myTaxiService
Requirement **A**nalysis and **S**pecification **D**ocument

A.Y. 2015/2016

Politecnico di Milano

Version 1.1

Cattaneo Michela Gaia, matr. 791685
Barlocco Mattia, matr. 792735

November 6, 2015

Contents

1 Introduction

1.1 Purpose

This document represents the Requirement Analysis and Specification Document (RASD). The aim of this document is to describe the system in terms of functional and non-functional requirements, modeling the system according to the real needs of the customer and showing the constraints and the limit of the software. This document serves as a contractual basis between the customer and the developer.

1.2 Actual system

The government of a large city has commissioned a system that is able to manage and optimize the taxi service.

1.3 Scope

The system aims at simplifying the access of the passengers and guaranteeing a fair management of taxi queues. In order to manage the queue optimization, the city is divided in zones, each one associated to a taxi queue, and the system computes the taxi distribution basing on their GPSs information.
The system allows the user of the application to request a taxi in an area of the city, even without logging in, guaranteeing a fast way to access the service. As soon as a taxi driver takes the request, the application notifies the user of the ID of the incoming taxi and the waiting time. The user can also register, by filling in a form, and exploit additional functionalities such as choose the payment method he prefers or make reservations: he only needs to indicate the origin and destination of the ride and the exact time.
On the other hand, the taxi driver always has to notify his availability and let the system know if he is going to take a certain request by a passenger. If the driver is not able to confirm, the system will forward the request to the second in the queue and will move this taxi driver in the last position of the queue.

1.4 Actors

The actors participating in the system are:

- Visitor: a person who can see the home page of the application, he can register, look up for information about the system and request a taxi, after providing e-mail address and phone number.

- Passenger: a regular registered user who wants to use the application in order to request a taxi in a specific area or make a reservation for a given time.

- Taxi Driver: a registered user who logged in with a specific ID, provided by the system. This area is reserved to taxi drivers who needs to use the application in order to inform the system of their availability or about what request they are going to take care of.

1.5 Goals

A user should be able to:

- Request a taxi.
- Check the waiting time.
- Visualize the information about the system.
- Sign up into the system.

A passenger should be able to:

- Log into the system.
- Reserve a taxi.
- Choose the payment method.

The taxi driver should be logged in with a specific ID and should be able to:

- Log into the system.
- Change his state from available to unavailable and vice versa.
- Visualize the queue of his area.
- Accept or decline the request that the system forwards to him.
- Visualize the position of the user he needs to reach.

1.6 Domain properties

We suppose that the following properties hold in the analyzed domain.

- We assume that there are not previous implementations managing this service.
- The visitor is expected to be where he made the request to the system when the taxi driver arrives.
- The passenger is expected to be where and when he made the request to the system when the taxi driver arrives.
- The passenger is not supposed to cancel the reservation less than 2 hours earlier he has submitted it.
- A taxi driver always inform the system that he is available only when he actually is.
- A taxi driver, who notifies that he is going to take a certain request, always carries it out.
- At least one taxi driver in a queue accept the incoming request within few minutes.
- When a taxi driver finishes a ride, he instantly changes his state from unavailable to available.
- If a user requests a service from the system, it will be within the borders of the city.

2 Overall Description

2.1 Product perspective

The product consists in a web application and a mobile app both based on the web. The web application is used only by the users, the mobile app instead is used also by taxi drivers.
There is a server which manages all the requests of the users along with the answers and the availability of the taxi drivers, computing their area distribution. There are several clients (visitors, passengers or taxi drivers) interacting with the server by using a graphical user interface, but there is no implementation of interfaces for administration. In fact, the application is only user-based.

2.2 User characteristics

The user can be a person who wants to request a taxi or access the system in order to use additional functionalities, such as make a reservation or easily request a taxi without having to give his information every time he exploits the service. The mobile app is used also by the taxi drivers who needs to receive the requests of the customers and to interact with the system, answering to the calls and updating their availability and their GPS position .

2.3 Constraints

2.3.1 Regulatory policies

The system must ensure the privacy of the users, without publishing their personal information.

2.3.2 Hardware limitations

The system requires the users to use a device provided with a GPS detector and an Internet connection.

2.3.3 Interfaces to other applications

The system interfaces with three applications:

- Google Maps in order to determine the position of the taxi drivers and the users.
- Facebook if a user wants to sign in with his Facebook account.
- Google+ if a user wants to sign in with his Google+ account.

2.3.4 Parallel operations

The system supports parallel operations of different clients, who access the application and make requests or reservations of taxis.

2.3.5 Documents related

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

2.4 Assumptions

- The user can only make one request at a time, which means that the system does not allow him to make requests until the ride is over.
- The system changes the state of a taxi driver from available to unavailable if he accept a request, but not vice versa.
- If the passenger makes a reservation less then two hours in advance the system will not accept it.
- The user has to turn on his GPS in order to make any request.
- The system is able to manage the taxi queues in a way that there is always a taxi driver available for the reservation within ten minutes before the ride.
- The system deducts the cost of the ride from the credit card of the passenger only when the ride is over, sending also the receipt of the payment to the e-mail address indicated in the passenger profile.
- If the first taxi of the queue does not take the request, the system will send the request to the second taxi of the queue and, at the same time the first taxi will be moved in the last position of the queue.
- The taxi driver has one minute to answer the request, if he does not make it in time, the system considers it as declined.
- The profile of the passengers and of the taxi drivers are private, no other users or taxi drivers can visualize it.
- The system constantly updates the position of all drivers of all areas, assigning their position to the correct queue of the area.
- When a user registers, the system does not instantly confirm the creation of a new passenger profile, but a confirmation e-mail is sent to the e-mail address provided by the user.
- When a passenger makes a reservation, the system retains it, then it forwards it to the first taxi driver in the queue of the designated area, ten minutes earlier the specified time.
- The taxi driver can not register in the application, but he will be added in the database, providing him his ID.
- There is always an available taxi driver in every area.

2.5 Future possible implementations

The application is open to future implementations, which could be:

- Taxi sharing: this is an additional functionality that allows different passengers to share the taxi with other passengers. The system must be notified that the passenger wants to use the taxi sharing option and it will ask him to indicate the destination of this rides. When it is all settled, the system is able to check if there are other persons in the same area that wants to reach the same destination, informing the taxi driver and the passengers. This can be a convenient solution, as the cost of the ride would be equally shared among the passengers.

- Price transparency: this is an additional functionality that allows the passenger to calculate the price of the ride in advance, inserting its origin and destination.

- Meet the driver: this is an additional functionality that lets the passenger know which taxi driver has answered his request. The system will send the profile of the designated driver to the passenger, along with his name and data, so that the customer would be able to recognize him or even call or text him if necessary.

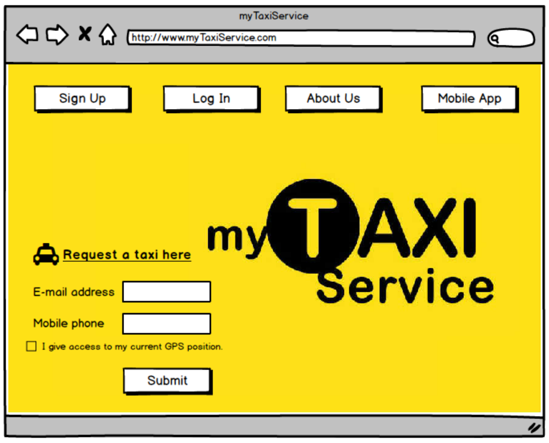
- Give us feedback: this is an additional functionality that allows the passenger to rate the driver after the ride and leave comments about it.

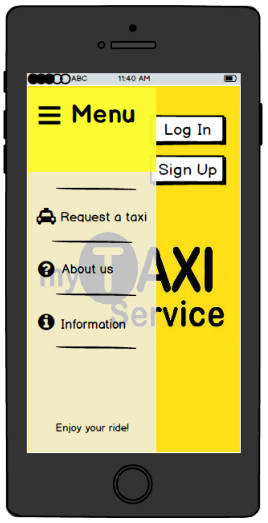
3 Specific Requirements

3.1 External interfaces requirements

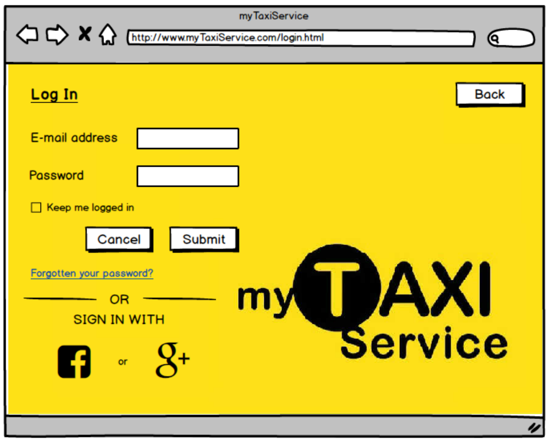
3.1.1 User interfaces

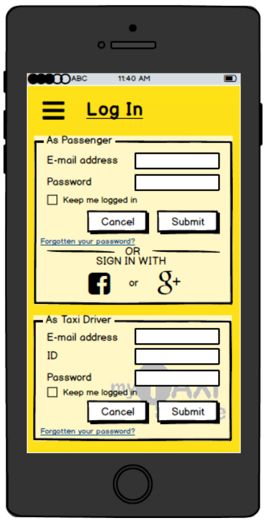
- **Home page** This mockup represents the home page of the mobile app and of the web application, where a user can request a taxi or have access to the log in or registration page, or look up information about the service.



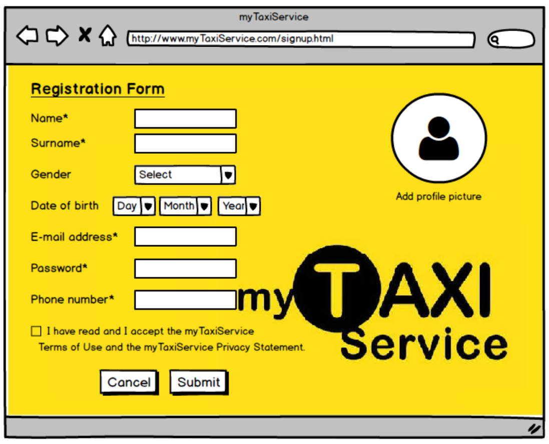


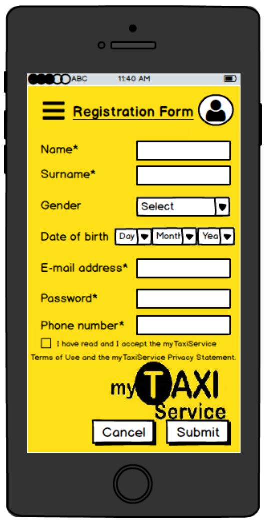
- **Log in** This mockup represents the log in page of the mobile app and of the web application, where a user can log into the application filling in the form or accessing with the Facebook or Google+ account. The taxi drivers can use only the mobile application.



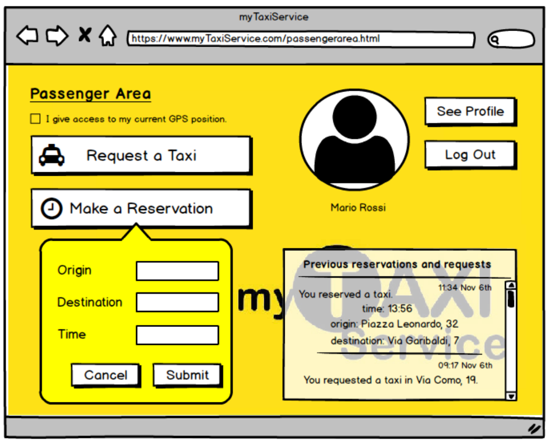


- **Registration** This mockup represents the sign up page of the mobile app and of the web application, where a user can register, filling in the form with his personal data.





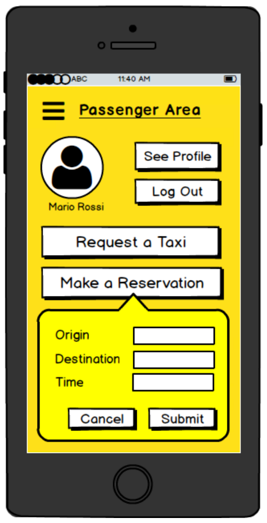
- **Passenger Area** This mockup represents the passenger area of the mobile app and of the web application, where a passenger can request or reserve a taxi or access to his profile.



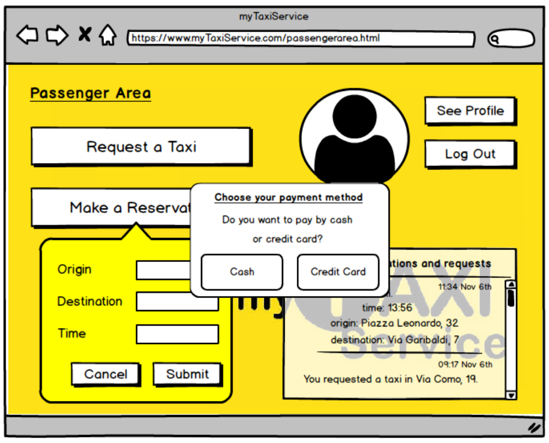
Previous reservations and requests

11:34 Nov 6th
You reserved a taxi.
time: 13:56
origin: Piazza Leonardo, 32
destination: Via Garibaldi, 7

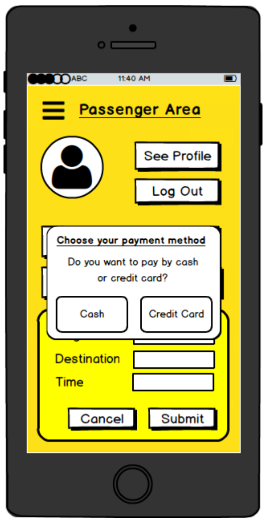
09:17 Nov 6th
You requested a taxi in Via Como, 19.



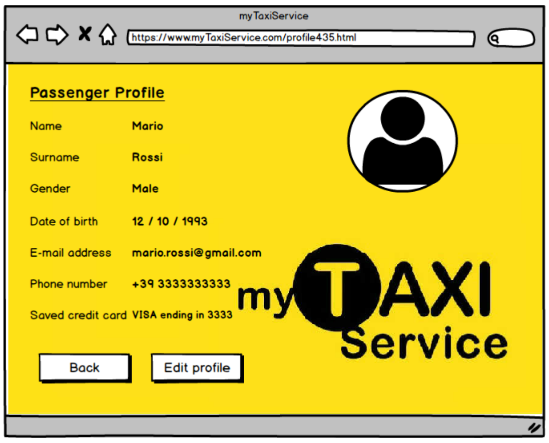
- **Payment Method** This mockup represents the passenger area of the mobile app and of the web application, where a passenger can choose the payment method of his request or reservation.



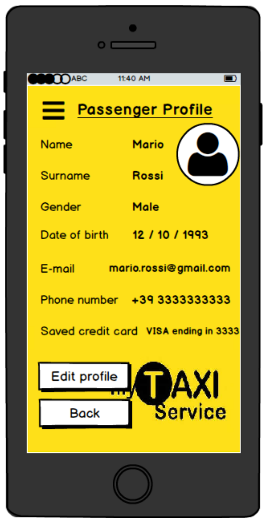
11:34 Nov 6th
time: 13:56
origin: Piazza Leonardo, 32
destination: Via Garibaldi, 7
09:17 Nov 6th
You requested a taxi in Via Como, 19.



- **Passenger Profile** This mockup represents the passenger profile mobile app and of the web application, where a passenger can see or modify his profile.

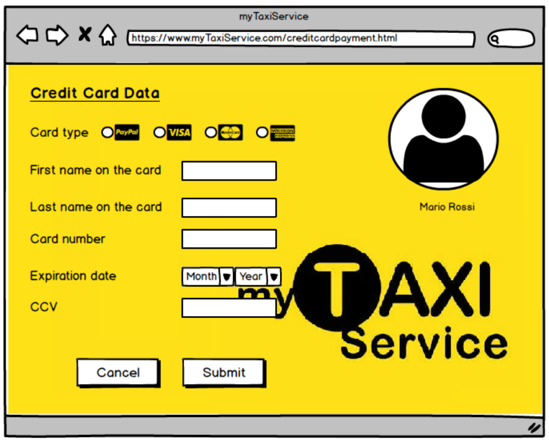


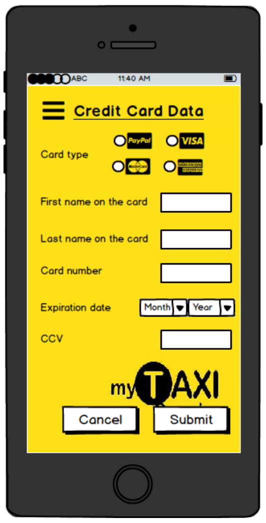
Name	Mario
Surname	Rossi
Gender	Male
Date of birth	12 / 10 / 1993
E-mail address	mario.rossi@gmail.com
Phone number	+39 3333333333
Saved credit card	VISA ending in 3333



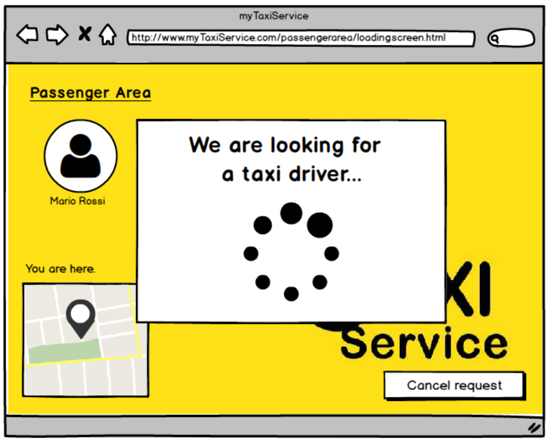
Name	Mario
Surname	Rossi
Gender	Male
Date of birth	12 / 10 / 1993
E-mail	mario.rossi@gmail.com
Phone number	+39 3333333333
Saved credit card	VISA ending in 3333

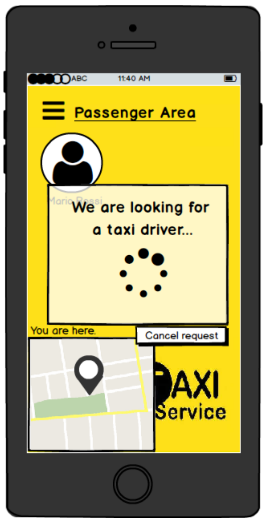
- **Credit Card Data** This mockup represents the passenger area of mobile app and of the web application, where a passenger can insert his credit card data.



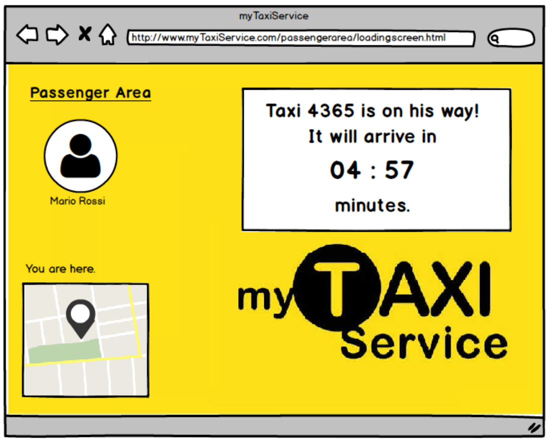


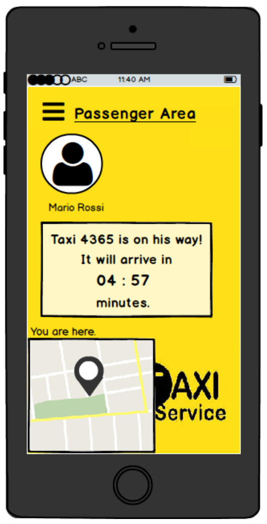
- **Loading Screen** This mockup represents the passenger area of the mobile app and of the web application, where a passenger waits for his request to be taken in consideration and can cancel it.


myTaxiService
http://www.myTaxiService.com/passengerarea/loadingscreen.html
Passenger Area
Mario Rossi
We are looking for a taxi driver...
You are here.
Service
Cancel request

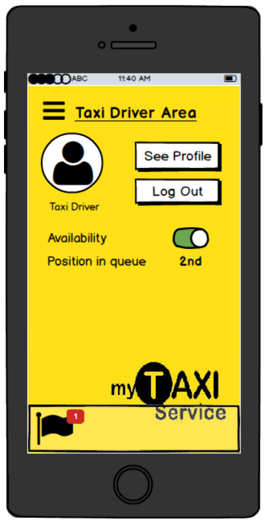


- **Waiting Time** This mockup represents the passenger area of the mobile app and of the web application, where a passenger see how much time he has to wait until the taxi arrival.

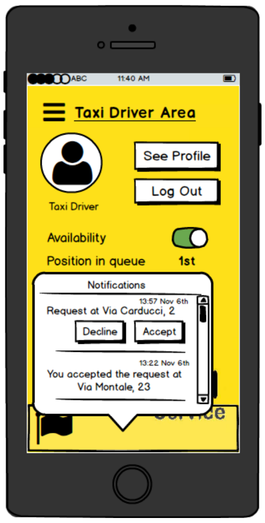




- **Taxi Driver Area** This mockup represents the taxi driver area of the mobile application, where a taxi driver can set his availability, see his position in the queue and access to his profile.



- **Taxi Driver Notifications** This mockup represents the taxi driver area of the mobile application, where a taxi driver can see the notifications of the forwarded requests and accept or decline them.



13:57 Nov 6th
Request at Via Carducci, 2

13:22 Nov 6th
You accepted the request at Via Montale, 23

3.1.2 API interfaces

The system exploits the Google Map APIs in order to track the movements of the taxi drivers and to get the position of the users that request a service. It is necessary to be always aware of the taxi position in the areas and to make sure that the system is able to calculate the exact time the user has to wait until the taxi arrival. Facebook and Google+ APIs are used to allow the users to log into the application without filling in any registration form or providing any further personal information.

3.1.3 Hardware interfaces

This project does not support any hardware interfaces.

3.1.4 Software interfaces

- Database Management System (DBMS):

- Name: MySQL.

- Version: 5.6.21

- Source: http://www.mysql.it/

- Java Virtual Machine (JVM).

- Name: JEE

- Version: 7

- Source: http://www.oracle.com/technetwork/java/javaee/tech/index.html

- Application server:

- Name: Glassfish.

- Version: 4.1.

- Source: https://glassfish.java.net/

- Operating System (OS).

- Application must be able to run on any SO which supports JVM and DBMS specified before.

3.1.5 Communication interfaces

Protocol	Application	Port
TCP	HTTPS	443
TCP	HTTP	80
TCP	DBMS	3306(default)

3.2 Functional requirements

- **Registration of the visitor.** The system displays the registration form to the user, who needs to fill in the mandatory fields of the form and press the "Submit" button. The user should provide his name, surname, password, e-mail address and telephone number. He can choose whether let the system know about his gender and date of birth or not.
- **E-mail confirmation.** The system sends a confirmation e-mail is sent to the e-mail address provided by the user. The user can follow the link written in the e-mail and the system creates the new account.
- **Look up information about the system.** The system displays a screen where the user can read the data of the application, in particular the description of its functionalities, the contacts of the managers of the system and information about the investors.
- **Log in as a passenger.** The system provides an input form where the passenger can insert e-mail and password. The system checks if the account is already registered and if the password is correct, if not an error message is displayed. If this conditions hold, the passenger logs in successfully.
- **Log in as a taxi driver.** The system provides an input form in the mobile application where the taxi driver can insert e-mail, ID and password. The system checks if the account is correctly registered, if the ID matches the e-mail address and if the password is correct, if not an error message is displayed. If this conditions hold, the taxi driver logs in successfully.
- **Log in with social networks** The system allows the user to sign up with his Facebook or Google+ account without inserting e-mail, telephone number and the other information, if he has already inserted them on his profile.
- **Retrieve password.** The system provide a link to recover the password of the account. The user clicks on the link and receives a recover e-mail with a link that redirects to a page where he can set his new password.
- **Request a taxi.** The user can make taxi requests enabling the GPS of his device so that the system can determine his position and send the call to the first taxi driver of the queue of the designated area. The visitor can exploit this service in the home page of the application, inserting his e-mail and telephone number. The passenger can use this service in the passenger area, displayed after the log in, without inserting any further information.
- **Reserve a taxi.** The system provides a "Make a reservation" button in the passenger area. The passenger presses this button and an input form is displayed, where he should indicate the origin, destination and time of the ride.

- **Choose payment method.** The system, after the passenger submits a request or a reservation, displays a panel to him, where he can choose whether to use cash or credit card as payment method. If the passenger decides to pay by cash, the taxi driver will take care of the payment. Instead the passenger needs to add his credit card data only if this is the first time he chooses this payment method, in fact the system adds the credit card data to his profile after the insertion and does not ask anymore.

- **View the profile.** The passenger or the taxi driver can see his profile by pressing the "See profile" button. The system displays a screen with his personal data already inserted and his profile picture.

- **Edit the profile.** The passenger can edit his profile by pressing the "Edit profile" button and the system displays a screen where he can change his name, surname, e-mail address, password, phone number and profile picture. He can edit or add his gender, date of birth and credit card data. In case the passenger modifies his e-mail address, the system will send an e-mail confirmation.

- **See waiting time.** The system displays to the user who has made a request the waiting time panel, from the time a taxi driver is found until he arrives. The user can see the panel to know how much time he has to wait till the taxi arrival.

- **Change availability.** The taxi driver can switch his state from available to unavailable and vice versa. The system does not consider him in the queue as long as his state is unavailable and, if he is available, the system shows him his queue positioning and keeps him in consideration for the requests of the users.

- **Notification of the requests.** The taxi driver can see the requests by a user as notifications in the notification panel. The system sends him a notification with the position of the user, the taxi driver can decide to accept the request or refuse it.

- **See position in the queue.** The system provides the taxi driver his position in the queue in his area, if his state is set to available, so that he is always able to know if there can be any incoming requests.

3.3 Non-functional requirements

3.3.1 Performance requirements

The system is composed of a server side hosting the database, where all the data of the passenger and of the taxi driver are stored, and it handles all the HTTP request. There is not a great amount of data to manage: there are the profile information of the passengers, of the taxi drivers and the log of the activities; so the size of the database is not a constraint for the system.
In order to provide a suitable service, the system has to be reactive and able to answer to a high number of simultaneous requests.
There are no time constraints that depends on the system, the time to process a transaction is less then one second. The only time constraint would be the reply message of the taxi driver that depends on his reflexes.

3.3.2 Software system attributes

1. Reliability: the server must remain up twenty-four hours a day in order to guarantee a suitable service. The system allows the administrators to fix any problem without compromising the functionality of the system.

2. Availability: in order to guarantee the continuous availability of the data, it has been arranged one backup system on a secondary database and the other on an external storage.

3. Security: privacy of the data exchanged between the server and the clients is guaranteed by a SSL encryption. The passwords and the sensitive data stored on the database are protected by MD5 encryption.

4. Portability: the client side of the system is compatible with the major mobile platforms on the market (e.g. Android, iOS) and any device (e.g. PC with Windows, Apple, etc.) that supports a browser and a GPS system.

4 Scenarios Identifying

4.1 Scenario 1: Registration and log in

Virginia is new to the city and is not used to use public transport, so she searches for the best taxi services via web. After a while, she decides to download the new application "myTaxiService", because it seems to offer the most interesting functionalities. Virginia registers in the application, without even filling the form, accessing with her Facebook account and adds her favorite photo with her boyfriend. Now she is able to log in, to make taxi reservations choosing the payment method she prefers and to request a taxi without having to insert her information every time she needs it.

4.2 Scenario 2: Password recovery

Charlie has registered in the application when he visited the city a few months ago, now he is back and needs to make a taxi reservation for the afternoon. But Charlie is really absent-minded and is not able to remember his password. He is very happy to see the "Forgotten your password?" link and he press it: this leads to another page where the system tells him that an e-mail for the recovery has been sent to his e-mail address. He checks his inbox and follows the link to set his new password. Now he can easily log in and make his reservation.

4.3 Scenario 3: Choose payment method

William often uses myTaxiService for the taxi requests, as it is a fast and reliable service. He is a very lazy person and does not want to register, as the application allows visitors to easily request a taxi, but he is very tired of paying by cash, so he finally make up his mind. He registers and adds his credit card data to his profile. William is happy to save his time and he can request or reserve a taxi without even worry about the money spent.

4.4 Scenario 4: Reserve a taxi

Pierre asks his friend next door, Marie, out for a date at the restaurant and then at the cinema. He really wants to make a good impression, but his car is at the mechanic. So Pierre decides to go anyway by bus, even tho it is not the most elegant solution, but when he will come back there will not be any public transport. He decides to register in the myTaxiService application and make a reservation for 00:30 pm, when the movie ends. The taxi is already there when they come out of the cinema and they are home in a few minutes, Pierre and Marie are very satisfied with the service and with the evening too.

4.5 Scenario 5: Taxi driver availability

Jonathan is used to take a coffee after his first ride of the day. Even if he is the first of the queue, he wants to maintain this habit, so he just switch his availability button to off. Now the system knows that he cannot be called for the requests and he can set his state available whenever he wants and see his position in the queue. Finally, when Jonathan takes a call, he knows that he does not have to bother about his availability because the system always sets his state unavailable in the exact time he takes the request of a passenger.

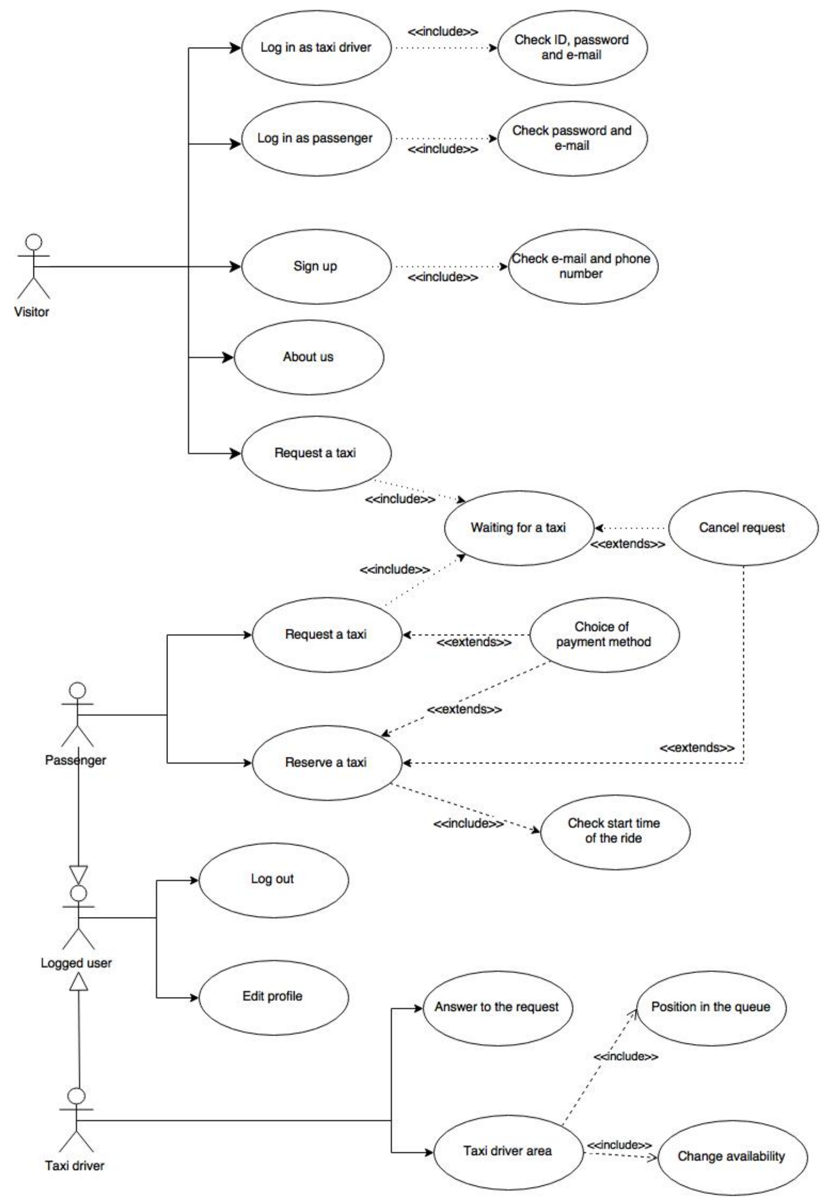
4.6 Scenario 6: Taxi drivers notification of the requests

Walter is the first of the queue, so he knows that he is going to take a request soon but he is on the phone with his mother and does not see the notification within the first minute, so the system forwards it to the second taxi driver of the queue. Now he is in the last position of the area and needs to be very careful when the positions change. Walter shifts fast to the first position again and he is now able to click on the "Accept" button before the time is over. He sees the position of the request and goes there with no trouble. At the end of the day he scrolls his notifications panel and he is very satisfied with his work today, as he has missed just one request.

5 UML Models

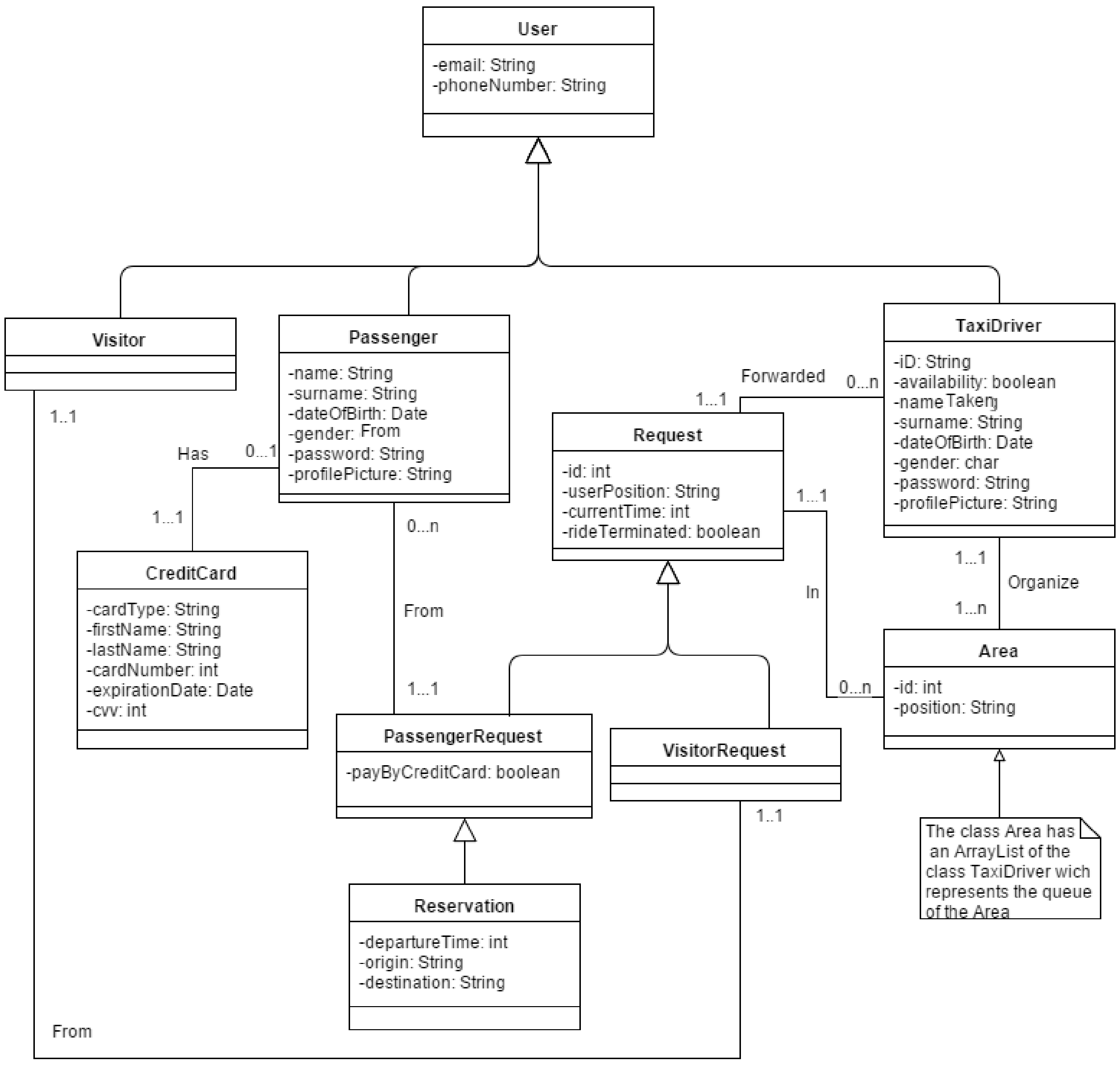
5.1 Use Case diagram

This Use Case diagram gives an overview on the main actors of the system and their functionalities.



5.2 Class diagram

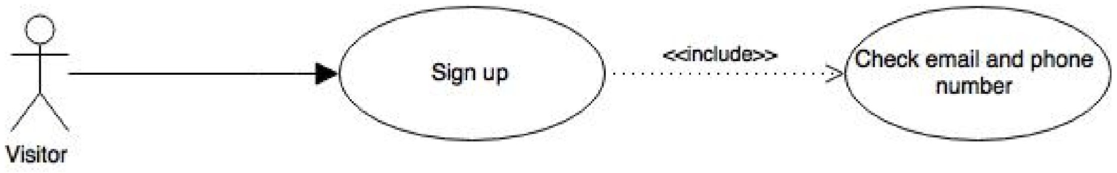
This Class diagram describes a basic structure of the entities of the system and their relations.

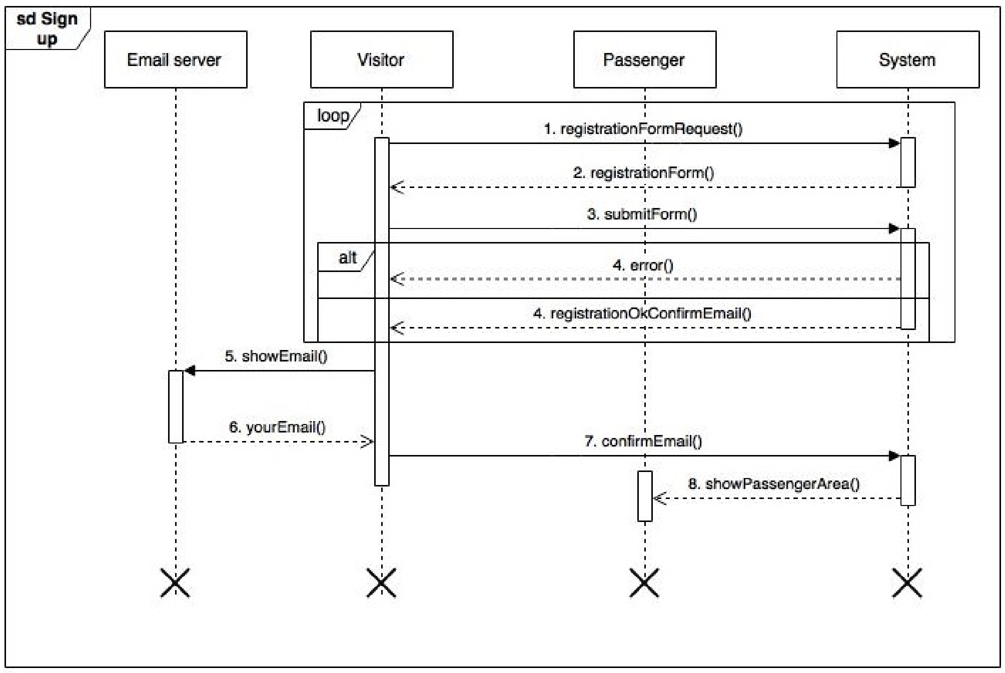


5.3 Sequence diagram

- User Registration

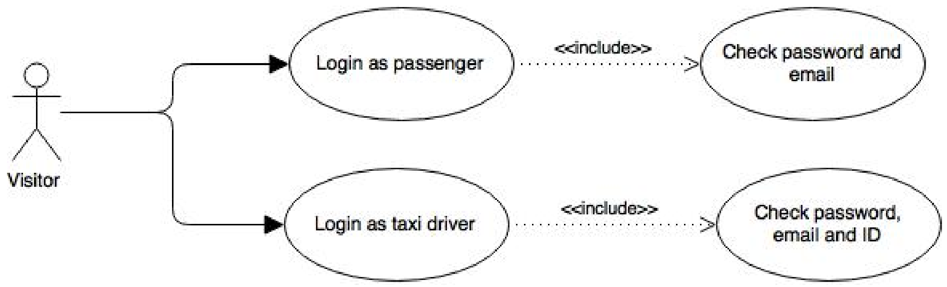
Actor	Visitor and passenger.
Goal	Sign up into the system.
Input Condition	The visitor is not already registered into the system.
Event Flow	1. The system shows the sign up page. 2. The visitor completes the sign up form and submits.
Output Condition	1. The system checks the information of th e visitor and allows him to log in 2. The visitor becomes a passenger.
Exception	If any information is incorrect, the system notifies the visitor and asks him to redo it.

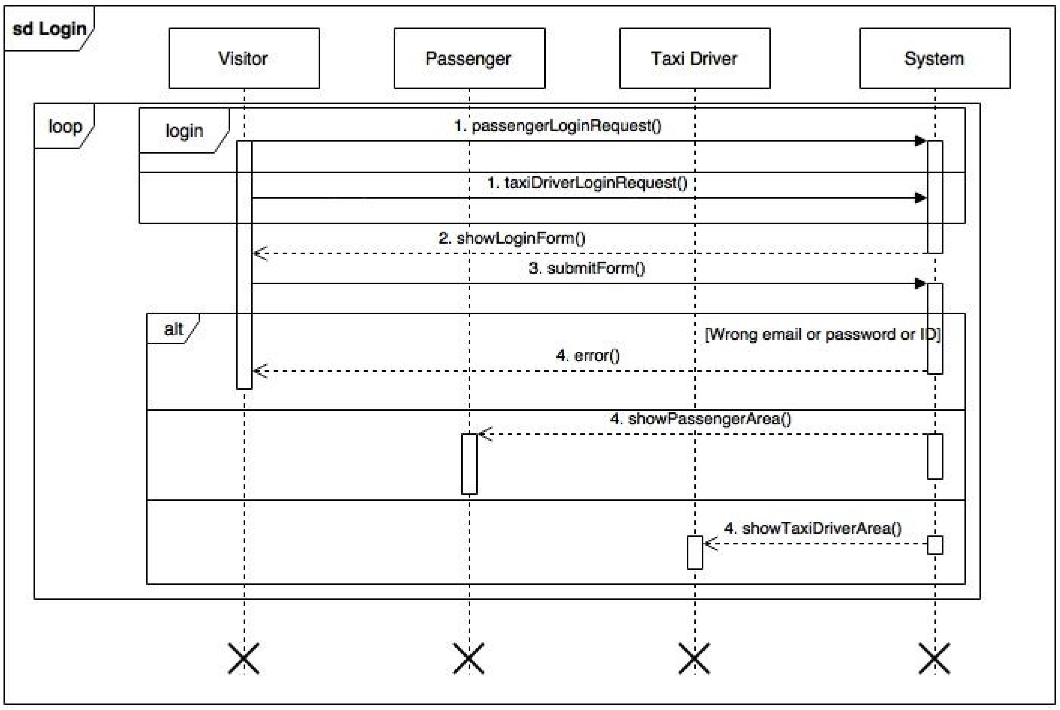



sd Sign up
Email server
Visitor
Passenger
System
loop
1. registrationFormRequest()
2. registrationForm()
3. submitForm()
alt
4. error()
4. registrationOkConfirmEmail()
5. showEmail()
6. yourEmail()
7. confirmEmail()
8. showPassengerArea()

- User Log In

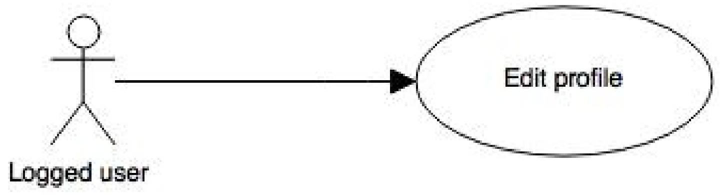
Actor	Visitor, taxi driver and passenger.
Goal	Log into the system.
Input Condition	The visitor is already registered into the system as a taxi driver or a passenger.
Event Flow	1. The system shows the login page. 2. The visitor completes the login form.
Output Condition	1. The system checks the credentials and shows the passenger or taxi driver area. 2. The visitor becomes a passenger or a taxi driver if the credentials are correct.
Exception	If the credentials are incorrect the system notifies the visitor.

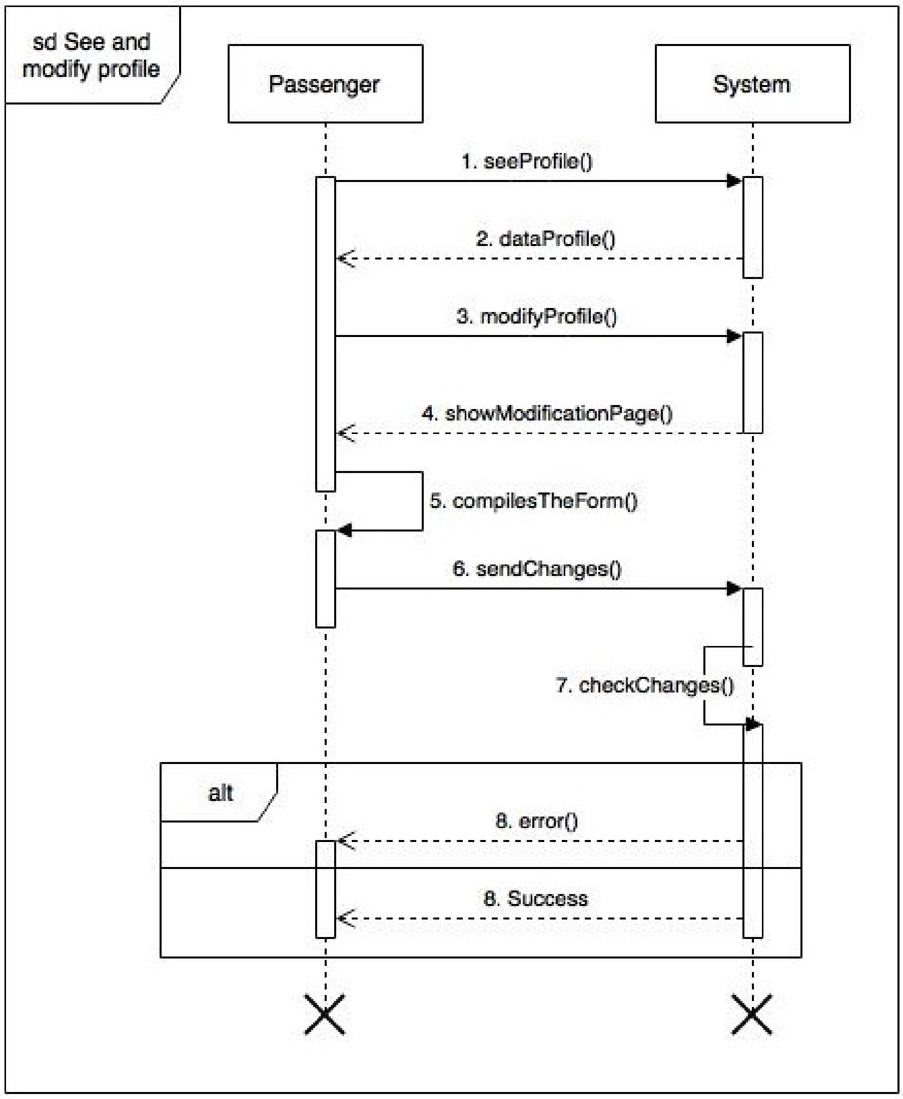



sd Login
Visitor
Passenger
Taxi Driver
System
loop
login
1. passengerLoginRequest()
1. taxiDriverLoginRequest()
2. showLoginForm()
3. submitForm()
alt
[Wrong email or password or ID]
4. error()
4. showPassengerArea()
4. showTaxiDriverArea()

- Passenger Profile

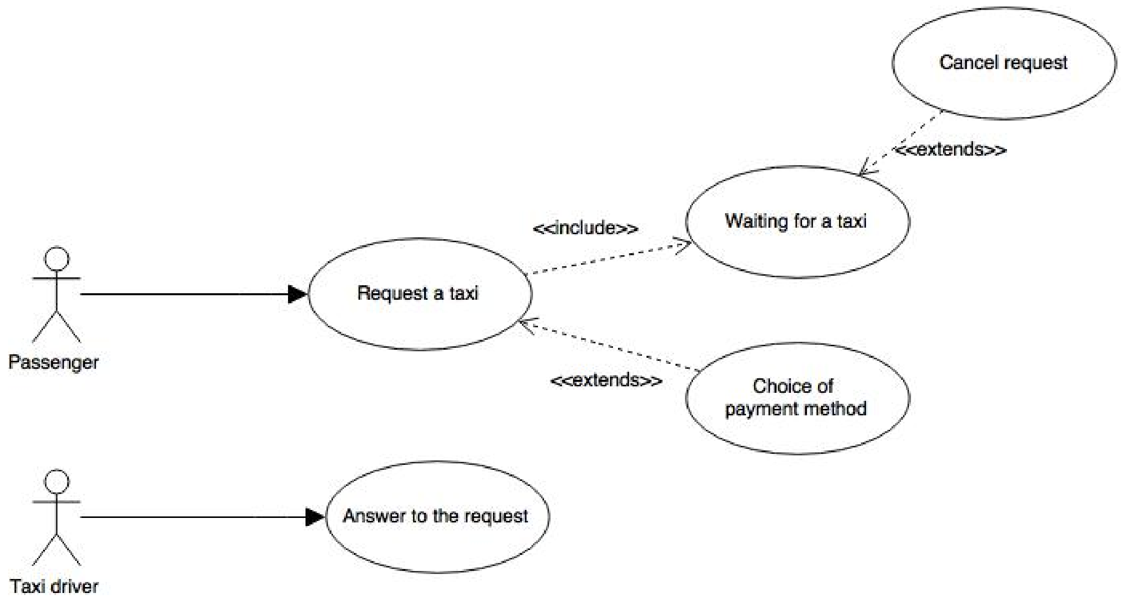
Actor	Passenger.
Goal	Edit his profile information.
Input Condition	The passenger is already logged in.
Event Flow	1. The system shows the Edit profile page. 2. The passenger modifies some information and submits the changes.
Output Condition	The system checks the changes and shows the passenger area.
Exception	If any information is incorrect, the system notifies the passenger and cancels the changes.

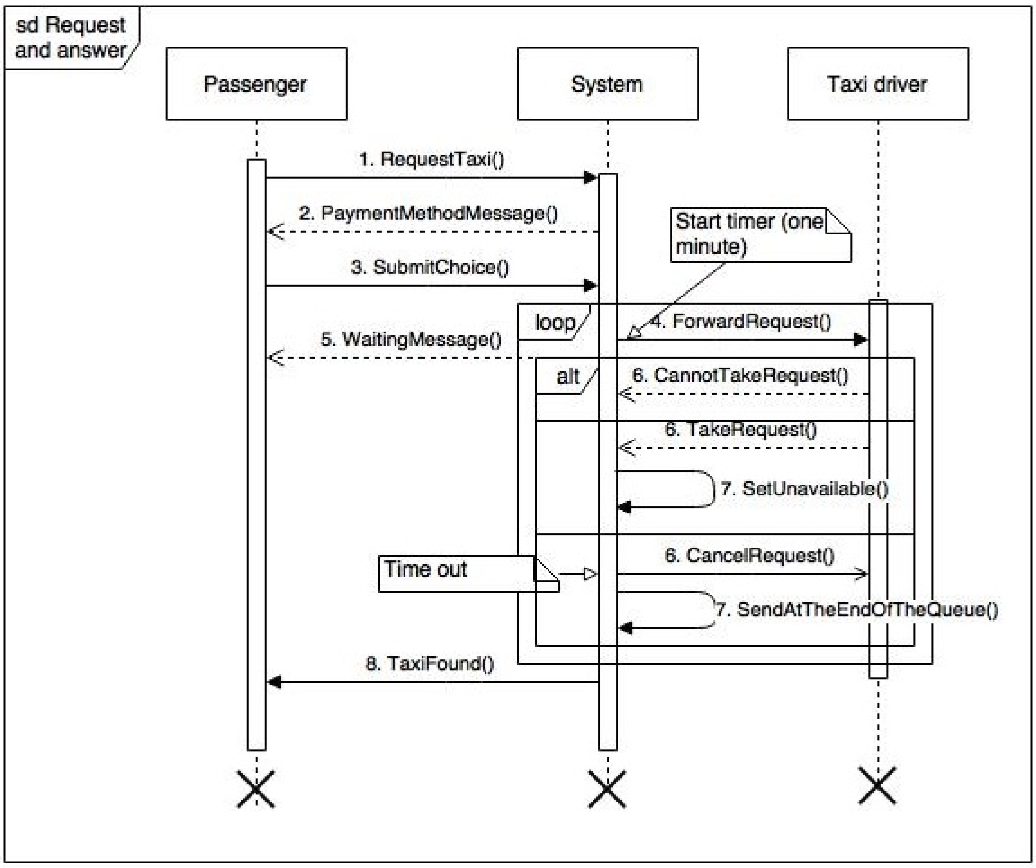



sd See and modify profile
Passenger
System
1. seeProfile()
2. dataProfile()
3. modifyProfile()
4. showModificationPage()
5. compilesTheForm()
6. sendChanges()
7. checkChanges()
alt
8. error()
8. Success

- Request By A User

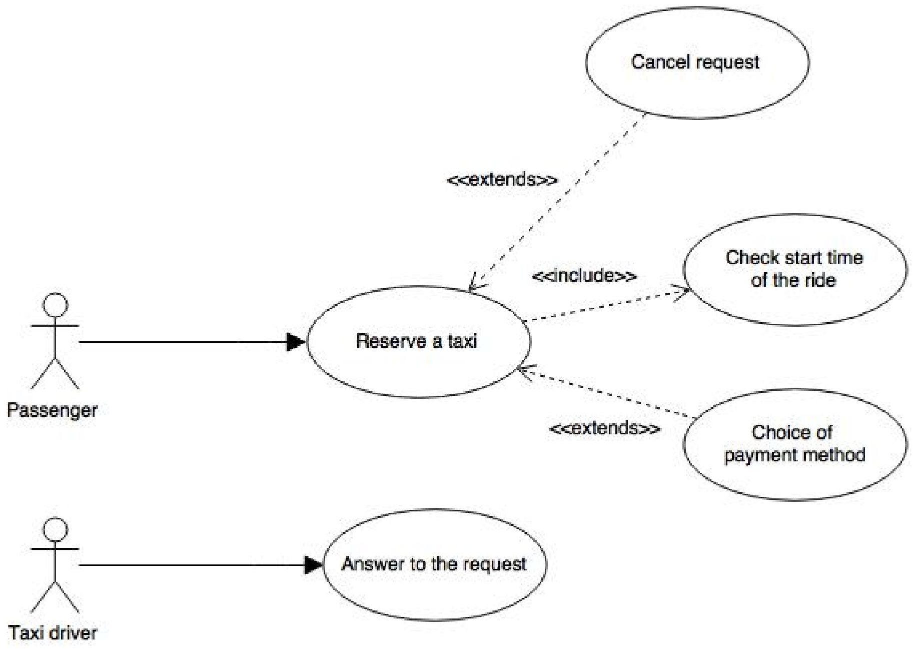
Actor	Taxi driver and passenger.
Goal	Request a taxi and answer to the request.
Input Condition	The passenger is already logged in and he has not made any request or reservation recently.
Event Flow	1. The passenger presses the request button. 2. The passenger waits for the system to look for an available taxi driver. 3. The taxi driver answers to the request. 4. The system sends a confirmation to the passenger if a taxi driver has accepted the request.
Output Condition	When the system finds a taxi driver, it notifies the passenger.
Exception	If a taxi driver declines the request, the system continues the research.

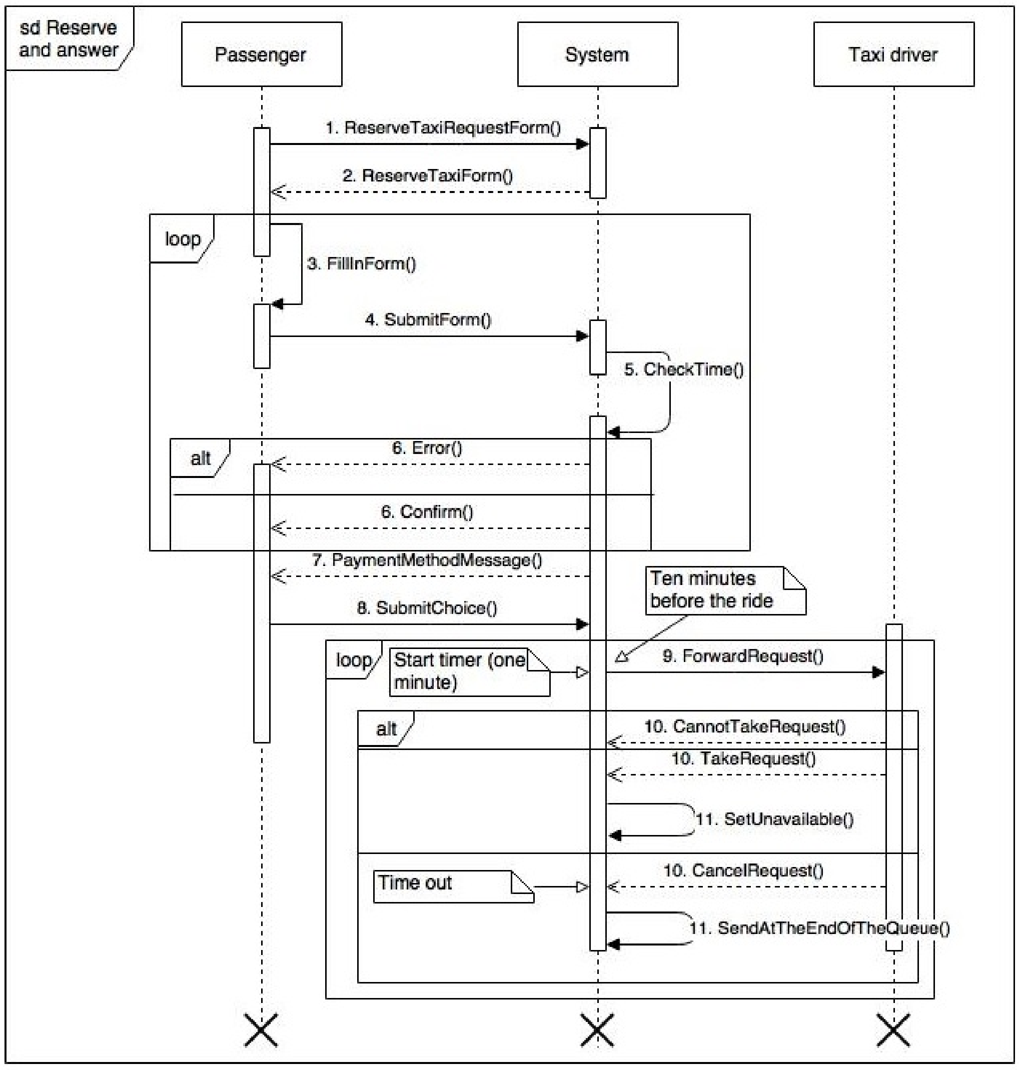



sd Request and answer
Passenger
System
Taxi driver
1. RequestTaxi()
2. PaymentMethodMessage()
Start timer (one minute)
3. SubmitChoice()
loop
4. ForwardRequest()
5. WaitingMessage()
alt
6. CannotTakeRequest()
6. TakeRequest()
7. SetUnavailable()
Time out
6. CancelRequest()
7. SendAtTheEndOfTheQueue()
8. TaxiFound()

- Reservation By A Passenger

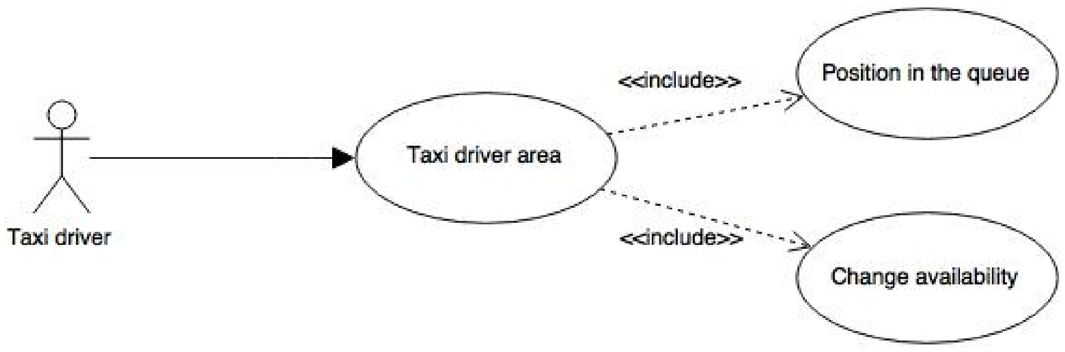
Actor	Taxi driver and passenger.
Goal	Reserve a taxi and answer to the request.
Input Condition	The passenger is already logged in and he has not made any request or reservation recently.
Event Flow	1. The passenger presses the Make a reservation button. 2. The system displays to the passenger the reservation form. 3. The passenger fills in the form and submits it. 4. The system checks the current and the departure time. 5. Ten minutes before the ride, the system looks for an available taxi driver.
Output Condition	When the system finds a taxi driver, it notifies it to the passenger.
Exception	If the departure time is wrong, the system does not allow the passenger to make the reservation.

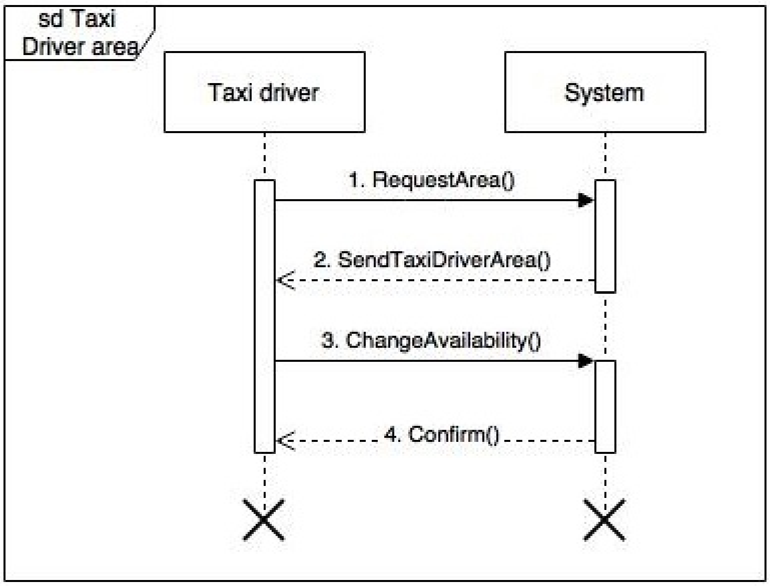



sd Reserve and answer
Passenger
System
Taxi driver
1. ReserveTaxiRequestForm()
2. ReserveTaxiForm()
loop
3. FillInForm()
4. SubmitForm()
5. CheckTime()
alt
6. Error()
6. Confirm()
7. PaymentMethodMessage()
Ten minutes before the ride
8. SubmitChoice()
loop
Start timer (one minute)
9. ForwardRequest()
alt
10. CannotTakeRequest()
10. TakeRequest()
11. SetUnavailable()
Time out
10. CancelRequest()
11. SendAtTheEndOfTheQueue()

- Taxi Area

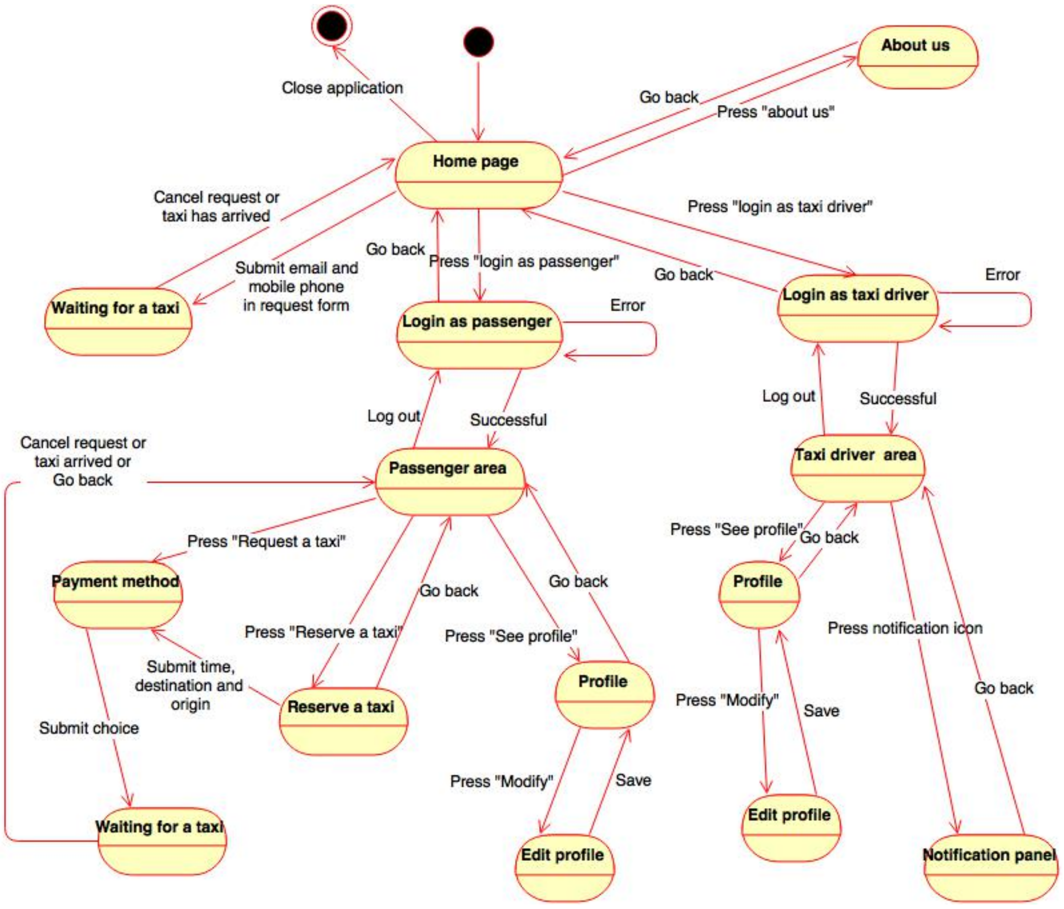
Actor	Taxi driver.
Goal	See the taxi driver area.
Input Condition	The taxi driver is already logged in.
Event Flow	The taxi driver requests his area page.
Output Condition	The system shows the area page to the taxi driver.
Exception	If the information in the area is incorrect, the system refreshes the page.




sd Taxi Driver area
Taxi driver
System
1. RequestArea()
2. SendTaxiDriverArea()
3. ChangeAvailability()
4. Confirm()

5.4 State Chart diagram

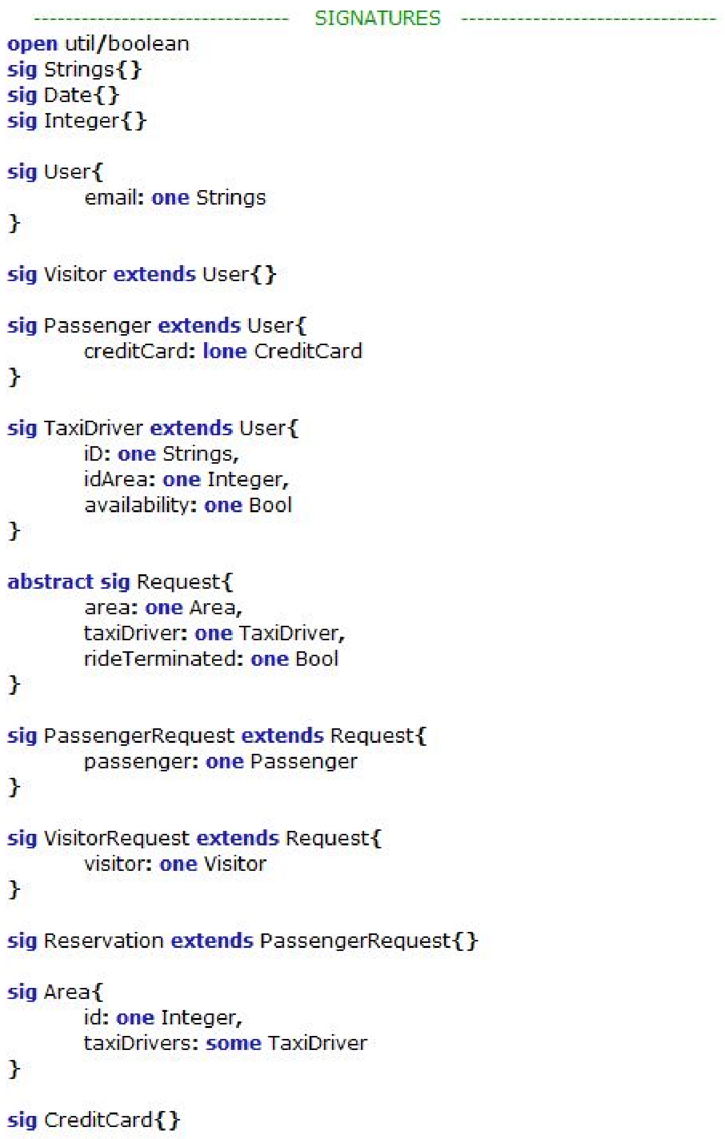
This State Chart diagram describes the behaviour of the system.



6 Alloy Modeling

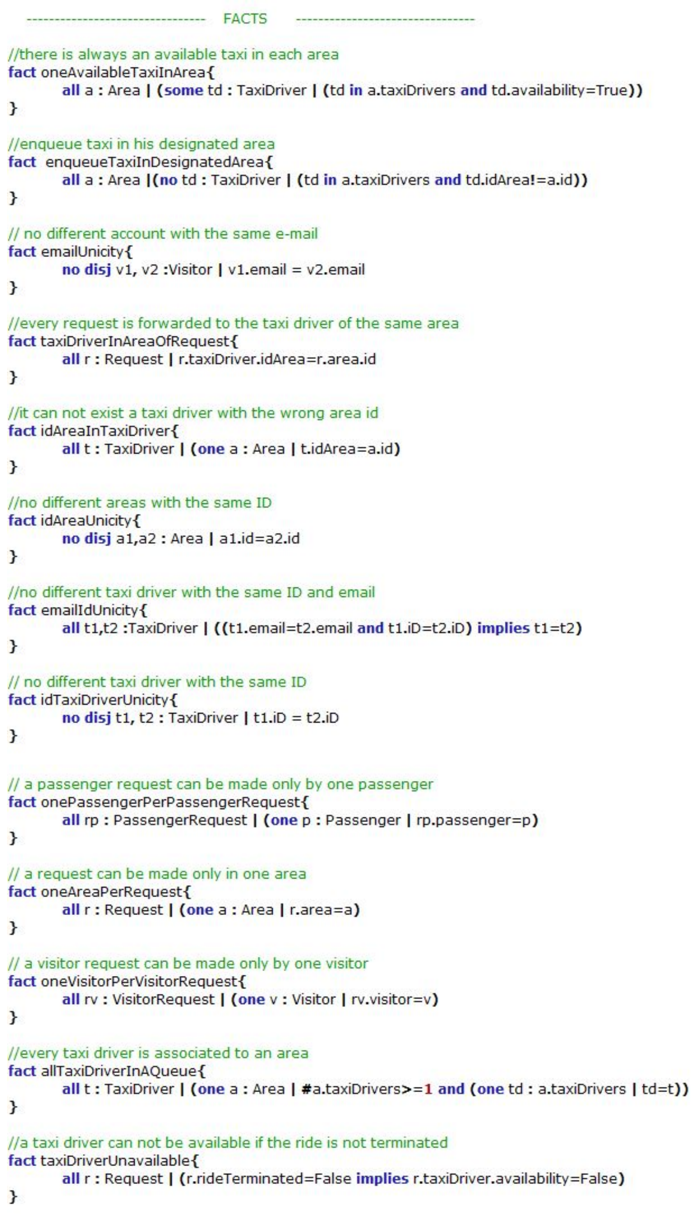
The alloy modeling has been used to test the consistency of the proposed Class diagram with the Alloy Analyzer 4.2. This report consists in the signatures of the classes, facts and assertions on the model and two examples of the world generated by our predicates.

6.1 Signatures



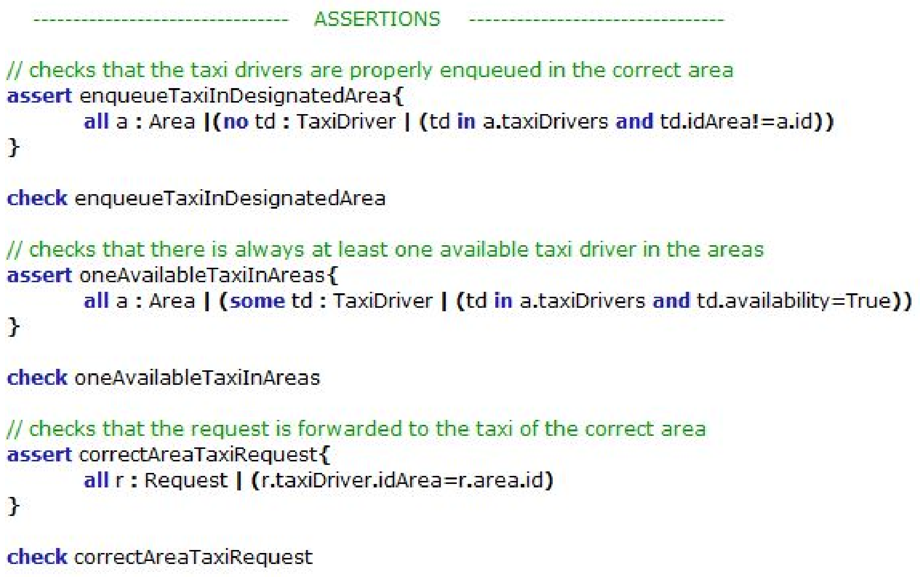
```
    --------------------------------   SIGNATURES   --------------------------------
open util/boolean
sig Strings{}
sig Date{}
sig Integer{}

sig User{
        email: one Strings
}

sig Visitor extends User{}

sig Passenger extends User{
        creditCard: lone CreditCard
}

sig TaxiDriver extends User{
        iD: one Strings,
        idArea: one Integer,
        availability: one Bool
}

abstract sig Request{
        area: one Area,
        taxiDriver: one TaxiDriver,
        rideTerminated: one Bool
}

sig PassengerRequest extends Request{
        passenger: one Passenger
}

sig VisitorRequest extends Request{
        visitor: one Visitor
}

sig Reservation extends PassengerRequest{}

sig Area{
        id: one Integer,
        taxiDrivers: some TaxiDriver
}

sig CreditCard{}
```

6.2 Facts



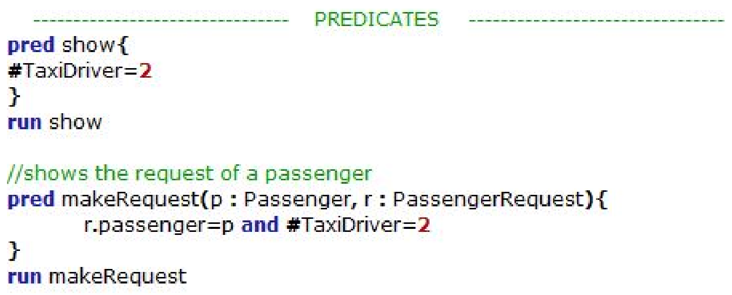
```
   --------------------------------   FACTS      --------------------------------

//there is always an available taxi in each area
fact oneAvailableTaxiInArea{
        all a : Area | (some td : TaxiDriver | (td in a.taxiDrivers and td.availability=True))
}

//enqueue taxi in his designated area
fact  enqueueTaxiInDesignatedArea{
        all a : Area |(no td : TaxiDriver | (td in a.taxiDrivers and td.idArea!=a.id))
}

// no different account with the same e-mail
fact emailUnicity{
        no disj v1, v2 :Visitor | v1.email = v2.email
}

//every request is forwarded to the taxi driver of the same area
fact taxiDriverInAreaOfRequest{
        all r : Request | r.taxiDriver.idArea=r.area.id
}

//it can not exist a taxi driver with the wrong area id
fact idAreaInTaxiDriver{
        all t : TaxiDriver | (one a : Area | t.idArea=a.id)
}

//no different areas with the same ID
fact idAreaUnicity{
        no disj a1,a2 : Area | a1.id=a2.id
}

//no different taxi driver with the same ID and email
fact emailIdUnicity{
        all t1,t2 :TaxiDriver | ((t1.email=t2.email and t1.iD=t2.iD) implies t1=t2)
}

// no different taxi driver with the same ID
fact idTaxiDriverUnicity{
        no disj t1, t2 : TaxiDriver | t1.iD = t2.iD
}


// a passenger request can be made only by one passenger
fact onePassengerPerPassengerRequest{
        all rp : PassengerRequest | (one p : Passenger | rp.passenger=p)
}

// a request can be made only in one area
fact oneAreaPerRequest{
        all r : Request | (one a : Area | r.area=a)
}

// a visitor request can be made only by one visitor
fact oneVisitorPerVisitorRequest{
        all rv : VisitorRequest | (one v : Visitor | rv.visitor=v)
}

//every taxi driver is associated to an area
fact allTaxiDriverInAQueue{
        all t : TaxiDriver | (one a : Area | #a.taxiDrivers>=1 and (one td : a.taxiDrivers | td=t))
}

//a taxi driver can not be available if the ride is not terminated
fact taxiDriverUnavailable{
        all r : Request | (r.rideTerminated=False implies r.taxiDriver.availability=False)
}
```

6.3 Assertions



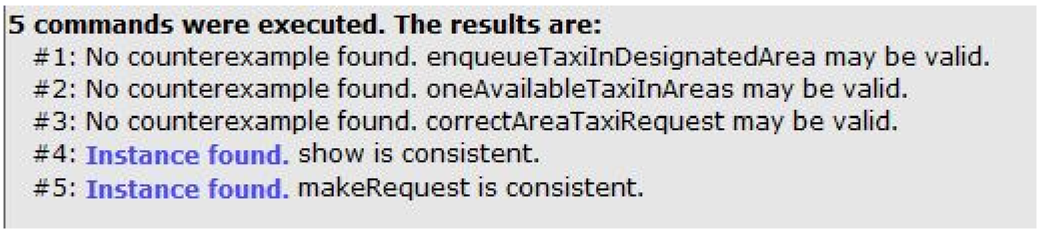
```
    ------------------------------    ASSERTIONS    ------------------------------

// checks that the taxi drivers are properly enqueued in the correct area
assert enqueueTaxiInDesignatedArea{
        all a : Area |(no td : TaxiDriver | (td in a.taxiDrivers and td.idArea!=a.id))
}

check enqueueTaxiInDesignatedArea

// checks that there is always at least one available taxi driver in the areas
assert oneAvailableTaxiInAreas{
        all a : Area | (some td : TaxiDriver | (td in a.taxiDrivers and td.availability=True))
}

check oneAvailableTaxiInAreas

// checks that the request is forwarded to the taxi of the correct area
assert correctAreaTaxiRequest{
        all r : Request | (r.taxiDriver.idArea=r.area.id)
}

check correctAreaTaxiRequest
```

6.4 Predicates



```
    ------------------------------    PREDICATES    ------------------------------
pred show{
#TaxiDriver=2
}
run show

//shows the request of a passenger
pred makeRequest(p : Passenger, r : PassengerRequest){
        r.passenger=p and #TaxiDriver=2
}
run makeRequest
```

6.5 Results

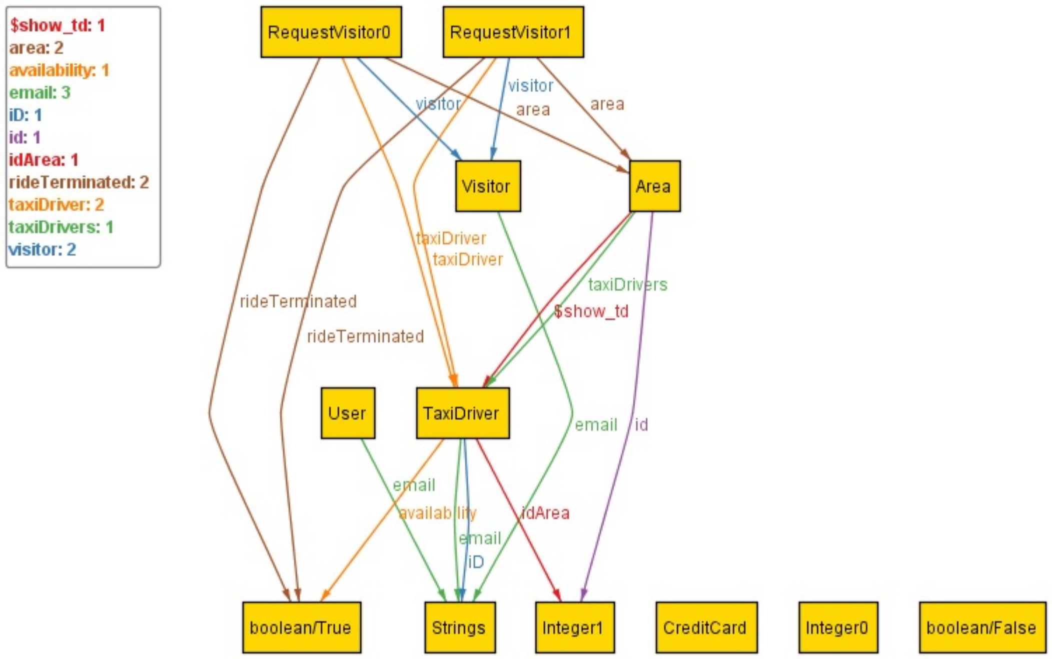


```
5 commands were executed. The results are:
  #1: No counterexample found. enqueueTaxiInDesignatedArea may be valid.
  #2: No counterexample found. oneAvailableTaxiInAreas may be valid.
  #3: No counterexample found. correctAreaTaxiRequest may be valid.
  #4: Instance found. show is consistent.
  #5: Instance found. makeRequest is consistent.
```

6.6 Generated world

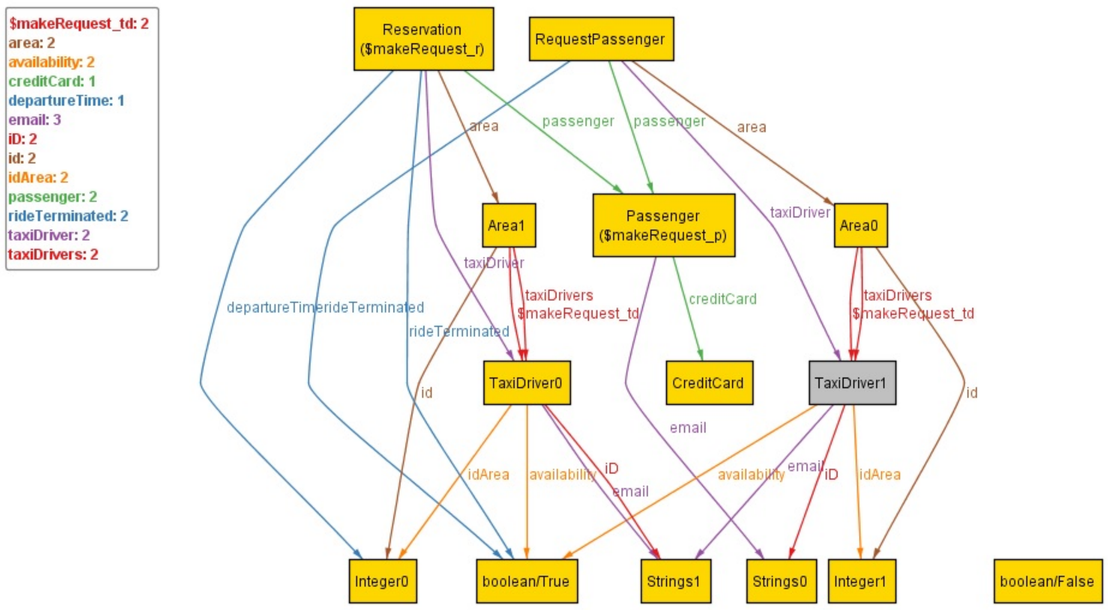
6.6.1 General World

In this image is shown the world generated by Alloy Analyzer via the execution of the predicate "show".



6.6.2 Request By A Passenger

In this image is shown the world generated by Alloy Analyzer via the execution of the predicate ”makeRequest”.



7 Appendix

7.1 Glossary

In order to avoid ambiguity, some words, which are often used in this document, are given a precise definition of what is the meaning in the contest of this project.

- User: a person who requests a service from the system. It can be a visitor or a passenger.
- Visitor: a person who is not registered in the application.
- Passenger: a person who is registered in the application.
- Taxi driver: a taxi driver who access the application with a specific ID.
- Request: the request of a taxi in a certain area and position in the city made by a user.
- Reservation: the reservation of a taxi in a certain area, place and time that can be made only by passengers.

7.2 Software and tool used

- TeXstudio (`http://www.texstudio.org/`): to redact and to format this document.
- draw.io (`https://www.draw.io/`):to create Use Case Diagrams, Sequence Diagrams, Class Diagrams and Statechart Diagrams.
- Balsamiq Mockups (`http://balsamiq.com/products/mockups/`): to create mockups.
- Alloy Analyzer (`http://alloy.mit.edu/alloy/`): to create Alloy models.

7.3 Hours of work

Time spent redacting this document:

- Cattaneo Michela Gaia: ~35 hours of work.
- Barlocco Mattia: ~35 hours of work.