POLITECNICO DI MILANO

SOFTWARE ENGINEERING 2

2015-2016

REQUIREMENTS ANALYSIS AND SPECIFICATIONS DOCUMENT (RASD)

Written by:

Bakti Melinda

Daniel Naveda

Chao Sun

Contents

1. Introduction

1.1. Description of the given problem

This document represents the Requirement Analysis and Specification Document (RSAD). The aim of this project is to determine the needs or conditions of the system, analyzing system requirements. Basically, the problem is the need to optimize the taxi service of a city, taking into account two main points, the first is to simplify the access of passengers to the service, and the second is guarantee a fair management of taxi queues. To address this issue, we will develop the project myTaxiService, which is a system able to optimize the taxi service of the city. As the main features of the system we have the UI for the Passengers and the Taxi Drivers. In this way, the system is going to receive the inputs from the Passengers and the Taxi Drivers and then, through a computation process, organize in the most efficient way possible how the resources (taxis) are going to be used by the Passengers.

In essence, the registered passengers only need to type in the name of the destination. The system will automatically get the exact taxi location data from the GPS API and schedule in the most efficient (cheapest, fastest) way to the destination.

On the other side, a taxi driver, when he/she is in the first place of the queue, will receive a “picking passenger up” request. If the driver accepts the request, then the deal is agreed upon. Otherwise, the request will send to the next taxi in the queue. This is the basic process about the taxi service and the advanced function will be introduced later in this paper.

1.2. Goals

Here is the list of goals that the mobile application should achieve:

- [G1] Help Passengers to find a taxi as soon as a taxi is available
- [G2] Help Taxi Drivers to get Passengers as soon as there is one available
- [G3] Allow an Unregistered Passenger become a Registered Passenger
- [G4] Allow an Unregistered Driver become a Registered Driver
- [G5] Allow a Registered Passenger to log in the web/mobile application
- [G6] Allow a Registered Driver to log in the web/mobile application
- [G7] Allow the Registered Passengers to find their location through GPS or by

typing in their address

- [G8] Allow Registered Passengers to find their destination through a search engine and/or a map
- [G9] Allow Registered Passengers to book a Taxi with 2 hours of anticipation or more
- [G10] Allow Registered Passengers to optionally share a Taxi with other Passengers to reduce the expenses
- [G11] Allow Registered Drivers to accept or deny a Registered Passenger's request
- [G12] Allow Registered Drivers to get the basic information of the upcoming Registered Passenger
- [G13] Validate the authenticity of the Taxi Drivers
- [G14] The system calculates the suitable route of the ride
- [G15] The system calculates the cost of the ride per each Passenger
- [G16] The system defines a Taxi queue per zone

1.3. Definitions, Acronyms, Abbreviations

1.3.1. Definitions

- Mobile App : Mobile Application
- Taxi : Vehicle used to transport passengers as a service provided
- Registered Passenger : A person who has registered as a Passenger on myTaxiService.
- Registered Driver : A Taxi Driver who has registered and has been authenticated by the System
- Driver Authentication : To make sure the driver is really existed and belong to the one taxi company
- Android application : An application created on the Android operating system
- Google map : An electronic map developed by Google company

1.3.2. Acronyms

- RASD : Requirements analysis and specification document
- GPS : Global Positioning System

- UI : User Interface
- DBMS : Database management system
- API : Application Programming Interface
- JVM : Java Virtual Machine
- JEE : Java Enterprise Edition
- SDK : Software Development Kit
- ADT : Android Development Tools

1.4. Actors

1. Visitor : All people who have not registered on myTaxiService System, regardless whether they are Taxi Drivers or Passengers. A visitor can only view the Log in screen. A registration is necessary to enjoy all the features provided by myTaxiService.

2. Registered passenger : After successfully logging in, this type of users can see their location on the map. Also, they can input the name of the destination and find this place on the map. Then, they can send the taxi request with the time (optional) and location. If the request is confirmed, then they can see the taxi code and the approximate arrival time to pick the Registered Passenger up.
3. Registered driver : After successfully logging, this type of user can see his/her location on a map. They can receive a Taxi Request and choose whether to accept it or not. If the request is accepted, they can get a basic information about the passenger's ride.
4. Administrator : This type of user is created to manage the user data and maintain the system. Also, the administrator has the right to confirm the registration of a driver.

1.5. References

- Specification document: "Assignments 1 and 2.pdf" assigned by the Professor of the course "Software Engineering 2".

- IEEE Std 830-1993 (Revision of IEEE Std 830-1984) IEEE Recommended Practice for Software Requirements Specifications

1.6. Overview

This document is organized in the following sections:

1. Introduction : Provide an overview of the entire document.
2. Overall Description : Describe the general factors that affect the product and its requirements.
3. Specific Requirements : Contain all the details required to design a system to satisfy these requirements.
4. Appendices : Contain extra information related to the document.

2. Overall Description

2.1. Product Perspective

MyTaxiService offers 2 separated but highly related applications, one for Passengers and one for Taxi Drivers. Both applications can only be used by people who have registered in the myTaxiService system. The core of myTaxiService is an automated based booking and dispatch platform targeted to be used for the Taxi Industry. It makes a city's taxi transportation system more efficient and safer.

2.2. Product Functions

Basically, this product is used for requesting taxi services by Registered Passenger, as well as finding passengers by Taxi Drivers. In addition to the basic functionality of getting a taxi and arrive at the destination, some other not-so-obvious features have been added, for example: sharing a taxi and divide the cost and reserve a taxi with a minimum of 2 hours of anticipation.

2.3. User Characteristics

We are expecting two kinds of users; the first one is a person who wants to easily request a taxi service, and the other kind of user is a taxi driver who wants to get passengers in a very efficient way.
Whether the user is a Passenger or a Taxi Driver, they must have a device (Smartphone for Taxi Drivers, Smartphone and PC for Passengers) with Internet connection.

2.4. Constraints

The constraints of this application are:

- Only users already registered can enjoy the features of this application.
- The city is divided in taxi zones of approximately 2 km^2
- Passengers can only request a taxi either through a web application or a mobile app.

2.5. Assumptions and Dependencies

There are few points that aren't really clear in the specification document, so we had to assume some facts. We assume that

- If multiple passengers are sharing a taxi, each Passenger's destination in on the same way
- There is only one account per user
- The taxi reservation has to occur at least two hours before the ride
- The system answers to the Passenger's request by sending the code and waiting time of the incoming taxi
- Taxi drivers manually use a mobile application to inform the system about their availability and to confirm that they are going to take care of a certain request.
- The system periodically assigns each taxi to their corresponding zone based on the GPS Information it receives from each taxi
- When a request arrives from a certain zone, the system forwards it to the first taxi queuing in that zone.

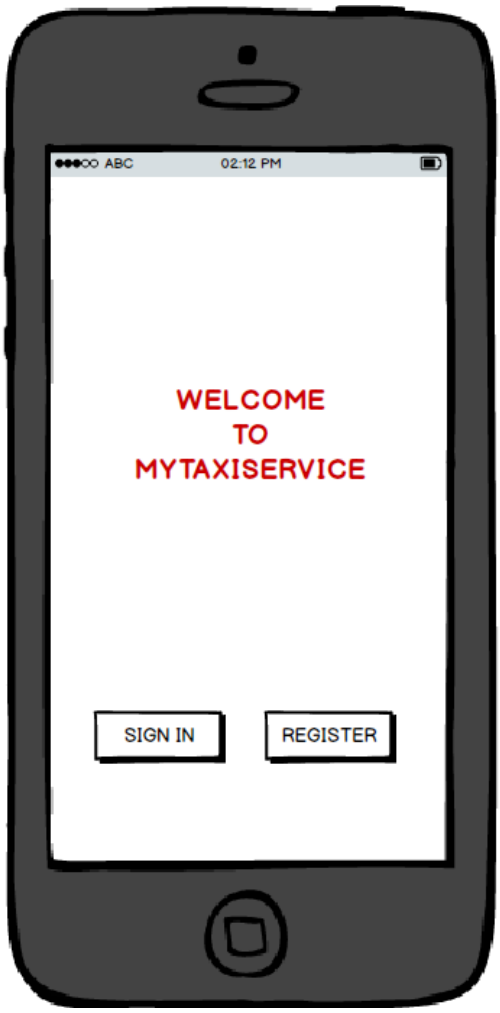
3. Specific Requirements

3.1. External Interface Requirements

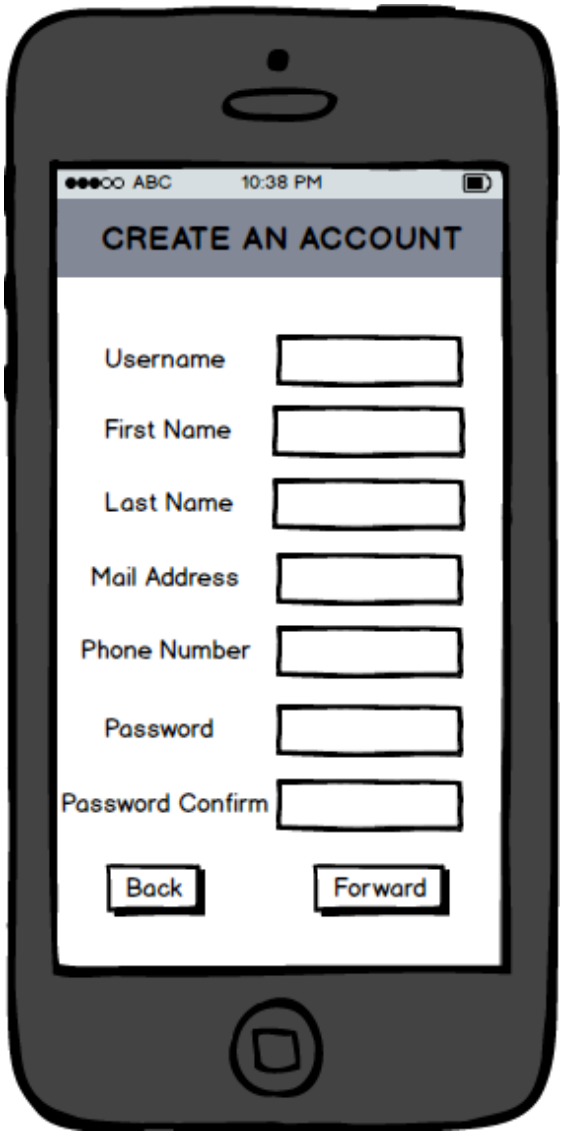
3.1.1. User Interfaces

We provide some mockups that can presents the structure of the application. And we divided it to two separated parts: Passenger Mode and Driver Mode.

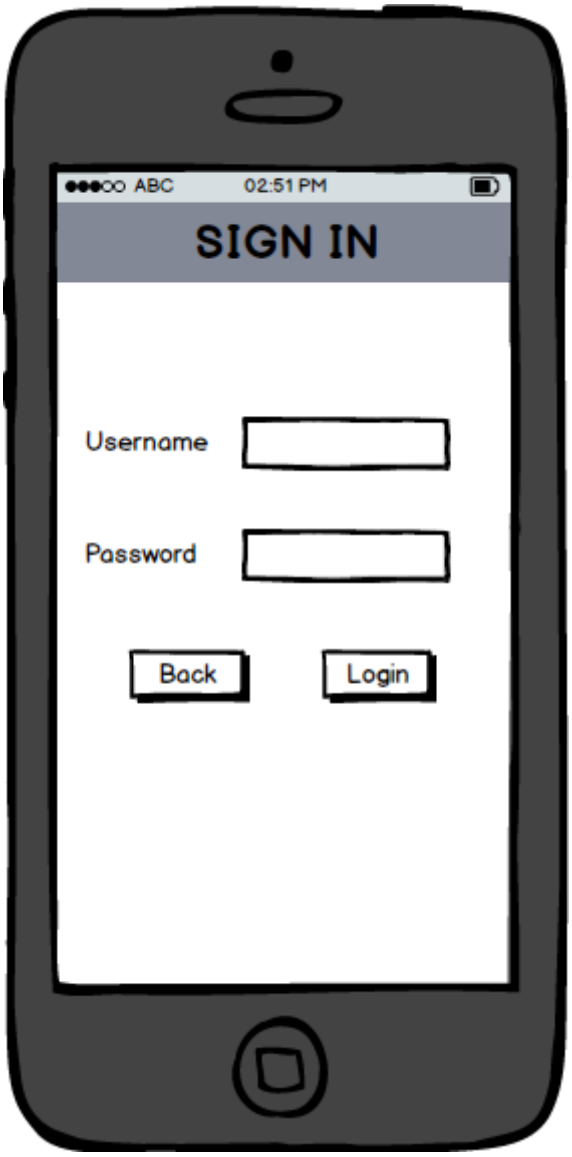
3.1.1.1. First Page screen (Passenger Mode)



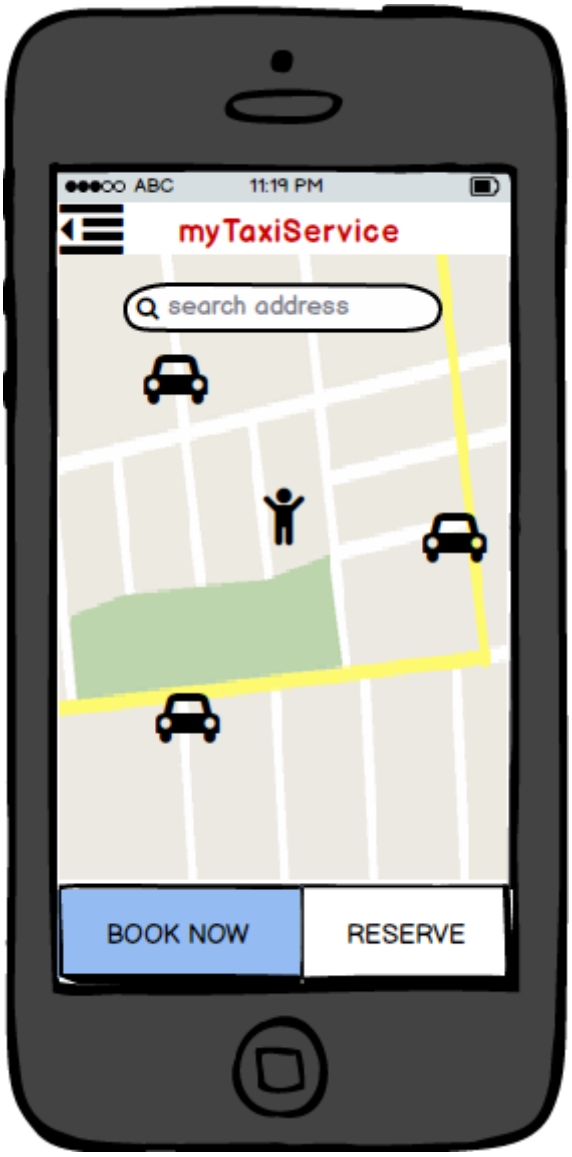
3.1.1.2. Sign up Page screen (Passenger Mode)



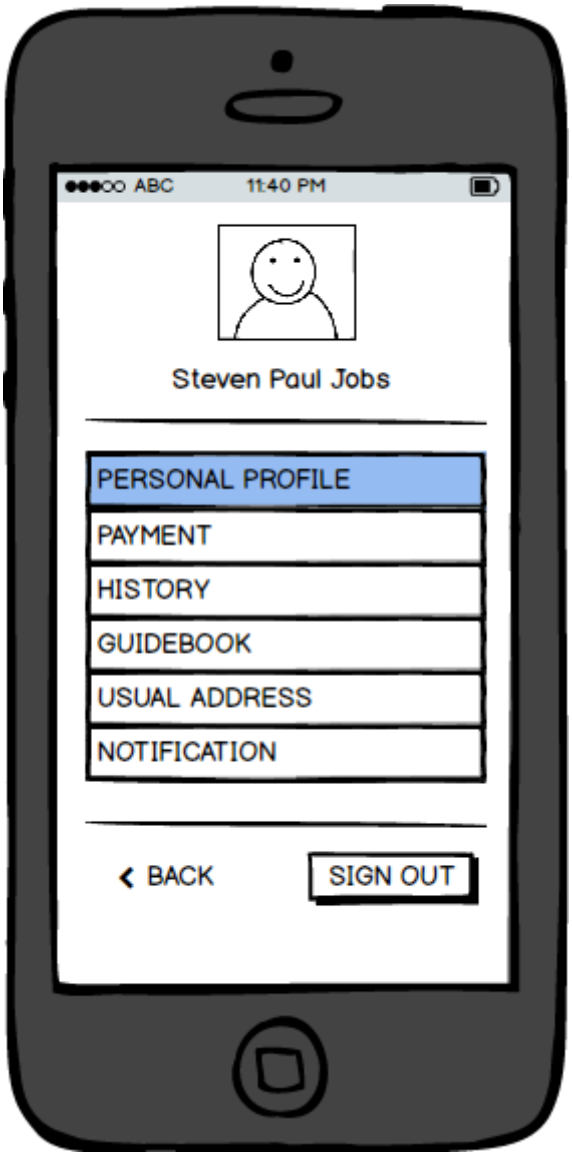
3.1.1.3. Log in Page screen (Passenger Mode)



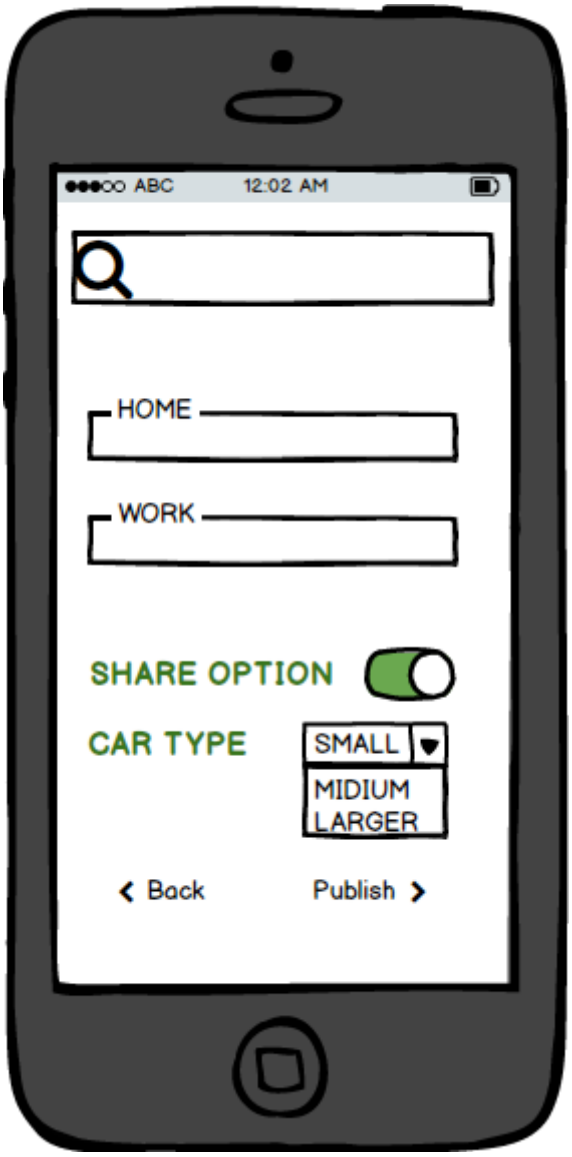
3.1.1.4. Main Page screen (Passenger Mode)



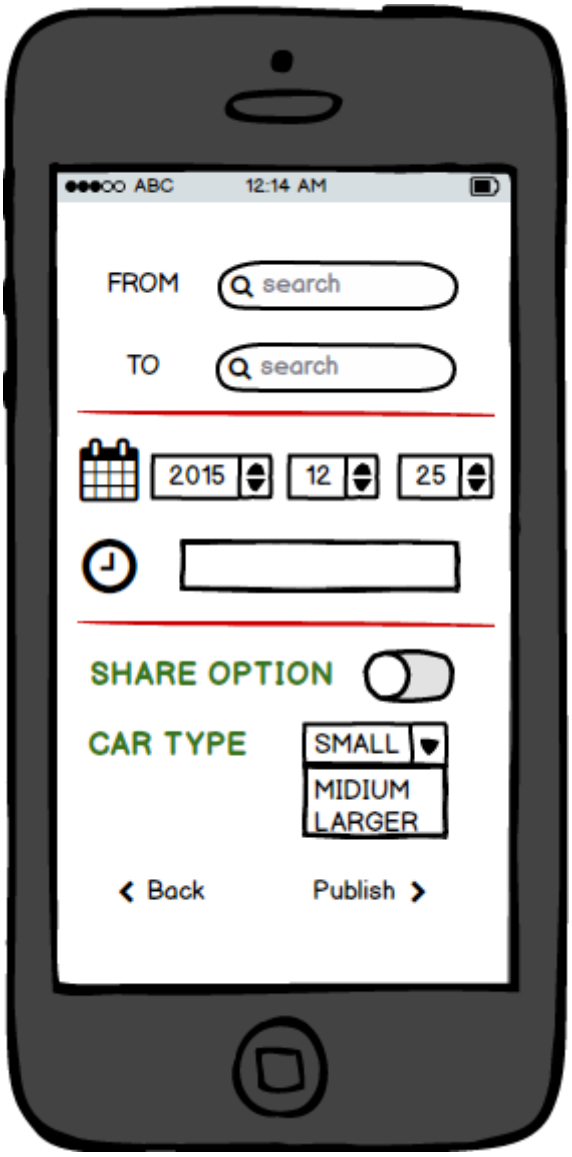
3.1.1.5. Account setting screen (Passenger Mode)



3.1.1.6. Order Right Now Page screen (Passenger Mode)

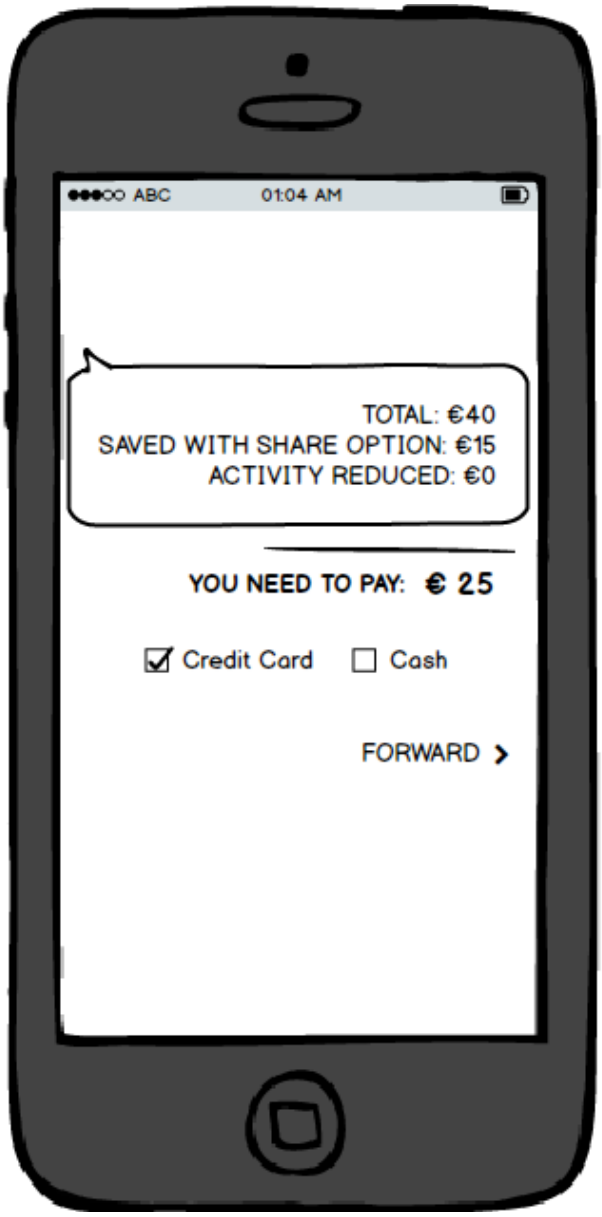


3.1.1.7. Reserve Page screen (Passenger Mode)



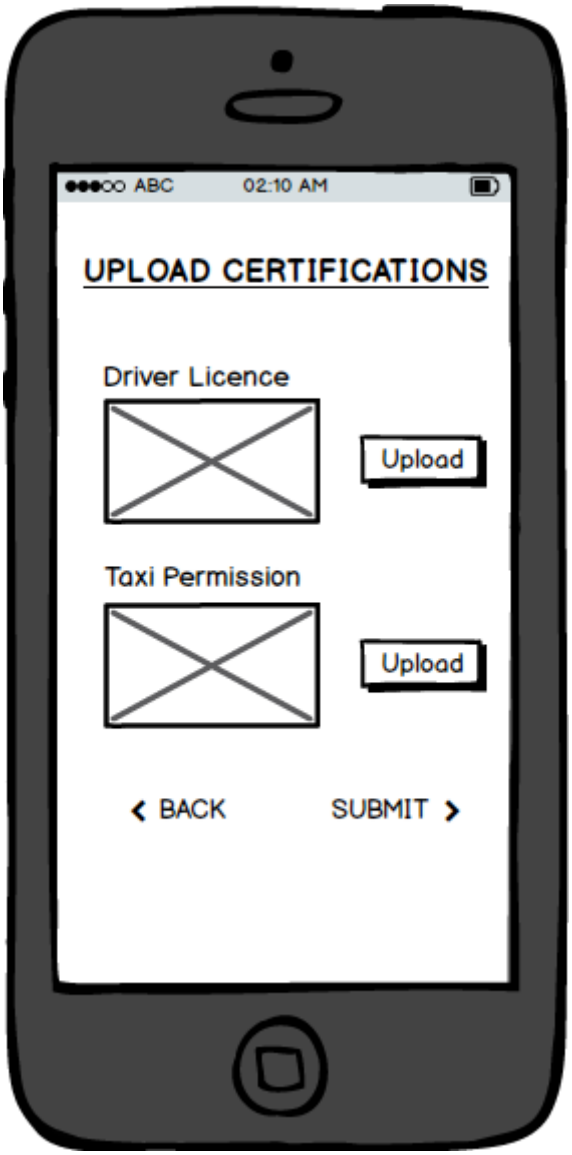
3.1.1.8. R*equest Page screen (Passenger Mode)*

3.1.1.9. Pay Page screen (Passenger Mode)

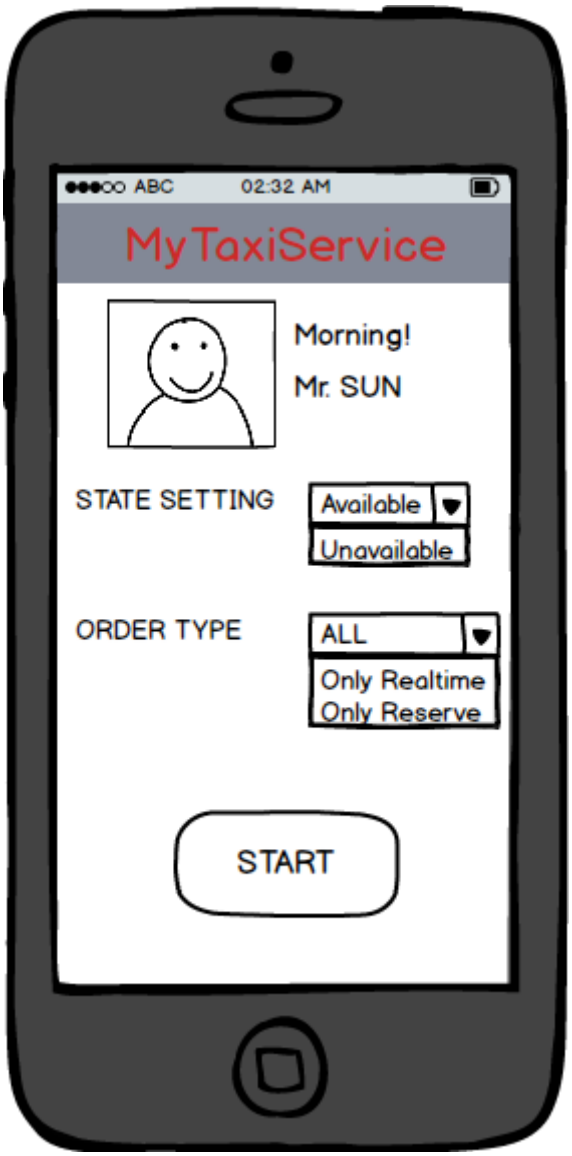


3.1.1.10. Upload Page screen (Driver Mode)

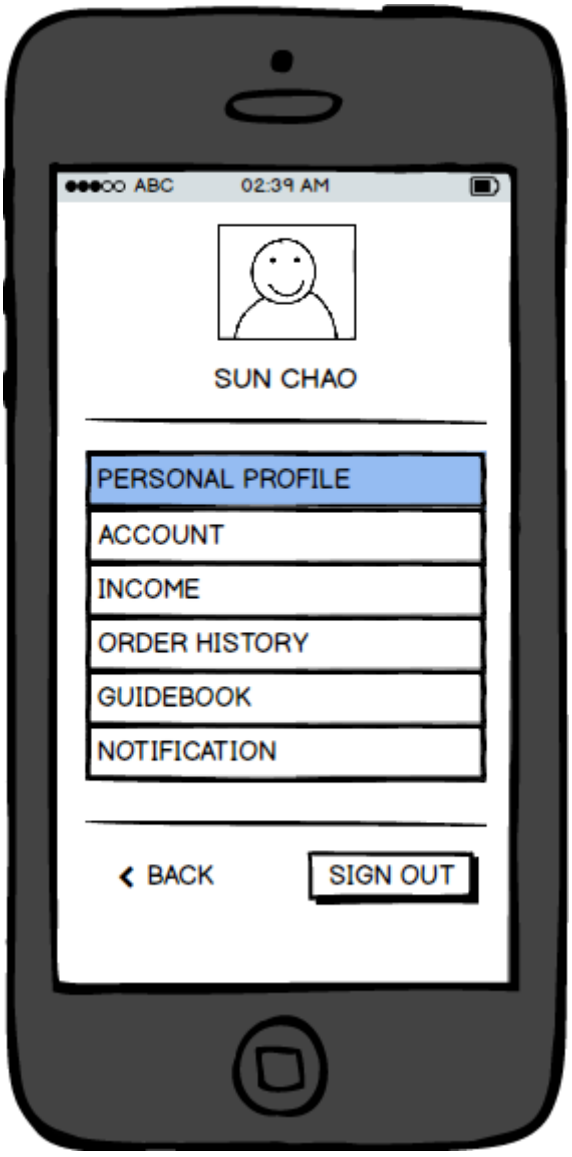
After guest registering to be a driver, he/she needs to upload some photos (documents) to prove his/her facticity. This step is completed on the “Upload Page”.



3.1.1.11. Main Page screen (Driver Mode)



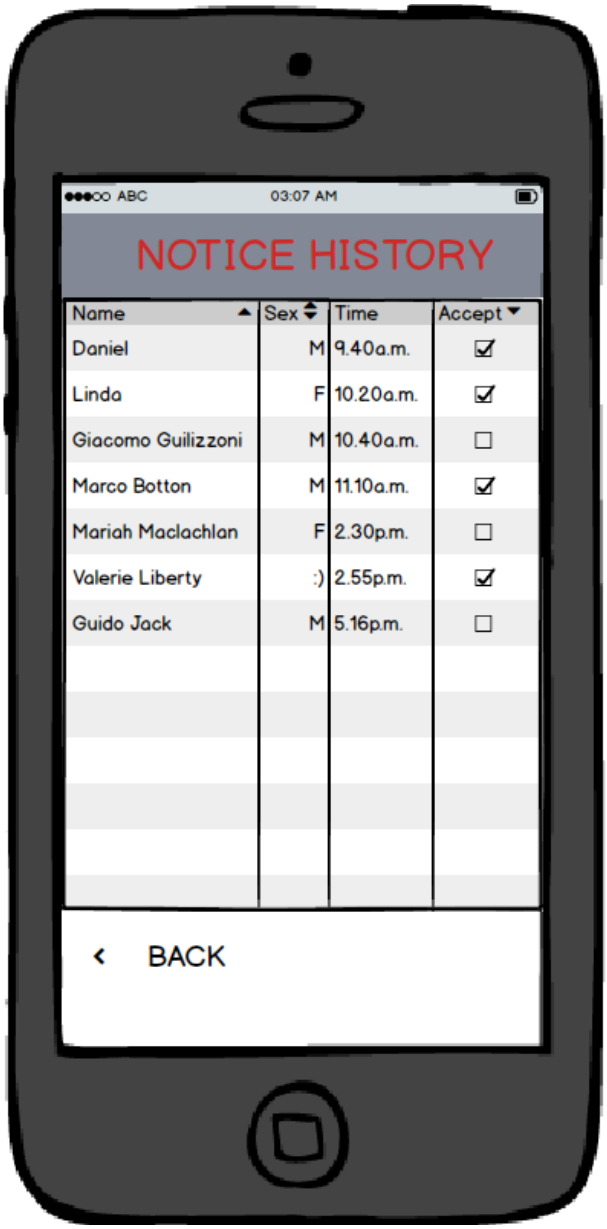
3.1.1.12. Personal Page screen (Driver Mode)



3.1.1.13. Waiting Page screen (Driver Mode)

3.1.1.14. Request Page screen (Driver Mode)

3.1.1.15. Request History Page screen (Driver Mode)



3.1.2. Hardware Interfaces

This project does not support any hardware interface.

3.1.3. Software Interfaces

For this stage of the project, the system does not have any Software Interface defined.

3.2. Functional Requirements

[G1] Help Passengers to find a taxi as soon as a taxi is available

- [R1] When a passenger requests a taxi, the system will assign the nearest taxi located through the GPS and which is first in the taxi queue
- [D1] The Passenger enters the correct data
- [D2] Accurate Taxi locations are provided by GPS

[G2] Help Taxi Drivers to get Passengers as soon as there is one available

- [R1] When a passenger wants to request a taxi service, the system will immediately assign the nearest taxi which is available for addressing the request
- [D1] When a passenger needs a taxi, the passenger will introduce the right information to request the ride

[G3] Allow an Unregistered Passenger become a Registered Passenger

- [R1] Visitor can register in the register field to become a member (passenger or driver) of myTaxiService
- [R2] Visitor use his email to register
- [R3] Visitor can only sign up once using correct email address

[G4] Allow an Unregistered Driver become a Registered Driver

- [R1] Visitor can register in the register field to become a member (passenger or driver) of myTaxiService
- [R2] Visitor use his email to register
- [R3] Visitor can only sign up once using correct email address

[G5] Allow a Registered Passenger to log in the web/mobile application

- [R1] User have to registered before
- [R2] User successfully login in login process
- [R3] User must to know his username and password to login
- [R4] Wrong username and password will not allowed to enter the aplication

[G6] Allow a Registered Driver to log in the web/mobile application

- [R1] User have to registered before
- [R2] User successfully login in login process
- [R3] User must to know his username and password to login
- [R4] Wrong username and password will not allowed to enter the aplication

[G7] Allow the Registered Passengers to find their location through GPS or by typing in their address

- [R1] User can see where they are now
- [R2] User can see the map

[G8] Allow Registered Passengers to find their destination through a search engine and/or a map

- [R1] User can see the route of their destination
- [R2] User can send their request to the system
- [R3] User can see available taxi near of them

[G9] Allow Registered Passengers to book a Taxi with 2 hours of anticipation or more

- [R1] User can reserve a taxi at least two hours before the ride

[G10] Allow Registered Passengers to optionally share a Taxi with other Passengers to reduce the expenses

- [R1] User can share taxi with another user in the same destination
- [R2] User can share the cost of the taxi

[G11] Allow Registered Drivers to accept or deny a Registered Passenger's request

- [R1] The driver can confirm or reject the request on the application

[G12] Allow Registered Drivers to get the basic information of the upcoming Registered Passenger

- [R1] For the sharing taxi, the driver get the information of the coming passenger from the first

[G13] Validate the authenticity of the Taxi Drivers

- [R1] The system must know the truth of the registered driver

[G14] The system calculates the suitable route of the ride

- [R1] The system must be ready to know the route for the passenger

[G15] The system calculates the cost of the ride per each Passenger

- [R1] The system must know how much the passenger pay for the taxi service
- [D1] The system must know how to calculate if the passenger share the taxi

[G16] The system defines a Taxi queue per zone

- [R1] The system guarantees a fair management of taxi queues

3.3. Scenarios

Here are some possible scenarios of myTaxiService:

3.3.1. Scenario 1

Rachel needs to go to a friend's wedding, since her car is being repaired, she is going to use myTaxiService app on her smartphone. She carefully read at the event's address and type in that information onto the app. After following its instructions, she is able to send the request to the system; few minutes after, she receives a code and a waiting time of 7 minutes.
After waiting approximately 7 minutes, she confirms his route with the taxi driver by giving him the code she had previously received. Rachel gets in the taxi and she arrives at the wedding in time.

3.3.2. Scenario 2

Mark, a taxi driver, is working late in the night and wants to take one last passenger. He looks at his myTaxiService app, and sees that there is a new passenger request. By clicking on "Accept" he has confirmed that he is going to pick that passenger up.

Mark goes to the passenger's location and verifies through the code that the passenger is the specified by the system. Mark takes the passenger to the right destination and then Mark goes home to have a rest after a long day of work.

3.3.3. Scenario 3

Susan is going to a famous concert in the city, but it is quite far away from her location. Since she knows that a lot of people are going there as well, she wants to book a taxi through myTaxiService and enable the sharing option. She knows that there is a big possibility to share a taxi with somebody else who is going whether to the concert or somewhere else near. That way she would pay less for the travel.

Through her laptop, she goes to the web application and logs into her account. She sets the destination address and enables the option “share taxi”. Afterwards, she sets also other destinations apart from the concert one to increase the possibility to share the taxi.

After few minutes of processing time, the system sends Susan a notification which specifies that there is another passenger who will join her to the way to the concert, and also the cost is split.

3.3.4. Scenario 4

John will travel to another country for business purposes. His flight will take off in 6 hours. He wants to make sure that he is not going to arrive late at the airport. For this reason he has decided to reserve a taxi through the myTaxiService app.

Once in his myTaxiService account, he sets the reservation for a taxi that should arrive in the next 3 hours from now. After setting the time and destination, the system informs him about the ride, and he gets his confirmation code.

Three hours later, the taxi is there waiting for him. John gets in the taxi and he gets prepared for his amazing travel.

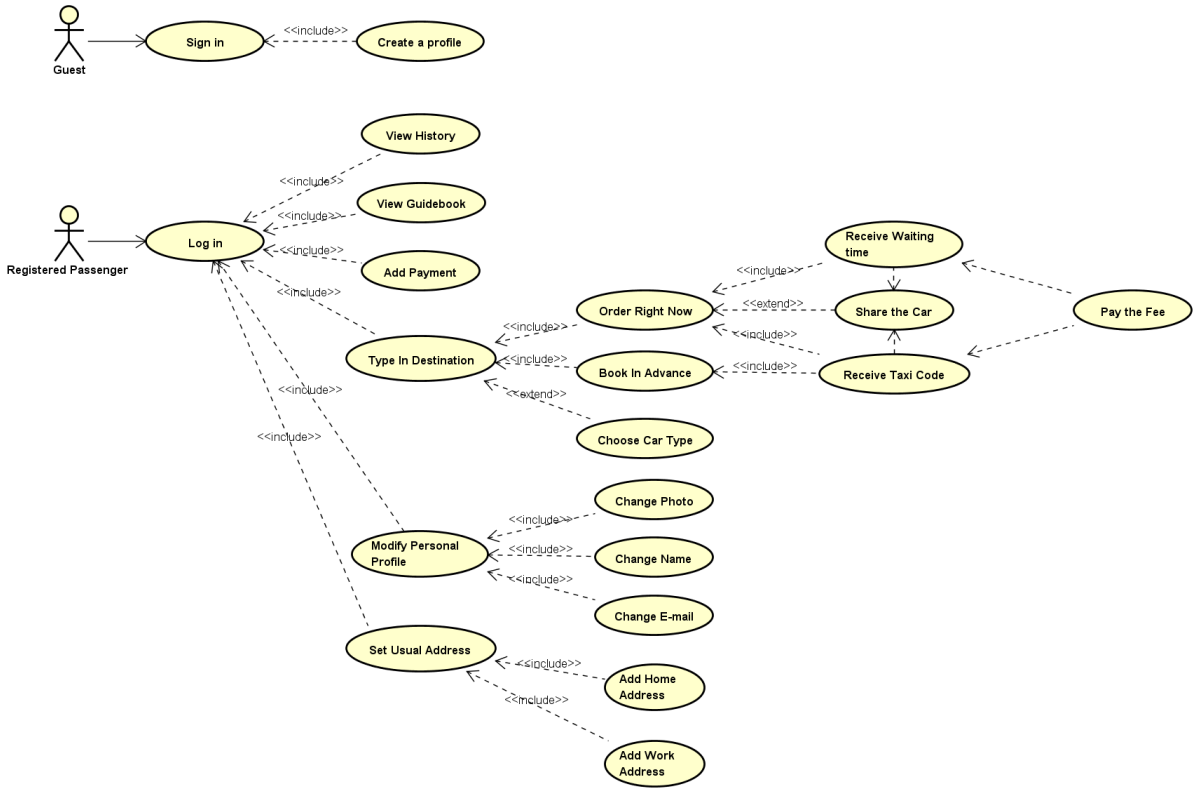
3.4. UML Models

3.4.1. Use Case Diagram

We can drive some use cases from the scenarios described above MyTaxiService System is composed with two separated applications, one for registered passenger and the other for registered driver. So split it into two independence use case diagrams, although they have connection inside.

3.4.1.1. Use Case Diagram of passenger

Here it is the Use Case Diagram of passenger:



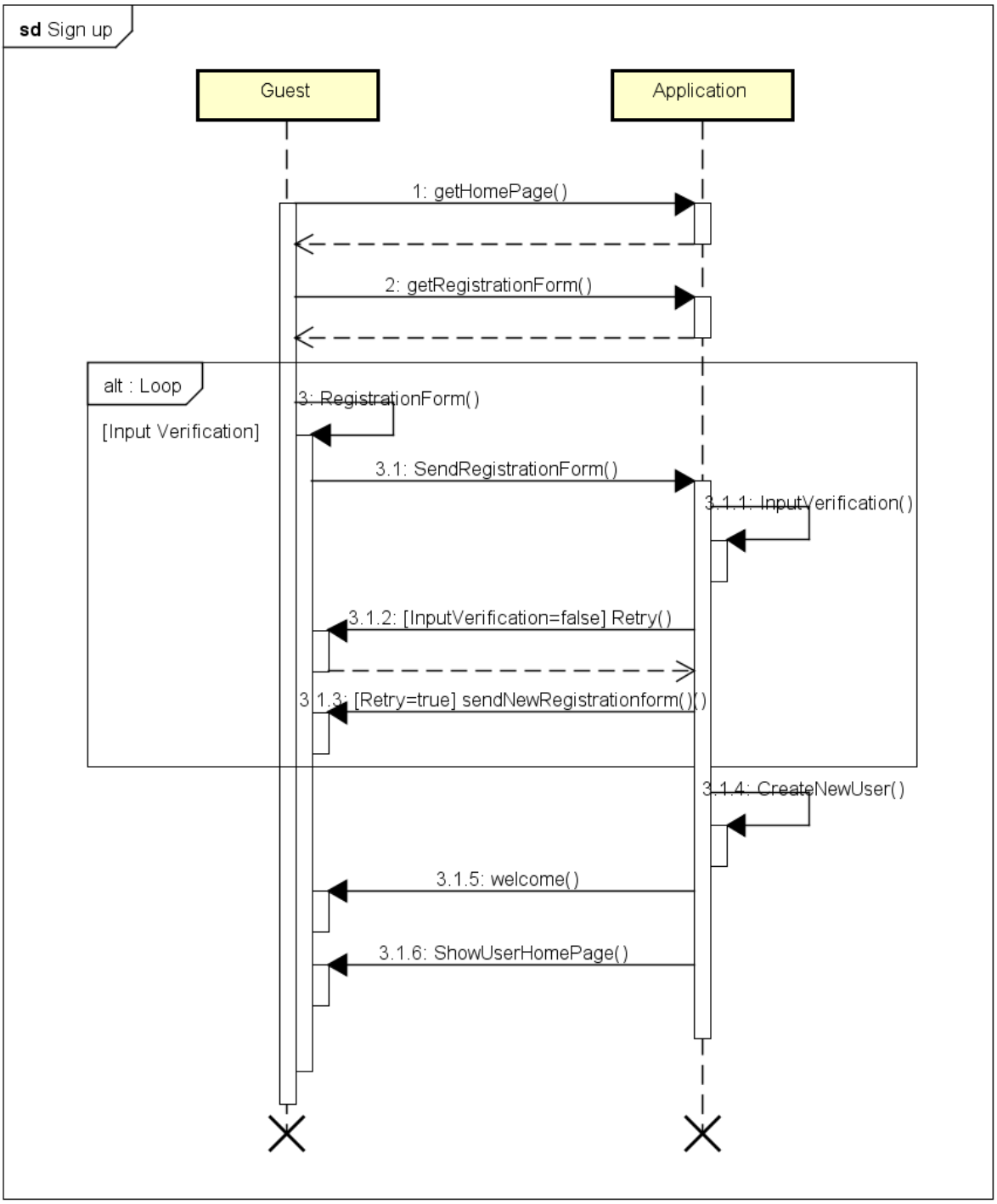
3.4.2. Use Case Description

Refine the use case “Sign up”:

Name	**Sign up**
Actors	Guest
Entry Conditions	The guest doesn’t register the application.
Flow of events	• The guest enters the application • The guest clicks on the “SIGN UP” button • The guest fills the registration form where he\she has to write: Mandatory ones: - First name - Last name

	- Mobile number - Email address - User name - Password - Confirm password Optionally ones: - Date of Birth - Address - Photo • The guest clicks “FINISH” button • The system shows him his personal page
Exit conditions	Registration successfully done.
Exceptions	The username which the guest inserts already exists. The mobile number has been associated to another account. The mandatory ones of the form are not be filled. The two passwords do not match.

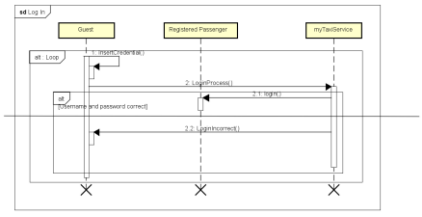
Here is the sequence diagram of “Sign up”



Refine the use case “Log in”:

Name	**Log in**
Actors	Guest, Registered Passenger, Registered Driver
Entry conditions	Guest has registered as a passenger or driver before.
Flow of events	• The guest enters the application; • The user fills in the text fields on the home page with username and password. • The user clicks on the “LOG IN” button.
Exit conditions	• Guest types the correct username and password, the application shows the main page after verified. • Guest are promoted to a registered user.
Exceptions	The password and/or username inserted by the user are wrong. The System notifies that to Guest and show the login page.

Here is the sequence diagram of “Log In”

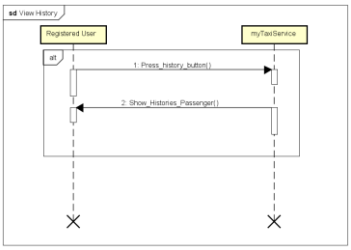


Refine the use case “View History”

Name	**View History**
Actors	Registered Passenger
Entry conditions	Registered Passenger must be logged in.
Flow of events	• The passenger enters the application;

	• The user clicks on the “PROFILE” button (present with a photo). The system will show him a setting interface with some new buttons; • The user clicks on the “HISTORY” button;
Exit conditions	The user clicks on the “X” button to go back to setting interface.
Exceptions	There are no possible exceptions.

Here is the sequence diagram of “View History”

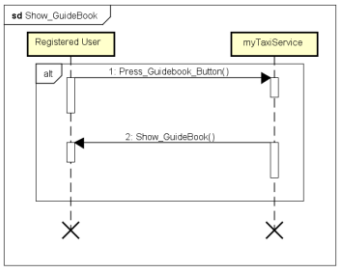


Refine the use case “View Guidebook”

Name	**View Guidebook**
Actors	Registered Passenger
Entry conditions	Registered Passenger must be logged in.
Flow of events	• The passenger enters the application; • The user clicks on the “PROFILE” button (present with a photo). The system will show him a setting interface with some new buttons; • The user clicks on the “Guidebook” button;
Exit conditions	The user clicks on the “X” button to go back to setting

	interface.
Exceptions	There are no possible exceptions.

Here is the sequence diagram of “View Guide Book”



Refine the use case “Add Payment”

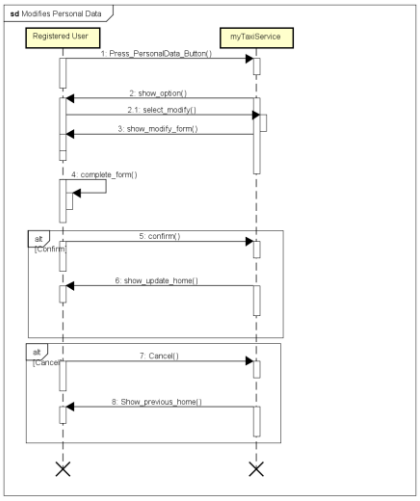
Name	**Add Payment**
Actors	Registered Passenger
Entry conditions	• Registered Passenger must be logged in. • The billing information doesn’t be inserted before.
Flow of events	• The passenger enters the application; • The user clicks on the “PROFILE” button (present with a photo). The system will show him a setting interface with some new buttons; • The user clicks on the “Payment” button. The system will turn to the Payment page. • The user fulfills the form with the related information of his credit card and clicks on “save”. • If the related information is valid, the page will turn

	back to profile interface and they will be stored. Else, the system will show the error page and reload.
Exit conditions	The page is reloaded with the correct information, the card information is stored. The information of the bank card is error.
Exceptions	The quantity of the digit is invalid. The System notifies that to user and asks to rewrite again after checking. The numbers of the bank card have been used before. The system notified that to user.

Refine the use case “Modify Personal Profile”

Name	**Modify Personal Profile**
Actors	Register Passenger
Entry conditions	Registered Passenger must be logged in.
Flow of events	• The registered passenger enters the application; • The user clicks on the “PROFILE” button (present with a photo). The system will show him a setting interface with some new buttons; • The user choose “MODIFY PROFILE” button; And the user only can modify the following things: - Photo; - First Name; - Last Name; - E-mail; • After the modification, click “SAVE” button to keep this change.
Exit conditions	The page is reloaded and new data has been stored.
Exceptions	The text field is empty. The System notifies that to user and asks to rewrite again after checking.

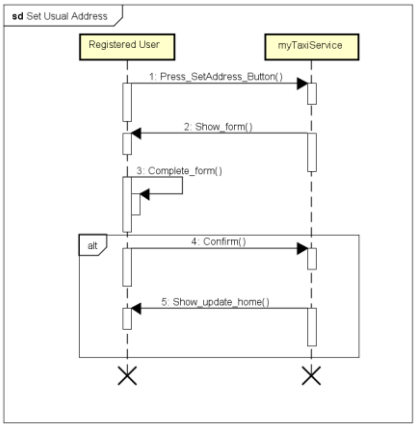
Here is the sequence diagram of “Modify Personal Profile”



Refine the use case “Set Usual Address”

Name	**Set Usual Address**
Actors	Register Passenger
Entry conditions	Registered Passenger must be logged in.
Flow of events	• The registered passenger enters the application; • The user clicks on the address field. Then, the application will turn to a new page. • On the new page, the user chicks on the “HOME” button and type in the address. Then, click on the “SAVE” button. • On the new page, the user chicks on the “WORK” button and type in the address. Then, click on the “SAVE” button.
Exit conditions	The page is reloaded and new data has been stored.
Exceptions	There are no possible exceptions.

Here is the sequence diagram of “Set Usual Address”



Refine the use case “Order Right Now”

Name	**Order Right Now**
Actors	Register Passenger
Entry conditions	The address typed in the destination field is valid.
Flow of events	• The registered passenger enters the application; • The user clicks on the address field. Then, the application will turn to an address page. • If the user have set the address for the “HOME” button and “WORK” button, he/she can choose it as the

	destination address. Then the map implanted will calculate and show the routes to the end. • If not, the user can type the address in the text field and press “SEARCH” button to find the exact place on the map. • After selected the destination address, the passenger clicks on the “ORDER RIGHT NOW” button to order a taxi right now.
Exit conditions	If there is no taxi available on this area, the passenger will keep waiting. Or he/she can press the “STOP” button to stop ordering a taxi. If there are some taxis available on this area, the passenger will soon get the taxi code and the probably waiting time.
Exceptions	The address cannot be found on the map. The user clicks on the button “cancel” and retype a new address for searching.

Refine the use case “Book in Advance”

Name	**Book in Advance**
Actors	Register Passenger
Entry conditions	The address typed in the destination field is valid.
Flow of events	• The registered passenger enters the application; • The user clicks on the address field. Then, the application will turn to an address page. • The passenger clicks on the “BOOK IN ADVANCE” button. Then, the application will turn to a reservation page. • Type in the exactly time that he/she want to book one taxi. • If the user have set the address for the “HOME” button and “WORK” button, he/she can choose it as the destination address. Then the map implanted will calculate and show the routes to the end. • If not, the user can type the address in the text field

	and press “SEARCH” button to find the exact place on the map. • Click on the button “Done” to finish the book.
Exit conditions	The system confirms the reservation to the user and allocates a taxi to the request 10 minutes before the meeting time with the user. If there is no taxi available on this area at that period, the system will reject the request and reflect this to the user. If there are some taxis available on this area, the passenger will get the taxi code and the probably waiting time 10 minutes before.
Exceptions	The address cannot be found on the map. The user clicks on the button “cancel” and retype a new address for searching. The exactly time typed in is less than 2 hours. The System notifies that to user and asks to retype the time 2 hours later.

Refine the use case “Choose Car type”

Name	**Choose Car type**
Actors	Register Passenger
Entry conditions	Registered Passenger must be logged in.
Flow of events	• The registered passenger enters the application; • The user chooses one car type shown on the main page.
Exit conditions	New data has been stored.
Exceptions	There are no possible exceptions.

Refine the use case “Receive Taxi Code”

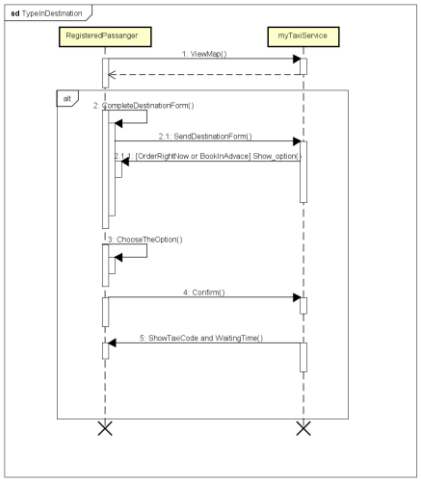
Name	**Receive Taxi Code**
Actors	Register Passenger
Entry conditions	A taxi has been booked in advance or ordered at present. There exists available taxi on that area.
Flow of events	If passenger books a taxi successfully (no matter in which way), they will receive a taxi code from the system. It is used

	to find the taxi.
Exit conditions	Passenger has received one.
Exceptions	There are no possible exceptions.

Refine the use case “Receive Waiting Time”

Name	**Receive Waiting Time**
Actors	Register Passenger
Entry conditions	A taxi has been booked in advance or ordered at present. There exists available taxi on that area.
Flow of events	If passenger books a taxi successfully (no matter in which way), they will receive a taxi code from the system. It is used to find the taxi.
Exit conditions	Passenger has received one.
Exceptions	There are no possible exceptions.

Here is the sequence diagram of “Type in Destination”

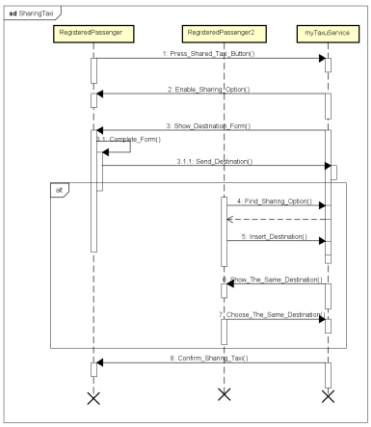


Refine the use case “Share the Car”

Name	**Share the Car**
Actors	Registered Passenger
Entry conditions	The registered passenger is logged in. The user will order a taxi right now. The address typed in the destination field is valid.

Flow of events	• The registered passenger enters the application; • The user clicks on the address field. Then, the application will turn to an address page. • Choose a valid destination address. • Pitch on “SHARE THE CAR” option. • Click on “ORDER RIGHT NOW” button
Exit conditions	New data has been stored and show “√” ahead of “SHARE THE CAR” option.
Exceptions	There are no possible exceptions.

Here is the sequence diagram of “Share the Car”

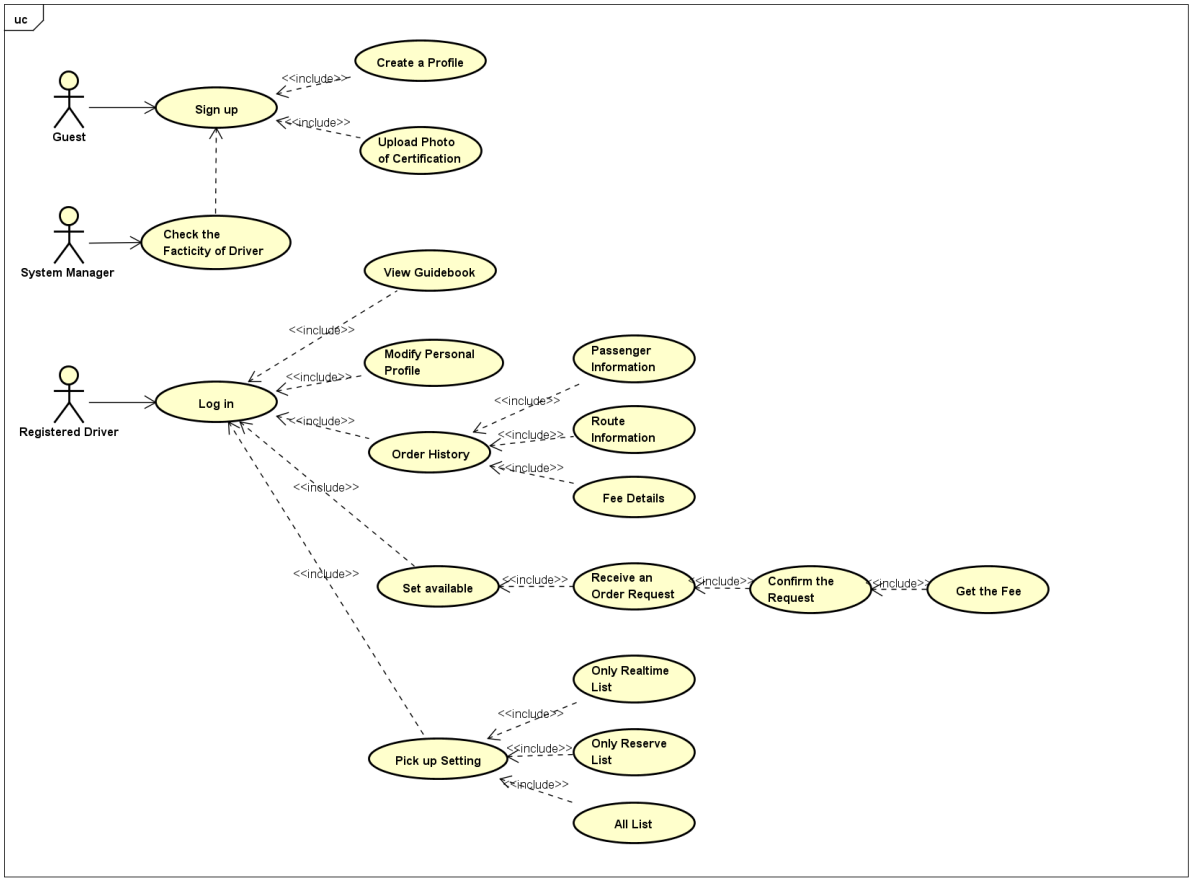


Refine the use case “Pay the Fee”

Name	**Pay the Fee**
Actors	Registered Passenger
Entry conditions	The user has book a taxi successfully. The user has been sent to the destination.
Flow of events	• Passenger books a taxi successfully, either in “ORDER RIGHT NOW” option or “BOOK IN ADVANCE” option. • The taxi sent the passenger to the destination. • Before getting off the car, passenger should pay for the cost. There are two ways for the user to pay: - By cash; - By credit card which has been added to the system already.
Exit conditions	Driver has confirmed the payment.
Exceptions	Remaining sum is not enough in the credit card.

3.4.1.2 Use Case Diagram of driver

Here it is the Use Case Diagram of Driver:



Some use cases in the subsystem of the driver part is similar to that in the subsystem of the passenger part. So we omit them and only draw the key ones to refine the main function.

Refine the use case “Upload Photo of Certification”

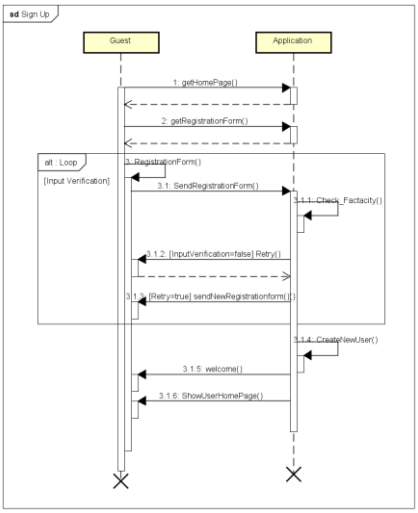
Name	**Upload Photo of Certification**
Actors	System Manager, Guest, Registered Driver
Entry conditions	The user doesn’t pass the authentication by the system manager. The Guest can upload the photos of certification when he signs up for the application. The registered driver can upload the photos of certification when he logs in the application.
Flow of events	• The guest enters the application • The guest clicks on the “SIGN UP” button • The guest fills the registration form where he\she has to write: Mandatory ones: - First name - Last name - Mobile number - Email address - User name - Password - Confirm password Optionally ones: - Date of Birth - Address - Photo • The guest clicks “NEXT” button. The system shows him the upload page. He should upload: - Driver license; - Taxi permission; • The guest clicks “FINISH” button to send all the information to the system. And the system manager will check his certificates later.

Exit conditions	Upload success
Exceptions	The registered driver have passed the detection.

Refine the use case “Check the Facility of Driver”

Name	**Check the Facticity of Driver**
Actors	System Manager, Guest, Registered Driver
Entry conditions	The guest has signed up to be a registered driver. The related certification documents have been uploaded successfully. The registered driver is not active.
Flow of events	• There are two ways for the users to upload the photos of their certification. 1. After signed in, the system will reload to the upload page. 2. If user skips, they can also find the upload page on the profile page. • System manager can see all the documents of each registered driver. Then they will check the authenticity from these documents.
Exit conditions	If passed, system manager will set the state of the registered driver to PASSED. Else, the state will be set to NOT PASSED and then system will notify this to user.
Exceptions	There are no possible exceptions.

Here is the sequence diagram of “Sign Up and Check Facticity of the Guest”

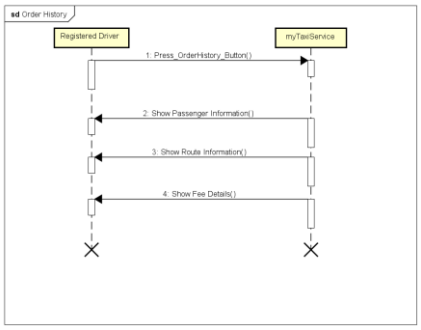


Refine the use case “Order History”

Name	**Order History**
Actors	Registered Driver
Entry conditions	Registered Driver must be logged in.

	The state of the registered driver should be “PASSED”
Flow of events	• The driver enters the application; • The user clicks on the “PROFILE” button (present with a photo). The system will show him a setting interface with some new buttons; • The user clicks on the “HISTORY” button; • In this section, driver user can see: - The information of the passengers who have ordered his/her taxi before. - The information of the routes. - The details of the car fare.
Exit conditions	The user clicks on the “X” button to go back to setting interface.
Exceptions	There are no possible exceptions.

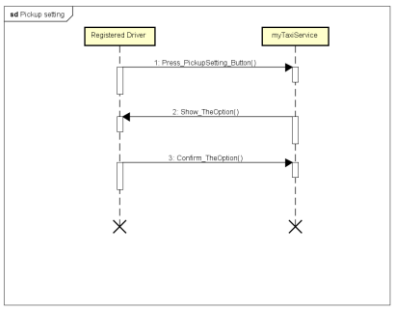
Here is the sequence diagram of “Order History”



Refine the use case “Pick up Setting”

Name	**Pick up Setting**
Actors	Registered Driver
Entry conditions	Registered Driver must be logged in. The state of the registered driver should be “PASSED”
Flow of events	Drivers can choose which type of passenger they want to pick up. Passengers on real-time list; Passengers on reserve list; Passengers on all list; After logged in, registered driver can change “Pick Up Setting” by selecting from three options.
Exit conditions	The option button is on the main page.
Exceptions	There are no possible exceptions.

Here is the sequence diagram of “Pick up Setting”



Refine the use case “Set Available”

Name	**Set Available**
Actors	Registered Driver
Entry conditions	Registered Driver must be logged in. The state of the registered driver should be “PASSED”
Flow of events	• Registered driver logs in the application and press “START” button to set him “Available”. • The system reload the page.
Exit conditions	The page is reloaded and new data has been stored.
Exceptions	There are no possible exceptions.

Refine the use case “Receive an Order Request”

Name	**Receive an Order Request**
Actors	Registered Driver
Entry conditions	Registered Driver must be logged in. The state of the registered driver should be “PASSED” It is “Available” for the driver to work The taxi driver is on the first place of the queue
Flow of events	System send the registered driver an order request from passengers.
Exit conditions	Driver can choose to confirm or reject the request
Exceptions	There are no possible exceptions.

Refine the use case “Confirm the Request”

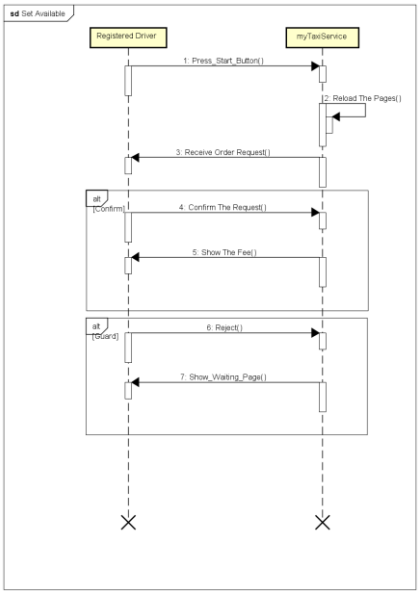
Name	**Confirm the Request**
Actors	Registered Driver
Entry conditions	Registered Driver must be logged in. The state of the registered driver should be “PASSED” It is “Available” for the driver to work The taxi driver is on the first place of the queue The taxi driver has received an Order Request
Flow of events	Registered driver received the request message. And in this

	dialog box, there two buttons. One is “CONFIRM” and one for “REJECT”. Driver clicks on the “Confirm” button.
Exit conditions	If driver clicks “CONFIRM” button, the taxi will be thrown out of the waiting queue and the system will reload to the counting page; If driver clicks “REJECT” button, the taxi will be put on the tail of the waiting queue and the system will reload to the waiting page again.
Exceptions	There are no possible exceptions.

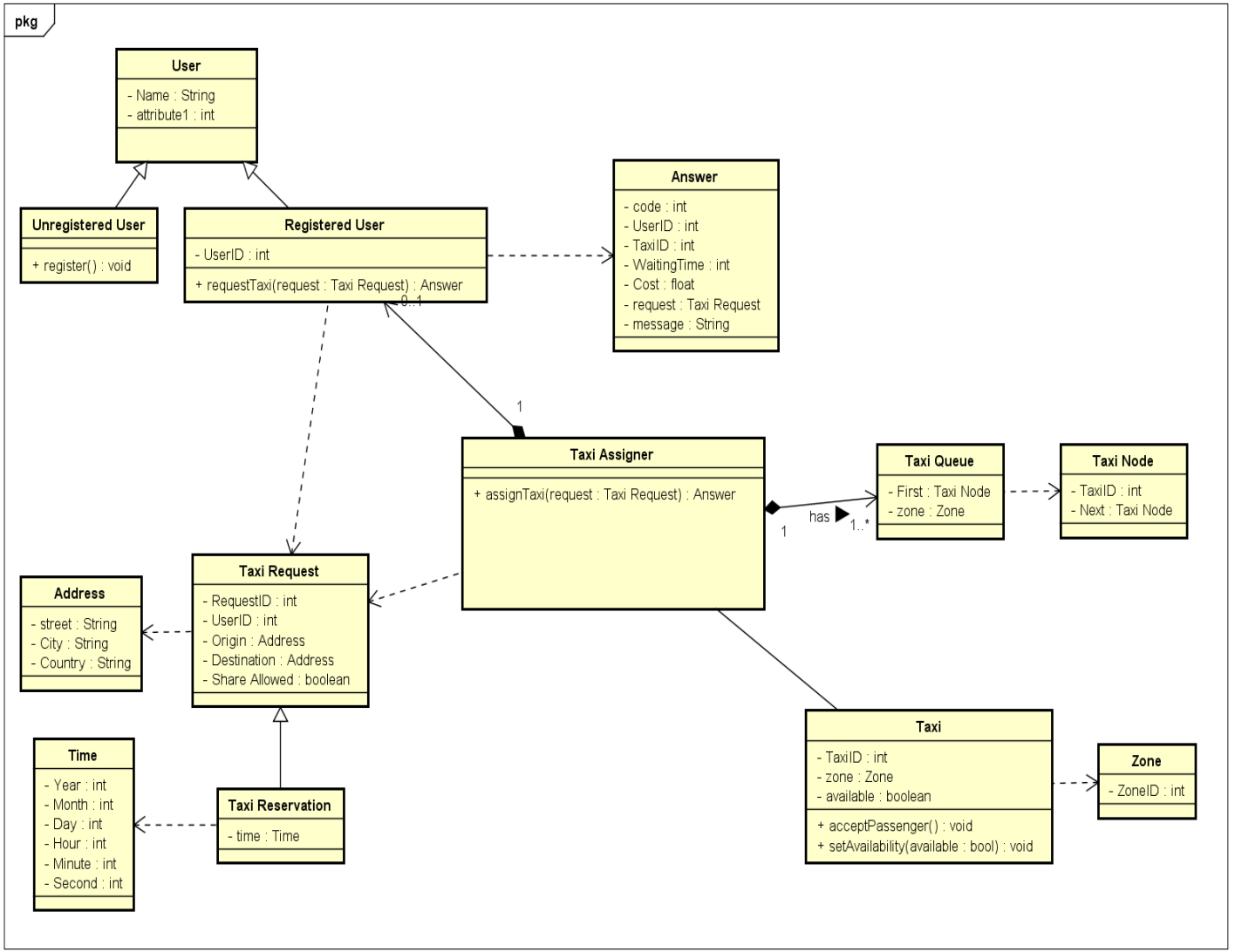
Refine the use case “Get the Fee”

Name	**Get the Fee**
Actors	Registered Driver, Registered Passenger
Entry conditions	Registered Driver must be logged in. The state of the registered driver should be “PASSED” It is “Available” for the driver to work The taxi driver is on the first place of the queue The taxi driver has received an Order Request The taxi driver has confirmed the request and has sent the passenger to the destination
Flow of events	After the payment by the passenger, the driver can see the final page of this process.
Exit conditions	The related data will be restored and the system will reload to the waiting page for the next passenger.
Exceptions	The remaining sum is not enough to pay the taxi fee.

Here is the sequence diagram of “Set Available”



3.4.3. Class Diagrams



3.5. Non Functional Requirements

3.5.1. Performance Requirements

The performance of the system must be optimal in order to guarantee an appropriate response approximately less than 15 minutes.

3.5.2. Design Constraints

3.5.3. Software System Attributes

3.5.3.1. Reliability

This application can simplify the access of passenger to the taxi service and guarantee a fair management of taxi queues.

3.5.3.2. Availability

The application is an online application. The user, both passenger and driver must have internet connection to use this application.

3.5.3.3. Security

myTaxiService application implements a login authentication to protect the information the user. With username and password each user can protect their data.

3.5.3.4. Maintainability

The developer must know how application works and haow it has been developed.

3.5.3.5. Portability

This application can run on web application or mobile application.

3.5.4. Documentation

During all the process of this project, the following documents should be created to define and record our work:

- **Assignments Document**: to release the tasks and define the requirements that should be satisfied. Also with the deadline of each intermediate products.

- **RASD**: Requirement Analysis and Specification Document, to understand the problem and make emphasis on the requirements that need to be fulfilled to address this issue.
- **DD**: Design Document, to provide the design of the system to be developed
- **User's Manual**: An easy-to-read document to explain how to use myTaxiService
- **Testing report**: A final report of the tests that were performed on myTaxiService

3.6. Alloy Modeling

3.6.1. Alloy Code

```
////// SIGNATURES //////

abstract sig User {
}



sig RegisteredUser extends User {
}



abstract sig TaxiRequest {
        assignTaxi: one TaxiAssigner,
        user: one RegisteredUser
}

sig TaxiImmediate extends TaxiRequest {
}
```

```
sig TaxiReservation extends TaxiRequest{
        date: Time
}


sig Answer {
        code:Int,
        waitingTime: Time,
        cost: Int,
        message: String
}


one sig TaxiAssigner{
        queue : some TaxiQueue
}


sig TaxiQueue {
        taxi : some Taxi,
        zone: one Zone
}


sig Taxi {
}
```

```
sig Zone {}


sig Address {
        street : String,
        City: String,
        Country: String
}


sig Time {
}


////// FACTS //////

fact TaxiQueue_UniqueZone {
        no tq1,tq2:TaxiQueue | tq1 != tq2 && tq1.zone = tq2.zone
}


factoneAssigner_per_queue {
        all tq1:TaxiQueue | one ta: TaxiAssigner | tq1 in ta.queue
}
```

```
facttaxi_belong_to_one_queue {
	all t1:Taxi | one ta: TaxiQueue | t1 in ta.taxi
}


//A registered user can't book two taxis for the same datetime
factunique_time_reservation_per_user {
	no  tr1,tr2 : TaxiReservation | tr1 != tr2 && tr1.user = tr2.user && tr1.date = tr2.date
}

factunique_time_reservation_per_user {
	no  ti1,ti2 : TaxiImmediate | ti1 != ti2 && ti1.user = ti2.user
}

////// ASSERTS //////

assert TaxiQueue_UniqueZone {
	no tq1,tq2:TaxiQueue | tq1 != tq2 && tq1.zone = tq2.zone
}

checkTaxiQueue_UniqueZone for 5


assertoneAssigner_per_queue {
	all tq1:TaxiQueue | one ta: TaxiAssigner | tq1 in ta.queue
}

checkoneAssigner_per_queue for 5
```

```
asserttaxi_belong_to_one_queue {
        all t1:Taxi | one ta: TaxiQueue | t1 in ta.taxi
}

checktaxi_belong_to_one_queue for 5



assertunique_time_reservation_per_user {
        no  tr1,tr2 : TaxiReservation | tr1 != tr2 && tr1.user = tr2.user && tr1.date = tr2.date
}

checkunique_time_reservation_per_user for 5



////// PREDICATES //////

predTaxiImmediate {
        #TaxiQueue> 3
        #TaxiImmediate> 3
        #TaxiReservation = 0
}

runTaxiImmediate for 5



predTaxiReservation {
```

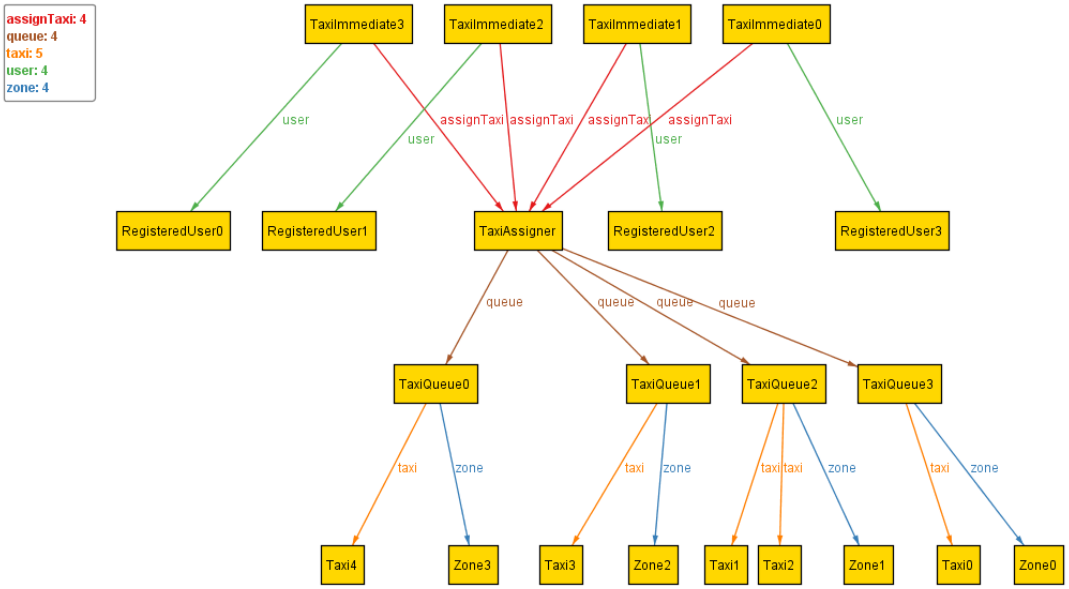
```
        #TaxiReservation> 3
        #TaxiImmediate = 0
}

runTaxiReservation for 5



pred show {
        #TaxiQueue> 3
        #RegisteredUser> 3
        #Time=#TaxiReservation
        #Zone=#TaxiQueue
        #RegisteredUser=#TaxiRequest.user
}

run show for 5
```

3.6.2. Code execution result

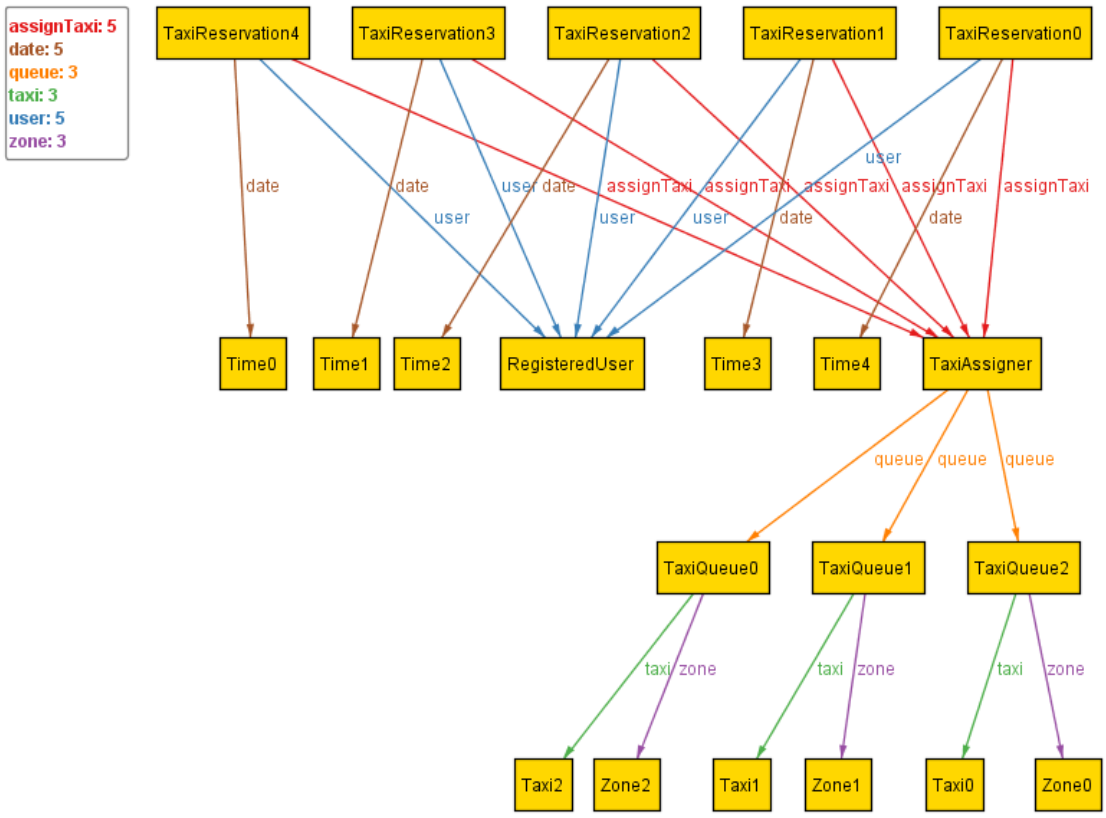
```
7 commands were executed. The results are:
  #1: No counterexample found. TaxiQueue_UniqueZone may be valid.
  #2: No counterexample found. oneAssigner_per_queue may be valid.
  #3: No counterexample found. taxi_belong_to_one_queue may be valid.
  #4: No counterexample found. unique_time_reservation_per_user may be valid.
  #5: Instance found. TaxiImmediate is consistent.
  #6: Instance found. TaxiReservation is consistent.
  #7: Instance found. show is consistent.
```

3.6.3. Worlds Generated

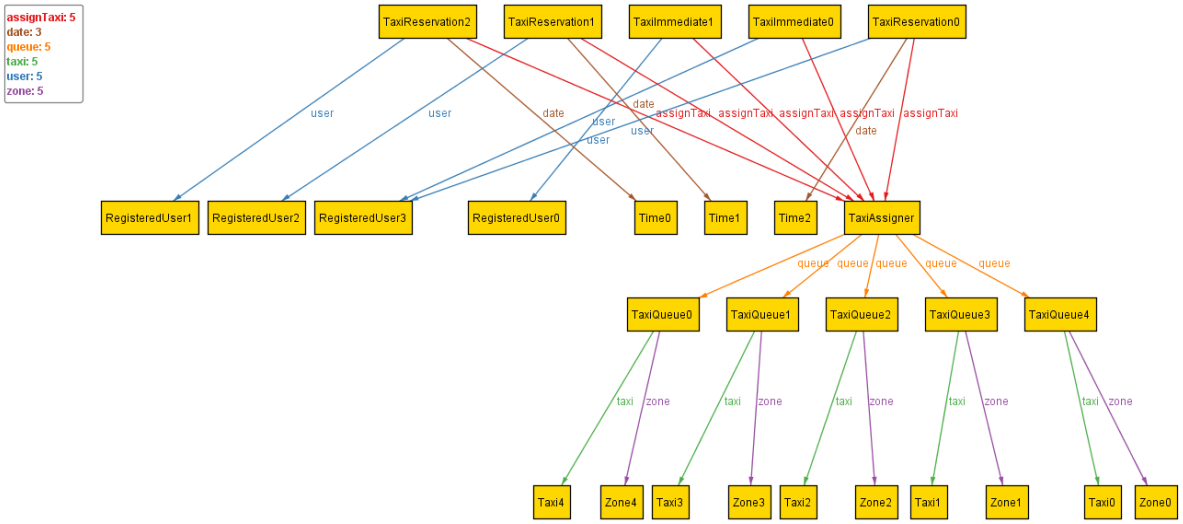
3.6.3.1. Immediate Taxi Request:



3.6.3.2. Reservation Taxi Request



3.6.3.3. Complete world



4. Appendices

4.1. Software and tools used

- Microsoft Word 2010 : To generate this document
- Alloy Analyzer (http://alloy.mit.edu/alloy/) : To demonstrate the consistency of our model.

4.2. Hours of work

- Bakti Ariani Melinda Pertiwi : ~38 hours
- Daniel Naveda : ~38 hours
- Chao Sun : ~38 hours