DormShare
Analysis and Design Document
Student: Eric Dumea
Group: 30433

Revision History

Date	Version	Description	Author
13/apr/18	1.0	Initial Version	Eric Dumea

Table of Contents

I. Project Specification

DormShare is a software solution that tries to deal with the problem of students not affording accommodation in big cities that usually have touristic attractions by sharing dorm rooms. The application will provide a easy to use, minimalistic interface for the students.

II. Elaboration – Iteration 1.1

1. Domain Model

The DormShare application will be developed obeying the OOP principles, but also the SOLID principles. Thus, multiple classes will be needed, most probably one for each table in the Database design, at least one for the GUI and some for the interaction between the basic models and the finite product. A preliminary class design diagram is:

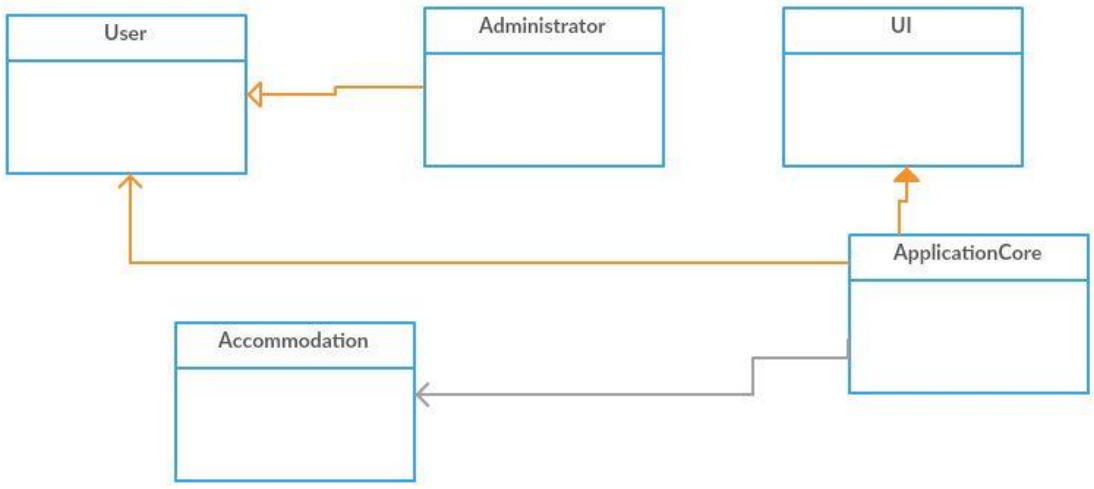
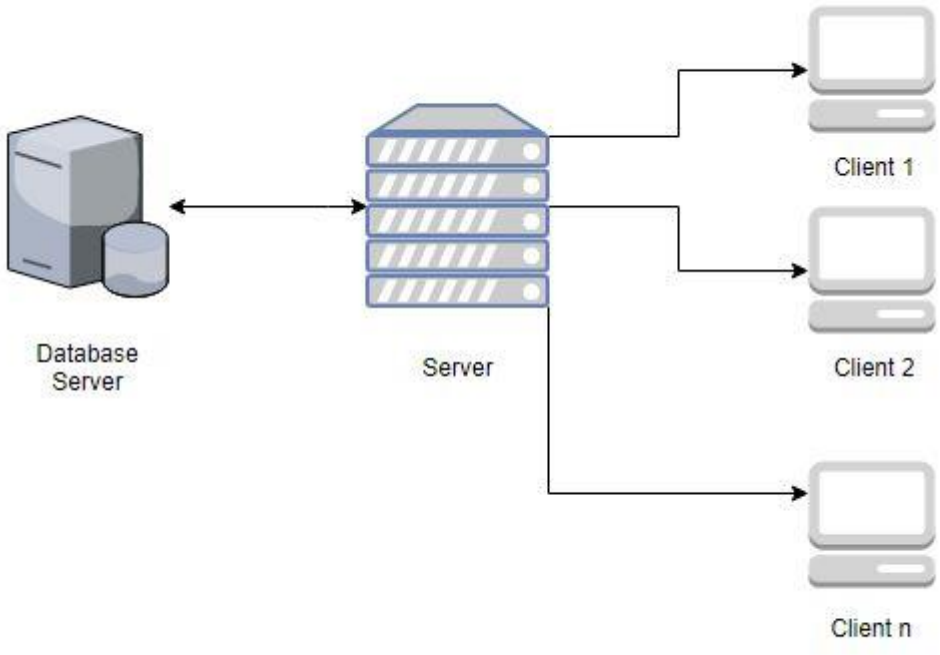


Figure 1: Class Diagram

2. Architectural Design

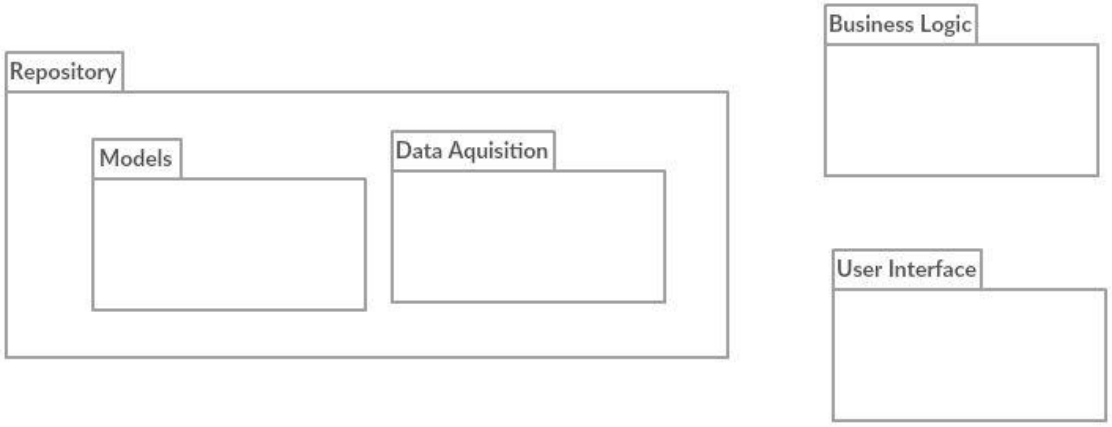
2.1 Conceptual Architecture

The project will be implemented using the Client-Server architectural pattern. This choice was made because the application manages the interaction between multiple users, and it needs a Server application to do the connection to the database and also to manage the users. The Server application will be the backend of the application. Provided is a system architecture diagram:



2.2 Package Design

Packages are used to separate components that are not related or loosely related.


Repository
Models
Data Aquisition
Business Logic
User Interface

2.3 Component and Deployment Diagrams

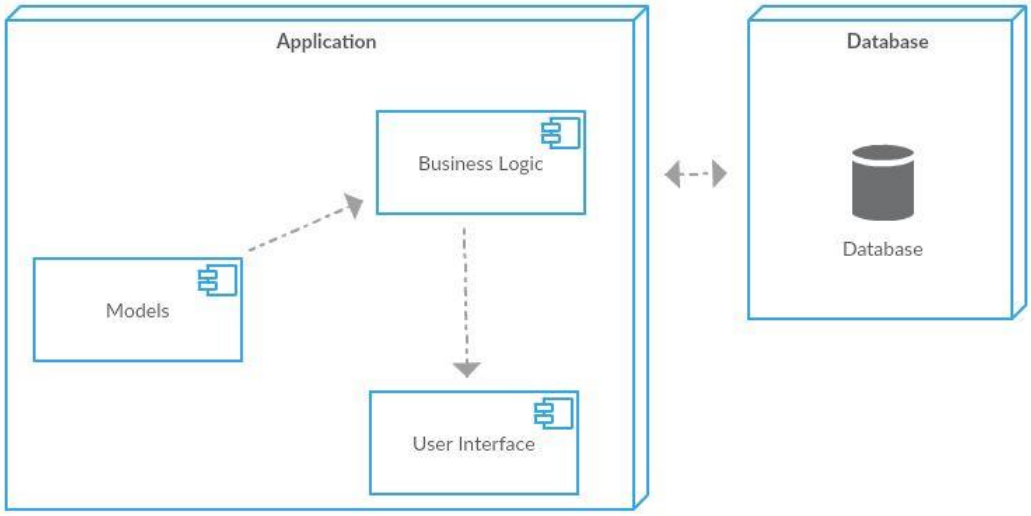


Figure 2 : Component Diagram

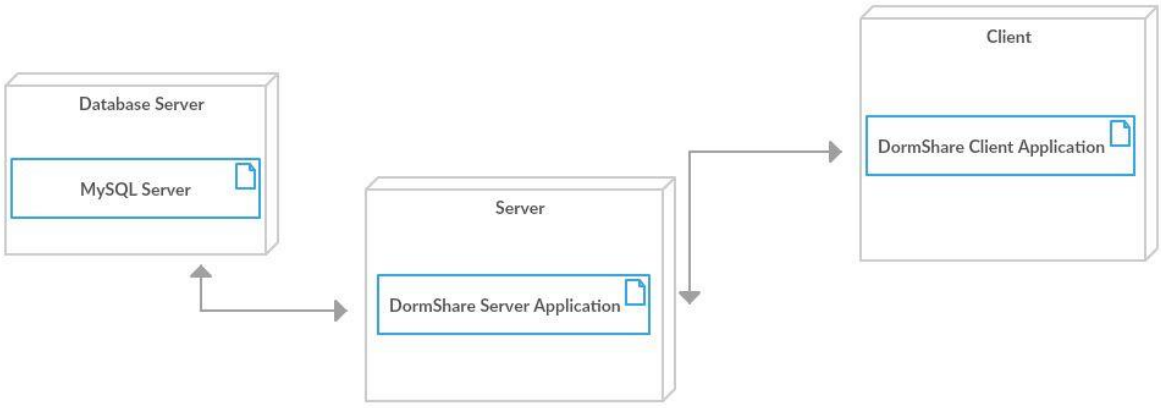


Figure 3 : Deployment Diagram

III. Elaboration – Iteration 1.2

1. Design Model

1.1 Dynamic Behavior

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

1.2 Class Design

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

2. Data Model

[Create the data model for the system.]

3. Unit Testing

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

IV. Elaboration – Iteration 2

1. 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.]

2. Design Model Refinement

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

V. Construction and Transition

1. System Testing

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

2. Future improvements

[Present future improvements for the system]

VI. Bibliography