PRACTICAL 1

- **Select and Write down the problem statement for a real time system of relevance**

Problem statement:

Hotel Management System

This study will support and manage real time Hotel Management system of ITC Mayur that will automate and optimize the operations, services, and transactions at the hotel. The computer technology students(BCA) will be able to develop a system programing using Python or Java software. This manual system is inefficient, prone to errors, and leads to delays, impacting both customer experience and operational efficiency.

The new system is automatic and integrate various tasks, such as room reservations, check-in/check-out procedures, billing, housekeeping, and service management, while providing real-time information for staff and management.

Inefficient Reservation Management:

The current manual system makes it difficult to track room availability and manage bookings in an organized manner. This often leads to overbookings or unavailable rooms. The system will automate the reservation process, track real-time room availability, and enable guests to make reservations online or through staff, reducing errors and improving guest satisfaction.

Manual Check-in/Check-out Process:

The current manual check-in/check-out procedure is time-consuming, leading to delays, guest dissatisfaction, and increased administrative workload. The system will automate guest check-ins and check-outs, updating room status and billing information in real-time, significantly reducing wait times and improving operational efficiency.

Problems:

1. **Billing and Payment Challenges:**
   - Billing is handled manually, often leading to mistakes in charges and delayed payments. The current system struggles to track additional services, such as meals, spa visits, and other amenities.
   - The system will automate the billing process, track all additional charges (room charges, food, services), and handle payments through multiple methods (credit/debit card, cash, online payment), improving accuracy and customer convenience.
2. **Lack of Real-time Updates:**
   - Without a real-time system, it is difficult for staff to track room availability, guest status, and service requests, leading to confusion and delays in fulfilling guest needs.

   - The new system will provide real-time updates on room availability, guest check-ins/outs, and service requests, ensuring that staff are always informed and can act promptly.
3. **Inefficient Resource Management (Housekeeping, Inventory, Services):**
   - The hotel's housekeeping, inventory management, and service delivery are currently managed manually, which often leads to mismanagement of resources and delays in fulfilling guest requests.
   - The system will automate housekeeping schedules, track inventory (e.g., linens, toiletries), and manage service requests (e.g., room service, laundry), ensuring resources are efficiently allocated and guests are satisfied with timely services.
4. **Limited Reporting and Analysis:**
   - The manual system makes it difficult to generate reports on hotel occupancy, revenue, guest preferences, and overall performance. This lack of data limits the hotel's ability to make informed decisions.
   - The system will provide detailed reporting tools for management, generating insights on occupancy rates, revenue, guest demographics, and service usage. This will help the hotel optimize operations and plan for future improvements.
5. **Lack of Integration with External Systems:**
   - The current system does not integrate with online booking platforms, payment gateways, or the hotel's accounting system, leading to discrepancies and inefficiencies.
   - The system will integrate with external systems such as online booking platforms (e.g., Booking.com, Expedia), payment gateways, and accounting software, allowing for seamless transactions and accurate data management across platforms.

Objectives of the Solution:

- **Automate key hotel operations** such as room reservations, guest check-ins/outs, billing, housekeeping, and inventory management.
- **Provide real-time updates** on room availability, guest status, and service requests, ensuring accurate information for staff at all times.
- **Streamline administrative processes** and reduce errors through automation of guest services, room assignments, and billing.
- **Enhance guest experience** by reducing wait times during check-in/check-out, offering quick and accurate billing, and ensuring timely service delivery.
- **Generate detailed reports and analytics** to monitor hotel performance, guest trends, and resource allocation, helping the hotel make informed business decisions.
- **Integrate with external systems** for seamless booking, payment processing, and accounting, ensuring a smooth operation from guest booking to post-stay payment.

PRACTICAL : 2

➢ **To perform the user's view analysis for the suggested system: use case diagram.**

AIM: -To perform the user 's view analysis for the suggested system:

Introduction to use case diagram:

Actor

Someone interacts with use case (system function).

Named by noun.

Actor plays a role in the business,

Similar to the concept of user, but a user can play different

roles

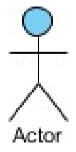
For example:

A prof. can be instructor and also researcher

plays 2 roles with two systems

Actor triggers use case(s).

Actor has a responsibility toward the system (inputs), and Actor has expectations from the system (outputs).



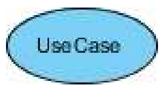
Use Case

System function (process - automated or manual)

Named by verb + Noun (or Noun Phrase).

i.e. Do something

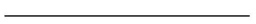
Each Actor must be linked to a use case, while some use cases may not be linked to actors.



Communication Link

The participation of an actor in a use case is shown by connecting an actor to a use case by a solid link.

Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages.



Boundary of system

The system boundary is potentially the entire system as defined in the requirements

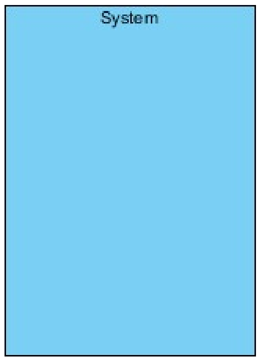
document. For large and complex systems, each module may be the system boundary.

For example, for an ERP system for an organization, each of the modules such as personnel, payroll, accounting, etc.

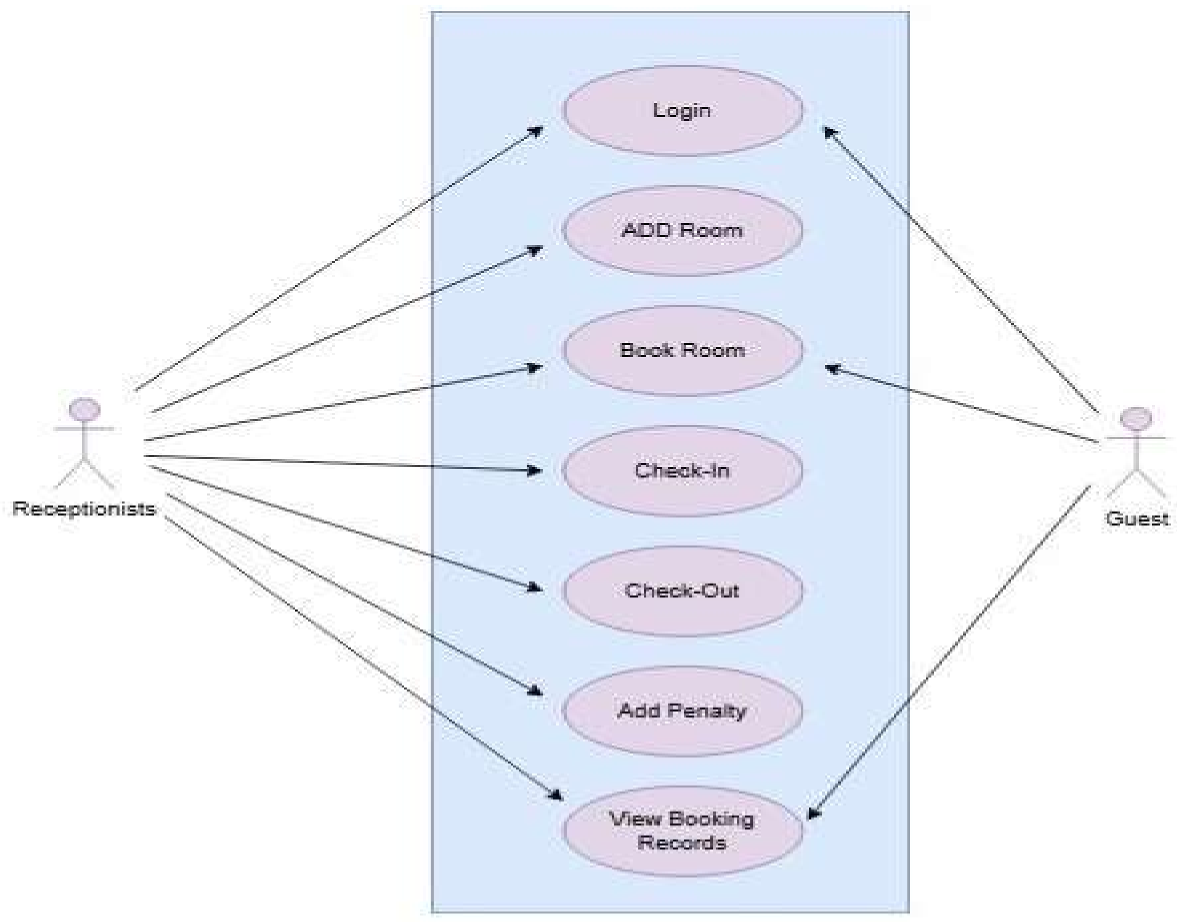
can form a system boundary for use cases specific to each of these business

functions. The entire system can span all of these modules depicting the overall

system boundary.



Use case diagram for hotel management system



<u>**DESCRIPTION OF USE-CASE DIAGRAM:**</u>

1. Login

Description:
This use case describes the process of logging into the hotel management system for customers (Guests) and hotel staff (Receptionists, Admin).

Actors:

- Guest
- Receptionist
- Administrator

Basic Flow:

1. The user navigates to the login page and enters their credentials (username and password).
2. The system verifies the user's credentials and checks their user type (Guest, Receptionist, or Admin).
3. The system redirects the user to their respective dashboard.

Alternative Flow:

- If the user enters an incorrect username or password, the system displays an error message, prompting the user to enter valid credentials.
- If the user's account is inactive, the system displays a message stating that the account is inactive and asks the user to contact the administrator.

Precondition:

- The user must have a valid username and password.

Postcondition:

- The user will be redirected to their respective dashboard.

Extension Points:
None.

2. Add Room

Description:
This use case represents the process of adding a new room to the hotel's inventory. Admin and Receptionist staff can input the room's details, such as room type, price, availability, and features, into the system.

Actors:

- Admin
- Receptionist

Preconditions:

- The user must be logged into the hotel management system.

Basic Flow:

1. The user navigates to the "Add Room" section within the system.
2. The user enters the room's details, such as room number, type (e.g., single, double, suite), price, and available features.
3. The system validates the entered information, ensuring mandatory fields are correctly filled.
4. If the information is valid, the system adds the room to the hotel's inventory.
5. The system updates the availability and pricing of the room in the system.

Alternative Flows:

- If the entered information is incomplete or invalid, the system displays an error message, prompting the user to correct the erroneous fields.

Postconditions:

- The new room is successfully added to the hotel's inventory and is available for booking.

3. Book Room

Description:
This use case represents the process of booking a room for a guest. Receptionists or guests can select a room from the available rooms and reserve it for a designated period.

Actors:

- Guest
- Receptionist

Preconditions:

- The user must be logged into the hotel management system.
- The room must be available in the hotel's inventory.

Basic Flow:

1. The user searches for available rooms within the hotel system by selecting check-in and check-out dates.
2. The system checks the availability of rooms for the selected dates.
3. The user selects a room type and the system shows the available options.
4. The user confirms the booking and enters guest details (name, ID, payment info).
5. The system processes the payment (if necessary), assigns a room, and confirms the booking.

Alternative Flows:

- If no rooms are available, the system displays a message notifying the user and offers alternative room options or dates.
- If payment fails, the system displays an error message and prompts the user to provide a valid payment method.

Postconditions:

- The room is successfully booked, and the booking details are recorded in the system.

4. Check-In

Description:
This use case describes the process of checking a guest into the hotel. The receptionist verifies the guest's booking details and assigns the room.

Actors:

- Receptionist

Preconditions:

- The guest must have a valid booking.
- The user must be logged into the hotel management system.

Basic Flow:

1. The receptionist retrieves the guest's booking details from the system.
2. The receptionist verifies the guest's identification and payment.
3. The system assigns the reserved room to the guest.
4. The receptionist provides the guest with a key or access to the room.

Alternative Flows:

- If the guest's booking is invalid or there are issues with the payment, the system displays an error message.
- If the guest's identification is incorrect, the system prompts the receptionist to request the correct details.

Postconditions:

- The guest is successfully checked in, and the system updates the room status to "occupied."

5. Check-Out

Description:
This use case describes the process of checking a guest out of the hotel. The receptionist verifies the guest's details, processes any final payments, and updates the room status.

Actors:

- Receptionist

Preconditions:

- The guest must have completed their stay and is logged into the hotel management system.

Basic Flow:

1. The receptionist retrieves the guest's booking details and verifies check-out.
2. The system generates the final bill, including room charges, services, and taxes.
3. The receptionist processes the payment (if any balance is due).
4. The guest checks out and returns the room key.
5. The system updates the room status to "available."

Alternative Flows:

- If the guest disputes the final bill, the system prompts the receptionist to review the charges.
- If payment is not processed correctly, the system will flag the error and prompt for re-processing.

Postconditions:

- The guest is successfully checked out, and the room status is updated to "available."

6. Add Penalty (Late Check-Out, Damaged Items)

Description:
This use case describes the process of applying penalties or fines to a guest's account for reasons such as late check-out or damaged hotel property.

Actors:

- Receptionist
- Admin

Preconditions:

- The user must be logged into the hotel management system.
- The guest must be checking out late or causing damage.

Basic Flow:

1. The receptionist identifies the reason for the penalty (e.g., late check-out, damages).
2. The receptionist applies the penalty to the guest's account and records the fine amount.
3. The system updates the guest's bill with the penalty charges.
4. The system may send a notification to the guest regarding the penalty.

Alternative Flows:

- If the penalty cannot be applied (e.g., no damage was found), the system displays a message indicating no charges are necessary.

Postconditions:

- The penalty is successfully applied to the guest's bill.

7. View Booking Records

Description:
This use case allows guests or staff to view the booking history, current bookings, and payment records.

Actors:

- Guest
- Receptionist
- Admin

Preconditions:

- The user must be logged into the hotel management system.

Basic Flow:

1. The user navigates to the "Booking History" section of the system.
2. The system retrieves the relevant records based on the user's query (current bookings, past stays, payments).
3. The system presents the data to the user in a readable format, such as a table or report.

Alternative Flows:

- If there are no records, the system displays a message indicating no booking history is available.

Postconditions:

- The user successfully views their booking or payment history.

PRACTICAL 3

- **To create the function-oriented diagram: Data Flow Diagram (DFD)**

THEORY:

Introduction to Data Flow diagram (DFD):

External entity: an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system.
They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as “Submit payment.”

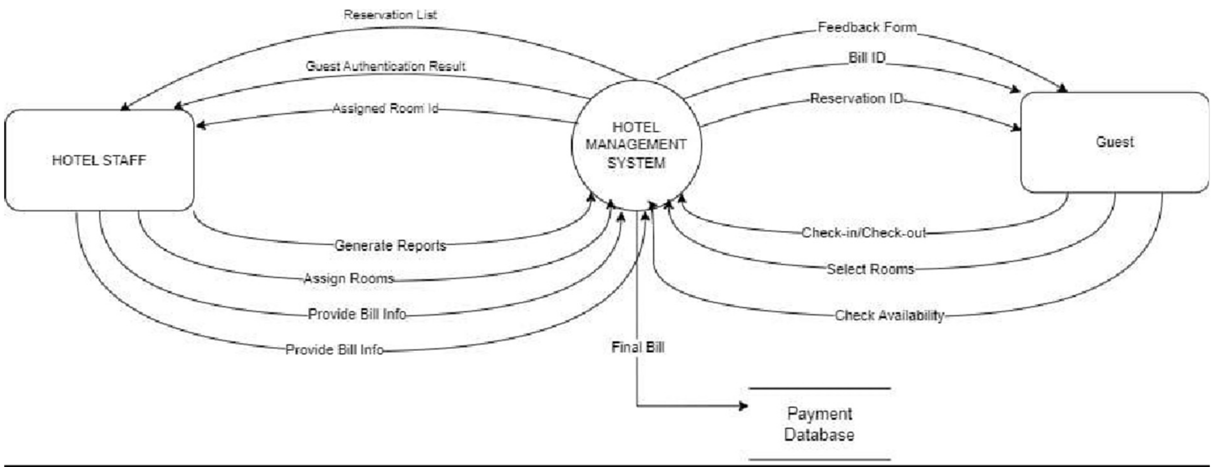
Data store: files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders.”

Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

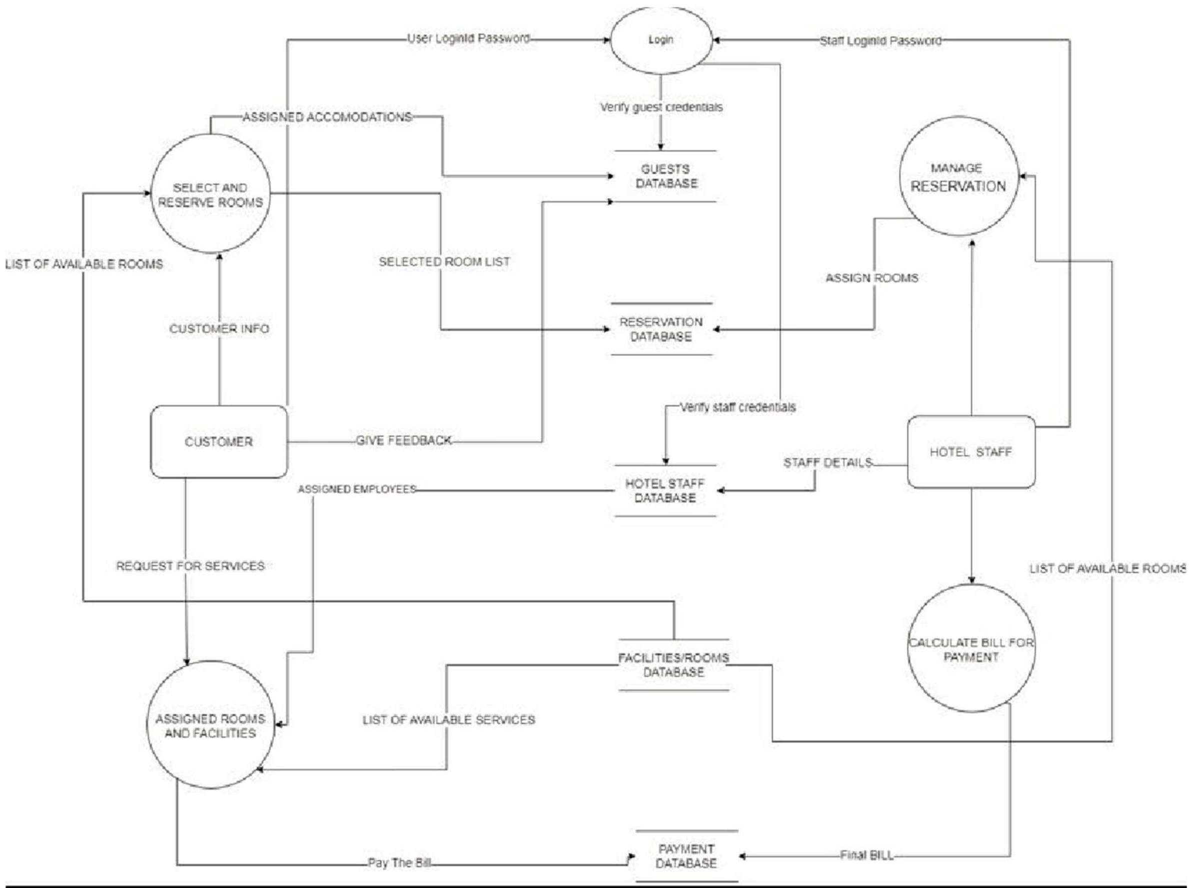
External Entity	▭
Process	○
Data Store	═
Data Flow	→

Data Flow Diagram

Level 0 (Conceptual level)



DFD Level 1

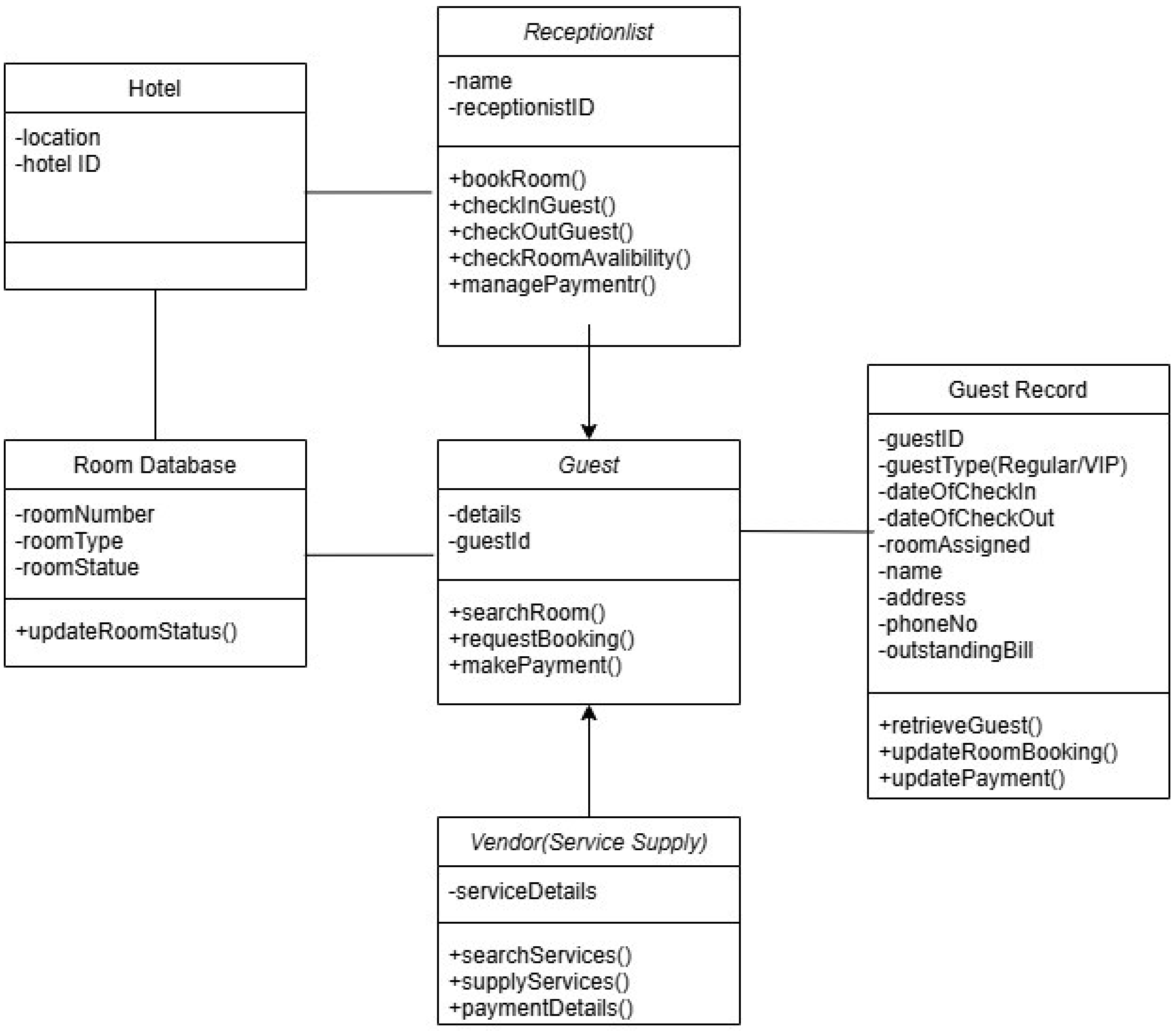


PRACTICAL 4

- **To draw the structural view diagram for the system: Class diagram**

Class Diagram:
A class diagram is a type of static structure diagram in UML (Unified Modeling Language) that represents the structure and relationships of classes, interfaces, and their associations in a system. It provides an overview of the classes and their attributes, methods, and associations in the system.

Components of a Class Diagram:
Class: A class represents a blueprint or template for creating objects.
Attributes: Attributes represent the characteristics or properties of a class.
Methods: Methods represent the behaviors or operations that a class can perform.
Associations: Associations represent the relationships between classes.
Multiplicity: Multiplicity defines the number of instances or objects that participate in an association.
Inheritance: Inheritance represents the "is-a" relationship between classes.
Interfaces: Interfaces define a contract or set of methods that a class must implement.
Dependency: Dependency represents a relationship where one class depends on another class.
Aggregation and Composition: Aggregation and composition represent the whole-part relationships between classes.
Stereotypes and Notes: Stereotypes and notes provide additional information or annotations to enhance the understanding of the class diagram.


Receptionlist
-name
-receptionistID
+bookRoom()
+checkInGuest()
+checkOutGuest()
+checkRoomAvalibility()
+managePaymentr()
Hotel
-location
-hotel ID
Guest Record
-guestID
-guestType(Regular/VIP)
-dateOfCheckIn
-dateOfCheckOut
-roomAssigned
-name
-address
-phoneNo
-outstandingBill
+retrieveGuest()
+updateRoomBooking()
+updatePayment()
Room Database
-roomNumber
-roomType
-roomStatue
+updateRoomStatus()
Guest
-details
-guestId
+searchRoom()
+requestBooking()
+makePayment()
Vendor(Service Supply)
-serviceDetails
+searchServices()
+supplyServices()
+paymentDetails()

PRACTICAL 5

- **To perform the behavioral view diagram for the suggested system: Sequence diagram**

SEQUENCE DIAGRAM

A sequence diagram is a type of UML (Unified Modelling Language) diagram that visually represents the sequence of interactions between objects or components in a system. It is used to model the dynamic behaviour of a system, showing how messages are exchanged between objects over time.

Sequence diagrams are widely used in software engineering to:

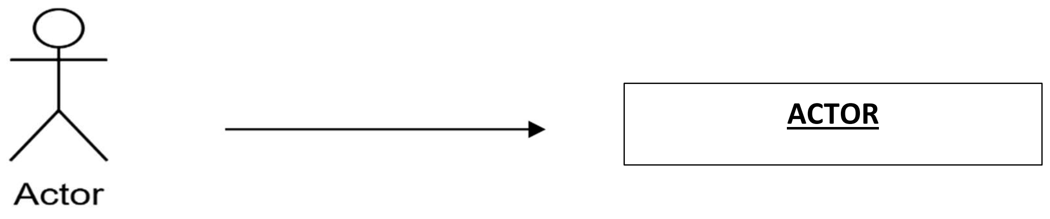
- Understand and document how a system works.
- Model specific use cases or scenarios.
- Provide a clear view of interactions and dependencies between components.

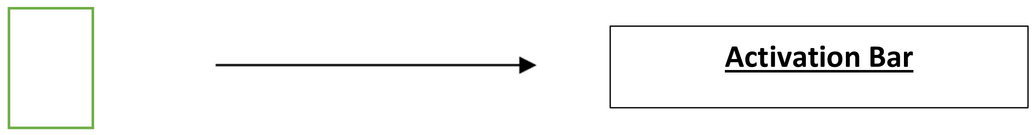
Components of a sequence diagram

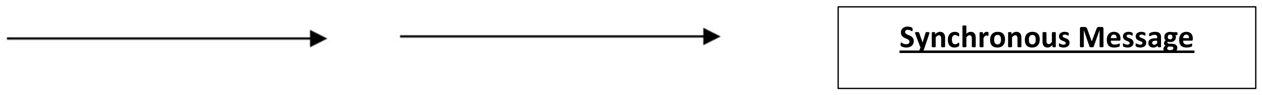
1. **Actors:**
   - Represent users or external systems that interact with the system.
   - Example: A customer, admin, or external API.
2. **Objects or Lifelines:**
   - Represent the various components, classes, or modules of the system.
   - Each lifeline has a vertical dashed line indicating its presence over time.
3. **Messages:**
   - Show communication between objects or actors.
   - Represented as horizontal arrows:
     - Solid Arrow: Represents a synchronous message (e.g., method call).
     - Dashed Arrow: Represents a return message.
4. **Activation Bars:**
   - Represent the time during which an object is actively performing a task or operation.
   - Drawn as vertical rectangles on a lifeline.
5. **Loops and Conditions:**

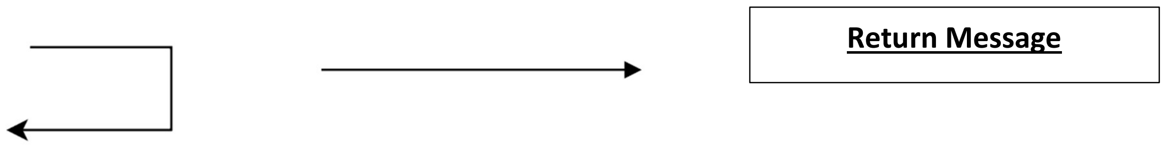
- Represent repetitive or conditional actions.
- Example: A loop to process a list of bookings.

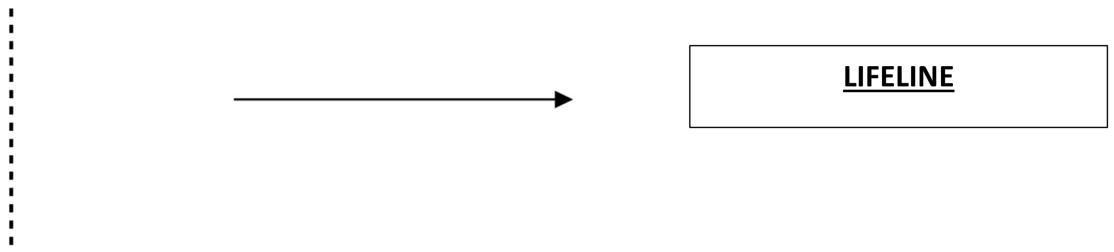
<u>Notations in a Sequence Diagram</u>

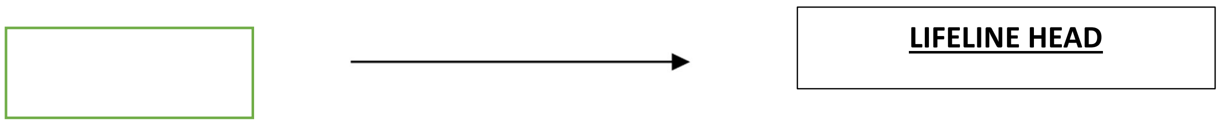






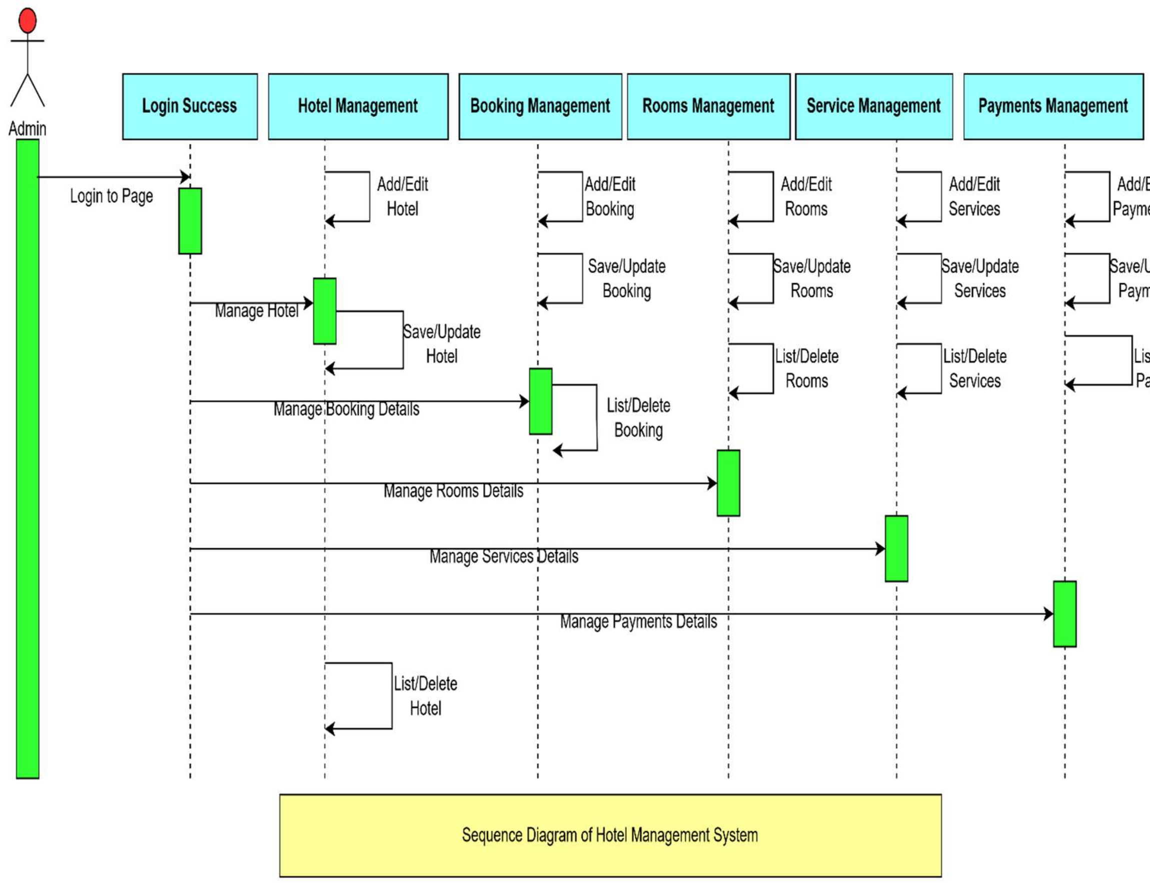






Text	$\longrightarrow$	**TEXT LABLES**

SEQUENCE DIAGRAM OF HOTEL MANAGEMENT SYSTEM



Hotel Management System – Sequence Diagram Description

Introduction

The sequence diagram of the Hotel Management System provides a visual representation of the interactions and processes involved in managing a hotel's operations. It outlines how different actors (e.g., admin) interact with the system to perform tasks such as hotel management, booking management, room management, service management, and payment management.

Actors in the System

Actors:

Admin: The primary actor who interacts with the system to manage various functionalities like hotels, bookings, rooms, services, and payments.

Lifelines:

1. **Login Success:** Handles the login authentication for the admin to access the system.
2. **Hotel Management:** Manages operations related to hotels, such as adding and editing hotel details.
3. **Booking Management:** Handles operations related to customer bookings, such as adding, editing, listing, and deleting bookings.
4. **Rooms Management**: Manages room-related functionalities, including adding, editing, listing, and deleting room details.
5. **Service Management:** Allows the admin to manage services offered by the hotel, including adding, editing, and listing services.
6. **Payments Management:** Manages payment operations, such as adding, editing, listing, and deleting payment details.

Flow of Interaction:

1. **Login Phase:**
   - The admin logs into the system by providing credentials.
   - If the credentials are valid, the system grants login success and gives access to the admin panel.
2. **Hotel Management:**
   - The admin can add or edit details of hotels using the Hotel Management module.
   - Changes are saved or updated in the system.
3. **Booking Management:**
   - Admin can manage booking details, such as adding or editing bookings.
   - The system allows the admin to list or delete existing bookings.
4. **Room Management:**
   - Admin can add or edit details of rooms, such as room types, availability, and rates.
   - The system also provides options to list or delete room information.

5. **Service Management:**
   - Admin can manage the services offered by the hotel, such as housekeeping, dining, and recreational services.
   - The admin can add or edit services and view or delete service details.
6. **Payment Management:**
   - The admin manages payment records, including adding and editing payment details.
   - The system supports listing and deleting payment transactions.

System Overview:

- Each operation follows a request-response mechanism. The admin initiates a request, and the system performs the corresponding action (e.g., add, edit, list, or delete).
- The diagram ensures a logical flow of operations and covers the main modules necessary for a functional Hotel Management System.
- The system emphasizes modularity, allowing the admin to handle each operation independently.

Conclusion

The sequence diagram and the detailed description provides a comprehensive understanding of the Hotel Management System as it ensures clarity and modularity in managing the hotel's operations. It demonstrates how the system processes requests and executes corresponding actions efficiently, making it a vital component of the software engineering documentation.

PRACTICAL 6

➢ **Analyze requirement for a system and develop Software Requirement Specification Sheet (SRS) for suggested system.**

1. Introduction

1.1 Purpose

The purpose of this document is to describe the requirements for the development of the **Hotel Management System (HMS)**. This document outlines the functional and non-functional requirements, system architecture, and constraints necessary to design and implement a hotel management system that streamlines various operations like booking, check-in/check-out, room availability management, and customer interaction.

1.2 Scope

The Hotel Management System is designed to automate the operations of a hotel. It will allow customers to make reservations, check in/out, view room availability, manage bookings, and perform related tasks. Hotel staff will be able to manage room details, view booking history, and process check-in/check-out efficiently.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS:** Software Requirement Specification
- **HMS:** Hotel Management System
- **ID:** Identification Number for guest or booking
- **Admin:** Administrator of the hotel management system
- **Guest:** Customer booking rooms at the hotel

1.4 References

- Use case diagram for the Hotel Management System
- Previous customer feedback reports
- Industry-standard hotel management practices

2. Overall Description

2.1 Product Perspective

The proposed Hotel Management System will manage the entire lifecycle of hotel operations from reservations, guest check-in, room service requests, billing, to guest check-out. The system will provide a seamless interface for both hotel staff and customers, ensuring smooth hotel management and a pleasant customer experience.

2.2 Software Requirements

Frontend:

- **Android Development Tools**
- **Advanced Java**

Backend:

- **MySQL**
- **PHP**

2.3 Hardware Requirements

- **Android Version 2.3 (Gingerbread) or higher**
- **2GB RAM (minimum)** for smooth operation
- **1.2 GHz Processor**
- **Intel i5 Processor**
- **Windows 7/8/8.1/10** Operating System

2.4 Functional Requirements

R.1 User Registration and Login

R.1.1: Sign Up

- **Description:** Guests will sign up for the system by providing their details.
- **Input:** Name, Address, Email, Phone Number.
- **Output:** Confirmation message with Membership ID and Password.
- **Processing:** The system checks for errors in input and if valid, generates a unique Membership ID and password.

R.1.2: Login

- **Input:** Membership ID, Password.
- **Output:** Successful login and access to hotel features.
- **Processing:** Validation of credentials with the stored data in the system.

R.2 Room Management by Guests

R.2.1: View Room Availability

- **Description:** Guests can check the availability of rooms based on dates.
- **Input:** Check-in date, check-out date.
- **Output:** List of available rooms for the specified dates.

R.2.2: Book Room

- **Description:** Guests can book a room for their stay.
- **Input:** Select room type, number of rooms, check-in date, check-out date.
- **Output:** Confirmation message with booking details.
- **Processing:** The system checks room availability, processes booking, and updates room status.

R.2.3: Modify Booking

- **Description:** Guests can modify their booking (e.g., change dates or room type).
- **Input:** Booking ID, new room preferences, new dates.
- **Output:** Confirmation of booking modification.

R.2.4: Cancel Booking

- **Description:** Guests can cancel their booking.
- **Input:** Booking ID, reason for cancellation.
- **Output:** Confirmation of cancellation.

R.3 Check-In and Check-Out

R.3.1: Check-In

- **Input:** Booking ID, guest details (ID proof, payment).
- **Output:** Room allocation and check-in confirmation.
- **Processing:** The system will verify the booking, check payment status, and allow the guest to check in.

R.3.2: Check-Out

- **Input:** Guest ID, room number, checkout details.
- **Output:** Final bill, payment receipt, room availability updated.

- **Processing:** The system will calculate the final bill, including room charges, service charges, and additional amenities, and process payment.

R.4 Room Management by Hotel Staff

R.4.1: Add New Rooms

- **Input:** Room details (room number, type, price, availability).
- **Output:** Confirmation of room addition.
- **Processing:** Admin adds new rooms to the system, updating the room inventory.

R.4.2: Update Room Details

- **Input:** Room number, new details (e.g., price change, room upgrade).
- **Output:** Confirmation of room update.

R.4.3: Remove Rooms

- **Input:** Room number to be removed from the system.
- **Output:** Confirmation of room removal.
- **Processing:** Admin removes rooms from the room inventory.

R.5 Billing and Payment

R.5.1: Generate Bill

- **Description:** The system generates a detailed bill for guests based on their room usage and services availed.
- **Input:** Room number, services used, number of days stayed.
- **Output:** Final bill, including taxes and additional charges.

R.5.2: Process Payment

- **Description:** The system processes payment and generates a receipt.
- **Input:** Payment method (credit card, cash, etc.), bill details.
- **Output:** Payment confirmation and receipt.

2.5 Non-Functional Requirements

- **Usability Requirements:**
  - The system should be easy to use and navigate, with no special training required. Both guests and hotel staff should be able to use the system intuitively.

- **Availability Requirements:**
    - The system should be available 24/7, with minimal downtime for maintenance.
- **Efficiency Requirements:**
    - The system should be able to handle multiple bookings and customer interactions concurrently without performance degradation.
- **Accuracy Requirements:**
    - The system should provide accurate information in real-time, such as room availability and billing information.
- **Performance Requirements:**
    - Response time for booking and check-in/check-out should be within 5 seconds for smooth operations.
- **Reliability Requirements:**
    - The system must be highly reliable, with minimal downtime. Backup mechanisms should be in place for data integrity.

2.6 User Characteristics

There are three types of users in the system:

1. **Guest Module (Customer):**
    - View room availability.
    - Make, modify, and cancel bookings.
    - Check-in/check-out.
    - View billing details and make payments.
2. **Hotel Staff Module:**
    - Manage room details (add, update, remove rooms).
    - View guest details and bookings.
    - Generate bills and process payments.
3. **Administrator Module:**
    - Register guests and staff.
    - Manage room inventory.
    - View all transactions, bookings, and guest details.

2.7 Constraints

- **Data Integrity:** All room and guest data must be accurately recorded, including booking status, check-in/check-out dates, and payment details.
- **Real-Time Availability:** The system must update room availability in real-time to ensure no double-booking occurs.

PRACTICAL 7

- **To draw the behavioral view diagram: State-chart diagram or Activity diagram.**

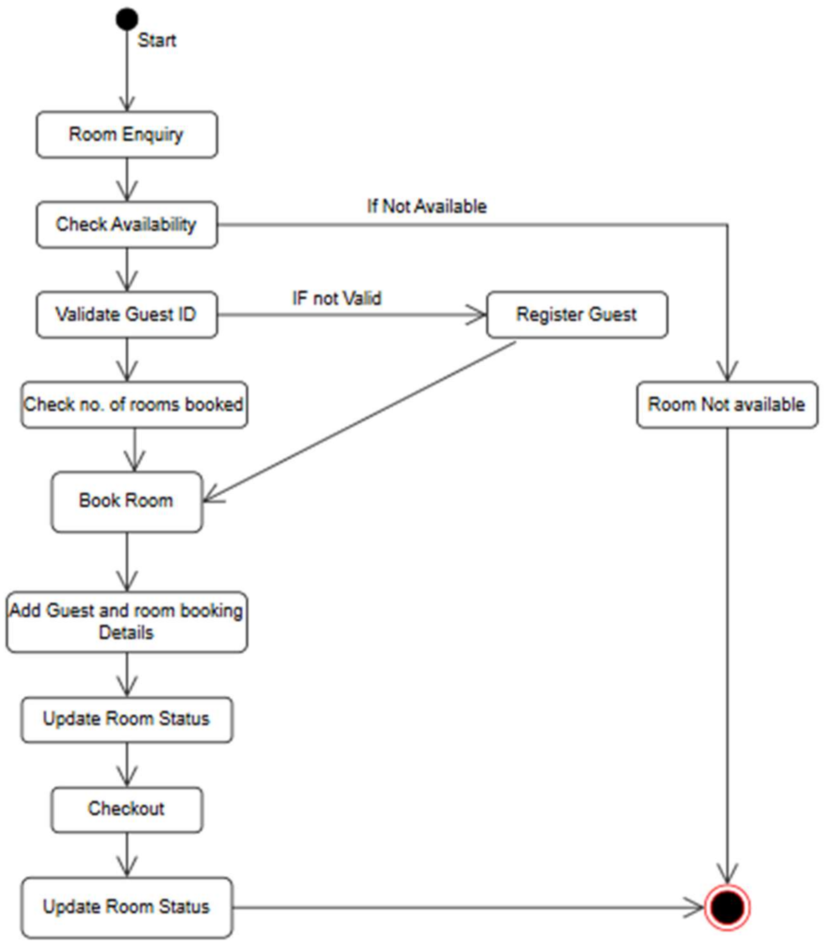
A state diagram, also known as a state machine diagram or state chart diagram, is a type of behavioral diagram in UML (Unified Modeling Language) used to model the behavior of a system or object over time. It represents the various states that an object or system can be in, as well as the transitions between those states based on events and conditions.

The components of a state diagram include:

1. State: A state represents a specific condition or mode of an object or system. It describes the behavior and attributes of the object or system at a particular point in time.
2. Initial State: The initial state indicates the starting point of the object or system when it is first created or initialized.
3. Final State: The final state represents the end point or termination of the object or system's behavior. Once the object or system reaches the final state, it cannot transition to any other state.
4. Transitions: Transitions depict the movement or change of an object or system from one state to another. They are triggered by events or conditions and may have associated actions or guards (conditions that must be met for the transition to occur).
5. Events: Events are stimuli or occurrences that trigger transitions in the state diagram. They can be external events (e.g., user input, system signals) or internal events (e.g., completion of an action).
6. Actions: Actions represent the behavior or activities associated with a state or transition. They can be actions performed upon entering or exiting a state, as well as actions performed during a transition.
7. Guards: Guards (also known as conditions or guards) are conditions that must be true for a transition to occur. They define the constraints or criteria that determine if a transition is allowed to happen.

State diagrams provide a visual representation of the dynamic behavior of a system, illustrating the possible states and transitions that can occur. They are useful for modeling complex systems, specifying the life cycle of an object, and understanding the behavior of a system in response to events and conditions

Diagram:


Start
Room Enquiry
Check Availability
If Not Available
Validate Guest ID
IF not Valid
Register Guest
Check no. of rooms booked
Room Not available
Book Room
Add Guest and room booking Details
Update Room Status
Checkout
Update Room Status