POLITECNICO
MILANO 1863

Software Engineering 2 A.Y. 2015-16

Requirements Analysis and Specification Document

for

MyTaxiService

by Francesco Picciotti (Matr. 854021)
and Marco Travaglini (Matr. 859186)

Table of Contents

1. Introduction

1.1 Purpose

The intent of the following document is describing properly the features of a software called MyTaxiService in an unambiguous way, showing off the whole system's requirements, such functional and non-functional ones, constraints, software's goals and modeling it with use cases, sequence diagrams and alloy, to proof the model consistency.
Moreover, here it is explained its usage and scenarios in order to provide a more detailed idea of what the software may be.
The document will be read as landmark for all that are going to be concerned with the developing, designing or integrate the software, because of the following document not only points out all project's details, reveals basic requirements and goals but also establish a contract between the customer and the developer.
*From the customer's point of view this is **what** he wants the software to do, from developer's one is **how** the software has to be.*

1.2 Product Scope

The whole system that is here described will be available via web application or mobile app, which will be used by users and taxis both.
The product's aim is offering a platform that covers a city's areas, giving the city's government the chance for better managing cabs and improving the service's quality through a modern and real-time system, which uses internet and GPS data for speeding-up the process of looking for a taxi or booking it.
Moreover providing this new service, not only it will lighten the other channels for contacting cabs like via mobile phone but it will even simplify the interaction between customer and cab's drivers.
The user will be able to ask for a ride or to reserve one for certain hour and on the other hand, the taxi's driver will be able to show his availability and confirm or decline the ask for a ride.

1.3 Definition, acronyms, abbreviations

- *User: general term used for stating a customer or driver*
- *Driver: user that has an account as taxi driver*
- *Customer: user that looks for taxi's rides or reservations*
- *Driver's availability: it means the taxi driver is working, free and potentially able to offer a ride*
- *State: A taxi can be in two state: available or busy*
- *FIFO: the policy through all queues are managed*
- *Area: A part in which the city is split, it's about 2 km^2 wide*

1.4 Reference documents

- *Specification document : MyTaxiService AA 2015/16.pdf*
- *IEEE Standard 830-1998 IEEE Recommended Practice for Software Requirements Specifications.*
- *IEEE Standard 1016 tm -2009 Standard for Information Technology-System Design-Software Design Descriptions.*
- *Rasd Meteocal example 2.pdf (past year's project).*

1.5 Overview

- *First Section: Introduction, it provides a general information about the system, its main future usage, the high-level features and other things like acronyms and abbreviations that will be used in the document and all the references.*
- *Second Section: Overall Description, here the document points out details about software's main goals, constraints and assumption imposed by the environment and the internal and external stakeholders even providing a representation of the system by the Jackson & Zave's model and certain common real situation with scenarios.*
- *Third Section: Specific Requirements, in this part all the software's requirements, both functional and non-functional, are exposed and widely discussed. Moreover, it shows system's model through UML diagrams, such use cases and sequence diagrams, and Alloy.*
- *Fourth Section: Alloy*
- *Fifth Section: Appendix*

2. Overall Description

2.1 Product Perspective

The system created will not replace an existing one and will not integrated with others but it will provide a programmatic interface in order to let develop new services in the future.
Moreover, it will be available both via web application and via mobile app, providing only two user interfaces, one for the customer and one for the taxi driver but any internal one for administration.

2.2 Product Functions

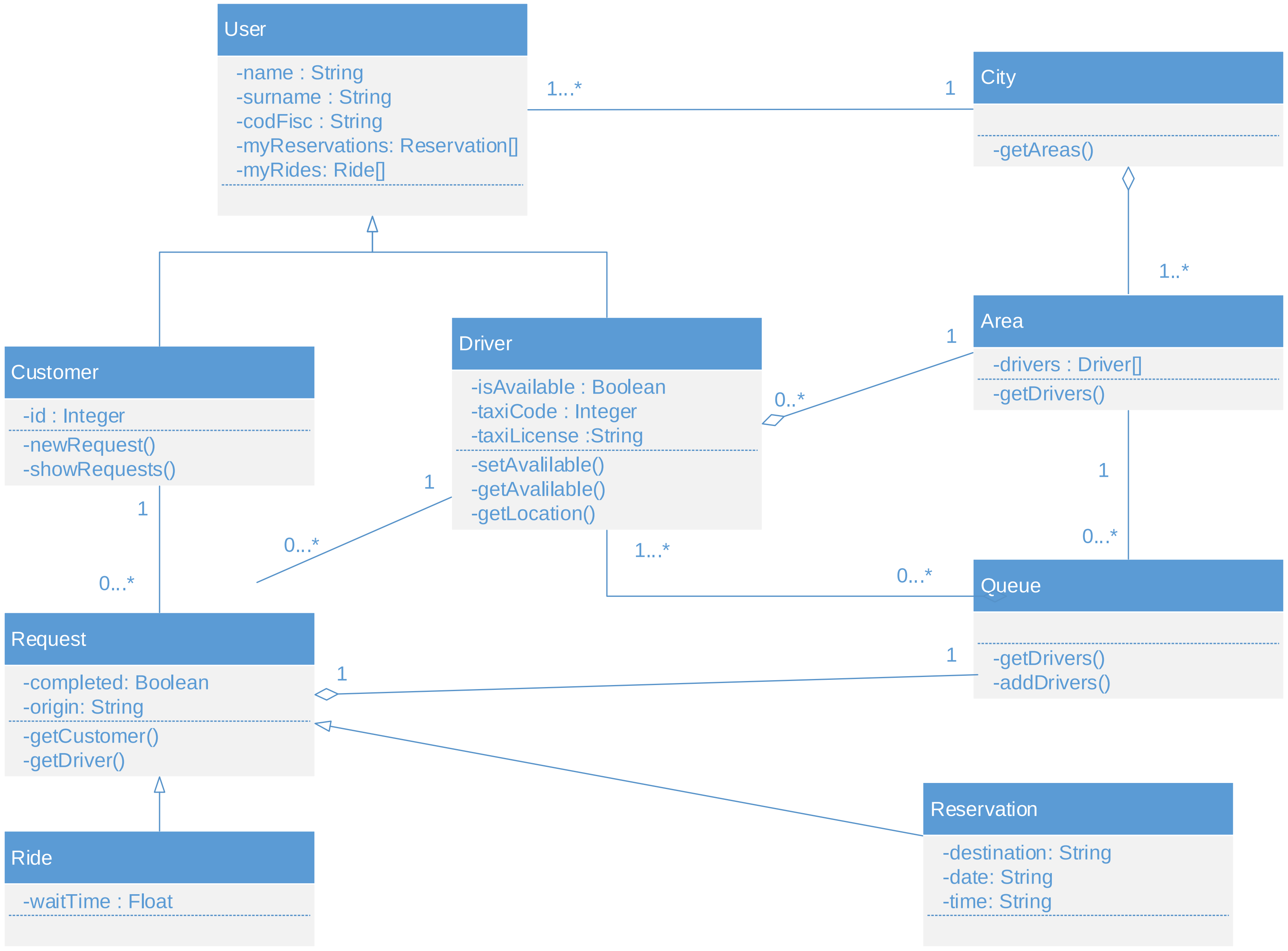
2.2.1 Goals:

Here are described the software's high level goals:

G1. Allow visitor to sign up as customer
G2. Allow visitor to sign up as driver
G3. Allow user to log in
G4. Allow customer to ask for a ride
G5. Allow customer to book a reservation for a certain date
G6. Allow customer to see the reservation's confirmation
G7. Allow customer to view the driver's confirmation to his request for a ride
G8. Allow driver to communicate and change his availability
G9. Allow driver to view and reply to a request for a ride
G10. For every customer's request, the system will create and manage a queue to which belong all the drivers in the request's area.

2.2.2 UML Diagram:

The following is the object class diagram at high-level:



2.3 User Characteristics

The software's users are taxi drivers or customers that are the main actors of the application. Intended drivers should want to improve the way to offer a ride or to be easily reachable by the customer, for using the application need to own a smartphone, in which it is installed the mobile application, with GPS and internet connection.
Intended customers should want to request quickly a ride to a location or book a taxi for a certain time, being in the city area, using the web application or the mobile app, respectively with a PC, with internet connection and a browser, or with a smartphone, with internet connection too.

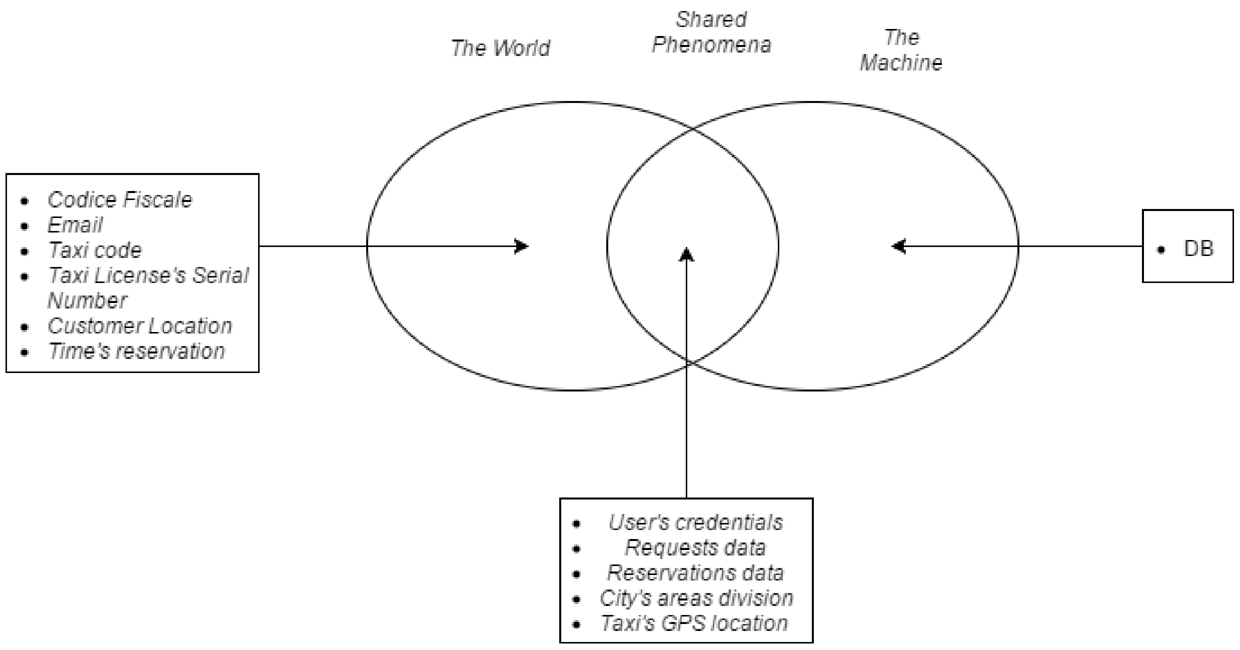
2.4 Constraints

The system must require to the user the permission to store and use all the data and the web cookies usage and in the case of the driver, it has to request the access to the GPS data. Moreover, the system must be developed using Java SE 8, tested with JUnit and runs on JVM with an Oracle Database.

2.5 Assumptions and Dependencies

- *The driver's GPS must be switched on using the mobile app*
- *The driver's GPS must work rightly*
- *The user must have a working internet connection*
- *All queues will be managed with a FIFO policy*
- *An available taxi driver using the app must be in a city's area*
- *The customer inserts an existing and real (the position where he actually is) position in a request*

2.6 The World and the Machine



2.7 Scenarios

2.7.1 Scenario 1

Giovanni is just arrived in Milan's airport; he needs a taxi to pick him up to Garibaldi railway station from Linate. As soon as the plane lands, Giovanni is aware to not be cheated by the annoying taxi drivers without license in the airport's entrance, but luckily he notices from an advertisement in the terminal about an app called MyTaxiService, offered by the city's government, which helps to find out quickly a taxi. Immediately he downloads it in his smartphone and starts to sign up, in the first attempt he submits a wrong email, mistyping the "@" letter, then the app notices it requiring to fill that form's field again until he inserts a right one.
Then Giovanni blames his smartphone's keyboard and inserts a valid mail, afterwards he receives a confirmation mail in the given email in order to endorse his subscription to the service. Finally, he creates a new ride request, after submitting his currently position (address) and waits for a taxi confirmation. A few minutes later, he receives a notification that confirms a taxi will pick him up and the estimate waiting time.

2.7.2 Scenario 2

It is 11:30 a.m. and it is raining in Milan. Michele has booked a train at 12:00 a.m. so he decides to reach the train station by metro. Because of a technical malfunction, the metro line that he has to take is blocked so he decides to take a taxi. Because of the weather, the phone line is busy so Michele decides to use MyTaxiService. He take his smartphone, open the relative APP, inserts the credentials, signs in and requests a ride inserting his position. Unfortunately, he has inserted a not valid street address so the system notifies it to Michele with a message and shows again the Ride-request page. This time Michele inserts the correct street address and a few moments later system notifies him about the confirmation of the passage, that is taken over by taxi n°13657. Waiting time is 7 minutes approximately.

2.7.3 Scenario 3

Maria is working on a project on her notebook. The next day at 10:00 a.m., she has a meeting with a technical director of an important Milan company exactly to present her project. The location in which she has to go is situated in the opposite part of the city from Maria's home and it is not well connected by means of transportation.
Because of Maria has recently transferred in Milan she decides to call her brother Francesco, who has been lived there for three years, to ask for advices to reach the location.
It is about two months that Francesco frequently uses MyTaxiService and he never had problems with it so he advises her sister to sign up in the service and to request a reservation in order to be sure to reach in time the location. Maria decides to trust his brother so she opens a browser from her notebook and connects to the service's website. She signs up without problems and request a reservation, inserting the street address of her home, the street address of the meeting and reservation time (9:20 a.m.). The System immediately informs that the request has been taken over and that ten minutes before the start time she will receive the confirm in which will be written the taxi number that will be pick her up.

2.7.4 Scenario 4

Silvio has been working as taxi driver for 11 years, he always been quite pleased with his work that always gave him much satisfaction. Two months ago, he started to use MyTaxiService. Since from the beginning he noticed the service efficiency; in a single day, on average, he carries out 4/5 rides more than when he did not use it and he can take some short break with his colleagues without staying in his taxi. It is 8:30 a.m. and Silvio has just sitting in the seat of his taxi, "Today is a good day, my dear" says him addressing to his car; he opens the MyTaxiService APP from his smartphone, inserts his credentials, correctly signs in and sets himself 'available'.
A few seconds later he receives a first ride request; he notices that the street address in which he should take the passenger is so far from his position despite being in the same work area so Silvio decides to reject the request. 30 seconds later he receives a reservation request in the same date at 2:20 p.m.; he notices that the itinerary which passenger has to do is very long and, thinking to earn a good amount of money, Silvio accept the request. Two minutes later, he receives another ride request; this time the street address is near to his position and he accepts the request, starts engines and starts his workday. Automatically the system sets Silvio 'not available'.

2.8 Future Extensions

- *Offer a feedback rating in order to provide a mark or share experiences with a given taxi driver*
- *Let the passenger share a ride or reservation with other people (partly done with the programmatic interface for developing "TaxiShare")*
- *Integration with Google Maps or Waze.*
- *Integration of a chat between the passenger and the taxi in order to provide a quick way to communicate in case of peculiar scenarios such that congestion, accident or car crash.*

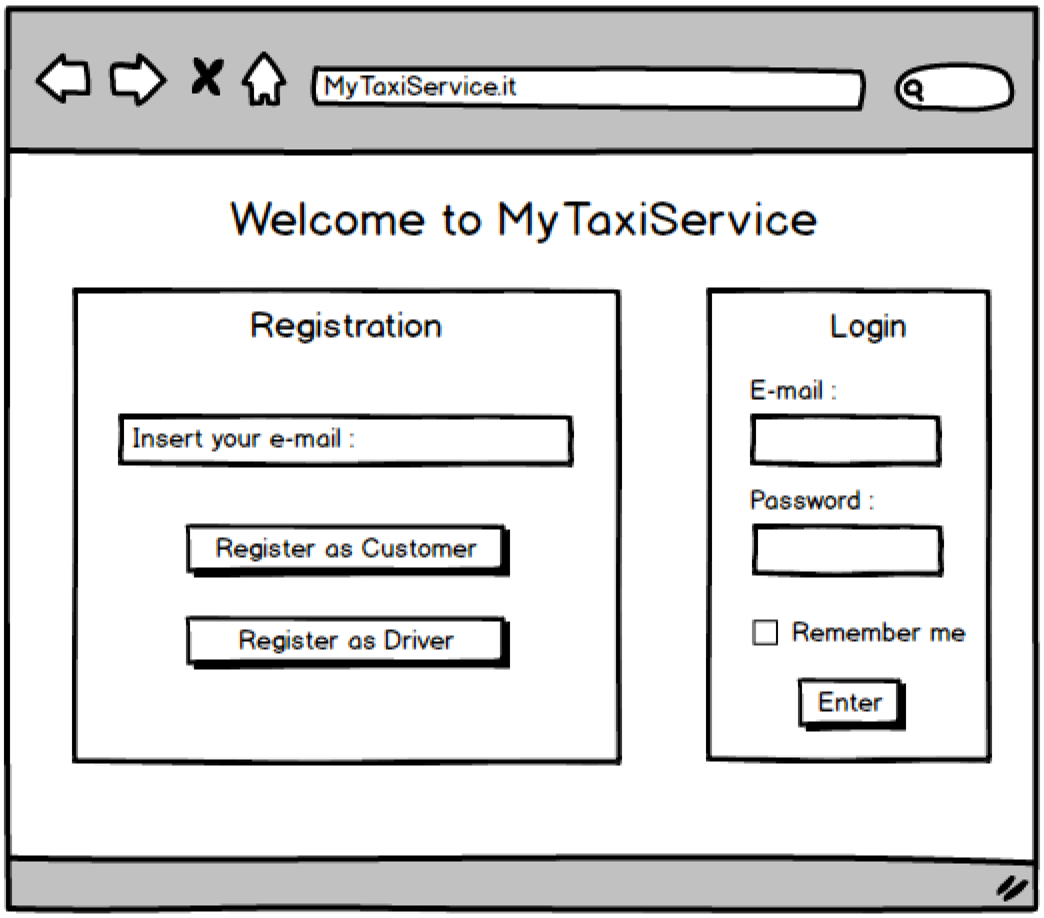
3. Specific requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

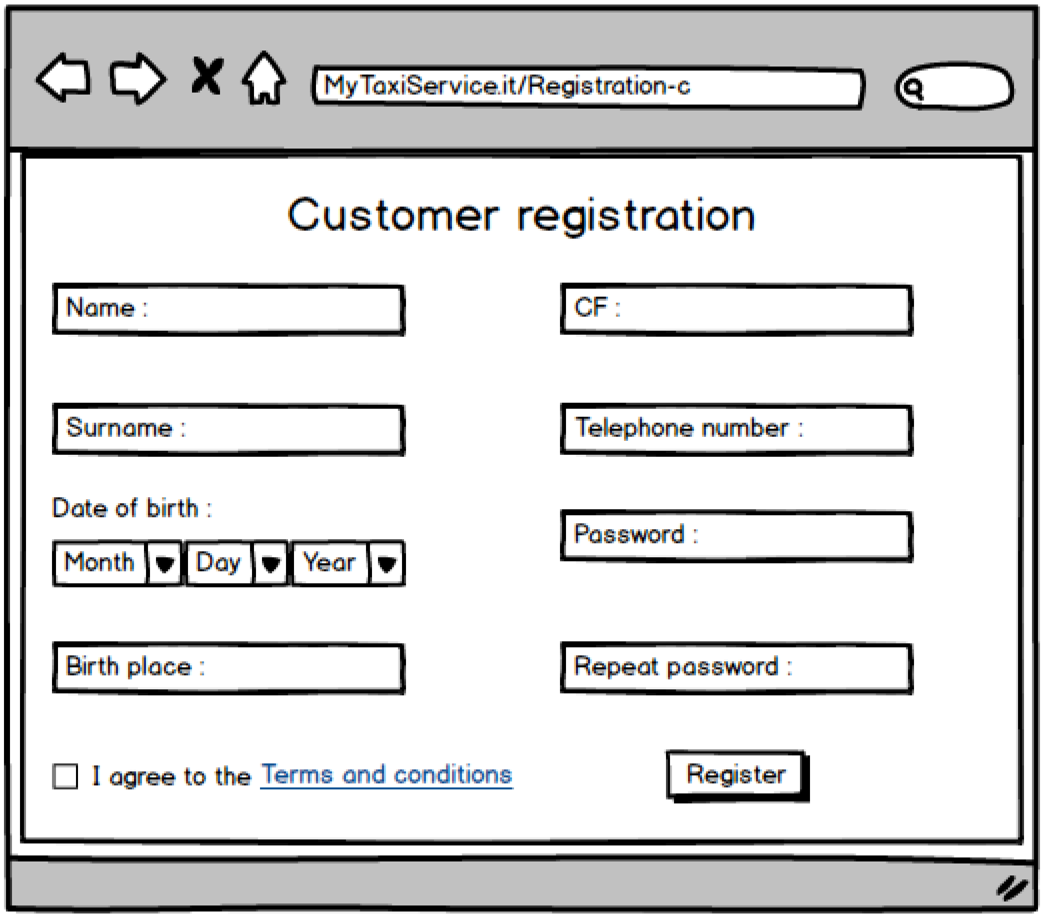
3.1.1.1 Homepage:

This mockup shows how visitors see the website. They can only sign up or sign in.



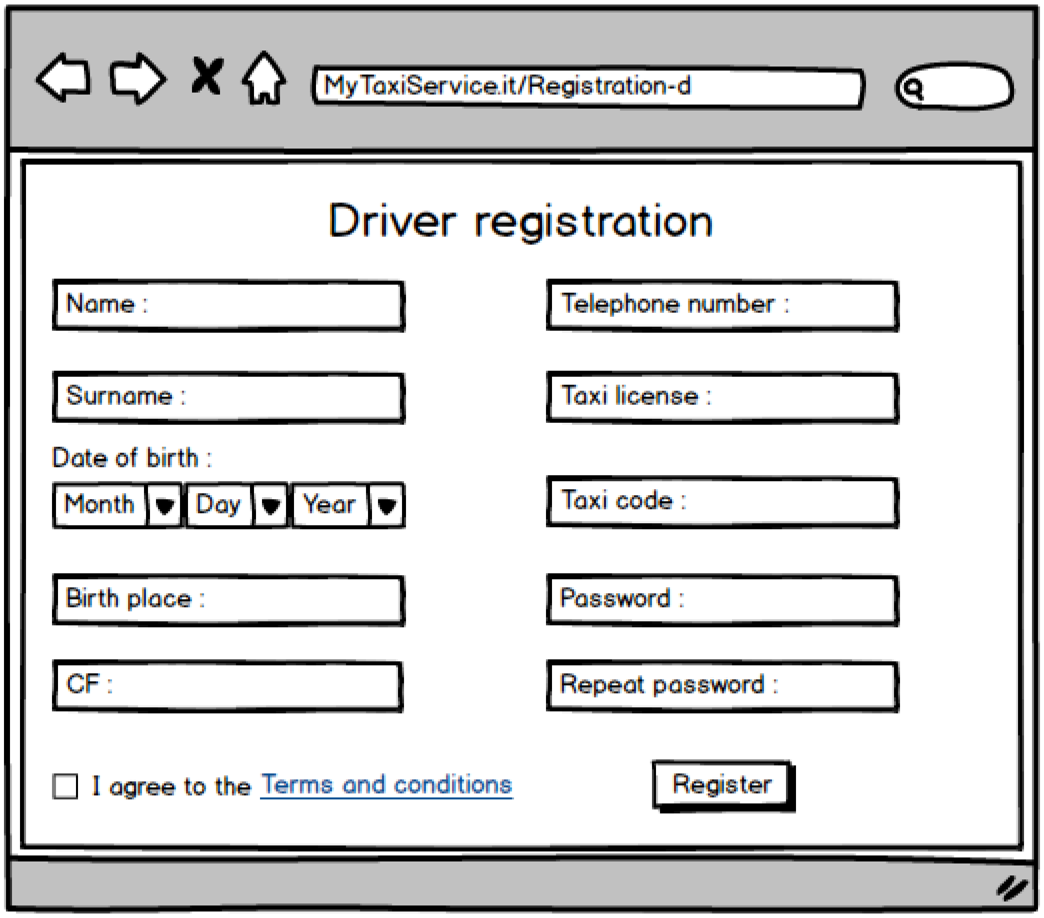
3.1.1.2 Registration as Customer:

This mockup shows the registration form page associated to the Customer.



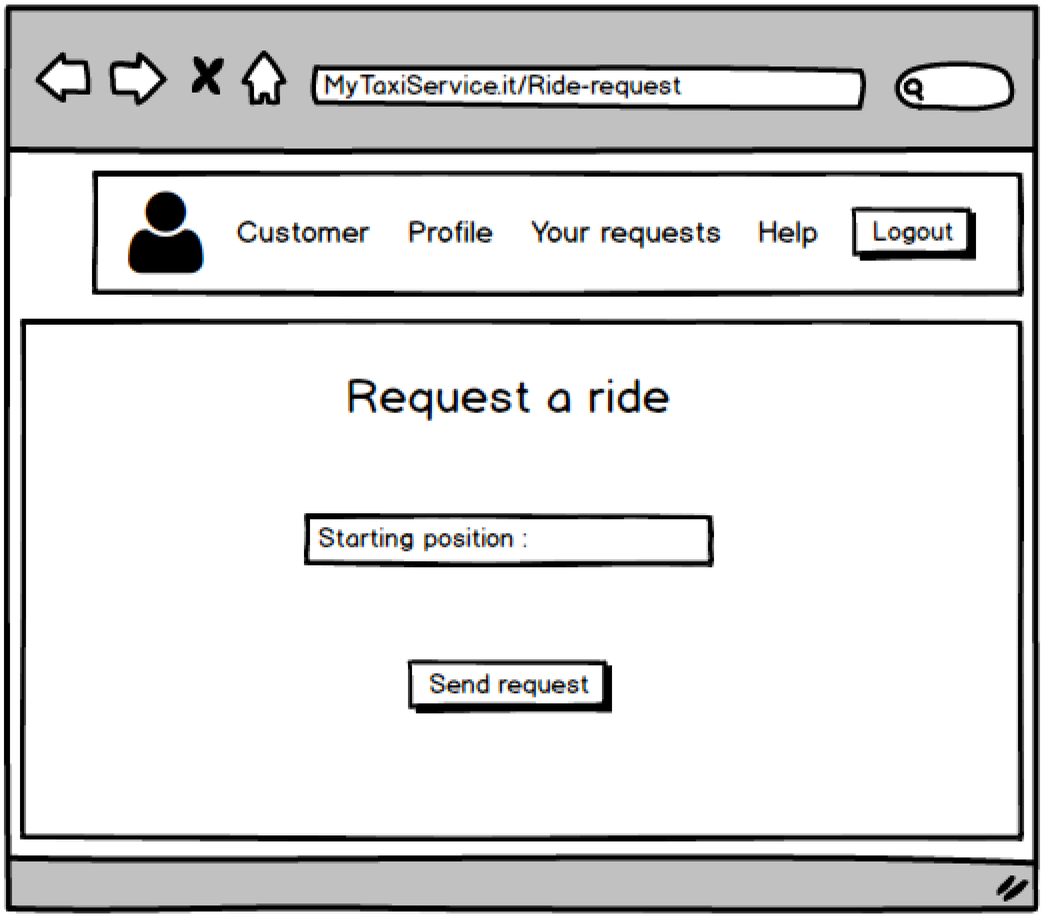
3.1.1.3 Registration as Driver:

This mockup shows the registration form page associated to the Driver.



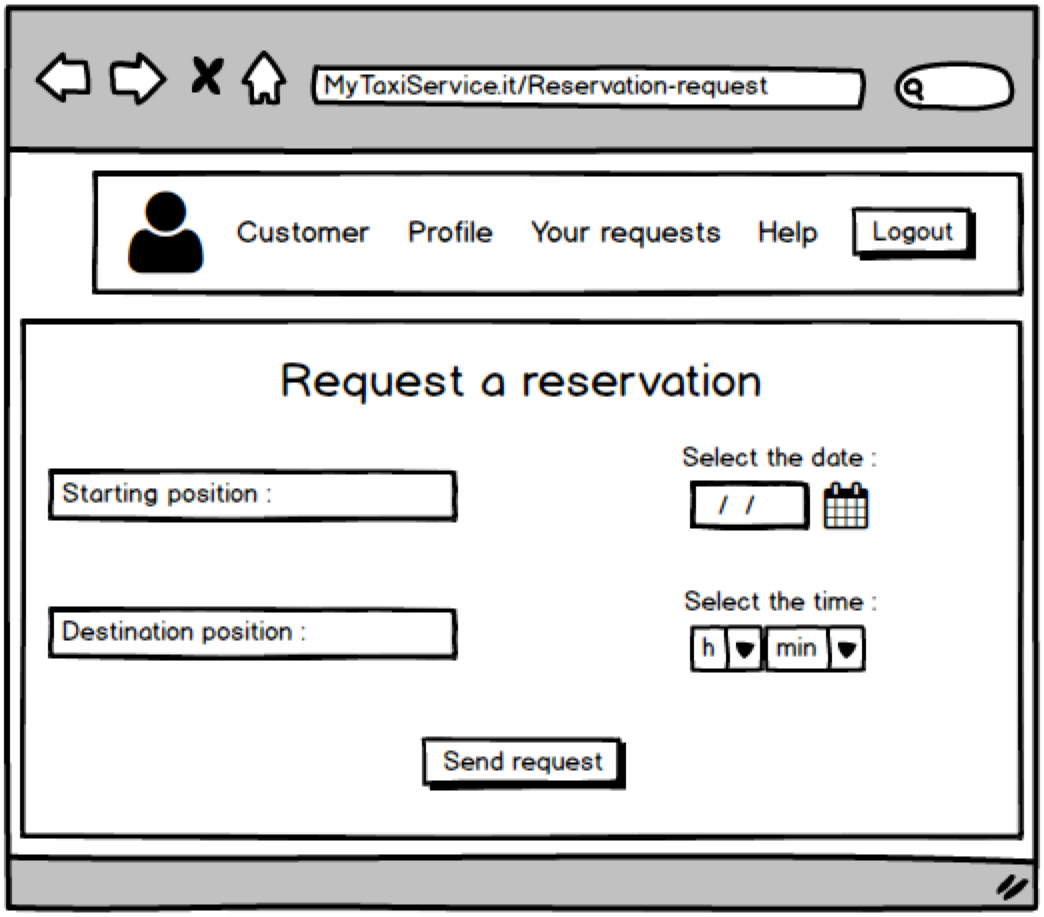
3.1.1.4 Request a ride (Customer):

This mockup shows the page in which a Customer can request a ride, inserting the starting position.



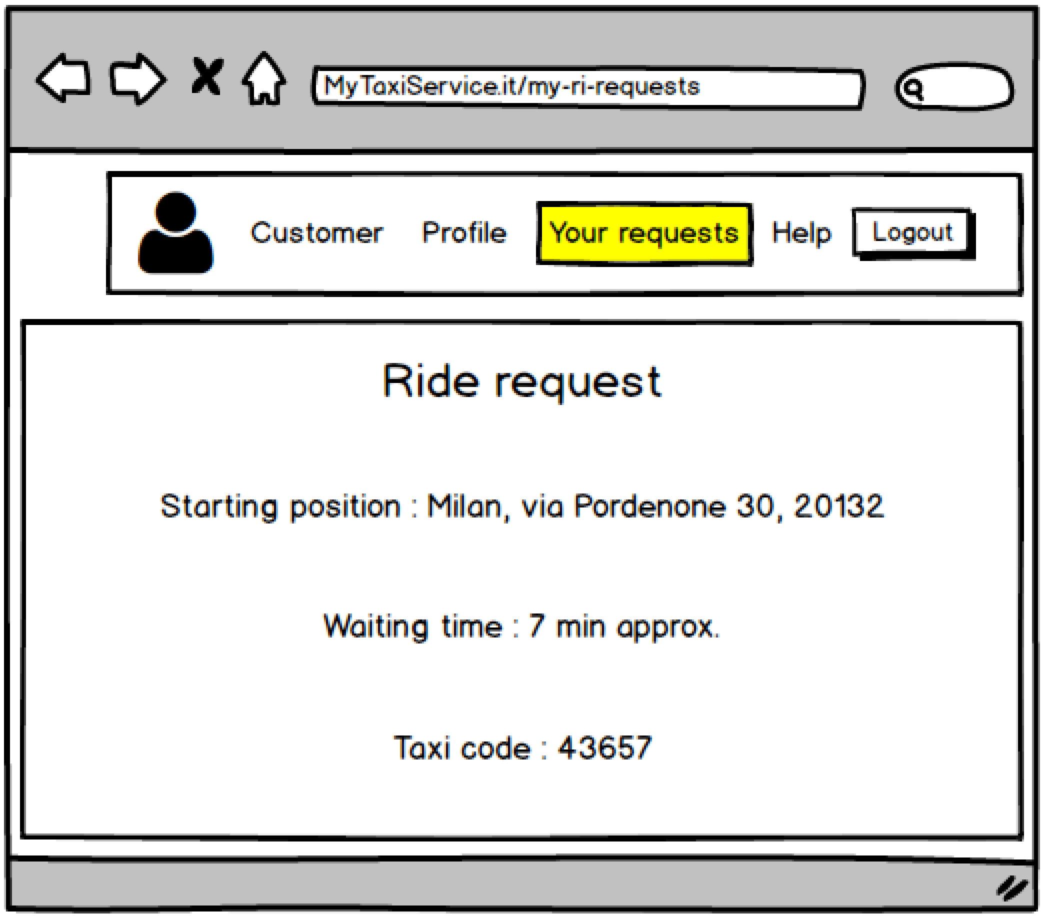
3.1.1.5 Request a reservation (Customer):

This mockup shows the page in which a Customer can request a reservation, inserting starting and destination positions, date and time.



3.1.1.6 Ride's details (Customer):

This mockup shows the page in which a Customer can see details, such as waiting time and code of incoming taxi, of a ride that he has previously requested.



Ride request

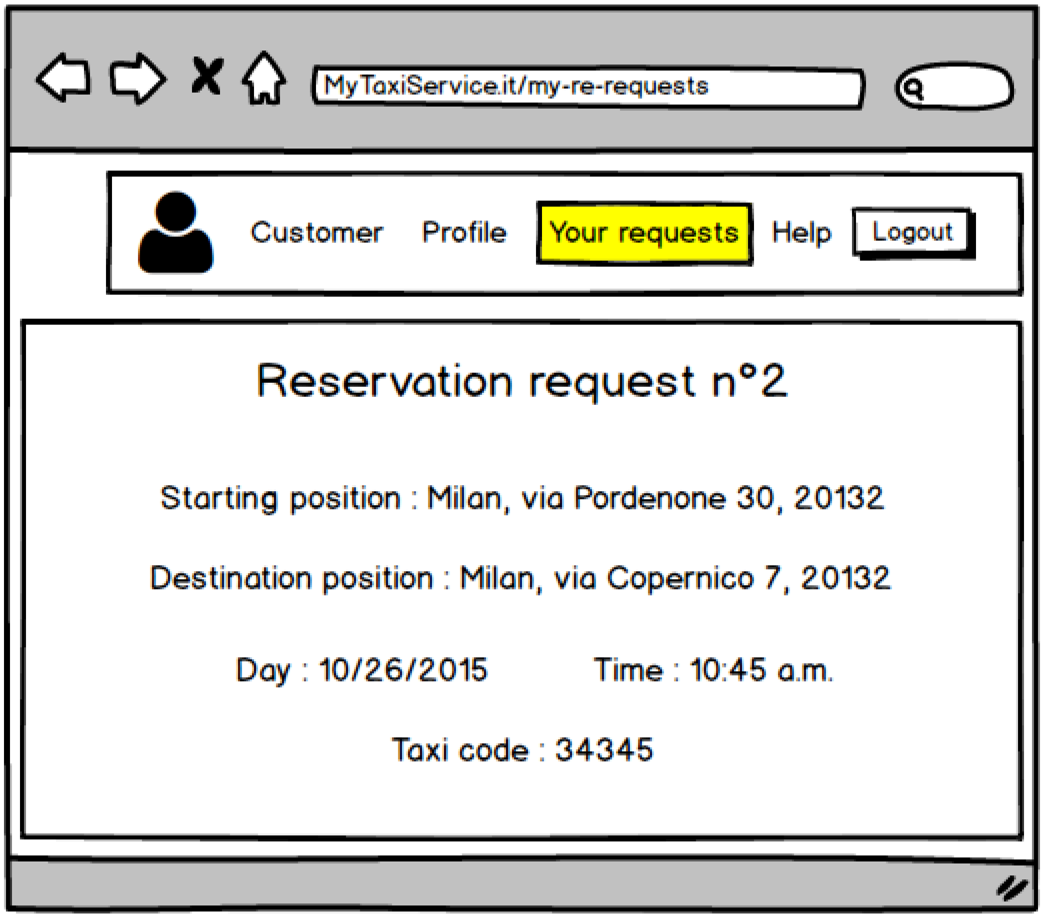
Starting position : Milan, via Pordenone 30, 20132

Waiting time : 7 min approx.

Taxi code : 43657

3.1.1.7 Reservation's details (Customer):

This mockup shows the page in which a Customer can see details, such as code of taxi, date and time, of a reservation that he has previously requested



Reservation request n°2

Starting position : Milan, via Pordenone 30, 20132

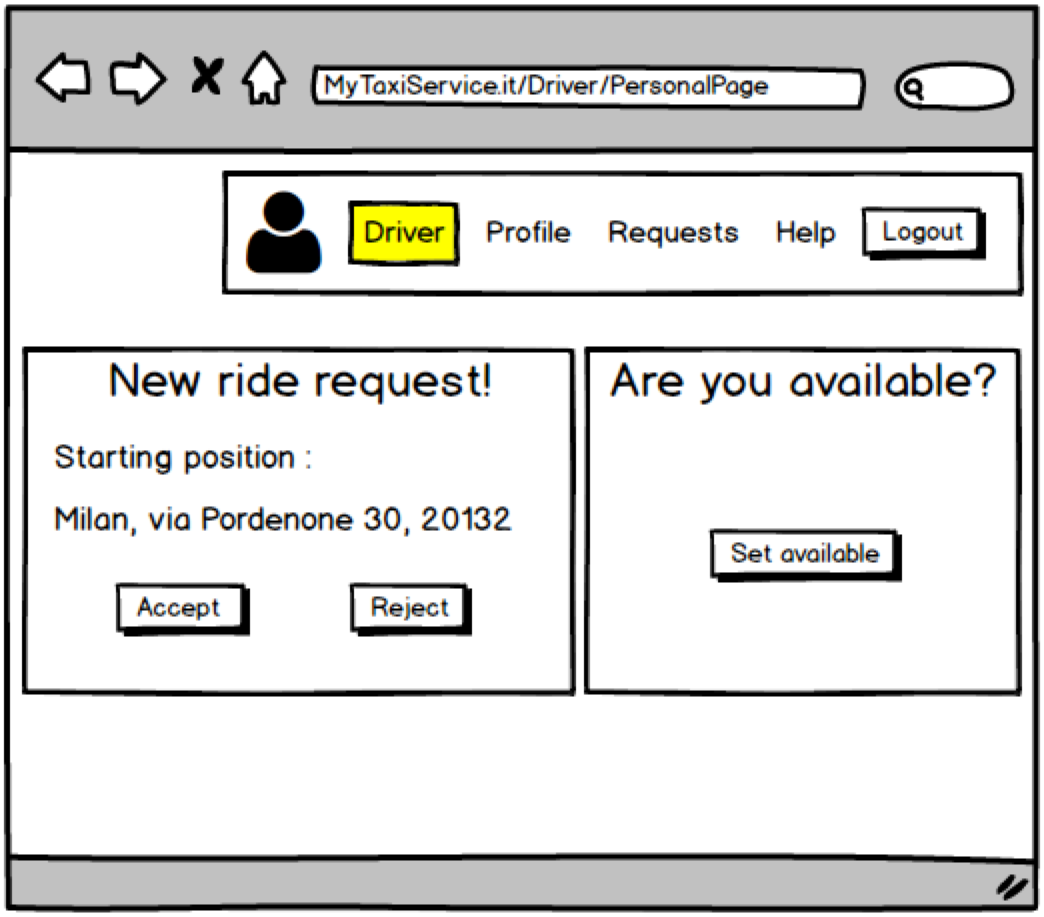
Destination position : Milan, via Copernico 7, 20132

Day : 10/26/2015 Time : 10:45 a.m.

Taxi code : 34345

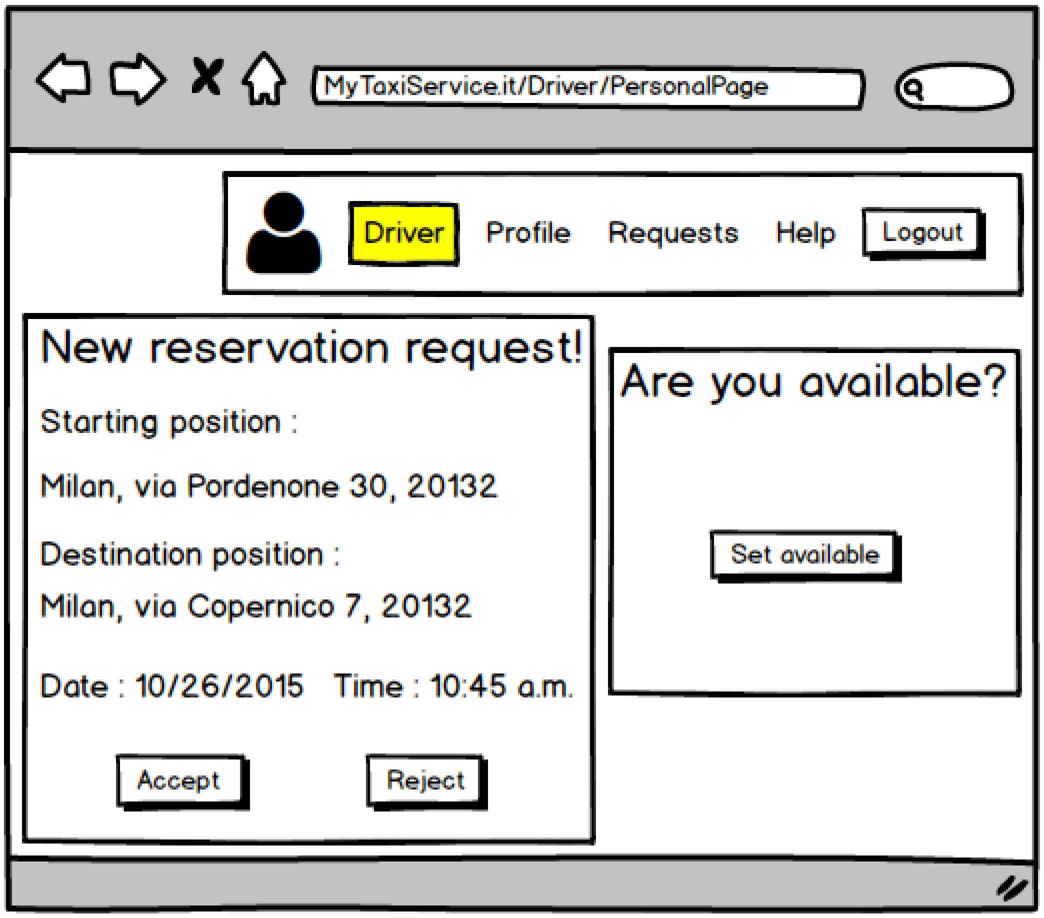
3.1.1.8 Personal page (1) (Driver):

This mockup shows a generic driver's personal page in which he can set his availability and see new requests. In this case, the Driver receive a new ride request.



3.1.1.9 Personal page (2) (Driver):

This mockup shows a generic driver's personal page in which he can set his availability and see new requests. In this case, a Driver receive a new reservation request.



MyTaxiService.it/Driver/PersonalPage

Driver Profile Requests Help Logout

New reservation request!

Starting position :

Milan, via Pordenone 30, 20132

Destination position :

Milan, via Copernico 7, 20132

Date : 10/26/2015 Time : 10:45 a.m.

Accept Reject

Are you available?

Set available

3.2 Goals and related Functional requirements

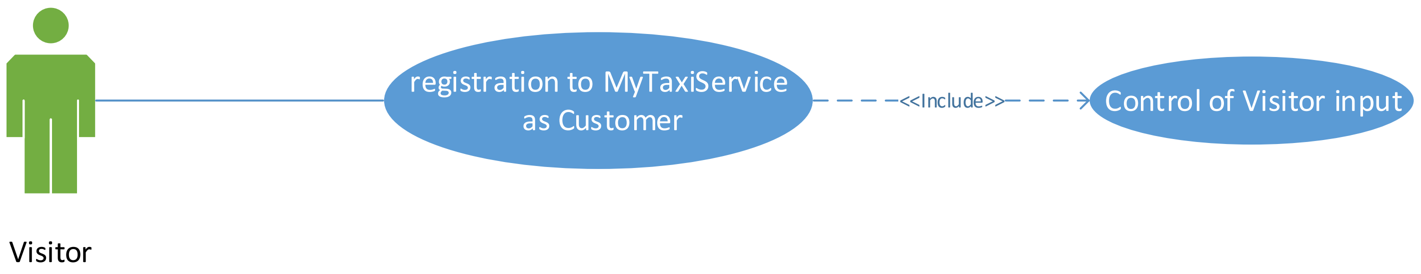
3.2.1 Sign up as Customer

Name	*Sign up as Customer*
Actors	*Visitor*
Entry condition	*NULL*
Event flow	*1. Visitor enters to the website or opens the APP* *2. Visitor click on the "register as Driver" button* *3. Visitor completes the form in which he has to write:* ❖ *Name* ❖ *Surname* ❖ *Email* ❖ *Password* ❖ *Date of birth* ❖ *Telephone number* *4. Visitor clicks on the "confirm" button*
Exit condition	*Visitor successfully completes the registration*
Exceptions	*1. The email is already used* *2. One or more mandatory fields are not filled or they are not valid*

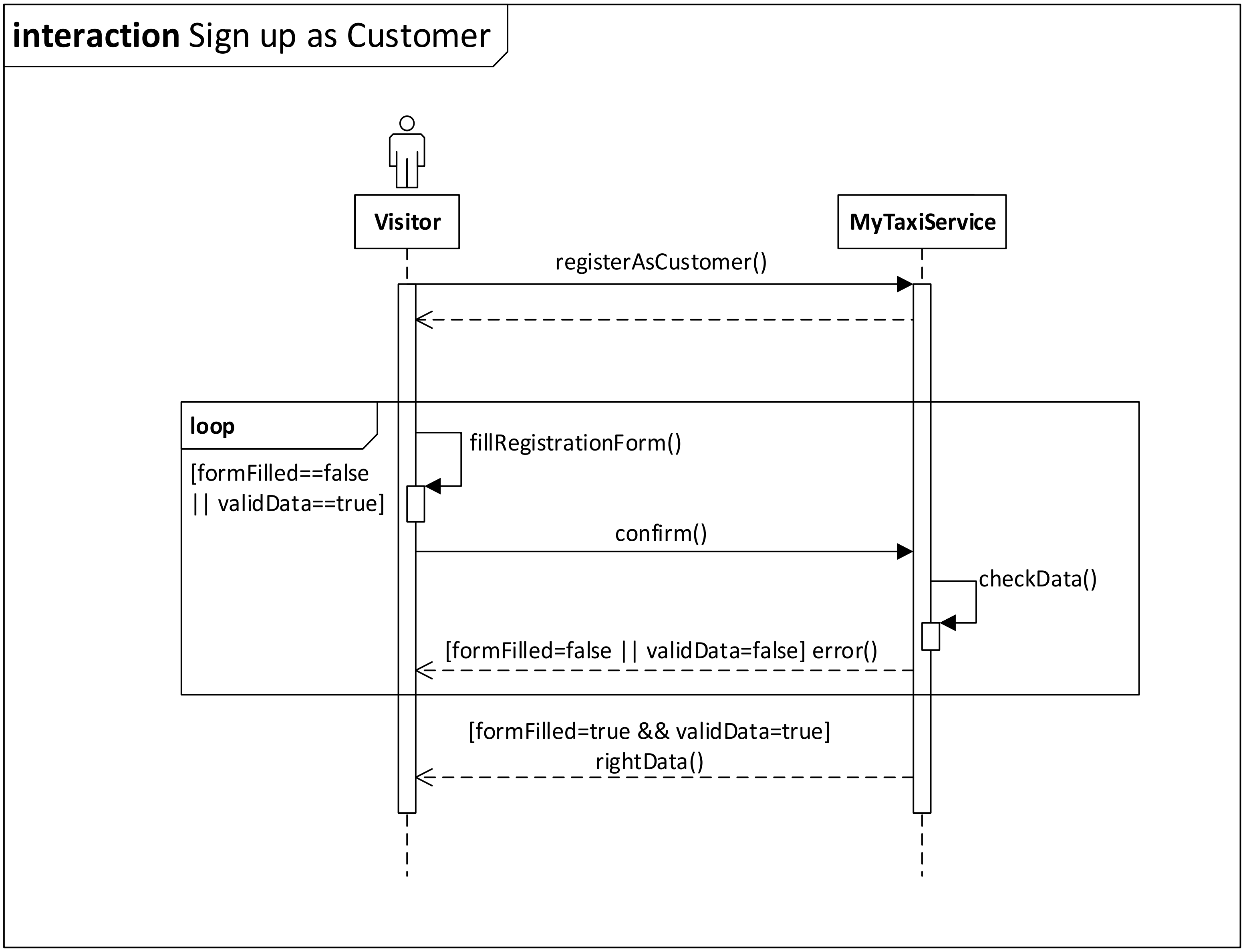
3.2.1.1 Functional requirements (Sign up as Customer)

[R1] CF must be in compliance to the inserted data
[R2] Visitor can only sign up or sign in

3.2.1.2 Use Case Diagram (Sign up as Customer)



3.2.1.3 Sequence diagram (Sign up as Customer)


interaction Sign up as Customer
Visitor
MyTaxiService
registerAsCustomer()
loop
[formFilled==false || validData==true]
fillRegistrationForm()
confirm()
checkData()
[formFilled=false || validData=false] error()
[formFilled=true && validData=true] rightData()

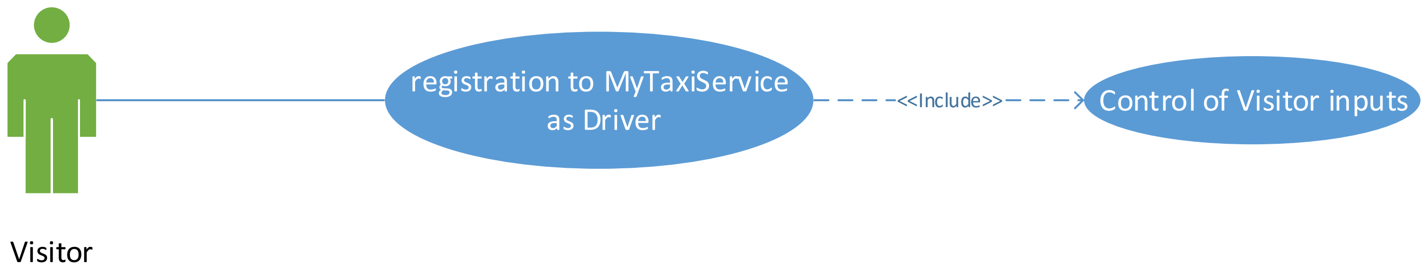
3.2.2 Sign up as Driver

Name	*Sign up as Driver*
Actors	*Visitor*
Entry condition	*NULL*
Event flow	*1. Visitor enters to the website or opens the APP 2. Visitor click on the "register" button 3. Visitor completes the form in which he has to write: ❖ Name ❖ Surname ❖ Email ❖ Password ❖ Date of birth ❖ Telephone number ❖ Taxi license ❖ Taxi code 4. Visitor clicks on the "confirm" button*
Exit condition	*Visitor successfully completes the registration*
Exceptions	*1. The email is already used 2. One or more mandatory fields are not filled or they are not valid 3. Taxi code isn't associated to the taxi license*

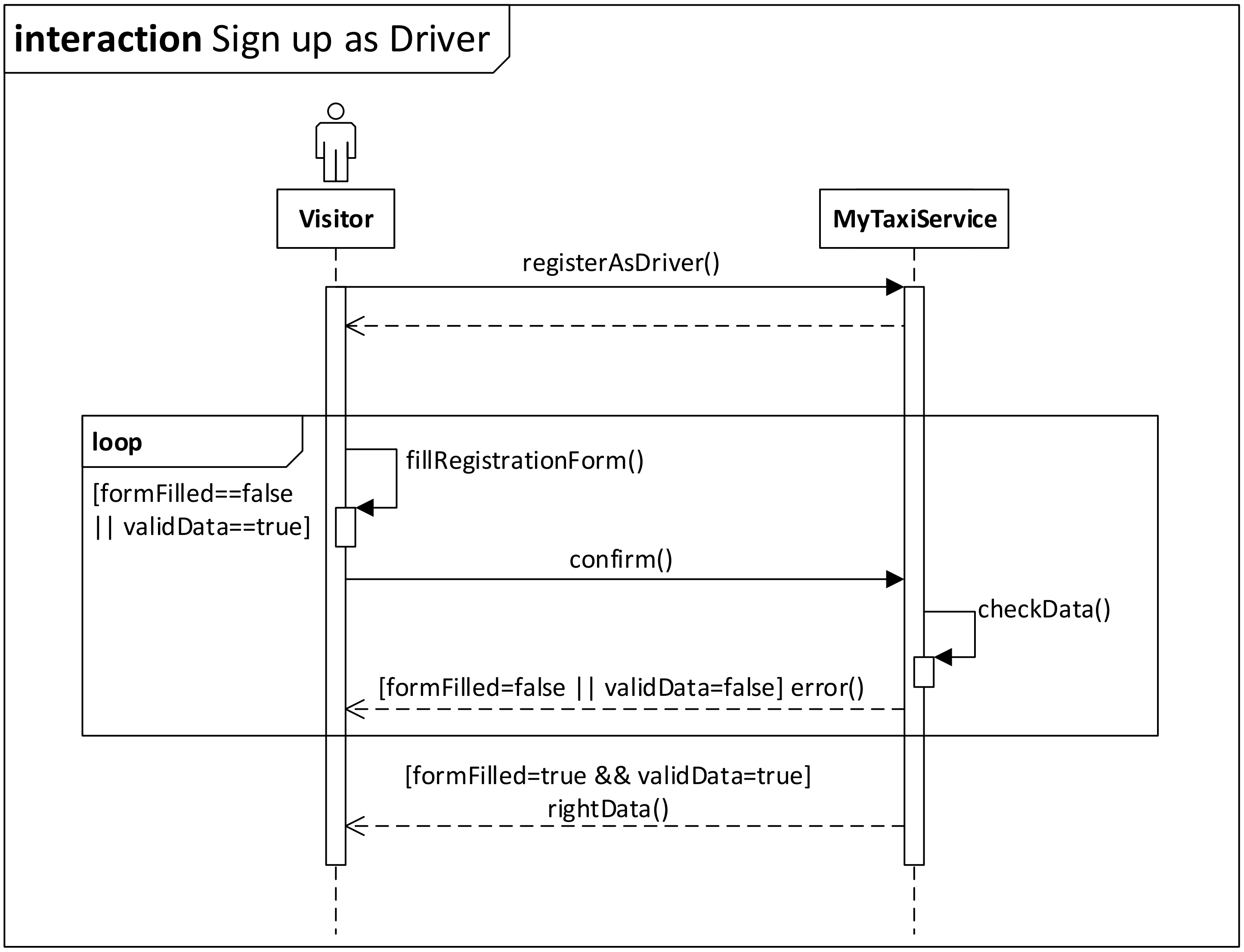
3.2.2.1 Functional requirements (Sign up as Driver)

[R1] CF must be in compliance to the inserted data
[R2] Visitor can only sign up or sign in
[R3] Taxi code must be associated to the Visitor's taxi license
[R4] Taxi license must belong to the Visitor

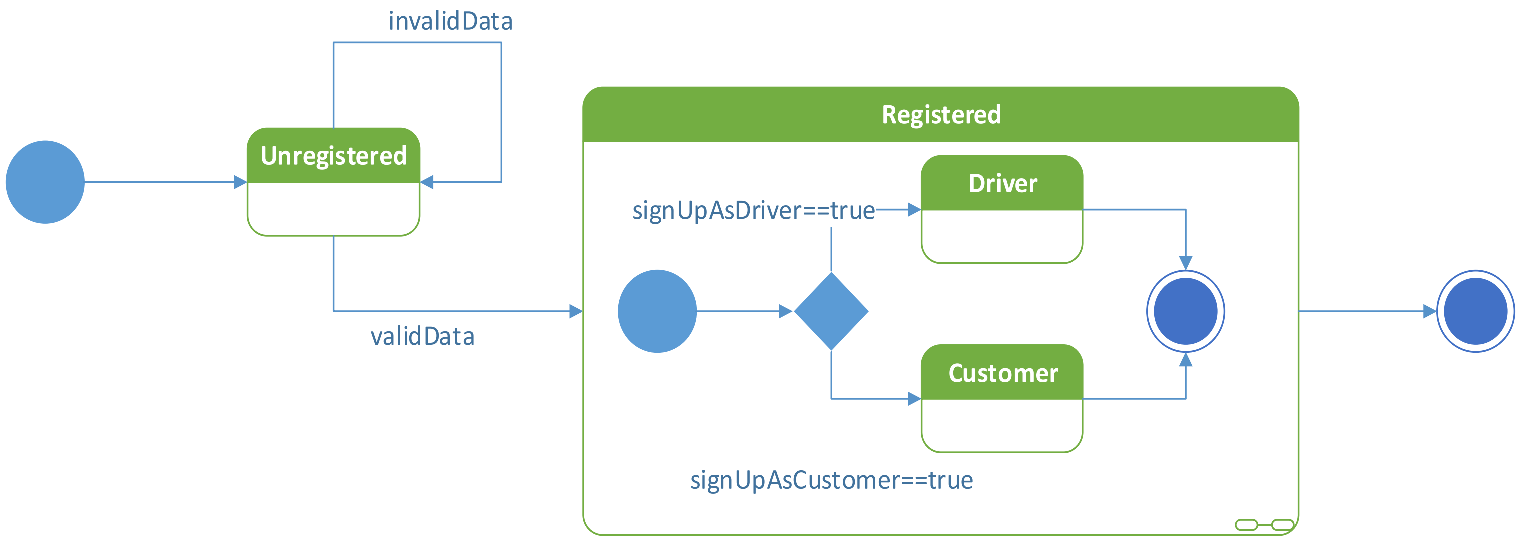
3.2.2.2 Use Case Diagram (Sign up as Driver)


registration to MyTaxiService as Driver
<<Include>>
Control of Visitor inputs
Visitor

3.2.2.3 Sequence diagram (Sign up as Driver)


interaction Sign up as Driver
Visitor
MyTaxiService
registerAsDriver()
loop
[formFilled==false || validData==true]
fillRegistrationForm()
confirm()
checkData()
[formFilled=false || validData=false] error()
[formFilled=true && validData=true] rightData()

3.2.2.4 State chart (general Sign up)



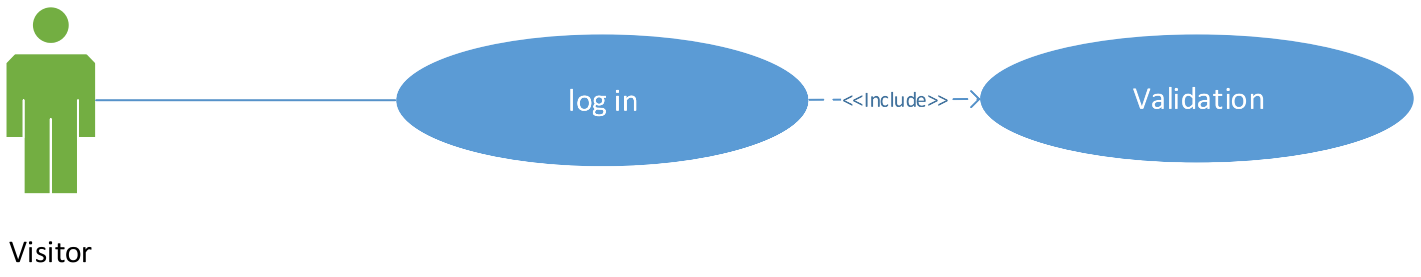
3.2.3 Login

Name	*Login*
Actors	*User*
Entry condition	*User is registered to the system*
Event flow	1. *User enters to the website or opens the APP* 2. *User inserts his username and password in the apposite fields* 3. *User clicks on the "login" button*
Exit condition	*The system show user's personal page*
Exceptions	1. *Username or password inserted by user are wrong. The system notifies an error and show the "login" page*

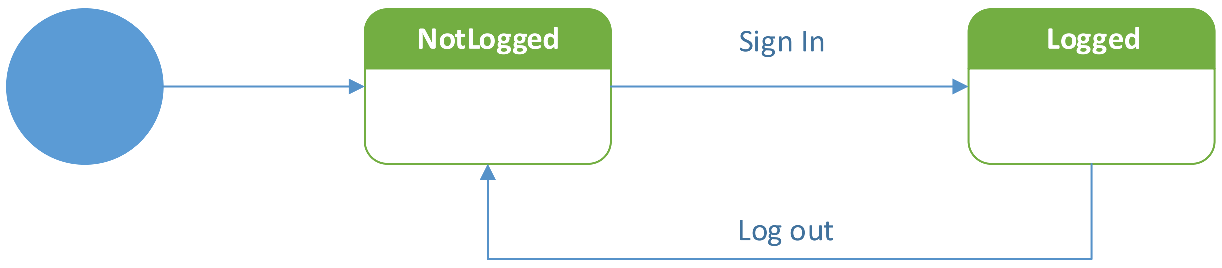
3.2.3.1 Functional requirements (Login)

[R1] Credentials must be correct
[R2] Visitor must not be already logged

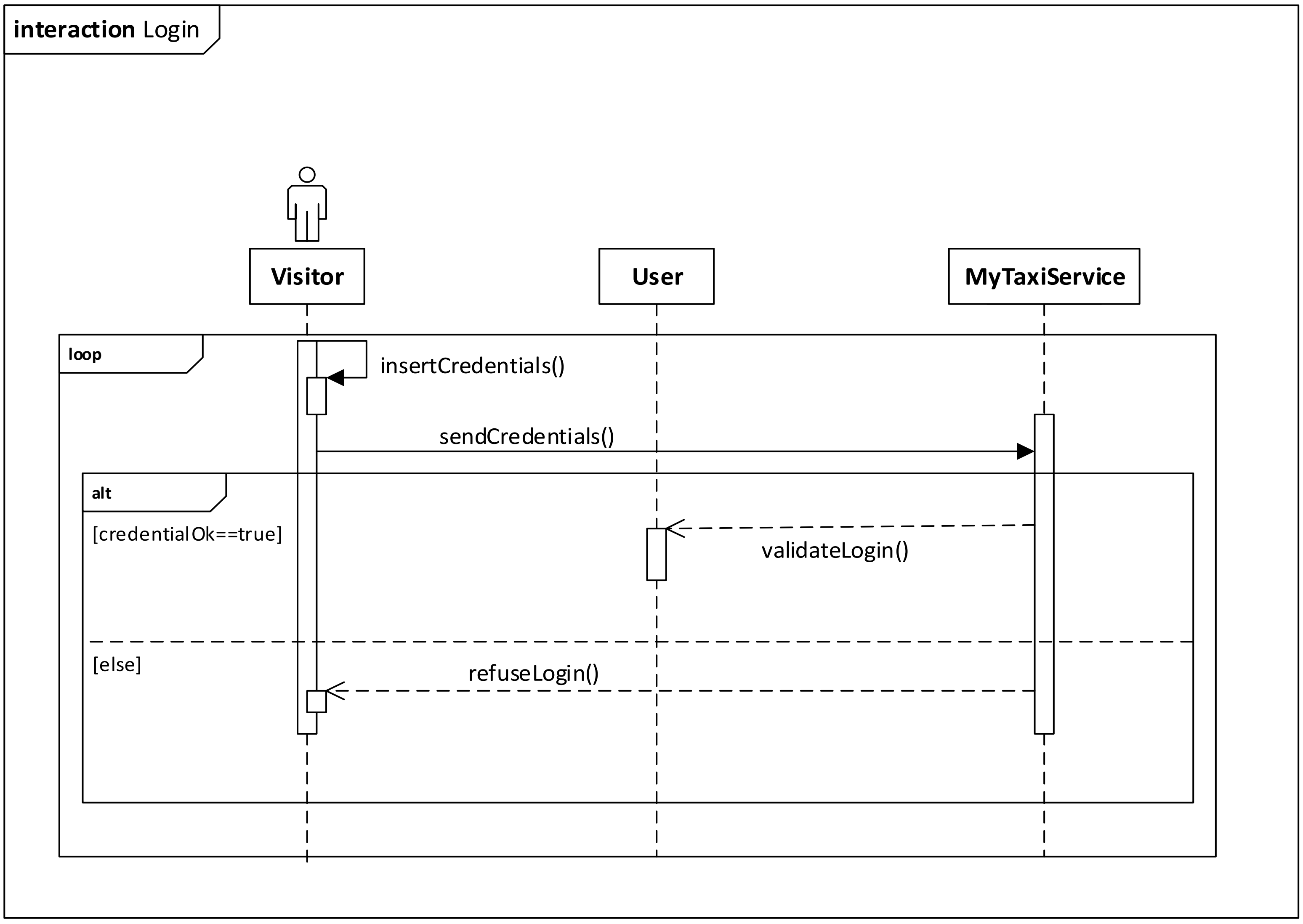
3.2.3.2 Use Case Diagram (Login)



3.2.3.3 State chart (Login)



3.2.3.4 Sequence diagram (Login)



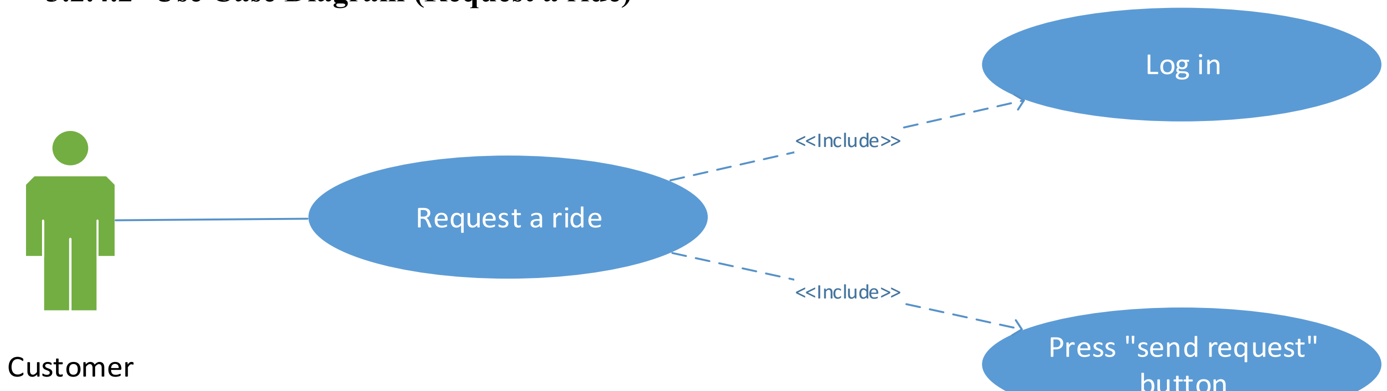
3.2.4 Request a Ride

Name	*Request a ride*
Actors	*Customer*
Entry condition	*User is logged into the system*
Event flow	*1. Customer clicks on the "Request a ride" button* *2. Customer inserts the position in which he wants to start the ride* *3. User clicks on the "Send request" button*
Exit condition	*The system notifies the Customer that the request has been forwarded to the drivers*
Exceptions	*1. The position inserted by the customer is not valid, the system notifies it to the user and asks to choose an existing position* *2. The position inserted by the customer isn't covered by the service, the system notifies it to the user and shows the homepage*

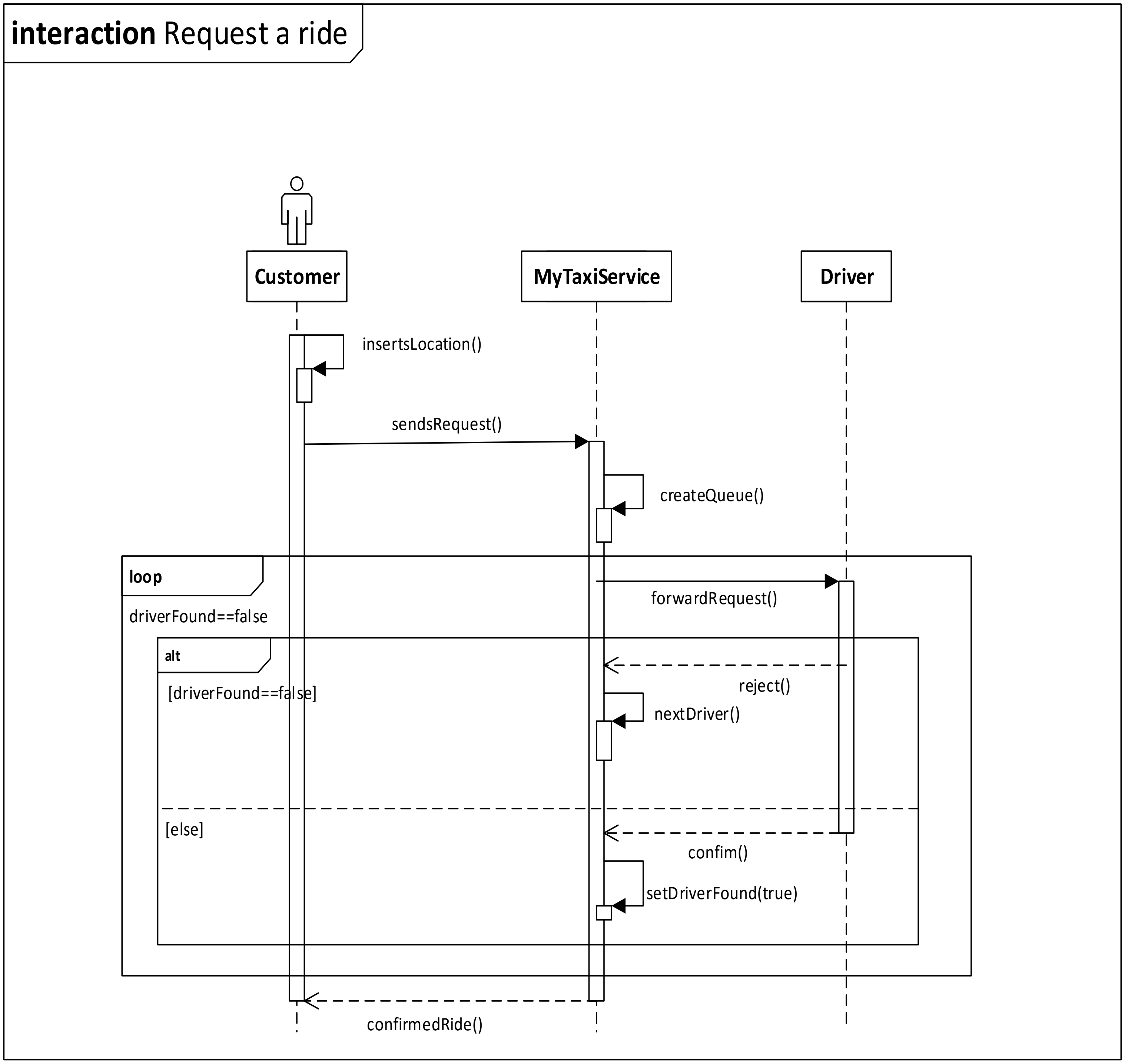
3.2.4.1 Functional requirements (Request a ride)

[R1] Customer must insert a position covered by the service
[R2] Customer must complete the form and send the request

3.2.4.2 Use Case Diagram (Request a ride)



3.2.4.3 Sequence diagram (Request a ride)



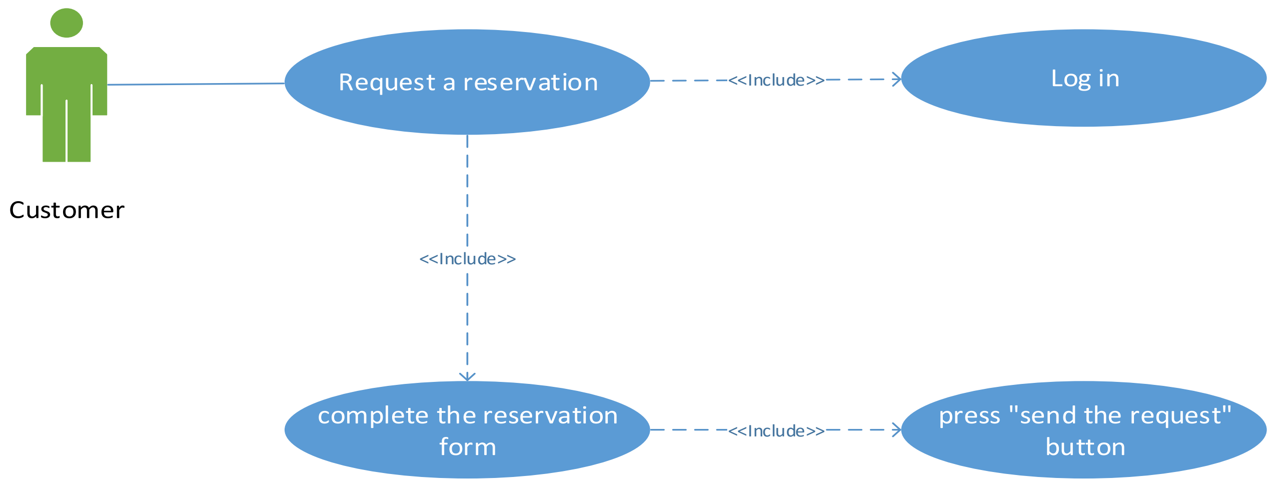
3.2.5 Request a Reservation

Name	*Request a reservation*
Actors	*Customer*
Entry condition	*User is logged into the system*
Event flow	*1. Customer clicks on the "Request a reservation" button 2. Customer inserts the following data in the apposite fields: ❖ Start position ❖ Arrive position ❖ Time of departure 3. User clicks on the "Send request" button*
Exit condition	*The system notifies the Customer that the request has been taken over*
Exceptions	*1. One or both the positions inserted by the customer are not valid. The system notifies it to the user and asks him to choose existing positions 2. One or both the positions inserted by the customer are not covered by the service. The system notifies it to the user and shows the homepage 3. Time of departure inserted by the use isn't valid. The system notifies it to the user and asks him to choose a valid time 4. Request is sent less than two hours before the departure time. The system notifies it to the user and asks him to choose a correct time.*

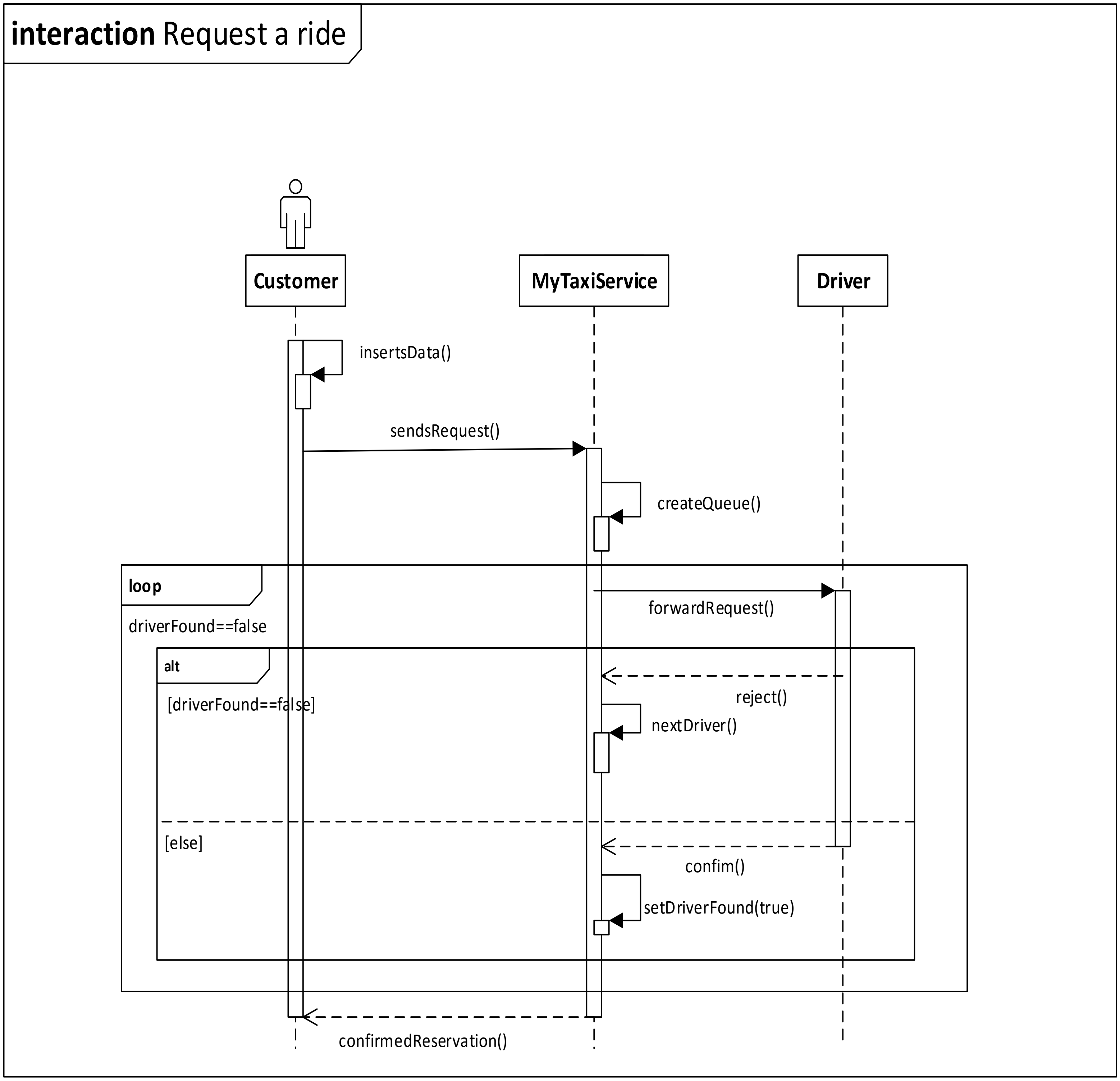
3.2.5.1 Functional requirements (Request a reservation)

[R1] Customer must insert a position covered by the service
[R2] Customer must complete the form and send the request
[R3] Request must be forwarded to the 1st Driver in queue

3.2.5.2 Use Case Diagram (Request a reservation)



3.2.5.3 Sequence diagram (Request a reservation)



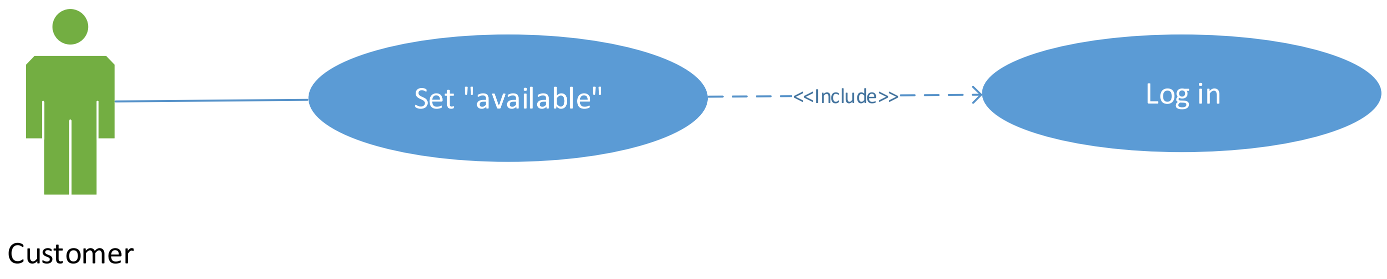
3.2.6 Set available

Name	*Set available*
Actors	*Driver*
Entry condition	*User is logged into the system*
Event flow	1. *User clicks on the "Set available" button* 2. *User clicks on the "Confirm" button*
Exit condition	*The system notifies the Driver that his state is successfully changed*
Exceptions	*NULL*

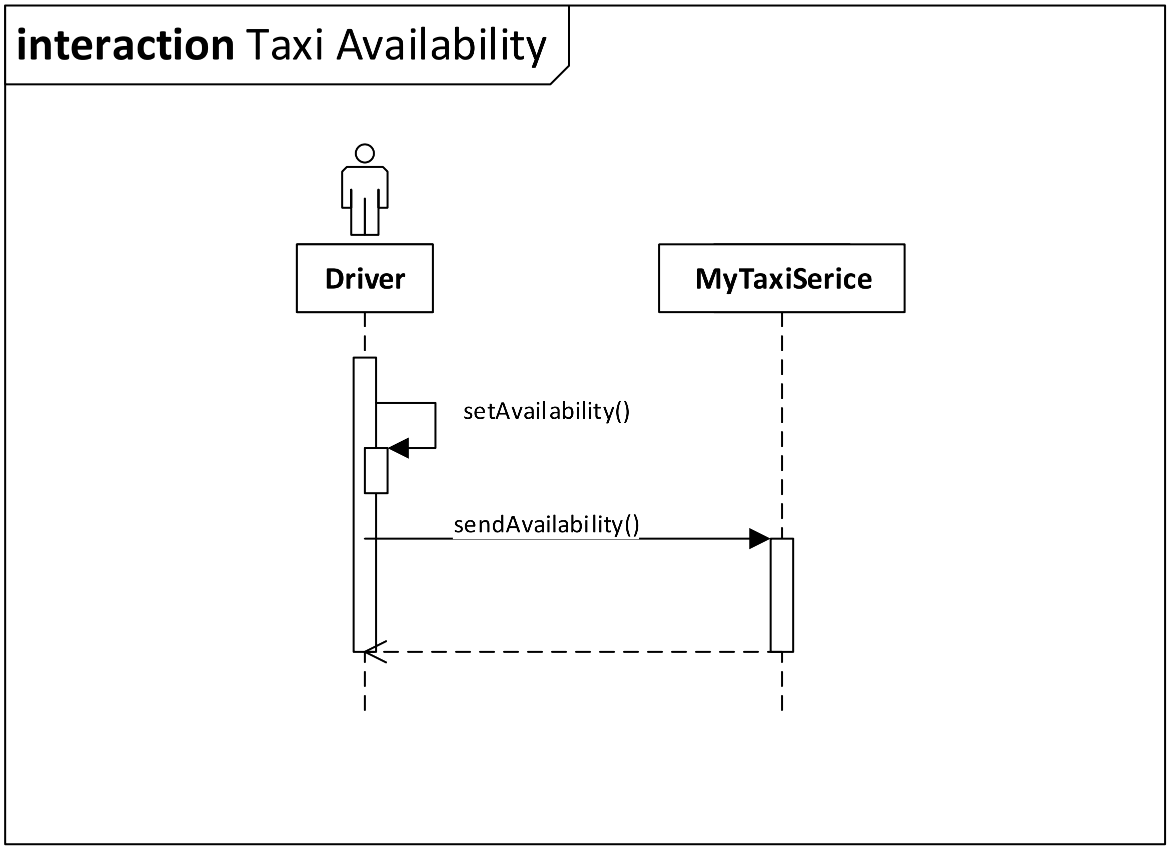
3.2.6.1 Functional requirements (Set available)

[R1] Driver must not be associated to another reservation in the same time

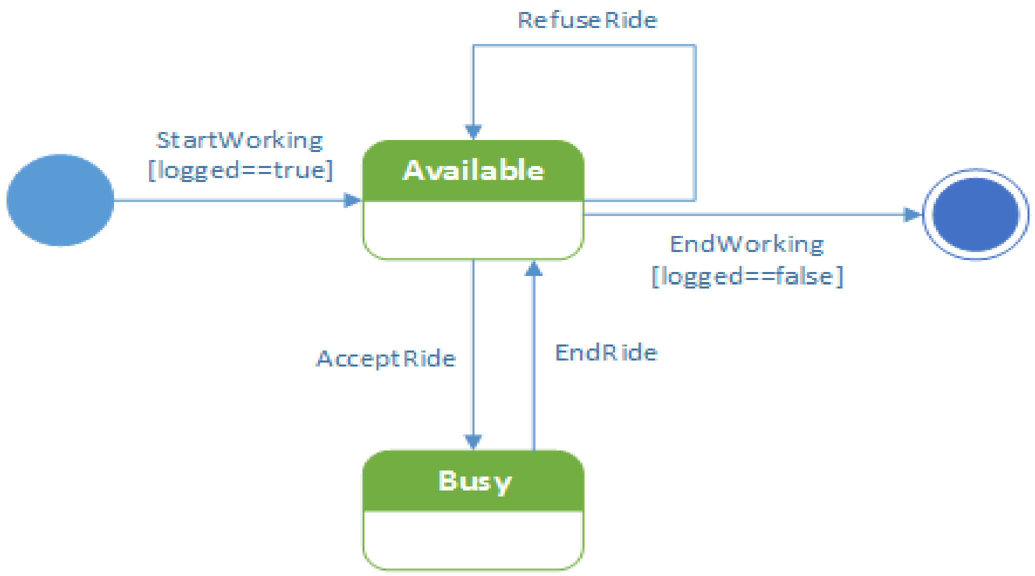
3.2.6.2 Use Case Diagram (Set available)



3.2.6.3 Sequence diagram (Set available)



3.2.6.4 State chart (general Sign up)



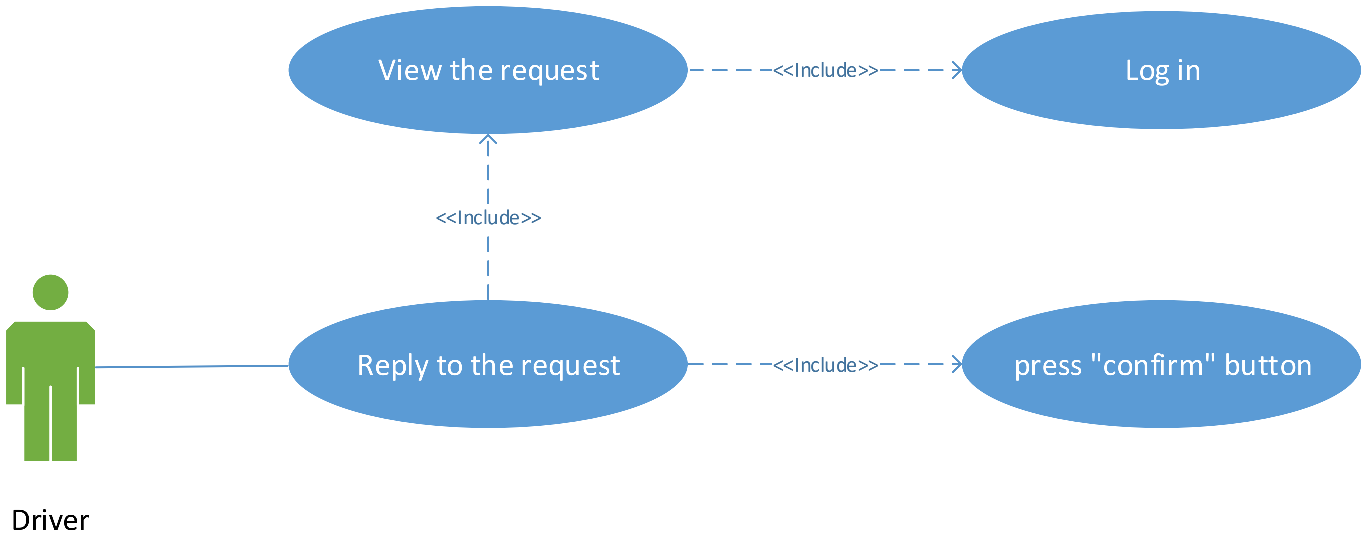
3.2.7 Respond to a request

Name	*Respond to a request*
Actors	*Driver*
Entry condition	❖ *User is logged into the system* ❖ *A customer must have requested a ride or a reservation in the zone in which the driver is* ❖ *Driver must be available*
Event flow	1. *User clicks on the "Accept request" button or on the "Reject request" button* 2. *User clicks on the "Confirm" button*
Exit condition	*The system notifies the Driver that his choice has been taken over*
Exceptions	*NULL*

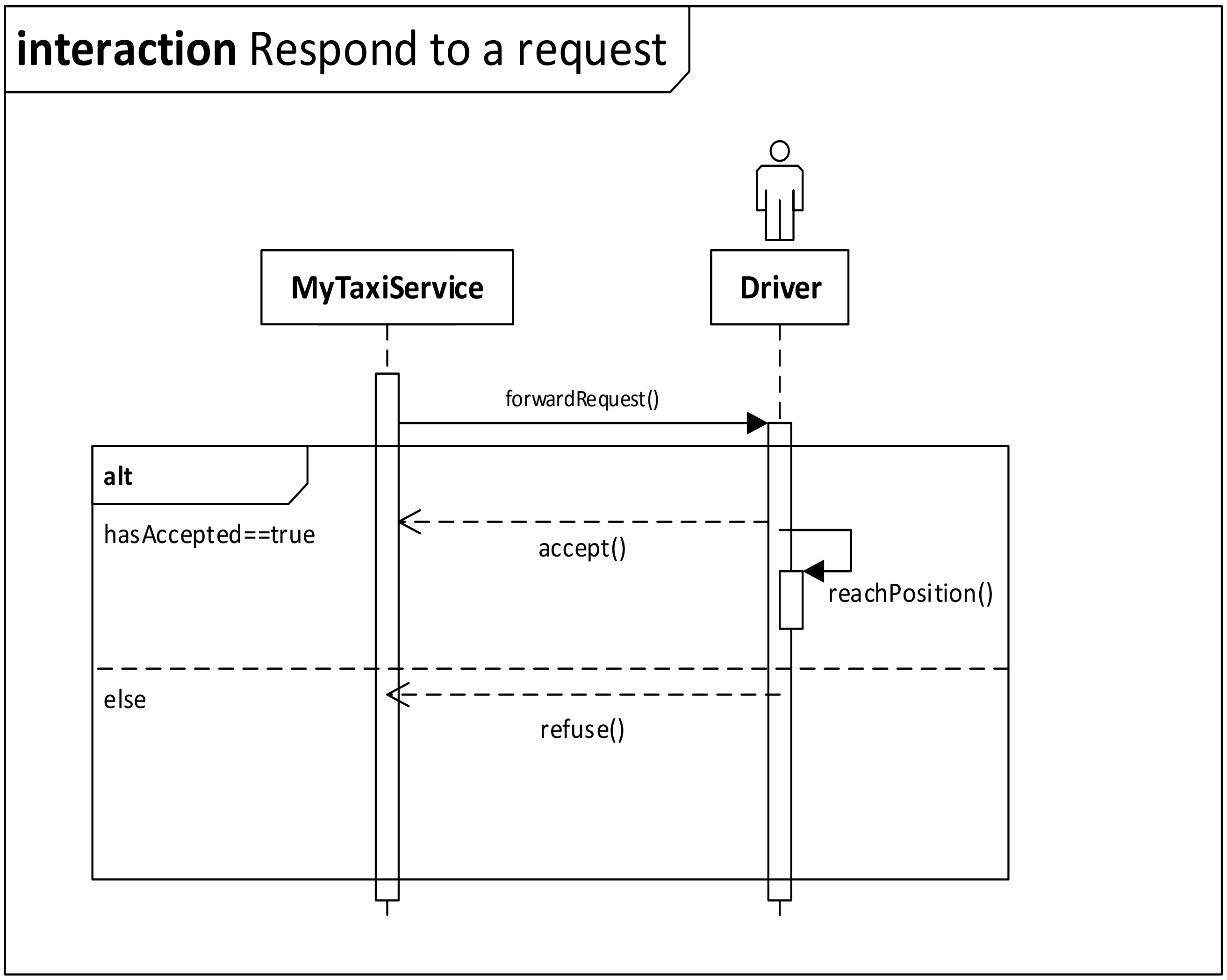
3.2.7.1 Functional requirements (Respond to a request)

[R1] Driver must respond to a request (he have to choose between accept or reject)

3.2.7.2 Use Case Diagram (Respond to a request)



3.2.7.3 Sequence diagram (Respond to a request)



3.3 Performance Requirements

- *Every user's request submission has to be successfully confirmed in at most 2 minutes; otherwise the user is notified his submission is discarded.*
- *User's login must be successful in at least 10 seconds.*
- *There's no limitation on registered users*
- *The system has to support at least 500 user connected in the same time*

3.4 Software System Attributes

3.4.1 Reliability

The system's reliability is strictly related to the server's one, so the server must work properly 24 hours every day.

3.4.2 Availability

The service must be up with a probability of at least 97%.

3.4.3 Security

The communication between clients and server must happen by a SSL protocol even more all the data must be properly encrypted. Most of all the system has to be not exposed to dangers like data steals such as man in the middle situations or SQL injections.

3.4.4 Maintainability

The software must be developed with a proper documentation with the JavaDoc and even more it must be developed with the Model-View-Control pattern, that ensures a great opportunities of extension.
However it must provide a programmatic interface for providing the chance to extends the service with new functions.

3.4.5 Portability

3.4.5.1 Hardware

- *Web application:*
  - *Laptop*
  - *2 Mb/s internet connection*

- *Mobile app:*
  - *2 GB of Ram*
  - *3g connection, at least 2 Mb/s*

3.4.5.2 Software

- *Web application:*
  - *Any O.S. supporting a web Browser*

- *Mobile app:*
  - *Android 4.0 or more*
  - *iOS 6.0 or more*

3.5 Other Requirements

- *After a ride request, the customer must receive the driver confirmation in at most 5 minutes.*
- *In case of reservation, the taxi arrives in the origin location with at most 10 minutes of delay.*

4. Alloy

We used alloy, for checking our model consistency and build up the overall UML class diagram previously shown. In fact, this top-down strategy is very useful and also has improved our class diagram and the right relation between classes.

4.1 Signs

```
//SIGNATURE

abstract sig User{
        name: one Stringa,
        surname: one Stringa,
        email: one Stringa,
        password: one Stringa,
        cf: one Stringa,
        dateOfBirth: one Stringa,
        birthPlace: one Stringa,
        telephoneNumber: lone Stringa,
        myReservations: set Reservation,
        myRides: set Ride,
}

sig Stringa{
}

sig Customer extends User{
        accountNumber: one Int,
}{ accountNumber>0}

sig Driver extends User{
        isAvailable: one Bool,
        taxiCode: one Int,
        taxiLicense: one Stringa,
        currentArea: one Area,
}{taxiCode>0}
```

```
abstract sig Request{
        completed: one Bool,
        passenger: one Customer,
        cabman: one Driver,
        relatedQueue: one Queue,
        origin: one Stringa,
        id: one Stringa,
        initTime: one Int,
        finishTime: one Int,
}{ initTime>0 and finishTime>initTime}

sig Ride extends Request {
        waitTime: one Stringa,
}

sig Reservation extends Request{
        destination: one Stringa,
        date: one Stringa,
}

abstract sig City{
        serialNumber: one Stringa,
        areas: set Area,
}

sig Area{
        identification: one Stringa,
        drivers: set Driver,
        queues: set Queue,
        cityBelonging: one City,
}

sig Queue{
        location: one Area,
        driversOnQueue: some Driver,
        request: one Request,
}
```

4.2 Facts

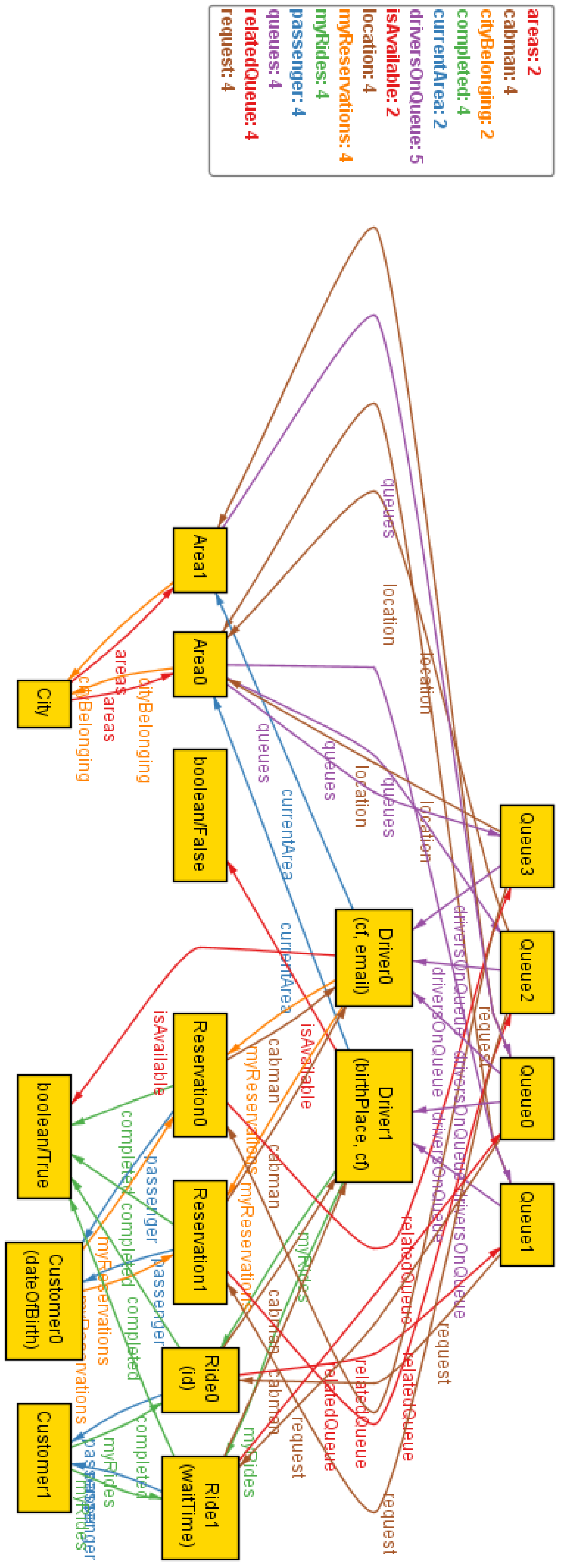
```
//FACTS

fact UniqueUser {
        no u1, u2: User | (u1!=u2 and u1.cf=u2.cf and u1.email=u2.email)
}

fact UniqueCustomerIdentifier {
        all c1, c2: Customer | (c1!=c2 implies c1.accountNumber=c2.accountNumber)
}

fact UniqueIDRequest{
        no r1, r2: Request | (r1!=r2 and r1.id=r2.id)
}

fact UniqueRequestForQueue{
        all r1, r2: Request | (r1!=r2 implies r1.relatedQueue!=r2.relatedQueue)
}

fact UniqueDriver {
        no d1, d2: Driver | (d1!=d2 and d1.taxiCode=d2.taxiCode and
d1.taxiLicense=d2.taxiLicense)
}

fact CorrispondenceQueueArea{
        all q: Queue | all a: Area | (q.location= a iff (a.queues & q) =q)
}

fact UniqueQueuePerRequest{
        all q1, q2: Queue | (q1!=q2 implies q1.request!=q2.request)
}

fact CorrespondenceQueueRequest{
        all q: Queue | all r: Request | (q.request = r iff r.relatedQueue= q)
}

fact UniqueCitySerialNumber{
        no c1, c2: City | (c1!=c2 and c1.serialNumber=c2.serialNumber)
}

fact UniqueAreaId{
        all a1,a2: Area | (a1!=a2 implies a1.identification!=a2.identification)
}

fact DriversAvailableInQueue{
        all q : Queue | all dr : Driver | all r : Request |
 (dr.myReservations + dr.myRides)&q.request = r implies (q.driversOnQueue & dr) = dr
}

fact NoRequestOverlapping{
        all u: User | (all r1, r2 : u.myRides+u.myReservations | r1!=r2 implies
        (r1.finishTime<r2.initTime or r2.finishTime<r1.initTime))

}

fact CorrespondenceUserRequest{
        all r: Request | all c: Customer | (r.passenger=c implies
(c.myRides+c.myReservations)&r=r)
        all r: Request | all dr: Driver | (r.cabman=dr implies (dr.myRides+dr.myReservations)&r=r)
}

fact AreaBelongsToOneCity{
        all a: Area | all c: City | (a.cityBelonging=c iff (c.areas)&a = a)
}
```

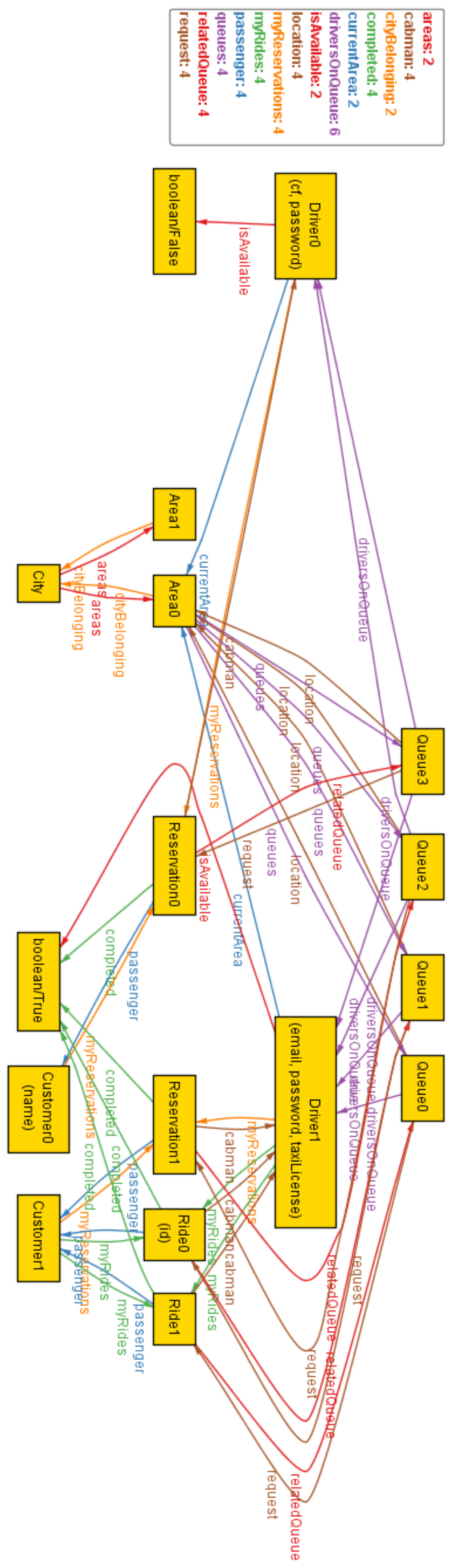
```
fact DriverHasAtLeastOneRide{
        all dr: Driver | (all r1, r2 :  dr.myRides| r1!=r2 implies (not(r1.completed=False and
r2.completed=False)))
}

fact CustomerHasAtLeastOneRide{
        all c :Customer | (all r1, r2 :  c.myRides| r1!=r2 implies(not(r1.completed=False and
r2.completed=False)))

}

fact DriverNotAvailable{
        all dr : Driver | all r : dr.myRides+dr.myReservations | (r.completed=False implies
dr.isAvailable=False)
}

fact CustomerTakeAtLeastOneRequest{
        all c: Customer | (all req1, req2 : c.myReservations| (req1.completed=False and
req2.completed=False) implies req1=req2)
}
```

4.3 Assertions

```
//ASSERTIONS

assert driverUnavailableDuringRide{
        all r : Request | r.completed= False implies r.cabman.isAvailable= False
}

assert driverInQueueAndInArea{
        all q :Queue | all dr : Queue.driversOnQueue | dr.currentArea= q.location
}

assert NoTemporalyContraddiciton{
        no r : Request | (r.initTime> r.finishTime)
}

assert noTwoRideActiveForDriver{
        all r1,r2 : Ride | ((r1.completed=False and r2.completed=False and r1!=r2) implies
r1.cabman!=r2.cabman)
}

assert noTwoCustomerInTheSameRide{
        all r1, r2 : Ride | r1!=r2 implies r1.passenger!=r2.passenger
}
```

4.4 Alloy Examples




areas: 2
cabman: 4
cityBelonging: 2
completed: 4
currentArea: 2
driversOnQueue: 6
isAvailable: 2
location: 4
myReservations: 4
myRides: 4
passenger: 4
queues: 4
relatedQueue: 4
request: 4
Driver0 (cf, password)
boolean/False
Area1
Area0
City
Queue3
Queue2
Queue1
Queue0
Reservation0
Reservation1
boolean/True
Customer0 (name)
Customer1
Driver1 (email, password, taxiLicense)
Ride0 (id)
Ride1
isAvailable
currentArea
driversOnQueue
cityBelonging
areas
cabman
queues
location
myReservations
relatedQueue
request
completed
passenger
myRides

5. Appendix

5.1 Tools

- *Microsoft Word : Used for create this document*
- *Microsoft Visio Professional : For all UML diagrams such as Class, Use Cases, Sequence and State Chart*
- *Alloy: For modeling the structure of the software*
- *Balsamiq Mockups 3: For generating the mockups*
- *Microsoft OneDrive: Used for sharing files among the group*

5.2 Hours of Work

- Francesco : 35
- Marco : 35