**Resort Management System (RMS)**

**1. Overview**

The Resort Management System (RMS) is a web-based application designed to streamline administrative and guest care operations in a resort. Designed for scalability and ease of use, the system is modular and follows the MVC architecture, making it compatible with both Java (Spring MVC) and .NET (ASP.NET Core MVC) frameworks. The core modules include:

1. Guest Management – Manages guest records and bookings.
2. Room Management – Handles room details and availability.
3. Booking Management – Enables guests to book rooms.
4. Billing and Payments – Manages guest billing and payments.
5. User Management – Handles authentication, authorization, and user roles.

**2. Assumptions**

1. The application will be deployed locally during development using a relational database (e.g., MySQL or SQL Server).
2. Role-based authentication will secure sensitive information.
3. ORM frameworks (Hibernate for Java or Entity Framework for .NET) will handle database interactions.
4. No containerization will be used for local deployment.

**3. Module-Level Design**

**3.1 Guest Management Module**

**Purpose**: Manages guest details and their booking history.

* **Controller**: GuestController
  + addGuest(guestData)
  + updateGuest(guestId, guestData)
  + getGuestDetails(guestId)
  + deleteGuest(guestId)
* **Service**: GuestService
  + Validate guest data.
  + Interact with the database for CRUD operations.
* **Model**: Guest
  + **Attributes**:
  + guestId (PK)
  + name (VARCHAR)
  + dateOfBirth (DATE)
  + gender (VARCHAR)
  + contactNumber (VARCHAR)
  + address (VARCHAR)
  + bookingHistory (TEXT)

**3.2 Room Management Module**

**Purpose**: Manages room details and their availability.

* **Controller**: RoomController
  + addRoom(roomData)
  + updateRoom(roomId, roomData)
  + getRoomDetails(roomId)
* **Service**: RoomService
  + Manage room details and availability.
* **Model**: Room
  + **Attributes**:
  + roomId (PK)
  + roomType (VARCHAR)
  + availabilityStatus (ENUM: AVAILABLE, BOOKED)
  + pricePerNight (DECIMAL)

**3.3 Booking Management Module**

**Purpose**: Facilitates booking of rooms by guests.

* **Controller**: BookingController
  + bookRoom(bookingData)
  + getBookingDetails(bookingId)
  + cancelBooking(bookingId)
* **Service**: BookingService
  + Validate and manage bookings.
* **Model**: Booking
  + **Attributes**:
  + bookingId (PK)
  + guestId (FK)
  + roomId (FK)
  + bookingDate (DATE)
  + checkInDate (DATE)
  + checkOutDate (DATE)
  + status (ENUM: CONFIRMED, CANCELLED)

**3.4 Billing and Payments Module**

**Purpose**: Handles billing and payment processing for guests.

* **Controller**: BillingController
  + generateBill(billData)
  + getBillDetails(billId)
  + processPayment(billId, paymentData)
* **Service**: BillingService
  + Generate bills and validate payments.
* **Model**: Bill
  + **Attributes**:
  + billId (PK)
  + guestId (FK)
  + totalAmount (DECIMAL)
  + paymentStatus (ENUM: PAID, UNPAID)
  + billDate (DATE)

**3.5 User Management Module**

**Purpose**: Manages user authentication and roles for secure access.

* **Controller**: UserController
  + registerUser(userData)
  + loginUser(username, password)
  + getUserProfile(userId)
* **Service**: UserService
  + Manage user credentials and roles.
* **Model**: User
  + **Attributes**:
  + userId (PK)
  + username (VARCHAR)
  + password (VARCHAR, Encrypted)
  + role (ENUM: ADMIN, GUEST, STAFF)

**4. Database Schema**

**4.1 Table Definitions**

1. **Guest Table**

CREATE TABLE Guest (

guestId INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

dateOfBirth DATE,

gender VARCHAR(10),

contactNumber VARCHAR(15),

address VARCHAR(255),

bookingHistory TEXT

);

1. **Room Table**

CREATE TABLE Room (

roomId INT PRIMARY KEY AUTO\_INCREMENT,

roomType VARCHAR(50),

availabilityStatus ENUM('AVAILABLE', 'BOOKED'),

pricePerNight DECIMAL(10, 2)

);

1. **Booking Table**

CREATE TABLE Booking (

bookingId INT PRIMARY KEY AUTO\_INCREMENT,

guestId INT,

roomId INT,

bookingDate DATE,

checkInDate DATE,

checkOutDate DATE,

status ENUM('CONFIRMED', 'CANCELLED'),

FOREIGN KEY (guestId) REFERENCES Guest(guestId),

FOREIGN KEY (roomId) REFERENCES Room(roomId)

);

1. **Bill Table**

CREATE TABLE Bill (

billId INT PRIMARY KEY AUTO\_INCREMENT,

guestId INT,

totalAmount DECIMAL(10, 2),

paymentStatus ENUM('PAID', 'UNPAID'),

billDate DATE,

FOREIGN KEY (guestId) REFERENCES Guest(guestId)

);

1. **User Table**

CREATE TABLE User (

userId INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) UNIQUE,

password VARCHAR(255),

role ENUM('ADMIN', 'GUEST', 'STAFF')

);

**5. Local Deployment Details**

**5.1 Environment Setup**

* Install JDK 17 or .NET SDK 7.0.
* Install MySQL or SQL Server.
* Use an application server (Tomcat for Java, Kestrel for .NET).

**5.2 Deployment Steps**

1. Clone the repository.
2. Configure the database connection string in application.properties (Java) or appsettings.json (.NET).
3. Run the provided SQL scripts to initialize the database schema.
4. Build and start the application locally.