



POLITECNICO
MILANO 1863

Software Engineering 2: MyTaxiService

Requirements Analysis and Specifications Document

Dimitar Anastasovski, Marco Colombo

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 2 |
| 1.1 | Purpose | 2 |
| 1.1.1 | Intended audience | 2 |
| 1.2 | Scope | 3 |
| 1.3 | Goals | 4 |
| 2 | Overall description | 5 |
| 2.1 | Product perspective | 5 |
| 2.2 | Product functions | 5 |
| 2.3 | User characteristic | 6 |
| 2.4 | Constraints | 6 |
| 2.5 | Assumptions and dependencies | 7 |
| 3 | Requirements | 9 |
| 3.1 | Functional requirements: | 11 |
| 3.2 | Nonfunctional requirements: | 12 |
| 3.2.1 | User interface | 12 |
| 3.2.2 | Software interface | 22 |
| 3.3 | Scenarios | 22 |
| 3.4 | Software system attributes | 23 |
| 3.4.1 | Reliability | 23 |
| 3.4.2 | Availability | 23 |
| 3.4.3 | Security | 23 |
| 3.4.4 | Usability | 23 |
| 3.4.5 | Maintainability | 23 |
| 3.4.6 | Portability | 23 |
| 4 | UML Models | 24 |
| 4.1 | Use Case Models | 24 |
| 4.2 | Class Diagram | 26 |
| 4.3 | Sequence Diagram | 27 |

| | | |
|----------|----------------------|-----------|
| 5 | Alloy | 30 |
| 5.1 | Signatures | 30 |
| 5.2 | Facts | 33 |
| 5.3 | Assertions | 34 |
| 5.4 | Predicates | 34 |

Chapter 1

Introduction

1.1 Purpose

This document is intended to help understand and communication of the requirements of the system, explaining both the application domain and the system that you want to accomplish. It explains the functional features of the MyTaxiService, along with interface details, design constraints and related considerations such as performance characteristics. It can be considered as a contract between the developer and the customer. It is the basis for the planning of the project and to estimate its duration and its cost. It is the base for the activities of testing, verification and validation. The RASD should in fact contain sufficient information to verify if the final system actually satisfies the requirements contained in the document itself. This document follows the IEEE standard for software requirements specification documents.

1.1.1 Intended audience

This document is intended for all individuals participating in and/or supervising the project:

- Developers who can review project's capabilities and more easily understand where their efforts should be targeted to improve or add more features to it (it sets the guidelines for future development).
- Project testers can use this document as a base for their testing strategy as some bugs are easier to find using a requirements document. This way testing becomes more methodically organized.
- End users of this application who wish to read about what this project can do.

The expected audience of this document are the developers and anyone who intends to develop on this program.

1.2 Scope

The main accent is to simplify and optimize the access of passengers to the system and to guarantee fair management of taxi queues. We will build flexible and user-friendly web application and a mobile application that will run on Android and iOS mobile phones. This application can be used by anyone who previously will be register on the registration page. After the registration is done the user will have a user name and password (????) that should remember for furthermore usage of the system. The passenger can call a taxi after a successful logging on the application. After that he can call a taxi and he will be informed about the code of the incoming taxi, waiting time. On the other hand taxi drivers will have a mobile application where the major purpose of them will be to inform the system about their availability, confirmation of a certain call and global map navigation.

1.3 Goals

We think that the system has to provide this features:

1. Allow the registration in the application to the taxi drivers and passengers
2. Simplify the access of passengers to the taxi service: Give a possibility to the users to reserve taxi from mobile application or web and to save time, passengers can call taxi in easier way and this can allow them to save their time.
3. Allow passenger to view if there is any taxi driver free
4. Divide the city into different areas covered by some taxi drivers, organized in "queues"
5. Guarantee a fair management of taxi queues
6. Allow the client to reserve taxi for an exact time

Chapter 2

Overall description

2.1 Product perspective

MyTaxiService is a mobile application (Android and iOS) and web application that require users to have access to a web browser on their smartphone or personal computer. The concept is based on the idea of establishing a direct connection between drivers and passengers to offer both sides a modern alternative to conventional booking processes. After research on the market we will provide unique system for booking a taxi that will benefit customers with cheap fast and satisfying usage of the system. (I don't know if we can write more???)

2.2 Product functions

- **Registration:** a guest user can register as passenger or Taxi driver: the user to be able to properly record must enter Name Last Name Email Phone and credit card and date of birth if he is of the passenger side ends its recording, or if he decides to be a taxi driver he will have to put the information about the taxi : VAT, id taxi number and driver's license. So the system'll confirm the registration by an email.
- **Call Taxi:** after the passenger chooses the mode of travel the system will report the price to the passenger that it will proceed with the payment, and then the resulting call taxi. According to information provided by the passenger completed the system will have to alert the driver to shift (shifts managed by a tail the first taxi driver who makes available will be the first to work FIFO) of the position of the passenger by a text message then the system will put him to instill tail.

- **Reserve taxi:** passengers can book a taxi at least 10 minutes before, at this point after the booking system will alert all passengers of the travel arrangements and will award a taxi from the queue at the time desired by passengers.

2.3 User characteristic

- **Guest:** a person that has not registered and can only exploit basic functionalities such as looking for help.
- **User:** a person that has registered and so has provided his personal information and a set of abilities.
 1. Passenger
 2. Taxi driver
- **Administrator:** the responsible for the web application. The only one that can accept new abilities from users.

2.4 Constraints

Major constraints are related to security and reliability of the system and customers. The Service may permit you to link to other websites, services or resources on the Internet, and other websites, services or resources may contain links to the Site. When you access third party websites, you do so at your own risk. These other websites are not under our control.

- Regulatory policies harmonized with Italian regulatory agency.
- Hardware and software limitations

| | Requirements |
|----------------------|--|
| Operating system | Windows(XP or higher), Linux(Edubuntu,Ubuntu or higher), Mac OS (X 10.6 or higher), Android(Froyo or higher), iOS(7.0 or higher) |
| CPU | 1.5Ghz or higher |
| Memory | 512 MB RAM(for PC) 32 MB RAM (for Android and iOS) |
| Hard drive | 2.2 GB free hard disk space |
| Browser requirements | Internet explorer 6 or higher, Google Chrome, Mozilla Firefox, Apple Safari 5+ |
| GPS | YES |
| Internet Connection | YES |

- Interfaces to other applications:
MyTaxiService doesn't meet with other application
- Parallel operation:
MyTaxiService must support parallel operations
- For testing purposes you will need a simulating tool.

2.5 Assumptions and dependencies

- If you close the application the booking of the taxi continues, push notification will be send before the taxi arrives.
- Administrator(or the system) had authority to delete account
- Taxi driver can cancel the booking
- If customer had more than three cancelation, administrator(or the system) can block his account.
- Customer can book maximum three taxis.
- Customers can cancel booking but the system must inform the taxi driver.
- Special code must send with the booking for easily recognition by the customers.
- MyTaxiService can be only use in Milan.

- If in the concrete region there are no available taxis system can ask the customer if he wants to automatically give the booking to the closest region and taxi with approximate waiting and additional cost.
- If there is no available taxi in the region customer can see queue of booked taxi near the customer with approximate waiting time.

Chapter 3

Requirements

Starting from the goals we explain the following requirements:

1. **Allow the registration in the application to the taxi drivers and passengers:**
 - The system should be able to provide a sign up functionality in order to create a new taxi driver or passenger profiles.
2. **Simplify the access of passengers to the taxi service:**
 - The system has to provide the possibility to the passenger to reserve a taxi
 - The system has to provide both a web application and mobile application in order to allow the passenger to save his time.
3. **Allow passenger to view if there is any taxi driver free:**
 - The system has to provide a functionality that allow a passenger to view if there is any free taxi driver for the desired route.
4. **Divide the city into different areas covered by some taxi drivers, organized in queues:**
 - The system has to provide a functionality to divide the city in some areas
 - The system has to organize the taxi of these areas into queues
5. **Guarantee a fair management of taxi queues:**
 - The system has to provide a functionality in order to manage the taxi queues in the best way has possible.

6. Allow the client to prenote taxi for an exact time:

- The system has to provide a functionality that allow the passenger to reserve a taxi for a desired time.

3.1 Functional requirements:

We specify here functional requirements for each actor:

1. Guest:

- Sign up
- Choose if he wants to register as a passenger or taxi driver

2. Passenger:

- Log in
- Manage his personal informations
- Search for free taxis
- Reserve a taxi
- Call a taxi

3. Taxi driver:

- Log in
- Manage his personal informations
- Share his position on the map
- Change his status (free or busy for a ride)

3.2 Nonfunctional requirements:

3.2.1 User interface

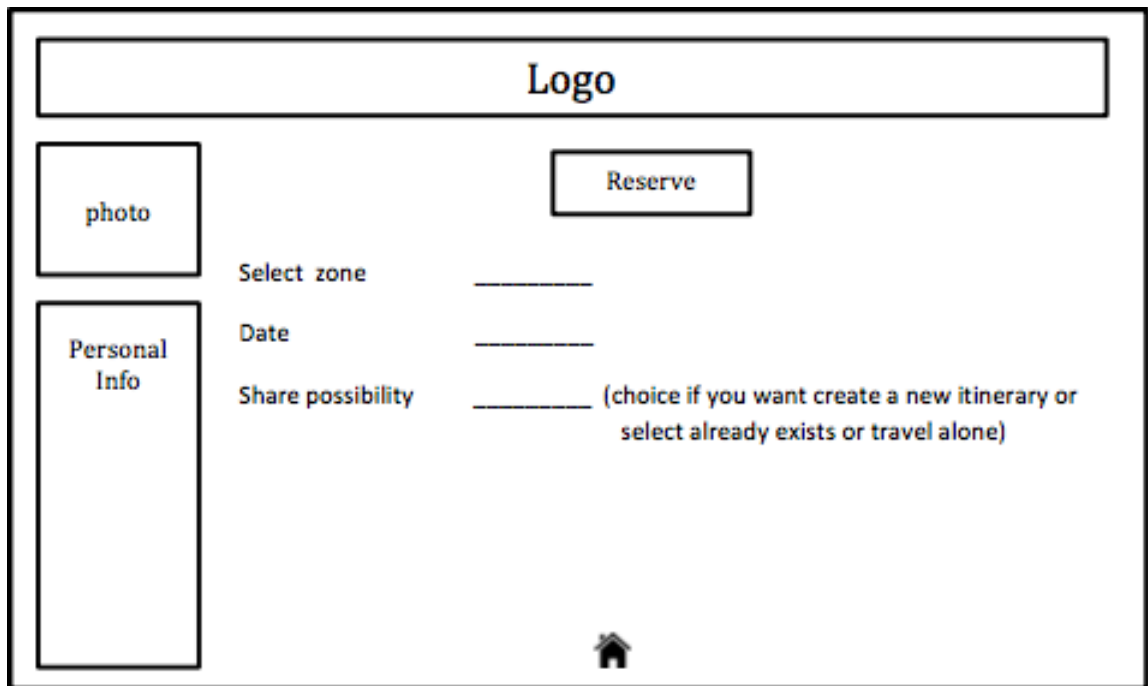
Passenger:

1. Interface to call taxi

First Page App

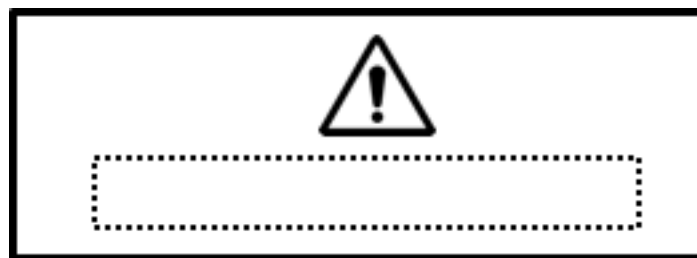


Reserve



The Reserve form layout includes a header bar with a "Logo" placeholder. On the left is a sidebar with a "photo" placeholder and a "Personal Info" section. The main content area contains a "Reserve" button, followed by three form fields: "Select zone", "Date", and "Share possibility". The "Share possibility" field has a detailed instruction: "(choice if you want create a new itinerary or select already exists or travel alone)". A home icon is located at the bottom center of the form.

Date specify the day and time if they don't respect bind of reserve 10 minutes before system show error message



If passenger chooses to travel alone, the system confirms reserve, else if chooses to share travel with new itinerary:

Logo

photo

Personal Info

Create itinerary

Map to choise itinerary

←

🏠

submit

If chooses to share travel with already exist itinerary

Logo

photo

Choose itinerary

Select itinerary(with its price)

Personal Info

submit

First user create the itinerary with all destinations he wants to do, then the second user who share already exist itinerary choose one. In the end system confirm the reserve with an email

Call taxi

Logo


photo

Personal Info

Call taxi

Select destination _____

Share possibility _____ (choice if you want share taxi)



submit

If there isn't shared traveller system advise user, otherwise:

Logo

photo

Choose itinerary

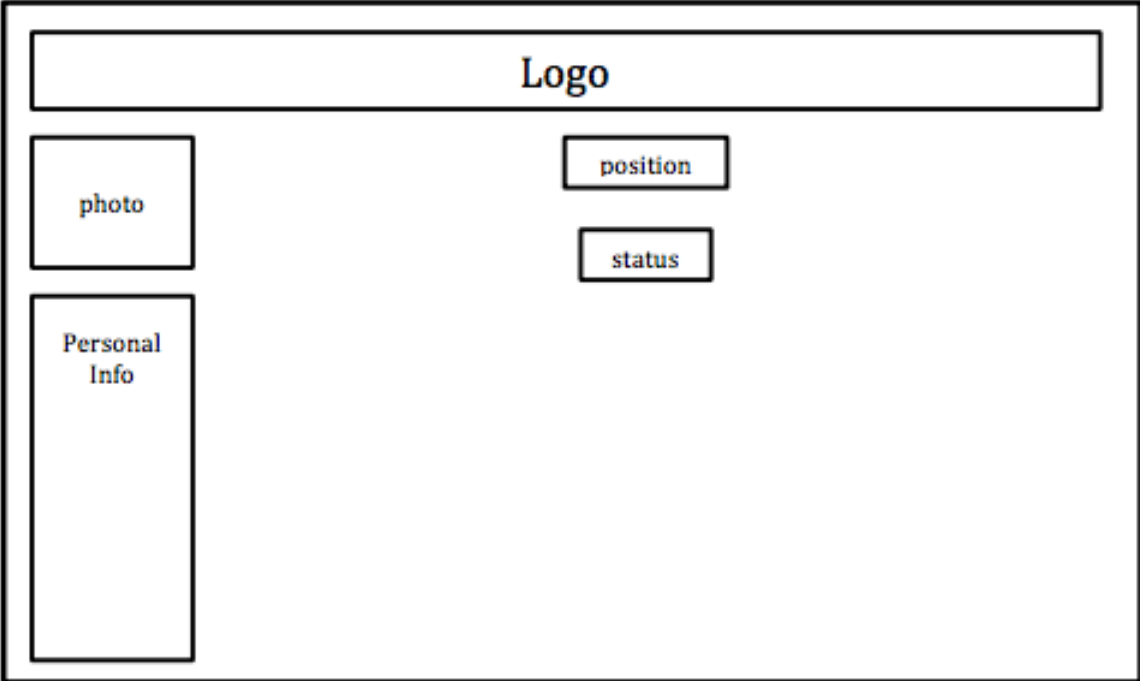
Select itinerary(with its

Personal Info

submit

In each possibility system confirm call request

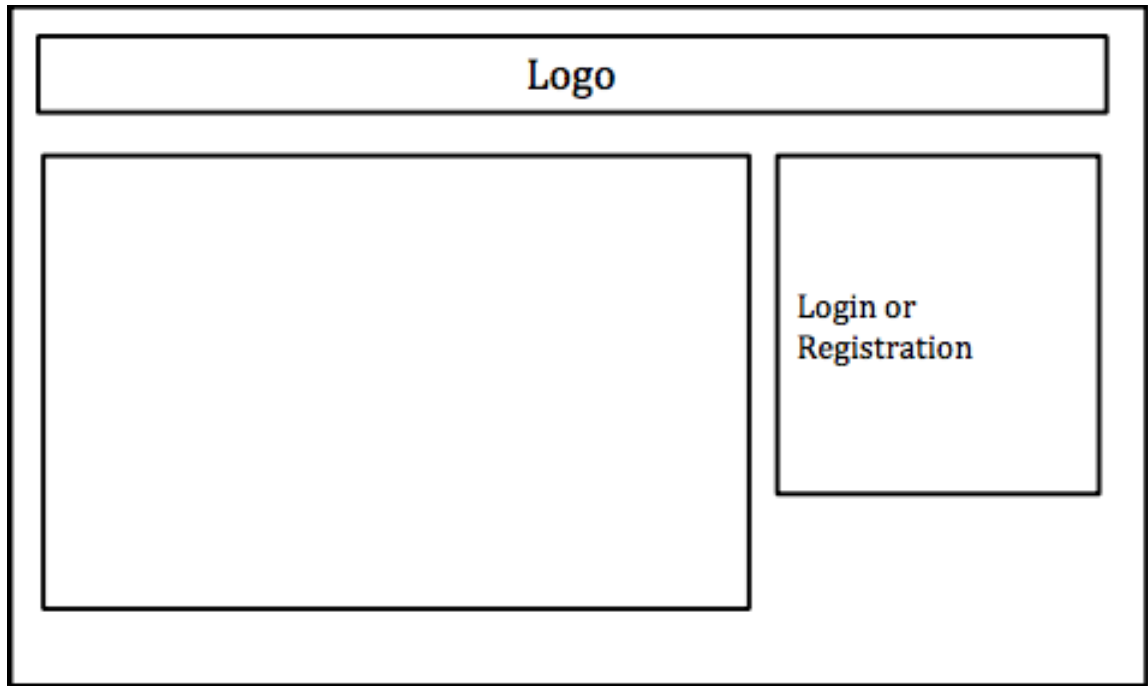
Taxi driver



A wireframe diagram of a form for a taxi driver. The form is enclosed in a large rectangular border. At the top, there is a wide horizontal box labeled "Logo". Below this, on the left side, is a vertical stack of two boxes: the top one is labeled "photo" and the bottom one is labeled "Personal Info". To the right of the "photo" box, there are two smaller boxes stacked vertically, labeled "position" and "status". The rest of the form area is empty.

Taxi drive use the system to confirm if he is free

Home page application



Once time users do the login the system will route or on passenger or taxi page

Registration page

| | |
|---------------------------------------|----------------------|
| Logo | |
| First name | <input type="text"/> |
| Second name | <input type="text"/> |
| Credit card | <input type="text"/> |
| Born date | <input type="text"/> |
| Kind of account | <input type="text"/> |
| Email | <input type="text"/> |
| Telephone | <input type="text"/> |
| <input type="button" value="submit"/> | |

Account can be taxi driver, or passenger. If it's the last one system will confirm the registration with an email, otherwise

Logo

Vat Number

Id license driver

Id taxi

submit

Then system confirm registration with an email

3.2.2 Software interface

Our software will have a database which saves members then serve a data management. In order to verify the data for the taxi drivers, the system must have access to the police database through and verify data on driving license and car. Enabling you to record your credit card the system will use bank pass that allows you to check the validity of information, the software have to use the maps google.

3.3 Scenarios

- **Scenario 1**

Alan has to leave for a trip from Milan to Rome and there is no one on arrival who can get him. Helping with his smartphone, he connects to myTaxiService app and does the log in using his data created in the registration phase. He watch the map and the available taxis in that area and decides to call one of them to get pick up. In few minutes Alan is in the taxi and can get where he wants to lead.

- **Scenario 2**

Alan has to leave for a business trip from Milan to Rome and on arrival he must be at an appointment in less that an hour. Before boarding the plane, Alan makes the access to the myTaxiService application, does the log in with his datas and makes a reservation for a taxi at the arrival time in Rome without activating the sharing option. Now Alan can make the trip with tranquility and will not miss his appointment.

- **Scenario 3**

Mario is a taxi driver and he usually uses the telephone just for receiving calls. He has just discovered from his colleague the exist of MyTaxiService App. He downloads it and registering in the app. During the registration he chooses taxi driver role and in 2 minutes he is ready to use it. He share is position on the map and his free or busy status.

- **Scenario 4**

Mario is a taxi driver and he received a call for a trip, he accessed the app and change his status from free into busy. When he will finish the ride, he will change the status into free and waits for the next call.

3.4 Software system attributes

3.4.1 Reliability

- Application must provide real time driver availability information.
- Application must provide accurate estimate time arrival of the driver.
- Application must provide accurate estimate fare(fare estimate).
- Application must send the booking request to the licensed taxi drivers.(??)

3.4.2 Availability

- Application must serve the customers 24 hours, 7 days in a week.

3.4.3 Security

- The system must use encryption to protect the customers and the drivers.

3.4.4 Usability

- User friendly interfaces (intuitive and easy to use)
- Fast access to content, information (in case of numbers request to the server)
- Presenting data and information?s in clear, logical and readable way.

3.4.5 Maintainability

- The program must be designed in a way that new addition can be done without changing the already developed structure.

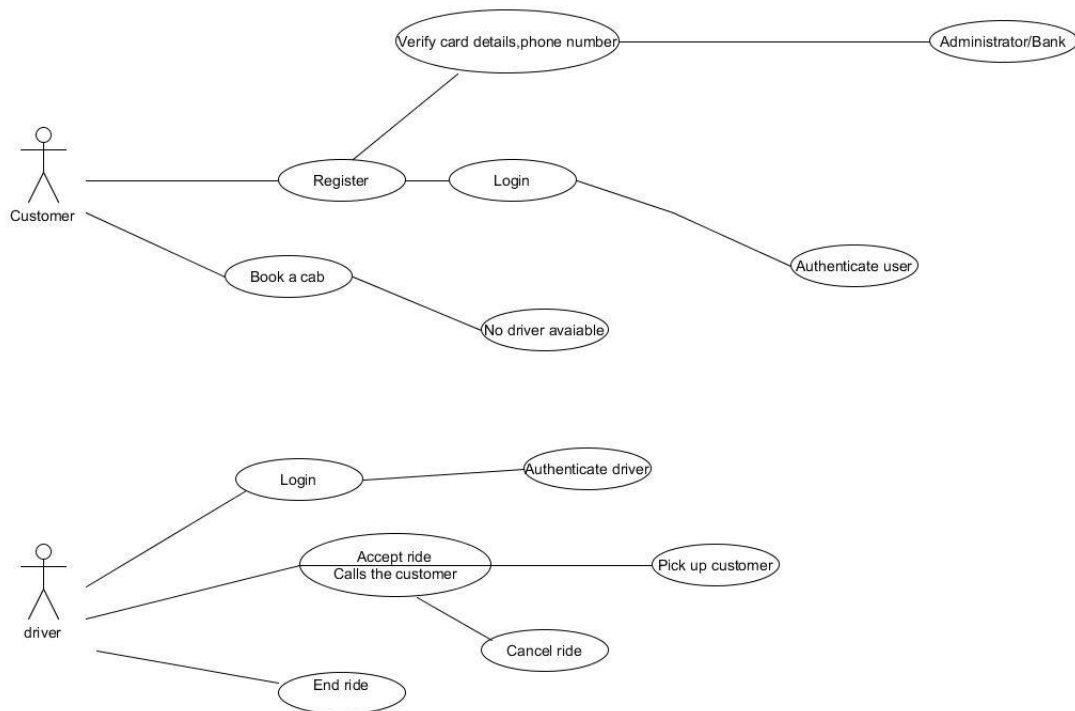
3.4.6 Portability

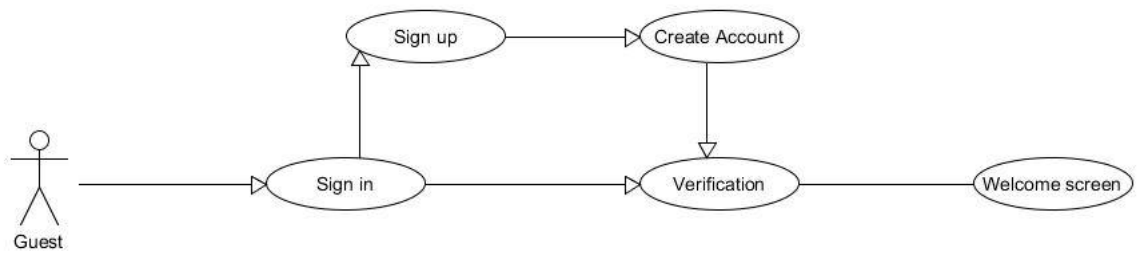
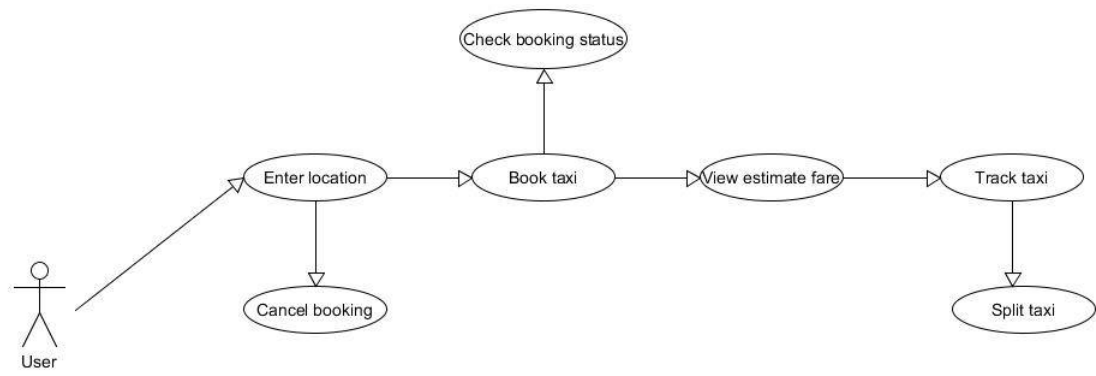
- MyTaxiService is completely portable.

Chapter 4

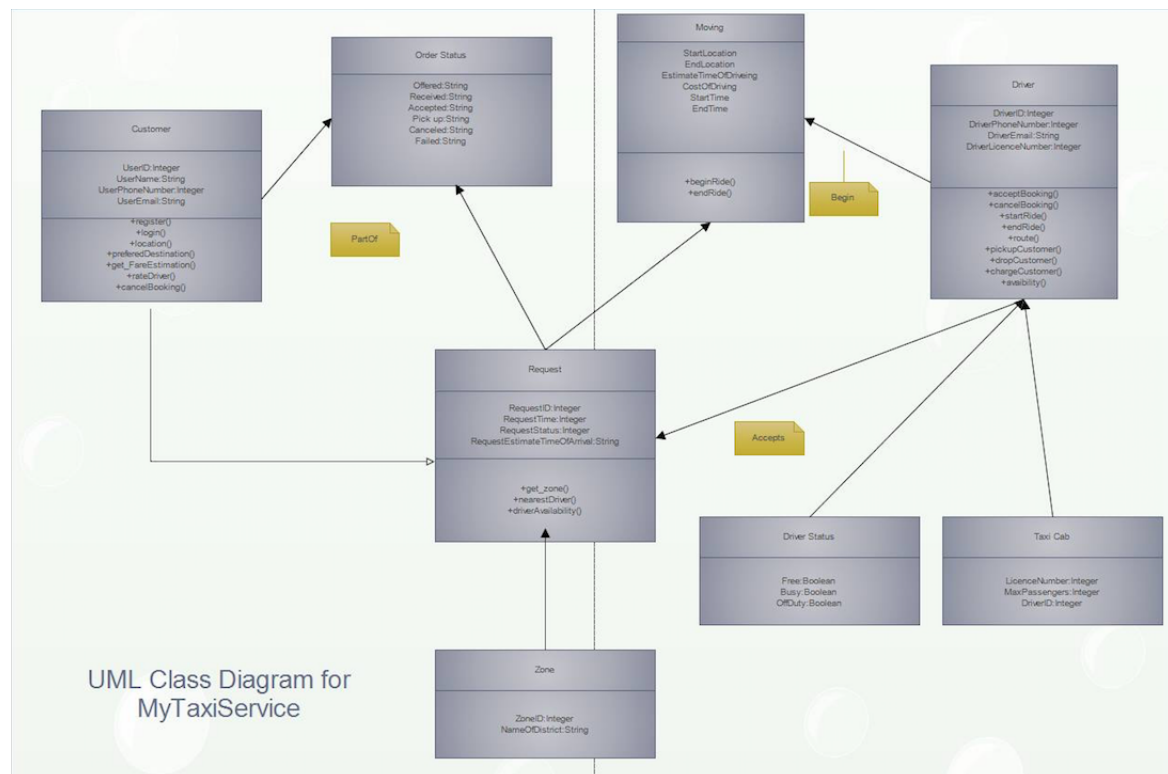
UML Models

4.1 Use Case Models

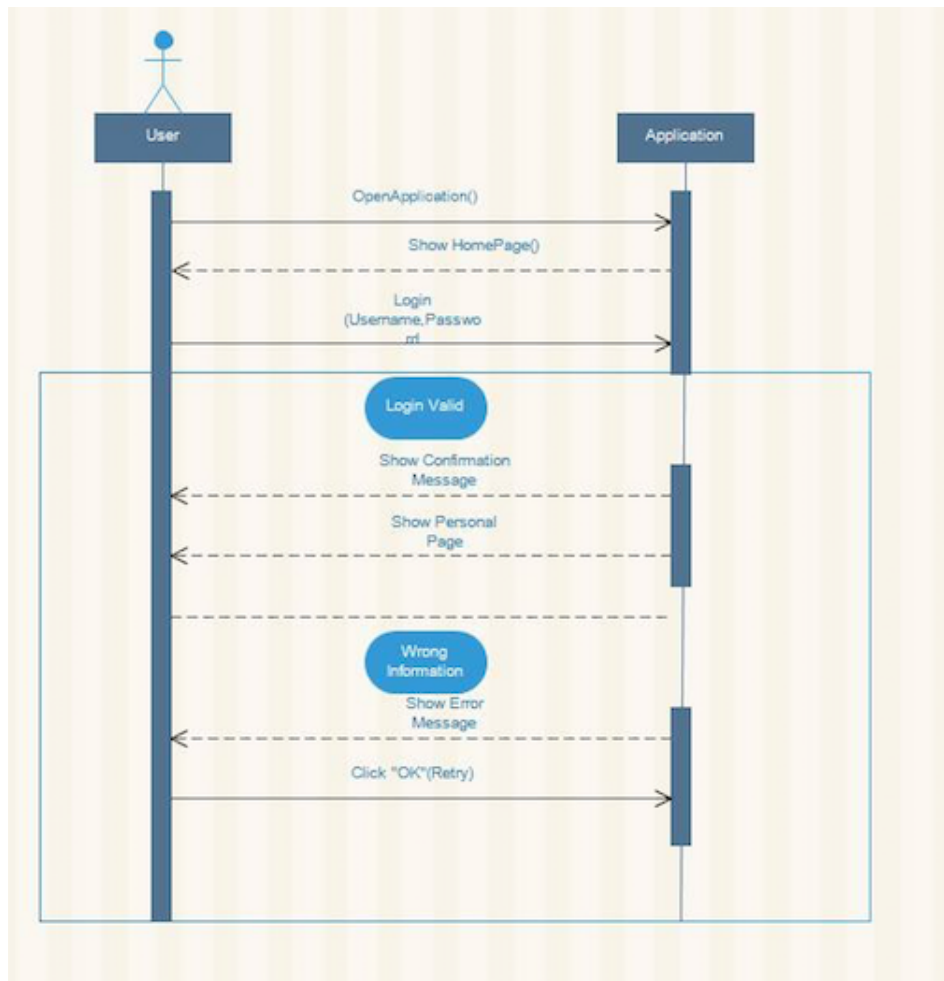


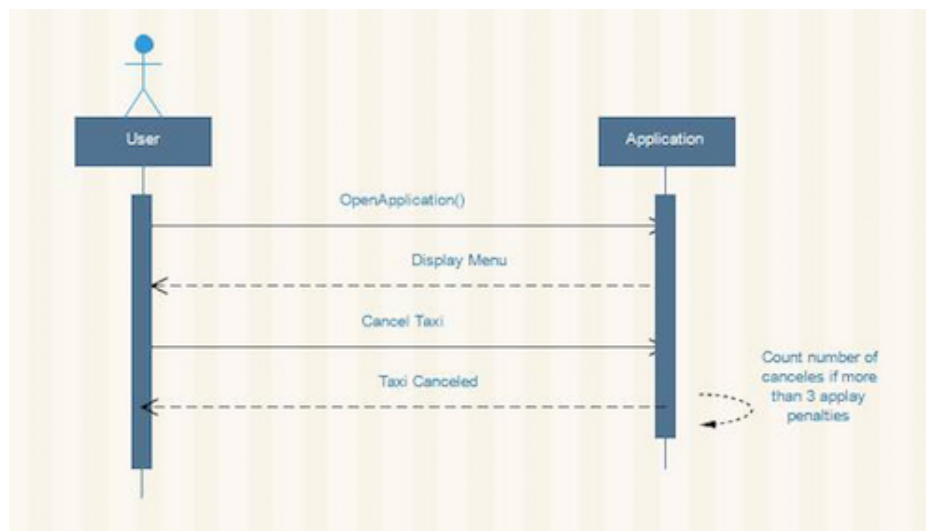
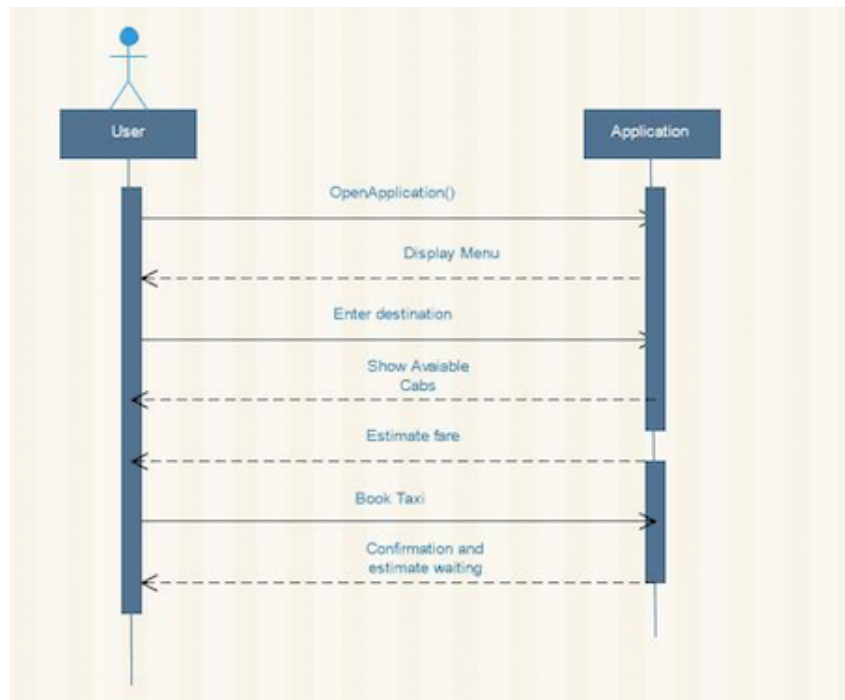


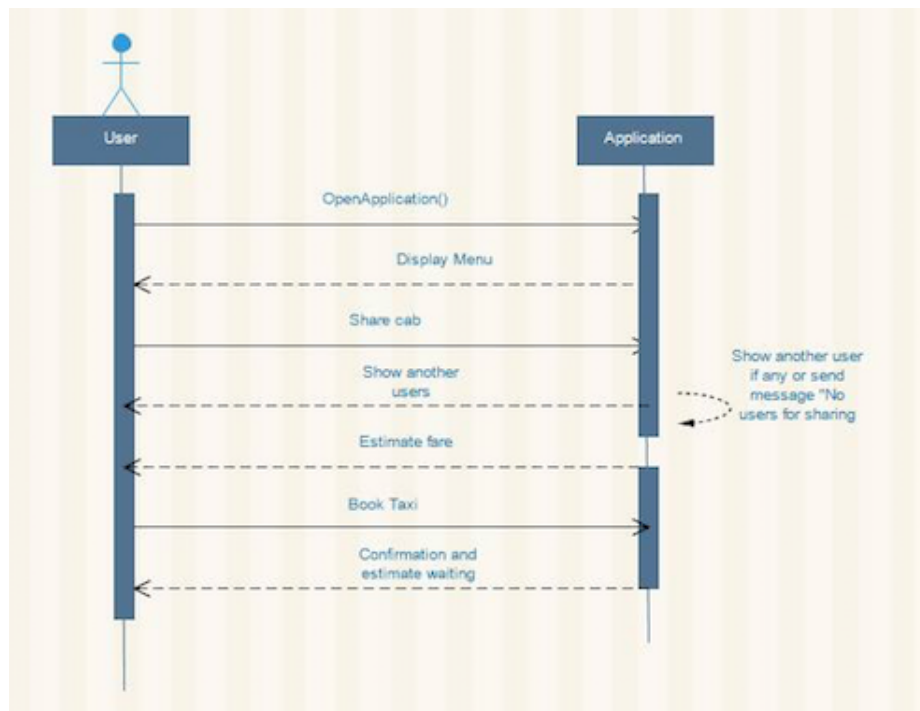
4.2 Class Diagram



4.3 Sequence Diagram







Chapter 5

Alloy

5.1 Signatures

```
sig Customer{
  userID: Int,
  userName: String,
  userPhoneNumber: Int,
  userEmail: String,
  request: one Request,
  orderStatus: one OrderStatus
}
```

```
sig RegisteredUser{
  name: one String,
  surname: one String,
  mail: one String,
  dateOfBirth: one String,
  username: one String,
  password: one String
}
```

```

    sig Driver{
driverID: Int,
driverPhoneNumber: Int,
driverEmail: String,
driverLicenseNumber: Int,
moving: one Moving,
request: one Request
}

sig Request{
requestID:String,
requestTime:String,
requestStatus:String,
requestEstimateTimeOfArrival:String,
moving: one Moving,
orderStatus: one OrderStatus
}

sig OrderStatus{
offered:String,
received: String,
accepted: String,
pickUp: String,
cancelled: String,
failed: String }

sig Moving{
startLocation: String,
endLocation: String,
endNumber: String,
estimateTimeOfDriving: String,
startTime: Int,
endTime: Int
}

```

```
sig Zone{
zoneID:Int,
nameOfDistrict:String,
request: one Request
}
```

```
sig DriverStatus{
Free: String,
Busy: String,
OffDuty: String,
driver: one Driver
}
```

```
sig TaxiCab{
licenseNumber:Int,
maxPassengers:Int,
driverID:Int,
driver: one Driver,
customer: one Customer
}
```

```
sig TimeDate{
day: one Int,
month: one Int,
year: one Int,
hours: one Int,
minutes: one Int,
}
```

```
sig Reservation{
Location: one String,
day: one Int,
month: one Int,
hours: one Int,
minutes: one Int,
}
```

```
sig Cancelbooking extends Driver{}
```

```
sig Requests extends Driver{}
```

sig RequestID extends Request{}

5.2 Facts

```
fact OneUserMaxTakeUps{
all u : Customer — (some t : TaxiCab — t.customer=u)
}
```

```
fact NoEmptyLocation{
all m : Moving — (#m.endLocation=1) and (#m.endNumber=1)
}
```

```
fact noDoubleUser{
no disj u1,u2:RegisteredUser — (u1.mail=u2.mail) and (u1.username=u2.username)
}
```

```
fact TaxiDriverCancel{
all r : Request — (some c: Cancelbooking — c.request=r)
}
```

```
fact NotifyTaxiDriver{ all r : Request — (some c: Cancelbooking — c.request=r)
}
```

```
fact OneCodeUser{
all r : RequestID — (one c : Customer — c.request = r)
}
```

```
fact StartEndDiffrent{
all m:Moving — not ( m.startLocation=m.endLocation )
}
```

```
fact noEmptyTime{
all t:TimeDate — (t.hours=1) and (t.minutes=1)
}
```

```
fact noEmptyDate{
all d: TimeDate — (d.day=1) and (d.month=1) and (d.year=1)
}
```

```
fact MinTwoHours{
all r:Reservation— (r.hours≥2) and(r.minutes≥2)
}
```

5.3 Assertions

```
assert OnlyRegisteredUser{
no r : Request — (no c : Customer — c.request=r)
}
check OnlyRegisteredUser
```

```
assert CancelBookingDriver{
all o : OrderStatus, r1, r2: Request — (o in r1.requestID) and DeleteOrder[o,r1,r2]
implies (o not in r2.requestID)
//all o1 : OrderStatus, o2 : OrderStatus— (o1=o2.cancelled) }
check CancelBookingDriver
```

```
assert NoLessTwoHours{
no r:Reservation—(r.hours≥2)
}
check NoLessTwoHours
```

```
assert AddBookingDriver{
all o : OrderStatus, r1, r2: Request — (o in r1.requestID) and AddBooking[o,r1,r2]
implies (o not in r2.requestID)
//all o1 : OrderStatus, o2 : OrderStatus— (o1=o2.cancelled)
}
check CancelBookingDriver
```

5.4 Predicates

```
pred show(){
#RegisteredUser=1
#Customer=1
#Request=1
#Driver=1
}
run show for 10
```

```
pred DeleteOrder(o:OrderStatus, r1,r2:Request){
  r2.requestID=r1.requestID-o
}
run DeleteOrder for 5

pred AddBooking(o:OrderStatus, r1,r2:Request){
  o not in r1.requestID implies r2.requestID=r1.requestID+o
}
run AddBooking for 5
```