Politecnico di Milano, A.A. 2015/2016

Software Engineering 2: My Taxi Service
Design **D**ocument

Belotti Nicola 793419
Chioso Emanuele 791621
Colombo Andrea 853381

December 2, 2015

Contents

1 Introduction

1.1 Purpose

The goal of this Design Document is to explain the design choice made during the architectural analysis and the functionalities that will be developed.

1.2 Scope

The aim of the Design Document is to present the main architecture of My Taxi Service application. The structure will be based on the requirements analysis we made in the RASD document.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Visitor Someone who visits my taxi service application's website but it's not logged in

User Someone who is registered and logged in my taxi service application

Passenger A type of logged user, who uses the application to call a taxi

Taxi driver Another type of logged user, who uses the application to answer calls from the system

1.3.2 Acronyms

DBMS Database Management System.

JEE Java Enterprise Edition.

API Application Programming Interface.

ER Entity-Relational Model.

EJB Enterprise Java Bean.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

JDBC Java Database Connectivity.

UML Unified Modeling Language.

UX User eXperience.

MVC Model View Controller.

JPA Java Persistence API.

XHTML eXtensible Hypertext Markup Language.

1.3.3 Abbreviations

1.4 Reference Documents

- Structure of the design document.pdf
- ISO/IEC/IEEE 42010 Systems and software engineering Architecture description
- IEEE Std 1016^{tm}-2009 IEEE Standard for Information TechnologySystems Design Software Design Descriptions

1.5 Document Structure

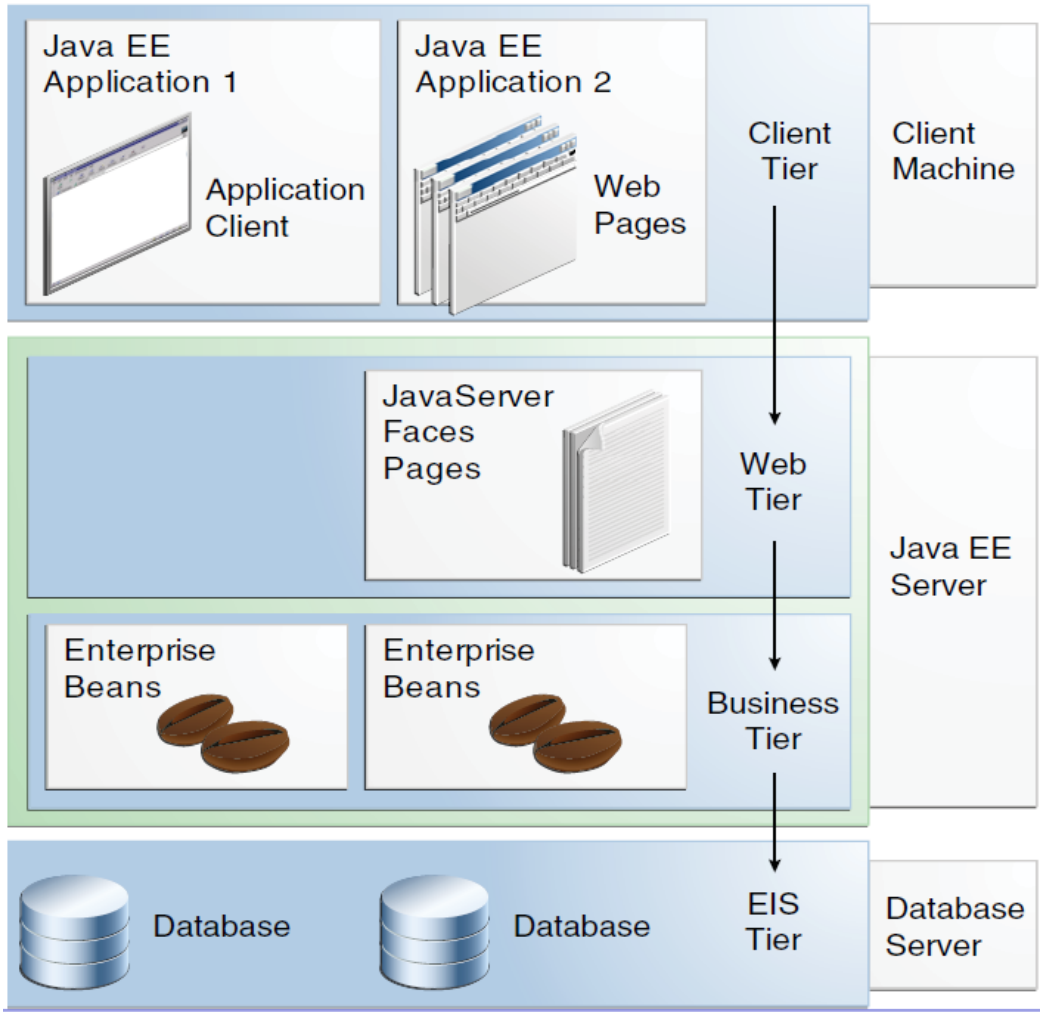
The document is structured in six parts

1. Introduction
2. Architectural Design
3. Algorithm Design
4. User Interface Design
5. Requirements Traceability
6. References

2 Architectural Design

2.1 Overview

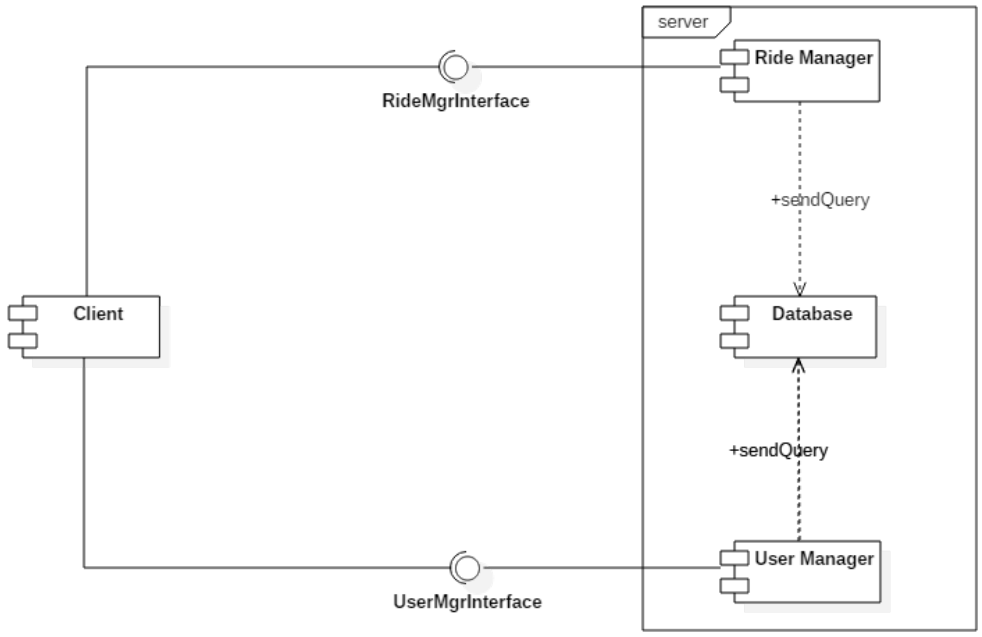
Our system will be designed like a classic four tier JEE application like explained in the figure below.



The client tier contains web pages (while browsing our site through a computer) or a application client (while using the application installed on a smartphone) The server tier is dived into two tiers: web tier that is in charge of generating web pages and sending them to the client; and the business tier where all the application's logic is placed. The last tier is the database tier that contains the database server.

2.2 High level components and their interaction

This diagram represents the main components of our system and shows how they communicate to each other. On the client side we have only the client component. On the server side we have the ride manager component that will take care of the taxi management side of our application, it will provide an interface to the client. The user manager component will provide functions to manage user account's, this component too will provide an interface to the client, finally the database will hold accounts', taxis' and rides' informations, it will be accessible by the other two server components.

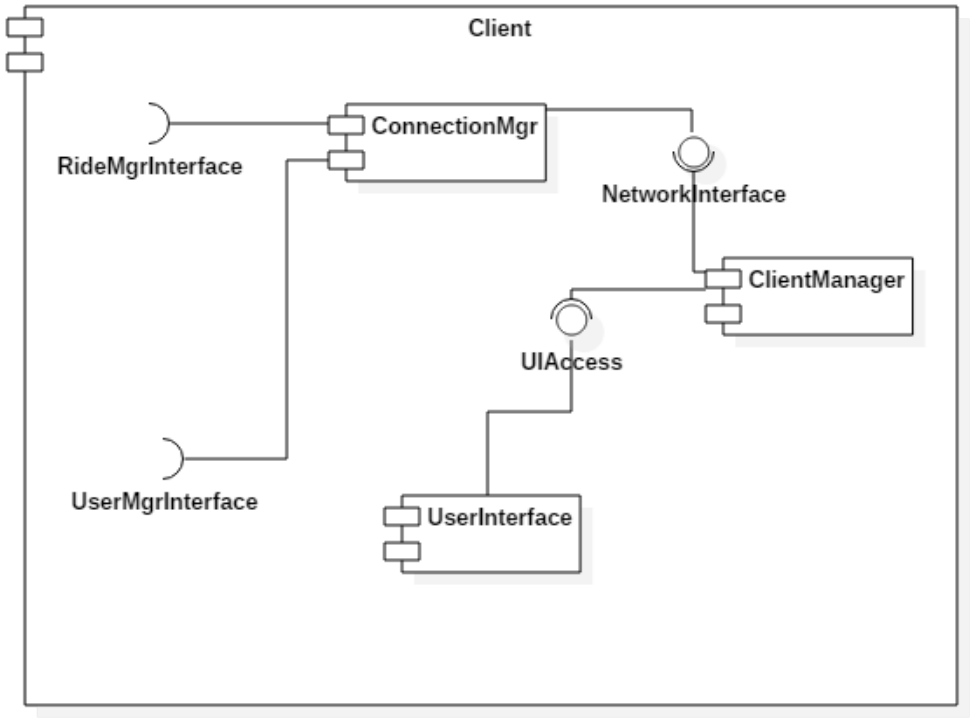


2.3 Component view

Here we will provide a more detailed description of the components presented in the previous section.

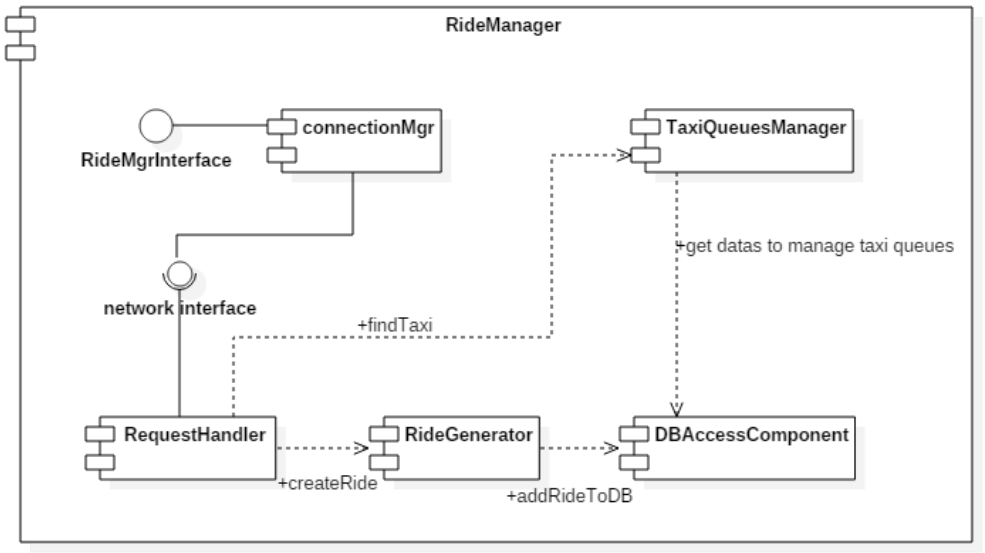
2.3.1 Client component

The client is composed by a main component which will have all the basics functions: create account, make request, make reservation; a user interface component that will serve to communicate with the user and finally a connection component that will take care of communicating with the server.



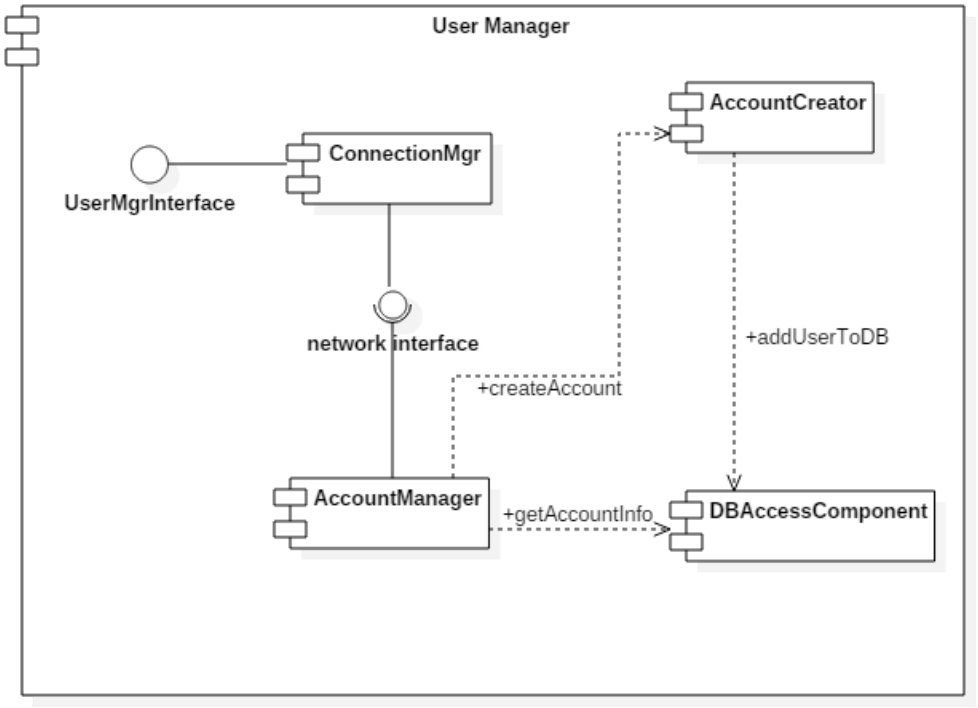
2.3.2 Ride manager component

The ride manager will have a request handler that will take requests or reservation from passengers and forward it to taxi drivers, once a driver has accepted it the ride will be created through the ride generator component. The taxi will be found using the taxi queues manager component. There will be also a connection manager component and a database access component used to access the database. The request handler will also take care of join and cancel reservation requests from a passenger.



2.3.3 User manager component

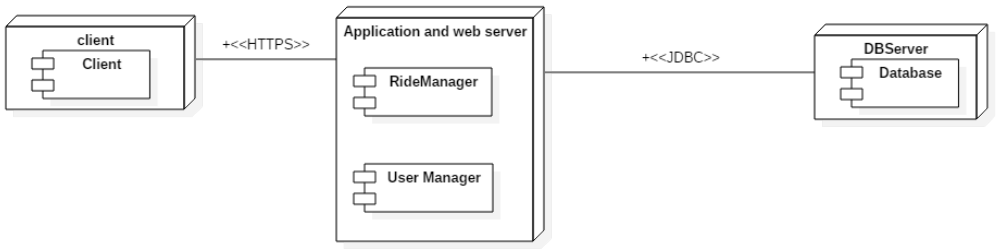
The user manager will have a account manager component which will provide login and registration functionalities, an account creator component to create new accounts and a database access component.



2.3.4 Database

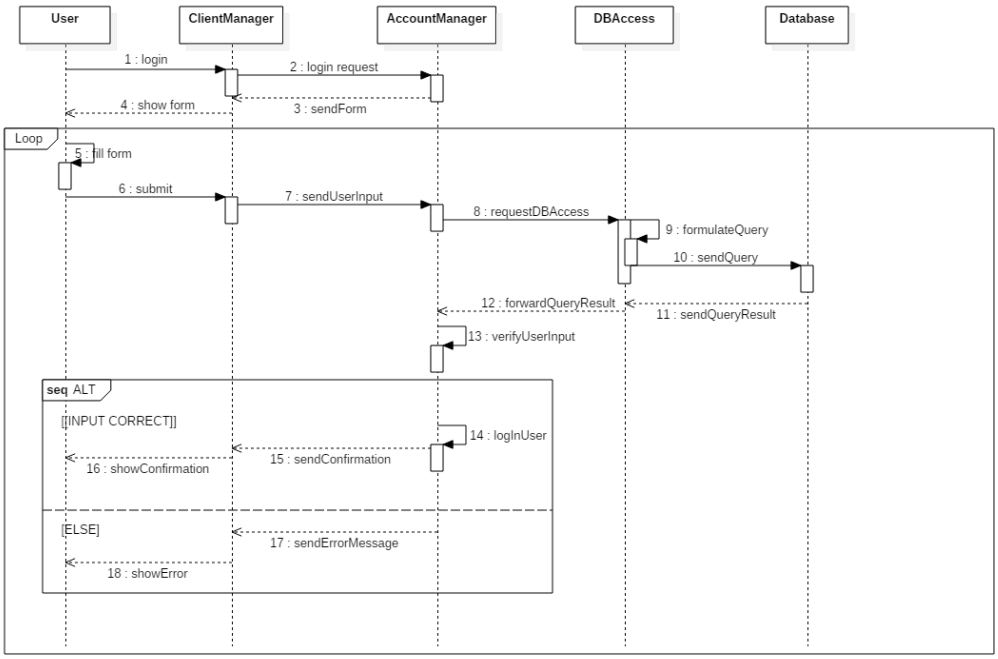
2.4 Deployment view

This diagram shows how the whole system will be deployed. The client communicates via HTTPS to the web server and the application server (both web and application servers are installed in the same machine), the web server is responsible to generate web pages, the application server contains all the user and taxi management application's logic and in the end the database.

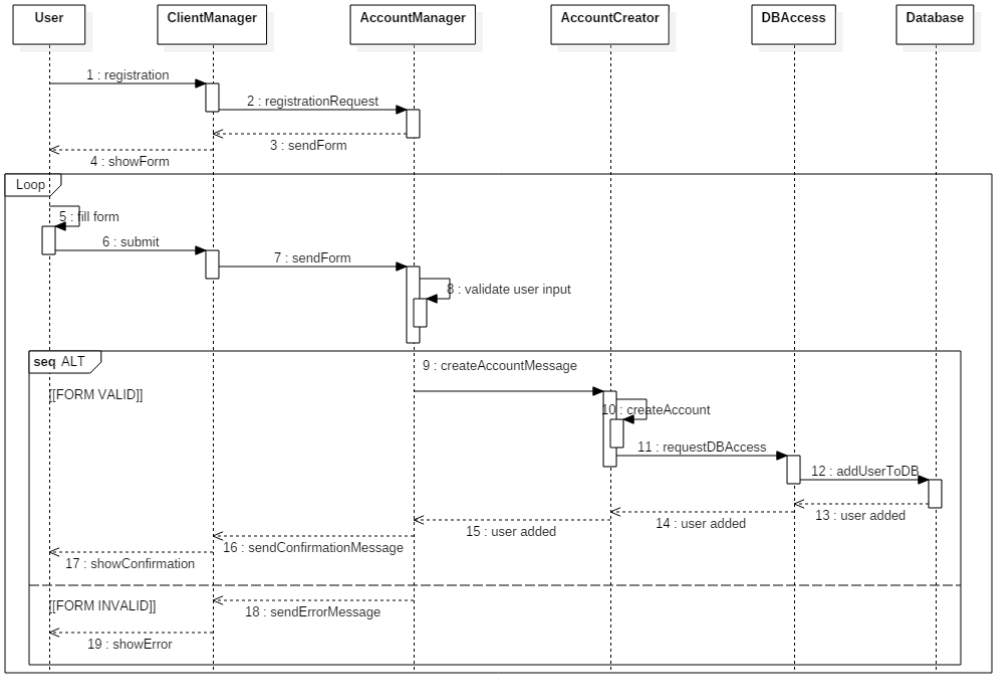


2.5 Runtime view

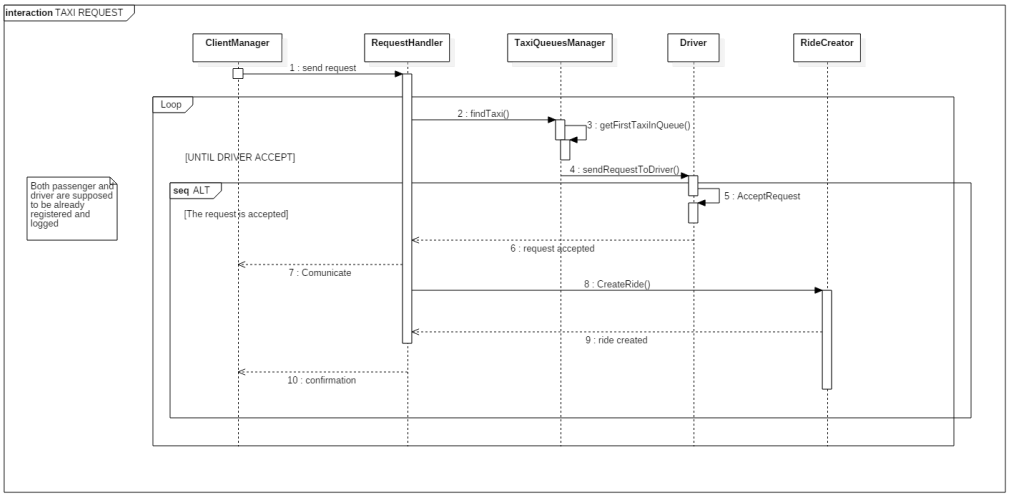
2.5.1 Login



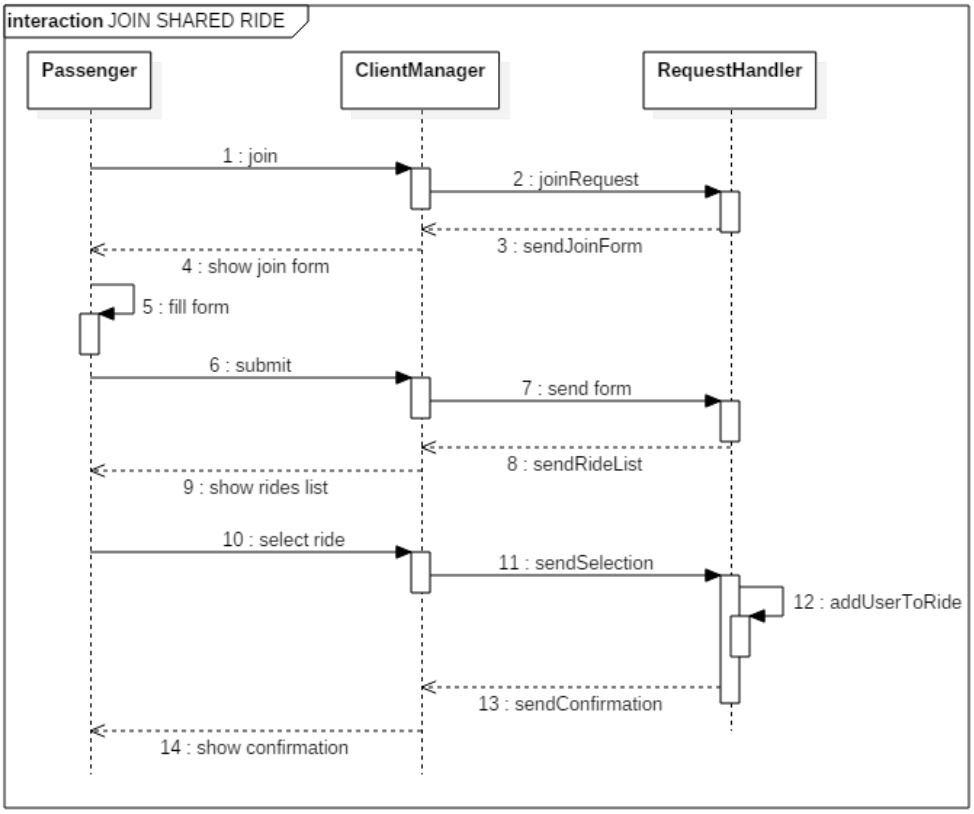
2.5.2 Registration



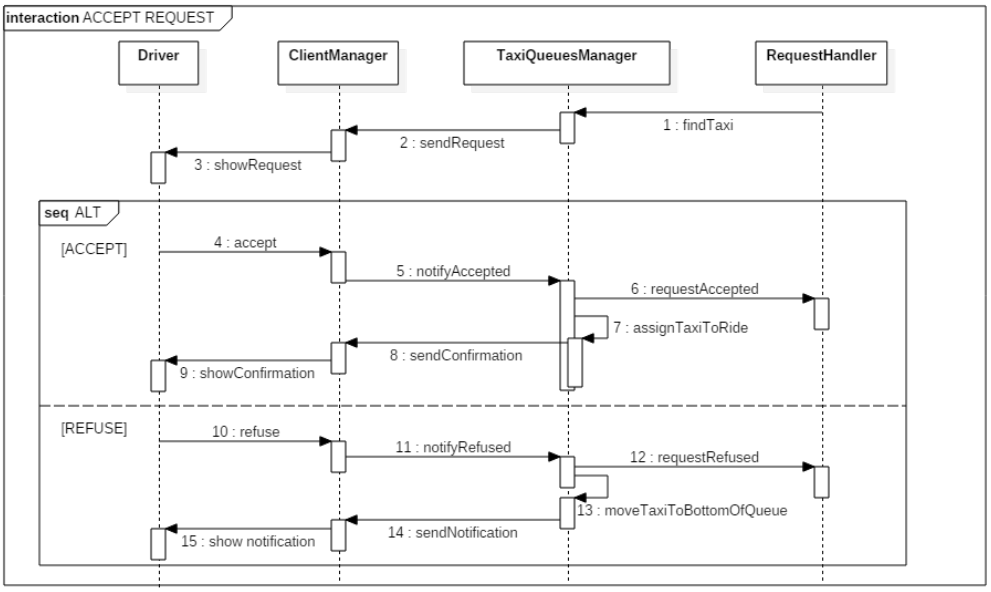
2.5.3 Taxi request



2.5.4 Join ride



2.5.5 Accept request



2.6 Component interfaces

In this section we provide a detailed description of the interfaces we use.

The network interface is used in all the three components, and it's a interface used to provide methods to send and receive user inputs, server's responses and data.

2.6.1 Client interfaces

1. **UIAccess**: This interface provides methods to get user input through the user interface

2.6.2 User Manager interfaces

1. **Ride Manager interface**: This interface provides to the client methods used to request services from the server, for example for calling a taxi.

2.6.3 Ride Manager interfaces

1. **User Manager interface**: This interface provides to the client methods used to request services from the server such as login and registration requests.

2.7 Selected architectural styles and patterns

2.8 Other design decisions

3 Algorithm Design

Here we will present the algorithm that will be used to create taxi queues:

First we divide the city zone in N areas, approximately equals one to each other. Then we assign a certain number of taxis to each zone, thus meaning that the taxi will work in the area he's assigned to. The number of taxis to assign to each zone will be computed dynamically based on the number of request that are coming from a certain zone, we'll call it request number, it may change from one day to another, or even from one period of time to another. The algorithm will always try to maintain the number of taxis in a zone equals to its request number.

If a taxi, while taking care of a request, leave his initial zone he will be automatically assigned to the zone he's into at the end of the ride, while assigning another taxi to the zone left by the first taxi, in order to have balance between zones. The choice of the taxi to be assigned to the zone will be made by minimizing the distance it will have to travel to enter the designated zone. If a zone's number of taxis drops below its request number the system will assign another taxi to that zone recursively till a point of balance is found. We will take preemptive measures to avoid infinite loops.

The taxis will be organized in queues, each queue assigned to a zone. When a request arrives from a zone it will be forwarded to the first taxi in queues, if he accept the taxi will be removed from the queue and will be placed at the end of the queue in the zone he will end up at the end of the ride. If he refuse it instead he will be placed at the end of the queue. If a taxi is moved to one zone to another he will take the position in the new queue equals to its previous position.

4 User Interface Design

5 Requirements Traceability

6 References