Summary

**1. INTRODUCTION ............................................................................................3**

1.1 DESCRIPTION OF THE GIVEN PROBLEM .........................................................................5

1.2 GOALS ................................................................................ ..............................................5

1.3 DOMAIN PROPERTIES ................................................................................................... 6

1.4 GLOSSARY ....................................................................................................................... 6

1.5 ASSUMPTIONS ................................................................................................................. 8

1.6 PROPOSED SYSTEM ........................................................................................................ 9

1.7 IDENTIFYING STAKEHOLDERS ..................................................................................... 9

**2. ACTORS IDENTIFYING ............................................................................. 11**

**3. REQUIREMENTS ........................................................................................ 11**

3.1 FUNCTIONAL REQUIREMENTS ................................................................................... 12

3.2 NON FUNCTIONAL REQUIREMENTS .......................................................................... 13

*3.2.1 User Interface ......................................................................................................... 13*

*3.2.2 Documentation ...................................................................................................... 16*

*3.2.3 Architectural considerations ........................................................................... 16*

**4. SCENARIOS IDENTIFYING ....................................................................... 18**

**5. UML MODELS .............................................................................................. 21**

5.1 USE CASE DIAGRAM .................................................................................................... 21

5.2 USE CASES DESCRIPTION ........................................................................................... 23

*5.2.1 Help Requests Managing ................................................................................... 24*

*5.2.2 Messaging ................................................................... .......................................... 29*

*5.2.3 Profile Managing ................................................................................................... 31*

*5.2.4 User Browsing and Friendship Managing .................................................. 36*

5.3 CLASS DIAGRAM .......................................................................................................... 40

5.4 SEQUENCE DIAGRAMS ................................................................................................ 41

*5.4.1 Log in ......................................................................................................................... 41*

*5.4.2 Search for help requests ..................................................................................... 42*

*5.4.3 Post a help request ................................................................................................ 43*

*5.4.4 Reply to a help request ....................................................................................... 44*

*5.4.5 Choose the best answer to a help request .................................................. 45*

*5.4.6 Provide a feedback ................................................................................................ 46*

5.5 STATE CHART DIAGRAMS ........................................................................................... 47

*5.5.1 Help request Class .................................................................................................. 47*

*5.5.2 Friendship Request Class ................................................................................... 48*

6. ALLOY MODELLING .................................................................... 42

6.1 ALLOY CODE ....................................................................................................... 42

6.2 ALLOY WORLDS .................................................................................................. 47

*5.5.1 General World* ................................................................................................ 47

*5.5.1 Invitation Properties* ....................................................................................... 48

*5.5.1 Notification Properties* ..................................................................................... 49

**8. USED TOOLS ....................................................................................... 57**

**1. Introduction**

**1.1 Description of the given problem**

The aim of this project is to improve the management of the taxi service in a large city, mostly by simplifying the access of passengers to the service and by simplifying the access of passengers to the service and by guaranteeing a fair management of taxi queues.

To further ensure the latter, passengers can also make a reservation by making a request at least 2 hours earlier and by specifying the origin and destination of the ride. To adapt to all situations, the system allows passengers to make requests either through a web application or a mobile app.

The requests will be passed on to the taxi drivers by the system and they will answer through an app.

The taxi drivers will be asked whether they want to accept a call only if they have previously notified the system about their availability.

**1.2 Goals**

We thought about the possible behaviors of those who will interact with the system and what myTaxiService should provide them, so we planned to give our system these features:

 Users should be able to:

• Sign up into the system;

• Log into the system;

• Request a taxi;

• Cancel a request;

• Reserve a taxi;

• View the reservations;

• Modify a reservation.

• Cancel a reservation.

 Taxi drivers should be able to:

• Inform the system about their availability;

• Accept a call from the system;

• Refuse a call from the system;

• Inform the system they can no longer take care of a call they’d previously accepted.

**1.3 Domain Properties**

We suppose that these conditions hold in the analyzed world:

• GPS localization is used to know where the taxis are;

• Taxi drivers must be inside the city;

• Taxi drivers should be able to use the necessary technology to communicate with the system.

• Users should be able to use the necessary technology to communicate with the system.

**1.4 Glossary**

Here are the definitions of some words that will be used in our documents.

• USER: by user we mean a person who is already registered in the system. This means they have a profile, through which they can make requests and reservations, and manage them.

• GUEST: a guest is a person who hasn’t signed up yet.

Unlike registered users, they cannot make requests and reservations, but they can still access the app, either to sign up or to read about the available features they will be able to access once they register.

• TAXI DRIVER: by taxi driver we mean a person who will interact with the system with the purpose of serving users. Taxi drivers can inform the system about their availability and they can confirm that they are going to take care of a call that the system is trying to assign to them.

• QUEUE: by queue we mean a queue of available taxis in a zone.

• ZONE: by zone we mean an area of the city. Each zone is approximately 2km and is associated with a single queue of taxis.

• REQUEST: a request can be made by a registered user when they want a taxi right away. Once a user has made a request, a taxi will come pick them up shortly after, at the location specified by the user.

• RESERVATION: a reservation can be made by a registered user when they want to book a taxi at a certain time. If a user wants to make a reservation, they will have to provide additional details, namely, the origin and destination of the ride, along with the meeting time. Reservations must be made at least two hours before the desired meeting time.

**1.5 Assumptions**

There are a few points that are not very clear in the specification document, so we will have to assume some facts.

We assume that:

• The user can cancel requests and reservations.

• The user can modify reservations. If a user wants to modify a reservation, they must notify the system at least 10 minutes before the meeting time, otherwise the reservation will be cancelled.

• If a taxi driver informs the system that he is available and then they decide to move from a zone to another, then the system will remove its identifier from the queue of the previous zone and move the taxi in the last position of the queue of the new zone.

• Once a call has been forwarded to an available taxi driver, a timer starts. If the taxi driver doesn’t confirm by the time the timer runs out, the system assumes the taxi driver is no longer available, it moves them to the last position in the queue and it forwards the request to the next available driver in the queue.

• If a taxi driver accepts a call, the system will automatically set the taxi driver as unavailable and therefore remove them from the queue.

• There always is at least one available taxi driver who will accept a request.

• If a taxi driver accepts a call and has a mishap before meeting up with the client, they will notify the system they can no longer take care of that call.

**1.6 Proposed system**

Our system will follow a client-server pattern. The server will generate dynamic pages for the clients, access data and directly manage queues and requests, so as to always find an optimal solution, by minimising the waiting time of passengers.

There will be two distinct versions of the application on the client side.

One version will only be available to taxi drivers and it will strictly be a mobile application. It is necessary that taxi drivers can use it to interact with the system, to let it know whether they’re available or not, and to let it know if they intend to take care of a call that has been forwarded to them.

The other version will only be available to passengers, registered users, to be precise. This version will be both accessible through a mobile application and a web application.

**1.7 Identifying stakeholders**

We can identify three main stakeholders. First and foremost, the government, who certainly have great interests in the success of this project, since they assigned us this task. In fact, by granting an improved service, the project will improve an aspect of the life in the city, that is transportation. Passengers are those who will actively enjoy this service, therefore we can assume they also belong to the category of the stakeholders without doubt. Lastly, taxi drivers, who will interact with the system because they will actually work for it and, as a consequence, it’s beneficial to them that the system, that is the outcome of this project, turns out to be as smooth as possible.

**2. Actors identifying**

The actors of our system are basically three:

 Guest: a guest is someone who hasn’t signed up yet. A guest can either sign up or read about the possible features they will be able to enjoy once they sign up.

User: a user is someone who has already signed up and therefore can use the service to book a taxi, either through a request or through a reservation.

Taxi Driver: a taxi driver is an employee of our service who gets paid for driving customers to their desired destinations upon request. Taxi drivers actively interact with the system by exchanging information about their availability.

**3. Requirements**

We’ve determined the following requirements according to the Jackson and Zave analysis, so assuming that the domain properties, which are described above, hold for our purposes, we’ve written the requirements in order to satisfy goals.

 Users are be able to:

• Sign up into the system

− The system has to provide a sign up functionality.

• Log into the system

− The system has to provide a log in functionality.

• Request a taxi

− The system has to provide a function to allow a user to make a request inside the city.

− The system forwards a request that comes from a certain zone to the first taxi queuing in that zone.

• Cancel a request

− The system has to provide a function to allow a user to cancel a request.

− The system has to provide a function to show the taxi driver that the call has been cancelled.

− The system has to provide a function to ask the taxi driver whether they’re available after they’ve been notified their call has been cancelled.

• Reserve a taxi

− The system has to provide a function to allow a user to make a reservation inside the city.

− The system forwards a reservation to the first taxi queuing in the zone of the origin of the ride, 10 minutes before the meeting time with the user.

• View the reservations

− The system has to provide a function that allows the user to see their reservations.

• Modify a reservation

− The system has to provide a function that allows the user to modify a reservation.

− The system rejects the modification, if it’s done less than two hours before the ride.

• Cancel a reservation

− The system has to provide a function to allow a user to cancel a reservation.

− The system has to provide a function to show the taxi driver that the call has been cancelled, in case the taxi driver has already been mobilized.

− The system has to provide a function to ask the taxi driver whether they’re available after they’ve been notified their call has been cancelled.

 Taxi drivers can:

• Inform the system about their availability;

− The system has to provide a function to store the identifier of the available taxi in the queue of taxis in the corresponding zone.

• Accept a call from the system;

− The system has to remove the taxi from the queue

− The system has to set the taxi driver as unavailable

− The system has to estimate the waiting time using the information provided by the GPS

− The system has to inform the user about the code of the incoming taxi and the waiting time.

• Refuse a call from the system;

− The system forwards the request to the second taxi driver in the queue.

− The system moves the first taxi in the last position in the queue.

• Inform the system they can no longer take care of a call they’d previously accepted;

− The system forwards the request to the second taxi in the queue.

− The system automatically sets the taxi driver as unavailable.

− The system removes the taxi from the queue.

**3.1 Functional requirements**

Now that we have defined the main feature of mytaxiService, we can find some functional requirement concerning each defined actor:

 Guest: he can

• Sign up.

 User: he can

• Log in;

• Modify his profile information;

• Request a taxi;

• Cancel a request;

• Reserve a taxi;

• View the reservations;

• Modify a reservation.

• Cancel a reservation.

 Taxi Driver: he can

• Inform the system about their availability;

• Accept a call from the system;

• Refuse a call from the system;

• Inform the system they can no longer take care of a call they’d previously accepted.

**3.2 non-functional requirements**

**3.2.1 user interface**

**3.2.2 documentation**

**4. Scenarios identifying**

Basic scenario

1-Bob needs to go to work but his car broke down the previous evening, therefore he decides to go by taxi. After having breakfast, he gets on his computer and logs in on his myTaxiService account to request a taxi, by filling a form where he opts for a simple request. Since he's making a simple request he only has to fill in a form saying the address where he lives. The request is sent to the system, which forwards it to the first available taxi driver in the zone where Bob lives. The taxi driver accepts the call and, as a consequence, Bob receives a confirmation of the request which contains the code of the incoming taxi and the waiting time. Some minutes later, a taxi shows up, Bob notices the code of the taxi is the one he's waiting for, so gets on it and he tells the driver the address of his workplace. The taxi driver takes Bob to work on time and eventually notifies the system that he is available again.

scenario with reservation

2-Alice is on a business trip, she will arrive in the city by train and she will have to stay at a hotel overnight. The train ride will be 4 hours long and she will arrive at the station late at night. Because she knows she will be tired, she wants to make sure she will find a taxi waiting for her right outside the station. Therefore, as soon as she gets on the train, she uses her phone to log in on her myTaxiService account to make a reservation. She fills in all the necessary fields, setting the station as the origin of the ride, the hotel as the destination, and specifying she wants the meeting time to be shortly after the train gets to the station. Around 4 hours later, just before getting off the train, the system forwards the reservation to the first available taxi driver near the station and Alice gets a notification which tells her the code of the taxi that will pick her up. She meets the taxi driver just outside the station and he drives her to the hotel. Right after this, the taxi driver is ready to accept requests again, therefore he notifies the system he’s available.

3-On a weekday Carl and his girlfriend are both stressed and tired from work and they agree on spending some relaxing time together during the weekend, therefore Carl suggests to go for a picnic on the beach this Saturday. Carl doesn't want to go to the beach by car because he knows finding a car park will stress him even more, so he goes on myTaxiService and makes a reservation for a taxi on Saturday morning. Friday night, right before going to sleep, Carl watches the weather forecast on TV, which warns of an incoming heavy storm that is going to make the weather rainy all through the weekend. Knowing that he will have to cancel his plans, Carl goes on myTaxiService application and cancels the reservation he had made a few days earlier.

4-John needs to go to the mall by taxi, so he requests a taxi on myTaxiService. The system checks the queue associated to the taxi zone where John lives and finds the first available cab in the queue, which is number 358. Mark, the taxi driver driving cab number 358, accepts the call and John receives a confirmation from the system which says a cab will be there to pick him up in 10 minutes. A couple of minutes later, Mark hears a noise coming from his cab and he notices that he got a flat tyre. Because of this, he decides he should give up the call and let the system assign the call to another cab, so he notifies the system of what's happened. The system assigns the call to the next available taxi driver and moves Mark to the last position in the queue. After doing this, the system also notifies John that there will be a delay of about 10 minutes; moreover John receives the code of the new cab that is coming to pick him up. Eventually, the new cab arrives and takes John to the mall.

5- chiama il taxi, ma passa un suo amico a prenderlo e quindi cancella la richiesta.

6- un tassista che non accetta la chiamata per un motivo bo

7- uno prova a mettere una via non all interno della città, il sistema lo blocca e quindi il tipo capisce di aver scaricato l app sbagliata ehehe...ed esce dal sistema

8- uno fa una prenotazione, si accorge di aver sbagliato l indirizzo di destinazione e gli viene permessa la modifica perché entro le due ore dalla corsa.

9- prenota e cancella quando ormai il tassista è già stato chiamato.

**5. UML Models**

**5.1 Use Case Diagram**

We can derive some use cases from the scenarios identified in the previous paragraph:

• Sign up;

• Log in;

• Request a taxi;

• Cancel a request;

• Reserve a taxi;

• Modify a reservation;

• Cancel a reservation;

**5.2 Use Case Description**

We describe in a detailed way the use cases that we derived from the scenarios.

We try to define them all.

It is important to understand that all the references to “pages”, “buttons” or “input forms” are only hypothesis to make the situation as clear as possible ad to help the reader to draw a visual picture in his mind of what we plan to do, but the real structures will be well defined in the Design Document.

We refine the use case “Sign up”:

|  |  |
| --- | --- |
| Name | Sign up |
| Actors | Guest |
| Entry Conditions | The guest isn’t registered to the myTaxiService. |
| Flow of events | • The guest enters the website or the mobile app.  • The guest clicks/touch the “SIGN UP” button.  • The guest fills in the form where he has to write:  -Name  -Surname  -Email  -Password  -Telephone number  • The guest clicks/touch the “DONE” button.  • The system shows him his personal page. |
| Exit conditions | Registration successfully done. |
| Exceptions | An exception can be caused if the format of e-mail or telephone number is not correct or the guest inserts already exists e-mail/telephone number or if some field that are not optional aren’t filled. |

We refine the use case “Log in”:

|  |  |
| --- | --- |
| Name | Log in |
| Actors | User |
| Entry Conditions | User has successfully signed up to the system. |
| Flow of events | • The user enters the web site/mobile app.  • The user fills in the text fields in the home page with e-mail and password.  • The user clicks/touch the “LOG IN” button. |
| Exit conditions | The system shows the user his personal page. |
| Exceptions | The password and/or e-mail inserted by the user are wrong. The System shows an error message to the user. |

We refine the use case “Request a taxi”:

|  |  |
| --- | --- |
| Name | Request a taxi |
| Actors | User, Taxi driver |
| Entry Conditions | User click/touch the “TAXI REQUEST” button. |
| Flow of events | • The user enters the web site/mobile app.  • The user fills in the text fields in the page the address of the origin of the ride.  • The user clicks/touch the “CONFIRM REQUEST” button.  • The system forwards the request to the first taxi queuing in the user’s zone.  • The system forwards the request to the first taxi queuing in the user’s zone.  • The taxi driver sceglie se accettare o meno la chiamata.  • Se il taxi driver conferma, the system will send a confirmation to the passenger,  altrimenti the system will forward the request to the second in the queue and will, at the same time, move the first taxi in the last position in the queue. |
| Exit conditions | The system shows the user the taxi code and the waiting time. |
| Exceptions | L’indirizzo inserito non è all’interno della città oppure non ci son taxi disponibili al momento. |
| Alternative flow | il tassista informa il sistema che non è piu disponibile alla chiamata precedentemente accettata, the system will forward the request to the second in the queue and will, at the same time, remove the taxi driver dalla coda e impostarlo automaticamente come non disponibile. |

We refine the use case “Cancel a request”:

|  |  |
| --- | --- |
| Name | Log in |
| Actors | User |
| Entry Conditions | User has successfully signed up to the system |
| Flow of events | • The user enters the web site.  • The user fills in the text fields in the home page with username and password.  • The user clicks on the “LOG IN” button. |
| Exit conditions | The system shows the user his personal page. |
| Exceptions | The password and/or username inserted by the user are wrong. The System shows an error message to the user. |

We refine the use case “Reserve a taxi”:

|  |  |
| --- | --- |
| Name | Log in |
| Actors | User |
| Entry Conditions | User has successfully signed up to the system |
| Flow of events | • The user enters the web site.  • The user fills in the text fields in the home page with username and password.  • The user clicks on the “LOG IN” button. |
| Exit conditions | The system shows the user his personal page. |
| Exceptions | The password and/or username inserted by the user are wrong. The System shows an error message to the user. |

We refine the use case “Modify a reservation.”:

|  |  |
| --- | --- |
| Name | Log in |
| Actors | User |
| Entry Conditions | User has successfully signed up to the system |
| Flow of events | • The user enters the web site.  • The user fills in the text fields in the home page with username and password.  • The user clicks on the “LOG IN” button. |
| Exit conditions | The system shows the user his personal page. |
| Exceptions | The password and/or username inserted by the user are wrong. The System shows an error message to the user. |

We refine the use case “Cancel a reservation.”:

|  |  |
| --- | --- |
| Name | Log in |
| Actors | User |
| Entry Conditions | User has successfully signed up to the system |
| Flow of events | • The user enters the web site.  • The user fills in the text fields in the home page with username and password.  • The user clicks on the “LOG IN” button. |
| Exit conditions | The system shows the user his personal page. |
| Exceptions | The password and/or username inserted by the user are wrong. The System shows an error message to the user. |

**5.3 Class Diagram**

Now that we have refined every use case we can draw a class diagram of our system:

**5.4 Sequence Diagrams**