

**MyTaxi**

**Design Document**

Authors:

Bucci Giovanni

De Togni Riccardo

**Summary**

[**1.** **Introduction** 3](#_Toc436323771)

[**1.1.** **Purpose** 3](#_Toc436323772)

[**1.2.** **Scope** 3](#_Toc436323773)

[**1.3.** **Glossary** 3](#_Toc436323774)

[**1.4.** **References** 4](#_Toc436323775)

[**1.5.** **Document Structure** 4](#_Toc436323776)

[**2.** **Architectural Design** 6](#_Toc436323777)

[**2.1.** **Overview** 6](#_Toc436323778)

[**2.2.** **High level components and their interaction** 8](#_Toc436323779)

[**2.3.** **Component View** 10](#_Toc436323780)

[**2.4.** **Component Interfaces** 11](#_Toc436323781)

[**2.5.** **Deployment View** 11](#_Toc436323782)

[**2.6.** **Runtime View** 11](#_Toc436323783)

[**2.7.** **Selected architectural styles and patterns** 11](#_Toc436323784)

[**3.** **Algorithm Design** 11](#_Toc436323785)

[**4.** **User Interface** 11](#_Toc436323786)

[**4.1.** **Design Overview** 11](#_Toc436323787)

[**4.2.** **User Experience** 11](#_Toc436323788)

1. **Introduction**
   1. **Purpose**

This document describes the specific architecture and design of “MyTaxi” project.

The focus will be on structural and design styles choices, expanding the thread already analysed in RASD document.

The design document in effect, starting from the requirements given in the RASD build a feasible architecture for the application.

* 1. **Scope**

The document will present different level views in order to describe clearly the architecture of the application. In particular will be presented the component view, both high and low level, the deployment view, the runtime view and a further description of user interface, analysed in its runtime flow.

* 1. **Glossary**

Below is reported the glossary already inserted in RASD:

* + - CUSTOMER: a generic person that use any part of the system service, it could be either a user or a guest.
    - DBMS: Database Management System, the set of machines and specific operation that allow the right Database working.
    - ADMIN/ADMINISTRATOR: it is a particular type of user that has administrative functions.
    - GUEST: a person who has not signed up yet. Guests have no power until they sign up with one exception. If a Guest just want to call a taxi, it could simply insert its identification data.
    - USER: a person that has already signed up as a customer. It could call a taxi, as guest does, but it also could reserve it in advance, compiling a specific form.
    - TAXI DRIVER: a person who has signed up as a taxi driver. In order to complete its registration it has to provide its identification data and its driving license too.
    - SYSTEM: the environment formed by the application itself and its features.
    - CITY ZONE: each city is divided in zones. Every zone has approximatively the same territorial extension, so a city zone is one of the portions of the metropolitan area.
    - QUEUE: an ordered list of taxi drivers that have previously provided their availability.
    - CALL A TAXI: the action which can be performed both by guests and user, that consists in asking for a single taxi ride without any advance.
    - RESERVE A TAXI: the action that could be performed only by Users. A user can forward the request for a taxi from a specified place to another in advance.
    - SERVICE: the service that is provided by the application.
    - DENY/DENIAL: when a request is not satisfied. It produce the shifting of the considered taxi driver to the bottom of the queue.
    - ACCEPT: when a request, both coming from a reservation or a taxi call, is taken by a taxi driver who assumes the charge to bring passengers to the destination.
    - UI: the user interface i.e. the set of web pages that constitute the meeting point for users and system.
  1. **References**
     + Requirements and Specification Document, RASD
     + IEEE Standards for Information Technology Systems, Design Document
  2. **Document Structure**

The Design Analysis is based on a Top-Down approach, therefore the Document structure will follow the same path. It will start from the high-level architecture, presenting the main components with their operations and mutual relations (2.2). Then it slides down to a lower level in which the high-level components are decomposed and analysed in detail (2.3).

After that in 2.4 paragraph will be presented the Deployment view of the system that show the execution architecture of the system representing the deployed software and hardware artifacts.

Runtime view presented in 2.5 paragraph will show the behaviour of the system during some typical situation. That will permit to understand in an easier way how the execution flow works.

The lowest level of analysis is reached in paragraphs 2.6 and 2.7 where are described the inner composition of the Interfaces and the Architectural choices which have been made to design the system.

Lastly, will be presented some of the main algorithms that are the fulcrum of the entire system. They will be analysed with UML Sequence Diagrams.

Chapter 4 will take the User Interface already presented in RASD and give further information about the design choices. It will also describe some UI flows clearing up the navigation through different web pages or mobile screens. The paragraph about User Experience will get over the mere “Look Requirements” presented in RASD and it will deeply analyse the structure of the Web Application.

1. **Architectural Design**
   1. **Overview**

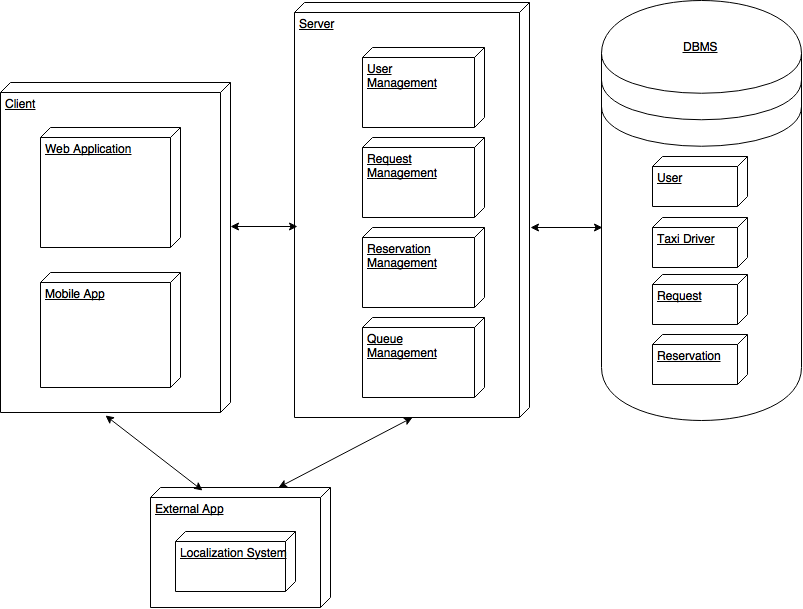
The design of the application is based on a 3-tier distributed system, where the three parts are Client-tier, Business Logic-tier and Entity-tier:

* + - CLIENT-tier: The different type of client application such as Web Application Client (Browser) or mobile app client composes client-tier. It collects the Users’ data and it is responsible of the delivery to the proper control unit that is part of the Business Logic. On the other hand, it is responsible for data receiving that consists in the safe delivery to the client, without loss or steal of data.
    - BUSINESS LOGIC-tier: as mentioned above Business Logic is the core of the application, it provides the information about how the system objects are related and how they interact, it shape the message format and it also coordinate the intercommunication between clients and entities.
    - ENTITY-tier: the entity tier contain the information about the system data model. It is responsible of database modification such as the insertion or the retrieving of any kind of data.

Given that architecture, it is possible to think that the Business Logics are represented by some Web Servers each one with a specific task. Entity-tier is represented by DBMS.

The situation described above is explained in the following diagram that provide a high-level view of the system structure:

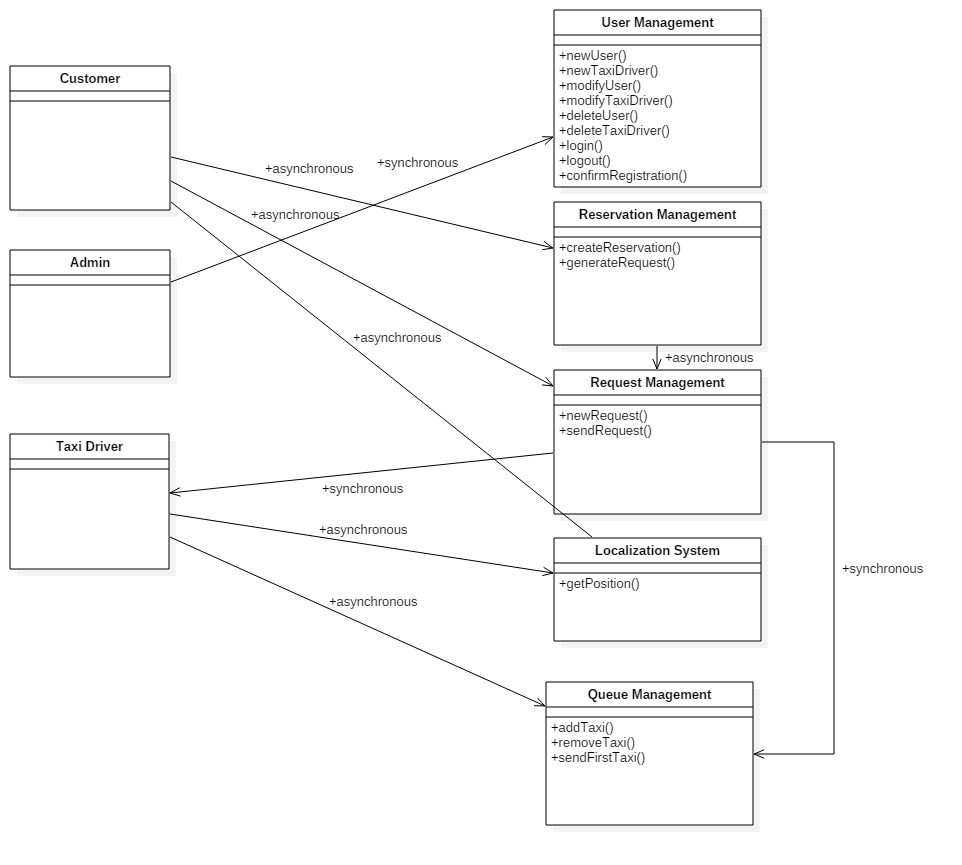
*System Overview*



* 1. **High level components and their interaction**

In this paragraph the raw architecture presented above will be decomposed and analysed, focusing on which parts are related and which kind of communication is. The main components will be also provided with the operations that each one can perform.

*High-Level Component view*

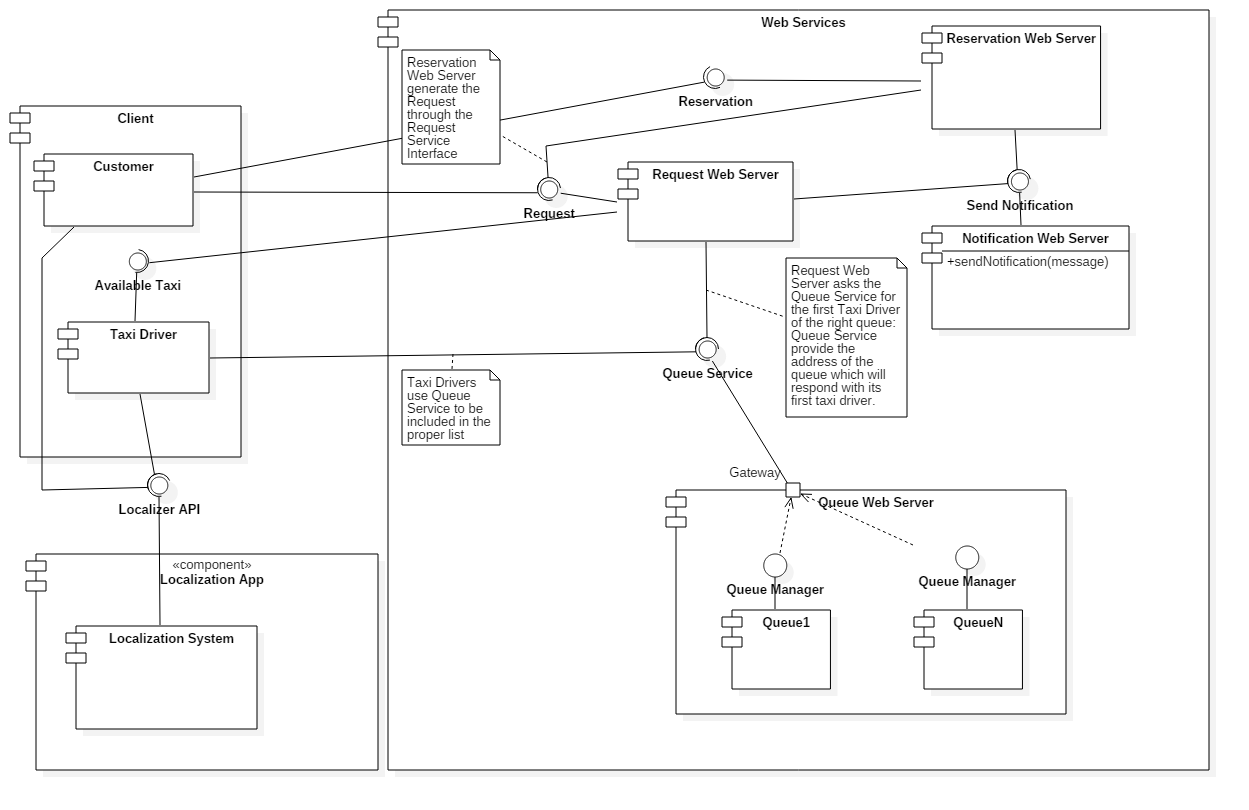


Now every relation will be described in detail:

* + - CUSTOMER – REQUEST MANAGEMENT: a generic costumer may start at any time a communication with the Request Management component, through the provided interface (That part will be described in the next paragraph). The communication is asynchronous since the Request Manager is always listening for new requests and it can handle more than one communication at a time.
    - CUSTOMER – LOCALIZATION SYSTEM: a customer start an asynchronous communication with the Localization System when it accesses the application.
    - CUSTOMER – RESERVATION MANAGEMENT: the reservation management system is always ready to accept a new reservation from any customer. To allow that the communication is asynchronous.
    - ADMIN – USER MANAGEMENT: an administrator can communicate with the User Management Component in order to view, modify or delete any registered user. Since the system permits multiple administrator the communication must be synchronous to grant concurrency.
    - TAXI DRIVER – LOCALIZATION SYSTEM: Taxi driver implicitly start a communication with Localization System when it provide its Availability. Since the Localization System has to handle multiple communications the message exchange is asynchronous.
    - TAXI DRIVER – QUEUE MANAGEMENT: As the communication with Localization System this relation starts when the taxi driver provide its availability. Queue Management handles all the taxi drivers so it can accept many request at the same time: the communication is asynchronous.
    - REQUEST MANAGEMENT – QUEUE MANAGEMENT – TAXI DRIVER: this relation is ternary because it represent a whole communication cycle. The request management has to associate its pending request with a taxi driver who will bring passengers to the destination. In order to find a feasible taxi driver the request management “ask” the queue management who is the first taxi driver in the specific queue, identified by the start address of the request. Once the request management receives the taxi driver information, it directly start a communication with Taxi Driver to send it the request. This loops until a taxi driver accepts the request. The communication between Request management and Queue management is synchronous in order to avoid simultaneous requests (Queue management respond a request with the same taxi driver). For similar reasons also the communication between Taxi Driver and Request Management is synchronous. In fact, it will be granted that the request management do not ask for another taxi driver until the former has denied the request.
  1. **Component View**

Now the High-Level Components view presented above will be decomposed and analysed in detail. In particular will be highlighted the subcomponents and how they behave. A component behaviour is described by provided and required interface, and its relation with other components.

*Component View*



The Component Diagram is divided in three macro components:

* + - Web Services
    - Client
    - Localization App

Web Services represent the main component. It contains many sub-components, one for each service that the system can provide. Even if it has been used the term “Server”, the sub-components are not independent hardware entities. They are just executable pieces of software contained in the same environment.

* + - The Request Web Server handle the Request Management already described in the High Level Component View (2.2). It provide a Request Interface available for Clients and for an inner component, the Reservation Web Server, which will be presented soon. It has to be able to communicate with the Queue Manager, from that comes its required interface Queue Service. The use of this interface is explained in a note inside the diagram and it will be deeply analysed in the next paragraph. Further when a request it is accepted by a taxi driver the Request Web Server has to send a notification to the User and to the taxi driver itself. That explain the second required interface: Send Notification.
    - The Reservation Web Server handle the Reservation Management, it interact with the Client component through its provided interface, Reservation, and with the Notification Web Server through the required interface Send Notification. It also communicate with the Request Web Server via Request interface, since the Reservation Web Server has to generate a request ten minutes before the related reservation time.
    - Queue Web Service hides its inner composition to the rest of the Web Servers since the only way to communicate is to use its provided interface Queue Service. It contains one software component for each queue, so for each city zone (see Glossary). When the Request Web Server ask for the first taxi driver of a particular queue the Queue Web Server transparently redirect the request to the right Queue. Each Queue provide a Queue Manager Interface that can respond the Request Web Server with the required data just sending it to the Queue Service interface through the Queue Web Sever gateway.
    - Notification Web Server handles all the notification functionality. It provide a Send Notification interface through which the other components may communicate their intention to send a notification. Both Request and Reservation Web Servers use this interface.

The second Macro Component represent all the kind of client that can access the service, so it integrate Customer component and Taxi Driver component.

* + - Customer symbolize a generic utilizer, which may be both a Guest and a User. It requires two different interfaces: one to make a request (Request Interface) and the other one to make a reservation (Reservation Interface).

When it make a Request or a Reservation using the mobile app, the position is automatically calculated by a Localization System, asked by the customer through its Interface.

* + - Taxi Driver component embodies the client-side service provider. The Request Web Server can communicate with it through the Available Taxi Interface. A taxi driver can give its availability to the Queue Web Server through its Queue Service Interface. In order to permit the Queue Web Server to redirect the Taxi Driver to the right queue, jointly with its availability, it has to provide its position too. For this reason it transparently(without any alert) connects with a Localization System through the specific interface which will provide the position.
  1. **Component Interfaces**
  2. **Deployment View**
  3. **Runtime View**
  4. **Selected architectural styles and patterns**

1. **Algorithm Design**
2. **User Interface**
   1. **Design Overview**

The idea, always present within the project, is to create a tool that is easy and immediate for the user, and the user interface is designed to meet these requirements.

From the mock-ups already presented in the RASD, it is in fact clear that the aim is to present the users a neat and minimal interface, in order to make the procedures intuitive and as quick as possible.

* 1. **User Experience**

Here it is presented a diagram that, with the proper stereotypes ("page" and "form"), shows how pages are related, which important components are present, every input form, and how the navigation through the website is structured.

The graph is presented as a class diagram, and the symbols have the same meaning as if they were used in that kind of diagrams; it is important to notice that <<page>> means that the class represents a web page, and <<form>> identifies an input form contained in a specific page.

The home page structure is the same for all the users, focused on the fast request for a taxi; every user home page is then developed from this point, adding links and features associated to the relative user.

Following these links every customer can navigate through the pages, but only through those for which he has permission.

Here are presented only direct flows, associations representing cancellations or links to previous or home pages are omitted in order to simplify the reading of the graph.