myTaxiService
-
Design Document

Davide Cremona (matr. 852365), Simone Deola (matr. 788181)

November 18, 2015

Contents

Chapter 1

Introduction

This chapter is intended to give an overall description of this document, it contains various sections like:

- **Purpose:** in this section is described the purpose of this document.
- **Scope:** in this section is described the scope of the myTaxiService system.
- **Definitions, Acronyms, Abbreviations:** here are listed all the definitions, all the acronyms and all the abbreviations that the reader will encounter in this document.
- **Reference Documents:** here are listed all the documents that the reader may need to read to better understand what is written in this document.
- **Document Structure:** here is explained the internal structure of this document, giving a fast description of each chapter.

1.1 Purpose

The MyTaxi Service Design Document is intended to provide an explanation of how the system has been designed. It's also destined to give to the software development team an overall guidance to the implementation of the architecture of the software project. These goals are chieved by describing:

- The system architecture;
- Architecture hig-level components;
- Interaction between components;
- Patterns and Styles used to design these components.

1.2 Scope

The scope of myTaxiService is to simplify the interaction between Taxi Drivers and Customers. The main goal is to provide a system to request a taxi and ensure that this will arrive in a small amount of time. Also, customers can reserve taxis for a future ride. The system also ensures a fair management of taxi queues that are divided in city zones in orded to reduce waiting times and to provide a fair division of work between Taxi Drivers.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- Customer: a Customer is the end-user (a taxi customer) that makes request or reservations to use taxis.
- Taxi Driver: a Taxi Driver is a driver of a taxi. He can receive requests and accept or decline them.

1.3.2 Acronyms

- RASD (or R.A.S.D.): is the Requirement Analysis and Specification Document.
- DD (or D.D.): is the Design Document (this document).
- MVC (or M.V.C.): is the Model-View-Controller software design pattern.
- DBMS (or D.B.M.S.): it's the Data Base Management System.
- UI (or U.I.): it's the acronym for User Interface.

1.3.3 Abbreviations

Definitions Acronyms Abbreviations section

1.4 Reference Documents

You can refer to the Requirement Analysis and Specification Document for a better understanding of what is described in this document (myTaxiService R.A.S.D. Link).

1.5 Document Structure

This document is divided in 6 main Chapters:

- Introduction: this chapter is used to give a general overview of this document (purpose, references etc..).

- Architectural Design: in this chapter is described the general architecture of the system and his components. It's also described how the components interacts with each other and the Design Patterns and Architectural Styles used to design them.
- Algorithm Design: here is described the most relevants algorithms used to create the system.
- User Interface Design: here is described the end-user interface of the mobile and web applications.
- Requirements Traceability: here is described how the requirements described in the RASD document are implemented in the various components of the system.
- References: in this chapter, there are useful references to external resources that helps to understand this document.

Chapter 2

Architectural Design

In this chapter there will be described the architecture of the system and the various components that will compose the system. Here it's also described how these components will interact to each other to perform the various tasks that the system have to do to provide to the users the functions described in the RASD document. Finally, there will be described what architectural styles and patterns have been used to design the entire system.

2.1 Overview

MyTaxiService system architecture has been designed as a 3-tier architecture (Data Tier, Logic Tier and Presentation Tier) to facilitate the development of the system using the MVC software design pattern. In fact the three tier represents in some way the Model (Data Tier), the Controller (Logic Tier) and the View (Presentation Tier) of the software.
The architecture of the system is described by the following figure:

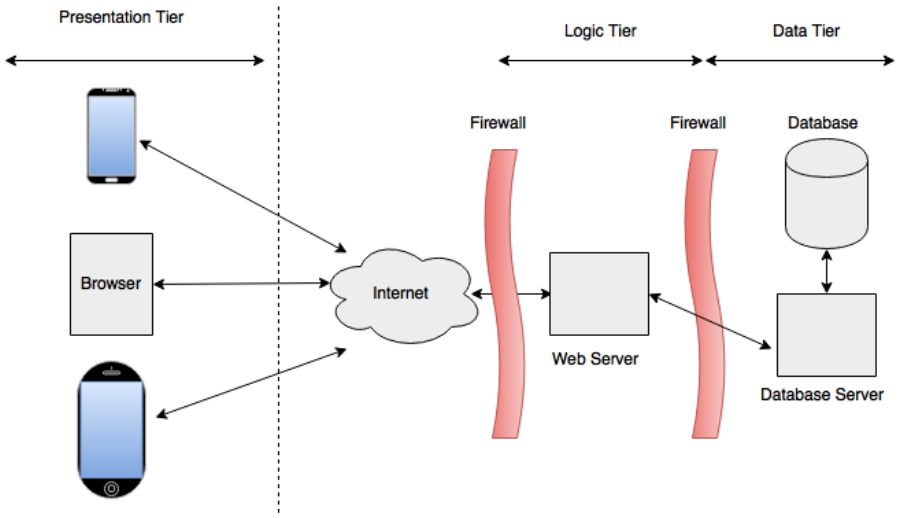

Figure 2.1: System Architecture

The components of the system are divided in the three tiers:

- In the Presentation Tier there are imoplemented classes and the objects that compose the view part (what the user see). From this tier users can perform actions to query the system and use the web and mobile applications.
- The Logic Tier is the place where are implemented classes and objects that compose the controller (actual logic of the system). This part is in charge of taking requests by the users applciations and perform changes on the data. This part is also in charge to notify the applications (presentation tier) of the data changes.
- The Data Tier is the place where are implemented the classes and the object that interacts directly with the database. These objects represents the entities of the database and are able to perform creation, modification and deletion of database records.

2.2 High Level Components and Their Interaction

In this section each Tier is decomposed in his high level components. For each component is given a short description of his purpose and how the component interacts with the others components.

2.2.1 Data Tier

Ther Data Tier is the part of the system that include the database and all the classes and the objects that will interact with it. It's composed by these components:

- **Database:** it's the data structure where all the system data are stored. The management of this component is delegated to the DBMS. The DBMS will provide a query language to extract data from the database.
- **Database Manager:** this component represents the Model of the MVC design pattern, it will include all the objects and the classes of the Data Model. This component also contains all the classes that are in charge to perform queries to the DBMS in order to make the data available to the Logic Tier.

2.2.2 Logic Tier

In this Tier there are implemented all the classes and the objects that are used to logically organize the requests and the responses from/to the Presentation Tier and the Data Tier. This Tier represents the Controller of the MVC software design pattern and it's composed by these components:

- **IO Manager:**
- **Requests an Reservations Manager:**
- **Zones Manager:**
- **Profile Manager:**
- **Facebook API Manager:**

2.2.3 Presentation Tier

The purpose of this tier is to make the data and their changes visible to the user. Presentation Tier is also in charge to make the system functions available to the users according to their permissions (see the RASD to read about functionality differences between Customers and Taxi Drivers). This Tier represents the View of the MVC design pattern and it's composed by these components:

- **Web Application UI:**
- **Mobile Application UI:**

2.3 Component View

component view section

2.4 Deployment View

deployment view section

2.5 Runtime View

runtime view section

2.6 Component Interfaces

component interfaces section

2.7 Selected Architectural Styles and Patterns

selected architectural styles and patterns section

2.8 Other Design Decisions

other design decisions section

Chapter 3

Algorithm Design

Heyheyhey

Chapter 4

User Interface Design

Heyheyhey

Chapter 5

Requirements Traceability

Heyheyhey

Chapter 6

References

Heyheyhey