Phase 3: Design Patterns and Implementation Phase

Note: *It is expected that all members of the group will make a reasonable contribution in writing parts of this report, that way one group member does not get overstretched. Furthermore, it ensures all members get practice of writing technical reports.*

Please submit 1, 2, 3.1, 3.3 below as a single MS word/PDF document so that it is easier for me to provide comments.

Submit code in 3.2 as a zipped file.

Submit JAVAdoc/PyDOC/etc. style documentation as another zip file

1. Introduction

1.1. Purpose

This document is written to show the implementation of the design of a hotel reservation application, coded by the Chaotic Coders group (also known as Group #3), with members Alex, Ethan, Minas, and Nura. We have used numerous algorithms and techniques to fulfill the intended design of the program. In this document, there will be a description of the detailed design patterns we used throughout this project, as well as technical documentation along with source code to further illustrate the results of our efforts. The important files will be listed together with documentation to explain each class and method to further improve understanding of the underlying details of the project. The final result of the project will be a hotel reservation system that allows for creating reservations, canceling reservations, modifying reservations, and more. There will be a complex account management system split between users and managers as well, and additional functionality with a complete GUI to demonstrate a full, complete application ready for use.

The contributions of each member to this phase of the project are as listed below:

Alex: In this phase, I was able to get the connection to the database working, Objects can now be stored in the database and all (except Reservations) can be pulled from the database. I also updated several files with the JavaDoc comments to generate documentation.

Ethan: In this phase, I have worked mostly on getting reservations completed. I have added the checking for room availability and occupancy accuracy. I have also done the screens that we need for managers to view all reservations and make edits if need be, as well as the users. Right now, all that needs to be done is advanced filtering but since it will be difficult to do that in a terminal window, I plan on getting more of it done once the GUI for that is started.

Minas: In this phase, I worked on the Edit Account screen for both users and managers. Now they can Edit their account, change name, username, password etc. They can also delete their accounts if they wish to do so. Also, added functionality to the new login screen we have, connecting it to user and manager panels it should lead to.

Nura: In this phase, I have completed the login screen, and from the login screen when the sign up button is clicked, it will take you to the sign up screen. And also, there's a login button from the sign up screen that can take you back to the login screen. I also did some editing on the screens.

As for this report, sections have been divided equally amongst team members:

Alex: For this document, I contributed my individual responses where necessary, as well as assisting with section 1.3. I have written the backend and database portion of 3.1 and generated JavaDoc documentation for the classes, Hotel, HotelManager, Room, RoomManager, Database, DatabaseManager, and DatabaseConnector.

Ethan: I have written section one, the introduction of this paper. I have also provided documentation for the classes Reservation, ReservationManager, User, Manager, and AccountManager.

Minas: I was responsible for writing the frontend section of 3.1, giving my individual responses where necessary and providing documentation for UserPanel, ManagerPanel, EditAccountInfo and EditAccountInfoForm classes.

Nura: I am responsible for part of the frontend. And my individual section was the design pattern. And also, Login.java, SignUp.java, Reservation.java, Hotel.java

Overall, the purpose of this document is to show our implementation and design patterns in depth

as we continue on the path to complete this project.

1.2 Definitions, Acronyms or Abbreviations

There are a few definitions, acronyms, or abbreviations that are included in this document.

List of definitions, acronyms and abbreviations:

JDBC - Java Database Connector

SQL - Structured Query Language

1.3 References

The references we have used for this paper are presented below. Works are referenced from their origin source.

List of references:

https://www.h2database.com/html/main.html

https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/

1.4 Overview

This paper will be sectioned into three parts. It will begin with the introduction for the paper, which serves as the current section in which this overview is being detailed. The next section, section two, will discuss the design patterns present in our project. There will be specific patterns identified, and outlined to further exemplify our implementation of this program. In the last section, section three, the technical documentation and source code will be addressed. There will be a listing of important files and provided documentation for each class we have currently made. Overall, we intend to illustrate the implementation and design of our project in these sections.

2. Design Patterns

In class we talked about various design patterns. Identify as many design patterns as possible that you have implemented in your project. Discuss in what context the design pattern is implemented in your project.

We used:

Singleton pattern: to ensure that the classes have instances and provide a global point of access to it. For example, Database Connection Manager, which was used to maintain instances of the database connection throughout the application.

Factory pattern: for creating users with different roles (e.g., Admin, Guest)

Decorator pattern: to add behavior and features to our objects.

MVC (Model-View-Controller) pattern: to manage our application's data

Adapter Pattern: which allows incompatible interfaces to work together by acting as a bridge.

3. Technical Documentation and Source Code

Good technical documentation is specific, concise, and relevant, providing all the information important to the programmers who write the code, and testers who verify that the software works as intended.

3.1 Important files

- Main
    - The entry point of the application, initializing the program, and setting up dependencies for both the backend and frontend.

Backend:

- Database.java
    - The main object used by the application is to store the objects to perform operations on them.
- DatabaseConnector.java
    - This class is used to connect to the database, create tables, add objects to the database, and query data from the database. Utilizes MySQL database using H2 and JDBC libraries.
- DatabaseManager.java
    - Serves as an intermediary for accessing and modifying database-related objects, abstracting database operations for other managers.
- AccountManager.java
    - Manages user accounts, including account creation, updating details, and authentication processes.
- User.java
    - This class represents the everyday user/customer of the Hotel application containing information like username, password, and personal details. This class is used to allow app navigation and placing reservations.
- Manager.java extends User.java
    - This class is a subclass of User with additional privileges to Edit Hotels, create

new Hotels, add/edit rooms, and adjust reservations. This account type is for employees and administrators.

- Hotel.java
    - This class represents a hotel entity with attributes like hotelID, hotelName, and list of hotelRooms. It represents the physical hotels and rooms that customers wish to reserve.
- HotelManager.java
    - This class contains methods for editing hotel information, creating new hotels, and managing hotel data.
- Room.java
    - This class represents a hotel room entity with attributes such as roomID, roomNumber, numOfBeds, and pricePerNight. It is responsible for managing room-specific information and tracks reservations associated with the room.
- RoomManager.java
    - This class contains methods to manage room creation, deletion, and pricing calculations.
- Reservation.java
    - This class represents a reservation entity, likely containing details like reservation ID, associated user, hotel, room, and reservation dates.
- ReservationManager.java
    - This class is responsible for handling reservation-related processes, such as creating, modifying, or deleting reservations.

Frontend:

- EditAccountForm.java
    - Description: This file contains the GUI logic for managing account-related actions. It provides options for editing account information, signing out, or deleting an account.
    - Purpose: Allows the user or manager to manage their account settings with an intuitive interface.
- EditAccountInfoForm.java

    - Description: This file handles the GUI for editing account details such as full name, username, and password.
    - Purpose: Provides a form to update user account details. Ensures proper validation and saves the changes to the backend.
- GUIManager.java
    - Description: This file acts as the bridge between the GUI and the backend operations. It includes methods to launch specific screens and manage user interactions with the GUI.
    - Purpose: Centralized control for handling the transitions between different GUI panels and backend functions.
- Login.java
    - Description: Contains the logic for the login interface, where users can input their credentials to gain access to the application.
    - Purpose: Facilitates user authentication, determining whether the user is a Manager or a regular User. Redirects to the appropriate dashboard after login.
- ManagerPanael.java
    - Description: Implements the Manager's dashboard. Provides options for viewing reservations, editing hotel details, and managing accounts.
    - Purpose: Acts as the main panel for all manager-level operations in the application.
- SignUp.java
    - Description: This file provides the interface for new users to create an account by entering their name, email, and password.
    - Purpose: Allows new users or managers to register into the system.
- UserPanel.java
    - Description: Implements the User's dashboard. Provides options for making reservations, viewing reservations, and editing account information.
    - Purpose: Acts as the main panel for user-level operations in the application.

Database:

- MySQL Database

- Stores persistent data, including hotels, rooms, users, and reservations.
- Key Tables: Hotels Table, Rooms Table, Users Table, and Reservations Table

3.2 Source Code with documentation

Source Code and Documentation in separate zip files

3.3 Individual contributions of each member

Alex: Database.java, DatabaseManager.java, DatabaseConnector.java, Hotel.java, HotelManager.java, Room.java, RoomManager.java, User.java, Manager.java

Ethan: Reservation.java, ReservationManager.java, AccountManager.java

Minas: UserPanel.java, ManagerPanel.java, EditAccountForm.java, EditAccountInfoForm.java

Nura: Login.java, SignUp.java, Reservation.java, Hotel.java