Politecnico di Milano, A.A. 2015/2016

Software Engineering 2: My Taxi Service
Requirements **A**nalysis and **S**pecification
Document

Belotti Nicola 793419
Chioso Emanuele 791621
Colombo Andrea 853381

November 17, 2015

Contents

1 INTRODUCTION

1.1 Description of the given problem

We will project My Taxi Service, an online service that will provide passengers an easy and reliable way to access taxi service, while allowing taxi drivers to organize themselves and make their job simple.

There will be two types of users: passengers and taxi drivers.

Passengers who wants to access the service should register in the system by providing information like name, address, phone number and e-mail. Once registered passengers will be able to:

- Request a taxi by specifying a valid location, the system will confirm the request by sending the user a code;

- Make a reservation for a taxi, in this case the passenger must provide the destination too, the reservation must be done at least 2 hours before the ride. If the reservation is successful the system will start searching for a taxi ten minutes before the meeting time.

- Enable taxi sharing option, meaning that he wants to share the ride with others and thus the cost of the ride. When making a reservation the user can decide to share the ride. Other passengers can view the shared rides and decide to join the ride. the system will create a path for a taxi.

To taxi drivers will be provided a different account with different functionalities. They will be able to set their availability, if they are available the system can call them to go to a specified location to pick up a passenger, a call that drivers can accept or refuse.

The system will optimize taxi queues by dividing the city in different zones of 2 km square size, and will assign to each zone a certain number of taxis organized in a queue. Each request is forwarded to the first taxi in the queue associated to the zone from which the request come.

1.2 Actors

- Visitor: a non registered user can only see the main page, log-in page and registration page. He can register himself by the compilation of the registration form.

- Passenger: a passenger has already an account, he can log-in in the system. After that he can call for a taxi using the apposite form.

- Taxi-Driver: a taxi driver has an account provided by the company. From this account he can manage his profile, set his availability, accept or refuse a call from the system.

1.3 Goals

Here's the list of the goals of our application

- [G1] Allow a visitor to register in the system and adding/managing his information.

- [G2] Allow a user to log in to application, either he is a passenger or a taxi driver.

- [G3] Allow a passenger to make a request for a taxi.

- [G4] Allow a passenger to make a reservation for a taxi and share the ride if he wants.

- [G5] Allow a passenger to join a shared ride.

- [G6] Allow a passenger to cancel a reservation/shared ride.

- [G7] Allow a user to leave a shared ride.

- [G8] Allow a taxi driver to set his availability.

- [G9] Allow a taxi driver to accept or refuse a call for a taxi-request from the system.

- [G10] Provide a fair management of taxi queues.

1.4 Glossary

We will give a specific definition of some crucial terms that we are going to often use in our documentation of the project to prevent some ambiguity of the natural language.

Visitor Every person that visits the website or downloads the application before registration. Registered users are seen as Visitors before the login.

Users Every single person registered in the Database of the service. It includes Passengers and Drivers

Passengers Clients registered in the database, they can only request a taxi and reserve one.

Drivers Taxi Owners registered in the database, they can set their availability depending on their needs and answer or refuse calls from the system.

Request When a Passenger uses the service to find a ride, he makes a Request. He must insert where he needs to go. The Request is sent to the available drivers who can accept or refuse it. It's the interaction that connects passenger and driver.

Reservation When a passenger uses the service to reserve a taxi so he's sure that the taxi will be available when he will need it.

1.5   Identifying Stakeholders

There are two big main Stakeholders for this project:

1. **Public Transport Administrators**
   Every city that hasn't got a good and reliable management of taxi queues is a possible stakeholder. Cities that have got a fair management of taxi queues but without web application or application are also possible stakeholders.

2. **Private Taxy Companies**
   Big taxy companies working on one or more cities may need our system to grant a more powerful service to passengers.

2 Overall Description

2.1 Product Perspective

The application will be released as a web application not integrated with other existing system, with a Client-Server architecture. The server will run the logic and generates web pages, a database system will be used to record information of the users. On the other side there will be several clients connecting using a web browser and a graphical user interface, or using a mobile application. There will be a internal interface for administration for maintenance. The application will also provide API for future development or integration with other project.

2.1.1 Hardware Interfaces

This project does not support any hardware interfaces.

2.1.2 Software Interface

- Database management system

  - Name: MySQL
  - Version: 5.7

- Java Virtual Machine

  - Name: Java enteprise edition
  - Version: 8

- Operating System

  - Mobile: Android (any version), iOS 6 or more recent, Windows 8 for phone.
  - Desktop: Any OS that supports a web browser.

2.1.3 Communication Interfaces

TCP protocol for communicating to DBMS and http/https.

2.2 Product Function

User Registration A normal user will register to the system by inserting username, e-mail and password.

Taxi driver registration There will be a registration functionality only for taxi driver's account, the registration will be done by the driver's employer company, then they will give the credentials to the driver so he can log-in.

Sharing System The shared ride will follow a route composed by the system, the route will contain all the position where the passengers will be picked up and all their destination.

Log-in system The login system will be the same for all the users but there will be different functionalities once logged in based on the account type: passenger or driver

Taxi Queues Management The system will assign each taxi to a queue associated to a zone, when receiving a request the system will always call the first taxi in the queue without skipping any driver that is available. When a driver finishes a ride he will be placed into the queue associated to his current zone.

2.3 User Characteristics

We expect users that need to access taxi service in an easy way. Users will need a web browser or a smartphone to use the application.

2.4 Constraints

2.4.1 Regulator Policies

My taxi service is an application developed after a directed request from the government of X, thus there should be no issues regarding regulator policies.

2.4.2 Hardware Limitation

My taxi service will require a internet connection fast enough to guarantee a fast response from the server.

2.4.3 Interfaces with other application

My taxi service doesn't need to interface to other applications.

2.4.4 Parallel operation

My taxi service must support parallel operations from different users that may require access to the database.

2.5 Domain Properties

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

- Once a request is done, it cannot be deleted.

- If a passenger makes a request and the request is accepted, he will show up at the established location in time.

- If a driver accepts a request, he will show up at the established location in time.

- If a passenger joins a shared ride, he will take part to the ride.

- A passenger will not ask for a taxi in any way at a certain time if he knows there will be a conflict in schedules.

2.6 Assumptions

- Users cannot have more than one request open at the same time.

- Reservations must be done at least two hours before the ride.

- Users cannot make requests if a reserved ride is taking place within 30 minutes.

- Reservations can be cancelled at most 30 minutes before the scheduled time.

- There will be a notification by e-mail and through the application 10 minutes before a reserved ride, when the server makes the request for him/them.

- There will be a notification sent to the current members of a shared ride when a new passenger joins the ride.

- Shared rides are reservations with the sharing option active.

- Every taxi has an attribute that shows the maximum capacity of the vehicle.

- A taxi will be placed in queue associated to a zone only if his driver notified the system he is available.

- A driver will be associated to only one taxi, this means that a taxi can't be driven by two different drivers.

- A driver that picks up a passenger in the normal way will set off his availability.

- There will always be at least a taxi driver available in a queue.
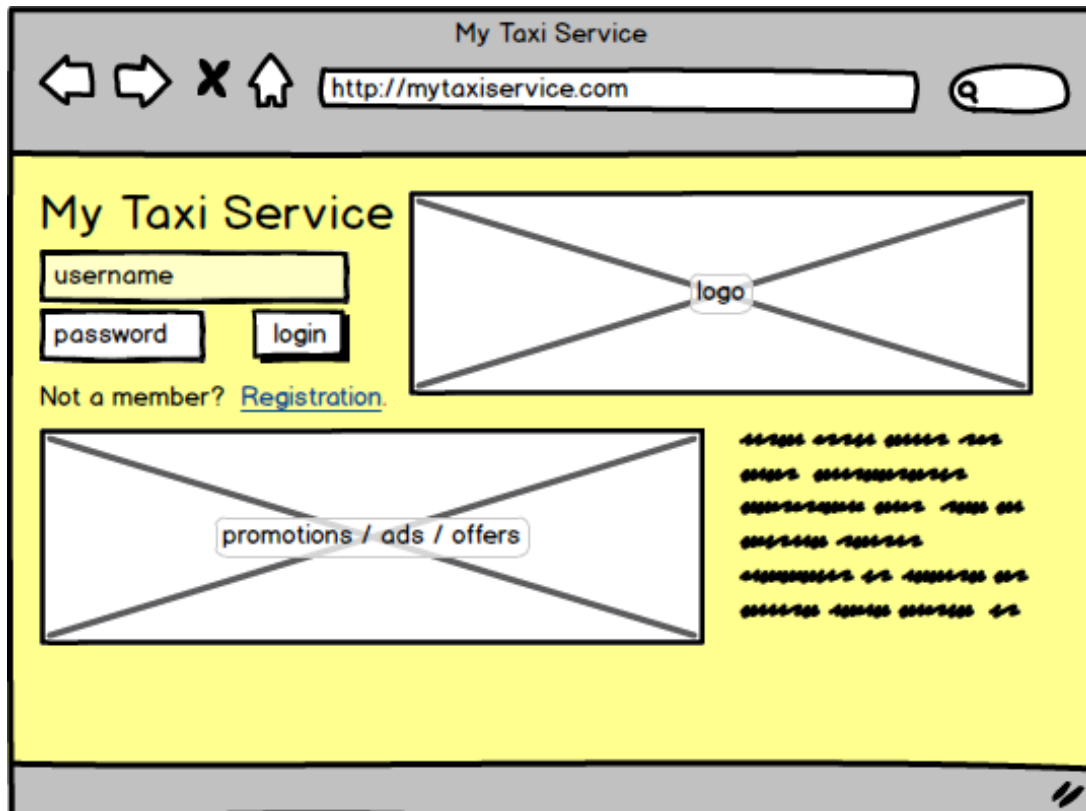
3 REQUIREMENTS

3.1 External Interfaces

Here are shown some mockup that should represent an idea of the structure of the web and mobile interfaces
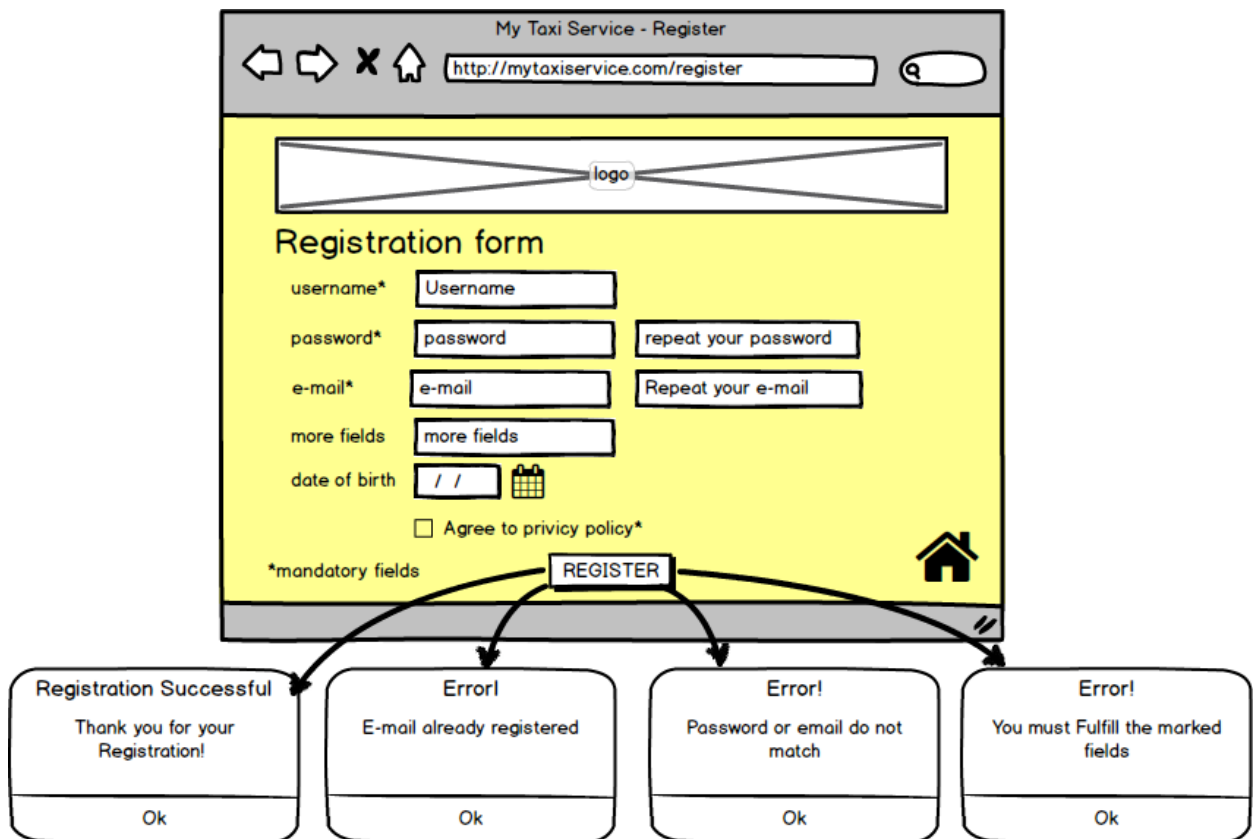
3.1.1 Web User interfaces

- **Visitor home page**

  This is the first page that every visitor will get. It's the homepage of myTaxiService and it allows visitors to login or register.

- **Registration Form**
  This mock shows the basic fields that the server needs to register a new user, some of them are mandatory and there could be more in future implementations.

- **Passengers home page**
  Registered users that have a passenger account will be lead here from the visitor home page. This mock shows the actions that they will be able to do: requesting a ride, reserving a taxi, finding a shared ride, going in the reservations page, logging out.
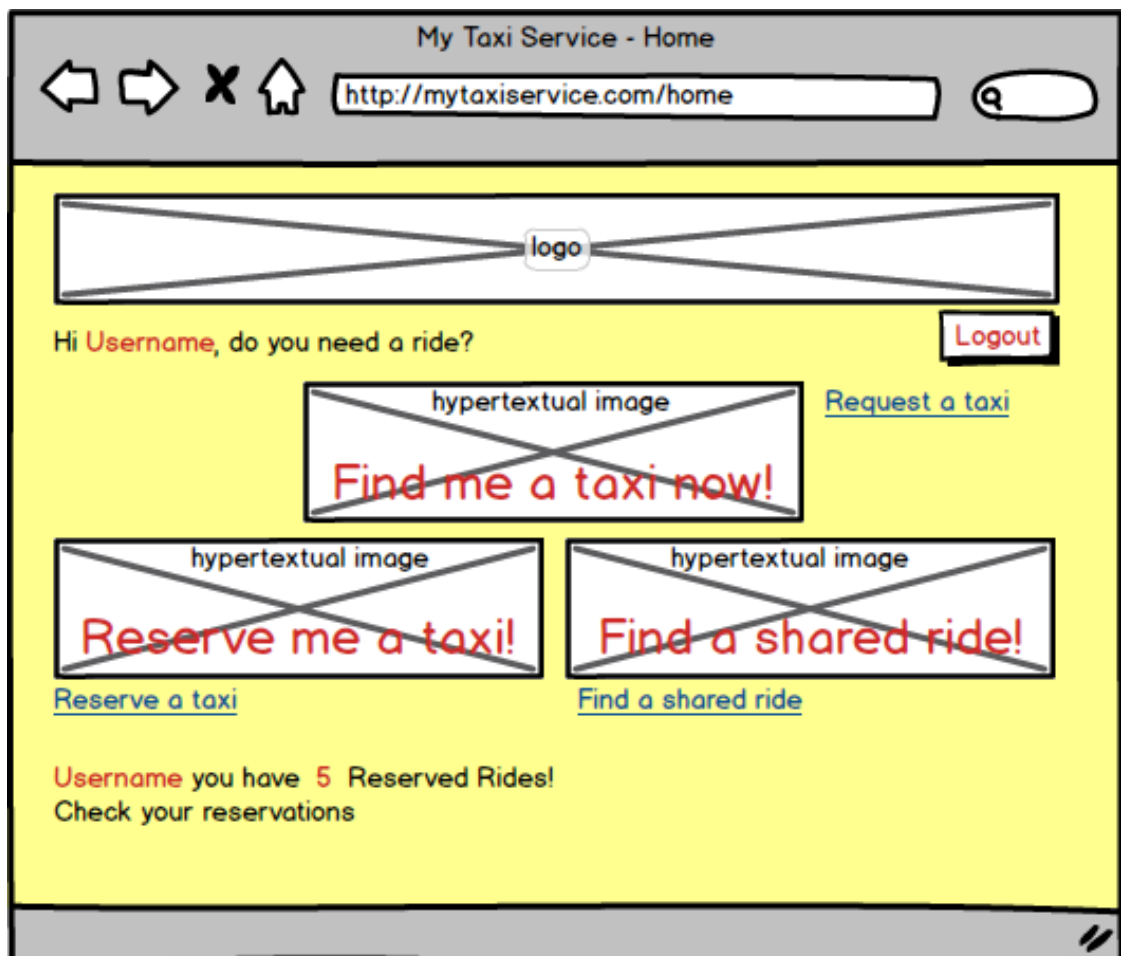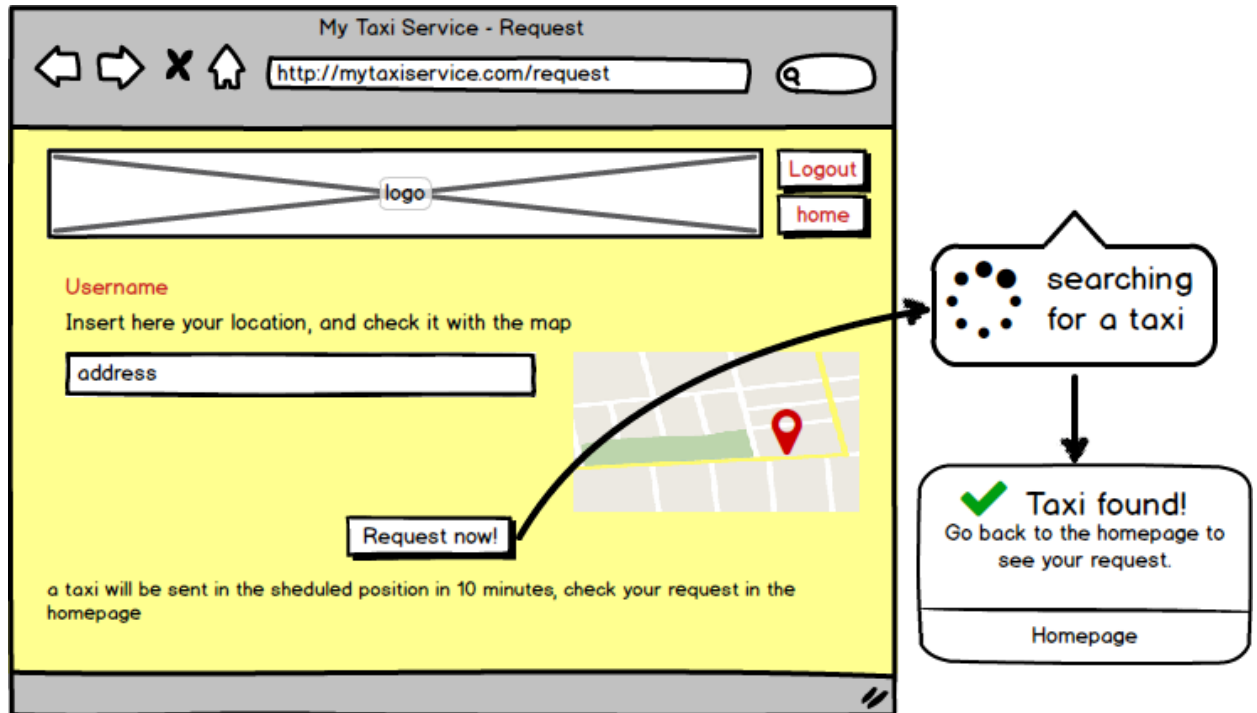
- **Passengers with a request home page**
  Passengers that already have a request active and are waiting for a taxi will have a different home page, that will not allow them to do another request.
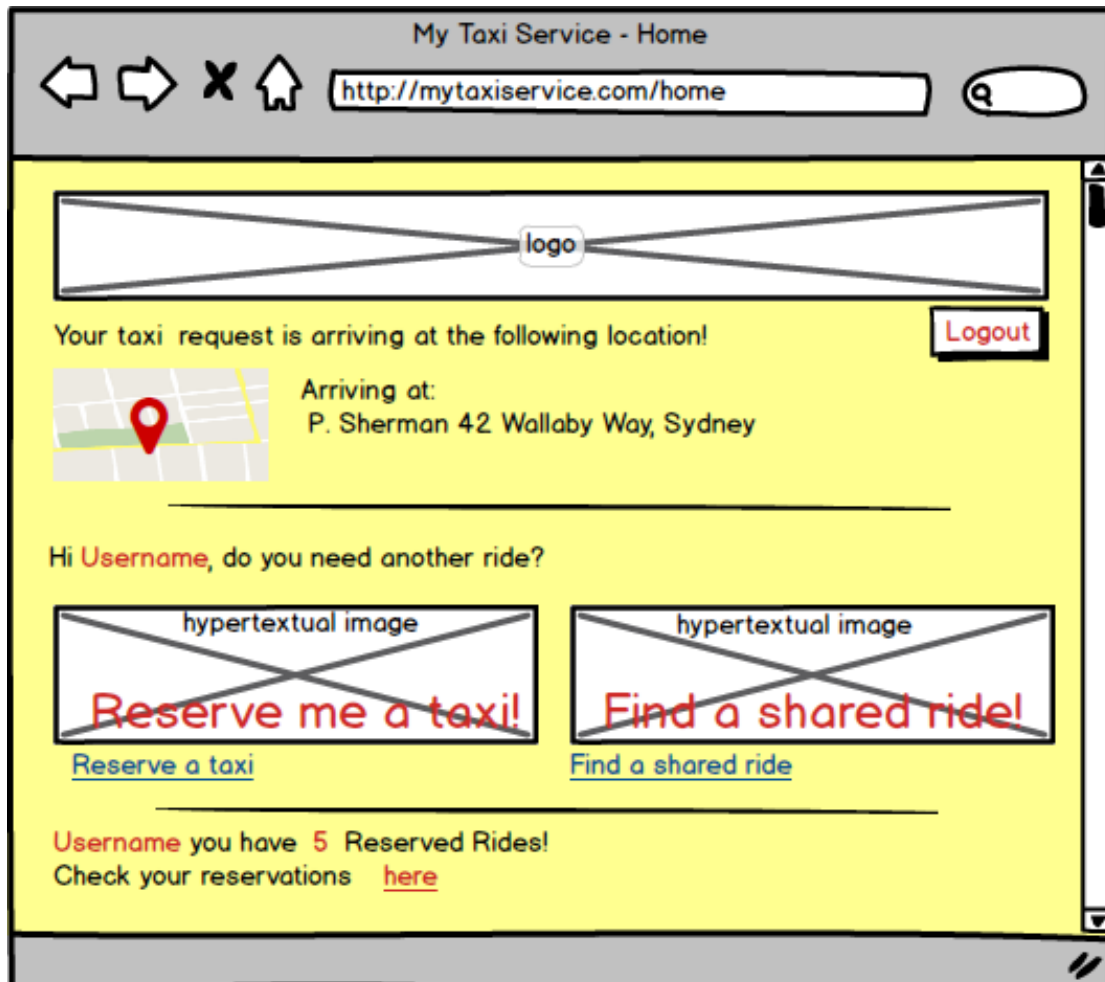
- **Request form**

  This is a simply and basic mock that allows the passenger to instantly call a taxi. the system will wait for the answer of a driver

- **Reservation Form**
  Here is an example of how reservations should be done. They all are mandatory except for the shared ride option.

- **Reservations**

  Once a reservation is done, the user is able to see it in the reservation page. they are ordered by date and hour and all the properties are listed. The example shows all the typologies of reservations, including the deletable and the not deletable ones.
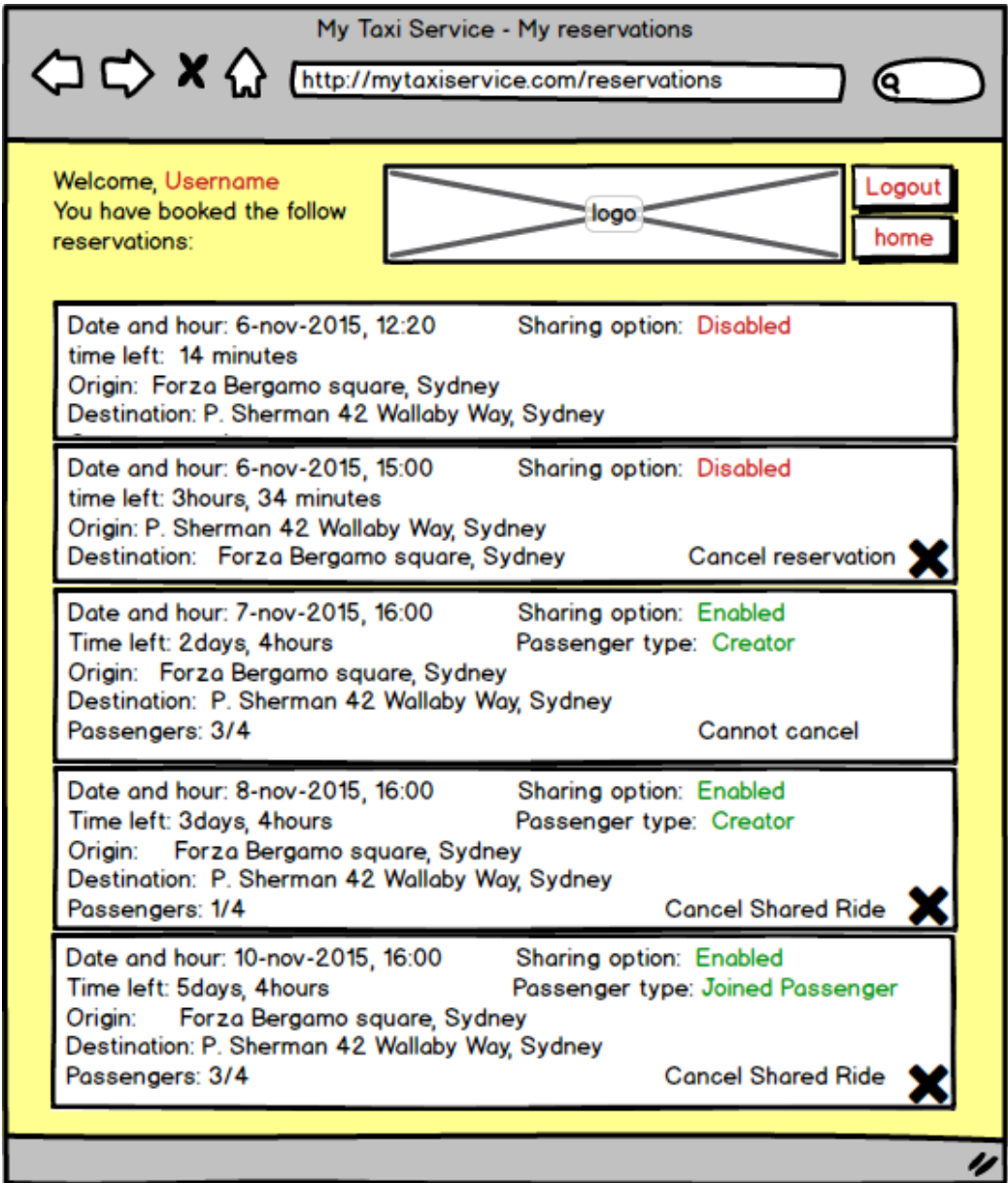
- **Shared rides**

  This mock is the template of the shared ride page. All the fields are mandatory, after filling them, there will be shown the result of the search. The user then can join his preferred ride.

3.1.2 Mobile Passenger interfaces

- **Visitor home pages**
  These mockups are the homepages for visitors with mobile devices, they will be lead to download the application to register and login with their smartphone.

- **Passenger home pages**

  These two homepages show what passenger will see in their application when they open it if they have logged in at least once. If they have a request active they will have a different homepage that will show them all the information they need and it will not let them do another request.
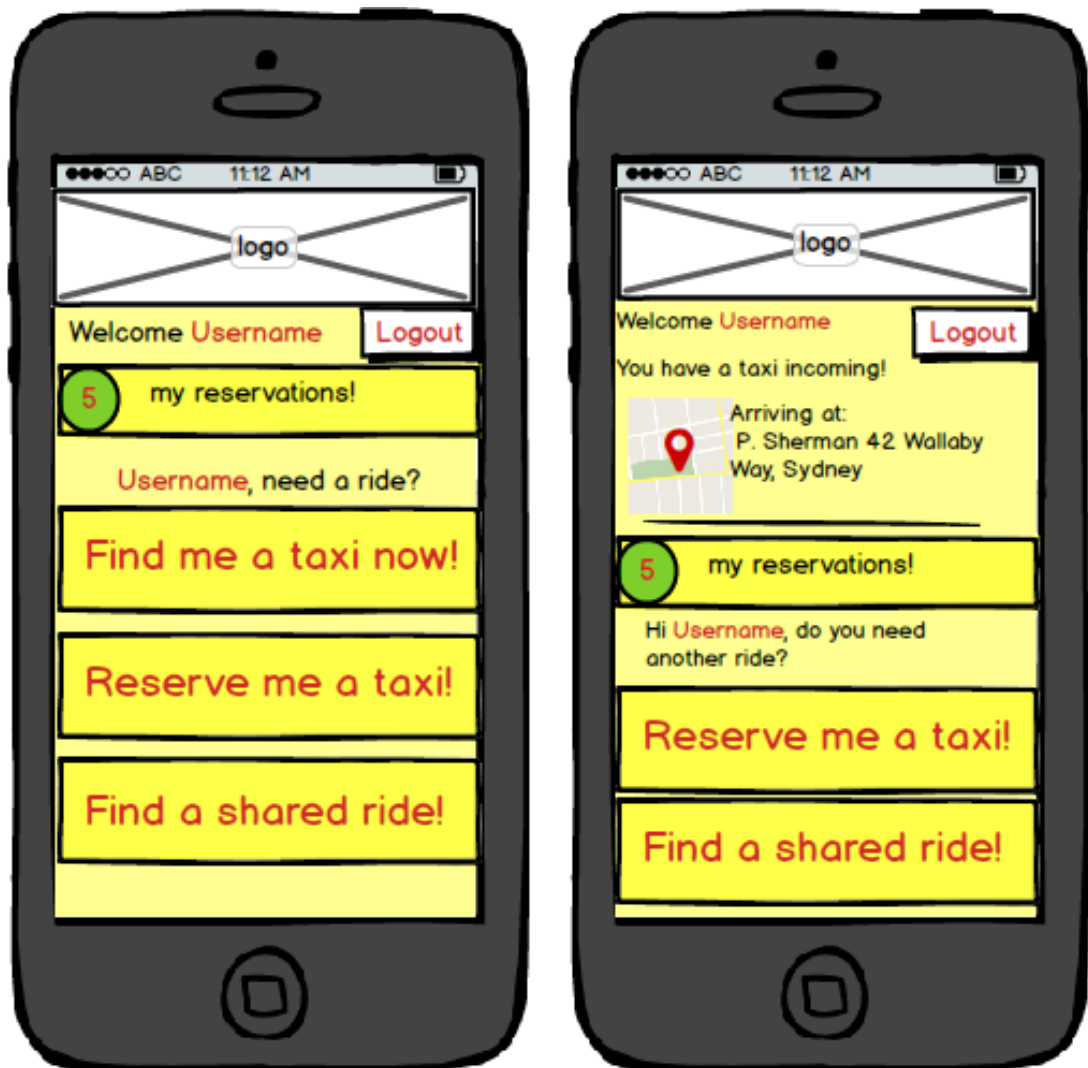
- **Request form**
  Like in the website, the request will be simply and fast, passengers will only need to write their location or use the map to start a request. Once the driver is found, a popup will notify it and they will be lead to the homepage again.
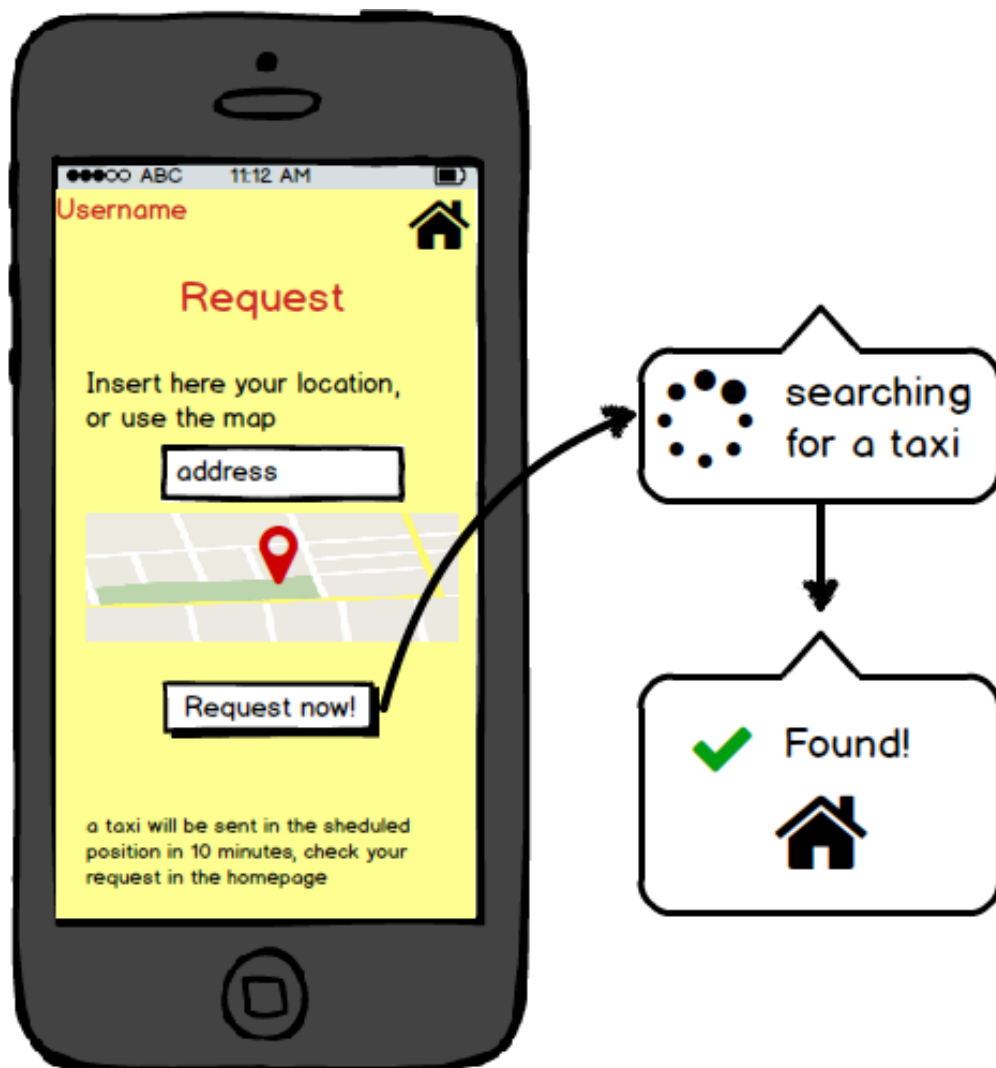
- **Registration form**
  These are the fields that the user must fill to register, like in the website.

- **Reservation form**

  Passengers that want to reserve a taxi must provide the mandatory informations and fill the fields that the app shows. They can also choice to share their reservation and make it a shared ride.

- **Reservations**
  Reservations booked by the passenger are shown in this reserved page, it will allow him to see them all and scroll down to find the one that to need to read. To see all the information he will need to expand the box by touching it. Touching again will make it small again.
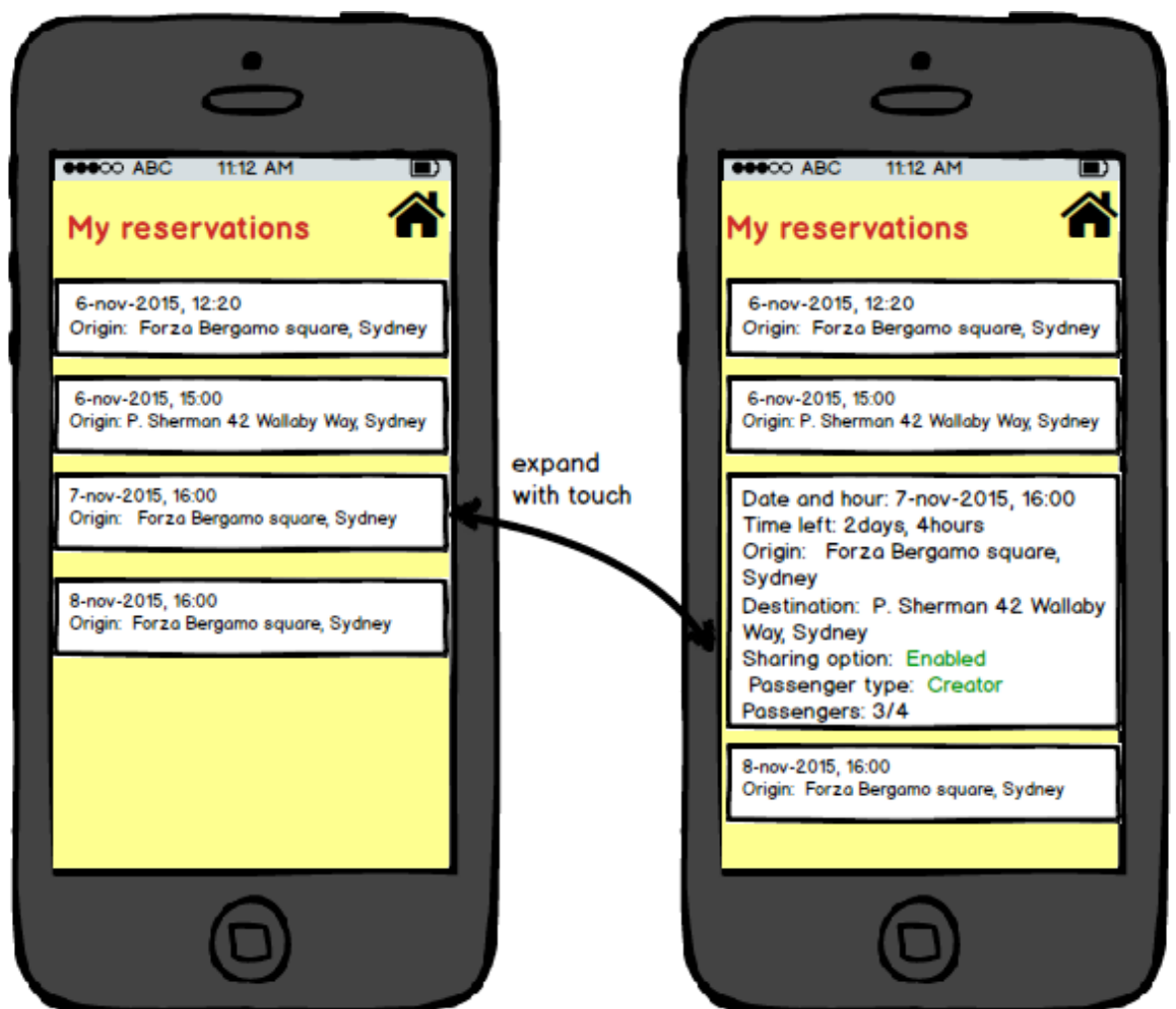
- **Shared ride pages**
  This is a mock of the page that allows the passengers to search for a shared ride and the page that will be displayed them after they hit the search button.
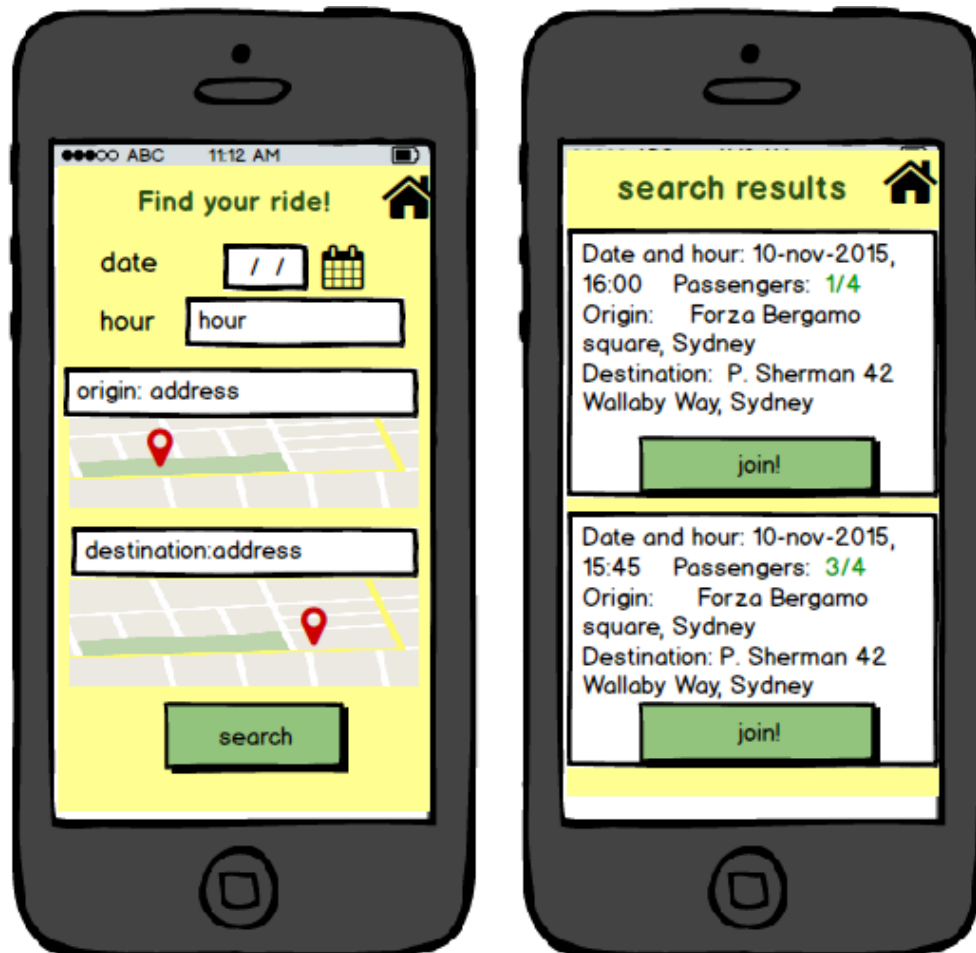
3.1.3 Driver mobile interface

- **Driver homepage**
  This is the home page where drivers are lead after logging in. they can change their availability and see their assigned zone and position in that queue.
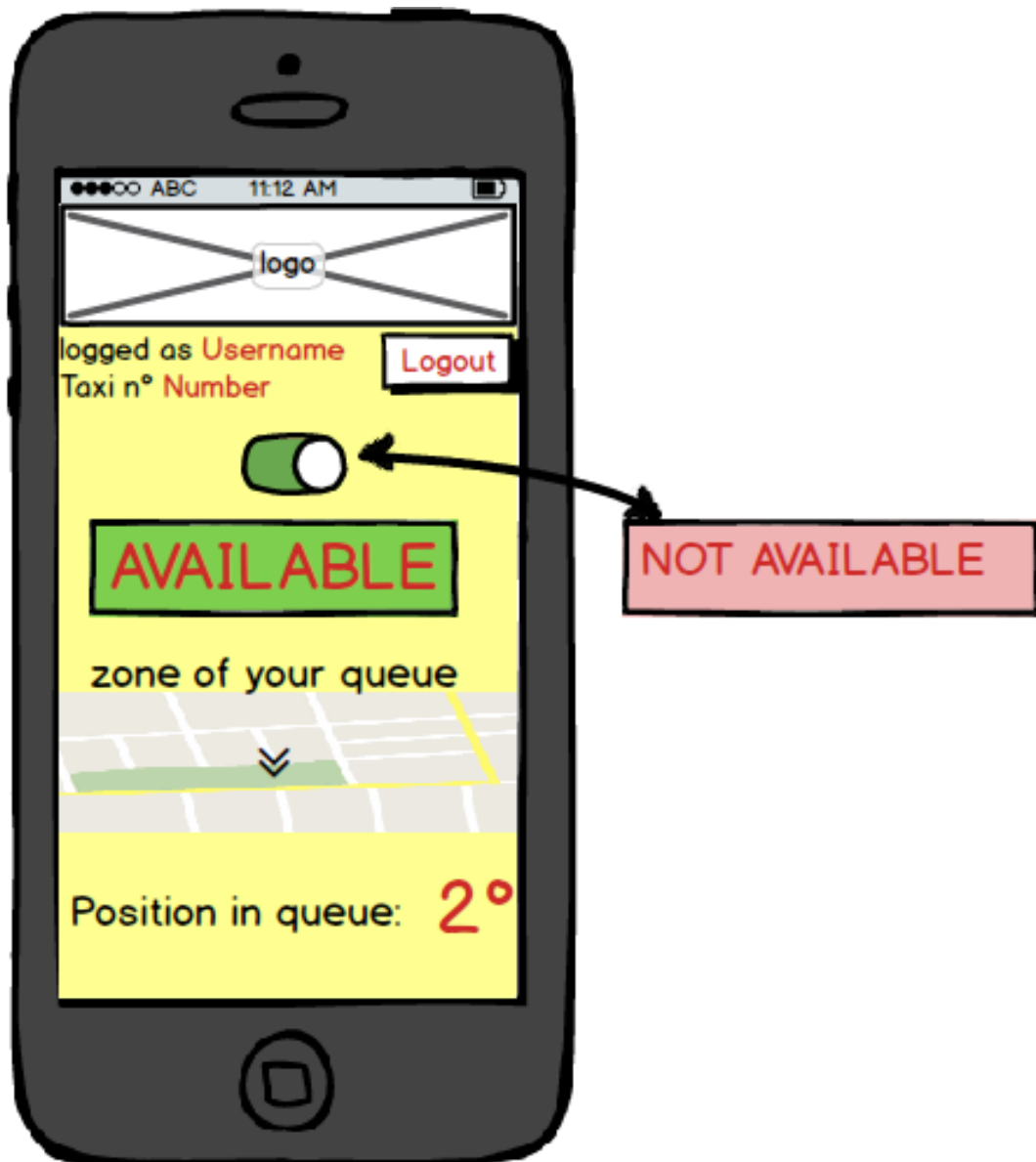
- **Request incoming page**

  When a passenger does a request or the system sends one for a reservation, these are the information that the driver should see in his mobile device. he can accept or refuse, if he refuses he will come back in the homepage, if he accepts he will follow his gps to the target location to start the ride.
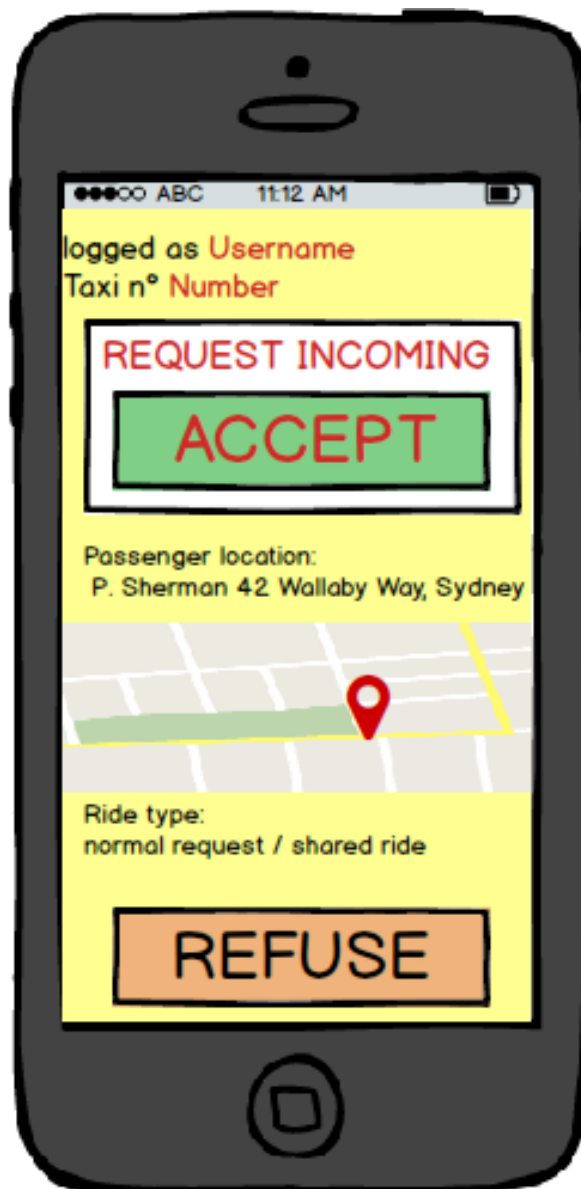
3.2 Functional Requirements

By analyzing the goals we came up with a list of requirements in order to achieve them:

- [G1] Allow a visitor to register in the system and adding/managing his information.

  - [R1] The system will provide a registration functionality.
  - [R2] System should check that user name must be unique, there cannot be two users with the same user name in the system.
  - [R3] System will not allow visitors to see other pages than the login page.
  - [R4] System will grant visitors access only to registration functionality.

- [G2] Allow a user to log in to application, either he is a passenger or a taxi driver.

  - [R1] The system will provide a log-in functionality.
  - [R2] System will check that the tuple username-password inserted by the user exists in the database.

- [G3] Allow a passenger to make a request for a taxi.

  - [R1] The system will not grant access to this functionality if the user is not logged in.
  - [R2] The system will forward a taxi request to a driver only if:
    * The passenger provides a valid location for a taxi.
    * Passenger is not waiting for another taxi called by a previous request.
    * Passenger does not have a reserved ride occurring within thirty minutes.

- [G4] Allow a passenger to make a reservation for a taxi and share the ride if he wants.

  - [R1] The system won't grant access to this functionality if the user is not logged in.
  - [R2] The system will accept the reservation if the passenger:

- ∗ Specifies starting position, destination and leaving time of the ride
- ∗ Completes the reservation two hours before the ride occurs.
- ∗ Did not make a reservation for a ride that occurs thirty minutes before or after the requested time.
  - [R3] If a user wants to share a ride the system will permit him to enable sharing option at the moment of the reservation, then wait until the taxi is full or until 10 minutes before the scheduled time for other users to join the ride and finally compute a path for the ride.
  - [R4] If the reservation is successful the system will call for a taxi via a normal request 10 minutes before the scheduled time, send a notification to the passenger, calculate the length and the cost of the ride and communicate it to all participants and to the taxi driver as well.

- [G5] Allow a passenger to join a shared ride.
  - [R1] The system won't grant access to this functionality if the user is not logged in.
  - [R2] If a passenger wants to join a shared ride the system will ask him the starting position and destination he's headed, show all the possible non full rides heading in the same direction, wait for user decision and then add the user to the shared ride.

- [G6] Allow a passenger to cancel a reservation/shared ride.
  - [R1] System will not allow a passenger to cancel a ride if it will occur in less than thirty minutes.
  - [R2] The system will reject a cancel request of a shared ride if someone has already joined it.

- [G7] Allow a user to leave a shared ride.
  - [R1] The system will allow a passenger to leave a shared ride if it won't occur within fifteen minutes.

- [G8] Allow a taxi driver to set his availability.
  - [R1] System will provide an interface to taxi drivers where they can notify the system that they are able to take care of a request.

- [G9] Allow a taxi driver to accept or refuse a call for a taxi-request from the system.

  - [R1] When a request occur the system will show a screen to the driver in which he can accept or refuse the call.

- [G10] Provide a fair management of taxi queues.

  - [R1] The system will divide the city in zones of two squared kilometers and will assign to each of them a certain number of taxis organized in a queue.

  - [R2] The system knows at every time which taxis are assigned to a zone by using a gps system installed on the vehicles.

  - [R3] When a request arrive from a zone the system will call the first taxi in the queue associated to that zone, the system will never skip to the second position in the queue.

  - [R4] If a taxi refuse a request the system will place it at the and of the queue and forward the request to the second tax in the queue.S

  - [R3] When a request arrive from a zone the system will call the first taxi in the queue associated to that zone.

  - [R4] If a taxi refuse a request the system will place it at the end of the queue and forward the request to the second tax in the queue.

3.3 Non Functional Requirements

3.3.1 Performance Requirements

Performance should be high to guarantee usability knowing that there will be a lot of real time request and computation. We assume that the response time of application is close to zero so the speed of the whole system depends only on user's internet connection.

3.3.2 Software System Characteristics

3.3.2.1 Availability
Everyone should be able to access the application online everytime, this means that a dedicated server could be necessary.

3.3.2.2 Maintainability
The application will be accurately documented to help future developers in maintain and apply changes to the code.

3.3.2.3 Portability
The application should be compatible to all major hardware and software platform present on the market.

3.3.2.4 User friendliness
The application does not expect an expert user so the interface will be simple and intuitive.

3.3.3 Data integrity, consistency and availability

The data should be always accessible. They should also be duplicated in case of a system fault to prevent data losses.

4 SCENARIO IDENTIFYING

In this section we will provide the description of some scenarios that may occur using my taxi service:
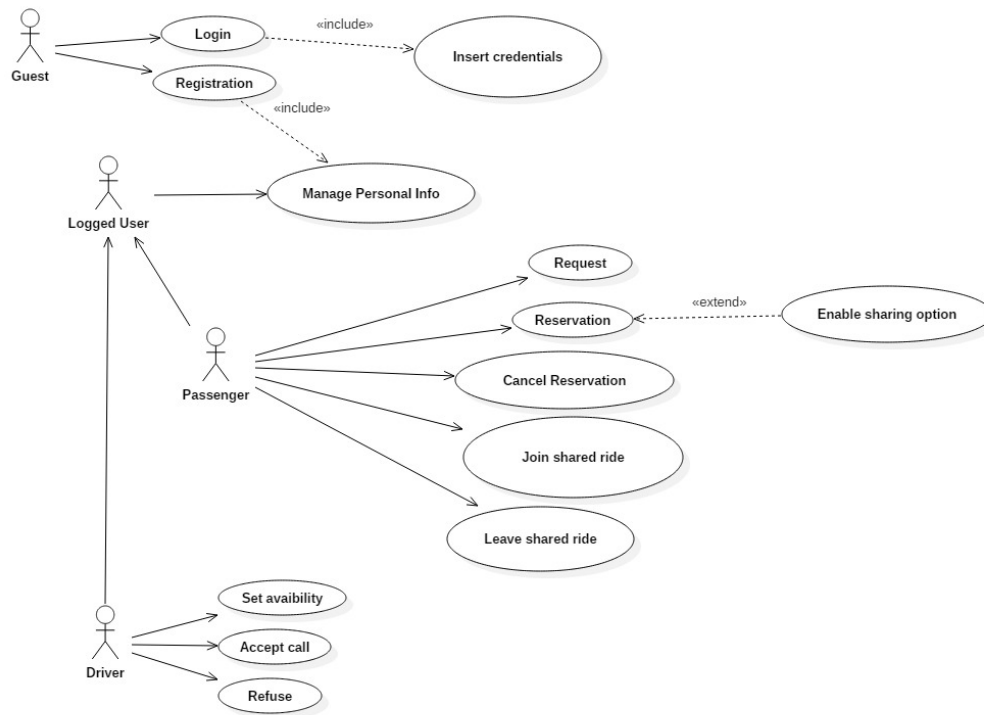
1. Pippo has just arrived to Naples as a tourist, he decides that he wants to go to the cinema, but the cinema is in a place not reachable using public transports, and the movie starts in ten minutes so he can't just walk there. Pippo has heard some rumors of a revolutionary application for managing taxi rides called my taxi service, he googles it, download the application on his phone, register to the system and then log in, then he requests a taxi inserting the address of his current location. The system forwards it to the first taxi available, the driver accept the call and arrives in less than two minutes to pick up Pippo and bring him to the cinema.

2. Nicholas is coming home to Milan tomorrow, he will arrive at Linate airport and there's no one to pick him up. So he logs in to my taxi service and makes a reservation for a taxi inserting Linate as starting position, his home's address as destination and the time he wants to take the ride, two o'clock pm. He then decides to make another reservation from his home to his office, inserting as time twenty minutes past two pm, the system denies the second reservation because he already have a ride occurring less than 30 minutes before and he probably won't be able to take the second ride. He creates a new reservation to get to work at half past two pm and the system accepts the reservation. When Nicholas arrives at the airport the taxi is there to pick him up and brings Nicholas home, where he will take the next taxi to get to work, after leaving his bags at home.

3. Mario wants to take a taxi to go from his home to the theatre, but he doesn't have enough money for a ride, so when making a reservation he enables the taxi sharing option, hoping someone is going in the same direction so they can split the ride's cost. Luigi is willing to go to the same theatre, and he lives not far from Mario, he sees the shared ride created by Mario and joins it, a notification is sent to Mario informing that Luigi joined the ride. Matthew needs to go to his office placed one kilometer north of the theatre, and he joins the ride too. Peach and Yoshi need to go in the same direction and they try to join the ride too, but the taxi is already full so the system denies the join request. Mario tries to cancel the shared ride but he can't cause other people has already joined it. Luigi decides he doesn't want to go to the theatre

and leave the shared ride, Matthew tries to leave the shared ride too but it's too late so he have to join it. The system now computes a route for the taxi based on all the location provided by the users and send it to the driver as request, also send to both driver and passenger the cost of the ride.

4. Oliver has an appointment in 30 minutes, so he decides to make a reservation for a taxi. After completing the form the system denies the reservation because the requested time is not at least two hours later the time Oliver completed the registration form.

5. Pippo requests a taxi inserting as starting location "Wallaby Way 42 - Sidney", the system does not find this location anywhere on the map and denies the request, asking Pippo a valid location.

6. Barry requests a taxi inserting his home's address, the system denies it because Barry has a reserved ride occurring in fifteen minutes.

7. Hooch is a taxi driver, he's waiting in his taxi at a station, he set his availability on and he didn't accept a call or something, suddenly a passenger shows up, Hooch is free so he picks him up, log in the application and set his availability off because he already has a passenger to take care of.
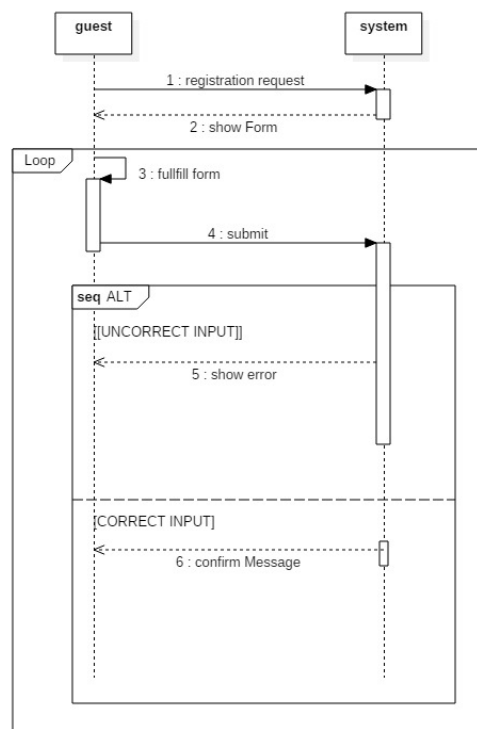
5 UML MODELS

5.1 Use case diagram
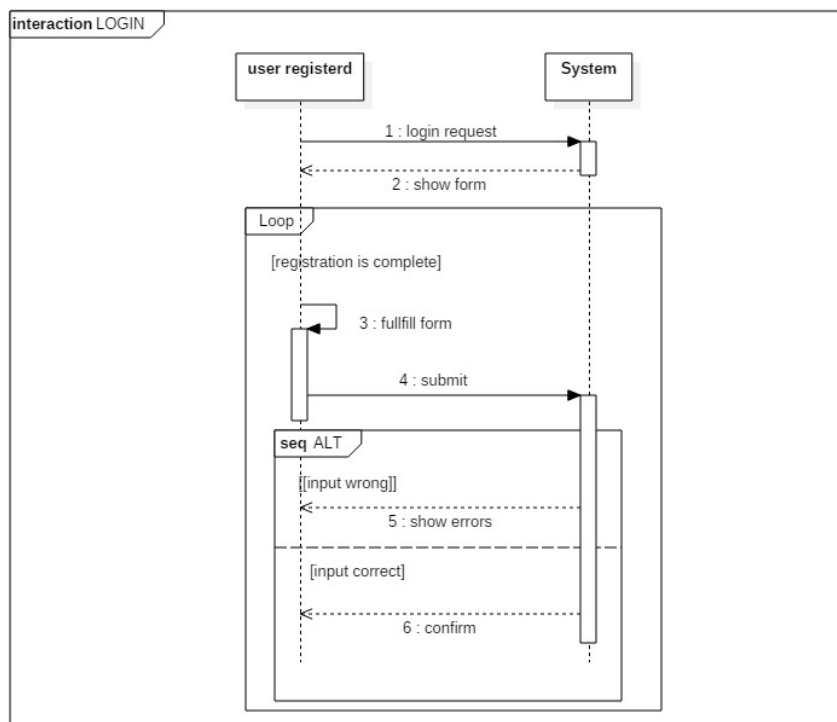
5.2 Use case Description and sequence diagram

5.2.1 Registation

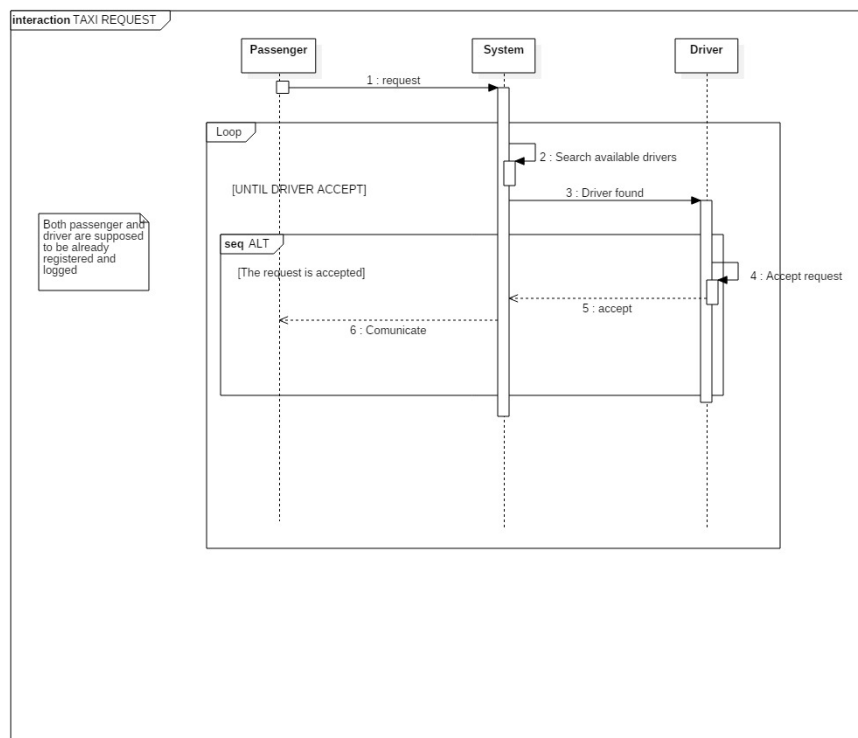| Goal	G1
Actor	Guest
Entry conditions	None
Flow of events	- The guest enters the website - The guest clicks the registration button - The guest fills the registrationo form - The Guest clicks on CONFIRM button
Exit conditions	The system shows the home page to the user and updates its database.
Exceptions	The email is already registered in the system or some fields are empty.

5.2.2 Login

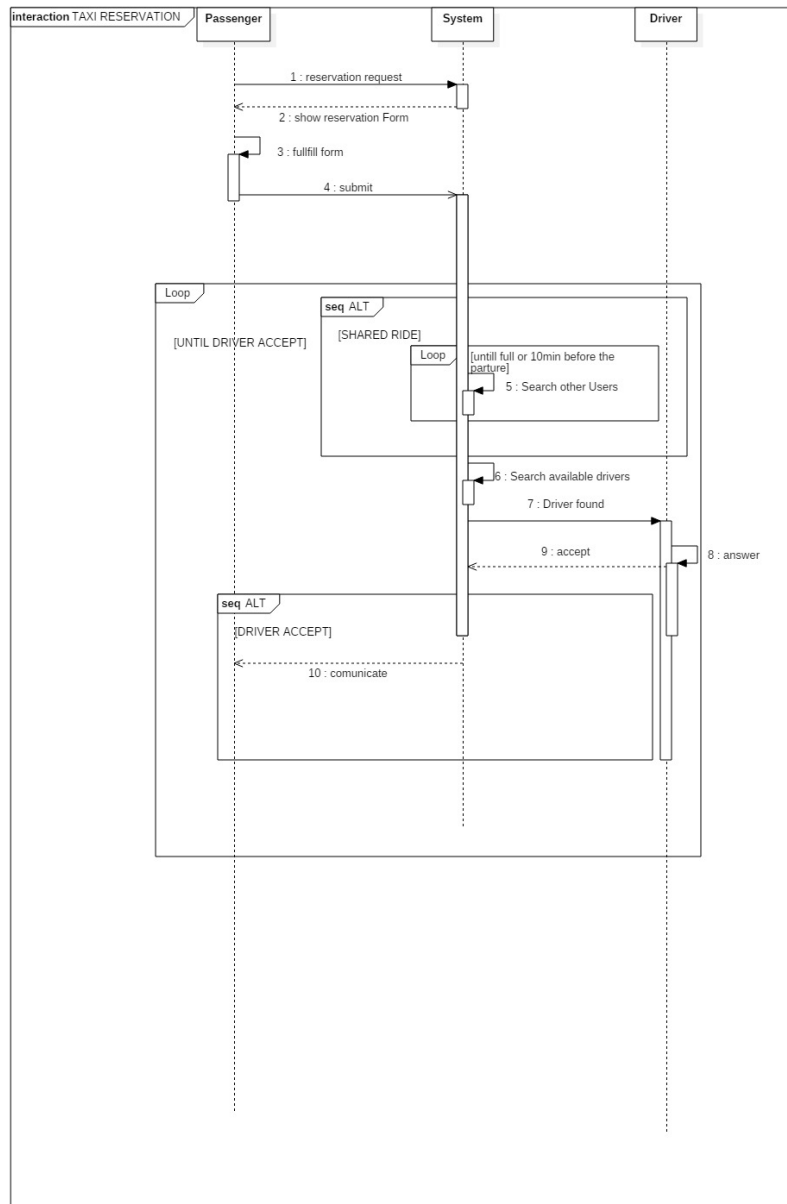| Goal	G2
Actor	Guest
Entry conditions	none
Flow of events	- The guest enters the website - The guest fills the fields Username/Password - The Guest clicks on login button
Exit conditions	The system shows the personal page to the user.
Exceptions	Username and password combination is invalid.

5.2.3 Request ride

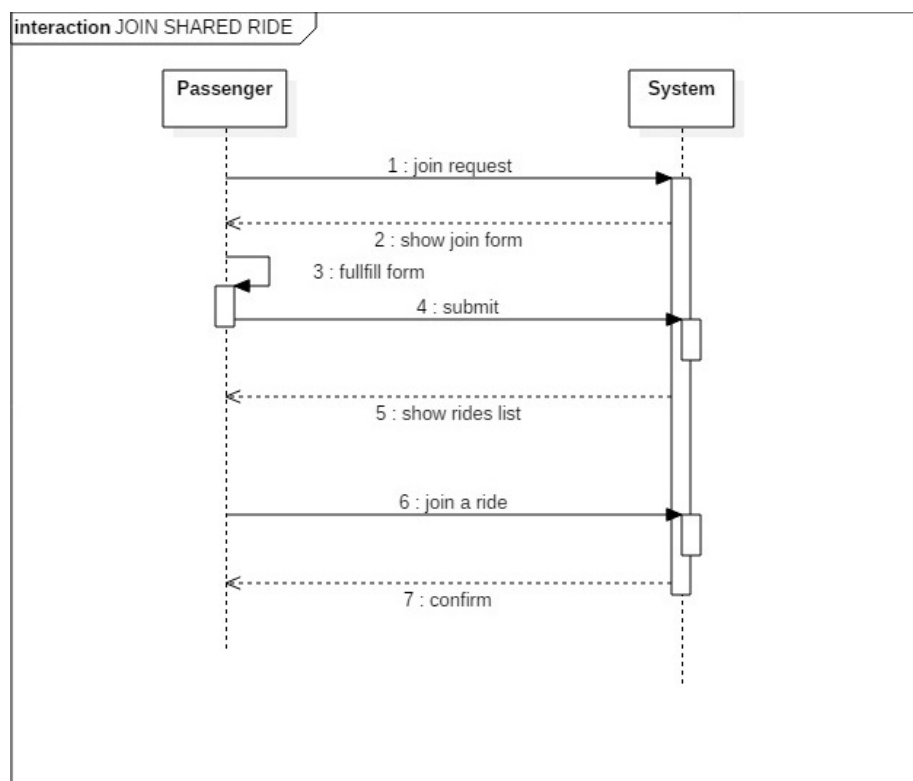| Goal	G3
Actor	Passenger
Entry conditions	User registered and logged in
Flow of events	- The passenger clicks on request button - The passenger insert his current location in the apposite field - The passenger clicks on confirm button
Exit conditions	The system find a taxi and send a confirmation to the user.
Exceptions	Location provided by the user is invalid

5.2.4 Reserve a ride

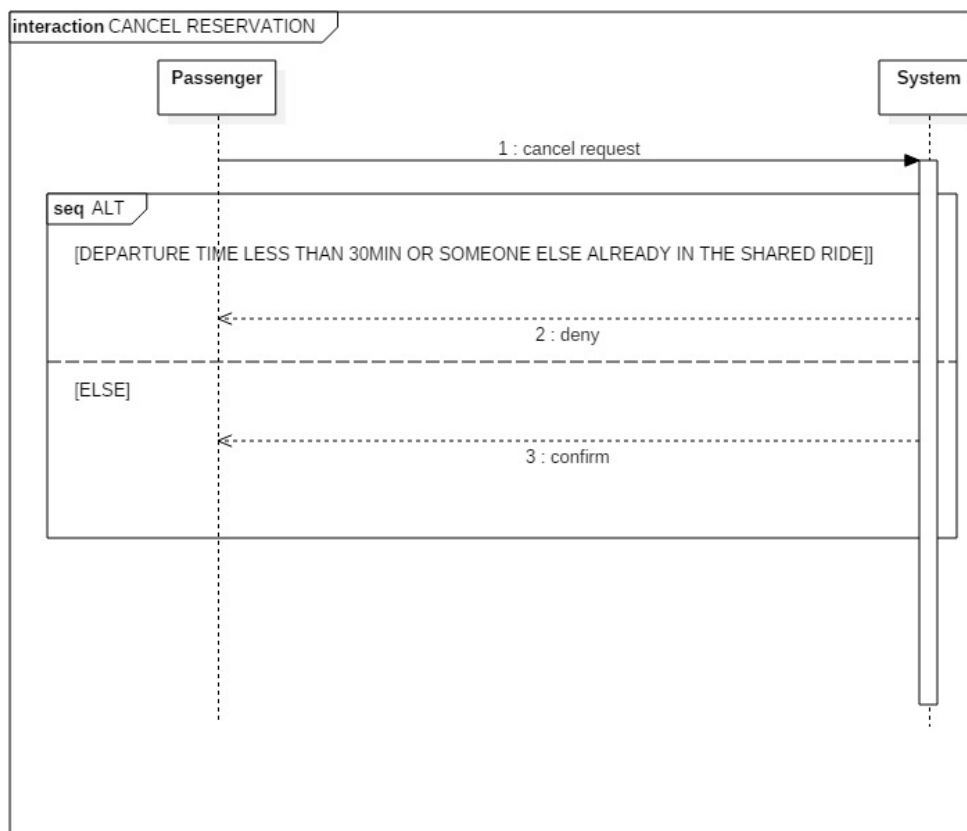| Goal	G4
Actor	Passenger
Entry conditions	User registered and logged in
Flow of events	- The passenger clicks on reservation button - The passenger insert his current location, destination and time of the ride in the apposite field - The passenger enable the taxi sharing option if he wants - The passenger clicks on confirm button
Exit conditions	If the sharing option is active the system wait for other passenger to join. Either it's shared or normal reservation, ten minutes before the meeting the system will find a taxi and send a confirmation to the user.
Exceptions	Location provided by the user is invalid, requested time is in less than two hours from the time the reservation is made, there are other rides occurring in 30 minutes from the requested time

interaction TAXI RESERVATION

Passenger | System | Driver

1 : reservation request

2 : show reservation Form

3 : fullfill form

4 : submit

Loop [UNTIL DRIVER ACCEPT]

seq ALT [SHARED RIDE]

Loop [untill full or 10min before the parture]

5 : Search other Users

6 : Search available drivers

7 : Driver found

8 : answer

9 : accept

seq ALT [DRIVER ACCEPT]

10 : comunicate

5.2.5 Join a ride

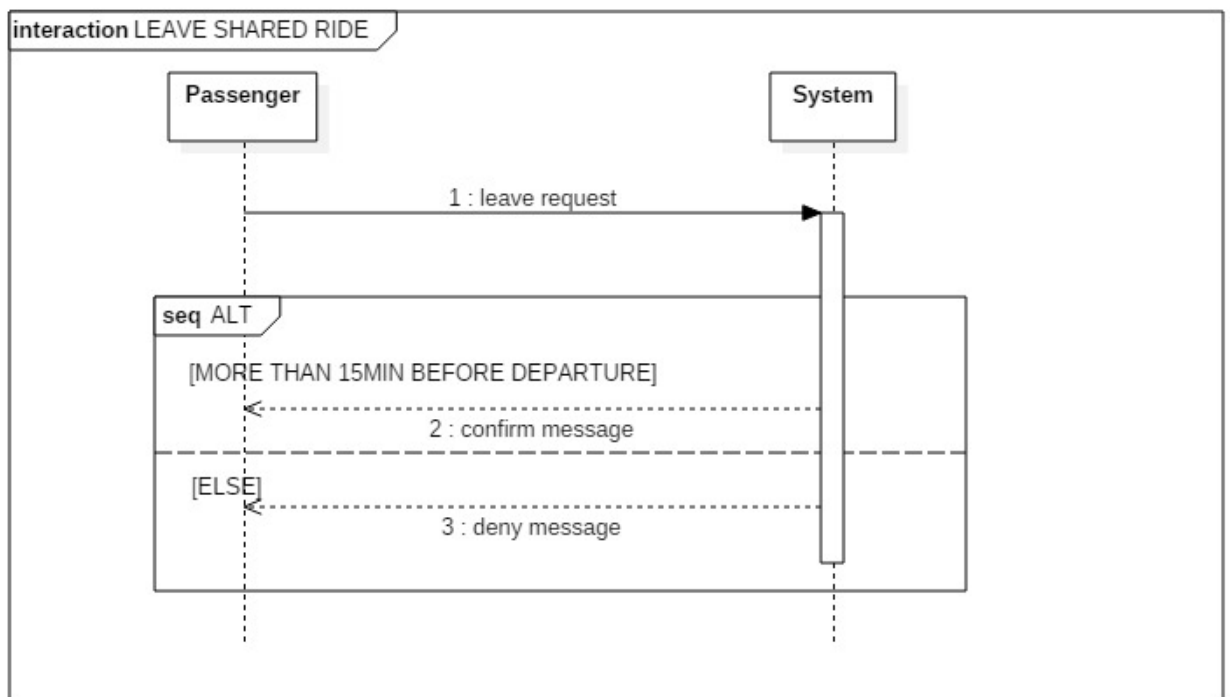| Goal	G5
Actor	Passenger
Entry conditions	User registered and logged in
Flow of events	- The passenger clicks on join button - The passenger insert starting position and destination - The passenger choose the ride to join - The passenger clicks on confirm button
Exit conditions	The system add the passenger to the ride and send a notification to all passengers that joined the ride
Exceptions	Ride is already full, passenger has a ride that is occurring 30 minutes from the shared ride time

5.2.6 Cancel a reservation/shared ride

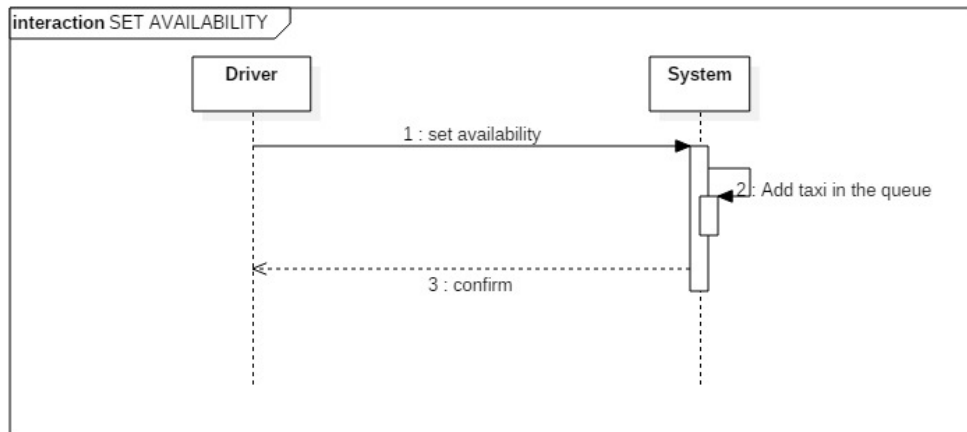| Goal	G6
Actor	Passenger
Entry conditions	User registered and logged in
Flow of events	- The passenger clicks on cancel reservation button - The passenger clicks on confirm button
Exit conditions	The system cancels the reserved ride
Exceptions	Ride is occurring in 30 minutes, other users have already joined the ride

5.2.7 Leave a shared ride

| Goal	G7
Actor	Passenger
Entry conditions	User registered and logged in
Flow of events	- The passenger clicks on leave ride button - The passenger clicks on confirm button
Exit conditions	The system remove the passenger from the ride and send a notification to all passengers that joined the ride
Exceptions	Departure time is in fifteen minutes.

5.2.8 Set availability

| Goal	G8
Actor	Driver
Entry conditions	User registered and logged in
Flow of events	- The driver set his availability - The driver clicks on confirm button
Exit conditions	The system add the driver to the taxi's queue.
Exceptions	none

5.2.9 Accept or refuse request

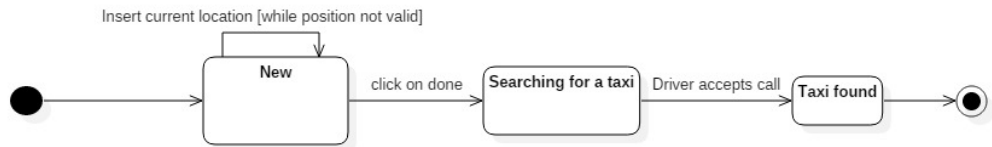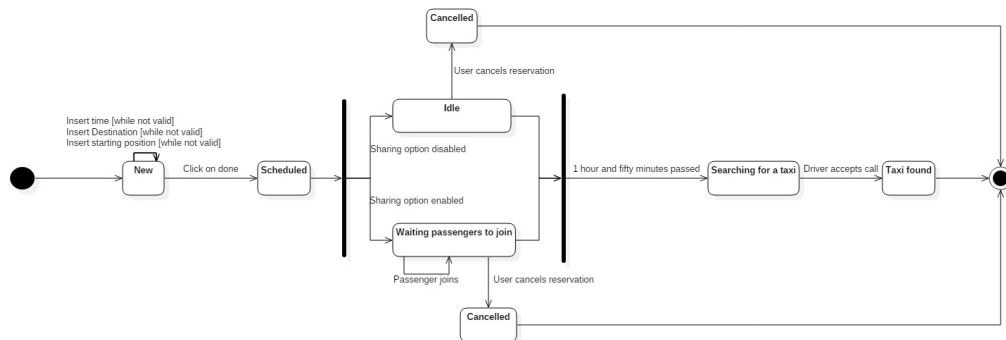| Goal	G9
Actor	Driver
Entry conditions	User registered and logged in
Flow of events	
	• The driver receive a call from the system • The driver accept or refuse it
Exit conditions	The system assign the driver to a ride
Exceptions	Driver doesn't answer in time
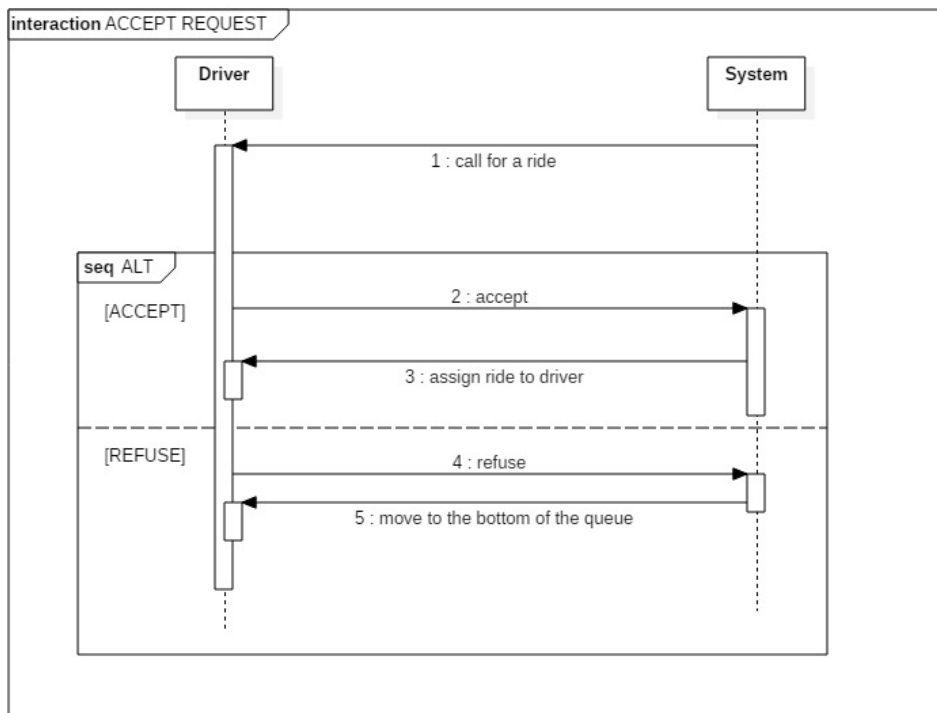
5.3   Class Diagram

++
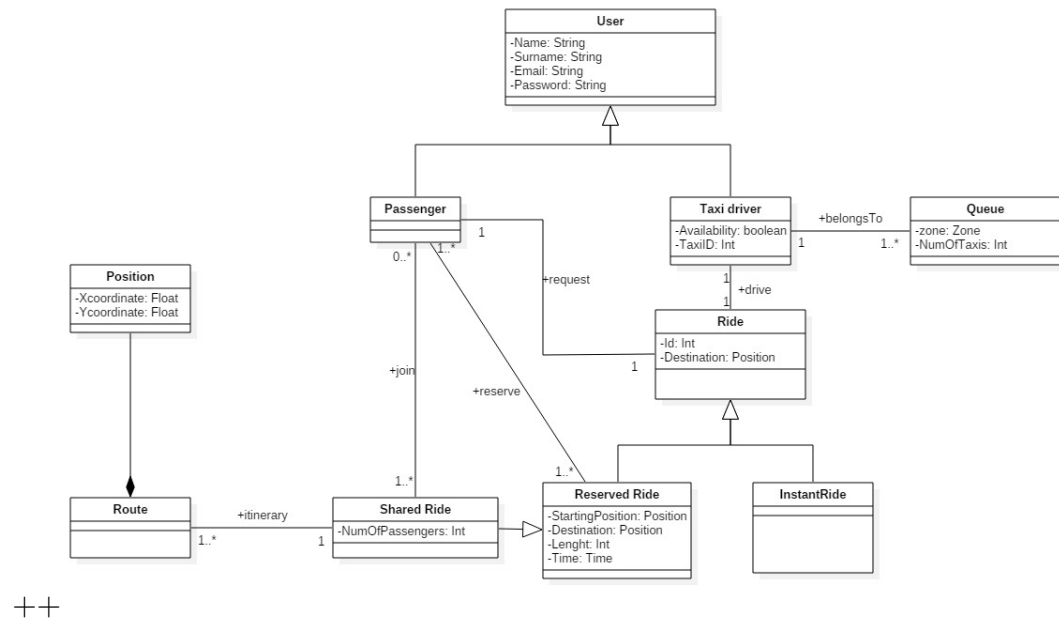
5.4 State chart diagram
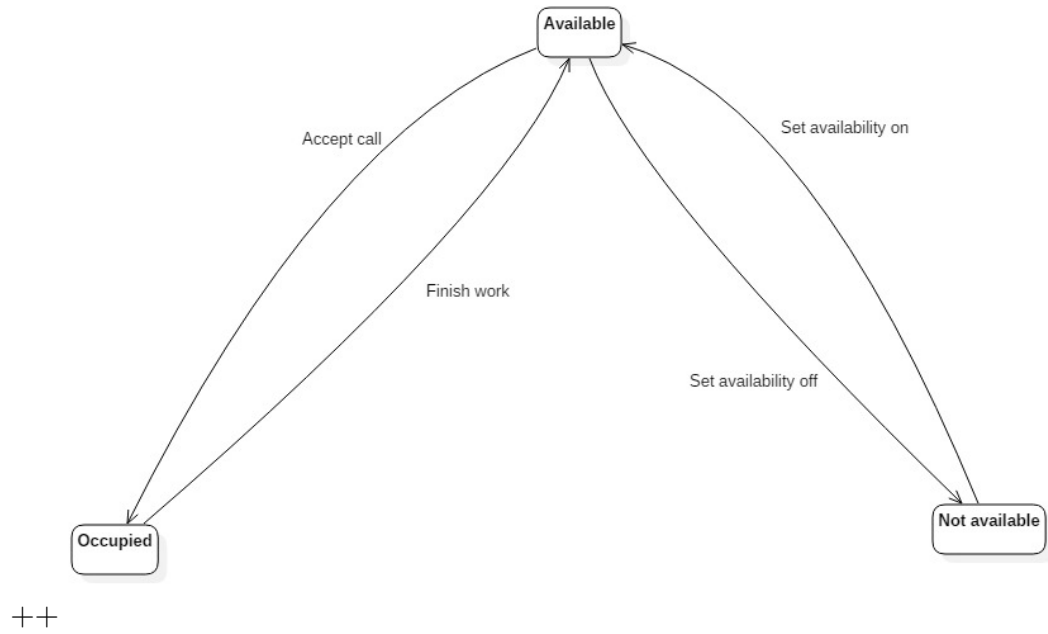
1. User makes a request

++

2. User makes a reservation

++

3. Taxi driver states

++

6 ALLOY MODELLING

The following alloy model is obtained by referring to our class diagram. Our code file has been structured by first inserting signatures, followed by facts, assertions and predicates. The complete code file is available on github. The results of our analysis, namely the generated worlds, are available at the end of this section.

6.1 Signatures

```
//SIGNATURES

abstract sig Ride{
start: one Position,
driver: one TaxiDriver
}

sig InstantRide extends Ride{
requestedBy: one Passenger,
}

sig ReservedRide extends Ride{
reservedBy: one Passenger,
destination: one Position,
Time: one Int
}

sig SharedRide extends ReservedRide  {
joinedBy: set Passenger,
itinerary: set Position
}

sig Position{
}

sig Passenger{
}

sig TaxiDriver{
}
```

6.2 Facts

```
//FACTS

fact NoNegativeTime{
all rr: ReservedRide | rr.Time >=0
}

fact NoNullPassenger{
all r: InstantRide | #r.requestedBy > 0
all rr: ReservedRide | #rr.reservedBy > 0
all sr: SharedRide | #sr.joinedBy>=0
}

fact NoReservationInLessThan30Minutes{

no disj rr1, rr2: ReservedRide | (rr1.reservedBy = rr2.reservedBy and !(rr1.Time-rr2.Time < 30 and rr2.Time-rr1.Time < 30))
}

fact OnlyOneRidePerTaxi {
all td: TaxiDriver | all disj r1, r2: Ride | assignRide [r1, td] implies (td in r1.driver implies td not in r2.driver)
}

fact StartingPositionAndDestinationNotEquals{
//Starting position and destination not the same
all rr: ReservedRide | !(rr.start=rr.destination)
}

fact JoinAndReservation{
//If a passenger reserves a ride he can't join it
all p: Passenger | all disj sr1,sr2:SharedRide | joinRide[p, sr1,sr2] implies (p in sr1.joinedBy <=> p not in sr1.reservedBy)
}
```

6.3 Assertions

```
//ASSERTIONS

assert NoMoreThanOneDriver {
all disj td1, td2: TaxiDriver | all r: Ride | (r.driver = td1 implies r.driver != td2) and (r.driver = td2 implies r.driver != td1)
}
check NoMoreThanOneDriver for 5

//Check that there can't be two rides created by the same user that have departure time closer that 30 minutes
assert NoTooCloseRide{
all disj rr1, rr2: ReservedRide | rr1.reservedBy = rr2.reservedBy implies (rr1.Time-rr2.Time > 30 or rr2.Time-rr1.Time > 30)
}
check NoTooCloseRide for 5
```

6.4 Predicates

```
//PREDICATES

//show all
pred show(){
#InstantRide>1
#ReservedRide>1
#SharedRide>1
#Passenger>1
#TaxiDriver>1
#Position>1 }
run show for 5

//show share rides
pred showShared(){
#ReservedRide > 1
#SharedRide = 2
#Passenger = 3
#TaxiDriver = 2
#InstantRide = 0
#reservedBy > 1
#Position = 2
}
run showShared for 5

//show rides
pred showRides(){
#ReservedRide >1
#InstantRide >1
#SharedRide > 1}
run showRides for 5

//Pred to assing a ride to a taxi driver
pred assignRide (r: Ride, td: TaxiDriver){r.driver = td}
run assignRide for 5

pred joinRide(p:Passenger, sr1, sr2: SharedRide){sr1.joinedBy = sr1.joinedBy+p}
run joinRide for 5
```
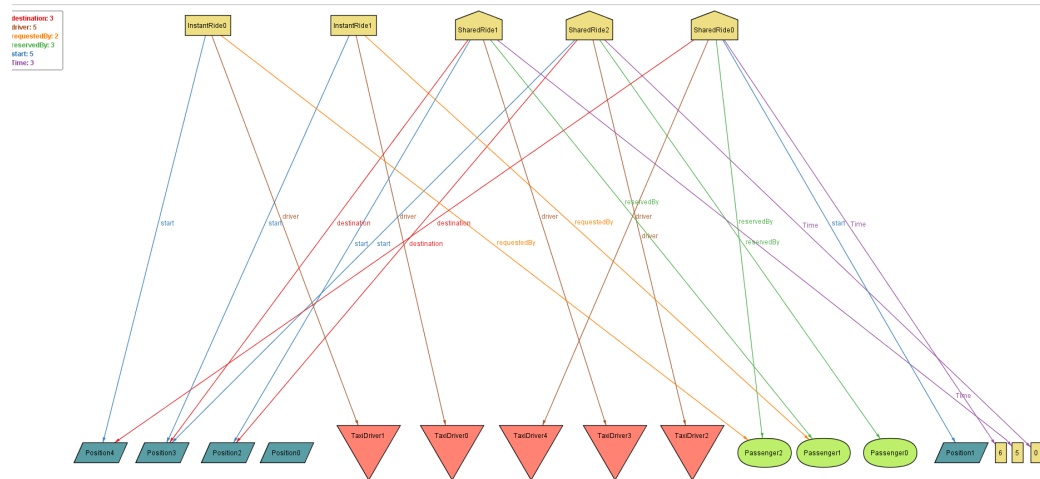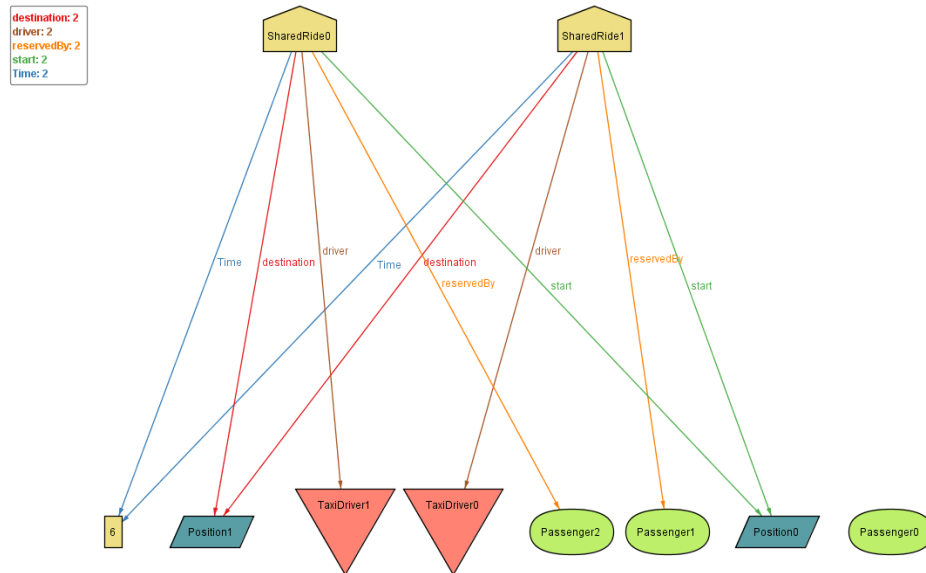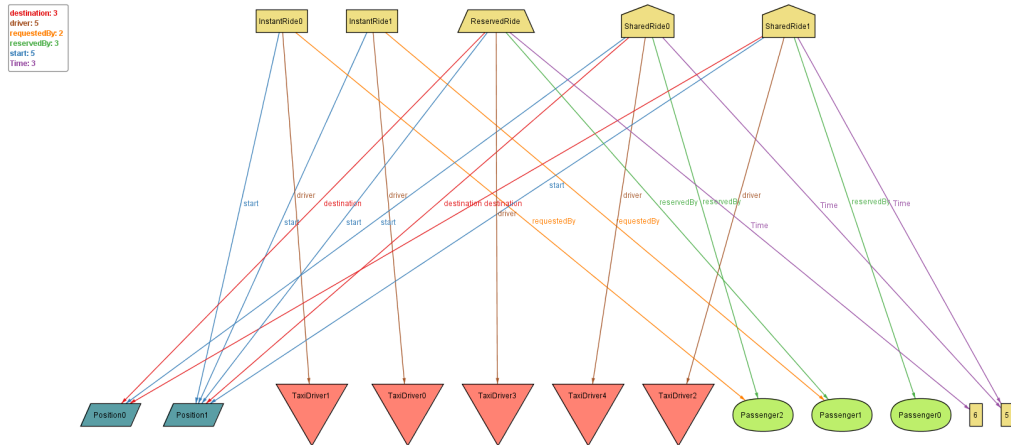
6.5 Generated Worlds

6.5.1 Show Predicate

6.5.2   Show Shared Predicate

6.5.3 Show Rides Predicate

7 APPENDIX

7.1 Used tools

To create the RASD we used the following tools:

- **MikTex**
  Distribution of the typesetting system LaTeX
  http://www.miktex.org/download

- **TexStudio**
  OpenSource cross-platform LaTeX editor we used to write the RASD.
  http://texstudio.sourceforge.net

- **StarUML**
  UML modelling tool we used to build the graphs
  http://staruml.io/download

- **Balsamiq**
  The mockup builder we used to design the mockups.
  https://balsamiq.com/download/

- **Alloy analyzer 4**
  used to build strong and substantial models
  http://alloy.mit.edu/alloy/download.html

- **GitHub desktop**
  Desktop application of the web-based Git repository hosting service.
  Used to collaborate in the team and to have a track of the changes.
  https://desktop.github.com/

7.2 Hours of work

This is the time we spent redacting the RASD

- Belotti Nicola ~31 hours

- Chioso Emanuele ~31 hours

- Colombo andrea ~31 hours