Taxi booking system

Analysis and Design Document

Student: Rus Darius

**Group: 30238**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

The taxi booking system is specialized in hiring cabs to customers. It is a system through which customers are allowed to book their taxis from everywhere. The users should create an account for ordering a cab. The platform offers an administrator interface where the taxi company can manage the content and access all bookings, driver information and customer information. All customer bookings are managed via an automated system which means they have an electronic record of past bookings.

# Elaboration – Iteration 1.1

# Domain Model

# Architectural Design

## Conceptual Architecture

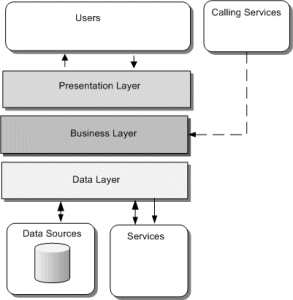
The components used in architecture are organized in layers. This structure organizes the project in three main categories: **presentation**, **business** and **data**. Each of the layers contains objects related to the particular concern it represents.

The presentation layer contains all of the classes responsible for presenting the UI to the end-user or sending the response back to the client.

The business layer represents the logical grouping of components and services that provide the business functionality in the application.

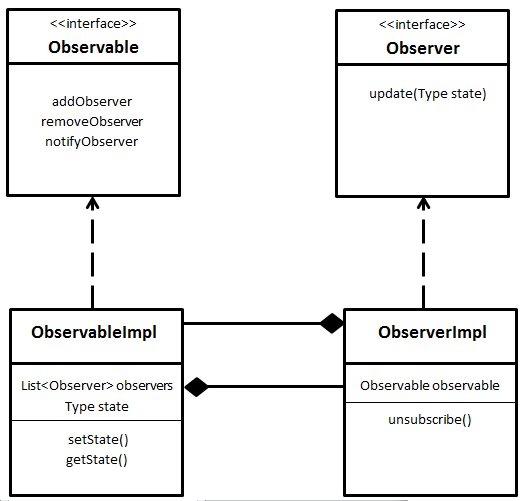
The data layer is the logical grouping of the components and services that provide data access functionality in the application.

I chose to use this layer because it should greatly simplify the way I manage the software infrastructure.



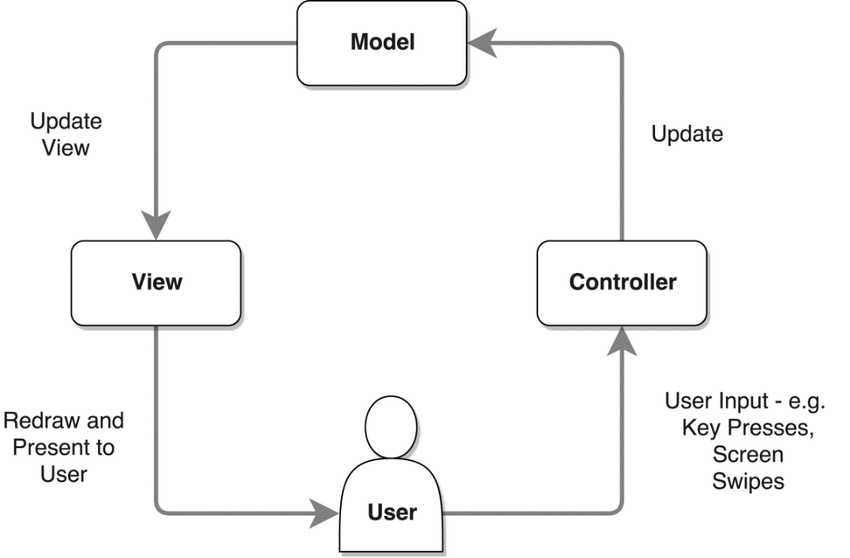
Observer Design Pattern is a behavioral design pattern among the Gang Of Four Design Patterns. Being a behavioral design pattern, the Observer pattern deals with how objects of the designed system interact with each other. This pattern is used when there are multiple subscribers observing updates from a publisher. The Subject publishes its change of state to its subscribers. The observers receive these changes and update themselves accordingly.

**Observer pattern diagram**



## Package Design

## 



## Component and Deployment Diagrams

# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

*[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]*

## Class Design

*[Create the UML class diagram; apply GoF patterns and motivate your choice]*

# Data Model

*[Create the data model for the system.]*

# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

# Future improvements

*[Present future improvements for the system]*

# Bibliography

<https://www.javabrahman.com/design-patterns/observer-design-pattern-in-java/>

<https://norsamsiah.files.wordpress.com/2010/01/lab-003-domain-modeling1.pdf>

<https://www.techrepublic.com/article/properly-package-your-java-classes/>