**Chapter 1: Introduction**

In the dynamic and fast-paced world of the hospitality industry, the efficient and seam-less management of a hotel's diverse operations is paramount to ensuring guest satis-faction, optimizing resource utilization, and ultimately, achieving business success. The Hotel Management System (HMS) is the technological backbone that empowers hotels, resorts, and other lodging establishments to achieve these objectives.

The Hotel Management System is a comprehensive software solution designed to streamline and automate the vast array of tasks and responsibilities that hotel staff and management encounter daily. From managing room reservations and guest check-ins to overseeing billing, staff coordination, and analytics, the HMS plays a central role in enhancing operational efficiency and guest experience.

The modern hospitality landscape is marked by high expectations and intense competi-tion. Guests demand a seamless and personalized experience, expecting their stays to be characterized by comfort, convenience, and exceptional service. In this context, the HMS emerges as a pivotal tool that not only helps hotels meet these expectations but also provides an edge in the ever-evolving industry.

Reservation and booking capabilities allow guests to plan their stays in advance, providing them with the flexibility to choose room types, specify dates, and personalize their bookings. By reserving rooms ahead of time, hotels can better anticipate and plan for guest arrivals, optimizing their room inventory and overall resource allocation.

In conclusion, the Hotel Management System is the cornerstone of modern hospitality management. It offers a holistic solution that empowers hotels to deliver top-tier guest experiences, streamline operations, and maintain comprehensive records of their daily activities.

Top of Form

**Chapter 2: System Requirements**

**2.1 Hardware Requirements**

Server Hardware:

•A dedicated server or cloud-based virtual machine may be needed to host the database and server-side components of the HMS. The hardware requirements will depend on the system's expected load and the chosen database management system (DBMS).

**2.1 Software Requirements**

Operating System:

• Server: The choice of the server's operating system depends on your preference and compatibility with your chosen database system. Common choices include Windows Server, Linux distributions (e.g., Ubuntu, CentOS), and macOS Serv-er.

**Java Development Kit (JDK):**

• The HMS is built using Java, so you'll need the appropriate JDK installed on the server and client devices. The version of JDK may vary based on the Java framework or libraries used, but Java 8 or later is recommended.

**Database Management System (DBMS):**

• You will need a DBMS to store and manage data. Common choices for Java-based HMS include MySQL, PostgreSQL, Oracle Database, or Microsoft SQL Server. The choice depends on factors like scalability, licensing, and specific requirements.

**Integrated Development Environment (IDE):**

• Developers will require an IDE for Java development. Popular options include Eclipse, IntelliJ IDEA, or NetBeans.

**Chapter 3: System Design**

**3.1 Proposed System**

**1. Modules:**

* Admin Module: Manages admin login and authentication.
* Room Management Module: Handles room allocation, check-in, check-out, and occupancy viewing.
* Database Module: Manages database connections, data retrieval, and data insertion.

**2. Database Design:**

* The system interacts with a MySQL database.
* Key tables include room\_allotment for storing guest information and room occupancy data.

**3. User Interface:**

* + The current code is text-based, but in a more comprehensive sys-tem, you'd likely implement a graphical user interface (GUI) for admin interactions.

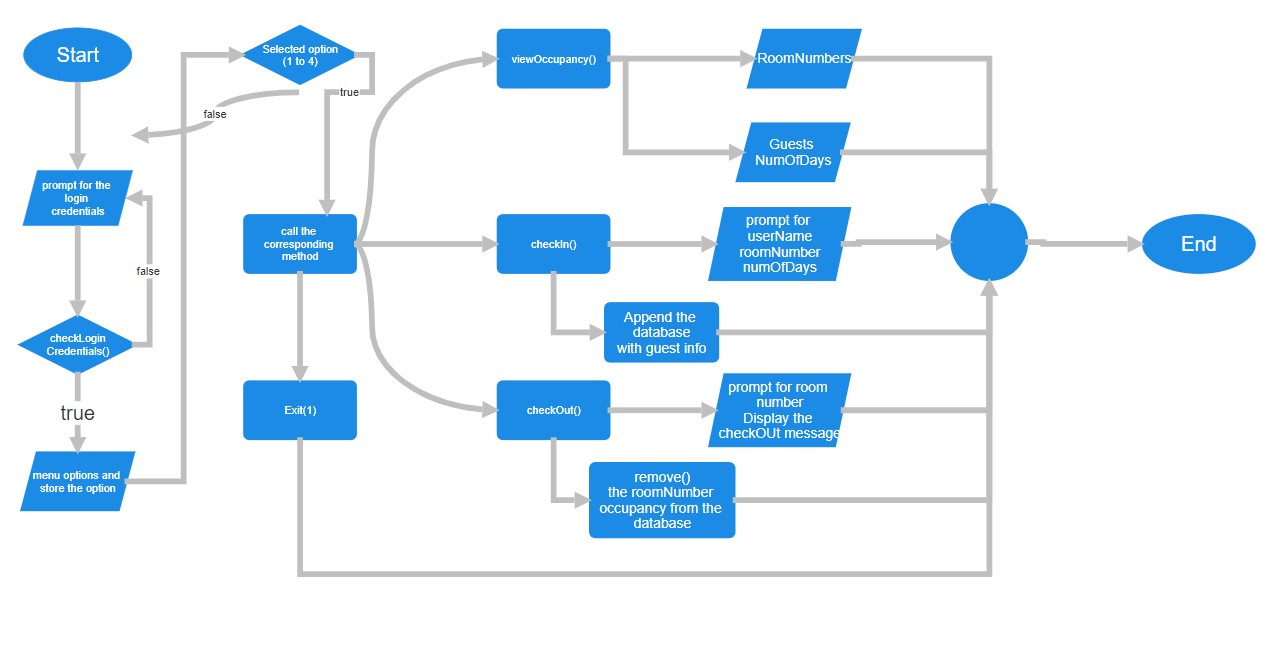
**4. Authentication and Security:**

* + Implement user authentication for admin login.
  + Secure sensitive data, such as login credentials, through hashing and encryption.

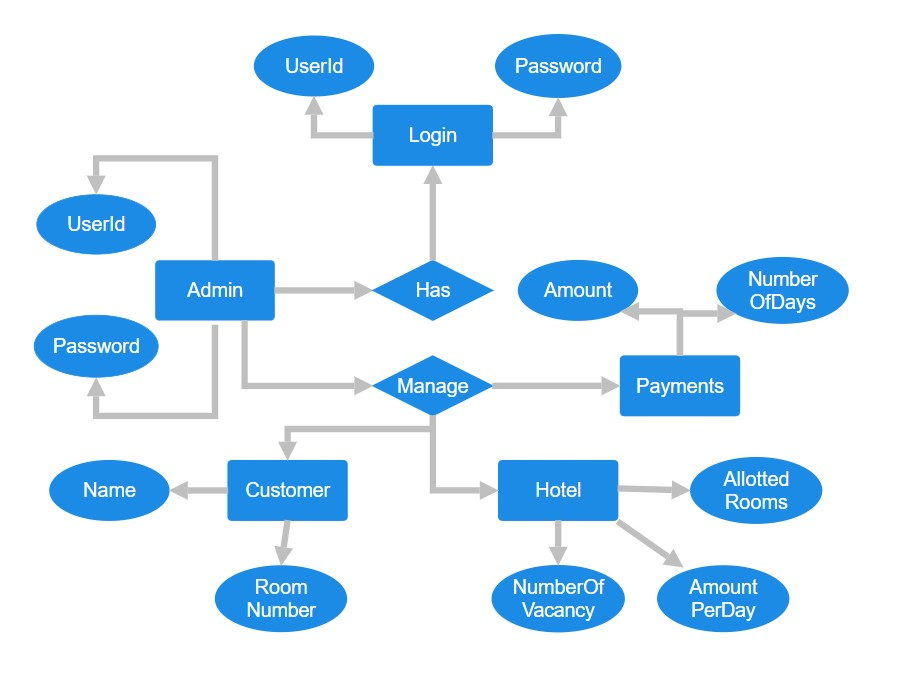
**5. Room Management:**

* + Enhance the room management module to handle a wider array of room-related operations, such as room type management, pricing, and maintenance schedules.

**3.1 Flow Chart and ER Diagram**

****

**Fig. 3.1 Flow Chart**



**Fig. 3.2 ER Diagram**

**Chapter 4: Implementation**

**4.1 Module Description**

**1. Admin Login Module:**

* + Description: The Admin Login module is responsible for authenticating admin-istrators who have access to the Hotel Management System (HMS). It ensures that only authorized personnel can use the system by verifying their login cre-dentials.
  + Key Functions:
  + Prompt the admin to enter a username and password.
  + Validate the provided credentials against predefined admin credentials (username: "suhas," password: "1234").
  + Grant access to the system if the provided credentials match the prede-fined ones.
  + Handle login failures and provide feedback to the admin.
  + Security:
  + The module ensures that access to the HMS is restricted to authorized users by implementing a username and password-based authentication process.

**2. Room Management Module:**

* + Description: The Room Management module handles the core operations relat-ed to hotel rooms. It allows administrators to view the status of rooms, assign rooms to guests during check-in, and mark rooms as vacant during check-out.
  + Key Functions:
  + Display the current status of all rooms in the hotel.
  + Enable administrators to allocate rooms to guests during check-in, pro-vided the room is vacant.
  + Update the room status from "Empty" to "Occupied" upon check-in.
  + Change the room status from "Occupied" to "Empty" during the check-out process.
  + Data Handling:
  + The module updates the rooms array in memory to reflect room occu-pancy status.
  + In a more advanced system, it would update the database with real-time room occupancy information.

**3. Database Interaction Module:**

* + Description: The Database Interaction module manages the connection to the MySQL database and handles interactions related to guest data, room allocation, and occupancy tracking.
  + Key Functions:
  + Establish a connection to the MySQL database.
  + Execute SQL queries to retrieve data about room occupancy.
  + Insert new records into the database when guests check in.
  + Delete records from the database when guests check out.
  + Data Management:
  + This module is responsible for maintaining the consistency of the data between the in-memory rooms array and the MySQL database.
  + Error Handling:
  + The module includes error handling to manage exceptions that may oc-cur during database interactions.

**4. 1Check-In and Check-Out Modules (Combined):**

* + Description: The Check-In and Check-Out modules handle the processes of guest check-in and check-out. These modules are closely related as they both in-teract with the Room Management and Database Interaction modules.
  + Key Functions:
  + Check-In:
  + Collect guest details, including the room number, guest name, and number of days.
  + Validate the room number, ensure the room is vacant, and store guest data.
  + Calculate the bill amount based on the number of days.
  + Update the room status to "Occupied" and add the guest record to the database.
  + Check-Out:
  + Prompt the admin to enter the room number of the departing guest.
  + Validate the room number, ensure the room is occupied, and re-trieve guest data.
  + Calculate the bill amount based on the number of days stayed.
  + Display the total bill amount and delete the guest record from the database.
  + Update the room status to "Empty."

**4.2 Methods and Process**

1. **Admin Login Process:**

* + Method: adminLogin()
  + Description: This method handles the process of authenticating administrators by prompting them to enter a username and password. It verifies the provided credentials against predefined admin credentials and grants access to the system if the login is successful. If the login fails, it prompts the admin to retry.

**2. Initialization Process:**

* + Method: initializeHotel()
  + Description: This method initializes the rooms array, representing the hotel's rooms. It populates the array with room numbers and sets their initial status to "Empty."

**3. Database Connection Process:**

* + Methods:
  + establishDatabaseConnection()
  + closeDatabaseConnection()
  + Description: These methods manage the connection to the MySQL database. establishDatabaseConnection() establishes the connection, while closeData-baseConnection() closes it. Proper error handling is essential to manage poten-tial database connection issues.

**4. Check-In Process:**

* + Method: checkIn()
  + Description: This process allows administrators to check in guests. It prompts the admin to enter the room number, validates the room's availability, collects guest information, calculates the bill amount based on the number of days, up-dates the room status to "Occupied," and inserts the guest's record into the data-base.

**5. Check-Out Process:**

* + Method: checkOut()
  + Description: This process handles guest check-out. It prompts the admin to en-ter the room number of the departing guest, validates that the room is occupied, calculates the bill amount, displays the total bill amount, deletes the guest's rec-ord from the database, and updates the room status to "Empty."

**6. View Occupancy Process:**

* + Method: viewOccupancy()
  + Description: This process displays a list of occupied rooms along with guest names, the number of days they've stayed, and the total bill amount for each oc-cupied room. It retrieves this information from the database.

**7. Guest Data Handling**:

* + Methods: Various database interaction methods
  + Description: The code manages guest data by interacting with the MySQL da-tabase. It includes processes for inserting, updating, and retrieving guest infor-mation based on room occupancy and check-in/check-out actions.

**8. Input Validation**:

* + Methods: getValidNumberOfDays()
  + Description: The code includes input validation to ensure that the admin's in-puts are valid. The getValidNumberOfDays() method, for example, ensures that the number of days provided for the guest's stay is a positive integer.

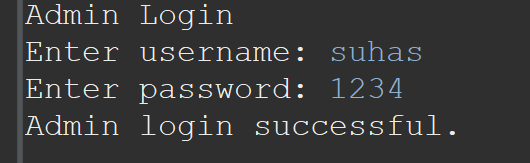
**9. Main Loop:**

* + Description: The code runs within a loop while the admin is logged in, allowing them to perform multiple operations such as check-in, check-out, and viewing occupancy. The loop continues until the admin chooses to exit the system.

**Chapter 5: Results**

**1.Admin Login:**

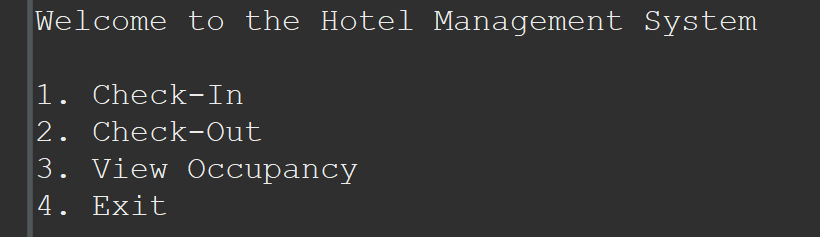
* The program starts with an admin login screen, prompting you to enter a username and password.
* If you enter the correct credentials (username: "suhas" and password: "1234"), you'll see "Admin login successful," and you'll gain access to the system. If the credentials are incorrect, you'll receive an "Invalid username or password. Ac-cess denied" message.



**Fig. 5.1 Admin Login**

**2.Main Menu:**

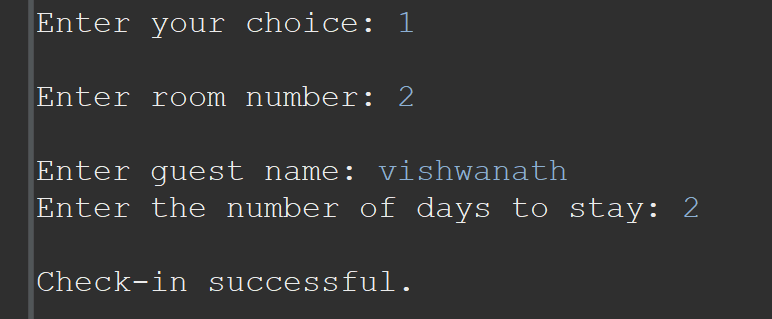
* + **After successful login, the program displays a main menu with options:**
* **2.1 Check-In**
* **2.2 Check-Out**
* **2.3 View Occupancy**
* **2.4 Exit**

****

**Fig. 5.2 Main Menu**

**Check-In:**

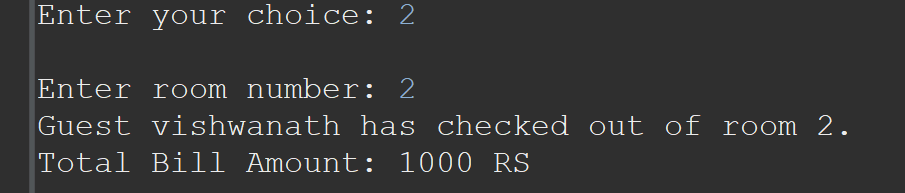
* + If you choose "Check-In," the program prompts you to enter the room number.
  + Then, you enter the guest's name and the number of days they will stay.
  + The program will calculate the bill amount and display a "Check-in successful" message.
  + The room status will be updated from "Empty" to "Occupied.

****

**Fig. 5.3 Check-In Window**

**Check-Out:**

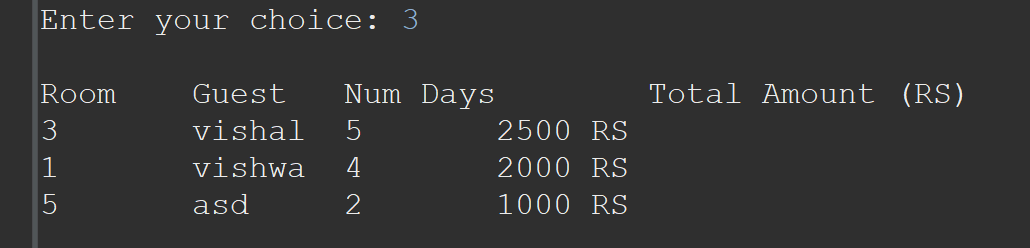
* + If you choose "Check-Out," the program prompts you to enter the room number.
  + It will calculate the bill amount for the guest based on the number of days they stayed and display the total bill amount.
  + The room status will be updated from "Occupied" to "Empty."
  + The guest's record will be deleted from the database.

****

**Fig. 5.4 Check-Out Window**

**View Occupancy:**

* + If you select "View Occupancy," the program displays a list of occupied rooms along with guest names, the number of days they've stayed, and the total bill amount for each occupied room

****

**Fig. 5.5 View Occupancy**

**Exiting the Program:**

* + If you choose "Exit," the program will close the database connection and termi-nate.

**Chapter 6: Conclusion**

The program provides a straightforward and text-based interface for managing room occupancy, catering primarily to administrators. It does not cover more advanced HMS features like reservations, billing, reporting, or extensive user management.

It integrates with a MySQL database, allowing for the storage and retrieval of guest information and room occupancy data. This aspect is essential for data persistence and management.

Future Enhancements:

• Graphical User Interface (GUI): Develop a user-friendly GUI that enhances the system's usability and provides an intuitive interface for staff and administrators.

• Reservations: Implement a reservations system to allow guests to book rooms in advance, manage reservations, and prevent overbooking.

• Billing and Invoicing: Extend the program to include billing and invoicing capabilities, generating invoices for guests upon check-out, and handling various billing scenarios.

• Reporting and Analytics: Create a reporting module for tracking hotel performance, including metrics like occupancy rates, revenue summaries, and guest demographics.

• User Management: Enhance the system to manage different user roles, such as administrators, front desk staff, and management, each with specific permissions and access rights.

• Security: Implement robust security measures, including data encryption, access control, and audit trails to protect guest data and ensure compliance with data privacy regulations.