myTaxiService

Design Document

Belluschi Marco 791878, Cerri Stefano 849945, Di Febbo Francesco 852389

December 2, 2015

Contents

1 Introduction **3**
- 1.1 Purpose . . . 3
- 1.2 Scope . . . 3
- 1.3 Definitions, Acronyms, Abbreviations . . . 4
- 1.4 Reference Documents . . . 4
- 1.5 Document Structure . . . 5

2 Architectural Design **6**
- 2.1 Overview . . . 6
- 2.2 High level components and their interaction . . . 6
  - 2.2.1 myTaxiService . . . 6
  - 2.2.2 Web app . . . 7
  - 2.2.3 Mobile app . . . 7
  - 2.2.4 Account manager . . . 8
  - 2.2.5 Ride manager . . . 8
  - 2.2.6 Taxi queues . . . 8
  - 2.2.7 Map services . . . 9
  - 2.2.8 Notification . . . 9
  - 2.2.9 Data manager . . . 10
  - 2.2.10 myTaxiDatabase . . . 10
- 2.3 Component view . . . 11
- 2.4 Deployment view . . . 11
- 2.5 Runtime view . . . 12
- 2.6 Component interfaces . . . 19
- 2.7 Selected architectural styles and patterns . . . 19
- 2.8 Other design decisions . . . 20

3 Algorithm Design **21**

4 User Interface Design **22**
- 4.1 Purpose . . . 22
- 4.2 UX diagram . . . 22
- 4.3 Mockups . . . 24

5 Requirements Traceability **29**

6 References **39**

A Appendix **40**

A.1 Software and tool used 40

A.2 Working hours 40

Chapter 1

Introduction

1.1 Purpose

This document contains the complete design description of myTaxiService. This includes the architectural features of the system down through details of what operations each code module will perform and the database layout. It also shows how the use cases detailed in the RASD will be implemented in the system using this design. The primary audiences of this document are the software developers.

1.2 Scope

MyTaxiService is a taxi service for a large city. The main goals of the system are:

- simplify the access of passengers to the service
- guarantee a fair management of taxi queues

The system architecture will be a three-tier architecture: client, server application and server database. It will be created by using the MVC architectural pattern.

The system will be divided into components with respect to the principles leading to good design:

- Each individual component will be smaller in order to be easier to understand
- Coupling will be reduce where possible
- Reusability and flexibility will be increase in order to make easier future implementation

The system will have efficient algorithm in order to increase its performance; in the document will be given special attention to the sharing algorithm.

1.3 Definitions, Acronyms, Abbreviations

Definitions

- User: person that uses the service applications
- Visitor: user that has not registered nor logged in
- Registered user: user that has registered to the service
- Passenger: passenger registered to the service
- Taxi driver: taxi driver registered to the service
- System: the union of software and hardware to be developed and implemented

Acronyms

- RASD: requirements analysis and specification document
- AES: Advanced Encryption Standard
- FIFO: First In First Out
- ETA: estimated time of arrival
- API: application programming interface
- GPS: Global Positioning System
- MVC: Model View Controller
- UX: User Experience

1.4 Reference Documents

- Software Engineering 2 Project AA 2015/2016: Project Description And Rules
- Software Engineering 2 Project AA 2015/2016: Assignments 1 and 2 (RASD and DD)
- Software Engineering 2 Project AA 2015/2016: Structure of the design document

1.5 Document Structure

This document is essentially structured in seven parts:

- Introduction: it gives a description of the document and some basical information about the system design and archicture.
- Architectural Design: This is the core of the document. It gives general information about the architectural design. It also describes how the system will be divided into components and how the components communicate. It also has a description of the design pattern and architectural styles that will be used.
- Algorithm Design: it gives a description of the main algorithm that will be implemented. More focus will be given in the sharing algorithm.
- User Interface Design: it gives a description of the user interfaces of the system and the flow from one interface to another.
- Requirements Traceability: this section documents the life of a requirement and provides bi-directional traceability between various associated requirements.
- References: it gives information on the guidelines used in order to redact this document.
- Appendix: it provides informations that are not considered part of the actual DD. It includes: software and tools used, project group organization.

Chapter 2

Architectural Design

2.1 Overview

This chapter describes the software system, the relationships between software components and the relationship to actors with the system. Each component is described by a specification and an interface design. The specification is a description of its purpose, its functionality, its attributes (including dependency on other components) and the constraints under which it must operate. It also describes resources, that is, any elements used by the component which are external to the design such as physical devices and software services. The interface design is the list of the services that it provides to clients. These services are methods (procedures and functions), each carefully documented.
Each component in turn may provide its services by having an internal architectural design with its own set of subordinate components. These components may be called sub-components. The decomposition of a higher-level component into subordinate component must be explicit. The algorithm that shows how each method of the larger component is performed by these components must be explicit. Any data stored in an component must be explicitly described.

2.2 High level components and their interaction

2.2.1 myTaxiService

Type System

Node myTaxiServer

Description This is the primary entrance to the system for a user. It provides all the functionalities of myTaxiService by the interaction of software components. It provide APIs for software development.

Dependencies myTaxiDatabase, Google Maps API

Resources Glassfish 4.1.1 on Linux Ubuntu Server 14.04.3L LTE on server

Operations

- Account API (sign-up, login, personal information editing, taxi driver availability handling)
- Ride request API (standard, shared, reserve ride request)
- Notification API (notifications from system)
- Map API (city zones, geolocation services)
- Queues API (taxi queues management)

2.2.2 Web app

Type Web app

Node client

Description The web application provide an easy way to reach myTaxiService functionalities. Through supported web browser, the passenger can request standard, shared or reserved rides.

Dependencies Account API, Ride request API, Notification API

Resources supported web browser (see RASD for further informations)

Operations

- Sign up, login, personal informations editing
- Ride requests (only functionalities for passenger)
- Notification from service

2.2.3 Mobile app

Type Mobile app

Node client

Description The mobile application extends web functionalities by adding taxi driver services.

Dependencies Account API, Ride request API, Notification API

Resources supported devices (see RASD for further informations)

Operations

- Sign up, login, personal informations editing (for taxi driver, taxi driver availability handling)
- Ride request (passenger: request for ride; taxi driver: request handling)
- Notification from service

2.2.4 Account manager

Type Subsystem

Node myTaxiServer

Description This component manages users connected to the service. It manages sign-up and login functionalities, profile editing and taxi driver availability handling.

Dependencies Data manager component

Resources

Operations

- Sign up, login, personal informations editing (for taxi driver, taxi driver availability handling)
- Ride request (passenger: request for ride; taxi driver: request handling)
- Notification from service

2.2.5 Ride manager

Type Subsystem

Node myTaxiServer

Description this component manages the incoming requests from passengers and responses from taxi drivers. It manages all types of requests, including standard, shared and reserved rides thanks to the interaction with Taxi Queues component. It takes care of messages dispatching to users by Notification component. It provides APIs for software development.

Dependencies Data manager, Map services, Taxi queues, Notification, Account manager components

Resources

Operations

–

2.2.6 Taxi queues

Type Subsystem

Node myTaxiServer

Description this component manages the queue of available taxis for each city zone.

Dependencies Map services component

Resources

Operations

–

2.2.7 Map services

Type Subsystem

Node myTaxiServer

Description this component manages the city map, the taxi zones, the location informations provided by GPS navigation systems of users.

Dependencies Data manager component, Google Maps API

Resources

Operations

–

2.2.8 Notification

Type Subsystem

Node myTaxiServer

Description this component takes care of messages dispatching to the users.

Dependencies Account manager component

Resources

Operations

–

2.2.9 Data manager

Type Subsystem

Node myTaxiServer

Description this component provides access to all of the data contained in the database. It provides various functions that allow entry, storage and retrieval of large quantities of information and provides ways to manage how that information is organized.

Dependencies myTaxiDatabase

Resource

Operations

- –

2.2.10 myTaxiDatabase

Type System

Node myTaxiDatabase

Description this system takes care of data storing and provisioning.

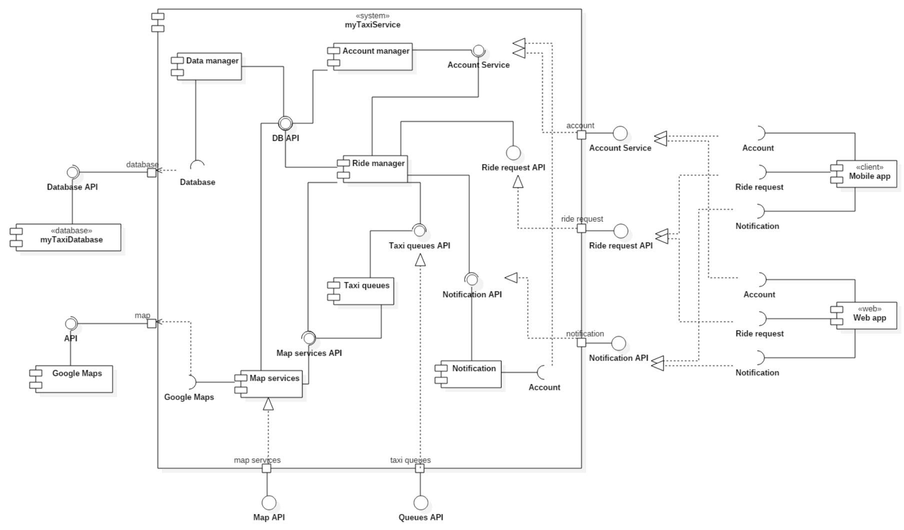
Dependencies

Resources Data warehouse

Operations

- –

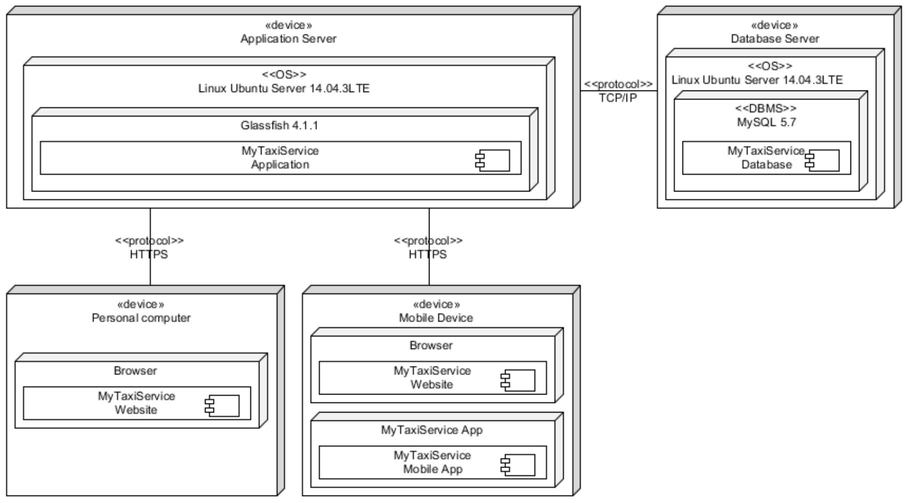
2.3 Component view



Component diagram of the entire system

2.4 Deployment view

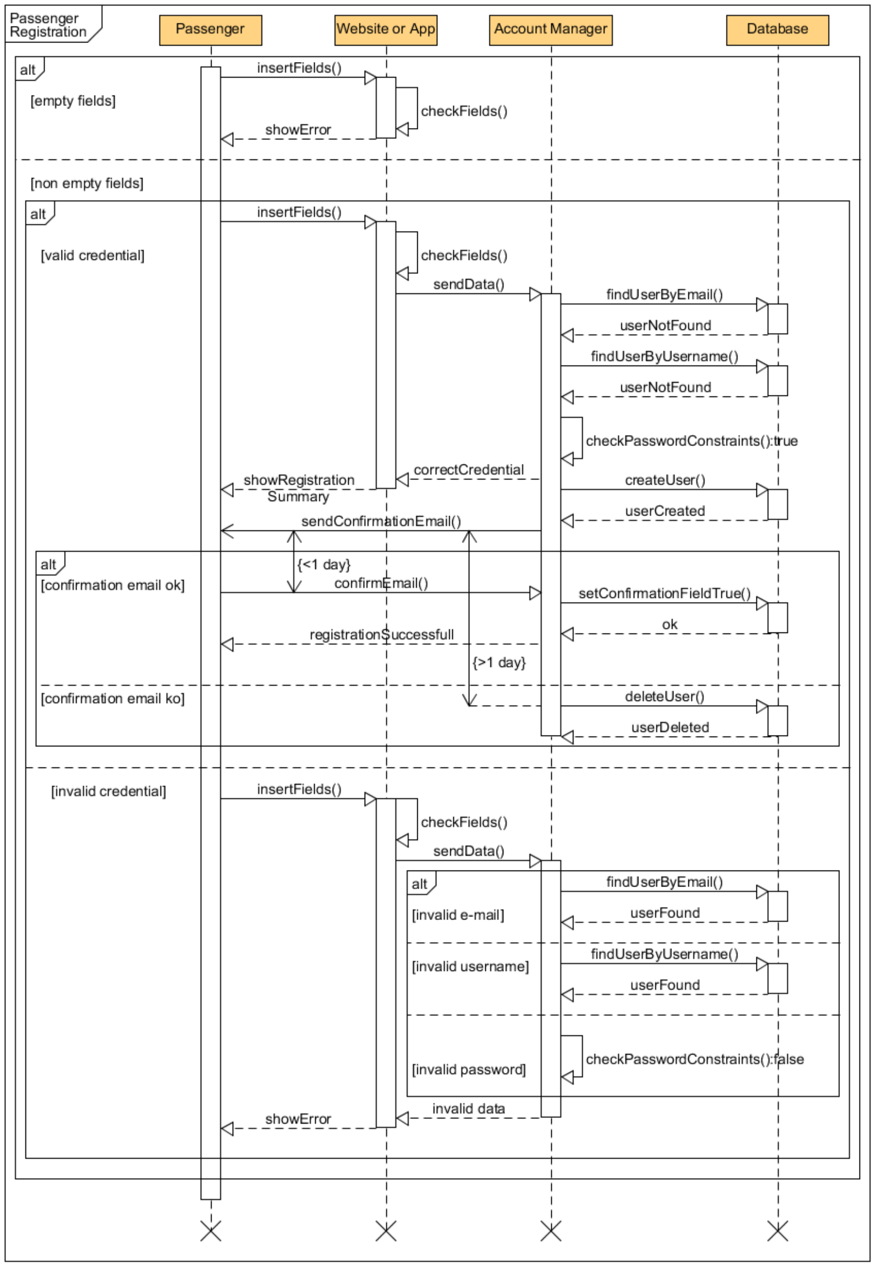
Here is the deployment diagram for MyTaxiService:

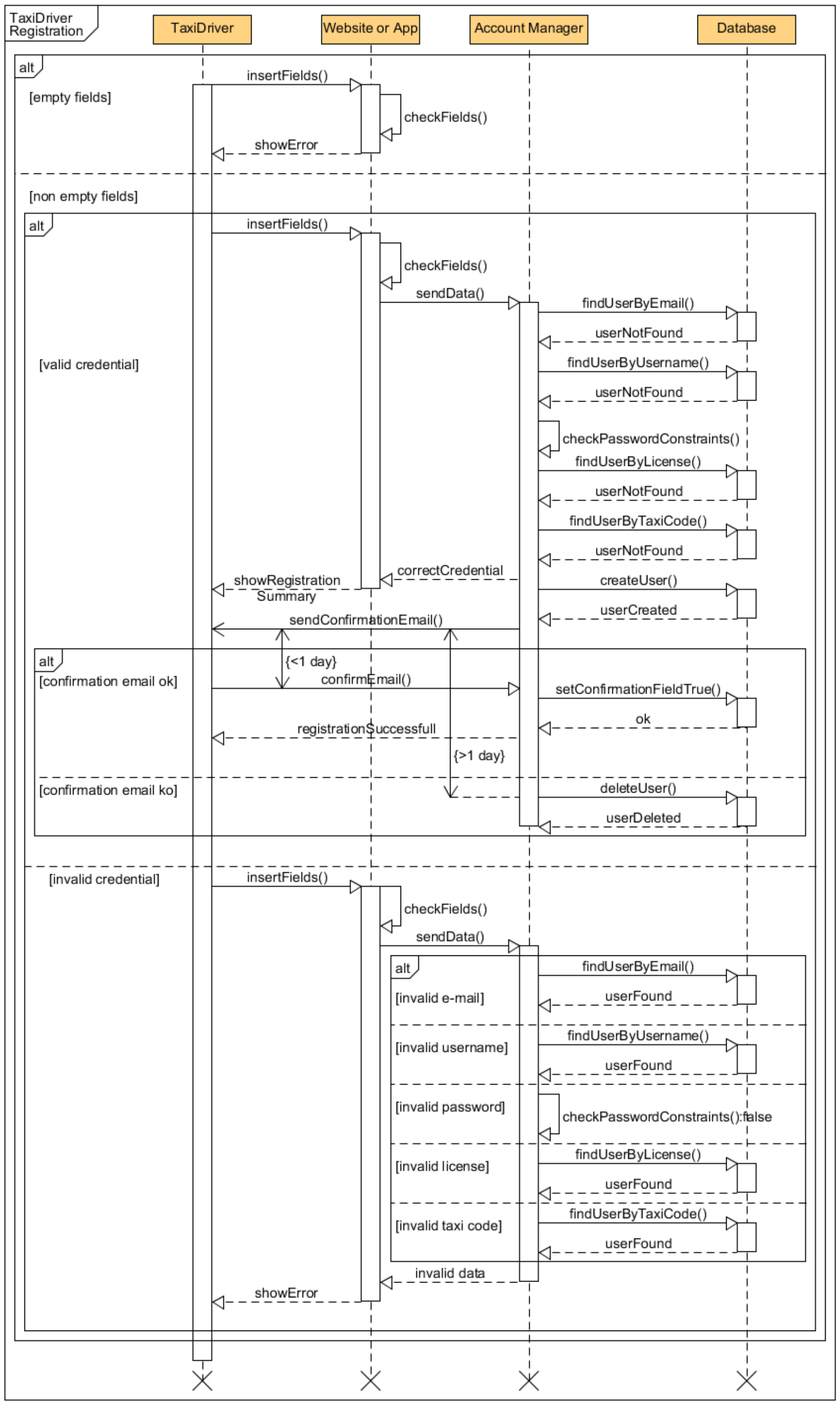


2.5 Runtime view

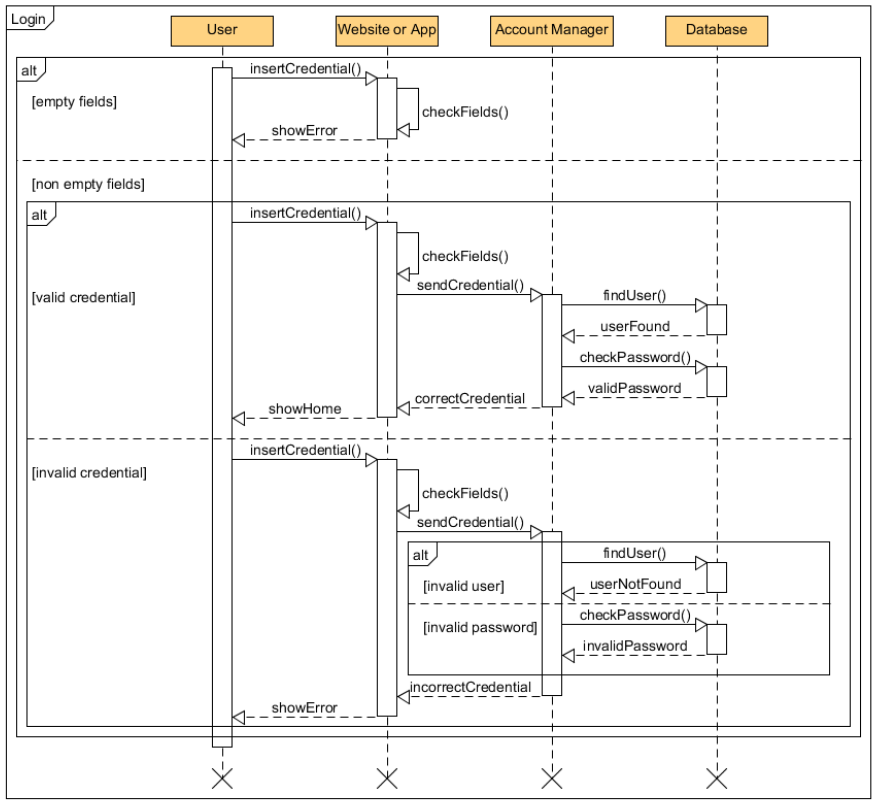
In this section will be presented the interactions between components usign some sequence diagram.

Registration



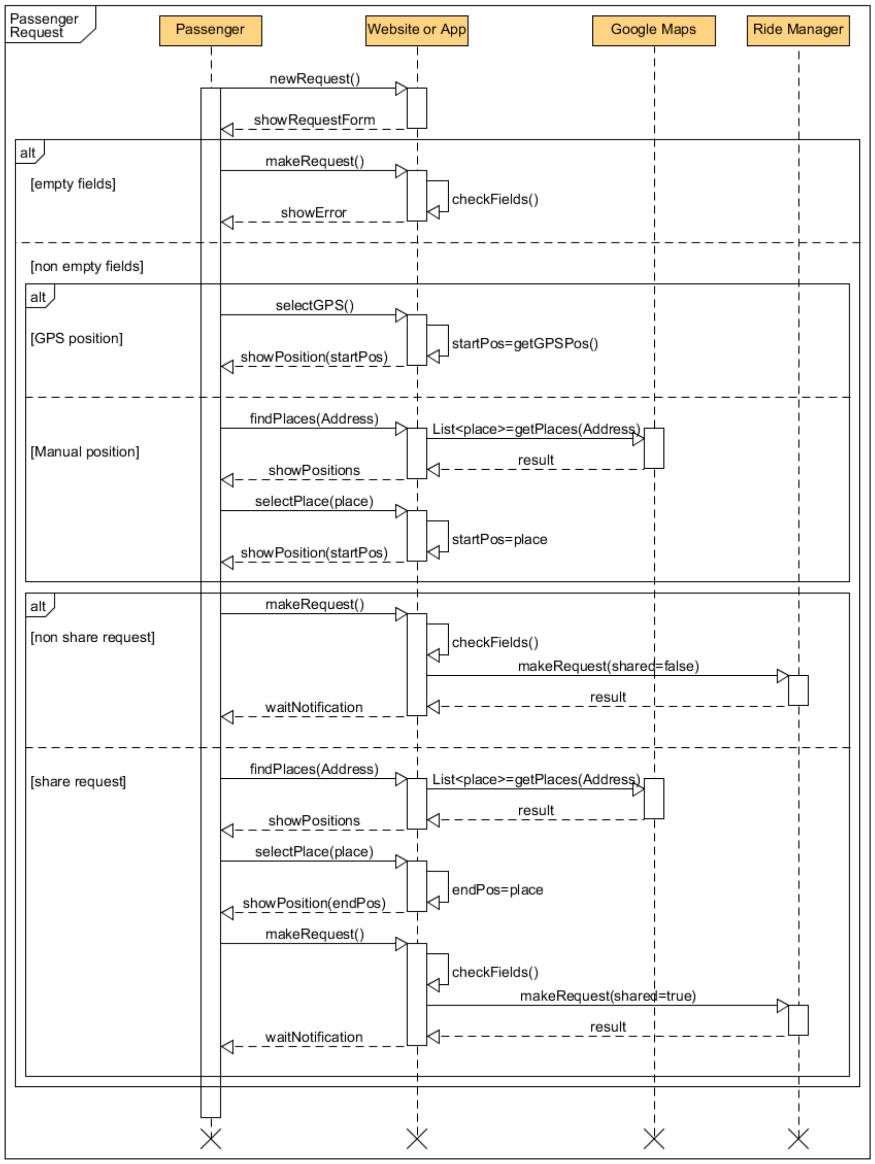

TaxiDriver Registration
TaxiDriver
Website or App
Account Manager
Database
alt
[empty fields]
insertFields()
checkFields()
showError
[non empty fields]
alt
[valid credential]
sendData()
findUserByEmail()
userNotFound
findUserByUsername()
checkPasswordConstraints()
findUserByLicense()
findUserByTaxiCode()
correctCredential
showRegistration Summary
createUser()
userCreated
sendConfirmationEmail()
[confirmation email ok]
{<1 day}
confirmEmail()
setConfirmationFieldTrue()
ok
registrationSuccessfull
{>1 day}
[confirmation email ko]
deleteUser()
userDeleted
[invalid credential]
[invalid e-mail]
userFound
[invalid username]
[invalid password]
checkPasswordConstraints():false
[invalid license]
[invalid taxi code]
invalid data

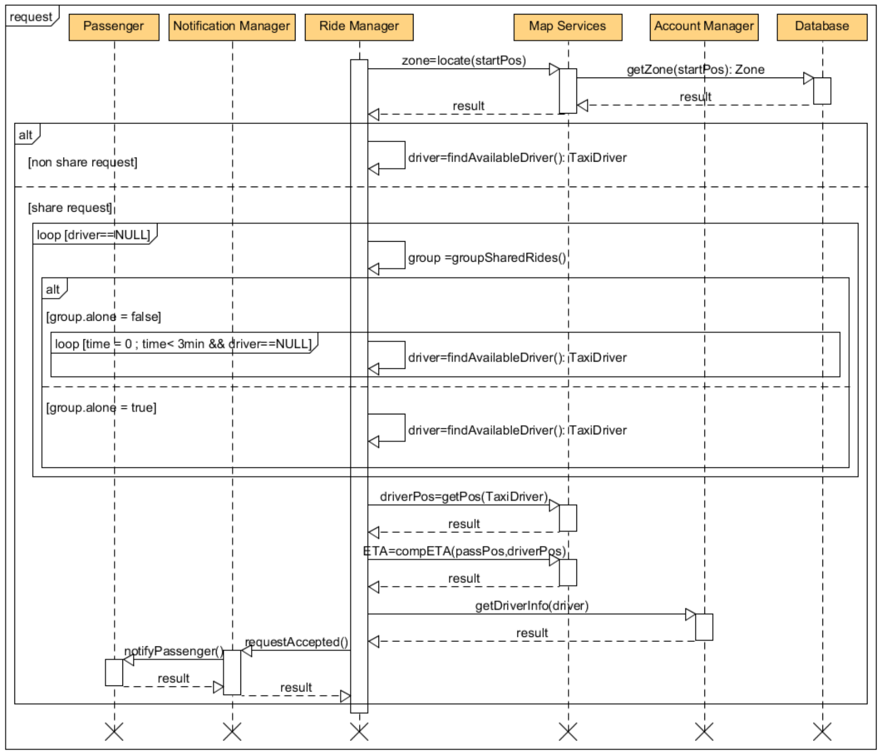
Login



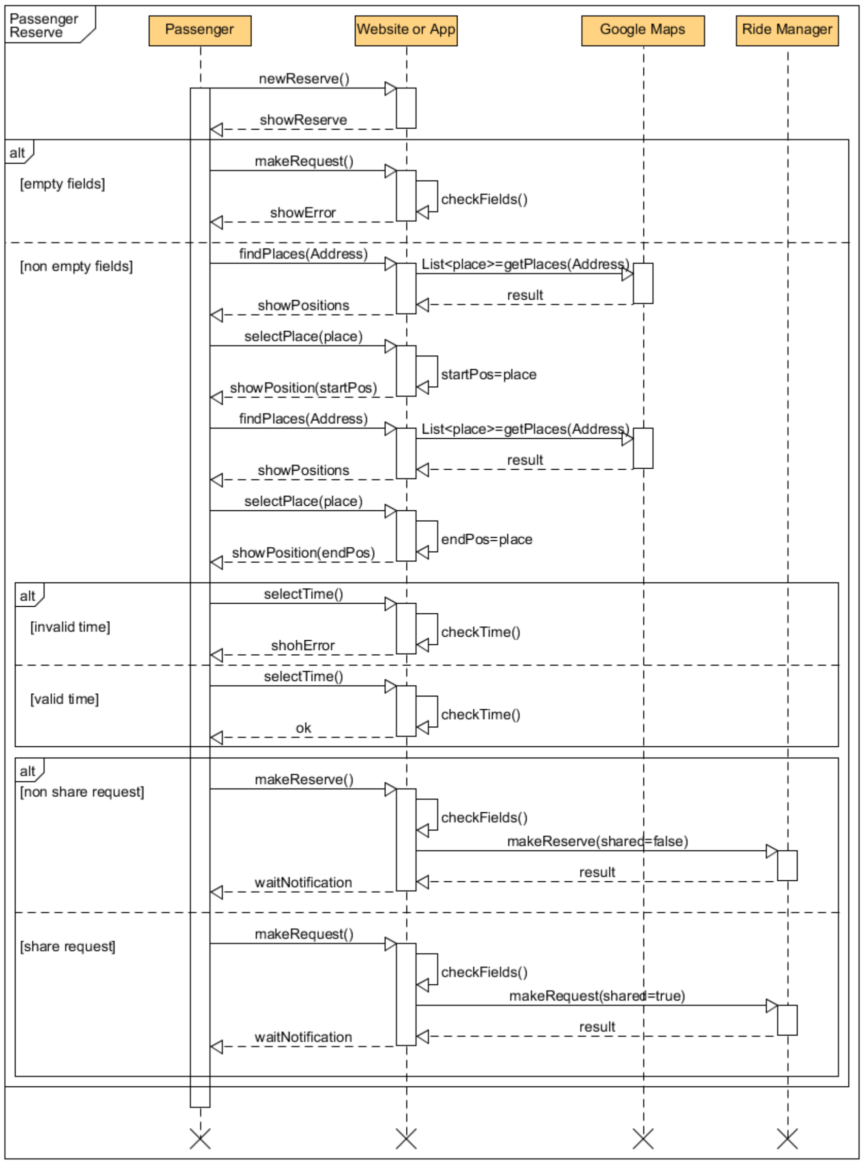
Find Available Driver

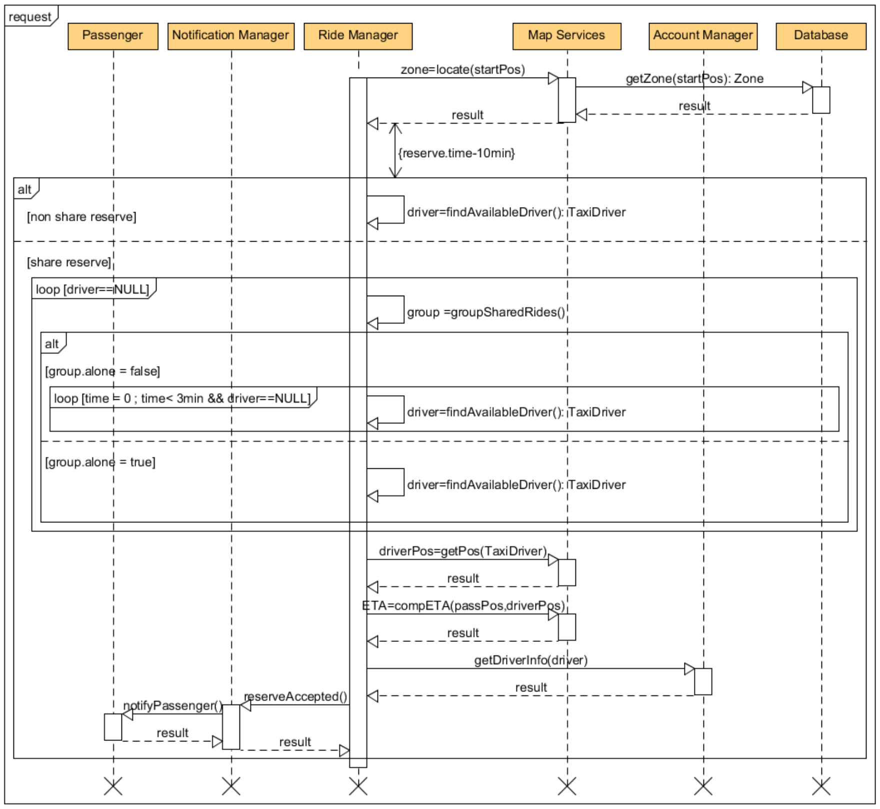
Taxi Request


Passenger Request
Passenger
Website or App
Google Maps
Ride Manager
newRequest()
showRequestForm
alt
[empty fields]
makeRequest()
checkFields()
showError
[non empty fields]
alt
[GPS position]
selectGPS()
startPos=getGPSPos()
showPosition(startPos)
[Manual position]
findPlaces(Address)
List<place>=getPlaces(Address)
result
showPositions
selectPlace(place)
startPos=place
showPosition(startPos)
alt
[non share request]
makeRequest()
checkFields()
makeRequest(shared=false)
result
waitNotification
[share request]
findPlaces(Address)
List<place>=getPlaces(Address)
result
showPositions
selectPlace(place)
endPos=place
showPosition(endPos)
makeRequest()
checkFields()
makeRequest(shared=true)
result
waitNotification



Taxi Reserve


Passenger Reserve
Passenger
Website or App
Google Maps
Ride Manager
newReserve()
showReserve
alt
[empty fields]
makeRequest()
checkFields()
showError
[non empty fields]
findPlaces(Address)
List<place>=getPlaces(Address)
result
showPositions
selectPlace(place)
startPos=place
showPosition(startPos)
findPlaces(Address)
List<place>=getPlaces(Address)
result
showPositions
selectPlace(place)
endPos=place
showPosition(endPos)
alt
[invalid time]
selectTime()
checkTime()
shohError
[valid time]
selectTime()
checkTime()
ok
alt
[non share request]
makeReserve()
checkFields()
makeReserve(shared=false)
result
waitNotification
[share request]
makeRequest()
checkFields()
makeRequest(shared=true)
result
waitNotification



2.6 Component interfaces

2.7 Selected architectural styles and patterns

Various architectural and logical choices have been justified as following:

- A **3-tier architecture** has been used: client, server application and server database.
  This is due to the fact that we're developing an overall light application that doesn't need a lot of computing power, especially on the client side. Therefore, it is possible to structure the system in a logically simple and easily understandable way, without having to lose much in terms of optimization. Besides, this allows to obtain a good compromise between thin client and database tiers, and a clear correspondence between tiers and layers (i.e. no layer is distributed among multiple tiers).

- For similar reasons, a **SOA (Service Oriented Architecture)** has been chosen for the communication of the application server with the front ends.

This improves flexibility, through modularity and a clearer documentation, and simplicity, through an higher abstraction of components.

- The **Plug-in logic** greatly improves modularity, allowing for a minimized application core to which adding, when needed, all the additional features and functionalities through apposite modules.
- The **Client&Server** logic is the most common, simple way to manage the communication both between client and application server, and between application server and database.
- The **MVC (Model-View-Controller)** pattern, besides being a common choice in object-oriented languages like Java, allows for a clear logical division of the various elements of the program.

2.8 Other design decisions

The system uses **Google Maps** to perform all the operations related to maps, i.e. map and position visualization, geolocalization, distance and ETA calculation. This is an easy and fast to develop solution that relies on a worldwide, well-known and well-established software.

Chapter 3

Algorithm Design

Chapter 4

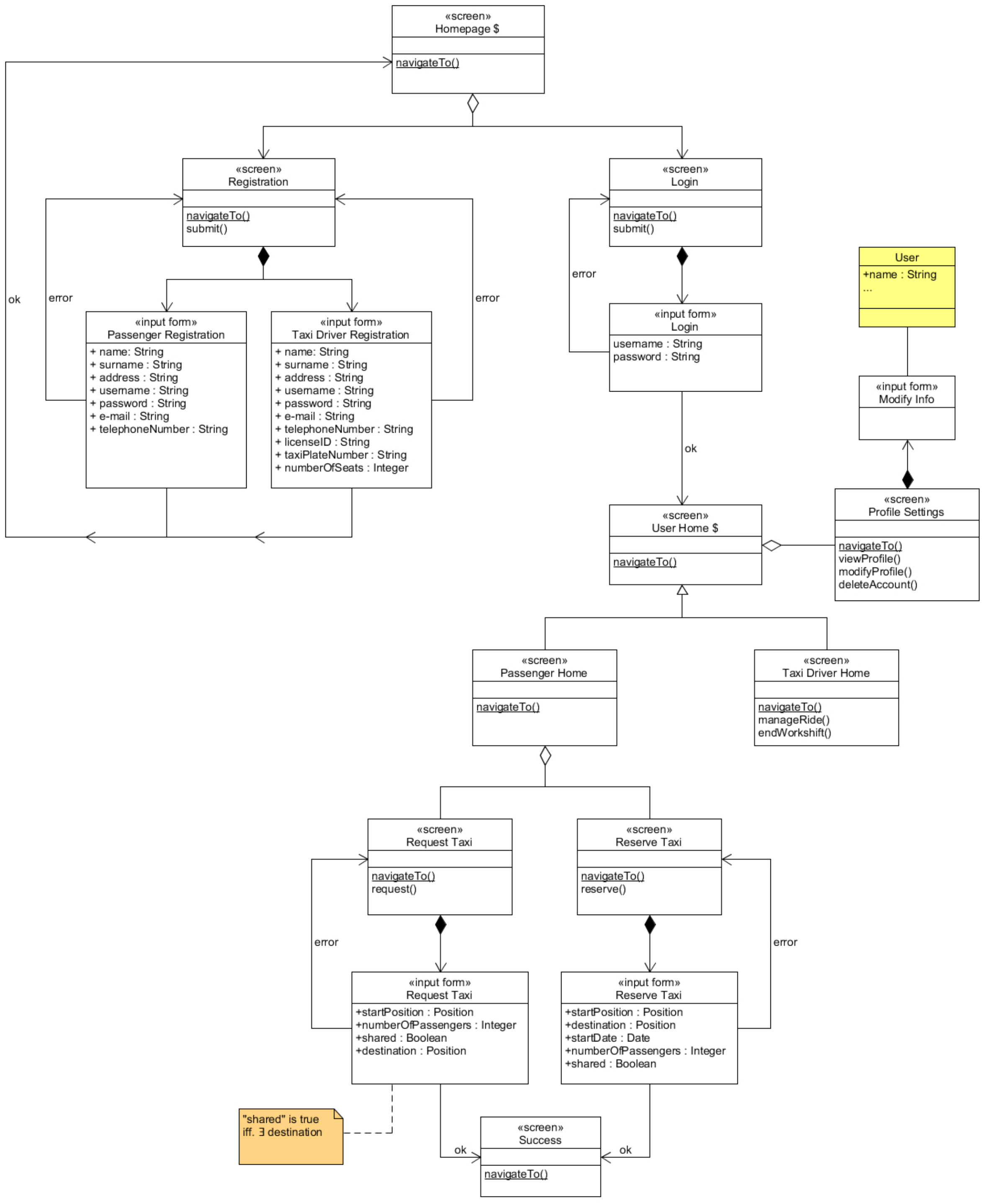
User Interface Design

4.1 Purpose

This chapter gives a description of the user interfaces of the system and the flow from one interface to another. The description includes an UX diagram and some mockups in order to understand how a user can do actions using the interface given by the system.

4.2 UX diagram

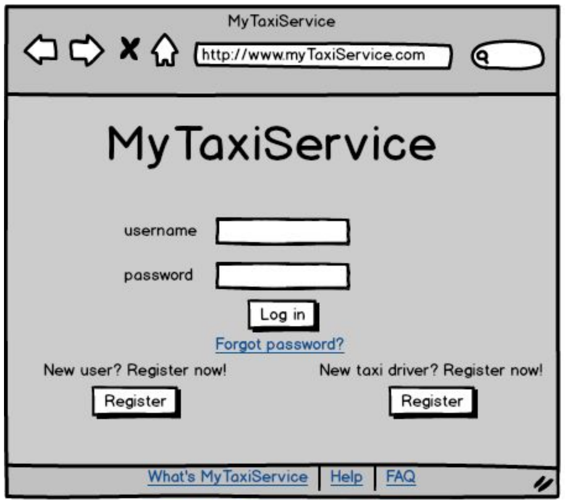
Here is the UX diagram:

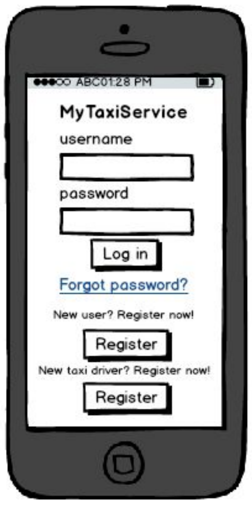

«screen» Homepage $
navigateTo()
«screen» Registration
navigateTo()
submit()
«screen» Login
navigateTo()
submit()
User
+name : String
...
ok
error
error
error
«input form» Passenger Registration
+ name: String
+ surname : String
+ address : String
+ username : String
+ password : String
+ e-mail : String
+ telephoneNumber : String
«input form» Taxi Driver Registration
+ name: String
+ surname : String
+ address : String
+ username : String
+ password : String
+ e-mail : String
+ telephoneNumber : String
+ licenseID : String
+ taxiPlateNumber : String
+ numberOfSeats : Integer
«input form» Login
username : String
password : String
«input form» Modify Info
ok
«screen» User Home $
navigateTo()
«screen» Profile Settings
navigateTo()
viewProfile()
modifyProfile()
deleteAccount()
«screen» Passenger Home
navigateTo()
«screen» Taxi Driver Home
navigateTo()
manageRide()
endWorkshift()
«screen» Request Taxi
navigateTo()
request()
«screen» Reserve Taxi
navigateTo()
reserve()
error
error
«input form» Request Taxi
+startPosition : Position
+numberOfPassengers : Integer
+shared : Boolean
+destination : Position
«input form» Reserve Taxi
+startPosition : Position
+destination : Position
+startDate : Date
+numberOfPassengers : Integer
+shared : Boolean
"shared" is true iff. ∃ destination
ok
ok
«screen» Success
navigateTo()

4.3 Mockups

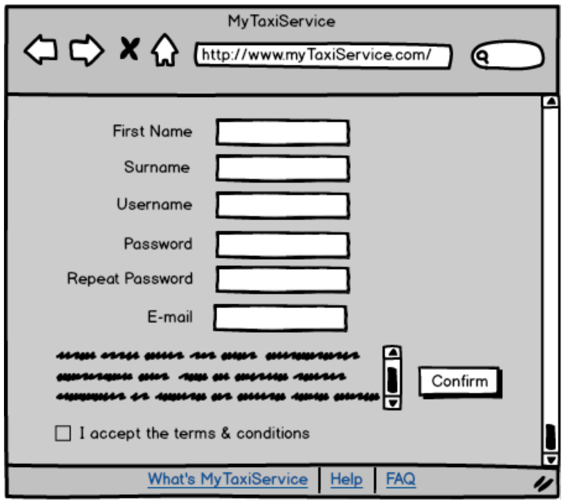
Here we will presented some mockups of MyTaxiService. Some of them referring to the RASD document Section 3.1.1 User Interface.

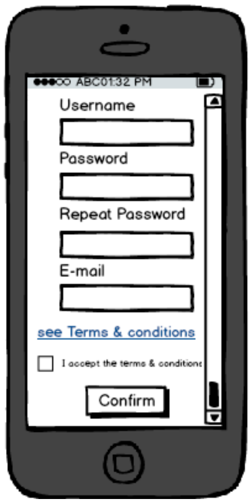
Log in: In the figure below is shown MyTaxiService's homepage



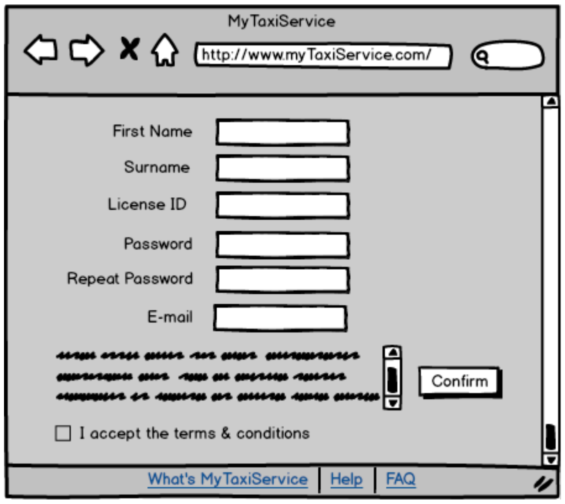


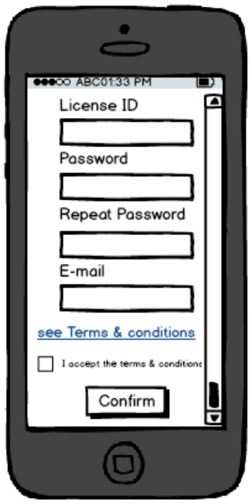
Registration passenger: View of the visitor that wants to register as a passenger



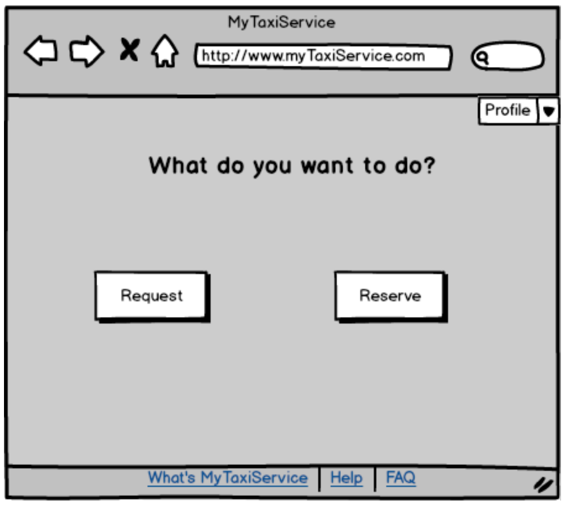


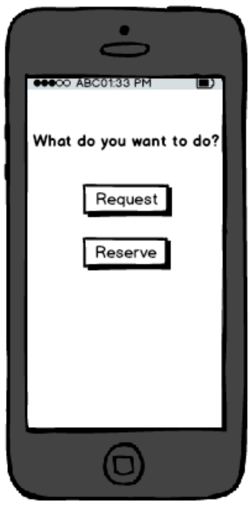
Registration taxi Driver: View of the visitor that wants to register as a taxi driver



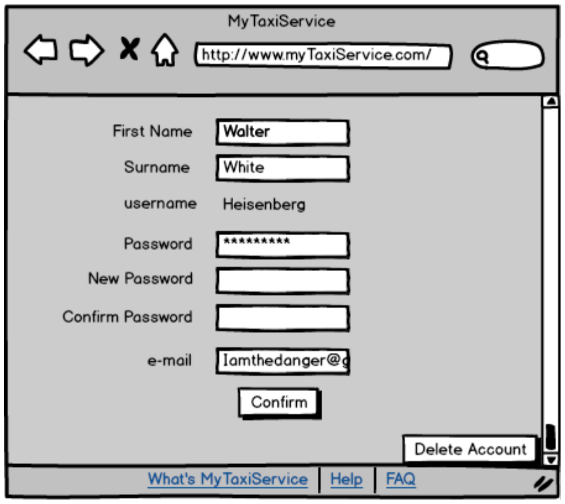


Passenger view: View of the passenger

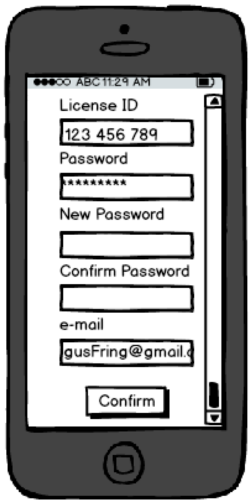




Profile: View of the profile of the user

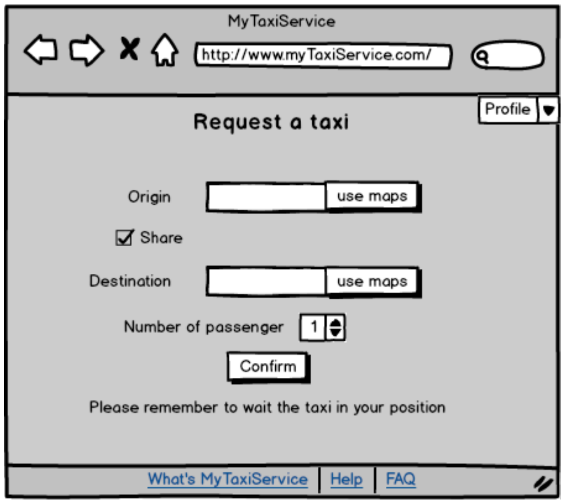


First Name Walter
Surname White
username Heisenberg
Password *********
New Password
Confirm Password
e-mail Iamthedanger@g

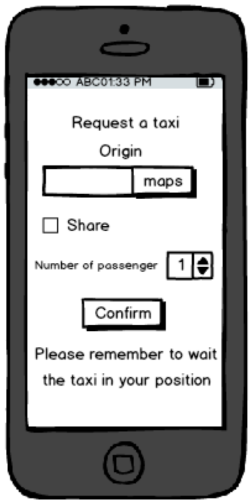


License ID 123 456 789
Password *********
New Password
Confirm Password
e-mail gusFring@gmail.c

Request a taxi: View of the passenger when he/she requests a taxi

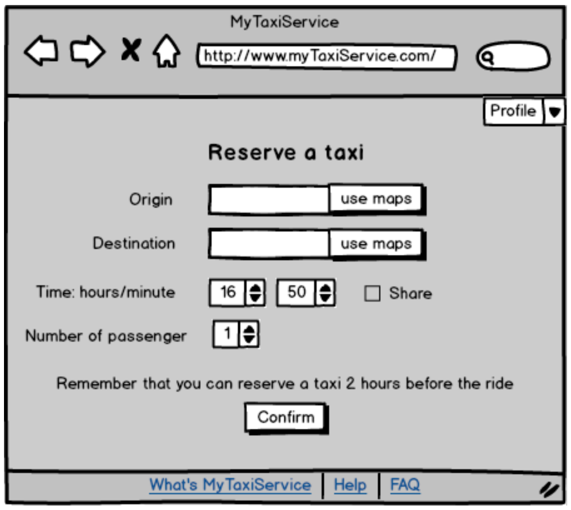


Request a taxi
Origin
☑ Share
Destination
Number of passenger 1
Please remember to wait the taxi in your position

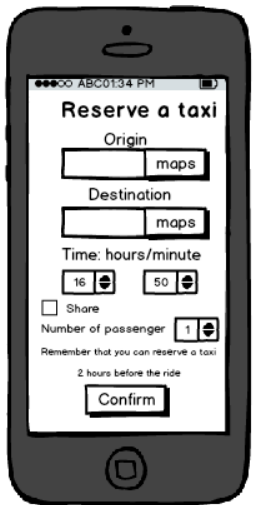


Request a taxi
Origin
☐ Share
Number of passenger 1
Please remember to wait the taxi in your position

Reserve a taxi: View of the passenger when he/she reserves a taxi

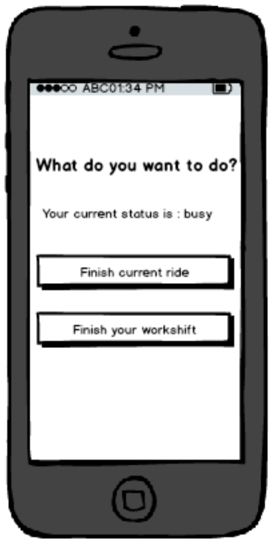


Remember that you can reserve a taxi 2 hours before the ride



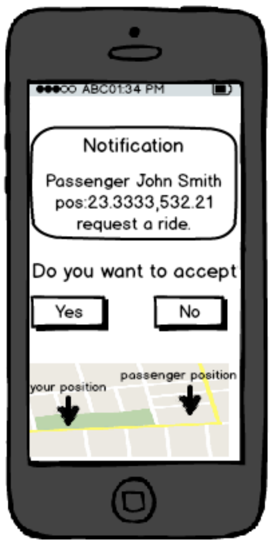
Remember that you can reserve a taxi 2 hours before the ride

Taxi driver view: View of the taxi driver

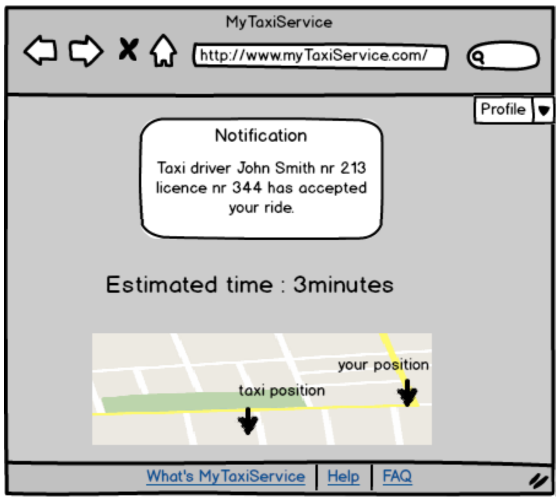


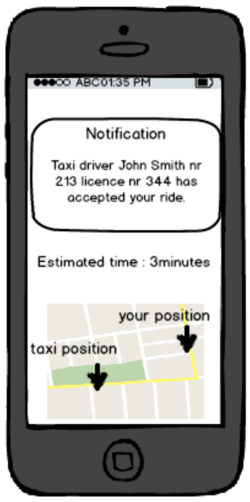
Your current status is : busy

Taxi driver notification: Notification that the taxi driver, choosen by the system, sees when a passenger request a ride.



Passenger notification : Notification that the passenger see when a taxi accept the ride


MyTaxiService
http://www.myTaxiService.com/
Profile
Notification
Taxi driver John Smith nr 213 licence nr 344 has accepted your ride.
Estimated time : 3minutes
your position
taxi position
What's MyTaxiService
Help
FAQ


●●●○○ ABC01:35 PM
Notification
Taxi driver John Smith nr 213 licence nr 344 has accepted your ride.
Estimated time : 3minutes
your position
taxi position

Chapter 5

Requirements Traceability

1. **Visitors can register either as passengers or as taxi drivers**
   Account manager permits visitors to be registered as passengers or taxi drivers during registration. User types have different functionalities. The clients applications use Account API.

2. **Visitors can abort the registration process at any time.**
   The applications permit user to abort registration process. No data is stored.

3. **The link in the confirmation email must be clicked within 1 day, otherwise the registration is deleted along with the visitor's info.**
   After the registration process is completed, Account manager takes care of sending the confirmation email to new user and if he/she clicks on link provided within 1 day, he/she will be capable to use myTaxiService, otherwise registration data will be deleted.

4. **Registration form requires user's info (fields). All fields must be contain valid inputs (see RASD for further informations)**
   Users data are stored in myTaxiDatabase. myTaxiservice can access data through myTaxiDatabase API.

5. **Email address and username cannot be the same as ones from other myTaxiService users**
   Account manager ensures the uniqueness of users through the control of same emails and usernames.

6. **Password must contain at least 8 characters**
   To ensure data security the user during the registration has to provide a password containing at least 8 characters.

7. **Password and password confirmation must match.**
   It requested to provide two times the same password to avoid typos.

8. **Visitors can log in either as passengers or as taxi drivers**
   Account manager allows access to service only to registered user.
9. **Visitors must fill the "username" field either with an existing username or with an existing email address in order to successfully log in.**
   The uniqueness of username and password permits user to log in with his/her username or password.
10. **Visitors must fill the "password" field with the only password corresponding to the submitted username/email address in order to successfully login**
    To access the service password is always requested
11. **The system will ignore log in requests if at least one of the "username" and "password" fields are left blank....**
12. **The system allows visitors to retrieve their password if they forget it, by clicking "Forgot password?"....**
13. **The system requires visitors to submit an existing email address in the "username" field in order to retrieve their password....**
14. **The system will take care of assigning the user a new password, when he/she states to have lost the previous one....**
15. **The system will take care of sending to the email address submitted by the visitor the new assigned password, when he/she states to have lost the previous one....**
16. **The system allows visitors to retrieve their password once a day....**
17. **The system allows standard taxi ride requests to passenger users....**
18. **The system allows standard taxi ride requests both on the web and on the mobile application....**
19. **The system allows taxi ride requests if and only if the passenger accepts to give info about his/her location, either through GPS or directly writing down a valid location....**

20. **The system allows taxi ride requests if and only if the passenger can be located in some definite position of some definite taxi zone....**

21. **The system uses default values for the number of passengers and sharing preferences of a ride (1 person, no sharing), unless the passenger does specify them....**

22. **The system uses a FIFO policy to manage forwarding of pending ride requests....**

23. **The system uses a FIFO policy to manage the order of taxi drivers in queues to send notifications to....**

24. **The system forwards a ride request to the first taxi driver in the considered zone queue if and only if he/she has a sufficient number of free seats available in his/her vehicle....**

25. **The system keeps the passenger(s) notified about the status of the ride request he/she sent....**

26. **Once a ride request has been accepted by some taxi driver, the system changes the request status from "Pending" to "Accepted"....**

27. **Once a ride request has been accepted by some taxi driver, the system calculates the ETA of the incoming taxi based on the distance between the taxi and the passenger(s), and the current traffic....**

28. **Once a ride request has been accepted by some taxi driver, the system notifies the passenger(s) about the ETA of the incoming taxi....**

29. **Once a ride request has been accepted by some taxi driver, the system keeps the passenger(s) notified about the current location of the incoming taxi, showing its position on a map....**

30. **Once a ride request has been accepted by some taxi driver, the system prevents the passenger(s) to make a new ride request until the taxi driver changes the status of the ride to "Completed"....**

31. **The system allows reserved taxi ride requests to passenger users....**

32. **The system allows reserved taxi ride requests both on the web and on the mobile application....**

33. **The system allows reserved taxi ride requests if and only if the passenger gives definite existing positions of some definite existing taxi zones both for "Origin" and "Destination" fields....**

34. **The system allows passengers to select locations either through GPS or directly writing down a valid location....**

35. **The system allows reserved taxi ride requests if and only if the passenger gives complete info about the date and the time of the meeting....**

36. **The system allows reserved taxi ride requests if and only if the time of the request occurs at least two hours before the ride meeting time....**

37. **The system forwards notifications to taxi drivers about reserved taxi ride requests 10 minutes before the ride meeting time....**

38. **The system uses default values for the number of passengers and sharing preferences of a reserved ride (1 person, no sharing), unless the passenger does specify them....**

39. **The system uses a FIFO policy to manage forwarding of pending ride requests....**

40. **The system uses a FIFO policy to manage the order of taxi drivers in queues to send notifications to....**

41. **The system forwards a reserved ride request to the first taxi driver in the considered zone queue if and only if he/she has a sufficient number of free seats available in his/her vehicle....**

42. **Once a reserved ride request has been accepted by some taxi driver, the system changes the request status from "Pending" to "Accepted"....**

43. **Once a reserved ride request has been accepted by some taxi driver, the system calculates the ETA of the incoming taxi based on the distance between the taxi and the passenger(s), and the current traffic....**

44. **Once a reserved ride request has been accepted by some taxi driver, the system notifies the passenger(s) about the ETA of the incoming taxi....**

45. **Once a reserved ride request has been accepted by some taxi driver, the system keeps the passenger(s) notified about the current location of the incoming taxi, showing its position on a map....**

46. **Once a reserved ride request has been accepted by some taxi driver, the system prevents the passenger(s) to make a new ride request until the taxi driver changes the status of the ride to "Completed"....**

47. **The system allows taxi ride sharing to passenger users....**

48. **The system allows taxi ride sharing both on the web and on the mobile application....**

49. **The system allows both standard shared taxi ride requests and reserved shared taxi ride requests, simply ticking the "Share" checkbox while doing any any kind of taxi ride request....**

50. **Both kinds of shared ride requests require no less data than their non- shared forms (see also 3.2.3, "Standard ride request" and 3.2.4, "Reserved ride request")....**

51. **The system allows standard shared taxi ride requests if and only if the passenger also gives a definite existing position of some definite existing taxi zone as "Destination" (reserved taxi ride requests already require this, even if non-shared)....**

52. **The system allows passengers to select the input for "Destination" either through GPS or directly writing down a valid location....**

53. **The system, at the time of forwarding any shared ride request, will use a special algorythm to calculate good arrangements between different shared ride requests....**

54. **The arrangement algorythm search different shared ride requests to form a single, grouped, shared ride request....**

55. **The arrangement algorythm considers for grouping only shared rides with close "Origin" positions and similar directions towards "Destination" po- sitions. Similar directions means that going from the origin to the farthest destination implies passing by the other destination(s) as well....**

56. **The arrangement algorythm also considers the total number of passengers in a taxi. Groups that would occupy too many seats are not eligible....**

57. **The arrangement algorythm can group even standard and reserved ride requests together, as long as all the others requirements are met....**

58. **The system will consider group requests as a single one when forwarding them to taxi drivers....**

59. **Taxi drivers can see all details of all shared requests when receiving noti- fication of a grouped request....**

60. **Taxi drivers can accept or refuse grouped requests as normal....**

61. **The system will automatically split up groups of shared requests if they're continuously refused, or simply not accepted, for 3 minutes since their forwarding....**

62. **The system will then proceed to recalculate possible groups with the spe- cial algorythm, but excluding the arrangements that were already tried....**

63. **The system calculates how to split taxi fees equally on all passengers on a shared ride, depending on the distance traveled by each one and the number of passengers during each part of the ride....**

64. **The system notifies the taxi driver how much of the total fee each passenger will have to pay, in percentage....**

65. **The system allows taxi drivers to receive ride request notifications on their mobile phone application and respond to them, either accepting or refusing....**

66. **The system notifies taxi drivers about all request notifications forwarded to them....**

67. **The system uses a FIFO policy to manage forwarding of pending ride requests....**

68. **The system uses a FIFO policy to manage the order of taxi drivers in queues to send notifications to....**

69. **The system forwards a ride request to the first taxi driver in the considered zone queue if and only if he/she has a sufficient number of free seats available in his/her vehicle....**

70. **Taxi zone queues contain only taxi drivers that currently have status "Ready"....**

71. **Taxi zone queues contain only taxi drivers that are currently located in that taxi zone....**

72. **Taxi ride notifications show all requesting passengers' username and position on a map....**

73. **The system gives taxi drivers one minute to accept or refuse request noti- fications, otherwise take it as a refusal. This is to avoid long waiting times for passengers....**

74. **Once a ride request has been accepted by some taxi driver, the system changes his/her status to "Busy" and he/she's removed from his/her taxi zone queue....**

75. **The system allows taxi drivers to notify the end of the ride, when they're doing one (i.e. when they accepted a ride request)....**

76. **Once a taxi driver notifies the end of a ride, the system changes his/her status to "Ready" and he/she's put on the bottom of his/her taxi zone queue....**

77. **The system uses a FIFO policy to manage taxi zone queues....**

78. **The system uses info provided by the GPS to locate taxis and decide their respective queues....**

79. **The system automatically inserts taxi drivers in queues when their status changes to "Ready"....**

80. **The system automatically removes taxi drivers from queues when their status changes to "Busy" or "Offline"....**

81. **The status automatically changes to "Busy" when the taxi driver accepts a ride request....**

82. **The status automatically changes to "Ready" when the taxi driver notifies the end of a ride....**

83. **When status is "Ready", the application notifies about ride requests....**

84. **When status is ”Ready”, the application enables the taxi driver to accep- t/refuse requests....**

85. **When status is ”Busy”, the application prevents ride requests notifica- tions....**

86. **When status is ”Busy”, the application enables the taxi driver to notify the end of the current ride....**

87. **Account settings are available to both passenger and taxi driver users....**

88. **Account settings are available both on the web and the mobile application....**

89. **Account settings are accessible from the start screen of both apps, through the ”Profile” button....**

90. **The system allows users to view all their profile info, submitted during registration (see also 3.2.1, ”Registration”)....**

91. **The system allows users to modify all their profile info, submitted during registration, with the only exception of username....**

92. **Modifying the password requires to write the old one, and the new one twice; if the former password is not correct or if the two new passwords submitted do not match, the system asks for all passwords again and notifies the user....**

93. **Modifying the email address, the taxi license ID or the taxi code requires that the new one doesn’t match with the one of another registered user....**

94. **Modifying the email address requires confirmation through an email sent to the submitted email address....**

95. **The system allows users to abort modifications at any time....**

96. **The system allows users to delete their account: confirmation is required to proceed....**

Chapter 6

References

Appendix A

Appendix

A.1 Software and tool used

- LaTeX (http://www.latex-project.org/) : to redact and to format this document
- Balsamiq Mockups (http://balsamiq.com/products/mockups/): to create mockups
- Eclipse Luna (https://eclipse.org/luna/): to draw global use case, class diagrams, sequence diagrams and UX diagram
- StarUML (http://staruml.io/): to draw UML diagrams

A.2 Working hours

This is the time spent for redact the document

- Belluschi Marco : xx hours
- Cerri Stefano : xx hours
- Di Febbo Francesco : troppe hours