Politecnico di Milano
Scuola di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Computer Science and Engineering
A.Y. 2015-2016

Software Engineering 2 Project
"myTaxiService"
Requirement **A**nalysis and **S**pecifications **D**ocument
Version 1.4

February 1, 2016

Principal Adviser: Prof. **Di Nitto**

Authors:
Davide Fisicaro 854043
Gianmarco Giummarra 852667
Salvatore Ferrigno 850130

Contents

1 Introduction **1**
1.1 Purpose . . . 1
1.2 Description of the given problem . . . 1
1.3 Scope . . . 1
1.4 Definitions, acronyms, abbreviations . . . 1

2 Overall description **3**
2.1 Purpose of the product . . . 3
2.2 Domain properties and assumption . . . 3
2.3 Identifying Stakeholders . . . 4
2.4 Constraints . . . 5
2.5 Actors identifying . . . 5

3 Requirements **7**
3.1 Functional requirements . . . 7
3.2 Non-functional requirements . . . 10
3.3 User interfaces . . . 11
3.3.1 Customer oriented interface . . . 11
3.3.2 Tablet interface for taxi drivers . . . 15
3.4 Administration Console . . . 17

4 Scenarios identifying **18**
4.1 Scenarios . . . 18
4.1.1 Gigi registers to myTaxiService . . . 18
4.1.2 Franco makes an immediate call . . . 18
4.1.3 Mimmo reserves a taxi . . . 18
4.1.4 Gabry shares a ride . . . 19
4.1.5 Alessio rejects a request . . . 19
4.1.6 Maurizio makes an Assistance call . . . 19
4.1.7 Tiziana makes a rate . . . 20

5 Models **21**
5.1 Use case models . . . 21
5.1.1 Registration . . . 24
5.1.2 Login . . . 24
5.1.3 Private immediate call . . . 25
5.1.4 Private reservation call . . . 26
5.1.5 Shared immediate call . . . 26
5.1.6 New Taxi Driver registration . . . 27
5.1.7 User information update . . . 28
5.2 Class diagram . . . 30
5.3 Sequence diagram . . . 31

6 Alloy modeling **35**
6.1 Modeling 35
6.2 Generated worlds 39
6.2.1 World 1 39
6.2.2 World 2 40
6.2.3 World 3 41

7 Used tools **41**

8 Version history **42**

1 Introduction

1.1 Purpose

The purpose of this document is to provide, organize and communicate the goals and the features of the system to be developed, specifically in terms of functional and non-functional requirements, using graphic models (Class diagram, Sequence diagram, Use case diagram...) and high level specifications of the system requirements. In the last section a formal model of the requirements and the analysis results using the declarative language Alloy will be provided. The document is in fact intended for all the developers, the system analysts and the stakeholders.

1.2 Description of the given problem

The purpose of the system to be developed is the optimization of a taxi service.

The system will be composed of a web app and a mobile app; the mobile app will be designed both from customers and taxi drivers, providing a different interface and different features depending on the kind of current user, while web app is intended for customers only.

A new user can register to the system by creating a new account and providing his personal data.

A registered member can provide informations about his location and reserve a taxi for a specific time, or make an immediate call. The reservation has to occur at least two hours before the ride.

Another feature for registered members is the possibility of sharing a ride with other members, specifying both the starting and the destination point for all the rides he wants to share.

Taxi drivers are able to log in the system with a identification code provided by the company; they will be notified about new customers' calls, and will be able to accept or reject the requests.

1.3 Scope

This project aims to optimize the taxi service of a large City, through the development of the application called myTaxiService; this service will improve the access to the taxi service, simplifying the steps and procedures that the customers have to follow. The benefit for the taxi drivers is the optimization of resources' consuption, thanks to the automatization of the customers' queues management.

1.4 Definitions, acronyms, abbreviations

- User: a person already registered to the system, with his own personal profile. He can login in order to get access to all the functionalities;

- Guest: a person that has never signed up to the system, so without a personal profile. He can only sign up;
- Customer: a person already logged to the system that wants to call a taxi, or a person who is actually using the taxi service;
- Taxi driver: a person who drive a taxi, with his own taxi driver ID provided by the government that hired him;
- System administrator: a person who has privileges in the system and can access in read/write mode to the database through an administration console;
- Login: the action of accessing the system using the e-mail and the related password provided during the registration to the service;
- Mobile application: the application that a user can download on his mobile device;
- Taxi driver tablet application: the application running on the taxi driver's tablet, used to access maps and navigator and to accept or decline customer's requests forwarded by the system;
- Web application: a web page accessible from any device, mobile or not, through a web browser.When the user logs in will have acess to all the functionalities provided in the mobile application;
- Private ride: a ride for only one customer. While the customer don't reach his destination no other customer will be picked up;
- Shared ride: a ride in which one or more customers (up to four) can join in, also in different points of the track: the added tracks will not modify the track of the users already in the car, but can eventually extend the main track. Every user will leave the car in his destination point and the system will calculate the fee for each customer, dividing the cost of the ride between all the customers, taking into account their path and the number of customers with him in every part of the track;
- Immediate call: a call for a taxi which has to reach the customer as soon as possible;
- Reservation call: a reservation for a particular track in a certain hour defined by the customer. This kind of call must be performed at least two hours before the desired time;
- Notification: a real-time alert used to give generic information, for instance, to notice a taxi driver when there is an incoming request of the taxi service;

2 Overall description

2.1 Purpose of the product

The system will not be integrated with other existing or larger systems, but will integrate external APIs (e.g. Google+ and Facebook for additional log in functions); it is designed to be used by the most general public, and in particular in mobility situations. For this purpose users will be able to use it through a mobile application, but also a web application will be available for Desktop environments.

2.2 Domain properties and assumption

In this section some ambiguous aspects of the domain (...) are clarified through the addiction of some hypotheses to better explain the interaction between the external environment and the developed application.

- Every person declares only true and correct personal information about himself;
- The system dosen't allow the registration of a minor member;
- An user have to book a taxi if and only if he is sure he will use the service;
- Every user can not book more than one taxi in overlapping times;
- A ride can be either shared or private;
- Each requested cab will surely reach the customer in the established time and will bring him to the destination;
- Every private customer's immediate taxi request is processed by the system and forwarded to the first available taxi driver only in the queue of the same zone where the request comes from;
- Every customer's call, except private immediate call, is processed by the system and forwarded to the first available taxi driver at first in the queue of the same zone where the request comes frome, and, if the search doesn't give results, it get extended to the closest zone and so on.
- When a taxi driver accepts a customer, the navigator in the application will show the minimum path for reaching the customer's destination;
- Each cab belonging to the service is homologated with five seats, including the driver's one;
- Each private reservation made by a customer refers to the single customer;
- Each shared reservation or shared immediate call made by a single customer corresponds to a single person;

- When a customer make a shared reservation or a shared immediate call he is really the only one to enjoy the service;
- The system analyses all the requests made with enabled "taxi sharing mode" and matches the ones that have a part of the path in common with others, trying to maximize the number of sharing customers;
- When the system matches the different routes of a shared ride, it takes in account a certain tolerance for eventually establish a detour of the route; the showed path will be refreshed by the system in case of eventual detours due to a new customer's addition;
- If a taxi driver notices any kind of health or technical issues must use the assistance service provided inside the application;
- When a user sign up the system will send a confirmation email to the address the user provided during registration;
- An user can complete the registration to the service only if has read and accepted the regulatory policies and terms and conditions of the contract, including the automatic payment authorization;
- Each registered user is related to one and only one e-mail address;
- Each customer is related to one and only one password;
- Each taxi driver is related to one and only one taxi driver's ID, provided by the government;
- Each taxi driver related to one and only one password, provided and periodically changed by the government;
- Each taxi driver is provided with a tablet device by the government;
- Each call concerns one and only one ride;
- The system can't store the same ride in his memory twice;
- The system can't store the same call in his memory twice;
- The system administrator registers to the system taxi drivers recognized by the government, and whose licenses are registered and validated;
- Each one of the taxi drivers recognized by the government is registered to the service from the system administrator;

2.3 Identifying Stakeholders

The stakeholder of this project is the government of a large city.

2.4 Constraints

The main constraints are the following:

- The use of J2EE platform and EJB (Enterprise Java Beans) for the implementation of the application;
- The access to an SMTP server for sending notifications to users via e-mail;
- The use of a DBMS to store all kind of data (e.g. users and operations);
- The use of a SSL (Secure Socket Layer) while transmitting sensitive data such as payment informations;
- The use of a handshake protocol (ack-nack);

2.5 Actors identifying

The system provides the interaction of two different types of actors who can exploit different functionalities of the application system. The types are set out below along with a brief description of the functionality of the basis that the system should them allow.

Guest

A guest is a person that is not registered. He can only sign up to the system through the registration form.

Registered user

A registered user is a person whose data is stored into the system's Database. There are two different types of registered user with different roles into the system, listed below including all the possible actions they can do:

Taxi driver

- login/logout
- accept or decline the customers' requests forwarded to him by the system
- ask for assistance
- update personal informations

Customer

- login/logout
- reserve a ride for a specific time in a specific position
- make an immediate call in his current position

- activate the "sharing function"
- update personal informations
- update payment methods
- rate taxi drivers
- having a look at his reservations' table
- having a look at the bills informations about a specific ride

System administrator

The system administrator access the system through a System administrator Console, which allows to do the following actions:

- login/logout
- add information about a new taxi driver
- manage user data
- general information management

3 Requirements

In this section will be analyzed in detail functional and non-functional requirements that the system developed has to satisfy, when the domain properties previously denoted hold and referring to the declared goals.

3.1 Functional requirements

The functional requirements include the functionalities that the system must necessarily have and describe the interactions between the system developed and the external environment independently from the implementation.

Registration of a person into the system:

- The system has to guarantee the registration to all the new major users who want to create a new account;
- In order to complete the registration process, the system sends a confirmation link to the new user via email;

Login of a person already registered to the system:

- The system has to allow the login to an already registered customer when he types the correct email and password in the login form.
- The system has to allow the login to a taxi driver when he types the correct identification code and password in the login form;

Providing a payment method:

- When a customer provides his payment informations, the system verifies the validity of the submitted informations and in case of success unlocks all the allowed functionalities;

Private immediate calling of a taxi:

- The system, before unlocking all the allowed functionalities, checks the validity of the submitted payment informations;
- The system notifies the confirm of the call with an email and has to assign the first taxi in the queue of the zone where the call comes from, if it is not empty;
- When a taxi is assigned to a ride, the system removes the selected taxi from the starting zone queue;
- If the queue of the zone is empty, the system has to notify it to the calling customer and aborts the current operation;

Shared immediate call of a taxi:

- The system, before unlocking all the allowed functionalities, checks the validity of the submitted payment informations;
- The system asks the user to specify his destination too;
- The system matches the route with the other shared calls (reservation or immediate calls), compatible both for the time and the path;
- When a taxi ends a shared ride, the system moves the selected taxi to the last position of the current zone queue;
- If there isn't any compatible shared ride to match with, the system normally assignes the first available taxi of the queue;
- If the queue of the zone corresponding to the starting position is empty, the system has to forward the request to the first available taxi of the closest zone.

Private taxi reservation:

- The system, before unlocking all the allowed functionalities, checks the validity of the submitted payment informations;
- The system asks the user both the starting position and the destination point;
- The system, before processing the request, checks the validity of starting and destination points;
- The system notifies the confirm of the reservation with an email and allocates a taxi to the request 10 minutes before the meeting time with the user, notifying the first available taxi of the queue of the starting address' zone; if, in the first 5 minutes of the 10, the system doesn't find any ride to match the path with, or any available taxi driver, it notifies a delay to the customer and retry in 5 minutes; if also the second try fails, aborts the operation and notifies the user.
- If the queue of the zone corresponding to the starting position is empty, the system has to forward the request to the first available taxi of the closest zone.

Shared taxi reservation

- The system, before unlocking all the allowed functionalities, checks the validity of the submitted payment informations;
- The system asks the user both the starting position and the destination point;
- The system, before processing the request, checks the validity of starting and destination points;

- The system matches the route with the other shared reservation calls, compatible both for the time and the path, and all the possible rides corresponding to immediate calls that will pass through the starting point at the specified time;
- If the queue of the zone corresponding to the starting position is empty, the system has to forward the request to the first available taxi of the closest zone.

Accepting of a request by the taxi driver:

- The system has to effectively assign the accepting taxi driver to the correspondant request;

Rejecting of a request by the taxi driver:

- The system has to forward the request to the second in the queue and, at the same time, has to move the first taxi in the last position in the queue;

Assistance calling by the taxi driver:

- Depending on the kind of signaled issue (e.g. accident, system failure, et cetera), the system has to get the taxi driver in contact with the correspondant service;

Payment transactions:

- When a ride ends the system automatically gets the corresponding amount from the customer's current account;

Rating a taxi driver:

- When any ride ends the system automatically saves the taxi driver's profile in the specific section of the user's app which did that ride in order to allow the user to rate the taxi driver;

Login of a system administrator to the administration Console:

- The system has to allow the login to a system administrator when he types the correct id and password in the login form of the administration Console.

Registration of a new Taxi driver into the system:

- The system has to guarantee the registration of a new Taxi driver to the System administrators;

Management of Registrated users information:

- The system has to guarantee the access to users information to the System administrators;

Management of general information:

- The system has to guarantee the access to general information to the System administrators;

3.2 Non-functional requirements

The non-functional requirements are those not related to the functionality, but rather consider the quality of the system to be implemented (Quality of Service, QoS), regardless of the application domain.

Reliability and stability:

- The system guarantees a good level of reliability during the use of the service, satisfying all the ACID properties for each transaction, fulfilling the functions and the tasks for which the system itself had been developed. Furthermore it will be able to cope with the possible loss of data due to non-desired or accidental events by ensuring atomicity of all the operations.

Concurrency management:

- The system guarantees data consistency in the DBMS associated with the system, facing multiple access of authenticated users to the same resources.

Security:

- Each user has a finite number of functionalities that can access: the system will deny any request of unauthorised access to data or functionalities. In order to preserve the security, the system will provide an authentication mechanism with e-mail and password for the authentication using the standard security protocols, and will use an SSL protocol to guarantee the security during sensible data transfer.

Performance:

- All the possible operations offered by the system are executed in acceptable time and don't cause any system block or failure. The system is able to balance all the above constraints and efficiency for the user at the same time. These requirements are guaranteed in presence of a decent internet connection, needed for most of the operations.

Portability:

- The mobile application is developed for all the principal mobile operating systems (Android, iOS, WP) supporting the APIs used in the development of the software.
- The web client is compatible with all hardware and software platforms, aiming to reach the maximum number of users as possible. It can be accessed by any web browser simply having an internet connection.
- The tablet application for taxi drivers is developed for a single operating system running on their devices.

3.3 User interfaces

The system will be provided of three different user interfaces:

- A mobile interface for customers, which includes both smartphones and tablet layout.
- A tablet interface for taxi drivers, which will not be compatible with smartphone.
- A web interface for customers.

3.3.1 Customer oriented interface

At the first opening of the application the user will be asked to register to the system, or login if already registered. The app provides two authentication methods: standard e-mail or social network login, through Facebook or Google+. Once logged, the user's experience begins with the start page showing a map with the actual position and some shortcuts for fast reservations. The sliding menu provides the basilar functions that almost any app should have, such as settings, FAQ, app rating and so on; furthermore will present shortcuts to the functionalities, e.g. taxi sharing service and taxi reservation. In each of those functionalities the user must provide all the required data to begin its experience. After a reservation, the new home page will provide, in addition to the usual start page functionalities, some information about the riding status. This interface will be available for Desktop systems and Mobile systems (both smartphone and tablet devices).

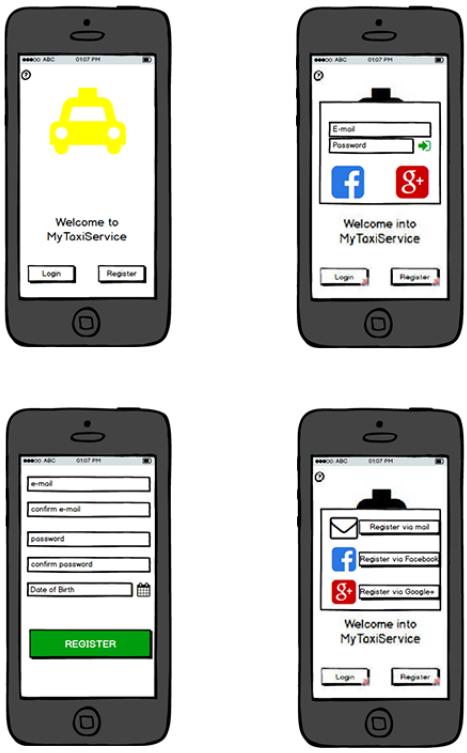

Figure 3.1: Starting page layouts. These layouts represent the first screens the users will interact with, and show the different registration and login methods previously described.

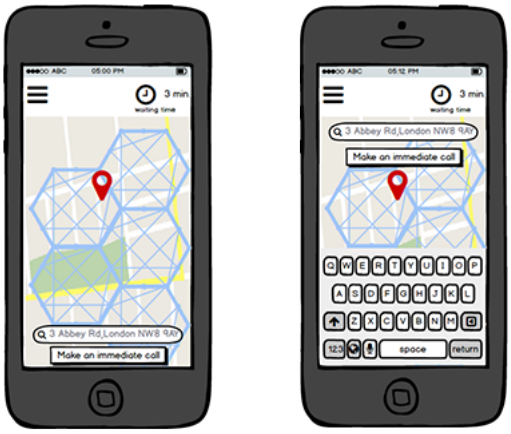

Figure 3.2: Immediate call screen. By default, the system get the customer's position from the GPS signal, but the customer can manually specify it by tapping the search area and type the address.

Figure 3.3: Reservation and location screens.

Figure 3.4: Sliding menu and taxi sharing screen, which will show all the possibility of sharing the reserved ride.

3.3.2 Tablet interface for taxi drivers

The taxi driver application is intended to be used only on tablets (that should be provided by the company) with a screen wider from 7" to 10" inches, in order to provide a more clear and usable layout while the user is driving. At every opening, the app will ask for credential in order to login, using an univocal taxi driver ID provided by the company. Once logged the application will show the map indicating the current location of the taxi while waiting for customers assigned by the system. When a request arrives the driver will be able to see the details of the ride (starting and destination point, arriving time and so on) and will decide whether to accept or decline the request. If the ride is accepted the application will show the map with driving directions, and no other things, in order to avoid the use of the application while driving with a customer. In every screen (except for the navigation one) there will be a sliding menu providing all the app functionalities, such as settings, FAQ, shortcuts to navigation mode, incomes and runs history, and an "assistance button" which will open a screen for reporting any kind of problem and eventually call the assistance.

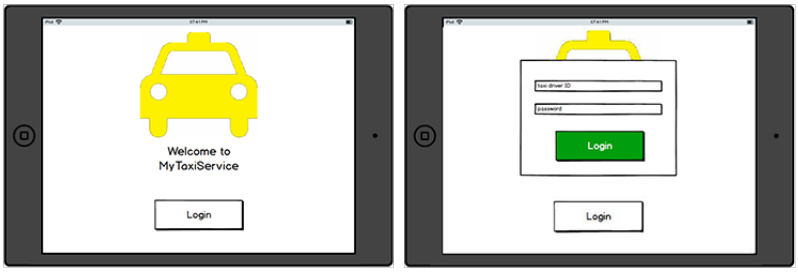

Figure 3.5: Login screens in taxi driver's device.

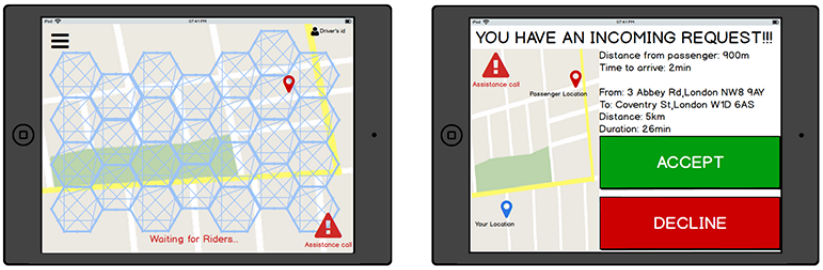

Figure 3.6: Waiting and request managing screens.

Figure 3.7: Riding and assistance calling screens.

3.4 Administration Console

The system provides a console, available only for system administrators with admin privileges, in order to allow the management of user such as the registration of a new Taxi Driver and the management of the registered users and of general system information.

For example, when a new Taxi driver gets hired by the company, an admin has to insert manually his data in order to provide him with all the functionalities of myTaxiService.

Another example could be the management of registered users when, for example, a user has a generic problem and his data have to be verified and eventually modified by an administrator.

The console is available only in desktop systems, accessible through the browser.

The mockups are omitted because the console must have a very essential design, with a simple login form and all the information available in a related section of the desktop screen.

4 Scenarios identifying

This section will present some possible situations that may occur from the interaction between a user and the system developed.

4.1 Scenarios

4.1.1 Gigi registers to myTaxiService

Gigi is a business man who needs constantly to move around the city for his meetings and decides to become an user of this new taxi service. He downloads the app from the mobile store and follows the steps proposed by the system to complete the process entering his own email, a password, his date of birth. When he finishes, the system notifies him via email sending the confirmation link to the email addres he gave during the registration process. He visits the confirmation link and become an effective user of the system, able to log in to the system.

4.1.2 Franco makes an immediate call

Franco, an already registered user, after logging in to the myTaxiService's mobile application and providing a valid payment method, decides to make an immediate call to go to the Prince of Wales Theatre (Coventry St, London W1D 6AS). In order to do so, he ensures that the position detected by the GPS and shown in the application's map is correct and, before tapping the "Make an immediate call" button, he checks the estimated waiting time for the taxi arrival. He decides to confirm and effectively tap the button.

The system processes the request, assings the first taxi in the queue of the zone where Franco has made the call from and notifies Franco.

The taxi driver receives the notification in his device and accepts the request related to Franco's call.

At the end of the ride the system automatically gets the corresponding amount from the customer's current account.

4.1.3 Mimmo reserves a taxi

Mimmo, an already registered user, after logging in to the myTaxiService's web application and providing a valid payment method, decides to reserve a taxi for his date with Donatella at 9 pm.

In order to do so, he chooses the reserve section from the app menu, provides the starting position and his destination and choose from the specific form the desired time.

He decides to confirm and effectively taps the "confirm reservation" button.

The system confirms the reservation to the user by sending a notifcation via email and allocates a taxi to the request 10 minutes before the meeting time with the user.

At the end of the ride the system automatically gets the corresponding amount from the customer's current account.

4.1.4 Gabry shares a ride

Gabry is at his home and has to go to his band's rehearsal room for playing some music.

His car is broken, so he decides to take a taxi.

He makes the log in to the service through the mobile application and has already submitted and being verified his payment informations.

He decide to enable the "sharing mode" in order to save money.

After submitting the starting and destination address, the system process the request, notifies to him that there are two person, Emanuele and Maria, which share a part of his ride and matches the Gabry's ride to the route of the taxi which the system previuosly selected according to the compatibility with the other person's ride.

Gabry decides to confirm and taps on the "confirm sharing" button.

The system defines the fee for all persons sharing the taxi and informs Gabry and the other passengers; furthermore the system informs the taxi driver about the new route to follow.

At the end of the ride the system automatically gets the corresponding amount from the customer's current account.

4.1.5 Alessio rejects a request

Mario, a head physician, is at his home and has to go to the Saint Mary's hospital.

In order to do so, after logging in to the myTaxiService's mobile application and providing a valid payment method, he decides to make an "immediate call".

The system processes the request and assings the first taxi in the queue of the zone where Mario has made the call from.

Alessio, the first taxi driver of the queue, receives the notification in his device and rejects the request related to Mario's call, then the system forwards the request to the second in the queue and , at the same time, move the Alessio's taxi in the last position in the queue.

The system confirms to Mario by sending a notifcation via email and at the end of the ride automatically gets the corresponding amount from the customer's current account.

4.1.6 Maurizio makes an Assistance call

Maurizio is a taxi driver who works at myTaxiService, during a ride he accidentally punctures a tyre.

He decides to make an "Assistance call" informing the myTaxiService system of his problem.

In order to do so, he taps on the "assistance call" botton on his device, then the system asks to Maurizio to specify the kind of problem occurred.

He taps on “Other” botton and writes in the special area :“puncuture of a tyre”, the systems puts the Maurizio’s taxi out of service and forward the request to the technical service which will fix the problem.

4.1.7 Tiziana makes a rate

Tiziana, an already registered user, at the end of a ride using myTaxiSerivce is very satisfied with taxi driver’s service and decides to rate him.

In order to do so he chooses the rate section from the app menu and gives her rate .

5 Models

In this section an abstraction from the previously seen scenario is made, in order to have a more high-level description. UML (Unified Modelling Language) diagrams are used for this purpose.

5.1 Use case models

From previously denoted scenarios and from the whole analysis done in this document, the main use cases of the system to be developed have been detected. In these pictures there are some use Cases diagram which represent actors, their use cases and their interactions. Then some of them are better explained using a less formal and more narrative way.

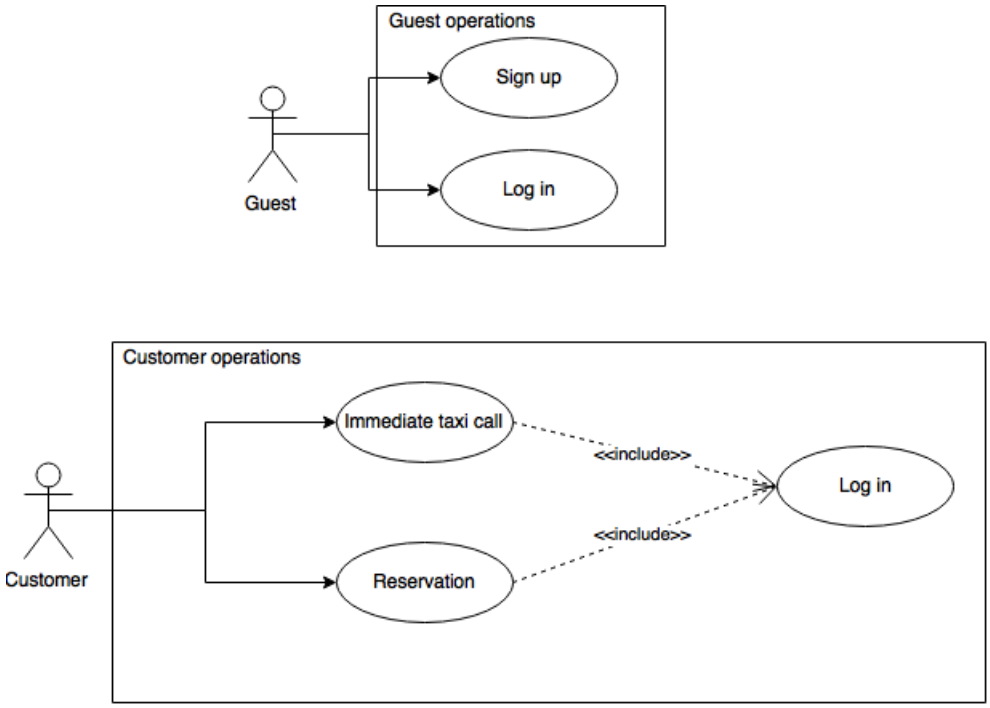

Figure 5.1: first use case model

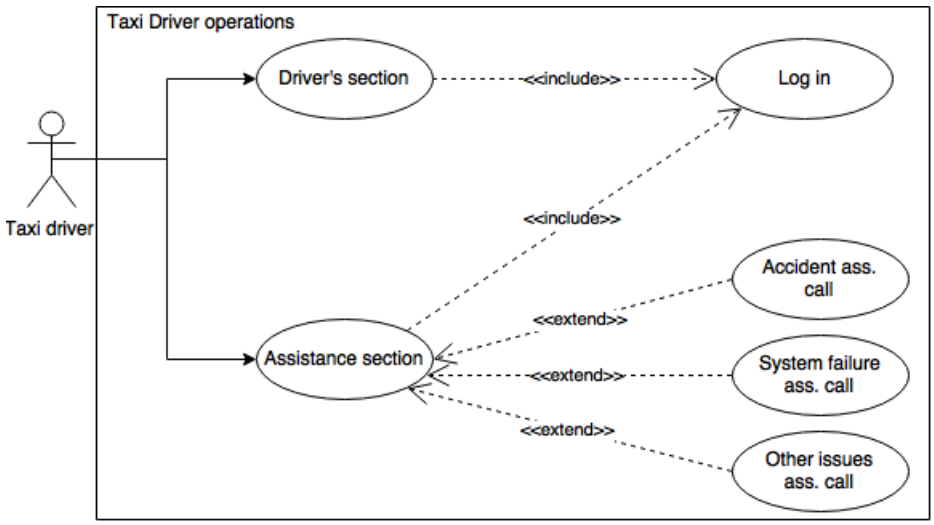

Figure 5.2: second use case model

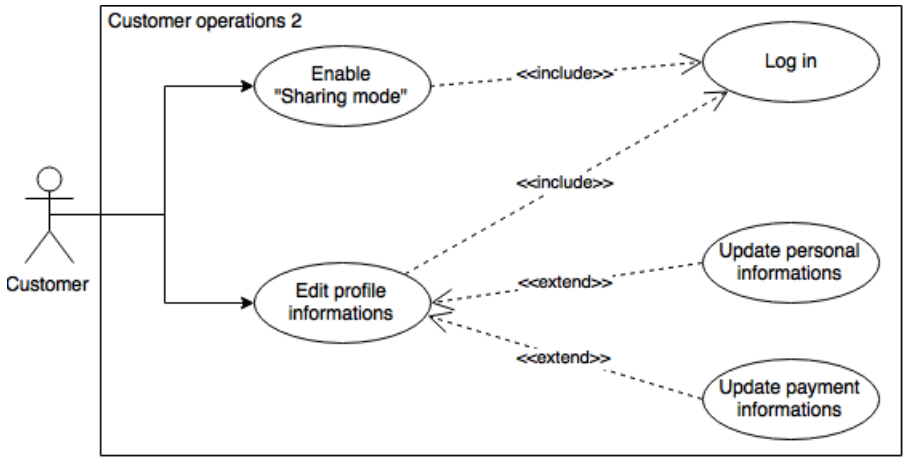

Figure 5.3: third use case model

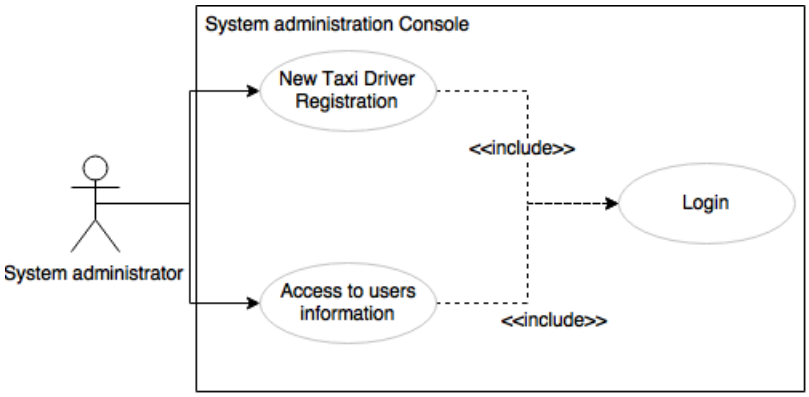

Figure 5.4: fourth use case model

5.1.1 Registration

Actors: Guest.

Preconditions: The user has not already signed up.

Events flow:

1. The guest opens the home page of the mobile app or the one of the web application
2. The guest taps the "register" button on the home page
3. The system shows the registration form in which requires the guest to enter his email, password and date of birth
4. The guest types the requested information and taps the "Register" button again
5. The system verifies the equality of the two typed email and of the two typed passwords
6. The system notifies the guest via email sending the confirmation link to the email addres submitted during the registration process
7. The guest visits the confirmation link and become an effective user of the system, then he logs in to the system

Postconditions: The guest has signed up and becomes a user.

Exceptions: The email or the nickname the guest typed has been already used. The second email does not match with the first one. The second password does not match with the first one. The date of birth does not corresponds to an adult person. In these cases the user can't complete the registration step. The system notifies the error and prepares the guest for the reintegration of data for the registration.

5.1.2 Login

Actors: User.

Preconditions: User has successfully signed up to the system.

Events flow:

1. The user opens the home page of the mobile app or the one of web application
2. The user taps the "login" button on the home page
3. The system shows the login form in which requires the user to enter his email (ID in the case of a taxi driver) and password

4. The user taps the "Login" button again
5. The system verifies the correctness of the personal information entered
6. The system redirects the user to his own personal page

Postconditions: The user is logged in.

Exceptions: Either the email (ID in the case of the taxi driver) or the password is not valid. The system notifies the error and prepares the user for the reintegration of data for log in

5.1.3 Private immediate call

Actors: Customer.

Preconditions: The customer has succesfully logged in, provided his payment informations and he is surfing his personal page.

Events flow:

1. The customer reachs the map of the app for make an immediate call
2. The system checks the validity of the submitted payment informations of the customer and unlocks all the allowed functionalities
3. The customer ensures that the position detected by the GPS and shown in the application's map is correct
4. The customer checks the estimated waiting time for the taxi arrival
5. The customer taps the "Make an immediate call" button
6. The system processes the request and assings the first available taxi in the queue of the zone where the request comes from
7. The system notifies the confirmation of the call via email
8. At the end of the ride the system automatically gets the corresponding amount from the customer's current account

Postconditions: The system updates the queue of the zone. The selected taxi driver results occupated now.

Exceptions: The payment information are not valid, then the system locks all the functionalities. The taxi driver rejects the request, than the system has to forward the request to the second in the queue and, at the same time, has to move the first taxi in the last position of the queue.

5.1.4 Private reservation call

Actors: Customer.

Preconditions: The customer has succesfully logged in, provided his payment informations and he is surfing his personal page.

Events flow:

1. The customer selects the reserve section from the app menu
2. The system checks the validity of the submitted payment informations of the customer and unlocks all the allowed functionalities
3. The systems requires the customer to enter the starting position and the destionation point of his ride, and the desired time
4. The customer types the requested information and press "Confrim reservation" button
5. The system confirms the reservation to the user by sending a notifcation via email
6. The system processes the request
7. The system allocates a taxi to the request, taken from the zone correspondig to the starting position, 10 minutes before the meeting time with the user
8. The system notifies the confirmation of the reservation via email
9. At the end of the ride the system automatically gets the corresponding amount from the customer's current account

Postconditions: The selected taxi driver results busy from 10 minutes before the reserved ride.

Exceptions: The starting position or the destination position or the time are not valid, then the system notifies the error and prepares the user for the reintegration of data for reservation. The payment information are not valid, then the system locks all the functionalities.

5.1.5 Shared immediate call

Actors: Customer

Preconditions: The customer has succesfully logged in, provided his payment informations and he is surfing his personal page.

Events flow:

1. The customer enables the sharing mode from the app menu

2. The customer reaches the map of the app in order to make a shared immediate call
3. The system checks the validity of the submitted payment informations of the customer and unlocks all the allowed functionalities
4. The systems requires the customer to enter the starting position and the destionation point of his ride
5. The customer types the requested information and press "Confirm reservation" button.
6. The system matches the customer's ride to the route of the taxi which the system previuosly selected according to the compatibility with the other persons' ride.
7. The system notifies to the customer that there are two person which share a part of his ride.
8. The customer decides to confirm and taps on the "confirm sharing" button
9. The system processes the request
10. The system confirms the shared call to the user by sending a notifcation via email
11. The system notifies the taxi driver the modification of his route
12. At the end of the ride the system automatically gets the corresponding amount from the customer's current account.

Postconditions: The shared taxi now increases of one the number of the passengers.

Exceptions: The starting position or the destination position is not valid, then the system notifies the error and prepares the user for the reintegration of data for reservation. The payment information are not valid, then the system locks all the functionalities.

5.1.6 New Taxi Driver registration

Actors: System Administrator

Preconditions: The system administrator has succesfully logged in.

Events flow:

1. The system administrator access to the Administration Console
2. The system administrator access to the Register New Taxi Driver section of the page

3. The system shows the registration form
4. The system administrator fills the fields of the form with the information about a new taxi driver
5. The system checks the validity of the information
6. The system checks that the taxi driver has not been registered before
7. The registration process ends
8. The system notifies via email the new taxi driver sending a confirmation link
9. The system notifies the System Administrator the success of the operation
10. The new taxi driver visits the confirmation link and become an effective taxi driver of myTaxiService

Postconditions: The taxi driver is signed up and is ready to configure the provided tablet device to use in his cab.

Exceptions: The information inserted are related to an already registered taxi driver. The inserted information are not valid. In both cases the system notifies the error to the system administrator and restart the operation.

5.1.7 User information update

Actors: System Administrator

Preconditions: The system administrator has succesfully logged in.

Events flow:

1. The system administrator access to the Administration Console
2. The system administrator access to the User Information section of the page
3. The system shows the selected functionality
4. The system administrator select the specific user to update
5. The system administator select the “update” action
6. The system shows the User Information in read/write mode
7. The system administrator fills the fields to update
8. The system checks the validity of the inserted information
9. The update process ends

10. The system notifies the user of the updates via email, with a list of all the changes
11. The system notifies the System Administrator the success of the operation

Postconditions: The user's information are updated, and he can visualize them in the Information section of his profile Page.

Exceptions:

5.2 Class diagram

The following class diagram doesn't take in account the classes needed for the system administration side of the system. This doesn't affect anything because that is not a big part of the system.

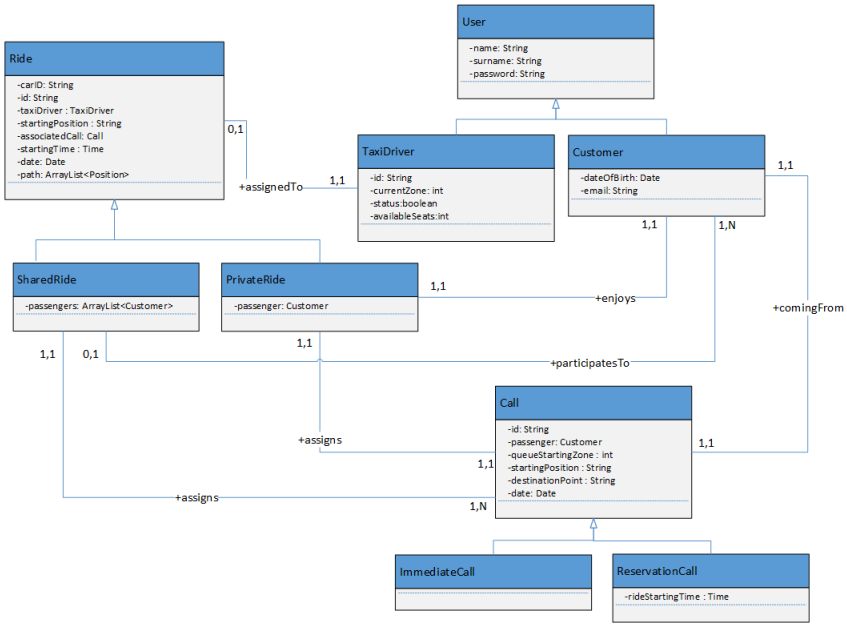

Figure 5.5: class diagram

5.3 Sequence diagram

This section presents the sequence diagram of the most important interaction, in order to have a dynamic sight of the main entities too. Some assumptions have been made in order to simplify the diagrams in some particular cases.

For example:

- If the log in or the registration procedure fails, the system will ask information resubmission again, until the information are valid or the guest decide to quit the operation;
- The immediate call sequence diagram (figure 5.7) refers to a private immediate call, and doesn't consider the shared call case. This assumption was made in order to clearify and simplify the sequence diagram;
- The reservation call sequence diagram (figure 5.8) refers to a shared reservation call and the payment information are assumed to be valid.

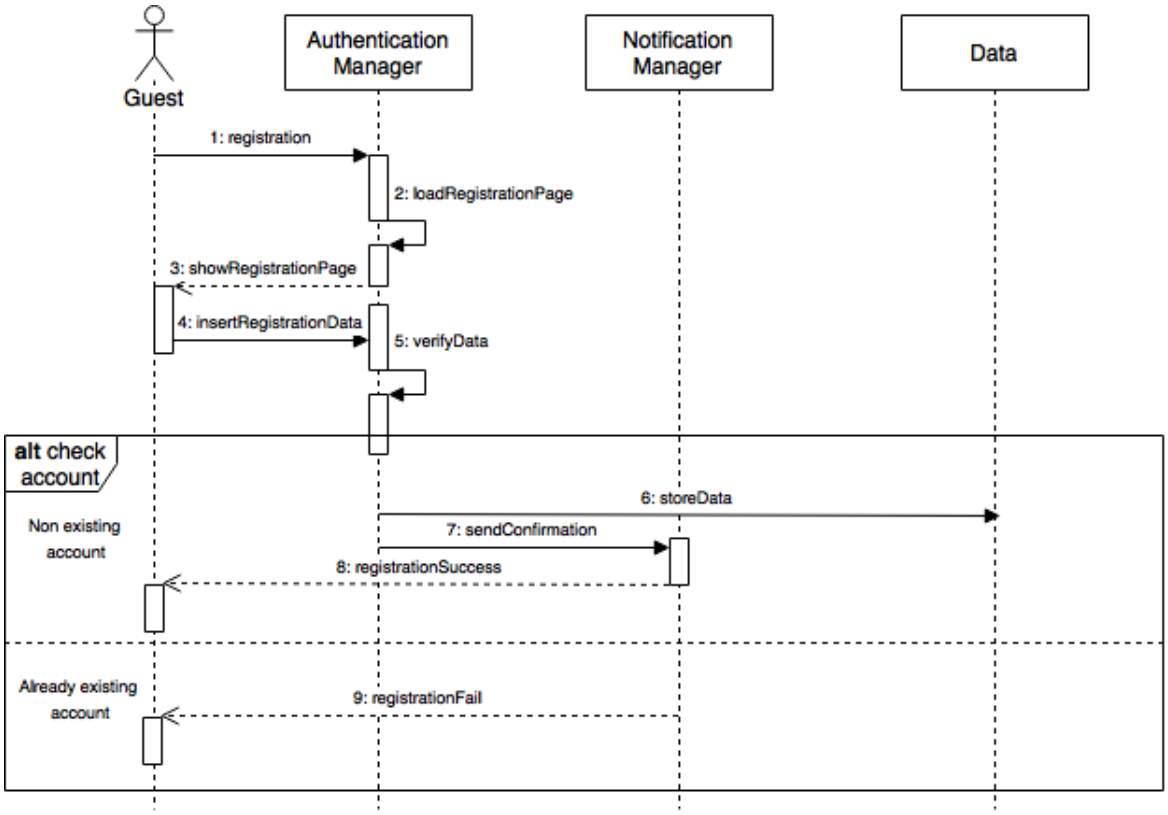

Figure 5.6: registration sequence diagram

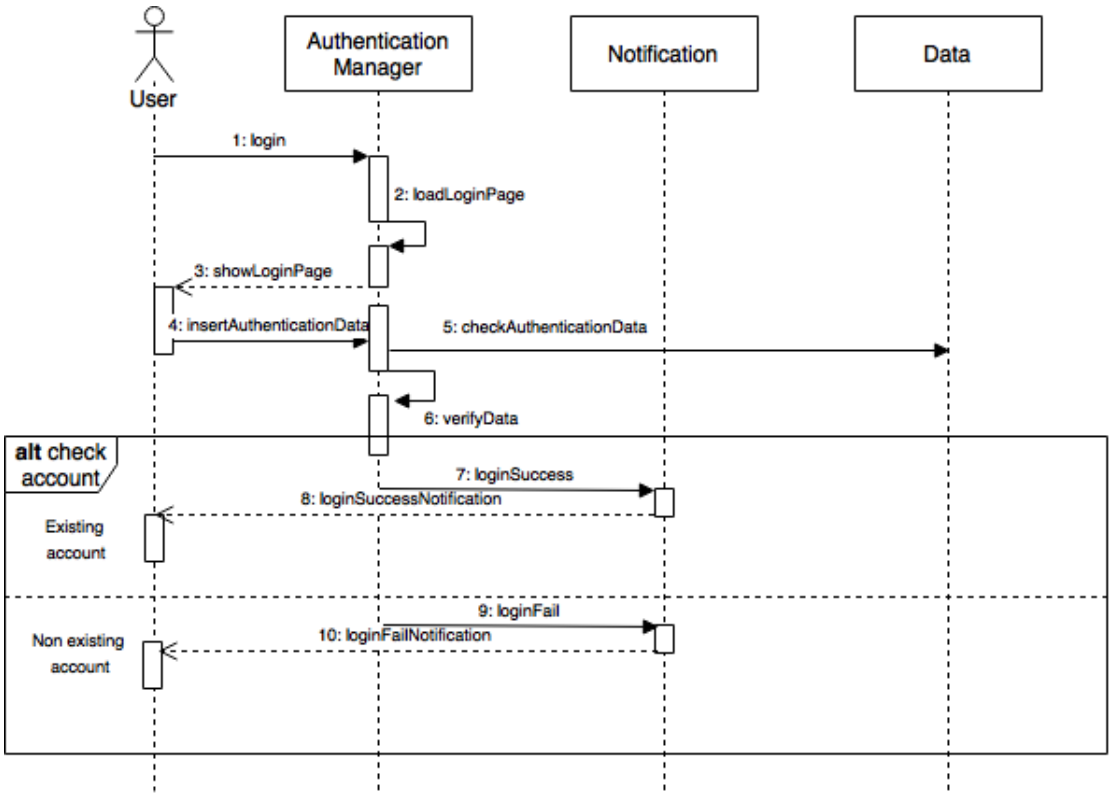

Figure 5.7: login sequence diagram

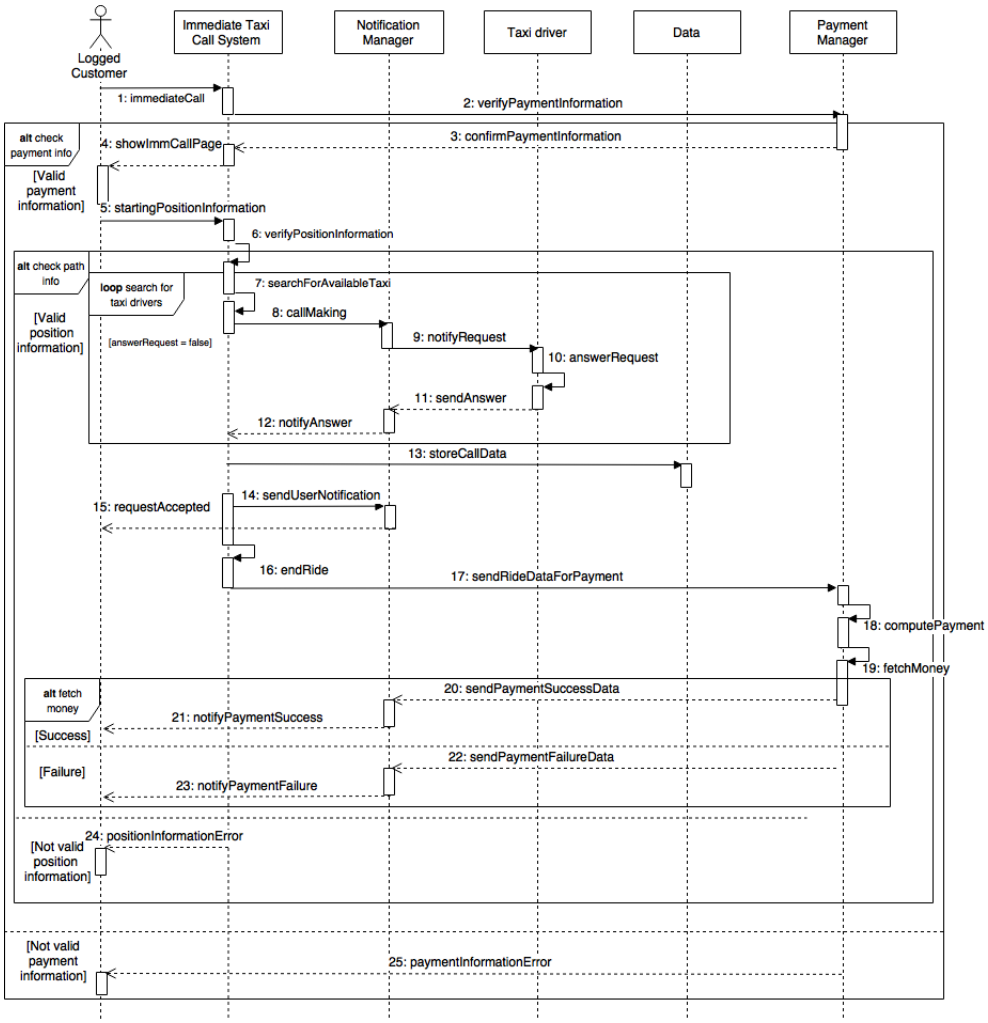

Figure 5.8: Private immediate call sequence diagram

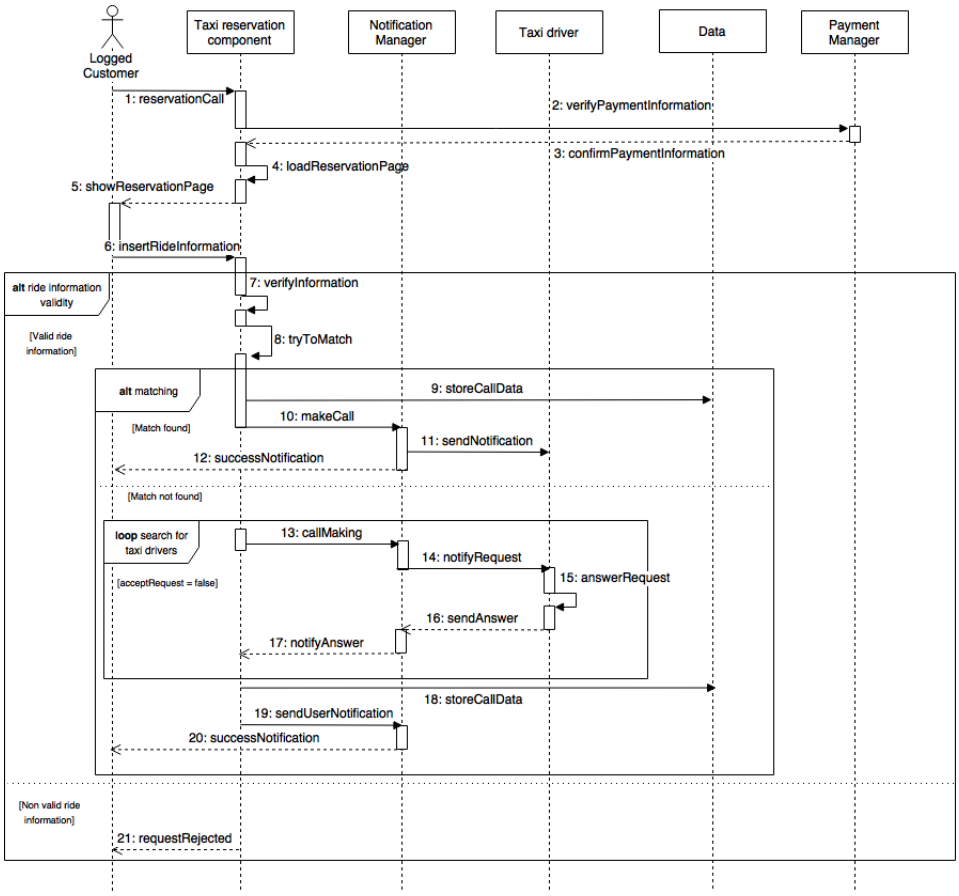

Figure 5.9: reservation call sequence diagram

6 Alloy modeling

6.1 Modeling

In this section is verified the consistency of the class diagram. In order to do this a formal model of the system had been realized, based both on the class diagram and in the previously done assumptions and considerations about constraints. The model has been realized using Alloy sintax, used to formally describe the domain of the application and its properties. Then, using the Alloy analizer, the consistency has been proved. Here there is the code of the model of the system.

```
module myTaxiService

//DATA TYPE

sig Text{}
sig Integer{}
sig Date{}
sig Time{}

//SIGNATURE

abstract sig User{
    name: lone Text,
    surname: lone Text,
    password: one Text
}

sig Customer extends User{
    email: one Text,
    dateOfBirth: one Date
}

sig TaxiDriver extends User{
    id: one Text,
    currentZone: one Integer,
    availableSeats: Integer
}

abstract sig Ride{
    id: one Text,
    carID: one Text,
    taxiDriver: one TaxiDriver,
    startingPosition: one Integer,
    startingTime: one Time,
    date: one Date,
    associatedCall: one Call
}

sig PrivateRide extends Ride{
    passenger: one Customer
}

sig SharedRide extends Ride{
    passengers: some Customer,
    steps: some Text
}

abstract sig Call{
    id: one Text,
    passenger: one Customer,
    queueStartingZone: one Integer,
    startingPosition: one Text,
    destinationPoint: one Text,
    date: one Date
}

sig ImmediateCall extends Call{}
```

```
sig ReservationCall extends Call{
    rideStartingTime: one Time
}

//ASSUMPTION

//Every user can not book more than one taxi in overlapping times.
assert noMoreThanOneTaxi{
    all r1,r2: Ride | (((r1.passenger = r2.passenger) or (r1.passenger in r2.passengers) or
                            (r1.passengers & r2.passengers != none)) and r1.date = r2.date
                            and
                            r1.startingTime = r2.startingTime) implies r1 = r2
}

/*
    Every customer's taxi request is processed by the system and forwarded to the first
    available taxi driver only in the queue
    of the same zone where the request comes from.
*/
fact taxiDriverCameFromTheCustomerZone{
    all c: Call, r: PrivateRide | (c.passenger = r.passenger) implies ( c.queueStartingZone
    = (r.taxiDriver).currentZone)
}

//Each cab belonging to the service is homologated with five seats, including the driver's
one.
fact maximumNumberOfAvailableSeats{
    #TaxiDriver.availableSeats >= 0
    #TaxiDriver.availableSeats <= 4
}

//Each registered user is related to one and only one e-mail address.
fact noDoubleUser{
    all c1, c2: Customer | c1.email = c2.email implies c1=c2
}

//Each customer is related to one and only one password.
fact onePasswordForEveryCustomer{
    all c1, c2: Customer | c1.password = c2.password implies c1 = c2
}

//Each taxi driver is related to one and only one taxi driver's ID.
fact noDoubleTaxiDriver{
    all t1,t2: TaxiDriver | t1.id = t2.id implies t1 = t2
}

//Each taxi driver related to one and only one password.
fact onePasswordForEveryTaxiDriver{
    all t1, t2: TaxiDriver | t1.password = t2.password implies t1 = t2
}

//Each call concerns one and only one ride.
fact noGhostCall{
    all r: Ride | # r.associatedCall <=1
}
```

```
//Each call concerns one and only one customer.
fact noGhostCustomer{
    all r: Ride | # r.passenger <=1
}

//The system can't store the same ride in his memory twice.
fact noDoubleRide{
    no disj r1, r2 : Ride | r1.id = r2.id
}

//The system can't store the same call in his memory twice.
fact noDoubleCall{
    no disj c1, c2 : Call | c1.id = c2.id
}

//A call and its associated ride must have the same date (the time can be different, e.g.
for shared rides)
fact sameDateForCallAndAssociatedRide{
    all r: Ride, c: Call | r.associatedCall = c implies r.date = c.date
}

//In a private ride every car can carry one and only one customer
fact oneCarForEveryPrivateRideCustomer{
    all r: PrivateRide | #r.passenger = 1
}

pred show{}
run show
```

The result found after running the code above is in the following picture:

```
Executing "Run show"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=4 Symmetry=20
  2690 vars. 255 primary vars. 4373 clauses. 479ms.
  Instance found. Predicate is consistent. 143ms.
```

This result confirm the validity of the model. While writing the code, particular attention has been paid in checking all the possible assertion we thought.

6.2 Generated worlds

Below there are some generated worlds useful to better analyse the model and verify its correctness. In this steps some entities have been simplified in order to make the pictures more watchable.

6.2.1 World 1

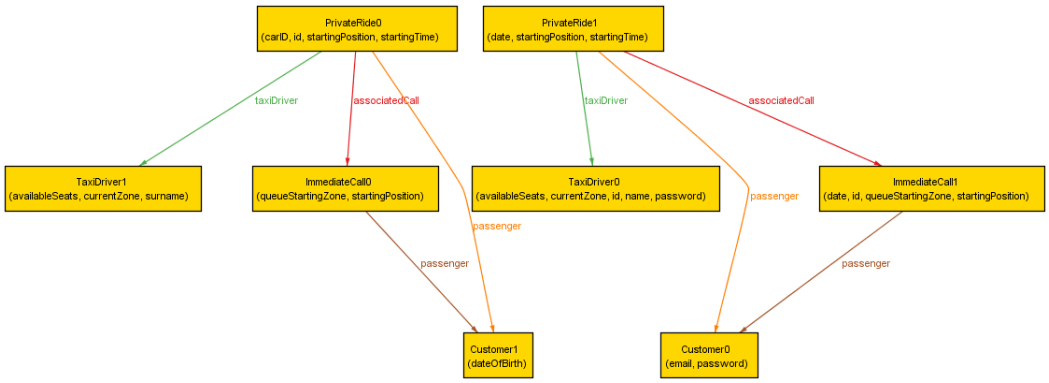

Figure 6.1: generated world 1

In this world is put in evidence the basic functionality of the system, and in particular the user's request for an immediate call, without sharing the ride. In this world there are:

- Two customers, each one with his own account (it was actually stated that each account is owned by one and only one registered user);
- Two immediate calls: ImmediateCall0 has been made by Customer1 and ImmediateCall1 has been made by Customer0;

- Two private rides: PrivateRide0 has been generated by ImmediateCall0 after the request of Customer1, PrivateRide1 has been generated by ImmediateCall1 after the request of Customer0;
- Two taxi drivers: TaxiDriver0 is assigned to PrivateRide1, TaxiDriver1 is assigned to PrivateRide0;

Both Customer0 and Customer1 have turned off the sharing mode.

6.2.2 World 2

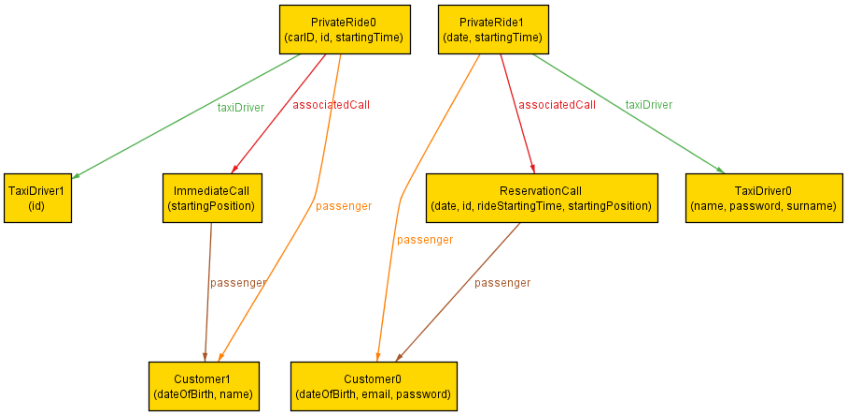

Figure 6.2: generated world 2

In this world has been put in evidence the second functionality of the system, that provides the reservation of a taxi. In this world there are:

- Two customers, each one with his own account (it was actually stated that each account is owned by one and only one registered user);
- One immediate call: ImmediateCall has been made by Customer1;
- One reservation call: ReservationCall has been made by Customer0;
- Two private rides: PrivateRide0 has been generated by ImmediateCall after the request of Customer 1, PrivateRide1 has been generated by Reservationcall after the request of Customer 0;
- Two taxi drivers: TaxiDriver0 is assigned to PrivateRide1, TaxiDriver1 is assigned to PrivateRide0;

Both Customer0 and Customer1 have turned off the sharing mode.

6.2.3 World 3

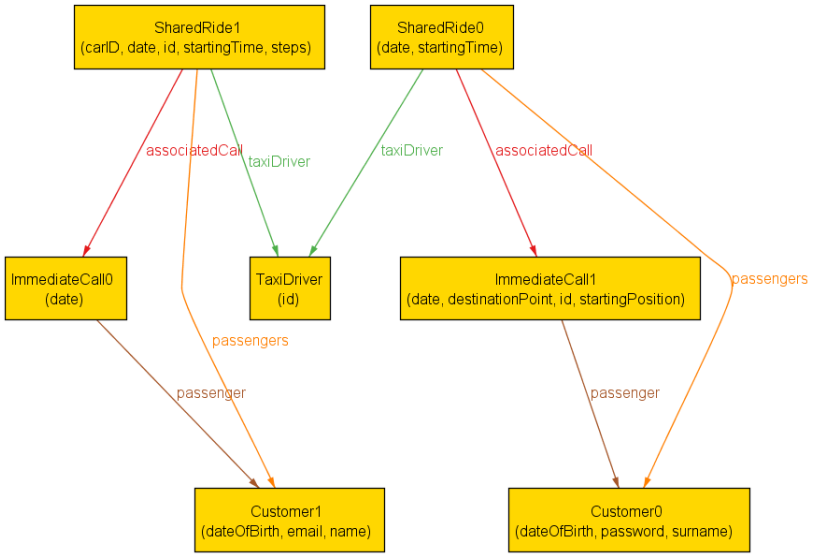

Figure 6.3: generated world 3

In this world has been put in evidence the third functionality of the system, that provides the taxi sharing. In this world there are:

- Two customers, each one with his own account (it was actually stated that each account is owned by one and only one registered user);
- Two immediate calls: ImmediateCall0 has been made by Customer1 and ImmediateCall1 has been made by Customer0;
- Two shared rides: SharedRide0 has been generated by ImmediateCall1 after the request of Customer 0, SharedRide1 has been generated by ImmediateCall0 after the request of Customer 1;
- One taxi driver: TaxiDriver is assigned to both the SharedRide0 and SharedRide1;

Both Customer0 and Customer1 have turned on the sharing mode.

7 Used tools

The used tools to create this RASD document are:

- *LyX* to generate the pdf document;

- *Microsoft Visio* to generate the class diagram;
- *draw.io* to generate the Sequence and the Use Case diagrams;
- *Alloy Analyzer 4.2* to prove the consistency of the model and to format the worlds generated;

8 Version history

- 6th November, 2015: version 1;
- 4th December, 2015: version 1.1;
- 7th December, 2015: version 1.2;
- 20th January, 2016: version 1.3;
- 1st February, 2016: version 1.4;