Politecnico di Milano
A.A. 2015-2016
Software Engineering 2 Project

My Taxi Service

Requirement Analysis and Specification Document (RASD)

Giovanni Brena (858328), Andrea Canale (858638)

Contents

1 Introduction

1.1 Purpose

The intention of this document is to provide a detailed description of the MyTaxiService system, starting from an analysis of the problem and the concerned domain assumptions to reach a clear Requirement Analysis. It will focus on software goals and objectives, functional and non-functional requirements, limits and possible future extensions. This document has to be perceived as a guideline for developers and analysts who will take care of implementation, as product overview for costumers and stakeholders. It will also offer previews of some UI elements required by the system and many scenarios regarding typical use of the software.

1.2 Actual System

An existing physical taxi service is supposed to be already present within the city, MyTaxiService will integrate it offering new interactive tools for customers and will use information given by the existing service.

1.3 Scope

MyTaxiService's main scope is to offer an easy, fast, secure and attractive way of interaction between taxis and their customers. It provides both a Web Client and mobile App. Users have the possibility to request a taxi from anywhere in the city, easily follow its position on a map, know the amount of time needed to wait and pay for the run using their preferred payment method. They can also submit reservations specifying origin, destination, day and time. Convenience and smartness also concerns the taxi drivers who are automatically informed by the system about incoming calls. Through the mobile app, drivers can accept or reject calls, retrieve information about customers and pick-up destinations, and easily manage payments. Fairness between taxi drivers and passengers is guaranteed using a specific queue management system.

1.4 Domain Properties

These properties have been considered as assumptions of the domain:

- Information about taxi drivers is supposed to be held by the existing system. MyTaxiService will not be concerned with verifying licenses, vehicles or any information about drivers
- Taxis positions are monitored in real-time by a GPS system and are always available. The GPS system is to be precise enough not to cause ambiguity in any case.

- Every driver is to have all the necessary hardware and software tools checked and working before starting a run.
- Incoming calls will only be received from areas of the city which are covered by the service.

1.5 Stakeholders

1. The city council
2. Taxi service management
3. Developers and technicians

1.6 Actors

Below are listed all possible entities interacting with the system:

- VISITOR [V]
  System entries defined as VISITOR are those made by anyone not recognized as being registered.
  VISITORS are only allowed to interact with the Web Client public area, with the mobile application Login Screen and registration form.

- REGISTERED USER [U]
  Registered users are entries into the system made by submitting correct login data for private customer service. Registered users have access to the Web Client public area, personal area, services area and to all functions of the mobile application (customer version).

- REGISTERED DRIVERS [D]
  Registered drivers are entries to the system made by submitting correct login data for the driver's service. They have access to Web Client public area, personal area and to all functions of the mobile application (driver's version).

- SYSTEM ADMINISTRATOR [A]
  Privileged actor reserved to taxi system management. Can modify data about drivers, add new ones and see run history. Has no access to users data.

1.7 Goals

List of MyTaxiService application goals:

1. EACH ACTOR should be able to:

- Easily find information about the system's functions and contacts/support for the system (tutorial, info line, email).

2. VISITOR should be able to:

   - [G-V1] Become a REGISTERED USER or REGISTERED DRIVER through a registration form.

3. REGISTERED USER and DRIVER should be able to:

   - [G-UD1] Log onto the application through his/her username and password.
   - [G-UD2] See/Update personal information.

4. REGISTERED USER should be able to:

   - [G-U1] Perform an immediate taxi call, cancel a pending call, know their request status and position of the reserved taxi.
   - [G-U2] Request a taxi reservation, cancel a reservation and see reservation info.
   - [G-U3] Choose payment method for each taxi run (cash, credit card, paypal…) and accept payment.

5. REGISTERED DRIVER should be able to:

   - [G-D1] Inform the system about his availability / unavailability
   - [G-D2] Accept or reject a taxi call
   - [G-D3] Know information about accepted calls (customer position and personal info)
   - [G-D4] Secure payment after each run using the method chosen by customers and confirming the payment.
   - [G-D5] Inform the system about problems related to runs

6. SYSTEM ADMINISTRATOR should be able to:

   - [G-A1] Add, remove or update taxi driver profiles
   - [G-A2] Have full visibility of calls, runs and reservations

1.8 Definitions, Acronyms, Abbreviations

Here are defined keywords often used within this document:

- USER
  Identifies a REGISTERED USER [U] actor.

- DRIVER
  Identifies a REGISTERED DRIVER [D] actor.

- SYSTEM
  If not differently specified, it identifies the MyTaxiService system.

- APP
  Identifies the MyTaxiService mobile client for USER use.
- DRIVER APP
  Identifies the MyTaxiService mobile client for DRIVERS use.

- RUN
  A single taxi journey using the system.

- RESERVATION
  A single reservation made by a user.

- TAXI AREA
  One of the city's areas in which available taxis are located.

- TAXI QUEUE
  Refers to a taxi area identifying the ordinated queue of available taxis in that area.

- ETA: Estimated time of arrival

- TD :Taxi Driver

- TC: Taxi Code

- DBMS: Data Base Management System

These are the used abbreviations:

- [V], [U], [D], [A] : Visitor, User, Driver, Administrator
- [G] : Goal
- [G-Yn] : Goal of index n referred to actor Y (Y is in Actors{V,U,D,A}).

These are acronyms:

- RASD: Requirement Analysis and Specification Document
- DB: Database
- API: Application Programming Interface
- OS: Operating System

1.9 Reference Documents

- Specification document: Assignements 1 and 2 (RASD and DD).pdf
- RASD guidelines: IEEE Standard For Requirement Specification.pdf
- Given examples: examples of RASD documents

1.10 Documents Related

These are all document related to myTaxiService in order to organize the work better and provide to stakeholders the main ideas during the development:

- RASD: Requirement Analysis and Specification Document, where the problem is analyzed in order to define which are the main goals and how to reach them.

- DD: Design Document, where the real structure of the web application its defined.

- Testing Document: a report of testing of another myTaxiService project.

1.11 Used Tools

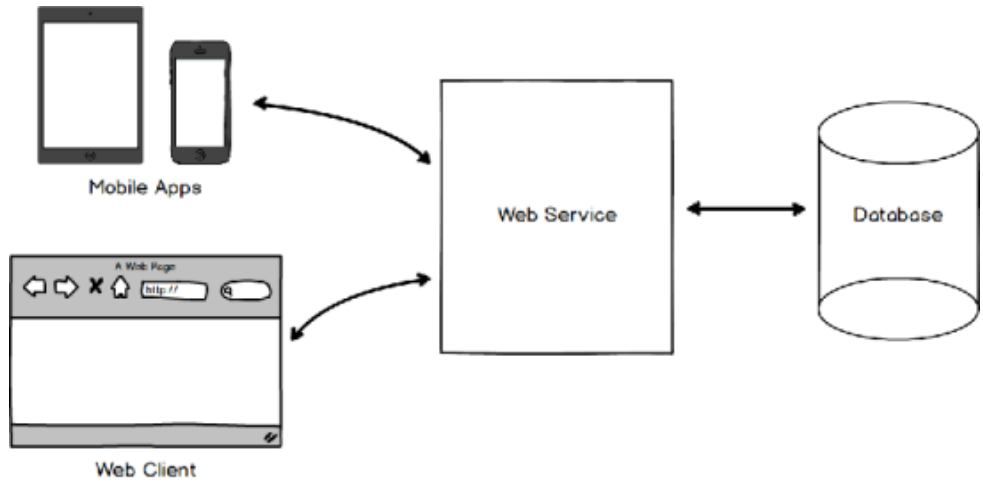
- Microsoft Word 2013 :to redact and to format this document.
- Gliffy (http://www.gliffy.com/): to create Use Cases Diagrams, Sequence Diagrams, Class, Diagrams and State Machine Diagrams
- Alloy Analyzer(http://alloy.mit.edu/alloy/): to prove the consistency of our model.
- Adobe Photoshop (http://www.photoshop.com/): to modelling some image.
- Balsamiq Mockups(http://balsamiq.com/products/mockups/): to create mockups.

2 Overall Description

2.1 Product perspective

Application modules will be as following:

1. A Web Service will contain system logic implementation and will offer methods to client applications.
2. DBMS will collect and provide system data. It will be accessible only by the Web Service.
3. A Web Client will be accessible from the web and will offer a User Interface providing services to recognized users.
4. A Mobile Application will offer services to users and drivers.



2.2 User characteristics

The target audience are people of any age and any type living or moving inside the city area. They must be able to use a web browser or mobile app and they need internet connection.

2.3 Constrains

2.3.1 Regulatory policies

The system will work under the rules and limitations imposed by taxi regulations. Their position is required. The use of personal information will be regulated by a privacy policy subscribed to by users during the registration process.

2.3.2 Hardware limitations

Usage of the system requires having a smartphone or tablet running one of the supported OS and supported versions. It also requires internet connection.

2.3.3 Interfacing with other applications

The mobile app has interfacial access to third party applications:

- GPS and Maps applications such as "Google Maps"
- SMS and texting applications
- Mail app

2.4 Assumptions and Dependencies

- The system do not allow new taxi requests for users who already have a pending request to be processed or another taxi run on progress
- Each run has maximum of 4 people partecipating
- In case of requests from areas in which no taxis are currently available the system will scan queues from adjacent areas. If unvailability persist the user is notificated and the request canceld.
- Supported payments methods are cash or electronic payments
- Amount of payment is typed by drivers inside the application at the end of the run. Both passenger and driver are notificated about the payment outcome
- System will not handle or pursue problems related to nonpayment by passengers
- A taxi call is "active" if it has been sent to the system, processed and if a driver has been accept it, otherwise the call is "pending". When the taxi journey starts the call switch to status "in progress" while at the end of the run it goes to "closed"
- In any time users can have information about a taxi call (availability changes between various status). The system will show status, position of the driver, driver's profile public section, expected arrival, actual journey distance and time, final price
- Reservations can't be modified. Changes on date/time/location will require to cancel reservation and create a new one
- In this Document we define as TaxiException events who lead a taxi driver to reject a previously accepted Call, these can be a "Car BreakDown" or also "Rejected Call". The system will just save these in order to provide a view of taxi driver's behavior to the stakeholder.

2.5 Future extensions and additional goals

These goals are not considered into this RASD but mentioned as possible extensions:

- Allow users to choose BOOKING OPTIONS as type/dimensions of car, multiple taxis/taxi bus, eco/electric vehicols, driver rating…
- Let the users to RATE drivers after their taxi experience
- Let the user to register some routes as FAVORITES
- Let the users to register FAVORITE DRIVERS
- Let the system prevent the misuse of the application by User who cancel a Taxi Call repeatedly and by Taxi Driver who raises TaxiException repeatedly.

3 Specific Requirements

3.1 Functional Requirements

3.1.1 [G-V1] Visitors should be able to become registered users through a registration form.

- [R1] Visitors can only access the login screen and sign up screen
- [R2] Visitors must provide a valid email address not already present in the system.
- [R3] Visitors must choose a username not already present in the system.
- [R4] Visitors must choose a password according to the minimum security standard.
- [R4] Visitors must confirm registration through a link sent to the provided email address.
- [R5] Visitors are asked to provide electronic payment information (credit card information or PayPal account). In case of no info provided the user will only be able to pay by cash.
- [R6] Visitor must accept Privacy Policies

3.1.2 [G-UD1] A registered user/driver should be able to log onto the application through his username and password.

- [R1] Both username and password must be present within the system
- [R2] System must provide a password retrieving service.
- [R3] No login attempts limit will be implemented.

3.1.3 [G-UD2] Registered user/driver should be able to see and update personal information.

- [R1] User/driver must be logged into the system
- [R2] Modifying a username is not allowed
- [R3] Modifying an email address will require verification of new email.

3.1.4 [G-U1] Registered user should be able to perform an immediate taxi call, cancel a pending call, know request status and reserved taxi position.

- [R1] User must be logged in.
- [R2] Users are allowed to perform a call only if they do not have any other pending or active calls in progress.
- [R3] User can cancel a taxi call if the status is “pending” or “active”

3.1.5 [G-U2] Registered user should be able to request a taxi reservation, cancel a reservation and see reservation info

- [R1] User must be logged in
- [R2] Users can perform a reservation correctly providing this information: date and time of departure, departure position, arrival position
- [R3] Date and time must be in the future and around 120 minutes after the time of the actual call.
- [R4] Reservations can only be cancelled with a minimum of 15 minutes before the assigned departure time.

3.1.6 [G-U3] Registered user should be able to choose payment method for each taxi run and accept payment

- [R1] User must be logged in
- [R2] User must be at the end of a run
- [R3] Electronic payment is allowed only for users who have provided credit card or paypal information

3.1.7 [G-D1] Registered drivers should be able to inform the system about his/her availability or unavailability

- [R1] Driver must be logged in
- [R2] Driver must not be associated with another active taxi call

3.1.8 [G-D2] Registered drivers should be able to accept or reject a taxi call

- [R1] Driver must be logged in
- [R2] Driver status must be “available”

- [R3] Driver must have received a taxi call

3.1.9 [G-D3] Registered drivers should be able to know information about accepted calls (customer position and personal info)

- [R1] Driver must be logged in.
- [R2] Driver must have an active taxi call.

3.1.10 [G-D4] Registered drivers should be able to set the payment amount after each run using the method requested by customers, and confirm payment.

- [R1] Driver must be logged in.
- [R2] Driver must have a closed taxi call.
- [R3] Payment method selection requires price to be defined.
- [R4] Electronic payments require customer confirmation.

3.1.11 [G-A1] System administrator should be able to add, remove or update driver profiles.

- [R1] System Administrator must be logged in.

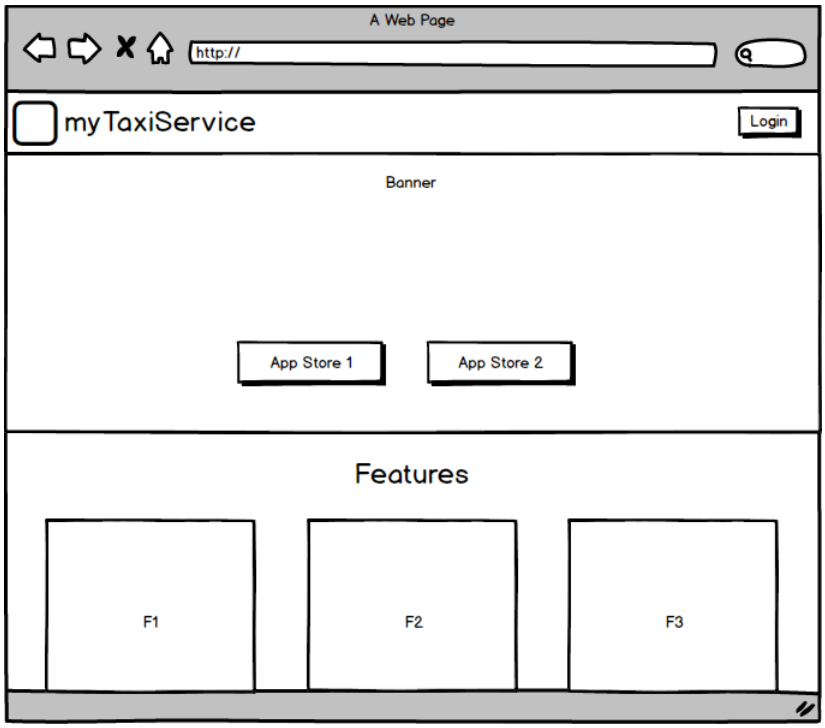
3.2 Non-Functional Requirements

3.2.1 User Interface

We will now demonstrate some UI samples to show how the system's design would be presented, both on the Website and mobile apps.

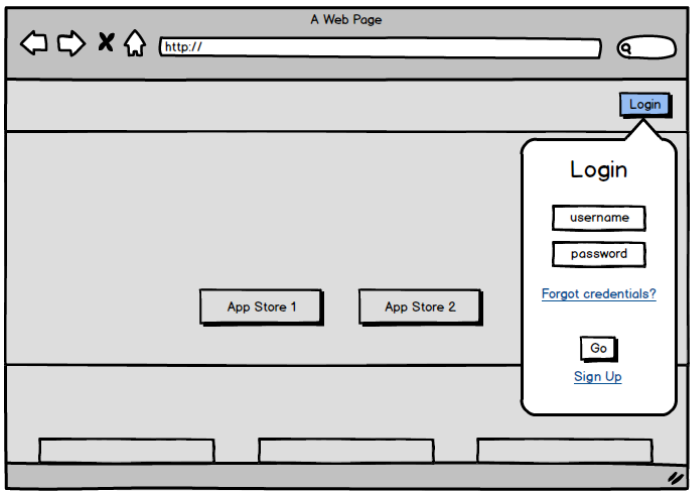
3.2.1.1 – Website Homepage

Here is an example of the MyTaxiService website homepage. It will contain links to app stores to download the application, a link to the login page and a short description of the system features.



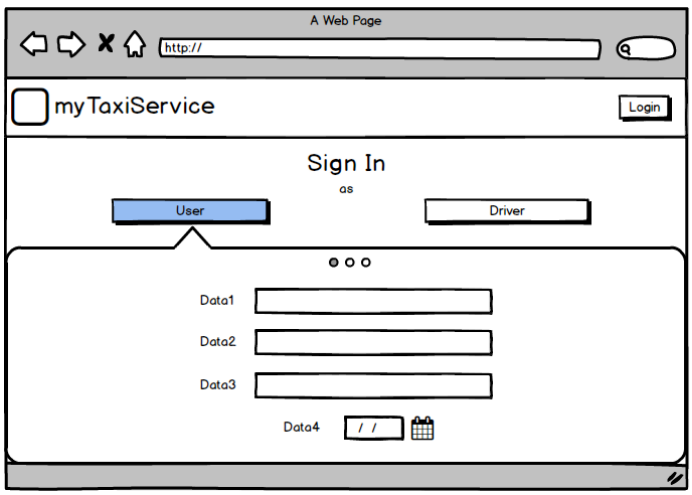
3.2.1.2 – Login form

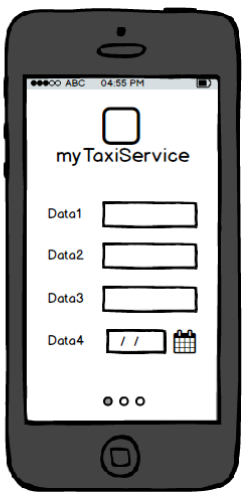
The log in form will allow access to the registered user's services. Users have to provide their username and password in the process. Links to signing in and password recovery have to be present. This screen will be included both on the website and app.



3.2.1.3 – Sign in form

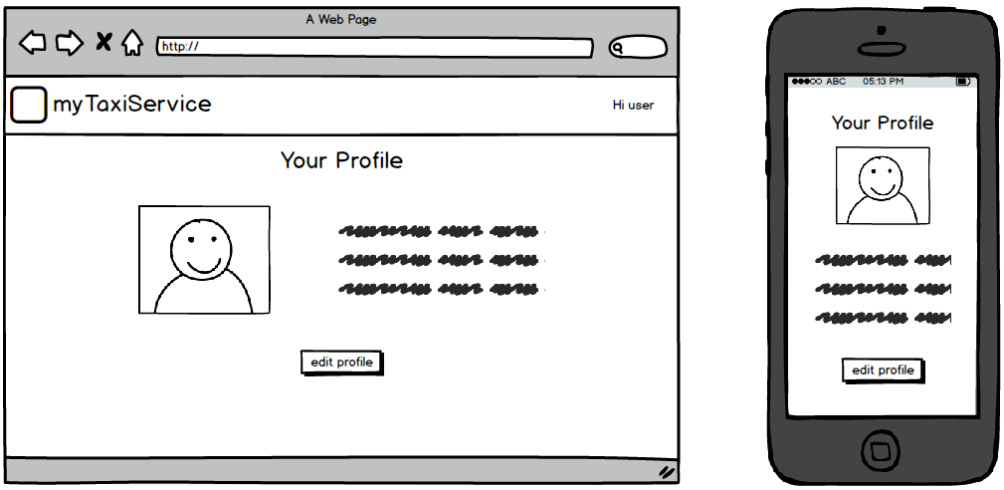
This screens show first part of Sing In process. Website also allow new drivers requests while application only allow user subscriptions.





3.2.1.4 – Profile section

Profile section will contain personal user information. Here is possible to edit part of these information.



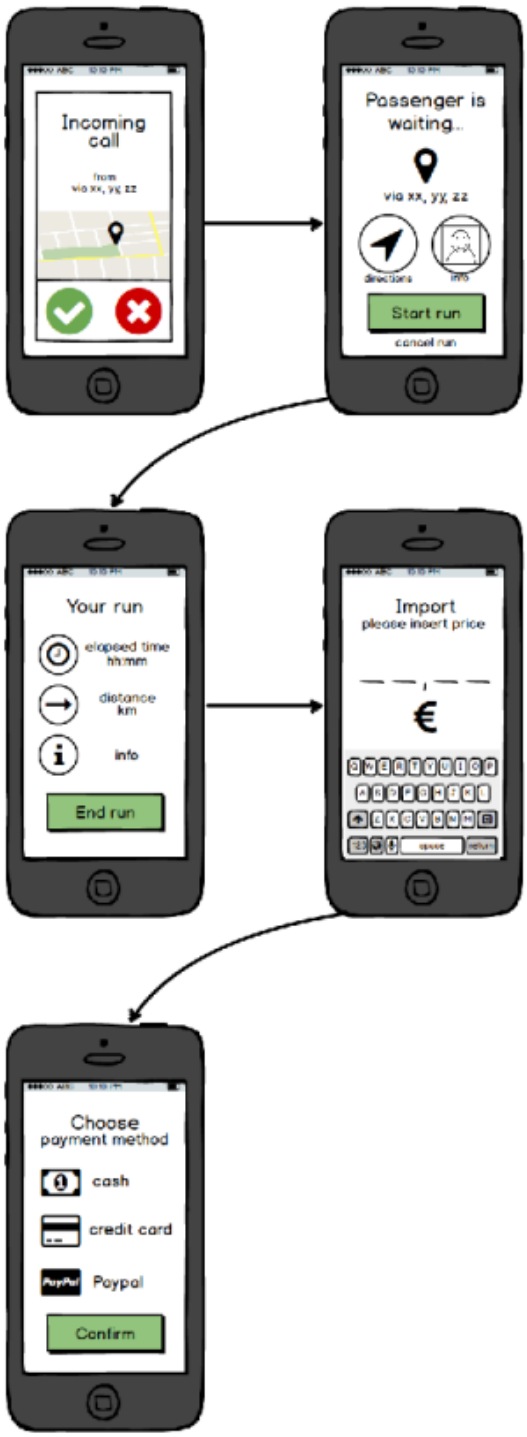
3.2.1.5 Reservation Screen

Allows taxi reservation. Require information about date, hour, departure and destination.

3.2.1.6 – Call taxi action screen
Perform call action. This screen should be as more simple as possible in order to inspire ease and safety to users.

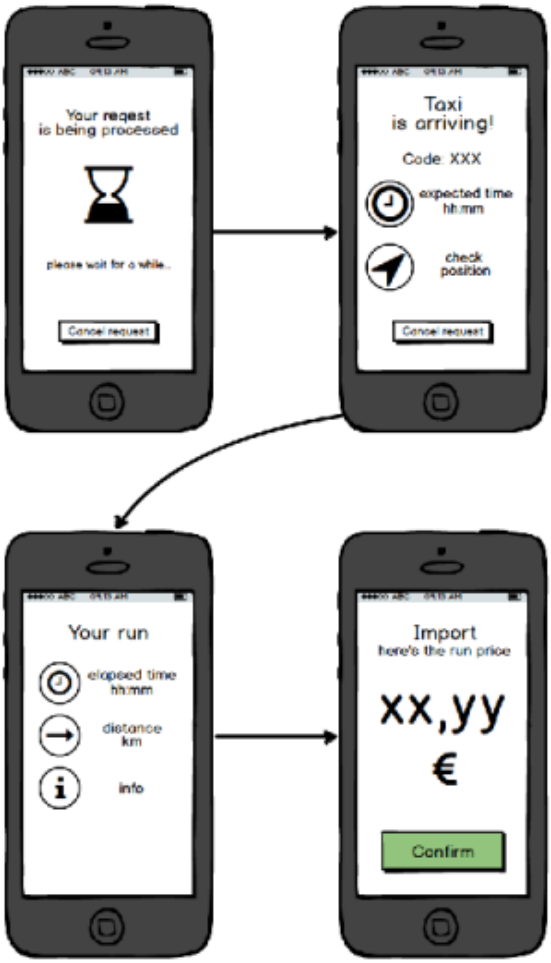
3.2.1.7 – Call screen flow DRIVER

This UI flow shows how does a call action should be handled by driver.



3.2.1.8 – Call screen flow USER

This UI flow shows how does a call action should be handled by user.



3.3 Scenarios

3.3.1 Scenario1

Rob after a party needs a taxi to go home. He open "myTaxiService" on his phone,login in it and accept the request "allow the application to use Geolocalization".Then he click on "Call Taxi" button and he was asked to wait a moment. After few seconds ,a pop-up appear with ETA and TC. Since Rob arrived at home the driver communicates him the ride fare, Rob pays it with cash and driver notify payment to the system.

3.3.2 Scenario2

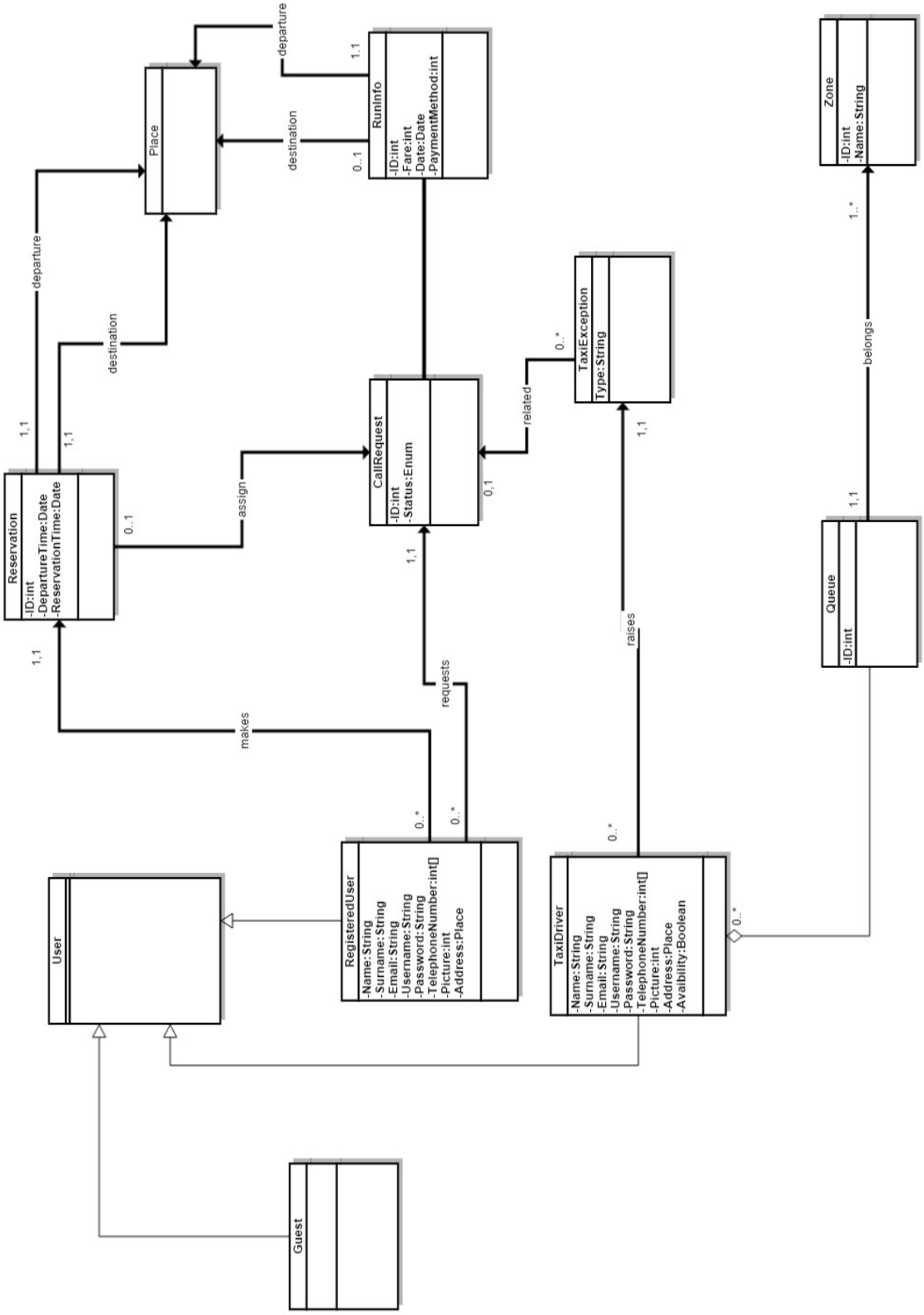
Sansa has to take a plane on Sunday for work but she realizes that no one could take her to the airport. She has to book a Taxi ,so she open "myTaxiService" from his laptop and click on "Reserve a Taxi ". She select the pick-up place(=Winterfell),destination(=airport),date and departing time. At the time she chosen the taxi arrived and brings her to airport. After she was communicated the fare she chooses to pay with credit card previously registered on app.Once she left she was notified on her phone the fare drawn from his credit card.

3.3.3 Scenario3

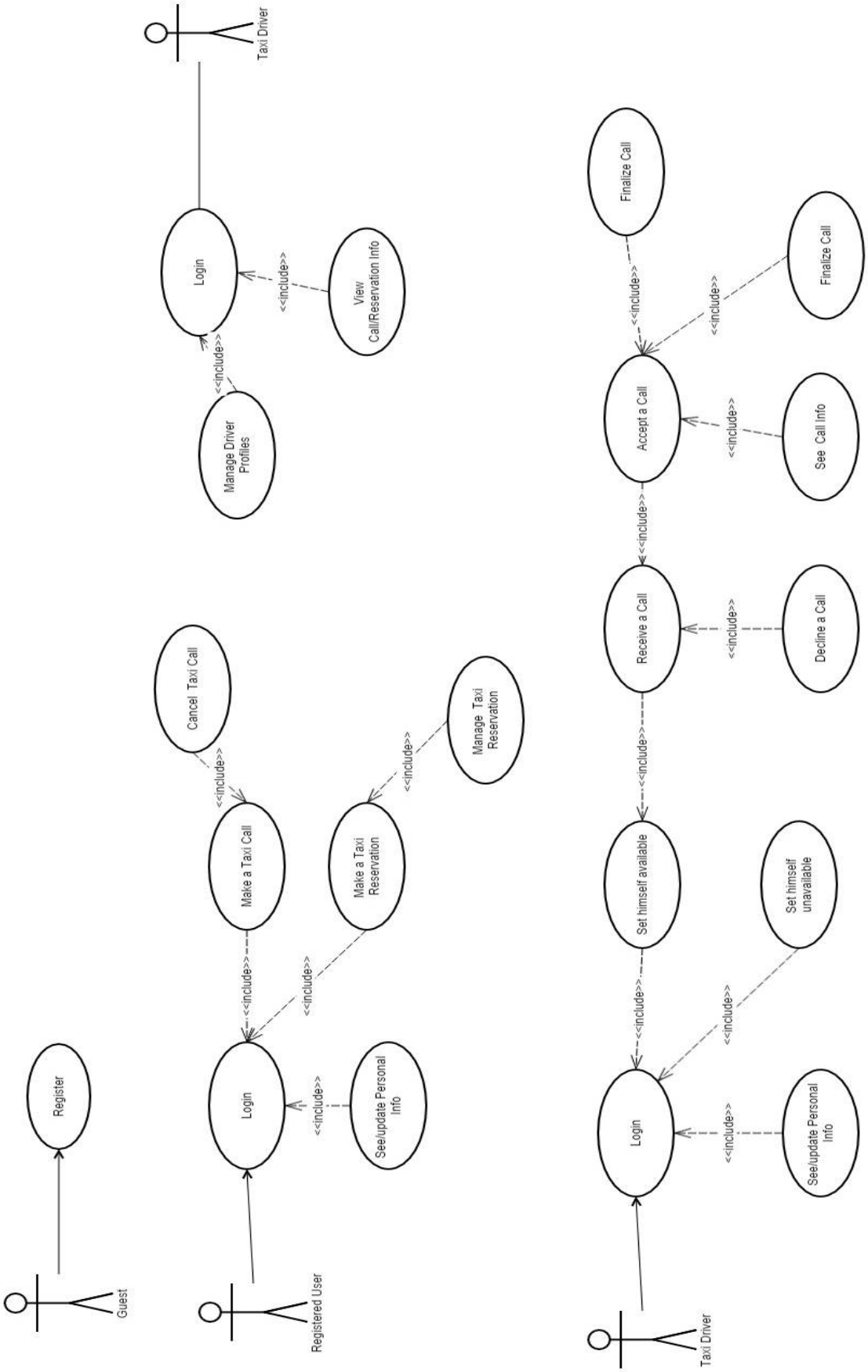
Mark is a Taxi Driver who is logged in myTaxiService through his mobile phone.He is bottled up in traffic jam due to a demonstration in DownTown.He receives a call from a new passenger but he decides to decline it due the fact that it will take too much time to reach him.After a while Mark can go away from the traffic,he receives a call from a new passenger .He accepts it and follows the GPS.

3.4 UML models and Use Cases models

3.4.1 Class Diagram



3.4.2 Use Cases diagram



3.4.2 Use Cases Description

In this paragraph some use cases will be described in a detailed way .These use cases can be derived from the scenarios and Use Case Diagram.

[G-V1] Guest registration:

NAME	Guest Registration
Actor	GUEST
Goal	[G-V1]
Input Condition	Null
Event Flow	1. Guest on the homepage clicks on "register" button to start the registration process 2. Guest visit fills at least all mandatory fields. 3. Guest clicks on "Confirm" button. 4. The application will save the data on DB.
Output Condition	Guest is successfully ends registration and become a Registered User. From now he/she can log in to the application using his/her credential and start using myTaxiService.
Exception	1. The Guest is already a Registered User. 2. One or more mandatory fields are not valid. 3. Username chosen is already used by another user. 4. Email chosen is already associated to another user.

[G-V1] Login:

NAME	Login
Actor	Registered User, Taxi Driver
Goal	[G-V1]
Input Condition	User/Taxi Driver are already registered in the system.
Event Flow	1. User on homepage clicks on "Login" button. 2. User fill "username" and "password" fields
Output Condition	The system show the user his personal page.
Exception	The password and/or username inserted by the user are wrong.The system shows an error message to the user.

[G-U1]Taxi call:

NAME	Make a Taxi Call
Actor	Registered User
Goal	[G-U1]
Input Condition	1. User is already logged in the application. 2. User enabled GPS localization.
Event Flow	1. User clicks on "Call a Taxi" button. 2. System show "Call a Taxi " page. 3. User clicks on "Call Taxi Now" button.
Output Condition	The system show user "Call info"
Exception	1.If the GPS localization is not enabled, the system before showing "Call a Taxi " page will ask to user to enable it showing the browser/OS related pop-up for enable it. 2. If system doesn't find any TD or time exceed ,the system show a message of noFoundTD to User.

[G-UD2] Call info:

NAME	See Call Info
Actor	Registered User
Goal	[G-UD2]
Input Condition	1. User has made a Call for a taxi 2. User is waiting for Response
Event Flow	1. Application finds an available Taxi Driver and retrieve his information. 2. Application show the information to the User.
Output Condition	The system show to User: • ETA • Taxi Code
Exception	If System don't find a Taxi Driver in a "reasonable" Time it notify user and invite him to try again later.

[

G-D2]Accept call:

NAME	Accept a Call
Actor	Taxi Driver
Goal	[G-D2]
Input Condition	1. TD is already logged in the application. 2. TD set himself as "available".
Event Flow	1. System show a "Call" pop-up on TD application with two button: • "Accept" button. • "Refuse" button. 2. TD clicks on "Accept" button.
Output Condition	The system assign the TD to the call,removing it from the queque.
Exception	

[G-D2] Refuse call:

NAME	Refuse a Call
Actor	Taxi Driver
Goal	[G-D2]
Input Condition	1. TD is already logged in the application. 2. TD set himself as “available”.
Event Flow	1. System show a “Call” pop-up on TD application with two button: • “Accept” button. • “Refuse” button. 2. TD clicks on “Refuse” button.
Output Condition	The system forward the request to another TD and move the TD in last position of the queque.
Exception	

[G-U2] Taxi reservation:

NAME	Make a Taxi Reservation
Actor	Registered User
Goal	[G-U2]
Input Condition	3. User is already logged in the application. 4. User click on “Reserve a Taxi button”.
Event Flow	1. System show him the form that the user has to fill with the following information: • Destination Place • Starting Place • Day • Hour • Payment Option 2. User fill the previous fields and click on “Reserve”
Output Condition	The system add the reservation into the database and shows a confirmation message.
Exception	If any of the fields is not correct, the system shows an error message to the user.//data e time 2 ore prima della corsa

[G-U2] Cancel taxi reservation:

NAME	Cancel a Taxi Reservation
Actor	Registered User
Goal	[G-U2]
Input Condition	1. User is already logged in the application.
Event Flow	1. User clicks “Reservation List” button. 2. User selects a Reservation. 3. User clicks on “Cancel Reservation” at least,and start deleting process.
Output Condition	The system delete the reservation from the DB and from Reservation List of the User
Exception	If User clicks “Cancel Reservation” less than 15 minutes before the scheduled departing time the system will show an error message.

[G-U1] Cancel taxi call:

NAME	Cancel a Taxi Call
Actor	Registered User
Goal	[G-U1]
Input Condition	1. User is already logged in the application. 2. User is in “Call info” page. 3. User clicks on “Cancel Call” button.
Event Flow	1. System show a Confirmation pop-up with two button: • Yes • No 2. User clicks on “Yes” button
Output Condition	The system //bo
Exception	If User clicks “Cancel Reservation” the system will show an error message.

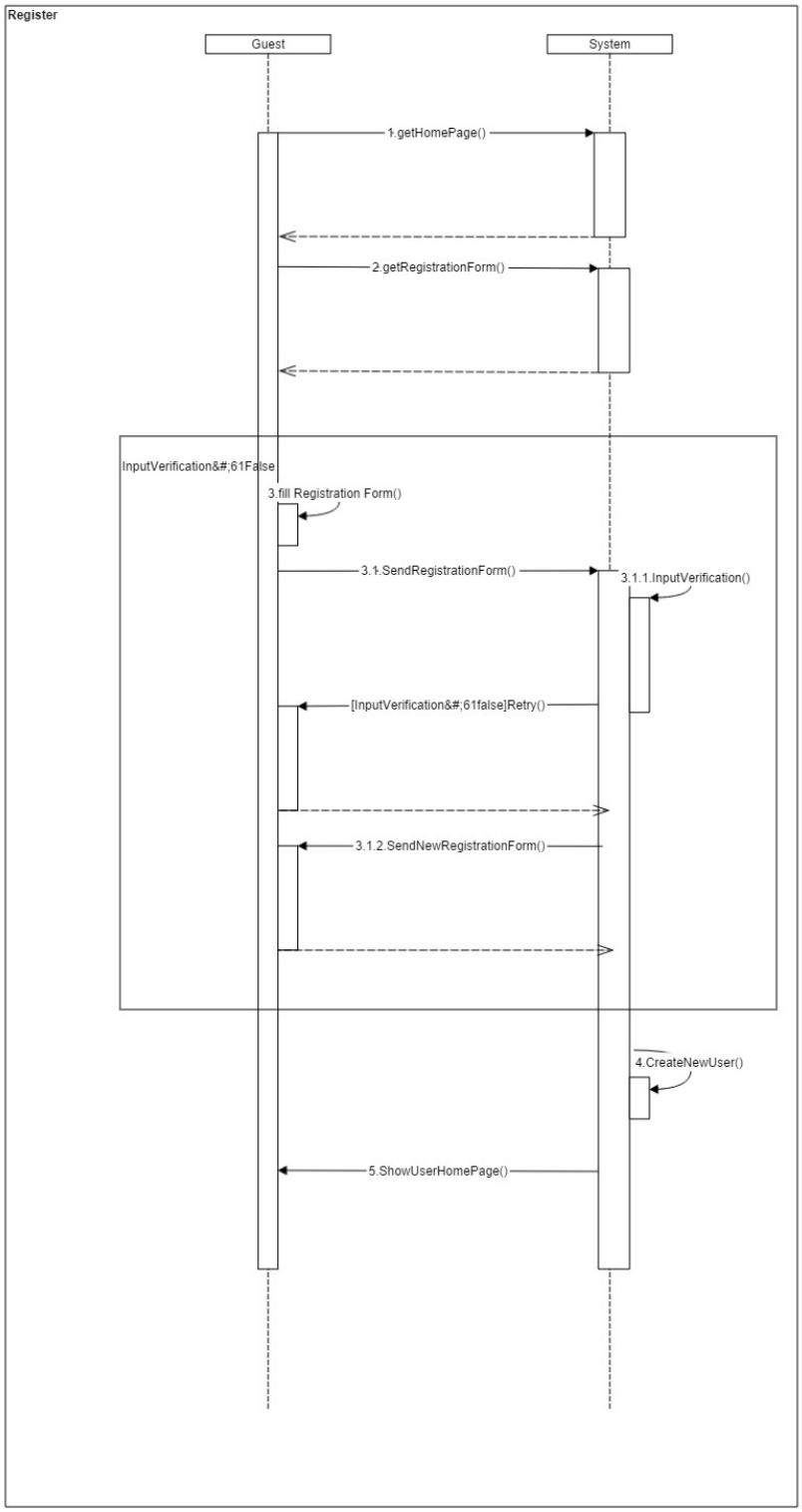
[G-D1] Driver Availability:

NAME	Set himself available
Actor	Taxi Driver
Goal	[G-D1]
Input Condition	TD is already logged in the application.
Event Flow	1. TD request Status Page. 2. System show Status Page. 3. TD clicks on “Set available” button. 4.
Output Condition	The system shows the updated Status page and assign the TD to the queue accordingly to his position.
Exception	

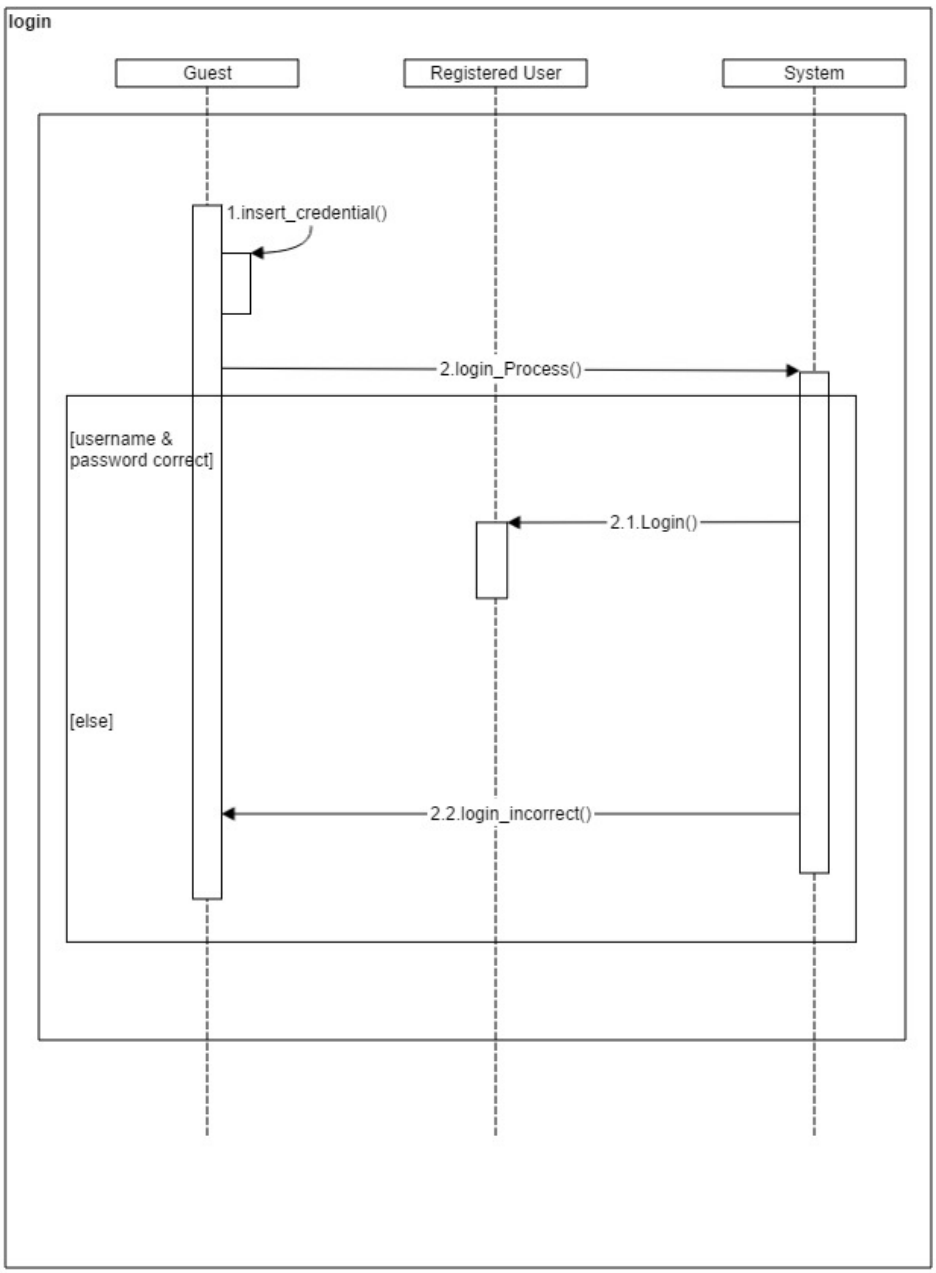
3.4.3 Sequence Diagrams

Here the Sequence Diagrams of the previous descibed use cases .

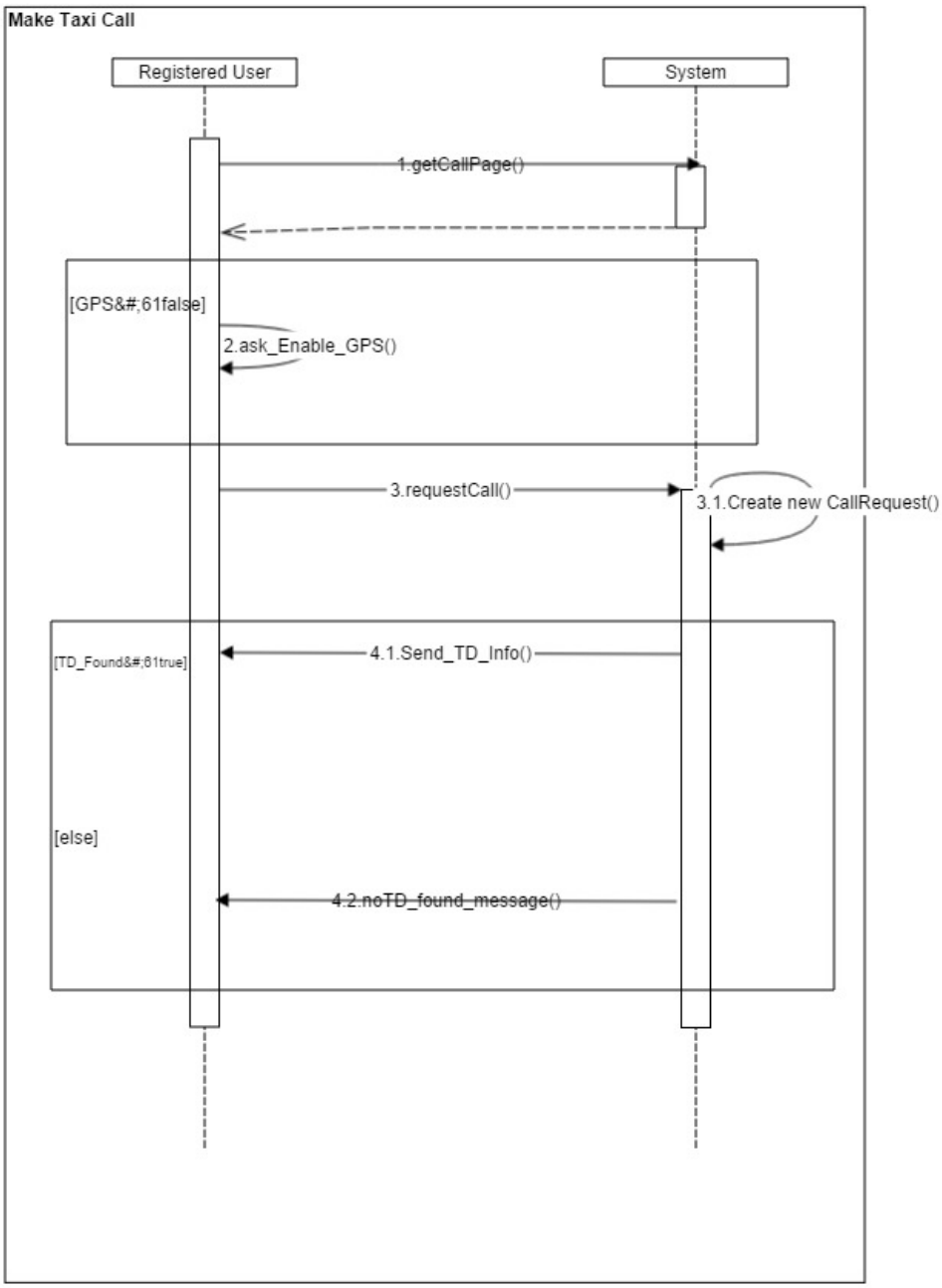
[G-V1] Register diagram:



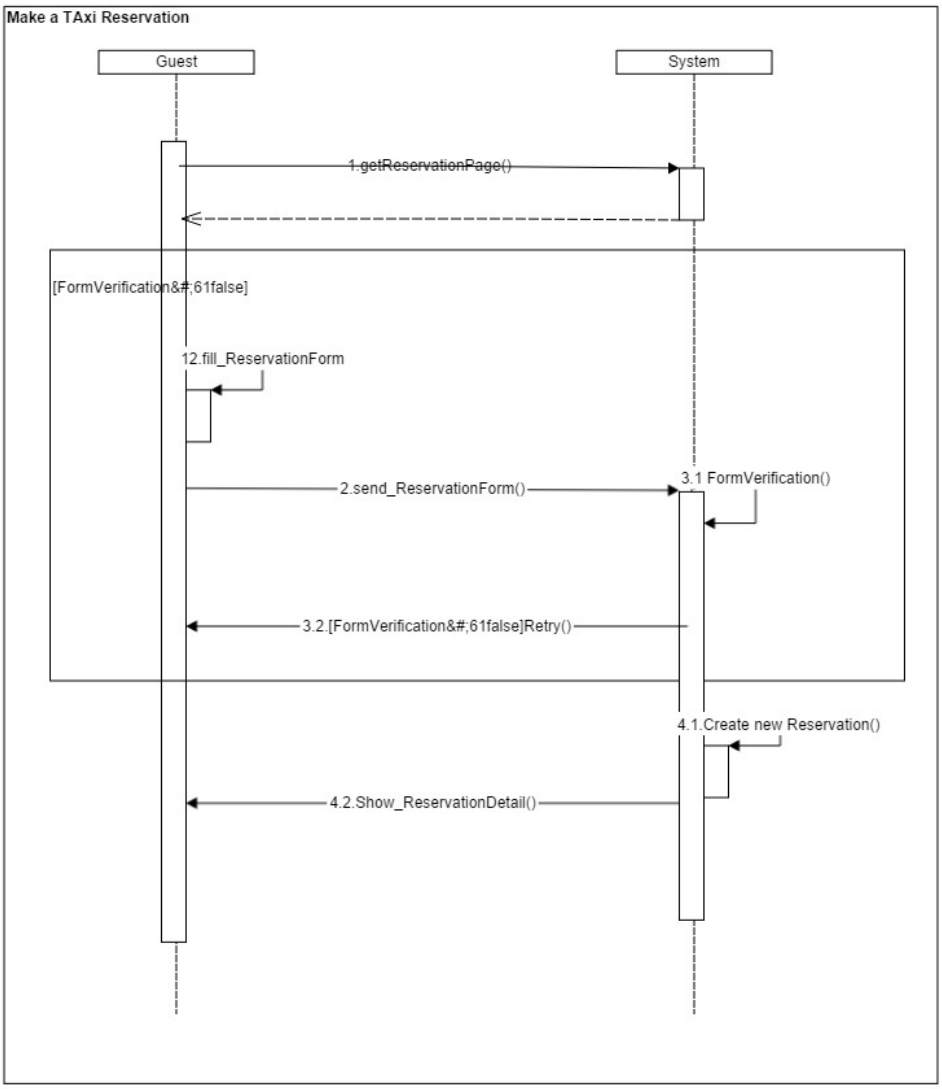
[G-V1] Login Diagram:



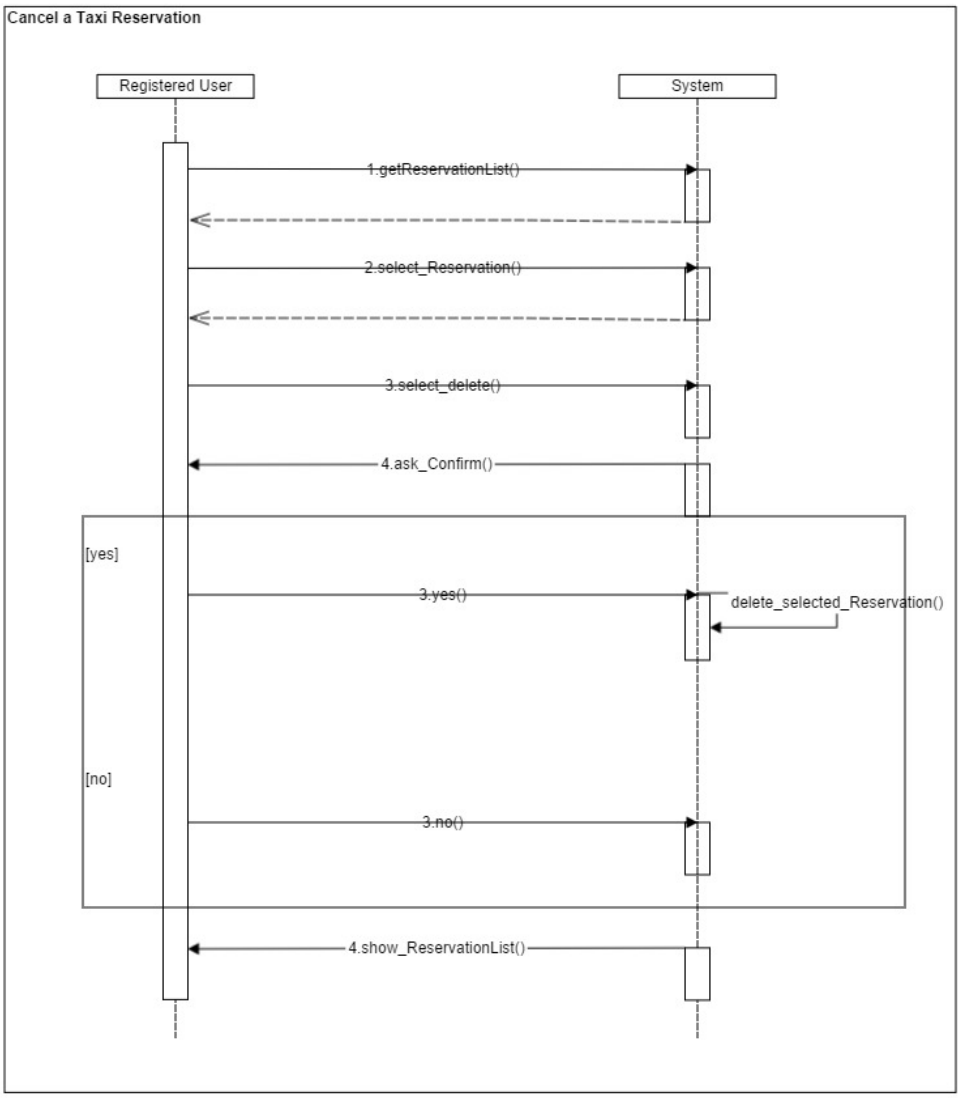
[G-U1] Taxi Call diagram:



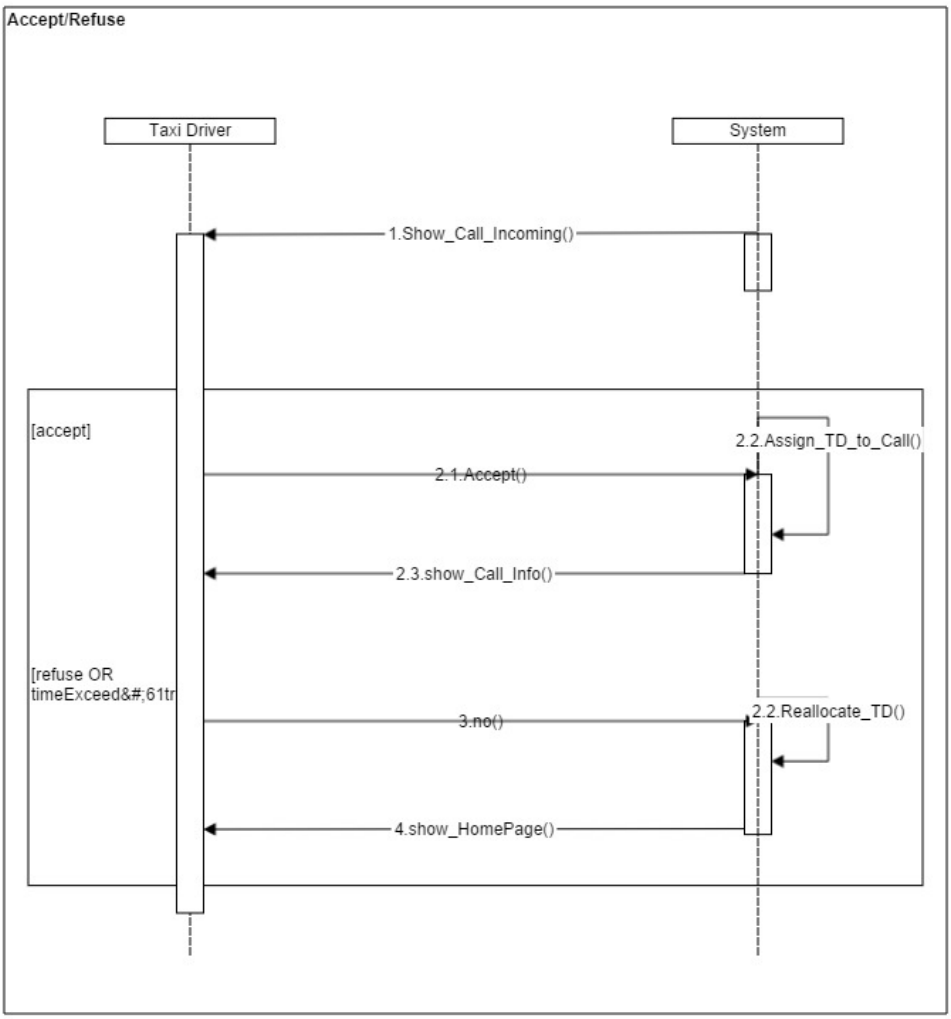
[G-U2] Reservation diagram:



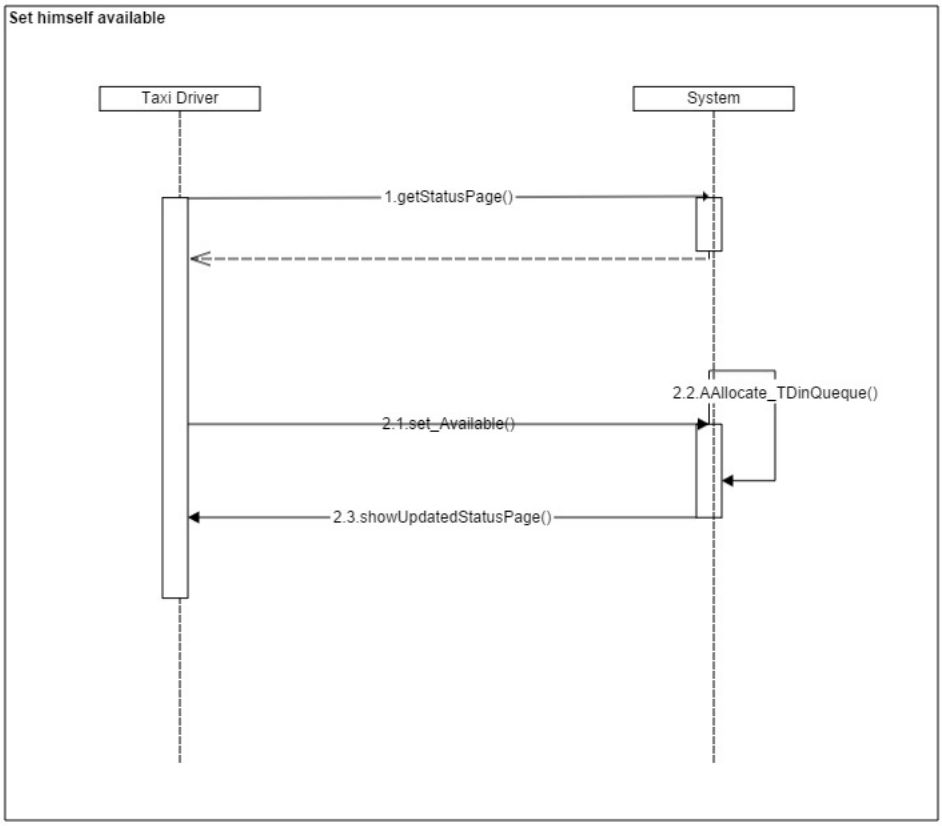
[G-U2] Reservation Cancel diagram:



[G-D2] Taxi accept or refuse call diagram:



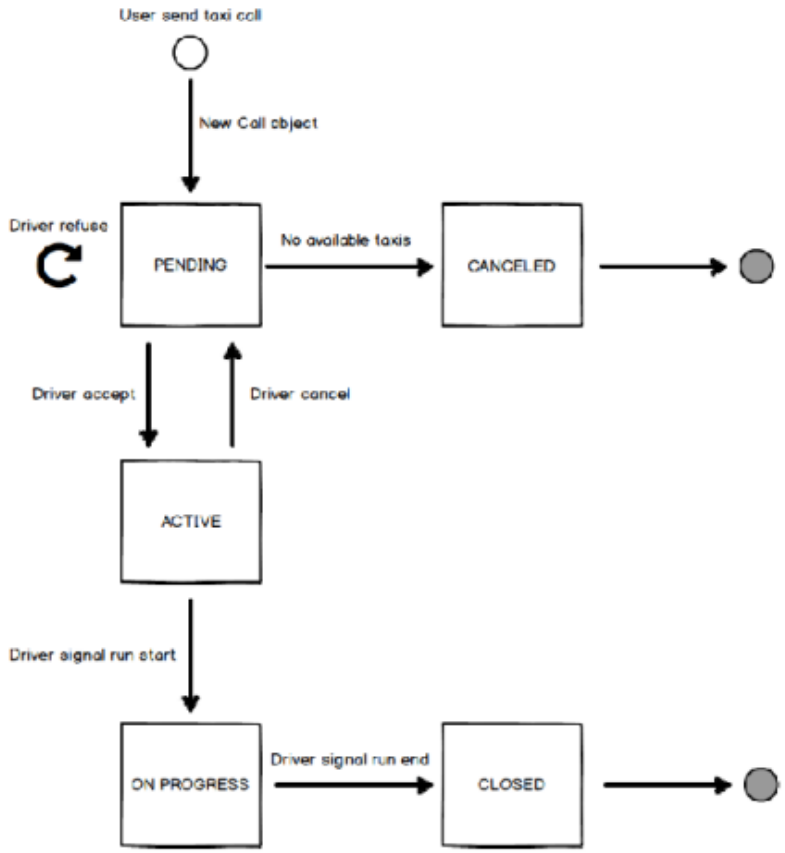
[G-D1] Driver set Availability diagram:



3.5 State Chart Diagrams

3.5.1 Taxi call state digram

Here is shown how a Call object can evolve through its possible status.



4 Alloy Modeling

The complete alloy file (.als) could be find on our Google code repository. The following alloy model has been created using class diagram. We divied the code into part logic parts (signatures, facts, asserts and predicates). The last section contain generated worlds.

4.1 Signatures

```
// SIGNATURES

sig User {}

sig Passenger extends User{
   activeCall: lone CallRequest,
   books:set Reservation,
}

sig TaxiDriver extends User{
   activeCall: lone CallRequest,
   exceptionRaised:set TaxiException
}

one sig Place{}

one sig City {
   zones: some Zone
}

sig Zone {
   taxiQueue: one TaxiQueue
}

sig TaxiQueue {
   taxiList: set TaxiDriver,
}

sig CallRequest {
   recap: one RunInfo,
}

sig Reservation{
   departure: one Place,
   arrival: one Place,
   call: lone CallRequest,
}

sig RunInfo{
   departure: one Place,
   arrival: lone Place,
               }

sig Exception{}

sig TaxiException extends Exception{
               related:CallRequest
}
```

4.2 Facts

```
// FACTS

fact OneCityPerZone{
   // every zone has exactly 1 city
   all z: Zone | (one c: City | z in c.zones)
}

fact OneZonePerQueue{
   // every queue has exactly 1 zone
   all q: TaxiQueue | (one z: Zone | z.taxiQueue = q)
}

fact OneQueuePerDriver{
   // ogni driver appartiene ad una sola queue
   all t: TaxiDriver | (lone q: TaxiQueue | t in q.taxiList)
}

fact OneRunInfoPerCall{
   //for every CallRequest must be only one RunInfo
   all i: RunInfo | one c: CallRequest |  i= c.recap
}

fact OneDriverForCall{ all t: TaxiDriver | #(t.activeCall)<2 }

fact NoDriverBusyOnQueue{
   //if a TaxiDrives has an active call ,he is  not in any TaxiQueue
   no t:TaxiDriver  |(#(t.activeCall)=1 and (some q:TaxiQueue | t  in q.taxiList ))
}

fact ReservationProperties{
         //Every Reservation has one Passenger as its creator
         all r: Reservation|one p:Passenger|r in p.books
         //Creator of Reservation and related Call Request must be the same
          all p:Passenger  | all r:Reservation | r in p.books implies r.call=p.activeCall
         //A CallRequest have at most one Reservation
         all c:CallRequest | lone  r:Reservation | r.call=c
}

fact CallRequestProperties{
            //All CallRequest must have one Passenger and at most one TaxiDriver
               all c:CallRequest | one  r:Passenger | r.activeCall=c
               all c:CallRequest | lone  r:TaxiDriver | r.activeCall=c
}

fact TaxiExceptionProperties{
            all e:TaxiException|one t:TaxiDriver|e in t.exceptionRaised
            all t:TaxiDriver|all e:TaxiException | all c:CallRequest |
               ( e.related=c and e in t.exceptionRaised) implies t.activeCall!=c
}
```

4.3 Predicates

```
//Predicates

//pred for assign a CallRequest to a Passenger
pred  addCallRequest (c:CallRequest,p:Passenger){
      #(p.activeCall)=0 implies p.activeCall=c
}

run addCallRequest  for 4


//pred for cancel ActiveCall of a Passenger
pred  cancelCallRequest  (p:Passenger){
   #(p.activeCall)=1 implies p.activeCall=p.activeCall-p.activeCall
}

run cancelCallRequest  for 5


//pred for assign a CallRequest to a TaxiDriver
pred assignCallRequestToDriver(t:TaxiDriver,c:CallRequest){
       all t1:TaxiDriver|(t1.activeCall!=c or #(t.activeCall)=0 )implies t.activeCall=c
}
run assignCallRequestToDriver for 5


//pred for reject a CallRequest from a TaxiDriver
pred rejectCallFromDriver(t:TaxiDriver,c:CallRequest){
      all t1:TaxiDriver|(t1.activeCall=c  )  implies t.activeCall=c-c
}
run rejectCallFromDriver for 5


//pred for add a TaxiException to  a TaxiDriver
pred  addTaxiException(t,t':TaxiDriver,e:TaxiException){
      e not in t.exceptionRaised implies t.exceptionRaised=t'.exceptionRaised+e

}

run addTaxiException  for 5


//pred for delete a TaxiException of a  TaxiDriver
pred  deleteTaxiException(t,t':TaxiDriver,e:TaxiException){
      e  in t.exceptionRaised implies t'.exceptionRaised=t.exceptionRaised-e

}

run deleteTaxiException  for 5


pred assignTaxiDriverToZone(z,z':Zone,t:TaxiDriver){
      t not in z.taxiQueue.taxiList implies z'.taxiQueue.taxiList= z.taxiQueue.taxiList+t

}

run assignTaxiDriverToZone  for 5


pred removeTaxiDriverFromZone(z,z':Zone,t:TaxiDriver){
      t  in z.taxiQueue.taxiList implies z'.taxiQueue.taxiList= z.taxiQueue.taxiList-t

}

run assignTaxiDriverToZone  for 5
```

4.4 Assertions

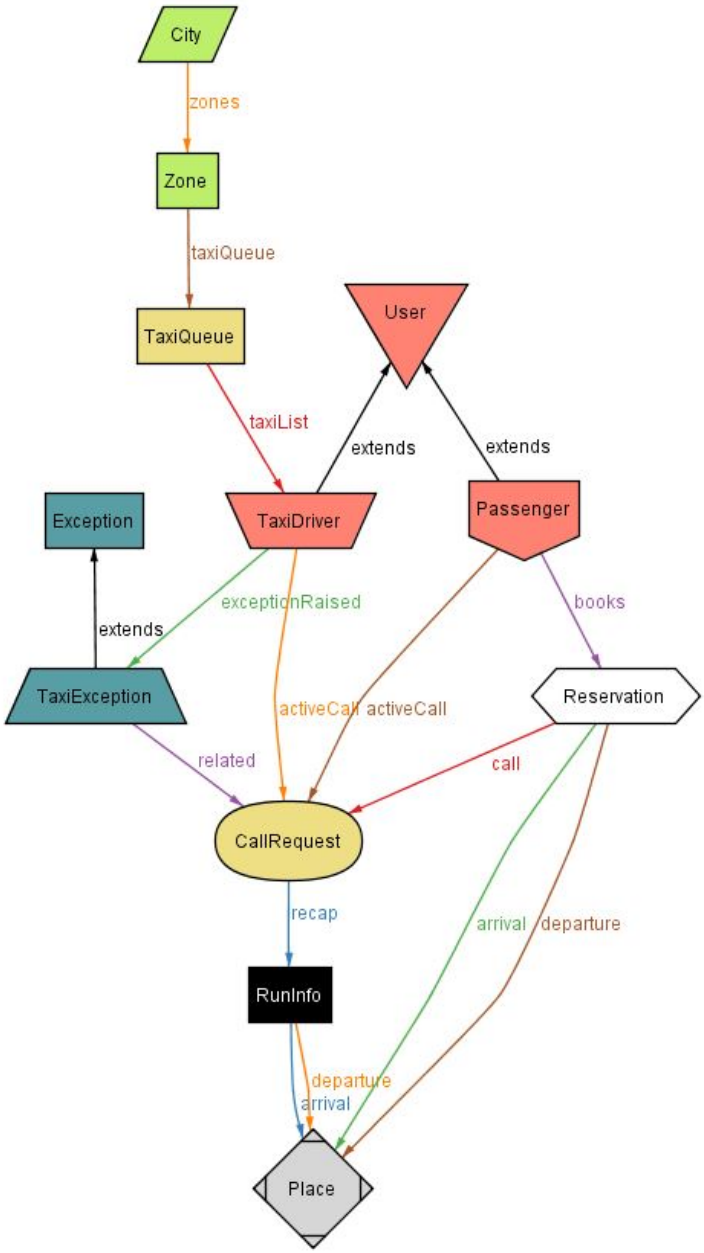
```
//--------ASSERT--------//


//no Driver with a ActiveCall in a Queue
assert noDriverQueueActive{
	all t:TaxiDriver,z:Zone | (#(t.activeCall)=1) implies t not in z.taxiQueue.taxiList
}
check  noDriverQueueActive for 10


//no more than 1 call for a Passenger
assert noMultipleCallForPassenger{
	all p:Passenger|#(p.activeCall)<2
}
check noMultipleCallForPassenger for 7


//to check add and delete TaxiException
assert checkUndoAddTaxiException{
    all t,t',t'':TaxiDriver,e:TaxiException| (e not in t.exceptionRaised and addTaxiException[t,t',e] and
         deleteTaxiException[t',t'',e]) implies t.exceptionRaised=t''.exceptionRaised
}

check  checkUndoAddTaxiException for 6

//check AddCallRequest
assert checkAddCallRequest{
	all p:Passenger,c:CallRequest| (#(p.activeCall)=0 and  addCallRequest[c,p]) implies p.activeCall=c
}


//check AddCancelRequest
assert checkCancelCallRequest{
	all p:Passenger| (#(p.activeCall)=1 and  cancelCallRequest[p]) implies #(p.activeCall)=0
}


check checkCancelCallRequest for 10


//check assignTaxiDriverToZone
assert checkassignTaxiDriverToZone{
	all t:TaxiDriver,z,z':Zone |(t not in  z.taxiQueue.taxiList) and
		assignTaxiDriverToZone[z,z',t] implies t in z'.taxiQueue.taxiList
}

check checkassignTaxiDriverToZone for 10


//check removeTaxiDriverFromZone
assert checkremoveTaxiDriverFromZone{
	all t:TaxiDriver,z,z':Zone |(t in  z.taxiQueue.taxiList) and removeTaxiDriverFromZone[z,z',t] implies
		t not in z'.taxiQueue.taxiList
}
check checkremoveTaxiDriverFromZone for 10


//check assignCallRequestToDriver
assert checkrassignCallRequestToDriver{
	all t:TaxiDriver,c:CallRequest |#(t.activeCall)=0 and assignCallRequestToDriver[t,c] implies t.activeCall=c
}
check checkrassignCallRequestToDriver for 10


//check rejectCallFromDriver
assert checkrejectCallFromDriver{
	all t:TaxiDriver,c:CallRequest |t.activeCall=c and rejectCallFromDriver[t,c] implies #(t.activeCall)=0

}

check checkrejectCallFromDriver for 10
```

4.5 Show

```
// Show

pred showPassengerRelation{

        #Passenger=1
        #TaxiDriver=1
        #Reservation=1
        #CallRequest=1
        #RunInfo=1
        #Place=1
        #Exception=0
        #City=1
        #Zone=1

}
run showPassengerRelation


pred show {

      #Passenger=8
      #TaxiDriver=6
      #Zone=5
      #CallRequest=6
      #TaxiException=2
}

run show   for 20
```

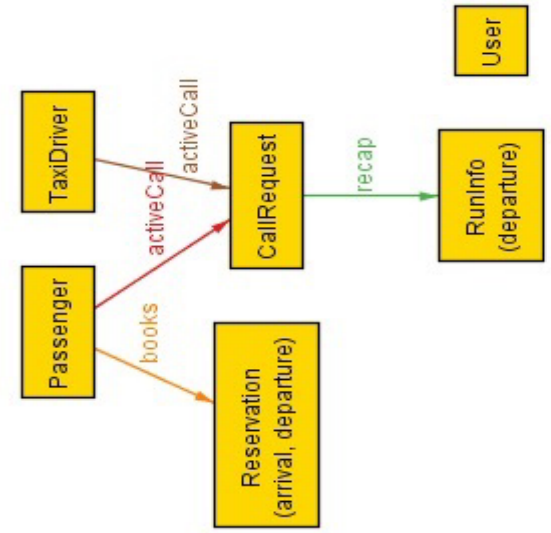
4.6 View models

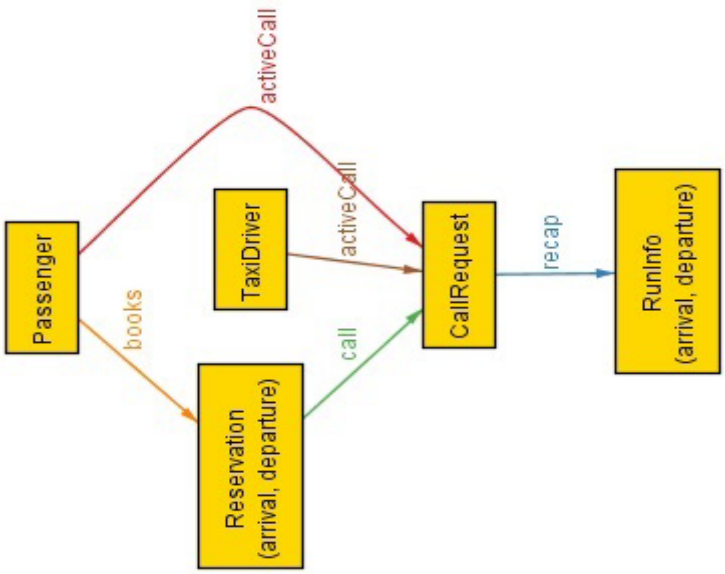
- 4.6.1 This diagram shows all entities and relations in our Alloy Model.



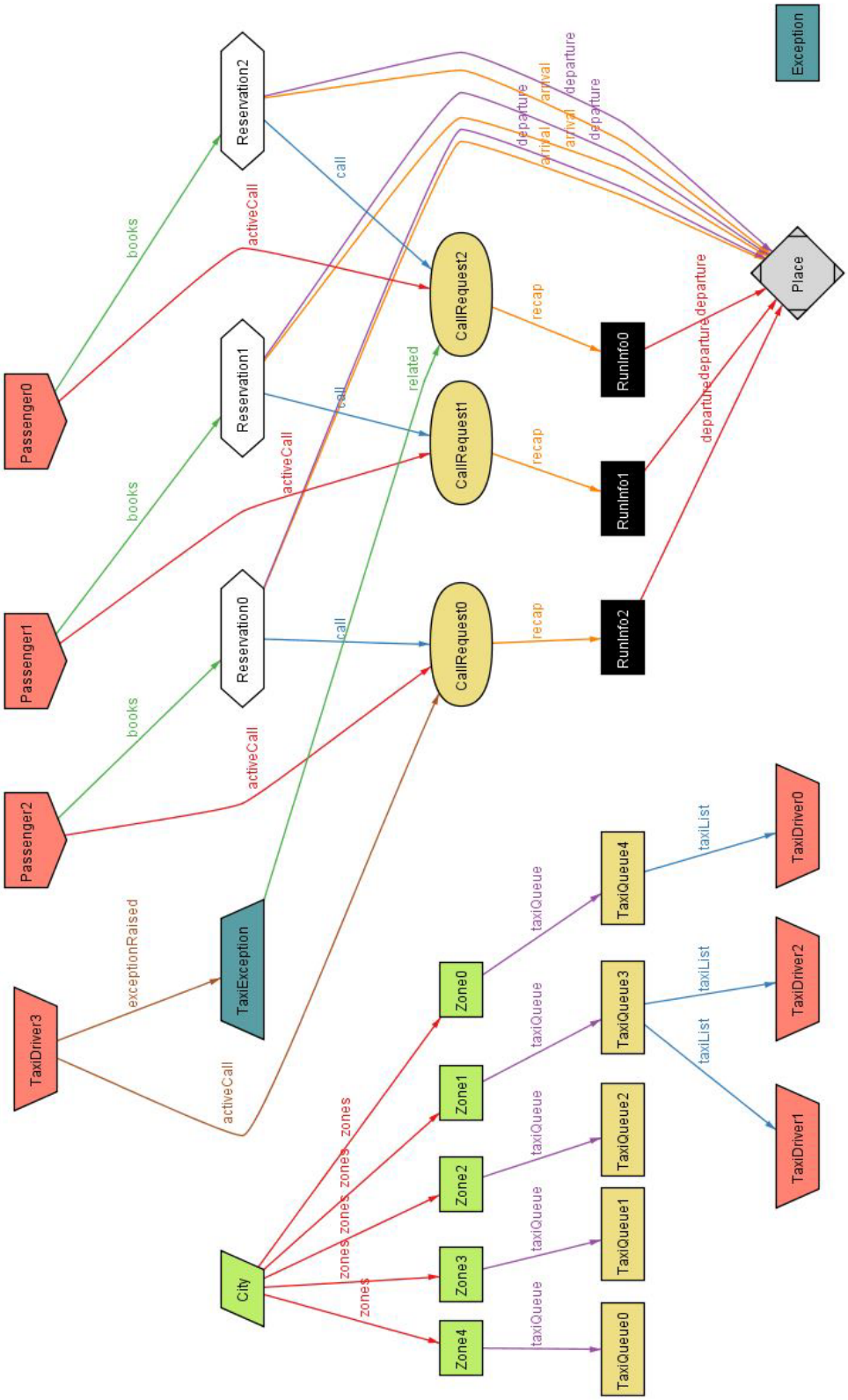
- 4.6.2 This diagram show in a clearer way how the class CallRequest are related to classes Reservation,Passenger and TaxiDriver.

  .

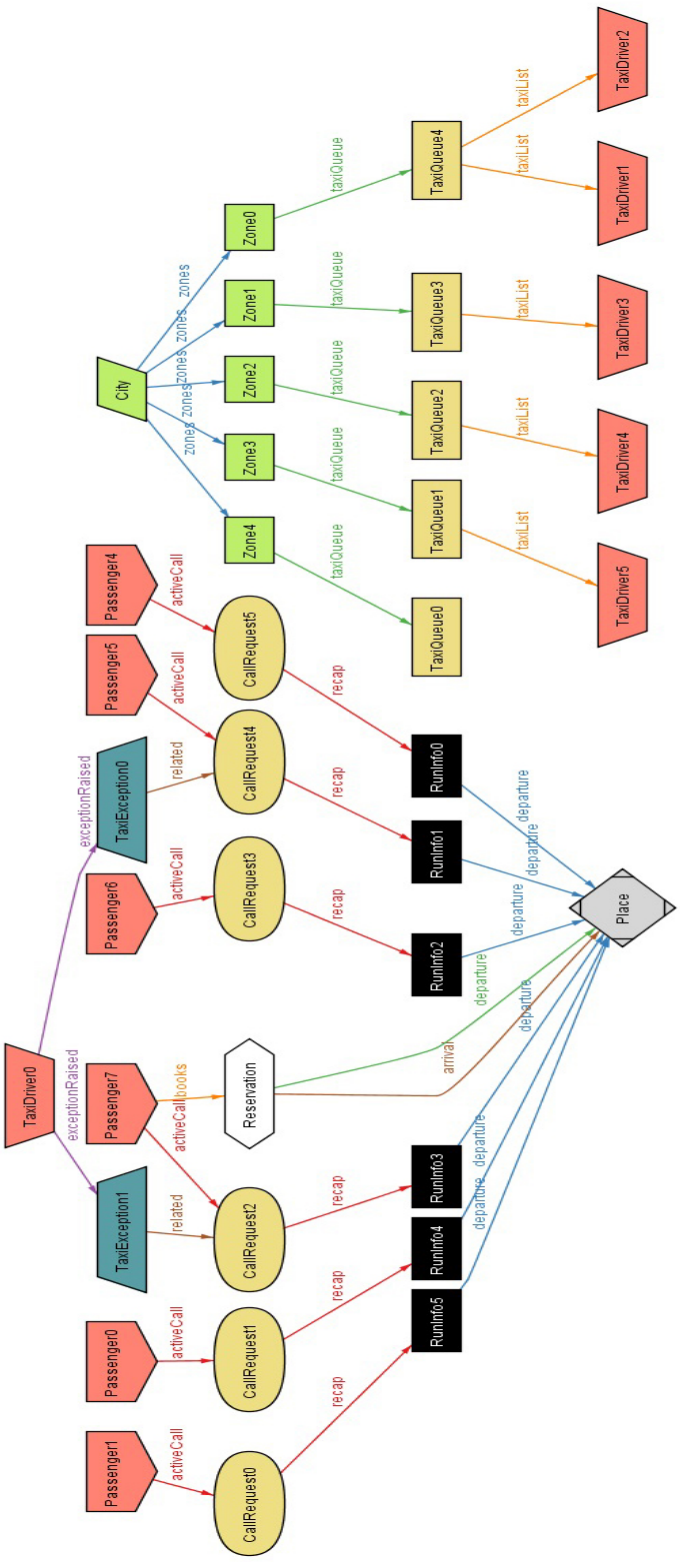




- 4.6.3 Scenario1: This is a full scenario



- 4.6.4 Scenario2: This is a full scenario



Politecnico di Milano, 6 Novembre 2015
Giovanni Brena e Andrea Canale