POLITECNICO DI MILANO

SOFTWARE ENGINEERING 2 (2015 - 2016)

MyTaxiService V1

DESIGN DOCUMENT

Authors:

Sun Chao

Daniel Naveda

Bakti Ariani Melinda Pertiwi

December 3rd 2015

Contents

1. INTRODUCTION **5**

1.1. Purpose 5

1.2. Scope 5

1.3. Definitions, Acronyms, Abbreviations 5

1.3.1. Definitions 5

1.3.2. Acronyms 5

1.3.3. Abbreviations 6

1.4. Reference Documents 6

1.5. Document Structure 6

2. ARCHITECTURAL DESIGN **7**

2.1. Overview 7

2.1.1. Client Tier 7

2.1.2. Web Tier 8

2.1.3. Business Tier 8

2.1.4. EIS Tier 8

2.2. Identifying Subsystem 8

2.3. High Level Components and Their Interaction 11

2.4. Component View 11

2.5. Deployment View 12

2.6. Runtime View 14

2.7. Component Interface 15

2.8. Architectural styles and patterns 16

2.8.1. Server 17

2.8.2. Database 17

2.8.3. Registered User 17

2.8.4. Registered Taxi 17

3. ALGORITHM DESIGN **18**

3.1. Fair Management of Taxi Queues 18

3.2. Taxi Fee Calculating 21

4. USER INTERFACE DESIGN **22**

4.2. Registered Driver 22

4.3. Taxi Client Android Application 22

5. REQUIREMENT TRACEABILITY **23**

6. REFERENCES..... 24

Table of figures

Figure 1 – Java Multitiered Applications (taken from Oracle's website) 7
Figure 2 – Structure Diagram 10
Figure 3 – Main three components 11
Figure 4 – Component view 12
Figure 5 – Deployment View 13
Figure 6 – Deployment View 13
Figure 7 – Runtime view (a) 14
Figure 8 – Runtime view (b) 15
Figure 9 – Component Interface 16
Figure 10 - Basic Diagram of the system 17
Figure 11 – Taxi Zones 18
Figure 12 – Allocating new taxi flowchart 19
Figure 13 – Taxi Request 20
Figure 14 – Taxi Fee Calculation 21
Figure 15 – Registered Driver's mobile UI 22
Figure 16 – Passenger's mobile UI 23

1. INTRODUCTION

1.1. Purpose

This software design document is intended to be used as a guideline for the implementation of myTaxiService app. The primary purpose is to show how myTaxiService will be structured to satisfy its requirements previously defined in the RASD. This DD can be used as the primary reference to understand the architecture and system design of myTaxiService application.

1.2. Scope

This document contains a complete and high level description of the design of myTaxiService. The basic architecture is a two-tier Server-Client model paradigm. The server will be our management system in charge of assigning the taxis, and the clients will be either a regular user or a taxi driver.

1.3. Definitions, Acronyms, Abbreviations

1.3.1. Definitions

- Mobile App : Mobile Application
- Taxi : Vehicle used to transport passengers as a service provided
- Registered Passenger : Passenger that have registered
- Registered Driver : Taxi drivers that have registered and have been authenticated by the system manager
- Authentication : To make sure the driver is really existed and belong to the one taxi company
- Android application : An application created on the Android operating system
- Google map : An electronic map developed by Google company

1.3.2. Acronyms

- RASD : Requirements analysis and specification document
- DD : Design Document
- GPS : Global Positioning System
- UI : User Interface
- DBMS : Database management system

- API : Application Programming Interface
- JVM : Java Virtual Machine
- JEE: Java Enterprise Edition
- SDK : Software Development Kit
- ADT : Android Development Tools

1.3.3. Abbreviations

- App : Software Application

1.4. Reference Documents

- Specification document: “Assignments 1 and 2.pdf” assigned by the Professor of the course “Software Engineering 2”.
- Template for the design document: “DD TOC.pdf” given by the Professor of the course “Software Engineering 2”.
- IEEE Std 1016-2009 (Revision of IEEE Std 1016-1998) IEEE Standard for Information Technology—Systems Design— Software Design Descriptions

1.5. Document Structure

This document is organized in the following sections:

1. Introduction

2. Architectural Design

3. Algorithm Design

4. User Interface Design

5. Requirements Traceability

6. References

2. ARCHITECTURAL DESIGN

2.1. Overview

For this project, we have decided to use Java EE. Its capabilities to build a large-scale, multi-tiered, scalable, reliable, and secure system are suitable to deploy myTaxiService. Moreover, JEE architecture is perfectly compatible with our application.

Due to the well-defined JEE architecture, our myTaxiService system architecture is shaped by the traditional Java EE. Therefore, before explaining our application's architecture we would like to focus on JEE Architecture. In this sense, we are also providing a fairly good description of myTaxiService system.

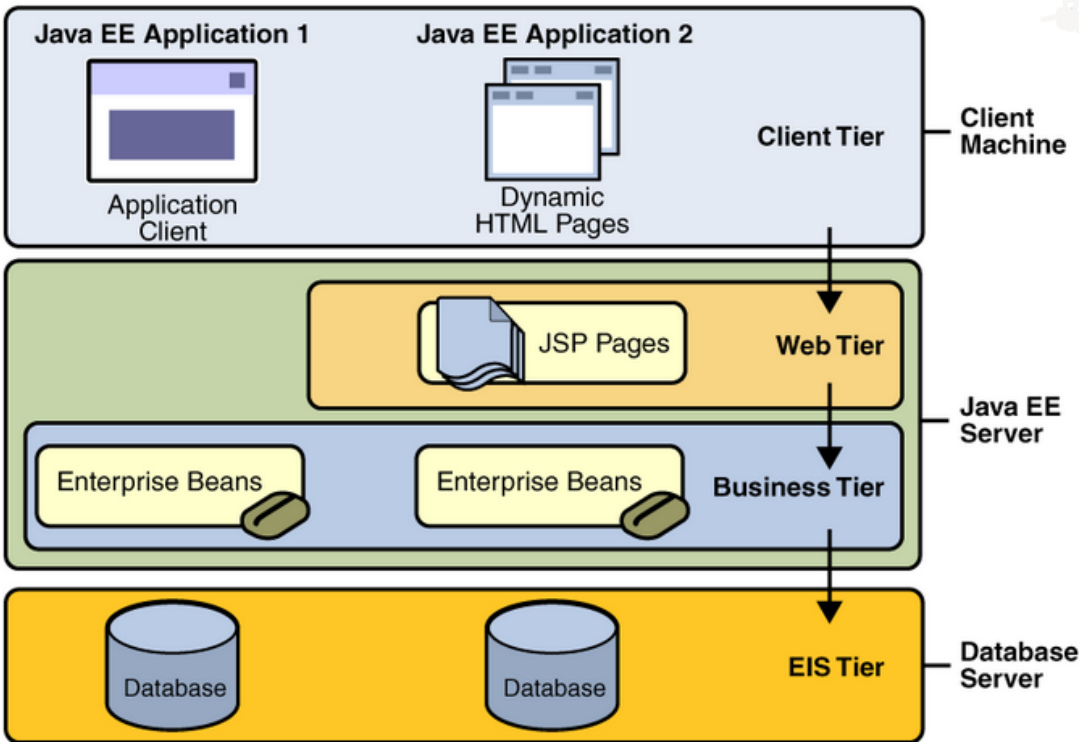


Figure 1 – Java Multitiered Applications (taken from Oracle's website)

As shown in the figure above, JEE has a multi-tiered architecture divided in:

2.1.1. Client Tier

The client tier consists of application clients that access a Java EE server and that are usually located on a different machine from the server. The clients send requests to the server. The server processes the requests and returns a response back to the client. Clients can be a web browser, a standalone application, or other servers, and they usually run on a different machine from the Java EE server.

2.1.2. Web Tier

The web tier consists of components that handle the interaction between clients and the business tier. Its primary tasks are the following:

- Dynamically generate content in various formats for the client.
- Collect input from users of the client interface and return appropriate results from the components in the business tier.
- Control the flow of screens or pages on the client.
- Maintain the state of data for a user's session.
- Perform some basic logic and hold some data temporarily in JavaBeans components.

2.1.3. Business Tier

The business tier consists of components that provide the business logic for an application. Business logic is code that provides functionality to a particular business domain, like the financial industry, or an e-commerce site. In a properly designed enterprise application, the core functionality exists in the business tier components.

2.1.4. EIS Tier

The enterprise information systems (EIS) tier consists of database servers, enterprise resource planning systems, and other legacy data sources, like mainframes. These resources typically are located on a separate machine than the Java EE server, and are accessed by components on the business tier.

2.2. Identifying Subsystem

In order to make it easy to understand the issues that we will meet in implementing detail functionalities, we prefer to decompose our system into other sub-systems.

- Sign up sub-system;
- Log in sub-system;
- Passenger sub-system;
    - Profile managing sub-system;
    - Payment managing sub-system;
    - History1 managing sub-system;
    - Order Taxi sub-system;
    - Map managing sub-system;
- Driver sub-system;
    - State managing sub-system;

- Request managing sub-system;
- Income managing sub-system;
- Profile managing sub-system;
- History2 managing sub-system;

- Administrator sub-system.

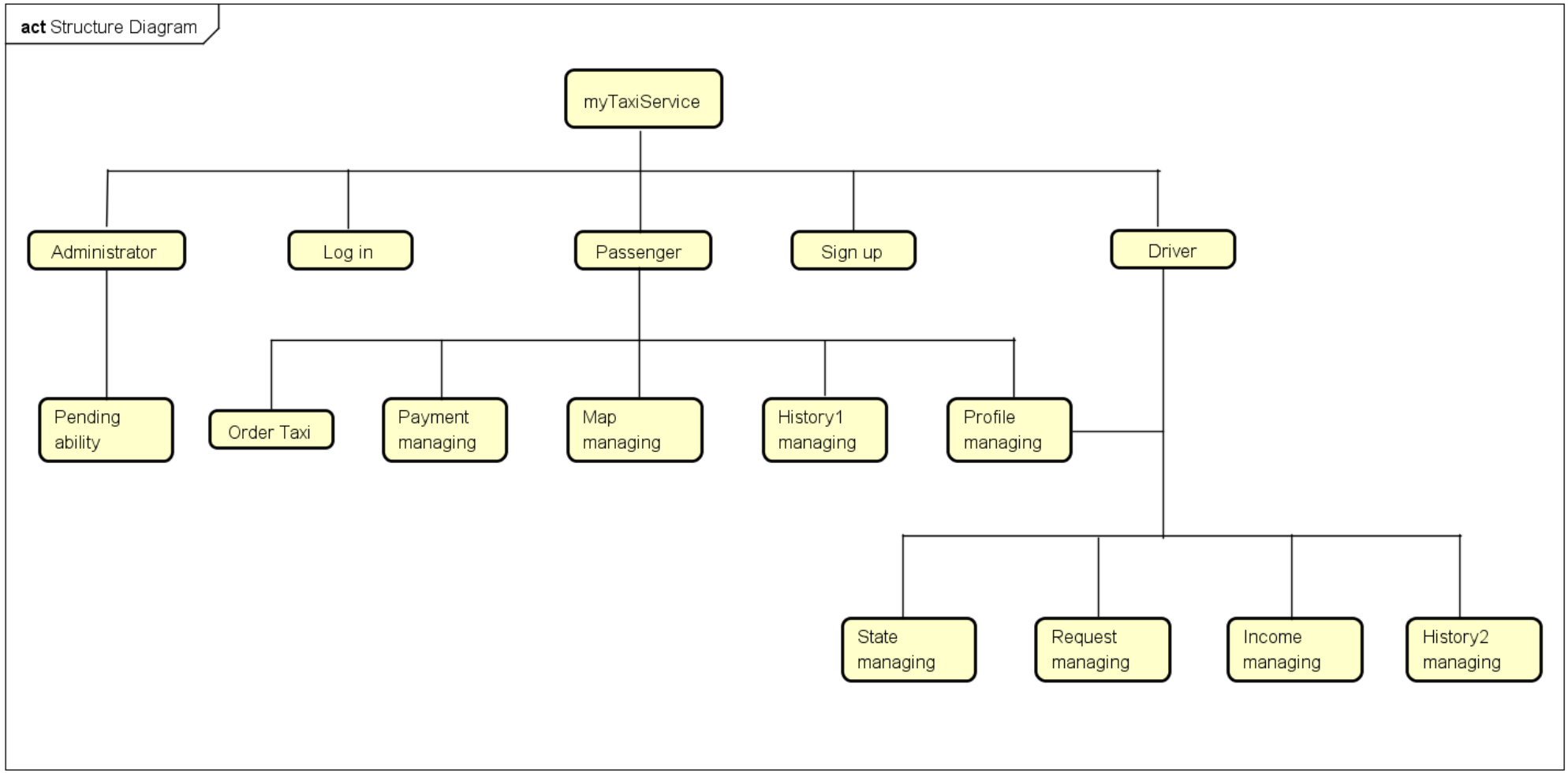


Figure 2 – Structure Diagram

2.3. High Level Components and Their Interaction

From a high level's point of view, we basically have three components: Server (myTaxiServiceServer) and 2 clients (Users and Taxis). This is depicted in the following figure:

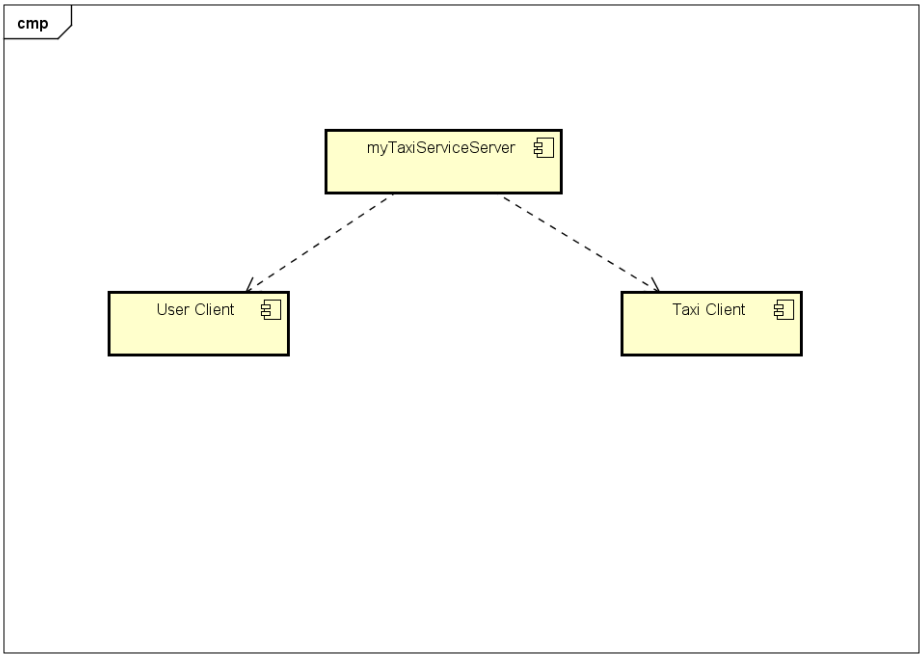


Figure 3 – Main three components

As a typical Server-Client architecture, we can only see communication between the Server and the Clients. As stated before, we have two kinds of clients: one for Passengers (User Client) and the other for Drivers (Taxi Client).

2.4. Component View

Product contains of three main modules, Customer client application, Taxi driver's client application and a main server which coordinates the communication between these two components and merges them into a complete and unique system. Taxi driver's application tracks taxi movement, updates taxi's location and enables taxi driver to get information about potential pickup requests. Customer's application allows customers to order a taxi and track the taxi selected for pickup. Main server collects and handles the data received from Taxi driver's application and Customers application. It manages virtual queues and forwards order requests received from customers to the taxi drivers. Main server also provides a web

application for customers. The web application has the same features as the Android client application for customers “myTaxiService”. It is used for ordering and tracking the taxi.

This kind of deployment is natural since the product contains of two parts, one that is going to be used by the customers and the other that is going to be used by the taxi drivers. Third part, the Main Server provides a way to integrate these two modules.

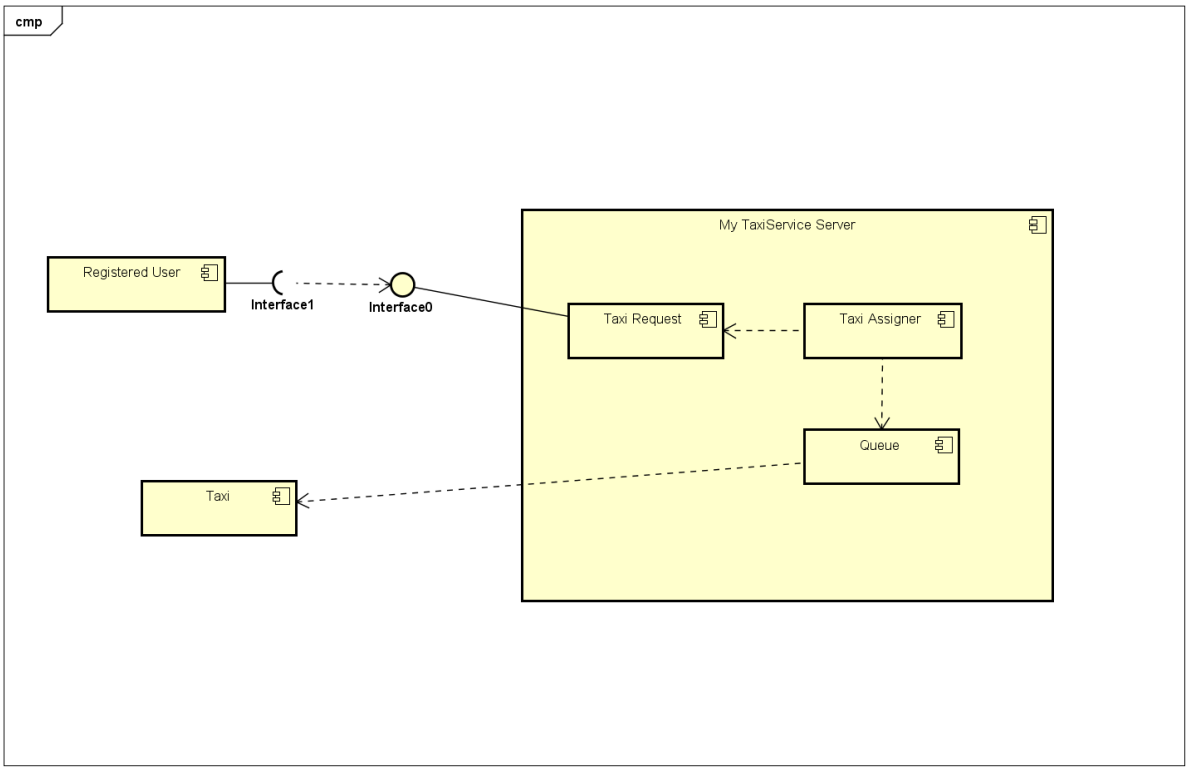


Figure 4 – Component view

2.5. Deployment View

The client can be available after deployed on Android device which the SDK edition should be later than 2.0. Moreover, GPS modules should be embedded on these devices in advance. The relations between different components are shown on the following deployment diagram.

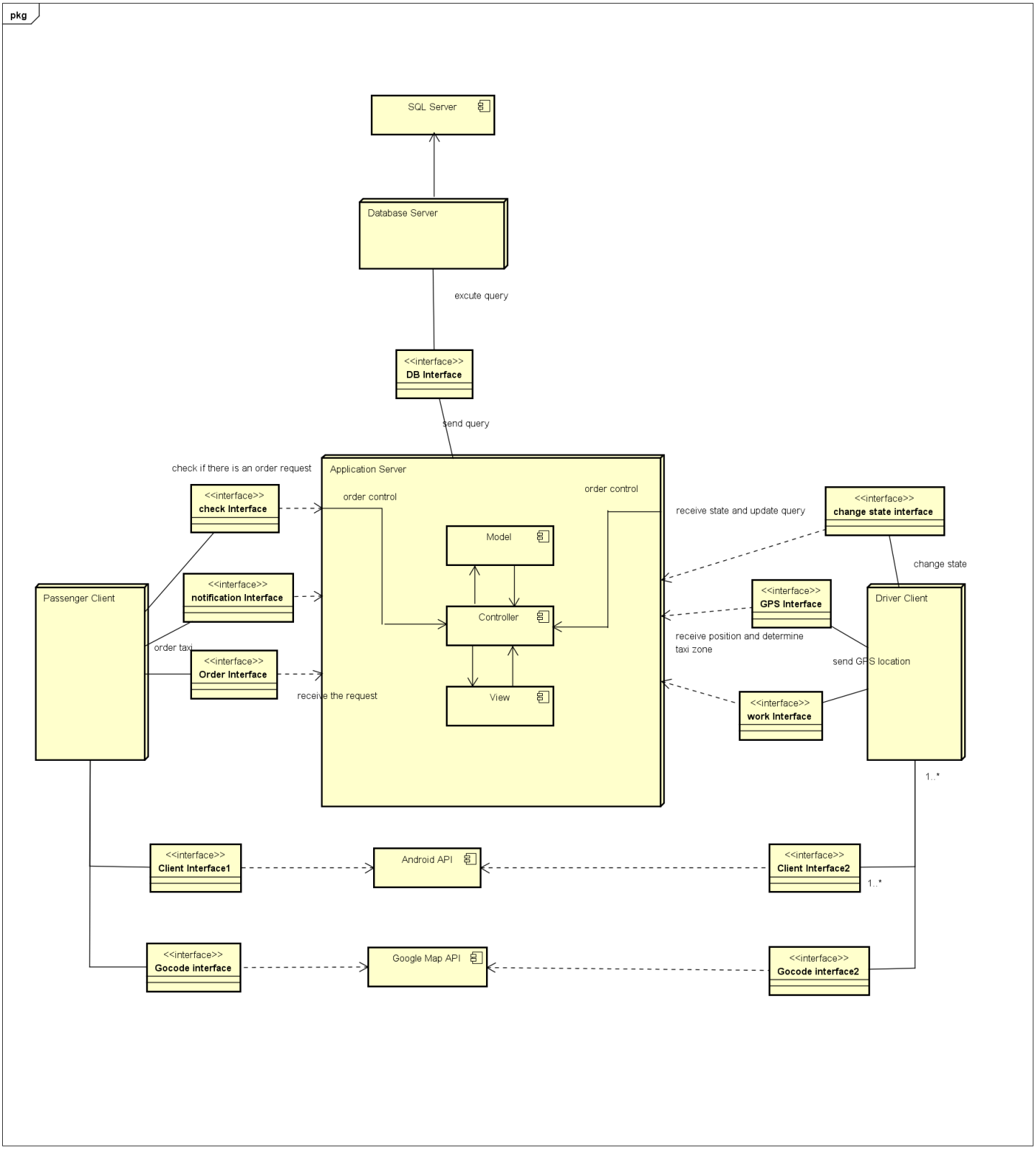


Figure 5 – Deployment View

2.6. Runtime View

This section of the document can be as extensive as we want to; this is because covering all the runtime cases is practically impossible. For this reason, we are going to show a couple of sequence diagram (quite similar to the ones shown in the RASD).

The first sequence diagram shows the behavior along time between a passenger (User Client) and the myTaxiService server (Server) during a sign up operation:

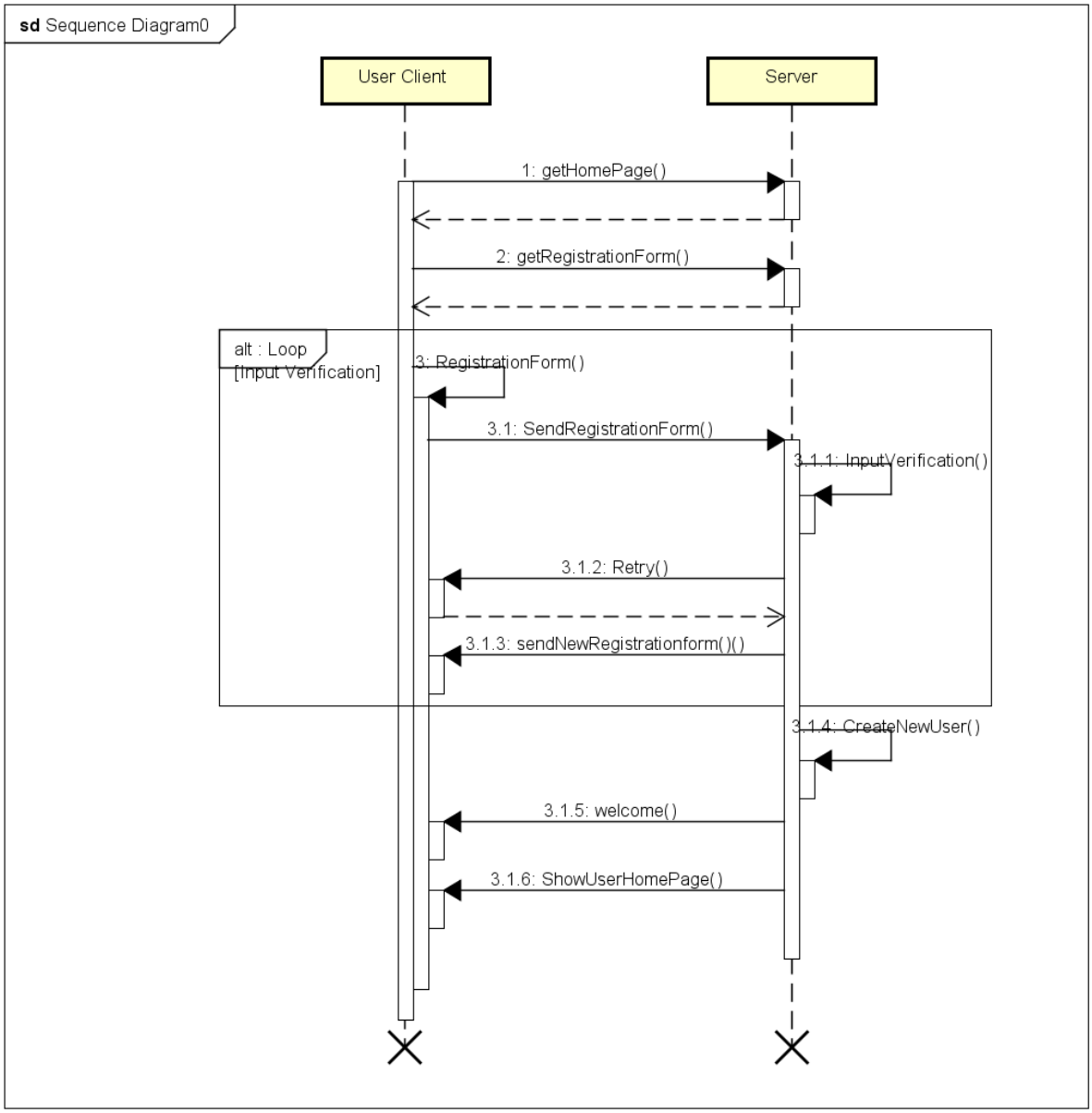


Figure 7 – Runtime view (a)

For the second diagram, the communication along time between a driver (Taxi) and the myTaxiService server (Server) is shown. Here we can see a basic “Pick up Setting” operation:

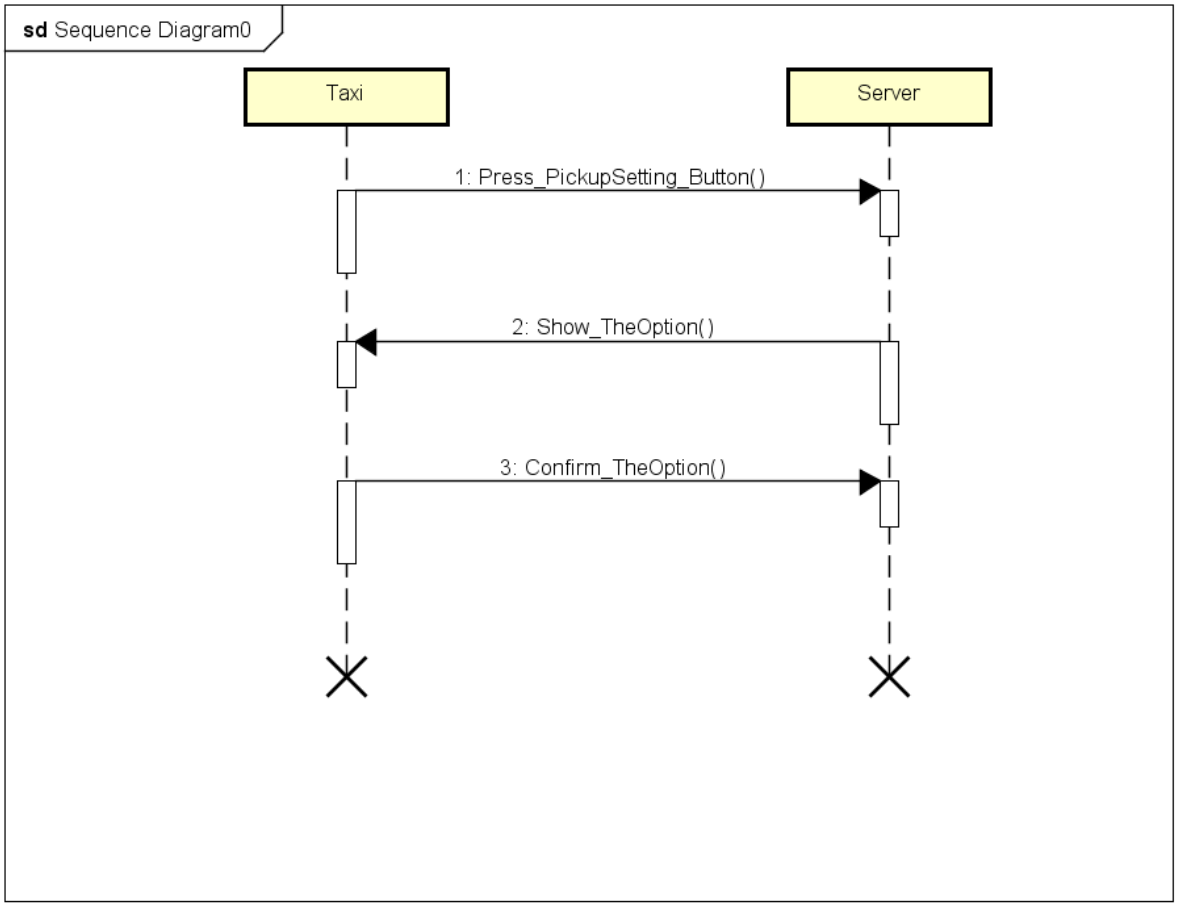


Figure 8 – Runtime view (b)

2.7. Component Interface

For the interface, the main components already mentioned are: Registered User (or Passengers), Taxi and Server. One important component which is not explicitly shown is the database; however, it is implicitly taken into account inside the Server component

The interface through which the Clients (Passengers and Taxis) and the server establish the communication is HTTP. This interface is not a surprise since this is almost the default interface when working with website applications.

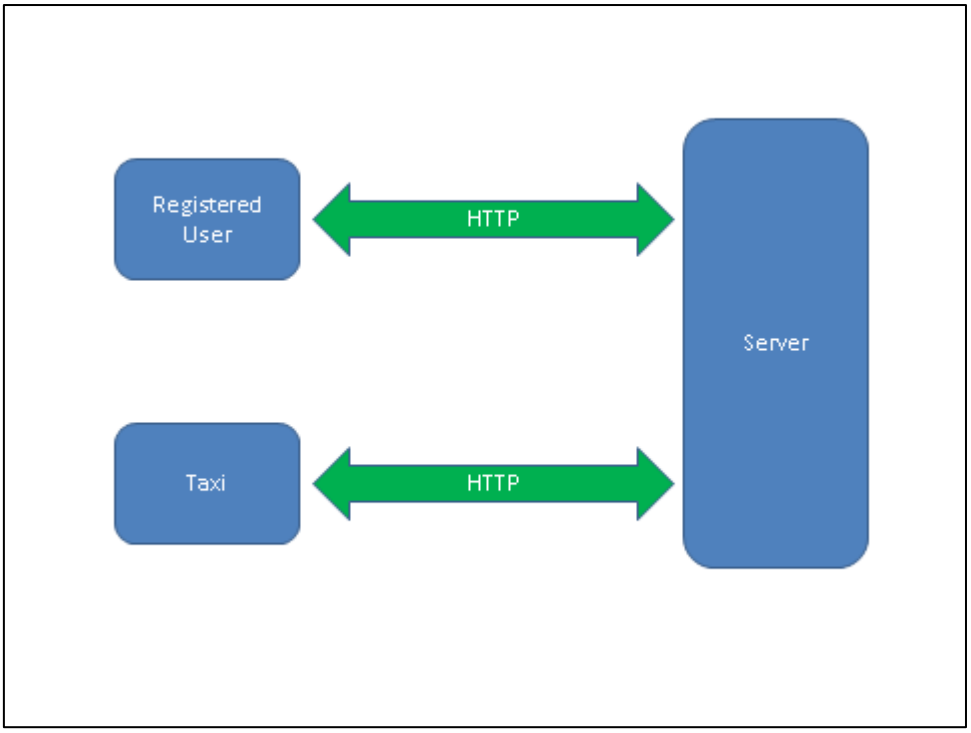


Figure 9 – Component Interface

2.8. Architectural styles and patterns

From a high level perspective, this architecture style of our application is Client and Server. In other words, we will have a Server, handling requests from two types of clients: Passengers and Taxi Drivers. Meanwhile, we decide adopt a top-down approach to break down this system to gain insight into its compositional subsystems. And then refine these subsystem in a greater detail way.

In the following figure, the basic structure of the system is shown:

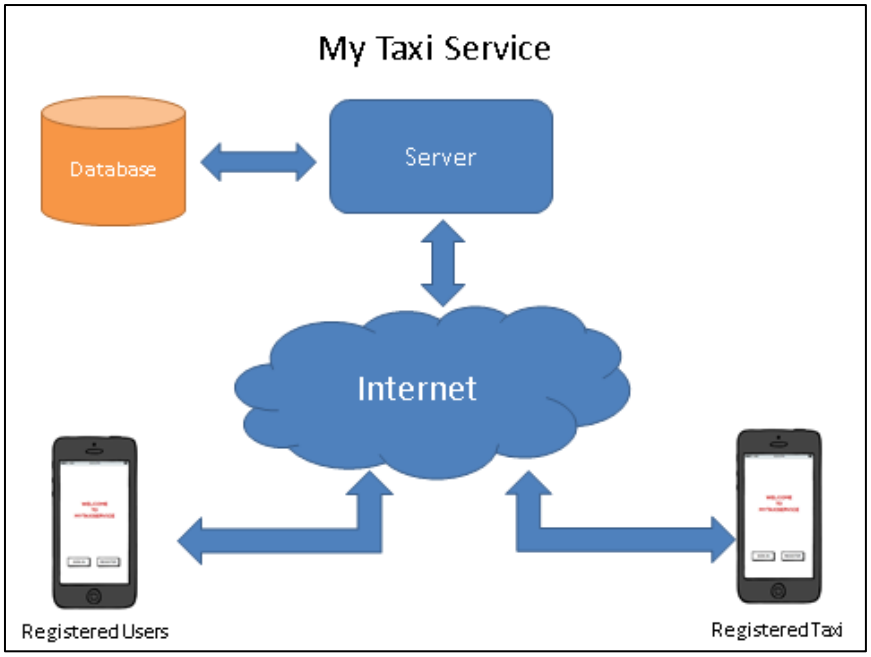


Figure 10 - Basic Diagram of the system

2.8.1. Server

The server is the core of the whole myTaxiService system. Its main purpose is receiving requests through Internet from clients (passengers or taxi drivers), processing them, and sending the results back to the clients.

One of the main components needed by the Server is the Database, which is explained in the following section.

2.8.2. Database

In our design the database is essential for the proper operation of the system. It is intended to storage all the clients' data in there, which is important for the typical operations of: signing up (store data), logging in (retrieve data), and requesting taxi and so on.

2.8.3. Registered User

As one of the clients depicted in the figure shown above, registered users (passengers), in essence, are the entity sending requests to the server in order to get a taxi as a mean of transportation.

2.8.4. Registered Taxi

This is the Taxi Driver application. The main reason to contact the server is to provide it with useful information such as: location, status, etc. Making sure that the server counts

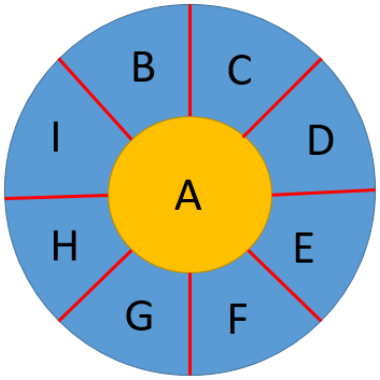
with all this data is crucial so the server can compute all the info gathered and take the best decision to assign the most suitable taxi to a client.

3. ALGORITHM DESIGN

3.1. Fair Management of Taxi Queues

The registered passengers order/reserve a taxi through our application, and the system will allocate a suitable taxi to them through the fair management algorithm. In this process, passengers are blind to this, they only need to know which one is ready for them.

First of all, the city should be divided into different zones.



If the blue circle represents a city, we can divide it into 9 different parts. Then, we can get 9 taxi zones in this way.

Figure 11 – Taxi Zones

Noted:

- One specific taxi can only belong to one taxi zone at a time.
- Region allocated is static and the area will not change when taxi is moving.
- Taxi belongs to one exactly zone, only after the state of the taxi switched to “ACTIVE”.
- The area of each taxi zone approximately 2km^2.

In each single taxi zone, all the taxis are waiting in a queue logically. Only on the first place can receive the request from passengers. Here we use FIFO queue to define the data structure of taxi queue in each zone.

```
typedef struct queue
{
    int queuesize;
    int head, tail;
    int *q;

}queue;
```

Allocate new taxi to the queue:

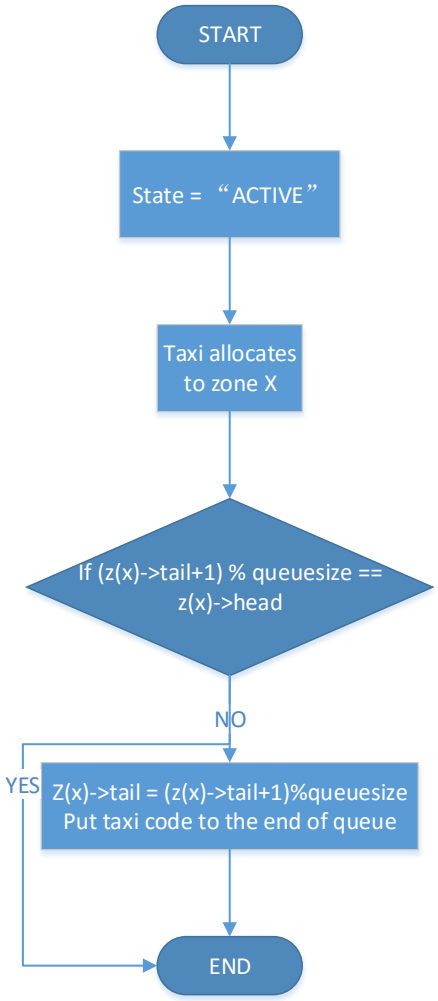


Figure 12 – Allocating new taxi flowchart

Reflection to taxi request:

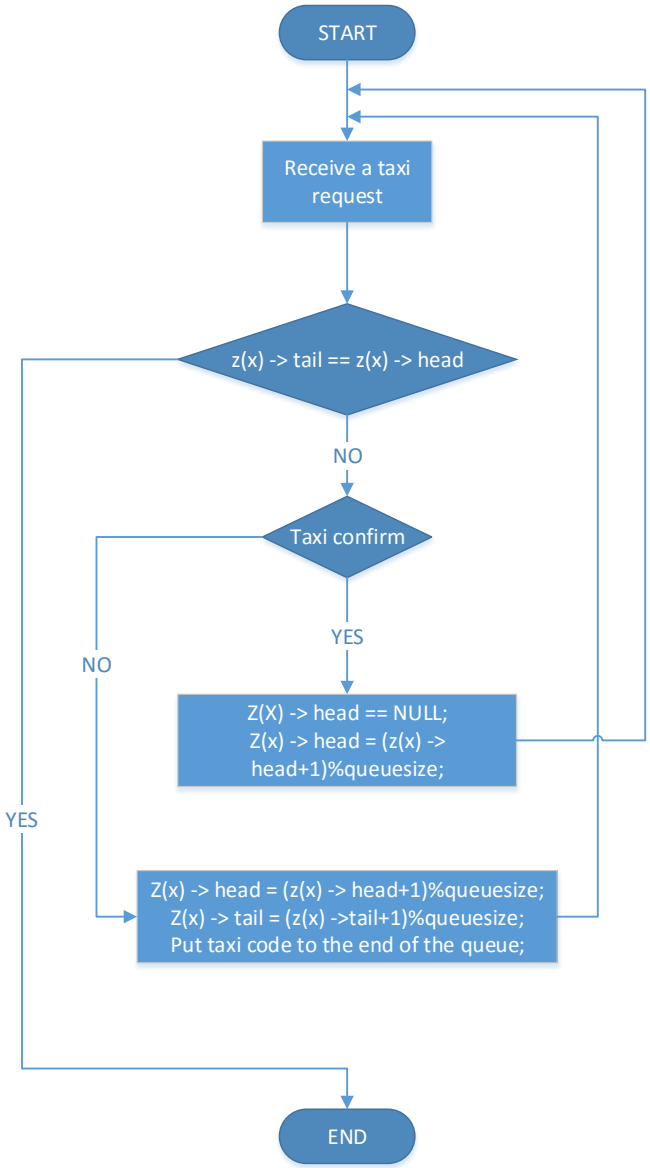


Figure 13 – Taxi Request

3.2. Taxi Fee Calculating

As we can see below, there are 3 separated ways which can lead to different cost.

- Order taxi service immediately;
- Reserve taxi service 2 hours ahead;
- Order taxi service and share it with other passenger;

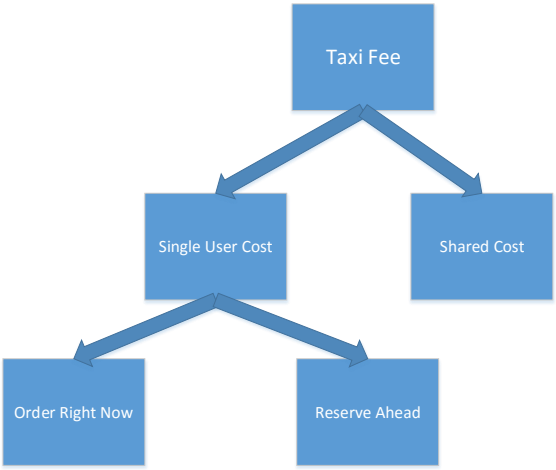


Figure 14 – Taxi Fee Calculation

In order to make this issue calculability, we define:
Taxi fee calculating method between "Order Right Now" option and "Reserve Ahead" option is the same.

Without sharing taxi:
D: total distance
d: constant distance
F: total taxi fee need to pay
x: passenger need to pay €.x
n1: €.n1for every kilometer
n2: €.n2 for every minite

when $0 < D \leq d$,

$$F = x$$

when $D > d$,

$$F = x + (D - d) \times n1 + t \times n2 \quad \text{when} \quad D>d$$

Sharing taxi option:
y: y passengers share one taxi
Fs: payment for every shard passengers
Fu: payment for ordered passenger

when $0 < D \leq d$,

$$F = \frac{x}{y} \times t$$

When $D > d$,

$$Fs = \frac{Ds \times n1 + Ts \times n2}{y}$$

$$Fu = F - Fs\times(y\text{-}1) = x + [(D - d) \times n1 + t \times n2] \times \frac{1}{y}$$

4. USER INTERFACE DESIGN

The user interfaces for myTaxiService were described in the RASD document. It can be easily accessed on:

https://github.com/daniel2121/SE2_Project_15-16/blob/master/Deliveries/myTaxiService_RASD.pdf

However, we will show next some of the user interfaces which might be handy.

4.2. Registered Driver

This mockup is the main screen for registered driver. The profile photo is the entrance to "Account Setting screen". It is worth noting that driver can press "START" button only after setting the state to available.

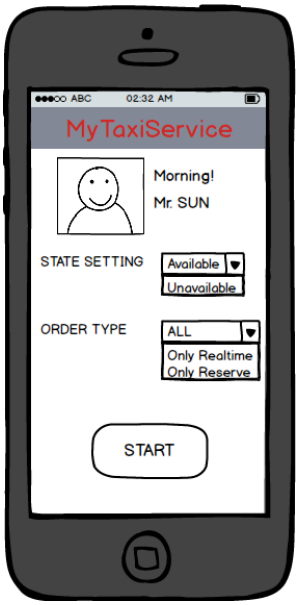


Figure 15 – Registered Driver's mobile UI

4.3. Taxi Client Android Application

This main screen mockup shows the first page after passenger sign in. They can send the request to book a taxi right now or later (2 hours ahead) through this page.

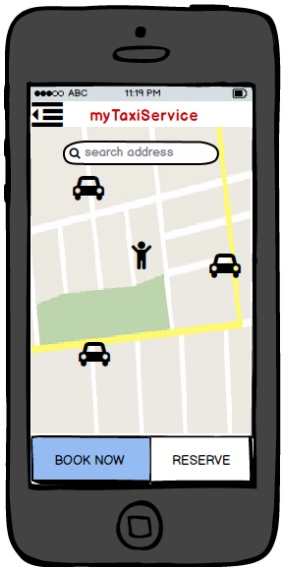


Figure 16 – Passenger's mobile UI

5. REQUIREMENT TRACEABILITY

Requirement	Description	Design Reference
Simplify the access of passengers to the service	Passengers can request a taxi by a web application or mobile application	The request will be information about the code of the incoming taxi and the waiting time by the system.
Guarantee a fair management of taxi queues	There are taxi zones in the city. Each zone is associated to a queue of taxis.	The system will compute the taxis in every zone. When the system receives information that the taxi is available, so it will be stored in the taxi queue in every zones.
Specifying the origin and the destination of the ride	Passengers have to confirm from where they will take the taxi and confirm they destination clearly.	Passenger can make the reservation two hours before the ride. Because it can allocates the time of taxi to take the passengers.
Taxi sharing option	The passengers want to share their taxi with another passenger if it is possible about the price and the ride.	Passengers who want to share their taxi have to specify the destination of the ride, so the system will arranges the route and the fee for all passengers.

6. REFERENCES

- https://docs.oracle.com/javaee/5/tutorial/doc/bnaay.html
- Hans van Vliet, 2007, Software Engineering: Principles and Practice
- https://en.wikipedia.org/wiki/Two-tier_system
- https://simple.wikipedia.org/wiki/Client-server