



POLITECNICO MILANO 1863

Politecnico di Milano
A.A. 2015/2016
Software Engineering 2: “*myTaxiService*”

Design Document

Alessandro Baldassari (mat. 841561)
Alberto Bendin (mat. 841734)
Francesco Giarola (mat. 840554)

November 26, 2015

Contents

	Page
1 Introduction	1
1.1 Purpose	1
1.2 Scope	1
1.3 Definitions, Acronyms, Abbreviations	1
1.4 Reference Documents	2
1.5 Document Structure.....	2
2 Architectural Design	2
2.1 Overview	2
2.2 High level components and their interaction	3
2.2.1 Identifying sub-systems.....	3
2.2.2 General Package design.....	4
2.3 Component view.....	5
2.3.1 Design Level Model.....	5
2.3.2 Detailed Package Design	6
2.4 Deployment view	8
2.5 Runtime view	8
2.6 Component interfaces	8
2.7 Selected architectural styles and patterns	8
2.8 Other design decisions	8
3 Algorithm design	8
4 User Interface Design	8
4.0.0.1 Sign-up	9
4.0.0.2 Login.....	10
4.0.0.3 Call a taxi	11
4.0.0.4 Plan a ride	12
4.0.0.5 Your reservations	13
4.0.0.6 Your profile.....	14
4.0.0.7 Manage jobs.....	15
4.0.0.8 Administrator panel	16
5 Requirements Traceability	17
6 References	17
7 Appendix	17
7.1 Software and tools used.....	17
7.2 Hours of work.....	17

1 Introduction

1.1 Purpose

This document presents the architecture on which *myTaxiService* will be developed; it describes the decisions taken during the design process and justifies them. The whole design process is described including also the improvements and modifications to provide additional valuable informations in case of future changes of the architecture structure.

1.2 Scope

Accordingly to the definition of the architecture design this document will focus on the non functional requirements of *myTaxiService*. Since the system architecture defines constraints on the implementation this document will be used to provide fundamental guidelines in the development phase of *myTaxiService*.

The system architecture will be organized in three categories corresponding to the functionalities of the different users: passengers, drivers and administrator.

- **Passengers**
The system allows the user to sign up, login, request or reserve a taxi with the customization of the ride.
- **Taxi drivers**
The system allows the user to login, set the “availability” and accept or reject jobs.
- **Administrator**
The system allows the user to login, manage the drivers’ list and supervise the system itself.

1.3 Definitions, Acronyms, Abbreviations

The following acronyms are used in this document:

- JEE: Java Enterprise Edition
- RASD: Requirements Analysis and Specification Document
- ER: Entity Relationship
- EIS: Enterprise Information System
- API: Application Programming Interface

The following definitions are used in this document:

- Thin client: is a computer that depends heavily on another computer (its server) to fulfill its computational roles.
- Fat server: is a type of server that provides most of the functionality to a client’s machine within a client/server computing architecture.
- Event-driven architecture: is a software architecture pattern promoting the production, detection, consumption of, and reaction to events, that is a change of state of an object.
- Availability: is the “status” of a taxi driver, s/he can be available to take care of new jobs, thus the system should forward compatible requests, or can be unavailable, or occupied, busy, meaning that the taxi is temporarily off line with respect to the system, meaning that the system should not consider that taxi for requests assignment. The availability is set to “off” also when the taxi is carrying passengers.

1.4 Reference Documents

- Specification document: myTaxiService project
- Template for the Design Document
- IEEE Std 1016-2009 - IEEE Standard on Design Descriptions
- Requirements Analysis and Specification Document for *myTaxiService*

1.5 Document Structure

This document specifies the architecture of the system using different levels of detail. It also describes the architectural decisions and justifies them. The design is developed in a top-down way, then the document reflects this approach. The document is organized in the following sections:

1. Introduction
Provides a synopsis of the architectural descriptions.
2. Architectural design
3. Algorithm design
4. User interface design
5. Requirements traceability
6. References

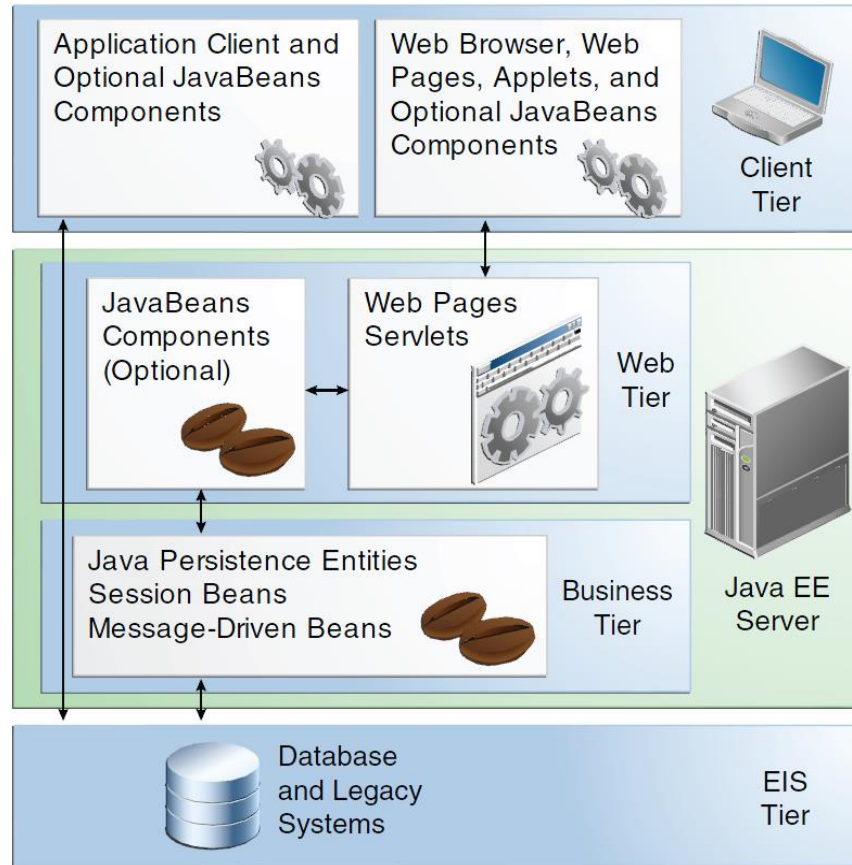
2 Architectural Design

2.1 Overview

This section of the design document provides a general description of the design of the system and its processes; it includes the general design context, the general approach and describes the overall design.

The architectural style adopted by *myTaxiService* is the well-known “client-server”, where the client is a “thin client” and the server is a “fat server”. The architecture is “event-driven”, thus the interaction between the components takes place through events, in other words asynchronous triggers.

JEE has a “four-tier” architecture divided as shown in the picture below:



1. **Client Tier:** contains Application Clients and Web Browsers and it is the layer designed to interact directly with the actors. *myTaxiService* is a web and mobile application, then the client will use a web browser or a smartphone to access the pages.
2. **Web Tier:** contains the Servlets and Dynamic Web Pages that need to be elaborated. This tier receives the requests from the client tier and forwards the pieces of data collected to the business tier waiting for processed data to be sent to the client tier, eventually formatted.
3. **Business Tier:** contains the Java Beans, which contain the business logic of the application, and Java Persistence Entities.
4. **EIS Tier:** contains the data source. In our case, it is the database allowed to store all the relevant data and to retrieve them.

When the second and third tier are considered together the architecture becomes a “three-tier” with *client tier*, *business logic tier* and *persistence tier*.

To design the system a top-down approach is used. After the identification of the main three layers, the system is decomposed in components that capture subsets of related functionalities. For each component is specified the role in the architecture and its interactions with the rest of the system.

2.2 High level components and their interaction

2.2.1 Identifying sub-systems

The functionalities of *myTaxiService* are divided into these functional areas:

- *myTaxiService*

This component describes the system that is going to be developed.

- Users

This component represents all the users that will use the service.

- Notification service

This component represents the notification mechanism that the system will use (text-messages, emails and push notifications).

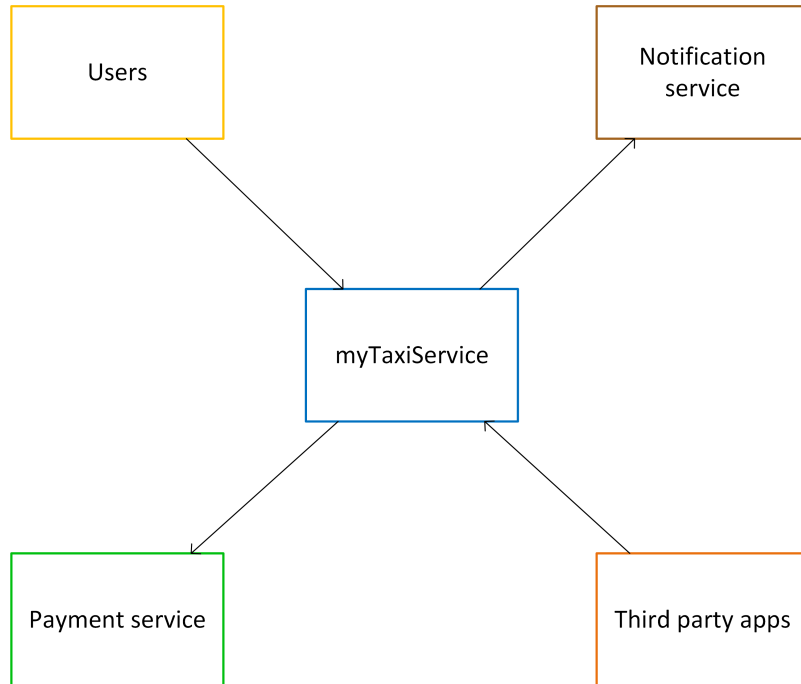
- Payment service

This component represents the external service in charge of managing the payments.

- Third party apps

This component represents generally the external apps that can possibly connect to the system via our public web APIs.

The following schema shows the above-mentioned components and their interaction.



The “notification service”, “payment service” and “third party apps” are external components that are not part of the *myTaxiService* system, therefore they will not be discussed.

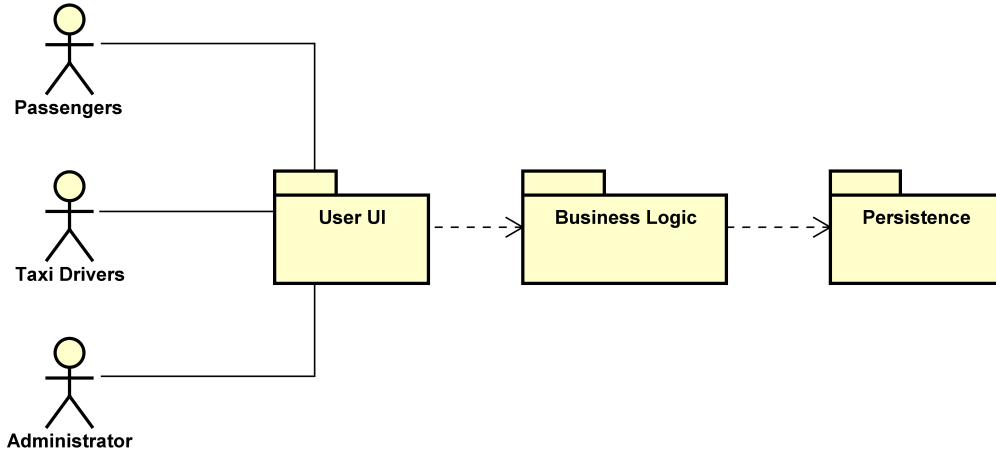
2.2.2 General Package design

Considering the three-tier view of the architecture three packages are identified:

- User UI: this package is in charge of interacting with the user; it receives the user requests, sends these to the business logic package, obtains the information needed from the latter and displays them to the user accordingly. In general the package contains the user interfaces.

- Business logic: this package is in charge of receiving and processing the User UI's package requests, accessing the Persistence package when needed and sending a response accordingly.
- Persistence: this package is in charge of managing the data requests from the Business logic package.

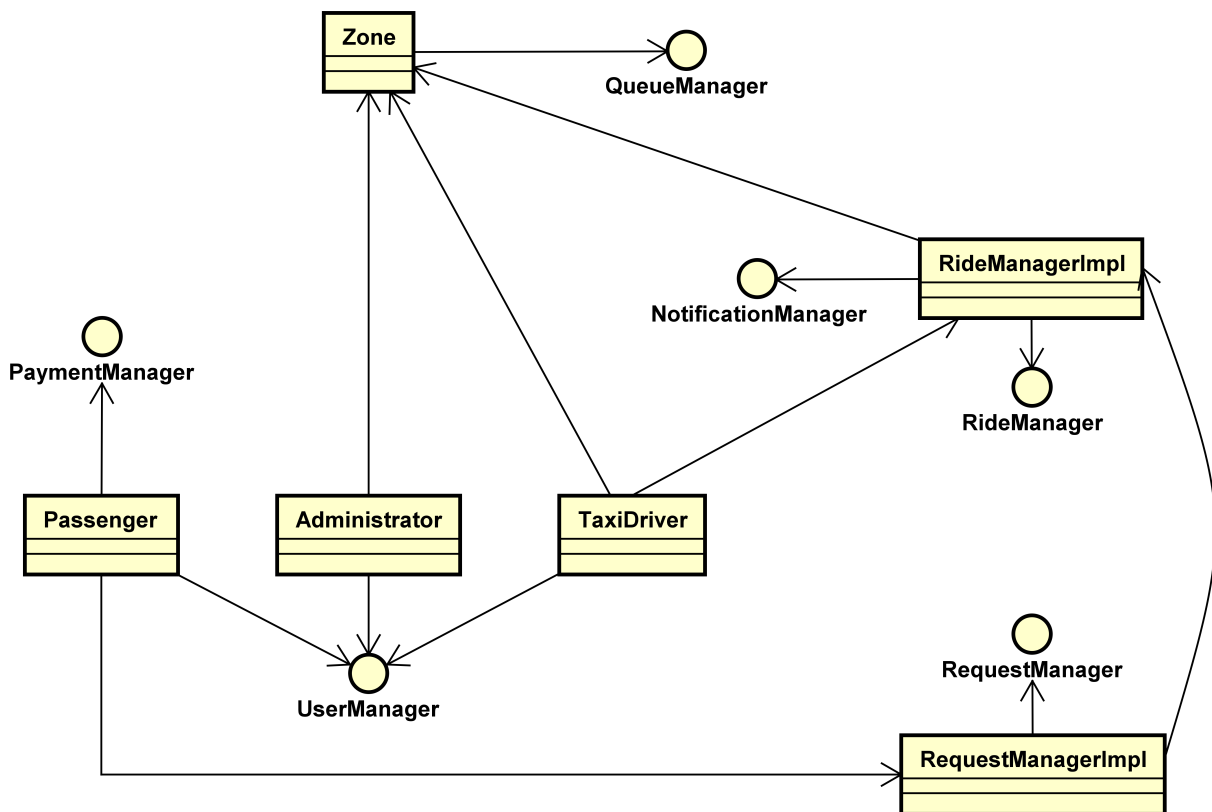
The main users: Administrator, passengers and taxi drivers directly access the User UI package but cannot see the other packages, as shown in the picture below.



2.3 Component view

2.3.1 Design Level Model

The picture below represents the main components and interfaces of *myTaxiService*.



From the diagram above it is evident that:

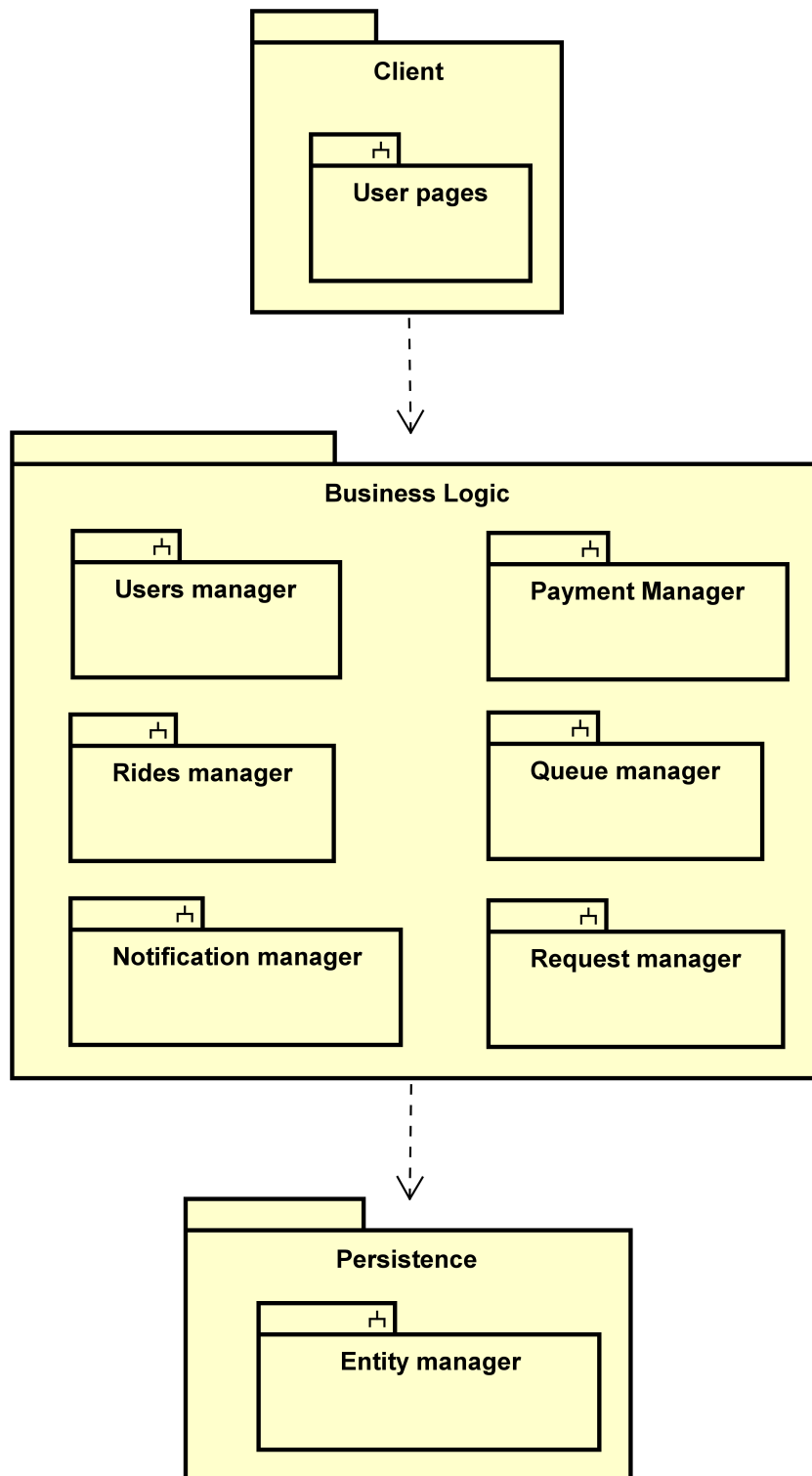
- ⇒ The “*UserManager*” interface contains all the information related the every user and his/her profile; it also offers the various “login”, “sign-up”, “changePassword”, “changeNotificationType” methods. It also enables the administrator to manage the drivers list.
- ⇒ The “*PaymentManager*” is the interface in charge of exchanging the information about the transaction with the payment service provider.
- ⇒ The passenger is able to make a request or reservation through the interface “*RequestManager*” that is in charge of receiving a complete and customized ride request.
- ⇒ The “*RideManager*” interface communicates with the “*RequestManager*” to prepare the physical ride; it communicates with the “*QueueManager*” interface to be able to select to which taxi the jobs are proposed and with the “*NotificationManager*” to notify all the users involved in ride.
- ⇒ The “*QueueManager*” interface is accompanied by a “*Zone*” class to control the sections of the city and their queue of available taxis.
- ⇒ The three types of users are represented by three respective classes with the same name: “*Passenger*”, “*TaxiDriver*” and “*Administrator*”.
- ⇒ The “*RequestManagerImpl*” and “*RideManagerImpl*” classes accompany their respective interfaces and keep track of the history of rides and request for every users.
- ⇒ The connection between the “*Administrator*” and the “*Zone*” class enables the administrator to supervise the situation of all the queues in real-time.
- ⇒ The connection between the “*TaxiDriver*” and the “*Zone*” and “*RideManagerImpl*” classes enables the system to keep track of the availability of the taxi drivers and allows the drivers to accept or reject job proposals.

2.3.2 Detailed Package Design

The inner packages are described as follows:

- User UI: this set of sub-packages is responsible for encapsulating the user’s actions and forwarding information requests to the Business Logic sub-packages.
- Business logic: this set of sub-packages is responsible for handling requests from the User UI package, processing them and sending back a response. These packages may access the Persistence package.
- Persistence: this set of sub-packages contains the data model for the system. It accepts requests from the Business Logic package.

A more detailed view of the system:



2.4 Deployment view

2.5 Runtime view

2.6 Component interfaces

2.7 Selected architectural styles and patterns

The decision of using the classical client-server architecture was straightforward. In a system like *myTaxiService* it is important to have one single point of failure (the server) and control, with respect to distributed architectures. This decision is also compatible with the fact of having only one centralized database which sustains the whole service.

For the type of interaction with clients required by the system, the “thin client” approach is employed; only a very simple layer of logic is implemented on the client side, for instance the acquisition of coordinates with the GPS sensor and the rendering of the graphical user interface. *myTaxiService* relies on an “event based” or “event-driven” architecture. The system reacts to the change of state of some objects, for instance the switch of “availability” for the taxi driver is detected by the zone manager that operates accordingly in the respective queue. Other examples could be the change of state of a request to a modified request that must be taken into account (eg. cancellation of the request), or the consequence of the approval/refusal of a job by a driver. All of these message-events are carried out by the notification manager as notifications.

Decisions list:

- patterns

2.8 Other design decisions

3 Algorithm design

Algorithms list:

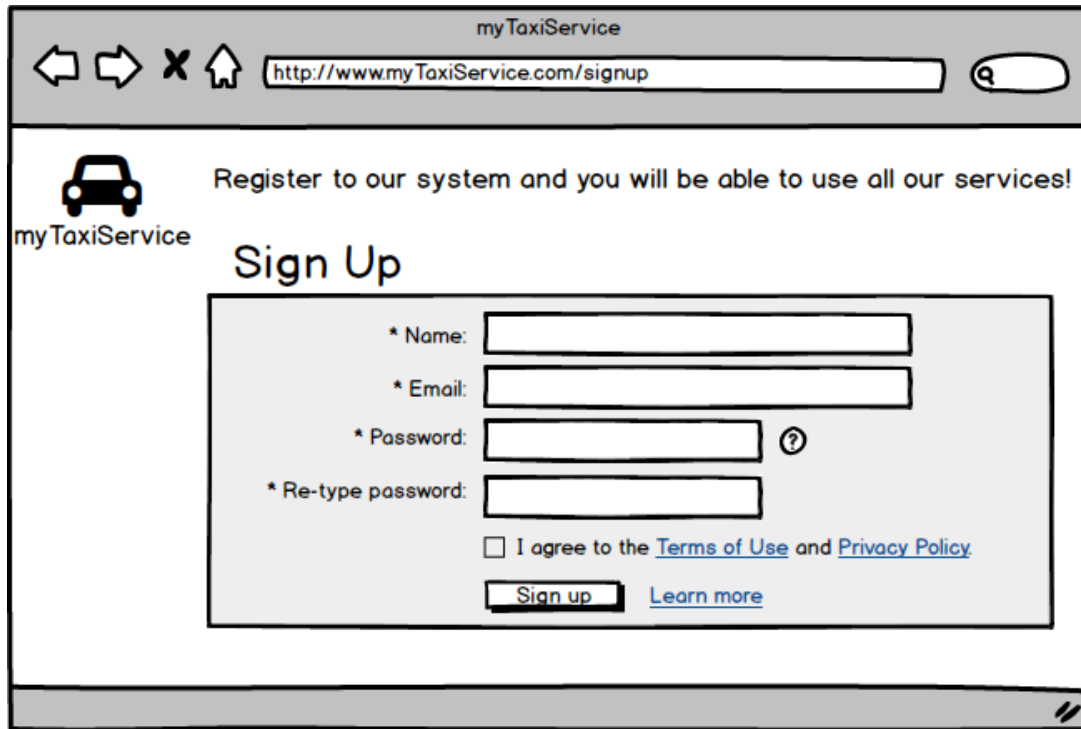
- Queue managing when driver accept/rejects a job
- fee managing

4 User Interface Design

This section is the same featured in the RASD.

Shown below are some mock-ups that preview the user interface of the main features the system shall provide.

4.0.0.1 Sign-up This page presents the sign up form for the clients. The user will need to provide his/her personal data, home and email addresses, phone number, favorite payment option, favorite notification type.



The image shows a web browser window with the title "myTaxiService". The address bar contains "http://www.myTaxiService.com/signup". The page content includes a car icon and the text "myTaxiService". Below this is the heading "Sign Up" and a subheading "Register to our system and you will be able to use all our services!". The form fields are: * Name: (text input), * Email: (text input), * Password: (text input with a question mark icon), and * Re-type password: (text input). Below the fields is a checkbox labeled "I agree to the [Terms of Use](#) and [Privacy Policy](#)". At the bottom of the form are two buttons: "Sign up" and "Learn more".



The image shows a mobile app interface on a smartphone. The status bar at the top displays "ABC" and "04:44 PM". The app header features a car icon and the text "myTaxiService". Below the header is the heading "Sign up". The form fields are: Name:* (text input), Password:* (text input), and Email:* (text input). At the bottom of the form is a button labeled "Sign up".

4.0.0.2 Login This page shows the login form for the final users.

A web browser window titled "myTaxiService" with a search bar containing "http://www.myTaxiService.com/login". The page features a car icon and the text "myTaxiService". A message reads: "Login to our system and you will be able to use all our services!". Below this is a "Welcome" dialog box with a help icon. The dialog contains two input fields labeled "Username" and "Password", and a "Login" button.

A mobile app interface for "myTaxiService" showing a "Login" screen. The screen includes a car icon, the app name, and a hamburger menu icon. The login form has two fields: "Name:*" and "Password:*", each with a corresponding input box. A "Login" button is positioned at the bottom of the form. The status bar at the top shows "ABC" and "04:44 PM".

4.0.0.3 Call a taxi The user can ask for a taxi providing the pickup location through a complete address or through the automatic detection of his/her location thanks to the browser or the GPS. The user can also choose from an history of “recent addresses”.



4.0.0.4 Plan a ride The user can plan a ride in advance providing the pickup and drop off locations, the date and time, the willingness to share the ride and the number of passengers.

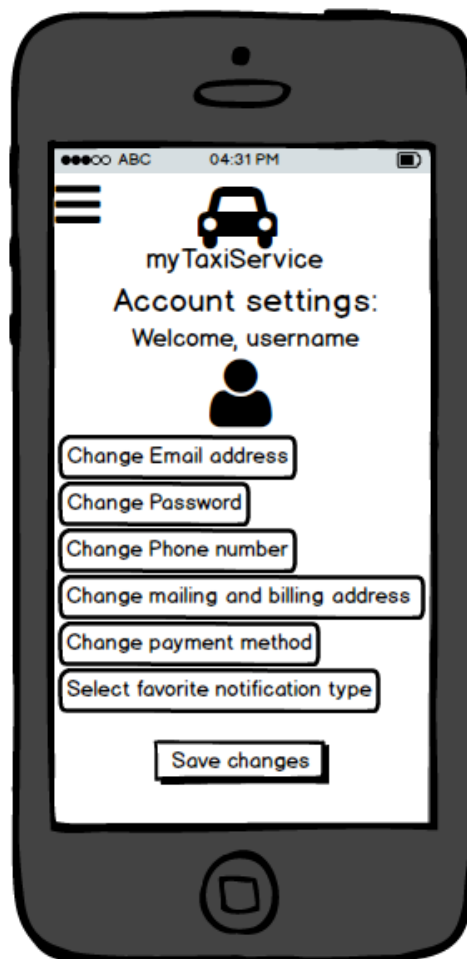
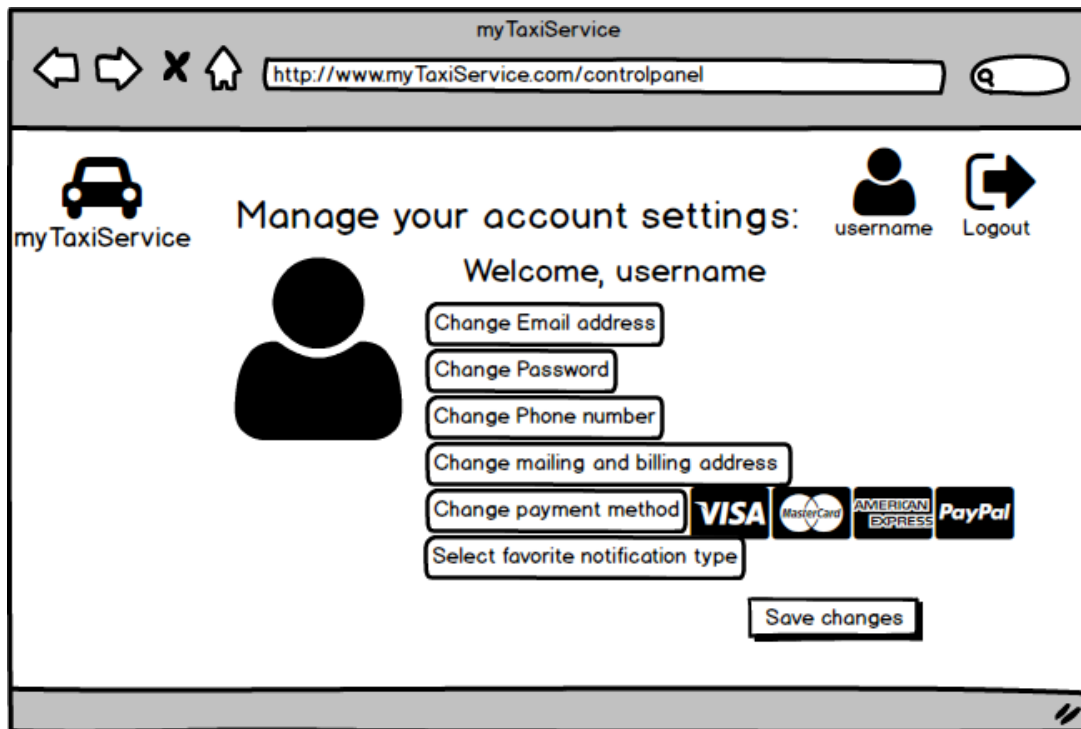
The image shows a web browser window for 'myTaxiService'. The address bar displays 'http://www.myTaxiService.com/bookride'. The page has a navigation bar with 'Request a taxi', 'Book a ride' (highlighted), and 'Your reservations'. A user profile icon is labeled 'username' and a 'Logout' button is present. The main heading is 'Make a reservation:'. The form includes fields for 'Pickup Address:*' (with a text input 'Enter a location' and a 'Recent Addresses' dropdown), 'Drop off Address:*' (with a text input 'Enter a location' and a 'Recent Addresses' dropdown), 'Date:*' (with a 'dd/mm/yyyy' input), 'Time:*' (with a '12:00' input), and 'Passengers:*' (with a '1' input). A checkbox 'Include shared rides' is checked. A map icon is visible on the right side of the form.

The image shows a mobile app interface for 'myTaxiService'. The status bar at the top shows 'ABC' and '11:14 AM'. The app has a hamburger menu icon on the left. The main heading is 'Make a reservation:'. The form includes fields for 'Pickup Address:*' (with a text input 'Enter a location' and a location pin icon), 'Drop off Address:*' (with a text input 'Enter a location'), 'Date:*' (with a 'dd/mm/yyyy' input), 'Time:*' (with a '12:00' input), and 'Passengers:*' (with a '1' input). A checkbox 'Include shared rides' is checked.

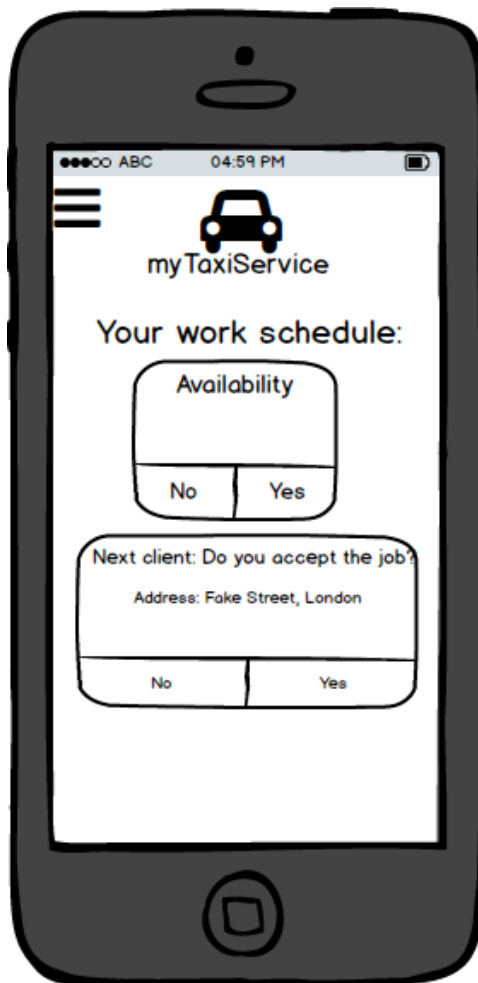
4.0.0.5 Your reservations The user can see both the active reservations and the past ones. S/he can edit the active reservations within the established time frame before the meeting time.



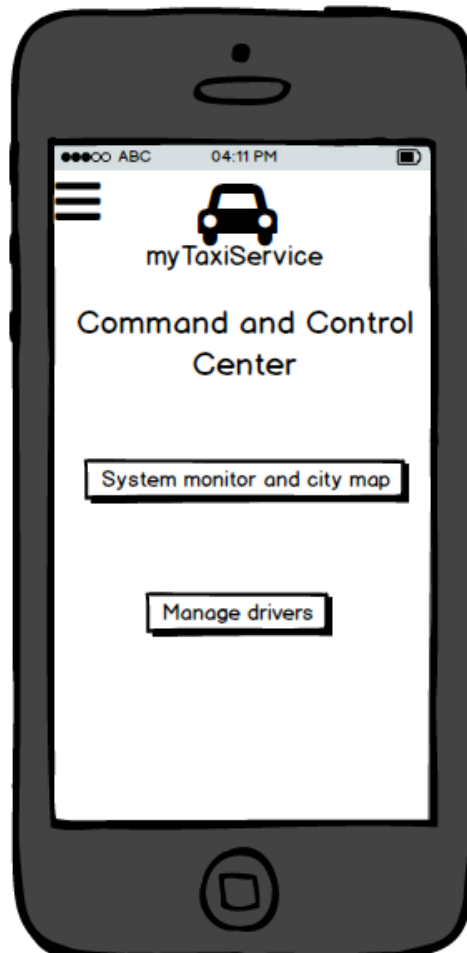
4.0.0.6 Your profile The user can edit the personal profile modifying the password, phone number, email address, permanent address, payment method and notification type.



4.0.0.7 Manage jobs The taxi driver's home page: s/he can accept or reject the requests forwarded by the system and set her/his own "availability".



4.0.0.8 Administrator panel The administrator's home page: s/he can monitor the whole system or manage taxi drivers inserting new employees in the system DB or modifying/deleting existing ones.



5 Requirements Traceability

6 References

7 Appendix

7.1 Software and tools used

- TeXstudio 2.10.4 (<http://www.texstudio.org/>) to redact and format this document.
- Astah Professional 7.0 (<http://astah.net/editions/professional>): to create Use Cases Diagrams, Sequence Diagrams, Class Diagrams and State Machine Diagrams.
- Microsoft Office Visio Professional 2016

7.2 Hours of work

The time spent to redact this document:

- Baldassari Alessandro: 35 hours.
- Bendin Alberto: 35 hours.
- Giarola Francesco: 35 hours.