# SOFTWARE ENGINEERING PROJECT

**COMPUTER SCIENCE & ENGINEERING**
**(Artificial Intelligence & Machine Learning)**

Submitted By
**23WH1A6605 Ms. E. ANKITHA**
**23WH1A6616 Ms. K. MAHIMA**
**23WH1A6619 Ms. M. JISHA CHOWDARY**
**23WH1A6620 Ms. G. SOWMYA SRI**
**23WH1A6624 Ms. S. SNEHITHA**
**23WH1A6633 Ms. U. SRI SURYA SAI TEJASWINI**
**23WH1A6640 Ms. B. THANASWI**
**23WH1A6649 Ms. T. SAHITHYA**
**24WH5A6601 Ms. D. SHREE VEDHA**

**under the esteemed guidance of**
**Ms. V. ASHA**
**Assistant Professor CSE (AI & ML)**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**
**(Artificial Intelligence & Machine Learning)**
**BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN**
**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)**
**Accredited by NAAC with A Grade**
**Bachupally, Hyderabad – 500090**

# BVRIT HYDERABAD
# COLLEGE OF ENGINEERING FOR WOMEN

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Accredited by NAAC with A Grade
Bachupally, Hyderabad – 500090

**Department of Computer Science & Engineering**
**(Artificial Intelligence & Machine Learning)**

## CERTIFICATE

This is to certify that the **Software Engineering Project** is a bonafide work carried out by **Ms. E. ANKITHA(23WH1A6605), Ms. K. MAHIMA(23WH1A6616), Ms. M. JISHA CHOWDARY (23WH1A6619), Ms. G. SOWMYA SRI(23WH1A6620), Ms. S. SNEHITHA(23WH1A6624), Ms. U. SRI SURYA SAI TEJASWINI (23WH1A6633),Ms. B. THANASWI (23WH1A6640), Ms. T. SAHITHYA(23WH1A6649),Ms. D. SHREE VEDHA(24WH5A6601)** in partial fulfilment for the award of B.Tech degree **in Computer Science & Engineering (AI & ML), BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad,** affiliated to Jawaharlal Nehru Technological University Hyderabad, under my guidance and supervision. The results embodied in the project work have not been submitted to any other**.**

| | |
|---|---|
| **Internal Guide** | **Head of the Department** |
| **Ms. V .ASHA** | **Dr. B Lakshmi Praveena** |
| **Assistant Professor** | **HOD & Professor** |
| **Dept of CSE(AI&ML)** | **Dept of CSE(AI&ML)** |

# INDEX

# E-TICKECTING

**Problem Statement:** An e-ticketing system is a streamlined platform that enhances ticket entry,processing, and marketing across various sectors, including airlines,railways, and entertainment venues. Electronic tickets, often referred to as "eTickets," can be delivered in digital formats such as PDFs via email or mobile apps. This system allows users to download their tickets, eliminating the need for physical mail. Users present either a printed ticket or a digital version on their mobile devices at the venue.

## Software Requirement Specification:

**Functional requirements:**

1. **User Registration and Authentication**
   **Account Creation :** Users must be able to create accounts by providing essential information such as name, email address, and password. The system will enforce strong password policies to enhance security.
   **Secure Login :** Users shall log in securely using their registered credentials (email and password). The system must utilize encryption to protect user information during transmission.
   **Password Recovery :** The system shall provide a password recovery feature that allows users to reset their passwords through a secure email verification process. Users must receive a link to reset their password within a specified timeframe.

2. **Event Search and Discovery**
   **Event Search Functionality :** Users can search for events based on various criteria such as location, date, and category. The search should be intuitive and responsive, providing relevant results.
   **Filtering and Sorting Options :** Users should have the ability to filter search results by date, price range, popularity, and other relevant attributes. Sorting options should also be available to enhance the user experience.

3. **Ticket Purchase**

**Selection Process :** Users should be able to select from different ticket types (e.g., general admission, VIP) and specify the desired quantity during the purchasing process.
**Secure Payment Processing** : The system must support a variety of payment methods, including credit/debit cards and digital wallets, ensuring a secure transaction process.
**Digital Ticket Confirmation :** Upon successful payment, users will receive a digital ticket via email and/or app notification, including a QR code for event entry. The ticket will contain essential details such as event name, date, time, and venue.

4. **Ticket Management**
   **Access to Tickets :** Users can view their purchased tickets within their account dashboard, allowing them to download or print tickets as needed.
   **Ticket Transfer Options :** Users should have the ability to transfer tickets to others, either via email or through the app, ensuring that all transfers are securely managed.
   **Refund Requests and Support :** Users can request refunds for canceled or rescheduled events. The system should provide a straightforward process for submitting refund requests and obtaining customer support, with timely responses expected.

**Non Functional requirements:**

**Performance :** The system should provide quick responses to user actions and efficiently handle a high volume of transactions. Performance metrics should ensure that users experience minimal delays during ticket searches and purchases, even during peak traffic.
**Scalability :** The architecture of the system must support scalability, allowing it to accommodate a growing number of users and transactions without compromising performance. This includes the ability to add resources as demand increases, ensuring consistent service quality.
**Availability :** High availability is crucial for an e-ticketing system, particularly during major events. The system should be designed to minimize downtime and provide users with reliable access to ticket purchasing at all times. Additionally,

it should include robust backup and recovery mechanisms to ensure data integrity and quick restoration in case of failures.

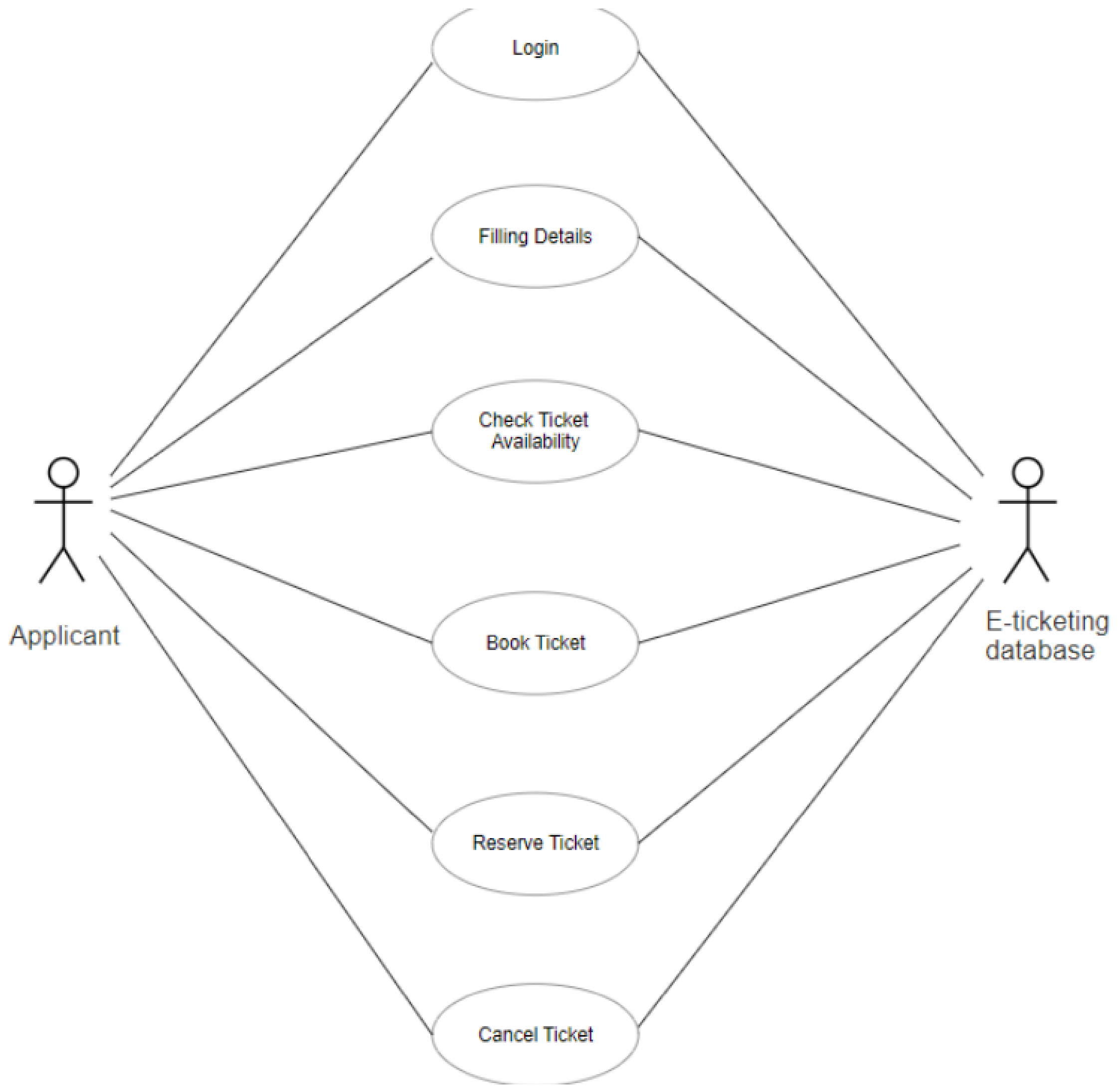
**Security :** Security measures are essential to protect user data and transaction details. The system should implement strong authentication methods, secure data transmission, and encryption to safeguard sensitive information. Regular security audits and compliance with industry standards will further enhance the system's resilience against threats.

**Usability :** A user-friendly interface is vital for encouraging ticket purchases and enhancing user satisfaction. The system should be intuitive, with clear navigation and minimal steps required to complete transactions. Attention to accessibility standards will ensure that all users, including those with disabilities, can effectively use the system.

## UML Diagrams

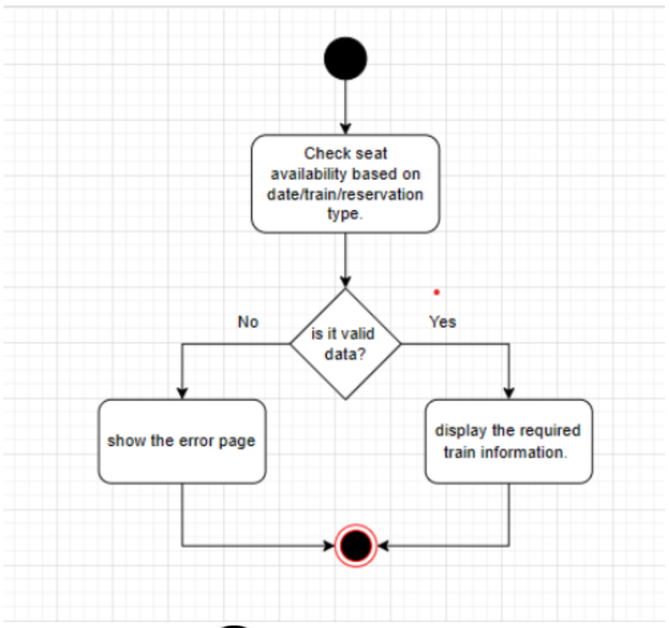
### Use Case Diagram:

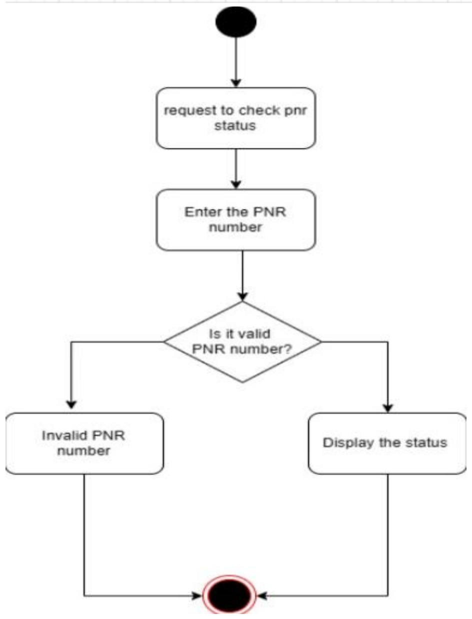
A use case diagram is a visual representation that means the different interactions between the user, or actors, and a system that actually describes the system's functional requirements. It, therefore, indicates all the use cases, or specific functionalities or processes, which the system provides and how various different actors are going to interact with those use cases

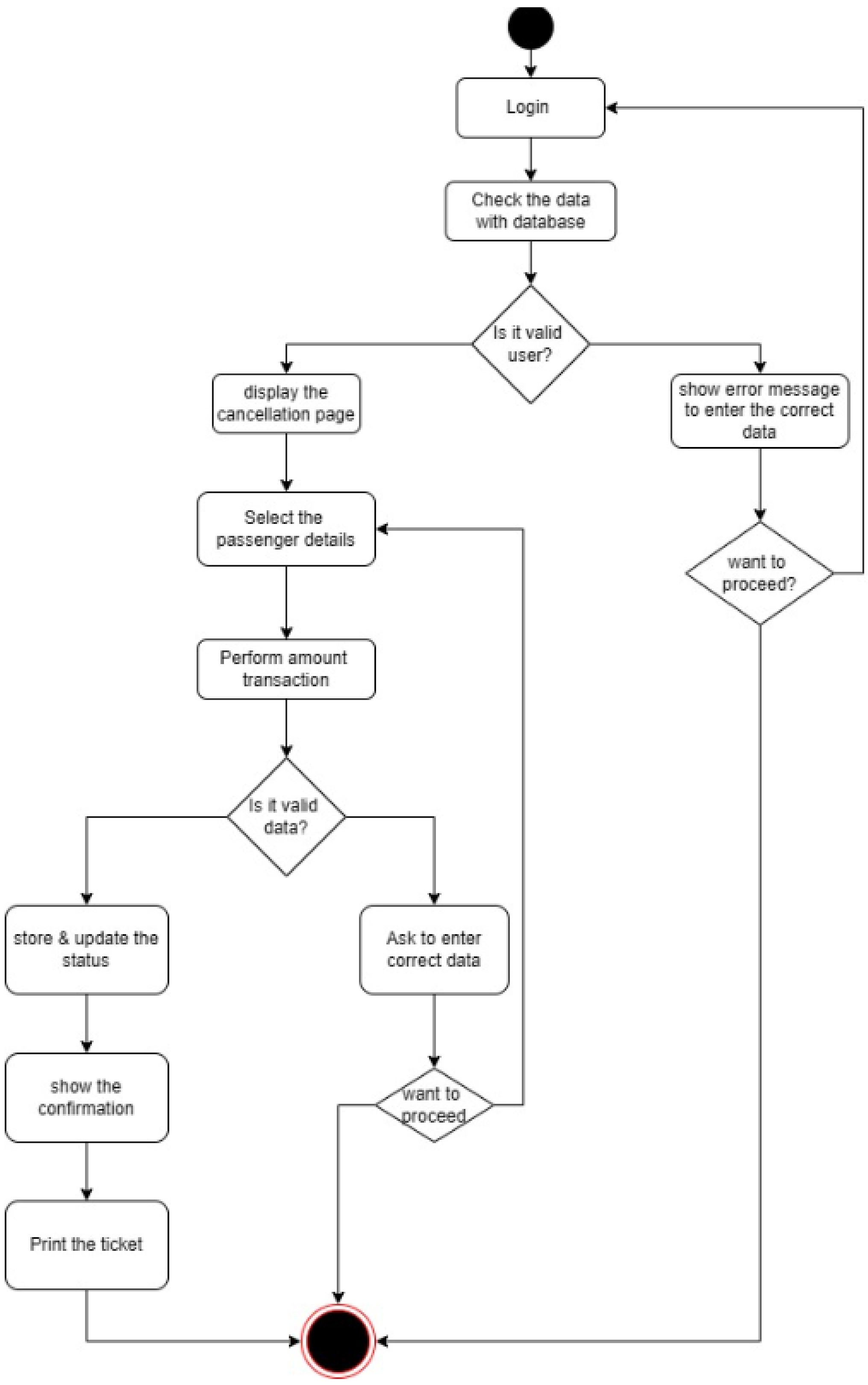


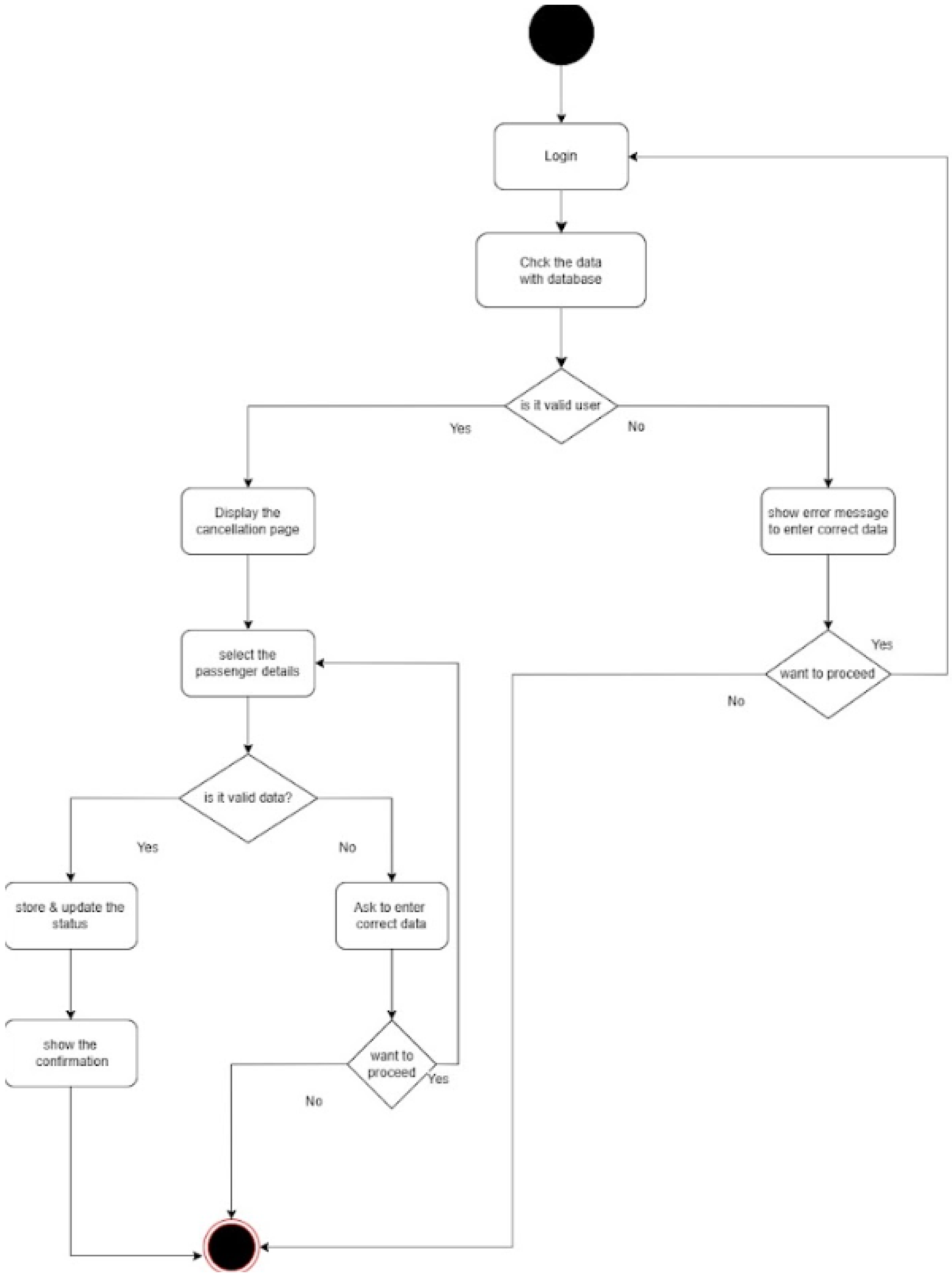
**Activity Diagram:**

An activity diagram is a UML diagram that depicts the flow of activities within a process, showing actions, decisions, and parallel workflows. It uses rounded rectangles for activities, arrows for transitions, and diamonds for decision points. Start and end nodes mark the process boundaries, while swimlanes can indicate different responsibilities. This diagram is valuable for visualizing complex workflows and system behavior.


Check seat availability based on date/train/reservation type.
No
is it valid data?
Yes
show the error page
display the required train information.

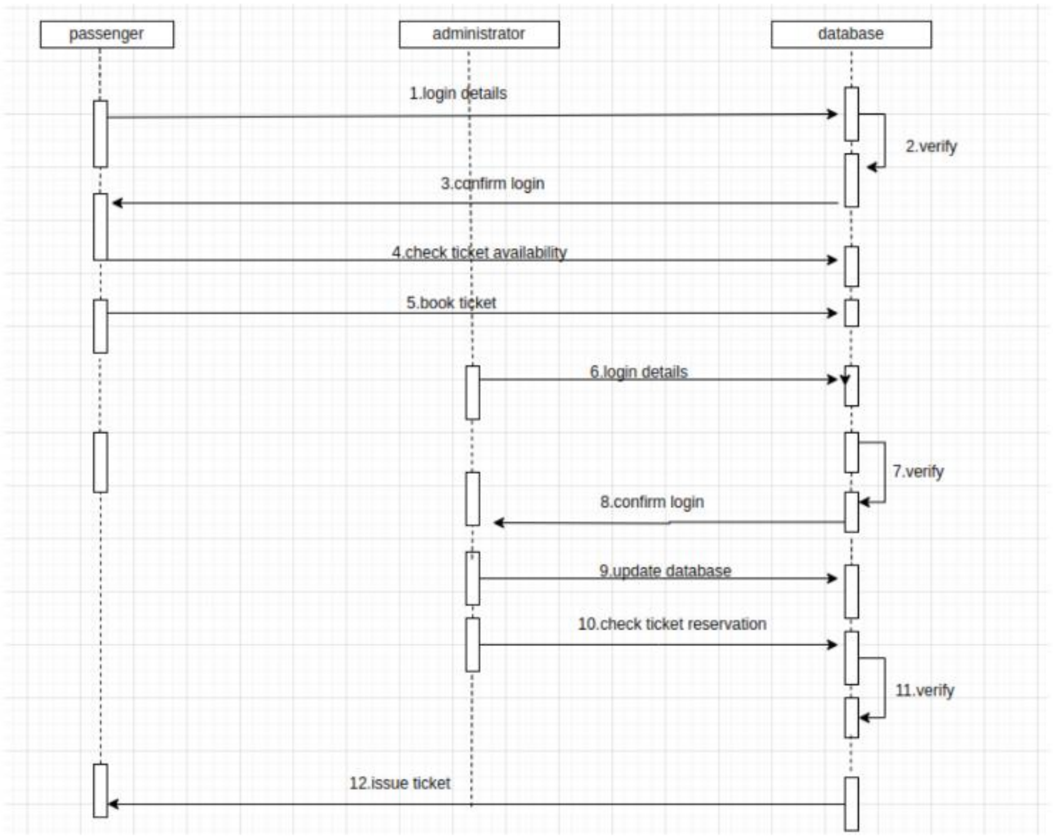

request to check pnr status
Enter the PNR number
Is it valid PNR number?
Invalid PNR number
Display the status


Login
Check the data with database
Is it valid user?
display the cancellation page
show error message to enter the correct data
Select the passenger details
want to proceed?
Perform amount transaction
Is it valid data?
store & update the status
Ask to enter correct data
show the confirmation
want to proceed
Print the ticket


Login
Chck the data with database
is it valid user
Yes
No
Display the cancellation page
show error message to enter correct data
select the passenger details
want to proceed
Yes
No
is it valid data?
Yes
No
store & update the status
Ask to enter correct data
show the confirmation
want to proceed
Yes
No

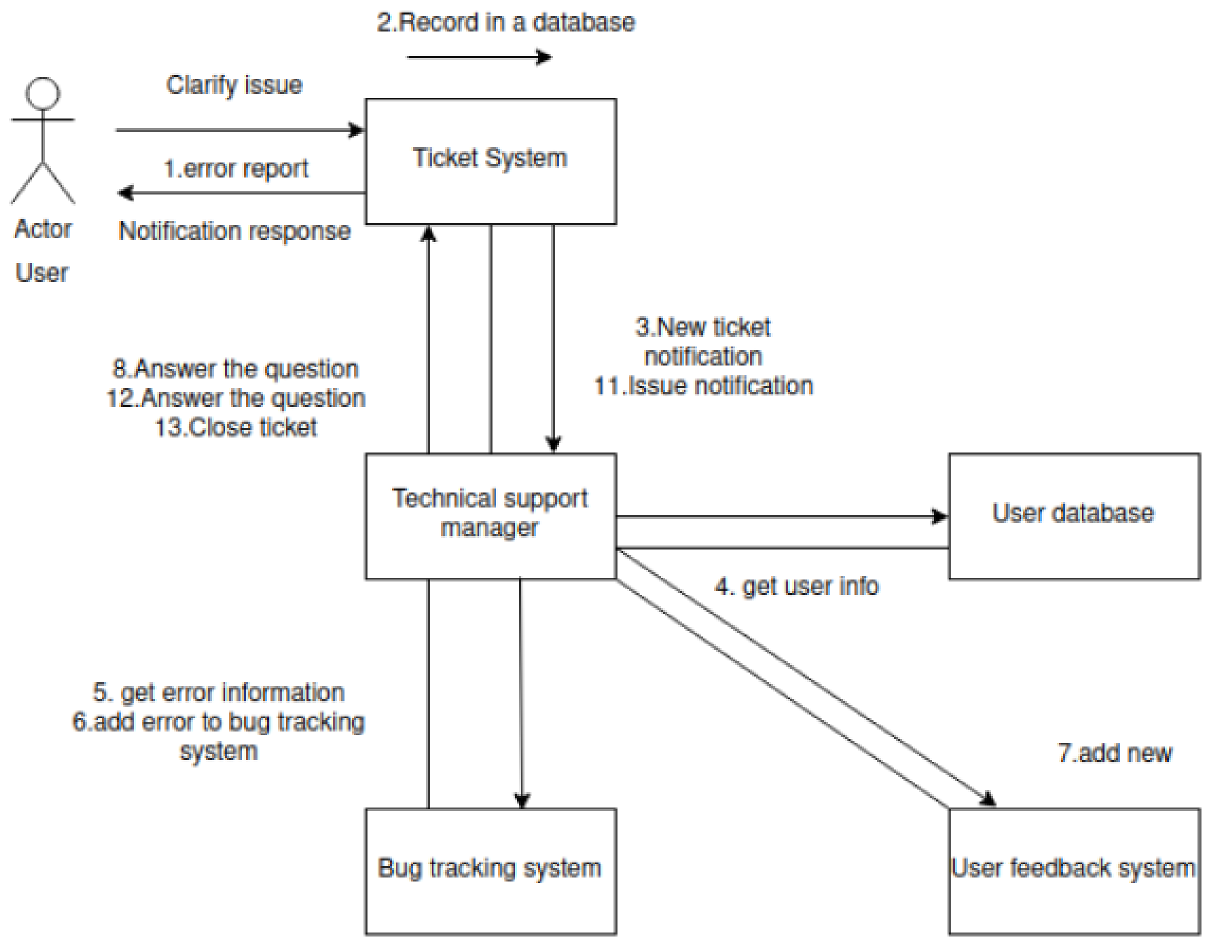
**Sequence Diagram:**

A sequence diagram is a type of UML diagram that illustrates how objects interact in a specific scenario over time. It shows the sequence of messages exchanged between objects, represented as vertical lines, with horizontal arrows indicating the messages or calls. The diagram captures the order of interactions, highlighting timing and the relationships between different components in a process. Sequence diagrams are useful for modeling dynamic behavior, clarifying system operations, and understanding the flow of control in complex systems.



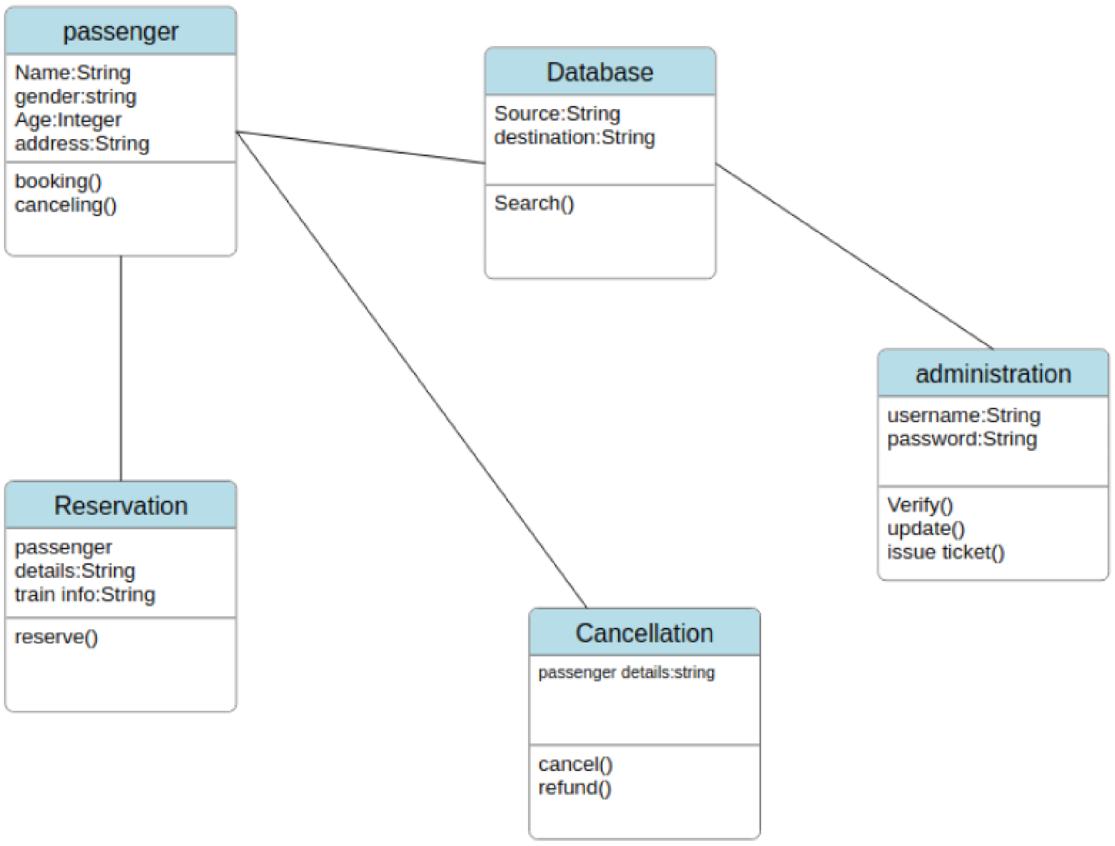
**Collaboration Diagram:**

A collaboration diagram, or communication diagram, illustrates the interactions between objects in a system, emphasizing their relationships. Objects are represented as circles, while messages are shown as labeled arrows connecting them. The diagram uses numbered arrows to indicate the sequence of message exchanges. It is useful for visualizing how objects work together to complete tasks within a process.



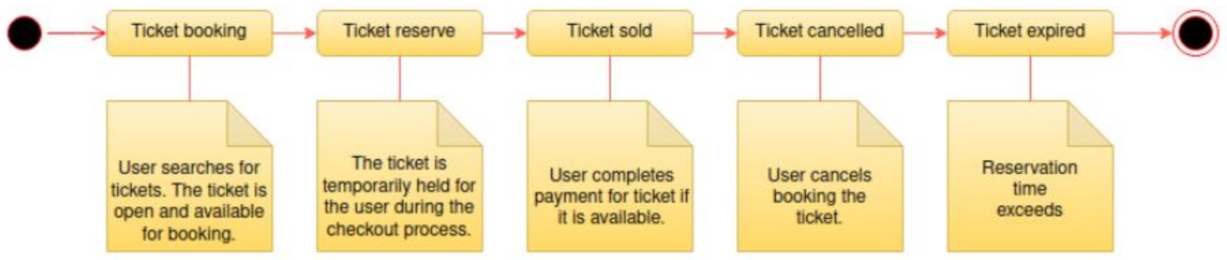
**Class Diagram:**

A class diagram is a UML diagram that depicts the static structure of a system, showing its classes, attributes, methods, and relationships. Classes are represented as rectangles divided into sections for the name, attributes, and operations. Relationships such as associations and inheritances are illustrated with lines and symbols. Class diagrams are essential for modeling system architecture and guiding development processes.



**State-chart Diagram:**

A state-chart diagram, or state machine diagram, illustrates the dynamic behavior of an object by showing its possible states and the transitions between them. States are represented as rounded rectangles, while arrows indicate transitions triggered by events. The diagram can also include entry and exit actions for each state. State-chart diagrams are valuable for modeling object lifecycles and system responses to various events.

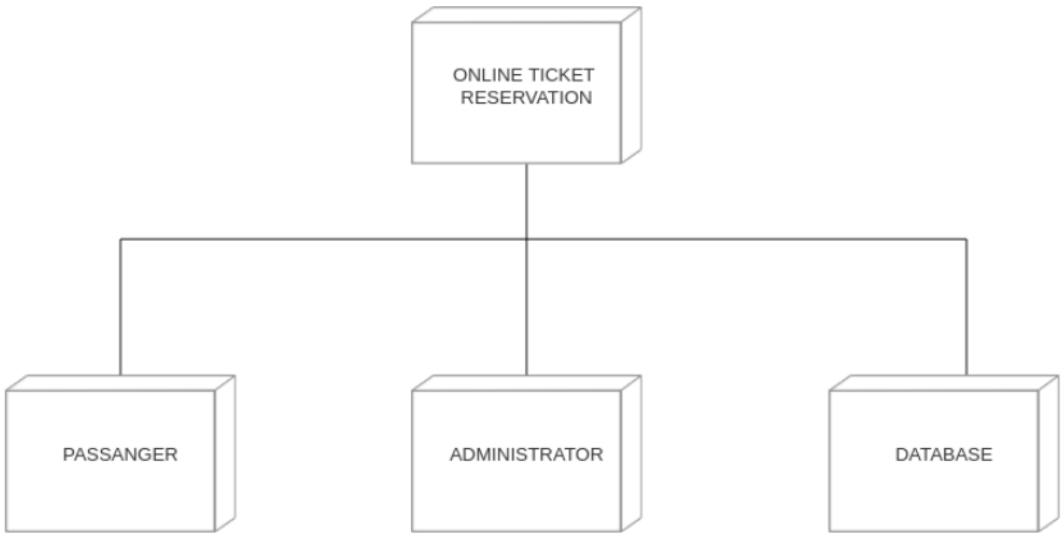

Ticket booking
Ticket reserve
Ticket sold
Ticket cancelled
Ticket expired
User searches for tickets. The ticket is open and available for booking.
The ticket is temporarily held for the user during the checkout process.
User completes payment for ticket if it is available.
User cancels booking the ticket.
Reservation time exceeds

State chart diagram for E-Ticketing system

**Deployment Diagram:**

A deployment diagram visualizes the physical deployment of software artifacts on nodes, such as servers or devices, within a system. Nodes are represented as three-dimensional boxes, while artifacts are shown as rectangles. The diagram highlights the relationships and communication pathways between components. Deployment diagrams are essential for understanding system architecture and planning effective infrastructure.

**DEPLOYMENT DIAGRAM:**



# HOTEL MANAGEMENT SYSTEM

**Problem statement:** The Hotel Management System is a web-based application designed to streamline hotel operations for managers through an interactive GUI, enabling efficient handling of room bookings, staff management, and various hotel activities. Tailored for busy managers, this system centralizes management tasks, eliminating the need for manual, paper-based processes. Managers can post available rooms, and customers can easily view and book them online. An admin feature allows for booking approvals, ensuring controlled and organized reservations. Additionally, customers can access and book other hotel services, making the system convenient for both managers and customers by providing an all-in-one platform to efficiently manage hotel activities.

- 
- **Features:**
- **Admin login & admin dashboard:** It has admin login who has the authority of the system & he is responsible for approving & disapproving
- the users request for room booking. Admin can add & delete notifications & updates in the system.
- **User registration:** There is user registration form available where new users can create their account by providing required information to the system.
- **Booking system:** User can request for the table booking for a particular date & time.
- **Approving/Disapproving request:** The booking requests are directly sent to admin account by the system. Admin can view all the requests along with respective user details & therefore make decisions for cancelling the requests.

**Software Requirements Specification:**

- **Functional requirements:**

These describe what the system should do, i.e., the core functionalities needed to manage the operations of a hotel.

1. User Management:
   - The system should allow administrators to create, update, and delete user profiles for different roles (e.g., admin, staff, guest).
   - The system should allow guests to create their own accounts and manage personal information.
2. Room Management:
   - The system should allow the hotel to manage room availability, types, and prices.
   - The system should display available rooms based on guest search criteria (e.g., room type, number of guests, check-in/check-out dates).
   - The system should allow staff to update room status (e.g., clean, dirty, maintenance).
3. Reservation Management:
   - Guests should be able to make room reservations online or at the hotel front desk.
   - The system should validate room availability during booking and prevent overbooking.
   - Guests should receive confirmation of their reservation via email or SMS.
   - The system should allow cancellations and modifications to bookings within specified policies.
4. Check-in and Check-out:
   - The system should support check-in and check-out processes, including issuing room keys (physical or digital).
   - The system should track check-in and check-out times.
   - The system should automatically calculate the total bill based on stay duration, room type, and any additional services.
5. Billing and Payments:
   - The system should support generating invoices based on room charges, services used (e.g., minibar, room service), and taxes.
   - The system should allow guests to pay using various methods (e.g., credit/debit cards, cash, online payments).

  - The system should allow guests to view a detailed breakdown of charges at check-out.
6. Inventory and Housekeeping Management:
   - The system should track inventory levels for consumables (e.g., linens, toiletries).
   - The system should allow housekeeping staff to track room cleaning schedules and requirements.
   - The system should notify staff if supplies are running low.
7. Customer Feedback and Complaints:
   - The system should allow guests to provide feedback or lodge complaints about their stay.
   - The system should track customer feedback and allow staff to respond or resolve complaints.
8. Reporting:
   - The system should provide reports on occupancy rates, revenue, guest demographics, and room usage.
   - It should allow administrators to generate financial reports, including income, expenses, and outstanding payments.
9. Multi-language Support:
   - The system should support multiple languages for international guests.

- **Non-functional requirements:**

These describe the qualities the system must exhibit, ensuring it performs well under various conditions and is maintainable.

1. Performance:
   - The system should handle up to [X] simultaneous users without significant degradation in performance.
   - The system should process room reservations and check-ins within [X] seconds to ensure quick service.
   - The system should be able to generate reports within [X] seconds/minutes.
2. Scalability:

- The system should be able to scale horizontally to accommodate increasing user load as the hotel chain grows (e.g., adding more locations or users).

3. Availability:
   - The system should be available 24/7 with an uptime of [99.9%] or better.
   - Maintenance windows should be scheduled and communicated in advance.
4. Security:
   - The system should comply with data protection regulations (e.g., GDPR, CCPA) to safeguard guest information.
   - The system should implement encryption for sensitive data (e.g., payment information, personal guest details).
   - Access control should be implemented to ensure that only authorized personnel can access specific data or functions (e.g., admin access, financial reports).
   - The system should support two-factor authentication (2FA) for sensitive user accounts.
5. Usability:
   - The system should have an intuitive and user-friendly interface, making it easy for staff and guests to use.
   - The system should support mobile devices and provide a responsive design for both hotel staff and guests.
6. Compatibility:
   - The system should be compatible with different browsers (e.g., Chrome, Firefox, Safari) and mobile operating systems (iOS, Android).
   - It should integrate with third-party tools (e.g., payment gateways, CRM systems, channel managers).
7. Maintainability:

- The system should be easy to update and maintain, with clear documentation for both users and developers.
- It should support modular development to add new features with minimal disruption.

8. Backup and Recovery:
    - The system should have an automated backup mechanism to ensure data is regularly backed up.
    - In the event of a failure, the system should have a disaster recovery plan to restore data within [X] hours.
9. Compliance:
    - The system should meet all local and international regulations for handling guest data, payments, and business operations.

**Software requirements:**

- Windows XP, Windows 7(ultimate, enterprise)
- Visual Studio 2010
- SQL 08

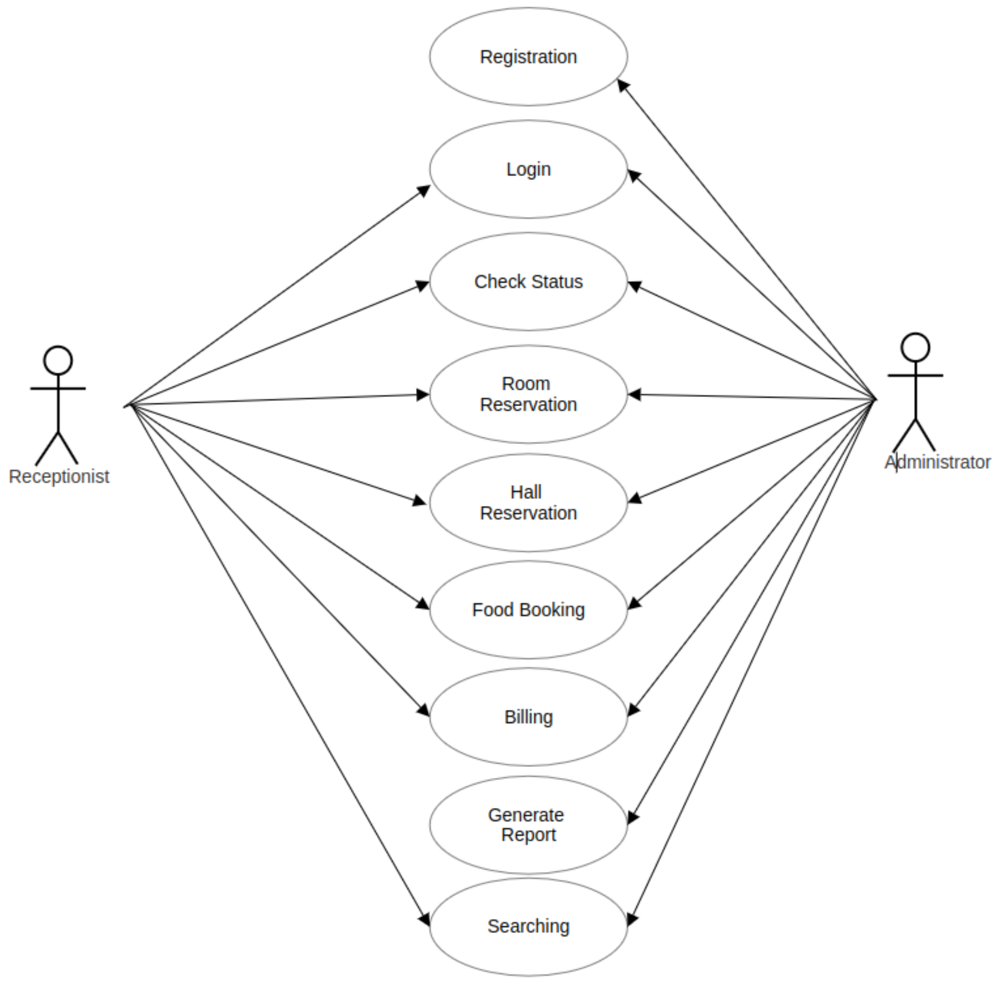
**Hardware components:**

- Process - p4
- Hard disk - 5GB
- Memory - 1GB RAM

## UML Diagrams:

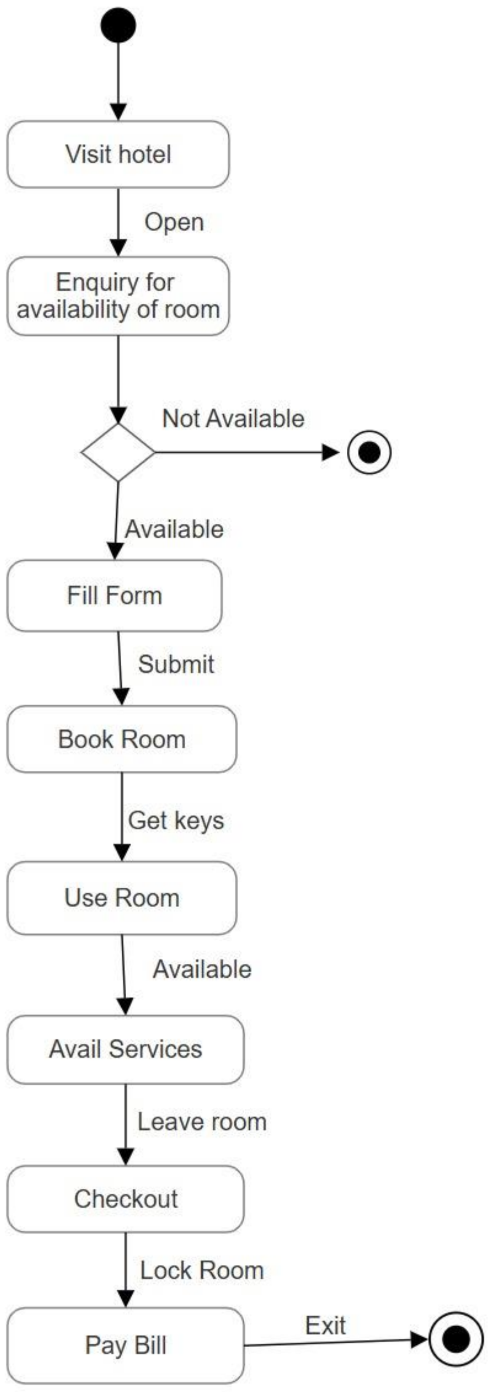
### Use-case diagram:

A use case diagram is a tool to visualize system requirements by showing how actors interact with the system. In the Hotel Management System, there are two main actors: the Receptionist and the Administrator. The Receptionist handles tasks like guest registration, booking rooms and halls, managing food orders, processing billing, and generating reports. The Administrator has control over all functions, including monitoring and adjusting operations. The diagram uses stick figures to represent actors, connecting them to system functions to clarify user interactions and system requirements.



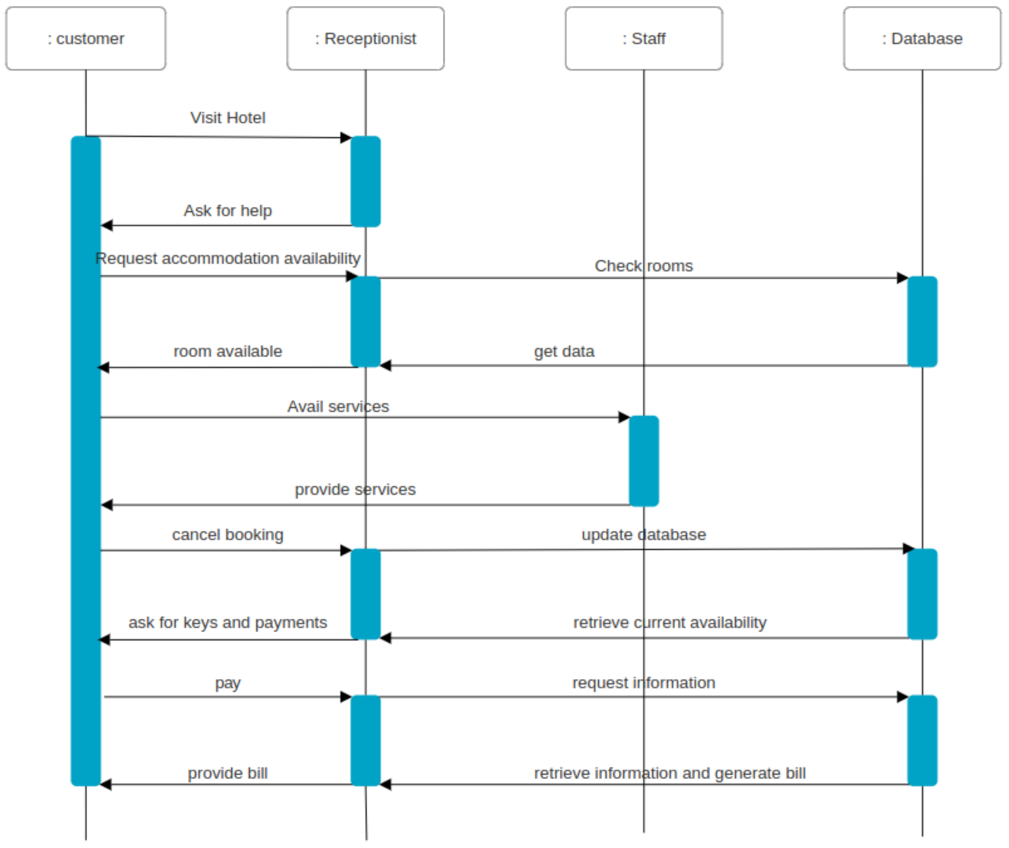
**Activity diagram:**

An activity diagram shows the flow of actions in a system. In the Hotel Management System, the Receptionist registers guests, checks availability, books room, manages orders, and processes billing. Decisions like availability or payment affect the next steps. The Administrator oversees and manages these tasks, ensuring smooth operations. The diagram uses actions and decision points to represent the workflow.



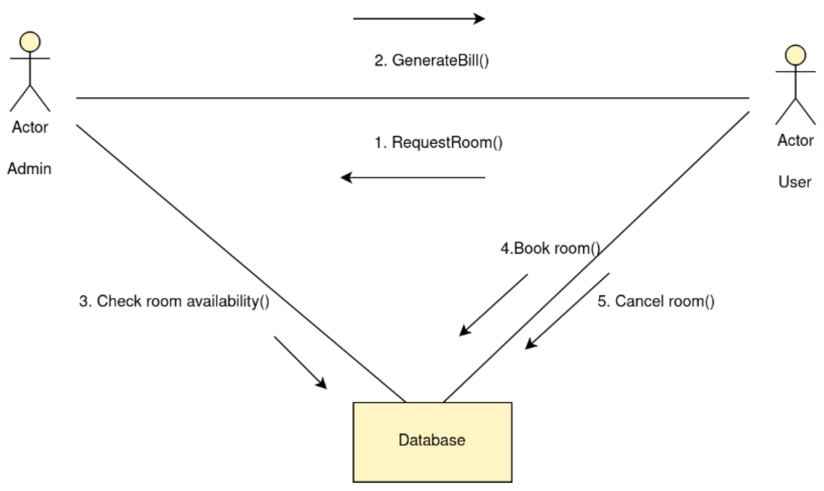
**Sequence diagram:**

A sequence diagram shows how actors interact with the system over time. It displays the order of messages exchanged between actors and the system to perform tasks such as booking rooms or processing payments. The diagram focuses on the sequence of actions, helping visualize the flow of operations in the system.



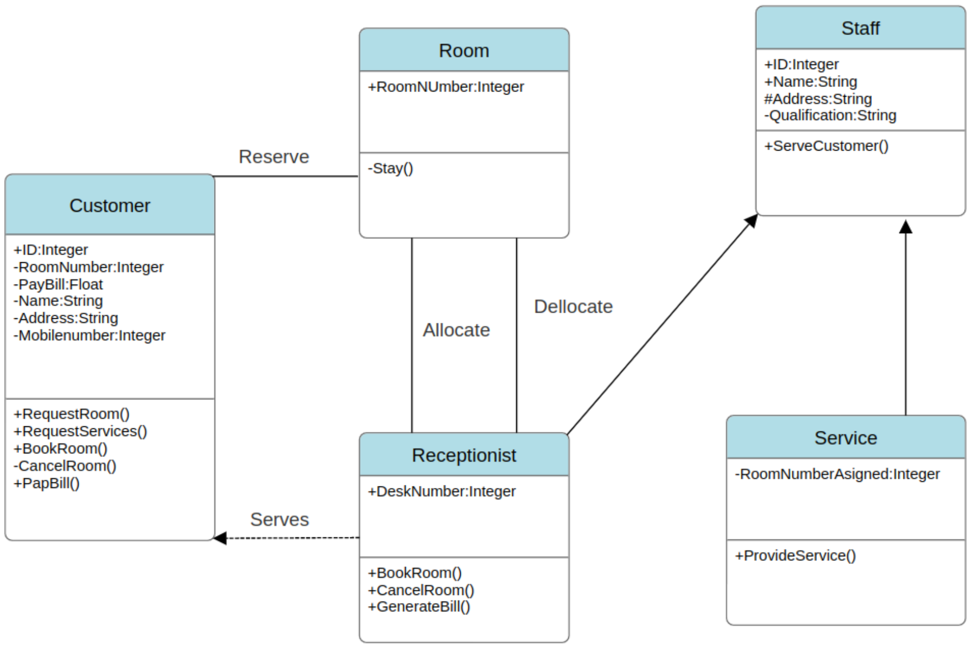
**Collaboration diagram:**

A collaboration diagram shows how actors and objects interact within a system. It focuses on the relationships between components, highlighting the messages exchanged to complete tasks. In the Hotel Management System, the diagram illustrates how the Receptionist and Administrator interact with different system objects to perform tasks, emphasizing the structure and flow of communication between them.



**Class diagram:**

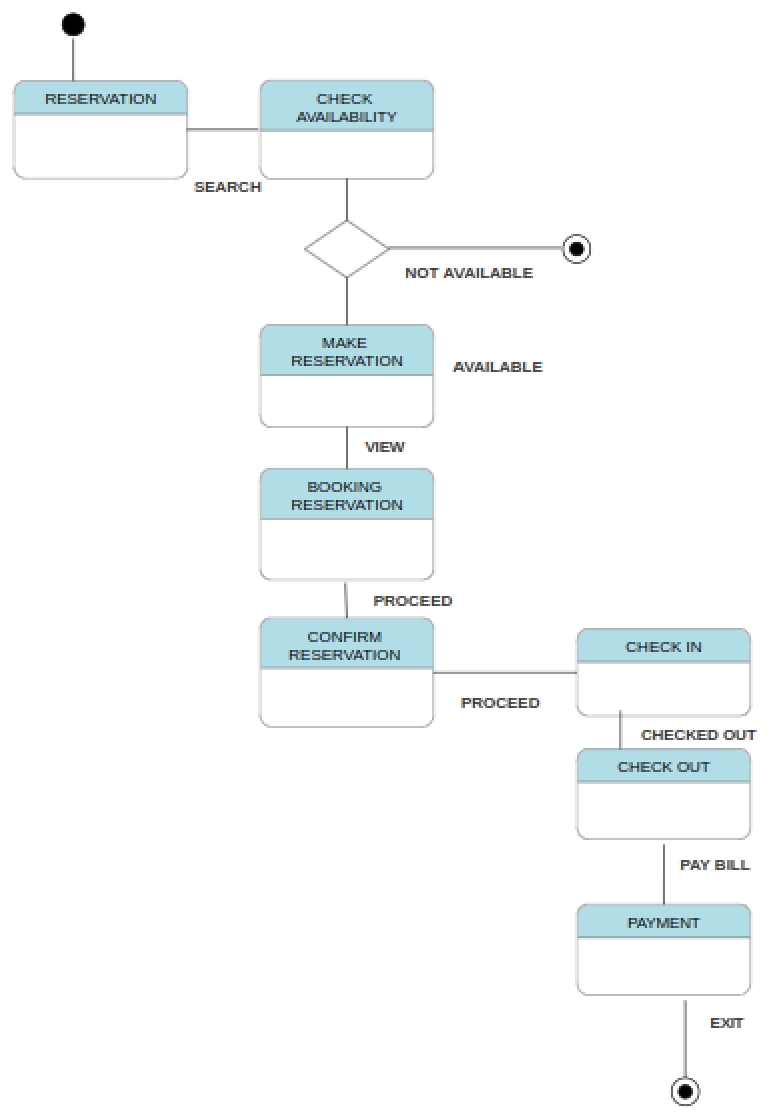
A class diagram shows the structure of a system by defining its classes, attributes, methods, and relationships. It represents real-world entities and how they interact within the system. The diagram helps visualize the system's architecture and the connections between its components. For a Hotel Management System, you'd have classes like Guest, Room, Reservation, and Billing, each with their own details, like the guest's name or the room number, and functions, such as checking in or processing payments.



Class Diagram for Hotel Mangement System

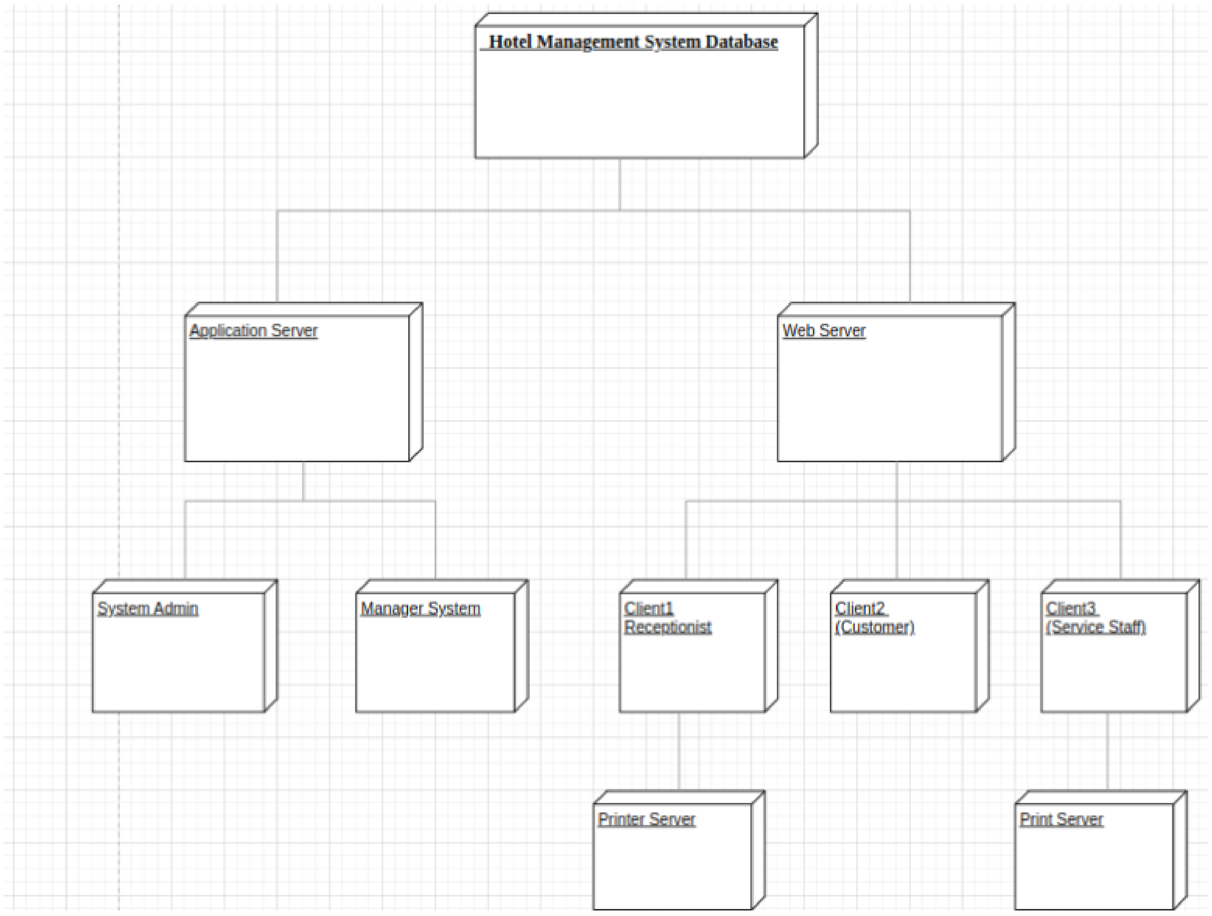
**State chart diagram:**

A state chart diagram shows the different states an object can be in and how it moves between those states based on events. In a Hotel Management System, a guest might move from "Reserved" to "Checked-in" or "Checked-out" depending on actions taken. It helps track the changes an object goes through over time.



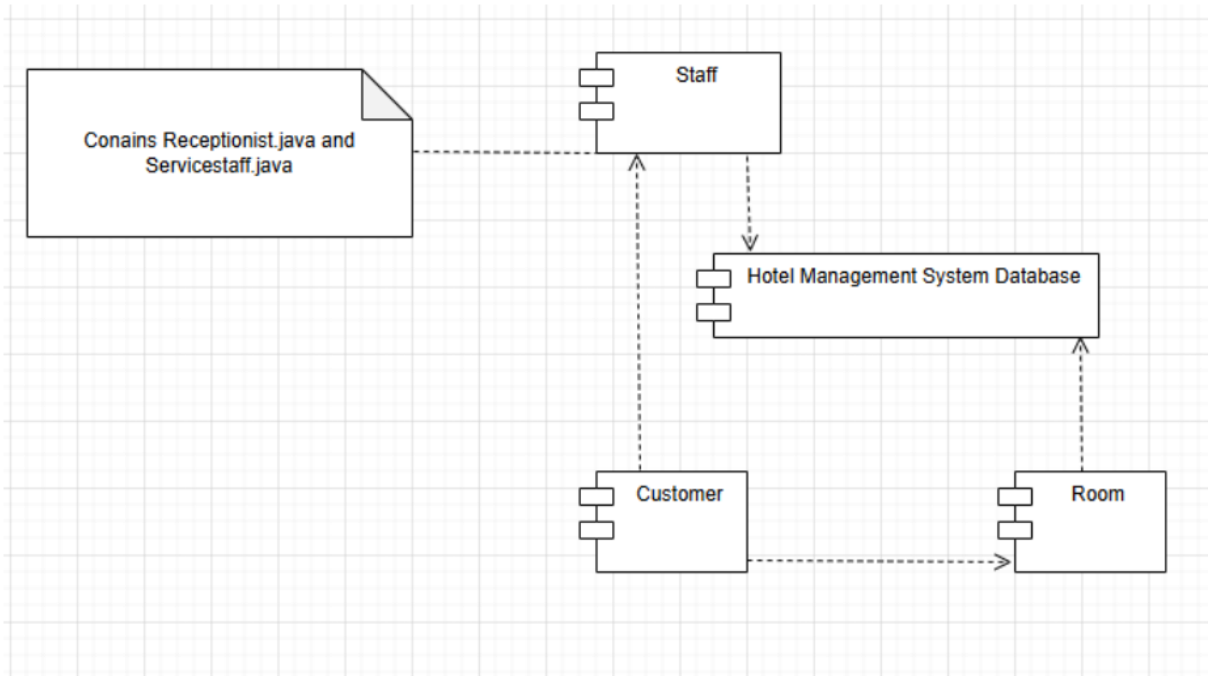
**Deployment diagram:**

A deployment diagram shows the physical setup of a system, including hardware components and their connections. In a Hotel Management System, it illustrates how devices like servers and reception computers are linked, showing the system's architecture.



**Component diagram:**

A component diagram shows the major components of a system and how they interact. It focuses on the system’s structure by breaking it down into smaller, functional pieces. In a Hotel Management System, components might include the booking system, billing system, and database, each connected to show how they work together to handle tasks like reservations and payments.


Conains Receptionist.java and Servicestaff.java
Staff
Hotel Management System Database
Customer
Room

# HOSPITAL MANAGEMENT SYSTEM

**Problem statement:** The Hospital Management System (HMS) addresses the challenges faced by healthcare facilities in managing patient records, appointments, billing, and inventory. Manual processes often lead to inefficiencies, data errors, and delays in service delivery, impacting patient care. The system aims to automate and centralize these functions, improving operational efficiency and communication. It provides seamless management of patient data, scheduling, and billing, ensuring accurate records and compliance with regulations. Ultimately, the HMS enhances the overall hospital experience for both patients and staff, optimizing healthcare delivery

**Features:**

- **Admin login & admin dashboard:** The admin can securely log in to the system, accessing a centralized dashboard to manage hospital operations, view statistics, and oversee users and appointments.
- **User registration:** Patients and hospital staff can register on the system, providing necessary personal details and medical information for efficient service management.
- **Booking system:** Patients can book appointments with doctors through an easy-to-use interface, choosing their preferred doctor and available time slots.
- **Approving/Disapproving request:** The admin or relevant authority can approve or disapprove patient requests, such as appointment bookings or service access, based on availability and requirements.

**Software Requirements Specification:**

- **Functional requirements:**

These describe what the system should do, i.e., the core functionalities needed to manage the operations of a Hospital Management System.

**1 Patient Management**

- **Registration:** Create and update patient profiles.
- **EHR Management:** Maintain and update patient's medical history, diagnoses, and treatment records.
- **Patient Search:** Easily retrieve patient records using patient ID, name, or contact details.

**2 Appointment Management**

- **Appointment Scheduling:** Patients can book, reschedule, or cancel appointments.
- **Doctor's Schedule:** Doctors can update their availability, view, and manage appointments.
- **Appointment Notifications:** Send reminders to patients and doctors via email or SMS.

**3 Billing and Payments**

- **Invoice Generation:** Generate invoices for consultations, procedures, and treatments.
- **Payment Processing:** Integrate with payment gateways for online payments.
- **Insurance Integration:** Handle insurance claims and generate insurance reports.

**4 Inventory and Pharmacy Management**

- **Stock Management:** Track and manage stock levels for medicines, medical supplies, and equipment.
- **Inventory Alerts:** Notify administrators of low stock or expired items.
- **Pharmacy POS System:** Simplify medicine dispensing and billing at the pharmacy.

**5 Reporting and Analytics**

- **Operational Reports:** Generate reports for patient inflow, financials, inventory levels, and staff performance.
- **Regulatory Compliance Reports:** Produce reports for healthcare regulations (e.g., HIPAA compliance).

- **Dashboard:** Visual display of hospital performance metrics, such as patient wait times and revenue.

**6 Security and Access Control**

- **Role-based Access Control:** Different user roles (e.g., Administrator, Doctor, Patient) with specific access permissions.
- **Audit Logs:** Track system usage and changes made to sensitive data.
- **Data Encryption:** Ensure patient data is encrypted during storage and transmission to maintain privacy.

**Non-functional requirements:**

These describe the qualities the system must exhibit, ensuring it performs well under various conditions and is maintainable.

1. Performance:
   - The system must provide high responsiveness and efficiency under normal and peak usage conditions. All user actions, such as booking appointments, retrieving medical records, and generating invoices, should have a response time of no more than 2 seconds. Additionally, the system should be capable of handling a significant number of concurrent users (e.g., 500+ simultaneous users) without experiencing performance degradation..
2. Scalability:
   - The HMS should be designed to scale as the hospital grows. It must accommodate an increasing number of patients, doctors, and other users. The system architecture should allow for the addition of new modules (e.g., additional branches, new departments, or specialized services) and should support horizontal scaling, allowing it to handle more users or transactions as needed, without requiring a complete redesign of the system.
3. Availability:
   - The system should be available 99.9% of the time, ensuring that users can access the hospital services without interruptions. This includes considerations for system uptime, backup mechanisms, and

disaster recovery procedures. Planned maintenance should be scheduled during off-peak hours, with advance notice given to users.

4. Security:

   - The system must meet high-security standards to protect sensitive patient data and comply with regulations such as HIPAA or GDPR. This includes secure user authentication (e.g., multi-factor authentication), encryption of patient records during transmission and storage, access control based on user roles, and regular security audits to detect vulnerabilities. The system must also prevent unauthorized access to medical and financial records.

5. Usability:

   - The system must be user-friendly, ensuring that all types of users, from patients to hospital staff, can navigate and operate the system easily. The interface should be intuitive, with clear labeling and minimal learning curves. It should also be designed to accommodate users with disabilities, meeting accessibility guidelines.

6. Compatibility:

   - The HMS should be compatible with a wide range of operating systems (e.g., Windows, macOS, Linux) and devices (desktop computers, laptops, tablets, and smartphones). It should also support different web browsers (e.g., Chrome, Firefox, Safari) and allow integration with external systems, such as third-party payment gateways, laboratory systems, and medical equipment..

7. Maintainability:

   - The system should be designed for easy updates and maintenance with minimal downtime and clear documentation..

8. Backup and Recovery:

   - The system must implement automatic daily backups of all critical data (e.g., patient records, appointments, billing) to ensure data integrity. In case of system failure or data loss, recovery processes should allow for quick restoration of data with minimal downtime, ensuring that operations can resume smoothly without significant impact on hospital services.

9. Compliance:
    - The system must adhere to healthcare regulations such as HIPAA and GDPR to ensure the privacy and security of patient data. It should also meet industry standards for data protection and accessibility.

**Software requirements:**

- Windows XP, Windows 7(ultimate, enterprise)
- Visual Studio 2010
- SQL 08

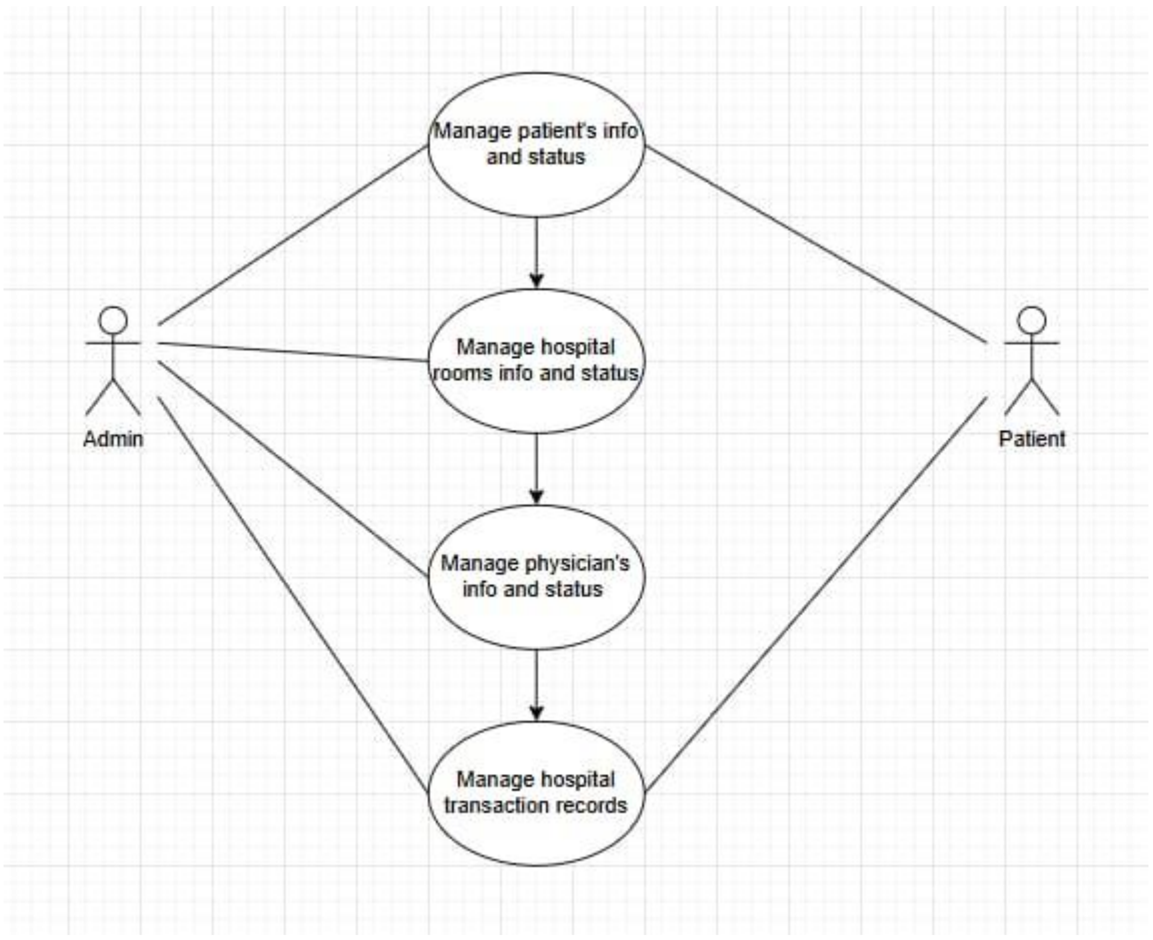
**Hardware components:**

- Process - p4
- Hard disk - 5GB
- Memory - 1GB RAM

## UML Diagrams:

### Use-case diagram:

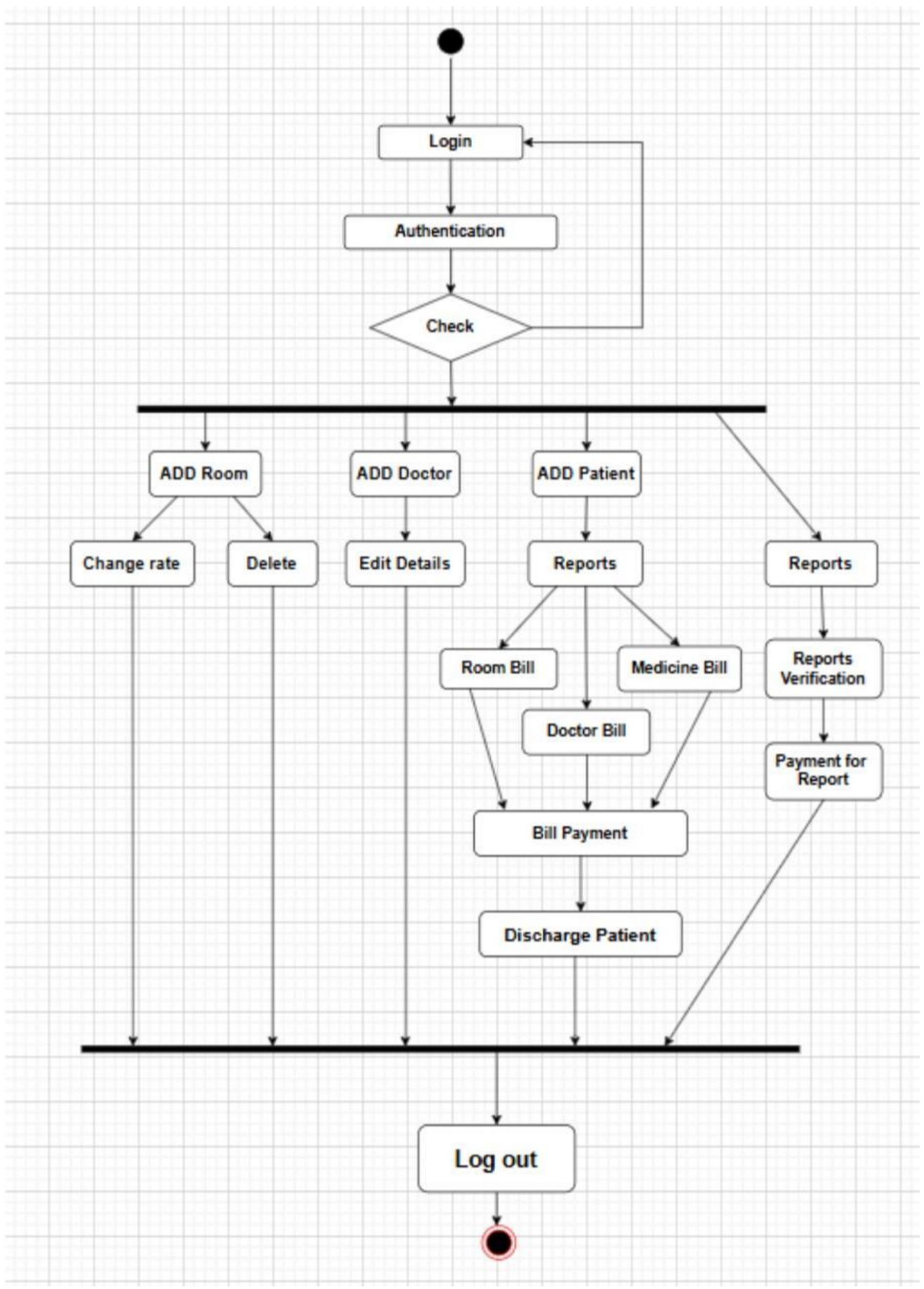
The  use case in a Hospital Management System enables patients to book consultations with doctors seamlessly. The patient logs into the system, selects the desired specialization, and views available doctors and time slots. After choosing a doctor and an appropriate time, the patient confirms the appointment, which is then updated in both the patient's and the doctor's schedules. Notifications are sent to both parties via SMS or email to ensure reminders. If no slots are available, the system suggests alternate dates or doctors with similar expertise. The receptionist can assist patients who are unable to book appointments themselves by performing the same process on their behalf. In case of emergencies or changes in the doctor's availability, the system notifies the patient to reschedule. This feature streamlines appointment management, reduces wait times, and enhances patient experience.


Manage patient's info and status
Manage hospital rooms info and status
Manage physician's info and status
Manage hospital transaction records
Admin
Patient

**Use-case diagram for hospital management system**

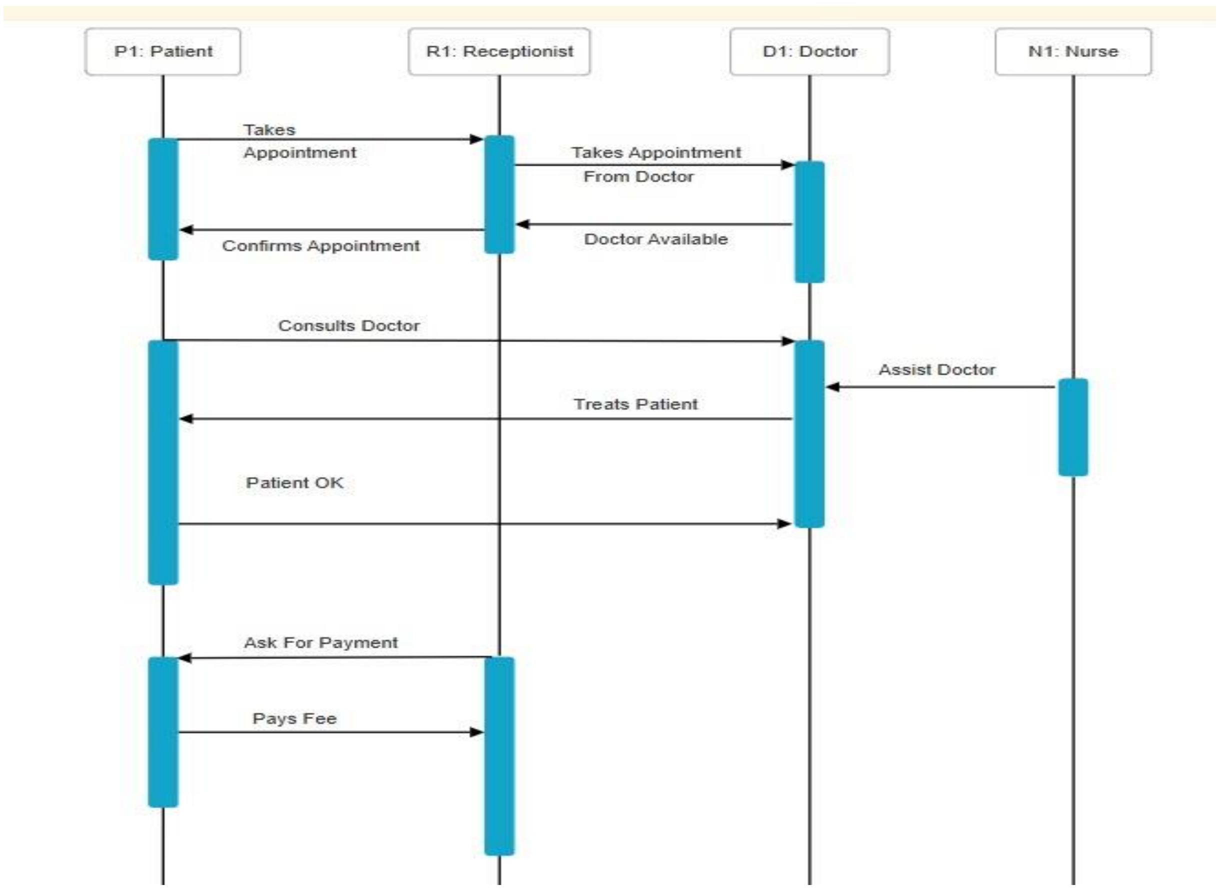
**Activity diagram:**

An activity diagram for a Hospital Management System outlines the workflow of processes such as patient registration, appointment scheduling, billing, and inventory management. It visualizes the flow of activities, from a patient booking an appointment to the completion of medical treatment and the final billing process. The diagram also includes decision points, like handling payment methods or managing inventory stock levels, ensuring clear understanding of system processes and roles.



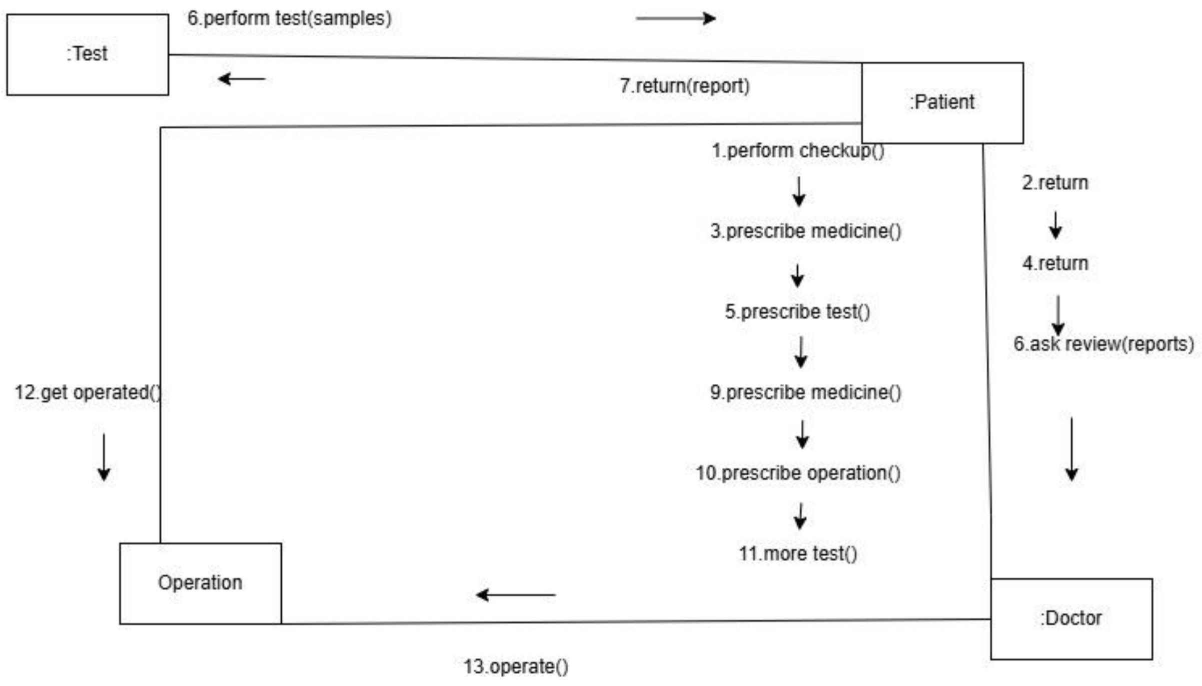
**Sequence diagram:**

A sequence diagram for a Hospital Management System illustrates the interaction between objects for processes like appointment scheduling. It shows the sequence of messages exchanged between the patient, the system, and the doctor, starting from the patient requesting an appointment to the system confirming the booking. This diagram helps visualize the step-by-step communication flow, ensuring that each actor and system component performs its role in a timely and synchronized manner.



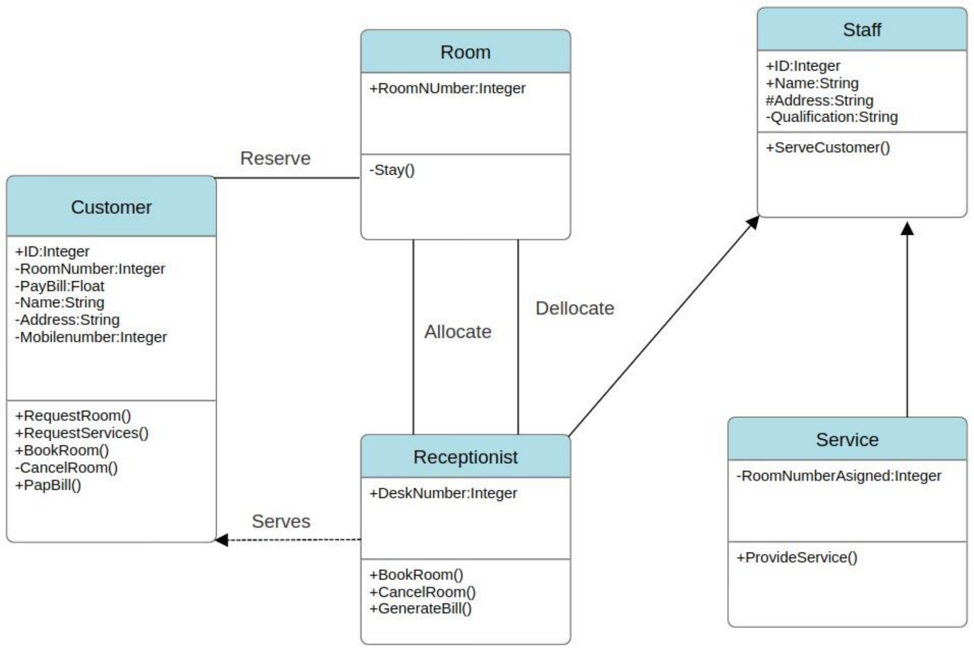
**Collaboration diagram:**

A collaboration diagram for a Hospital Management System represents the interaction between system components and actors, focusing on how objects collaborate to complete a process. For example, when a patient schedules an appointment, the diagram shows the interactions between the patient, the appointment system, the doctor's schedule, and notification services. This diagram highlights the relationships and communication paths between objects, making it easier to understand the flow of data and responsibilities within the system.



**Class diagram:**

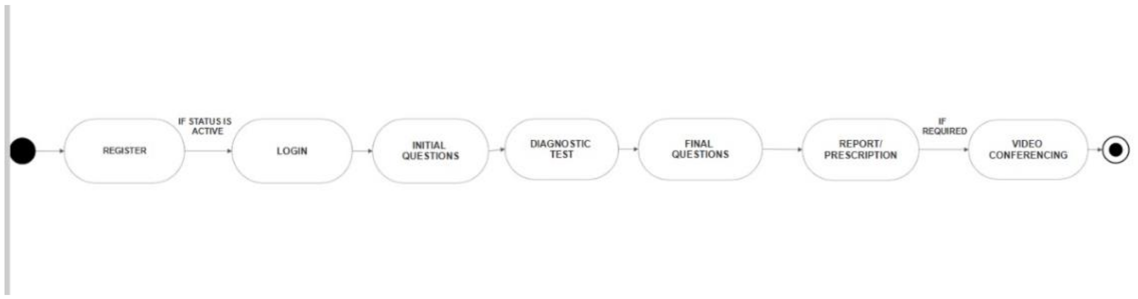
A class diagram for a Hospital Management System defines the structure of the system by representing key classes, their attributes, and methods. It includes classes such as Patient, Doctor, Appointment, Billing, and Inventory, with relationships like associations, inheritances, and dependencies between them. This diagram provides a blueprint for the system's object-oriented design, helping to visualize how different entities within the hospital interact and collaborate to manage operations efficiently..



Class Diagram for Hotel Mangement System

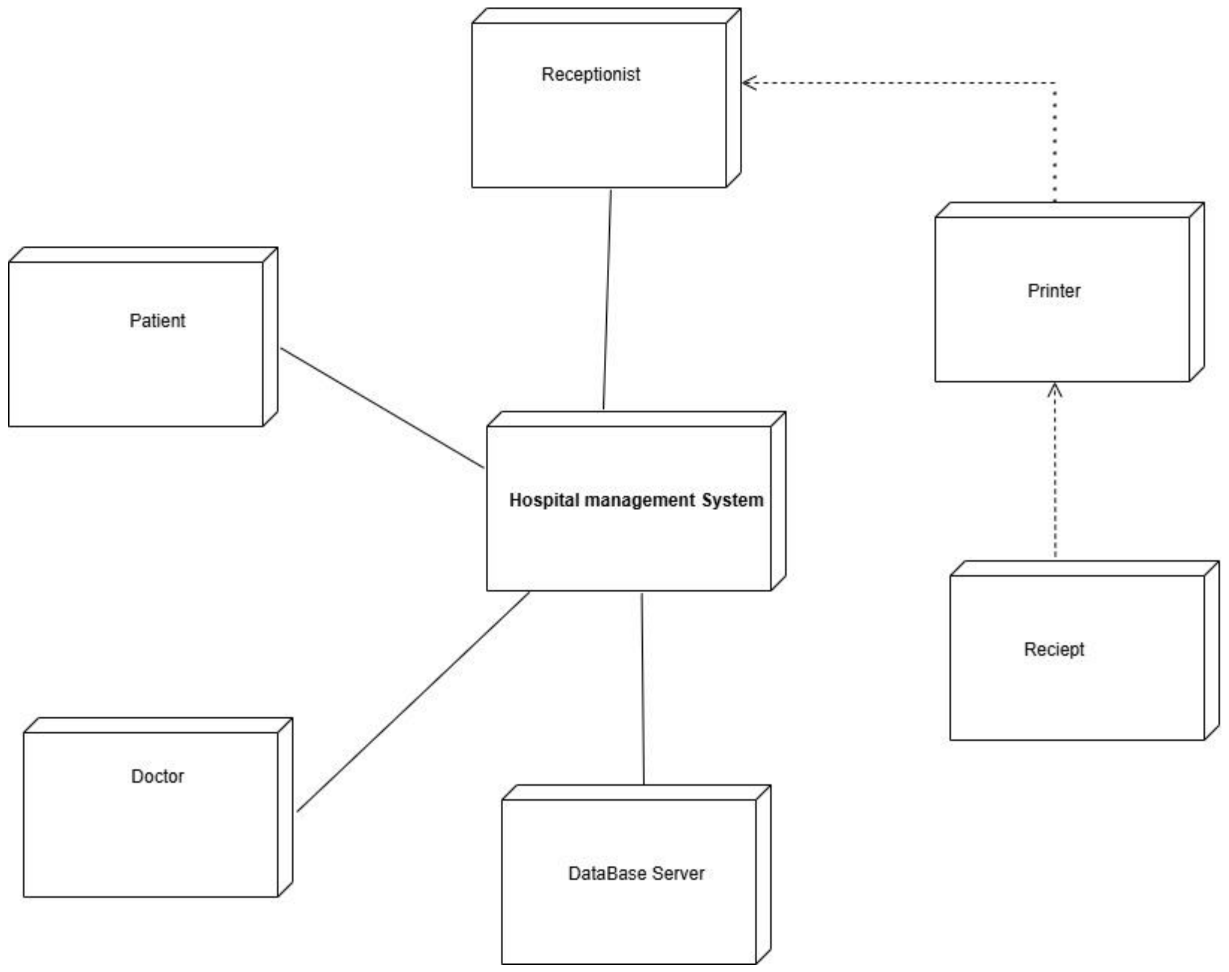
**State chart diagram:**

A state chart diagram for a Hospital Management System illustrates the different states an object, such as an Appointment or Patient, can transition through during its lifecycle. For example, an appointment can move through states like Scheduled, In Progress, Completed, and Canceled based on interactions with the system. This diagram helps visualize the dynamic behavior of objects, ensuring that the system responds appropriately to different events or changes in state throughout hospital operations.



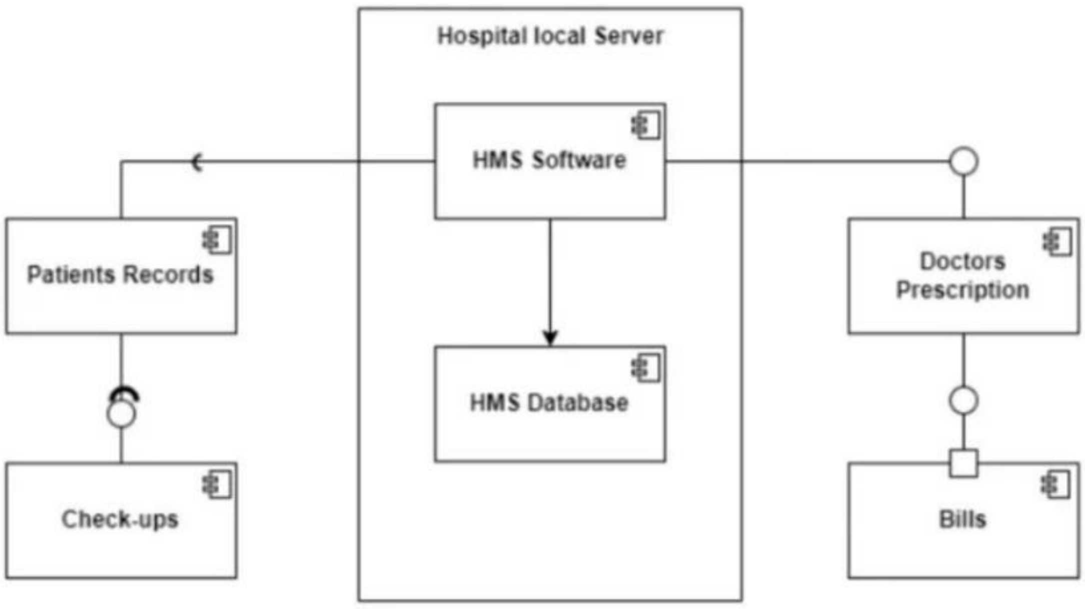
**Deployment diagram:**

A deployment diagram for a Hospital Management System outlines the physical deployment of the system components across hardware nodes. It shows how the system’s software, including the web application, database server, and client devices, is distributed across different servers and devices. This diagram helps visualize the system's architecture, ensuring efficient resource allocation, scalability, and communication between various components such as the user interface, application server, and database server..


Receptionist
Patient
Printer
Hospital management System
Reciept
Doctor
DataBase Server

**Component diagram:**

A component diagram for a Hospital Management System depicts the high-level structure of the system by showing the main components and their relationships. It includes components such as the User Interface, Backend Server, Database ,Payment Gateway, and Notification Service. This diagram helps to understand how each part of the system interacts with others, ensuring smooth data flow and integration between modules like patient management, billing, and appointment scheduling..


Hospital local Server
HMS Software
HMS Database
Patients Records
Check-ups
Doctors Prescription
Bills