

Release date 04-12-2015

Software Engineering 2: MyTaxiService

Design Document V1.0

Dimitar Anastasovski, Marco Colombo

Contents

1	Introduction	2
1.1	Purpose	2
1.2	Scope	2
1.3	Definitions	3
1.4	Abbreviations	3
1.5	Acronymous	3
1.6	Document Structure	3
2	Architectural Design	5
2.1	Overview	5
2.2	High level components and interactions	5
2.3	Component view	6
2.4	Deployment view	7
2.5	Runtime view	8
2.6	Component interfaces	11
2.7	Selected architectural styles and patterns	13
3	Algorithm Design	16
4	User Interface	18
4.1	User interface design	18
5	Requirements traceability	28
5.1	From requirements to diagrams	28
6	References	33
6.1	Software and used Tools	33
6.2	Working hours	33

Chapter 1

Introduction

1.1 Purpose

This document describes the system architecture and design of MyTaxiService system. Each component is described in terms of its external interfaces and the dependencies on other components. It will form the basis of the planning of the development phase and will be updated using the results from the various development iterations to come to complete architectural design. The architecture review will allow us to identify what are the weaknesses or strengths of the proposed architecture as well as we hope to obtain a list of suggested changes to improve it. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This document applies to the overall design of the system. It contains information relating to the architectural design of the software, the Structure of the Database, and of the physical servers hosting the site. The main accent is to simplify and optimize the access of passengers to the system and to guarantee fair management of taxi queues. We will build flexible and user-friendly web application and a mobile application that will run on Android and IOS mobile phones. This application can be used by anyone who previously will be register on the registration page. After the registration is done the user will have a user name and password that should remember for furthermore usage of the system. The passenger can call a taxi after a successful logging on the application. After that he can call a taxi and he will be informed about the code of the incoming taxi, waiting time. On the other hand taxi drivers will have a mobile application where the major purpose will be to inform the system about their availability, confirmation of a certain call and global

map navigation. City is divided into taxi zones that are uniquely associated with corresponding taxi queues for efficient usage of the system.

1.3 Definitions

- **Request:** Passenger filled form for immediate ride
- **Reservation:** Passengers can request for a vehicle at least 2 hours before the ride and can reserve his ride
- **User:** Is a customer who already registered and logged into the system
- **Taxi driver:** Is a person who legally drives taxi (with driver license and work license) already registered and logged into the system as a driver
- **System:** Is the system that has to be designed
- **Taxi zone:** Are the zones in which the city is divided in

1.4 Abbreviations

No abbreviations are been used in this document

1.5 Acronymous

- IEEE Standard 1016-2009 for Information Technology–Systems Design–Software Design Descriptions
- Specification Document: myTaxiService Project

1.6 Document Structure

This document is divided in four parts with clean and non-ambiguity description of the whole system.

- Chapter 1: General description and basic information of the system
- Chapter 2: Architectural design, which introduces the components of the system and describes the way components interact with specific tasks.
- Chapter 3: Algorithm design, focused on the most interesting part of the system

- Chapter 4: User interface design, providing an overview on how the user can interact with the system.
- Chapter 5: Requirements traceability, that explains the requirements defined in the RASD into the design elements.

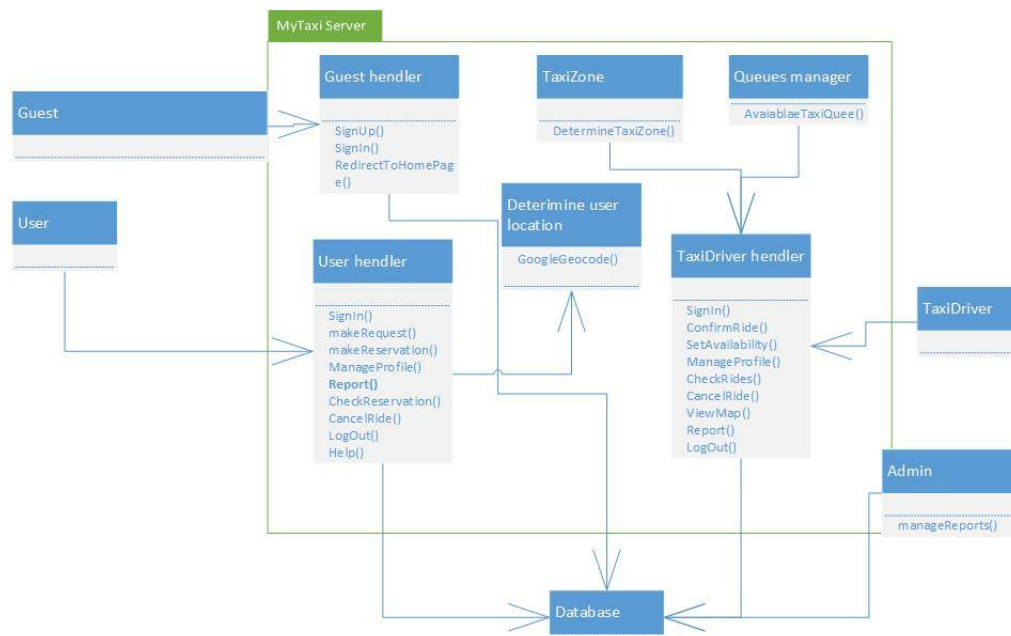
Chapter 2

Architectural Design

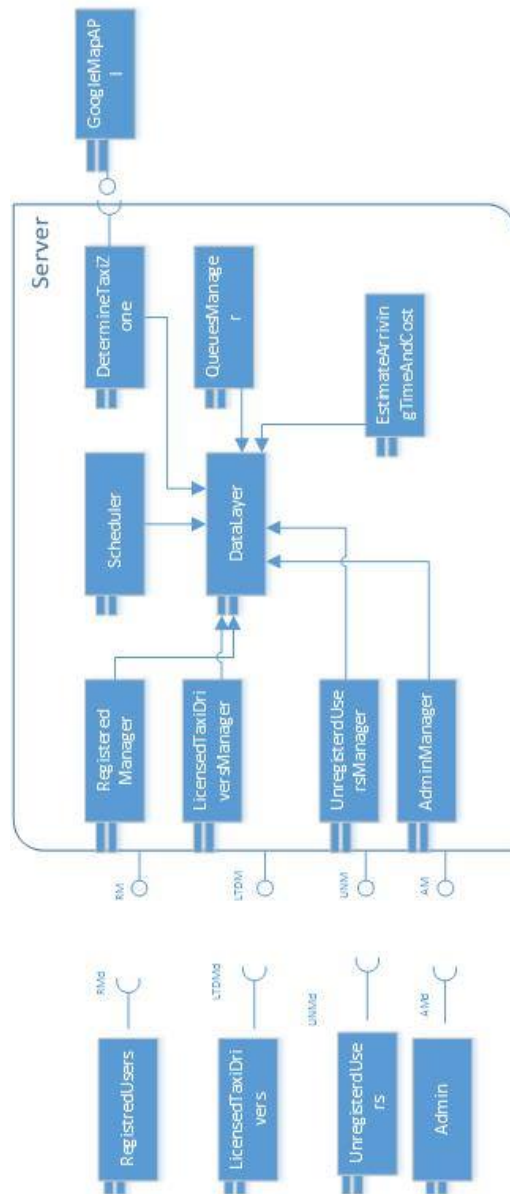
2.1 Overview

This document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

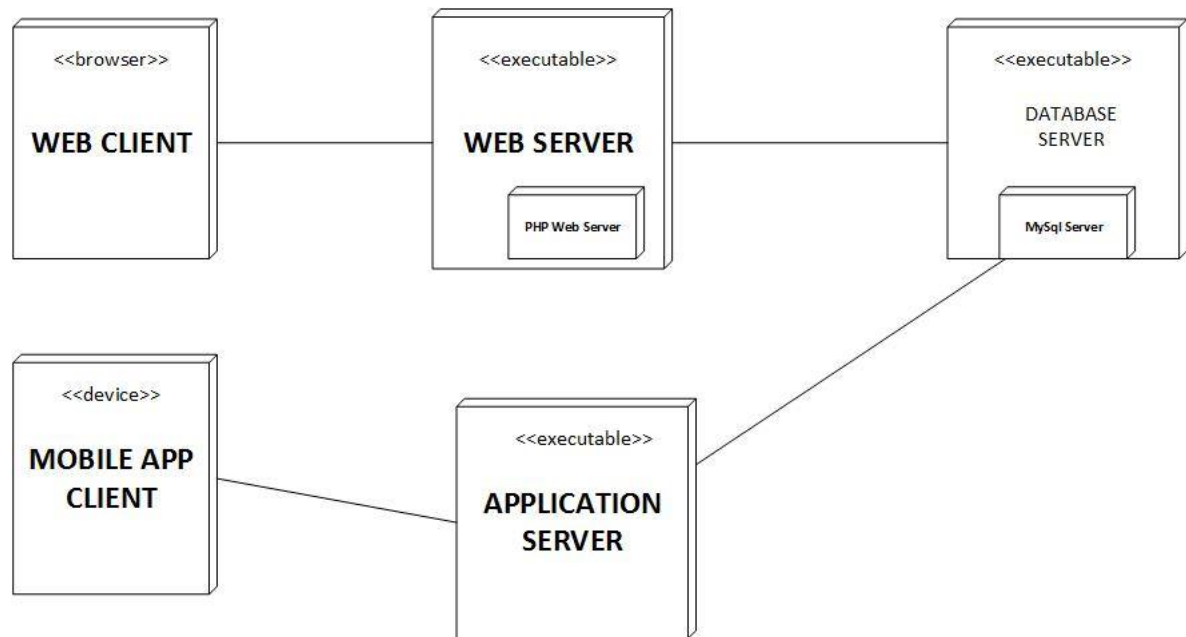
2.2 High level components and interactions



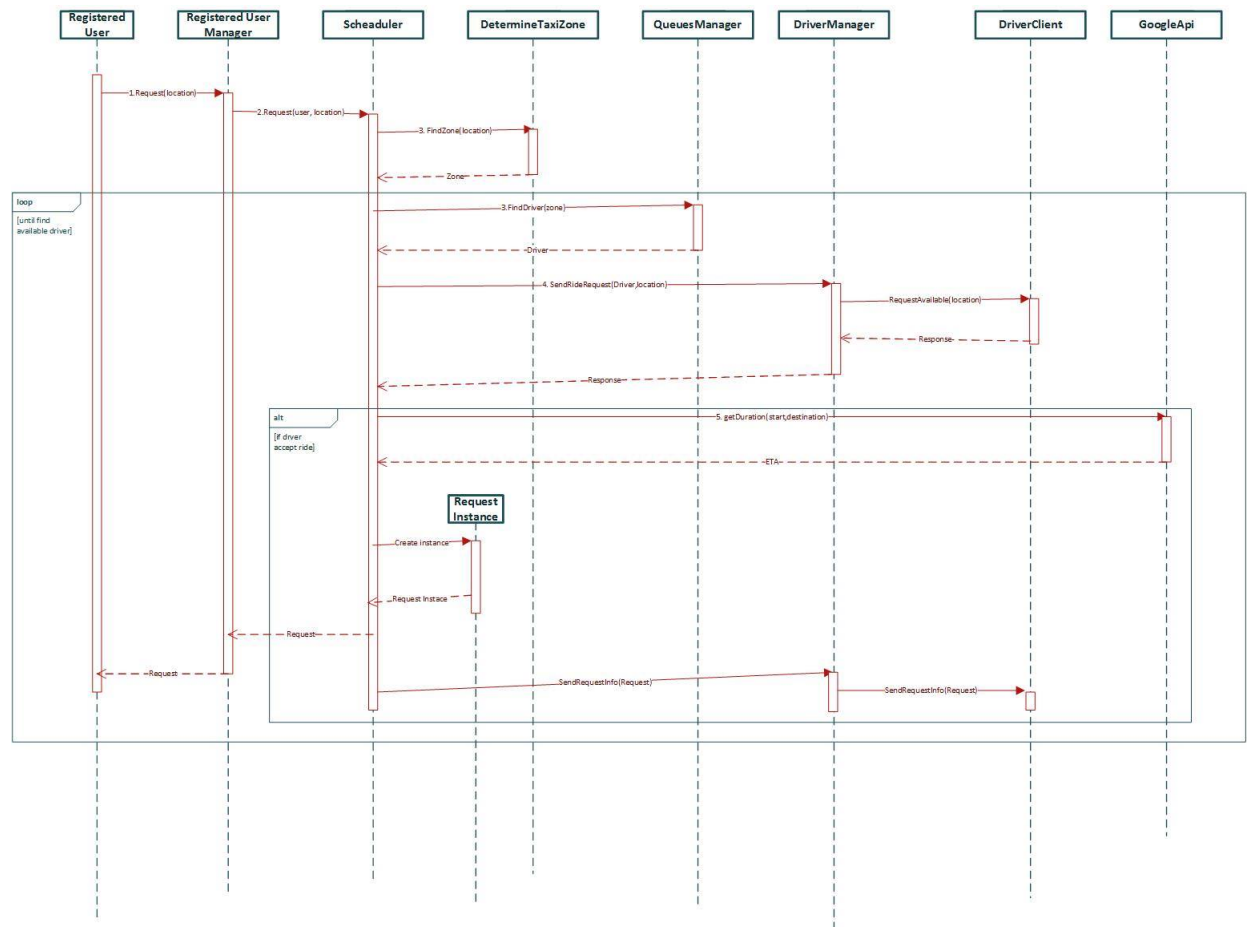
2.3 Component view

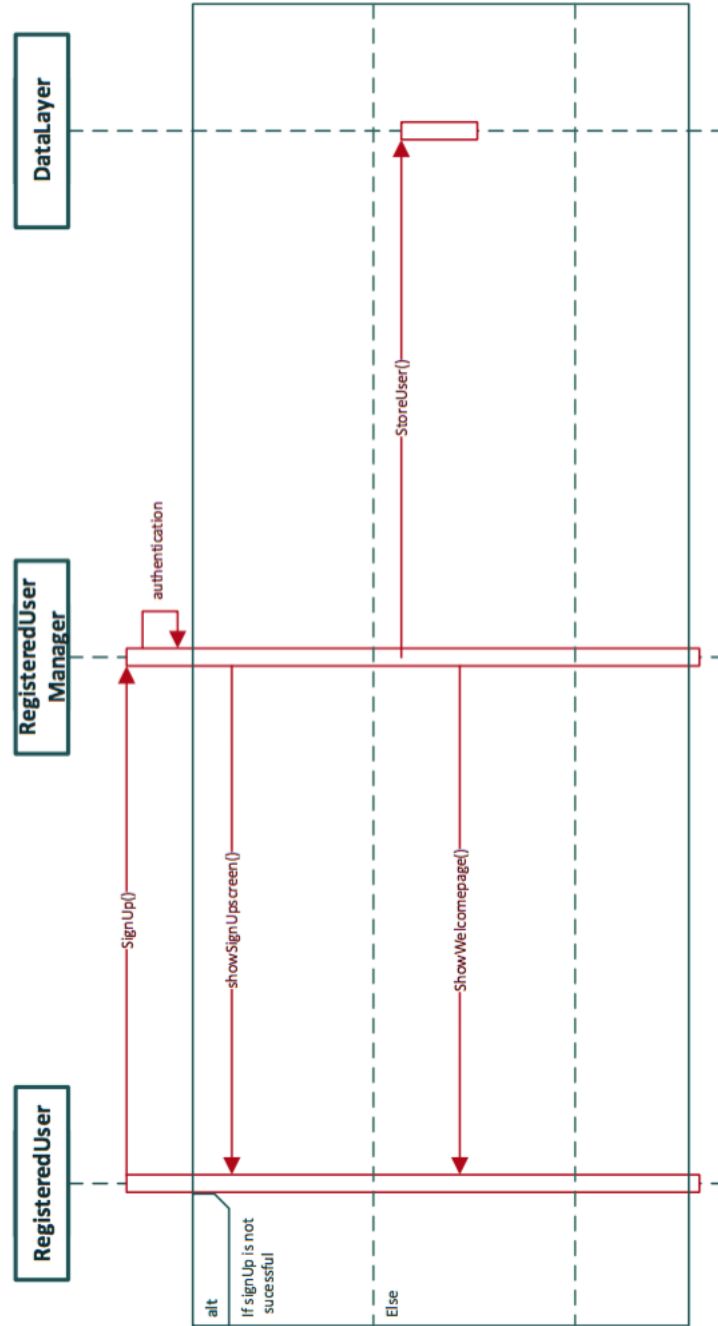


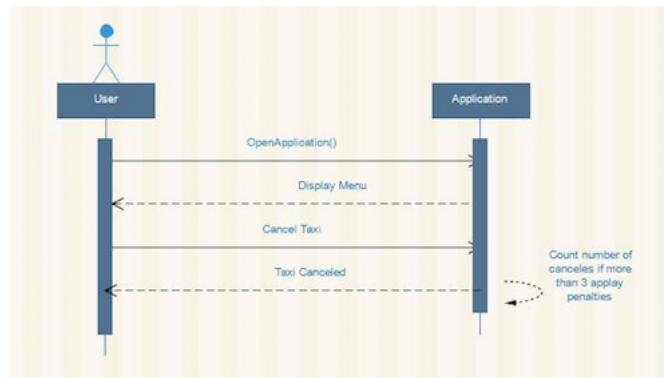
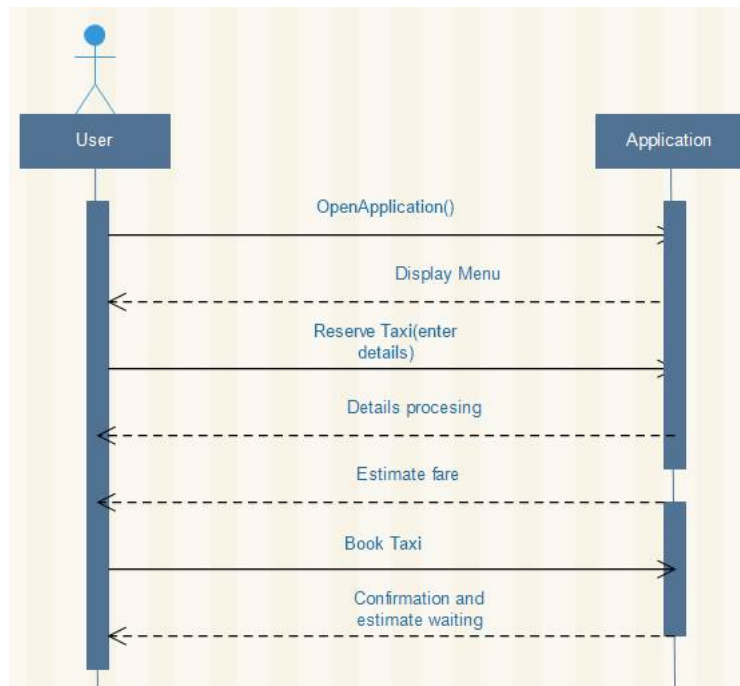
2.4 Deployment view



2.5 Runtime view





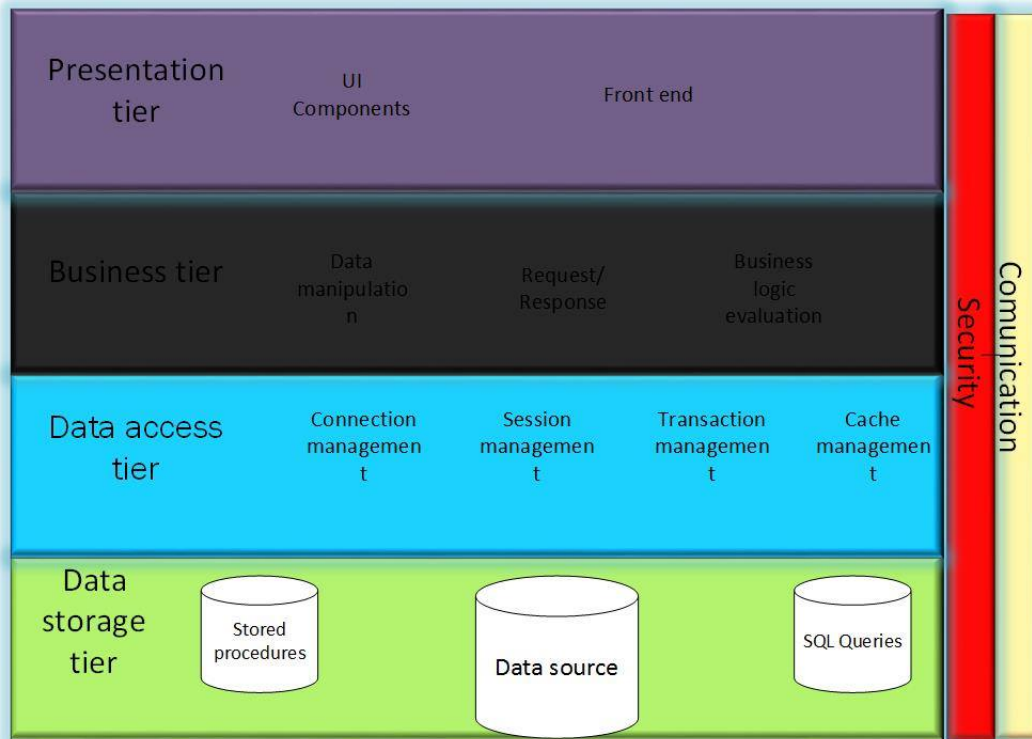


2.6 Component interfaces

- **RegisteredUsers:** is a module where clients who already sign up can access their account and can use the features of the application. This component contains sign in module for authentication the users and its communicating with **RegisteredManager**.
- **LicensedTaxiDrivers:** is a module where licensed taxi drivers access their account and can use the features of the application. This component contains sign in module for authentication the users and its communicating with **LicensedTaxiDriversManager**.
- **UnregisteredUsers:** is a module where guest who don't have an account can sign up for using the features of the application. If the guest click on the sign up button it will be redirected to the sign up form for entering basic information.
- **RegisteredManager:** this module is connected with registered users who already sign up and have their own account. It managing all registered users showing them the home page, notifying the users and etc. and is communicating with **DataLayer** for storing, deleting or retrieval information that is needed.
- **LicensedTaxiDriversManager:** is managing licensed taxi drivers who are already sign up and have their own account. It is communicating with the taxi drivers notifying them about taxi requests, reservations and their cancellation. It can send and receive notification from the taxi drivers, update information about their availability and accept or declined requests. Also it's communicating with **DataLayer** for storing, deleting or retrieval information that is needed.
- **UnregisteredUsersManager:** this module is managing the whole registration process for the guests who don't have an account. It is communicating with **DataLayer** for notifying the system for a new profile and to store the data in the database system.
- **DataLayer** module is managing the whole data of the system. This module is encrypted and can create, delete, modify, store and update informations when it is required.
- **Scheduler** module is managing the informations that is needed for the taxi drivers and users.

- **DetermineTaxiZone** module is determine the zone for the taxies so when the user request taxi this module is response for calling the taxi nearest to the customer. Also is taking care to have availably taxis in every zone.
- **QueuesManager:** this module is providing functions to manage all the taxis queues of the city. It communicate with **LicensedTaxiDriversManager** in order to retrieve information about taxi?s availability and eventually move it to the end of the queue.
- **EstimateArrivingTimeAndCost:** this module is estimating the arrival time and the cost after a user successful book a taxi.

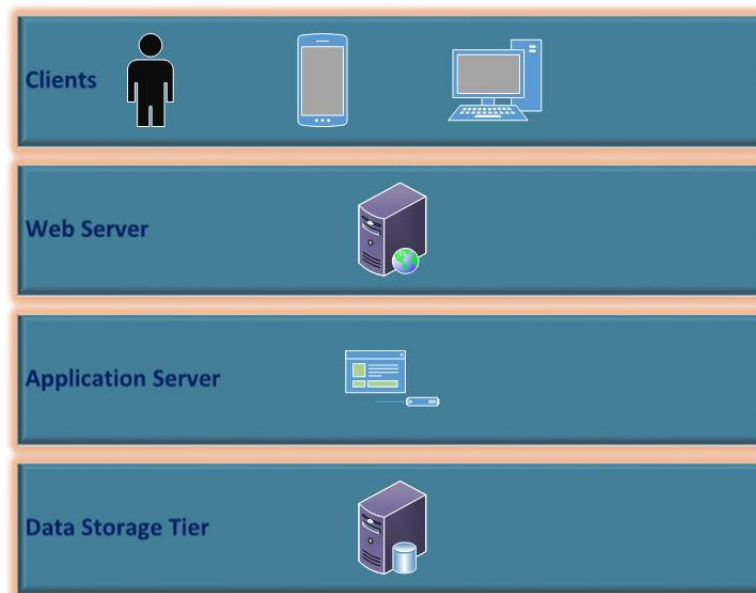
2.7 Selected architectural styles and patterns



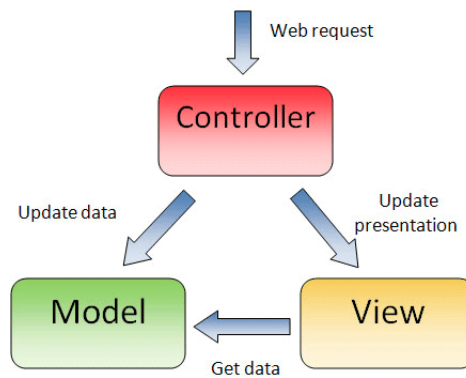
- **N-Tier** data applications are data applications that are separated into multiple tiers. Also called "distributed applications" and "multitier applications," n-tier applications separate processing into discrete tiers that are distributed between the client and the server. A typical n-tier application includes a presentation tier, a middle tier, and a data tier. In addition to the advantages of distributing programming and data throughout a network, n-tier applications have the advantages that any one tier can run on an appropriate processor or operating system platform and can be updated independently of the other tiers.
- **Database (Data) Tier** - At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier** - At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of

any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

- **User (Presentation) Tier** - End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.
- **Multiple-Tier** database architecture is highly modifiable, as almost all its components are independent and can be changed independently.



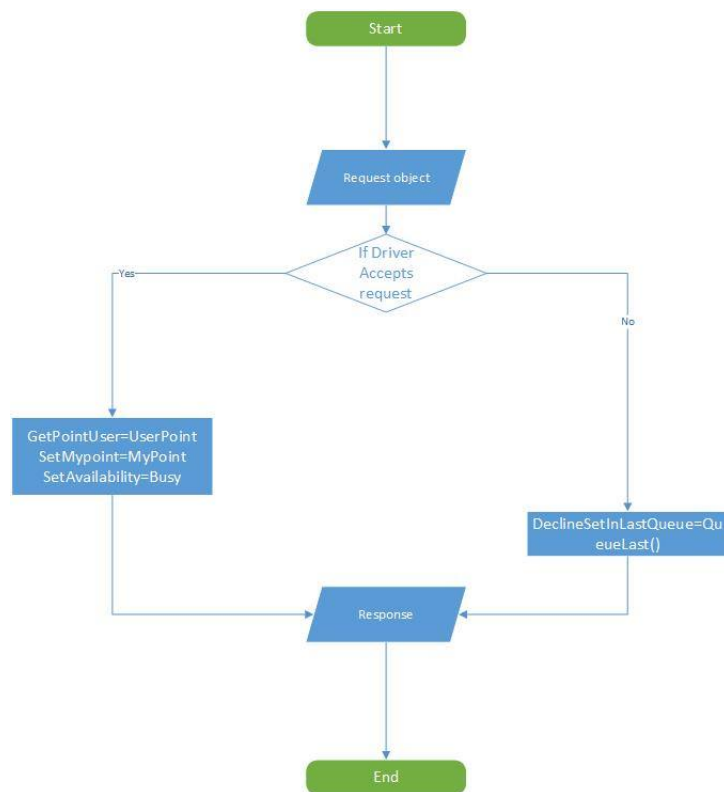
The model-view-controller pattern proposes three main components or objects to be used in software development:

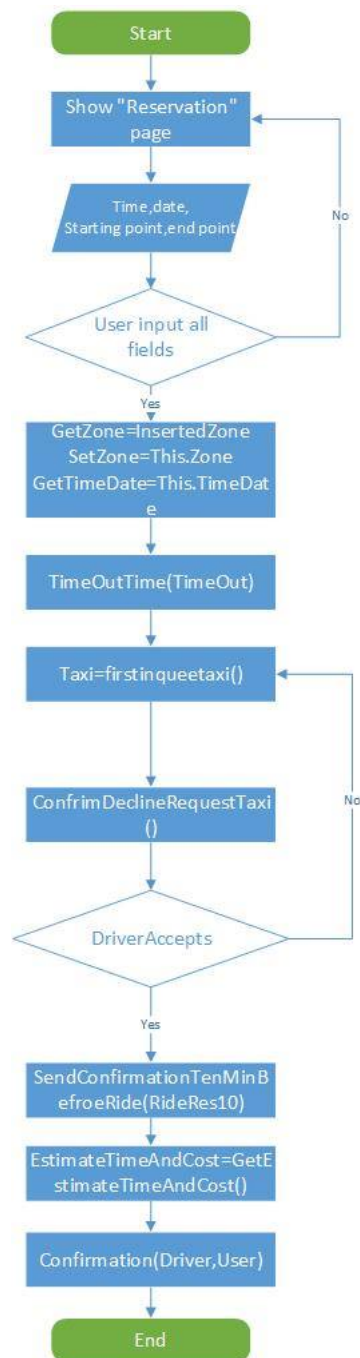


- **Model**, which represents the underlying, logical structure of data in a software application and the high-level class associated with it. This object model does not contain any information about the user interface.
- **View**, which is a collection of classes representing the elements in the user interface (all of the things the user can see and respond to on the screen, such as buttons, display boxes, and so forth)
- **Controller**, which represents the classes connecting the model and the view, and is used to communicate between classes in the model and view.

Chapter 3

Algorithm Design



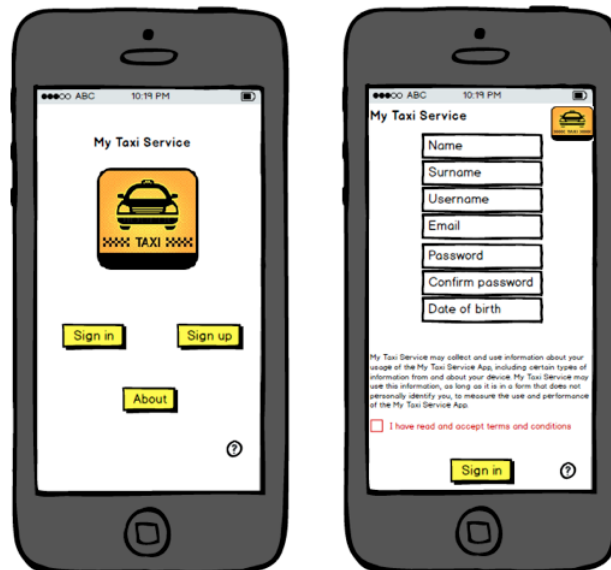


Chapter 4

User Interface

4.1 User interface design

Homepage & registration page



The passenger can access to the registration page and register himself to the application.

Login



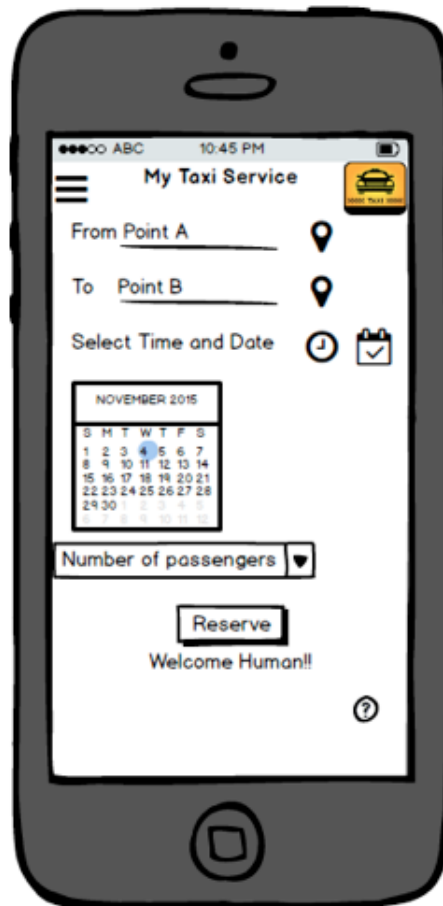
This is the login page. Here the user after entered his data can sign in the application.

Menu



This is the menu page, here the user can navigate and choose what to do. There are several options like: reserve taxi, check reservation, profile, help or log out.

Reservation



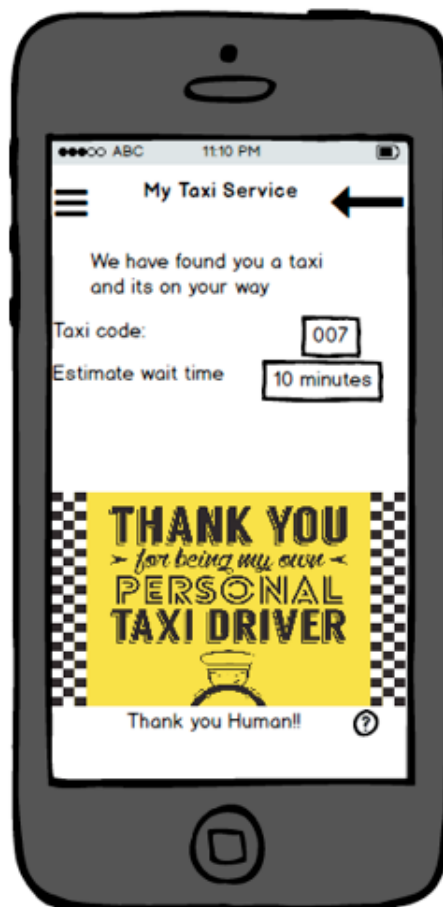
Here the passenger can reserve a taxi. He needs to insert the starting and destination point, the date and the number of passenger.

Confirm reservation



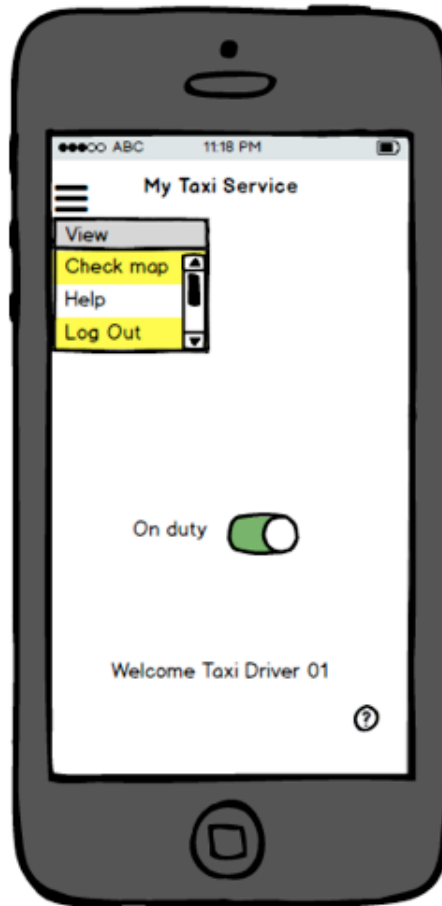
In this page the passenger can show a recap of his trip and confirm the reservation by clicking on the button

Recap reservation



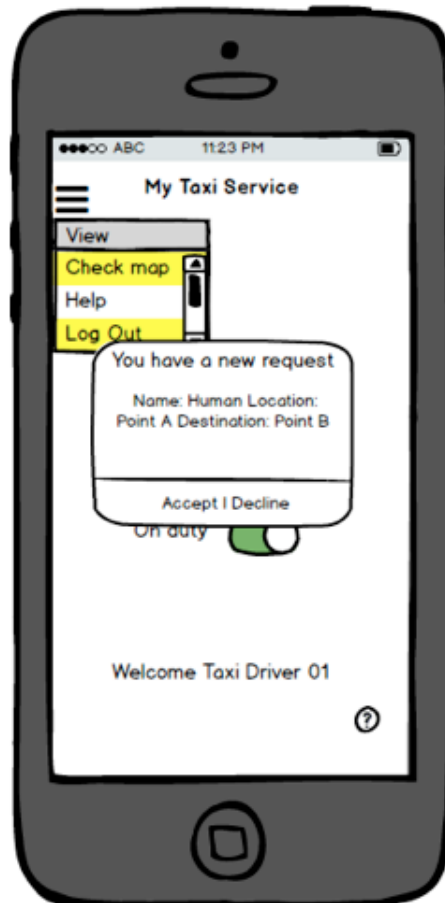
Here the passenger can see the estimate waiting time and the code of the taxi that will pick him up.

Taxi driver homepage



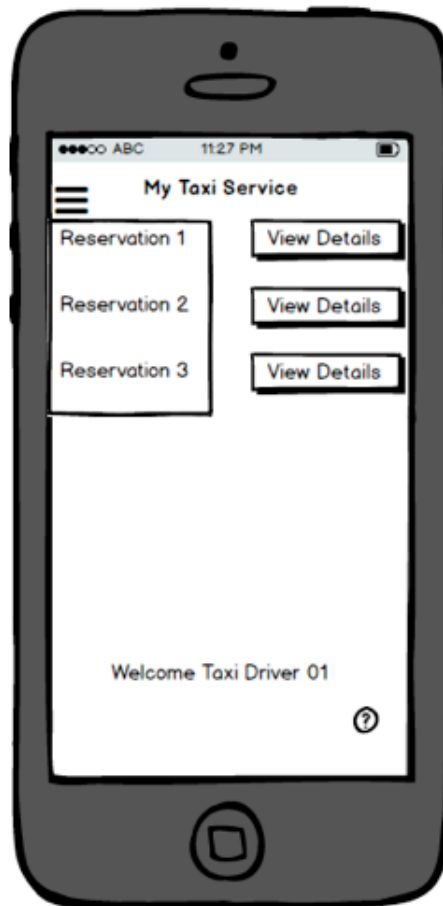
This is the home page for the taxi driver. He can choose from different option such as: Check map, Help or Log out. Also he can change his status in busy or free.

Taxi driver request



This is the notification that the taxi driver will receive when there is a request of a taxi. He can choose if accept or decline this request.

Reservation list for taxi driver



Here the taxi driver can give a look at the reservation he has in list and its details.

Taxi driver map



In this page the taxi driver can see the map of the itinerary for the reservation he chooses.

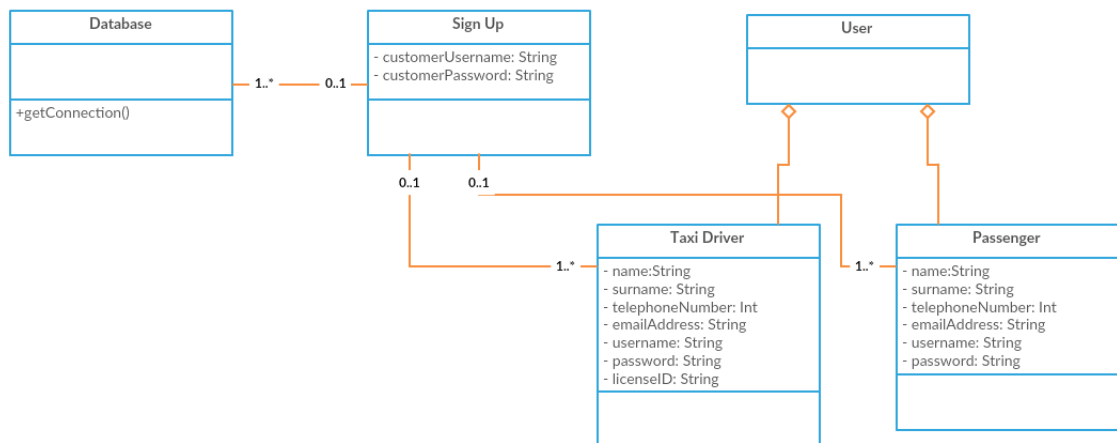
Chapter 5

Requirements traceability

5.1 From requirements to diagrams

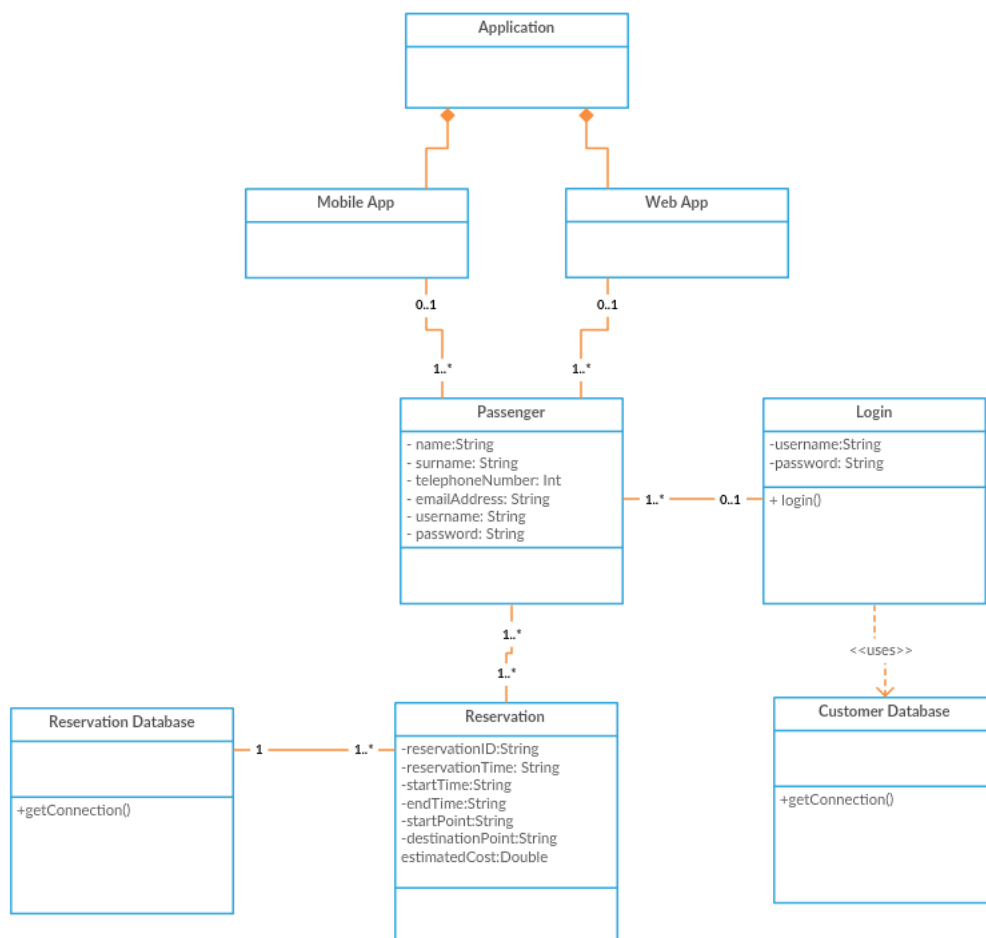
1. Allow the registration in the application to the taxi drivers and passengers:

- The system should be able to provide a sign up functionality in order to create a new taxi driver or passenger profiles.



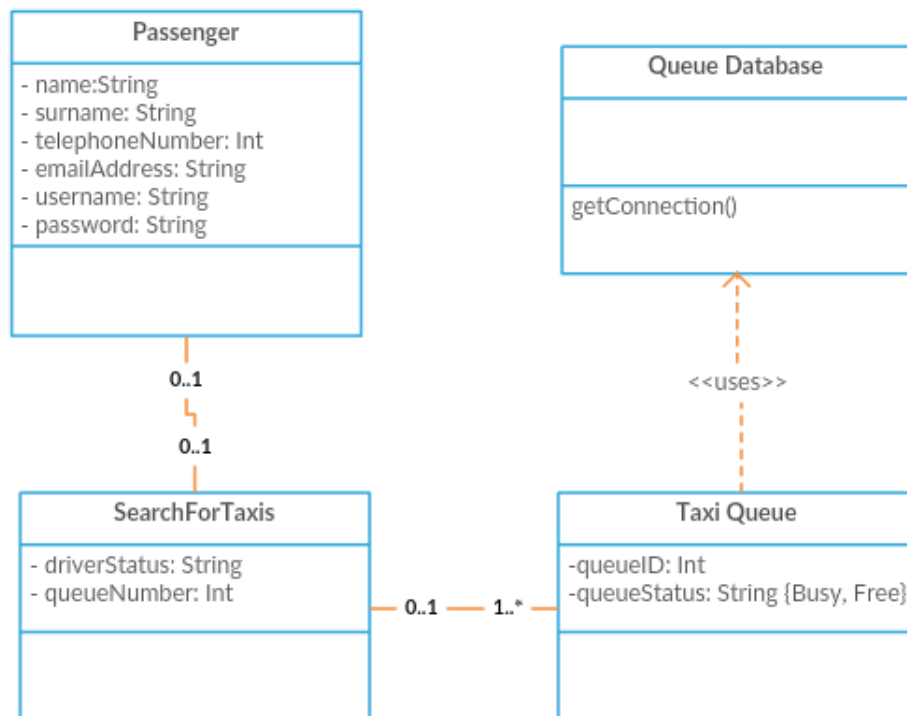
2. Simplify the access of passengers to the taxi service:

- The system has to provide the possibility to the passenger to reserve a taxi
- The system has to provide both a web application and mobile application in order to allow the passenger to save his time.



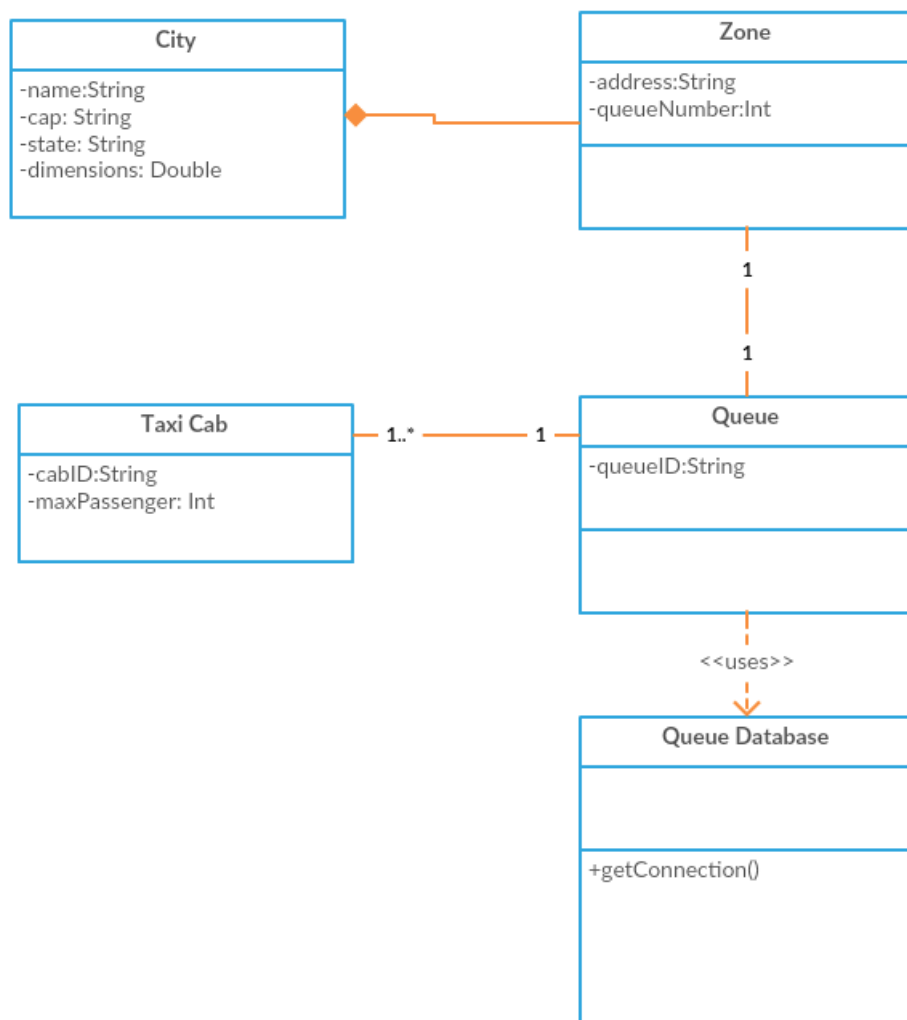
3. Allow passenger to view if there is any taxi driver free:

- The system has to provide a functionality that allow a passenger to view if there is any free taxi driver for the desired route.



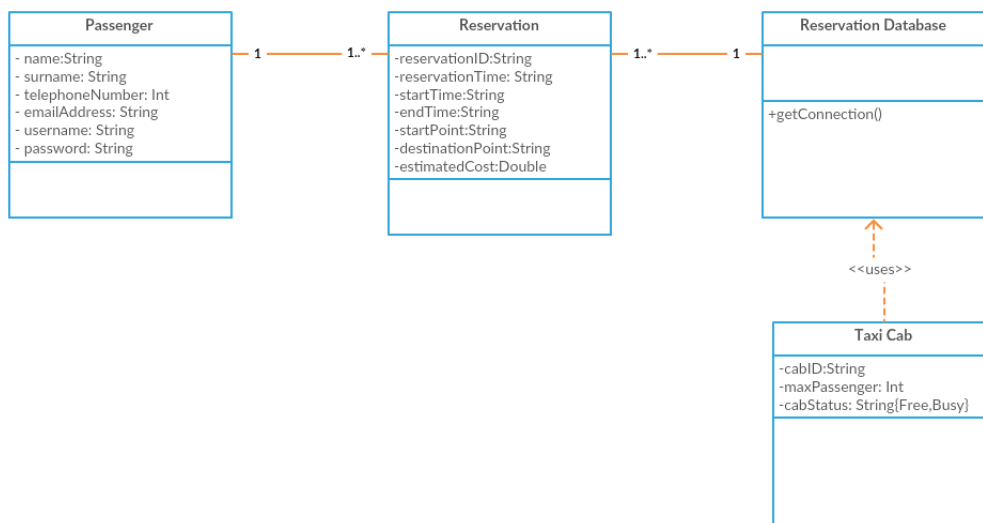
4. Divide the city into different areas covered by some taxi drivers, organized in queues:

- The system has to provide a functionality to divide the city in some areas
- The system has to organize the taxi of these areas into queues



5. Allow the client to reserve taxi for an exact time:

- The system has to provide a functionality that allow the passenger to reserve a taxi for a desired time.



Chapter 6

References

6.1 Software and used Tools

- TexShop (<http://pages.uoregon.edu/koch/texshop/>), to redact this document
- CreatelyApp (<https://creately.com/app/>), to create the class diagrams
- Balsamiq Mockups (<https://balsamiq.com/products/mockups/>), to create the user interface mockup
- Edraw Max(<https://www.edrawsoft.com>), to create the sequence diagrams
- Microsoft Visio (<https://products.office.com/it-it/visio/flowchart-software>), to create the algorithmic part

6.2 Working hours

Dimitar Anastasovski: ~30 hours

Marco Colombo: ~20 hours