SOFTWARE ENGINEERING PROJECT

COMPUTER SCIENCE & ENGINEERING

(Artificial Intelligence & Machine Learning)

Submitted By

23WH1A6601 K. LAKSHMI CHAITANYA

23WH1A6607 P. MAHIMA MEGHANA

23WH1A6610 M. SRIJANI

23WH1A6615 N. AKSHITHA

23WH1A6651 K. GEETA REDDY

23WH1A6662 B. ABHIGNA

23WH1A6663 B. RUTHVIKA

24WH5A6603 K. THRIPADA

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)

VISHNU
UNIVERSAL LEARNING
BVRIT H

CERTIFICATE

This is to certify that the Software Engineering project is a bonafide work carried out **by Ms. K. LAKSHMI CHAITANYA(23WH1A6601), Ms. P. MAHIMA MEGHANA(23WH1A6607), Ms. M. SRIJANI (23WH1A6610), Ms. N. AKSHITHA (23WH1A6615), Ms. K. GEETA REDDY (23WH1A6651), Ms. B. ABHIGNA (23WH1A6662), Ms. B. RUTHVIKA (23WH1A6663), Ms. K. THRIPADA (24WH5A6603)** 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
Ms. V. ASHA
Assistant Professor
Dept of CSE(AI&ML)

Head of the Department
Dr. B Lakshmi Praveena
HOD & Professor
Dept of CSE (AI&ML)

INDEX

CREDIT CARD PROCESSING

PROBLEM STATEMENT**:**

Credit card processing through offline involves the merchant collecting order information (including credit card numbers), storing this in a database on your site, and entering it using their on-site merchant credit card processing system. Takes time to manually enter credit card information for each order. This solution creates following cons:

• Insecure – there is a possibility that a skilled hacker could break into the database and steal an entire list of credit card numbers, thereby damaging the merchant's reputation with current client.

• There is a higher risk of customer charge backs with no signature

• Higher risk of fraud for using stolen credit cards

• Many discerning online shoppers will not give their credit card to an "entrusted" online merchant (you may want to consider being part of the Better Business Bureau or similar organization to add credibility). So there is a need of online and trusted credit card processing.

1. INTRODUCTION

1.1 PURPOSE**:**

Credit card processing software is used to manage credit card details of a bank. This includes the sequence of phases involved in the creation of a credit card and the usage details of the card owned by the bank's customers.

1.2 SCOPE:

Credit card processing is used in banks all over the world to process their respective credit cards. The simplicity under the use of credit

cards have engrossed a mass of people into using them bringing a great deal of profit to the bank organizations.

1.3 DOCUMENT OVERVIEW:

This document of software requirement specification contains the functional requirements, use case identification and the non-functional requirements. The non-functional requirements would be focussed at a greater detail in a Business based development environment.

2. GENERAL DESCRIPTION

2.1 SYSTEM ENVIROMNENT:

The system composes of a central server for a bank through which the bank itself and the internet server for credit cards is connected. The credit card purchase is done over the internet.

2.2 USER CHARACTERISTICS:

The actors associated with the system and their descriptions are as follows:

Card Holder:

This is the owner of a credit card. This is a primary actor. The cardholder is the only one who can purchase through his/her card.

Verifier:

This is a supporting actor who verifies the details given by an applicant.

Card Issuer:

This is a primary actor whose duty is to issue, renew and replace credit cards.

2.3 NON-FUNCTIONAL REQUIREMENTS

The following are the requirements that focus on the overall quality of the software.

2.3.1 PRODUCT REQUIREMENTS:

Usability**:**

The product should be simple enough to be operated by the actors concerned.

Efficiency:

The product should occupy the least possible memory space and provide greater performance.

Reliability:

The database must not be corrupted in case of power failure or any malfunctioning of the underlying system.

2.3.2 ORGANIZATIONAL REQUIREMENTS

Deliverability**:**

The deliverable product must include the database pack necessary to handle the software along with the software itself.

Implementation**:**

The product should be safe to use and should not be easily subjected to hacking.

2.3.3 EXTERNAL REQUIREMENTS

Interoperability:

This product must include the database pack necessary to handle the software along with the software itself.

Implementation:

The product should be safe to use and should not be easily subjected to hacking.

3. SPECIFIC REQUIREMENTS

3.1 FUNCTIONAL REQUIREMENTS

The following are the requirements necessary for the system to function for its purpose.

- There must be database to store the details of credit cards along with the details.
- There must be a way in which an applicant can produce his/her details to the bank to apply for a credit card.
- The details given by the applicant should be put through verification.
- Applications that pass the police verification should be put for processing.
- The statuses of the credit card phases must be indicated.
- The fully processed credit card must be issued to the applicant.
- The credit card details must be maintained even after issue.
- The credit cards that are obsolete must be put through for renewal.

4. Design Phase tool

StarUML:

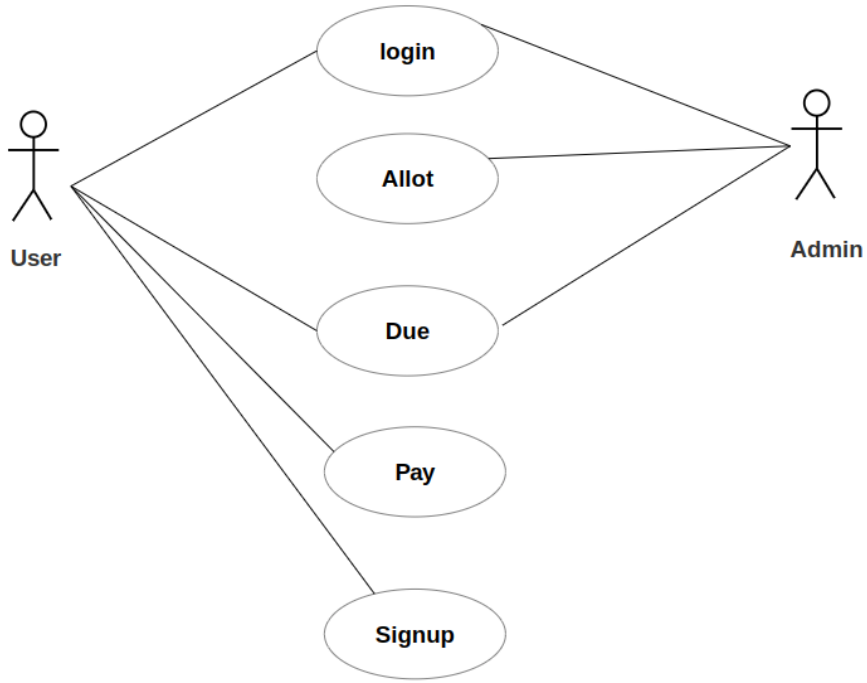
StarUML is an open-source software modelling tool that supports the UML (Unified Modelling Language) framework for system and software modelling. It is based on UML version 1.4, provides different types of diagrams and it accepts UML 2.0 notation. It actively supports the MDA (Model Driven Architecture) approach

by supporting the UML profile concept and allowing to generate code for multiple languages.
StarUML supports the following diagram types:

UML DIAGRAMS**:**

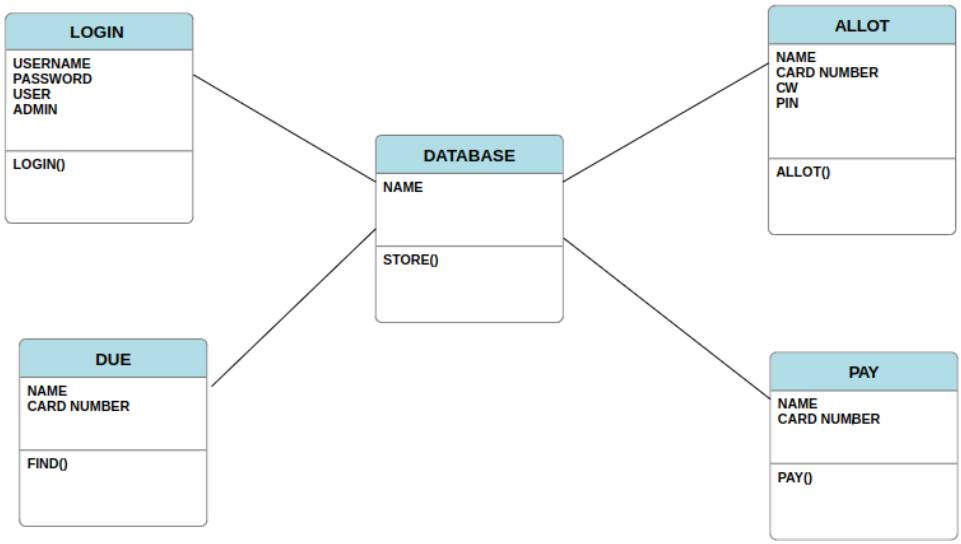
1. Use Case diagram
2. Class diagram
3. Sequence diagram
4. Collaboration diagram
5. State-chart diagram
6. Activity diagram
7. Deployment diagram

I. Use case diagrams
A use case diagram is a diagram that shows a set of use cases and actors and their relationships.
Common Properties**:**
A use case diagram is just a special kind of diagram and shares the same common properties as do all other diagrams - a name and graphical contents that are a projection into a model. What distinguishes a use case diagram from all other kinds of diagrams is its content.
Contents:
Use case diagrams commonly contain
1.Use cases
2. Actors
3.Dependency, generalization, and association relationships
Common Uses:
The use case diagrams are used to model the static use case view of a system. This view primarily supports the behaviour of a system - the outwardly visible services that the system provides in the context of its environment.



2. CLASS DIAGRAM

A class diagram for a credit card processing system models the key entities and their relationships, showing how the system processes transactions and manages accounts, cards, customers, and merchants. This class diagram represents a modular and secure approach to managing the flow of transactions, with each component responsible for a distinct aspect of the payment process. It allows for scalability, where more functionality (such as enhanced fraud detection) can be added without disrupting the core transaction flow.



3.SEQUENCE DIAGRAM

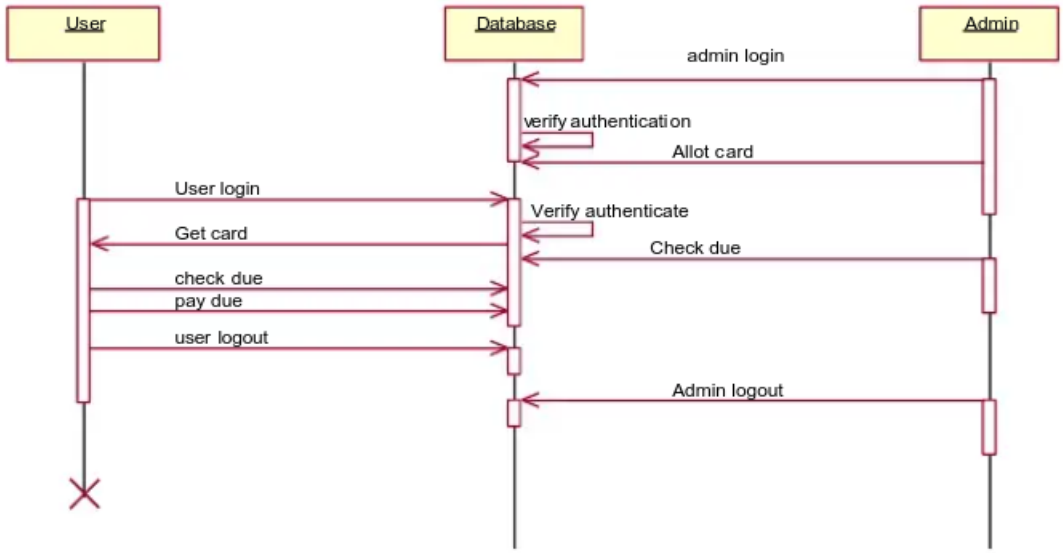
A sequence diagram for credit card processing shows the interactions between the primary entities involved in processing a payment transaction from start to finish. This includes the steps taken from when a customer initiates a transaction to when the payment is completed or declined.

Here's a description of each step in the sequence:

- Parallel Processing: Fraud detection and authorization can occur in parallel or sequentially, based on system design.
- Synchronous and Asynchronous Calls: Requests for authorization and fraud detection are synchronous, whereas settlement can be asynchronous (it can be processed in batches after authorization).

- Error Handling: If the card fails validation or if fraud is detected, the transaction is immediately declined, and appropriate messages are sent to the Customer.

This sequence enables secure and efficient payment processing, handling steps for both authorization and settlement, while minimizing risk through fraud detection.



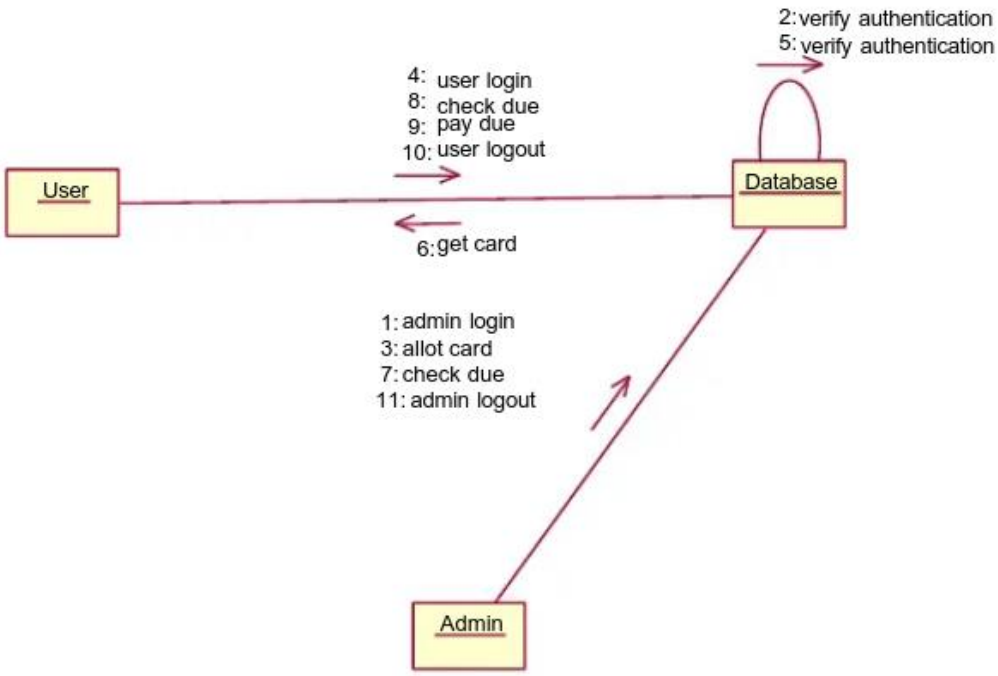
4.COLLABORATION DIAGRAM

A collaboration diagram (or communication diagram) for a credit card processing system shows how objects and components within the system interact to complete the transaction process. Here's a high-level outline of the objects and interactions that could be represented in the diagram:

Key Components in a Credit Card Processing System

1. Customer: Initiates the payment by submitting their card details.

2. Point of Sale (POS) System: Takes customer details and initiates the transaction.
3. Payment Gateway: Acts as the intermediary, transferring transaction data securely.
4. Bank (Customer's Bank): The customer's bank that will authorize or decline the transaction.
5. Merchant's Bank: Receives funds for the merchant once the transaction is approved.
6. Credit Card Network: Connects customers and merchants banks, and manages transaction authorization.

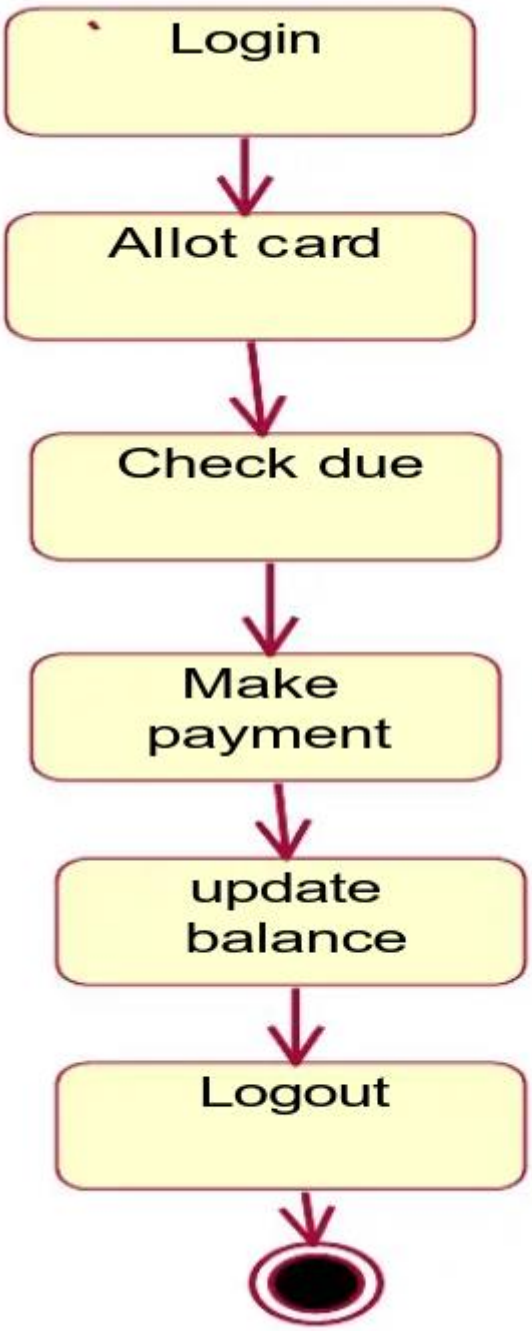


5.STATE - CHART DIAGRAM

A state chart diagram for a credit card processing system can illustrate the various states and transitions involved in handling credit card transactions. Below is a description of a typical state chart for a credit card processing system. Each of these states would have associated transitions that define the flow of a credit card transaction. This chart helps visualize the credit card processing lifecycle and highlights points of failure or success within the process.

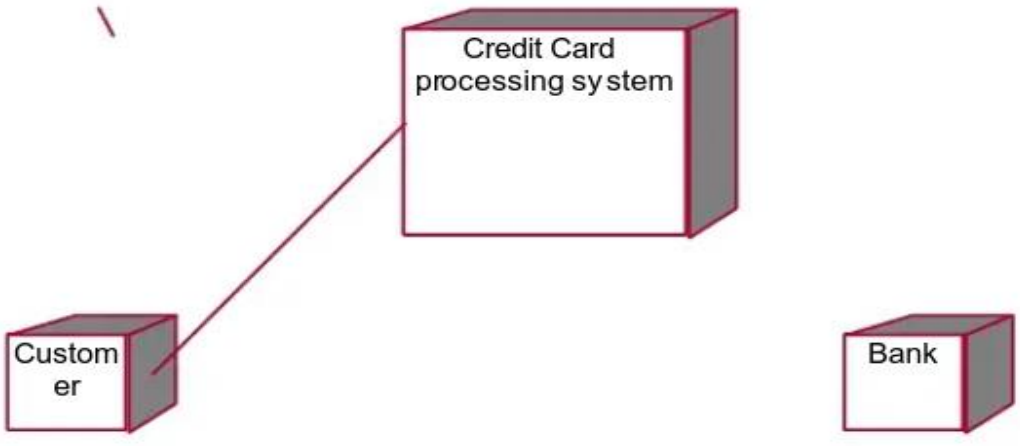
6. ACTIVITY DIAGRAM

An activity diagram for a credit card processing system visually represents the steps involved in processing a transaction, from initiating a purchase to completing or rejecting the transaction. Here’s a description of the main components typically included If there are enough funds, the bank approves the transaction; otherwise, it is denied. The bank or payment processor then notifies the merchant of the approval or denial status. Upon approval, the merchant completes the transaction, and funds are transferred from the customer's account to the merchants. If denied, the transaction is cancelled, and the customer is informed. This activity flow ensures secure, efficient transaction processing while verifying the customer's payment capability.


Login
Allot card
Check due
Make payment
update balance
Logout

7.DEPLOYMENT DIAGRAM

A deployment diagram for a credit card processing system illustrates the various hardware and software components involved in transaction processing and their interactions across different nodes. At the forefront is the Customer Device, which can be a mobile phone, computer, or POS terminal where customers initiate transactions through a web browser or mobile app. This device communicates with the Merchant Server, consisting of an application server that handles customer requests and a database server that stores transaction and customer data. The merchant server interacts with the Payment Gateway Server, which includes a transaction processor for routing requests to financial institutions and an encryption service to secure sensitive information. The Bank Server is responsible for validating card details and checking available funds through its authorization server and database server. This deployment setup creates a robust framework for securely processing credit card transactions, protecting customer information while providing efficient service.



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

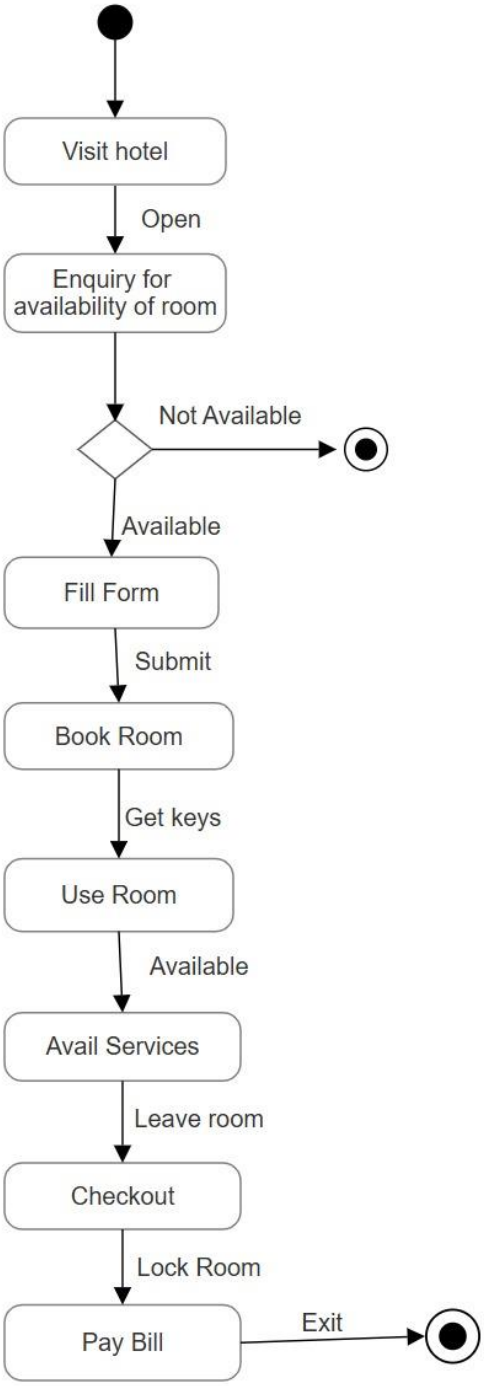
UML Diagrams:

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.

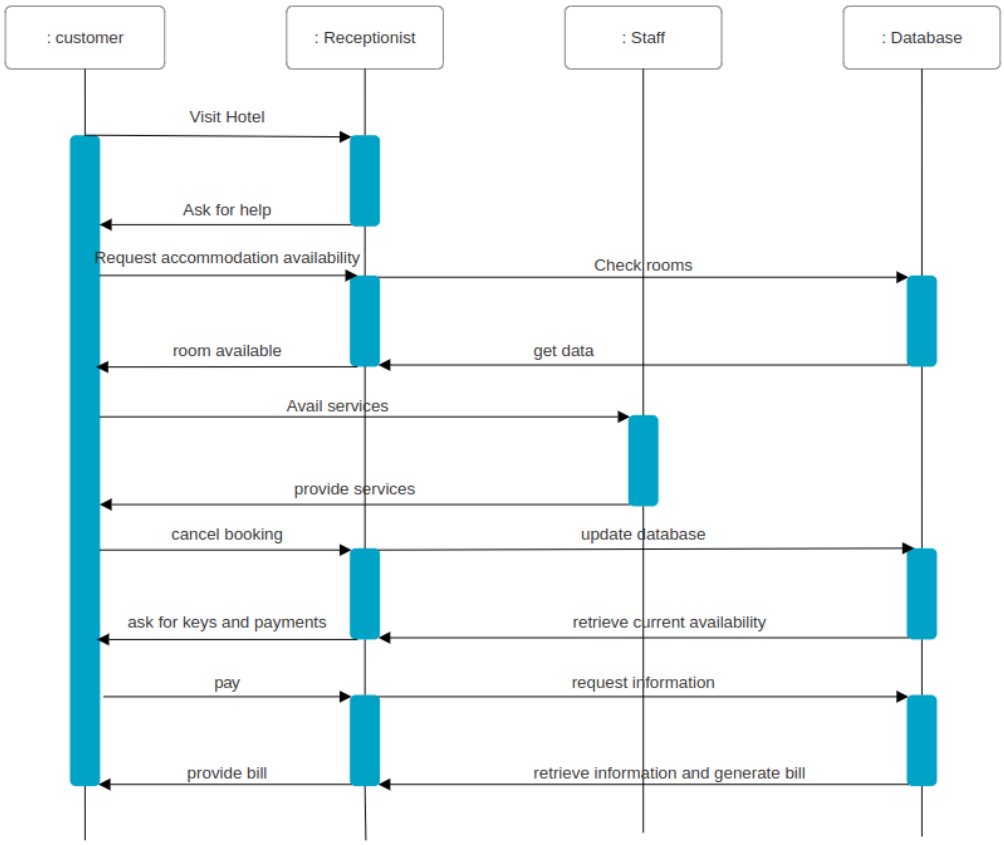
Activity diagram:

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.



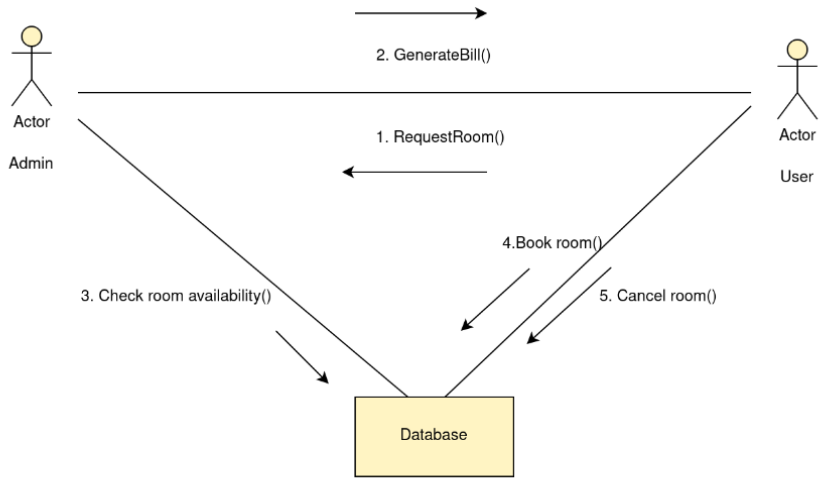
Sequence diagram:

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.



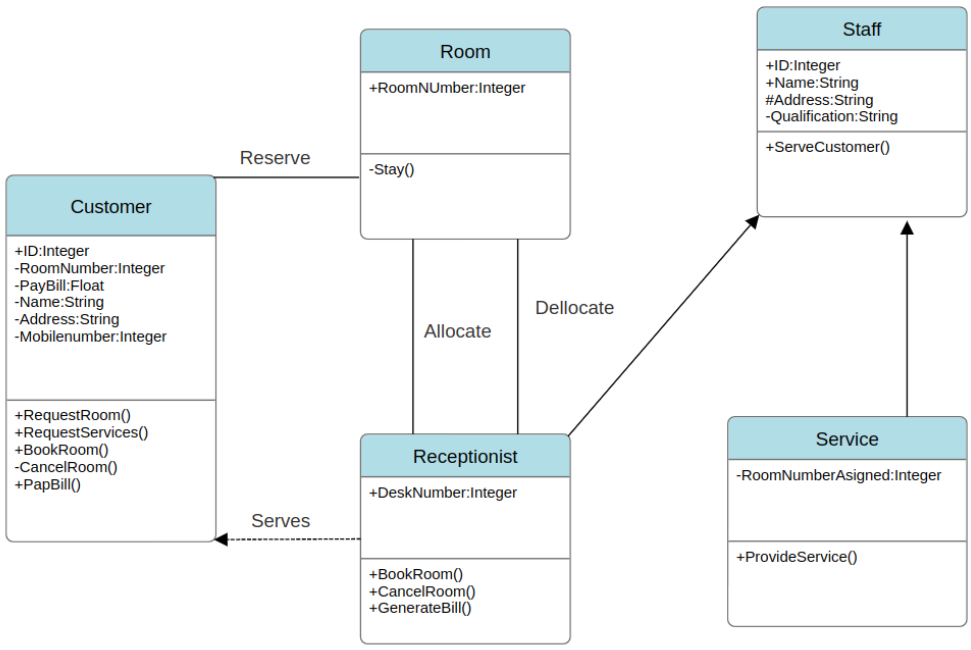
Collaboration diagram:

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:

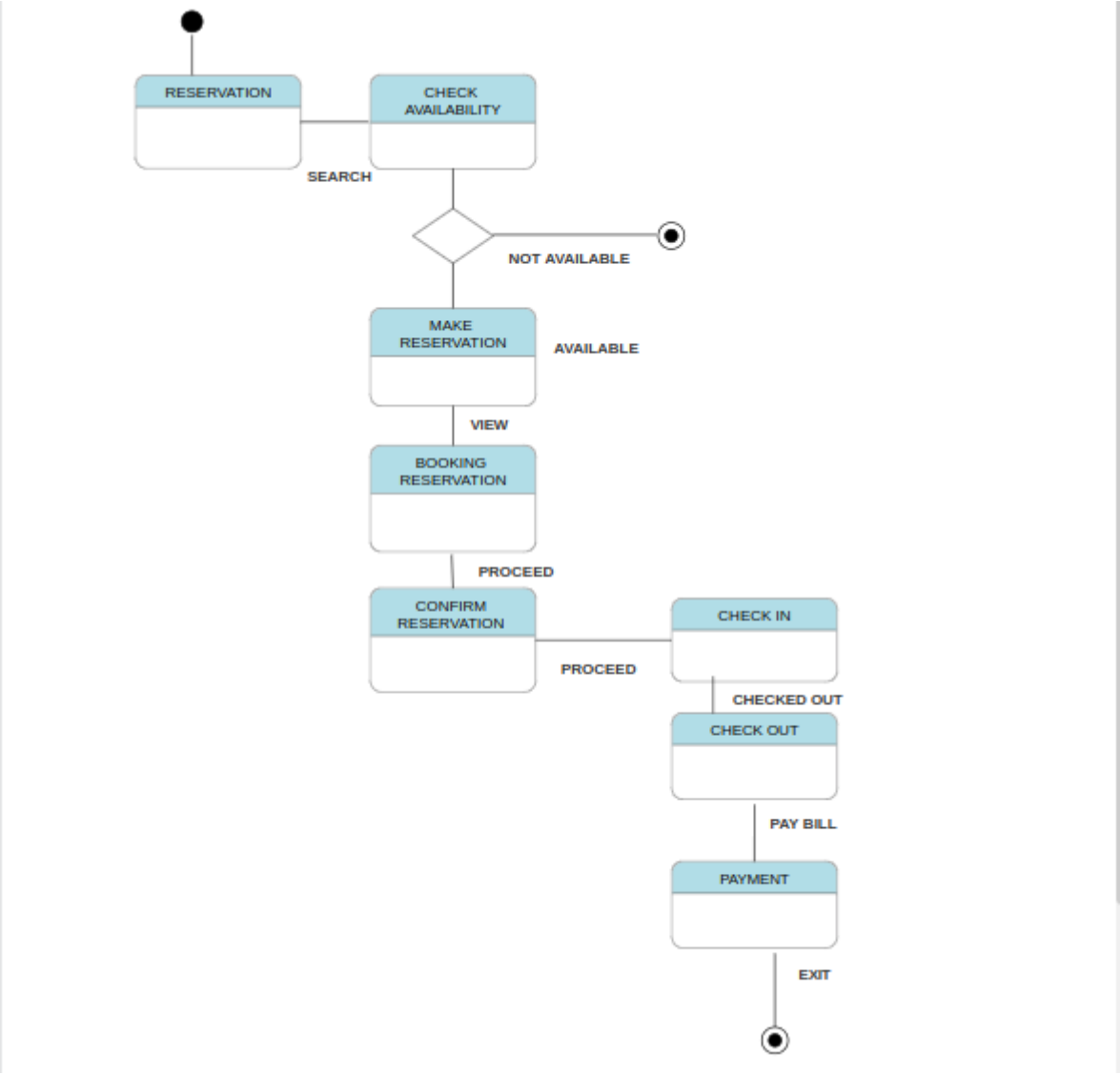
A class diagram shows the structure of a system by defining its classes, attributes, methods, and relationships. It represents real-world entities and how they interact within the system. The diagram helps visualize the system's architecture and the connections between its components. For a Hotel Management 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

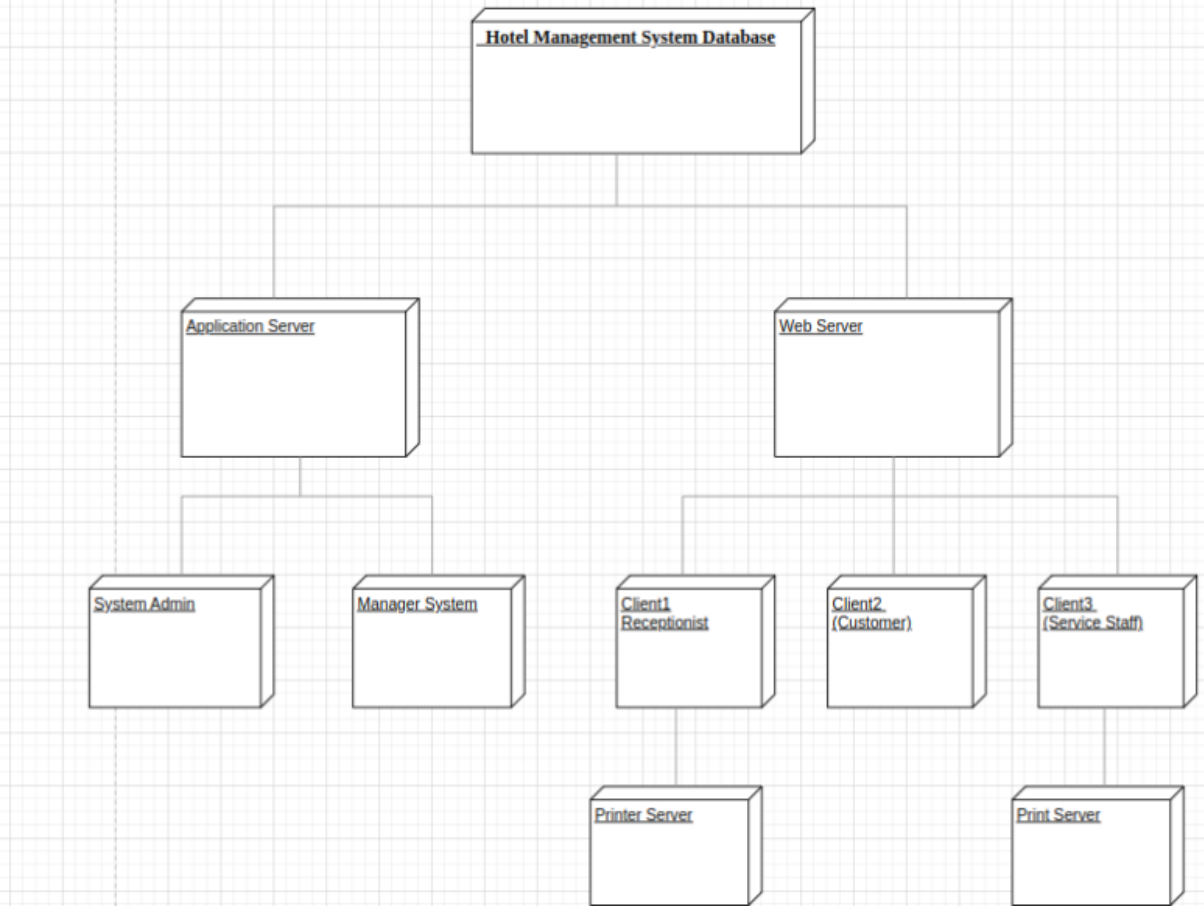
State chart diagram:

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.



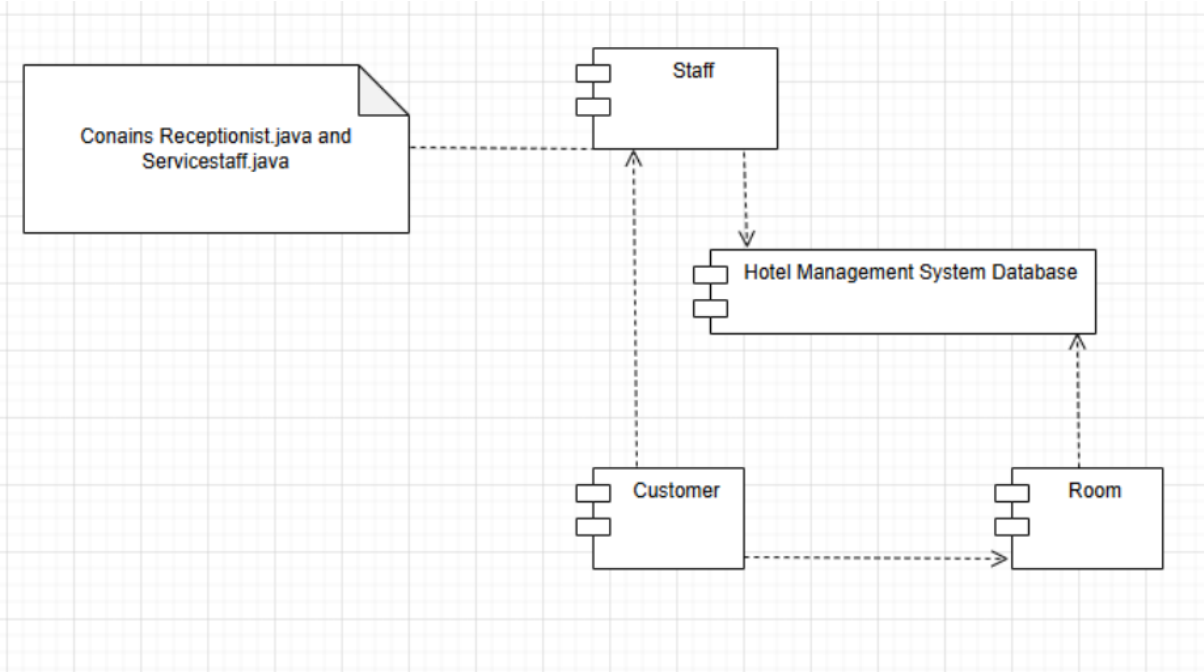
Deployment diagram:

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.



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

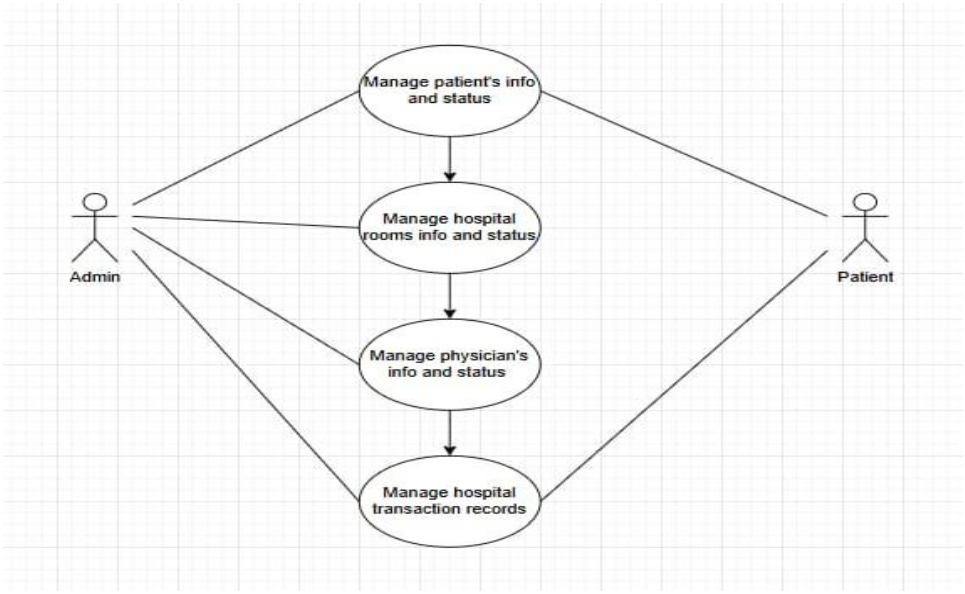
Hardware components:

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

UML Diagrams:

Use-case diagram:

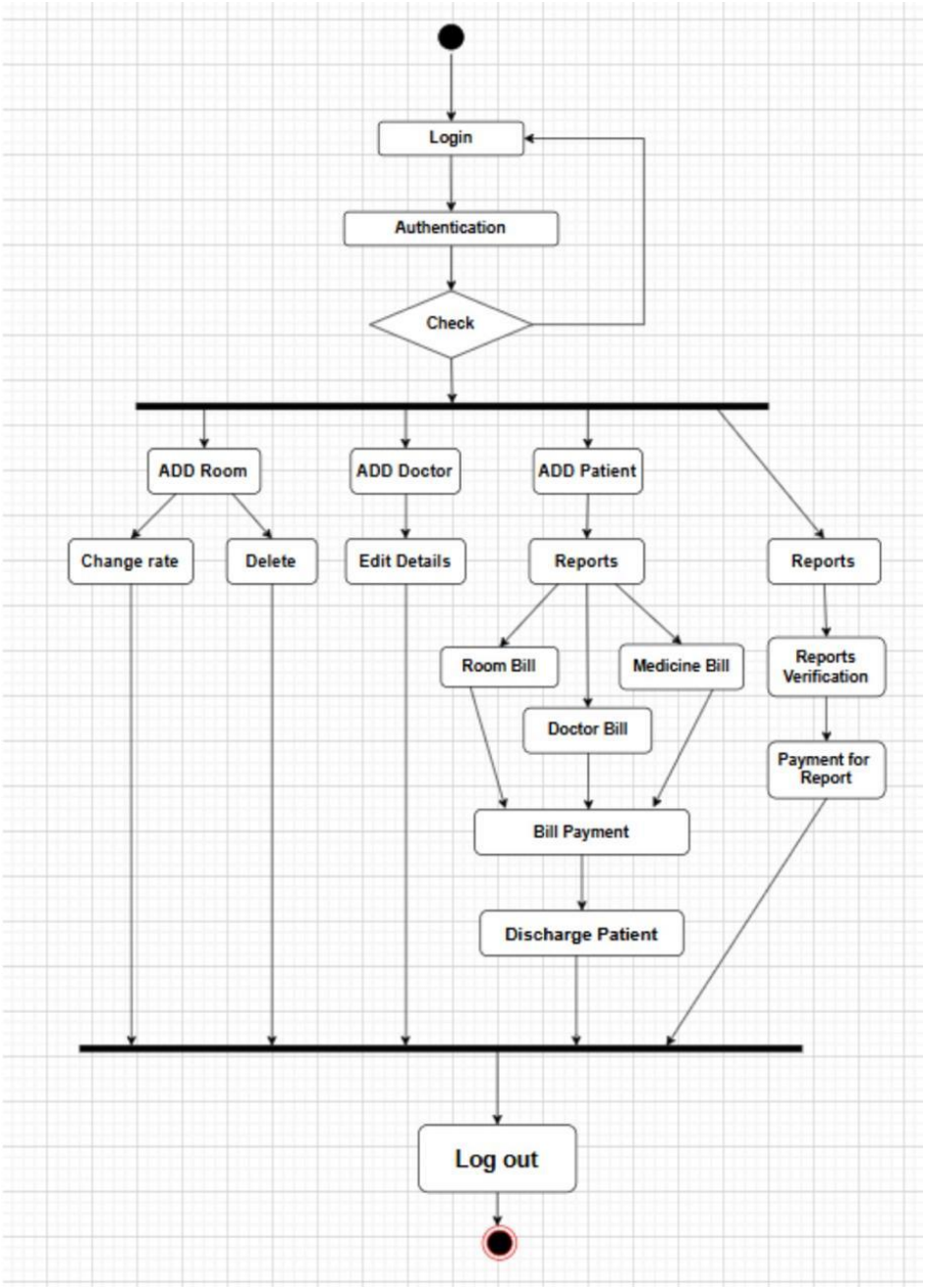
The use case in a Hospital Management System enables patients to book consultations with doctors seamlessly. The patient logs into the system, selects the desired specialization, and views available doctors and time slots. After choosing a doctor and an appropriate time, the patient confirms the appointment, 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.



Use-case diagram for hospital management system

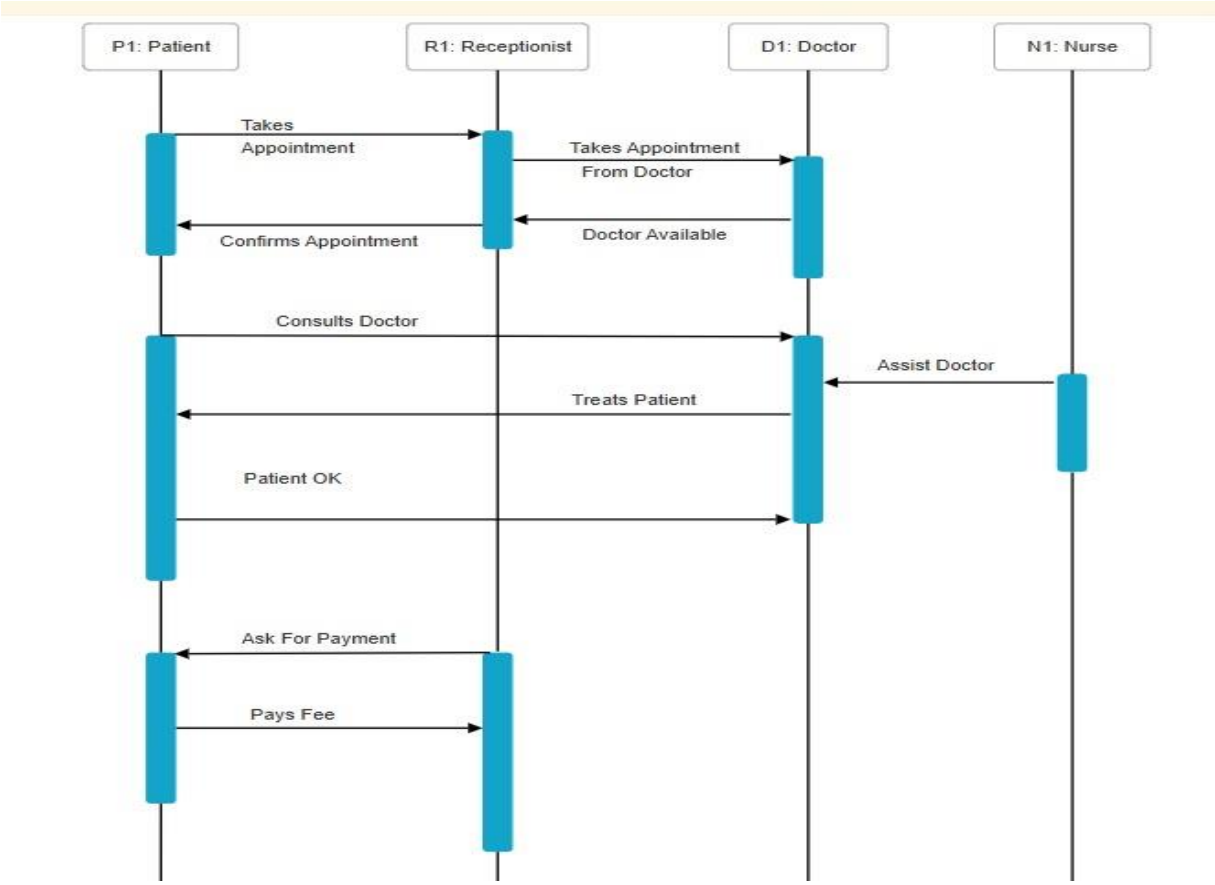
Activity diagram:

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.



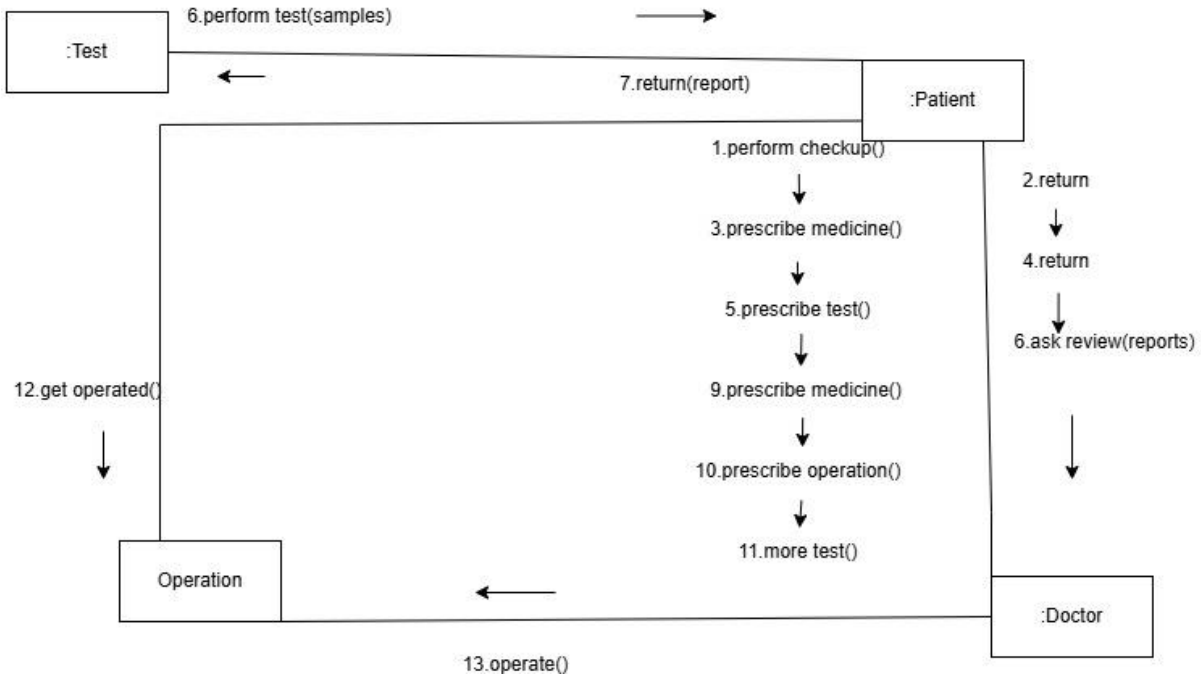
Sequence diagram:

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.



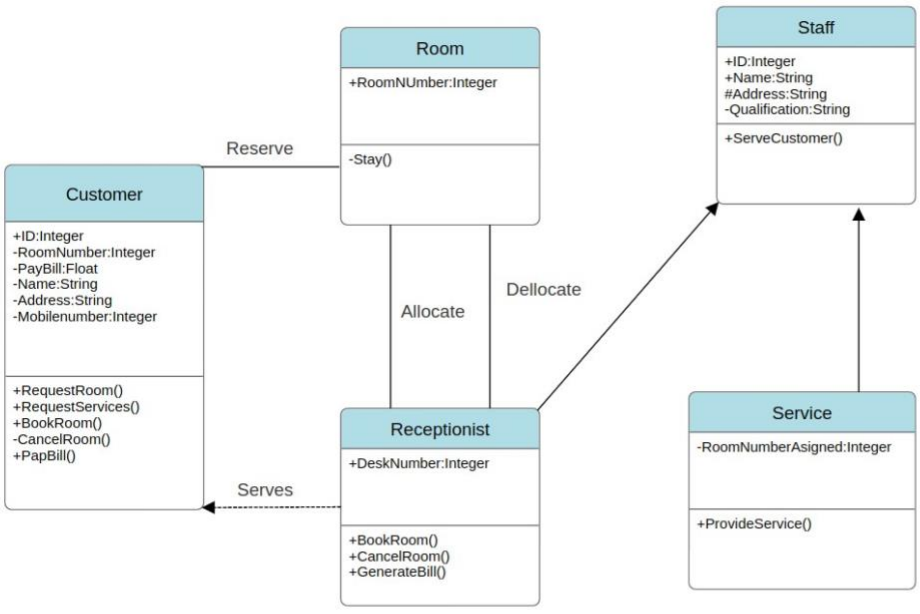
Collaboration diagram:

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:

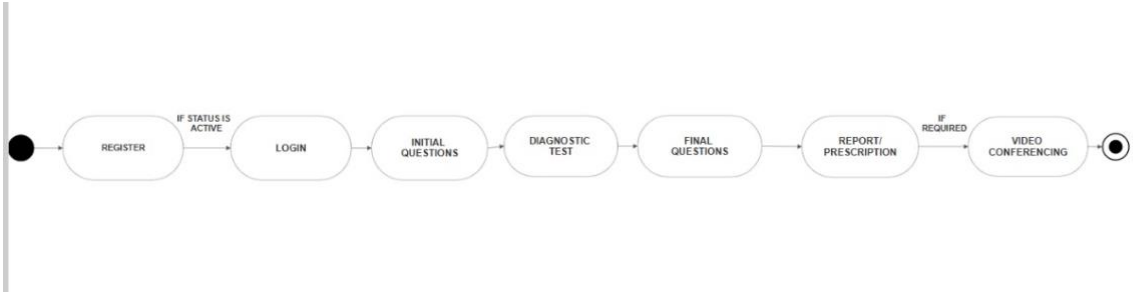
A class diagram for a Hospital Management System defines the structure of the system by representing key classes, their attributes, and methods. It includes classes such as Patient, Doctor, Appointment, Billing, and Inventory, with relationships like associations, inheritances, and dependencies between 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

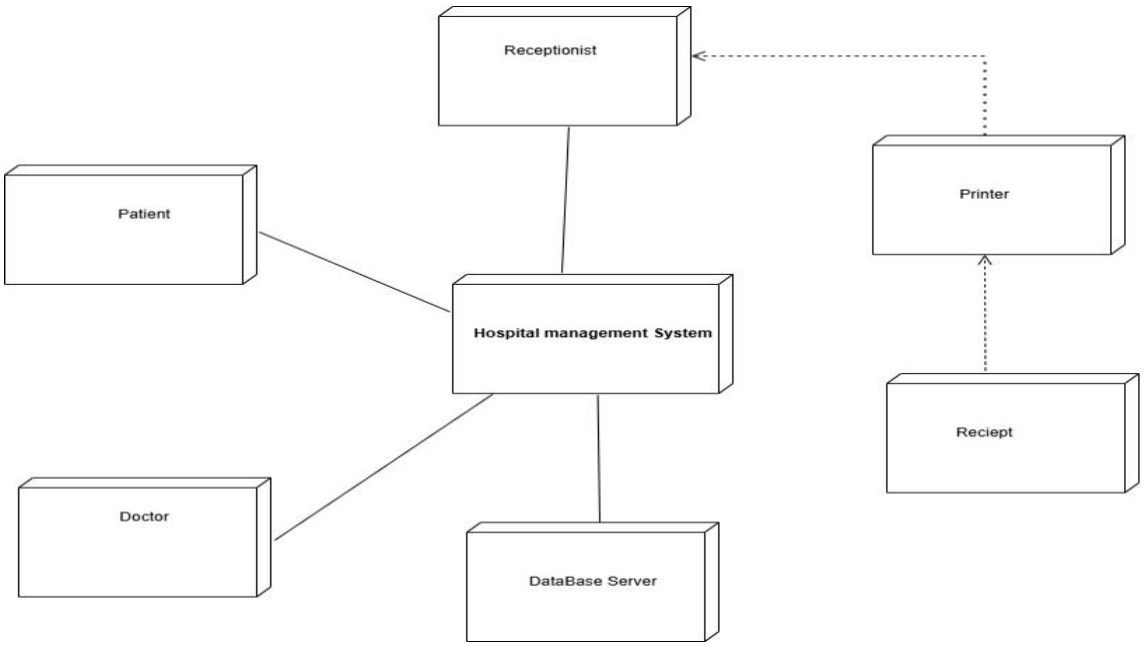
State chart diagram:

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..



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..