Politecnico di Milano
A.A. 2015-2016
Software Engineering 2: “MyTaxiService”
Requirements Analysis and Specifications Document

Mirko Basilico, Federico Dazzan

Contents

1.Introduction

1.1 Scope

We will project and implement myTaxiService, which is a software system that should optimize the availability of taxis in a given city, and guarantee a
fair management of taxi queues.
By using this service, passengers will be able to request a taxi either through a web application or a mobile app. The system answers to the request by informing the passenger about the code of the incoming taxi and the waiting time.
Taxi drivers use another mobile application to inform the system about their availability and to confirm that they are going to actually pick up a passenger.
To manage the taxi queues in a fair way the system uses a division of the city in taxi zone (approximately 2 km^2 each) . Each zone is associated to a queue of taxis. The system automatically computes the distribution of taxis in the various zones, based on the GPS information it receives from each taxi. When a taxi is available, its identifier is stored in the queue of taxis in the corresponding zone.
When a request arrives from a certain zone, the system forwards it to the first taxi queuing in that zone. If the taxi confirms, then the system will send a confirmation to the passenger. If not, then the system will forward the request to the second taxi in the queue and will, at the same time, move the first taxi in the last position in the queue.
A user can also reserve a taxi by specifying the origin and the destination of the ride. The reservation has to occur at least two hours before the ride. In this case, the system confirms the reservation to the user, and allocates a taxi to the request 10 minutes before the meeting time with the user.

1.2 Actors

Passenger guest: a passenger guest is someone who uses the web or mobile application for passengers and who hasn't signed up yet. All passenger guests can only see the login page and access to passenger's registration form.

Driver guest: a driver guest is someone who uses the mobile application for taxi drivers and who hasn't signed up yet. All driver guests can only see the login page and access to driver's registration form.

Registered passenger: this type of user, after a successful login, is the only one that can request a taxi immediately or reserve a taxi for later (at least two hours before the ride).

Registered driver: they are the taxi drivers of the city. This type of user, after successful login, can accept or refuse a passenger's request and, in the first case, he must go to the place indicated by the passenger to take him.

1.3 Goals

User should be able to:

- Register to the service
- Login to the service
- Request a taxi in an arbitrary position
- Reserve a taxi at least two hours in advance
- Receive a confirmation for the request/reservation

Tax drivers should be able to:

- Register to the service
- Login to the service
- Make themselves available in the zone they are currently in.
- Accept a request from an user

1.4 Definitions

- City: physical area of operations of our software, in this case it's the entire territory governed by the city council.
- Taxi Zone: one of the squares of approximately 2km^2 the city is divided in.
- Passenger App: the mobile or web app the users need to book a taxi.
- Driver App: the mobile or web app the drivers need to receive and acknoledge the passenger request
- Reservation: the function used by the passengers in the mobile or web application to book a taxi.
- Applicazione per tassisti: applicazione mobile utilizzata dai tassisti della città per ricevere richieste di taxi da parte dei passeggeri e dare una risposta ad esse.
- Passenger: user of the passengers' applications.
- Taxi driver: user of the taxi drivers' application.

1.5 Stakeholders

Our principal stakeholder is the government of the city, the institution that commissioned us to develop this software. It expects to receive a working software that fulfills the requests, the documentation about requirements analysis, design, development, testing, project reporting and a final presentation.
The purpose of the stakeholder is to optimize the taxi service of the city, simplifying the access of passengers to the service, and guaranteeing a fair management of taxi queues.

1.6 Reference Documents

•Specification Document: Assignments 1 and 2 (RASD and DD).pdf.
•IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.

1.7 Document overview

This document is essentially structured in three parts:
•Section 1: Introduction, it gives a description of document and some basic information about software.
•Section 2: Overall Description, gives general information about the software product with more focus about constraints and assumptions.
•Section 3: Specific Requirements, this part list requirements, typical scenarios and use cases. To give an easy way to understand all functionality
of this software, this section is filled with UML diagrams.

2. Overall Description

2.1 Product perspective

Il nostro sistema si interfaccerà con il sistema attraverso 3 differenti applicazioni, di cui 2 dedicate ai passeggeri (1 web e 1 mobile), e una dedicata ai tassisti (mobile).

2.2 User characteristics

The user that we expect to use our applications is a person who want an easy
way to access to the service we offer. Le uniche capacità che ci aspettiamo dai nostri utenti sono quelle di saper utilizzare un browser web e di saper scaricare un applicazione mobile.

2.3 Constraints

2.3.1 Regulatory Policies
mytaxyservice has to comply with all the applicable privacy laws to store the sensible user data.
the drivers need a valid taxy license issued by the local government to use mytaxiservice

2.3.2 Hardware Limitation
A device with a modern web browser to access the web application or smartphone running iOS/Android/WinPhone to access the mobile app.

2.3.3 Interfaces to other applications
mytaxiservice will provide an api to enable additional services (eg. taxi sharing)

2.3.4 Documents related
• Requirements and Analysis Specification Document (RASD).
• Design Document (DD).
• Testing report.

2.4 Assumptions

- The city is contained in a square that is divided into square zones of 2 km2.

- If no taxi is available in the passenger zone the system will keep looking in the adjacent zones until one is found.

- The same request can't be sent to the same taxy more than once.

- Every taxi driver has a valid taxi license with an ID number.
- Accurate taxi drivers' and users' position are known by GPS.

- Passengers and drivers are handled separately by the system, this means that a user registered as passenger won't be able to use the driver's application and vice versa. A user already registered as taxi driver, who wants to use the taxi service of the city as a passenger, must create a passenger account, using, if he wants, the same email address of the taxi driver account.

- To handle registration and login we use an already existing control access system, which has access to a database in order to verify the users data and taxi licenses.
  This system guarantees that:

  - ➢ The guest passengers will be able to register using their personal data and email address as user id, they will be able to choose a password of their liking and they will receive an email confirmation if the process is successful.For the process to be successful al the provided data must be valid as they will be checked by the system.

  - ➢ The guest drivers will be able to register using their personal data, license id and email address as user id, and they will receive an email confirmation if the process is successful. For the process to be successful al the provided data must be valid as they will be checked by the system.

- A reservation made by a passenger is handled like this: when there are 10 minutes left the system sends out a regular request to the taxi with the address specified in the reservation.

- After login the drivers are automatically set as "unavailable to receive taxi requests", so they need to explicitly acknowledge their availability to take requests.

- As soon as a driver is available they are added to the queue and they are ready to confirm requests

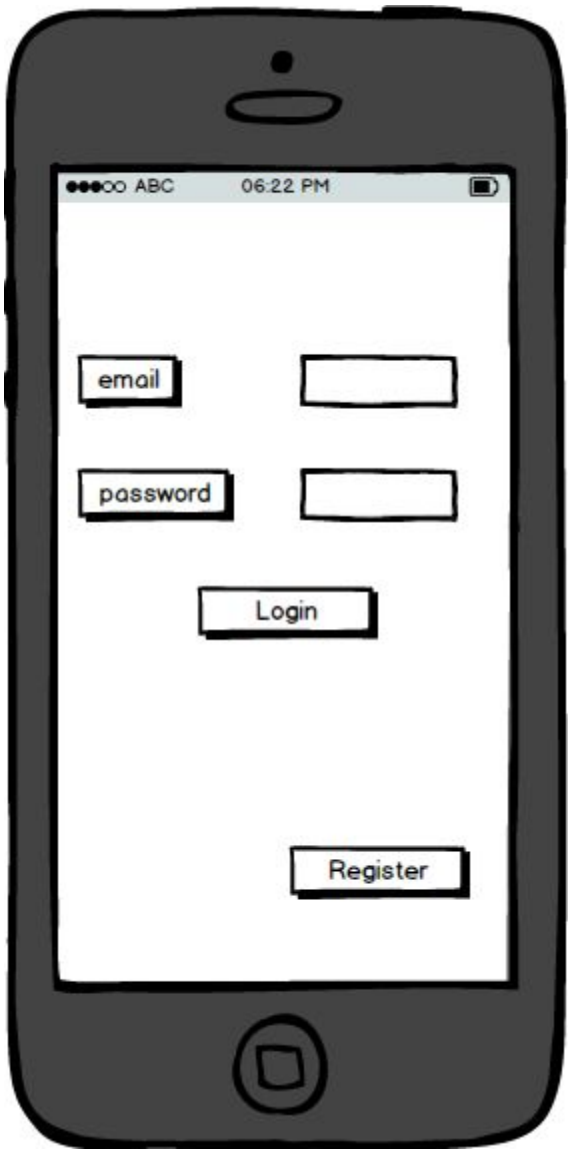
3. Specific Requirements

3.1 User interfaces

Here are presented some mockup that represent an idea of the structure of the application pages:

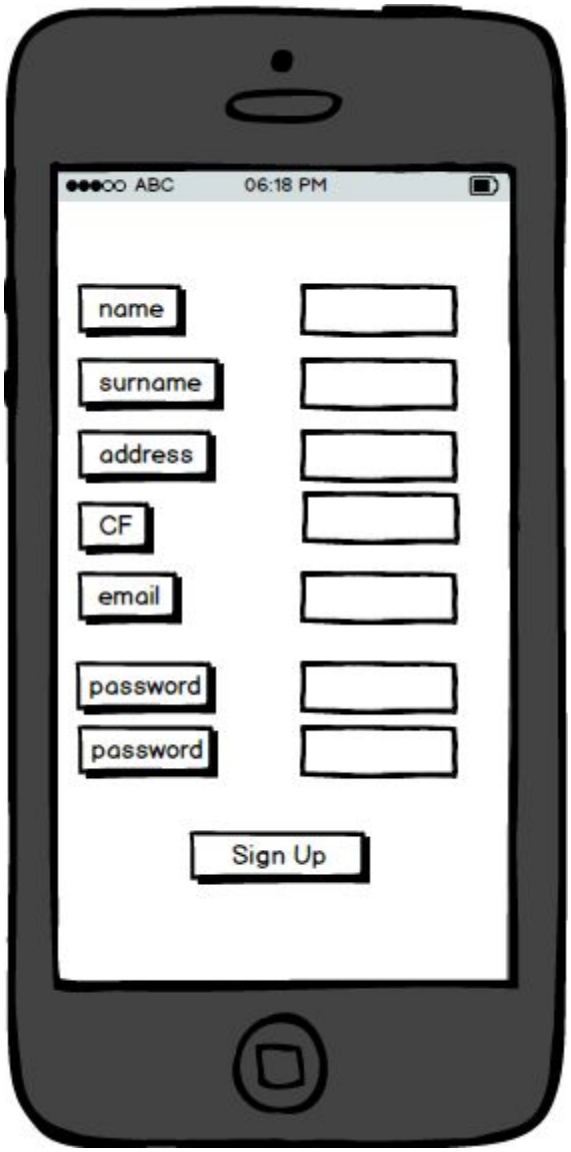
3.1.1 Login

The mockup shows the first screen of myTaxi. Here users can log in to the service and guests can access the registration form.



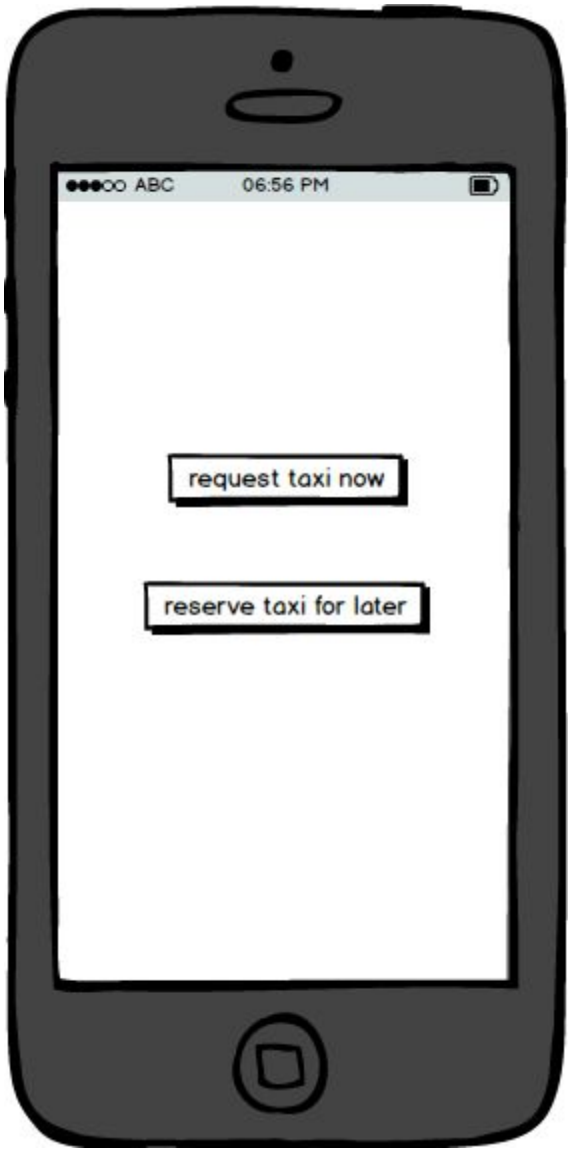
3.1.2 Passengers Registration

The mockup shows the registration screen of myTaxi for passengers. Here users can register to the service with their personal data.



3.1.3 Passengers Main Page

The mockup shows the main page of myTaxi for passengers. Here users can decide if they want a taxi immediately or if they want to reserve it for later.



3.1.4 Taxi Request

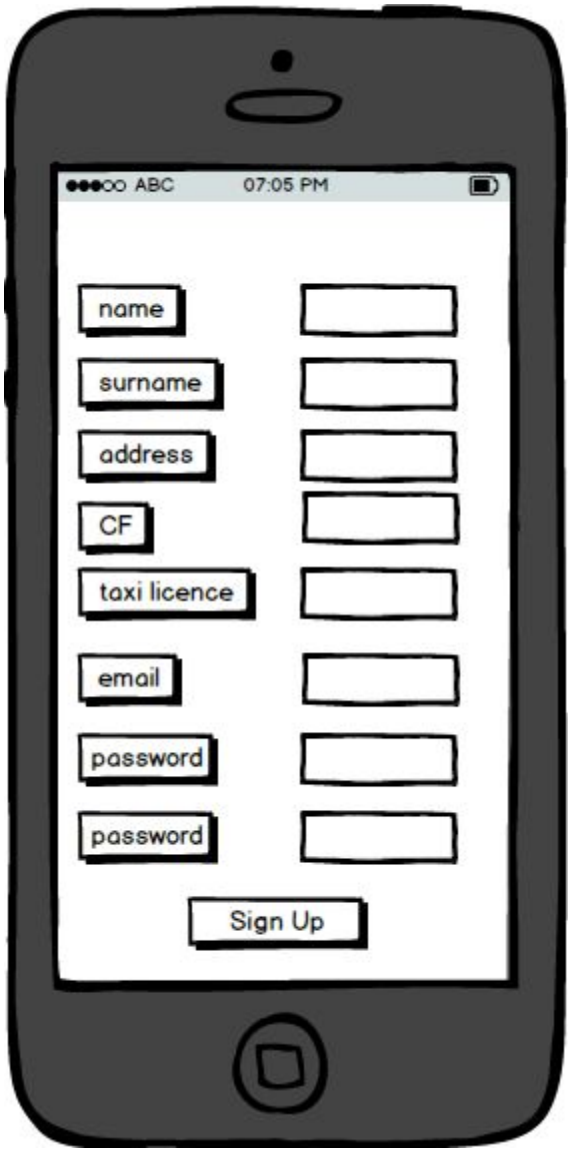
The mockup shows the request screen of myTaxi for passengers. Here users can request a taxi to a location of their liking.

3.1.5 Taxi Reservation

The mockup shows the reservation screen of myTaxi for passengers. Here users can reserve a taxi to a location of their liking.

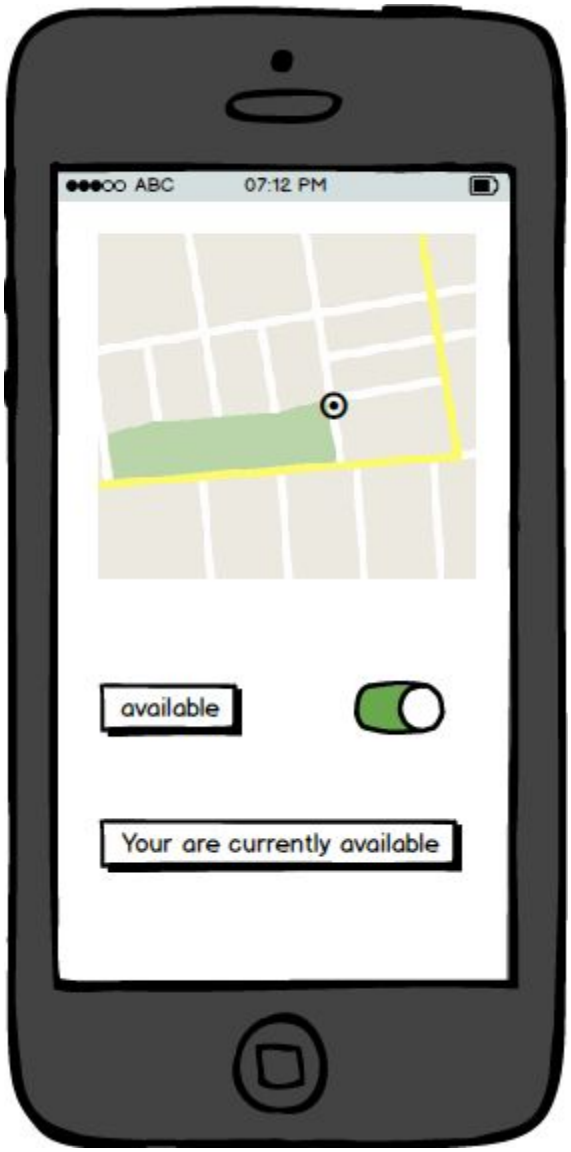
3.1.5 Driver Registration

The mockup shows the registration screen of myTaxi for drivers. Here drivers can register to the service with their personal data.



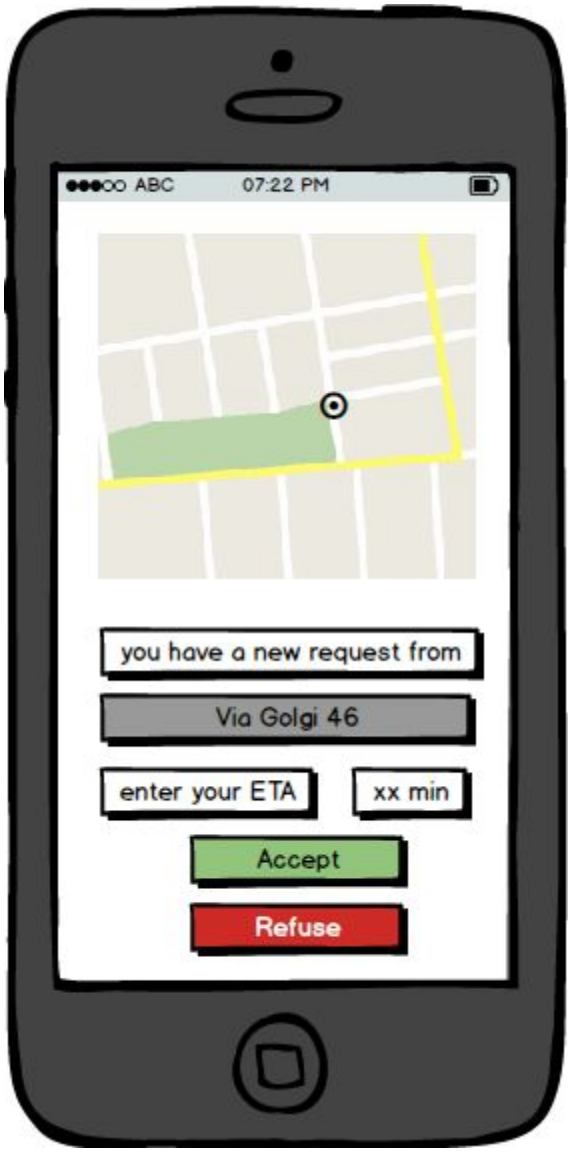
3.1.6 Driver Main Page

The mockup shows the main page screen of myTaxi for drivers. Here drivers can make themselves available to take requests.



3.1.7 Driver Request Notification

The mockup shows the request notification screen of myTaxi for drivers. Here drivers can accept or refuse requests from passengers



3.2 Functional Requirements

We indicate here the functional requirements identified for each goal:

Goal: Users should be able to register to the service

- The system must allow all guest passengers to compile the registration form

Goal: Users should be able to login to the service

- Users should be already registered to the service
- The system should allow all registered passengers to login

Goal: User should be able to request a taxi in an arbitrary position

- Users must be already registered and logged in the application;
- The system should allow the user to choose the pickup location based on the gps coordinates or an address manually entered
- if an address was manually entered it should be a valid one in the serviced area.
- After choosing the pickup location the user should send the pickup request

Goal: User should be able to reserve a taxi with a reservation made at least two hours in advance

- Passengers must be already registered and logged in the application;
- Passengers must specify the pickup time
- The pickup time should be at least two hours into to future from when the reservation it's actually made
- The system should allow the passenger to specify both the starting and arriving location
- The specified starting and arriving location should be both valid addresses in the serviced area
- After choosing the starting and arriving location the user should send the pickup request

Goal: Passengers receive a confirmation for the request/reservation

- Passengers must be already registred and logged in the application;
- The request should be already accepted by the driver
- The confirmation should include the ETA and the taxi number

Goal: Drivers should be able to register to the service

- The system must allow all guest drivers to compile the registration form

Goal: Drivers should be able to login to the service

- Drivers should be already registered to the service
- The system should allow all registered drivers to login

Goal: Drivers should be able to make themselves available in the zone they are currently in.

- Drivers must be already registered and logged in the application;
- The system should allow the driver to set the following states:
  - ❖ available to take requests
  - ❖ not available to take requests

Goal: Drivers should be able to accept a request from an user

- Drivers must be already registered and logged in the application;
- There is a request from a passenger
- The request was not accepted by another taxi
- The request is notified by the system to the first driver in the zone queue
- The request should include the pickup location
- The system should allow the driver to confirm or deny the request
- If one driver confirms the request, the same request isn't notified to another driver.

3.3 Scenarios

3.2.1 Scenario 1

Marco is a citizen of the city, already registered to MyTaxiService, who needs a taxi to go to the city center. He is chilling at home watching youtube videos on his computer, so he goes to the web site of MyTaxiService and logs in into the system. He needs a taxi immediately, so he selects the service to request a taxi as soon as it's available. He selects his current position as starting address of the taxi trip and confirms the request.

The system accepts the request and notifies it to Giuseppe, the first taxi driver in the queue associated to the city zone containing the request address. Giuseppe sees the request, but he can't accept because he must go to pick up his from school. So he refuses the request and the system notifies it to Luca, the second taxi driver in the queue. Luca accepts the request, inserts his taxi's code in the system and indicates 5 minutes as waiting time. The system notifies Marco that Luca has accepted his request, and shows him the taxi number and the waiting time.

Luca goes to the address provided by Marco and takes him to the city center.

3.2.2 Scenario 2

Patrick is a tourist visiting the city. He is planning to see the cathedral in the afternoon and the science museum in the evening. The distance between the cathedral and the museum is too long to be covered by foot, so Patrick wants to reserve a taxi. He sees on a billboard that is available an application to do this (MyTaxiService). So he downloads the version for passengers of this application on his mobile phone, and he registers himself in the system putting his personal data, an email address and a password. So he logs in into the system and he selects the reserve a taxi in a second moment.

Right now it's 5.00 pm. Patrick inserts into the system that he needs a taxi at 6.30 pm, but the system notifies him that he can only reserve a taxi at least two hours after the current time. So he inserts 7.00 pm as starting time, the cathedral's address as starting address and the museum's address as arrival address, and he confirms his reservation.

At 6.50 pm the system creates a taxi request, putting the cathedral address as starting address, and notifies it to Giuseppe, the first taxi driver in the queue associated at the city zone containing the request's address. Giuseppe accepts the request, inserts his taxi's code in the system and indicates 5 minutes as waiting time. The system notifies Patrick that Giuseppe has accepted his request, and shows him the taxi's code and the waiting time.

Giuseppe goes to the cathedral and takes Patrick to the science museum.

3.2.3 Scenario 3

Giuseppe is a taxi driver who offers his service in our city. He has just started his shift, so he opens the mobile application for taxi driver of MyTaxiService and logs in into the system. He sets his state as "available to receive taxi requests", so, the system, puts him at the end of the queue of the zone where Giuseppe is actually in. After five minutes Giuseppe is on the top of the queue, and he receives immediately a notification of taxi request at an address in his same zone. He accepts the request, inserts his taxi's code in the system and indicates 5 minutes as waiting time. So Giuseppe goes to the address indicated in the request, takes the passenger who ask Giuseppe to take him at the airport. Once arrived at the airport, the passenger pays Giuseppe and leaves the taxi. After this ride, Giuseppe is very tired, so he decides to take a coffee in the airport's bar. To avoid receiving taxi requests while he is at the bar, he opens the mobile application for taxi driver of MyTaxiService and sets his state as "unavailable to receive taxi requests".

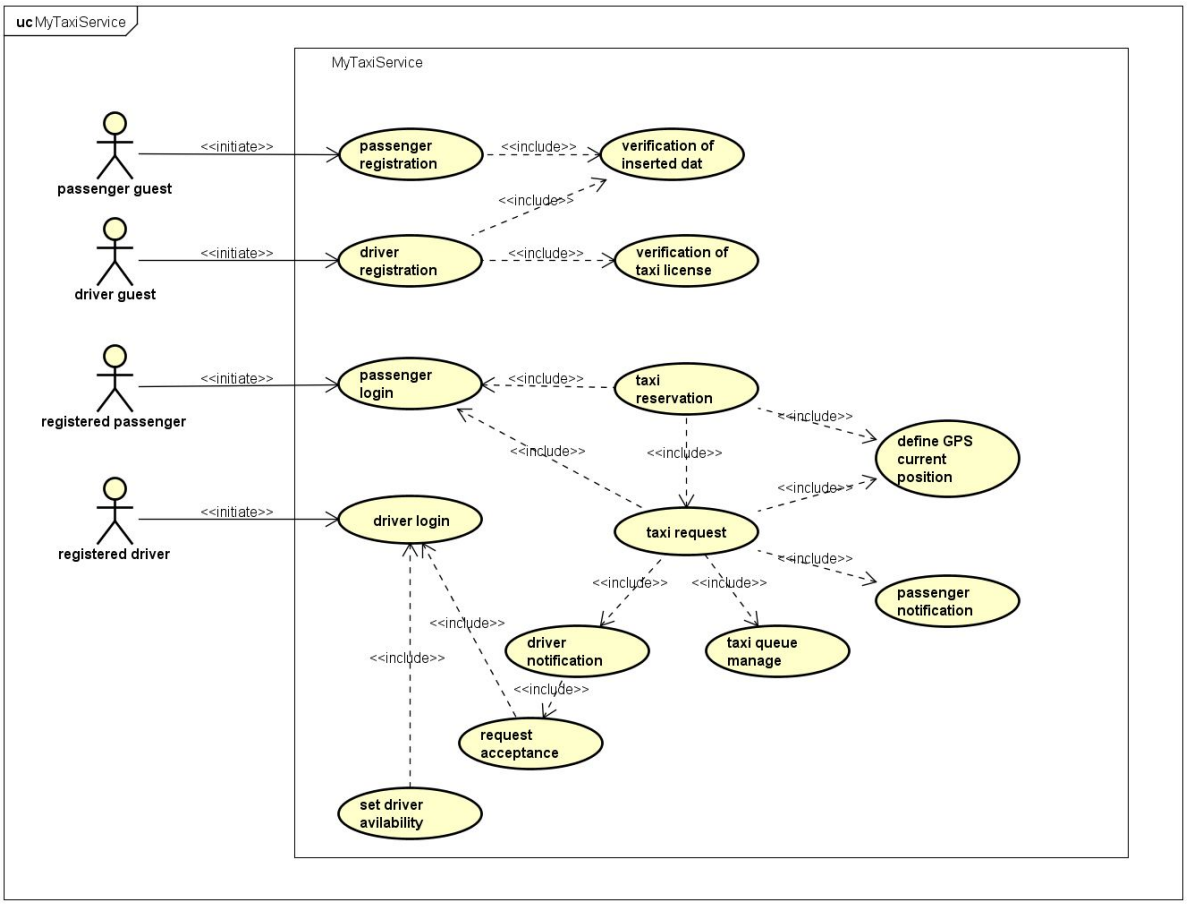
3.4 UML models

3.4.1 use cases

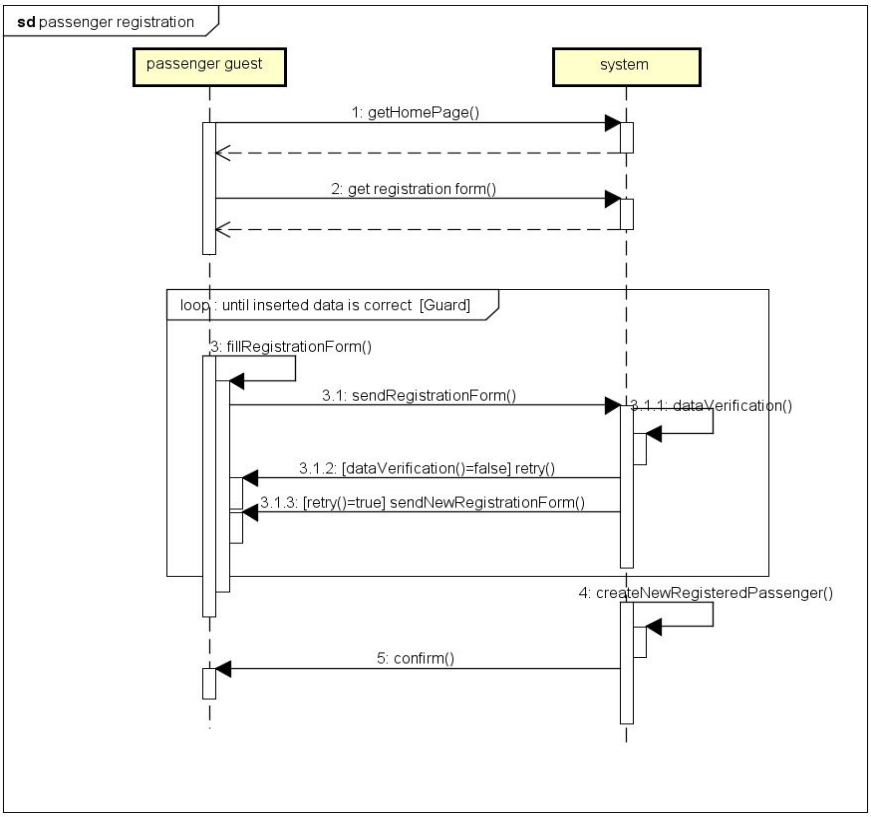
We can derive some use cases from the scenarios identified in the previous paragraph:

- passenger registration;
- driver registration;
- passenger login;
- driver login;
- taxi request;
- taxi reservation;
- taxi driver's setting availability.

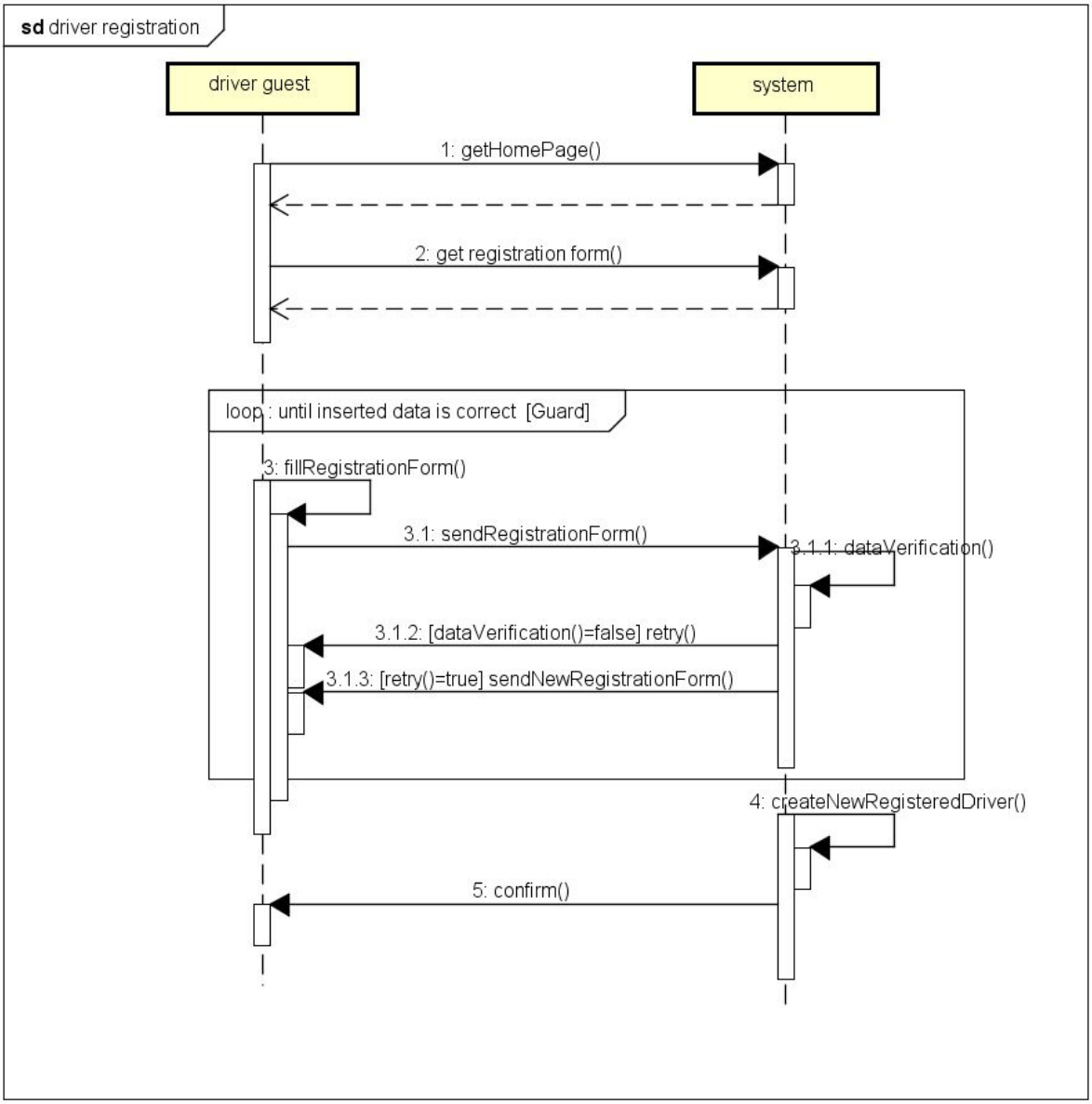
Here it is a Representation of the Use Case Diagram:



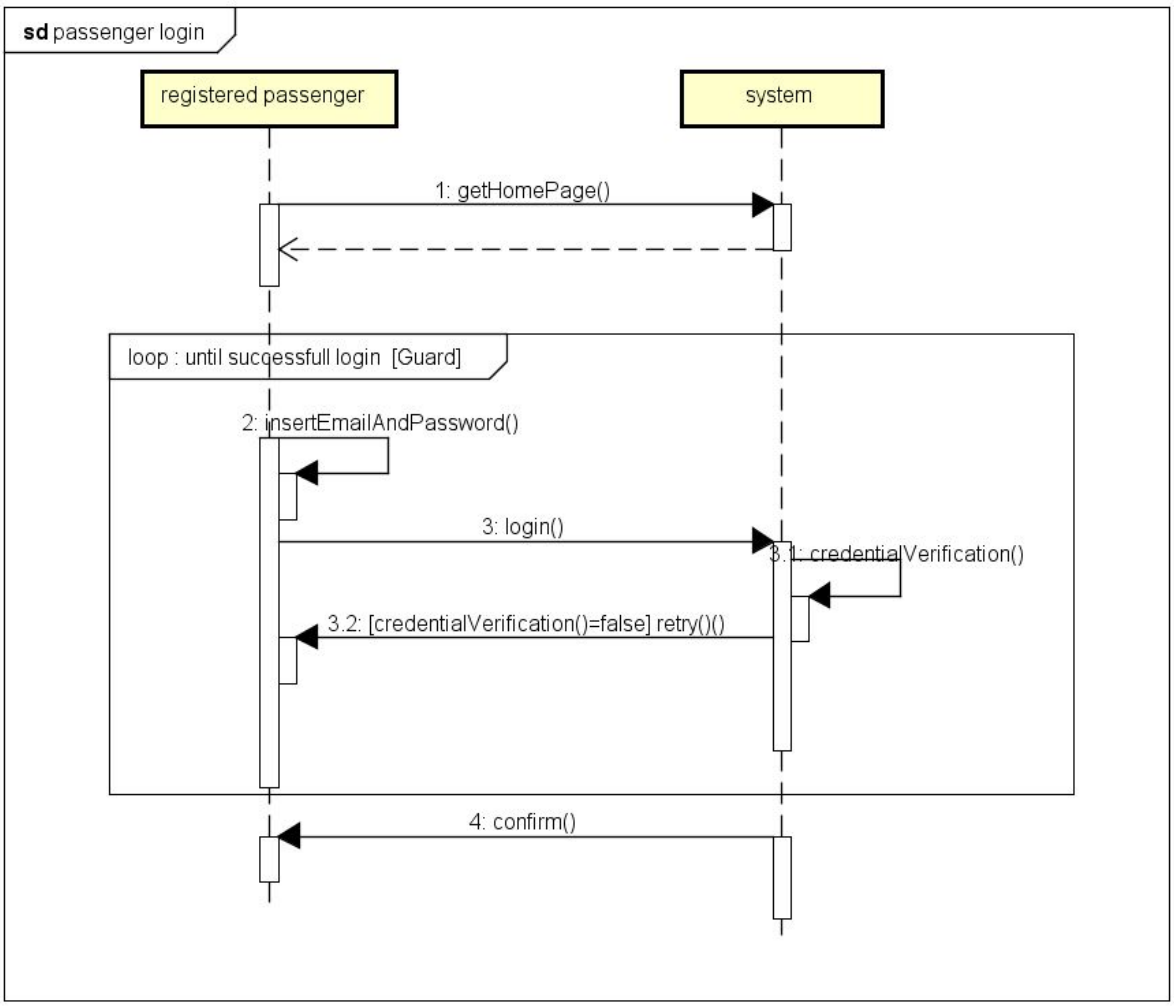
Name	Passenger registration
Actor	passenger guest
Entry condition	passenger guest isn't registered
Flow of events	1. passenger guest opens the web application or the mobile app for passengers of MyTaxiService; 2. passenger guest clicks on the "registration" button; 3. passenger guest inserts into the registration form his email address, the mandatory personal data and picks a password 4. passenger guest confirms the registration; 5. the system confirms the successful registration
Exit condition	registration successfully done
exceptions	● Passenger guest is already registered; ● One or more mandatory fields are not valid; ● Email inserted is already associated to another user; All exception are handled alerting the visitor of the problem and application goes back to point 2 of EventFlow



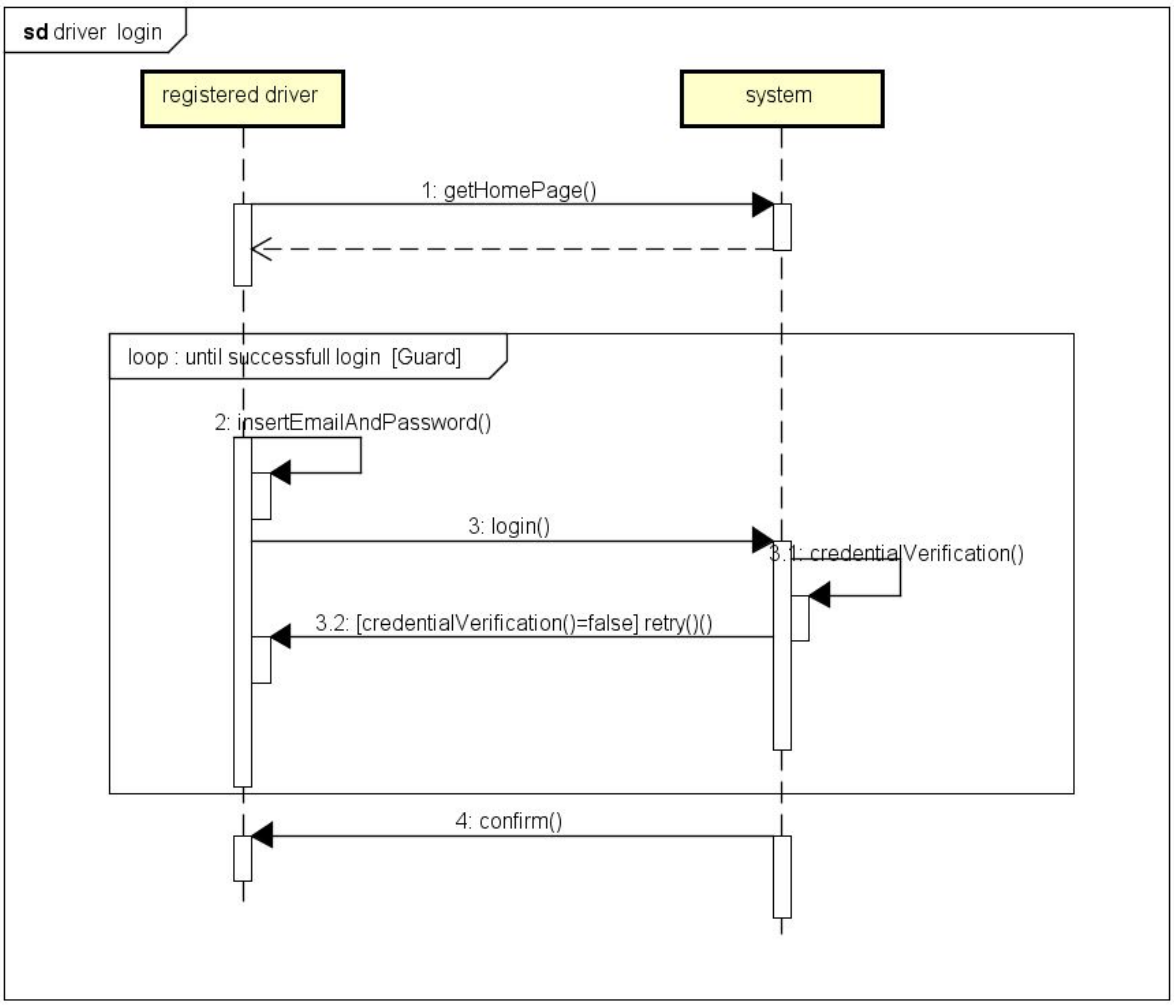
Name	driver registration
Actor	driver guest
Entry condition	driver guest isn't registered
Flow of events	1. driver guest opens the mobile app for driver of MyTaxiService; 2. driver guest clicks on the "registration" button; 3. driver guest inserts into the registration form his email address, taxi license number, the mandatory personal data and picks a password 4. the system check if the taxi licence number corresponds to the inserted personal datas; 5. driver guest confirms the registration; 6. the system confirms the successful registration
Exit condition	registration successfully done
exceptions	● driver guest is already registered; ● One or more mandatory fields are not valid; ● Email inserted is already associated to another user; ● taxi license's code is already associated to another user or it doesn't correspond to the inserted personal data; All exception are handled alerting the visitor of the problem and application goes back to point 2 of EventFlow


sd driver registration
driver guest
system
1: getHomePage()
2: get registration form()
loop : until inserted data is correct [Guard]
3: fillRegistrationForm()
3.1: sendRegistrationForm()
3.1.1: dataVerification()
3.1.2: [dataVerification()=false] retry()
3.1.3: [retry()=true] sendNewRegistrationForm()
4: createNewRegisteredDriver()
5: confirm()

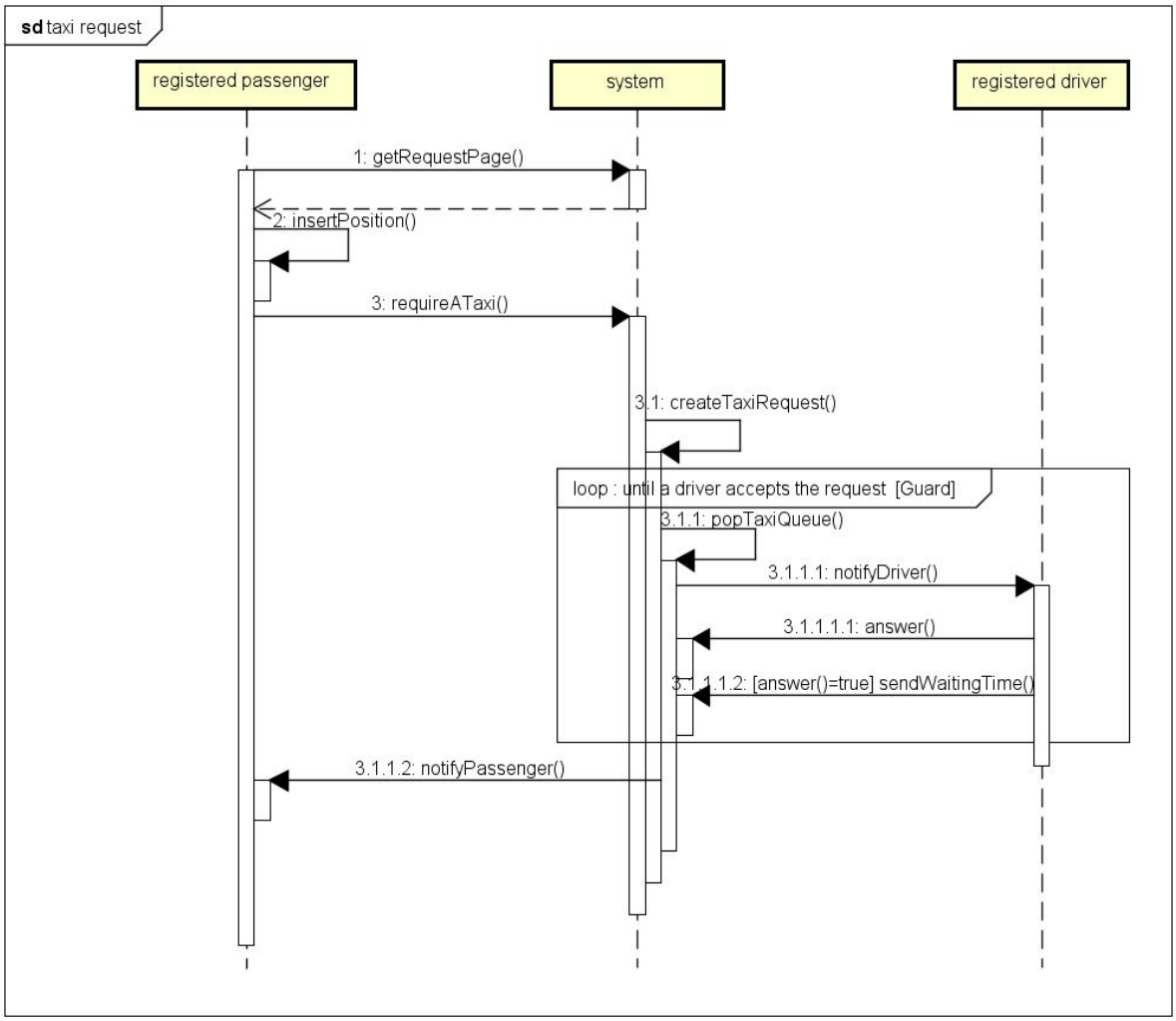
Name	passenger login
Actor	registered passenger;
Entry condition	registered passenger is already registered into the system
Flow of events	1. registered passenger opens the web application or the mobile app for passenger of MyTaxiService; 2. registered passenger inserts his email address and password in the appropriate fields; 3. registered passenger clicks on “login” button; 4. registered passenger is promoted to registered passenger;
Exit condition	the system shows the passenger’s main page.
exceptions	If username and/or password are incorrect the system notifies it to registered passenger and doesn’t allow the login.



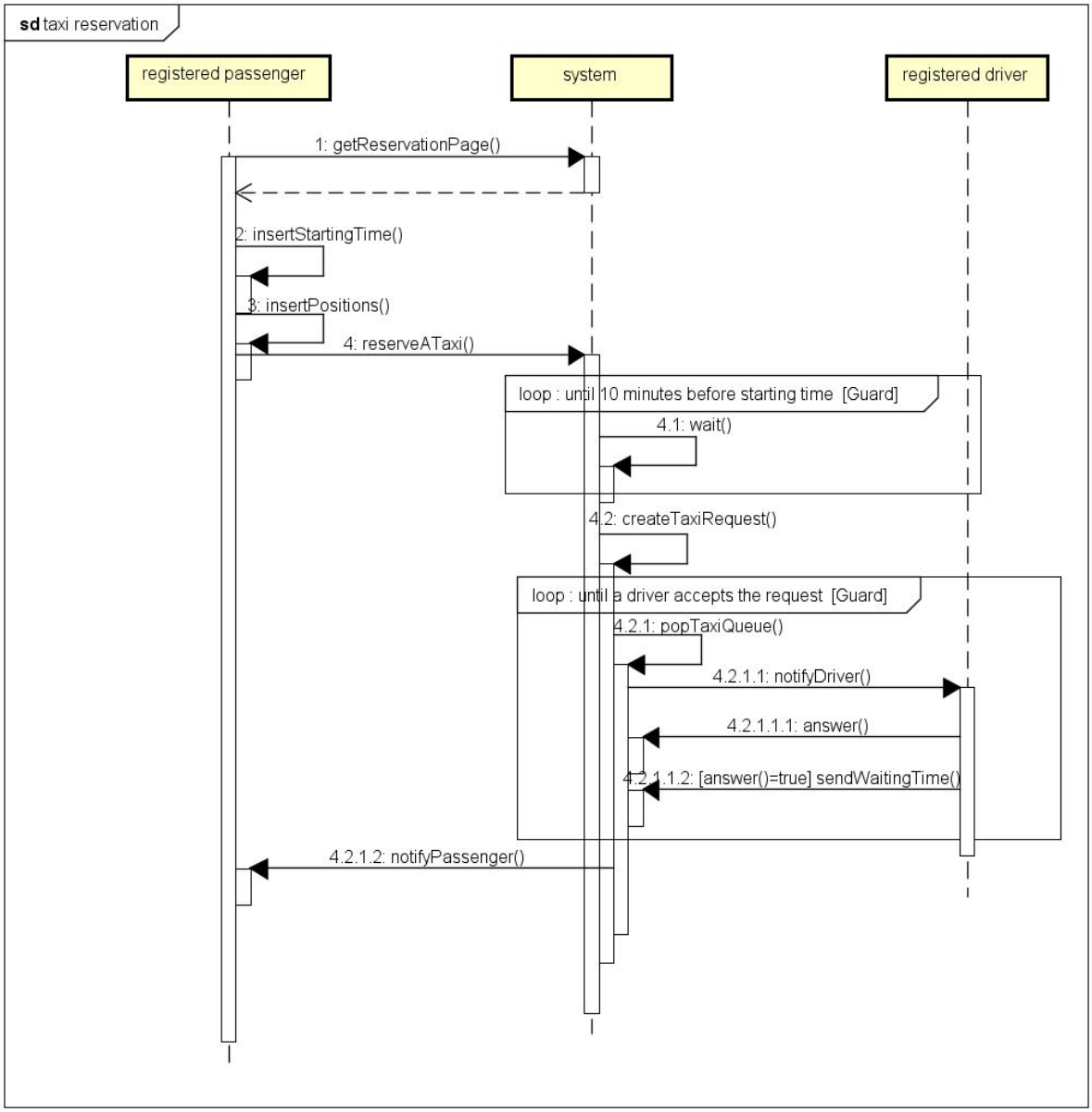
Name	driver login
Actor	registered driver;
Entry condition	registered driver is already registered into the system
Flow of events	1. registered driver opens the mobile app for driver of MyTaxiService; 2. registered driver inserts his email address and password in the appropriate fields; 3. registered driver clicks on “login” button; 4. registered driver is promoted to registered driver;
Exit condition	the system shows the driver’s main page.
exceptions	If username and/or password are incorrect the system notifies it to registered driver and doesn’t allow the login.



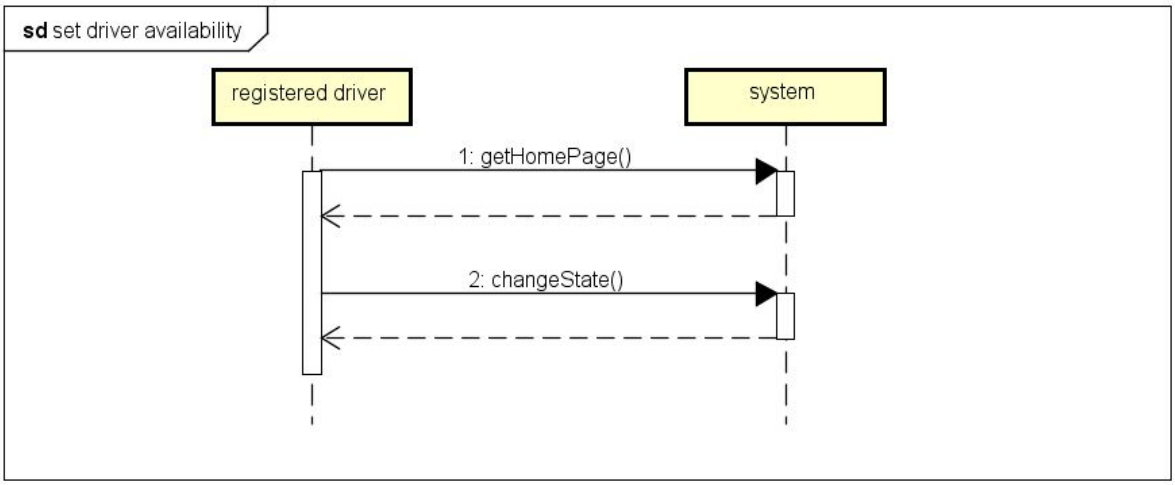
Name	taxi request
Actor	● registered passenger; ● registered driver;
Entry condition	registered passenger select “request taxi now” button in the passenger’s main page of the passenger’s app.
Flow of events	1. registered passenger insert the starting position manually or selecting his current position; 2. registered passenger confirms the request clicking on “request taxi” button; 3. the system checks if the inserted address is valid; 4. the system notifies the taxi request to the first taxi in the queue of the zone containing the address indicated by registered passenger; 5. registered driver who receives the notification accepts the registered passenger’s taxi request; 6. registered driver inserts an estimated arrival time
Exit condition	the system notifies to the registered passenger that a driver has accepted his request, showing him the incoming taxi code and the waiting time
exceptions	● the inserted address is not valid or is not contained in the city. In this case the system notifies that to registered passenger and goes back to point 1 of EventFlow. ● there is no taxi in the zone’s queue. In this case the system searches for other taxi in the adjacent zones until one is found. ● if a registered taxi refuses a taxi request the system notifies it to the next taxi in the queue, until a registered taxi has accepted the request.


sd taxi request
registered passenger
system
registered driver
1: getRequestPage()
2: insertPosition()
3: requireATaxi()
3.1: createTaxiRequest()
loop : until a driver accepts the request [Guard]
3.1.1: popTaxiQueue()
3.1.1.1: notifyDriver()
3.1.1.1.1: answer()
3.1.1.2: [answer()=true] sendWaitingTime()
3.1.1.2: notifyPassenger()

Name	taxi reservation
Actor	● registered passenger; ● registered driver;
Entry condition	registered passenger selects “reserve a taxi for later” button in the passenger main page of the passenger’s app.
Flow of events	1. registered passenger inserts the time he wants to reserve a taxi; 2. registered passenger inserts the starting and the arrival positions manually or selecting his current position; 3. registered passenger confirms the request clicking on “request taxi” button; 4. the system checks if the inserted addresses are valid; 5. 10 minutes before the time indicated by the registered passenger, the system automatically creates a taxi request, putting the starting address indicated by the registered user as starting address; 6. the system notifies the taxi request to the first taxi in the queue of the zone containing the address indicated in the taxi request; 7. registered driver who receives the notification accepts the registered passenger’s taxi request; 8. registered driver inserts his taxi number and an estimated arrival time
Exit condition	the system notifies to the registered passenger that a driver has accepted his request, showing him the incoming taxi number and the waiting time
exceptions	● the inserted hour isn’t at least two hours after the current time. In this case the system notifies that to registered passenger and goes back to point 1 of EventFlow. ● one of the inserted addresses is not valid or is not contained in the city. In this case the system notifies that to registered passenger and goes back to point 2 of EventFlow. ● there is no taxi in the zone’s queue. In this case the system search for other taxi in the adjacent zones until one is found. ● if a registered taxi refuses a taxi request the system notifies it to the next taxi in the queue, until a registered taxi has accepted the request.

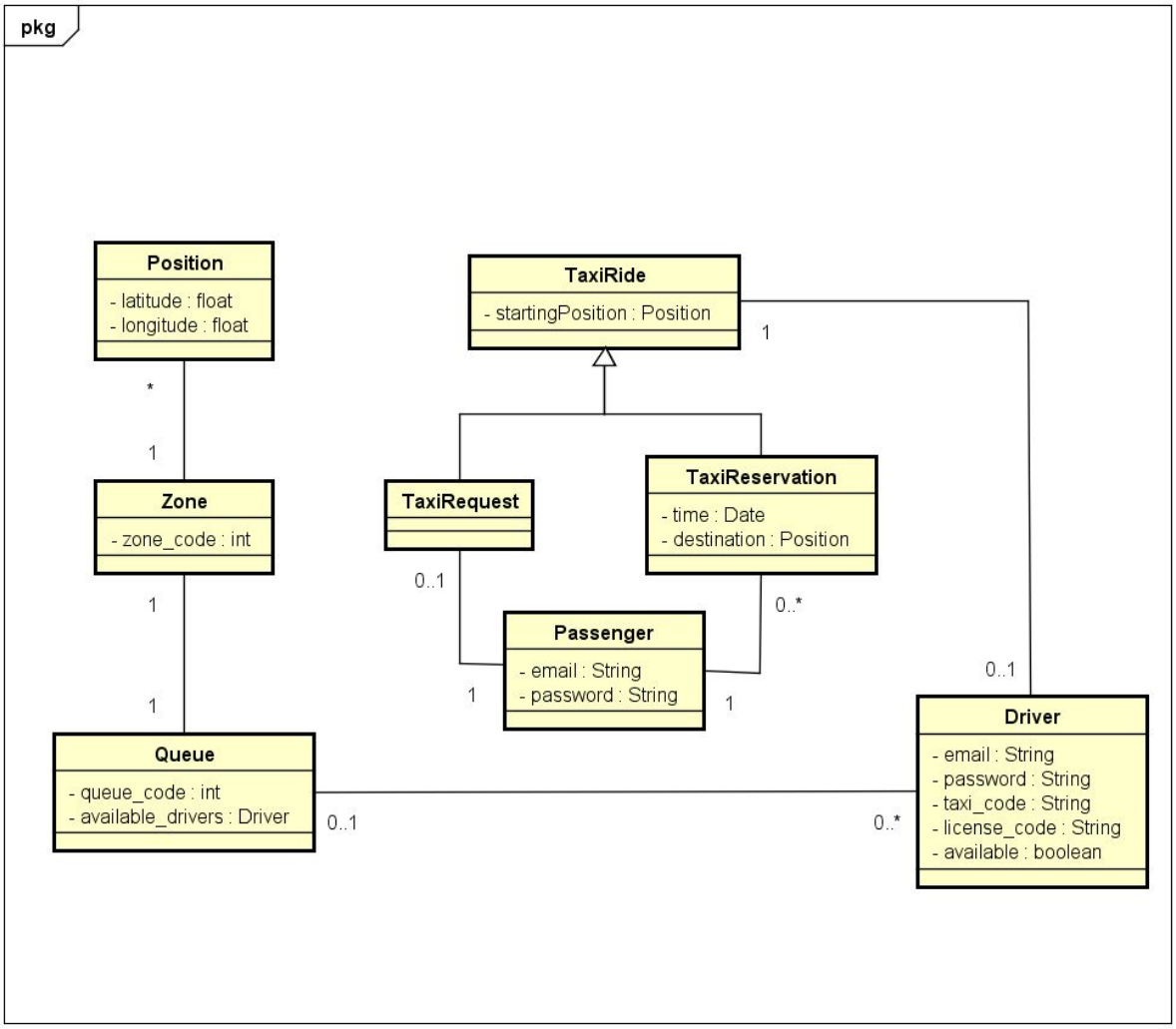

sd taxi reservation
registered passenger
system
registered driver
1: getReservationPage()
2: insertStartingTime()
3: insertPositions()
4: reserveATaxi()
loop : until 10 minutes before starting time [Guard]
4.1: wait()
4.2: createTaxiRequest()
loop : until a driver accepts the request [Guard]
4.2.1: popTaxiQueue()
4.2.1.1: notifyDriver()
4.2.1.1.1: answer()
4.2.1.1.2: [answer()=true] sendWaitingTime()
4.2.1.2: notifyPassenger()

Name	taxi driver's setting availability
Actor	registered driver
Entry condition	Registered driver has logged in into the mobile application for taxi driver and is into the driver's main page of the app
Flow of events	1. registered drive clicks on "available" switch botton, to indicate his desired status of availability.
Exit condition	Registered driver has set his desired status of availability
exceptions	



3.4.2 class diagram

Here is presented the class diagram. This diagram will be updated during the developing process expecially adding all method.



4. Alloy

4.1 Modeling

```
//Signatures
open util/boolean
sig string{}
sig float{}
sig Driver{
        email: one string,
        password: one string,
        taxi_code: one string,
        licence_code: one string,
        available: one Bool
}

sig Passenger{
        email: one string,
        password: one string,
}

abstract sig TaxiRide{
 starting_position: one string,
 has: one Passenger,
 assigned_to: lone Driver
}

sig TaxiRequest extends TaxiRide{}

sig TaxiReservation extends TaxiRide{
        time: one Int,
        destination: one string
}

sig Position{
        latitude: one float,
        longitude: one float
}

sig Zone{
        zone_code: one Int,
```

```
        queue: one Queue,
        edge: some Position
}{ #edge=4
}

sig Queue{
queue_code: one Int,
available_drivers: set Driver
}




//Facts
fact Location{
        //Two Position can't have the same latitude and longitude
        no disj l1,l2:Position | l1.latitude=l2.latitude and l1.longitude=l2.longitude
}

fact Zone{
// Two or more Zone can't have the same idzone
        no disj z1,z2:Zone | z1.zone_code=z2.zone_code
//Two or more Zone can't be associated to the same queue
        no disj z1,z2:Zone | z1.queue=z2.queue
//Two ore more Zone can't have the same edge
        all disj z1,z2:Zone | z1.edge!=z2.edge
}

fact Queue{
//Two or more queue can't have the same queue code
        no disj q1,q2:Queue | q1.queue_code=q2.queue_code
//One queue is associated to one zone
        all q:Queue | one z:Zone | q in z.queue
}

fact TaxiDriver{
//Two or more Driver can't have the same email
        no disj t1,t2:Driver | t1.email=t2.email
//Two or more Driver can't have the same licence code
        no disj t1,t2:Driver | t1.licence_code=t2.licence_code
//Two or more Driver can't have the same username
```

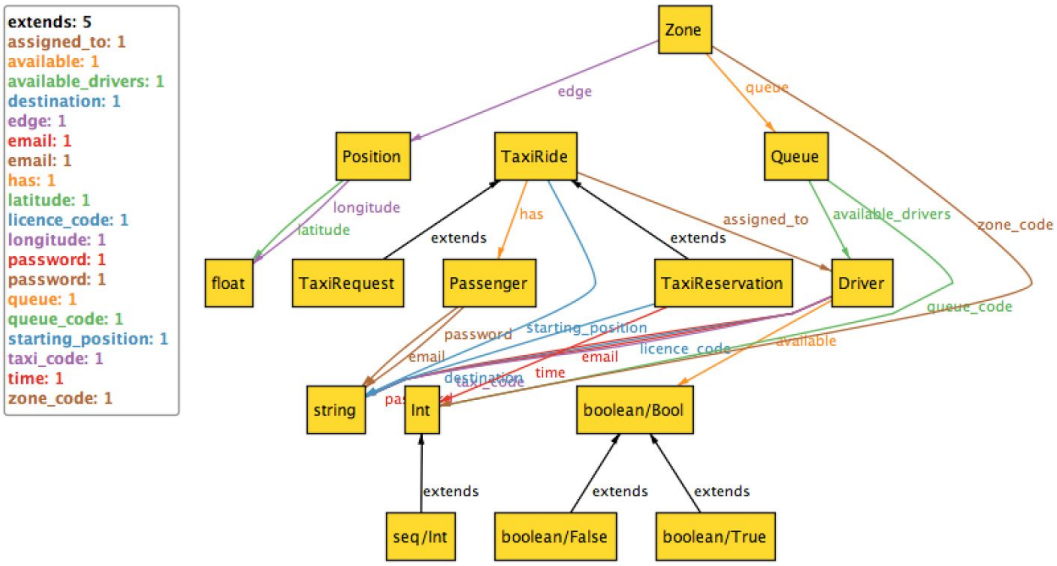
```
        all disj t1,t2:Driver | t1.taxi_code!=t2.taxi_code
//One Driver can be at most in one queue at the same time
no t:Driver | some disj q1,q2: Queue | t in q1.available_drivers and t in q2.available_drivers
//If one Driver is assigned to a request, he must not belong to any queue
all t:Driver,r:TaxiRide,q:Queue | r.assigned_to=t implies t not in q.available_drivers
//A Driver can serve at most one request at the same time
no disj r1,r2:TaxiRide | r1.assigned_to=r2.assigned_to and r1.assigned_to!=none
//If a Driver has an assigned ride, they are not available
all t:Driver,r:TaxiRide | t in r.assigned_to implies t.available=False
//If a Driver is in a queue, they are available
all t:Driver,q:Queue | t in q.available_drivers implies t.available=True
}

fact Passenger{
//Two or more Passenger can't have the same email
no disj p1,p2:Passenger | p1.email=p2.email
//One Passenger can't have two rides with same time
all disj r1,r2:TaxiRide | r1.has=r2.has implies r1.time!=r2.time
//One Passenger can have only one outstanding request
no disj r1,r2:TaxiRequest | r1.has = r2.has
}

fact TaxiRide{
//ride must have a destination different from the starting point
all disj r1,r2:TaxiRide | r1=r2 implies r1.starting_position!=r2.destination
//A passenger can have only one ride assigned at the same time
all disj r1,r2:TaxiRide | (r1.has=r2.has and r1.assigned_to!=none) implies
r2.assigned_to=none
//If a Passenger has one Request they can't have a Reservation assigned
all r1:TaxiRequest,r2:TaxiReservation | r1.has=r2.has implies r2.assigned_to=none
}


//Assertions and Predicates


assert DriverAssignedToTwoOrMoreQueue{
no t:Driver | some disj q1,q2: Queue | t in q1.available_drivers and t in q2.available_drivers
}
check DriverAssignedToTwoOrMoreQueue

assert PassengerWithTwoOustandingRides{
```

```
no disj r1,r2:TaxiRide | r1.has=r2.has and r1.assigned_to!=none and r2.assigned_to!=none
}
check PassengerWithTwoOustandingRides

pred PassengerMoreRide{
#Queue>1
}
run Passenger MoreRide for 5 but 2 Passenger, 10 Driver


pred MoreRide{
#TaxiRide>6
#Passenger=3
}
run MoreRide for 5 but 10 TaxiRide
pred GenericWord {
#Queue>1
#TaxiRide>1
some q:Queue | q.available_drivers!=none
}
run GenericWord for 10
```

4.2 Analyzer

```
Executing "Check DriverAssignedToTwoOrMoreQueue"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  3248 vars. 321 primary vars. 5496 clauses. 17ms.
  No counterexample found. Assertion may be valid. 2ms.

Executing "Check PassengerWithTwoOustandingRides"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  3254 vars. 318 primary vars. 5547 clauses. 17ms.
  No counterexample found. Assertion may be valid. 2ms.

Executing "Run PassengerMoreRide for 5 but 2 Passenger, 10 Driver"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  11493 vars. 787 primary vars. 19636 clauses. 44ms.
  Instance found. Predicate is consistent. 33ms.

Executing "Run MoreRide for 5 but 10 TaxiRide"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  14252 vars. 860 primary vars. 22852 clauses. 82ms.
  Instance found. Predicate is consistent. 35ms.

Executing "Run GenericWord for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  38527 vars. 2100 primary vars. 66176 clauses. 140ms.
  Instance found. Predicate is consistent. 98ms.

5 commands were executed. The results are:
  #1: No counterexample found. DriverAssignedToTwoOrMoreQueue may be valid.
  #2: No counterexample found. PassengerWithTwoOustandingRides may be valid.
  #3: Instance found. PassengerMoreRide is consistent.
  #4: Instance found. MoreRide is consistent.
  #5: Instance found. GenericWord is consistent.
```

4.3 Worlds Generated



5. Appendix

5.1 Software and tools used

- Google Docs: to redact and to format this document.
- Astah Professional (http://astah.net/editions/professional): 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.
- Balsamiq Mockups(http://balsamiq.com/products/mockups/): to create mockups.

5.2 Hours of works

This is the time we spent to redact this document:
• Mirko Basilico: ~37 hours.
• Federico Dazzan: ~37 hours.