Software Requirements Specification (SRS)

Hotel Management System

---

1. Introduction

The Hotel Management System is a comprehensive, Java-based application designed specifically for internal use by hotel staff. The primary goal of the system is to streamline and simplify essential hotel operations, ensuring that daily tasks such as room bookings, customer and employee management, and food and drink services are handled efficiently. By providing a user-friendly command-line interface (CLI), the system enables managers, receptionists, and administrative personnel to quickly access and update critical information without the need for complex software solutions.

Key features of the system include:

- **Room Management**: The system tracks hotel room statuses—whether they are occupied or available—and allows staff to book or vacate rooms. Room details such as type and price are maintained, and all changes are persistently stored in text files.
- **Customer Management**: Customer details (such as name, age, phone number, and assigned room) are captured and maintained. When a room is booked, the system automatically updates the customer records and room status, ensuring accurate tracking of occupancy.
- **Employee Management**: Employee information—including names, job titles, and salaries—is recorded and updated within the system. This feature helps hotel management keep an up-to-date log of all staff members.
- **Food and Drink Services**: The system provides a menu for food and drink items, complete with prices, allowing hotel staff to process orders and calculate totals for services rendered.
- **File Management**: To ensure data persistence, the system reads from and writes to text files using a dedicated FileManager class. This guarantees that all changes to rooms, customers, and employees are saved across sessions.

The application is built using an organized, modular approach, separating responsibilities across several classes—each handling a specific aspect of hotel operations. This design not only simplifies maintenance and future enhancements but also ensures that the system remains scalable and reliable for day-to-day use.

1.1 Purpose

The purpose of this document is to define and describe the software requirements for the Hotel Management System. The system is designed to be a simple, user-friendly, Java-based application that enables hotel staff to manage essential hotel operations such as:

- Room bookings and room status management
- Customer and employee record management
- Food and drink ordering services
- Persistent data storage using text files

This SRS is intended for developers, testers, project managers, and hotel staff to ensure that the system meets the operational needs of the hotel.

1.2 Scope

The Hotel Management System is an internal application (not customer-facing) that provides the following capabilities:

- **Room Management:**
    - Represent each hotel room (via the **Room** class) with a unique number, type, price, and occupancy status.
    - Enable booking of rooms for customers and marking them as occupied or available.
    - Persist room status between sessions using file storage.
- **Customer Management:**
    - Capture and store customer details (via the **Customer** class), including first name, last name, age, phone number, and assigned room number.
    - Update and remove customer records as needed (for example, when a customer checks out).
- **Employee Management:**
    - Manage employee data (via the **Employee** class), including personal details, job title, and salary.
    - Add or remove employee records with persistent storage.
- **Food and Drink Services:**
    - Provide a food menu (via the **Food** class) and a drink menu (via the **Drink** class) with item names and prices.

    - Process orders for food and drinks and calculate order totals.

- **File Management:**
    - Use the **FileManager** class to read from and write to text files, ensuring that all changes to room, customer, and employee records persist across sessions.
- **User Interface:**
    - Offer a command-line interface (CLI) via the **HotelManagementSystem** class that displays a main menu and guides the user through various operations.

1.3 Intended Audience

This document is intended for:

- **Developers and Testers:** Who will build and verify the system.
- **Project Managers:** To review and ensure the system meets business requirements.
- **Hotel Management and Staff:** To understand the system's functionality and operation.
- **Quality Assurance Personnel:** To ensure the system complies with the requirements.

2. Overall Description

2.1 Product Perspective

The Hotel Management System is a stand-alone Java application built using object-oriented principles. All the code is maintained in a single source folder, making the project structure simple and easy to manage. The system is composed of eight interrelated classes that work together to provide the essential functionality needed for managing hotel operations. These classes are:

- **Room**: Manages information about hotel rooms, including room numbers, types, prices, and occupancy status. It provides methods to mark a room as occupied or vacant.
- **Food**: Maintains a list of food menu items and their corresponding prices. It offers methods to display the menu and retrieve the price of a selected item.
- **Drink**: Similar to the Food class, it manages drink menu items and prices, and provides methods to display the menu and look up drink prices.
- **Employee**: Represents employee data, storing personal and job-related details such as names, gender, age, job title, and salary. It includes methods for accessing these details and displaying them in a readable format.

- **Customer**: Captures customer information such as first name, last name, age, phone number, and assigned room number. This class provides access to customer details and a formatted representation of the customer record.
- **FileManager**: Handles all file input/output operations for the system. It is responsible for reading from and writing to text files that store data related to rooms, customers, and employees, ensuring that information persists between sessions.
- **Hotel**: Serves as the central component that coordinates the various operations within the system. It manages room bookings, customer and employee records, and updates the files whenever changes occur.
- **HotelManagementSystem**: Contains the main method and provides a command-line interface (CLI) for hotel staff to interact with the system. This class displays menus, processes user input, and calls the appropriate methods in the other classes to perform tasks such as booking rooms, managing records, and processing orders.

Together, these classes provide a comprehensive solution for managing hotel operations, all within a straightforward project structure. By keeping all the classes in one source folder, the application remains simple, making it ideal for small-scale deployments, educational purposes, or environments where ease of maintenance is a priority.

2.2 Product Functions

The system provides the following functions:

- **Room Booking and Management:**
    - Display available rooms.
    - Book a room for a customer and mark it as occupied.
    - Vacate a room when a customer leaves.
    - Persist room status in a file.
- **Customer Management:**
    - Add new customer details when booking a room.
    - Remove a customer when they check out.
    - Display all customer information.
- **Employee Management:**
    - Add and remove employee records.
    - Display employee details.

- **Food and Drink Services:**
    - Display food and drink menus.
    - Process orders and compute total cost.
- **File Persistence:**
    - Read and write hotel data (rooms, customers, employees) to text files.
    - Update files automatically after any operation.
- **User Interaction:**
    - Present a main menu that offers options such as viewing available rooms, booking a room, managing customers and employees, and ordering food and drinks.
    - Accept user input via a CLI.

2.3 User Characteristics

The intended users are hotel staff with basic computer literacy. They will interact with the system using a command-line interface, and the interface is designed to be intuitive and straightforward.

2.4 Operating Environment

- **Hardware:** Standard desktop or laptop computers.
- **Operating System:** Platform-independent (Windows, macOS, Linux).
- **Software:**
    - Visual Studio Code or any other Java IDE.
- **Dependencies:**
    - File I/O access with appropriate read/write permissions.

2.5 Design and Implementation Constraints

- The system uses fixed-size arrays for storing customer and employee data, which may limit scalability.
- Data persistence is handled via text files rather than a database.
- The interface is text-based (CLI), which may limit usability compared to a graphical interface but simplifies implementation.

2.6 Assumptions and Dependencies

- The application is deployed on workstations with Java installed.

- Users will follow provided prompts and input valid data.
- The file system is accessible with appropriate permissions for reading and writing data.

3. Specific Requirements

3.1 Functional Requirements

1. Room Management (Room Class)

- **FR1.1:** The system shall represent each hotel room with a unique room number, a room type (e.g., Suite, Double, Single), a price, and an occupancy status.
- **FR1.2:** The system shall allow hotel staff to book a room for a customer. When a room is booked, the system shall mark it as occupied and update the corresponding file.
- **FR1.3:** The system shall allow hotel staff to vacate a room when a customer checks out, updating the room's status accordingly.

2. Customer Management (Customer Class)

- **FR2.1:** The system shall store customer details including first name, last name, age, phone number, and room number.
- **FR2.2:** The system shall add a customer record when a booking is made and remove a customer record when the customer leaves.
- **FR2.3:** The system shall update the customer file after each addition or removal.

3. Employee Management (Employee Class)

- **FR3.1:** The system shall store employee information including first name, last name, gender, age, job title, and salary.
- **FR3.2:** The system shall support adding new employees and removing existing employees, updating the employee file accordingly.

4. Food and Drink Services (Food and Drink Classes)

- **FR4.1:** The Food class shall display a formatted menu of food items with corresponding prices.
- **FR4.2:** The Drink class shall display a formatted menu of drink items with corresponding prices.
- **FR4.3:** The system shall allow hotel staff to input orders (using item numbers) and calculate the total cost for food and drink orders.

5. File Management (FileManager Class)

- **FR5.1:** The FileManager class shall provide methods to read from and write to text files.
- **FR5.2:** The system shall use these methods to save room, customer, and employee data persistently.
- **FR5.3:** All file operations shall use proper resource management (e.g., try-with-resources) to avoid resource leaks.

6. Booking and Overall Hotel Operations (Hotel Class)

- **FR6.1:** The Hotel class shall manage arrays of Room, Customer, and Employee objects.
- **FR6.2:** The system shall process room bookings by associating a Customer with an available Room.
- **FR6.3:** The system shall update the corresponding files after any booking, customer removal, or employee addition/removal.
- **FR6.4:** The system shall display available room details, customer information, and employee records upon request.

User Interface (HotelManagementSystem Class)

- **FR7.1:** The main menu shall present options including:
    - Viewing available rooms.
    - Booking a room.
    - Removing a customer.
    - Viewing customer information.
    - Adding an employee.
    - Viewing employee information.
    - Ordering food.
    - Ordering drinks.
    - Removing an Employee
    - Exiting the system.
- **FR7.2:** The system shall accept and process user input via a command-line interface.

- **FR7.3:** The system shall display confirmations and error messages for each operation.

3.2 Non-Functional Requirements

Performance

- **NFR1:** The system shall respond to user commands within 2 seconds under normal conditions.

Usability

- **NFR2:** The CLI must be clear and easy to navigate, with straightforward prompts and error messages.
- **NFR3:** Minimal training shall be required for hotel staff to operate the system.

Maintainability

- **NFR4:** The system shall be modular with a clear separation between the model, service, utility, and main layers.
- **NFR5:** Code shall follow standard Java naming and documentation conventions.

Reliability

- **NFR6:** The system shall validate user inputs to prevent invalid operations (e.g., booking an already occupied room).
- **NFR7:** The system shall handle file I/O exceptions gracefully, displaying appropriate error messages.

Portability

- **NFR8:** The application must be platform-independent and run on any operating system supporting Java SE 8 or later.

Security

- **NFR9:** Data files shall be stored in a secure location with restricted access (given the internal, controlled environment).

4. System Architecture and Class Descriptions

4.1 Architectural Overview

The Hotel Management System is structured as a collection of interrelated Java classes, all contained within a single source folder. Each class has a specific role in handling different aspects of the hotel's operations. The key components of the system include:

- **Room**: Stores details about hotel rooms, such as room number, type, price, and occupancy status.
- **Food & Drink**: Manage the available menu items, including food and drink options, along with their prices.
- **Employee**: Maintains employee records, including personal details and job-related information.
- **Customer**: Stores customer details, including personal information and assigned room numbers.
- **FileManager**: Handles reading and writing data to files, ensuring that hotel information is stored and retrieved correctly.
- **Hotel**: Acts as the central class, managing room bookings, customer and employee records, and overall system operations.
-   Provides the main entry point for the program and serves as a command-line interface (CLI) for interacting with the system.

This straightforward structure ensures that all hotel-related operations are efficiently managed within a single source folder, making the system easy to understand and maintain.

4.2 Class Descriptions

1. **Customer**
    - **Purpose:**
      Represents a hotel customer and stores basic information.
    - **Attributes:**
        - firstName (String)
        - lastName (String)
        - age (int)
        - phoneNumber (String)
        - roomNumber (int)

- **Constructor:**
  Initializes a Customer object with all attributes.
- **Methods:**
  - Getter methods for each attribute.
  - toString(): Returns a formatted string summarizing customer details.

2. **Employee**
   - **Purpose:**
     Represents a hotel employee's personal and professional details.
   - **Attributes:**
     - firstName, lastName (String)
     - gender (String)
     - age (int)
     - jobTitle (String)
     - salary (double)
   - **Constructor:**
     Initializes an Employee with specific attribute values.
   - **Methods:**
     - Getter methods for each attribute.
     - toString(): Provides a formatted string representation.
     - Overridden equals() and hashCode() methods.
3. **Food**
   - **Purpose:**
     Manages the hotel's food menu.
   - **Attributes:**
     - menuItems: An array of food item names.
     - price: An array of corresponding prices.
   - **Methods:**
     - showMenu(): Displays the food menu with indices and prices.
     - getMenuItems(): Returns the array of food items.

        - getPrice(int choice): Returns the price for the selected food item.

4. **Drink**
    - **Purpose:**
      Manages the hotel’s drink menu.
    - **Attributes:**
        - menuItems: An array of drink item names.
        - price: An array of corresponding prices.
    - **Methods:**
        - showMenu(): Displays the drink menu in a formatted manner.
        - getMenuItems(): Returns the array of drink items.
        - getPrice(int choice): Returns the price for the selected drink item.
5. **FileManager**
    - **Purpose:**
      Handles file operations for data persistence.
    - **Methods:**
        - readFile(String fileName): Reads data from the specified file and returns a String array of lines.
        - writeFile(String fileName, String[] data, boolean append): Writes the provided data array to the specified file using buffered I/O.
    - **Features:**
      Uses try-with-resources for proper resource management and error handling.
6. **Room**
    - **Purpose:**
      Represents a hotel room with details and occupancy status.
    - **Attributes:**
        - roomNumber (int)
        - roomType (String)
        - roomPrice (double)
        - isOccupied (boolean; default is false)

- **Constructor:**
  Initializes a Room with the given number, type, and price.
- **Methods:**
  - Getter methods for each attribute.
  - occupyRoom(): Marks the room as occupied.
  - vacateRoom(): Marks the room as available.
  - toString(): Returns a formatted string including room number, type, price, and occupancy status.

7. **Hotel**

- **Purpose:**
  Coordinates hotel operations such as room bookings, customer/employee management, and file updates.
- **Attributes:**
  - rooms: An array of Room objects.
  - customers: An array of Customer objects (capacity: 100).
  - employees: An array of Employee objects (capacity: 100).
  - customerCount and employeeCount: Track the current number of records.
- **Constructor:**
  - Hotel(int numRooms): Initializes the hotel with the specified number of rooms (each room is assigned a type and price based on its number).
  - Initializes arrays for customers and employees.
- **Methods:**
  - bookRoom(Customer customer): Books a room for a customer if available and updates files.
  - removeCustomer(String firstName, String lastName): Removes a customer record and vacates the corresponding room.
  - addEmployee(Employee employee) / removeEmployee(String firstName, String lastName): Manages employee records and updates persistence.

        - File update methods such as updateCustomerFile(), updateRoomFile(), and updateEmployeeFile().

8. **HotelManagementSystem**
    - **Purpose:**
      Provides the entry point for the application and manages the command-line interface.
    - **Main Flow:**
        - Displays the main menu with options:
            - Show available rooms
            - Book a room
            - Remove a customer
            - View customer information
            - Add an employee
            - View employee information
            - Order food
            - Order drink
            - Exit the system
        - Processes user input and calls the appropriate methods in the Hotel class.
    - **Helper Methods:**
        - Methods to display the menu and handle specific operations (e.g., bookRoomForCustomer(), removeCustomerFromHotel(), addEmployeeToHotel(), orderFood(), and orderToDrink()).

5. Appendices

5.1 Glossary

- **Booking:** Assigning a room to a customer.
- **CLI:** Command-Line Interface.
- **I/O:** Input/Output operations.
- **Occupancy:** Whether a room is currently booked.

- **Persistence:** Saving data to files for use across sessions.

5.2 References

- IEEE Std 830-1998 for Software Requirements Specifications.
- Java SE Documentation.
- Best practices for Java application design and architecture.

Summary

The **Hotel Management System** is a Java-based application that facilitates the efficient management of hotel operations. It allows users to handle essential tasks such as room booking, customer management, employee records, and food and drink services. The system ensures smooth operation by maintaining and processing relevant hotel data. It provides a **command-line interface (CLI)** for user interaction, making it easy to navigate and manage various hotel services. Additionally, the system utilizes file handling to store and retrieve data, ensuring persistence and reliability. This project aims to streamline hotel management processes and improve operational efficiency