EXPERIMENT-7

HOTEL MANAGEMENT SYSTEM

Software Design Document (SDD)

1. Introduction

1.1 Purpose

The purpose of this Software Design Document (SDD) is to provide a detailed design for the Hotel Management System. It is intended for software engineers and developers who will implement the system, and for stakeholders to understand the system's structure and functionality.

1.2 Scope

The Hotel Management System (HMS) allows users to register, login, search for rooms, book or cancel reservations, make payments, and check reservation status. Admins and hotel staff manage room details, availability, and customer records. A payment gateway handles secure online payments.

1.3 Definitions, Acronyms and Abbreviations

- HMS: Hotel Management System
- UI: User Interface
- DB: Database
- MVC: Model-View-Controller
- CRUD: Create, Read, Update, Delete

2. Overall Description

2.1 Product Perspective

This system is a standalone web-based application. It follows the MVC architecture, separating logic, UI, and data management. It is designed to be scalable and secure, integrating with third-party payment systems.

2.2 Product Functions

User Functionality:

- Register and login/logout
- Search room availability

- Book and cancel reservations
- Make secure payments
- View reservation status

Admin/Staff Functionality:

- Update room details (availability, pricing)
- Generate booking reports

System Functionality:

- Session management
- Role-based access control
- Integration with external payment gateways

2.3 User Classes and Characteristics

- **User:** Casual or registered guests using the platform to make or cancel reservations.
- **Admin:** Full system access to modify room details, manage bookings, and generate reports.
- **Hotel Staff:** Limited access for operational booking/cancellation tasks.
- **Payment Gateway (External Actor):** Handles payment processing.

2.4 Operating Environment

- **Frontend:** HTML5, CSS3, JavaScript
- **Backend:** Node.js / Python Flask / PHP
- **Database:** MySQL / PostgreSQL
- **Server:** Apache/Nginx on Linux
- **Browser:** Compatible with Chrome, Firefox, Edge, Safari

2.5 Constraints

- Secure authentication
- Must handle 60+ concurrent users
- Must support real-time booking updates
- Payment integration via secure API

2.6 Assumptions and Dependencies

- Users have internet and digital payment access
- Admin updates room details manually

3. System Features and Use Cases

3.1 Use Case List

1. User Login
2. User Registration
3. Search Room Availability
4. Book Room *(includes payment, modify booking)*
5. Cancel Booking *(<> Book Room)*
6. View Reservation Status
7. Update Room Info *(Admin/Staff)*
8. Logout *(Handled by session manager)*

3.2 Use Case Relationships

- <<*include*>>: Make Payment is always part Booking a Room.
- <<*extend*>>: Cancel Booking is an optional extension after booking.
- External Actor: Payment Gateway (interacts during Make Payment), Database (ontaining all the data of rooms, user accounts, bookings etc.).

3.3 Use Case Diagram Summary

Actors: Customer, Manager, Receptionist, Database, Payment Gateway
Use Cases: 10 major functionalities listed above with include and extend relationships.

4. System Design

4.1 System Architecture

- *Model Layer*: Handles business logic and database interactions.
- *View Layer*: HTML/CSS UI pages for each actor.
- *Controller Layer*: Processes input, interacts with model, returns appropriate view.

4.2 Database Design

Tables	Description
Users	user_id, name, email, password, phone
Rooms	room_id, room_type, price, status
Reservations	reservation_id, user_id, room_id, check_in, check_out
Payments	payment_id, reservation_id, amount, payment_status
Services	service_id, reservation_id, service_name, service_price

4.3 UI Design

Screens:

1. Login / Register
2. Dashboard (User/Admin/Staff)
3. Room Search
4. Booking Summary + Room Selection
5. Payment Confirmation
6. Reservation Status Pages
7. Admin Panel for room updates and reports

5. Component-Level Design

5.1 Booking Module

Inputs: Room type, check-in date, check-out date
Process: Search → Select room → Confirm booking → Make payment
Output: Booking confirmation with reservation details

5.2 Cancellation Module

Inputs: Booking ID
Process: Validate booking → Cancel reservation → Initiate refund
Output: Cancellation receipt

5.3 Admin Management Module

Functionality: Add/update room info; Generate booking/cancellation report

6. Security and Performance

6.1 Security Requirements

1. HTTPS protocol for secure communication.
2. Encrypted passwords (bcrypt/sha256).
3. Secure API calls to payment gateway.
4. Role-based permissions for all routes.

6.2 Performance Requirements

- Handle 100+ concurrent room bookings.
- Cache search results for faster lookup of room availability.
- Optimize database with indexes for quick retrieval of booking and customer data.

7. Future Enhancements

- Mobile app version for hotel reservations.
- Integration with third-party travel platforms (e.g., Expedia, Booking.com).
- Push notifications for booking confirmations and reminders.
- Loyalty program for frequent customers.

8. Appendices

8.1 Tools Used

- Visual Studio Code (development).
- XAMPP / Flask for backend testing.
- MySQL Workbench (database design).
- Draw.io (for use case diagram).