

Name	Request a taxi.
Actors	User, Taxi driver1, Taxi driver2.
Entry Conditions	User clicks/touches the “REQUEST A TAXI” button.
Flow of events	• The system shows to the user the origin address taken by GPS coordinates or fills in the text fields in the page the address of the origin of the ride. • The user clicks/touches the “CONFIRM ORIGIN” button. • The system forwards the request to Taxi driver 1 queuing in the user’s zone. • Taxi driver 1 accepts the call. • The system removes Taxi driver 1 from the queue. • The system estimates the time of arrival. • the system sends a confirmation to the passenger.
Exit conditions	The system shows the user the taxi code and the waiting time.
Exceptions	The inserted address is not inside the city.
Alternative flow	• Taxi driver 1 doesn’t accept the call. • The system moves Taxi driver 1 to the last position in the queue. • The system forwards the request to Taxi driver 2 in the queue. • Taxi driver 1 accepts the call. • The system removes Taxi driver 2 from the queue. • The system estimates the time of arrival. • the system sends a confirmation to the passenger.

5.2.4Cancel a request

Name	Cancel a request.
Actors	User, Taxi driver.
Entry Conditions	User clicks/touches the “CANCEL REQUEST” button.
Flow of events	• The user clicks on “CANCEL REQUEST” button. • The system notifies the taxi driver that the call has been cancelled.
Exit conditions	The system notifies the taxi driver that the call has been cancelled.
Exceptions	None.

5.2.5Modify a reservation

Name	Modify a reservation.
Actors	User.
Entry Conditions	User clicks/touches the “Reservations” button and then User clicks/touch the “Modify” button next to the reservation.
Flow of events	• The user modifies the reservation. • The system notifies the user the modification was successful.
Exit conditions	The system notifies the user the modification was successful.
Exceptions	The user writes in the origin and/or destination field an address that is not inside the city. The user tries to modify the meeting time by changing it to an earlier time.

5.2.6Reserve a taxi

Name	Reserve a taxi.
Actors	User, Taxi driver.
Entry Conditions	User clicks/touches the “RESERVATIONS” button and then user clicks/touches the “NEW RESERVATION” button.
Flow of events	• The user writes the address of the origin and the destination of the ride in the text fields in the page. • User clicks/touches the “CONFIRM” button. • the system sends a confirmation to the passenger. • The system forwards the request to the first taxi driver queuing in the user’s zone 10 minutes before the meeting time with the user. • The taxi driver accepts the call.
Exit conditions	The taxi driver accepts the call.
Exceptions	The user writes in the origin and/or destination field an address that is not inside the city. The user tries to set the meeting time too early.

5.2.7Cancel a reservation

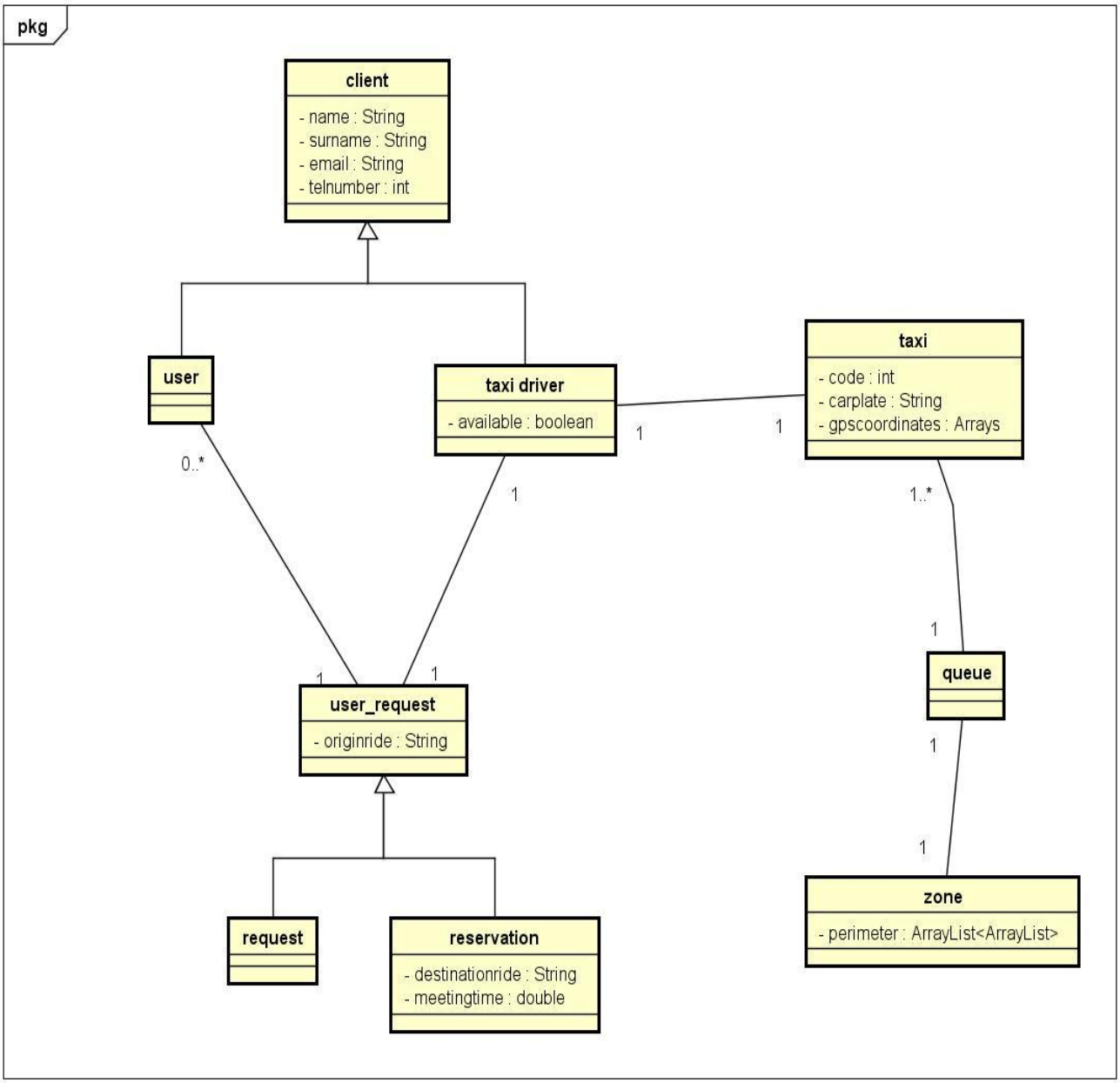
Name	Cancel a reservation.
Actors	User, Taxi driver.
Entry Conditions	User clicks/touches the “Reservations” button and then User clicks/touches the “Cancel” button next to the reservation.
Flow of events	• The user cancels the reservation. • The system hasn’t called the taxi driver yet. • The system notifies the user about the successful cancellation of the reservation.
Exit conditions	The system notifies the user about the successful cancellation of the reservation.
Exceptions	None.
Alternative flow	• The user cancels the reservation. • The system realises it has already called the taxi driver. • The taxi driver notifies the taxi driver about the cancellation.

5.2.8Inform about the availability

Name	Inform about the availability.
Actors	Taxi driver.
Entry Conditions	The taxi driver goes on the web mobile app and do the log in.
Flow of events	• The taxi driver touches the “AVAILABLE” button. • The system adds him to the queue of the corresponding zone. • The system makes the “AVAILABLE" button unavailable, preventing the taxi driver from touching it again. The system enables the “NOT AVAILABLE” button.
Exit conditions	The system notifies the taxi driver that his status has been successfully changed to available.
Exceptions	None.
Alternative flow	• The taxi driver touches the “NOT AVAILABLE” button. • The system removes the taxi driver from the queue. • The system makes the “NOT AVAILABLE" button unavailable, preventing the taxi driver from touching it again. • The system enables the “AVAILABLE” button.

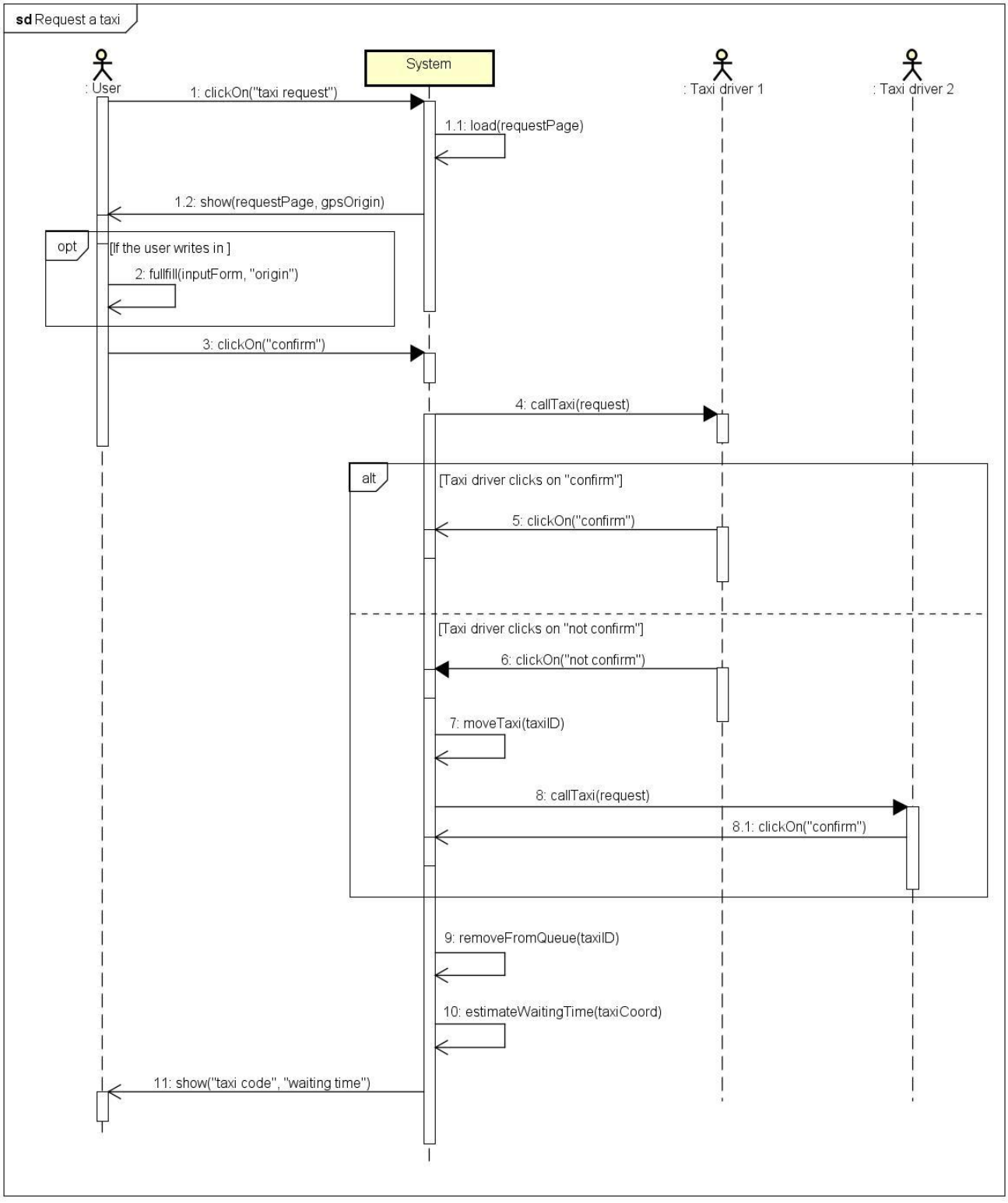
5.3 Class Diagram

Now that we have refined every use case we can draw a class diagram of our system:

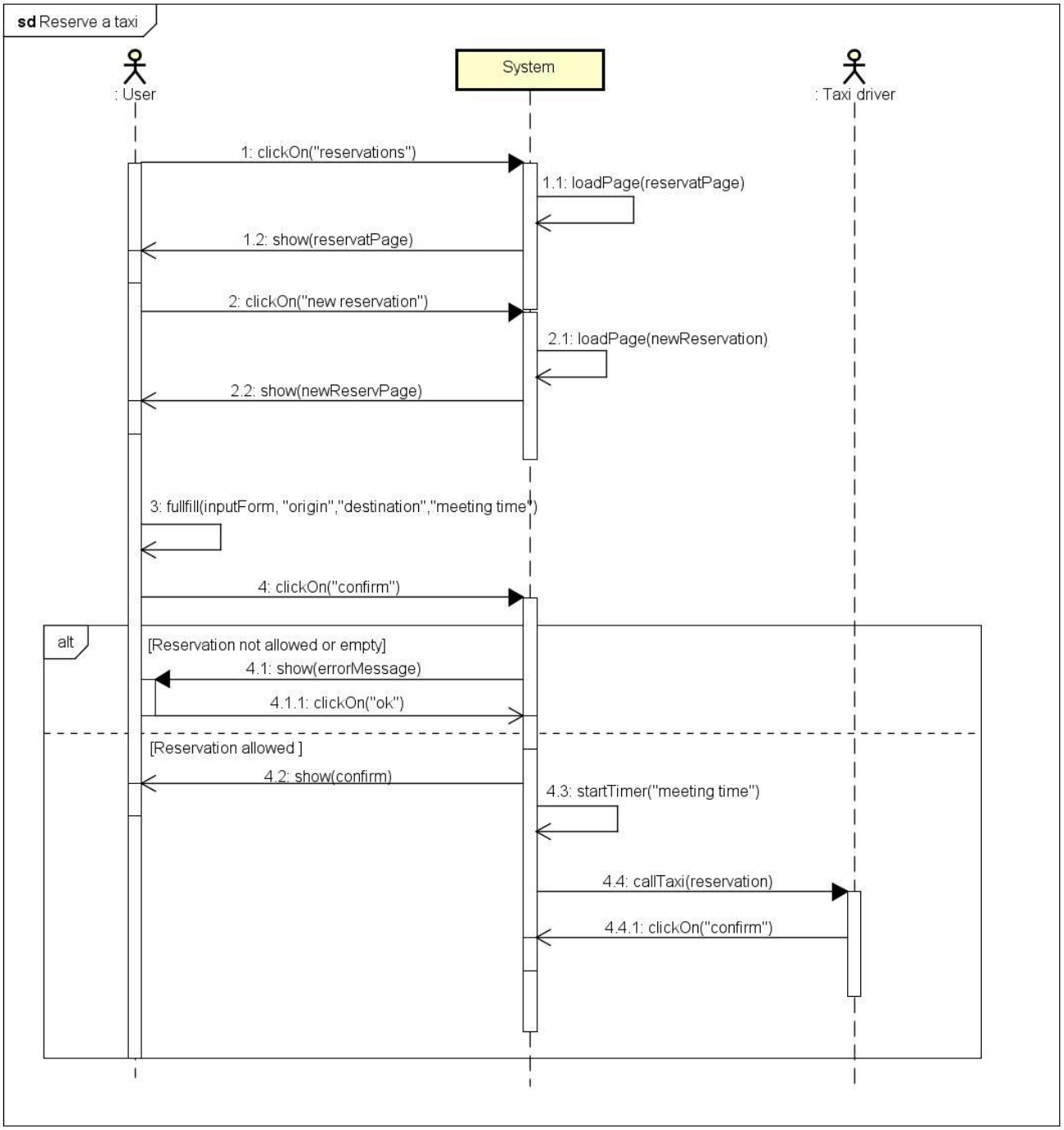


5.4 Sequence Diagrams

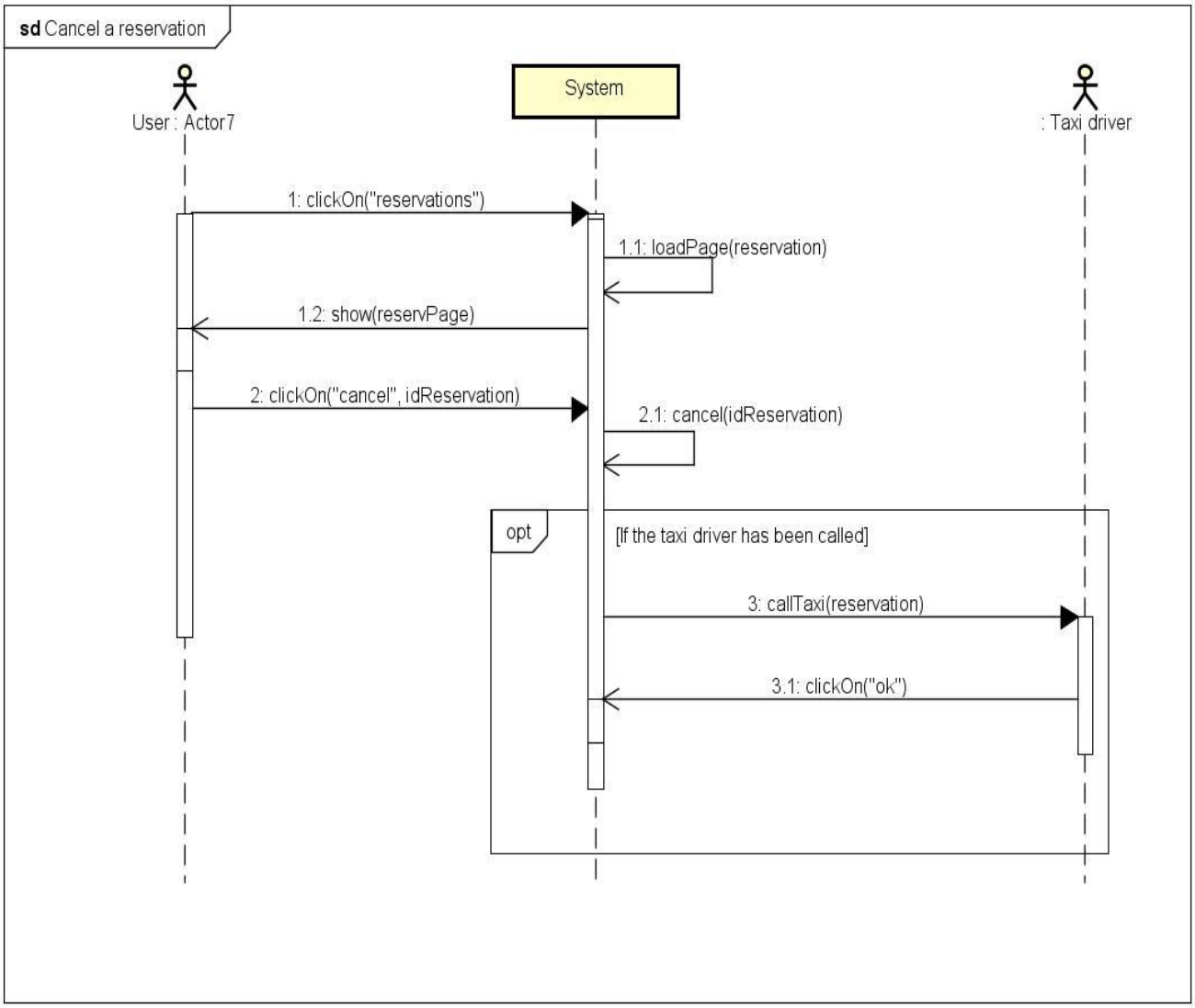
5.4.1 Request a taxi



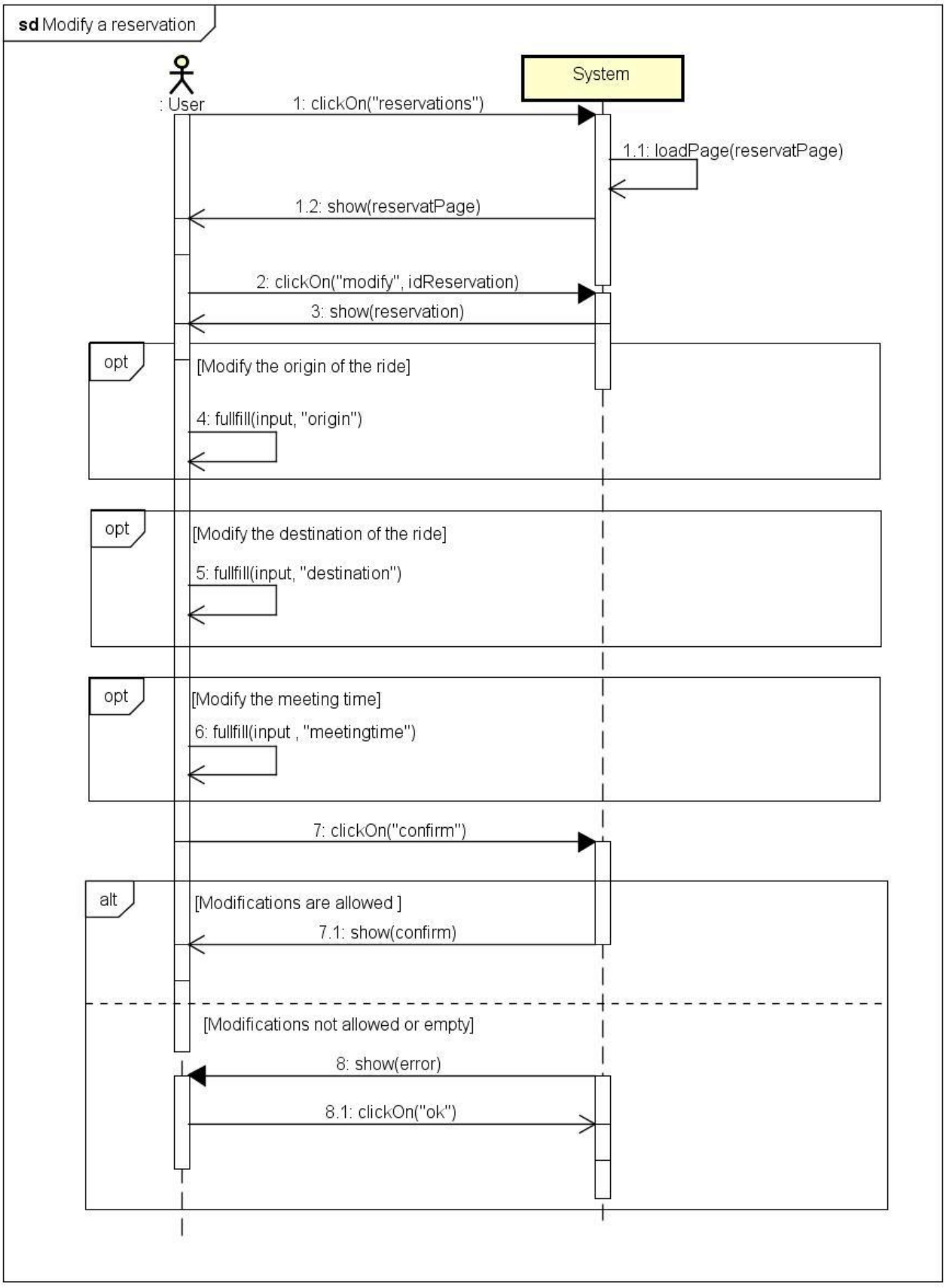
5.4.2 Reserve a taxi



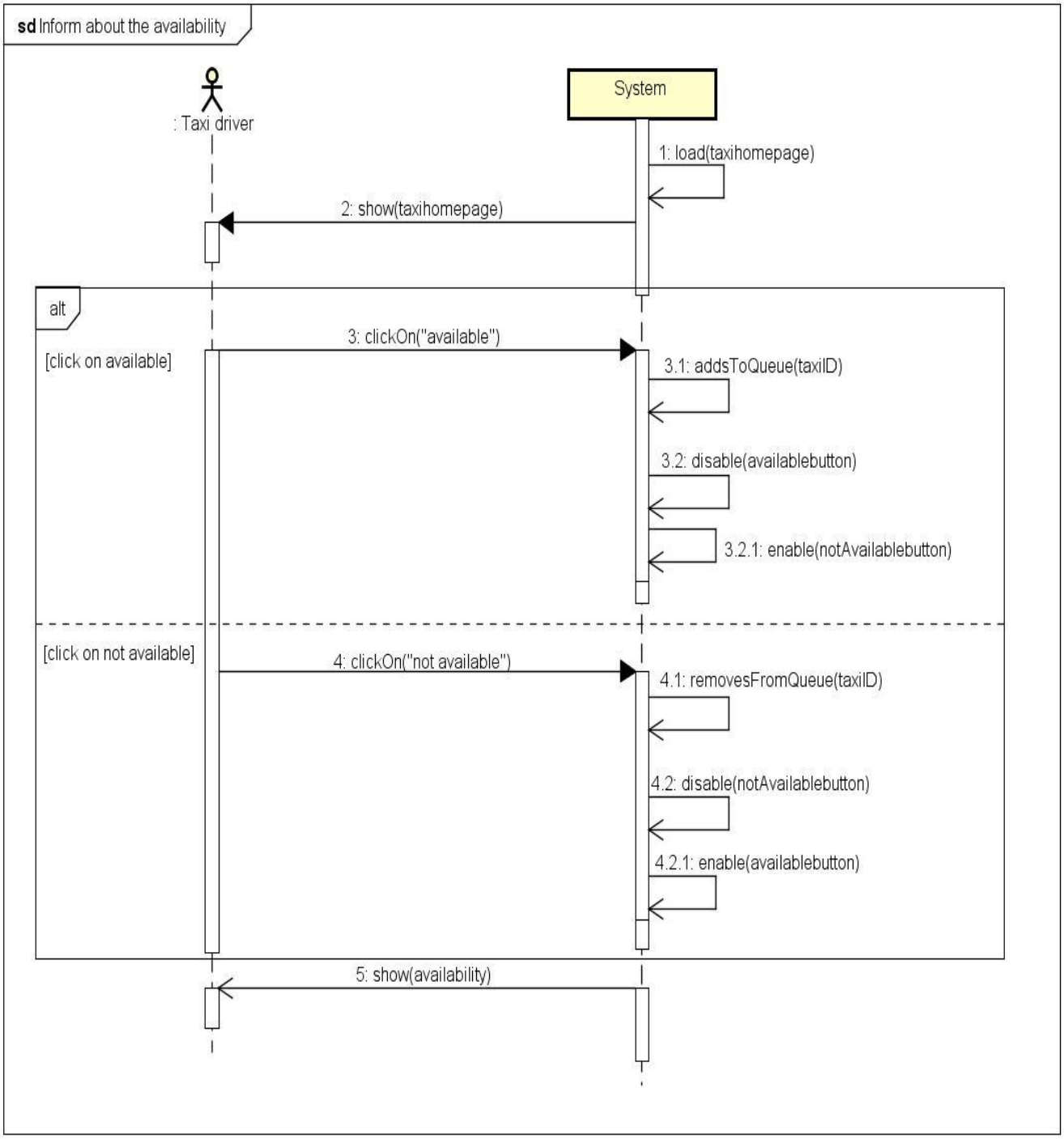
5.4.3 Cancel a reservation



5.4.2 Modify a reservation

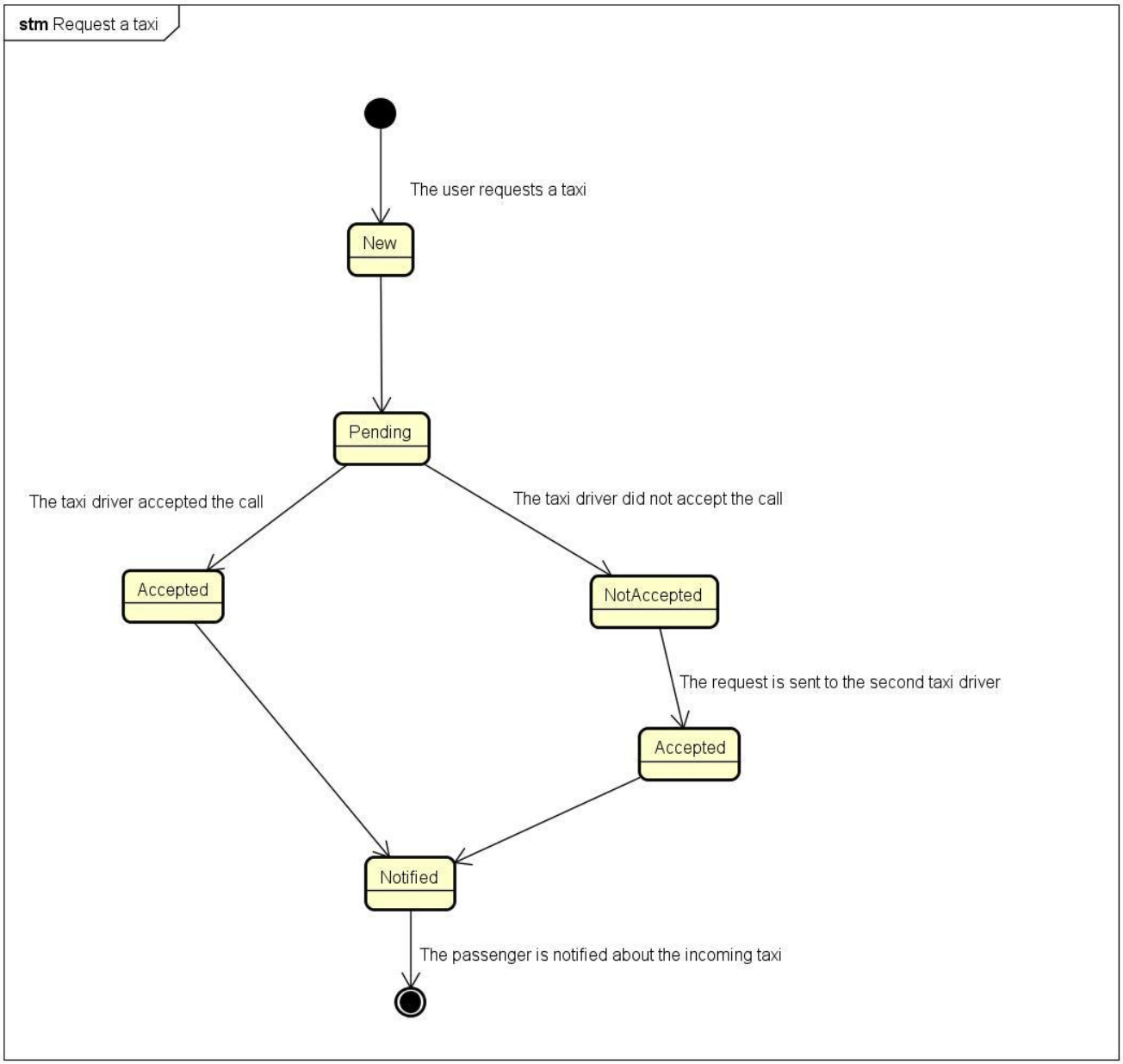


5.4.2 Inform about the availability

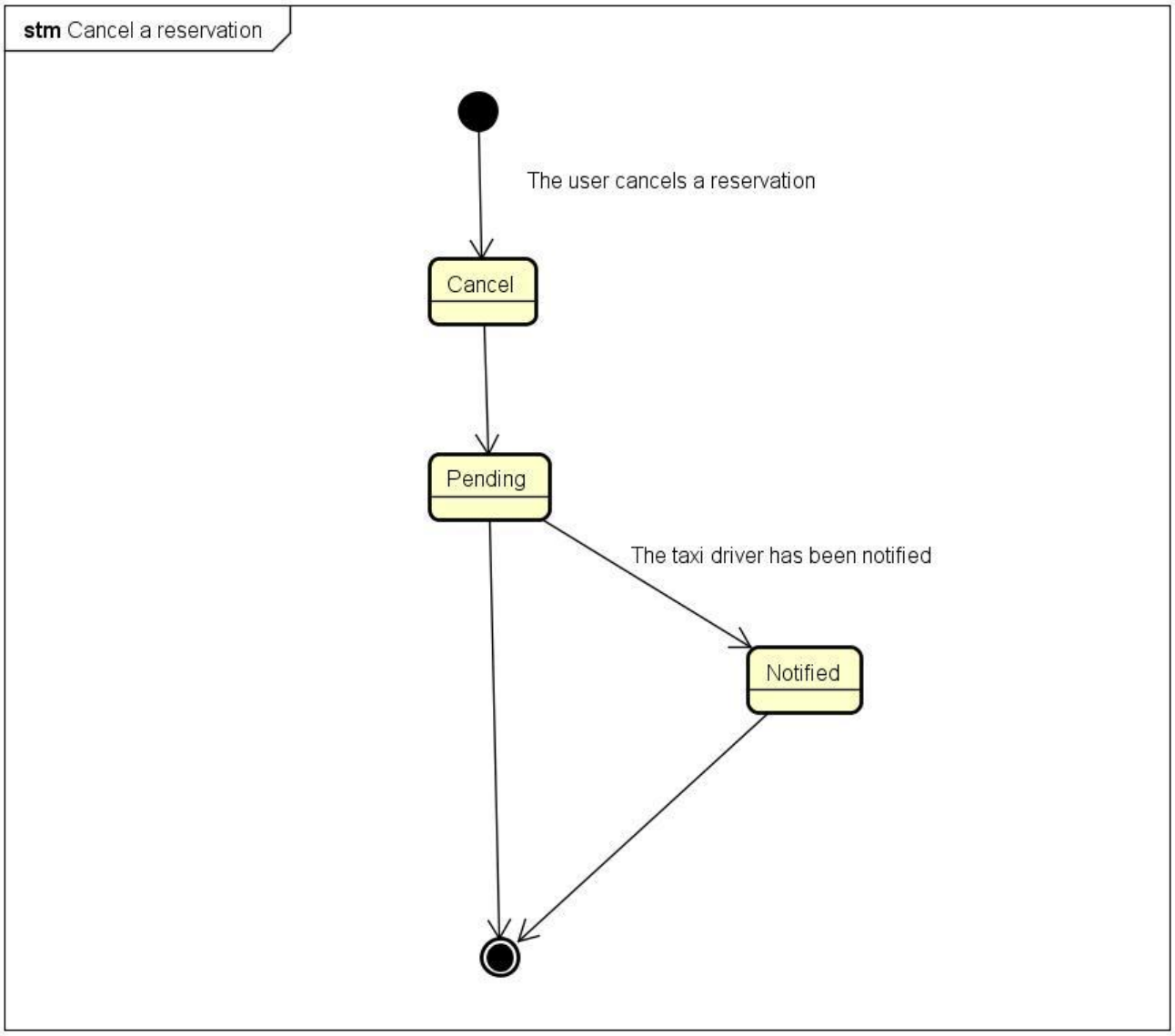


5.5 State chart Diagram

5.5.1 Request a taxi



5.5.2 Cancel a reservation



6. Alloy Modelling

We used Alloy Analyzer to specify the properties of our application, mostly getting ideas from the representation of the Class Diagram. In this paragraph we reported the alloy code used for the analysis and some examples of worlds generated with the analyzer.

6.1 Alloy Code

```
//SIGNATURES

sig Passenger{
	request: one Request
}

sig Request{
	passenger: one Passenger,
}

sig TaxiDriver{
	request: lone Request,
	available: oneBool,
	taxi: one Taxi
}

sig Taxi{
	queue: lone Queue,
	coordinates: one Coordinates
}

sig Zone{
	queue: one Queue,
	coordinates: one Coordinates
}

sig Queue{
}

sig Coordinates {}

abstract sigBool{}
```

```
onesig True extendsBool{}
one sig False extendsBool{}

//FACTS

//Every taxi is driven by exactly one taxi driver
factOneTaxiPerTaxiDriver {
        all t: Taxi | (one d: TaxiDriver | d.taxi = t)
}

//Every queue is assigned to exactly one zone
factOneQueuePerZone {
        all q: Queue | (one z: Zone | z.queue = q)
}

//Each request can only be made by one single passenger
factPassengersAreOnlyAssociatedWIthOneRequest{
        nodisjoint p1,p2: Passenger | p1.request = p2.request
}

//Each passenger can only have one single request assigned to them
factPassengersAreOnlyAssociatedWIthOneRequest{
        nodisjoint r1,r2: Request | r1.passenger =r2.passenger
}

//The only request associated with a certain passenger, is the one he/she made
fact{
        no disjoint r1, r2: Request | some p: Passenger | r1.passenger=p
&&p.request=r2
}

//A taxi driver is available if and only if he/she hasn't accepted any requests
factTaxiDriverAvailability{
        all t: TaxiDriver | some v: True | (#t.request=0 implies v int.available) and
(v int.availableimplies #t.request=0)
}



//A taxi driver is unavailable if and only if he/she has accepted a request
factTaxiDriverUnavailability{
```

```
	all t: TaxiDriver | some f: False | (#t.request=1 implies f int.available) and
(f int.availableimplies #t.request=1)
}

//If a taxi driver is unavailable then his/her taxi is not in a queue.
factUnavailableTaxiDriversCannotBeInAQueue{
	all t: TaxiDriver| some f: False | f int.availableimplies #t.taxi.queue=0
}

//If a taxi driver is available, then his/her taxi is in a queue
factAnAvailableTaxiDriverMustBeInAQueue {
	all t: TaxiDriver| some v: True| v int.availableimplies #t.taxi.queue=1
}

//If a taxi is in a queue, the queue is the one associated with the zone where the
taxi is
facttestqueue{
all t: TaxiDriver| one z: Zone| some v: True| v int.availableimplies
(t.taxi.coordinates = z.coordinatesandt.taxi.queue = z.queue )
}

//There must be no coordinates that are not associated with any zone
//and the number of zones must be equal to the number of coordinates
factThereAreNoCoordinatesWithNoCorrespondingZone {
	all c: Coordinates | one z: Zone | c = z.coordinatesand #Zone =
#Coordinates
}

//ASSERTIONS

assertNoUnavailableTaxiDriversInAQueue{
	no t: TaxiDriver | some f: False | f int.available&& #t.queue=1
}

checkNoUnavailableTaxiDriversInAQueue

assertNoZonesThatShareASingleQueue{
	no disjoint z1,z2: Zone| z1.queue=z2.queue
}

checkNoZonesThatShareASingleQueue
```

```
assertNoRequestWithoutPassenger {
	no r: Request | #r.passenger = 0
}

checkNoRequestWithoutPassenger

//PREDICATES

//Show a world where taxi driver t1 hasn't accepted any requests and therefore
is available
//and taxi driver t2 has accepted a request and therefore is not available
predshowAvailability (disjoint t1,t2: TaxiDriver, v: True, f: False) {
	#TaxiDriver = 2
	#t1.request = 0
	v in t1.available
	#t2.request = 1
	f in t2.available
}
runshowAvailability

//Show a world where taxis located in distinct zones, belong to distinct queues
//(the queues assigned to the zone where they are)
predshowTaxiLocalization(disjoint  t1,t2: TaxiDriver, disjoint  z1,z2: Zone){
	t1.taxi.coordinates!=t2.taxi.coordinates
}

runshowTaxiLocalization

pred showCoordinates () {
	#Coordinates=3
	#Zone = 3
	#Request = 1
}
run showCoordinates
```

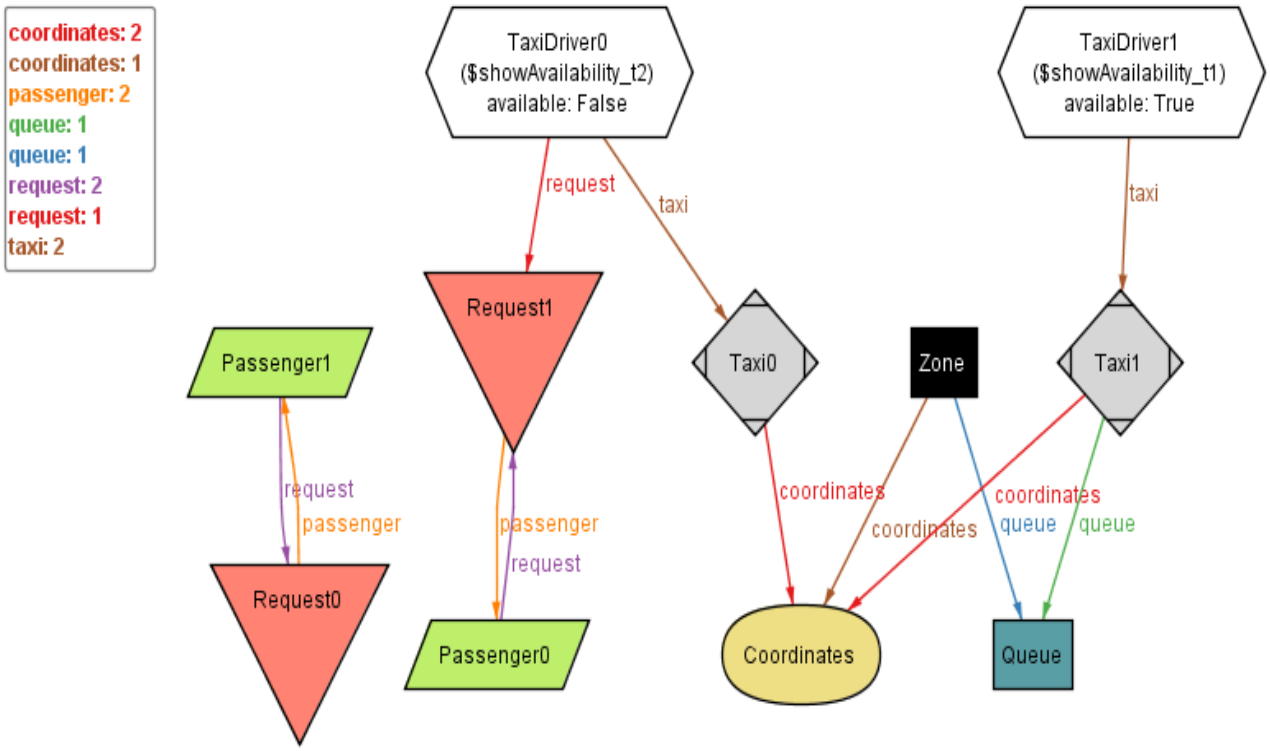
This is the report we got in Alloy Analyzer:

```
6 commands were executed. The results are:
  #1: No counterexample found. NoUnavailableTaxiDriversInAQueue may be valid.
  #2: No counterexample found. NoZonesThatShareASingleQueue may be valid.
  #3: No counterexample found. NoRequestWithoutPassenger may be valid.
  #4: Instance found. showAvailability is consistent.
  #5: Instance found. showTaxiLocalization is consistent.
  #6: Instance found. showCoordinates is consistent.
```

6.2 Alloy Worlds

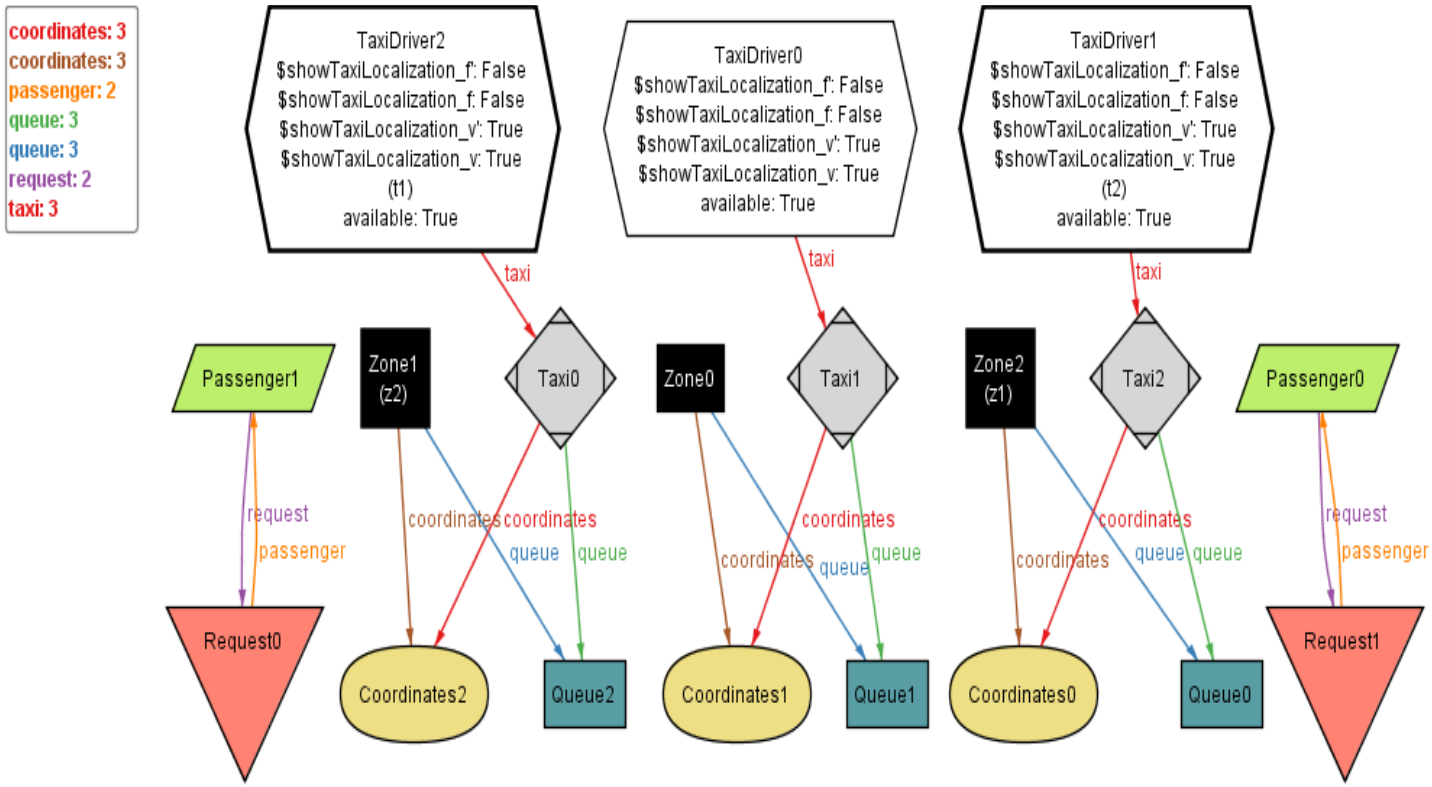
6.2.1 Availability

The following world is a general world generated with the analyzer. It was generated using the predicate showAvailability. As we can see in the picture, the first taxi driver hasn't accepted any requests and therefore is available, and the second taxi driver has accepted a request and therefore is unavailable. Moreover, only the available taxi driver belongs to a queue. Lastly, the first request is not associated with a taxi driver, which means it is actually a reservation and it currently hasn't been forwarded to a taxi driver yet.



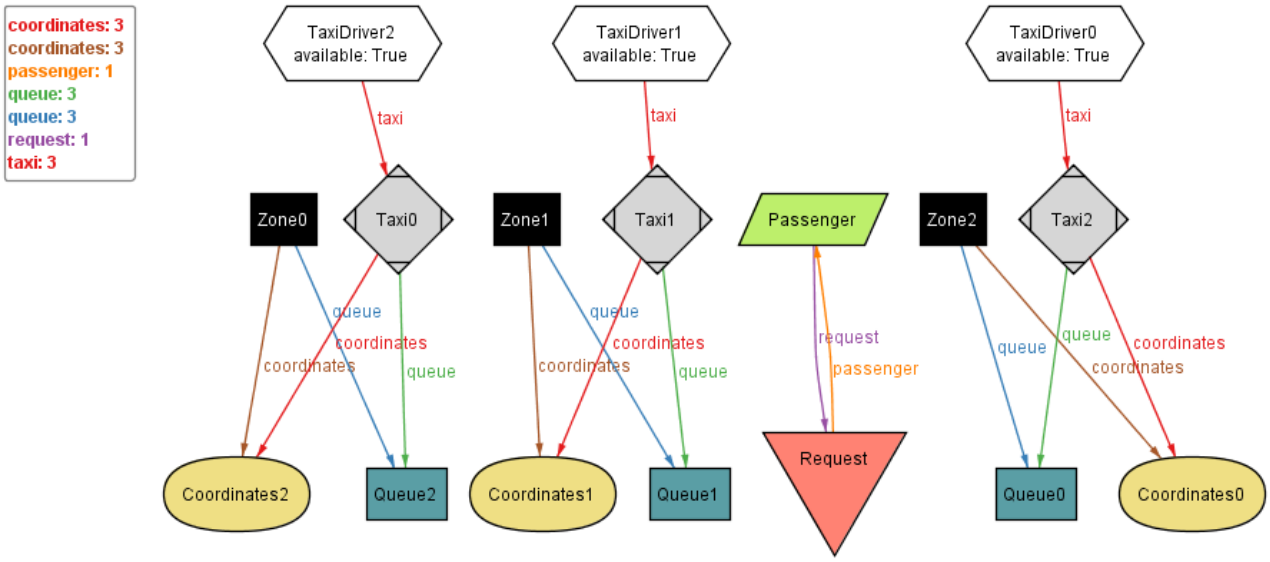
6.2.2 TaxiLocalization

The following world shows a situation in which there are distinct taxis in distinct zones. Because they are in distinct zones, they belong to different queues, so that each taxi belongs to the queue associated with the zone where it currently is. We know that the system is aware of the locations of the taxis (thanks to the GPS) and can correctly assign taxis to the right queue. This world was generated using the predicate showTaxiLocalization.



6.2.3 Coordinates

This world was created with no specific requirements, except the existence of three coordinates. It was generated using the predicate showCoordinates.



7. Used tools

The tools we used to create the RASD document are:

- Microsoft Office Word 2010: to redact and to format this document;
- Astah: to create the Use Case Diagram, the Sequence Diagrams, the Class Diagram and the State Charts;
- Alloy Analyzer 4.2: to prove the consistency of our system;
- Moqups: to create the UI sketches.