myTaxiService
-
Requirements Analysis and Specification Document

Davide Cremona, Simone Deola

November 2, 2015

Contents

1 Introduction **4**
1.1 Purpose 4
1.2 Actual System 4
1.3 Scope 4
1.4 Actors 5
1.5 Goals 5
1.6 Definitions, Acronyms, Abbreviations 6
1.6.1 Definitions 6
1.6.2 Acronyms 6
1.6.3 Abbreviations 6
1.7 Reference documents 6
1.8 Document overview. 6

2 Overall Description **7**
2.1 Product perspective 7
2.2 User Characteristics 7
2.3 Constrains 7
2.3.1 Regulatory Policies 7
2.3.2 Hardware Limitations 7
2.3.3 Software Limitations 8
2.3.4 Parallel Operations 8
2.3.5 Documents Related 8
2.4 Assumptions 8
2.5 Future possible Implementations 9

3 Specific Requirements **10**
3.1 External Interface Requirements 10
3.1.1 User Interfaces 10
3.1.1.1 Login 10
3.1.1.2 Customer registration 12
3.1.1.3 Taxi Drivers registration 14
3.1.1.4 Customer home page 16
3.1.1.5 Reservation page 18
3.1.1.6 My reservation Page 20

3.1.1.7 Customer notification pop-up 22
3.1.1.8 User page 24
3.1.1.9 Taxi Driver home page 26
3.1.1.10 Taxi Driver notification 28
3.1.1.11 Taxi Driver end ride page 30
3.1.2 Hardware Interfaces 32
3.1.3 Software Interfaces 32
3.1.4 Interfaces to Others Application 32
3.1.5 Communication Interfaces 33
3.2 Functional Requirements 33
3.2.1 Functional Requirements for Guest Users 34
3.2.1.1 [G.1 - Allow guest users to become a customer creating a myTaxiService Account] 34
3.2.1.2 [G.2 - Allow guests user to become a customer using his Facebook Account] 34
3.2.1.3 [G.3 Allow guest users to become a taxi driver] . . 35
3.2.2 Functional Requirements for Registered Users 35
3.2.2.1 [G.4 Allow registered users to log in with myTaxiService account.] 35
3.2.2.2 [G.5 Allow a Registered User to view or modify his username and email.] 36
3.2.2.3 [G.6 Allow a Registered User to retrieve his password if he doesn't remember it.] 37
3.2.2.4 [G.7 Allow a Registered Users to signal another one if he has made a bad use of the system.] 37
3.2.3 Functional Requirements for Customers 38
3.2.3.1 [G.8 Allow customers to log in with Facebook account.] 38
3.2.3.2 [G.9 Allow customers to require a taxi.] 38
3.2.3.3 [G.10 Allow customers to reserve a ride.] 38
3.2.3.4 [G.11 Allow customers to delete a previous reservations.] 39
3.2.4 Functional Requirements for Taxi Drivers 39
3.2.4.1 [G.12 Allow Taxi Drivers to accept or decline a ride request.] 40
3.2.4.2 [G.13 Allow Taxi Drivers to notify their availability.] 40
3.3 Scenarios 40
3.3.1 S.1 Registration as a Customer creating a myTaxiService account 40
3.3.2 S.2 Registration as a Customer using a Facebook Account . 41
3.3.3 S.3 Registration as a Taxi Driver creating a new Taxi Driver myTaxiService account. 41
3.3.4 S.4 Login with a myTaxiService account. 41
3.4 UML Models 42
3.4.1 Use-Case Diagrams 42
3.4.2 Class Diagrams 42

3.4.2.1 Registration through the creation of a new myTaxiService account. 42
3.4.2.2 Registration using a Facebook account. 43
3.4.2.3 Taxi Driver Registration 45
3.4.2.4 Registered User Login 46
3.4.3 State Machine Diagrams 47
3.5 Non Functional Requirements 47
3.5.1 Performance Requirements 47
3.5.2 Design Constraints 47
3.5.3 Software System Attributes 47
3.5.3.1 Availability 47
3.5.3.2 Maintainability 47
3.5.3.3 Portability 47
3.5.4 Security 47
3.5.4.1 External Interface Side 47
3.5.4.2 Application Side 47
3.5.4.3 Server Side 47

4 Appendix **48**

Chapter 1

Introduction

1.1 Purpose

This document is the R.A.S.D. (Requirement Analysis and Specification Document). The purpose of this document is the description of the "myTaxiService" system. At first, it will provide functional and non-functional requirements, a complete overview of the constraints of the system and its limits. Then it will explain in detail the dynamics of the system using real-life use cases. Finally this document will provide a base for the developers that concretely have to implement the system.

1.2 Actual System

The functionality that the new system will provide is now not supported. So the entire system must be developed without using or modifying existing system.

1.3 Scope

The objective of myTaxiService is to provide an interface between customers and taxi drivers to optimize their interaction and provide a fair management of taxi queues. The users, once registered through the mobile application or the web application, can request a taxi for their travel or reserve one, specifying the origin and the destination. The reservation can be done at least two hour before the ride; if the reservation can take place, the system will allocate a taxi 10 minutes before the meeting time. On the other side, taxi drivers can inform the system that they are waiting for a client and accept or decline a ride request. If the request has been accepted, a notification will be sent to the requesting customer with the identification number of the incoming taxi and the time he has to wait. Otherwise, if the request has been rejected it will be forwarded to the next taxi in the queue. The system has to optimize the management of customers requests giving the rides to the taxi with the highest priority that has to be evaluated in function of avaiability and the nearness of the taxi driver.

1.4 Actors

- **Guest User:** guest users are unlogged or unregistered users. They can visit the login page or the registration forms.
- **Registered User:** this kind of user identify either a Guest User or a Taxi Driver.
- **Customer:** this kind of user is the end-user of the service. He can perform request for taxis or reserve a ride. In his personal page he can view his requests and the system responses.
- **Taxi Driver:** this kind of user is composed by the actual taxi drivers that can only see customers requests that has been forwarded by the system. He can accept or decline these requests. Also, he's considered a special kind of user because one can register as a "Taxi Driver" only if he provide a valid Taxi licence.

1.5 Goals

- **[G.1]** Allow guest user to become a customer creating a myTaxiService Account.
- **[G.2]** Allow guest user to become a customer using his Facebook Account.
- **[G.3]** Allow guest user to become a taxi driver.
- **[G.4]** Allow registered user to log in with myTaxiService account.
- **[G.5]** Allow a Registered User to view or modify his username and email.
- **[G.6]** Allow a Registered User to retrieve his password if he doesn't remember it.
- **[G.7]** Allow a RegisteredUser to signal another one if he has made a bad use of the system.
- **[G.8]** Allow customer to log in with Facebook account.
- **[G.9]** Allow customers to require a taxi.
- **[G.10]** Allow customers to reserve a ride.
- **[G.11]** Allow customers to delete a previous reservations.
- **[G.12]** Allow taxi drivers to accept or decline a ride request.
- **[G.13]** Allow taxi drivers to notify their availability.

1.6 Definitions, Acronyms, Abbreviations

1.6.1 Definitions

1.6.2 Acronyms

- RASD: Requirement Analysis and Specification Documents.
- DD: Design Document.
- UML: Unified Modeling Language.
- OS: Operative System.
- API: Application Program Interface.
- GPS: Global Positioning System.
- HTTP: Hypertext Transfer Protocol.
- HTTPS: Secure Hypertext Transfer Protocol.

1.6.3 Abbreviations

- G.x is the x-Goal
- Req.x is the x-Functional Requirement
- Dom.x is the x-Domain Assumption

1.7 Reference documents

- (IEE830) IEEE Recommended Practice for Software Requirements Specifications

1.8 Document overview.

Until now, we have given a general explanation about the software functionalities and a brief description about this document. Now we will describe what the rest of this RASD contains.
In Section 2 we will focus more about system constraints and assumptions.
In Section 3 we will describe requirements, typical scenarios and use-cases. In this section there is also a collection of UML diagrams that describes in particular the functionalities of the system.
//TODO SECTION 4

Chapter 2

Overall Description

2.1 Product perspective

The system will be composed of a web application and a mobile application developed for the three major OS (Apple iOS, Android, Windows 10). The system will provide some API with the purpose of a future connection with another travel planning systems.

2.2 User Characteristics

The users that we suppose will use our system are of two types. the ones who want to find a taxi for a travel in the simplest way (customers). The others are taxi drivers that want to increment their productivity. The first ones must be able to access to a web browser or download and using a mobile application, the second ones also must have a taxi license.

2.3 Constrains

2.3.1 Regulatory Policies

myTaxiService has to meet regulatory policies about taxies in the countries where it will be used.

2.3.2 Hardware Limitations

The only hardware limitation that the myTaxiService mobile application has to meet will be the mobile phones characteristics. the rest of the system will be no affected by particular hardware limitations.

2.3.3 Software Limitations

myTaxiService mobile application has to be compatible with all major mobile operating systems (Android, Apple iOS, Windows 10). Also myTaxiService web application has to be compatible with all major browser (Chrome, Safari, Firefox, Microsoft Edge).

2.3.4 Parallel Operations

Our system must be able to perform parallel operations on the database to satisfy all the requests from multiple users.

2.3.5 Documents Related

- Requirements and Analysis Specification Document (RASD)
- Design Document (DD)

2.4 Assumptions

- Every taxi driver has equipped a smartphone during working hours.
- Every taxi driver has a unique taxi license.
- Every taxi has a GPS locator to send GPS information to the central server.
- Android, Apple iOS or Windows 10 is avaiable on the registered users smartphones.
- Every registered users can be connected to the Internet with a mobile device when outside.
- When a customer require a taxi, the GPS informations about his location are automatically sended to the central server.
- The reservation of a ride is made at least two hours before the ride.
- Deletion of a reservation is made at least two hours before the ride.
- Requests from customers are automatically notified to the first taxi driver in the zone queue.
- If a taxi driver declines a request he will be placed in the bottom of the zone queue.
- If a request is declined it will be forwarded to the next taxi driver in the zone queue.
- If a customer make a bad use of the taxi request system, he can be reported as a bad customer.

- If a taxi driver notifies his availability is because he is actually avaiable
- If a taxi driver notifies his availability is because he wants to be notified of customers that needs a ride.
- If a taxi driver accept a request, the requesting customer will be notified

2.5 Future possible Implementations

A possible future implementation can be a complex feedback system that permits to the customers to leave a comment about the taxi driver and vice versa. For example taxi drivers can be interested in knowing the punctuality or how is the behave of the customer that requests the ride.

Chapter 3

Specific Requirements

This chapter contains a detailed description of how the applications works and its features. It also gives a specification of the functional and quality requirements.

3.1 External Interface Requirements

This section gives a description of the various inputs and relative outputs of the system. It also gives a description of the hardware, software and communication interfaces that are necessary to make the system work. It will also provide a generic visualization of the user interface in the various user platforms.

3.1.1 User Interfaces

Here we describe in particular how the application should look like either for mobile and web application. To make an easier explanation of the aspect of the various screen of the application we are going to use Mockup.

3.1.1.1 Login

On this page the users can log in with username and password or with Facebook (only if as registered user is a Customer). If the user is not already registered can access to the registration pages (for Customer or for Taxi Drivers). Also if the users forgot the password from this page can go to the forgot password page.

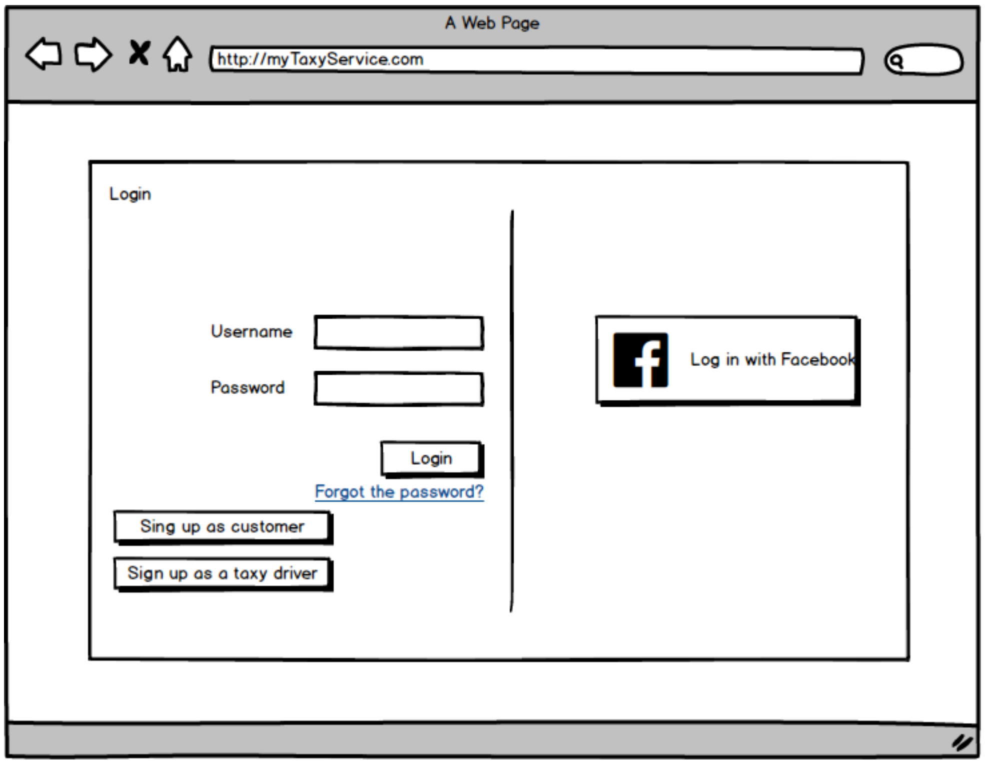


Figure 3.1: Login Page, web version

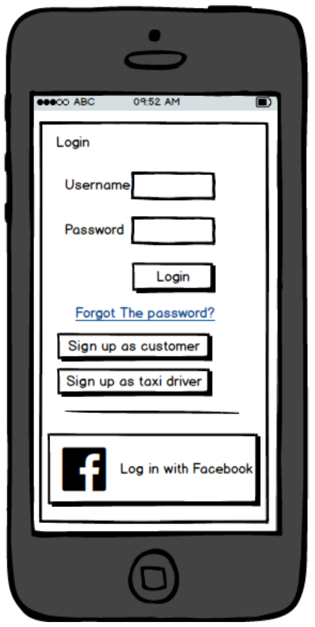


Figure 3.2: Login Page, mobile version

3.1.1.2 Customer registration

On this page the users can register itself. This page must provide two way of registration, registration with the standard form (e-mail, password and username) of with Facebook API.

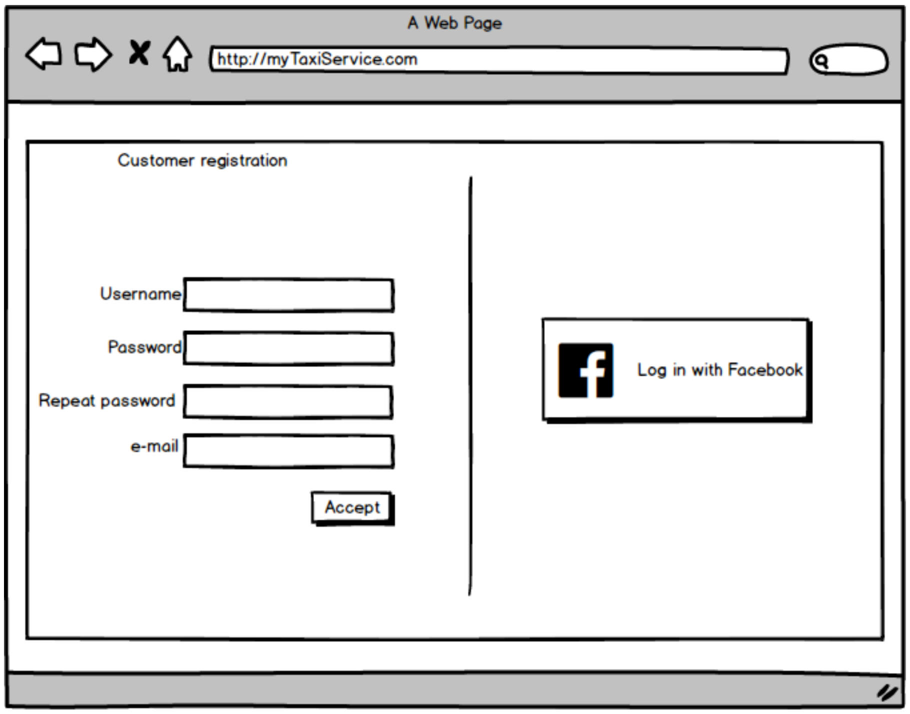


Figure 3.3: Customer registration Page, web version

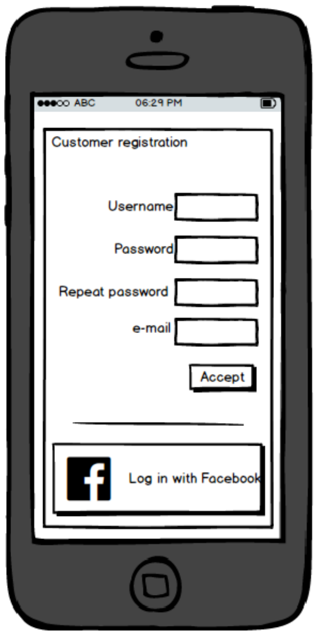


Figure 3.4: Customer registration Page, mobile version

3.1.1.3 Taxi Drivers registration

On this page the users can register itself as a taxi driver. Taxi driver have a special form to been registered because of the additional information the user must provide (taxi license number). Just because the additional information the registration can be done just with the standard form.

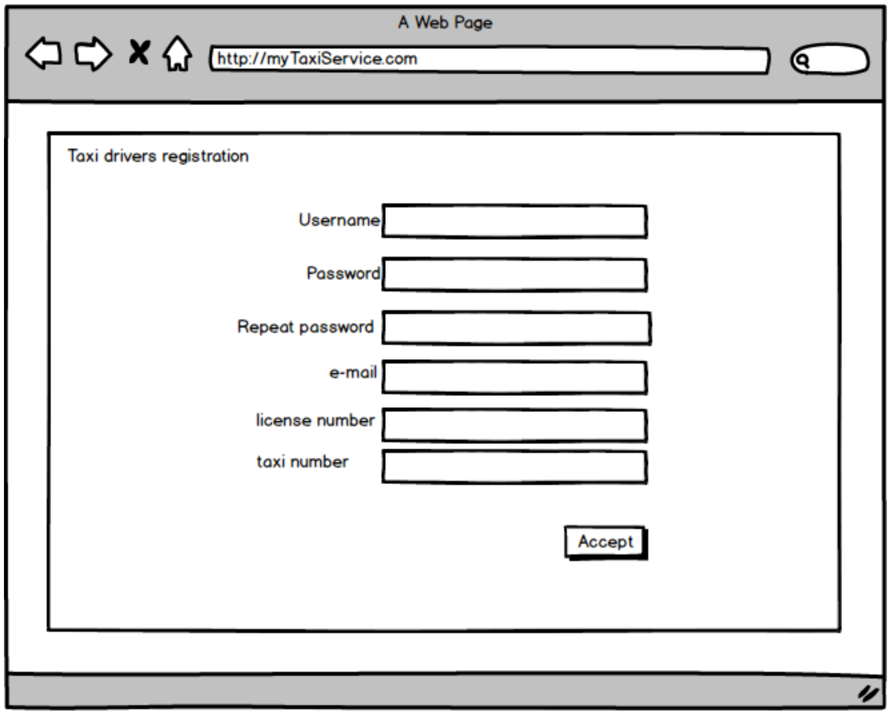


Figure 3.5: Taxi Driver registration Page, web version

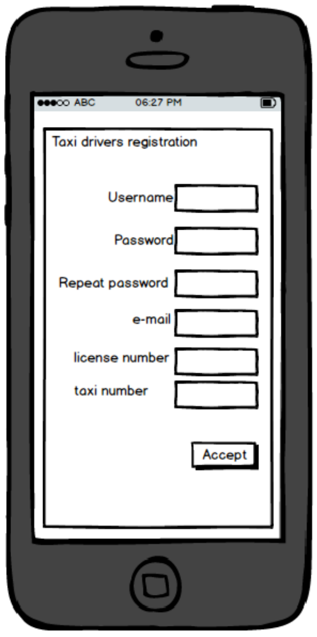


Figure 3.6: Taxi Driver registration Page, mobile version

3.1.1.4 Customer home page

On this page the Customer can see his position (information from his GPS) and perform all the main operation that he can perform. He can Request a taxi on the position showed, can go to the reservation page (in which can reserve a ride), can visit his personal page (in which can see all the information of his profile), can go to the information page (information about myTaxiService) or go to the 'my reservation page' (in which can manage all the previous reserved ride).

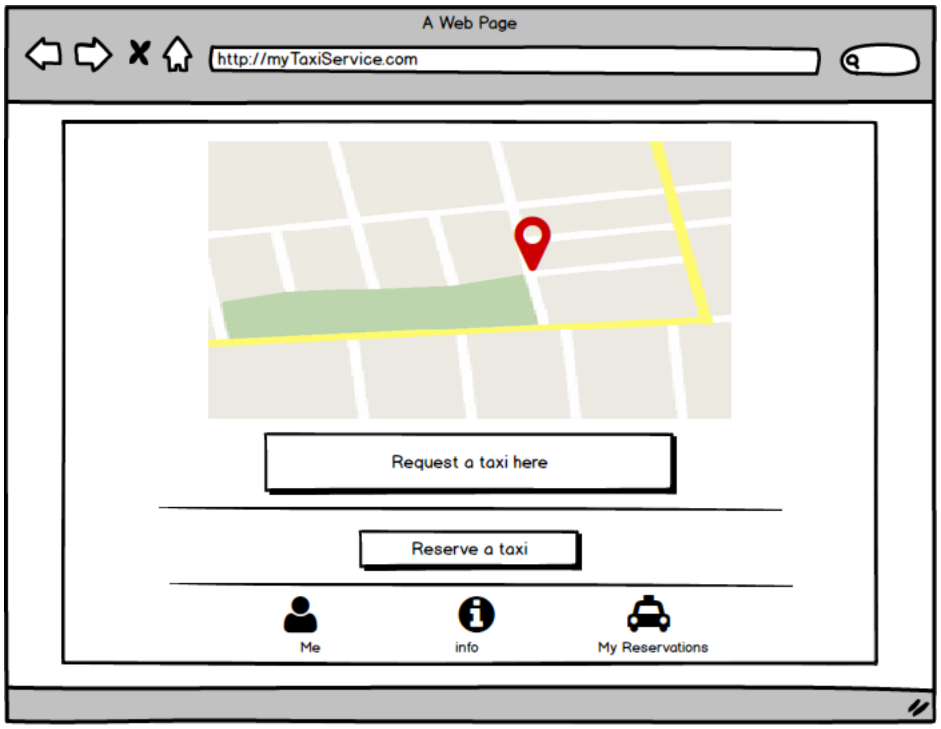


Figure 3.7: Customer home page, web version

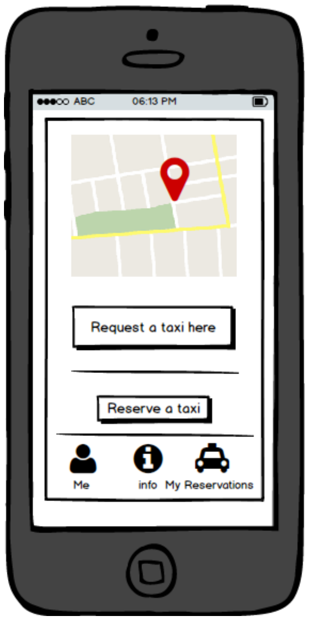


Figure 3.8: Customer home page, mobile version

3.1.1.5 Reservation page

On this page the Customer can reserve a taxi ride. To do this he must insert a Origin position, a Destination position, a Data and a Time. The reservation must be at least two hours after the current Time, so the system must avoid to reserve a previous ride.

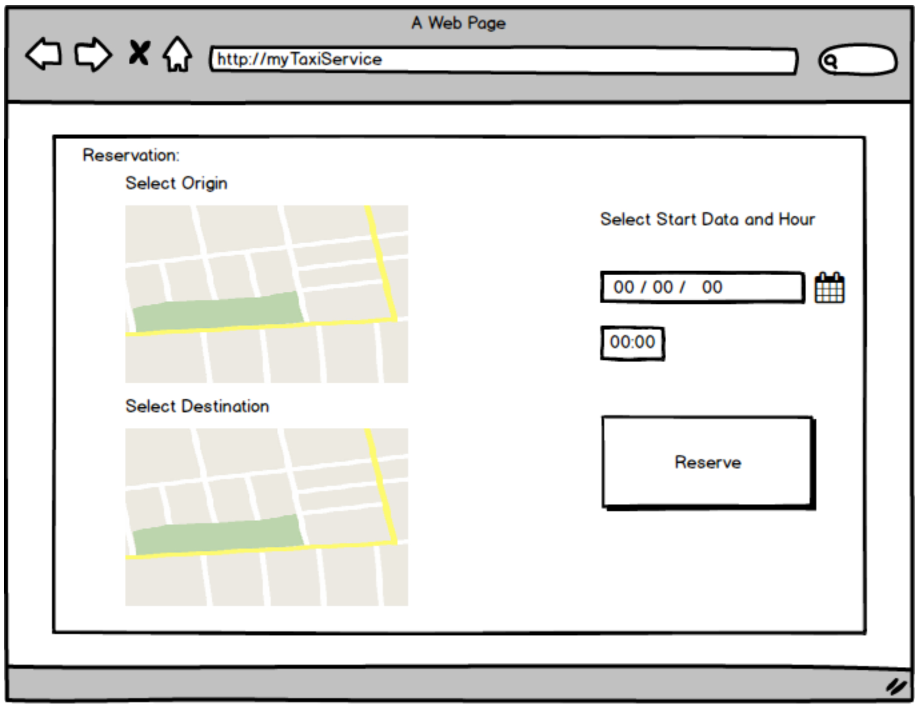


Figure 3.9: Reservation page, web version

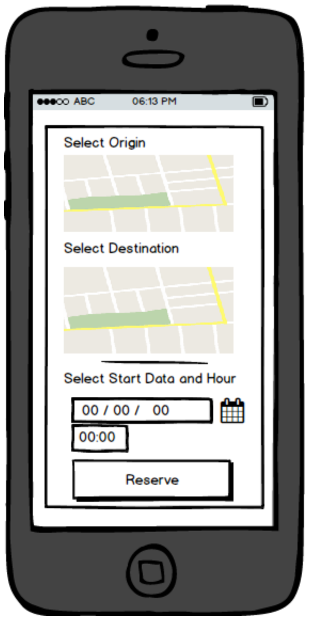


Figure 3.10: Reservation page, mobile version

3.1.1.6 My reservation Page

On this page Customer can see all the reservation he have already done and adding some new. For each previous reservation can see all the information and, if the Date is before the next two hours, can delete it.

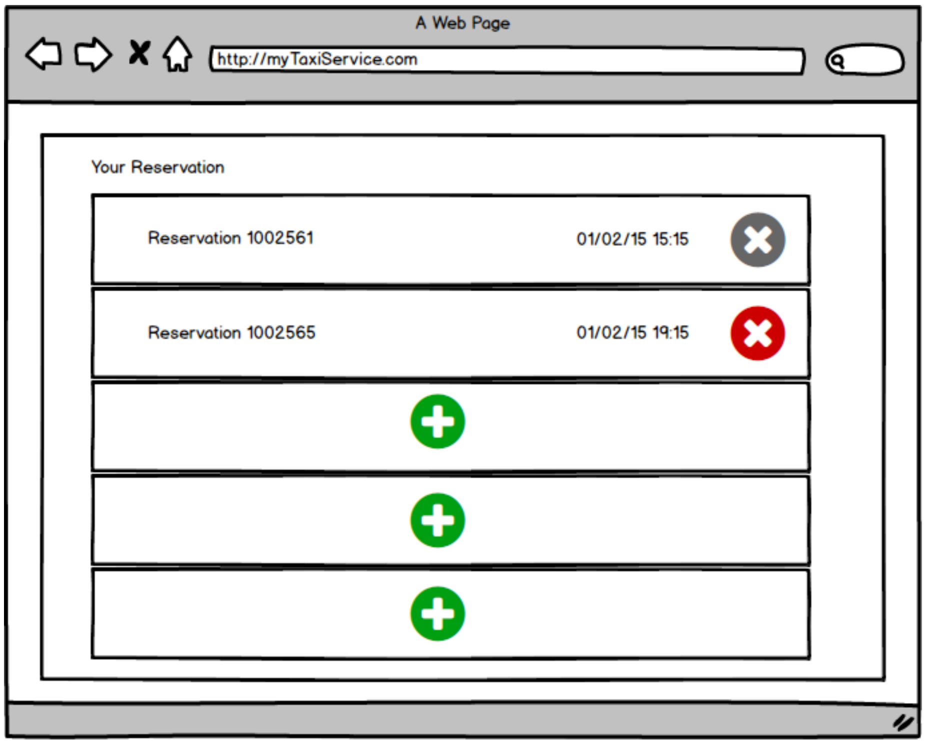


Figure 3.11: My reservation, web version

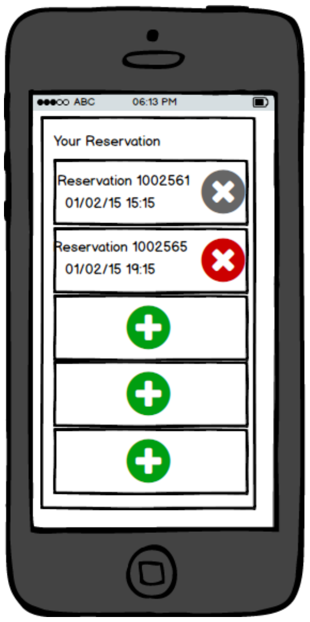


Figure 3.12: My reservation, mobile version

3.1.1.7 Customer notification pop-up

This pop-up is showed to the requesting Customer when his request has been handled. On this pop-up the system show also the number of the incoming taxi and an approssimative waiting time.

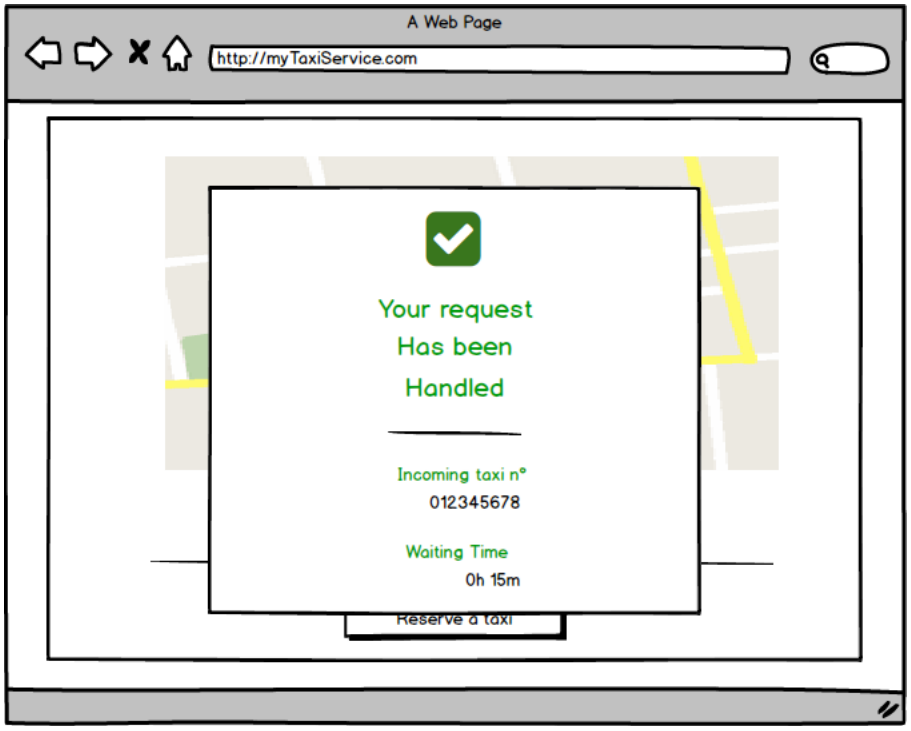


Figure 3.13: Customer nofication, web version

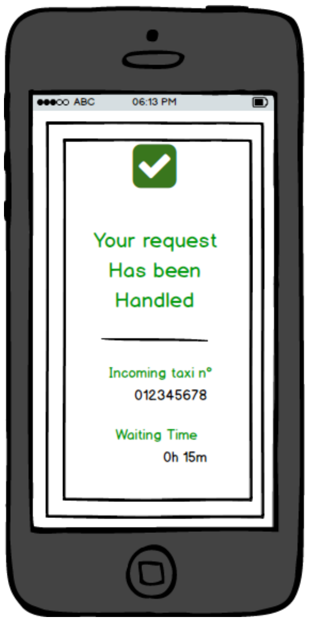
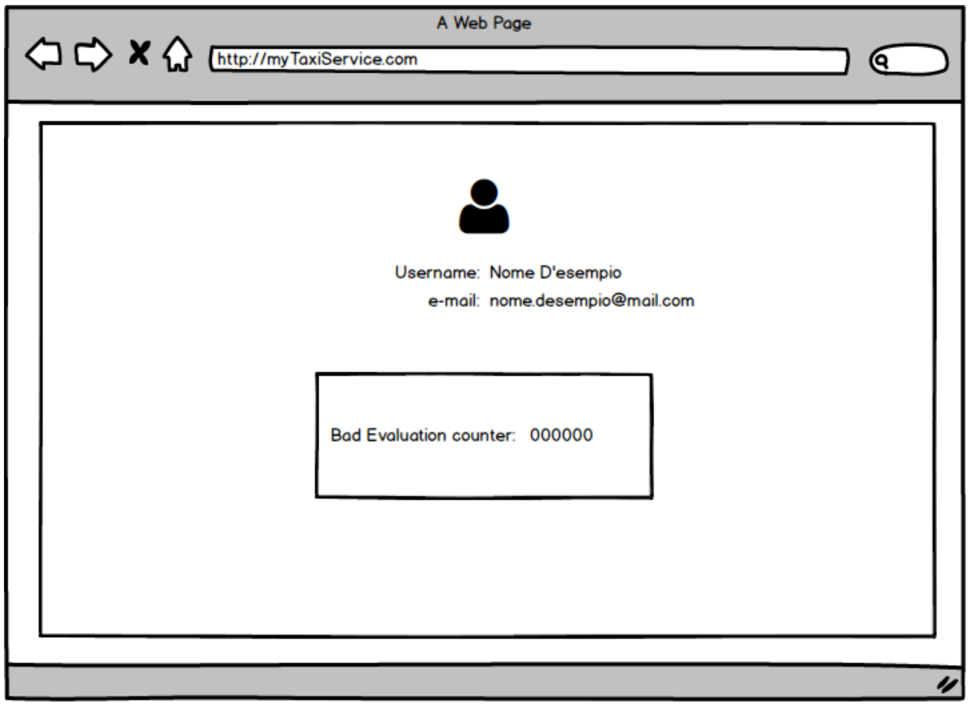


Figure 3.14: Customer notification, mobile version

3.1.1.8 User page

This page show all the information about your user and show how many time you've been notified as bad user.



Username: Nome D'esempio
e-mail: nome.desempio@mail.com

Bad Evaluation counter: 000000

Figure 3.15: User Page, web version

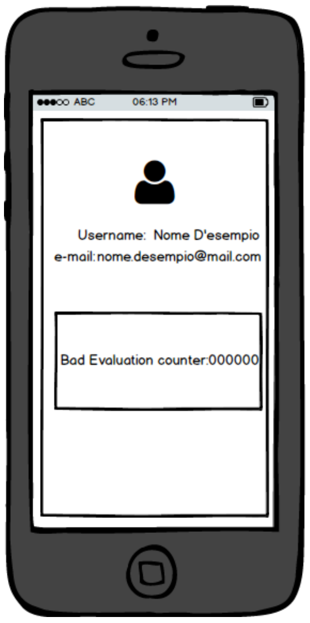


Figure 3.16: User Page, mobile version

3.1.1.9 Taxi Driver home page

On this page the Taxi driver can see his position (information from his GPS) and inform the system about his availability.

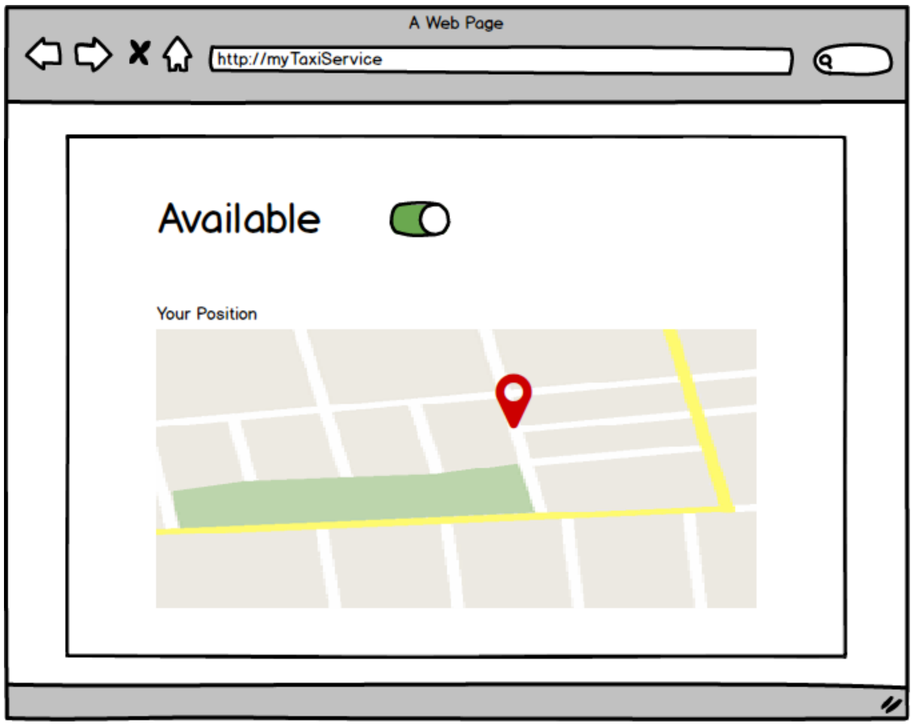


Figure 3.17: Taxi driver home page, web version

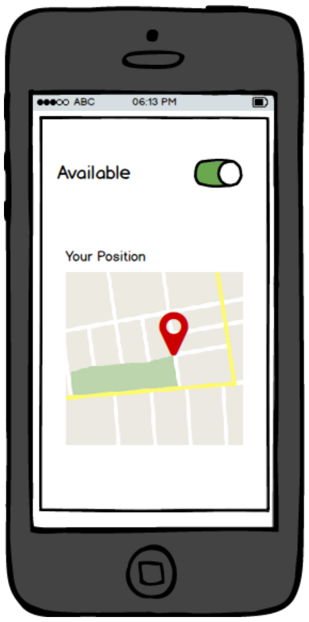


Figure 3.18: Taxi driver home page, mobile version

3.1.1.10 Taxi Driver notification

On this pop-up the taxi driver will notified of a request that he can handle. On this pop-up is showed also two button, with which the taxi driver can accept or decline the request.

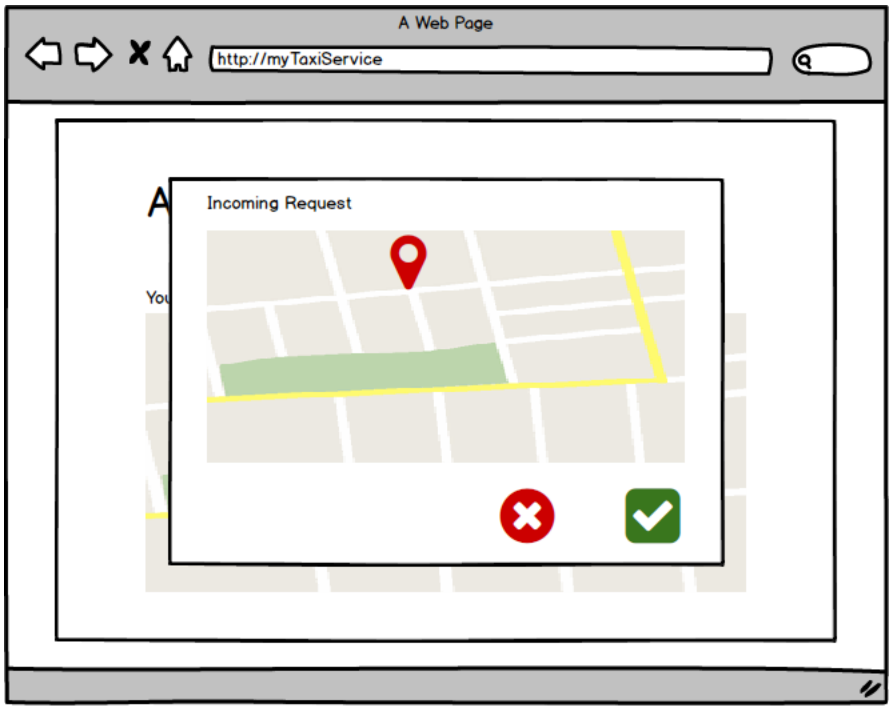


Figure 3.19: Taxi driver notification pop-up, web version

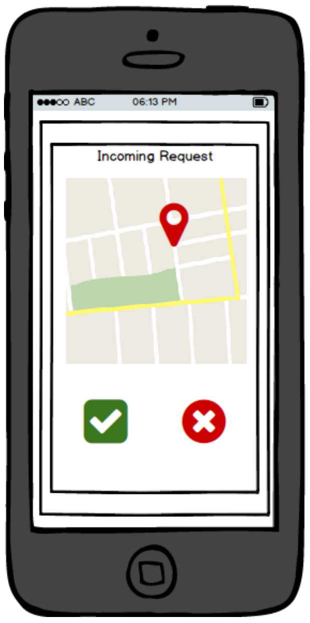


Figure 3.20: Taxi driver notification pop-up, mobile version

3.1.1.11 Taxi Driver end ride page

When a taxi driver accept a ride this page is showed up. On this page the taxi driver can check the name of the customer and the Origin position of his journey. When the ride is ended he can give a bad evaluation or simply end the ride and return to the main page.

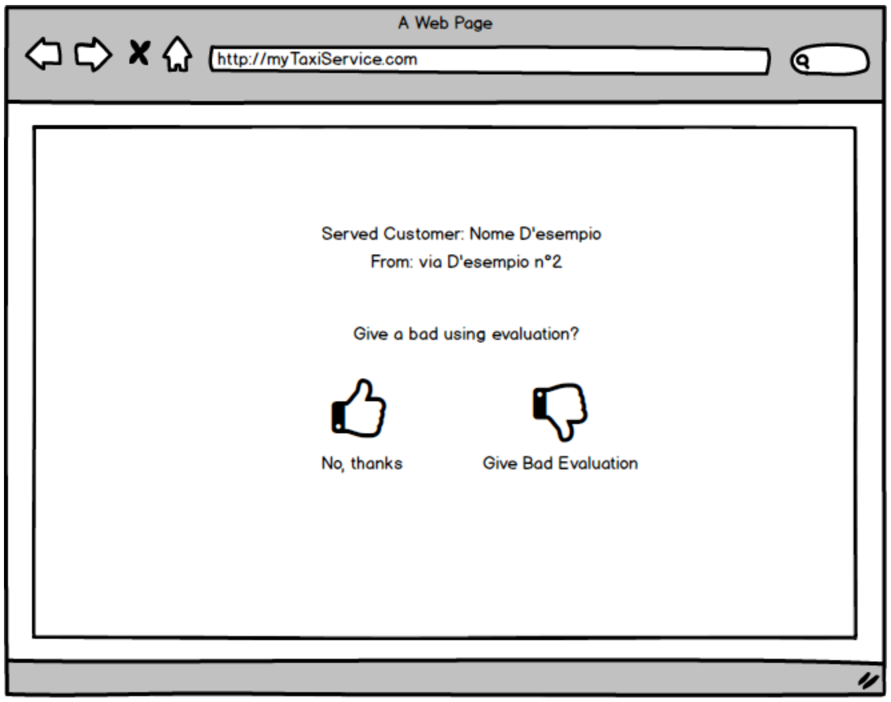


Figure 3.21: On Ride Page, web version

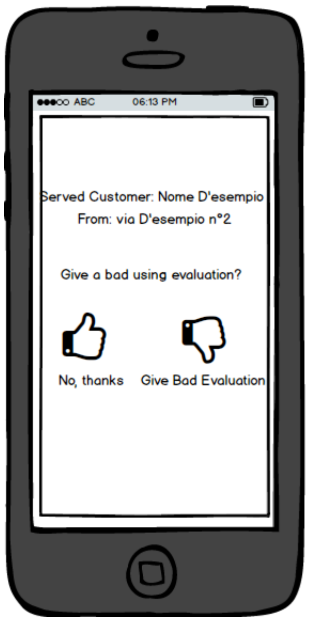


Figure 3.22: On Ride page, mobile version

3.1.2 Hardware Interfaces

Since mobile and web applications don't have any dedicated hardware we have not designed any hardware interfaces for our system. The interaction with the central database is performed by connections handled by the already installed operating system on the mobile devices or the users computers.

3.1.3 Software Interfaces

- The mobile application communicates with the GPS application in order to get geographical informations about the user.
- The web application communicates with the browser in order to get geographical informations about the user.
- Mobile and web applications communicates with the database through HTTP requests to the server.

3.1.4 Interfaces to Others Application

- myTaxiService web application require that at least one of these browsers is installed on the user Personal Computer:

Table 3.1: Browsers

Name	Version	Company	Source
Safari	9.0.1	Apple Inc.	Get Safari
Firefox	41.0	Mozilla	Get Firefox
Chrome	46.0.2490	Google	Get Chrome
Microsofr Edge	20.10240.16384.0	Microsoft	Get Edge

- myTaxiService mobile application require that at least one of these operating systems is installed on the user Smartphone:

Table 3.2: Mobile Operative Systems

Name	Version	Company	Source
Android	KitKat 4.4W.2 or later	Google	Android Info
iOS	9.1 or later	Apple Inc.	iOS Info
Windows 10	10.0.10572.0 or later	Microsoft	Windows 10 Info

- To give an additional LogIn method, we use also the "LogIn with Facebook" API relased by Facebook. Facebook Login for Apps is a fast and convenient way for people to create accounts and log into our system across multiple platforms. It is well described at Facebook Login API Page.

3.1.5 Communication Interfaces

The communication between system pieces is not specified because it is handled by the underlying operating systems for both the mobile application and the web portal.

In particular, the web and mobile applictaions will communicate with the server through HTTP/HTTPS requests.

- HTTP communicate through the port number 80 and is handled by the operating system.
- HTTPS communicate through the port number 443 and is handled by the operating system.

3.2 Functional Requirements

In this section are described, for every Actor, the Functional Requirements needed to reach the linked Goal.

3.2.1 Functional Requirements for Guest Users

Here are listed all the Functional Requirements referring to the Goals that affects Guest Users.

3.2.1.1 [G.1 - Allow guest users to become a customer creating a myTaxiService Account]

To allow the guest user to perform a successful registration the system has to:

- [Req.1] check if the selected username has not already been taken by another user to perform a successful registration.
- [Req.2] check if the selected password is at least 8 characters long.
- [Req.3] check if the selected password contains either digits and alphabetic characters.
- [Req.4] check if the provided email has not already been used by another user.
- [Req.5] check if the provided email respects this regular expression:
  “\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$”
- [Req.6] ensures that the user cannot access to pages different from registration and login.
- [Req.7] provide a registration page containing:
  1. A text box where the user must insert his username.
  2. Two text box where the user must insert his password (the second one is for security check).
  3. A text box where the user must insert his email.
  4. A button to submit informations to the system.
- [Dom.1] The email used by the Guest User is a valid one.

3.2.1.2 [G.2 - Allow guests user to become a customer using his Facebook Account]

To allow the guest user to perform a successful registration using Facebook API, the system has to:

- [Req.1] check if the Guest User is not already registered as a Customer.
- [Req.2] delegate to the Facebook system all the checks concerning the existence of the user.
- [Req.3] ensures that the user cannot access to pages different from registration and login.
- [Req.4] provide a registration page containing:
  1. A button that calls the Facebook Registration API.

3.2.1.3 [G.3 Allow guest users to become a taxi driver]

To allow the guest user to become a Taxi Driver, the system must:

- [Req.1] check if the selected username has not already been taken by another user.
- [Req.2] check if the selected password is at least 8 character long.
- [Req.3] check if the selected password contains either digits and alphabetic characters.
- [Req.4] check if the provided email has not already ben used by another user.
- [Req.5] check if the provided email respects this regular expression: "\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$"
- [Req.6] ensures that the user cannot access to pages different from registration and login.
- [Req.7] check if the provided taxi license has not already been used by another one.
- [Req.8] provide a registration page containing:
    1. A text box where the user must insert his username.
    2. Two text box where the user must insert his password (the second one is for security check).
    3. A text box where the user must insert his email.
    4. A text box where the user must insert his taxi license.
    5. A text box where the user must insert his taxi number.
    6. A button to submit informations to the system.
- [Dom.1] The email used by the Guest User is a valid one.

3.2.2 Functional Requirements for Registered Users

Here are listed all the Functional Requirements referring to the Goals that affects Registered Users.

3.2.2.1 [G.4 Allow registered users to log in with myTaxiService account.]

To allow a registered user to log in with his myTaxiService account, the system must:

- [Req.1] check if the given username is registered.
- [Req.2] check if the given password is related to the given username.

- [Req.3] ensures that the user cannot access to different pages from login and registration.
- [Req.4] provide a login page containing:
  1. A text box where the user must insert his username.
  2. A text box there the user must insert his password.
  3. A button to submit informations to the system.

3.2.2.2 [G.5 Allow a Registered User to view or modify his username and email.]

To allow a Registered User to view or modify his username and email, the system must:

- [Req.1] check if the Registered User is correctly logged in.
- [Req.2] provide an homepage that contains:
  1. A button that, if clicked, shows the page that contains the Registered User's informations.
- [Req.3] provide a page used to show Registered User's informations that contains:
  1. A clickable label showing the username of the Registered User.
  2. A clickable label showing the email of the Registered User.
  3. A box with a label inside that shows how many times the Registered User has been reported as a bad user.
- [Req.4] once clicked, the username label must become a text box in wich the Registered User can write his new username. The system must also ensure that the new username is not already taken by another registered user. If it was taken, then the system will cancel the update. Otherwise the system will have to update the record in the database that refers to the registered user with the new username.
- [Req.5] once clicked, the email label must become a text box in witch the Registered User can write his new email. The system must also ensure that the new email is not already taken by another registered user and respects the regular expression:
  “\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$”.
  If it was taken, then the system will cancel the update. Otherwise the system will have to update the record in the database that refers to the registered user with the new username.

3.2.2.3 [G.6 Allow a Registered User to retrieve his password if he doesn't remember it.]

To allow a Registered User to retrieve his password, the system must:

- [Req.1] on the log in page, provide a link that will call the function of the system that will send an email to the Registered User with his password.

3.2.2.4 [G.7 Allow a Registered Users to signal another one if he has made a bad use of the system.]

To allow a Registered User to signal another one, the system must:

- [Req.1] check if the Registered User is correctly logged in.
- [Req.2] for Taxi Drivers, provide an end-ride page that contains:
    1. A label showing the username of the Customer that he has just served.
    2. A label showing the starting location of the current ride.
    3. A label that asks to the Taxi Driver if he want to report the Customer.
    4. A button that, if clicked, ends the current ride calling the function of the system that sets to endedthe current ride without reporting the Customer.
    5. A button that, if clicked, A button that, if clicked, ends the current ride calling the function of the system that sets to taxi-endedthe current ride and increments the Customer's Bad Evaluation Counter.
- [Req.3] After that one of the Taxi Driver end-ride page button is pressed, the system must ensure that the page is not reachable anymore and the Taxi Driver must access only to his homepage.

- [Req.4] for Customers, provide an end-ride page that contains:
    1. A label showing the username of the Taxi Driver that has just served him.
    2. A label showing the starting location of the current ride.
    3. A label that asks to the Customer if he want to report the Taxi Driver.
    4. A button that, if clicked, ends the current ride calling the function of the system that sets to endedthe current ride without reporting the Taxi Driver.
    5. A button that, if clicked, ends the current ride calling the function of the system that sets to customer-endedthe current ride and increments the Taxi Driver's Bad Evaluation Counter.
- [Req.5] After that one of the Customer end-ride page button is pressed, the system must ensure that the page is not reachable anymore and the Customer must access only to his homepage.
- [Dom.1] The ride is ended after a Taxi Driver or a Customer action.

3.2.3 Functional Requirements for Customers

Here are listed all the Functional Requirements referring to the Goals that affects Customers.

3.2.3.1 [G.8 Allow customers to log in with Facebook account.]

To allow a customer to log in with Facebook account, the system must:

- [Req.1] provide a button that calls the Facebook Login API.
- [Req.2] delegate to the Facebook system all the checks about the existence of the user.

3.2.3.2 [G.9 Allow customers to require a taxi.]

To allow the customers to require a taxi, the system must:

- [Req.1] check if the customer is correctly logged in.
- [Req.2] provide an homepage that contains:
    1. A map showing the current location of the customer.
    2. A button that calls the function of the system to require a taxi (called "require button").
- [Req.3] once the "require button" is clicked (or tapped) the system must memorize the request in the remote database.
- [Req.4] once the request is memorized in the database, the system must send a request notification to the first Taxi Driver available on the zone queue.
- [Req.5] once the request has been accepted by a Taxi Driver, the system must send a notification to the customer with the incoming taxi informations and the supposed waiting time.

3.2.3.3 [G.10 Allow customers to reserve a ride.]

To allow the customers to reserve a ride, the system must:

- [Req.1] check if the customer is correctly logged in.
- [Req.2] provide an homepage that contains:
    1. A button that, if clicked, shows the page used to reserve a ride.
- [Req.3] provide a page used to reserve a ride that contains:
    1. A map where the customer can select the starting location for his ride.
    2. A map where the customer can select the ending location for his ride.

3. A field where the customer can insert the starting date for the ride.
4. A field where the customer can insert the starting time for the ride.
5. A button that calls the function of the system to reserve a ride (called "reserve button").

- [Req.4] "reserve button" must be clickable if and only if the reserve date and time is at least two hours after the current time. (for instance: if the current date is 10/10/2015 and current time is 10.00, the reservation time must be at least 12.00 of the 10/10/2015).
- [Req.5] after the "reserve button" is clicked (or tapped) the system must memorize the reservation in the remote database.
- [Req.6] once the reservation is memorized, the system must send a request notification 10 minutes before the starting time of the ride to the first Taxi Driver available on the zone queue.
- [Req.7] once the ride has been accepted by a Taxi Driver, the system must send a notification to the customer with the incoming taxi informations and the supposed waiting time.

3.2.3.4 [G.11 Allow customers to delete a previous reservations.]

To allow a customer to delete a previous reservation, the system must:

- [Req.1] check if the customer is correctly logged in.
- [Req.2] provide an homepage that contains:
  1. A button that, if clicked, shows the page that contains all the reservations made by the customer.
- [Req.3] provide a page that contains all the reservations made by the customer:
  1. All reservations that have a difference between the start time and the current time of less than two hours are not erasable
  2. Other reservations are erasable.
  3. The page also contains a button to add a new reservation. If clicked that button leads the customer to the page used to reserve a ride (described in the Functional Requirements Req.3 of G.4)
- [Dom.1] The reservation is deleted after the Customer action.

3.2.4 Functional Requirements for Taxi Drivers

Here are listed all the Functional Requirements referring to the Goals that affects Taxi Drivers.

3.2.4.1 [G.12 Allow Taxi Drivers to accept or decline a ride request.]

To allow Taxi Drivers to accept or decline ride requests, the system must:

- [Req.1] check if the Taxi Driver is correctly logged in.
- [Req.2] check if the state of the Taxi Driver is: Available.
- [Req.2] if the Taxi Driver is at the top of his zone queue, the system must notify him with a request notification as described in Req.4 of G.9.
- [Req.3] once the notification has been sent, the system must wait for the Taxi Driver acceptance or declination for 10 minutes, after that time, the request is forwarded to the next Taxi Driver in the zone queue.
- [Req.4] if the request has been accepted, the Taxi Driver must be redirected to the end-ride page where he can end the ride when he want. Also he is moved to the bottom of his zone queue and his state is setted to Not Available.
- [Req.3] if the request has been rejected, the Taxi Driver must be moved at the bottom of the zone queue and the Customer request is forwarded to the next Taxi Driver in the zone queue. Also the Taxi Driver is redirected to his home page and his state is setted to Available.

3.2.4.2 [G.13 Allow Taxi Drivers to notify their availability.]

To allow Taxi Drivers to notify their availability, the system must:

- [Req.1] check if the Taxi Driver is correctly logged in.
- [Req.1] provide an home page that contains:
    1. An ON/OFF button that signals to the system that the Taxi Driver is Available/Not Available.
    2. A map showing his current zone and position.

3.3 Scenarios

Here are described in natural language some useful scenarios that shows how the system should work in different cases:

3.3.1 S.1 Registration as a Customer creating a myTaxiService account

To read the Functional Requirements for this scenario, refer to G.1 Functional Requirements Specification

Aristotle is new in the system and he wants to simplify the way he calls for a taxi. So he download the "myTaxiService" application from the Play Store (since he has a smartphone with the fancy Android 5.0 Lollipop Operative System) and he open it. The application shows to Aristotle the login screen and he taps on the "Sign Up as a Customer" button (see Login Screen) going to the registration page. In that page, he has to fill the form (see Customer Registration Screen (Mobile)) and after that he clicks on the "Accept" button. If the data entered by Aristotle are ok, he is automatically logged in and he can start using myTaxiService. Otherwise, he is redirected to an error page, that describes what is the problem.

3.3.2 S.2 Registration as a Customer using a Facebook Account

To read the Functional Requirements for this scenario, refer to G.2 Functional Requirements Specifications

Pythagoras is new in the system and he wants to simplify the way he calls for a taxi, but he doesn't want to loose time inserting his credential into a boring registration form, so he decide to register with his Facebook Account. He download the application from the Apple App Store (since he has a glorious iPhone 6 Plus) and he open it. The application shows to Pythagoras the login screen and he taps on the "Log in With Facebook" button calling the Facebook Login API. If the registration is successful, the Facebook System will reply with the informations about the user and the myTaxiService system can create a record in the database with Pythagoras informations. Otherwise, he is redirected to an error page, that describes what is the problem.

3.3.3 S.3 Registration as a Taxi Driver creating a new Taxi Driver myTaxiService account.

To read the Functional Requirements for this scenario, refer to G.3 Functional Requirements Specifications

Thales is a great taxi worker and he wants to improve his service to the customers so he decide to join myTaxiService. First of all he visit the myTaxiService web page from his pc. That page shows the login screen (see Login Screen) and he clicks on the "Sign up as a Taxi Driver" button. The web application redirect him to the Taxi Drivers registration page. Thales fills up the form (see Taxi Driver Registration Screen (Web)) and clicks on the "Accept" button. The system perform a special check for the validity of the given taxi licence and taxi number. If the data entered by Thales are ok, he is automatically logged in and he can start using myTaxiService as a Taxi Driver. Otherwise, he is redirected to an error page, that describes what is the problem.

3.3.4 S.4 Login with a myTaxiService account.

To read the Functional Requirements for this scenario, refer to G.4 Functional Requirements Specifications

Epicurus is a Registered User and he wants to login to use the myTaxiServer Application with his smartphone. So he taps on the application icon and start using it. myTaxiService Application shows to him the login screen and Epicurus fills up the form with his credentials. After that he submits the data by pressing the "Login" button. The Application calls the server function to check the credentials of Epicurus and if them are ok, the server reply to the application with a success message and the homepage is shown to Epicurus. Otherwise the server reply with an error message that is shown to Epicurus.

3.4 UML Models

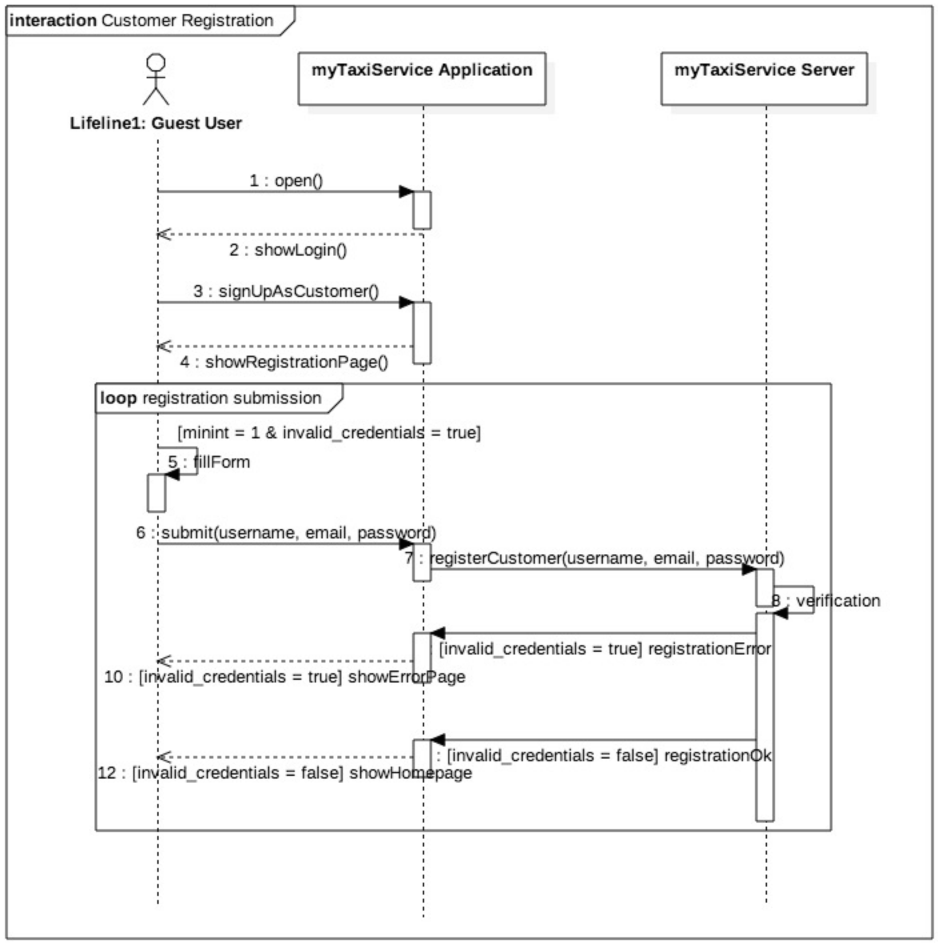
3.4.1 Use-Case Diagrams

3.4.2 Class Diagrams

3.4.2.1 Registration through the creation of a new myTaxiService account.

To accompany this diagram, read the Scenario S.1.

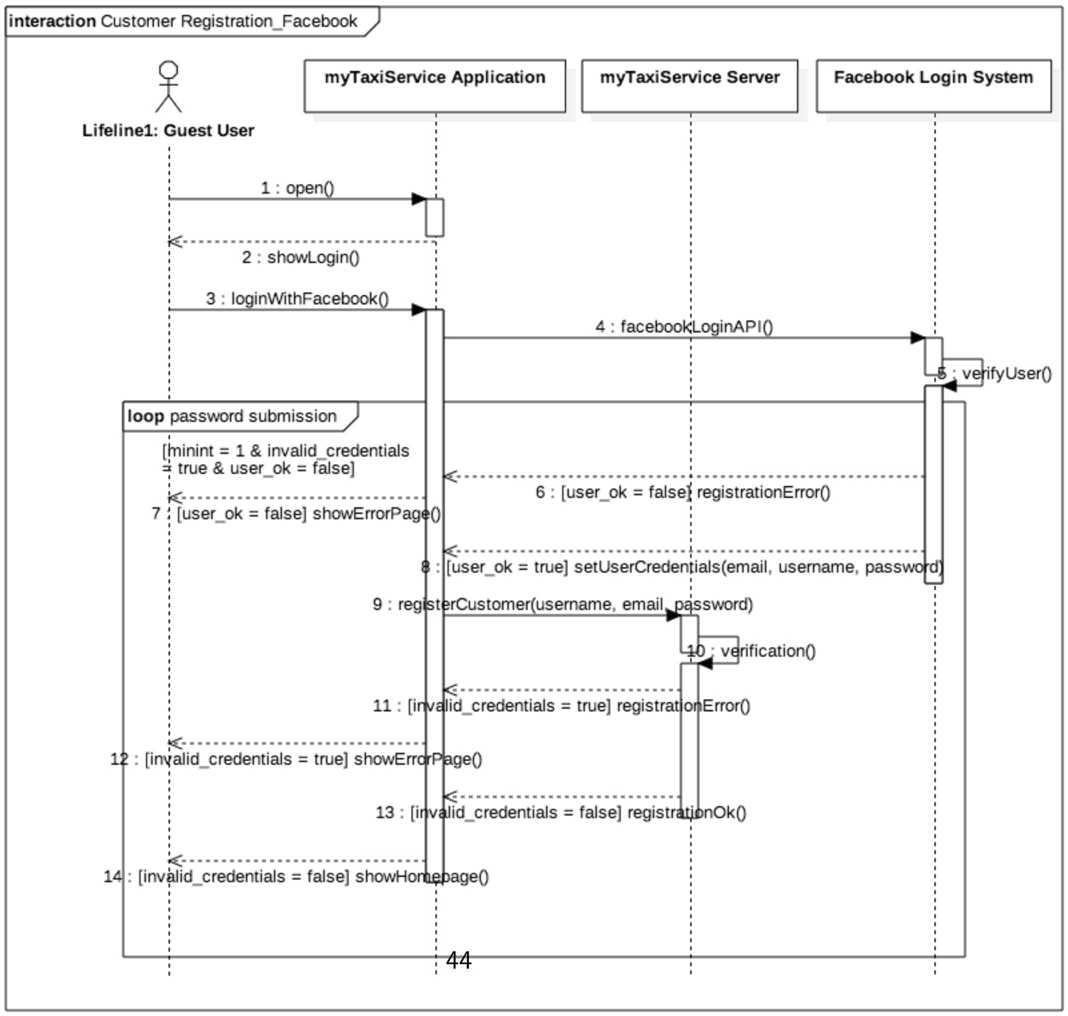
Subject	**Description**
Actors	Guest User, myTaxiService Application, myTaxiService Server
Preconditions	Guest User must not be already registered
Execution	1. Guest User open the myTaxiService Application. 2. Guest User taps on the "Sign Up as a Customer" button. 3. myTaxiService Application shows the registration page. 4. Guest User fills up the registration form. 5. Guest User taps on the "Submit" button.
Postconditions	The Guest User is now a Customer, he is registered in the database and he's now logged in.
Exceptions	1. The email regex is not optimal (no regex for email is optimal) 2. The connection is lost or the Guest User doesn't complete the registration clicking the button.


interaction Customer Registration
Lifeline1: Guest User
myTaxiService Application
myTaxiService Server
1 : open()
2 : showLogin()
3 : signUpAsCustomer()
4 : showRegistrationPage()
loop registration submission
[minint = 1 & invalid_credentials = true]
5 : fillForm
6 : submit(username, email, password)
7 : registerCustomer(username, email, password)
8 : verification
[invalid_credentials = true] registrationError
10 : [invalid_credentials = true] showErrorPage
: [invalid_credentials = false] registrationOk
12 : [invalid_credentials = false] showHomepage

3.4.2.2 Registration using a Facebook account.

To accompany this diagram, read the Scenario S.2.

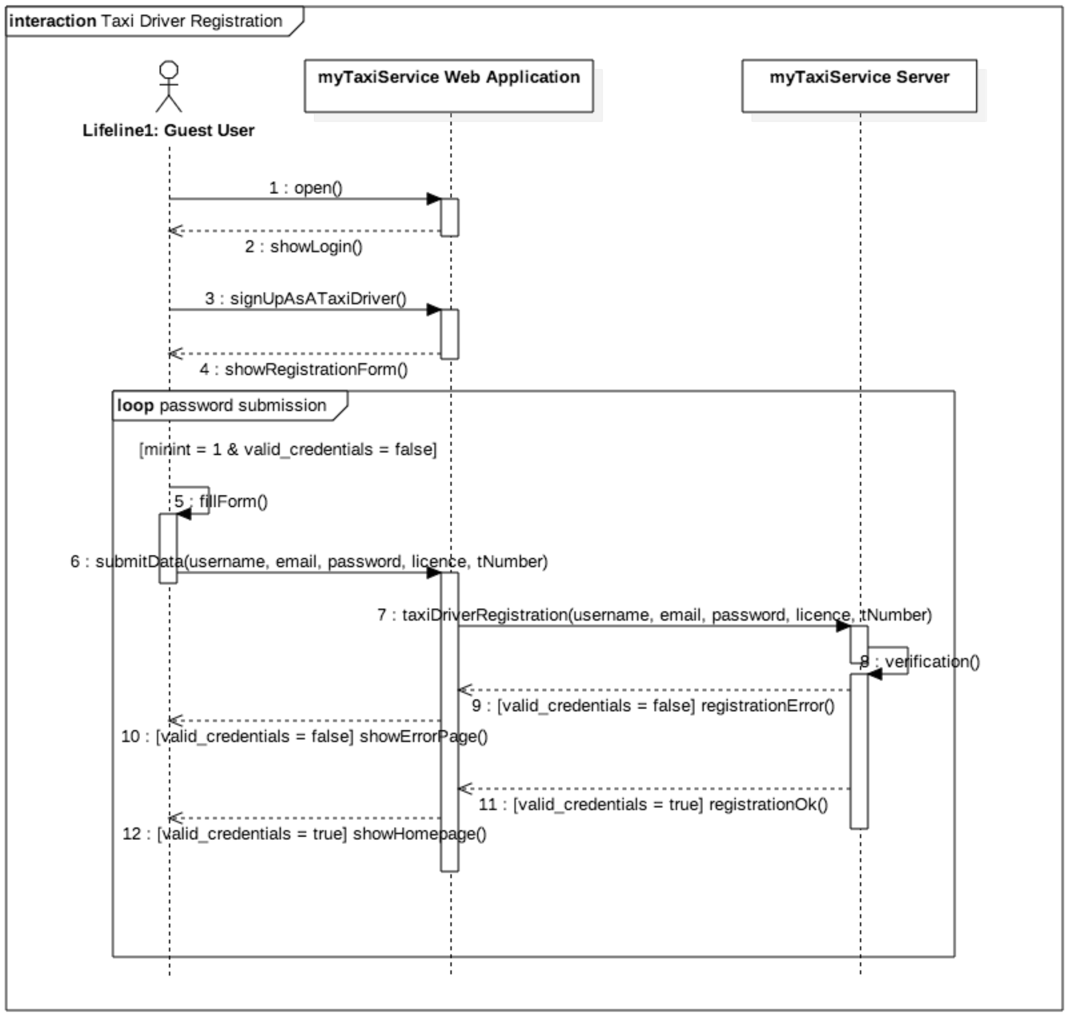
Subject	Description
Actors	Guest User, myTaxiService Application, myTaxiService Server, Facebook Login System
Preconditions	Guest User must not be already registered
Execution	1. Guest User open the myTaxiService Application. 2. Guest User taps on the "Log in with Facebook" button. 3. myTaxiService Application calls Facebook Login API 4. Facebook Login System makes his stuff 5. Facebook Login System reply 6. myTaxiService Application shows the result to the Guest User
Postconditions	The Guest User is now a Customer, he is registered in the database and he's now logged in.
Exceptions	1. The connection is lost or the Guest User doesn't complete the registration clicking the button.


interaction Customer Registration_Facebook
Lifeline1: Guest User
myTaxiService Application
myTaxiService Server
Facebook Login System
1 : open()
2 : showLogin()
3 : loginWithFacebook()
4 : facebookLoginAPI()
5 : verifyUser()
loop password submission
[minint = 1 & invalid_credentials = true & user_ok = false]
6 : [user_ok = false] registrationError()
7 : [user_ok = false] showErrorPage()
8 : [user_ok = true] setUserCredentials(email, username, password)
9 : registerCustomer(username, email, password)
10 : verification()
11 : [invalid_credentials = true] registrationError()
12 : [invalid_credentials = true] showErrorPage()
13 : [invalid_credentials = false] registrationOk()
14 : [invalid_credentials = false] showHomepage()

3.4.2.3 Taxi Driver Registration

To accompany this diagram, read the Scenario S.3.

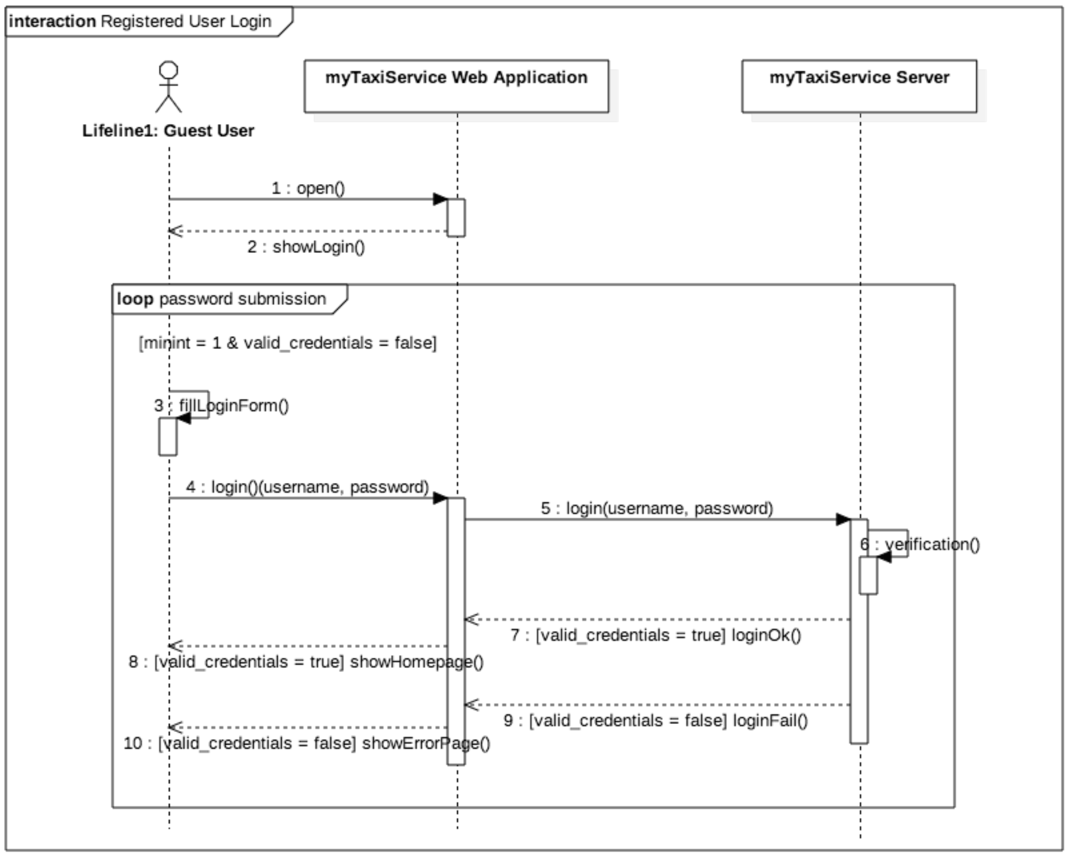
Subject	**Description**
Actors	Guest User, myTaxiService Web Application, myTaxiService Server
Preconditions	Guest User must not be already registered as a Taxi Driver
Execution	1. Guest User open the myTaxiService Web Application. 2. Guest User taps on the "Sign Up as a Taxi Driver" button. 3. myTaxiService Web Application shows the registration page. 4. Guest User fills up the registration form. 5. Guest User taps on the "Submit" button. 6. myTaxiService Web Application calls the server registration service. 7. myTaxiService Server verify the credentials 8. myTaxiService Server reply with an ok or an error 9. myTaxiService Web Application shows to the Guest User the result of his registration.
Postconditions	The Guest User is now a Taxi Driver, he is registered in the database and he's now logged in.
Exceptions	1. The email regex is not optimal (no regex for email is optimal) 2. The connection is lost or the Guest User doesn't complete the registration clicking the button.


interaction Taxi Driver Registration
Lifeline1: Guest User
myTaxiService Web Application
myTaxiService Server
1 : open()
2 : showLogin()
3 : signUpAsATaxiDriver()
4 : showRegistrationForm()
loop password submission
[minint = 1 & valid_credentials = false]
5 : fillForm()
6 : submitData(username, email, password, licence, tNumber)
7 : taxiDriverRegistration(username, email, password, licence, tNumber)
8 : verification()
9 : [valid_credentials = false] registrationError()
10 : [valid_credentials = false] showErrorPage()
11 : [valid_credentials = true] registrationOk()
12 : [valid_credentials = true] showHomepage()

3.4.2.4 Registered User Login

To accompany this diagram, read the Scenario S.4.

Subject	Description
Actors	Guest User, myTaxiService Web Application, myTaxiService Server
Preconditions	1. Registered User must be registered on the system. 2. Registered User must not be already logged in.
Execution	1. Registered User open the myTaxiService Web Application. 2. Registered User fills the login form. 3. Registered User press on the "Login" button. 4. myTaxiService Application submits the data to the server. 5. myTaxiService Server checks the data. 6. myTaxiService Server reply with an error or with success. 7. myTaxiService Application shows an error notification or a success one.
Postconditions	The Registered User is now logged in and is able to use the application.
Exceptions	1. The connection is lost or the Guest User doesn't complete the login clicking the button.


interaction Registered User Login
Lifeline1: Guest User
myTaxiService Web Application
myTaxiService Server
1 : open()
2 : showLogin()
loop password submission
[minint = 1 & valid_credentials = false]
3 : fillLoginForm()
4 : login()(username, password)
5 : login(username, password)
6 : verification()
7 : [valid_credentials = true] loginOk()
8 : [valid_credentials = true] showHomepage()
9 : [valid_credentials = false] loginFail()
10 : [valid_credentials = false] showErrorPage()

3.4.3 State Machine Diagrams

3.5 Non Functional Requirements

3.5.1 Performance Requirements

3.5.2 Design Constraints

3.5.3 Software System Attributes

3.5.3.1 Availability

3.5.3.2 Maintainability

3.5.3.3 Portability

3.5.4 Security

3.5.4.1 External Interface Side

3.5.4.2 Application Side

3.5.4.3 Server Side

Chapter 4

Appendix

//TODO