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 . . . 3
- 1.4 Definitions . . . 4
- 1.5 Abbreviations . . . 4
- 1.6 Acronymous . . . 4
- 1.7 Document overview . . . 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: . . . 10
- 3.2 Nonfunctional requirements: . . . 11
  - 3.2.1 User interface . . . 11
  - 3.2.2 Software interface . . . 21
- 3.3 Scenarios . . . 21
- 3.4 Software system attributes . . . 22
  - 3.4.1 Reliability . . . 22
  - 3.4.2 Availability . . . 22
  - 3.4.3 Security . . . 22
  - 3.4.4 Usability . . . 22
  - 3.4.5 Maintainability . . . 22
  - 3.4.6 Portability . . . 22
  - 3.4.7 Future implementation . . . 23

4 UML Models **24**
4.1 Use Case Models 24
4.2 Class Diagram 26
4.3 Sequence Diagrams 27

5 Alloy **30**
5.1 Abstract signatures 30
5.2 Signatures 31
5.3 Facts 33
5.4 Assertions 34
5.5 Predicates 35
5.6 Results 35
5.7 Generated world 36

6 Appendix **37**
6.1 Software and used Tools 37
6.2 Working hours 37

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:

- Expected audience of this document is the developers and anyone who intends to develop on this program
- 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.

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 will be to inform the system about their availability, confirmation of a certain call and global map navigation. City is divided into taxi zones that are uniquely associated with corresponding taxi queues for efficient usage of the system.

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 aplication 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

1.4 Definitions

- Request: Passenger filled form for immediate ride
- Reservation: Passengers can request for a vehicle at least 2 hours before the ride and can reserve his ride
- User: Is a customer who already registered and logged into the system
- Taxi driver: Is a person who legally drives taxi (with driver license and work license) already registered and logged into the system as a driver
- System: Is the system that has to be designed
- Taxi zone: Are the zones in which the city is divided in

1.5 Abbreviations

No abbreviations are been used in this document

1.6 Acronymous

- IEEE Standard 830-1998 Recommender Practice for Software Requirements Specifications
- Specification Document: myTaxiService Project

1.7 Document overview

This document is divided in four parts with clean and non-ambiguity description of the whole system.

- Chapter 1: General description and basic informartion of the system
- Chapter 2: Explanation of the main functionalities with constraints, assumptions and hardware dependencies.
- Chapter 3: Specifications of the system functionalities and non-system functionalities possible scenarios
- Chapter 4: Use cases and UML models
- Chapter 5: Alloy modelling and generated world

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.

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

We expect massive usage from people who use public transport on a daily basis regardless age, gender and nationality. Expected users must have basic knowledge of using web browser and have access to internet.

- **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

Mayor constrains 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 polices 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, Mozila 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
- Customer can only reserve taxi at least two hours before the ride
- Taxi driver can receive ride request while he is unavailable
- Taxi driver that is currently on ride can't receive and ride request
- Only available taxi drivers can receive ride requests
- Taxi driver can cancel the booking
- User can only have one account
- If customer had more than three cancellation, administrator(or the system) can block his account

- If customer wants to share cab his location will be visible public
- If customers wouldn't like to share costs after a booking it will automatically charged full price to every client in the cab
- 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 (including Malpensa and Linate airports)
- If in the concrete region there are no available taxis,the 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 nera 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 taxy 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 reserve 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)

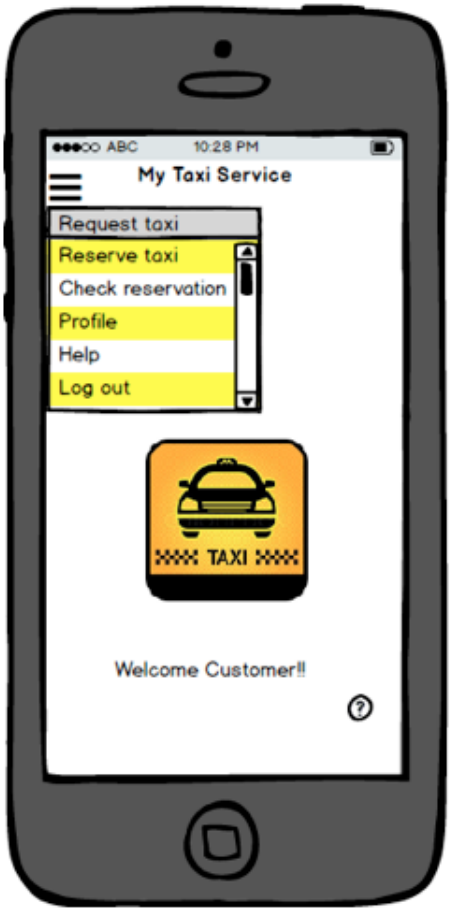
3.2 Nonfunctional requirements:

3.2.1 User interface

Homepage & registration page

Login

Menu

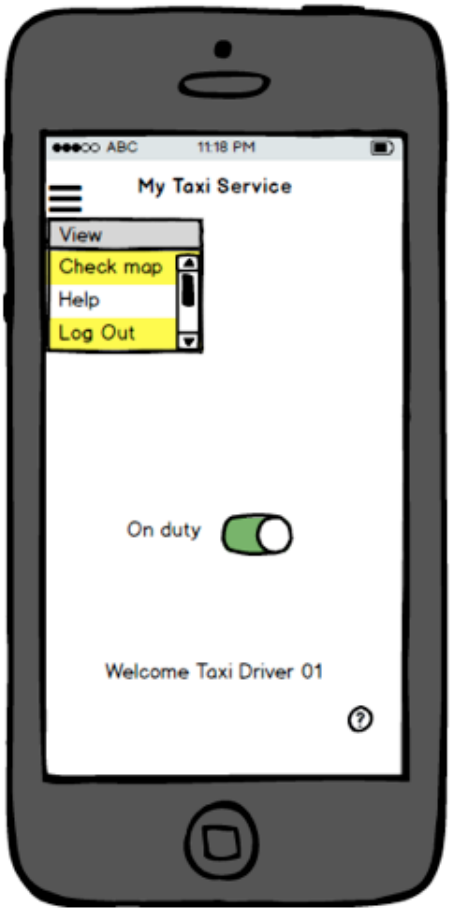

ABC
10:28 PM
My Taxi Service
Request taxi
Reserve taxi
Check reservation
Profile
Help
Log out
TAXI
Welcome Customer!!

Reservation page

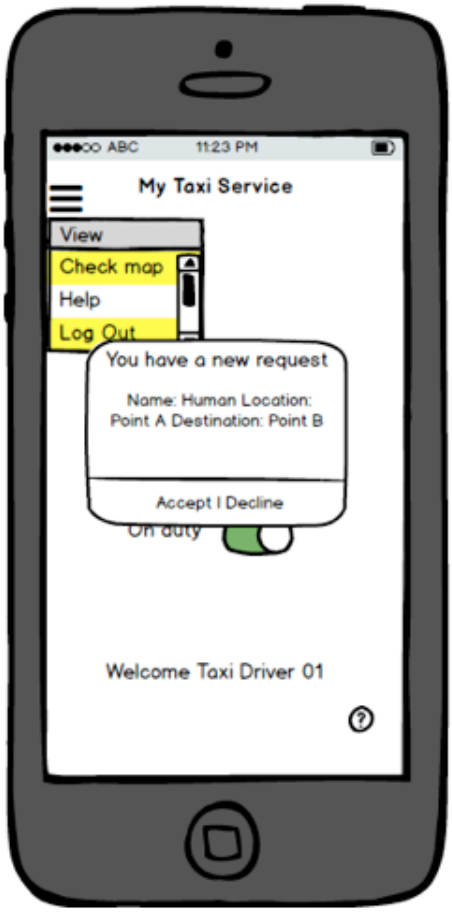
Confirm reservation

View reservation

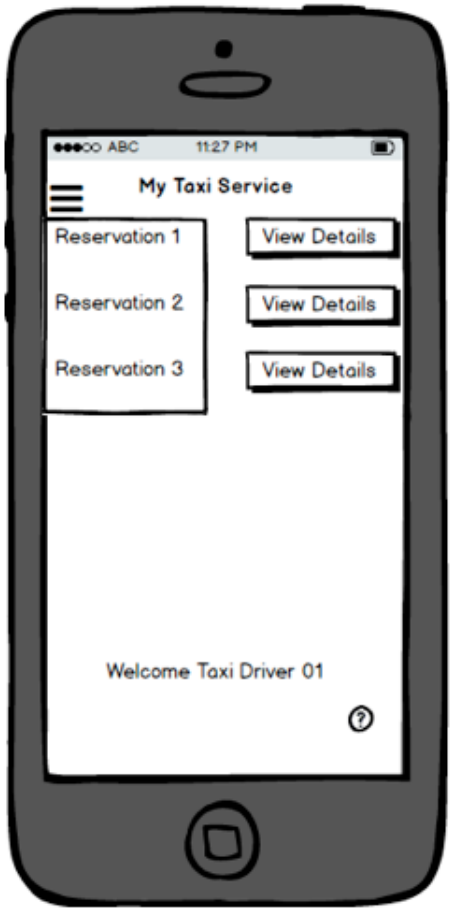
Taxi driver homepage



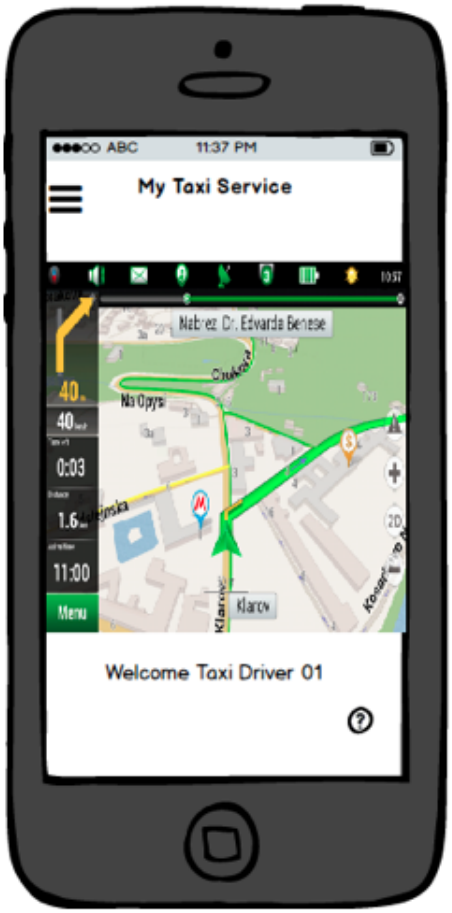
Taxi driver request



Reservation list for taxi driver



Taxi driver map



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 Rome to Milan 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 Rome to Milan 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 Milan. Now Alan can make the trip with tranquility and will not miss his appointment.

- **Scenario 3**
  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.

- **Scenario 4**
  Mario is a taxi driver and he must be at the Linate airport to pick up Alan at the arrival. During the travel the car has a problem and stops to work. Mario immediatly try to repair the car but there isn't anything to do. He sends a notification to Alan and cancel the reservation. Alan will be refunded on his credit card.

- **Scenario 5**
  Luca listen one of his friends talking about MyTaxiDriver App so he decided to download it and use it because he need to make a reservation for the next day. After the download he registers himself into the application and in few

minutes he is ready to go. He logs in and make the reservation. The next day he will find the taxi cab at the desired point.

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.

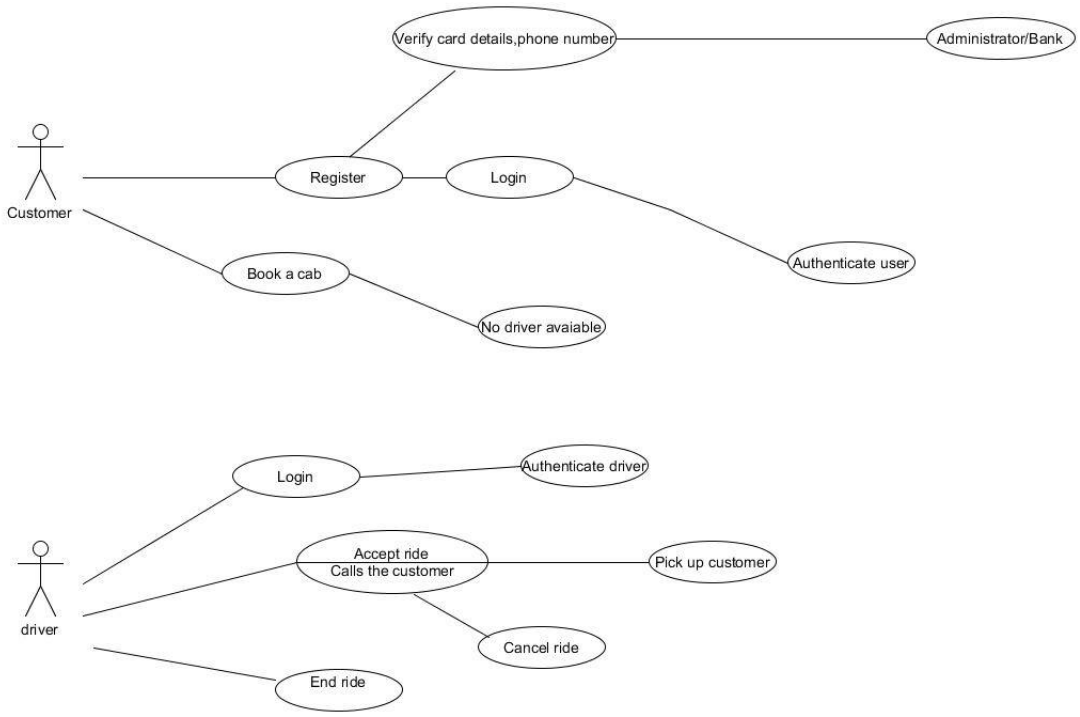
3.4.7 Future implementation

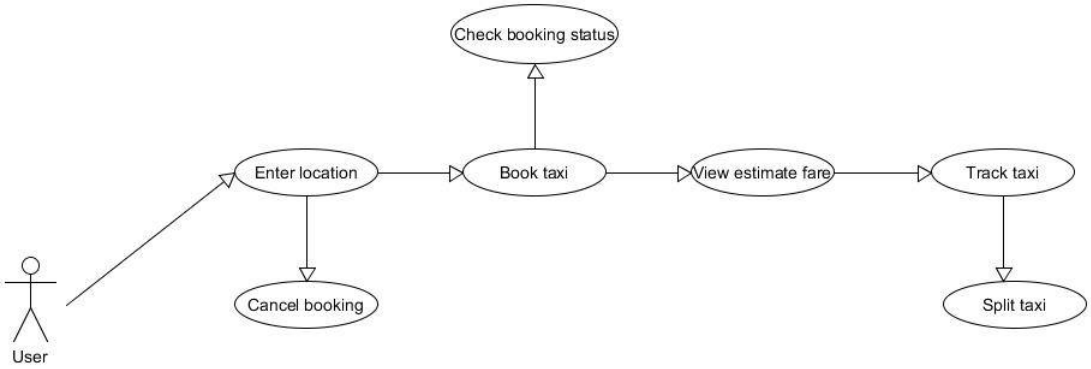
- Customers can use the option taxi sharing
- More available operating zones for the taxis
- Users can pay online
- You can add favorite drivers

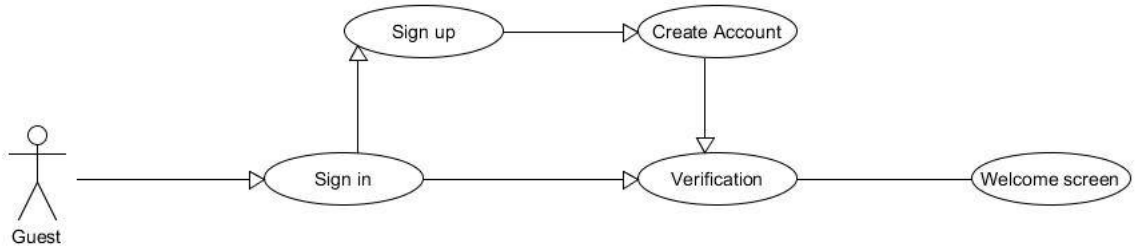
Chapter 4

UML Models

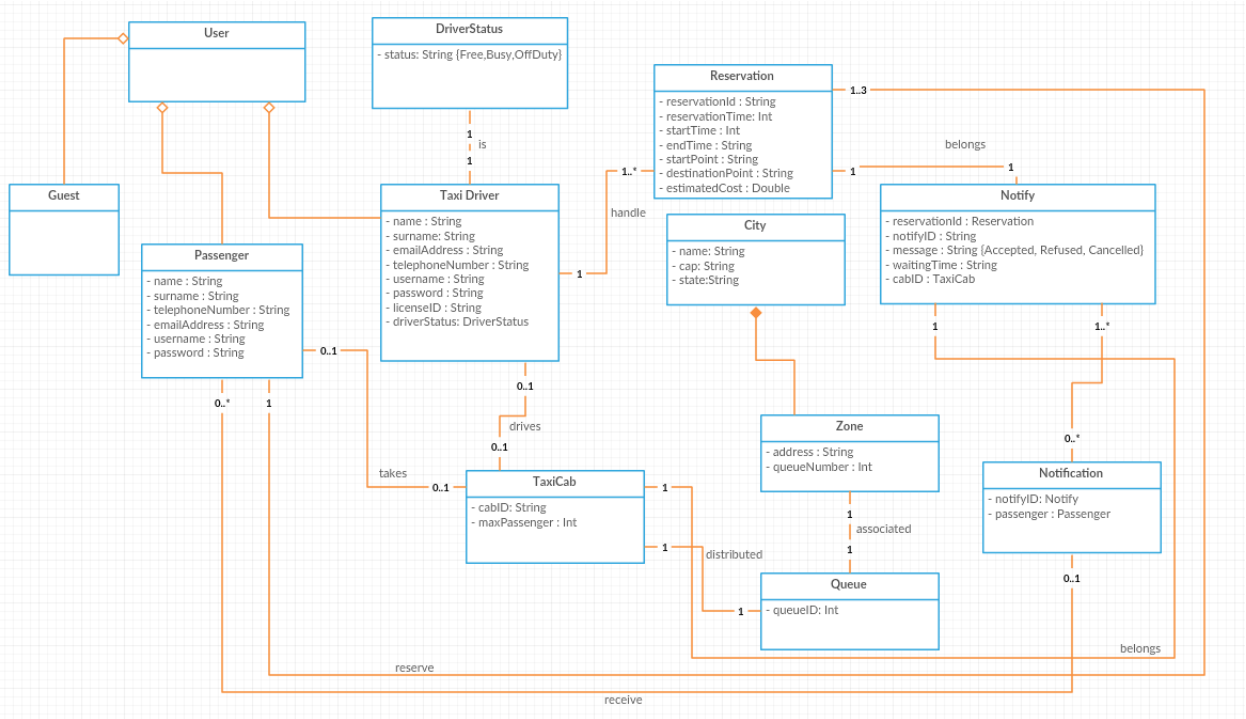
4.1 Use Case Models



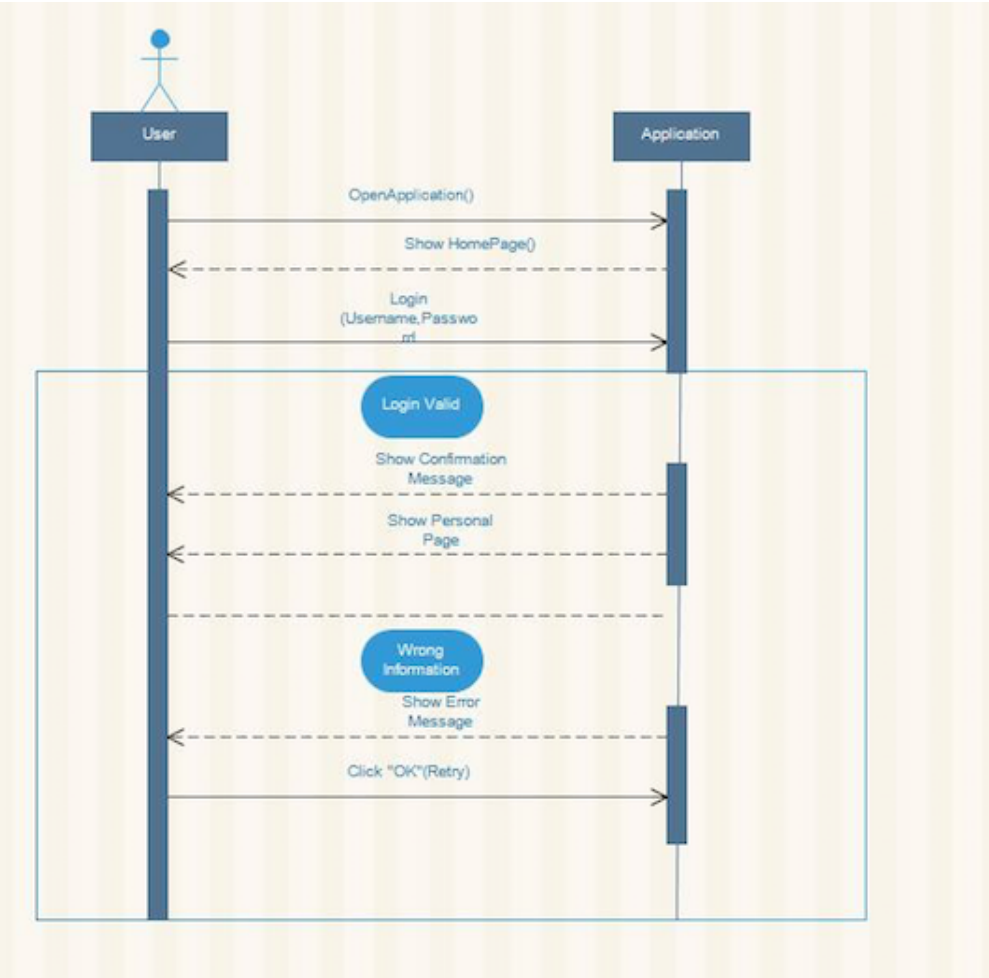

Check booking status
Enter location
Book taxi
View estimate fare
Track taxi
Cancel booking
Split taxi
User

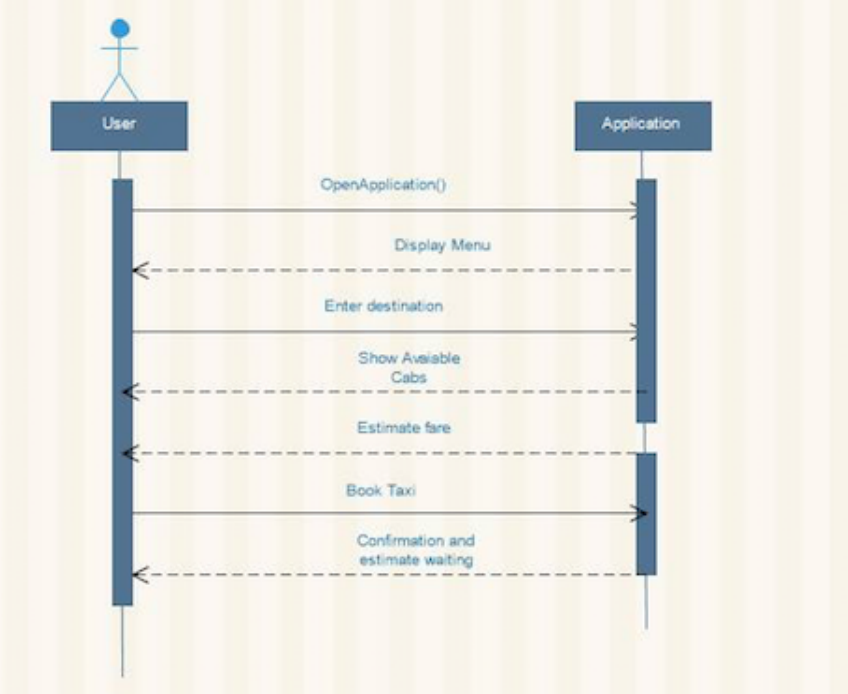

Sign up
Create Account
Sign in
Verification
Welcome screen
Guest

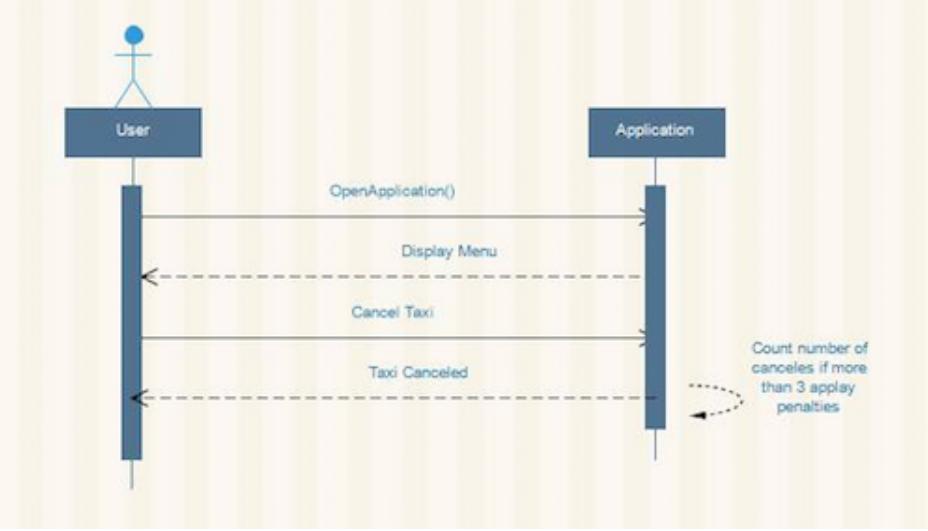
4.2 Class Diagram

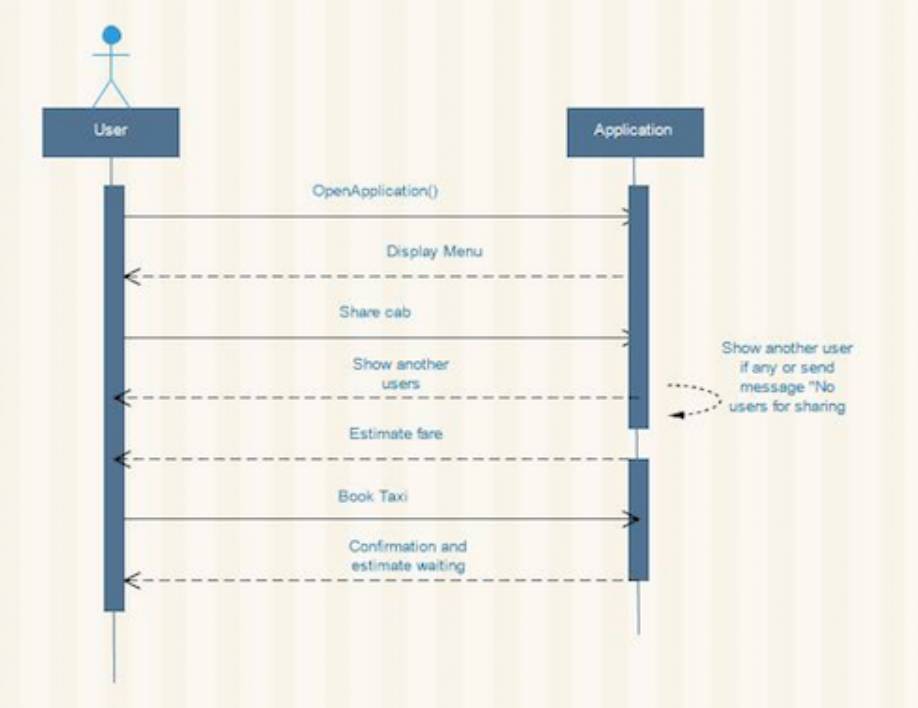


4.3 Sequence Diagrams




User
Application
OpenApplication()
Display Menu
Enter destination
Show Avaiable
Cabs
Estimate fare
Book Taxi
Confirmation and
estimate waiting


User
Application
OpenApplication()
Display Menu
Cancel Taxi
Taxi Canceled
Count number of
canceles if more
than 3 applay
penalties


User
Application
OpenApplication()
Display Menu
Share cab
Show another
users
Show another user
if any or send
message "No
users for sharing
Estimate fare
Book Taxi
Confirmation and
estimate waiting

Chapter 5

Alloy

5.1 Abstract signatures

```
abstract sig User{}

abstract sig Notify{
    resID: Reservation,
    notifyID: String,
    message: String,
    waitingTime: String,
    cabID: TaxiCab,
    reservationID: Reservation,
    notification: Notification
}

abstract sig Reservation{
    reservationID: String,
    reservationTime: Int,
    startTime: Int,
    endTime: String,
    startPoint: String,
    destinationPoint: String,
}

abstract sig DriverStatus{
    status: one String
}
```

5.2 Signatures

```
sig Guest extends User{}

sig Passenger extends User{
    name: one String,
    surname: one String,
    telephoneNumber: one String,
    emailAddress: one String,
    userID: one String,
    password: one String,
    reservationID: Reservation,
    notification: Notification
}

sig TaxiDriver extends User{
    name: one String,
    surname: one String,
    emailAddress: one String,
    telephoneNumber: one String,
    username: one String,
    password: one String,
    licenseID: one String,
    driverStatus: one DriverStatus,
    reservationID: Reservation,
    taxiID: TaxiCab
}

sig TaxiCab{
    cabID: String,
    maxPassenger: Int
}

sig City{
    name: String,
    cap: String,
    state: String,
    zone: Zone
}
```

```
sig Zone{
    address: String,
    queueNumber: Int,
    queueID: Queue
}

sig Queue{
    queueID: Int,
    cabID:TaxiCab
}

sig Notification{
    notifyID: Notify,
    passenger: Passenger
}
```

5.3 Facts

```
fact StartEndDiffrent{
   all r:Reservation| not ( r.startPoint=r.destinationPoint)
}

fact OnePassengerMaxTakeUp{
   no disj p1,p2:Passenger, t:TaxiCab | t.cabID=p1.userID and t.cabID=p2.userID and p1.userID=p2.userID
}


fact noEmptyLocation{
   all r : Reservation | (#r.destinationPoint=1)
}

fact noDoubleIDNotify{
   no disj n1,n2:Notify | n1.notifyID = n2.notifyID
}

fact noDoublePassenger{
   no disj p1,p2:Passenger | (p1.userID=p2.userID)
}

fact noEmptyNotification{
   all n:Notify | (#n.message=1)
}

fact noEmptyTime{
   all r:Reservation | (#r.startTime=1) and (#r.endTime=1)
}
```

5.4 Assertions

```
assert noDoublePassenger{
   no disj p1,p2:Passenger | p1.userID=p2.userID
}

check noDoublePassenger for 5

assert addNewPassenger{
   all p,p1,p2: Passenger | (p not in p1) and addNewPassenger[p, p1,p2] implies (p in p2)
}
check addNewPassenger for 5

assert NoLessTwoHours{
   no  r:Reservation|(r.startTime-r.reservationTime>2)
}
check  NoLessTwoHours

assert OnlyRegisteredUser{
   no r : Reservation, p:Passenger | (p.userID=r.reservationID)
}
check OnlyRegisteredUser
```

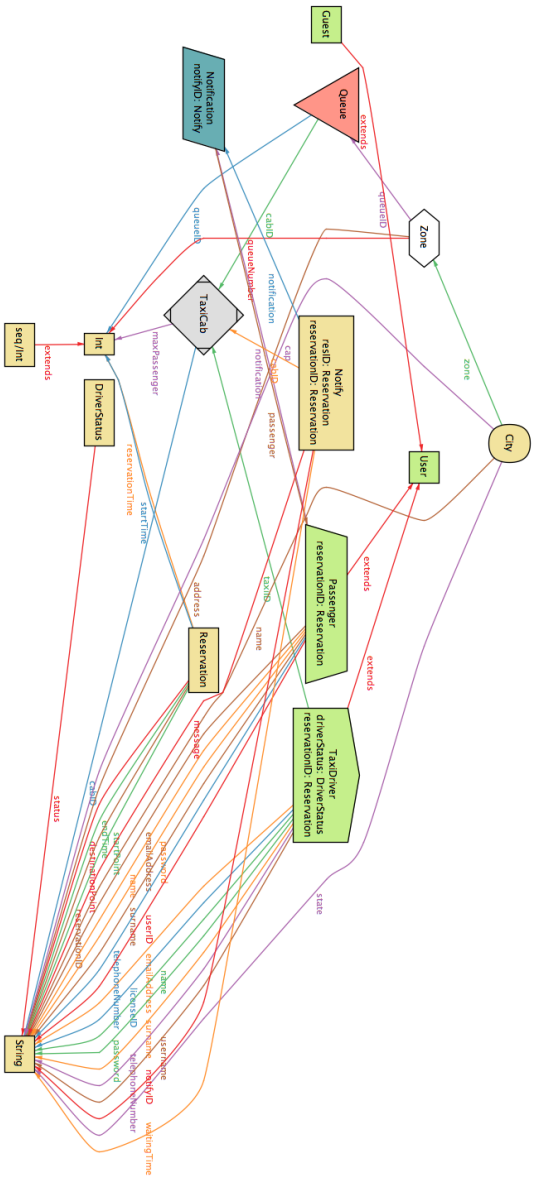
5.5 Predicates

```
pred show(){
   #User > 1
}
run show for 30

pred addNewPassenger(p,p1,p2:Passenger){
   p.userID not in p1.userID implies p.userID=p2.userID
}
run addNewPassenger for 10

pred noLessTwoHours(r:Reservation){
   r.startTime-r.reservationTime>2
}
run noLessTwoHours for 10
```

5.6 Results

```
7 commands were executed. The results are:
   #1: No counterexample found. noDoublePassenger may be valid.
   #2: No counterexample found. addNewPassenger may be valid.
   #3: No counterexample found. NoLessTwoHours may be valid.
   #4: No counterexample found. OnlyRegisteredUser may be valid.
   #5: Instance found. show is consistent.
   #6: No instance found. addNewPassenger may be inconsistent.
   #7: No instance found. noLessTwoHours may be inconsistent.
```

5.7 Generated world



Chapter 6

Appendix

6.1 Software and used Tools

- TexShop (http://pages.uoregon.edu/koch/texshop/): to redact this document
- Alloy analyzer 4.2 (http://alloy.mit.edu/alloy/): to prove the consistency of the model
- CreatelyApp (https://creately.com/app/): to create Class diagram

6.2 Working hours

Dimitar Anastasovski: 40 hours

Marco Colombo: 30 hours