CS 3004 - Software Design and Analysis

Coding Task 2 – Java: Hotel Management System (MIS) with File and Database Integration.

The document contains the following sections

Objective

This assignment aims to enhance the existing console-based Hotel Management System (MIS) by introducing file handling and SQL database integration. By doing this, data such as room details, guest information, and bookings can be saved and retrieved from external storage, ensuring that the system's data remains available even after the program is terminated for long-term storage. Another key objective is to implement a ***2-tier architecture*** to achieve ***modularity*** by clearly separating the data handling layer (DAL) from the business logic layer (BLL), making the system more organized, and easier to maintain.

Prerequisites

Before diving into the assignment, here's what you should have in place:

1. **Java OOP Concepts**: Make sure you're comfortable with fundamental OOP principles like encapsulation, inheritance, and polymorphism.
2. **File Handling Skills**: You should know how to read from and write to files using classes like *FileReader*, *FileWriter*, *BufferedReader*, and *BufferedWriter*. Additionally, understanding how to serialize objects with *ObjectOutputStream* will be helpful.
3. **JDBC Knowledge**: Familiarize yourself with setting up and managing database connections using Java Database Connectivity (JDBC).

4. **Basic SQL Skills**: Have a grasp of essential SQL commands for creating, updating, and querying tables in your database.
5. **Understanding 2-Tier Architecture**:
    - **Business Logic Layer (BLL)**: This is where you'll implement core operations such as handling bookings and managing guests.
    - **Data Access Layer (DAL)**: This layer will take care of interactions with both the database and file storage.
6. **Development Setup**:
    - Install the **Java JDK** on your machine.
    - Set up a **SQL database** (choose the one taught in the class).
    - Use an IDE like **Eclipse** or **IntelliJ** for coding, along with a database management tool like **MySQL Workbench**.

Outline of Tasks

1. Prompt the user at the start of the program to choose between using a **database** or **text files** for data storage.

2. Implement 2-Tier Architecture

   Busisness Logic Layer handles core functionalities such as room management, guest management, and booking transactions.

   Data Access Layer (DAL) manages interactions with the chosen storage mechanism (either SQL database or text files-.txt or .csv).

3. Design and create tables in the database to store

   i. Rooms: Attributes to capture essential information about various room types, their status, pricing, and available amenities.

   ii. Guests: Attributes to store guest information, including identification details, contact information, and any relevant booking history.

   iii. Bookings: Attributes to maintain records of booking transactions, linking guests to the rooms they have reserved, along with stay duration and associated costs.

4. Implement CRUD Operations for both storage mechanisms.

   i. Create: Implement functionality to add new rooms and guests, ensuring proper validation of input data.

      Example: a method that adds a new guest that includes detailed validation checks for their name, email format, and phone number. Additionally, ensure that the guest's booking history is initialized correctly upon creation.

   ii. Read: Develop methods to retrieve and display information about available rooms, guests, and their booking histories with filters for specific criteria.

      Example: a function that retrieves a guest's booking history, showing details such as room type, stay duration, and total costs. Implement search filters for guests based on booking dates or room types.

   iii. Update: Create functionality to modify existing room and guest information with the ability to track changes and maintain a log of updates.

      Example: Implement a method that updates a guest’s email and phone number, including a check to ensure the new email is unique in the database.

iv. Delete: Ensure robust functionality for removing rooms or guests, including cascading effects on related bookings and data integrity checks.

Example: Implement a method that removes a guest after confirming that all their bookings are completed, ensuring that any associated booking records are also deleted. Provide a confirmation prompt to prevent accidental deletions.

The examples given for each CRUD operation are just for your reference. You should build on the features that you have already created in your Hotel Management System. Try to make those functions better by adding extra checks, logs, and ways to keep the data safe and correct. Think about real-life situations to make your system work well and be easy to use.

5. Implement try-catch blocks to manage exceptions gracefully, ensuring the program doesn't crash due to unexpected input or database issues.

6. Extend your already implmented console-based menu for user interactions, making it easy for users to navigate between different functionalities.

Deliverables

1. Complete source code for the Hotel Management System, organized with clear structure and proper comments.
2. A description of the database schema, including table definitions and relationships between tables for Rooms, Guests, and Bookings.
3. Sample text files (CSV or TXT) with data for Rooms, Guests, and Bookings to facilitate testing and demonstration of the system's functionality.

Rubric

Criteria	Description	Marks
File Handling Integration	Successfully implements file handling for data storage, ensuring proper reading and writing of room, guest, and booking data.	10
Database Integration	Effectively integrates SQL database functionality, allowing seamless CRUD operations and data persistence for rooms, guests, and bookings.	10
Menu	Creates a user-friendly console menu that facilitates easy navigation between functionalities, enhancing the user experience.	5
CRUD Operations	Implements comprehensive Create, Read, Update, and Delete operations for rooms and guests, with thorough validation and error checks.	20
2-Tier Architecture	Successfully separates business logic and data access layers, ensuring modularity and maintainability in the system design.	10
Database Schema	Designs a well-structured database schema with clear relationships and attributes for Rooms, Guests, and Bookings, properly documented.	10
Error Handling	Implements robust error handling throughout the application, utilizing try-catch blocks to manage exceptions gracefully and maintain stability.	10
Flow of Execution	Ensures a logical flow of execution within the application, with clear pathways for user inputs and expected outputs.	10
Code Quality	Maintains high code quality with adherence to coding standards, including consistency, readability, and efficient use of resources.	10
Comments	Provides clear and meaningful comments throughout the codebase, enhancing readability and understanding of the logic implemented.	5
Total		100