myTaxyService®

Design Document

Matteo Greselin matr. 773600 ,
Seyeed Mohsen Kashfi Haghighi matr.859085

Version 1.0

Contents

1. Introduction

1.1 Document Description

In this document we focused on the design of myTaxiService® . Our goal is the interaction between users and the system. In this document the Architecture, Algorithm and User Interface of myTaxiService® will be provided with their descriptions.
Also this software design document will focus on the Taxi Service Management System.

1.2 Scope

The scope of this document is to provide Technical Design Document for the project myTaxiService® which is an online Taxi Service Management System. Bellow some scopes are explained.

1. Design Information detail should be well explained for the development phase.
2. There should not be any ambiguity in the information.
3. Representation of architectural system and data should be done in detail.
4. Representation of User interface and Algorithm Design of the system is concerned.

1.3 Document Structure

This document have three main chapters.
Chapter 2, Architectural design will be presented in details with sufficient subsections. Chapter 3, we will focus on algorithmic part, in our case will be Queue algorithm.
Chapter 4, the design of our user interface will be explained and shown more than what we have done in RASD. At the end of this document there will be some clarification about the consistency between RASD and DD.
Also in different parts of our document we will propose some diagrams such as Sequence diagrams or BCE diagrams. We should divide the entire process of myTaxyService® into sub-systems in order to be clear.

1.4 Glossary

- JEE : Java Enterprise Edition
- DB : Data Base
- DBMS : Data Base Management System
- RASD : Requirements And Specification Document
- TD : Taxi Driver
- TDs : Taxi Drivers

1.5 Reference Document

IN THIS DOCUMENT we refer to the following documentation:

- IEEE Std 1016™-2009 (Revision of IEEE Std 1016-1998) Standard for Information Technology—Systems Design— Software Design Descriptions
- International Standard ISO/IEC/IEE 42010 - First edition 2011-12-01 - Systems and software engineering — Architecture description
- Template For The Design Documentprovided by prof. Di Nitto
- RASD of myTaxyService® created by Kashfi Haghighi and Greselin

2. Architectural Design

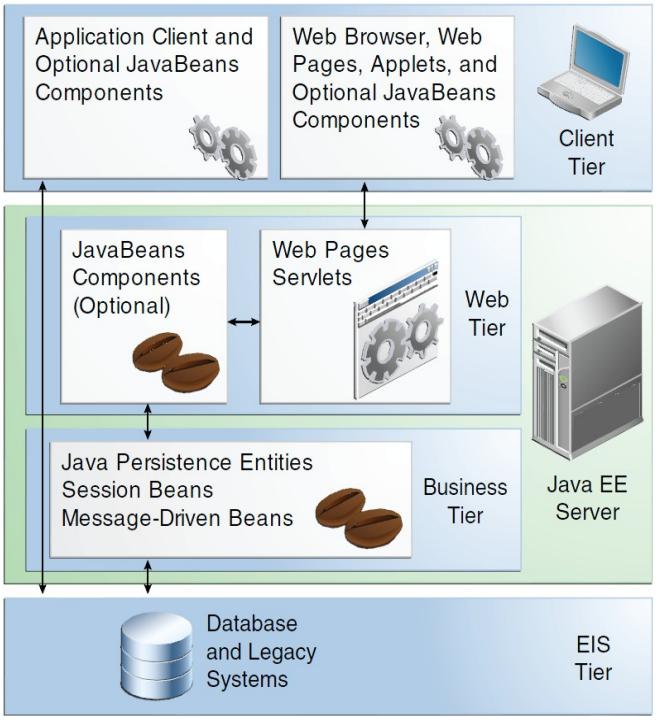
2.1 Architecture Overview

THE ARCHITECTURE of myTaxiService® will be based on JEE Architecture that has a four tiered architecture divided as:

- ***Client Tier:*** it contains Application Clients and Web Browsers and it is the layer that interacts directly with the actors. As far as our project is a web application and Mobile application the client will use a web browser and an application to access pages respectively.
- ***Web Tier:*** it contains the Servers and the Dynamic Web Pages that needs 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
- ***Business Tier:*** it contains the Java Beans that contain the business logic of the application, and Java Persistence Entities.
- ***EIS Tier:*** it contains the data source. In our case it is the database allowed to store all the relevant data and to retrieve them.

We have chosen a 4-Tier architecture in order to increase scalability and security, support multiple protocols and obviously different client and resources. All these things are possible with the linear relation between element in this type of architecture.

In the next image we can see the fundamental structure of JEE.



2.2 High Level Components

•N AN HIGH LEVEL description we are going to explain principal interaction between entity
1 of our application.
In the client side we could have four types of interface:

1. Guest
2. Passenger
3. Taxi Driver
4. Administrator

First one should be only represent general functions, such as informition about service, seeing system situation (zones, queue, waiting time..) or registration.
Second one should perform more funtionalities, such as submit request or reservation, receive messages from admin or confirmation request (or a reservation), see reports about usage of application relatively to own account and do actions that a guest are allowed to do
Third one should recieve request (or reservation) from system, decide to accept or refuse it, recive messages from admin, set availability or unavailability option:
Fourth must be principally a management interface: admin could not request or reserve a taxi but could send messages, delete user, edit datas or see reports

In the server side there is a system that process each command which arrive from client side. In fact it should be able to recieve requests or reservations from passengers, forward them to taxi driver, should be able to recieve decision from taxi drivers and each message involves an interaction with DB for require data and then updating the DB. System should also be able to modfy directly datas in case of profile management or setting availability.

Regarding synchronous and asynchronous processes we can divide them in the following way:

- All statements that involve requests and reservation must be synchronous, because system will always know what is the current situation of the queue or the availability status of a driver. There are some constraint to follow, for example a Taxi Driver must recieve only one request or reservation.
- All statements that involve messaging service should be asynchronous, with the exception of confirm messages (after that a driver has accept a request/reservation)

2.3 Components view

REGARDING THE COMPONENTS view we have decided an approach step-by-step and have worked in three main steps.

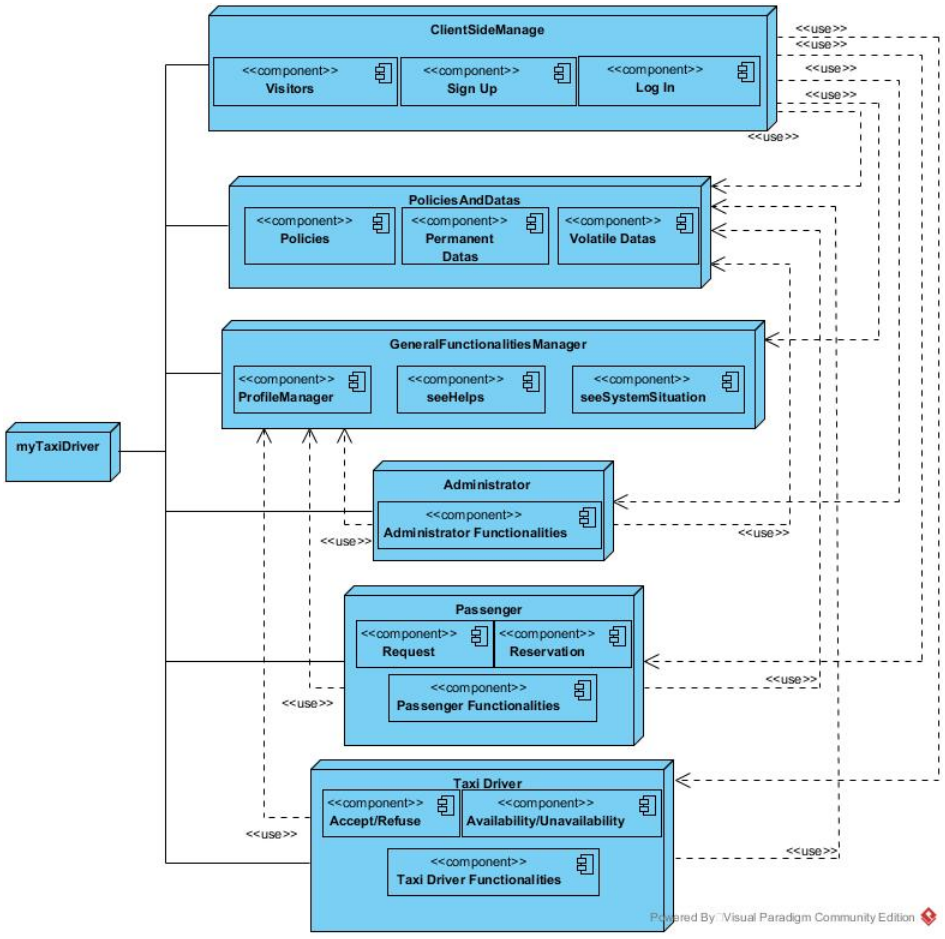
In the first one we have identify the subsytem of myTaxyService® in a very high level way, depending on the functionalitis that they should process. Some subsystems should be shared and each one will contain all that function that will be necessary to reach its goal. Bellow we have identified eight subsystems:

- Visitor sub-system;
- Sign up sub-system;
- Log in sub-system;
- Passenger sub-system;
- Taxi Driver sub-system;
- Administrator sub-system;
- Data sub-system.
- Policies sub-system

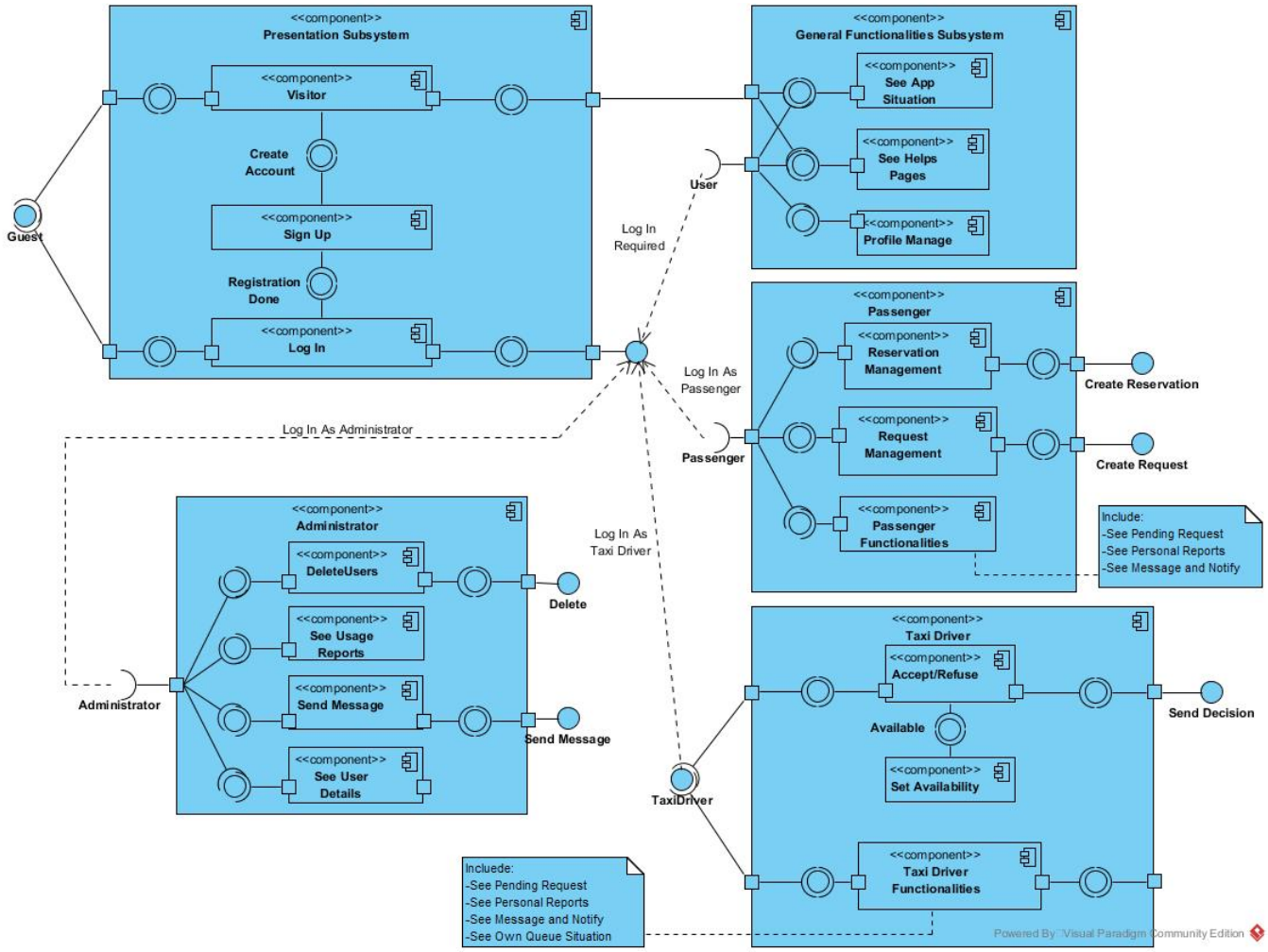
After the first idea we have done the following amalgamations :

- Visitors, Log In and Sign Up subsystem could be considered as a unique entity that will allow to connect people with system. This subSystem will be installed in presentation server and each request, message or decision from user to system or from system to user will pass here.
- Policies subsystem should be considered as a part of DB subsystem because, for example for datas in the DB, each subsystem will need to access on policies rules.
- Some general functionalities, which is independed by the type of log in, will merge in a subsystem called 'General Functionalities Manager'. For general functionalities we mean 'profile managing', 'see queue situation' and 'see helps' pages

After this clarification we have put in evidence of which part of each subsystem will need.We have indicate with <<use>> this type of relation.



In the last one we have added more details in order to clarifty all.



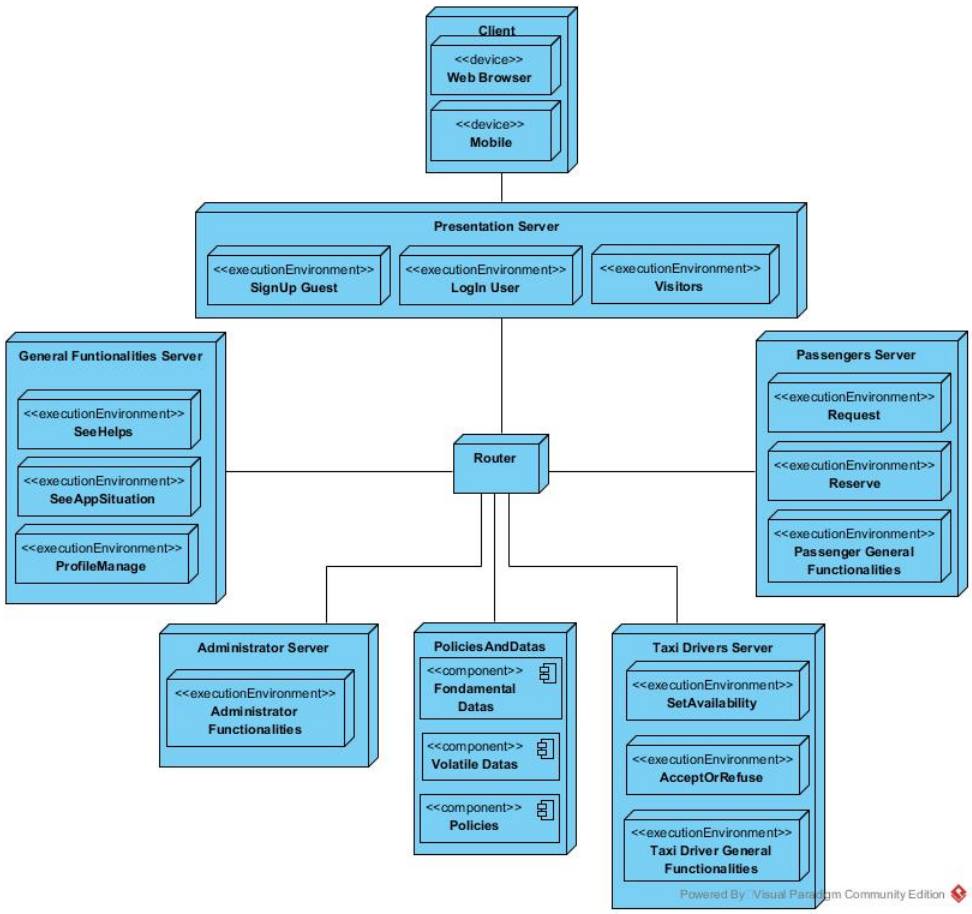
2.4 Deployment view

PHYSICALLY, CLIENT APPLICATION myTaxyService® should be installed on mobile device, such as smartphone or tablet, or should be executed via browser.
Regarding server side we must consider the use of different servers which will execute different operation in a concurrent way. In this way, there should be a backup server if a server was down.
The access to DB (that will be only one) must be synchronous, so we should avoid that two servers should both work on the same record.
We should also provide to divide our DB in two part:

1. A more 'persistent' part which will contain all datas that must be stored for long period, such as profile datas or ride information by drivers or requests/reservations information by passenger

2. A more ' volatile' part which will contain all datas regarding queues, pending request or reservation that are waiting for the taxi driver allocation.

In this way we could protect the most important datas: if the volatile part will go out of work, most probable being that is subject to continious update, the foundamental datas will not lost.



Our choices have been dictated by following ideas:

- we have decided to dedicate a server to forwarding operations between system and users, in this way we have added another level of protection because user cannot comunicate directly with server that perform the service.
- there will be a server dedicated to general functionalities: profile managment will require privileges (given by sign up and log in), seeHelps and seeAppSituation could be used by all, also guest that will use visitors privileges.
- servers will comunicate via a router and all comunication for user must pass by Presentation Server. During the document Presentation Server

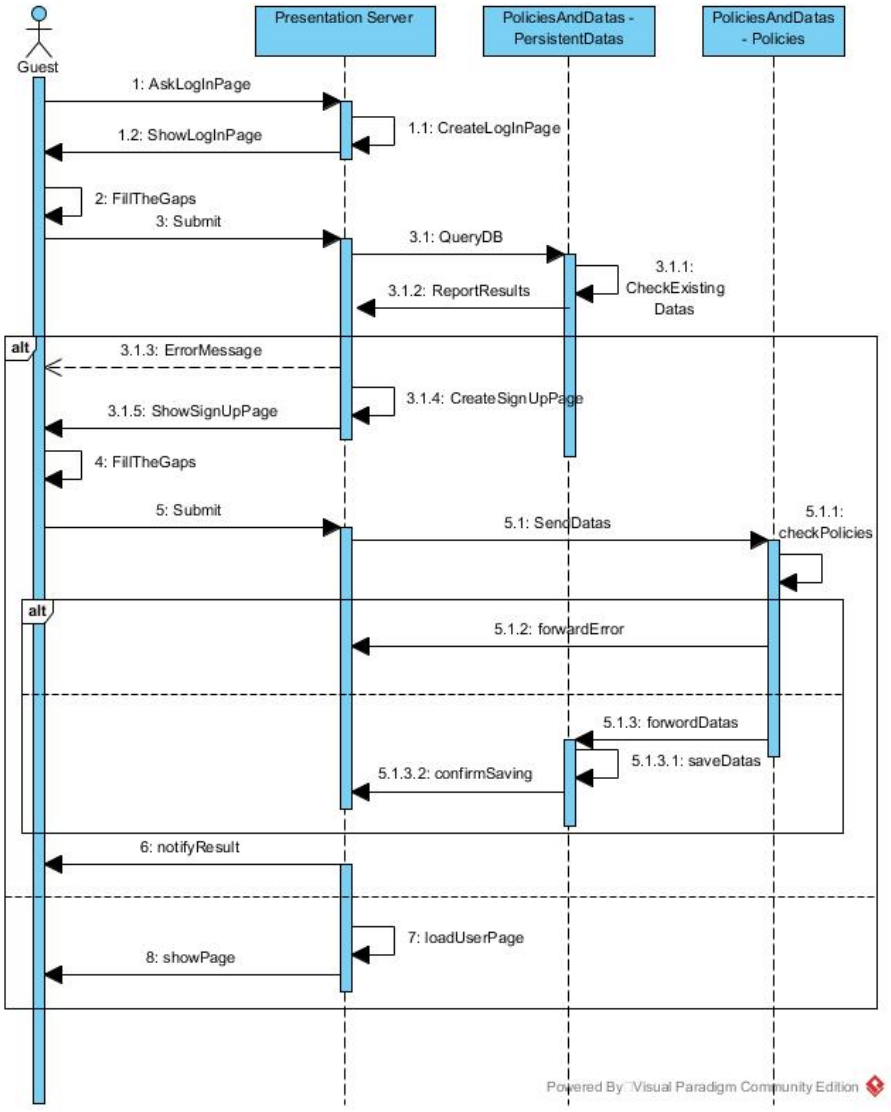
2.5 Runtime view

Sequence diagram of RASD but more in detail

Now we want to focus our attention on how components must act in order to reach their goals and for doing this we have decided to use sequence diagrams.
In next sections we will propose sequence diagram with the concept already presented in RASD but with more details with respect to the new assumption done dure the writing of this DD.

2.5.1 Sign Up or Log In

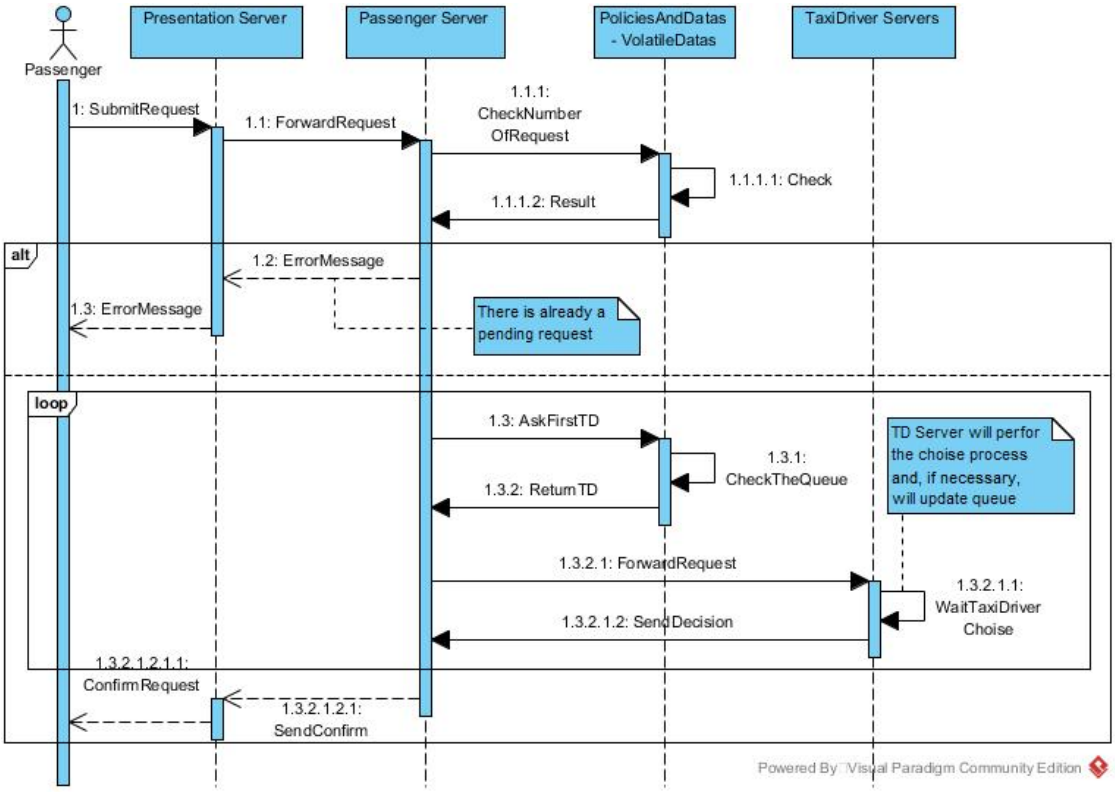
The diagram explain how components interact when a guest tries to log in, but he does not have any account .

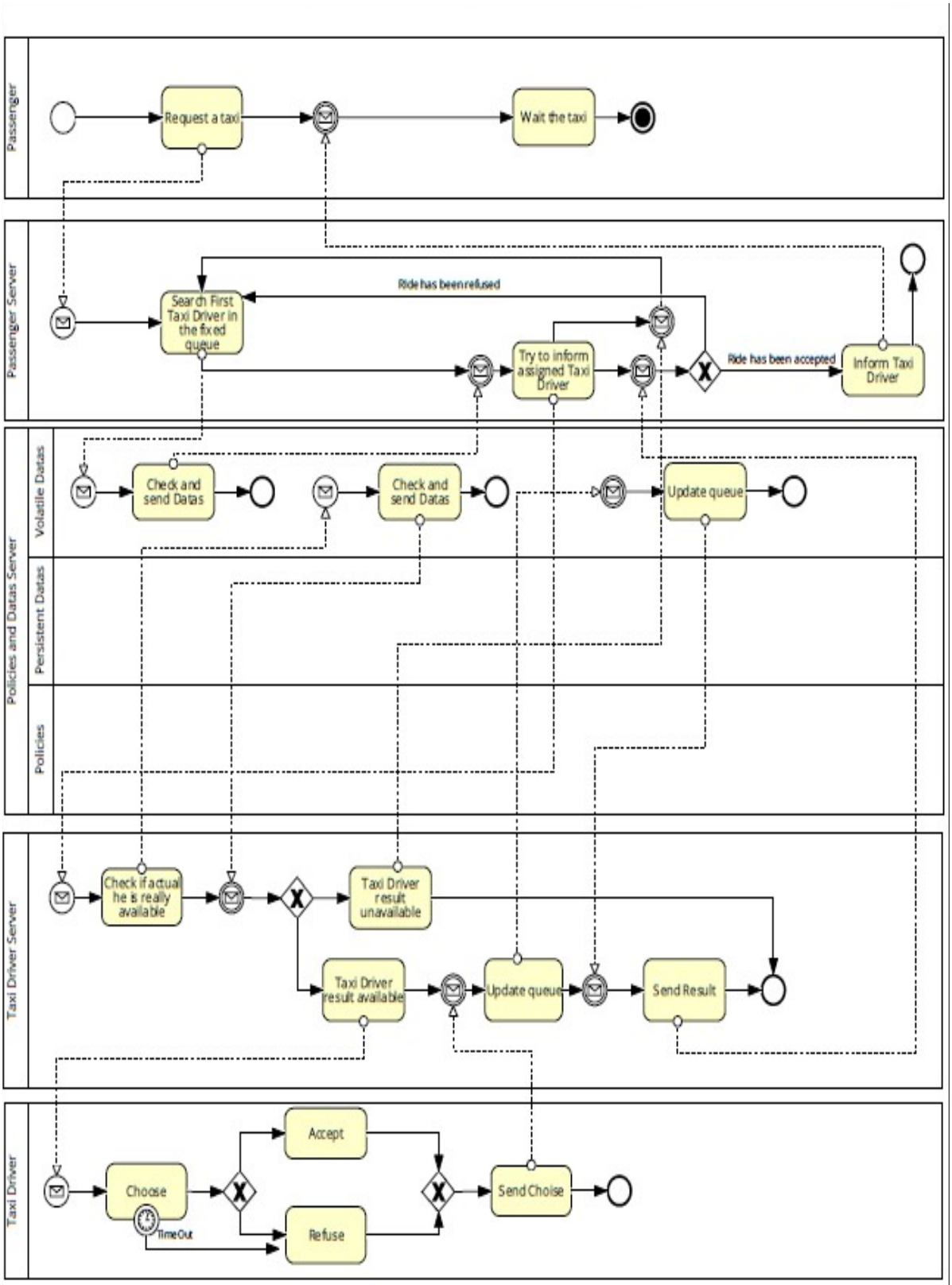


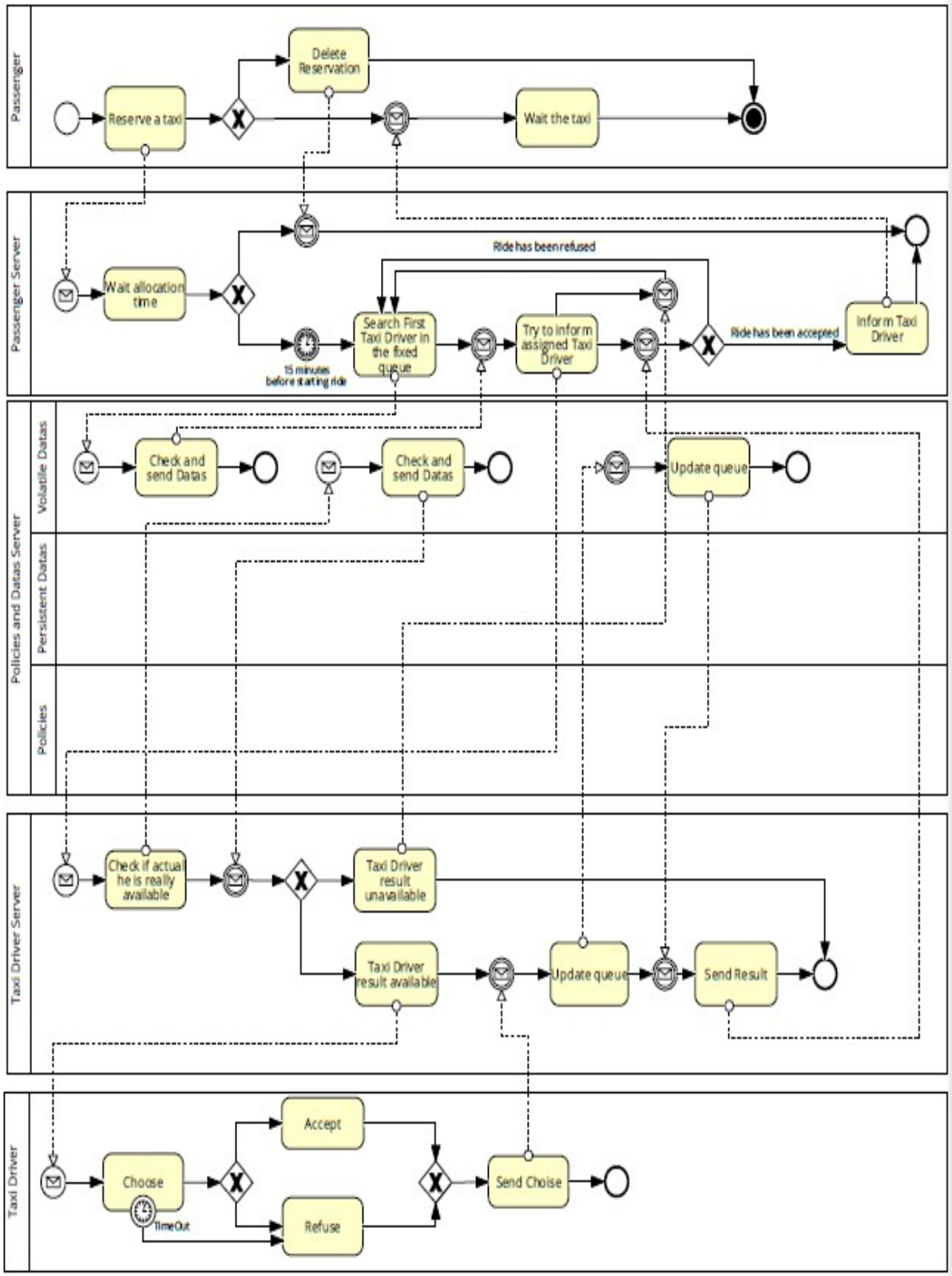
2.5.2 Request, Reservation and Taxi Driver

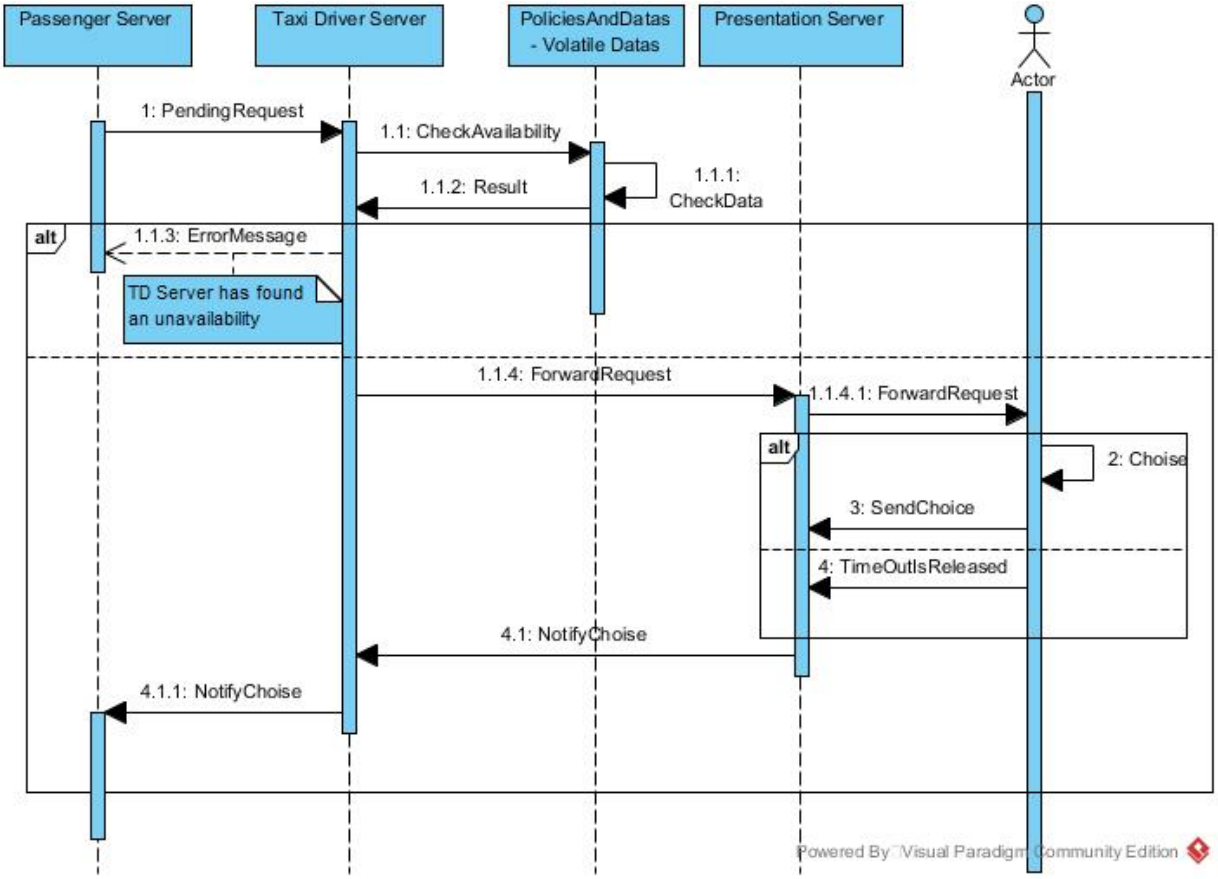
Request and Reserve

For request and reservation we have decided that it is better to propose both type of diagrams: sequence diagram restrict us to focus only on one part, BPMN diagrams allow to have a global view of situation because of merging request (or reservation) with the Taxi Driver Choise.



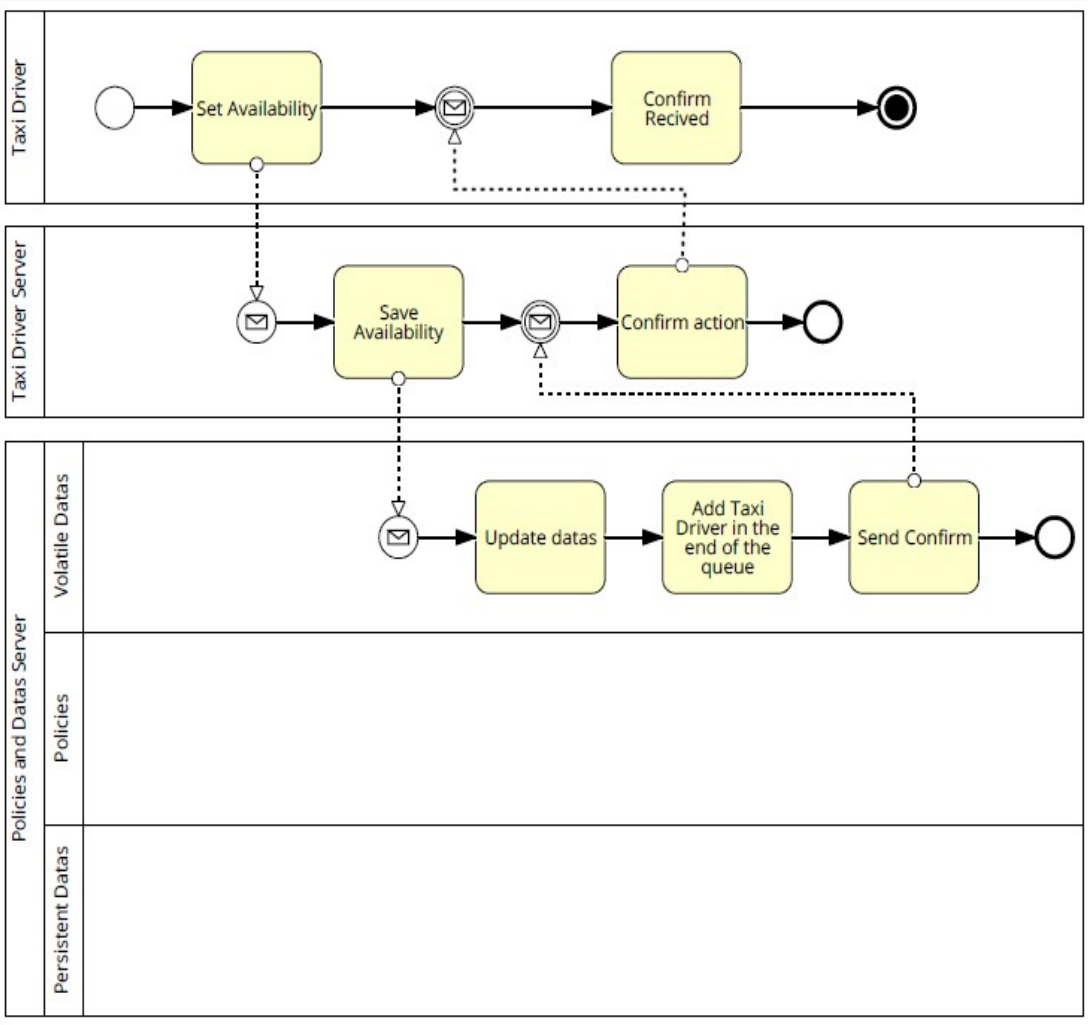

Passenger
Request a taxi
Wait the taxi
Passenger Server
Ride has been refused
Search First Taxi Driver in the fixed queue
Try to inform assigned Taxi Driver
Ride has been accepted
Inform Taxi Driver
Policies and Datas Server
Volatile Datas
Persistent Datas
Policies
Check and send Datas
Check and send Datas
Update queue
Taxi Driver Server
Check if actual he is really available
Taxi Driver result unavailable
Taxi Driver result available
Update queue
Send Result
Taxi Driver
Choose
TimeOut
Accept
Refuse
Send Choise


Passenger
Reserve a taxi
Delete Reservation
Wait the taxi
Passenger Server
Wait allocation time
Ride has been refused
15 minutes before starting ride
Search First Taxi Driver in the fixed queue
Try to inform assigned Taxi Driver
Ride has been accepted
Inform Taxi Driver
Policies and Datas Server
Volatile Datas
Persistent Datas
Policies
Check and send Datas
Check and send Datas
Update queue
Taxi Driver Server
Check if actual he is really available
Taxi Driver result unavailable
Taxi Driver result available
Update queue
Send Result
Taxi Driver
Choose
TimeOut
Accept
Refuse
Send Choise


Passenger Server
Taxi Driver Server
PoliciesAndDatas - Volatile Datas
Presentation Server
Actor
1: PendingRequest
1.1: CheckAvailability
1.1.2: Result
1.1.1: CheckData
alt
1.1.3: ErrorMessage
TD Server has found an unavailability
1.1.4: ForwardRequest
1.1.4.1: ForwardRequest
alt
2: Choise
3: SendChoice
4: TimeOutIsReleased
4.1: NotifyChoise
4.1.1: NotifyChoise
Powered By Visual Paradigm Community Edition

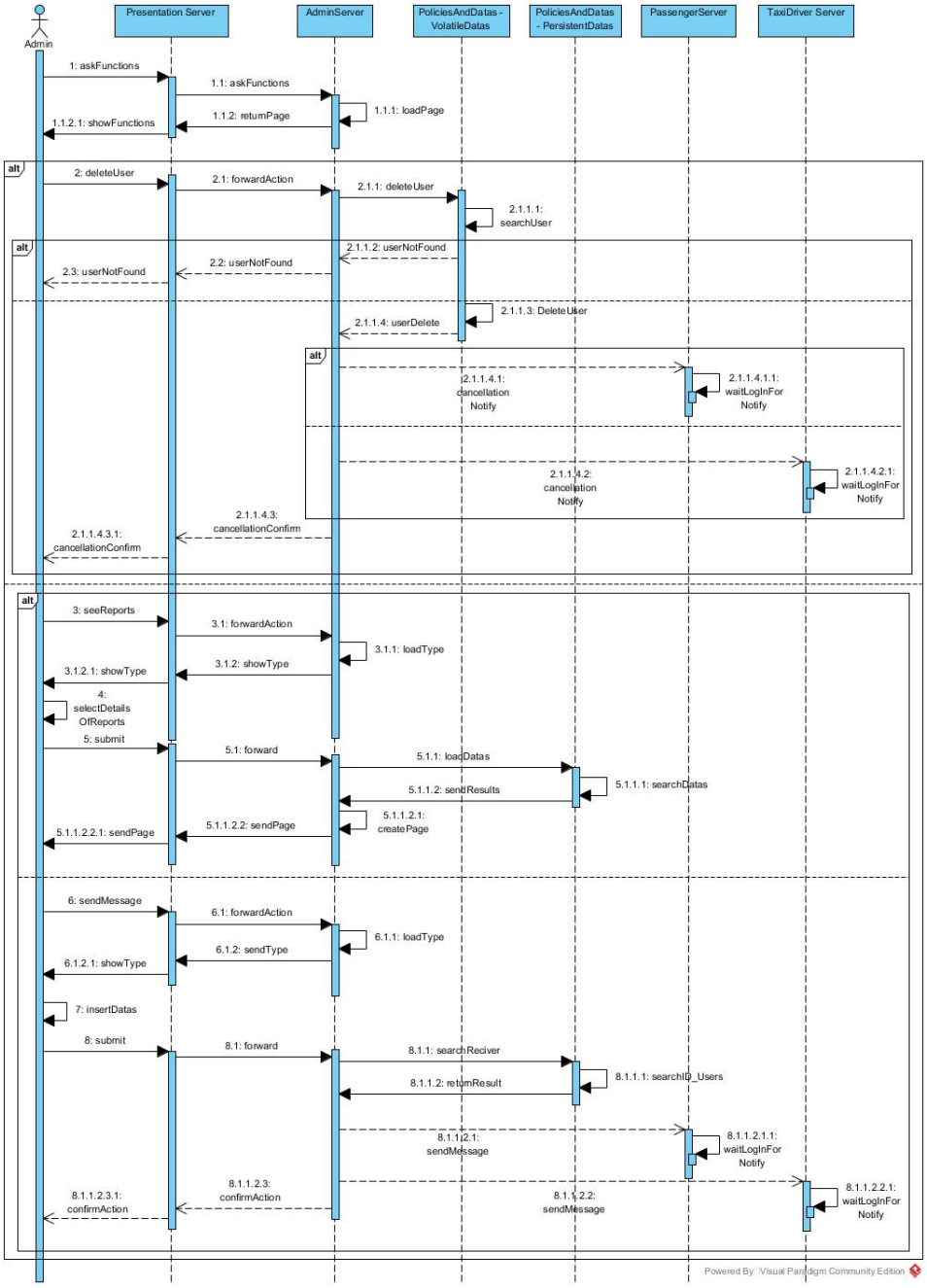
Availablity

For availability it is better to use the BPMN diagram because it allows to explain in which part of DB we have decided to store that datas that change frequently during service usage



2.5.4 Admin Functionalities

For this part we have decided to use sequence diagrame because it is more clear to explain multiple possibilities of admin with it.

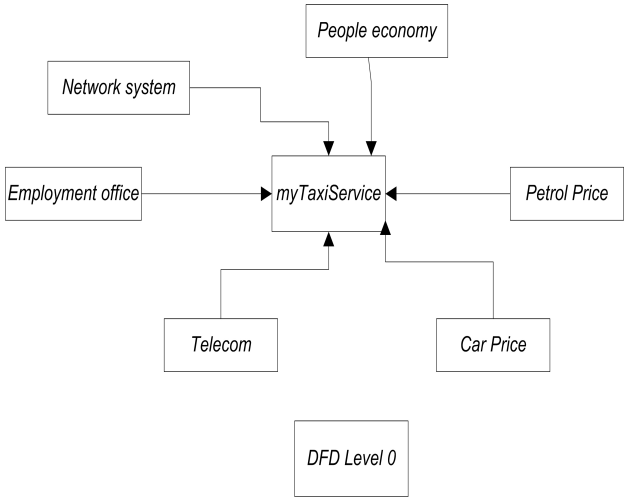


2.6 Component interfaces

In this section we are going to draw DFD diagram in two levels: Level 0 and Level 1.

Level 0:

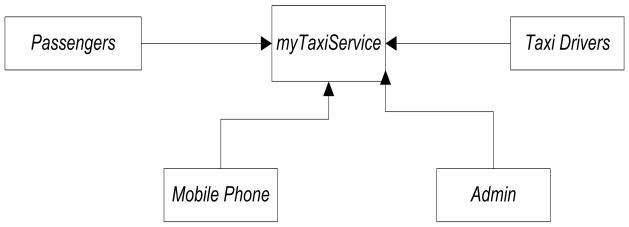
in this level the outside factors which affect on our system will be discussed. Bellow we can see some outside factors.



DFD Level 0

Level 1:

in this level we are going to explain the factors that affect our system from inside of the system.



DFD Level 1

2.7 Selected architectural styles and patterns

IN THIS FINAL part there is a short explanation of our design decision.
For all the sections we have used a step-by-step approach because we have faced the document as a refining job. We have exposed our ideas and then we have tried to clarify each part more in details.

- The choise of this architecture (JEE) has been done in order to increase the security of our datas and because we have different types of client to manage (web browser and mobile).
- In Component view we have started the analysis of each component, we have divided our system with respect to functionalities, in order to create a balanced product, increase scalability and explain how these components will interact.
- In the deployment view we have used an architectural model presented as 'Deployment Diagram' by our tool (Visual Paradigm) and we have tried to clarify which should be the physical structure that will accomodate our system and how server will be connected. We have thought different ideas for this part but as specify in details in the section reserved, that one could be the best.
- The runtime view has been shown through sequence diagram because in our opinion it is a good way to clarify how the subsystems that we have thought will be coordinated. With this approach we don't need to use long description or less clear diagram.
- Component interface is been presented in order to explain any input or output paramenters used and created by our components.

Regarding patterns, we have identified that, the following one could be useful for our system:

- Singleton : it allows us to create only one instance of an object, so all proccess should refer to that one. In our case, we could create only one instance for the Database, and only when it is necessary
- Proxy: it allows us to insert an additional layer between our server and user. we have already decided that there will be a presentation server that will be positioned between the router and the user but with a proxy also that one will be protected. In the specific situation, a Protection Proxy could provide different clients with different levels of access to some objects
- Facade: it permit sus to hide the complexity of our subsystem providing simple interface to modules that must interact with them
- Adapter: it is useful reguarding user interfaces, we will use it to convert an interface to another becuse we don't know preventively if a user will use a mobile device or a web browser.
- Strategy: it permits us to manage in a dynamic way the difference between request algorithm and reservation algorithm: we will create a class for each algorithm and each one will implement the same interface, for example 'createRide'.

3. Algorithm Design

In this phase we will propose our idea about algorithm for the management of the queue. We have decided to use natural languages and clarify all with some diagrams. Concept that we should apply is not very sophisticated but it is not necessay for managing a concatenated sequence of records.. If we should see in detail what is stored in our DB, we should see a lot of records that are concatenated each other with some reference.We have decided that DB will save each time a record that will contain 4 datas:

- *ID_TD:* that contains the ID_TaxiDriver of the TaxiDriver that we are considered.
- *ID_Previously:* that must contain ID_TaxiDriver about the Taxi Driver that is before him in the queue. If Taxi Driver is the first of the queue, this data will be empty (it will contain NULL).
- *ID_Last:* that will be set with something different to NULL only while Taxi Driver is the last in the queue, in the other cases it will contain NULL.
- ID_Zone: here is stored the ID of the zone of the queue where Taxi Driver has been added.

Now we will proceed with the analysis of principal cases in which our algorithm should be necessary.

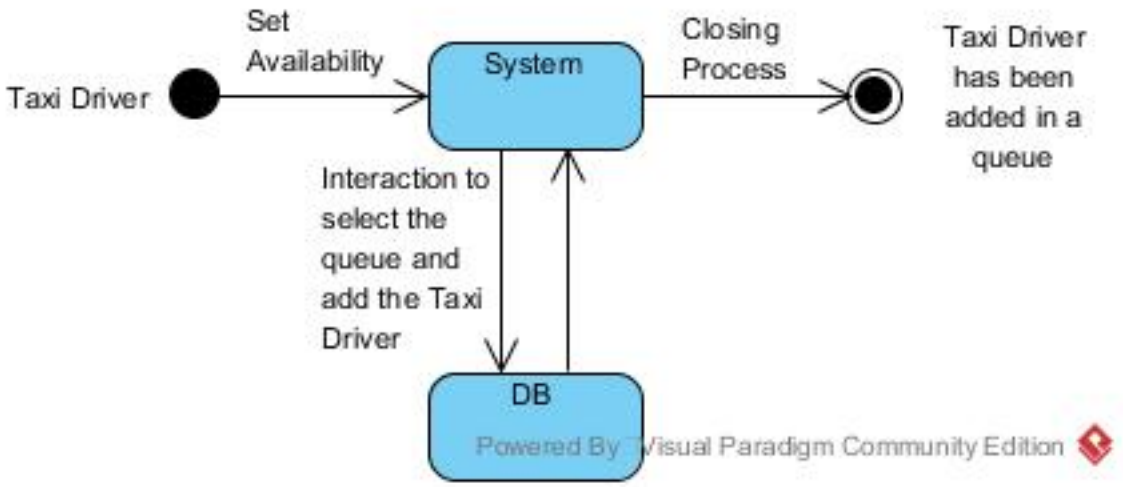
- **AVAILABILITY/UNAVAILABILITY**

  When a Taxi Driver will set availability option, the action will be recived by Taxi Driver Server that will check, with an interaction with DB, which zone needs to a largest number of Taxi (in order to satisfy our clients) and will comunicate to DB to create a new record and set ID_Zone equals to the zone selected. After the choise, Taxi Driver must be added on the queue.
  Let us suppose that Tommaso is the last Taxi Driver in the queue and Giacomo is the new entry. Taxi Driver Server will do following steps:

  - it will query DB on which is the Taxi Driver with ID_Last different to NULL.
  - DB will find a record which contain ID_TaxiDriver of Tommaso and comunicate it.
  - Taxi Driver Server comunicates to DB to modify the record that has just been created and setting ID_TD with ID_TaxiDriver of Giacomo, ID_Previously with ID_TaxiDriver of Tommaso and ID_Last with NULL.

- At the end ID_Last of the record that contain ID_TD equals to ID_TaxiDriver of Tommaso will be set with something different to NULL.
- If the query had been empty, both ID_Previously and ID_Last will be set on NULL.
- DB informs Taxi Driver Server that the action has been performed.
- Taxi Driver Server informs Taxi Driver that he is in the queue.

In digram we want to bring our the events flow, so 'System' will indicate the group of servers that interact with DB.



- **INFORMING A TAXI DRIVER ABOUT A REQUEST**

  In this moment we have a Passenger that have done a request, Passenger Server must find the first TD in the queue after an interaction with DB. The stept that will be compure are:

  - Passenger Server queries DB on which is the record with ID_Previously equal to NULL : the corrisponding ID_TD will be the Taxi Driver on top of the queue.
  - Passenger Server forward requet with the ID_TD found to Taxi Driver Server
  - TD Server check if TD is really available.
  - If he is not available (maybe he change his state in that moment), TD Server informs Passenger Server and it wil try to check the queue situation a second time.
  - If he is really available Taxi Driver System informs that Taxi Driver about the request and will start a timeout while waiting an answer.

- **UPDATE QUEUE**

  In this moment our system will have a pending request and should happen three different cases:

  1) Taxi Driver accept the request

  Here system should update the entire queue. This action will be done by Taxi Driver Server before to send a confirm message to Passenger Server. We can do the following step: Taxi Driver Server will order to DB to delete record that have ID_TD equals to the ID_TaxiDriver of TaxiDriver that has accepted the allocation. As final system will order to DB to set on NULL the data which has ID_Priviusly equal to to the ID_TaxiDriver of TaxiDriver that system is removing from the queue

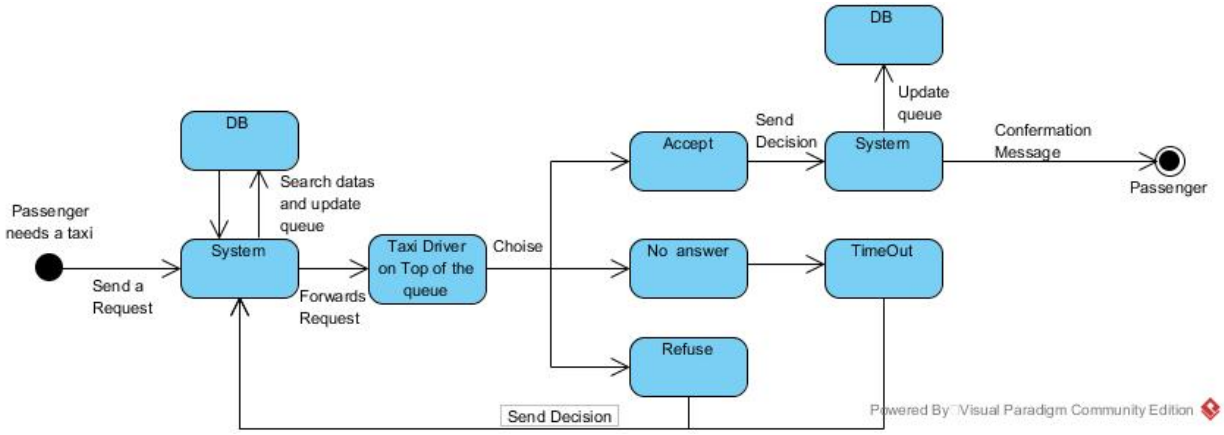
  2) Taxi Driver refuse the request

  This action will be done by Taxi Driver Server before to send a negative message to Passenger Server. TD Server will query to DB which is the record of that queue that has ID_Last different to NULL. The system will set ID_Previously of the Taxi Driver that has refused the request with the ID_TD of the last Taxi Driver in the queue and invert the two ID_Last (so there will be always only one ID_Last different to NULL). As final action ID_Previusly that contain ID_TaxiDriver of Taxi Driver that has refused the request, will be set on NULL, so the associated Taxi Driver will be the new first of the queue

  3) TimeOut is released

  Actions will be like if Taxi Driver had refused the request

As for first diagram, now we will try to bring out the event flow, without components details.



4. User Interface Design

As explained and discussed in RASD the user interface of myTaxiService® is expected to be used via Web App and Mobile App, below it is explained in more detail the User Interface of both methods.

4.1 Web Application:

4.1.1 General Pages

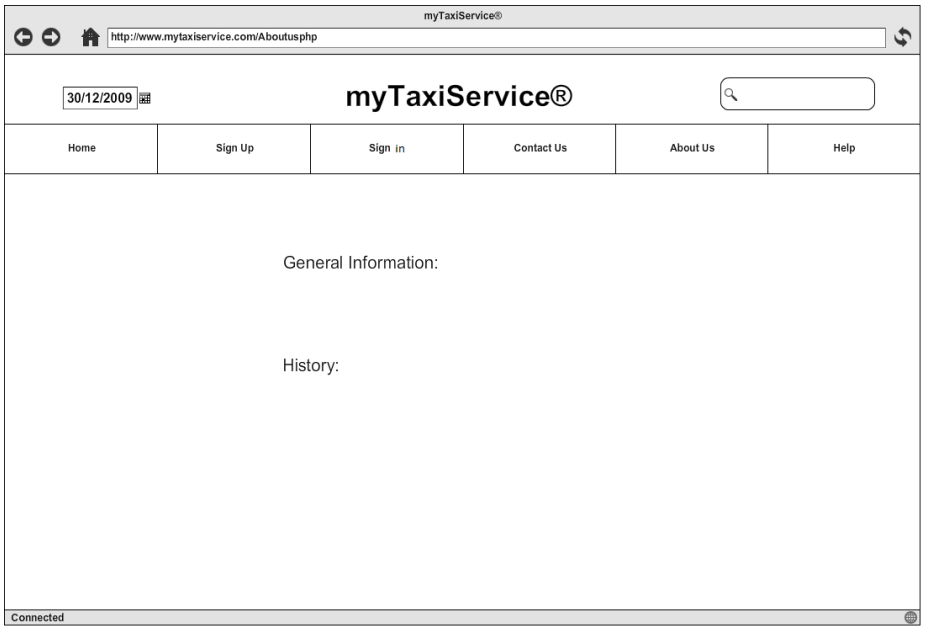
This section includes some pages that everyone can see and use, with some basic features of the Website.

4.1.1.1 Home Page

This page is the main page which shows the advertisement of the system.
A guest could see some basic feature in the home page. In this page there are several items and include lots of linked pages such as: Sign up, Login, contact, About us...

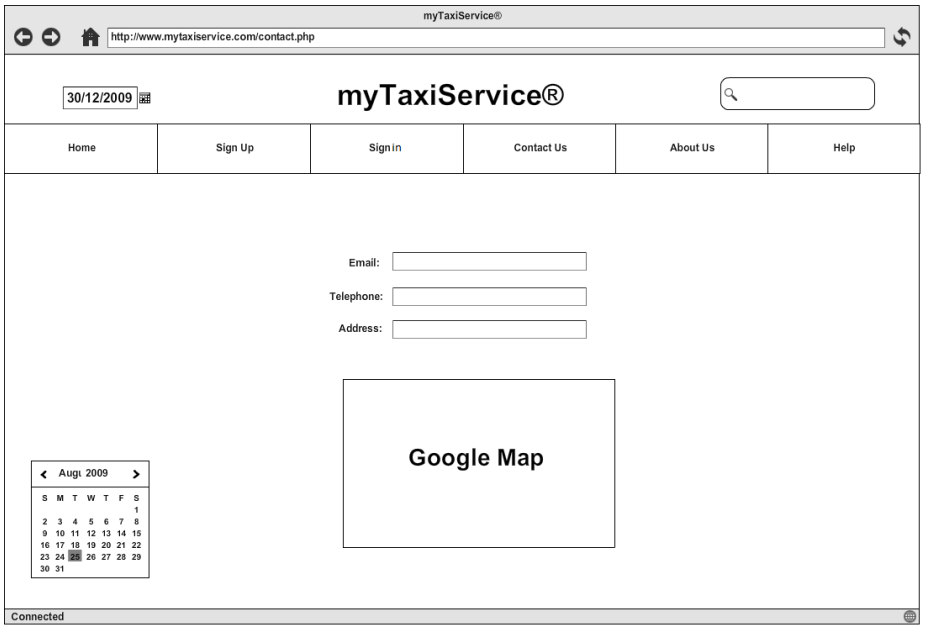
4.1.1.2 About Us

This page shows some information about how the system is working, who can use this website and what is the benefits of this system.



4.1.1.3 Contact Us

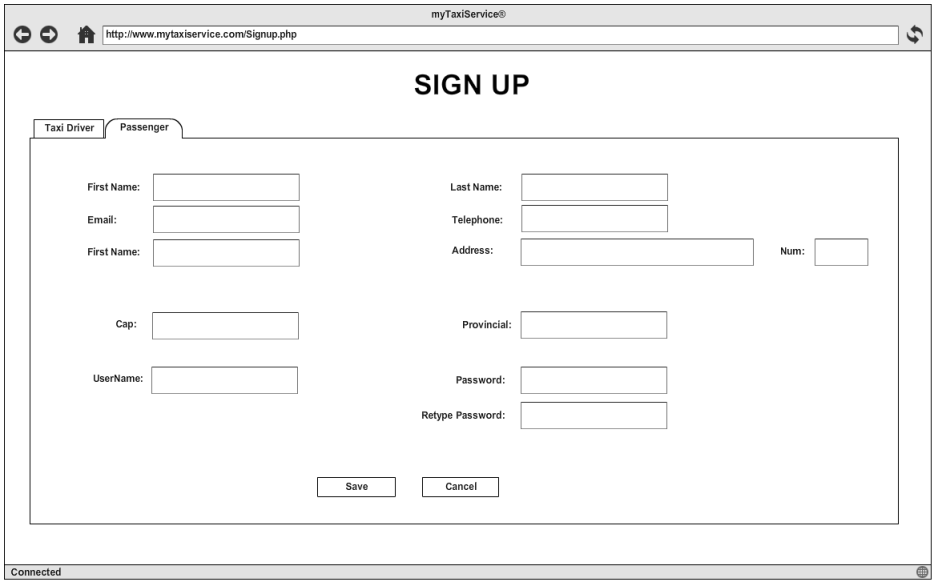
This page shows several ways of making connection with the operator by Email, Telephone number or Facebook.



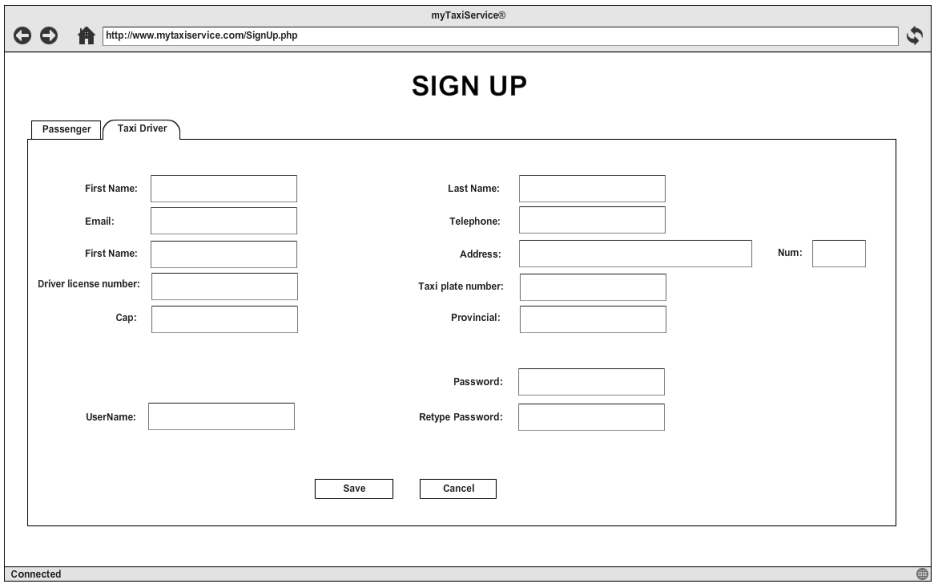
4.1.1.4 Sign Up

The Sign up page is divided into two categories. Passengers and Taxi Drivers. Regards to the user type, some specific information will be needed.

Passenger
Passenger sign up includes the following information : First name, Last name, Email, Telephone, Address, Num, Cap, Provincial, Username, Password, Retype Password

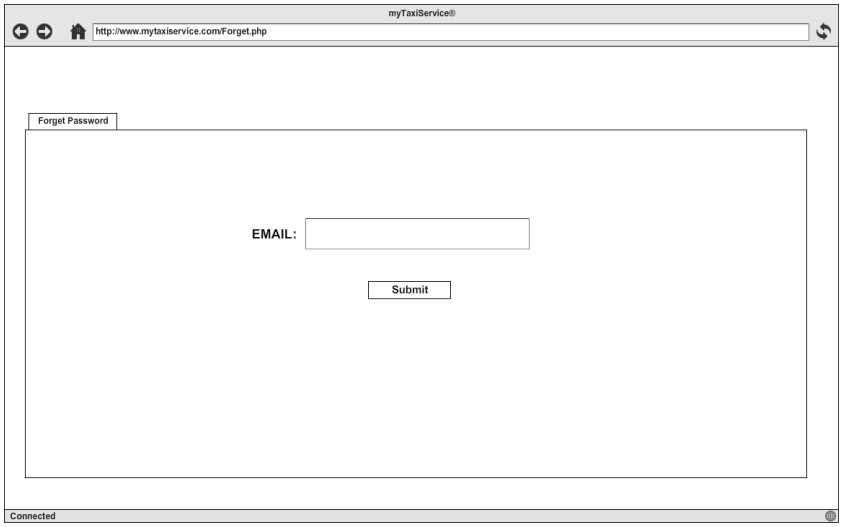


Taxi Driver
Taxi Driver sign up includes the following information : First name, Last name, Email, Telephone, Address, Num, Driving license number, Taxi plate number, Cap, Provincial, Username, Password, Retype Password



4.1.1.5 Password Forgot

This page is designed for who forgot their password.

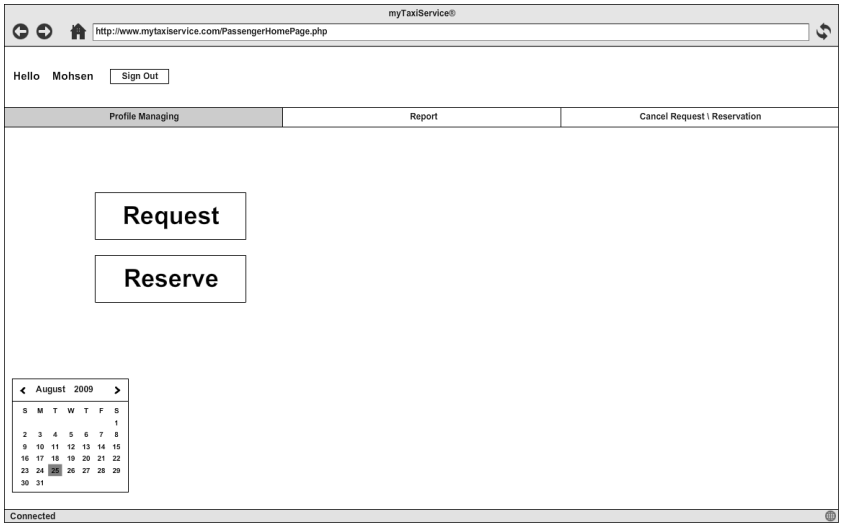


4.1.2 User Specific Pages

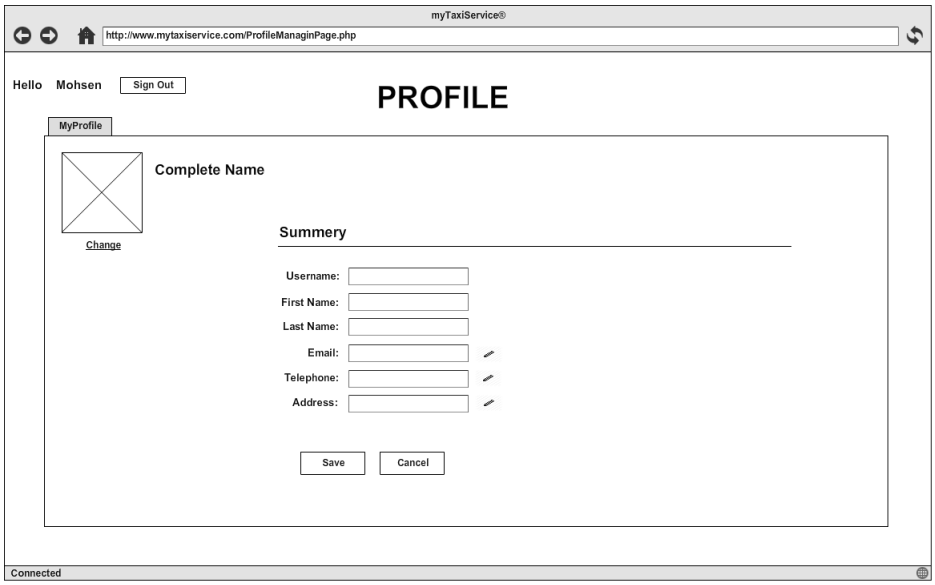
This part includes some pages in which a registered user can access to them.

4.1.2.1 Passenger pages

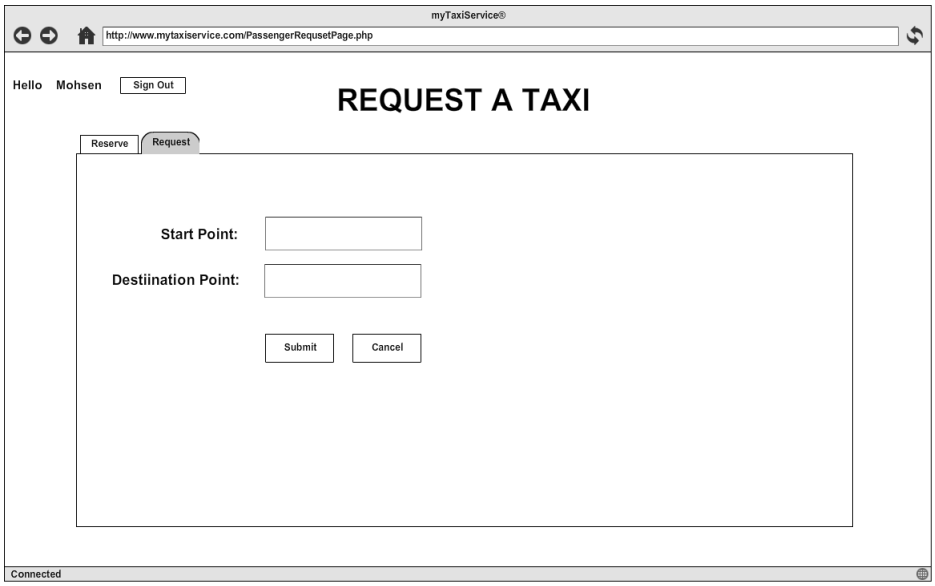
Home Page
This page is designed specifically for passengers which allow them to do variety actions such as: Request or Reserve a taxi, Profile managing, Report and Cancel Request or Reservation.



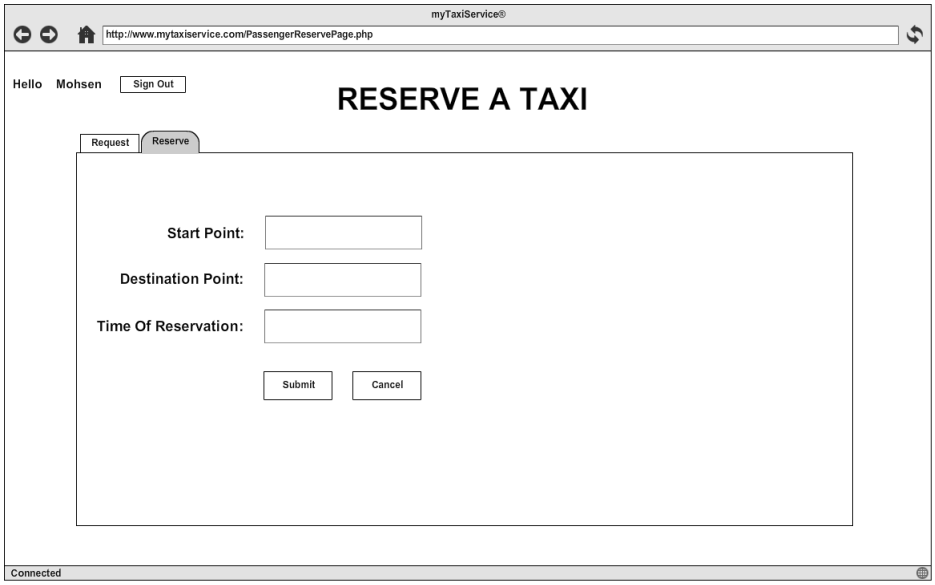
Profile Managing
This page is made for Passengers in which profile information such as Edit, Add and Delete can be done. Profile photo is changeable and some pencils can be seen beside the text boxes which mean that, only these information can be modified.



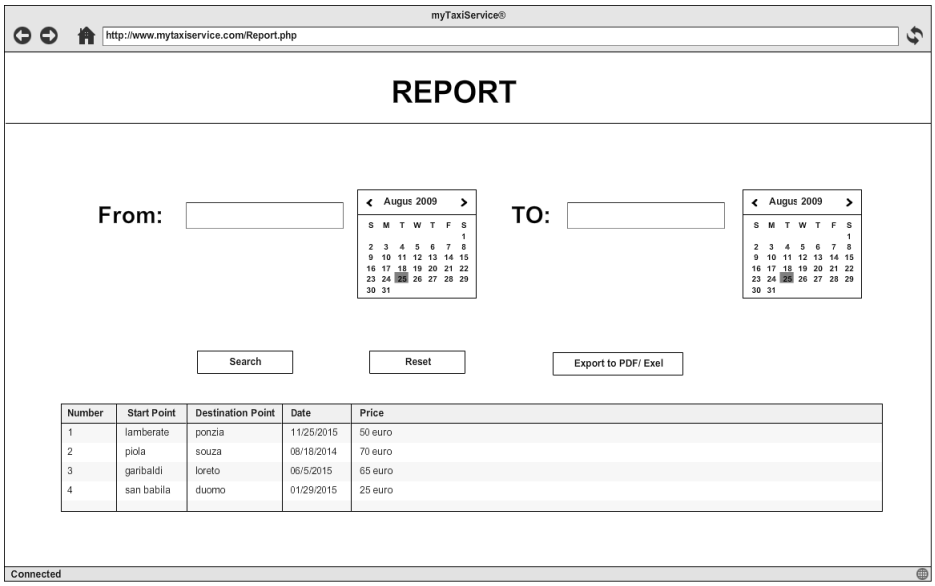
Taxi Request
This page is designed for Requesting a taxi. There are some spaces that should be fulfilled by passenger for example, the start point and the destination of the ride.



Taxi Reservation
This page is designed for taxi Reservation. In addition to determining start point and the destination point a passenger should specify the time of the reservation.



Report
This page is designed for Passengers to get report of how much money do they spend for riding in the specific period of time and get report from the detail of any rides include Start point, End point and the date of the ride.



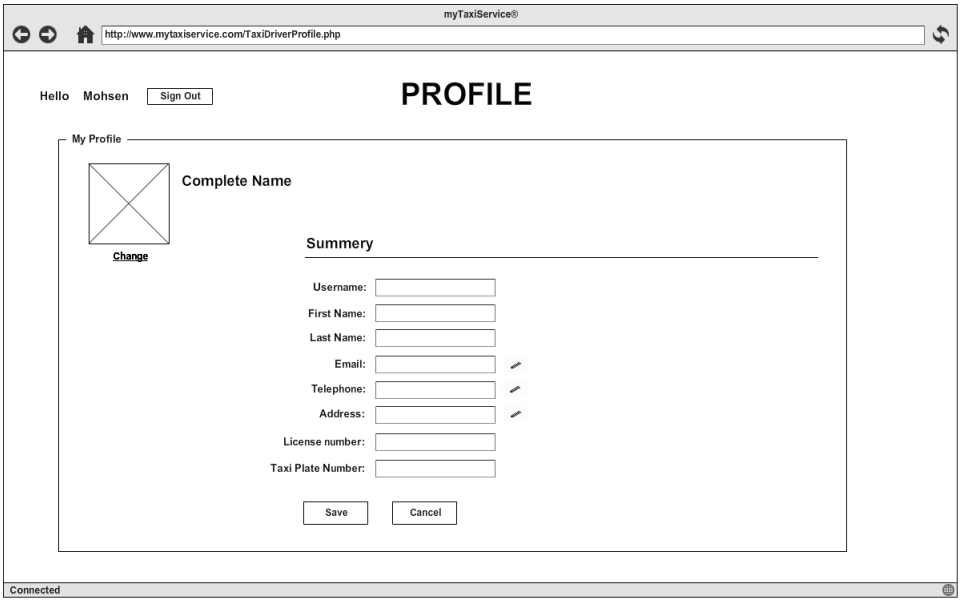
4.1.2.2 Taxi Driver pages

Home Page

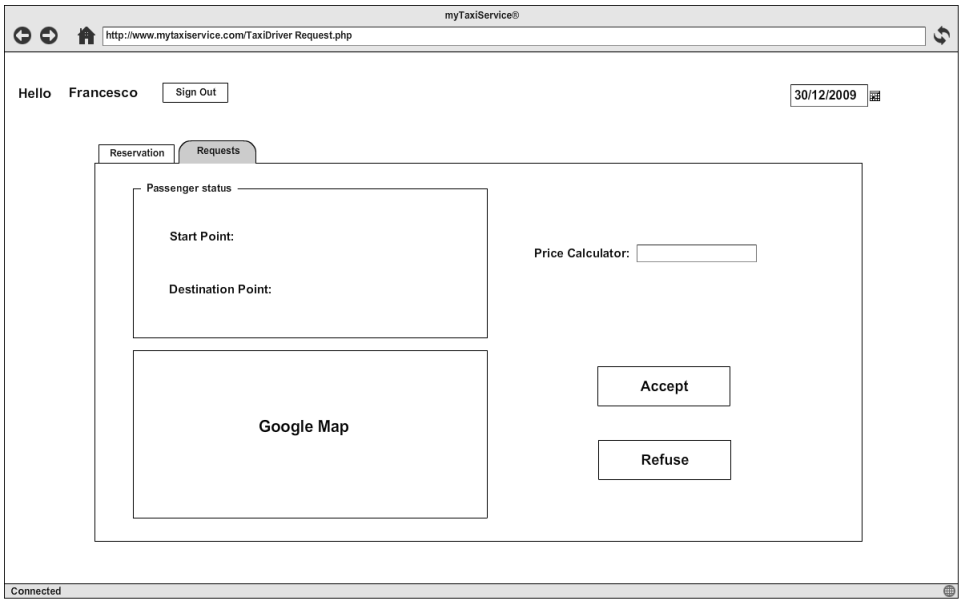
This is the main page of Taxi Drivers in which lots of buttons and links can be seen. In addition to specifying the availablity or unavailability option it includes profile managing, request\reservation form and report pages. Moreover a taxi driver can see the Request and Reservation notification in this page.

Profile Page

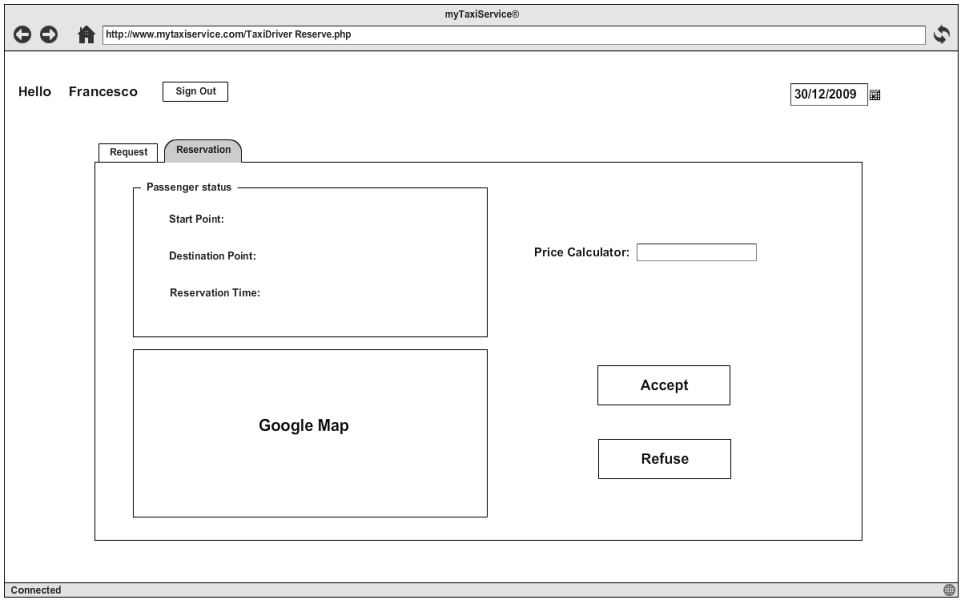
In this page a taxi driver can Edit, Add and Delete some personal information such as: Email, Telephone and Address



Passenger Request
This page is designed for Taxi drivers in order to know about the details of the rides such as: Start and destination point and the price of the ride. Google map is created to help taxi drivers find their passengers.



Passenger Reserve
This page is the same as the previous page but the only difference is that the time of reservation can be seen by a taxi driver. In addition, a google map is located in the bottem of the page.

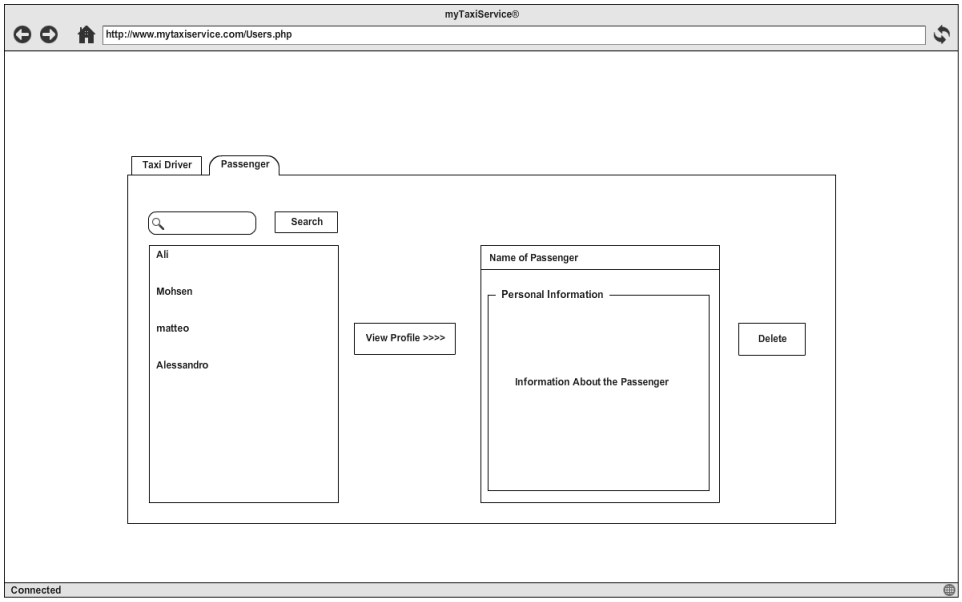


4.1.2.3 Administrator Pages

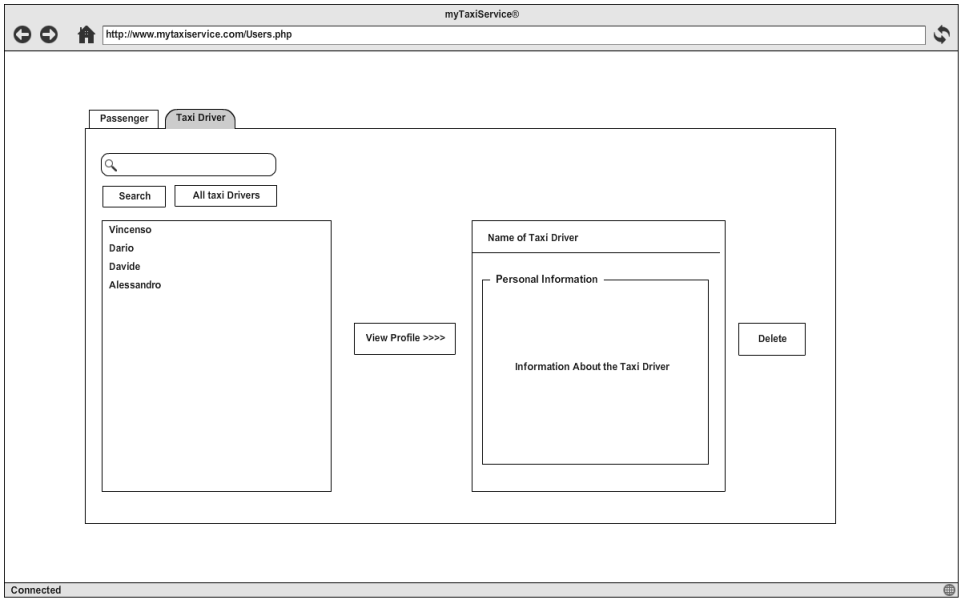
Home Page
This page is designed for administrator in which he\she can do some actions on profile managing, users and get report from the system. In the middle of the page some cutting edge news will be shown.

Users Managment
This page is divided into two tabs, Passengers and Taxi Drivers.

Passengers:
In the passenger tab the administrator can search any passengers and see their profile and delete if something is done against the rules.

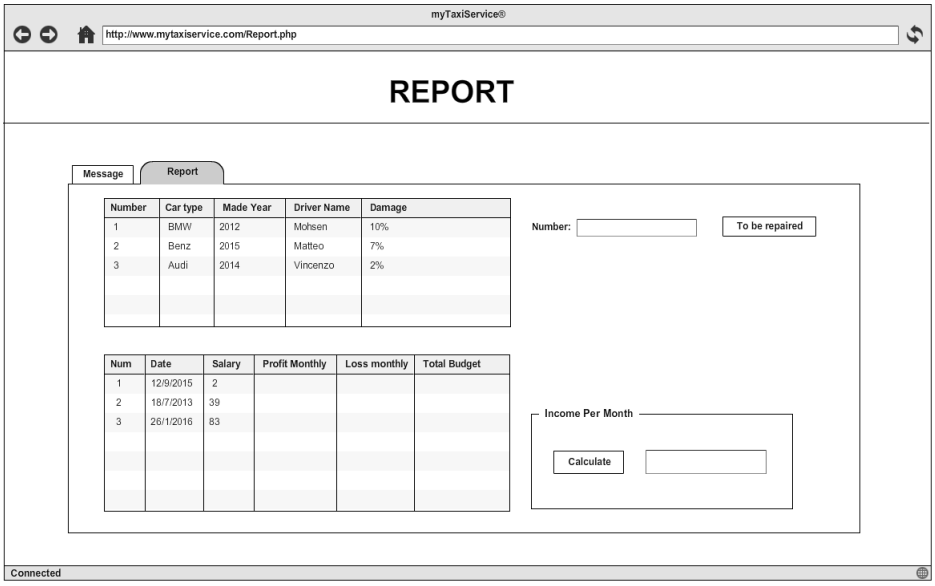


Taxi Drivers:
In this tab the administrator has the power of searching a taxi driver, see his\herprofile page and delete his\her account if it is needed.

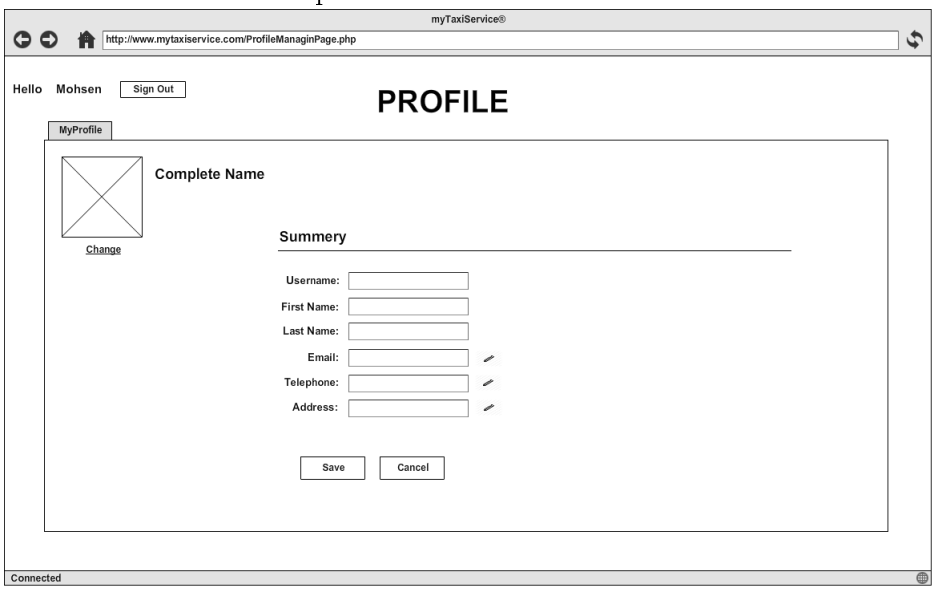


Message
In this tab the Administrator can send a message to every user, only taxi drivers or passengers in order to inform them about the possible changes in the future.

Report
In this tab the administrator can awar of cars status and if needed send damaged cars to be repaired. Also administrator can get report of the monthly income of the service, profits, losses and total budget of the system



Profile
In this page the administrator can change some information such as his\her profile photos or the fields that have a pencil beside them.



4.2 Mobile Application

In this part we are going to draw Mobile interface but with some restriction in compare with the Web interface.

4.2.1 General Pages

In this page everyone can use these pages in other words they are public pages.

4.2.1.1 Home Page

This is our main page which includes: Sign up, Sign in, Contact us and About us pages with some slide show in the middle of the page and a calendar.

4.2.1.2 About Us

In this page we can see some General Information about the way that system works and the benefits of using this system and a brief history of our system.

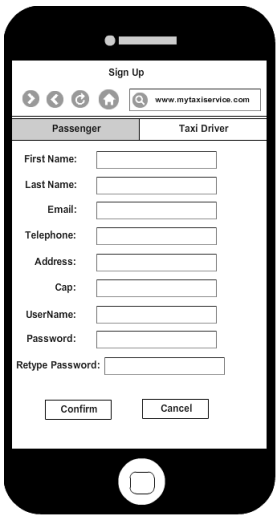
4.2.1.3 Contact Us

By using this page everyone can get in touch with our system by sending an email, Telephone and address.

4.2.1.4 Sign Up

The Sign up page is divided into two categories. Passengers and Taxi Drivers. Regards to the user type, some specific information will be needed.

Passenger
This page is designed specifically for passengers which includes particular information that should be fullfiled bu a passenger such as : First name, Last name, Email, Telephone, Address, Cap, Username, Password, Retype Password



Taxi Driver
In this page a Taxi Driver should fullfil some specific information which is designed only for them these information include: First name, Last name, Email, Telephone, Address, Num, Driving license number, Taxi plate number, Cap, Username, Password, Retype Password

4.2.2 User Specific Pages

This part includes some pages in which a registered user can access to them.

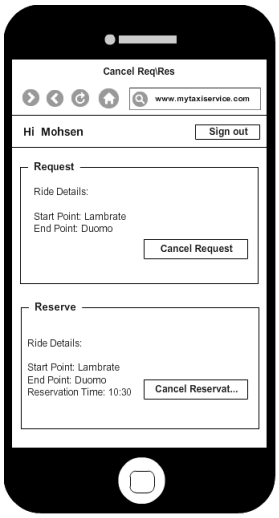
4.2.2.1 Passenger pages

Home Page
As can be seen this page is designed for passengers which includes Request, Reserve and cancelation part. Also there is a google map that can be used with a calendar below that.

Taxi Request
In this page a passenger can request a taxi by entering start point and destination point. Morover there is an check mark option for disable people that want to request and equiped taxi.

Taxi Reservation
This page is the same as request page with this difference that a passenger should specify the time of reservation.

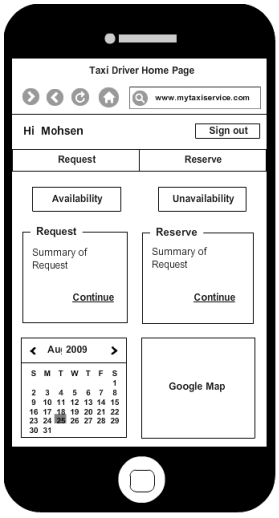
Cancel Request\Reservation
This page is designed for Passengers in order to allow them to cancel their request or reservation.



4.2.2.2 Taxi Driver pages

Home Page

In this page a taxi driver can set the availability\unavailability option and there are two links that a taxi driver will decide whether accept\refuse a request or a reservation. Moreover a taxi driver can see the Request and Reservation notification in this page.



Accept\Refuse Request

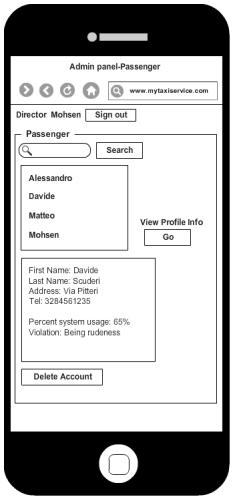
This page is designed for accepting\refusing a request. A taxi driver can see the start point and the destination point of a passenger. Also there is a button which calculate the price of the ride. Moreover a Google map is created to help taxi drivers find the location of passengers.

Accept\Refuse Reserve
This page is the same as the previous page but the only difference is that the time of reservation can be seen by a taxi driver. In addition a google map is located in the bottom of the page.

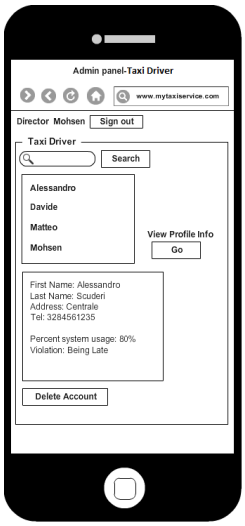
4.1.2.3 Administrator Pages

Home Page
This page is the panel of Administrator in which there are two main links in it: 1) Users 2) Report. Also there is an slide show about news and thechnology advertisment with a google map and a calendar on the bottom of the page.

Users Managment
This page is divided into two tabs, Passengers and Taxi Drivers.
Passengers:
In the passenger tab the administrator can search any passengers and see their profile and delete if something is done against the rules.



Taxi Drivers:
In this page the administrator can search any taxi driver and see his\her profile page and delete his\her account if it is needed.



Message
In this page the Administrator can send a message to every user, only taxi drivers or passengers in order to inform them about the possible changes in the future.

Report
In this page the administrator can see the cars status and if needed send damaged cars to be repaired. Also administrator can get report of the monthly income of the service, and total budget of the system

5. Requirements Traceability

REGUARDING REQUIREMENTS that we had exposed in RASD, we will propose the consistency between that document and DD.

Functional Requirement are explained in the following parts:
Sign Up Requirements can be verified in subsection *2.5.1* and *4.1.4;*
Log In Requirements can be verified in subsection *2.5.1* and *4.1.1*;
Request Requirements can be verified in subsection *2.5.2* and *4.2.1*;
Reservation Requirements can be verified in subsection *2.5.2* and *4.2.1*;
Accept/Refuse Requirements can be verified in subsection *2.5.2* and *4.2.2*;
Queue Management Requirements can be verified in section *3*, that is reserved totally to the queue algorithm;
Administrator Functionalities Requirements can be verified in subsection *2.5.3* and *4.2.3*;
Profile Management Requirements can be verified in subsection *4.2.1, 4.2.2* and *4.2.3.*

Non Functional Requirment are also verified.
Reguardig *User Interface requirements*, they can be verified in section *4*. We have the main page, as we had supposed in RASD, and we have done some updating in personal pages described in the previus document. We have decided to create more than one page (with all informations) , so the application will be more user-friendly but we have respected all details expected.
Users Characteristics requirements have been respected with the insertion of an age-control system during sign up and the possibility to require a special taxy during request or reservation.
As last, it can be observed that *Costraints* are guidelines of all DD : we have reserved a part for all types of Policies and in case of future expansions will be easy add new local policies, privacy policies or other; we have provided a multi servers product so we could have a balanced load during days with striks; we have provided to reserve a part of DB for those datas that need more protections

6. Final Considerations and References

In our opinion is not necessary to have a document with a lot of details in the first version. In thi way during the future update will be more easly review the concept.
During the drafting of the document we have keep on mind what we had explained with RASD, in order to not create any contradiction.
In architectural part we have tried to create a very scalable product : with differents subsystem should be less difficult add new functions on one of them or change the structure of one of them.
For the algorithmic part we have prefered natural languages in order to not insert any constraints reguarding programming language.
In user interface part we have followed the idea that we had already presented in RASD, in order to create a sort of continuity between documents.
In some sections we had a very cautious approach becouse we had never seen the parts that were requested. In future version we may bring also importand upgrade with the objective of obtain a more complete document

Extra documentation consulted:

- Business Process Model and Notation (BPMN) - Version 2.0 provided by OMG (Object Management Group)