mytaxy

Requirement Analysis Specification Document

Matteo Maria Fusi, Matteo Locatelli

Contents

1 Document Overview

1.1 Document Purpose

This document is the Requirements Analysis Specification Document (RASD) for mytaxi, the system that must be developed. The purpose of this document is to describe the requirements of this system (both functional and non functional), to identify its main goals and the users that will mainly interact with it, especially its stakeholders. This document will also analyze the properties of the domain in which the system will be used and the constraints that have to be respected. The main application scenarios will be analyzed to clearly explain how the system works. This document is intended for developers and other people that will work on this system and need to know the fundamental information about it.

1.2 Document Structure

Section 1 explains what a Requirements Analysis Specification Document (RASD) is and how this paper is structured. Part 2 contains a little introduction of mytaxy service and general information such as the scope of the service, identified stakeholders and the terminology used. The Section 3 is composed by the analysis of the requirements, from the domain properties to functional and non functional requirements. In Section 4 some common scenarios will be proposed to the reader, then use cases will described. The Section 5 focuses on the exposure of sequence diagrams and the class diagram. Section 6 introduces the alloy model proposed in the attachement. Section 7 contains some mockups of a future implementation of the application.
Appendix A contains links to the software used.
Appendix B contains the amount of time spent for the creation of this document.

1.3 Document version

Current version is 1.0 .

2 The mytaxi service

2.1 Scope

The purpose of the project is to develope a web-based service that manages the taxi service. Using their smartphone, or a web-app, users can request a taxi ride in a simple way. Also taxi drivers will use their smartphone to accept or reject requests of performance.

2.1.1 Goals

The main goals of the proposed service will be:

1. Allow customers to make a reservation.
2. Allow customers to make a request.
3. Allow customers to use the system via the web-app version.
4. Allow customers to use the system via the smartphone app version.
5. Guarantee a fair management of taxi queues for each taxi zone.
6. Allow taxi drivers to use a mobile application to access to the system.
7. Allow taxi drivers to accept a request.
8. Allow taxi drivers to refuse a request.
9. The service must be extendible.

2.2 Stakeholders Identification

The financial stakeholder of this project is probably the taxi company, that wants to use a system to improve its service in order to satisfy its customers and to increase the number of people who use its taxi. Another possible stakeholder is the government of the city, that wants to employ mytaxi service for the same reasons explained above. The only difference is that the taxis are managed by the government and not by a private company. Citizens or tourists of a city can also be stakeholders.

2.3 Glossary

Customer/Passenger: a citizen who requires a taxi ride using the service.

Confirmation: after a request is taken in charge by a taxi driver, the system generates a confirmation that will be sent to the customer. The confirmation informs the customer about the waiting time and the taxi code.

mytaxy: is the service proposed in this document.

Performance: (Taxi related) in this paper is synonym of the term Ride.

Request: when a customer requests for a ride to the system, this will generate a request that will be forwarded to a taxi driver, containing information about the customer (such as his/her name and surname provided during the registration to the system, etc.) and ride origin source.

Reservation: when a customer wants to book a ride, the system will generate a reservation instance. The reservation must be done at least 2 hours before the ride, but the request is processed by the system only 10 minutes before the performance takes place. Despite of the request, the customer must also provide to the system the destination of the ride

Ride: performance made by a taxi driver that take a customer from a place to the desired destination.

Driver: (taxi related) someone who drives, and also usually owns a taxi. He will use the service to find customers.

Zone: a zone is a section which the city is divided in. Zones are 2 km^2 wide.

3 Requirement Analysis

3.1 Domain Properties

1. The City is divided in taxi zones.
2. Zones are 2 km^2 wide.
3. The position of a taxi is known by receiving GPS coordinates from the smartphone that the taxi driver owns.
4. A taxi is associated to a zone if its location is inside the zone's boundaries.
5. Costumers don't lie about a request made, which means they are present at the location submitted when the taxi arrives.
6. Costumers use real information for making requests.

3.2 Assumptions

This section will clarify some points that aren't explained in the specification document.

1. Taxi drivers are inserted in the system from the beginning. They don't need to make any kind of registration to the service. Credentials needed for log into the app are provided by sending an e-mail to them.
2. A taxi driver can only drive the taxi associated to him/her.
3. Every taxi driver should have installed the mytaxi application on their smartphone in order to detect the vehicle position and, so, assign him to a zone of the city.
4. We assume that taxi drivers will accept or refuse a request in an admissible time such as 1 minute.
5. We assume that the location where the service will be installed and set up is near the city in which it will be exploited.
6. A taxi can arrive at the origin of a reservation with a tolerance of 5 minutes, this mean 5 minutes before or after the scheduled time.

3.3 Functional Requirements

3.3.1 Overview

Passengers can use an app installed on their phone or a web-app to request or reserve a taxi ride. Taxi drivers will also use the same app to inform the system about their availability and to confirm requests made by customers. A customer can send a request for a performance to the system; the system will forward the request to the first taxi available. The taxi driver can accept or refuse the request: in the case of rejection the system will contact the second available taxi and so on. When the request is accepted by a driver, the customer is notified with the waiting time and the taxi code. If a customer wants to book a ride, he can also make a reservation. If there is no available taxi in the zone, the system will try to contact the three nearest taxis in the city outside the inquired zone.

3.3.2 Analysis

1. To allow customers to make a request.
   - (a) A customer can make a request specifying his/her name and the origin of the ride.
     - i. If the customer is using the smartphone app, he can also specify the origin of the ride using the GPS.
   - (b) When the request is accepted by a taxi driver a confirm message will sent to the customer.
     - i. If an available taxi is found, the customer is notified with the code of the taxi and the estimated waiting time.
     - ii. If the available taxi is in a different zone from the one in which the customer is, when the passenger receives the confirmation he can decide to accept or refuse the ride.
2. To allow customers to make a reservation.
   - (a) A customer can make a reservation specifying the name, the origin and the destination of the ride and the hour he wants to make it.
   - (b) If the customer is using the smartphone app, he can also specify the origin of the ride using the GPS.
   - (c) The system will confirm the reservation to the customer.
   - (d) The system will automatically make a request 10 minutes before the time chosen by the customer.
3. To allow customers to use the system via web-app version.
   - (a) A customer can use the web-app version of the system which will allow him to access to every functionality of the system.

4. A customer can use the smartphone app version of the system which will allow him to access to every functionality of the system.

5. To guarantee a fair management of taxi queues for each taxi zone.

   (a) When a taxi enters a zone, if it's available, it's added to the queue of that zone.
   (b) When a taxi exits a zone, it is removed from the queue of that zone.
   (c) If a taxi driver refuses a request from a customer, the taxi is moved to the last place of the queue.
   (d) If there isn't any available taxi in the customer's zone or all the taxis in it refuse his request, the request will be forwarded to taxis in other zones.

6. Allow taxi drivers to use a mobile application to access to the system.

   (a) Taxi drivers are given a code and a password, thanks to them they can log into the system.
   (b) Using the app, taxi drivers can choose to appear available to make a ride.
   (c) Taxi drivers can also decide to appear unavailable.

7. Allow taxi drivers to accept a request.

   (a) Taxi drivers can accept a request from a customer.

8. Allow taxi drivers to refuse a request.

   (a) Taxi drivers can refuse a request from a customer.

9. The service must be extendible.

   (a) To guarantee programmatic interfaces to enable the development of additional services.

3.4 Non Functional Requirements

1. **Same UX**: The UX should be the same in every system in which the application is available. For example, the user interface should be the same in IOS and Android apps. A similar UX should be reproduced on the web-app if used by a desktop computer.

2. **Acceptable performance**: The system must maintain a reasonable time to manage information. Which means, the time of processing a request made by a customer (the time elapsed from the submission to the forwarding of the requesto to a taxi driver) should resolve in no more than 30 seconds. The same amount of time must not pass between the acceptance or rejection of the request and the arrival of the confirmation message to the customer.

3. **Freeware**: The application must be free to improve its spread.

3.5 Constraints

1. Internet connection is required for accessing the service.
2. The reservation must be made at least 2 hours before the time of the ride.
3. If the customer is using the system with the web-app version, the origin of the ride must be inserted manually because desktop PCs usually don't have a GPS system.
4. The GPS of the smartphone used by taxi drivers must be always turned on, in this way the system always is informed about the position of the taxis.

3.5.1 Functional constraints

1. All the rides must be stored by the system.
2. A taxi cannot make multiple rides at the same time. So, the taxi driver cannot accept other requests while he/she's performing a ride.

3.5.2 Legal Constraints

- Taxis and taxi drivers are subjected to the actual law in the country where mytaxi is installed and used.
- Privacy of users is guaranteed by the law in the country where mytaxi is installed and used.

3.5.3 Software Constraints

- Software design of mytaxi will be affected by the language choosen for the application development.

4 Use Case Analysis

4.1 Scenarios

4.1.1 Scenario 1

Marcello is a citizen who wants to request a taxi ride to go to work. He is at home, so he turns on his computer and opens the mytaxi website to call the ride. In order to complete his request, Marcello fills in the form with his name and address. As soon as the system has received Marcello's request, it starts looking for a taxi in the same zone of the customer's house. An available taxi has been found in that zone, so the request is sent to the smartphone of the driver who is driving that taxi. The taxi driver decides to accept Marcello's request, so a confirmation message is sent to the customer with the taxi code and the estimated waiting time. Marcello receives the confirmation and waits for the taxi. When the taxi arrives to Marcello's position, the taxi driver checks that Marcello is the person who made the request and then use his app to register the start of the ride, this also makes the taxi appear unavailable. Marcello tells the driver his destination and the driver takes him there. At the end of the ride, the customer pays the taxi driver, who then uses again his mytaxi app to register the end of the ride (and appear available again). In this way the ride will be registered and the taxi driver will find it when he checks all the rides that he has made.

4.1.2 Scenario 2

Oscar is really tired, he always gets lost in the huge city he lives in. He decides to request a taxi ride using the mytaxi service, so he downloads the app on his smartphone and fills the form for the request: he inserts his name e uses the GPS system to detect his position. The system finds a taxi in the same zone where Oscar is, but the taxi driver is having a break, so he decides to refuse the request. The request is then forwarded to the next taxi in the queue and the driver accepts the call. The taxi driver makes himself appear as unavailable and then takes Oscar to his destination, who pays for the service and thanks kindly the driver for saving him.

4.1.3 Scenario 3

Gwyn is a foreign traveler who arrives to Milan for a city tour. He decides to request a ride to go back to the hotel, after spending the whole day walking around. Gwyn downloads the mytaxi app and fills the request form submitting is name and his position (he doesn't use the GPS because his smartphone is low on battery). Unfortunately, there isn't any free taxi in Gwyn's zone, so the system forwards his request to the nearest taxi in a different zone. The taxi driver accepts the request, so a confirmation message is sent to the customer. When Gwyn receives the waiting time he gets very angry, because he has to wait for more than an hour. He decides to refuse the ride and visit Milan a little more, waiting for a closer taxi the be avaialble. After half an hour, Gwyn tries again to make a request and he's lucky this time because there's a free taxi in his zone. He submits all the needed information so he can make the ride and reach his hotel.

4.1.4 Scenario 4

Jack wants to request a taxi ride, because he has to go to university but there is a strike and the public transport isn't an option. He makes a request using the app installed on his smartphone, but the system can't find any available taxi inside Jack's zone. The request is sent to the nearest taxi, but the taxi driver refuses the request because he's stuck in traffic; so the request is forwarded two more times but the two drivers refuse Jack's call because of the same problem. Jack is notified that the service is unavailable for the moment, so he has to decide either not to go to university or wait a little to make another request and hope that there's some taxi available.

4.1.5 Scenario 5

Bob wants to go to visit a friend living in Bob's town but far away from his house. Bob decides to take a taxi, then uses the mytaxi web-app to request a ride. There are three taxi available in Bob's zone, but all of the taxi drivers refuse the call of the customer (such things occures very rarely). The request is forwarded to the nearest available taxi, the driver accept the request and Bob decides to take that taxi because he doesn't have to wait much time. The driver takes Bob to his friend's house and Bob pays the driver for the service.

4.1.6 Scenario 6

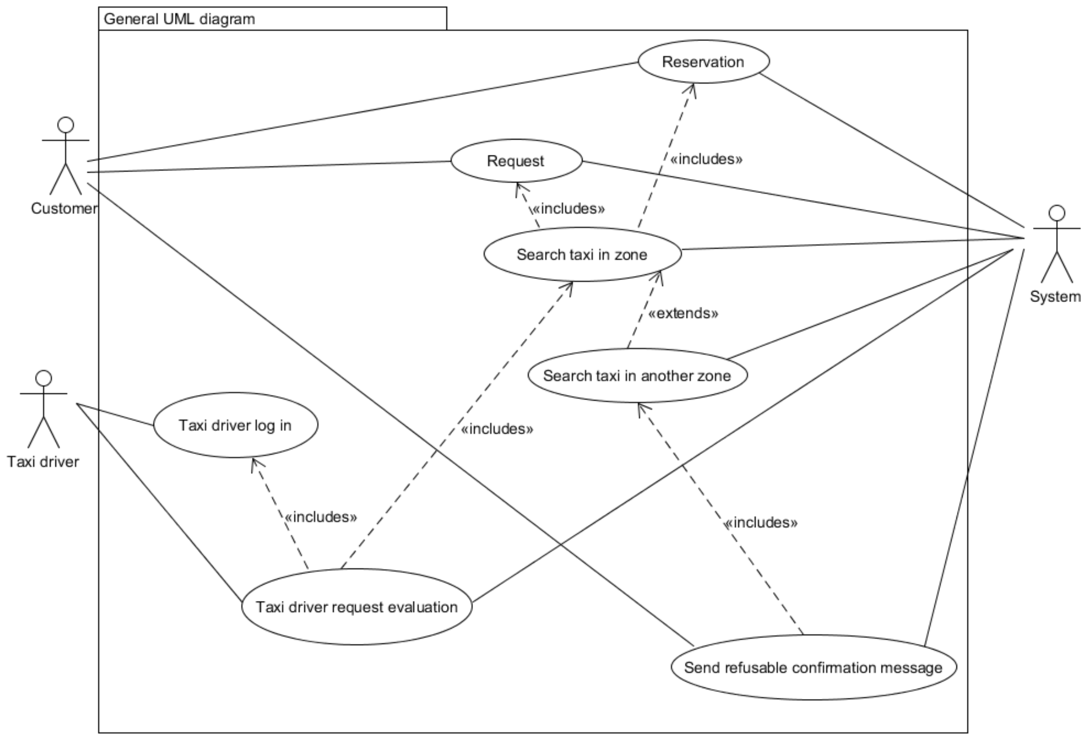
Lautrec wants to book a request in the city of Lordran, so he decides to use the mytaxi app that is already installed on his smartphone. It's 4 pm and Lautrec wants to make a reservation for 5 pm but, because reservations can only be made at least two hours before the ride, the system notifies him with an error. Lautrec then decides to postpone is ride and books a ride for 6.30 pm, filling the form with the hour, his name, the origin and the destination. 10 minutes before 6.30 pm, the system makes a request for Lautrec's ride and a taxi is found in the same zone of the starting point. At 6.30 pm Lautrec is at the decided place for the start of the ride and the taxi driver takes him to his destination.

4.2 Actors

- **Customer**. He/She's the one who exploit the taxi service. We can say that he/she is the end user. He can use the application for request or reserve a taxi ride. He/She interacts at first with the system when he requests for a ride, then with the taxi driver during the performance.
- **Taxi driver**. He/She has the duty of carrying one customer from the origin of the ride to the destination. He/She can accept or refuse a ride offer sent by the system.
- **System**. It manages the various requests and reservations made by customers and establishes the connection between customers and taxi drivers.

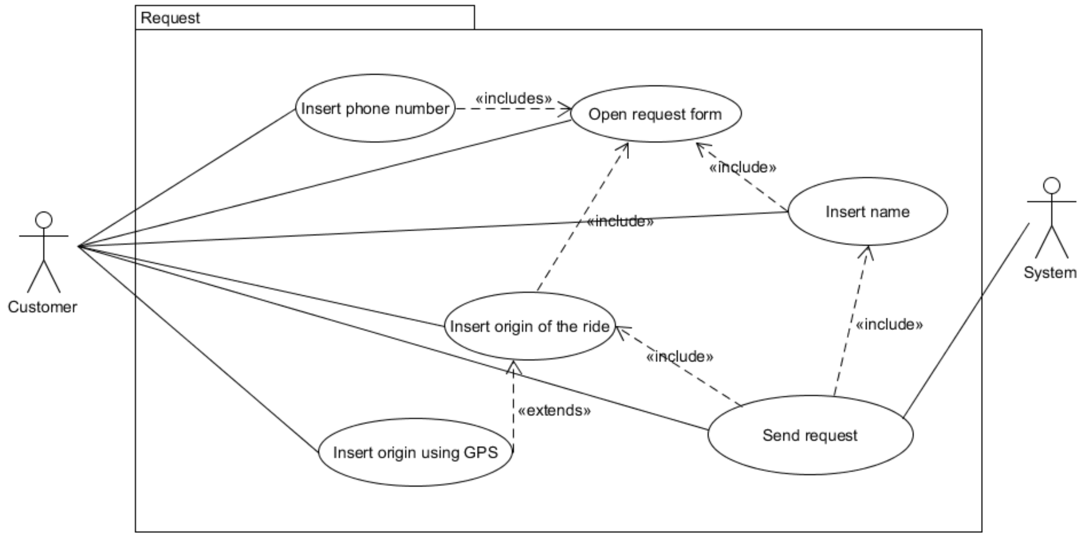
4.3 Use cases

4.3.1 General use case diagram



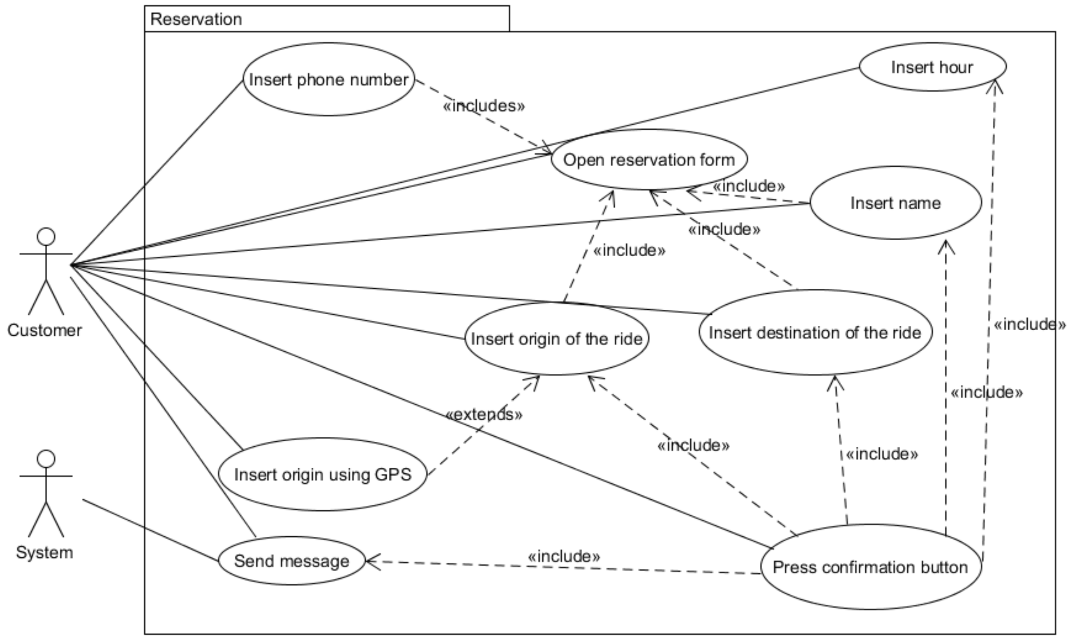
4.3.2 Request

Actor	Customer
Goal	Goal number 1
Input condition	The customer is using the web-app or the smartphone app version of the system.
Event flow	1. The customer, once he has opened the application, open the form to make a request. 2. The customer inserts his name into the form. 3. The customer inserts the origin of the ride into the form, manually or using the GPS system if he's using the app on his smartphone. 4. The customer presses the confirm button.
Output condition	The request has been done correctly and the system starts looking for an available taxi.
Exception	None.



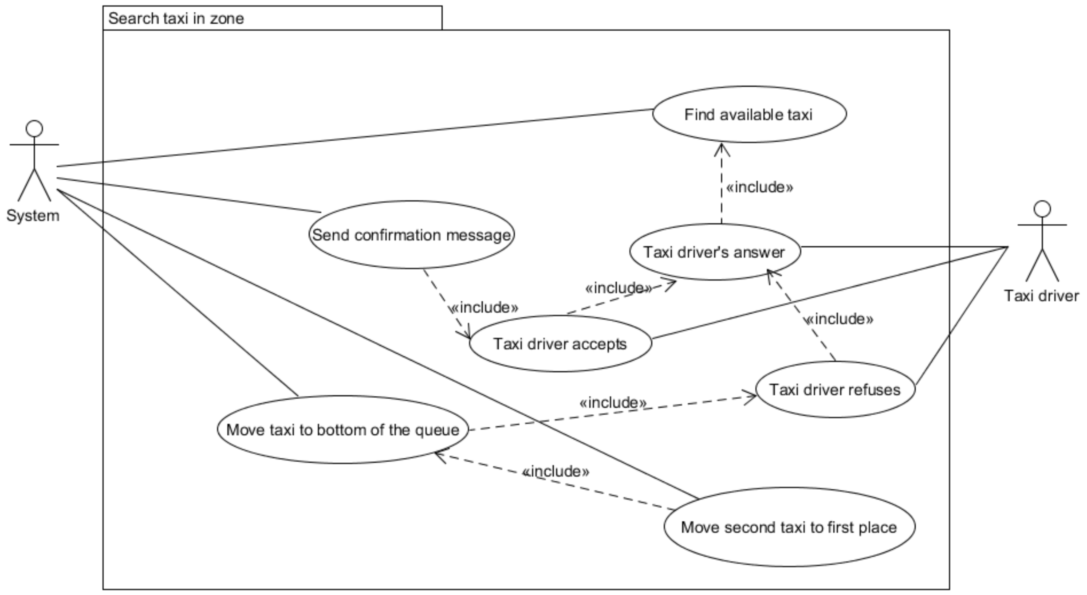
4.3.3 Reservation

Actor	Customer, system
Goal	Goal number 2
Input condition	The customer is using the web-app or the smartphone app version of the system.
Event flow	1. The customer, once he has opened the application, open the form to make a reservation. 2. The customer inserts his name into the form. 3. The customer inserts the origin of the ride into the form, manually or using the GPS system if he's using the app on his smartphone. 4. The customer inserts the destination of the ride into the form. 5. The customer inserts the hour at which he wants to make the ride. 6. The customer presses the confirm button. 7. The system will notify the customer with a message if the reservation has been done correctly.
Output condition	Ten minutes before the time of the reserved ride, the system will look for an available taxi for the customer.
Exception	The customer has to make the reservation at least two house before the time of te ride, if this condition isn't respected the customer will be notified with an error.


Reservation
Customer
System
Insert phone number
«includes»
Open reservation form
Insert hour
Insert name
«include»
«include»
«include»
Insert origin of the ride
Insert destination of the ride
«include»
«include»
«extends»
«include»
«include»
Insert origin using GPS
Send message
«include»
Press confirmation button

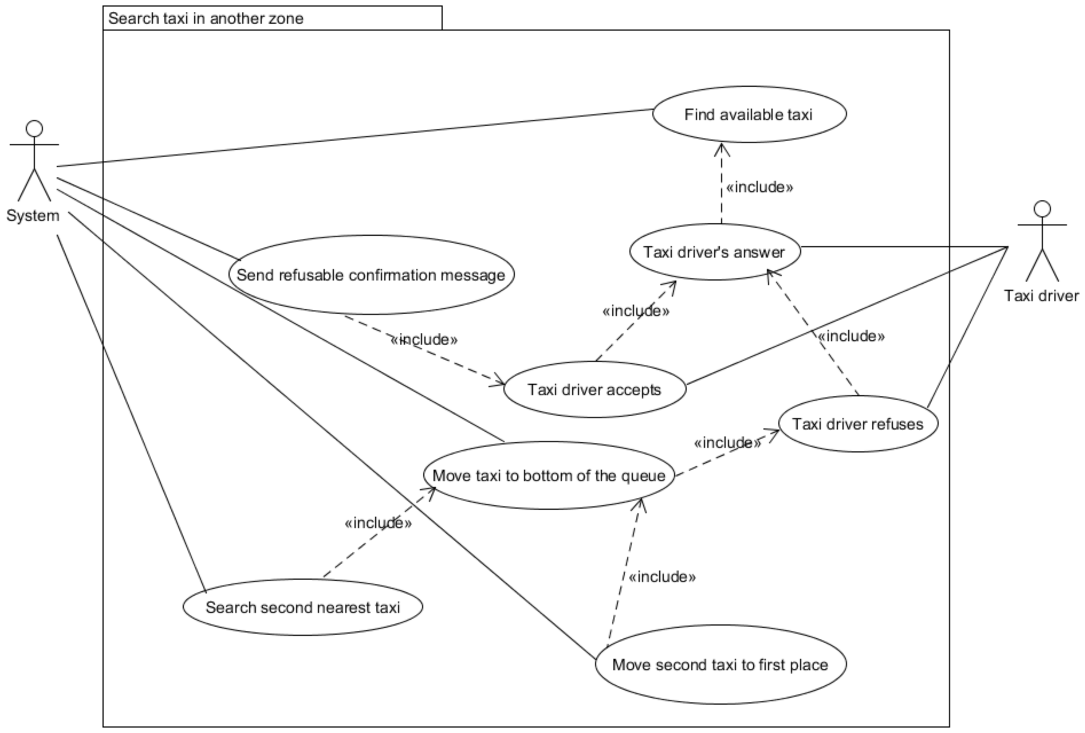
4.3.4 Search taxi in zone

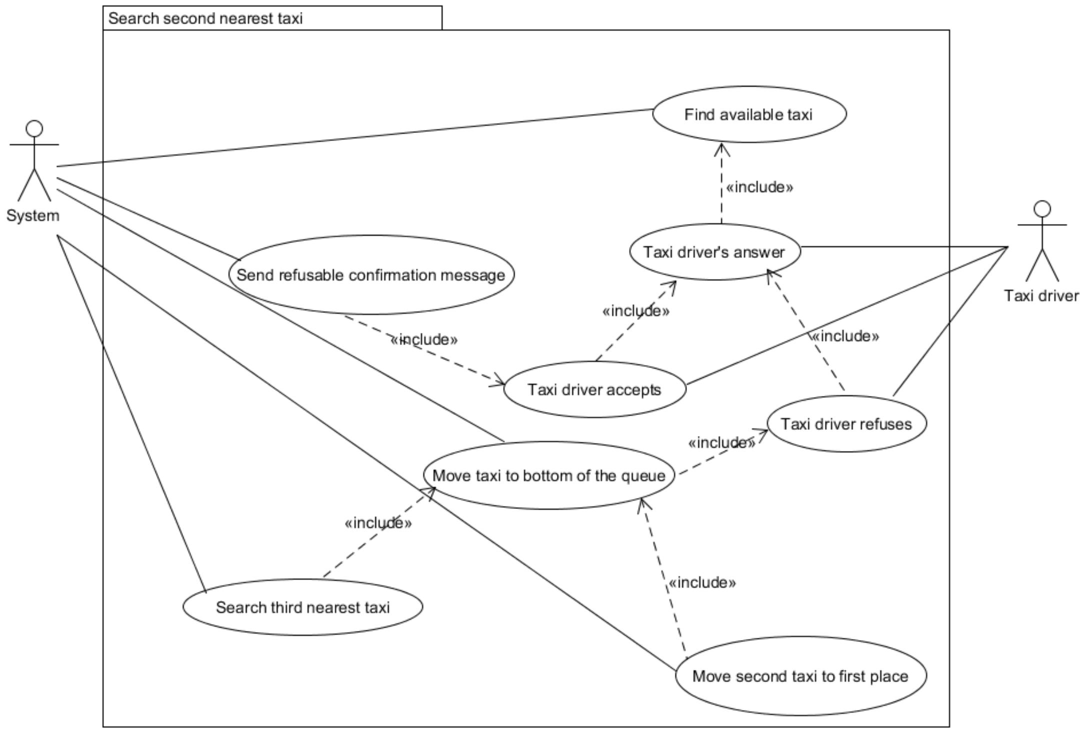
Actor	System, Taxi driver
Goal	Goal number 5, 7 and 8
Input condition	The customer has made a request or a reservation (in that case the system does this procedure autmatically ten minutes before the time of the ride).
Event flow	1. The system looks for an available taxi inside the customer's zone. 2. If there's an available taxi, the system sends to the taxi driver the request for the ride. 3. If the taxi driver refuses the request, his taxi is moved to the bottom of the queue and the request is forwarded to the next taxi in the queue (that is now at the first place of the queue). 4. When a taxi driver accepts the request, a confirmation message is sent to the customer with the code of the taxi and the waiting time.
Output condition	1. The system finds a taxi in the customer's zone that can take the customer to his destination. 2. The system doesn't find any available taxi in the customer's zone. 3. Every available taxi driver in the customer's zone refuses the customer's request.
Exception	The taxi driver doesn't answer to the request in the within 1 minute. In that case it's like the driver has refused the request.

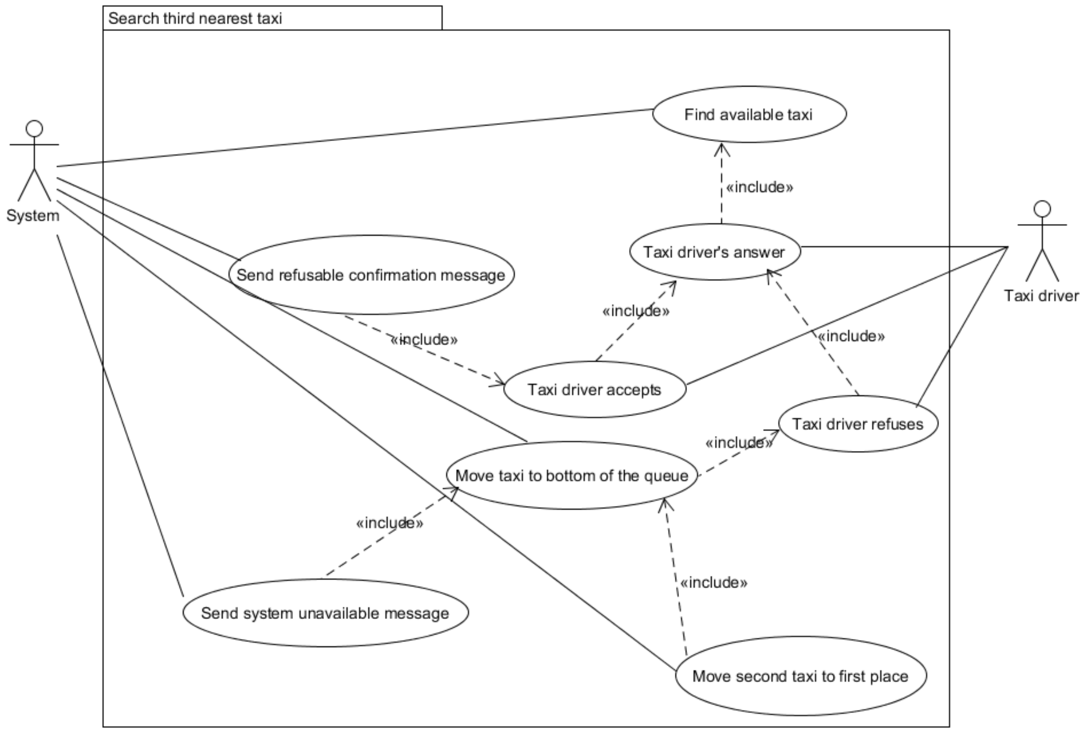

Search taxi in zone
System
Taxi driver
Find available taxi
«include»
Send confirmation message
Taxi driver's answer
«include»
«include»
«include»
Taxi driver accepts
Taxi driver refuses
«include»
Move taxi to bottom of the queue
«include»
Move second taxi to first place

4.3.5 Search taxi in another zone

Actor	System, Taxi driver
Goal	Goal number 5, 7 and 8
Input condition	The system has not found any available taxi in the customer's zone.
Event flow	1. The system forwards the customer's request to the nearest taxi located in a zone different from the one in which the customer is. 2. If the taxi driver refuses the call, it's moved to the last position of the queue it belongs to and the request is sent to the second nearest taxi to the customer's position. 3. If the second taxi driver refuses the call too, it's moved to the last position of the queue it belongs to and the request is sent to the third nearest taxi to the customer's position. 4. If the third taxi driver refuses the call too, it's moved to the last position of the queue it belongs to and the customer is notified that the system is temporarly anavailable. 5. If one of the taxi drivers accepts the request, the system sends to the customer a refusable confirmation message.
Output condition	1. The system finds a taxi in the that can take the customer to his destination. 2. The system doesn't find a taxi available to take the customer to his destination.
Exception	None.

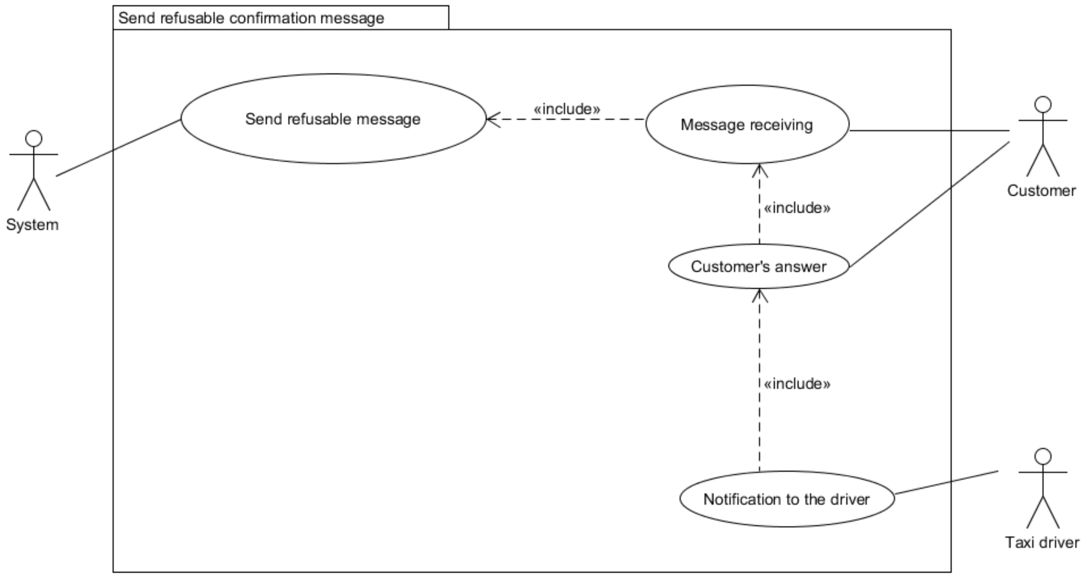

Search taxi in another zone
System
Taxi driver
Find available taxi
«include»
Taxi driver's answer
Send refusable confirmation message
«include»
«include»
«include»
Taxi driver accepts
Taxi driver refuses
«include»
Move taxi to bottom of the queue
«include»
«include»
Search second nearest taxi
Move second taxi to first place


Search second nearest taxi
System
Taxi driver
Find available taxi
«include»
Taxi driver's answer
Send refusable confirmation message
«include»
«include»
«include»
Taxi driver accepts
Taxi driver refuses
«include»
Move taxi to bottom of the queue
«include»
«include»
Search third nearest taxi
Move second taxi to first place


Search third nearest taxi
System
Taxi driver
Find available taxi
«include»
Taxi driver's answer
Send refusable confirmation message
«include»
«include»
«include»
Taxi driver accepts
Taxi driver refuses
«include»
Move taxi to bottom of the queue
«include»
«include»
Send system unavailable message
Move second taxi to first place

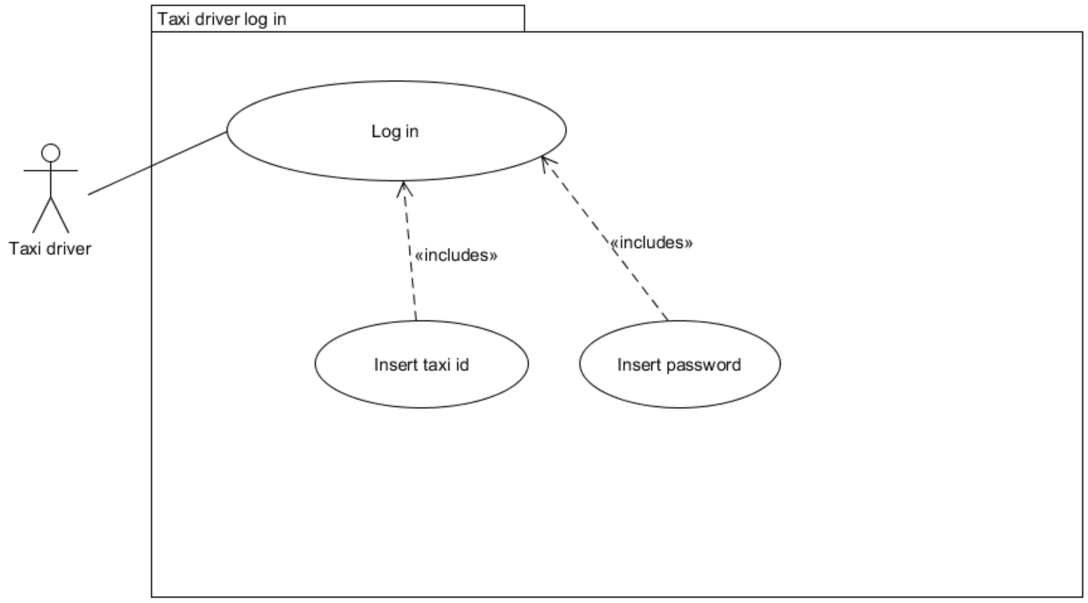
4.3.6 Send refusable confirmation message

Actor	System, customer
Goal	Goal number 1, 2 and 5
Input condition	The system has found an available taxi outside of the customer's zone and the driver has accepted the call.
Event flow	1. The system sends a message with the taxi code and the waiting time to the customer. 2. The customer decides to accept or refuse the ride. 3. The taxi driver is notified with the decision of the customer.
Output condition	1. If the customer accepts the ride, the taxi driver takes him to the desired destination. 2. If the customer refuses the ride, the request is cancelled.
Exception	None.



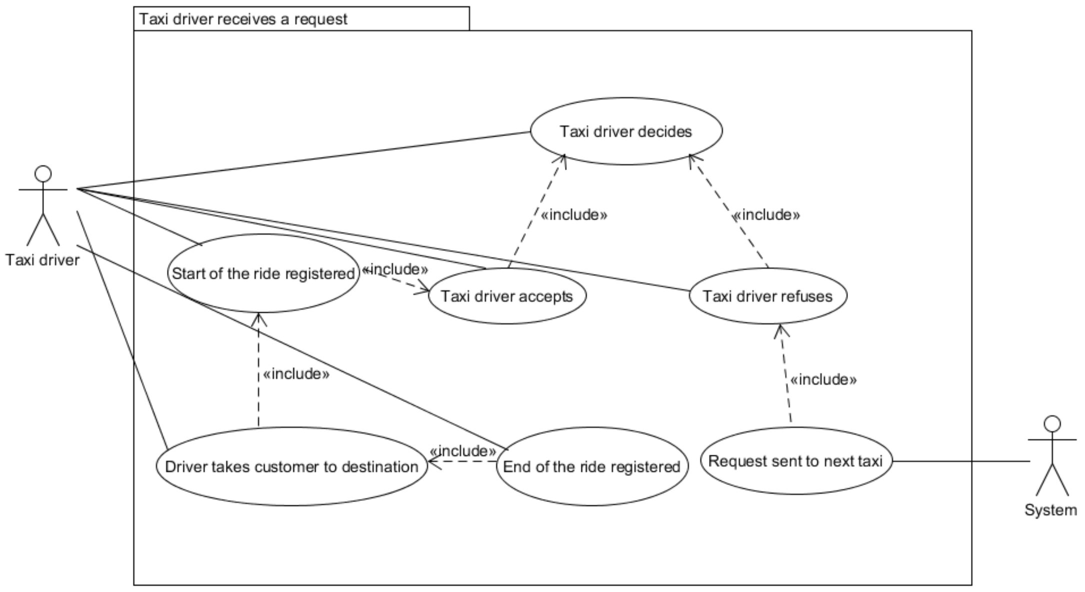
4.3.7 Taxi driver log in

Actor	Taxi driver
Goal	Goal number 6
Input condition	The taxi driver has the application installed on his smartphone.
Event flow	1. The taxi driver inserts the taxi id. 2. The taxi driver inserts his password.
Output condition	The taxi driver successfully logs into the system.
Exception	None.



4.3.8 Taxi driver receives a request

Actor	Taxi driver
Goal	Goal number 6, 7 and 8
Input condition	The system has sent to a taxi driver the request for a ride.
Event flow	1. The taxi driver can accept or refuse the customer's request. 2. The taxi driver can refuse the customer's request. 3. If the taxi driver accepts, when he reches the customer he registers the start of the ride. 4. The taxi driver takes the customer to the destination. 5. When the driver has taken the customer to his destination, he will register the end of the ride.
Output condition	1. If the taxi driver refuses, the request will be sent to the next taxi available by the system. 2. If the taxi driver accepts, the customer is taken to his destination.
Exception	None.


Taxi driver receives a request
Taxi driver decides
«include»
«include»
Taxi driver
Start of the ride registered
«include»
Taxi driver accepts
Taxi driver refuses
«include»
«include»
Driver takes customer to destination
«include»
End of the ride registered
Request sent to next taxi
System

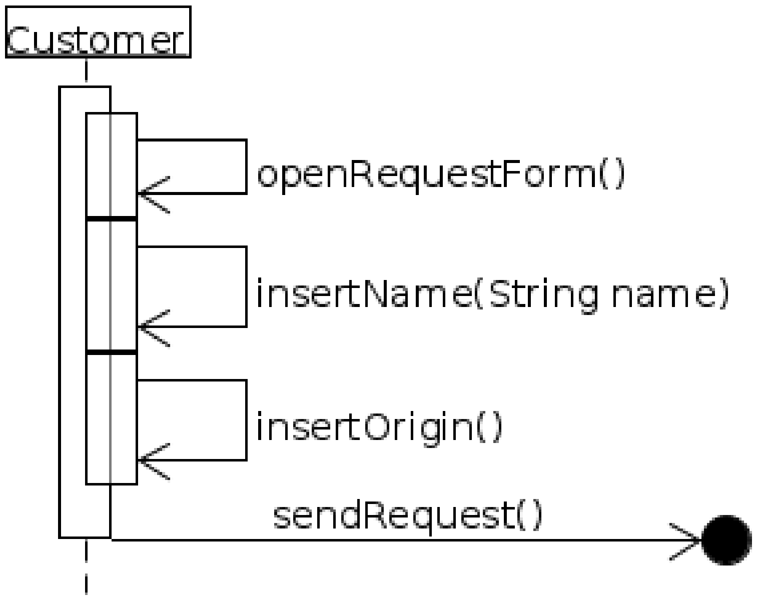
5 Other UML diagrams

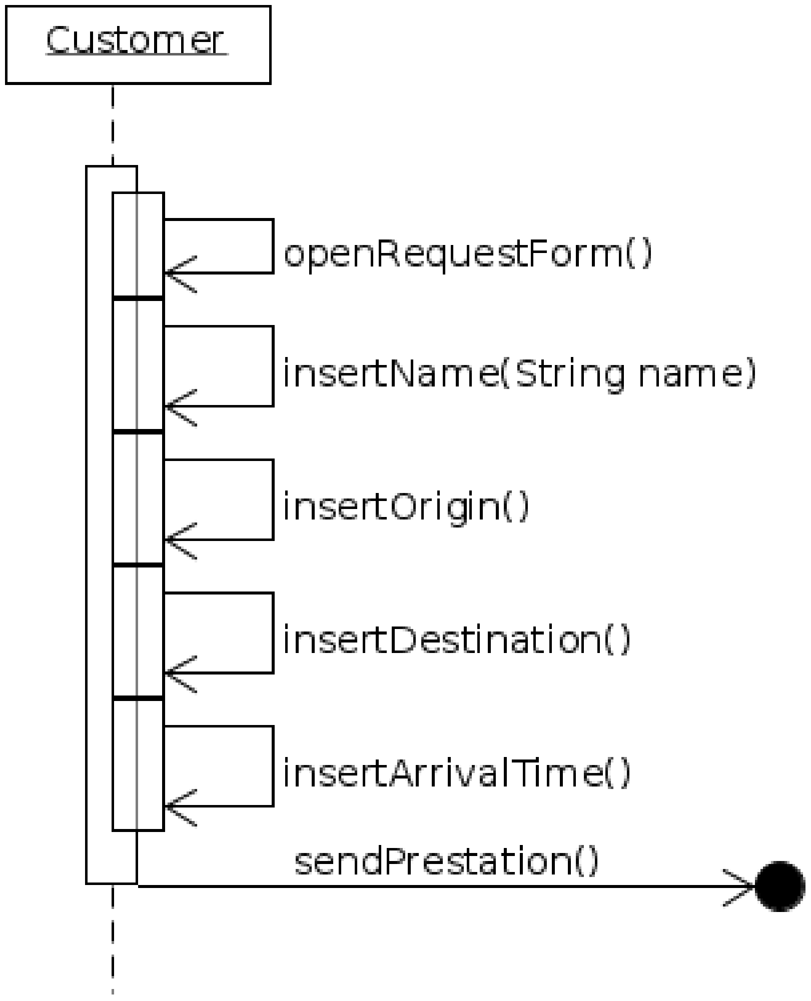
5.1 Sequence diagrams

For clarifying the interactions between actors in use cases will be proposed sequenced diagrams.

5.1.1 Customer operations

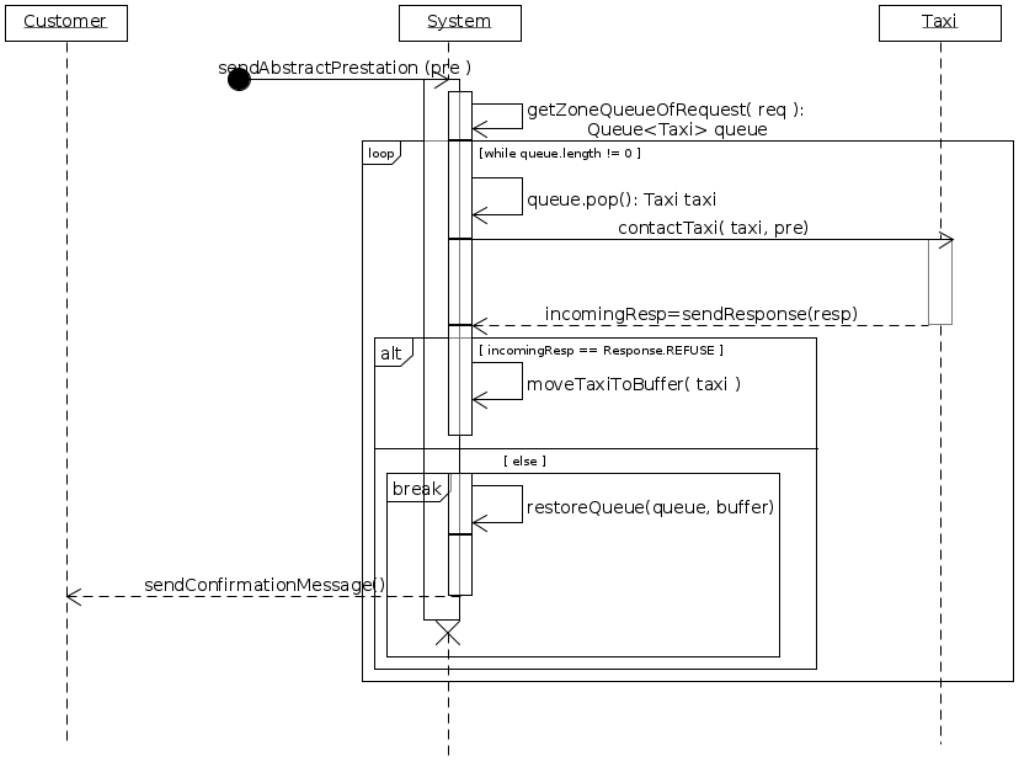
The first diagram regards the request for a ride made by a customer, the second explains how the reservation process must be executed.

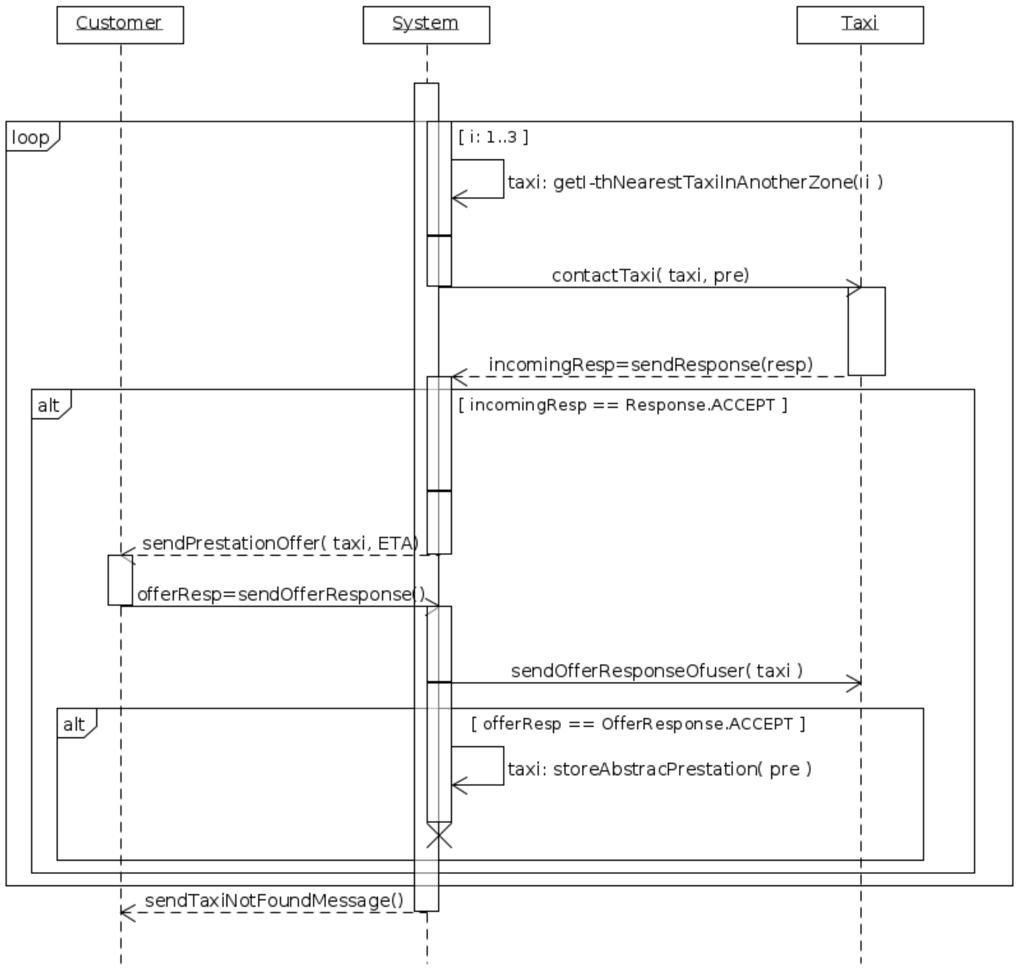




5.1.2 System operations

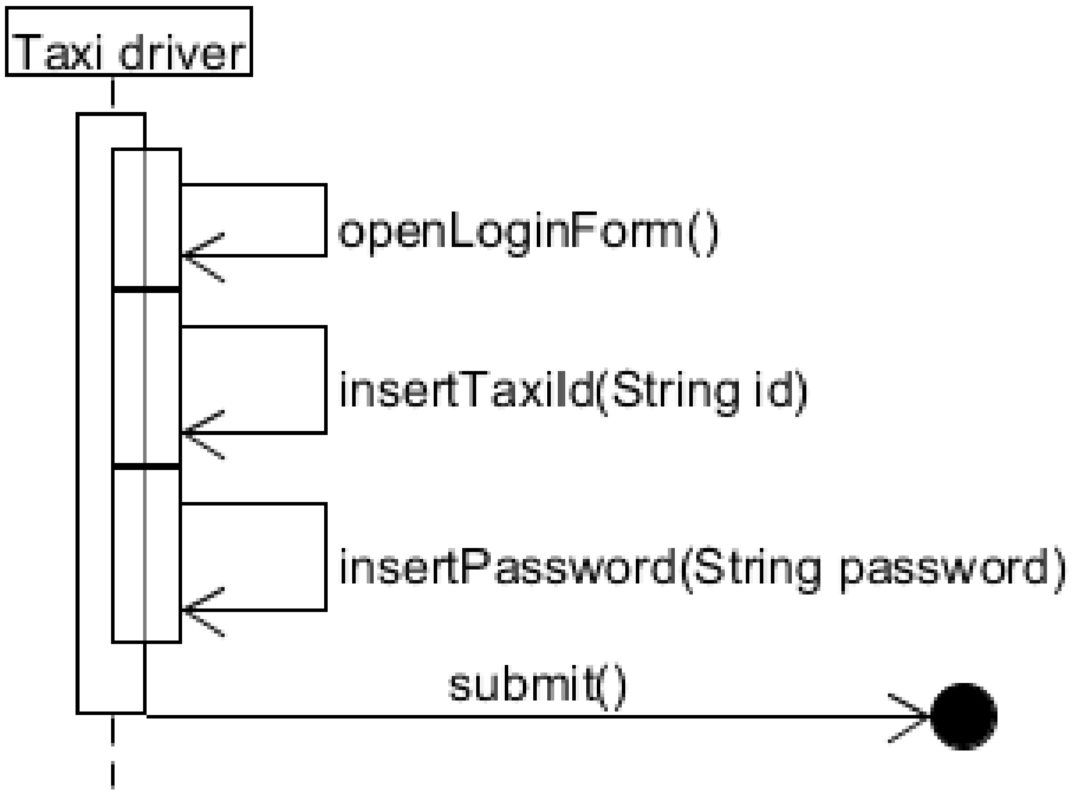
The system must pair customers and taxi driver using the first the taxi-queue of a zone, then the *“3-time out-of-zone”* method. The first diagram explains how the system handles a queue of a zone. The second diagram explains how the system manages the 3 attemps of calling a taxi out of the zone of the inquiring customer.


Customer
System
Taxi
sendAbstractPrestation (pre)
getZoneQueueOfRequest(req):
Queue<Taxi> queue
loop
[while queue.length != 0]
queue.pop(): Taxi taxi
contactTaxi(taxi, pre)
incomingResp=sendResponse(resp)
alt
[incomingResp == Response.REFUSE]
moveTaxiToBuffer(taxi)
[else]
break
restoreQueue(queue, buffer)
sendConfirmationMessage()

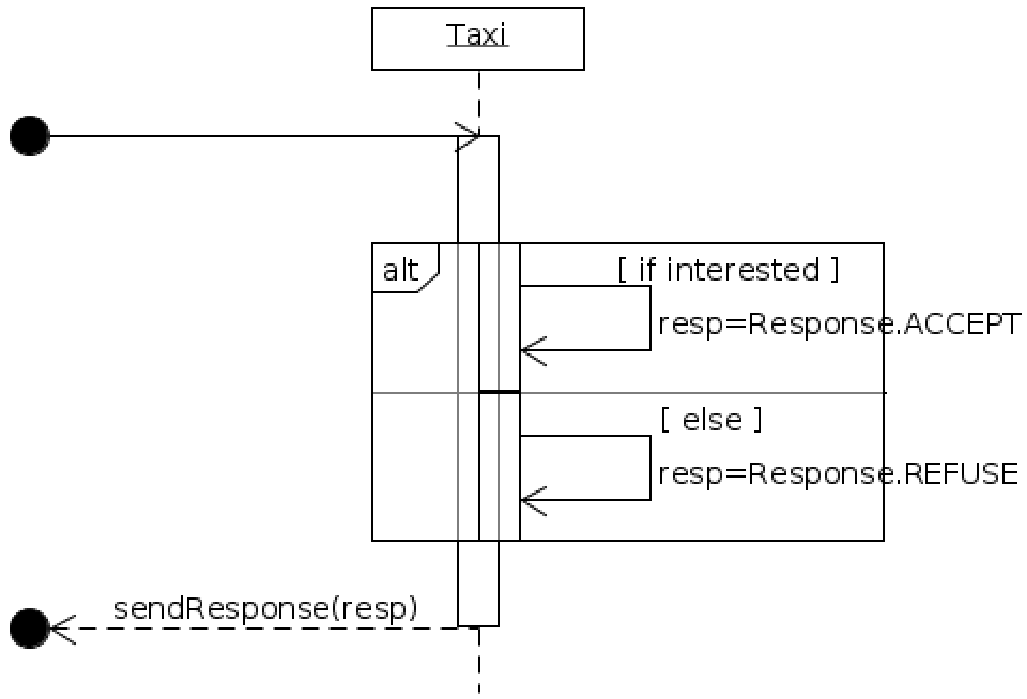

Customer
System
Taxi
loop
[i: 1..3]
taxi: getI-thNearestTaxiInAnotherZone(i)
contactTaxi(taxi, pre)
incomingResp=sendResponse(resp)
alt
[incomingResp == Response.ACCEPT]
sendPrestationOffer(taxi, ETA)
offerResp=sendOfferResponse()
sendOfferResponseOfuser(taxi)
alt
[offerResp == OfferResponse.ACCEPT]
taxi: storeAbstracPrestation(pre)
sendTaxiNotFoundMessage()

5.1.3 Taxi Driver operations

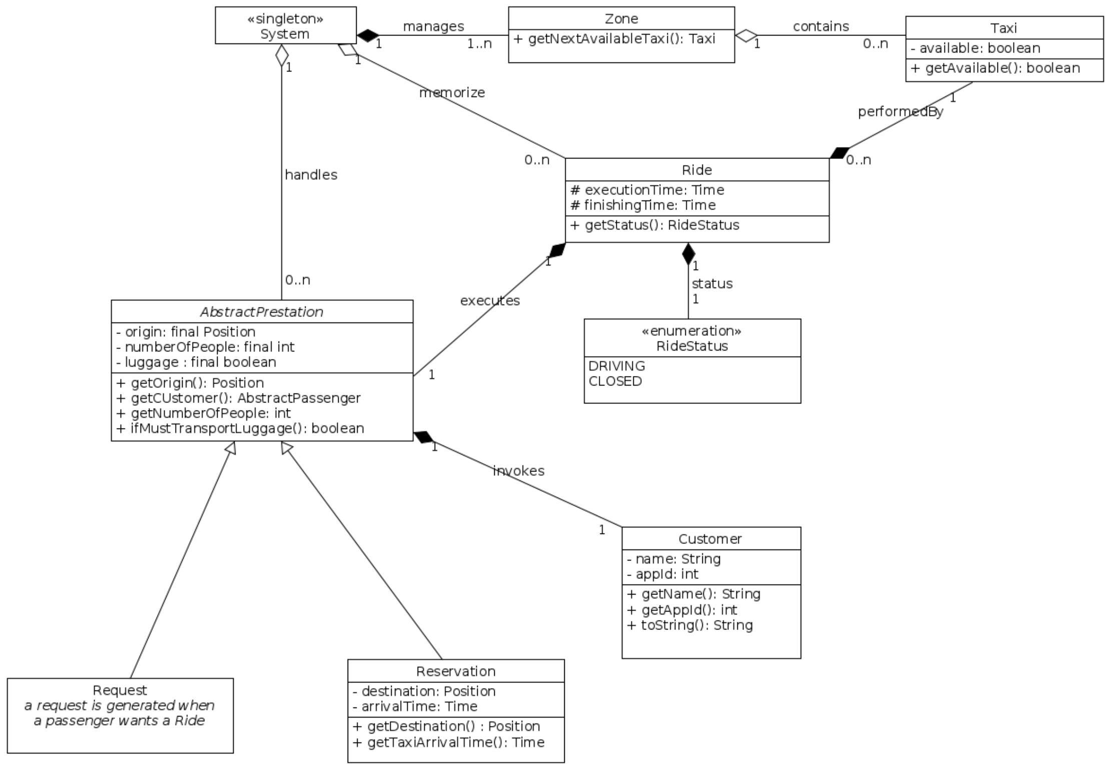
Taxi driver can log into the system.



Taxi driver can simply accept or reject the request proposed by the system.


Taxi
alt
[if interested]
resp=Response.ACCEPT
[else]
resp=Response.REFUSE
sendResponse(resp)

5.2 Class diagram



6 Alloy

Using the class diagram referred in the section above an alloy model has been developed for the sake of checking the correctness of our model. The model proposed can be found as attachement of this document. To be remarked that some signatures do not contain some attributes for simplifying the modelization.

```
/*****************************************************************
mytaxy ALLOY MODEL
AUTHORS: Matteo Maria Fusi, Matteo Locatelli
    *****************************************************************/
/**
NOTES
- Binary variables such as available and status has been
    modeled
```

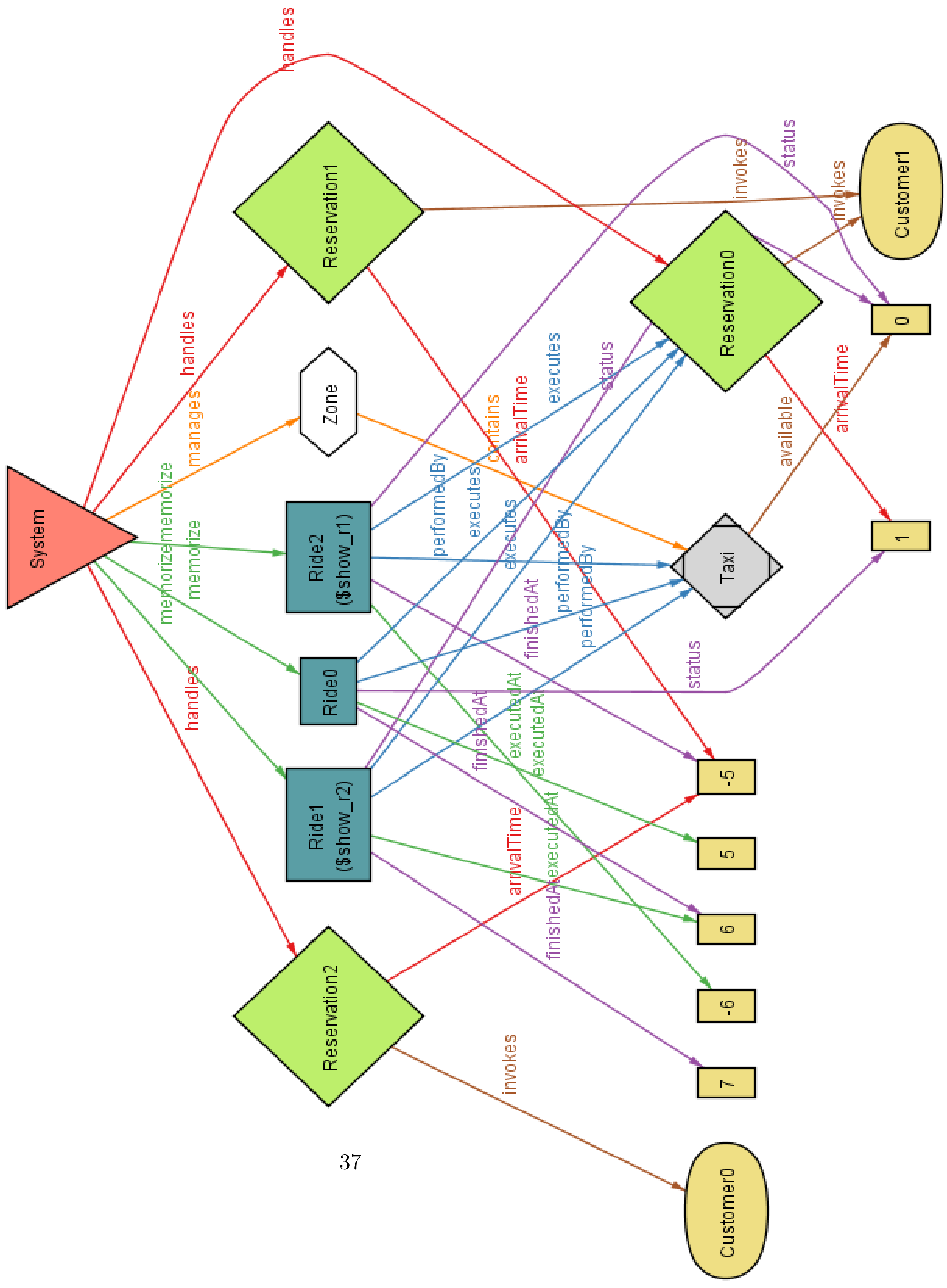
```
        with Int that assumes value 0 or 1
- Time variables such as executedAt, finishedAt and
    arrivalTime
        are Int that express milliseconds after the unix
            date
*/
/** SIGNATURES */
//taxis in the city
sig Taxi{
        0 -> not available, 1 -> available available: Int
            //binary
}
//zones of the city
sig Zone{
        contains: set Taxi //taxis in a zone
}

//when a prestation takes place a ride is generated
sig Ride{
        executedAt: Int, //time
        finishedAt: Int, //time
        performedBy: one Taxi,
        status: Int, // 0-> ride ended, 1 -> ride is
            taking place
        executes: one AbstractPrestation //prestation
            that a ride executes
}

//who requests prestations
sig Customer{}

//it manages zones, rides and prestations
sig System{
        memorize: set Ride,
        handles: set AbstractPrestation,
        manages: set Zone
}

//prestations
abstract sig AbstractPrestation{
        invokes: one Customer //who requests the
            prestation
}

//request is the base implementation
sig Request extends AbstractPrestation{}
```

```

//reservation
sig Reservation extends AbstractPrestation{
        arrivalTime: Int //time
}

/** FACTS */

//support facts
fact availableIsBinary{
        all t:Taxi | (t.available = 0) or (t.available =
            1)
}

fact rideStatusIsBinary{
        all r:Ride | (r.status = 0) or (r.status = 1)
}

//modelization
//system must be unique
fact oneSystem {
        #System=1
}

fact allZonePointedBySystem{
        all z :Zone | z in System.manages
}

fact allRidesMemorizedBySystem{
        all r :Ride | r in System.memorize
}
fact allTaxisBelongsToAZone{
        all t :Taxi | t in Zone.contains
                and all z1, z2:Zone |
                        (t in z1.contains and t in z2.
                            contains) implies (z1=z2)
}

fact allAbstractPrestationsHaveCustomer{
        all p:AbstractPrestation | one c: Customer  | c
            in p.invokes
}

fact allCustomersHaveAPrestation{
        all c:Customer | some p:AbstractPrestation | c in
            p.invokes
```

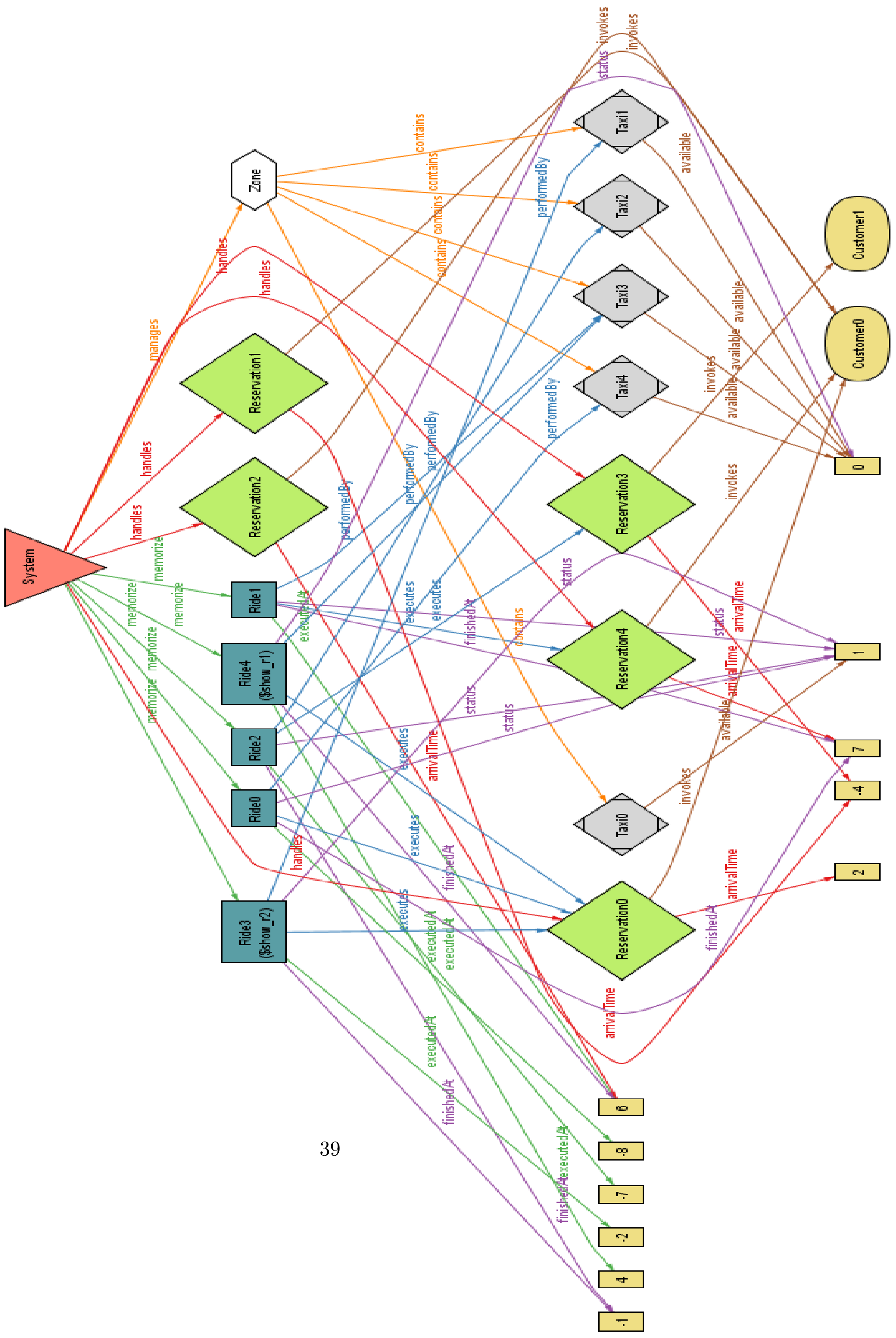
```
}

fact AllAbstractPrestationsAreLinkedToSystem{
        all p: AbstractPrestation | p in System.handles
}

fact allRideMustHaveAPrestation{
        all r : Ride | one p : AbstractPrestation | p in
            r.executes
}
/** NOTE As said above, all rides must have an
    AbstractPrestation
instance associated, but the inverse is not necessary
    true.
For instance, when an AbstractPrestation doesn't take
    place
(all taxi drivers refuse the request) the ride doesn't
    take place*/

fact aTaxiCannotMakeTwoRidesAtTheSameTime{
        some r1:Ride |
                some r2:Ride |
                        r1.executedAt=r2.executedAt
                            implies r1.performedBy != r2.
                            performedBy
}

fact theEndOfARideIsAfterItsBeginning {
        all r:Ride | r.executedAt < r.finishedAt
}

fact aTaxiIsAvailableIfIsNotRunning{
        all t:Taxi |
                (t.available = 0) implies ( one r:Ride |
                    (t in r.performedBy) and r.status = 1)
}

//tolerance is set to 5 minutes= 300000 milliseconds
fact maximumToleranceOfArrivalTime{
        all rid: Ride |
                some res: Reservation |
                        (res in rid.executes)
                        implies
                                ((res.arrivalTime <= rid.
                                    executedAt + 300000)
                                or
                                (res.arrivalTime >= rid.
                                   executedAt - 300000))
}

//2 hours are 7200000 milliseconds
fact reservationMustBeMadeAtLeastTwoHoursBeforeTheRide{
        all res:Reservation |
                some rid:Ride |
                        (res in rid.executes) implies (
                            res.arrivalTime < rid.
                            executedAt - 7200000)
}

/** PREDICATES AND ASSERTIONS */

pred addRequest[s, s': System, r:Request]{
        s'.handles = s.handles + r
}

pred addReservation[s, s': System, r:Reservation, time:
    Int]{
        r.arrivalTime = time and
        s'.handles = s.handles + r
}

pred addRideMadeFromReservation[s, s': System, r:Ride, ex
    :Int, fin:Int, t:Taxi, stat:Int, res:Reservation]{
        r.executedAt = ex and
        r.finishedAt = fin and
        r.performedBy = t and
        r.executes = res and
        s'.memorize = s.memorize + r
}

assert checkAddedRequest{
        all s, s': System, r: Request |
                addRequest[s, s', r] implies (r in s'.
                    handles)
}

check checkAddedRequest


assert checkAddedReservation{
        all s, s': System, r: Reservation |
                addReservation[s, s', r, 1000]
                implies
                (r in s'.handles)
}

check checkAddedReservation


//try to add an illegal ride.
// this should violate the
    reservationMustBeMadeAtLeastTwoHoursBeforeTheRide fact
    .
assert checkWrongRide{
        no s,s':System , r:Ride, t:Taxi , res:Reservation
            |
                all exTime, resTime:Int |
                        addRideMadeFromReservation[s,s',r
                            ,exTime,exTime+10,t,0,res]
                        and
                        addReservation[s,s',res,resTime]
                        and(exTime - resTime < 7200000)
}

check checkWrongRide


pred show(){}

/** RUN COMANDS */
run show
run addRequest
run addReservation
run addRideMadeFromReservation
```

6.1 Generated worlds

The first world is generated to show all the relations between signatures.

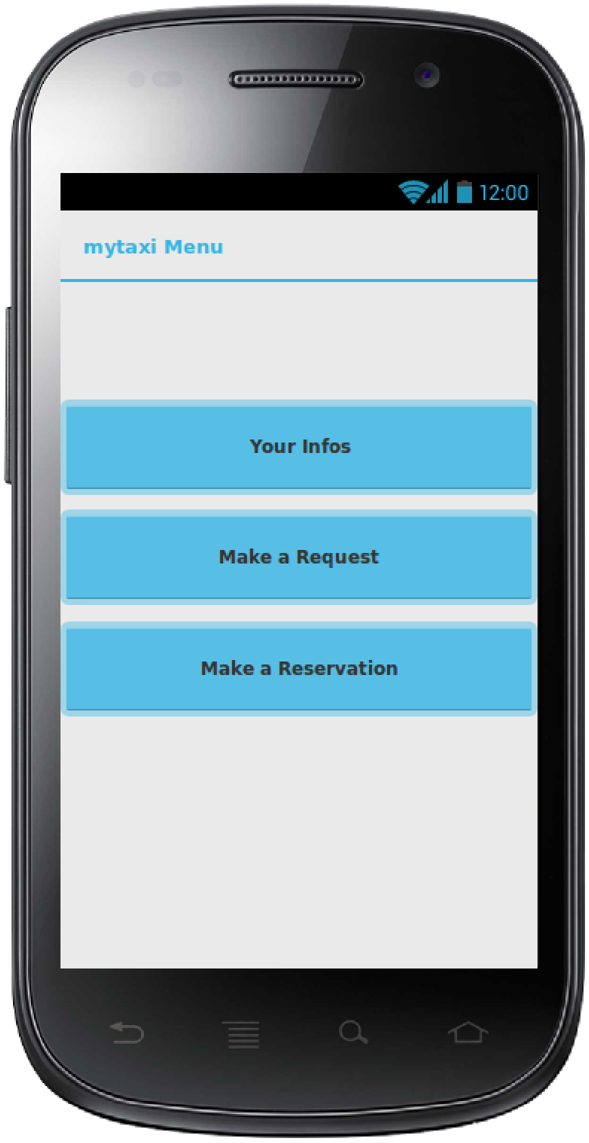

System
Reservation0
Reservation1
Reservation2
Zone
Taxi
Ride0
Ride1
($show_r2)
Ride2
($show_r1)
Customer0
Customer1
0
1
-5
5
6
-6
7
handles
handles
handles
handles
manages
memorize
memorize
memorize
memorize
contains
executes
executes
executes
performedBy
performedBy
performedBy
status
status
status
finishedAt
finishedAt
finishedAt
executedAt
executedAt
executedAt
arrivalTime
arrivalTime
arrivalTime
invokes
invokes
invokes
available

The second world is generated to show how the system is modeled with multiple instance of taxis, customers etc.

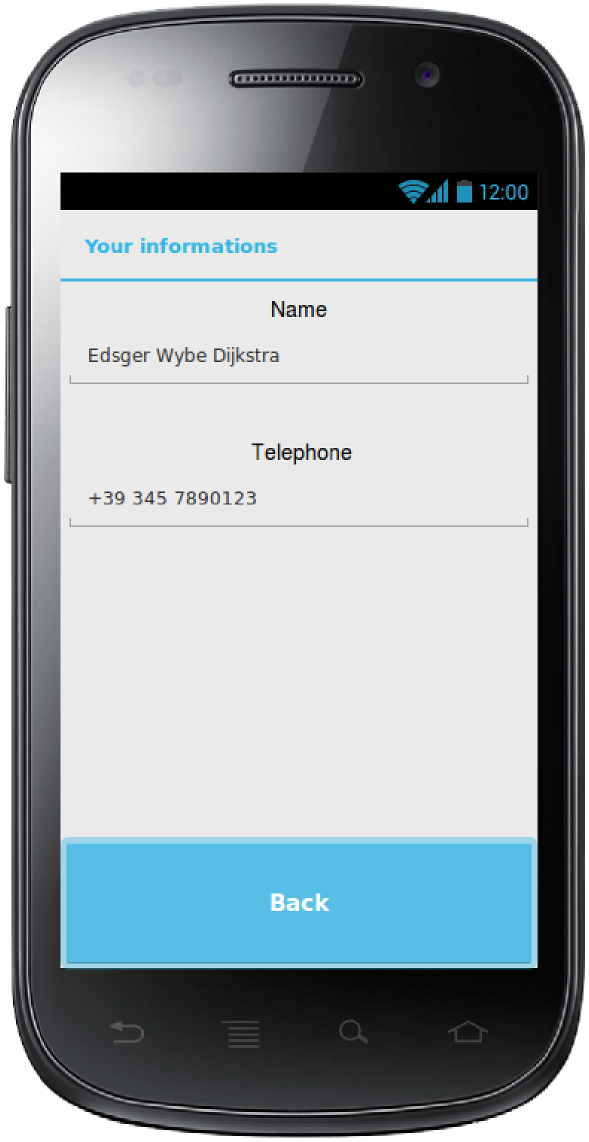

System
Zone
Taxi0
Taxi1
Taxi2
Taxi3
Taxi4
Customer0
Customer1
Reservation0
Reservation1
Reservation2
Reservation3
Reservation4
Ride0
Ride1
Ride2
Ride3 ($show_r2)
Ride4 ($show_r1)
0
1
7
-4
2
6
-8
-7
-2
4
-1
handles
manages
memorize
contains
performedBy
executes
executedAt
finishedAt
status
arrivalTime
invokes
available

7 Mockups

Main Menu



Info Page



Your informations

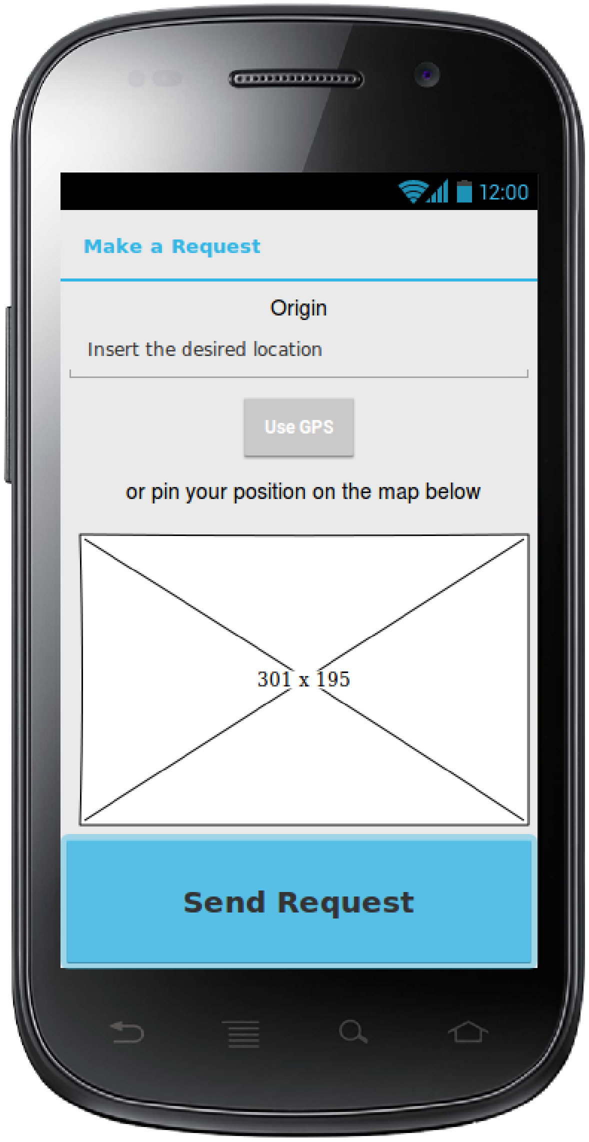
Name

Edsger Wybe Dijkstra

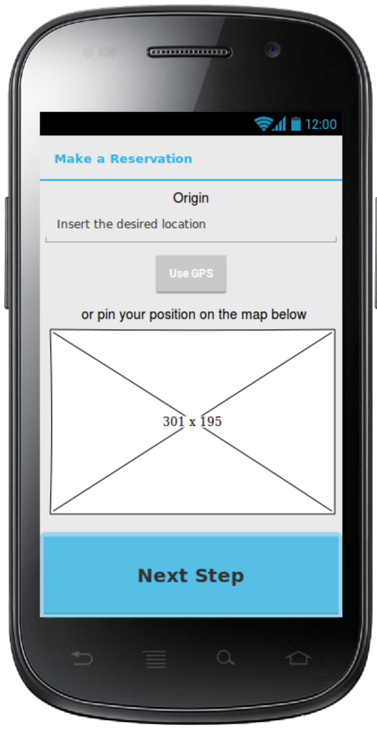
Telephone

+39 345 7890123

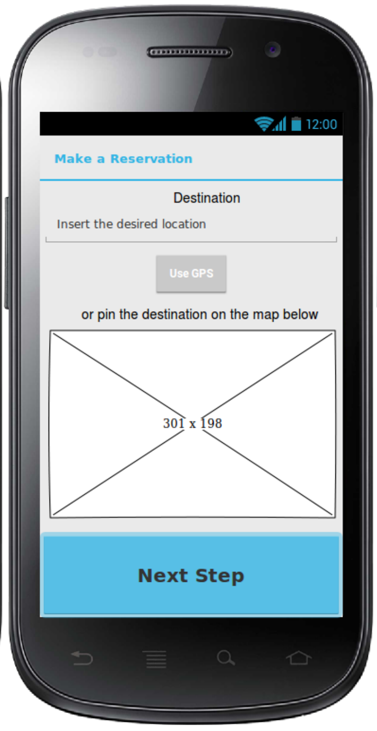
Request mockup


12:00
Make a Request
Origin
Insert the desired location
Use GPS
or pin your position on the map below
301 x 195
Send Request

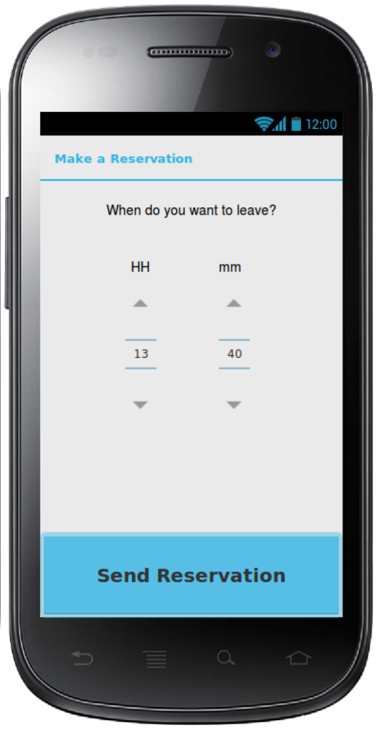
Reservation mockups



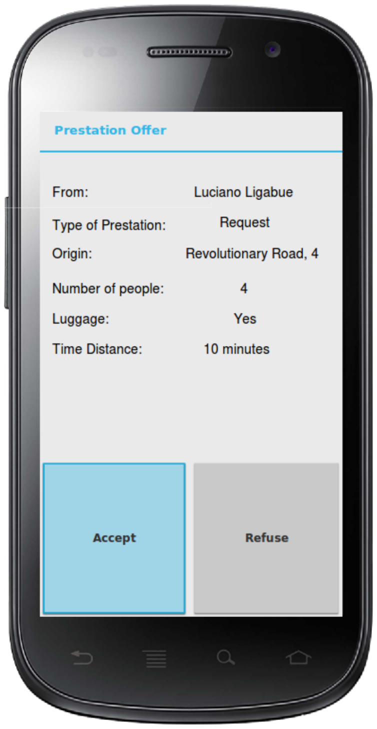
12:00
Make a Reservation
Origin
Insert the desired location
Use GPS
or pin your position on the map below
301 x 195
Next Step



12:00
Make a Reservation
Destination
Insert the desired location
Use GPS
or pin the destination on the map below
301 x 198
Next Step

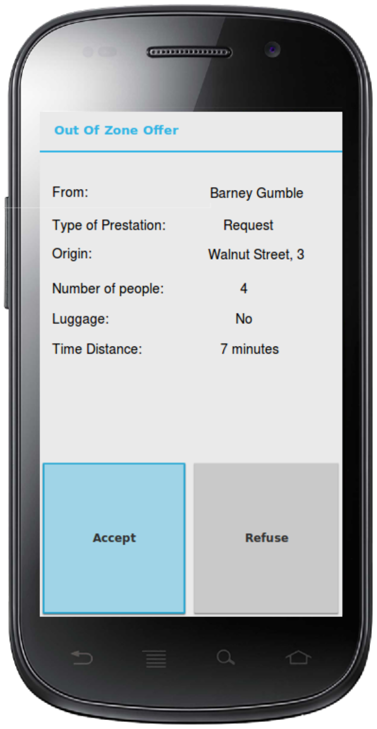


Taxi Requests



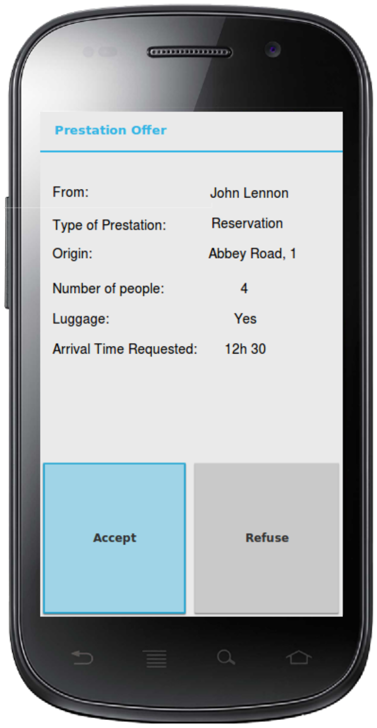
Prestation Offer

From:	Luciano Ligabue
Type of Prestation:	Request
Origin:	Revolutionary Road, 4
Number of people:	4
Luggage:	Yes
Time Distance:	10 minutes



Out Of Zone Offer

From:	Barney Gumble
Type of Prestation:	Request
Origin:	Walnut Street, 3
Number of people:	4
Luggage:	No
Time Distance:	7 minutes



Prestation Offer

From:	John Lennon
Type of Prestation:	Reservation
Origin:	Abbey Road, 1
Number of people:	4
Luggage:	Yes
Arrival Time Requested:	12h 30

A Software used

For the production of this document the sofware elencated below has been used:

- Document production and layout: LyX `www.lyx.org`
- UML diagrams (sequence, use cases, class): UMLet `www.umlet.com`
- Structoure and model analyzer: Alloy `alloy.mit.edu/alloy`
- Mockups: Pencil `pencil.evolus.vn`

B Time spent

- Matteo Maria Fusi: ~25 h
- Matteo Locatelli: ~25h