



POLITECNICO MILANO 1863

Requirements Analysis and Specifications Document of Software Engineering 2 Project

Andrea Gussoni

Federico Amedeo Izzo

Niccolò Izzo

February 15, 2016

Contents

1	Introduction	4
1.1	Purpose	4
1.2	Scope	4
1.3	Definitions, Acronyms and Abbreviations	4
1.3.1	Definitions	4
1.3.2	Acronyms	5
1.3.3	Abbreviations	5
1.4	References	5
1.5	Overviews	5
2	Overall Description	6
2.1	Product perspective	6
2.1.1	System interfaces	6
2.1.2	User interfaces	6
2.1.3	Hardware interfaces	6
2.1.4	Software interfaces	7
2.1.5	Communication interfaces	7
2.1.6	Memory	7
2.1.7	Operations	7
2.1.8	Site adaptation requirements	8
2.2	Product functions	8
2.3	User characteristics	8
2.4	Constraints	8
2.4.1	Regulatory policies	9
2.4.2	Hardware limitations	9
2.4.3	Interfaces to other applications	9
2.4.4	Parallel operation	9
2.4.5	Control functions	9
2.4.6	Reliability requirements	9
2.4.7	Criticality of the application	9
2.4.8	Safety and security considerations	9
2.5	Assumptions and dependencies	9
3	Specific Requirements	10
3.1	External interface requirements	10
3.1.1	User interfaces	10
3.2	System features	14
3.2.1	[G1] Customer Registration	14
3.2.2	[G2] User login	15
3.2.3	[G3] Standard ride	15
3.2.4	[G4] Reserved ride	16
3.2.5	[G5] Sharing-enabled ride	17
3.2.6	[G6] Taxi cab enqueueing	17
3.2.7	[G7] Taxi driver Confirmation of a ride	17
3.3	UML Models	19
3.3.1	Use Case	19

4	Alloy	40
4.1	Alloy Code	40
4.2	Generated World	45
5	Appendix	48
5.1	ToolBox	48
5.2	Hours of Work	48

1 Introduction

1.1 Purpose

This documentation consists in the RASD. The RASD has the main purpose of describing the functional and non-functional requirement and to analyze the client needs. Another purpose is to understand and estimate the size and the costs of this project, factors that are crucial for the development of the whole system. This document is aimed to all the people involved in the project, such as:

- The stakeholders of the project.
- The domain experts that will collaborate in defining the requirements.
- The developers that have to implements the requirements.
- The future maintainers of this system.
- All the people involved in possible extensions of the system.

1.2 Scope

The system developed will be **myTaxiService**. This service aims to offer a simplified and reliable access to the preexisting taxi infrastructure of the city. Both taxi drivers and customers will benefit from this product. In fact the customers will be able to:

- Have access to a taxi more quickly.
- Reserve in advance a taxi ride with fixed origin and destination.
- Share the taxi fee with others passengers going in the same direction.

The taxi drivers will also benefit from the usage of **myTaxiService**. In particular the system will guarantee the following advantages:

- Get an even distribution of the customers' service demand.
- Automatic route planning proposal in case of shared taxi ride.

1.3 Definitions, Acronyms and Abbreviations

1.3.1 Definitions

- Customers: those who will require the ride through the web application or the mobile application.
- Taxi drivers: registered users of the mobile application. They will upload their position and their availability to take rides to the system.
- Standard ride: action that begin with the customer's ride request and end with the customer's payment at the end of the ride.
- Reserved ride: a ride that has been reserved at least two hours before the starting time. It begins from the reservations and ends with the customer's arrival at his destination.

- Shared ride: a different type of ride in which the customer give his availability to share the ride. A ride is considered shared when at least two customers are travelling in the same taxi cab.
- Smartphone: a mobile device capable of connecting to the Internet and making and receiving calls and SMS.
- Geolocalization: the act of obtaining a user's geographic coordinates, eventually uploading them to an online service.
- Application: mobile or web app running on the user's device.
- System: refers to the part of the application logic that runs on the remote server.
- Taxi Zone: is an area of approximately $2km^2$ for which the taxi-queue is unique.

1.3.2 Acronyms

- RASD: Requirement Analysis and Specification Document.
- API: Application Programming Interface.
- UI: User Interface.
- OS: Operating System.

1.3.3 Abbreviations

- Web app: Web-based application.

1.4 References

- Specification Document: *Project Description And Rules.pdf*.
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.

1.5 Overviews

The structure of this document is inspired at the IEEE standard for redacting RASD documents. So its structured basically in this way:

- Section 1: Introduction, gives a general description of the document and the system to be developed.
- Section 2: Overall Description, gives a general view on all the factors that affects the product and its requirements, though not going in a deeper description of specific requirements.
- Section 3: Specific Requirements, in this section there are all the system's requirements. These requirements are specified with enough details to enable the system designers to satisfy them. This section will be supported with a set of UML diagrams, such as *Use Case Diagram*, *Flowchart* and *State Diagram*.

2 Overall Description

2.1 Product perspective

From the customers' point of view, the final product will be composed of:

- a web application, only a browser is required to use this interface.
- a mobile application for smartphones and tablets (available on AndroidTM and iOS).

2.1.1 System interfaces

The system will not depend on third-party agents for the basic functions. In fact the system will not rely on pre-existing infrastructures to manage the taxi calls (e.g. all the information regarding the geolocalization of customers and taxi cars will be obtained by using the GPS capabilities of the mobile devices running the app). Implementing some APIs for our service will have some positive impacts on future development. An example of these extensions could be the integration of the system in a bigger infrastructure (e.g. a national or an international service).

2.1.2 User interfaces

The available interfaces will be:

- the web app (oriented to the customers)
- the mobile app (oriented to both customers and taxi drivers).

The customers' web app will be implemented with a responsive design; therefore it will be able to resize itself according to the device used by the customer, be it a smartphone, a tablet or a PC. The idea is to develop them in order to be as similar as possible, so the user can simply switch from an interface to another without feeling confused. In addition to this, the taxi drivers' mobile application will have a clear UI design with the most used function keys in the main page, to let them use as little time as possible for interacting with the system. The use of the web app while driving won't be necessary at all. A taxi driver will be able to use the application even without a training course. The enqueueing process will take up to 5 minutes to be performed by a taxi driver, while the customer's ride request will take from 5 to 10 minutes.

2.1.3 Hardware interfaces

The main hardware interaction concerns the geolocalization feature, and in particular the GPS hardware of the smartphone. Both the web app and the mobile app will not interact directly with the computer or smartphone GPS hardware but will make use of specific localization frameworks, depending on the platform (refer to Software Interfaces 2.1.4 for details).

2.1.4 Software interfaces

As stated before the software requirements will be:

- An HTML5 compliant browser for the correct execution of the the web app.
- An up-to-date version of Android™ or iOS operating system to run the mobile app, both OSes must support geolocation services.

In particular for the geolocalization features

- The web app will make use of the HTML Geolocation API.
- The mobile app will make use of respectively the Android Location Service for Android smartphones. and the Core Location Framework for iOS smartphones

2.1.5 Communication interfaces

A working internet connection on the smartphone is required for the mobile app or web app to work properly, in particular:

- The web app will be entirely web hosted, it will accessible only via internet.
- The mobile app will require communication with the application server for the various services (e.g. getting the position of taxi cars or uploading the customer's position).

2.1.6 Memory

- client-side applications will not require a large permanent file storage. The only requested memory will be the one necessary to store the mobile app itself.
- while loaded, the web app will not require any secondary memory space.
- the server-side hardware should have enough primary memory to handle the main executable.

2.1.7 Operations

The system operations will be divided into two subsets:

- taxi ride request.
- taxi cab management.

The first subset will cover both the standard and reserved rides, consinting in an upload of the user position and a download of the taxi reservation results. The second subset will include the enqueueing operations in which every taxi driver submits its position and receives a notification of the successfull insertion in the current area's queue. In addition to this the second subset will cover the confirmation of the client's ride requests, this operation will imply downloading of the customer's route and the upload of the taxi driver choice.

2.1.8 Site adaptation requirements

Due to the fact that the Italian traffic laws forbid to use a cellular phone while driving, the application will not require interaction while driving, to ensure this, it will not accept any input when it will detect that the car is moving.

2.2 Product functions

The system will address various needs, all related to taxi travels. First, it will allow the customers to register on the service, by doing so, customers gain access to the provided services. Among those services are the taxi ride reservation, that allows a customer to request a taxi ride and get a short waiting time. After the reservation, the user will get the incoming taxi's number, so that he can recognize it when it will show up. The customer will also have the possibility to reserve a ride in advance. By doing so he will assure himself a timely service even in overcrowded places or during late hours. For being able to reserve a ride, the user must specify the origin and destination of its travel. The customer may also select the shared-ride option, in that case, if he has selected a destination, he accepts to share his taxi ride with other people. But not only customers benefit from the system's features, in fact, the system also handles the taxi drivers' side. Everything begins when the taxi driver decides via its mobile app to set himself as an active taxi into the system. When he does so, the system enqueues the driver into a local queue, determined by the taxi cab position into the city. Customers will request various taxi rides and the system will dispatch those requests to the enqueued drivers, the first to enqueue is the first to get a ride. When a taxi driver receives a request from the system, he views the position of the customer via its mobile app and can accept the ride or refuse it. The system inserts the refusing taxi drivers into the back of the queue.

2.3 User characteristics

The main system's users can be divided in two classes, each one has different characteristics:

- Customers.
- Taxi Drivers.

The mobile app will be used by different kind of customers, with various ages and educational levels. To be used by all these different users, the mobile app must be simple, streamlined and visually appealing. While the taxi drivers tend to be more uniform considering the age. And they are used to handle complex tools like the taxi meter or the navigation system. Considering this second class of agents we can develop a taxi drivers' app that gives many useful informations regarding the queues status, the drive requests and the suggested routes.

2.4 Constraints

The constraints to be kept in consideration for the development of the project are:

2.4.1 Regulatory policies

myTaxiService will respect the common privacy policies, by means keeping the user personal data and informations private.

2.4.2 Hardware limitations

The mobile app is designed to run on smartphones, that usually have limited computational power and memory resources, For this reason the mobile app should avoid intensive use of cpu or memory resources to prevent possible battery drains or overheating of the smartphone.

2.4.3 Interfaces to other applications

myTaxiService will provide a general API to interface with other services

2.4.4 Parallel operation

myTaxiService will be required to support simultaneous access and operations by multiple users

2.4.5 Control functions

myTaxiService will not provide a function of direct control and supervision, instead it will provide an email address for troubleshooting.

2.4.6 Reliability requirements

myTaxiService should provide a standard level of reliability for an online service.

2.4.7 Criticality of the application

The myTaxiService Platform is not meant to be critical, because it is only aimed at improving taxi activity, not for the taxi infrastructure to rely on it.

2.4.8 Safety and security considerations

As said in the User Interface section, the taxi driver client application will avoid the use of the smartphone while driving, according to the driving laws.

2.5 Assumptions and dependencies

The assumptions made during the creation of this document regards the software platform used: As said in section overall description/software interfaces

- An HTML5 compliant browser for the correct execution of the the web app.
- An up-to-date version of AndroidTM or iOS operating system to run the mobile app, both OSes must support geolocation services.

In case a newer version of these platforms became available, the requirements will be updated accordingly.

3 Specific Requirements

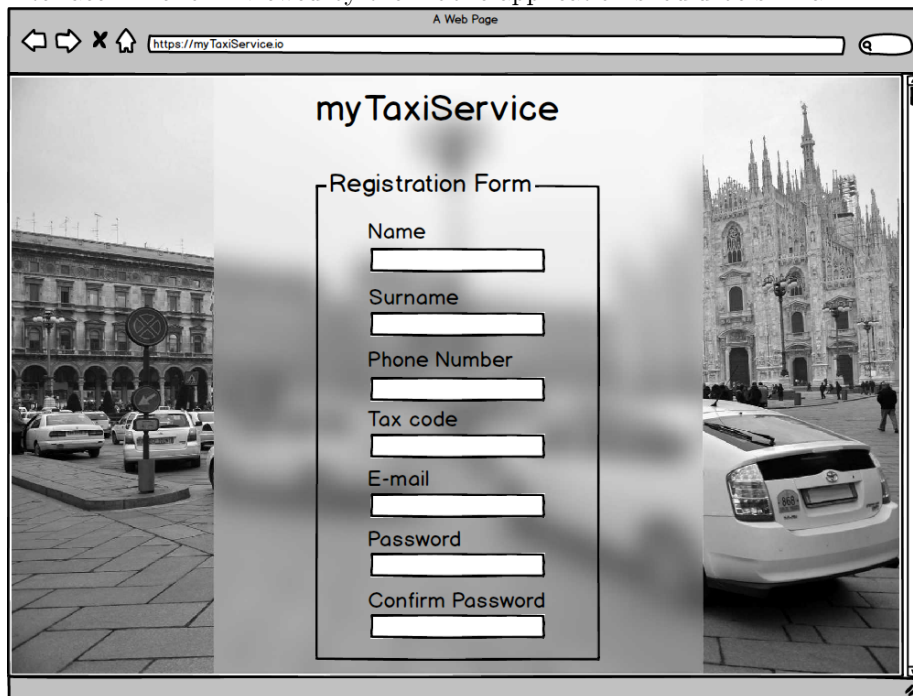
3.1 External interface requirements

3.1.1 User interfaces

Here are presented some mockups of the user interface (both for the web and the mobile one):

3.1.1.1 Registration

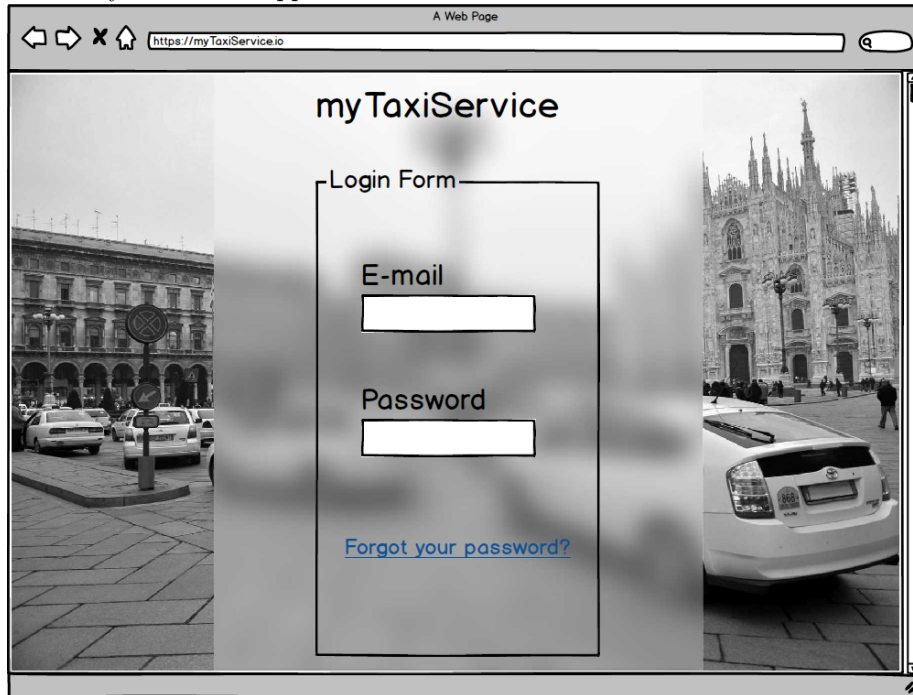
Figure 1: The mockup shows the registration form of myTaxiService of the web interface. The form viewed by the mobile application should be similar.



The mockup shows a web browser window with the address bar displaying `https://myTaxiService.io`. The page title is "A Web Page". The main content area features the "myTaxiService" logo at the top center. Below the logo is a "Registration Form" with the following fields: Name, Surname, Phone Number, Tax code, E-mail, Password, and Confirm Password. The form is set against a background image of a city street with a large cathedral and a white taxi car.

3.1.1.2 Login

Figure 2: The mockup shows the login form of the web interface. The form viewed by the mobile app should be similar.



3.1.1.3 Enqueueing

Figure 3: The mockup shows the alert that a taxi driver receives when he enters in a new taxi zone.



3.1.1.4 Taxi Request

Figure 4: The mockup shows the "map view" from which the customer can select the pickup location and make a taxi-call.



3.1.1.5 Ride acceptance

Figure 5: The mockup shows the alert the driver receives when he is selected by the system from the area queue.



3.2 System features

3.2.1 [G1] Customer Registration

3.2.1.1 Purpose of the feature

Allow an unregistered customer to create an account into the system.

3.2.1.2 Stimulus/Response sequence

The unregistered customer downloads the mobile app or accesses the web app and opens the *registration* section of the application.

The application shows a registration form with fields corresponding to all the requested informations.

3.2.1.3 Associated functional requirements

- The system will request some information such as:
 - Name and Surname
 - Valid email address
 - Tax code
 - Telephone number
 - Password for the account
 - Confirm password
- The application must show the informations input form.
- The application must verify the validity of the email address:
 - The application will check that the entered address is legal.
 - After completing the form, the system will send an email to the user's address containing a verification link.
 - If the user visits the link in 24 hours the email address is verified.
- The application checks that the password must respect some security standard (e.g. it must be alphanumeric and must have at least one special character).
- The system checks that the two passwords entered in the password field and verification field matches.

3.2.2 [G2] User login

3.2.2.1 Purpose of the feature

Allow a registered user to authenticate in the application.

3.2.2.2 Stimulus/Response sequence

The registered customer open the mobile app or accesses the web app and opens the login section of the application.

The application shows a login form with fields corresponding to all the requested informations.

3.2.2.3 Associated functional requirements

The system will request the following information:

- email address used for registration.
- password chosen during registration.

3.2.3 [G3] Standard ride

3.2.3.1 Purpose of the feature

Allow the registered customer to request a taxi ride. The system will provide the customer with the code of the incoming taxi and the estimated time he has to wait until the requested taxi arrive.

3.2.3.2 Stimulus/Response sequence

The customer requests a taxi ride using the application.

The system allocates a taxi driver taking it from the local queue.

3.2.3.3 Associated functional requirements

- The application must check that the customer is logged in.
- The application must upload the customer's position to the system.
- The system must obtain the zone corresponding to the customer's position.
- The system must get the taxi queue corresponding to the zone the customer is in.
- The system will pop the first taxi driver from the selected queue.
- The system will send a ride confirmation to the selected taxi driver.
- The system will send the taxi driver's code to the customer.
- The application will show the estimated waiting time.

3.2.4 [G4] Reserved ride

3.2.4.1 Purpose of the feature

Allows the registered customer to reserve a taxi for a ride that will take place at least 2 hours after the reservation. The system will provide the customer with a confirmation of the reservation.

3.2.4.2 Stimulus/Response sequence

The customer uses the application to reserve a taxi and the departure time is at least two hours in the future.

The system sends a confirmation to the customer in case of successful reservation.

3.2.4.3 Associated functional requirements

- The application must check that the user is logged in.
- The application must check that the customer has specified an origin and a destination for the reserved ride.
- The application must check that the reservation takes place at least 2 hours before the desired ride.
- The system must allocate a taxi at least 10 minutes before the desired beginning of the ride.
- From that moment on the ride proceeds as the standard ride.

3.2.5 [G5] Sharing-enabled ride

3.2.5.1 Purpose of the feature

The system gives the ability to eventually share the ride with other customers.

3.2.5.2 Stimulus/Response sequence

The customer requests a taxi ride using the application and he enables the shared ride option from the application UI.

The system enables the customer to share the ride.

3.2.5.3 Associated functional requirements

- The system checks that the customer has flagged the shared ride option.
- The system verifies that the customer has selected a destination for his ride.

3.2.6 [G6] Taxi cab enqueueing

3.2.6.1 Purpose of the feature

The application gives the taxi drivers the opportunity to give their availability to accept rides.

3.2.6.2 Stimulus/Response sequence

The taxi driver enters in a taxi zone and enables through the application his availability to accept rides.

The system enqueues the taxi driver in queue of the current area.

3.2.6.3 Associated functional requirements

- The application must check that the driver is logged in.
- The application retrieves the current location of the taxi.
- The system enqueues the taxi in the queue of the zone where the taxi is located.

3.2.7 [G7] Taxi driver Confirmation of a ride

3.2.7.1 Purpose of the feature

The application gives the opportunity to the taxi driver to accept/decline a ride.

3.2.7.2 Stimulus/Response sequence

The system selects the first taxi of the queue of a certain zone.

The driver accepts or not the ride.

3.2.7.3 Associated functional requirements

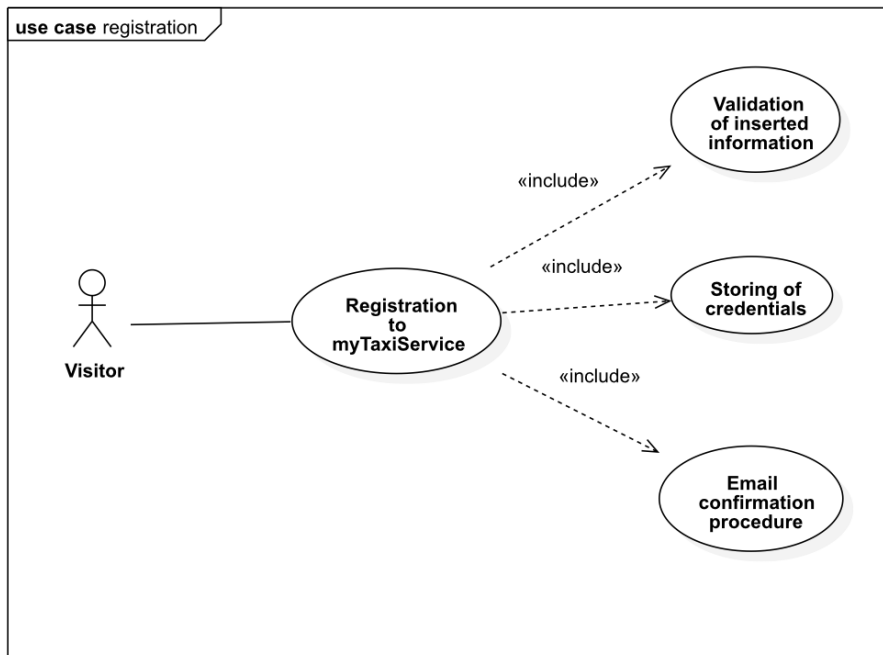
- The system sends a notification to the application of the first taxi driver in the queue.
- The applications displays to the taxi driver the informations about the ride requested and two buttons, one for accepting the ride and one for declining it.
- In case the taxi driver accepts the ride the system arranges the ride.
- In case the taxi driver declines the ride the system eliminates it from the top of the queue and places it to the bottom of the queue.

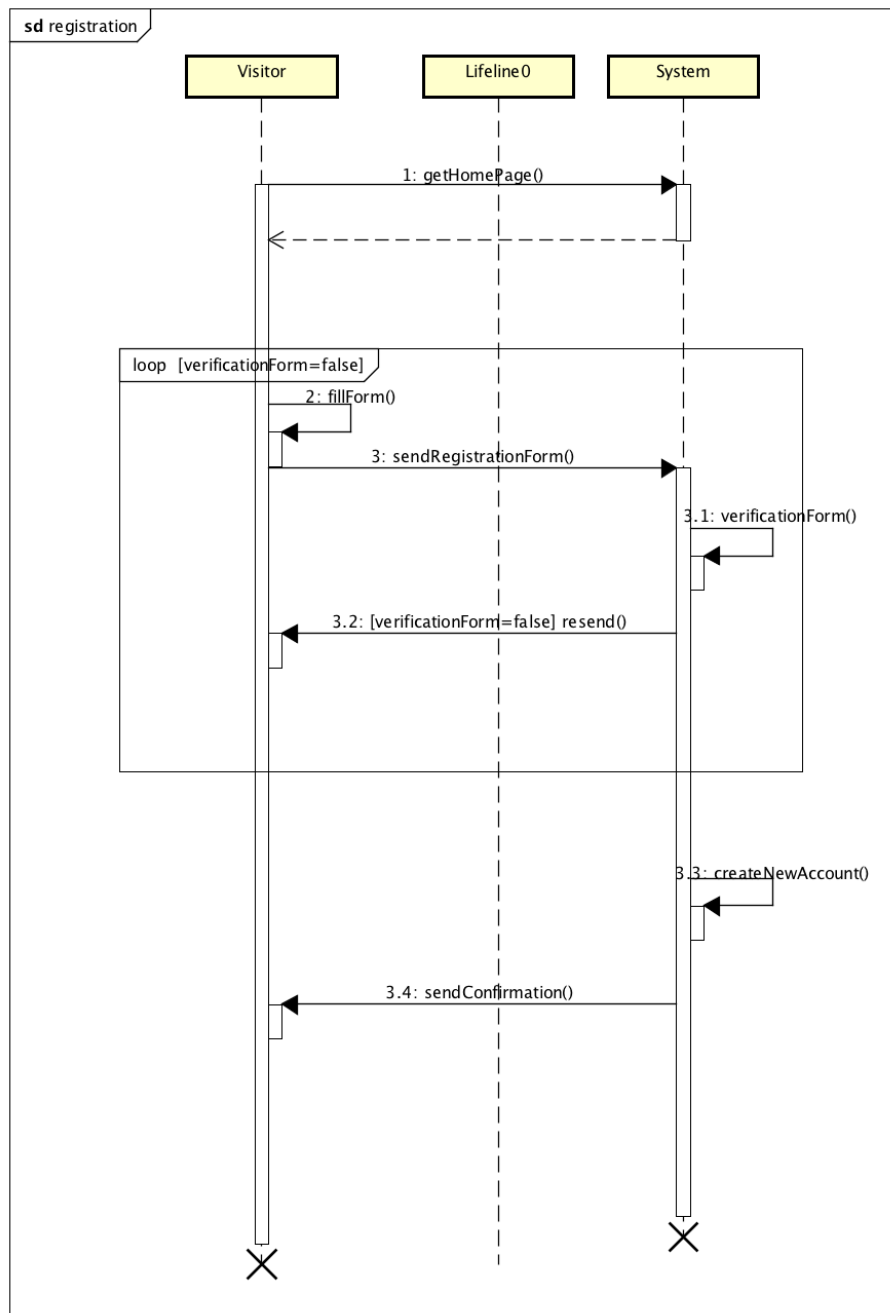
3.3 UML Models

3.3.1 Use Case

3.3.1.1 Customer registers to myTaxiService

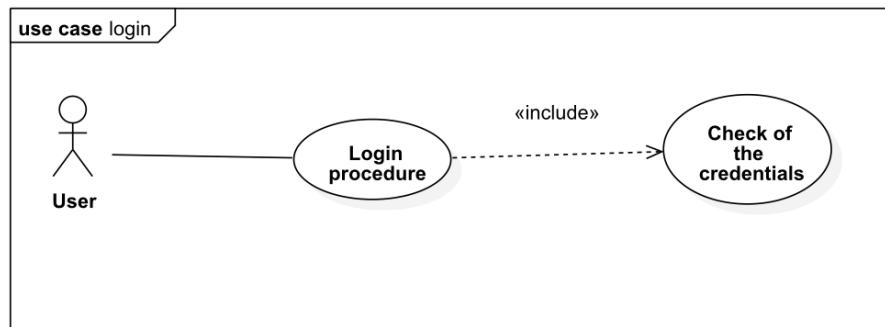
Name	Customer registers to myTaxiService
Actor	Customer
Goal	[G1]
Input Condition	Absent
Event Flow	<ol style="list-style-type: none">1. myTaxiService displays the registration page.2. User fills in the required input fields with the required informations.3. User follows the confirmation procedure (clicks on the link sent him by the system via e-mail)4. The system store the credentials in his DBMS
Output Condition	<ol style="list-style-type: none">1. The application checks the validity of the input inserted.2. If all goes right the customer is now registered in the system and from now on can log in in the application.
Exception	<ol style="list-style-type: none">1. The system refuses the registration requests because the data inserted in the registration form is not compliant to the requirements.2. The user doesn't complete the confirmation procedure before the time-out.

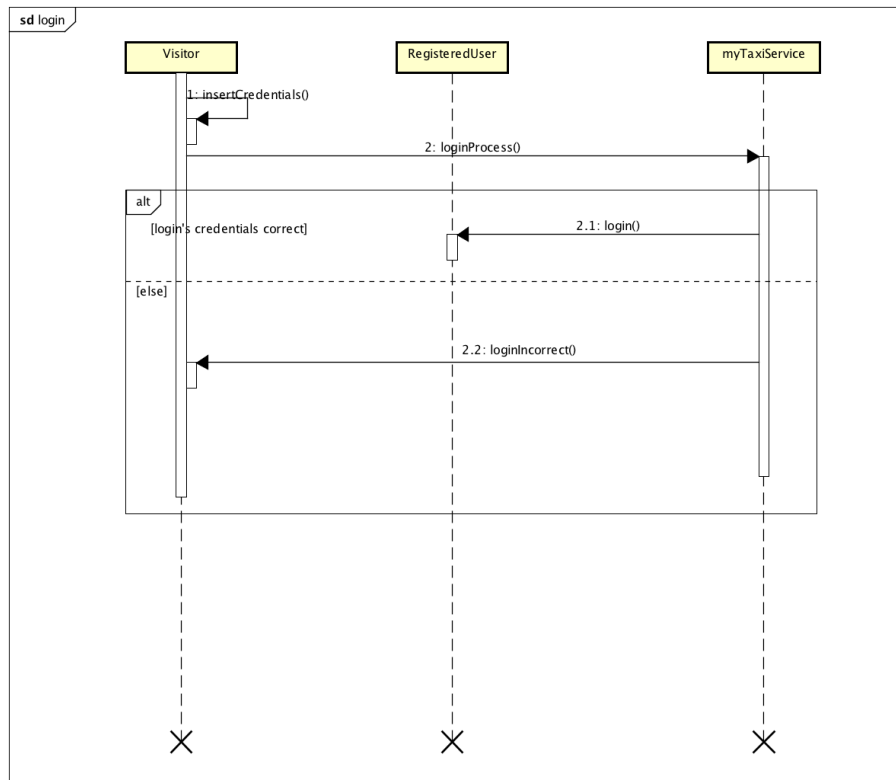




3.3.1.2 User login in myTaxiService

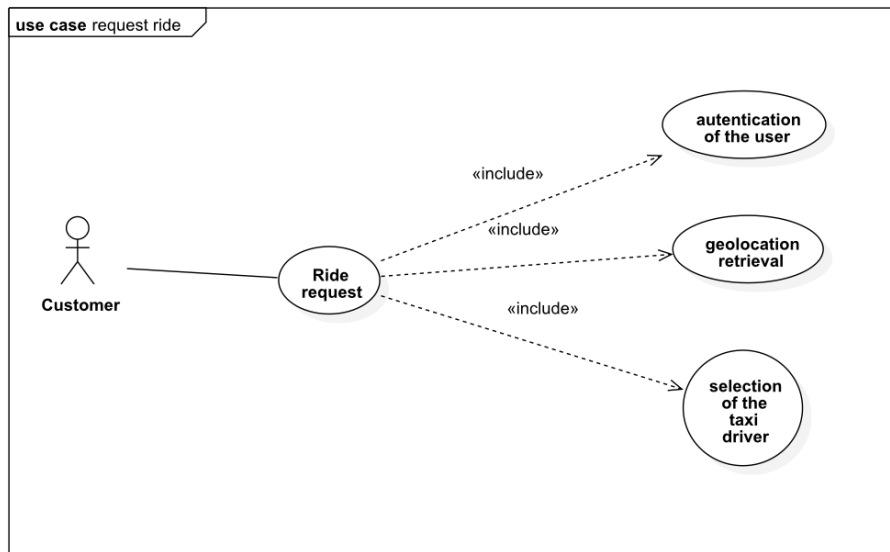
Name	User login in myTaxiService
Actor	User
Goal	[G2]
Input Condition	The user must be already registered in the system.
Event Flow	<ol style="list-style-type: none">1. The application displays the login form.2. The user inserts his credentials in the given fields and proceeds with the login.
Output Condition	<ol style="list-style-type: none">1. The application contacts the system in order to verify the user credentials.
Exception	<ol style="list-style-type: none">1. The inserted credentials are wrong.

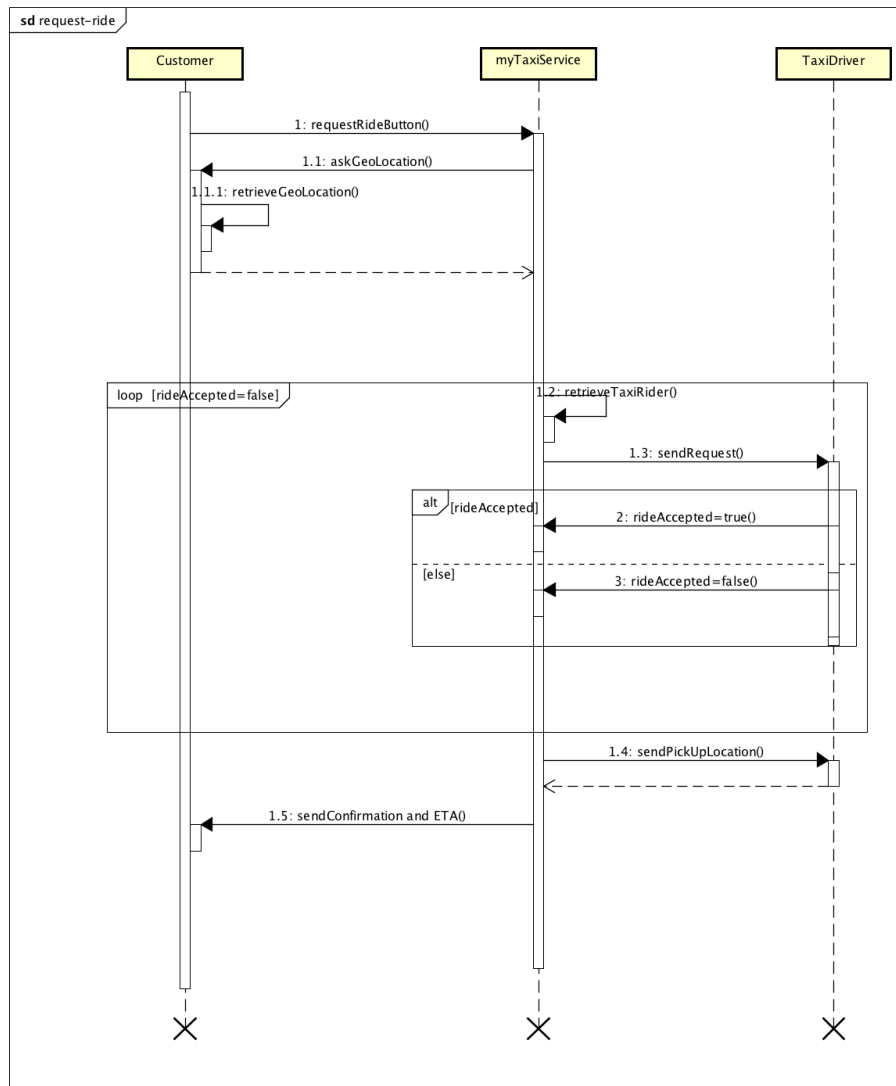




3.3.1.3 User requests a standard ride

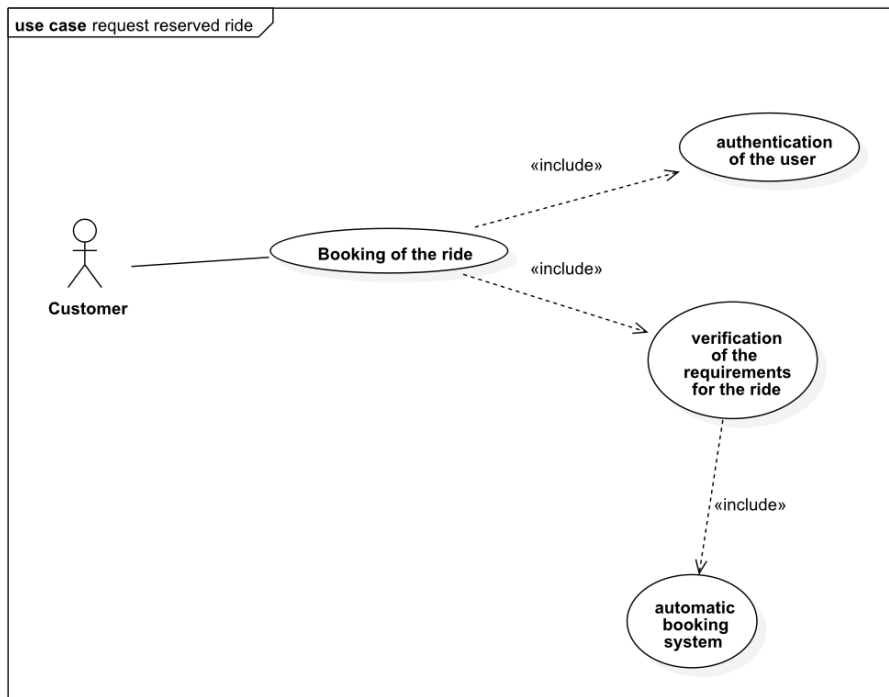
Name	User requests a standard ride
Actor	Customer
Goal	[G3]
Input Condition	The customer must be authenticated in the system and the application is able to retrieve his location.
Event Flow	<ol style="list-style-type: none">1. The user clicks on the button "call a taxi" on the application interface.2. The application retrieves the current location of the customer and sends it to the system.3. The system does all the necessary actions to retrieve the first taxi in the queue for the zone of the customer.4. The system sends to the selected driver a request to accept the ride.5. In case that the first driver denies the request, the system go on and cycles through the queue until at least one driver accepts the request.
Output Condition	<ol style="list-style-type: none">1. If the driver selected by the system accepts the request, the customer will receive a notification containing the assigned taxi code.2. Both the customer and the taxi driver receive a confirmation from the application.
Exception	<ol style="list-style-type: none">1. The system cannot find a suitable driver. After a reasonable timeout the system should notify the customer of the inability to complete the operation.2. The system cannot retrieve the user location. In this case the application must inform the customer and suggest to move in another area.

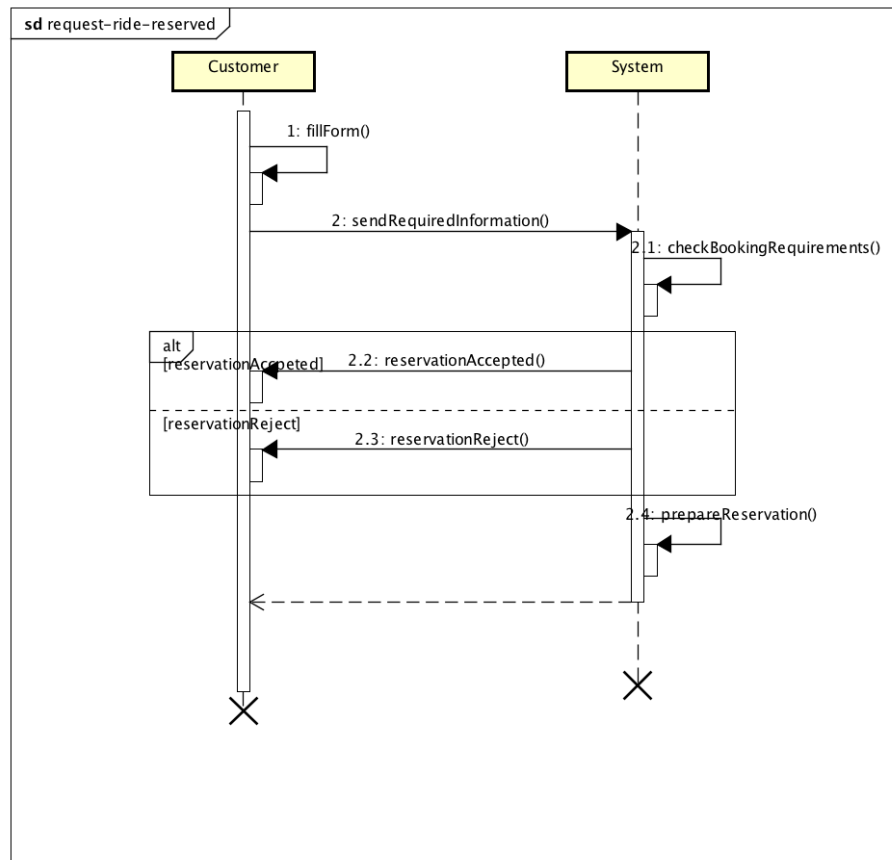




3.3.1.4 User requests a reserved ride

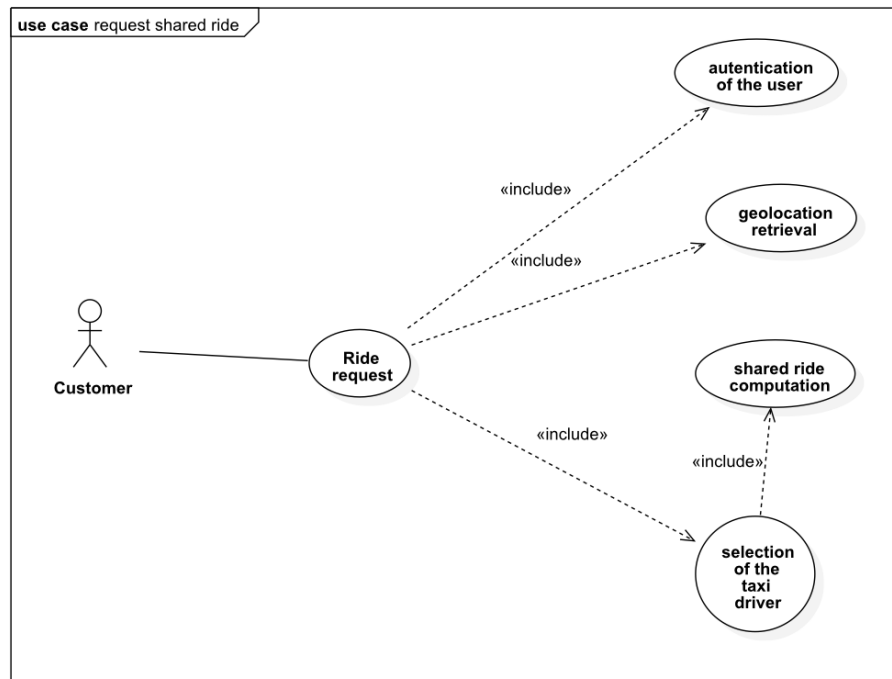
Name	User requests a reserved ride
Actor	Customer
Goal	[G4]
Input Condition	The customer must be authenticated in the system.
Event Flow	<ol style="list-style-type: none">1. The customer clicks on the "reserve a ride button" on the application.2. The application displays a form containing the fields to be filled in.3. The user inserts the starting point and the destination of the ride and the start time chosen for the ride.
Output Condition	<ol style="list-style-type: none">1. The application verifies that the chosen time for the start of the ride is at least two hours after the current time.2. The customer receives a notification of the successful reservation.3. The system enables an automatic process that reserves a taxi 10 minutes before the time chosen for the ride.
Exception	<ol style="list-style-type: none">1. The procedure fails due to the fact that at least one of the conditions for the reserved ride is not respected.2. At the moment of the reservation of the taxi car, the system can't reserve a taxi for the request.

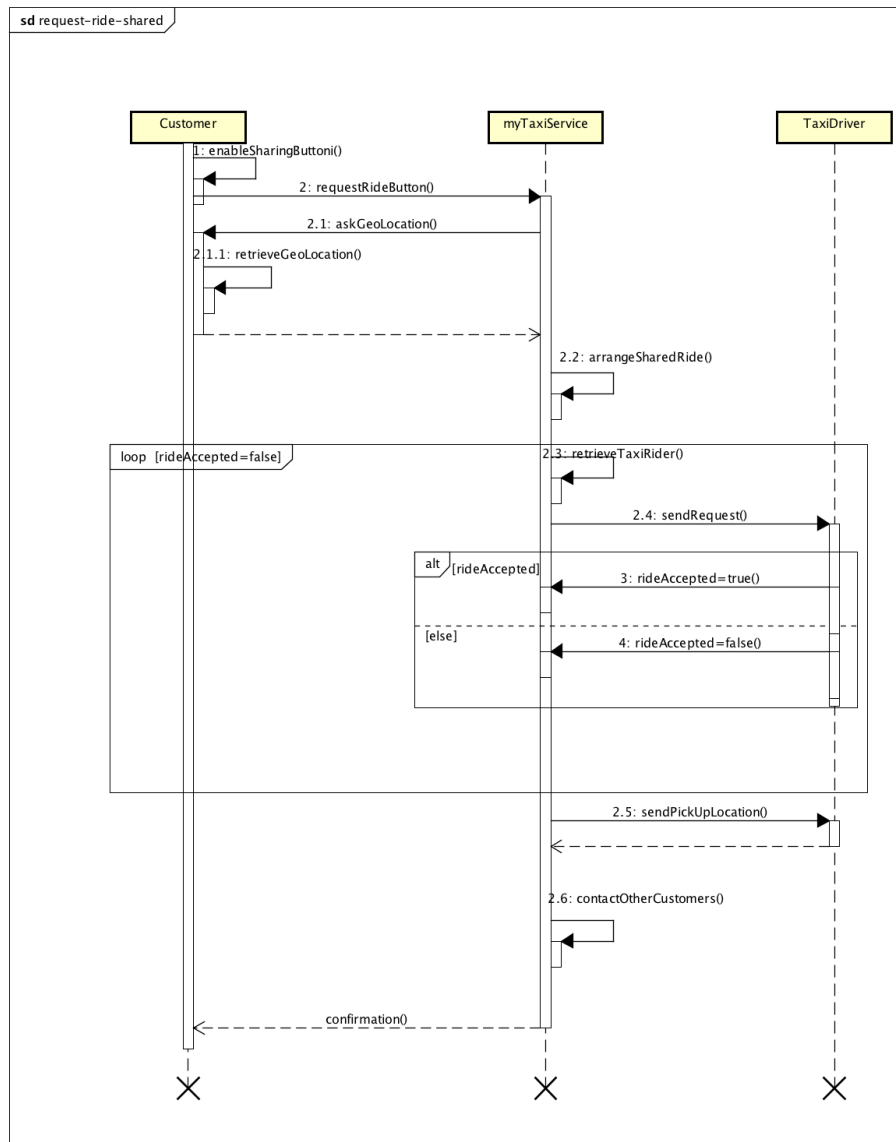




3.3.1.5 User selects the shared ride option

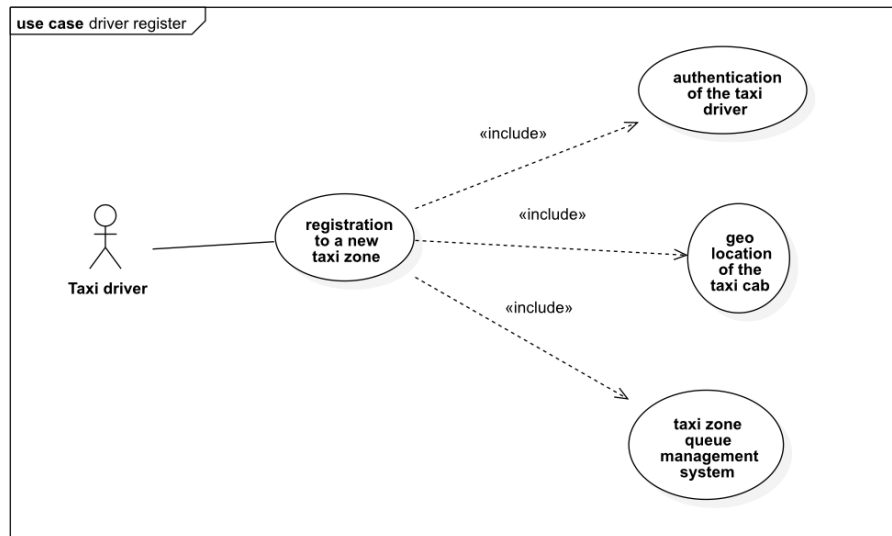
Name	User selects the shared ride option
Actor	Customer
Goal	[G5]
Input Condition	The customer must be authenticated in the system.
Event Flow	<ol style="list-style-type: none">1. The first phase of the event flow is similar to the standard ride procedure.2. The customer selects a check-box to express his intention of sharing the ride. In this case he must specify the destination of the ride.
Output Condition	<ol style="list-style-type: none">1. The system evaluates the possibility of a shared ride.2. The system sends to the customer application a notification of the operation result.3. The system arranges the route for the shared ride and sends it to the selected taxi-driver.
Exception	<ol style="list-style-type: none">1. The customer does not specify a destination.2. All the exceptions of the standard ride case.

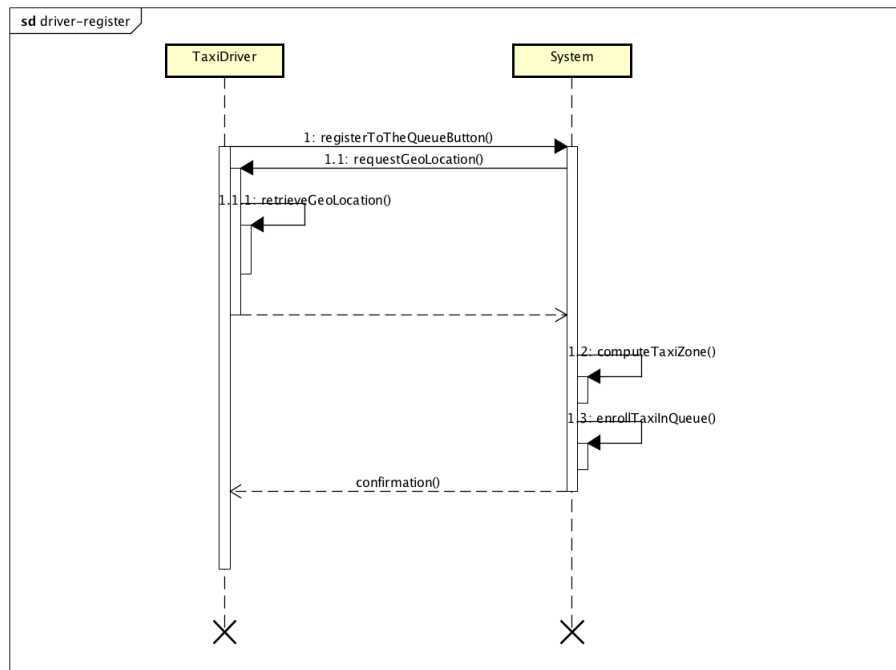




3.3.1.6 Taxi driver registers on the system

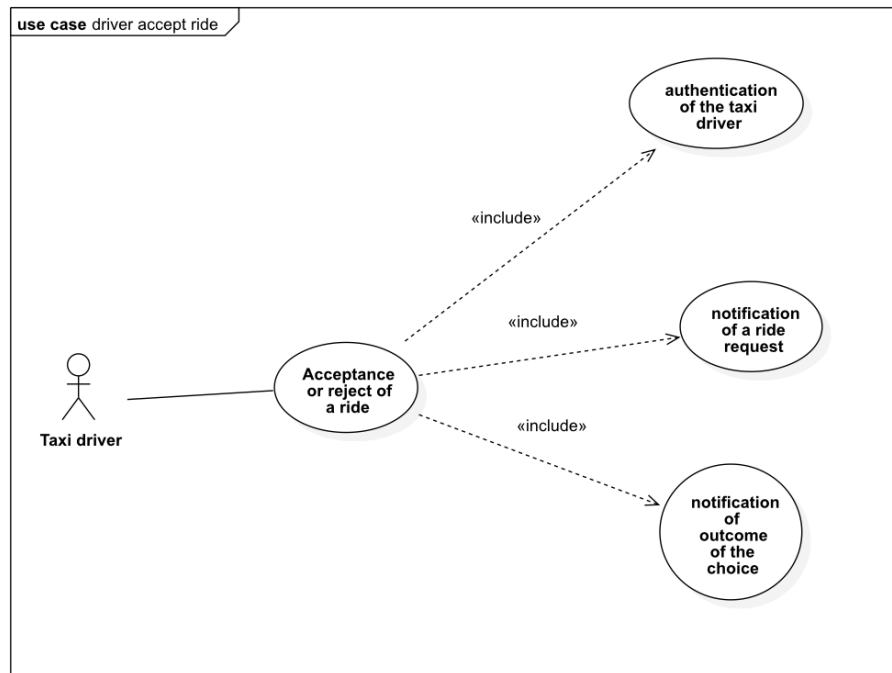
Name	Taxi driver registers on the system
Actor	Taxi driver
Goal	[G7]
Input Condition	The taxi driver must be authenticated in the system.
Event Flow	<ol style="list-style-type: none">1. The taxi driver activates from the mobile application interface the button that enrolls him in the queue of the taxis available to take a ride.
Output Condition	<ol style="list-style-type: none">1. The system sends a confirmation of the success of the operation to the driver's application.
Exception	<ol style="list-style-type: none">1. The system encounters problems with the verification of the driver's credentials.2. The application can not retrieve from the smart-phone the geo-location of the taxi driver.

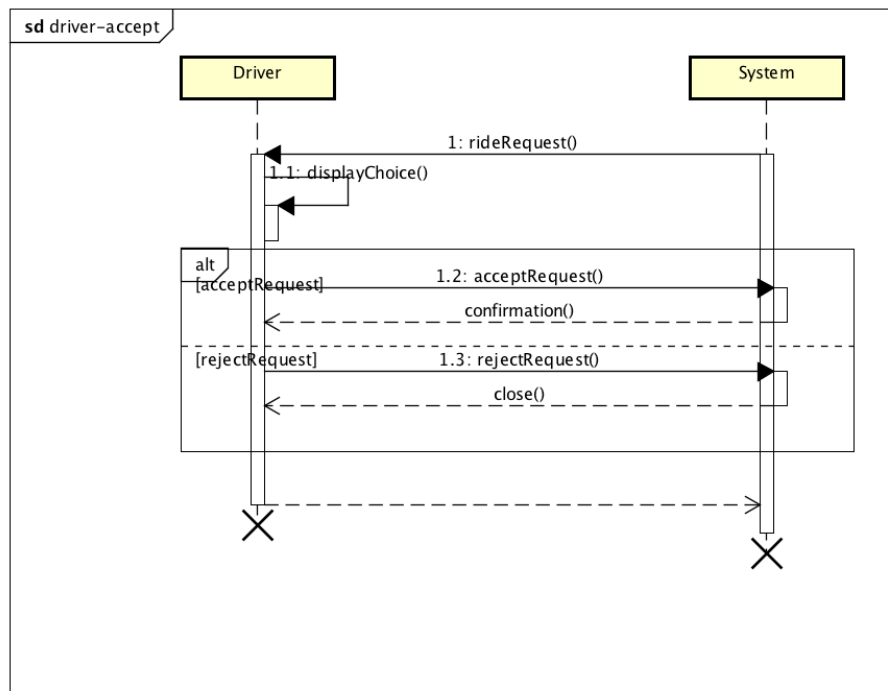




3.3.1.7 Taxi driver accepts/denies a request

Name	Taxi driver accepts/denies a request
Actor	Taxi driver
Goal	[G8]
Input Condition	The taxi driver must be authenticated in the system.
Event Flow	<ol style="list-style-type: none">1. The system, after selecting the first driver from the zone queue, sends to the application of the selected taxi driver a request to accept the ride.2. The application shows to the driver the option to accept or reject the ride.
Output Condition	<ol style="list-style-type: none">1. If the taxi driver accepts the request, the system arranges the ride as seen previously.2. If the taxi driver rejects the request, the system forwards the request to the next taxi driver in the queue and places the taxi driver who rejected the request at the bottom of the queue.
Exception	<ol style="list-style-type: none">1. The taxi driver doesn't make a choice before the timeout. In this case the system treats the situation as if the driver had selected the "reject" option.





4 Alloy

4.1 Alloy Code

In this section we have collected a basic Alloy model used to simulate and verify the consistency of the structure of the model of the system.

// Signatures

```
abstract sig User {
    id: one Int,
}

sig Customer extends User{
}

sig TaxiDriver extends User {
    taxi: one Taxi,
    availability: one TaxiDriverAvailability,
    taxiqueue: lone ZoneQueue,
}

sig Taxi {
    code: one Int,
    seats: one NumSeats,
}
abstract sig NumSeats {}

one sig FourSeats extends NumSeats{}

one sig SixSeats extends NumSeats{}

abstract sig TaxiDriverAvailability {
}

one sig TaxiDriverAvailable extends TaxiDriverAvailability {
}

one sig TaxiDriverNotAvailable extends
    ↪ TaxiDriverAvailability {
}

abstract sig Ride {
    id: one Int,
    passenger: some Customer,
    taxidriver: one TaxiDriver,
}

sig StandardRide extends Ride {
}

sig SharedRide extends Ride{
}

sig ReservedRide extends Ride{
```



```

}

sig ZoneQueue {
    id: one Int,
    nexttaxidriver : lone TaxiDriver,
}

// Facts

// The next taxi driver in the queue must be available
fact nextDriverAvailable {
    all q : ZoneQueue, d : TaxiDriver | q.nexttaxidriver
    ↪ = d implies isAvailable[d]
}

// Only one Customer is allocated to a standard ride
fact standardRideOneCustomer{
    no r : StandardRide | #r.passenger ≠ 1
}

// Only one Customer is allocated to a reserver ride
fact reservedRideOneCustomer{
    no r : ReservedRide | #r.passenger ≠ 1
}

// No rides without a passenger
fact noRideWithoutPassenger {
    no r: Ride | #r.passenger < 1
}

// Shared ride has more than one passenger
fact sharedRideHasMorePassengers {
    no r : SharedRide | #r.passenger =< 1
}

// Shared ride can not have more than 7 passengers
fact maxPassengersSharedRide {
    no r : SharedRide | #r.passenger >7
}

// One taxi driver for each ride
fact oneDriverForRide {
    no r : Ride | #r.taxidriver ≠1
}

// If a taxi driver is associated to a ride he is not
    ↪ available
fact unavailability {
    all d: TaxiDriver | isInRide[d] implies
    ↪ isNotAvailable[d]
}

```

```

// A customer cannot participate in two rides simultaneously
fact customerOnlyOneRide {
    all c : Customer | lone r: Ride | c in r.passenger
}

// Two users(Customers or TaxiDriver) cannot have the same
    ↪ id
fact uniqueIDs {
    all u1 : User, u2 : User | u1 ≠ u2 implies u1.id ≠
    ↪ u2.id
}

// Two taxis cannot have the same code
fact taxiIDs {
    all t1 : Taxi, t2 : Taxi | t1 ≠ t2 implies t1.code ≠
    ↪ t2.code
}

// Two taxi zone cannot have the same id
fact zoneQueueIDs {
    all z1 : ZoneQueue, z2 : ZoneQueue | z1 ≠ z2 implies
    ↪ z1.id ≠ z2.id
}

// Two taxi drivers cannot drive the same taxi
fact singleDriverTaxi{
    all d1 : TaxiDriver, d2 : TaxiDriver | d1 ≠ d2
    ↪ implies d1.taxi ≠ d2.taxi
}

// Each taxi must have exactly one driver
fact taxiHaveDriver {
    all t : Taxi | isDriven[t]
}

// Taxi driver must be in a queue to be available
fact availableQueue {
    all d : TaxiDriver | isNotAvailable[d] implies
    ↪ isNotInQueue[d]
}

// Not available taxi drivers cannot be in zone queues
fact notAvailableQueue {
    no d : TaxiDriver | isNotAvailable[d] and !(
    ↪ isNotInQueue[d])
}

// A Taxi driver can serve one ride at a time
fact notSharedDriver {
    all r1 : Ride, r2 : Ride | r1≠r2 implies r1.
    ↪ taxidriver≠r2.taxidriver
}

```

```

// Predicates

pred isDriven [ t: Taxi ] {
    some d : TaxiDriver | d.taxi = t
}

pred isAvailable [ d: TaxiDriver ] {
    some a: TaxiDriverAvailable | d.availability in a
}

pred isNotAvailable [ d: TaxiDriver ] {
    some a: TaxiDriverNotAvailable | d.availability in a
}

pred isInRide [ d: TaxiDriver ] {
    some r:Ride | d in r.taxidriver
}

pred isNotInQueue[ d: TaxiDriver]{
    #d.taxiqueue = 0
}

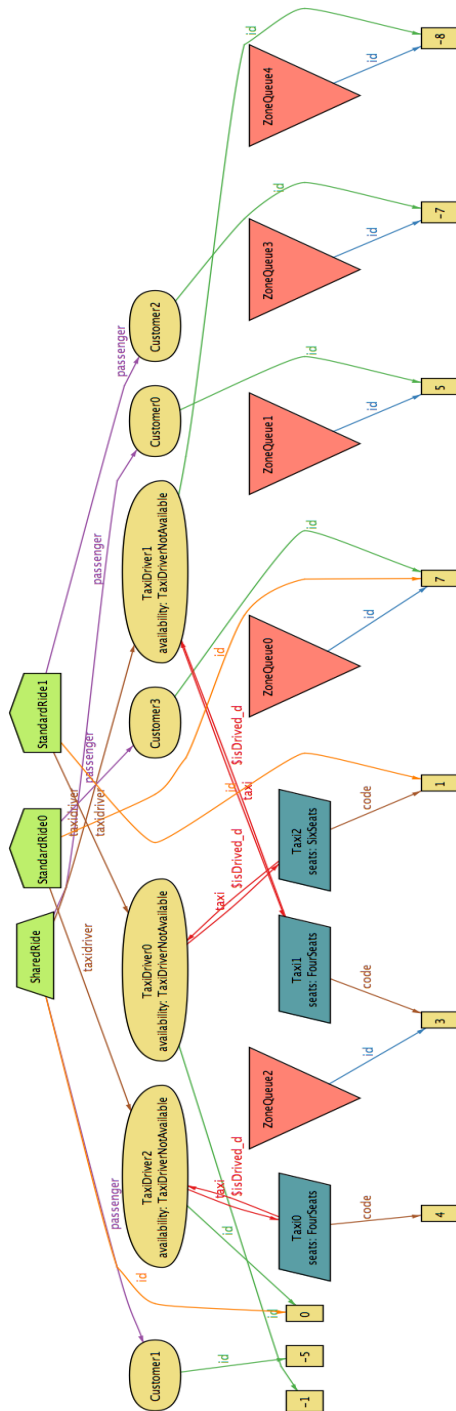
pred show {
    #Customer > 2
    #SharedRide = 1
    #StandardRide > 1
    #TaxiDriver >2
    #ZoneQueue >1
}

run show for 10

```


4.2 Generated World

Follows a set of images created using the Alloy Analyzer







5 Appendix

5.1 ToolBox

Follows a list of software, services and tools used during the redaction of this document.

- L^AT_EX framework (MacTeX on OS X and TeX Live on GNU/Linux) to generate the document.
- Various editors for editing the source : Vim, Sublime Text, Atom.
- Self-Hosted ShareLaTeX to collaboratively edit the document.
- Git to version the source, GitHub to host the repo.
- Balsamiq Mockups to create the mockups of the user interface.
- StarUML to create the use-case diagrams.
- Astah Professional to create the sequence diagrams.
- Gimp, Inkscape and ImageMagick to edit some images (.svg to .png)
- Official Alloy Analyzer to write and verify a basic model for the system
- Teamspeak used to organize conference-calls in order to work together.
- A modified version of this style sheet <https://github.com/Angtrim/alloy-latex-highlighting> to render the Alloy syntax.

5.2 Hours of Work

On average we tried to spend the same amount of hours on the project. For most of the parts we tried to do the work together, this is why the majority of the commits has been done from a single computer. For some parts we splitted the work and we did it by ourself or with the support of conference-calls. On average every one of us spent 30 hours in the redacting of this document.