Hotel Management System

Project Description :

The Hotel Management System is a software solution designed to simplify and automate various hotel management operations. This project is designed to create a powerful and efficient system to support the management of rooms, guests, billing, housekeeping and advertisements.

The main purpose of hotel management is to increase efficiency, improve customer traffic and increase revenue by providing hotel staff with a centralized management system to manage all aspects of hotel operations. The system will automate the manual process, eliminating data and increasing the access time to important data.

Main Functions:

Room Management: This system allows the receptionist to manage the room, check-in, check-out and order. It provides great information on viewing room details, assigning rooms to guests.

Guest Management: The system provides active guest registration, guest information management and guest history tracking. It stores important information about visitors, such as personal information, contact information, preferences and special requests.

Billing: The system automates the invoicing process by creating invoices, calculating charges, and invoicing. It supports multiple payment methods and integrates payment gateways for security and hassle.

Reporting and Analysis: The system provides comprehensive reporting and analysis to support decision making information. It generates information about room occupancy, revenue, guest response and other statistics. This information will help hotel management identify gaps, measure performance and improve operations.

User roles and permissions: The system uses role-based access controls to ensure effective access and data security. It defines user roles such as administrators, receptionists, and housekeepers, each with specific permissions and rights.

Hotel management System will be developed with modern technology and methods. The front-end will be built using HTML, CSS and JavaScript, providing a responsive and intuitive user interface. The backend will be implemented using Java and Spring Boot framework to ensure the performance and management of the system. MySQL database will be used for data storage and management.

The focus throughout the project is availability, reliability and security. Various tests, including unit tests, integration tests, and user tests, will be conducted to ensure the functionality and functionality of the system. A continuous integration and deployment approach will be used to provide frequent updates and improvements. As a result, hotel

management system aims to provide powerful and efficient software solutions to streamline hotel operations, improve guest experience and stimulate business development. By automating key processes and providing real-time data access, the system will enable hoteliers to deliver exceptional service and improve resource utilization.

Architecture :

The hotel management system follows a client-server architecture. The front-end is built using HTML, CSS, and JavaScript, while the back-end is built using Java and Spring Boot. The system uses a MySQL database to store and manage data.

Project Structure :

The project is divided into several folders:

Frontend: Contains the HTML, CSS and JavaScript files responsible for the user interface.

Backend: contains Java source code for implementing functions and APIs.

Databases: Contains SQL scripts for creating and managing MySQL databases.

Pre-requisites :

Here are the prerequisites for a Java Spring Boot project with MySQL and MongoDB, along with relevant links for download and installation:

1-Java Development Kit (JDK):JDK is required to compile and run Java applications, providing the necessary tools and libraries. Download and install the latest JDK version from Oracle's website.

- Download JDK: https://www.oracle.com/java/technologies/javase-jdk11-downloads.html

2-Integrated Development Environment (IDE): An IDE offers a comprehensive development environment for writing, debugging, and managing code. IntelliJ IDEA, Eclipse, or Visual Studio Code are popular choices for Java development.

- IntelliJ IDEA: https://www.jetbrains.com/idea/download/
- Eclipse: https://www.eclipse.org/downloads/
- Visual Studio Code: https://code.visualstudio.com/download

3-Spring Boot: Spring Boot simplifies Java application development by providing predefined configurations, automatic dependency management, and a streamlined development experience. Use Spring Initializr or Spring Tools for your IDE to create a Spring Boot project.

- Spring Initializr (Online): https://start.spring.io/

- Spring Tools 4 for Eclipse: https://spring.io/tools

- Spring Tools for Visual Studio Code: Install via Extensions in Visual Studio Code

4-MySQL Database: MySQL is a popular relational database management system. Install MySQL Community Server and optionally MySQL Workbench, a graphical tool for managing MySQL databases.

- MySQL Community Server: https://dev.mysql.com/downloads/installer/

- MySQL Workbench: https://dev.mysql.com/downloads/workbench/

5-MySQL Connector/J: MySQL Connector/J is the official JDBC driver for connecting Java applications to MySQL databases. Include this dependency in your project to enable connectivity and interaction with MySQL.

- Maven:

  - Add the following dependency to your project's pom.xml:

    xml

    ```
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.27</version>
    </dependency>
    ```

  - Maven Repository: https://mvnrepository.com/artifact/mysql/mysql-connector-java

- Gradle:

  - Add the following dependency to your project's build.gradle:

    ```
    implementation 'mysql:mysql-connector-java:8.0.27'
    ```

  - Gradle Repository: https://search.maven.org/artifact/mysql/mysql-connector-java/8.0.27/jar

6-MongoDB: MongoDB is a NoSQL document database. Install MongoDB Community Edition or use MongoDB Atlas, a cloud-based service for managing MongoDB databases.

- MongoDB Community Edition: https://www.mongodb.com/try/download/community

- MongoDB Atlas (Cloud-based service): https://www.mongodb.com/cloud/atlas/register

7-MongoDB Java Driver: The MongoDB Java Driver allows Java applications to connect and interact with MongoDB databases. Include this dependency to enable MongoDB integration in your Java Spring Boot project.

- Maven:

  - Add the following dependency to your project's pom.xml:

    xml

    <dependency>

      <groupId>org.mongodb</groupId>

      <artifactId>mongo-java-driver</artifactId>

      <version>3.12.12</version>

    </dependency>

    - Maven Repository: https://mvnrepository.com/artifact/org.mongodb/mongo-java-driver

- **Gradle:**

  - Add the following dependency to your project's build.gradle:

    implementation 'org.mongodb:mongo-java-driver:3.12.12'

  - Gradle Repository: https://search.maven.org/artifact/org.mongodb/mongo-java-driver/3.12.12/jar

Make sure to download and install the appropriate versions based on your system and project requirements. With these prerequisites in place, you'll be ready to start building your Java Spring Boot project with both MySQL or MongoDB as the database.

Role Based Access:-

Roles of Admin,Manager and Customer can be defined for Hotel management system:

Administrator:

Responsibilities: Administrators have the highest level of access and control over the Hotel Management System. They are responsible for system configuration, user management, and overall system administration.

Permissions:

Manage User Accounts: Administrators can create, modify, and delete user accounts within the system.

Define User Roles and Permissions: Administrators can assign roles and define the permissions associated with each role.

System Configuration: Administrators can configure system settings, such as room types, pricing, taxes, and other global parameters.

Generate Reports and Analytics: Administrators have access to generate comprehensive reports and analytics on various aspects of hotel operations, such as revenue, occupancy, guest feedback, and performance metrics.

Manage System Security: Administrators can manage security settings, including password policies, access controls, and data backups.

Manage Room Inventory: Administrators have the authority to add, edit, or remove rooms from the hotel's inventory.

Manage Hotel Facilities: Administrators can update information about hotel facilities, amenities, and services.

Hotel Manager:

Responsibilities: Hotel Managers oversee the day-to-day operations of the hotel and are responsible for ensuring smooth functioning and guest satisfaction.

Permissions:

Manage Bookings: Hotel Managers can view and manage room reservations, including modifying booking details and handling cancellations.

Monitor Room Occupancy: Hotel Managers can track room availability, check-in/check-out status, and ensure optimal occupancy rates.

Manage Hotel Rates: Hotel Managers have the authority to set and adjust room rates, special offers, and discounts.

Guest Relationship Management: Hotel Managers oversee guest satisfaction, handle VIP requests, and manage customer loyalty programs.

Customer:

Responsibilities: Customers are individuals or guests who interact with the Hotel Management System to make reservations and manage their bookings.

Permissions:

Create and Manage Bookings: Customers can search for available rooms, make reservations, and manage their booking details (e.g., check-in/check-out dates, room preferences).

Update Personal Information: Customers can update their personal information, contact details, and preferences within their user profile.

View and Pay Invoices: Customers can view their invoices, including room charges and additional services, and make payments.

Provide Feedback: Customers can share feedback and ratings about their stay, services, and overall hotel experience.

Project Flow:

- Frontend Development
- Backend Development
- Integration
- Deployment

Milestone 1: Frontend Development:

Frontend development involves building the user interface (UI) and implementing the visual elements of the Hotel Management System. It focuses on creating an intuitive and engaging user experience that allows users to interact with the application seamlessly. The following activities are part of the frontend development process:

Activity 1: UI Design and Layout

- Design the overall user interface (UI) for the Hotel Management Application.
- Create wireframes and mockups to visualize the layout and structure of the application.
- Determine the color scheme, typography, and overall visual style.
- Implement responsive design to ensure the application is compatible with different devices.

Activity 2: Room booking UI

- Develop the room booking functionality, allowing users to book rooms according to dates and availaibility.
- Implement the room booking page, displaying details about the rooms with a picture of the interior.

Activity 3: User Authentication and Profile UI

- Implement user authentication and registration functionality, including login and signup forms.
- Create user profile pages, allowing users to view and edit their personal information, manage bookings, and track their booking history.
- Design password reset and email verification flows for user account management.

Milestone 2: Backend Development:

Backend development involves building the server-side components and logic of the Hotel management system. It focuses on handling the business logic, processing requests from the frontend, and interacting with the database. The following activities are part of the backend development process:

Activity 1: Server Setup and API Development

- Set up the server environment and choose a suitable backend framework such as Java Spring Boot.
- Develop the RESTful APIs for room search, booking management, user authentication, and profile management.
- Implement server-side validation and error handling for API requests and responses.
- Integrate with external services such as payment gateways if required.

Activity 2: Database Integration and ORM

- Set up the database server (e.g., MySQL or MongoDB) and establish the necessary database connections.
- Design the database schema based on the application requirements, including tables/entities and their relationships.
- Implement object-relational mapping (ORM) techniques (e.g., Hibernate or Spring Data) to interact with the database.
- Develop database queries and CRUD operations for room and services data, user information, and bookings.

Milestone 3: Integration:

Integration is the process of combining and connecting the frontend and backend components of the Hotel management system to create a unified and fully functional system. It involves establishing communication channels, exchanging data, and ensuring seamless interaction between the frontend UI and backend APIs. The following activities are part of the integration process:

Activity 1: Frontend-Backend Integration

- Integrate the frontend UI components with the backend APIs, ensuring proper communication and data exchange.
- Implement API calls from the frontend to retrieve data, make bookings, and manage user-related actions.
- Handle data validation and error responses between the frontend and backend components.
- Conduct thorough testing to ensure seamless integration and compatibility between frontend and backend.

Milestone 4: Containerization of the Application

Containerization is the process of packaging an application and its dependencies into a standardized unit called a container. Containers offer a lightweight and consistent runtime environment that can be easily deployed across various platforms and environments.

Activity 1: Dockerfile Creation

- Create a Dockerfile that defines the necessary steps to build the application image.
- Specify the base image, install dependencies, and configure the container environment.

Activity 2: Building the Docker Image

- Build the Docker image using the Dockerfile.
- Include all the required application files and dependencies in the image.

Activity 3: Container Testing

- Run and test the container locally to ensure it functions as expected.
- Verify that the application runs within the container environment without issues.

Activity 4: Publishing the Docker Image

- Push the built Docker image to a container registry (e.g., Docker Hub) to make it accessible to the Kubernetes cluster.
- Ensure proper tagging and versioning of the image for easy identification.

Milestone 5: Deployment to Kubernetes Cluster

Kubernetes provides a platform for automating the deployment, scaling, and management of containerized applications. Deploying the application to a Kubernetes cluster enables efficient orchestration and scalability.

Activity 1: Kubernetes Manifest

- Create Kubernetes manifest files (YAML or JSON) to define the deployment specifications.
- Specify details such as the container image, resource requirements, and desired replicas.

Activity 2: Deploying the Application

- Use the Kubernetes command-line interface (kubectl) or deployment tools to apply the manifest files and deploy the application.
- Verify that the application pods are running and healthy within the cluster.

Activity 3: Exposing the Application

- Configure a Kubernetes service to expose the application internally within the cluster.
- Define appropriate service type (ClusterIP, NodePort, LoadBalancer) based on requirements.

Activity 4: Verification

- Test the deployed application to ensure it functions correctly in the Kubernetes environment.