Requirement Analysis and Specification Document

Beretta Carolina 852650
Brizzolari Cecilia 852399

January 5, 2016

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to explain in detail what the project "myTaxiService" is about, specifying all requirements and specifications useful for the development of the software and letting the stakeholders have an idea of how much work will be needed for the actual implementation of the application.

1.2 Scope

The main goal of this project is to optimize the taxi service of a city. This application will allow customers to request a taxi in the easiest way possible, passing through a system which will keep track of drivers' availability. The system will choose the taxi for the customer and notify both parties, without them needing to have direct contact. We want this application to be accessible from both smartphones and computer browsers. It will also need two different interfaces, one for each type of user: customer and taxi driver.

The specific goals of this application, thus, are

- [G1] allow a potential user to register
- [G2] allow an user to log in
- [G3] allow an user to manage his profile
- [G4] allow a passenger to call for a taxi
- [G5] allow a passenger to reserve a taxi in advance
- [G6] allow a taxi driver to accept or decline a call
- [G7] allow a taxi driver to change his status

1.3 Stakeholders

This document is meant for the founders of the project, the programmers who will implement the algorithms and functionalities and the architects of the system. From this RASD they should be able to gauge the complexity of the problem, in terms of time and work needed to bring it forth. The main stakeholder is the company which requested the development of this application: they requested it to be user-friendly, and most of all fast and efficient so that it gains customers quickly.

1.4 Definitions, acronyms and abbreviations

In order to avoid any ambiguity in the description and specification of the project, a list of definitions and acronyms is provided.

Guest: generic unregistered person

User: generic registered person, who is logged into the system

Passenger: user registered as a passenger, who will be able to book a taxi through both the mobile application and the website

Driver: user registered as a taxi driver, who will be able to use the system functions only through the mobile application

License's number: the license that allows a person to work as a taxi driver

Driver license: the official document or card which shows that a person has the legal right to drive a vehicle

Call: immediate booking, the request must be fulfilled as soon as possible

Reservation: advanced booking

System: the software system to be developed comprehensive of its all modules and function, it is the provider of the actual services the user can access through mobile applications and website

Zone: part of the city in which the customer and the taxi is situated. The city is defined as the disjoint union of all the zones

Waiting Queue: a list of available taxis in a specific zone, ordered from the one who has been available the longest to the one one who has just finished carrying a customer around

MTS: acronym for myTaxiService

ETA: estimated time of arrival

1.5 References Documents

- Specification document: Assignments 1 and 2 (RASD and DD)
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications

1.6 Overview

This document is divided in to four parts:

- **First section** briefly describes the main idea of this project, with no technical information
- **Second section** shows, still on an abstract level, what will be needed in the development of the software: the constraints of the system, the assumptions about the "world" it will interact with, along with the type of users we are expecting to see it work for
- **Third section** illustrates the project specific requirements, going into deeper detail, with diagrams used to better understand the entities used by each functionality and how they will transmit information between each other
- **Appendix** will include the Alloy models, based on the class diagram, and generated worlds

2 Overall Description

2.1 Product perspective

The application will have two main interfaces, one for the passenger and the other for the drivers. The look and functionalities of the two will be different, but they will have to inter-operate with each other as well as the system. The passengers' interface must be accessible from both browser and mobile app, while the drivers' target platform is mobile only.

2.2 Software Interfaces

The website will be accessible from any browser. In order to reach the maximum number of potential costumers, the mobile application must be available for Android, iOS and Windows Phone.

2.3 Hardware Interfaces

In order to fulfill its main functionalities, the system acquires the user's location whenever possible. The mobile app must therefore be able to acquire the GPS coordinates of the user's phone, both driver and passenger.

2.4 Product functions

The product will have two sets of functions, one for the drivers' side and one for the customers' side.

The application for the drivers must allow them to

- update their status (available/offline)
- accept or decline ride requests from the system

while the application for passengers must allow them to

- call a taxi
- make a reservation

and for both, there will need to be the functions

- register
- log in
- modify their account
- delete account

2.5 User characteristics

There are two types of intended users for this app: *passengers* and *taxi drivers*.

The passenger will be of all age and educational level, no restriction whatsoever. The drivers on the other hand, will need to have a valid license, their own vehicle, and will have to respect all requirements needed with their profession, as stated from the taxi rules of the city.

2.6 Constraints

2.6.1 Regulatory policies

The system must ask the permission to acquire and store personal data and web cookies. Moreover the rules of taxi service of the city must be respected.

2.6.2 Hardware limitations

The limitations the system must face are correlated to the specific interface. The mobile application must be able to run on devices with 1GB of RAM and three inches displays. Moreover it should not require more than 150MB of free space and should have a reasonable loading time with a 3G connection. On the other hand the website should support at least 1024×768 screen resolution and be able to load pages in a reasonable amount of time given a 5Mb/s connection.

2.7 Assumptions and dependencies

The system will work under these assumptions

1. The city is divided cleanly into zones of 2 km^2
2. All vehicles are provided with a GPS
3. All drivers have a cellphone with MTS app installed on it
4. The passenger will always be at the meeting point on time
5. No passenger is trying to pull a prank on the drivers making them waste their time
6. All drivers will always notify the system when they pick up a passenger and when they end the trip, by selecting the corresponding functionalities offered by the system
7. All drivers will be able to arrive at the meeting point in 10 minutes, if they are in the same zone of the pick-up point
8. All drivers will be able to arrive at the meeting point in 15 minutes, if they are in a neighboring zone of the pick-up point's zone
9. If a driver is running late due to traffic conditions or other problems, he/she will notify the passenger with a phone call

10. The city has an archive of all taxi driver li
11. If there's no taxi available, the system will choose it from a neighboring zone
12. If no taxi is available even in the neighboring zone, the system will tell the customer to try again later
13. If it is a holiday or if there's a strike, the app will tell the customer promptly that there's no taxi available for the day

2.8 Future Implementations

The MTS system could be expanded with future implementations of the following functionalities:

1. **Online Payment**: the system could provide the possibility of online payment. The Passenger should have the possibility to associated a credit card or a PayPal account to his/her profile and should be able to update his/her preferred payment method at any moment. At the end of the ride the mobile application should provide the passenger the "Pay by App" option. If this functionality is selected a screen with the payment request should be shown. The passenger should then verify the payment details and after a successful verification, the system should send the recipit to the user's email account. Since this function would require to store sensitive data, the security and privacy policies should be upgraded accordingly.
2. **Rating**: Passenger would be able to write a review and rate the driver after a ride. When a request is accepted the system should include the rating into the taxi's info sent to the passenger.
3. **Fidelity Points**: passengers would be able to acquire points for each ride booked through the MTS system and would be able to spend the accumulated points for discounts

3 Specific Requirements

3.1 Interface Requirements

The user interface has to be as simple and intuitive as possible. Any user should be able to easily understand the basic functions without any previous knowledge of the system.

The web application must be accessible from all most used browser such as Chrome, Firefox, Safari and Opera. It should also be compatible with the latest version of HTML and CSS, and most common screen resolutions. The mobile app must be able to run on smartphone with a screen of at least three inches without any problem.

In both cases every screen should provide a link to the main page, where the user is able to easily access all the functionalities of the system.

3.2 Scenarios

3.2.1 First scenario

Tsukishima is training along with his teammates, and the exercise is to run uphill until you reach the finish line. Unluckily, Tsukishima really isn't feeling it today, and doesn't want to build up a sweat running. He could walk all the way, but it would still take too much effort, or he could try and find a different solution. He takes out his smartphone and searches for a way to call a taxi. He stumbles upon the MTS site and decides to give it a try. He downloads the app and proceeds to fill up the passenger's registration form, inserting his name, surname, email, telephone and the username "DinoNerd". Sadly that username is not available and the system notifies Tsukishima that he must select another one. He takes a few minutes to think and inserts a new one. Now the user "MoonySaur11" will be able to skip running exercises any time he wants.

3.2.2 Second Scenario

Lately business hasn't been good for Igor. He decided to give up on his limousine service and try the standard taxi market by using the MTS app. He downloads the phone app and as soon as he opens it, he selects the driver registration form. He inserts all the needed data, including the license's number, number-plate of his new taxi and his driving license and completes the process by agreeing to the Terms of Use and privacy policy. The system validates his license and sends him a confirmation email. Igor checks his mailbox and activates his account, ready to go pick up new Persona protagonists any time they need it.

3.2.3 Third Scenario

Paolo has a date a fateful Friday afternoon, and wants to get the tube from Cadorna to Loreto. But Paolo decided to have a date exactly the day in which ATM had scheduled a strike. Seeing the awful situation that is happening inside

the station, he remembers about the MTS app he has on his phone and decides to use it. He logs in and select the call taxi option. Few seconds later he receives confirmation along with an estimated waiting time and the taxi code and decides to go wait outside, even though it's raining. Few minutes later he receives a call from the taxi driver, who tells him that, due to traffic conditions, she is running a bit late. Minutes passes but there is no sign of the taxi. Paolo decides to check the MTS app and notices that the estimated time is growing. The customer waits another fifteen minutes and is getting quite angry, when the driver finally shows up, with sunglasses and Starbucks, to pick him up. Paolo arrives to his destination late and his date thought that they had been stood up, so they left. Paolo is fuming when he opens the MTS app to complain about the service offered by taxi number 852399. However the system doesn't support this function yet, so Paolo decides to delete his account, erasing all traces of the "GiovanePaoloPunto" user from MTS data base.

3.2.4 Fourth Scenario

John has just finished a very important mission and is quite fatigued. Since his faithful buddy can't take him back home, he decides to call a taxi through the MTS app. He logs in, selects the "call Taxi" function, confirms his GPS position and successfully completes the request. The system forwards the ride request to "ADAM", the first available taxi in the waiting queue of the zone in which John is located. However "ADAM" is currently having a discussion about hamburgers with his friend and doesn't reply within 30 seconds, triggering the timeout. The system marks the request as declined, updates the driver position in the queue and proceeds to select a new taxi. Few seconds later the driver "Pequod" receives a ride request from the passenger "BB" and promptly accepts it. The system notifies John that "Pequod" will be arriving shortly at the pick-up point.

3.2.5 Fifth scenario

Jennifer Wilson needs to go to London to meet one of his lovers. Since she's leaving later in the day and doesn't want to risk not having a taxi ready, she decides to reserve a taxi with the MTS app. She logs in, chooses the right functionality and inserts the origin and destination of the trip and the time she wants to leave. After she confirms the reservation, she goes to finish preparing her pink suitcase. After a couple of hours a cabbie receives notification from the system that a certain "PinkLadyRach" needs a ride to London. He opens the app, accepts the job and gets on the road. The cabbie picks up the lady, but instead of taking her to her destination he brings her to a secluded place, and offers to play a game with her. Poor PinkLady couldn't know that her driver was going to be a murderer that kills his victims with mind games and poisonous pills. The cabbie wins the game, killing the lady, and thus he no longer is servicing a customer. He takes out his smartphone and selects the option "end the ride", ready to go meet another unsuspecting victim.

3.2.6 Sixth scenario

Kaname isn't feeling very well but he decides to go to work anyway. He gets on his taxi, takes out his smartphone, logs in the taxi app and changes his status from offline to available. Few minutes later the system notifies him that the passenger "Zero" needs a ride. He reluctantly accepts the booking and goes to the pick-up point. As soon as the passenger is on board he notifies the system by selecting the start ride option on his phone. Few minutes later the taxi arrives at the destination and the driver ends the ride. While Kaname is waiting for another customer, he decides to check his favorite blogging site. As he is scrolling through his dashboard he notices a post saying that his beloved fanfiction has finally been updated. Poor Kaname had been waiting this moment for months and he can't wait any longer. He takes out his phone, opens the app, updates his status to offline, and starts reading right away, drowning in feels given to him by his OTP.

3.2.7 Seventh scenario

Seijuro Akashi suffers from a double personality disorder. It has been a couple years since he's registered on the MTS application with the username "AbsoluteEmperorEye", but due to recent shocking events, i.e. losing a basketball match, he suddenly switched personalities. Now he feels that the username he used in all his applications and social networks does not represent him, so he decides to change them all. Being rich and powerful means that he is able to blackmail his former teammates to do the boring work for him. After an incredible amount of time one of the poor victims, Midorima, finally reaches the MTS app. He logs in, chooses to go into the profile and presses the button to modify it. The system answers by giving him an already compiled form with the data stored inside the database. He recompiles it and confirms that he wants to save the changes, and then waits for the server's response. As soon as the OK arrives, he passes the phone to Akashi who finally puts it down, happy that he had been able to change all his usernames in "JoinTheZone4" without troubles.

3.3 Functional Requirements

3.3.1 Use Case Diagram

Figure 1 illustrates the main functions offered by the system and the main actors involved.

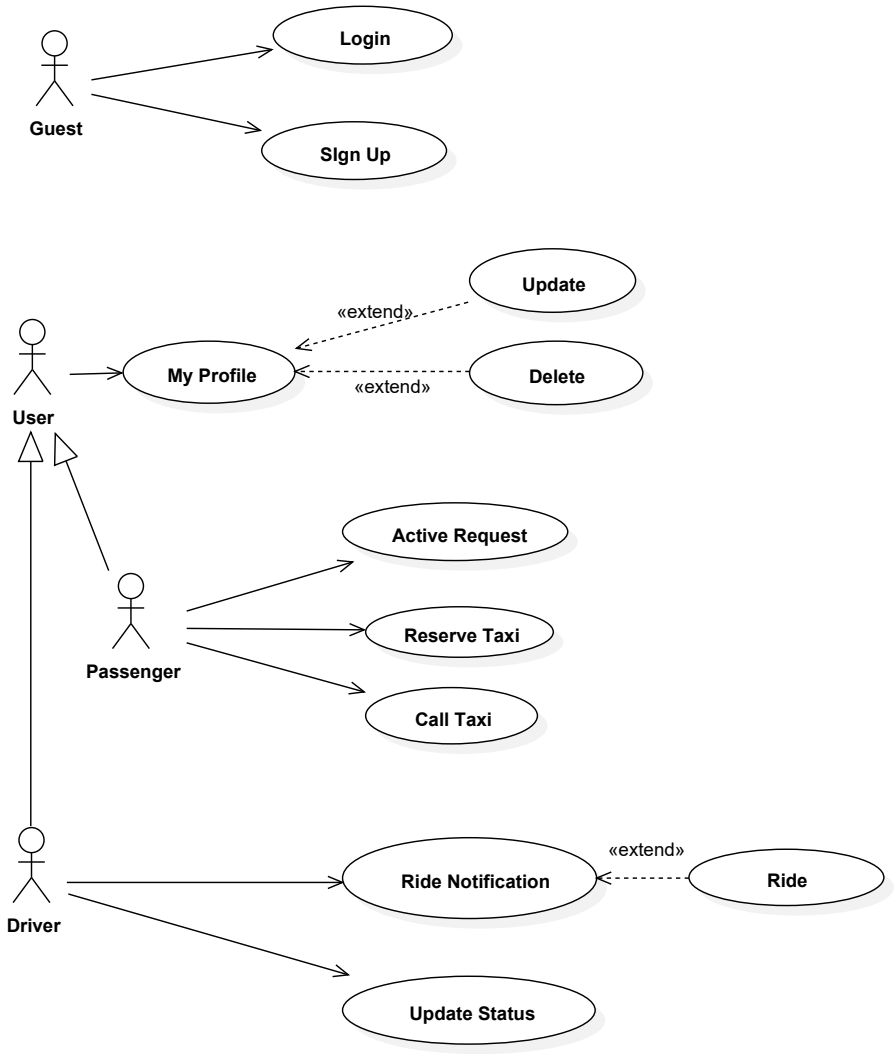

Figure 1: MyTaxiService Use Case

3.3.2 Sign Up

Actor	Guest
Goal	G1
Entry Condition	The guest doesn't have an account
Exit Condition	The guest successfully ends the registration process
Exceptions	One or more mandatory fields not filled, the guest is already registered, the guest didn't agree to Term of Condition and privacy policy

Any guest can sign up through the website or the mobile application. The first step of the registration process is to select the form the guest is interested in: driver or passenger. Both forms have mandatory fields(email, password, username, name, surname, telephone). The driver's form has additional fields, where the guest must specify the license's number, the driver license and the taxi number-plate. The potential user's account enters a pending status after the registration form is correctly submitted. For passengers' accounts the system immediately sends a confirmation email to check the availability of that email address.

On the other hand, drivers' accounts can't be activated until the authenticity of the driver's license is verified. Once the license is validate, by checking in the city's license archive, the activation link is sent to the email provided in the registration form. If the validation process has a negative result an email indicating that the provided license is not valid is sent. After the driver confirms his/her account the system assigns an ID to the driver.

Related requirements

1. The system must guarantee that there aren't two different users with the same credentials
2. The system must guarantee that there aren't two different users with the same email
3. The system must guarantee that there aren't two users with the same username
4. The systems must verify that the email address is formally correct
5. The system must verify that the telephone number is formally correct
6. For the driver's form, the system must verify that the license is formally correct
7. The system only accepts passwords that are a combination of 8-32 alphanumeric characters

8. If there is any invalid mandatory field, the system must reject the form, displaying an error
9. If the guest doesn't agree to Term of Use and privacy policy, the registration process can't be completed
10. The system must validate the taxi driver license within 48h
11. The system must allow registration only to authorized taxi drivers
12. The activation link expires after 72h
13. If the activation link is expired or the driver's license is not valid, the system must erase all guest's data
14. The system must guarantee that there aren't two different drivers with the same ID

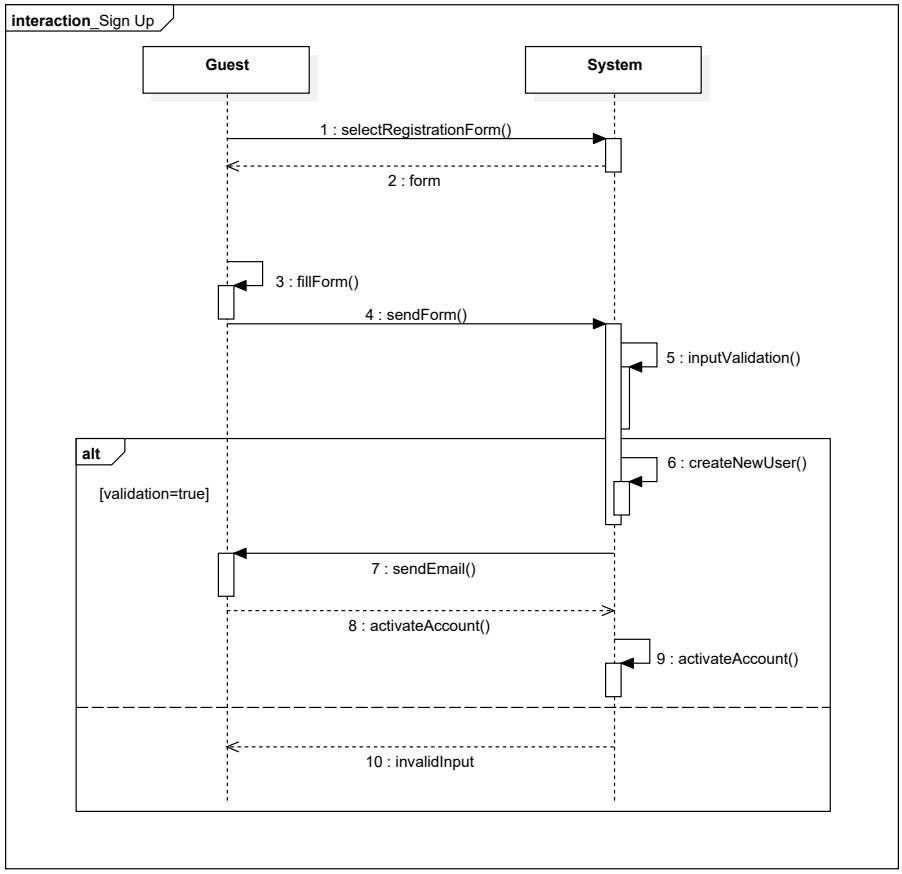

Figure 2: Sequence diagram of the passenger's registration process

3.3.3 Log In

Actor	Guest
Goal	G2
Entry Condition	The guest has an account and isn't already logged in the system
Exit Condition	The guest inserted the right credential or the retrieve password function
Exceptions	The credentials are wrong

Any registered user can log into the system with his/her username or email and password. Passengers can access the system's functionalities from both the mobile app and the website, whereas drivers must use the mobile application. If an user forgets his/her password the system offers the possibility to retrieve it, by sending a temporary password to the user's email account.

Related Requirements

1. The system must grant access only to users who correctly input the username or email and associated password
2. If the retrieve password option is selected, the system must send an email with the temporary password to the user's email account
3. The temporary password expires after one hour
4. After the user has logged in with the temporary password, the system must ask to set a new one

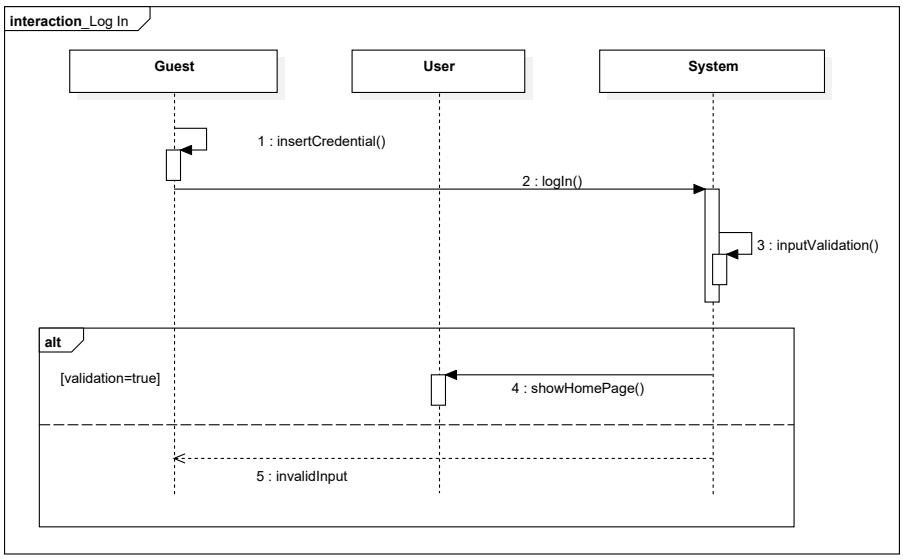

Figure 3: Sequence diagram of the log in process

3.3.4 My Profile

Actor	User
Goal	G3
Entry Condition	The user is logged in the system
Exit Condition	The user's data is updated or the profile has been successfully erased
Exceptions	Email or password not valid, the new username is already used

Users are able to update and delete their profile. They can change their email, username, telephone and password.

Related Requirements

1. Before deleting an account, the system must ask the user to confirm the operation
2. Before accepting a new email, the system must guarantee that it is formally correct
3. When an user wants to change his/her password, the system must ask for both the old and the new one
4. The system only accepts passwords that differ at least one character form the previous one

5. If a user deletes his/her account, the system must erase all his/her data from the system database

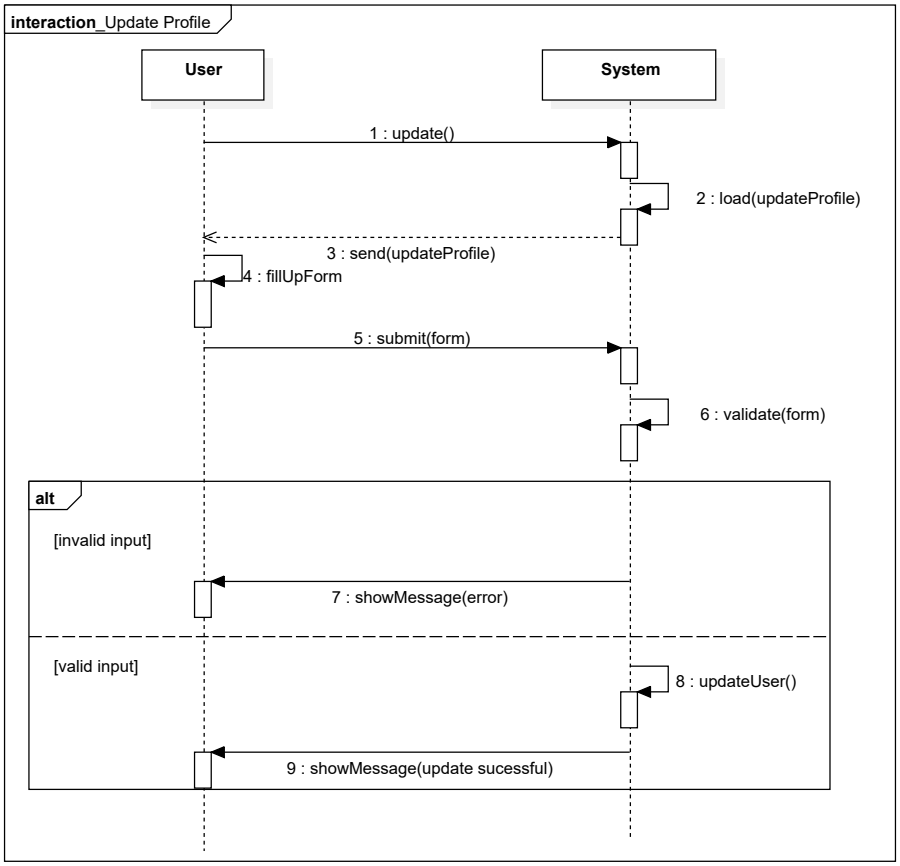

Figure 4: Sequence diagram of update profile function

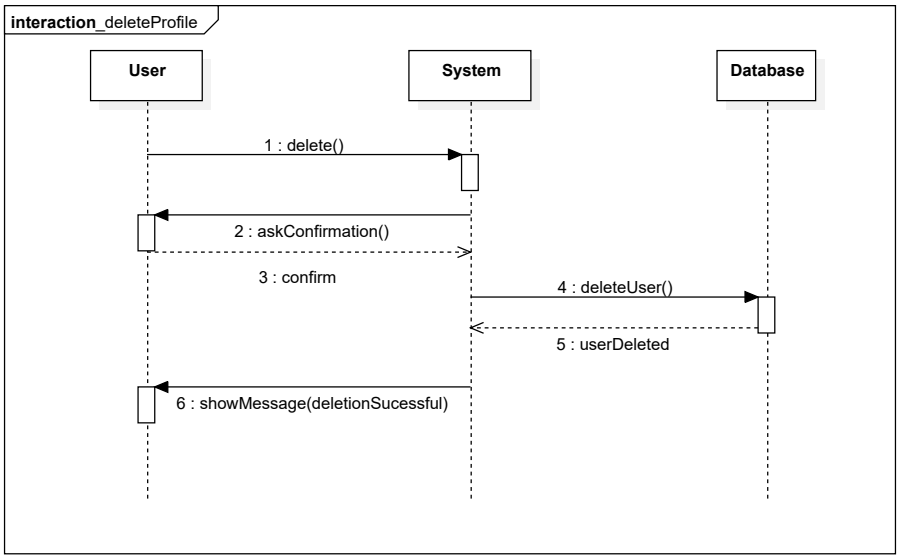

Figure 5: Sequence diagram of delete profile function

3.3.5 Call Taxi and Active Request

Actor	Passenger, Driver
Goal	G4
Entry Condition	The passenger is logged in the system
Exit Condition	The driver notifies the system that the passenger is aboard
Exceptions	The pick-up point is outside the city's zone, there's no taxi available

Passengers can book a taxi through the website or the mobile app. They have to specify the pick-up point, or confirm their GPS position, and the number of passengers. The system forwards the request to the first available taxi in the waiting queue of the pick-up point's zone. If no driver accepts the ride request within a given time, the system notifies the passenger that no taxi is available at the moment. Once the booking is confirmed the systems sends the passenger the taxi's info i.e. the driver's ID and the ETA. The passenger can check the ETA, which is periodically updated by the system, at any moment through the active request function. When the driver arrives at the pick up point he/she can select the "start ride" option in order to notify the system that the passenger is aboard. From this moment onward the passenger will no longer be able to see the ETA. The driver takes the passenger to destination and at the end of the ride he/she will select the "end ride" option to notify the system that he is now available for another booking.

Related Requirements

1. The system automatically locates the passenger via GPS, if possible
2. If the passenger's GPS coordinates are available they are automatically selected as the initial pick-up point
3. The passenger can confirm his/hers location, if any is available, or change the pick-up address manually
4. The system only accepts pick-up points inside the city's zones
5. After the passenger's confirmation, the system delivers the request to the first taxi in the queue of the pick-up zone
6. Requests are processed with a FIFO policy
7. If the system doesn't find a taxi within five minutes, it must notify the user that there is no taxi available
8. After the driver accepts the request, the system must forward the passenger a notification with the taxi's info(ID, ETA)
9. The ETA is computed by the system considering the current position of the taxi and traffic conditions
10. The ETA is updated every two minutes
11. When the driver finishes the ride his/her status is automatically set to "available"
12. When the driver finishes the ride inside the city he/she is automatically inserted in the waiting queue of his/her last position's zone

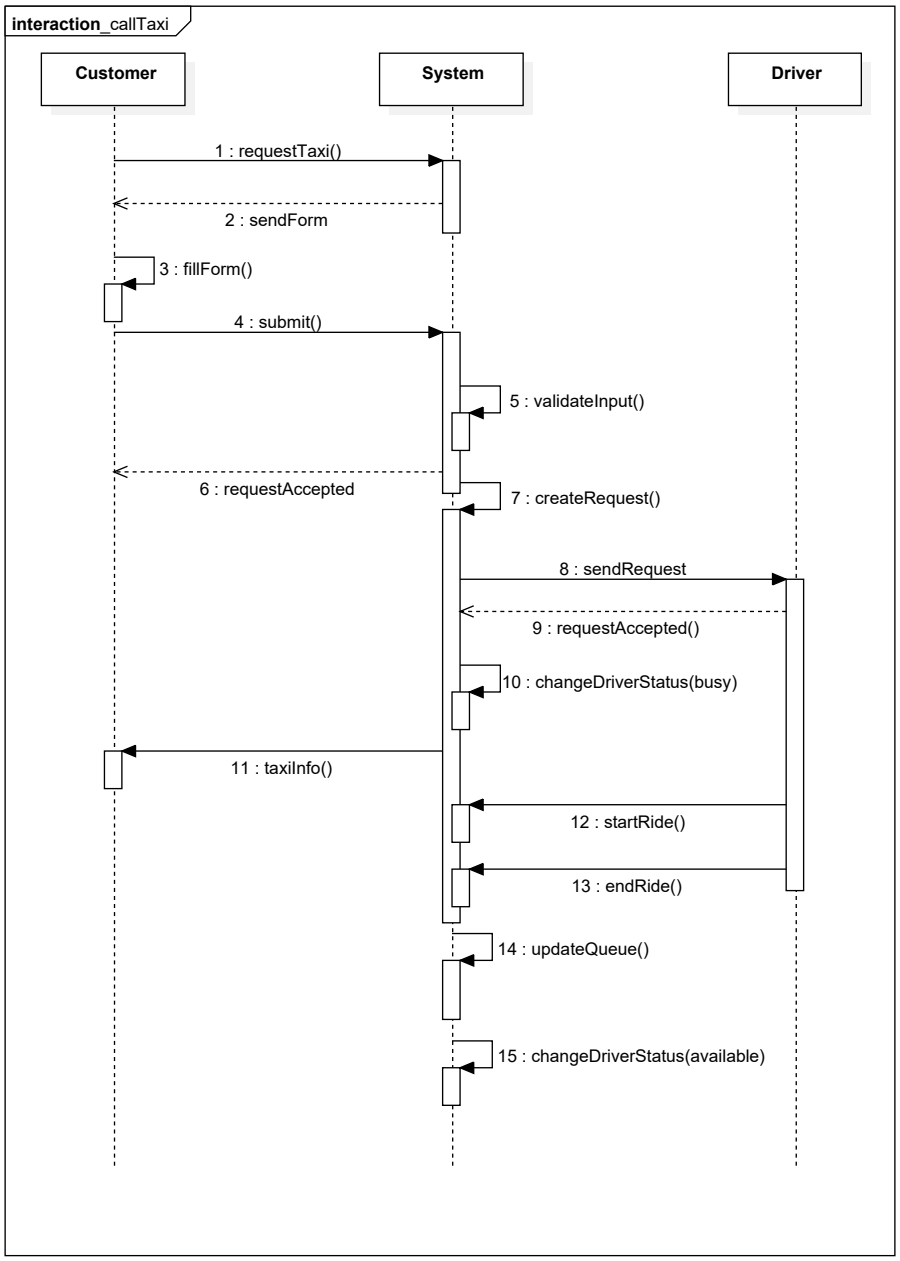

Figure 6: Sequence diagram of the whole process of a taxi booking, including the driver side

3.3.6 Reserve Taxi

Actor	Passenger
Goal	G5
Entry Condition	The passenger is logged in the system
Exit Condition	The passenger successfully books a taxi
Exception	The destination is not valid, the reservation's time is not valid

Related Requirements

1. The system only accepts reservations made at least 2 hours and at most 48 hours before the pick-up time
2. The system must check that the pick-up point is inside the city's zones
3. The system must check that the destination is within 50km from the pick-up point
4. The passenger can abort the operation anytime before the final confirmation
5. Before accepting the reservation, the system must ask for confirmation
6. Once the reservation is accepted the system does not allow the possibility to delete it
7. The system must allocate a taxi 10m before the pick-up time
8. The system must manage the waiting queue in such a way to guarantee that all reservations will be fulfilled

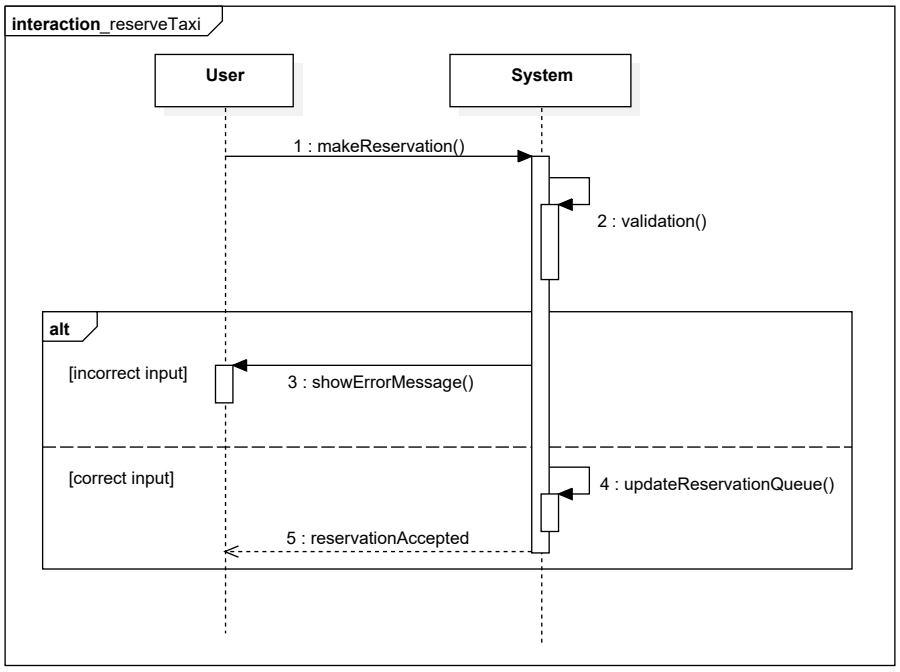

Figure 7: Sequence diagram of the reservation process, when the system forwards the request to the driver it behaves in the same way as in Figure 6, action 8 onward.

3.3.7 Ride Notification and Ride

Actor	Driver
Goal	G6
Entry Condition	The driver is logged in the system and his/her status is set to “available”
Exit Condition	The driver accepts or declines the request
Exception	No taxi accepts the request

When the system forwards a request to a driver, the mobile app will show a pop-up notification. The driver has a limited amount of time to accept or decline it before the system forwards that request to the next driver in the waiting queue. If the Driver accepts the request, then the system offers the functions “star ride” and “end ride”, to allow the driver to notify the system of the beginning and end of the ride.

Related Requirements

1. The system must provide the drivers the option to accept or decline a request
2. The system must deliver request only to drivers whose status is set to "available"
3. The system must send the request to the first driver in the queue
4. If the driver declines the request, the system forwards it to the next taxi in the waiting queue
5. After a driver declines a request the system moves him/her in the end of the queue
6. The system must wait 30s before forwarding the request to the next driver in the queue
7. If the time-out is reached, the driver request is automatically refused
8. The system offers the driver the option to start the ride
9. The system offers the driver the option to end the ride

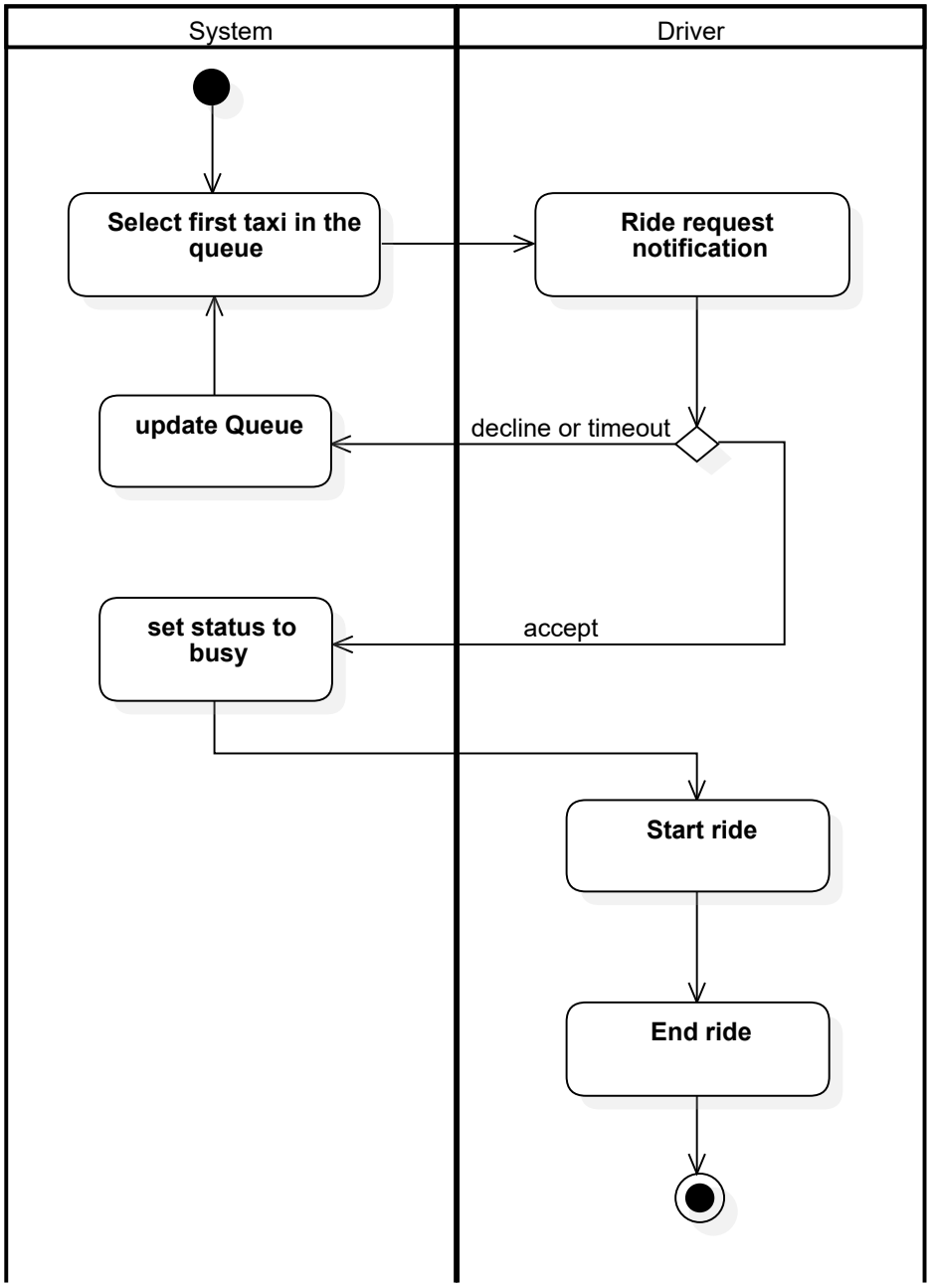

Figure 8: Request forwarding flow chart

3.3.8 Update Status

Actor	Driver
Goal	G7
Entry Condition	The driver is logged in the system
Exit Condition	The driver changed his/her status
Exception	None

The driver can update his status at any moment, but can only set his/her status to available or offline.

Related Requirements

1. The system only allows three possible status: available, busy and offline
2. After a driver accepts a request, his/her status is automatically set to "busy"
3. After a driver ends a ride, his/her status is automatically set to "available"
4. At the end of the shift the driver can set his status to "offline"
5. The driver can't set his/her status to "busy"

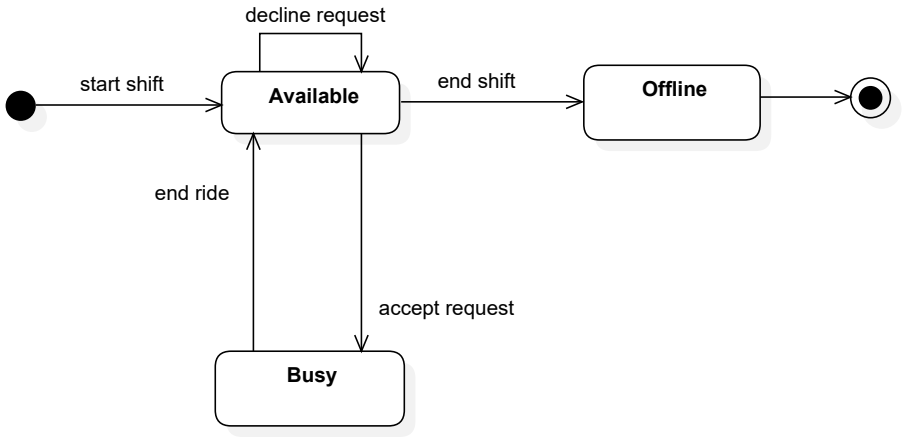

Figure 9: State chart of the driver's possible status

3.4 Non Functional Requirements

3.4.1 User Interface

The user interface has to be as simple and intuitive as possible. Any user should be able to easily understand the basic functions without any previous knowledge of the system. This requirement is to be taken into particular consideration for the driver's interface. Drivers should in fact be able to easily and promptly accept or decline the ride requests due to the confirmation time-out.

3.4.2 Performance Requirements

In order to provide a suitable service, the system has to be reactive and capable of handling hundreds of simultaneous requests. The nature of the application will require to manage real-time request and thus the response time must be such that users feel that the application is reacting almost instantaneously.

In particular the part of the system linked to the drivers' response shall be as quick as possible, to maintain synchronization with the algorithms selecting an available driver from the queue.

3.4.3 Reliability

The system should be available 24/24, 7/7. The server farm should allow daily maintenance without compromising the system's functionalities, since some of them will work in series. Some of the most critical parts, like those who gather and execute passengers' requests, should be made to work in parallel, since with the high number of probing they are the most likely to receive damage.

3.4.4 Security

The system must guarantee that personal and sensible data remains private. In order to fulfill this requirement the system implements a login authentication to protect the information of users and passwords are stored using hashing mechanisms.

3.4.5 Portability

In the programming phase of the project the abstract, logical part of the application and the interfaces must be cleanly divided, since the same software has to work on both smartphones and computer browsers. Moreover the software would gain scalability value, and adding a function in the future will not require double the work to make it usable on both interfaces.

A Alloy

In this appendix the consistency of the following class diagram will be tested using the tool "Alloy Analizer".

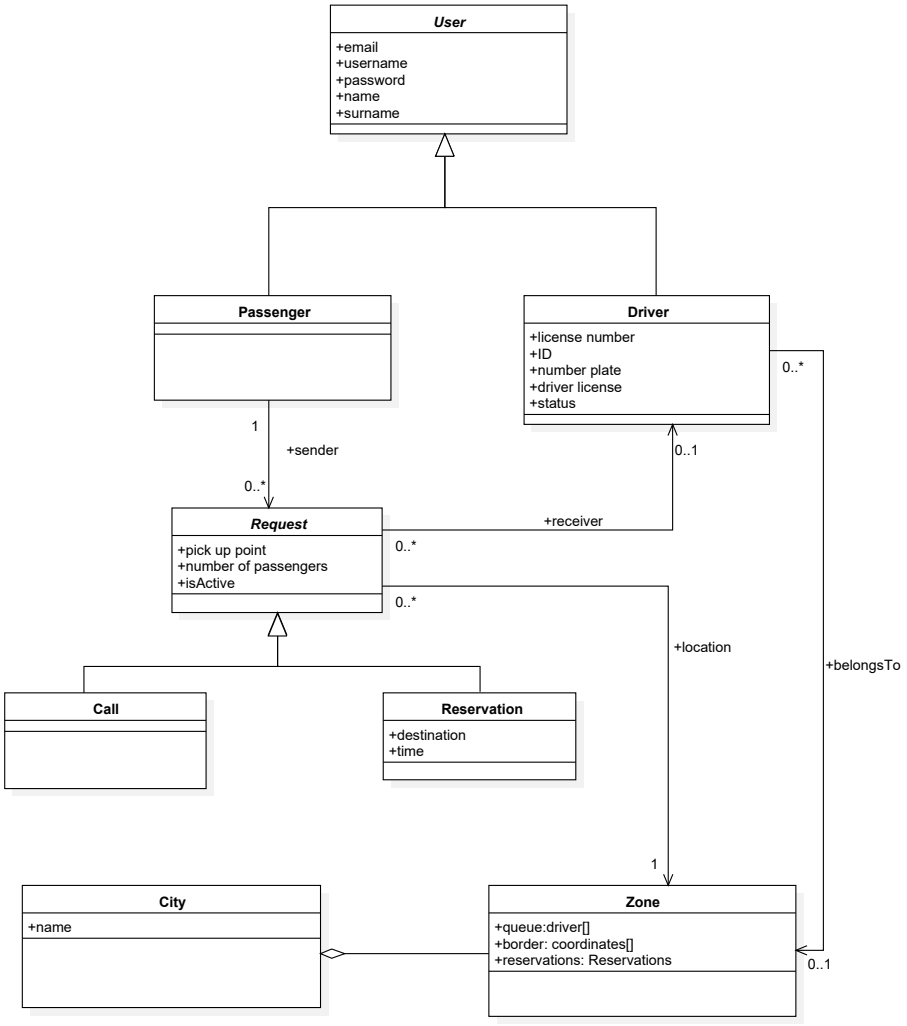

Figure 10: Class Diagram

A.1 Code

Figures 11, 12 and 13 show the alloy code used to model the class diagram. In this model there are non-relevant attributes so that it can be as similar to the class diagram proposed as possible, even though these attributes are not used

to prove the validity and consistency of the model.

```
module myTaxiService

//SIGNATURES//

sig Text{}
enum driverStatus {available, busy, offline}
enum requestStatus {active, pending, inactive}

abstract sig User{
	email: one Text,
	username: one Text,
	password: one Text,
	name: one Text,
	surname: one Text,
	telephone: one Text,
	location: lone Coordinates}

sig Passenger extends User{}

sig Driver extends User{
	licenseNumber: one Int,
	ID: one Int,
	numberPlate: one Text,
	driverLicense: one Text,
	status: one driverStatus}

abstract sig Request{
	pickUpPoint: one Coordinates,
	numberOfPassengers: one Int,
	sender: one Passenger,
	receiver: lone Driver,
	isActive: one requestStatus}

	{numberOfPassengers>=1}

sig Call extends Request{}

sig Reservation extends Request{
	destination: one Text,
	time: one Text}

sig Coordinates{
	latitude: one Text,
	longitude: one Text,
	belongsTo: one Zone}

sig Zone {
	queue: set Driver,
	border: set Coordinates,
	reservations: set Reservation}

one sig City {
	name: one Text,
	zones: set Zone}
```

Figure 11: Signatures

```
//FACTS//

fact userProperties{
//every email is unique
        no disj u1, u2 : User | u1.email = u2.email
//every username is unique
        no disj u1, u2 : User | u1.email = u2.email
//every telephone number is unique
        no disj u1, u2 : User | u1.telephone = u2.telephone
        }

fact driverProperties{
//every license number is unique
        no disj d1, d2 : Driver | d1.licenseNumber = d2.licenseNumber
//every ID is unique
        no disj d1, d2 : Driver | d1.ID = d2.ID
//every plate number is unique
        no disj d1, d2 : Driver | d1.numberPlate = d2.numberPlate
//every driver license is unique
        no disj d1, d2 : Driver | d1.driverLicense = d2.driverLicense
//if a driver is a receiver of an active request, he is busy
        all d : Driver, r : Request | r.isActive = {active} and r.receiver = d implies d.status = {busy}
//if a driver is not offline, he surely has the location attribute
        all d : Driver | d.status!={offline} implies #d.location=1
        }

fact requestProperties{
//there can't be two active requests with the same sender
        no r1, r2 : Request | r1.isActive = {active} and r2.isActive = {active} and r1.sender = r2.sender
//there can't be two active requests with the same receiver
        no r1, r2 : Request | r1.isActive = {active} and r2.isActive = {active} and r1.receiver=r2.receiver
        }

fact zoneProperties{
//the zones are clearly divided
        all disj z1, z2 : Zone | z1 & z2 = none
//in the queue there are only available drivers
        all z : Zone, d : Driver | d in z.queue implies d.status = {available}
//in the reservations set there are only pending reservations
        all z : Zone, r : Reservation | r in z.reservations implies r.isActive = {pending}
        }
```

Figure 12: Facts

```
//FUNCTIONS//

fun activeRequests [] : set Request {
//returns the set of all active requests
	{r: Request | r.isActive = {active}}
	}


//ASSERTIONS//

assert availableDriversNoReceivers{
//a driver who is available is not a receiver of an active request
	no d : Driver | d.status = {available} and d in activeRequests.receiver
	}
check availableDriversNoReceivers for 3

assert offlineDriver{
//if a driver is offline, he is neither a receiver of an active request nor in the queue of a zone
	no d : Driver | d.status = {offline} and d in activeRequests.receiver+Zone.queue
	}
check offlineDriver for 3

assert coordinateInOneZone{
//a coordinate belongs only to one zone
	no disj z1, z2 : Zone, c : Coordinates | c.belongsTo in z1 and c.belongsTo in z2
	}
check coordinateInOneZone for 4


//PREDICATES//

pred showRequest {
	#Passenger>1 and #Driver>1 and #Request>1
	}

run showRequest for 4

pred show {}

run show for 3
```

Figure 13: Functions, assertions, predicates

A.2 Assertions results

The check of the assertions gave these results:

1. **Executing "Check availableDriversNoReceivers for 3"** Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20 3256 vars. 372 primary vars. 6309 clauses. 844ms. **No counterexample found. Assertion may be valid.** 125ms.

2. **Executing "Check offlineDriver for 3"** Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20 3271 vars. 372 primary vars. 6333 clauses. 547ms. **No counterexample found. Assertion may be valid.** 141ms.

3. **Executing "Check coordinateInOneZone for 4"** Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 5304 vars. 584 primary vars. 10124 clauses. 469ms. **No counterexample found. Assertion may be valid.** 47ms.

Since the analizer didn't find any counterexample, the model hasn't been invalidated.

A.3 Worlds generated

These are worlds produced by the alloy model along with the consistency result given by the tool.

Figure 14 shows a world generated by the predicate "showRequest".
The response of the analizer was:

> "**Executing "Run showRequest for 4"** Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 5229 vars. 572 primary vars. 10067 clauses. 297ms. **Instance found. Predicate is consistent.** 343ms.

Figure 15 shows world generated by the predicate "show".
The response of the analizer was:

> "**Executing "Run show for 3"** Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20 3200 vars. 369 primary vars. 6205 clauses. 141ms. **Instance found. Predicate is consistent.** 78ms.

Both predicates are deemed consistent by the analizer. These results, along with the ones given by the assertions, prove that the model presented is both consistent and probably valid.

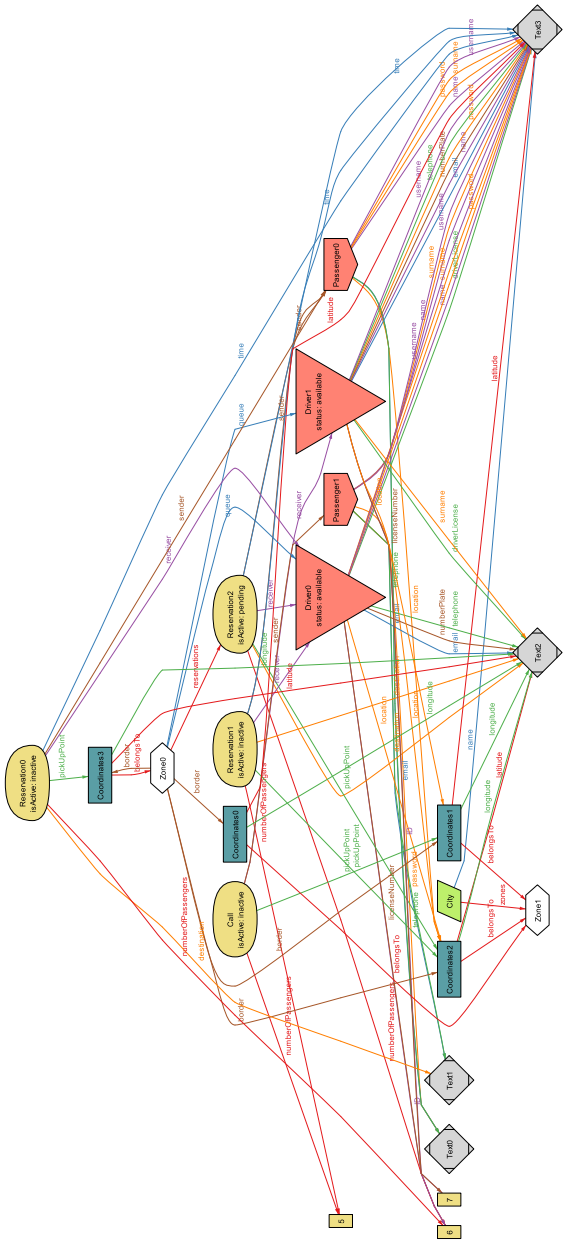

Figure 14: World generated by “showRequest”

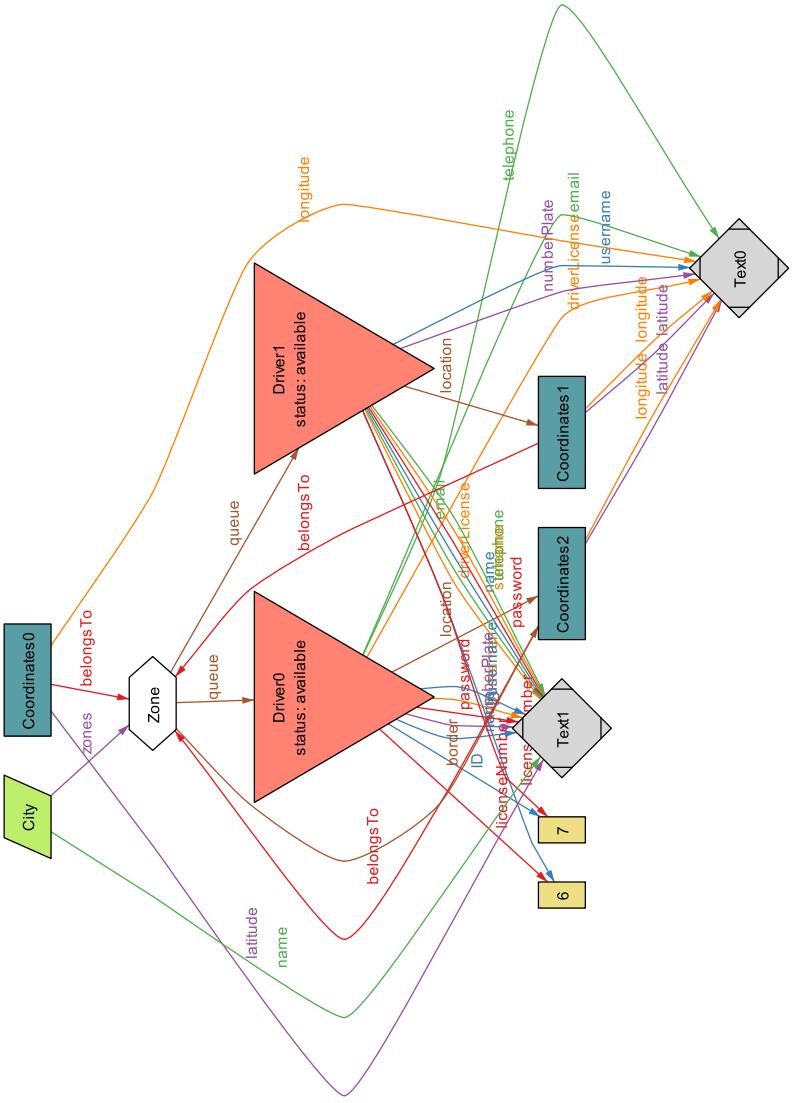

Figure 15: World generated by “show”

B Appendix

B.1 Software and tools

1. *Lyx* to redact and format the document
2. *StarUML* for all the diagrams
3. *Alloy Analyzer* to prove the consistency of our model

B.2 Hours of work

- Carolina Beretta ~ 20h
- Cecilia Brizzolari ~ 20h

B.3 Updates

The following changes have been made in respect to the previous version of the RASD:

1. Added assumption 8, 9 and 10
2. Updated the use case diagram and added the use case “active request”