MyTaxiService

Requirement Analysis and Specification Document

Software Engineering 2 Project AA 2015-2016

Elis Bardhi **790135**

Andrea Cavalli **841512**

Mario Dolci **773705**

1. Introduction

1.1. Description of given problem

We will project MyTaxiService, which is an online service used by people in order to take a taxi in a large city. The system that will be projected has to allow the registration of new users; each one can request a taxi through a web application or a mobile application.
Every user can create, modify and delete a taxi request, and can also share a taxi with other people in order to divide the cost of the ride.
After an user request, the taxi driver receives a notification from the system and can accept or decline the request. The system automatically arranges the requests dividing the city in taxi zones 2 km^2 big, each one associated to a taxi queue. If a taxi driver accepts the request, the system will send a notification to the user, otherwise it will forward it to the next taxi in the queue until a taxi driver accept the request.

1.2. Goals

MyTaxiService has to provide these main functionalities:

[G1] Registration of a person into the system;
[G2] Visualization of the personal homepage;
[G3] Allowing clients to request a taxi;
[G4] Allowing clients to reserve a taxi;
[G5] Allowing clients to share a taxi;
[G6] Allowing clients to edit a taxi request/reservation;
[G7] Allowing taxi drivers to accept or refuse a request/reservation.
[G8] Calculation of the costs done by the system;
[G9] Calculation of the routes done by the system;
[G10] Fair management of taxi queues;
[G11] Allowing clients to choose a shared ride;
[G12] Allowing clients to see the code and the ETA of the incoming taxi.

2. Overall Description

2.1. Domain Properties

We suppose some conditions that must take place in analyzed world:

- Requests created in MyTaxiService actually take place;
- The places specified on taxi requests must be real and must be reachable using a car;
- Clients can invite to their taxi sharing requests only people registered in MyTaxiService.

2.2. Glossary

We have to define some key words that will be used in our documents.

Abbreviations and definitions	Description
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair
EWT	Estimated Waiting Time
ETA	Estimated Time of Arrival
GPS	Global Positioning System - uses satellites to triangulate positions on earth.
Taxi Request	An event that happens when a client asks for a taxi service. There are two types of taxi requests: standard request, when a client asks for a taxi ride for a maximum of four people, and a taxi sharing request when a client asks to share a ride with other clients.
Guest	A person who has not signed up yet
Client	A client is an authenticated person who is capable of reserving a ride from a certain location to another. He is also able to share a ride with other passengers or to look for shared rides.
Taxi Driver	A person who has signed up into the system. Every taxi driver has only one taxi which is characterized by an identifier code. Taxi drivers can visualize the clients' requests and decide if to accept or decline it.

2.3. Assumptions

There are some few points that are not fully specified in the project presentation. These are our assumptions:

- All clients have got an email address;
- Each client can request only one taxi at a time;
- A client can take a shared taxi only if his ride has the same direction of the client who asked for the taxi first;
- There are enough taxies in order to cover all the city zones;
- Taxis can carry at most four passengers;
- Taxi drivers are characterized by credentials;
- Taxi drivers are already registered in the system;
- Taxi drivers always drive the same car;
- Taxis only operate inside the city;
- Taxi drivers always answer instantly to a system notification;
- System automatically understands if a person is a client or a taxi driver according to his email addresses;
- The system can manage clients time zones;
- The system is modular: the development team can add new services to the system without modifying the entire system;
- All the functions provided by the current system (such as the automatically distribution of taxis in the various zones based on the GPS) are assumed to be correct;
- Costs are proportional to the ride path.

2.4. Proposed System

We propose a project that will provide the services described below. Clients and taxi drivers will have access to the functionalities via a web browser or using the dedicated mobile application.

2.5. Identifying Stakeholders

Our main stakeholder is the government of the city, which gave us the commission to develop the project in order to simplify the access of passengers to the service and guarantee a fair management of taxi queues in the city zones.
Other stakeholders can be the clients and the taxi drivers.

2.6. Identifying Actors

There are four types of actors used in our system:

- **Guest**: a person who has not registered yet;
- **Client**: a person who can request, reserve or share a taxi;
- **Taxi driver**: a person who has already registered and can have access to the system functionalities in order to accept or decline taxi requests;
- **Dispatcher System:** the part of the system that manage taxi queues and assign requests to taxis;

3. Proposed system

3.1. Functional requirements

The system has got these functional requirements concerning the different type of user:

- **Guest**:
  - Sign up;
- **Client**:
  - Log in;
  - Log out;
  - Modify his credentials;
  - Ask for a taxi:
    - Create a taxi request;
      - Insert origin position and destination;
      - Insert number of passengers;
  - Manage the taxi reservation option:
    - Reserve a taxi:
      - Insert origin position, destination and time;
  - Manage his requests:
    - Modify a taxi request:
      - Modify origin position or destination or time;
    - Delete a taxi request;
  - Manage the taxi sharing option:
    - Enable or disable taxi sharing option;
    - Receive notifications about other passengers;
  - Receive a confirmation notification;
- **Taxi Driver**:
  - Log in;
  - Log out;
  - Modify his credentials;
  - Manage the taxi service status:
    - Take or suspend the service;
  - Manage clients' requests:
    - Accept or reject tasks;
    - View the route calculated by the system;
- **System**:
  - Management of taxi queues;
  - The system must correct and offer suggestions (i.e. allow address abbreviations) to clients during the data entry;
  - Calculation of the cost for each passenger;
  - Calculation of the best route to reach the destination;

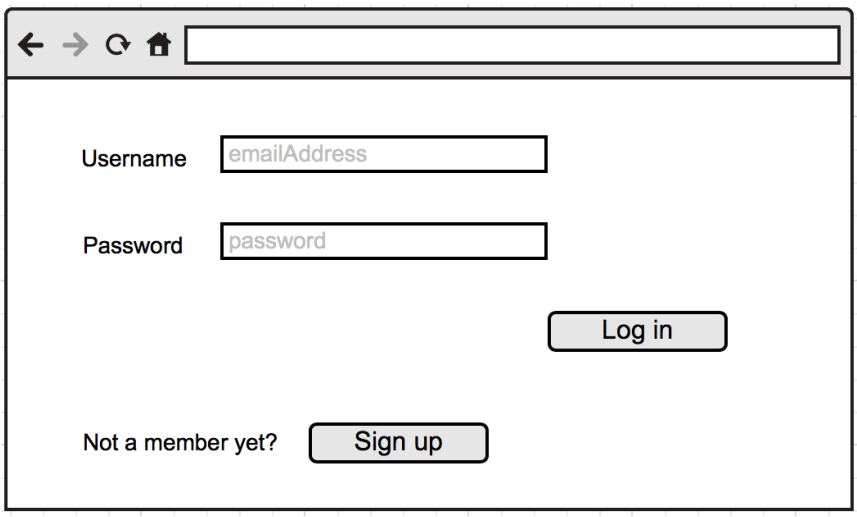
3.2. Non Functional requirements

3.2.1. User interface: navigational paths and Screen Mock Ups

The user interface will have the following characteristics:

- **Login page**: first, when the guest loads the login page he must fulfill the login format writing his username and password and then he can access to the service page pressing the "Log in" button. If a guest has not registered yet, he must use the registration button "Sign up" in order to do it. The login page can be represented through these mockups:

Web Interface



Mockup figure 3.1

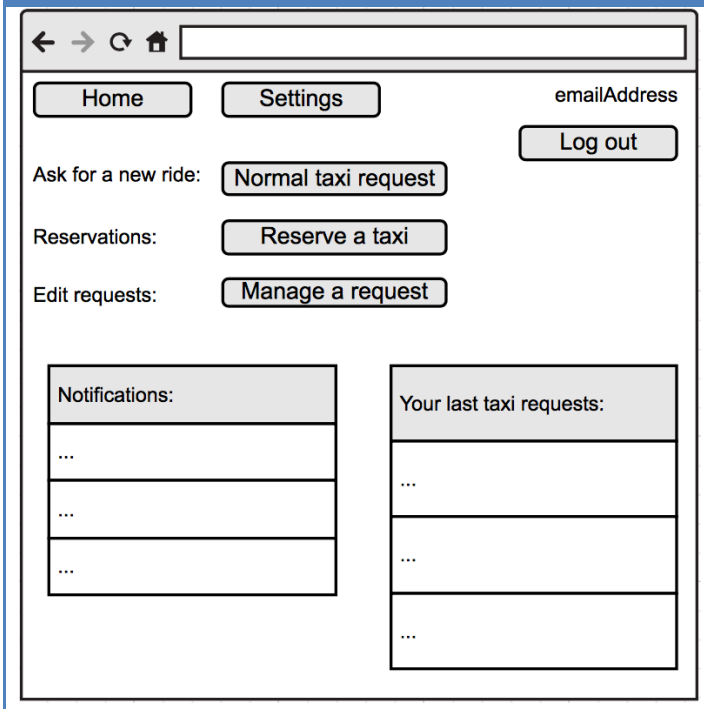
App Interface

Mockup figure 3.2

When a guest has logged in, a different type of home page will be loaded depending on the type of the user:

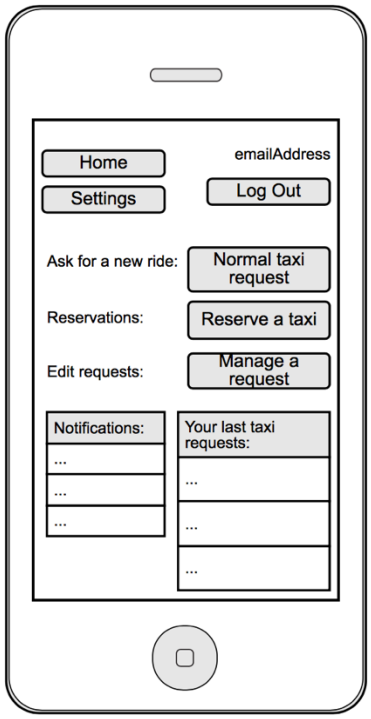
- **Client's home page**: clients can ask for a new taxi ride, reserve a taxi and edit their requests by using the dedicated buttons; it also has a notification list that contains the notifications sent by the system and a list of old taxi requests. The client's home page can be represented through these mockups:

Web Interface



Mockup figure 3.3

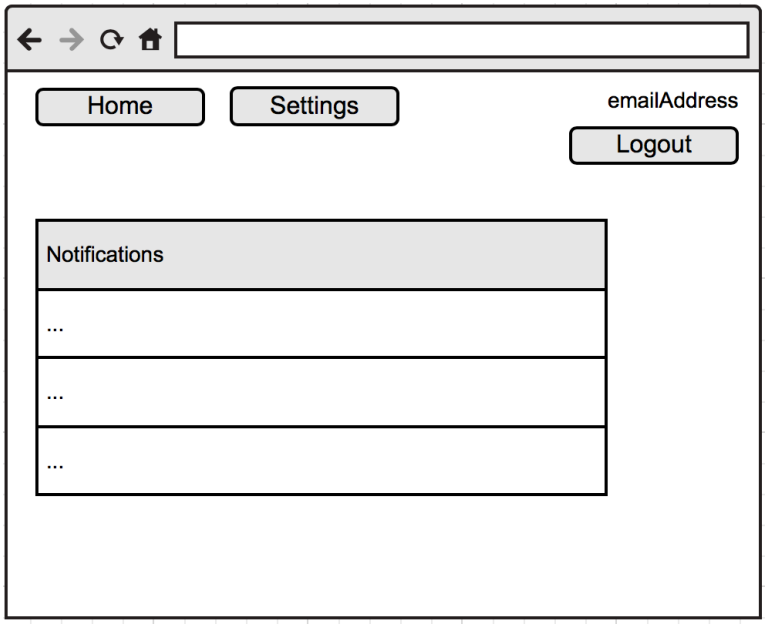
App interface



Mockup figure 3.4

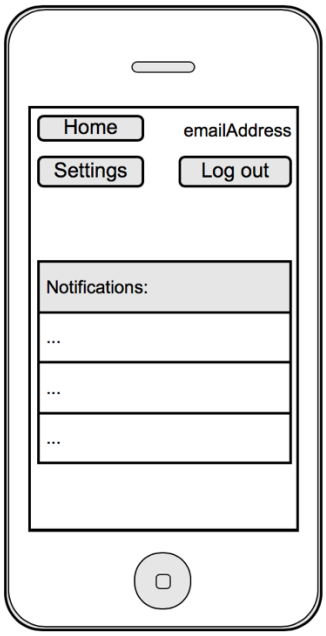
- **Taxi driver's home page**: taxi drivers can see the notifications about clients from the system; they can have more information about each one notification by tapping on it and the notification's page will be loaded. The taxi drivers' home page can be represented through these mockups:

Web interface



Mockup figure 3.5

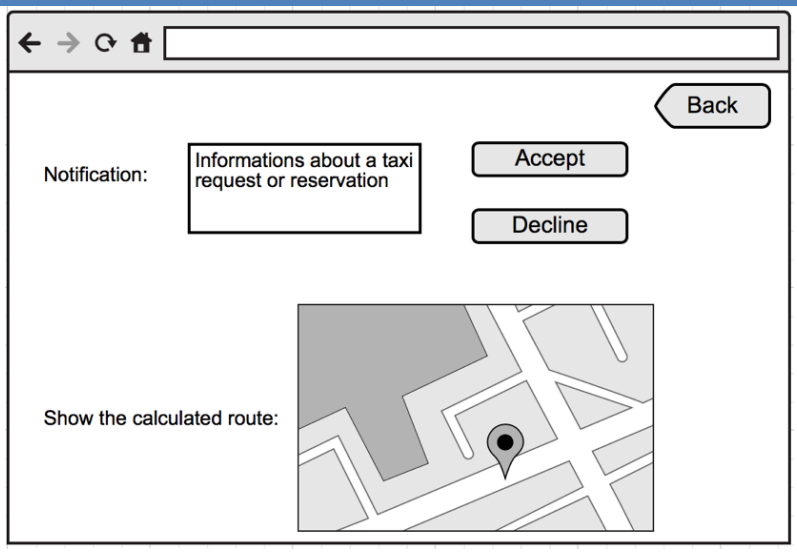
App interface



Mockup figure 3.6

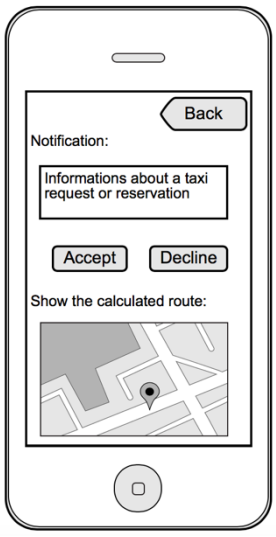
- **Taxi Drivers' notifications**: taxi drivers can have more details about clients' requests or reservations and they can accept or refuse each of them.

Web interface



Mockup figure 3.7

App interface

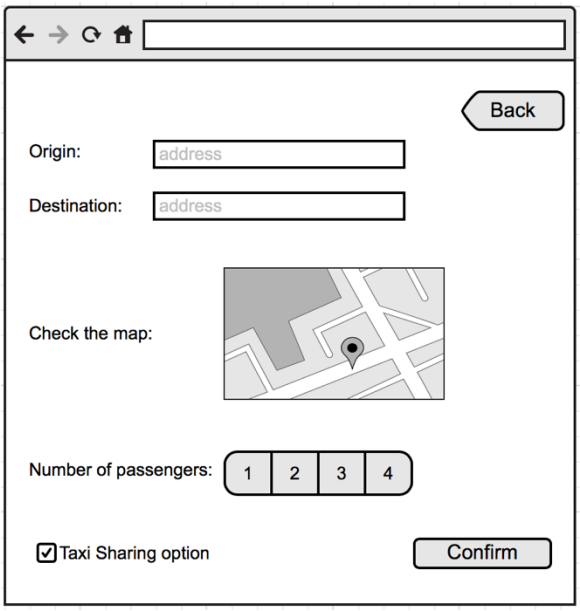


Mockup figure 3.8

Then, when a client taps a button the following pages will be opened:

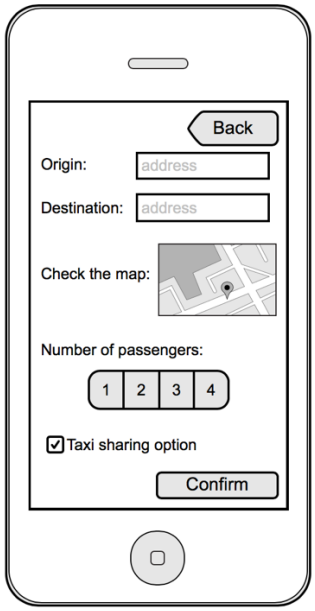
- **Request a taxi**: the client can insert the origin and the destination addresses, can set the number of passengers that need a taxi ride and can choose if to activate the taxi sharing option or not. This page can be represented through these mockups:

Web interface



Mockup figure 3.9

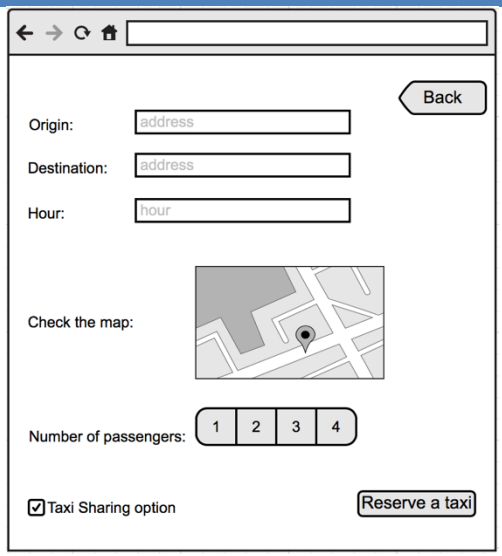
App interface



Mockup figure 3.10

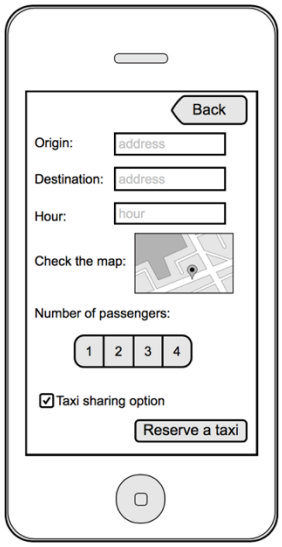
- **Reserve a taxi:** the client can insert origin and destination addresses and can choose the hour of the reservation; he can set the number of the passengers and choose if to activate the taxi sharing or not. This page can be represented through these mockups:

Web interface



Mockup figure 3.11

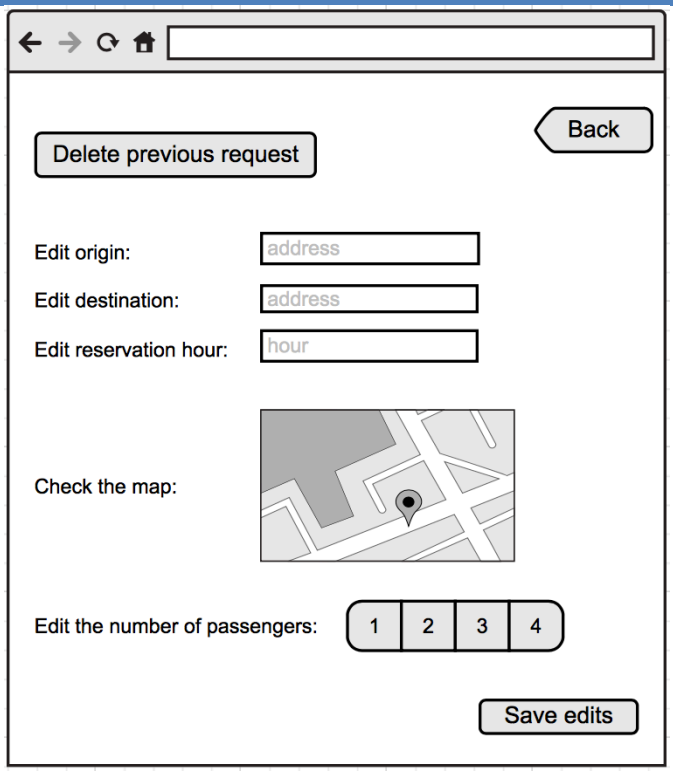
App interface



Mockup figure 3.12

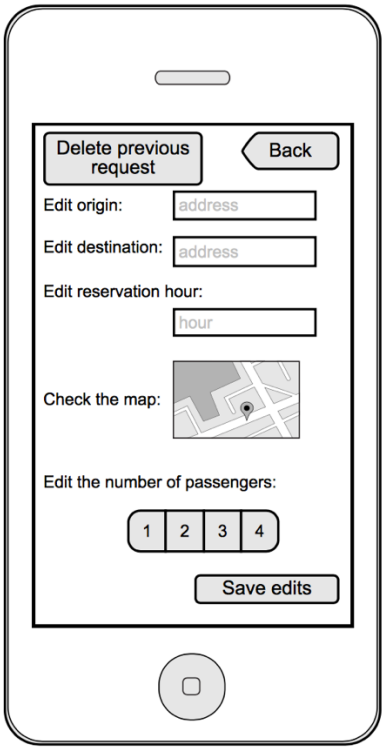
- **Manage requests**: the client can edit information added before and delete a request using the dedicated button. The editing pages can be represented using these following mockups:

Web interface



Mockup figure 3.13

App interface



Mockup figure 3.14

3.2.2. Performance requirements

3.2.2.1. Usability

- Simple to operate: the software must be easy to learn and operate; the user (client, taxi driver) must not require special skills or training in order to work with the system;
- Simple design: the user interface must be kept as simple as possible and the application must not be too confusing for the user to understand (friendly user interface);

3.2.2.2. Reliability

- The system must be up and running 24x7x365 and must be crash safe during its runtime.
- MTBF: if any, it must not be less than 6 months;
- MTTR: in case of a failure that leads to a system outage, the MTTR must not be more than 2 hours.

3.2.2.3. Performance

- Short response time: the load time of any page of the application (web page or mobile app) must not be more than 4 seconds.

3.2.2.4. Legal

- Data from users must adhere to the rights of data privacy or the user. All the content must be procured through legal channels and there must not be copyright violations.

3.2.2.5. Security

- The system must be secure; the data must be stored in a highly secure manner and must be immune from any hacking attempts.

3.3. Documentation

In order to illustrate the project development, the following documents will be provided:

- Requirements analysis and specification document (RASD), to describe the requirements of the project and how they will be satisfied;
- Design document (DD), to describe the structure of the system.

3.4. Architectural considerations

The services can be accessed through a web page or a mobile application: the user will need only a desktop browser or a smart device and an internet connection to have access to all functionalities of the system.

3.5. Scenarios

Here are described some possible scenarios of MyTaxiService.

3.5.1. Registration of a person into the system

Client registers into the system	
Goal	[G1] Registration of a person in the system
Assumptions	1-The client has not registered yet
Tom, who does not own a car, has to move to difference places because of his job. Sometimes he has to met a client by night in specific areas that are not reachable by using public transportation. He discovers MyTaxiService and decides to go to the website and to download the MyTaxiService mobile application. He fills in the registration form, providing his e-mail address and a password of his choice.	

3.5.2. Visualization of the personal home page

Client views his personal homepage	
Goal	[G2] Visualization of the personal homepage
Assumptions	1-A reservation has already been created
Tom has to move to a district of the town in the afternoon to meet a client. He knows that he has already created a reservation but he does not remember the right timetable, so he opens MyTaxiService mobile application or website. He fills in the login form with his e-mail address and password. Then, the home page provides an overall view about all the most recent reservations that has been done by Tom. He finds out that he has a reservation at 2 PM and he can see the taxi code and the ETA .	

3.5.3. Allowing a client to request a taxi

Client wants to send a taxi request	
Goal	[G3] Allowing clients to request a taxi [G2] Visualization of the personal homepage
Assumptions	1- Client has already registered and logged in
Tom does not want to be late at a client meeting, so he decides to use MyTaxiService instead of taking the bus. He taps on the "Request a taxi" button and then he fills in the origin and the destination addresses. Then the system sends him a notification with the number of the taxi and the ETA.	

3.5.4. Allowing a client to reserve a taxi

Client wants to send a taxi reservation	
Goal	[G3] Allowing clients to request a taxi [G2] Visualization of the personal homepage
Assumptions	1-Client has already registered and logged in
Tom has to be at home at 8.00 PM and he does not want to be late. So he decides to request a taxi: he taps on the "Reserve a taxi" button and then he fills in the origin address, the destination address and the hour when he wants to take the taxi.	

3.5.5. Allowing clients to share a taxi

Client wants to share a taxi ride with other users	
Goal	[G4] Allowing clients to reserve a taxi [G2] Visualization of the personal homepage [G5] Allowing clients to share a taxi
Assumptions	1-Client has already registered and logged in
Sometimes Tom has to take the taxi more than one time in the same day, but he wants to save his money. So he decides to share a taxi with other people. He creates a new taxi request and switch on the taxi sharing option. Therefore, if other clients will ask for a sharing ride, they will be notified by the system if their ride direction is the same so that them they can share the ride and divide the costs.	

3.5.6. Allowing clients to edit a taxi request/reservation

Client edits a reservation	
Goal	[G6] Allowing clients to edit a taxi request/reservation [G2] Visualization of the personal homepage [G4] Allowing clients to reserve a taxi
Assumptions	1-A reservation has already been created 2-Client has already registered and logged in 3-Client is in time to edit his request
Tom has to meet a client in the evening, but unfortunately he gets sick so he decides to postpone the meeting for the next week. He finds out in the homepage that he has a reservation. Then he taps on the "Edit" button in order to change the reservation timetable. Therefore the system will send him a new notification to confirm his new reservation and the successfully deleting of the previous one.	

3.5.7. Allowing a taxi driver to accept or refuse a client request/reservation

Allowing taxi drivers to accept or refuse a request or a reservation	
Goal	[G7] Allowing taxi drivers to accept or refuse a request/reservation [G2] Visualization of the personal homepage
Assumptions	1-Taxi driver has already registered and logged in
The taxi driver Paul is free at the moment, so he opens MyTaxiService app or website. He sees in the homepage that he has received a notification from the system in which there is a request of a taxi ride with the number of passengers, the origin and final destination. So Paul decides to take care of this request and he taps on the "Accept" button.	

3.5.8. Calculation of the costs done by the system

System calculates the cost of a ride	
Goal	[G8] Calculation of the cost done by the system [G7] Allowing taxi drivers to accept or refuse a request/reservation [G3] Allowing clients to request a taxi [G2] Visualization of the personal homepage
Assumptions	1-Taxi drivers and clients have already registered and logged in; 2-A Taxi request has already asked for; 3-Taxi driver has already accepted the request;
Tom has asked for a taxi in order to move from a city zone to an other one and Paul is the taxi driver arrived to bring him to the destination. During the route the system calculates the cost of the ride basing on the kilometers of the travel and the time spent to cover them and at the end of the ride the system will send a notification to Paul in order to tell him the total amount of the cost of the ride.	

3.5.9. Calculation of the routes done by the system

System calculates the route of a ride	
Goal	[G9] Calculation of the routes done by the system [G3] Allowing clients to request a taxi [G2] Visualization of the personal homepage
Assumptions	1-Taxi drivers and clients have already registered and logged in; 2-Client has already asked for a taxi;
Tom has asked for a taxi using MyTaxiService app and his request has been analyzed by the system. It finds the best route in order to cover the asked distance and sends a notification to the taxi driver Paul on the top of the queue of the zone where the client wants to start the ride. Paul in the notification sees the calculated route and then he can decide if to take care of the request or if to reject it.	

3.5.10. Fair management of the queues

System arranges zone queues	
Goal	[G10] Fair management of the routes done by the system [G3] Allowing clients to request a taxi [G7] Allowing taxi drivers to accept or refuse a request/reservation [G2] Visualization of the personal homepage
Assumptions	1-Taxi drivers and clients have already registered and logged in; 2-Client has already asked for a taxi;
Tom has asked a taxi and the system has already received his request and has already informed the taxi driver Paul about it. For some reasons Paul decides to not take care of the request and to reject it. Then the system receives the will of Paul and automatically sends him to the end of his zone queue and sends the same taxi request to the new taxi driver Jim on the top of that queue.	

3.5.11. Allowing clients to choose a shared ride

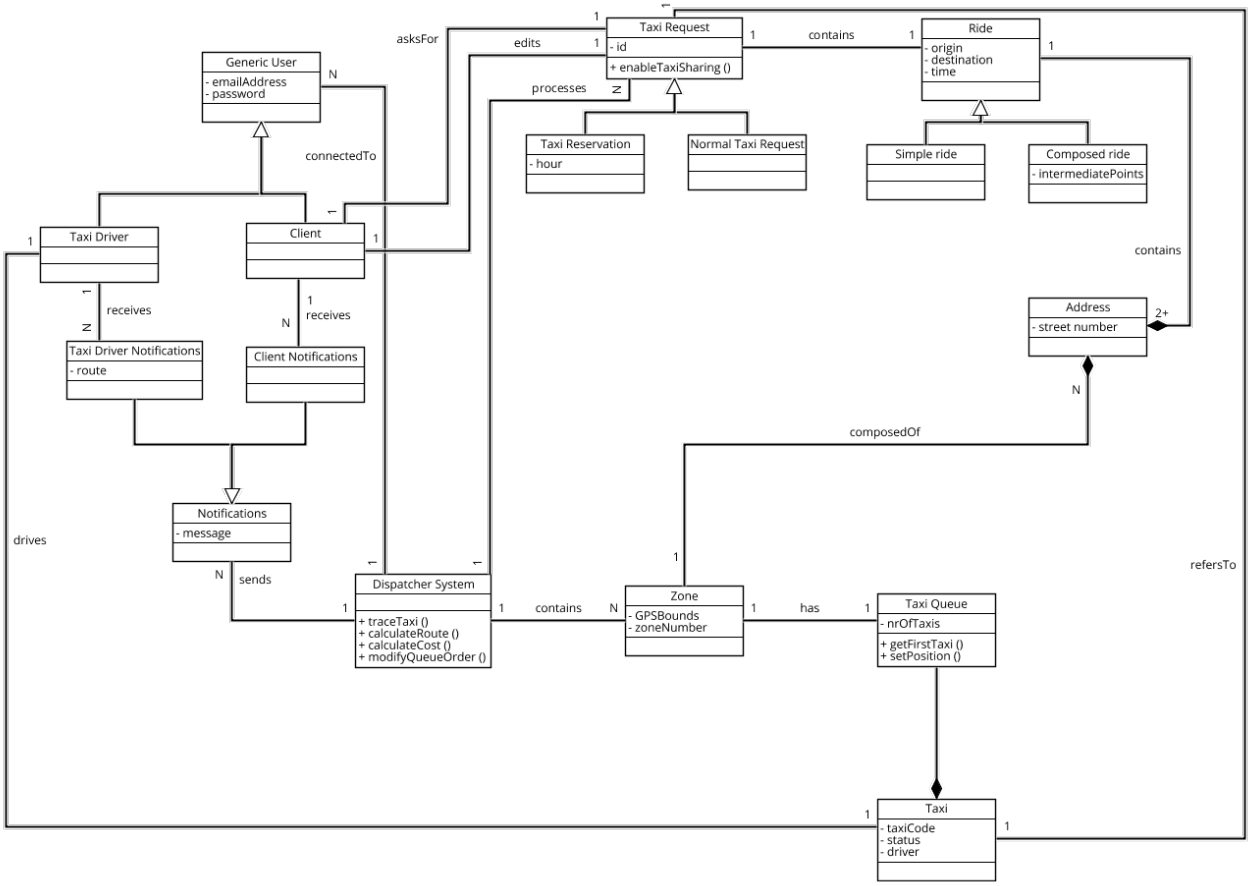
Client must be able to choose a shared ride	
Goal	[G11] Allowing clients to choose a shared ride [G3] Allowing clients to request a taxi [G5] Allowing clients to share a taxi [G2] Visualization of the personal homepage [G9] Calculation of the routes done by the system
Assumptions	1-Taxi drivers and clients have already registered and logged in 2-Clients have already asked for a taxi
Tom has requested a taxi and has the taxi sharing option set as active. The system has received his notification and it calculates the corresponding route; during the ride of Tom, an other client Billy asks for a sharing taxi and the system, calculating the route, checks that the origin position of Billy's ride is on the path of Tom's and that both rides have got the same direction. So the system sends them a notification that they will share the ride.	

3.5.12. Allowing clients to see the code and ETA of the incoming taxi

Client receives taxi code and ETA	
Goal	[G7] Allowing taxi drivers to accept or refuse a request/reservation [G3] Allowing clients to reserve a taxi [G7] Allowing taxi drivers to accept or refuse a request/reservation [G2] Visualization of the personal homepage
Assumptions	1-Taxi drivers and clients have already registered and logged in; 2-Client has already request a taxi 3-Taxi driver has already accepted a taxi request
Tom has asked for a taxi and the system has sent the notification to the taxi driver Paul. Paul decides to accept it and sends the confirmation to the system. Then the system sends a notification to Tom in order to inform him about the Paul's taxi code and the ETA.	

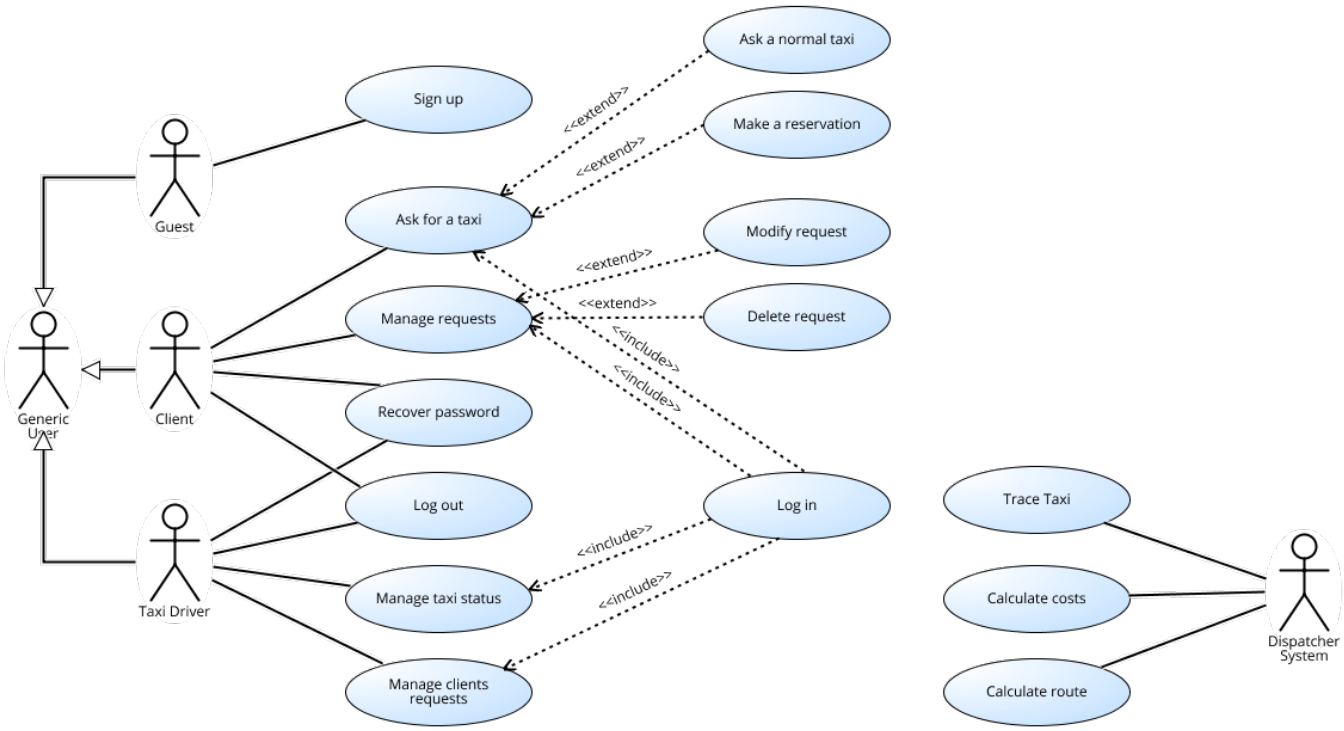
3.6. Class Diagram

Before analyzing the use cases, we provide the class diagram of our system, in order to clarify the structure of the project problem.
The diagram is the result of the analysis of the actors and other elements involved into the problem.



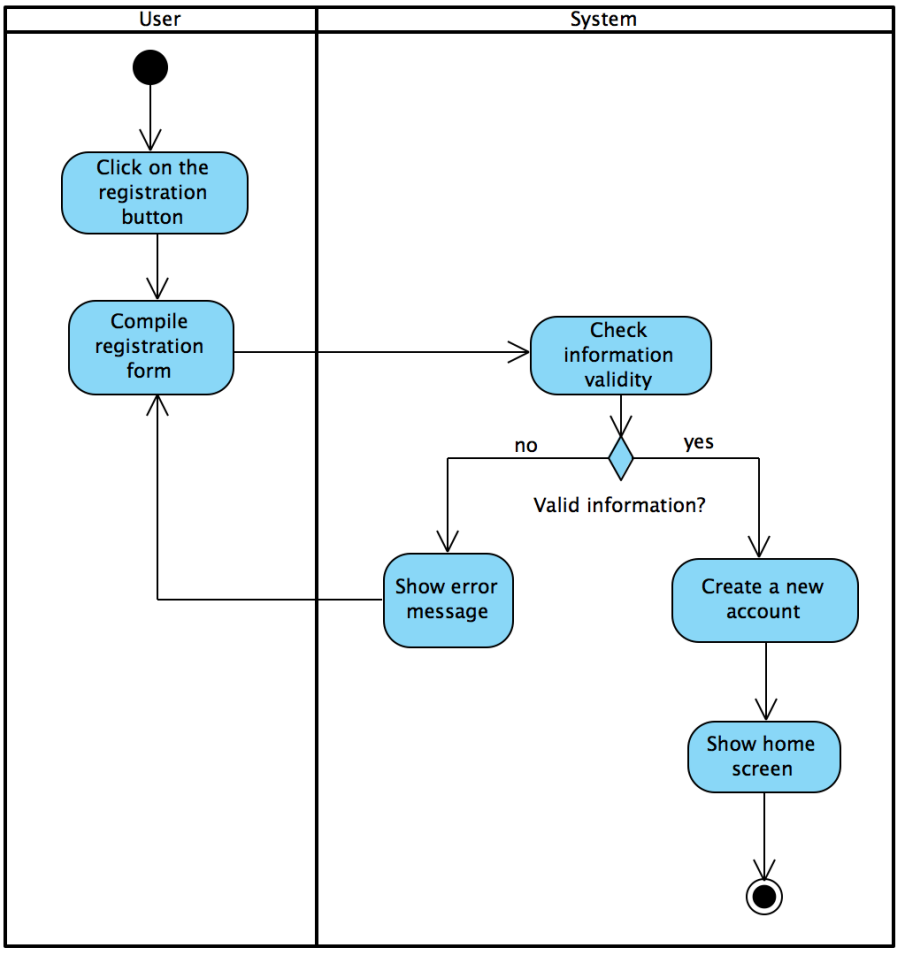
3.7. Use cases

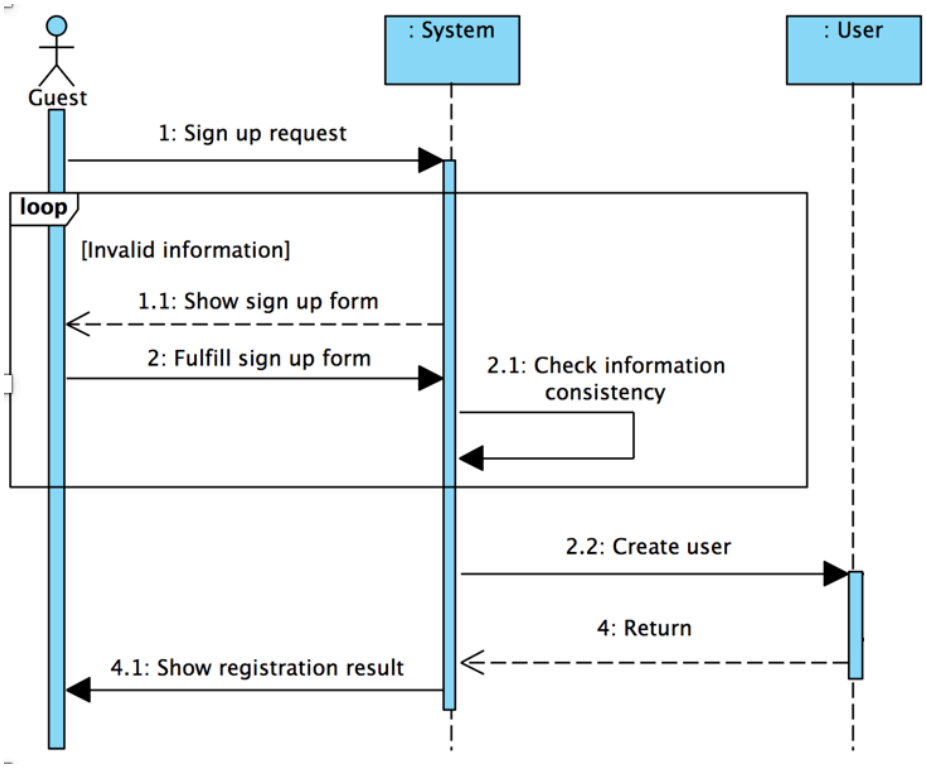
We now provide a general use case diagram with macro use cases in order to directly represent the whole MyTaxiService system. Then, each use case will be analyzed more in detail in the next sections.



Use Case figure 3.1

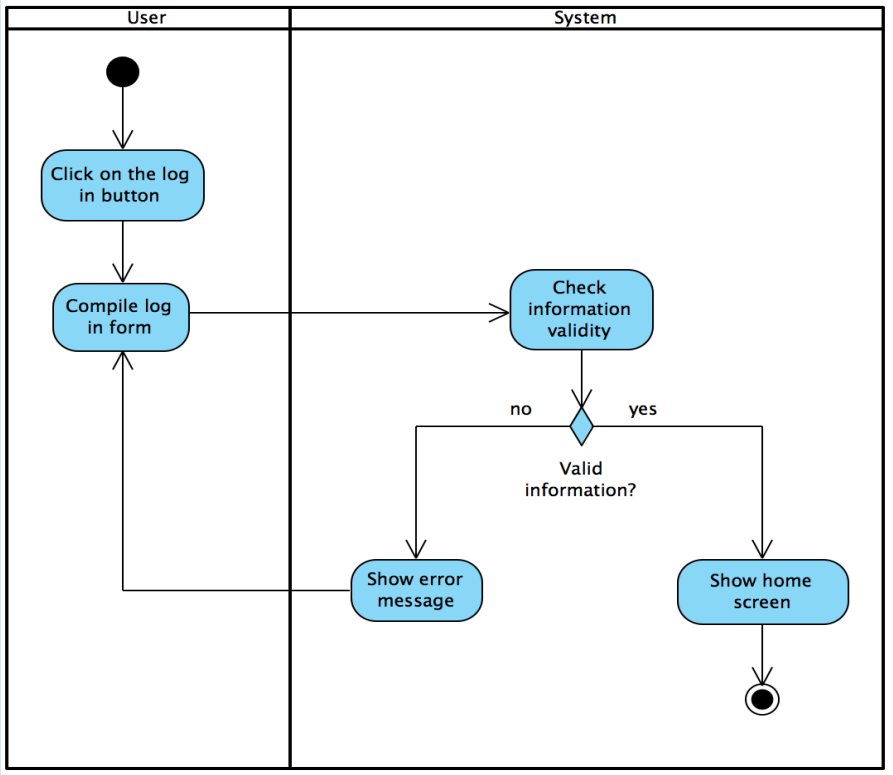
3.7.1. Sign up

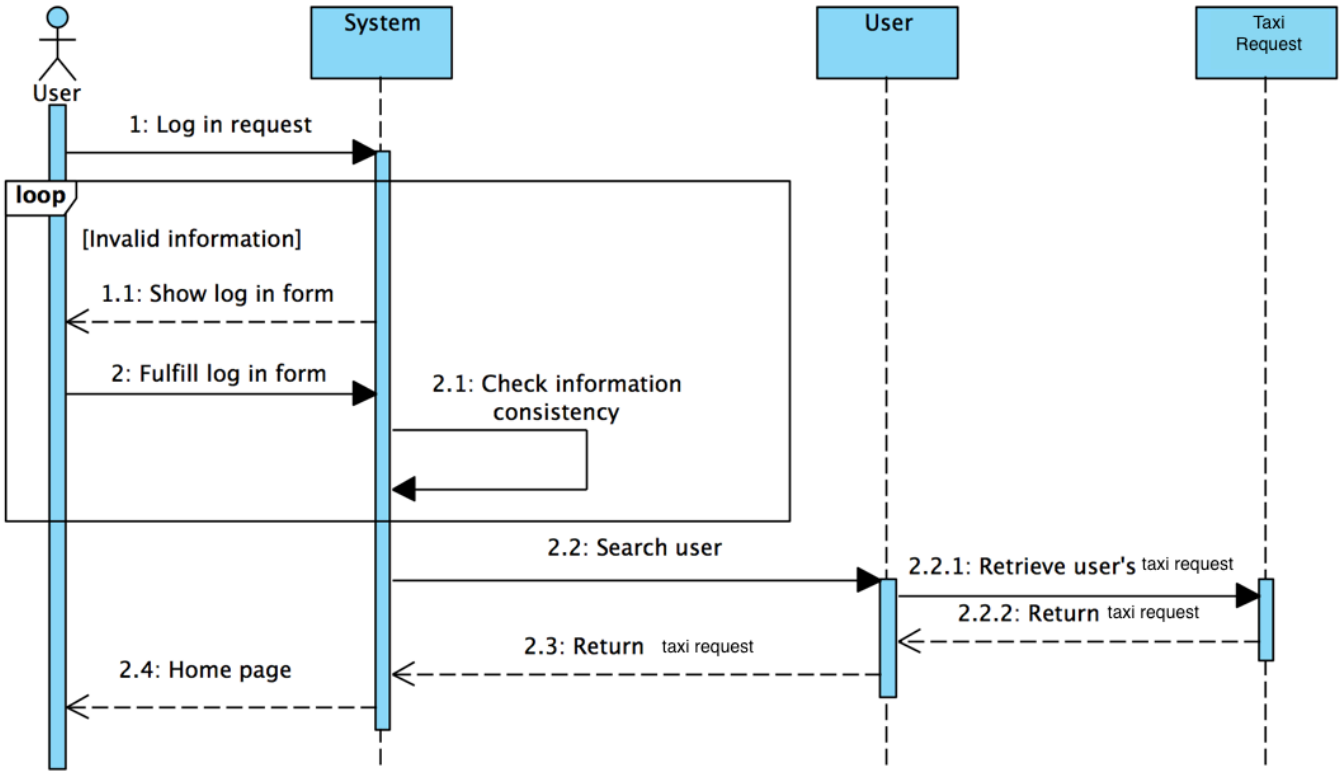
Sign up	
Actors	Guest
Entry condition	The guest is in the login page of MyTaxiService
Flow of Events	
Exit condition	The new account has been created and user can start to use MyTaxiService
Exceptions	Guest does not fill some fields of the registration form. In this case, the registration form appears again until the information is not valid.



Sequence diagram figure 3.1

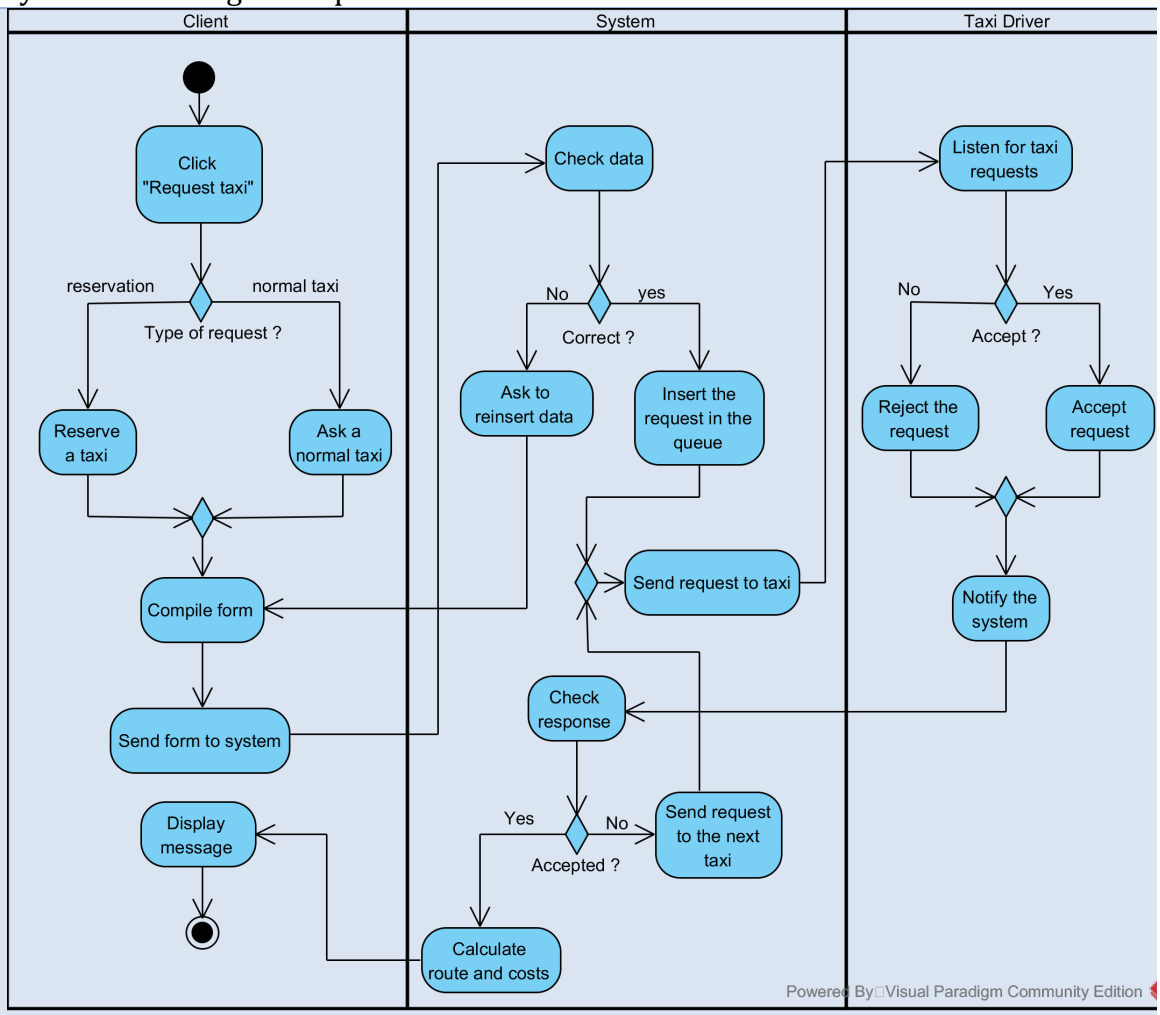
3.7.2. Log in

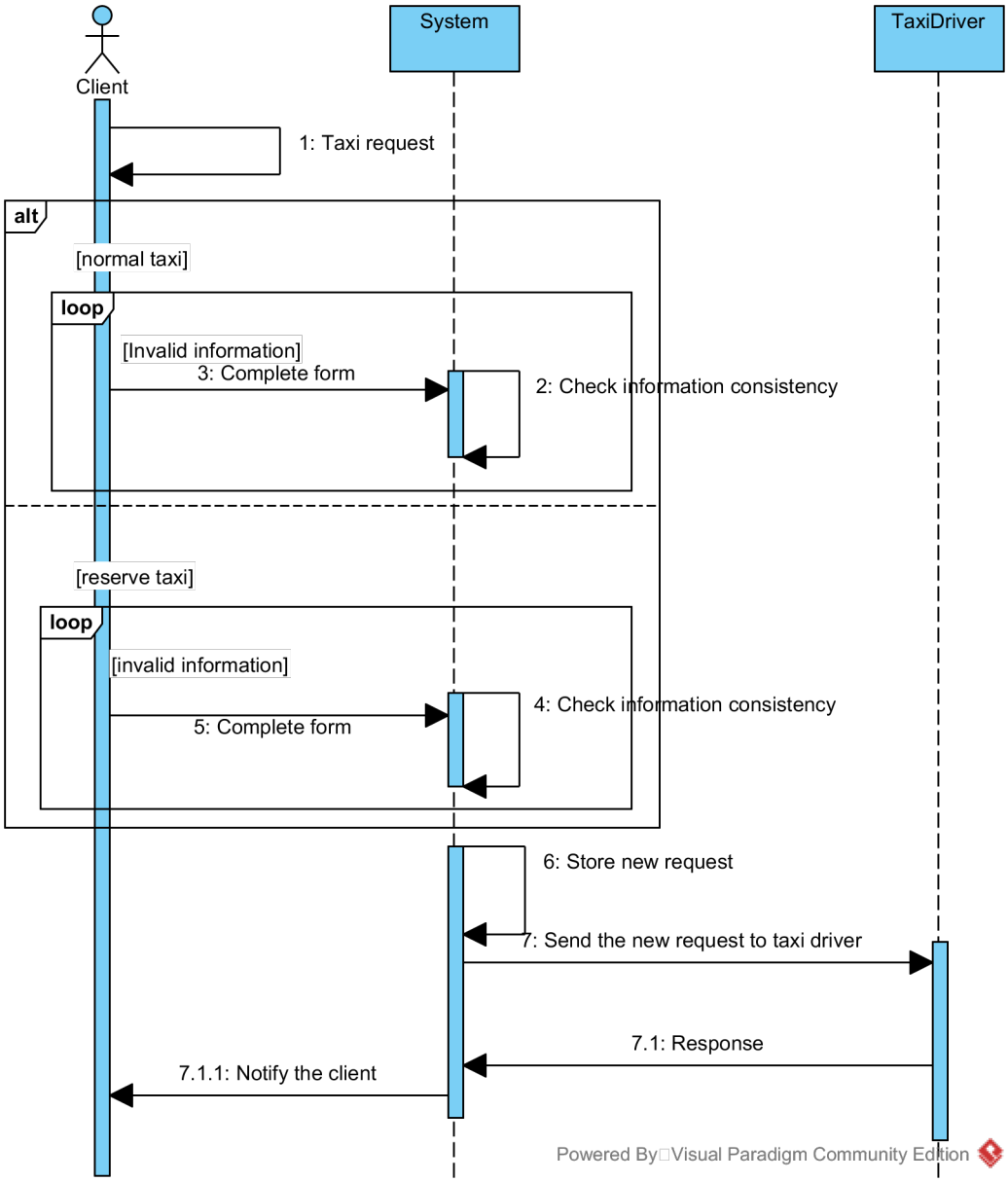
Log in	
Actors	Client, Taxi Driver
Entry condition	User has opened the program
Flow of Events	
Exit condition	User can now use the entire set of functionalities provided by MyTaxiService
Exceptions	The provided information is not valid; in this case, the system shows an error message.



Sequence diagram figure 3.2

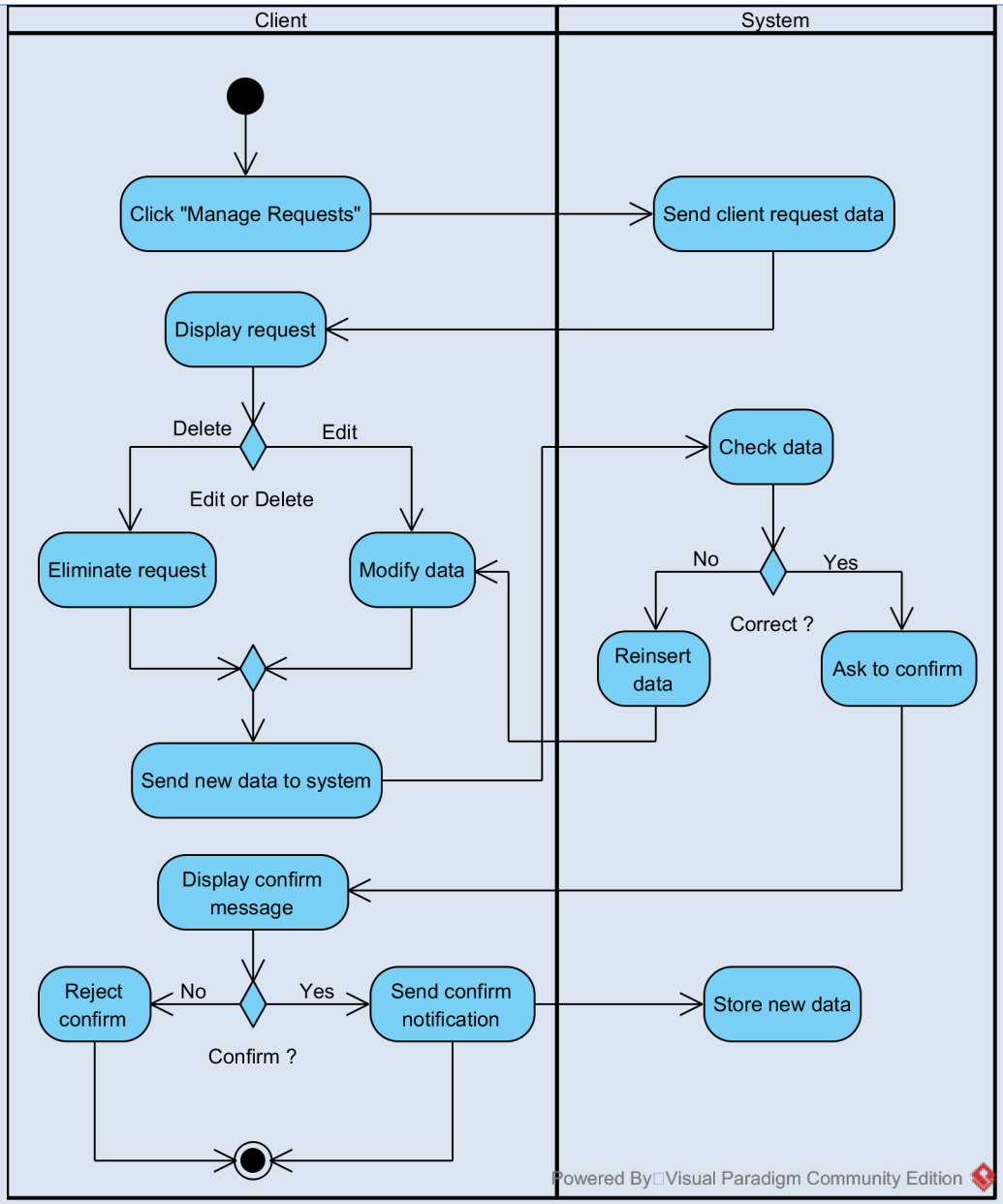
3.7.3. Ask for a taxi

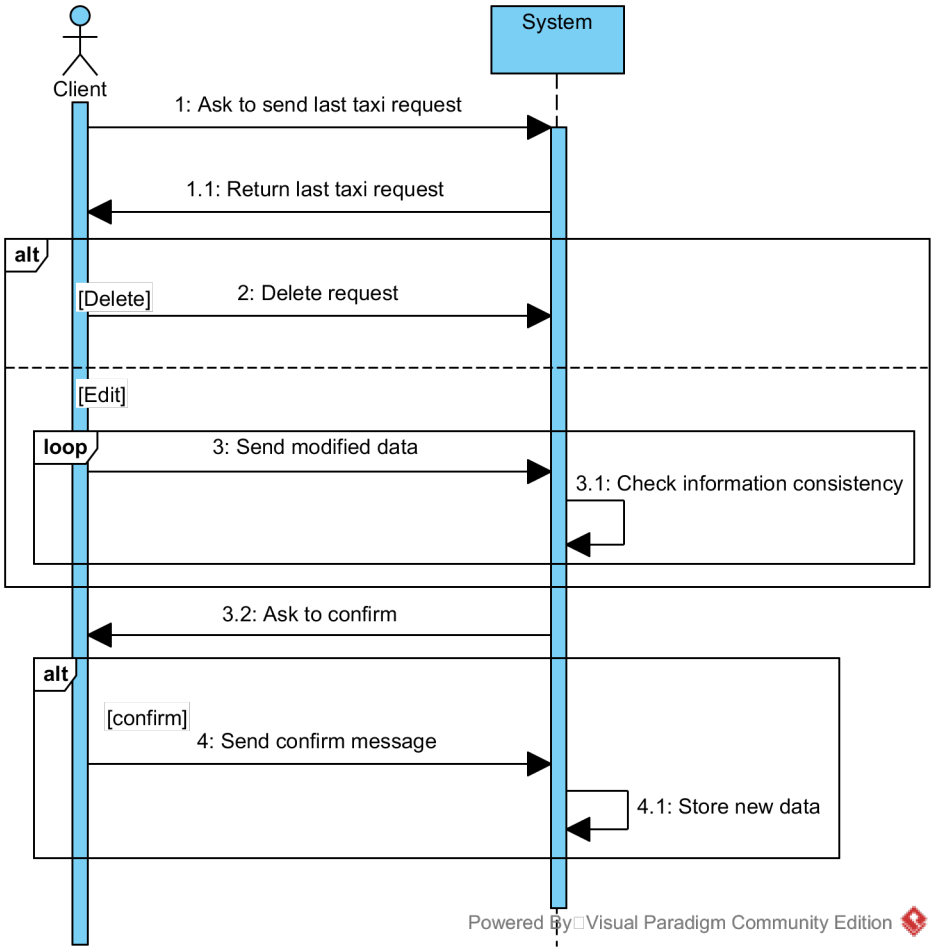
Ask for a taxi	
Actors	Client, Taxi Driver, System
Entry condition	Client has opened the program Taxi Driver is in service System is waiting for requests
Flow of Events	
Exit condition	Client is/is not assigned a taxi Taxi status is occupied/free
Exceptions	The provided information is not valid; in this case, the system shows an error message.



Sequence diagram figure 3.3

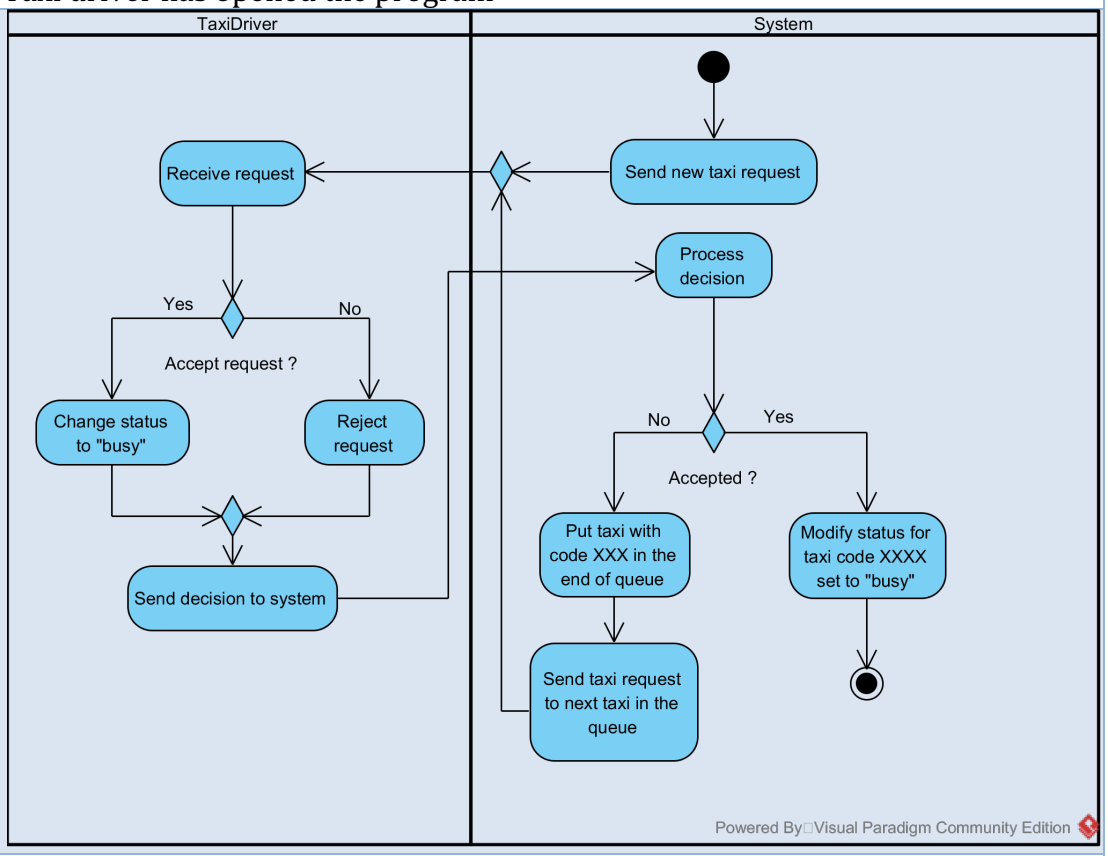
3.7.4. Manage requests

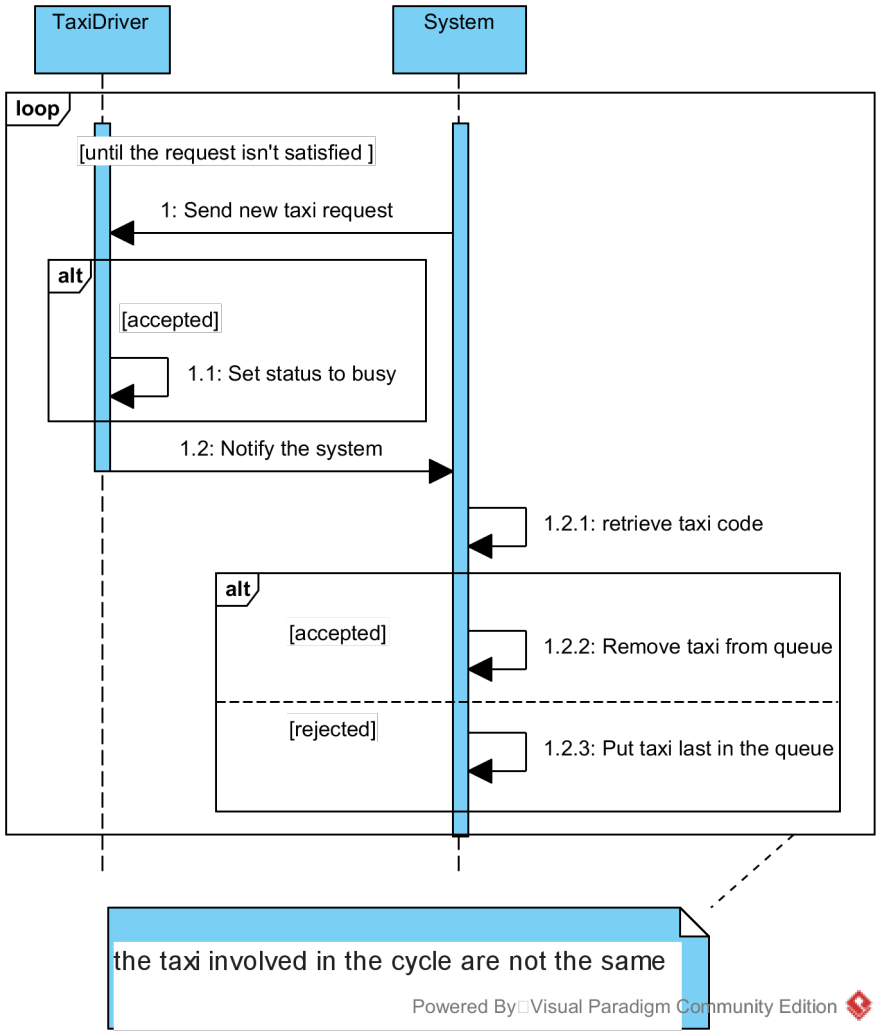
Manage requests	
Actors	Client, System
Entry condition	Client has opened the program and he is looking his request
Flow of Events	
Exit condition	Client is/is not assigned to a taxi
Exceptions	N/A



Sequence diagram figure 3.4

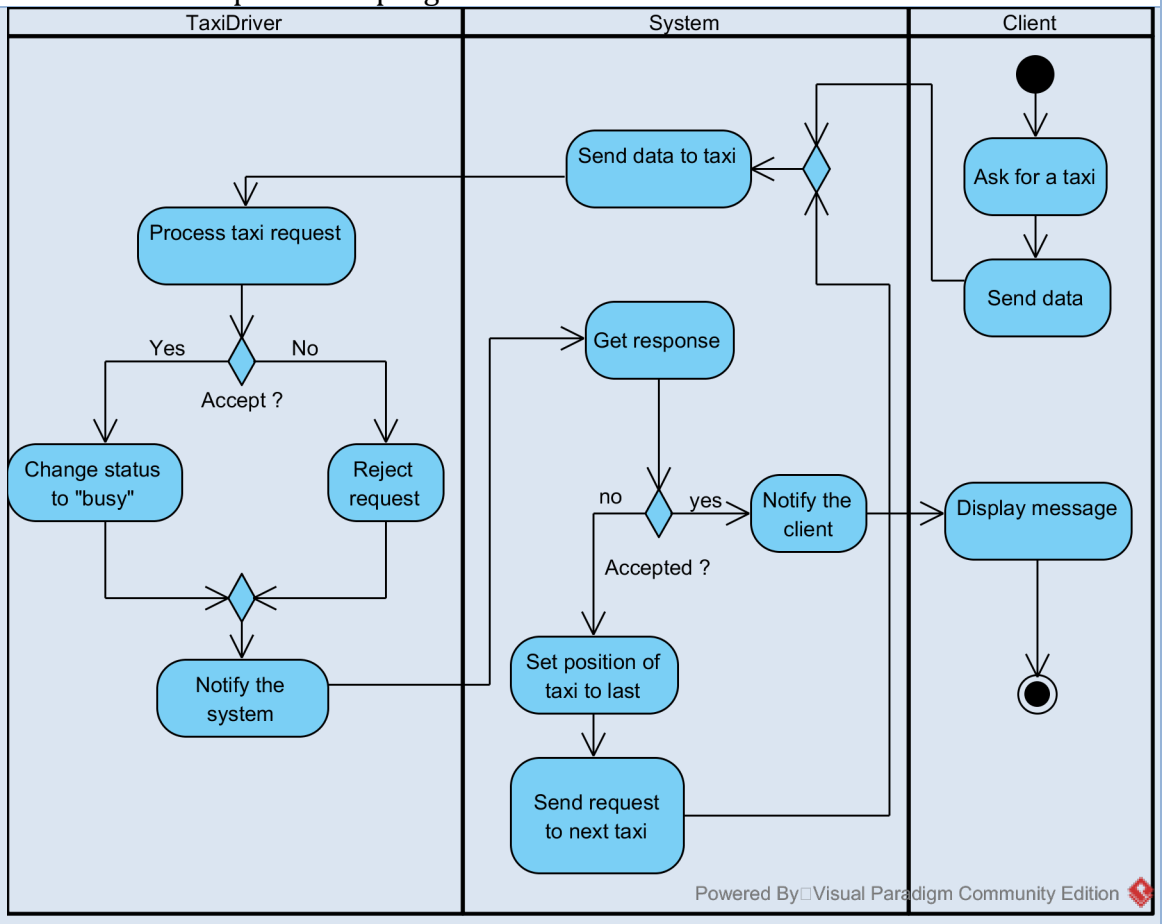
3.7.5. Manage taxi status

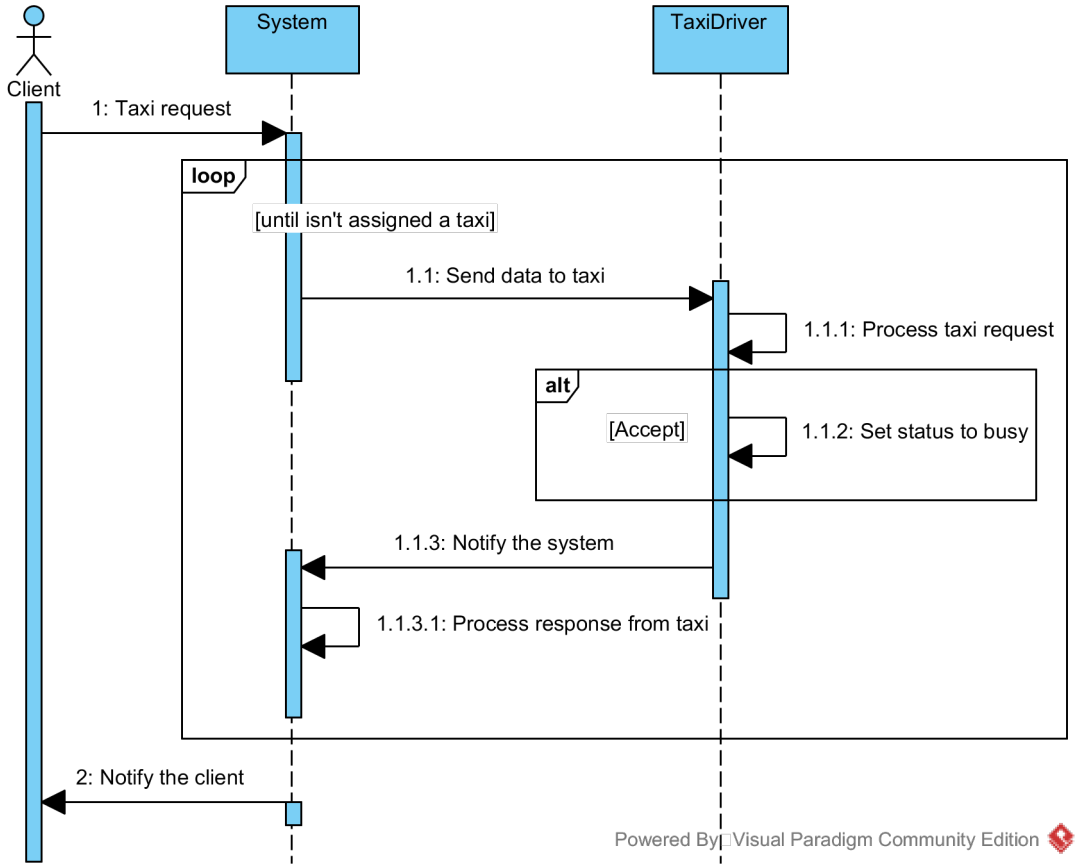
Manage taxi status	
Actors	Taxi driver, System
Entry condition	Taxi driver has opened the program
Flow of Events	
Exit condition	Taxi status is busy/ free
Exceptions	N/A



Sequence diagram figure 3.5

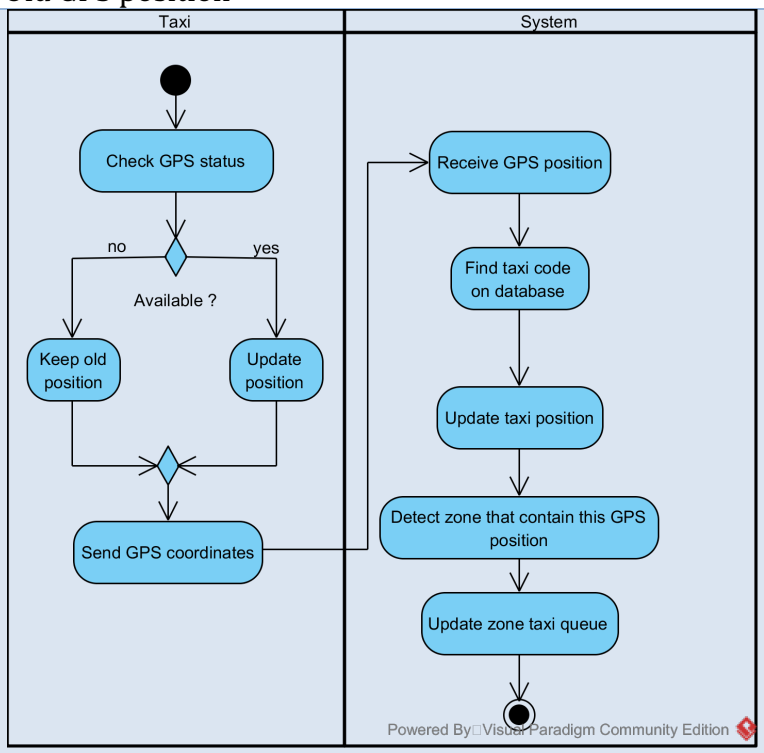
3.7.6. Manage clients' requests

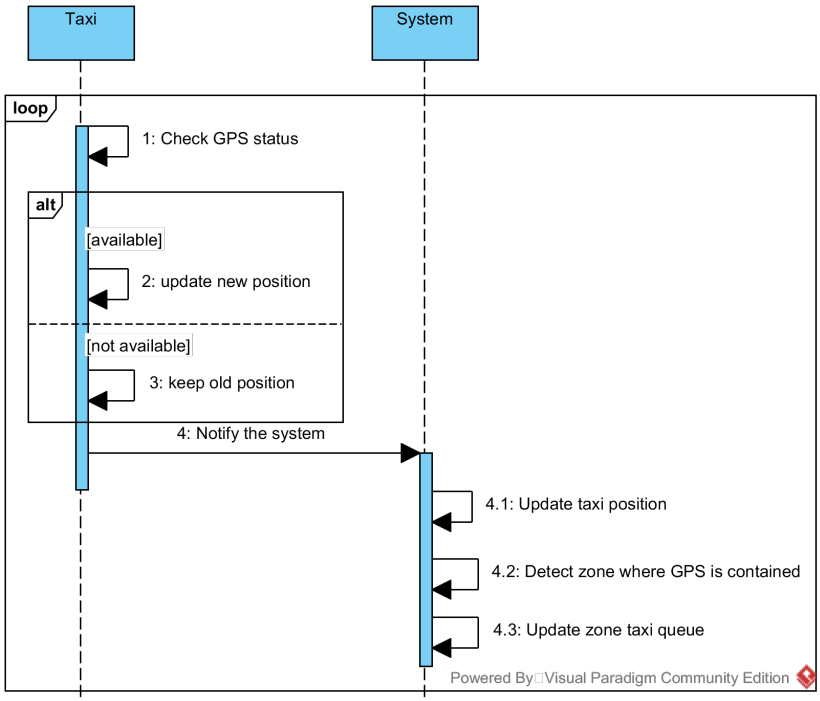
Manage clients requests	
Actors	Taxi driver, System, Client
Entry condition	Client has opened the program (app or web) Taxi driver has opened the program
Flow of Events	
Exit condition	Taxi status is busy/ free Client is waiting for taxi
Exceptions	N/A



Sequence diagram figure 3.6

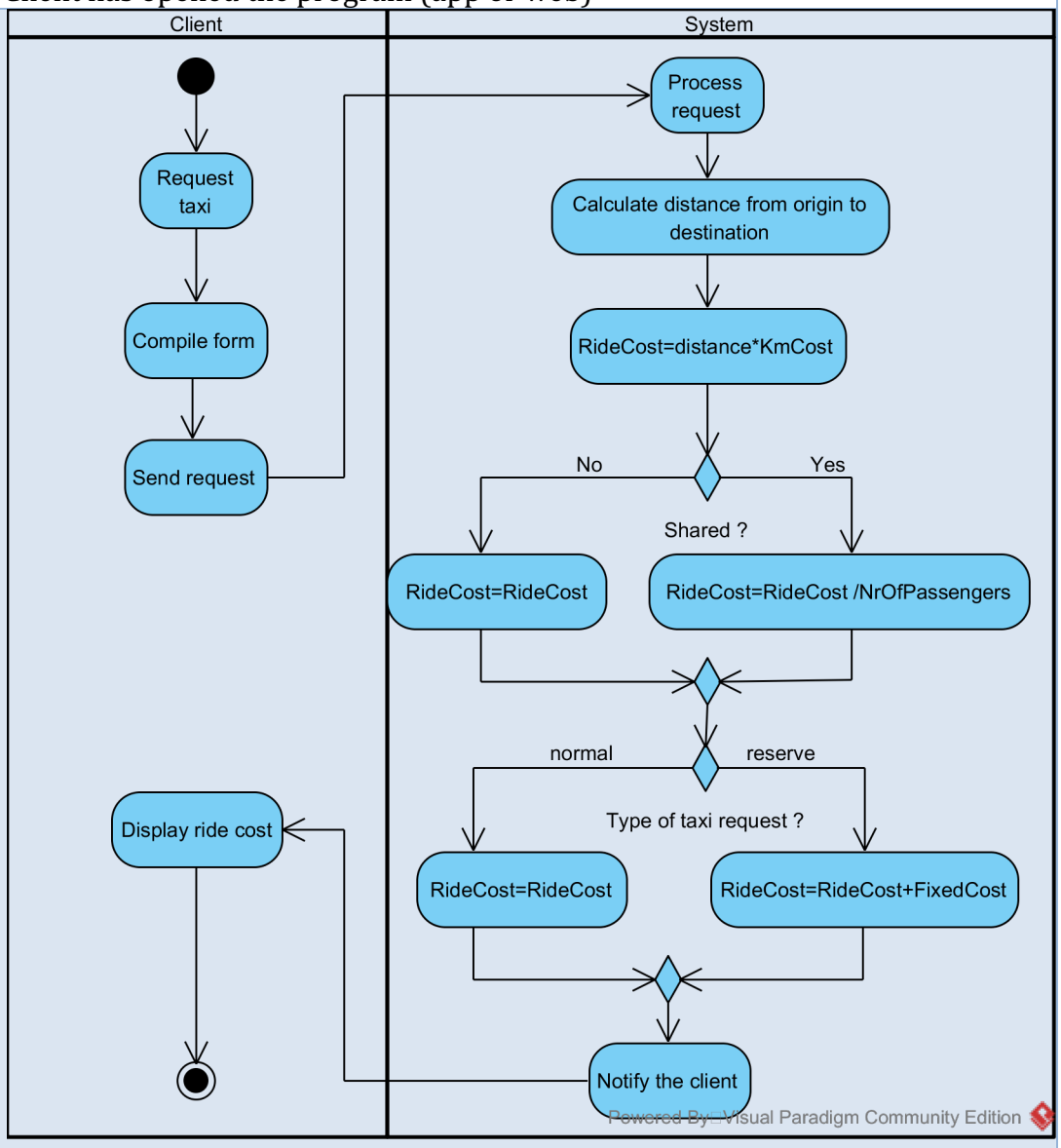
3.7.7. Trace a taxi

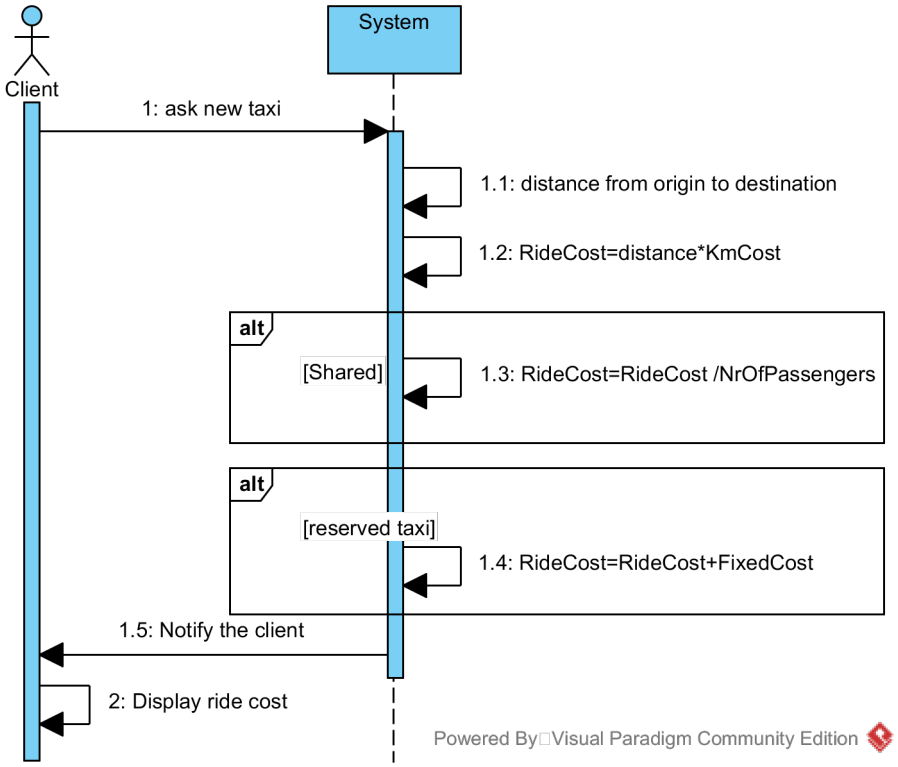
Trace a taxi	
Actors	Taxi, System
Entry condition	Old GPS position
Flow of Events	
Exit condition	Zones taxi queue updated
Exceptions	When the taxi is underground the GPS position cannot be traced, so the system set the old position as the new position.



Sequence diagram figure 3.7

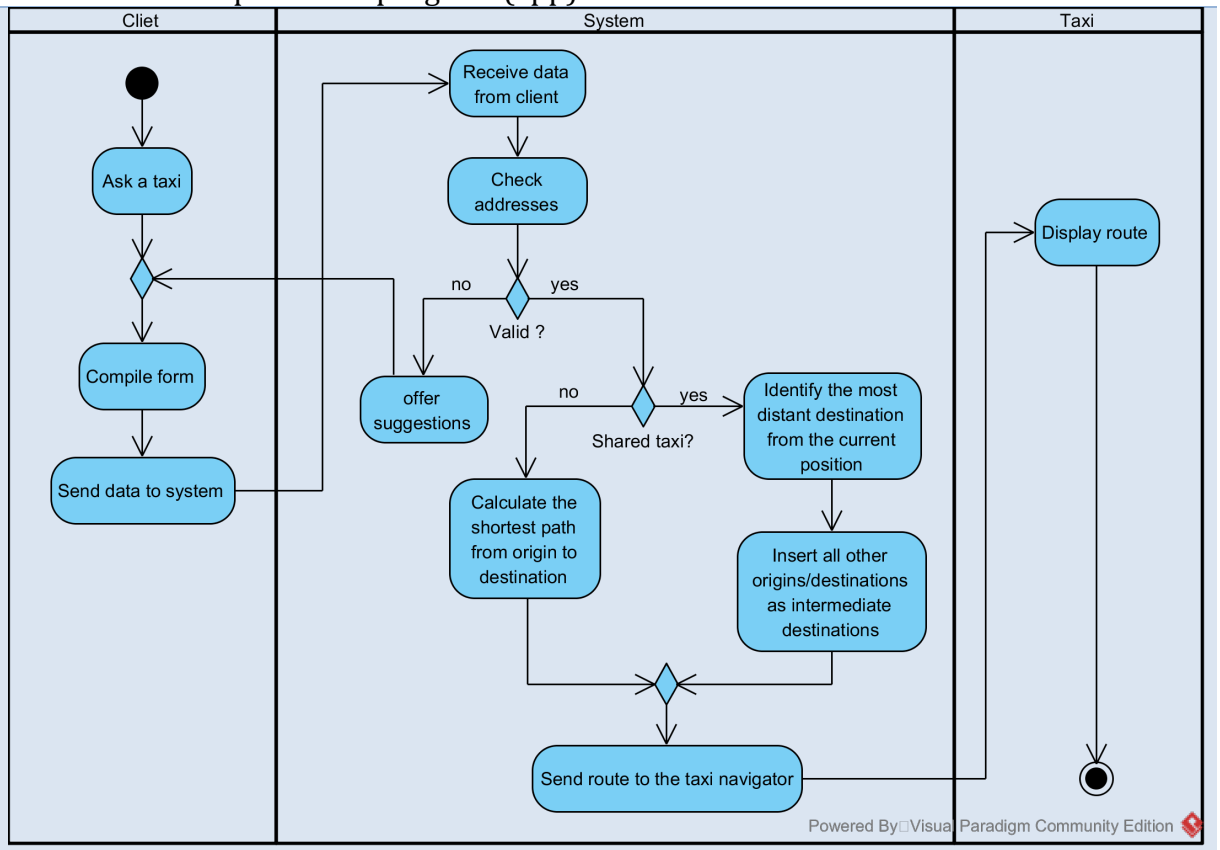
3.7.8. Calculate costs

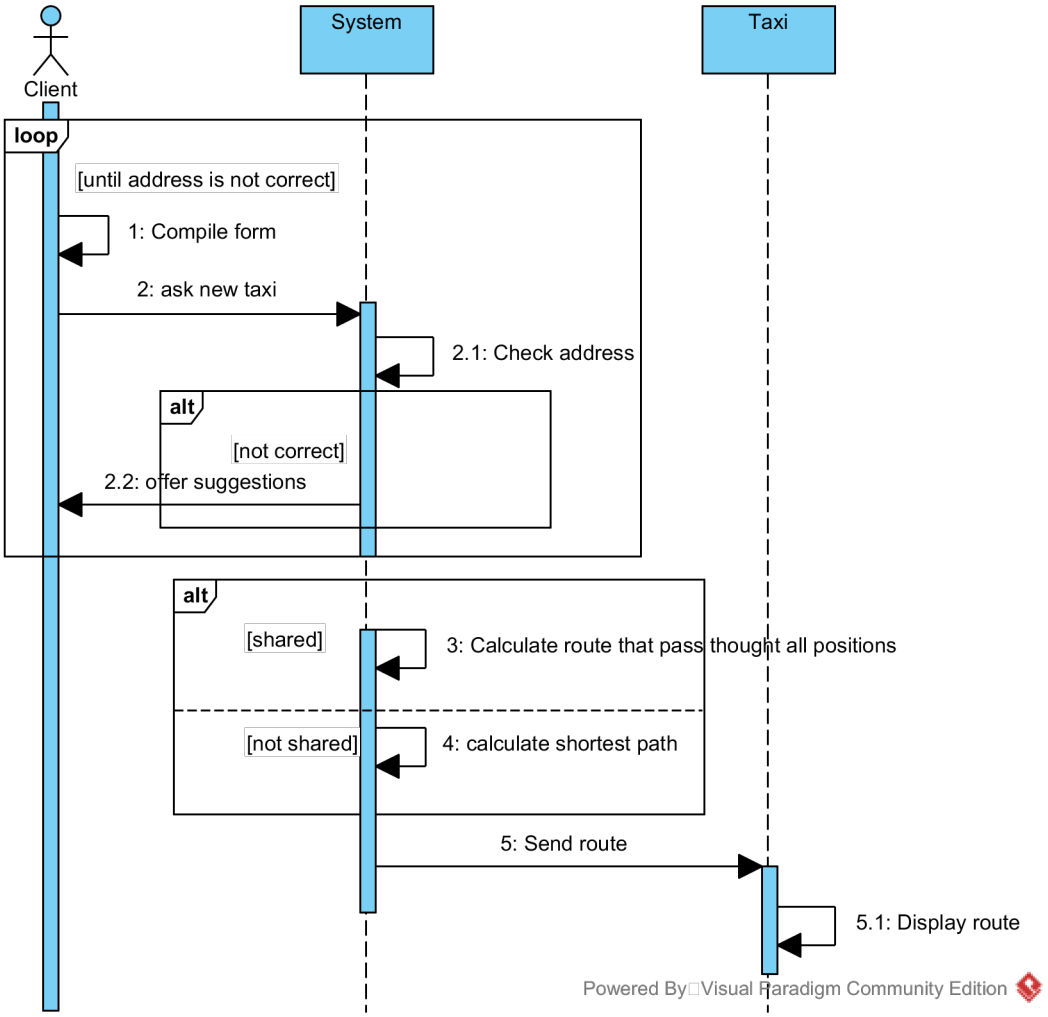
Calculate costs	
Actors	System, Client
Entry condition	Client has opened the program (app or web)
Flow of Events	
Exit condition	Cost of the ride
Exceptions	N/A



Sequence diagram figure 3.8

3.7.9. Calculate route

Calculate route	
Actors	Client, System, Taxi Driver
Entry condition	Client has opened the program (app or web) Taxi driver has opened the program (app)
Flow of Events	
Exit condition	Shortest route to destination
Exceptions	N/A

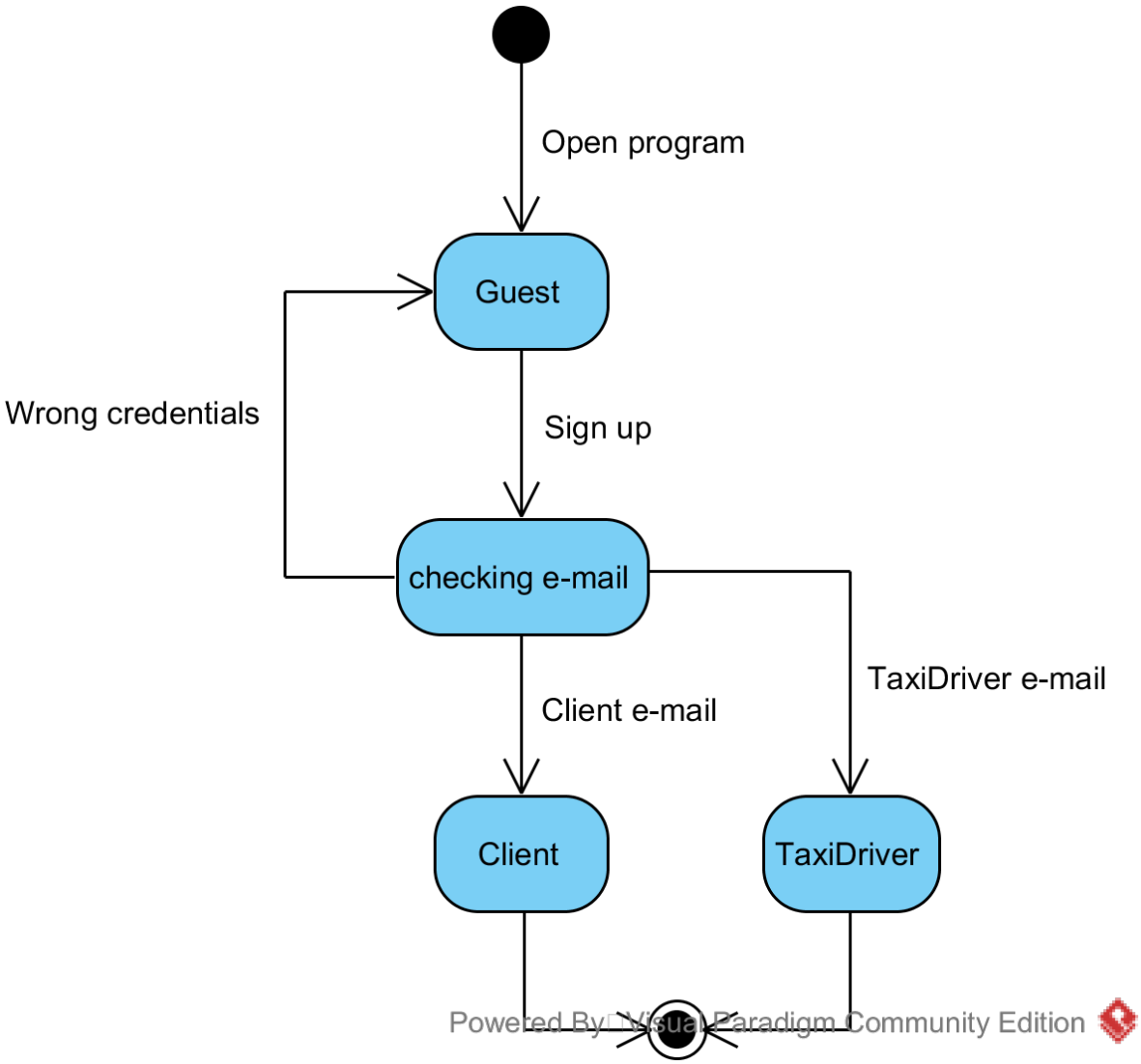


Sequence diagram figure 3.9

3.8. State machine diagrams

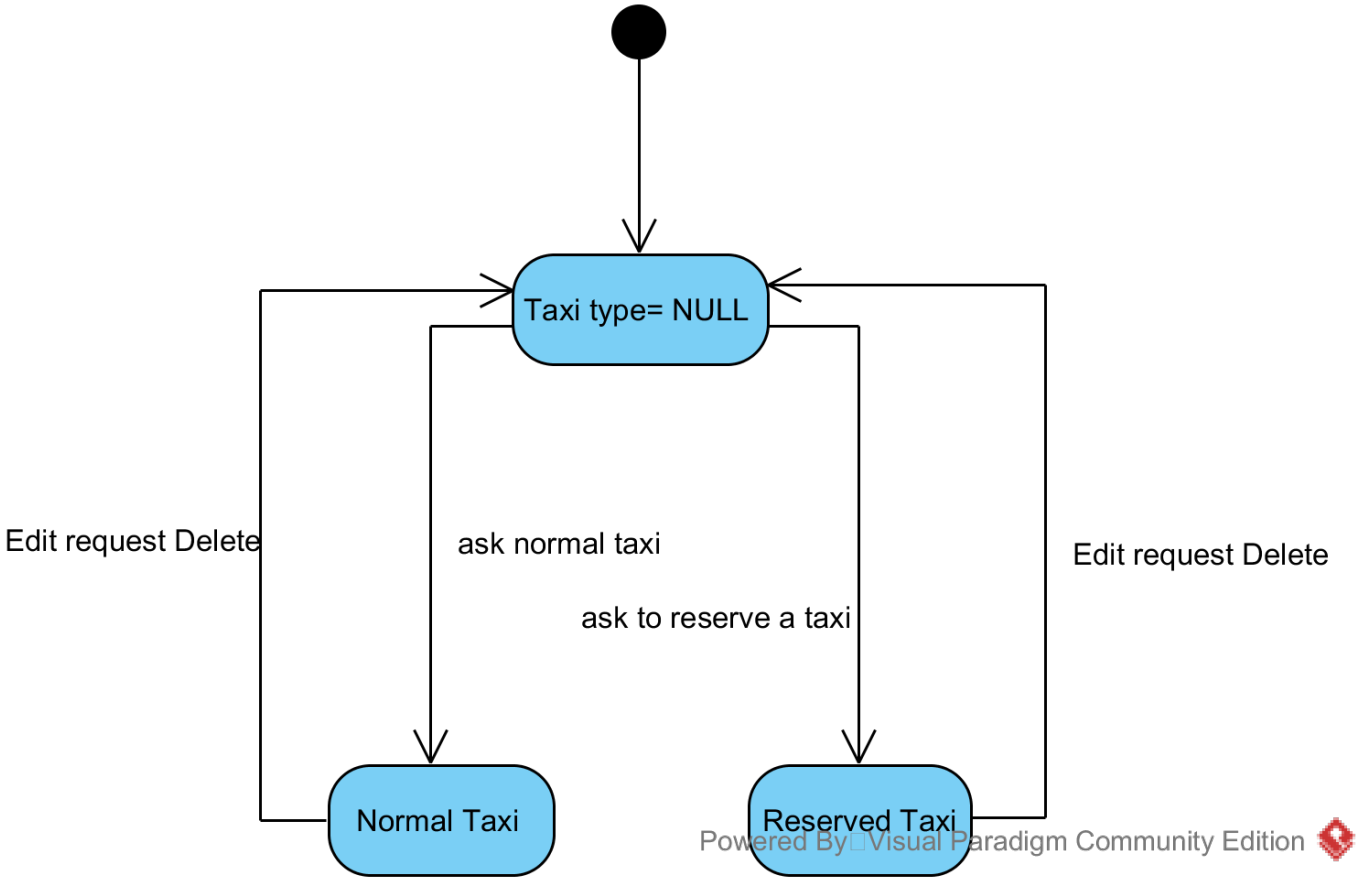
3.8.1. Actors

The following state diagram describes the login process: when a guest tries to sign up, the system automatically checks if his email address is stored in the database of taxi drivers' email and understands if the guest is a client or a taxi driver.



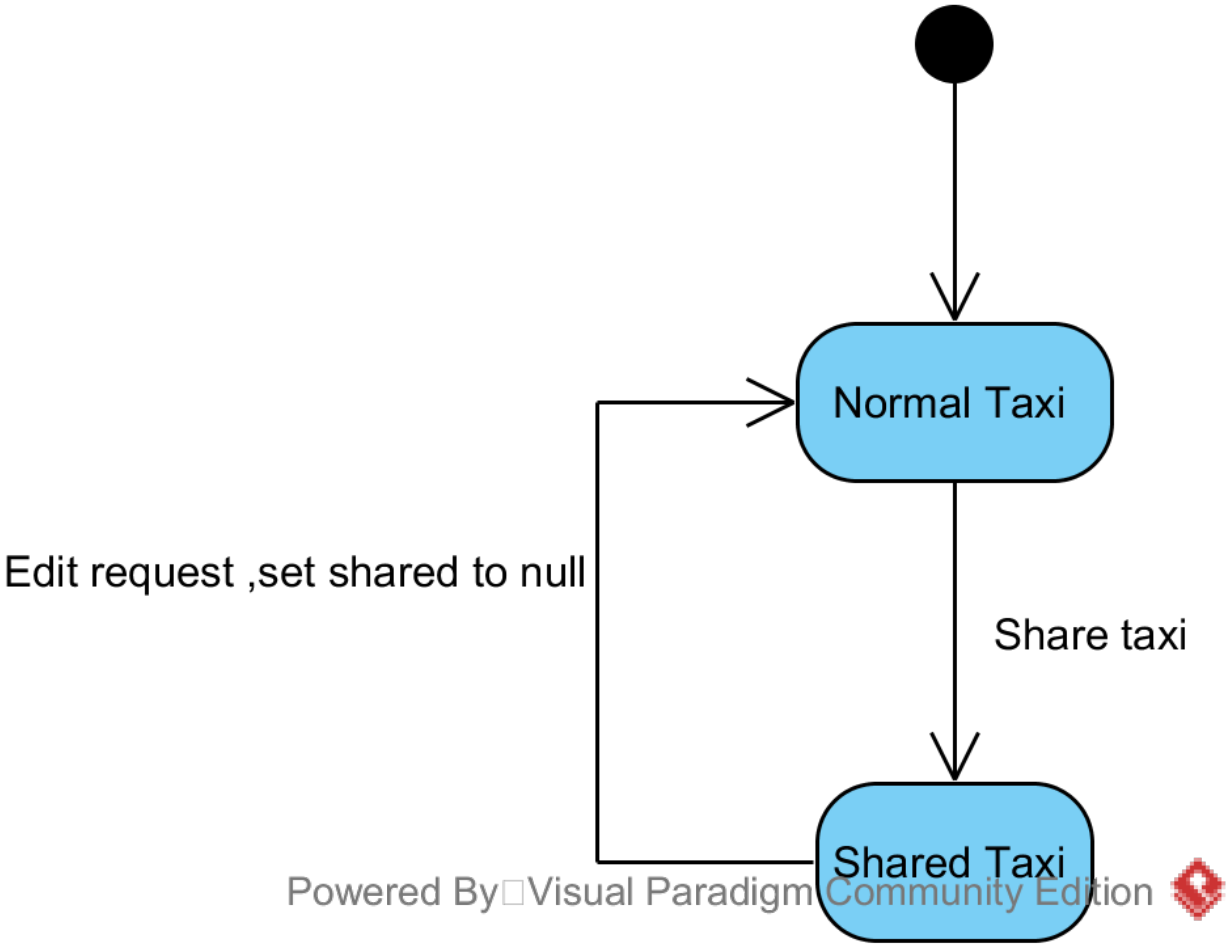
State machine diagram figure 3.1

3.8.2. Taxi Request/Reservation



State machine diagram figure 3.2

3.8.3. Taxi sharing request



State machine diagram figure 3.3

3.9. Alloy modeling

3.9.1. Code

```
open util/integer as integer

sig StreetName{}
sig Email{}
sig Password{}
sig Strings {}

sig GenericUser{
      emailAddress: one Email,
      password: one Password,
      connected: one DispatcherSystem
}

sig TaxiDriver extends GenericUser{
     receive: some TaxiDriverNotification,
       drive: one Taxi
}

sig Client  extends GenericUser{
    receive : some ClientNotification,
      asksFor:  one TaxiRequest,
      edit: one TaxiRequest
}

sig Notification {
      message: one Strings
}

sig TaxiDriverNotification extends Notification {}
sig ClientNotification extends Notification{}

sig DispatcherSystem {
      sends: some Notification,
      process: some TaxiRequest,
      contain:some Zone
}

sig Zone {
```

```
      composedOf: some Address,
      has: one TaxiQueue
}

sig TaxiQueue{
      composedOf: some Taxi,
}

sig Taxi {
      taxiCode: one Int ,
}{taxiCode>0}

sig TaxiRequest{
      contain: one Ride,
      satisfiedBy:one Taxi
}

sig ReservationTaxiRequest  extends TaxiRequest{}
sig NormalTaxiRequest  extends TaxiRequest{}

sig Ride {
       origin: one Address,
       destination : one Address
}{origin!=destination}

sig SimpleRide extends Ride{
}

sig ComposedRide extends Ride{
            intermediatePoints: some Address
}

sig Address{
     name : one StreetName
}

//-------------------------------------------------------------------------

//There must be only one dispatcher system
fact System{
      #DispatcherSystem = 1
}
```

```
//No repeated e-mails
fact noRepeatedEmail{
      no disj g1,g2 : GenericUser| (g1.emailAddress= g2.emailAddress)
}

fact TaxiDriverTaxi{
      //Every taxi must correspons to a exactly one taxi driver
      no t:Taxi , td1,td2:TaxiDriver |td1!=td2 and td1.drive=t and
      td2.drive=t

      //The number of taxis must be equal to the number of taxi drivers
      #Taxi = #TaxiDriver
}

fact TaxiQueue{
      //There cannot be a taxi in two different queues
      no t:Taxi , td1,td2:TaxiQueue | td1!=td2 and td1.composedOf=t and
      td2.composedOf=t

      //There must be at least one taxi for each queue
      all q: TaxiQueue | #q.composedOf >=1

      //all taxis has a queue
      all t:Taxi {one tq:TaxiQueue | t in tq.composedOf}
}

fact QueueZone{
      //The number of taxi queues must be equal to the number of zones
      #TaxiQueue = #Zone

      //one queue cannt be in two different zones
      no t:TaxiQueue , td1,td2:Zone | td1!=td2 and td1.has=t and
      td2.has=t
}

fact TaxiDriverNotification{
      //Every TaxiDriverNotification must correspons to a exactly one
taxi driver
      no t:TaxiDriverNotification , td1,td2:TaxiDriver |    td1!=td2 and
      td1.receive=t and td2.receive=t
```

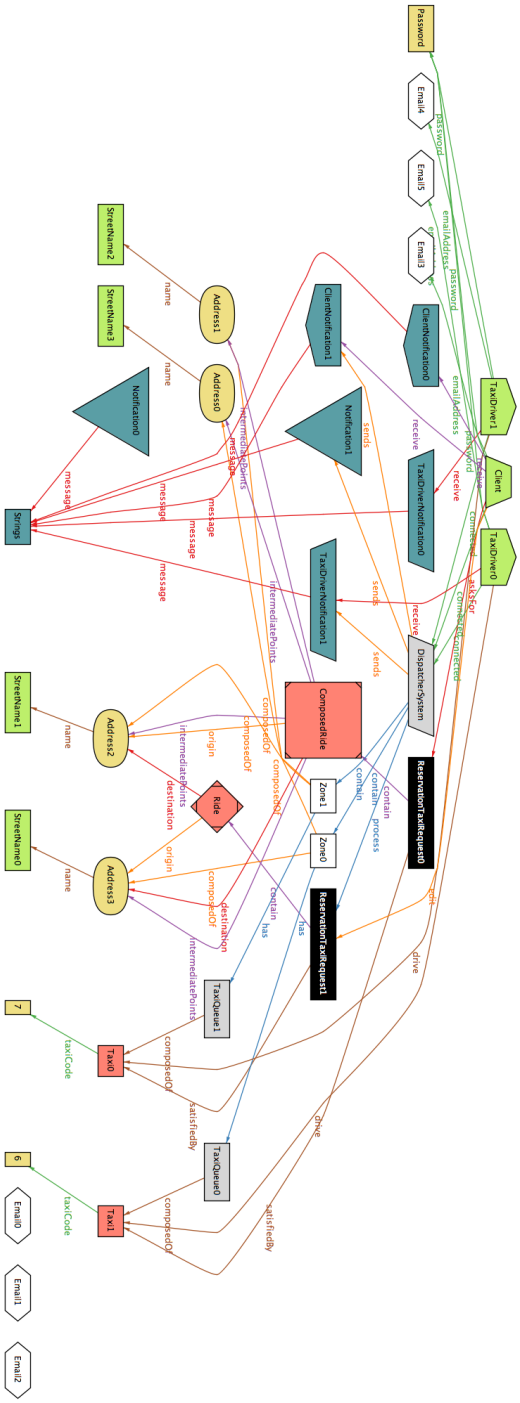
```
	//No taxi notification without taxi
	all tn:TaxiDriverNotification {one td:TaxiDriver | tn in
	td.receive}
}

//Notifications must not be empty
fact NoEmptyNotification {
	 all n : Notification |(#n.message>=1)
}

//no street name without address
fact noStreetNamewithoutAddress {
	 all sn:StreetName {one a:Address | sn in a.name}
}

fact TaxiRequest{
	//no taxi request without client
	all tr:TaxiRequest{one c:Client | tr in c.asksFor or tr in c.edit}

	//no taxi request without taxi
	all t:Taxi{one tr:TaxiRequest | t in tr.satisfiedBy or t in
	tr.satisfiedBy}
}

fact ClientNotification{
	//Every TaxiDriverNotification must correspons to a exactly one
	taxi driver
	no t:ClientNotification , td1,td2:Client | td1!=td2 and
	td1.receive=t and td2.receive=t

	//No taxi notification without taxi
	all tn:ClientNotification {one td:Client | tn in td.receive}
}

fact Ride{
	//no raid without taxirequest
	all r:Ride{one tr:TaxiRequest | r in tr.contain or r in tr.contain}

	//one ride one taxirequest
	no r:Ride , tr1,tr2:TaxiRequest |     tr1!=tr2 and tr1.contain=r and
	tr2.contain=r
}
```

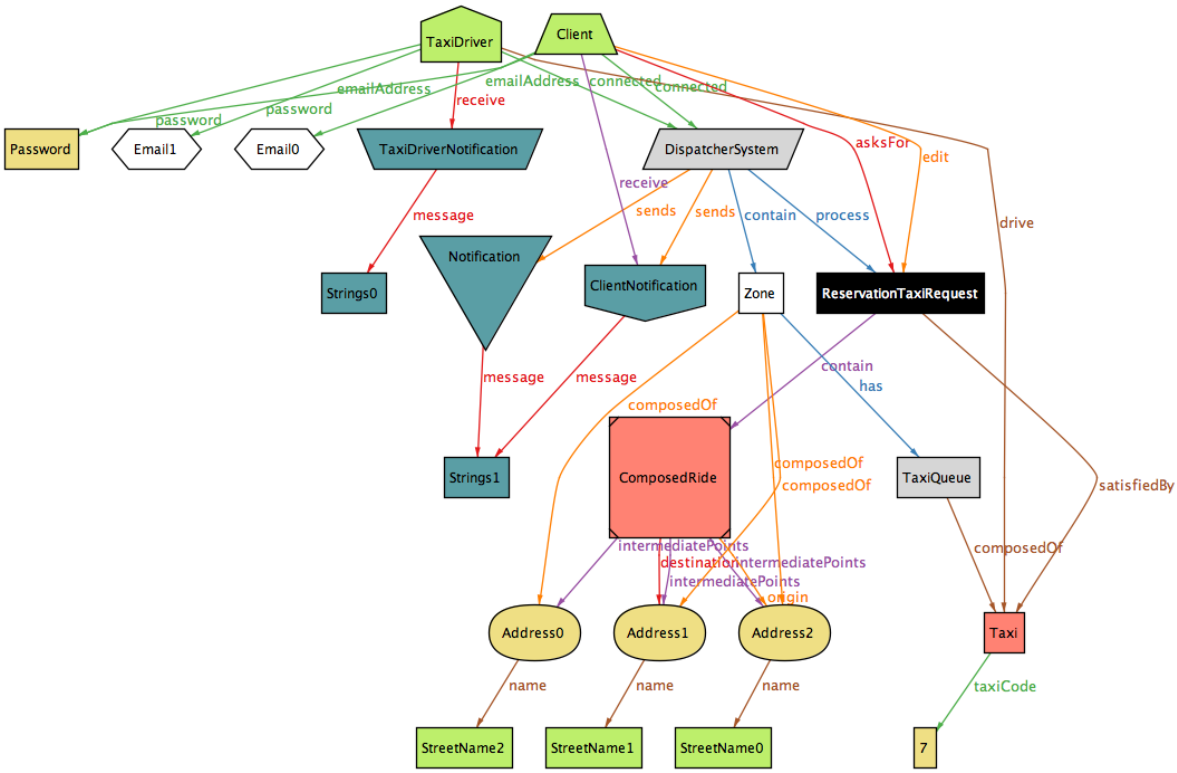
```
//There must be at least one taxi driver
fact numTaxiDrivers{
      #TaxiDriver >= 1}

//There must be at least one client
fact numClient{
      #Client >= 1}

//All addresses must be different from each other
fact noDoubleAddress{
      no disj a1,a2: Address | (a1.name = a2.name)
}

//------------------------------------------------------------------------

pred show(){
#Client<6
}

run show for 6
```

3.9.2. Report of Alloy Analyzer

```
Executing "Run show for 6"
  Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20
  12760 vars. 978 primary vars. 23025 clauses. 856ms.
  Instance found. Predicate is consistent. 311ms.
```

3.9.3. Generated worlds




TaxiDriver
Client
Password
Email1
Email0
TaxiDriverNotification
DispatcherSystem
emailAddress
password
password
emailAddress
receive
connected
connected
asksFor
edit
receive
sends
sends
contain
process
drive
message
Notification
Strings0
ClientNotification
Zone
ReservationTaxiRequest
message
message
contain
has
composedOf
Strings1
ComposedRide
composedOf
composedOf
TaxiQueue
satisfiedBy
intermediatePoints
destination
intermediatePoints
intermediatePoints
origin
composedOf
Address0
Address1
Address2
Taxi
name
name
name
taxiCode
StreetName2
StreetName1
StreetName0
7

4. Used tools

We have used the following tools in order to create the RASD document:

- Microsoft Word 2016 for Mac and Windows: to write and format the document;
- Moqups: to create UI mockups (https://moqups.com)
- Signavio Academic: to create the class diagram and use case diagrams;
- Visual Paradigm 12.2: to create sequence diagrams and state machine diagrams;
- Allow Analyzer 4.2: to write the Alloy model and format the generated worlds.

5. Hours of work

This section presents the hours of work for each group component.

- Elis Bardhi: 29 hours
- Andrea Cavalli: 25 hours
- Mario Dolci: 24hours