Experiment 1

Aim: Write down the problem statement for a suggested system of relevance.

Hospital Management System

A comprehensive Hospital Management System (HMS) needs to be developed to streamline and automate various manual processes within a healthcare facility. The system should be designed to provide the following key functionalities:

1. **Patient Management:**
   - Registration: Patients should be able to register efficiently, capturing essential information like name, contact details, and medical history.
   - Appointment Scheduling: A user-friendly interface for both staff and patients to schedule appointments, ensuring optimal utilization of resources.
   - Medical Records: Securely maintain and manage patient records, including medical history, test results, and prescribed medications.
2. **Billing and Insurance:**
   - Generate and manage bills for medical services rendered.
   - Integration with insurance providers to facilitate seamless claims processing.
3. **Staff Management:**
   - Maintain a centralized database of staff details, including doctors, nurses, and administrative personnel.
   - Schedule and manage staff shifts efficiently.
4. **Inventory Management:**
   - Keep track of medical supplies, ensuring timely restocking to avoid shortages.
   - Integration with vendors for streamlined procurement processes.
5. **Appointment Reminders:**
   - Automated reminders for upcoming appointments through SMS or email.
6. **Reports and Analytics:**
   - Generate various reports, including patient demographics, revenue summaries, and resource utilization statistics.
   - Analytics tools for data-driven decision-making and performance monitoring.
7. **User Authentication:**
   - Secure login for different users with role-based access control.

8. **Emergency Handling:**
    - Protocols for handling emergency situations, ensuring swift response and patient care.

In conclusion, this problem statement lays the foundation for the development of a Hospital Management System, addressing the diverse needs of both patients and healthcare providers. Following the outlined steps ensures a clear understanding of the project's objectives among all stakeholders.

Experiment 2

Aim: Software Requirement Specification Sheet (SRS) for Hospital Management System

SOFTWARE REQUIREMENTS SPECIFICATION

HMS

VERSION 1.0

SEPTEMBER 12,2023

HOSPITAL MANAGEMENT SYSTEM

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to comprehensively define the functional and non-functional requirements of the Hospital Management System (HMS). It serves as a foundational document that provides clear and detailed guidance to software developers, testers, project managers, and all stakeholders involved in the design, development, and implementation of the HMS.

1.2 Scope

The scope of the HMS encompasses the development of a robust and user-friendly software application tailored for Hospital management. The HMS aims to automate and streamline various Hospital-related processes, facilitating efficient and error-free management of student accommodations, fees, inventory, and reporting.

1.3 Definitions, Acronyms, and Abbreviations

- HMS (Hospital Management System): The software application designed to automate and streamline Hospital management processes.
- SRS (Software Requirements Specification): A document that outlines the functional and non-functional requirements of the HMS.
- UI (User Interface): The graphical or command-line interface through which users interact with the system.
- DBMS (Database Management System): Software used to manage the storage and retrieval of data in the system.
- API (Application Programming Interface): A set of rules and protocols that allows different software applications to communicate with each other.

- GDPR (General Data Protection Regulation): European Union data protection and privacy regulation.

1.4 Document Conventions

Throughout this SRS document, the following conventions will be adhered to for clarity and consistency:

- Section Structure: Sections and subsections will be organized using numbered headings (e.g., "2. Overall Description," "3. Specific Requirements") for easy navigation and reference.
- Appendices: Appendices will be used to provide additional information or reference materials and will be labelled as "Appendix A”.
- Emphasis: Key terms or phrases will be italicized (e.g., User Registration) for emphasis.
- Lists: Bulleted and numbered lists will be used for presenting items and requirements in a structured manner.

1.5 References

This SRS document may refer to the following documents, standards, or external sources:

- studylib.net
- slideshare.net

1.6 Overview

The Hospital Management System (HMS) is designed to address the complexities and challenges associated with Hospital administration and management. By offering a centralized and automated platform, the HMS aims to enhance the efficiency, accuracy, and transparency of Hospital-related operations.

This SRS document presents a detailed account of the requirements that the HMS must fulfil. It outlines the functionalities that the system will offer, the interactions it will support, and the quality attributes it must adhere to. Additionally, this document provides a comprehensive understanding of the system's context, including its users, constraints, assumptions, and dependencies.

The subsequent sections of this SRS will delve into specific aspects of the HMS, including functional requirements, non-functional requirements, system features, and external interfaces. It will serve as a reference point for all project stakeholders, guiding the development team in the creation of a reliable and user-friendly Hospital Management System that meets the needs of both Hospital staff and students.

2. Overall Description

2.1 Product Perspective

The Hospital Management System (HMS) is a stand-alone software application that integrates seamlessly into the existing Hospital management infrastructure. It interacts with other systems, such as the database management system (DBMS) for data storage, payment gateways for fee processing, and external services for communication.

2.2 Product Functions

The HMS is designed to automate and streamline various Hospital management tasks, enhancing efficiency and accuracy. Its core functions include:

2.2.1 User Registration

Description: Users, including Hospital staff and students, can register their profiles in the system, providing essential information such as personal details and roles.

2.2.2 Room Allocation

Description: Hospital staff can allocate rooms to registered students based on their preferences and availability, ensuring efficient room management.

2.2.3 Fee Management

Description: The system manages student fees, including fee calculations, payment processing, and automated reminders for overdue payments.

2.2.4 Check-In and Check-Out

Description: The HMS facilitates the check-in and check-out processes for students, providing a smooth transition into and out of Hospital accommodation.

2.2.5 Inventory Management

Description: Hospital staff can efficiently track and manage Hospital inventory, ensuring timely restocking of essential items.

2.2.6 Reporting

Description: The system generates comprehensive reports for administrators, allowing them to monitor Hospital occupancy, financials, and inventory.

2.2.7 Security and Access Control

Description: Robust security measures are implemented to safeguard user data and control access to the system based on user roles.

2.3 User Characteristics

The HMS caters to three main types of users, each with specific roles and responsibilities:

- Administrator: Administrators have full access to system features and are responsible for overseeing Hospital operations.
- Hospital Staff: Hospital staff members manage room allocations, fees, inventory, and other day-to-day operations.
- Students: Students use the system to check room availability, make payments, and interact with the Hospital management process.

2.4 Design/implementation constraints

Description: The HMS operates within the following constraints:

- The system must be accessible via standard web browsers (e.g., Chrome, Firefox, Safari).
- It must be compatible with the latest versions of these web browsers.
- Compliance with data security and privacy regulations, such as GDPR, is mandatory.

2.5 Assumptions and Dependencies

Description: Assumptions made during the development of the HMS and its dependencies include:

- The availability of suitable hosting infrastructure and resources.
- Integration with external payment gateways for secure transactions.
- Availability of personnel responsible for system maintenance and support.

3. External Interface Requirements

3.1User Interfaces

3.1.1 User Registration Interface

Description: The user registration interface allows Hospital staff and students to create accounts.

Inputs:

- User details, including name, username, password, email, and role.

Processing:

- Validates user inputs for data accuracy and security.
- Registers users by storing their details in the database.

Outputs:

- Confirmation message upon successful registration.
- Appropriate error messages if registration fails (e.g., duplicate username, invalid email).

3.1.2 Dashboard Interface

Description: The dashboard serves as the main interface for users to access and manage Hospital-related tasks.

Inputs:

- User credentials for authentication.
- User interactions to perform tasks such as room allocation, fee management, check-in/out, and inventory management.

Processing:

- Authenticates users based on provided credentials.
- Provides access to specific functionalities based on user roles (e.g., student, staff, administrator).

Outputs:

- Display of relevant data and functionality options based on user role.

3.1.3 Room Allocation Interface

Description: This interface allows staff to allocate rooms to students.

Inputs:

- Student details and room preferences.

Processing:

- Checks room availability.
- Suggests suitable rooms based on preferences.
- Assigns a room to the student and updates room status.

Outputs:

- Confirmation of room allocation.
- Notification if no suitable rooms are available.

3.2Hardware Interfaces

3.2.1Database Server

Description: The HMS interacts with a database server to store and retrieve data.

Inputs:

- SQL queries for data retrieval and modification.

Processing:

- Executes SQL queries to fetch or update data.

Outputs:

- Retrieved data or confirmation of data modification.

3.2.2 Payment Gateway Integration

Description: The system communicates with payment gateways for processing fee payments.

Inputs:

- Payment details provided by students.

Processing:

- Sends payment information to the payment gateway.
- Receives payment status (success or failure) from the gateway.

Outputs:

- Confirmation of successful payment or error message in case of payment failure.

3.3 Software Interfaces

3.3.1 Database Management System (DBMS)

Description: The system relies on a DBMS to manage the database.

Inputs:

- SQL queries for database operations (e.g., SELECT, INSERT, UPDATE).

Processing:

- The DBMS executes SQL queries and manages data storage.

Outputs:

- Data retrieved or modified based on SQL queries.

3.3.2 Payment Gateway Integration APIs

Description: The HMS integrates with payment gateway APIs for processing online fee payments.

Inputs:

- Payment request data, including payment amount and student information.

Processing:

- Communicates with the payment gateway APIs using secure protocols (e.g., HTTPS).
- Receives and processes payment responses from the gateway.

Outputs:

- Payment confirmation or failure status received from the payment gateway.

4. System Features

4.1 User Registration

Description: This feature allows users (Hospital staff and students) to create accounts with their personal details.

Use Cases:

- **User Registration:** Users provide their first name, last name, username, password, email, and role during registration.
- **Validation:** Input data is validated to ensure data integrity and security.
- **Data Storage:** Valid user details are stored in the database.

Outputs:

- Users receive a confirmation message upon successful registration.
- In case of registration failure (e.g., due to duplicate username or invalid email), users are informed with appropriate error messages.

4.2 Room Allocation

Description: This feature allows Hospital staff to assign rooms to registered students based on their preferences and availability.

Use Cases:

- **Room Allocation Request:** Staff initiates room allocation by selecting a student and specifying their preferences (e.g., room type).
- **Availability Check:** The system checks room availability and suggests suitable rooms.
- **Room Assignment:** Staff assigns a room to the student, and the room status is updated (e.g., from "Vacant" to "Occupied").

Outputs:

- Confirmation of room allocation is sent to the student.
- If no suitable rooms are available, the system informs staff and the student.

4.3 Fee Management

Description: This feature manages student fees, including tracking payments, generating invoices, and sending reminders.

Use Cases:

- **Fee Calculation:** The system calculates fees based on predefined criteria (e.g., room type, duration of stay).
- **Fee Payment:** Students make payments using integrated payment gateways.
- **Payment Tracking:** The system tracks payments and updates the student's fee status.
- **Overdue Notifications:** Automatic reminders are sent to students for overdue payments.

Outputs:

- Payment confirmations are sent to students.
- Overdue notifications are sent with details on outstanding fees.

4.4 Check-In and Check-Out

Description: This feature facilitates the check-in and check-out processes for students.

Use Cases:

- **Check-In:** Students provide their ID and room details upon arrival. Staff verifies room availability and records the check-in.
- **Check-Out:** Students request check-out, and staff verifies room condition, updates the status, and records the check-out.

Outputs:

- Check-in and check-out confirmations are provided to students.
- Room status is updated to reflect occupancy changes.

4.5 Inventory Management

Description: This feature tracks Hospital inventory and sends restocking alerts when necessary.

Use Cases:

- **Inventory Updates:** Staff record inventory changes, including additions and deductions.
- **Inventory Status:** The system maintains up-to-date inventory status.
- **Restocking Alerts:** When inventory levels fall below a certain threshold, the system sends restocking alerts to staff.

Outputs:

- Inventory status reports are available for staff.
- Restocking alerts are sent via notifications or emails.

4.6 Reporting

Description: This feature generates various reports for administrators to monitor Hospital occupancy, financials, and inventory.

Use Cases:

- **Generate Reports:** Administrators can select report types (e.g., occupancy report, financial report) and specify date ranges.

- **Data Retrieval:** The system retrieves relevant data, generates reports in various formats (e.g., PDF, Excel).

Outputs:

- Reports are provided to administrators, allowing them to make informed decisions about Hospital management.

4.7 Security and Access Control

Description: This feature ensures the security of user data and controls user access to the system.

Use Cases:

- **Authentication:** Users are required to log in with their usernames and securely hashed passwords.
- **Authorization:** User roles determine their access levels (e.g., student, staff, administrator).
- **Access Control:** Unauthorized access attempts are blocked, and access is limited to specific functionalities based on user roles.

Outputs:

- Users receive access denied messages if they attempt unauthorized actions.
- User sessions are secured with authentication tokens.

5. Other Non-Functional Requirements

5.1 Performance

5.1.1 Response Time

Description: The system should provide timely responses to user interactions, ensuring a smooth and responsive user experience.

Requirement:

- Critical functions (e.g., user authentication, room allocation) should have a response time of less than 2 seconds.
- Non-critical functions (e.g., generating reports) should provide responses within 5 seconds.

5.1.2 Scalability

Description: The system should be scalable to accommodate potential growth in the number of users and data volume.

Requirement:

- The system should handle a minimum of 100 concurrent users without significant performance degradation.
- Scalability planning should allow for future expansion of the database and infrastructure to support increased load.

5.2 Usability

5.2.1 User Interface

Description: The user interface (UI) should be intuitive and user-friendly to minimize user training requirements.

Requirement:

- The UI should follow standard design conventions, making it easy for users to navigate and perform tasks.
- User training materials, including documentation and tutorials, should be provided to aid users in using the system effectively.

5.3 Reliability

5.3.1 Availability

Description: The system should be available to users with minimal downtime for maintenance or unexpected failures.

Requirement:

- The system should aim for 99.9% uptime, excluding scheduled maintenance windows.
- Maintenance windows, if required, should be scheduled during off-peak hours to minimize disruption.

5.4 Security

5.4.1 Data Security

Description: The system should ensure the confidentiality, integrity, and availability of user data.

Requirement:

- User data should be encrypted during transmission and storage using industry-standard encryption protocols.
- Passwords should be securely hashed and stored.
- Access to sensitive data should be restricted based on user roles, following the principle of least privilege.

5.5 Compatibility

5.5.1 Browser Compatibility

Description: The system should be compatible with a range of web browsers to accommodate user preferences.

Requirement:

- The system should support the latest versions of major web browsers, including Chrome, Firefox, Safari, and Edge.
- Browser-specific issues should be addressed promptly to ensure consistent functionality.

5.6 Data Backup and Recovery

5.6.1 Regular Backups

Description: Regular backups of the system data should be performed to prevent data loss in case of hardware failures or other emergencies.

Requirement:

- Daily backups of the database should be scheduled, with backup data stored securely off-site.
- A backup retention policy should be established to manage backup data.

5.7 Compliance

5.7.1 Regulatory Compliance

Description: The system should comply with relevant data protection and privacy regulations.

Requirement:

- The system should adhere to data protection regulations such as GDPR, including providing mechanisms for user consent and data deletion requests.
- Regular audits and compliance checks should be conducted to ensure ongoing adherence to regulations.

6. Other Requirements

Appendix A: Terminology/ Glossary / Definition List

A.1 Terms and Definitions

- **HMS (Hospital Management System):** The software application designed to automate and streamline Hospital management processes.
- **SRS (Software Requirements Specification):** A document that outlines the functional and non-functional requirements of the HMS.
- **UI (User Interface):** The graphical or command-line interface through which users interact with the system.
- **DBMS (Database Management System):** Software used to manage the storage and retrieval of data in the system.
- **API (Application Programming Interface):** A set of rules and protocols that allows different software applications to communicate with each other.
- **GDPR (General Data Protection Regulation):** European Union data protection and privacy regulation.

A.2 Acronyms and Abbreviations

- **UI:** User Interface
- **DBMS:** Database Management System
- **API:** Application Programming Interface
- **GDPR:** General Data Protection Regulation
- **HTTPS:** Hypertext Transfer Protocol Secure

Experiment 3

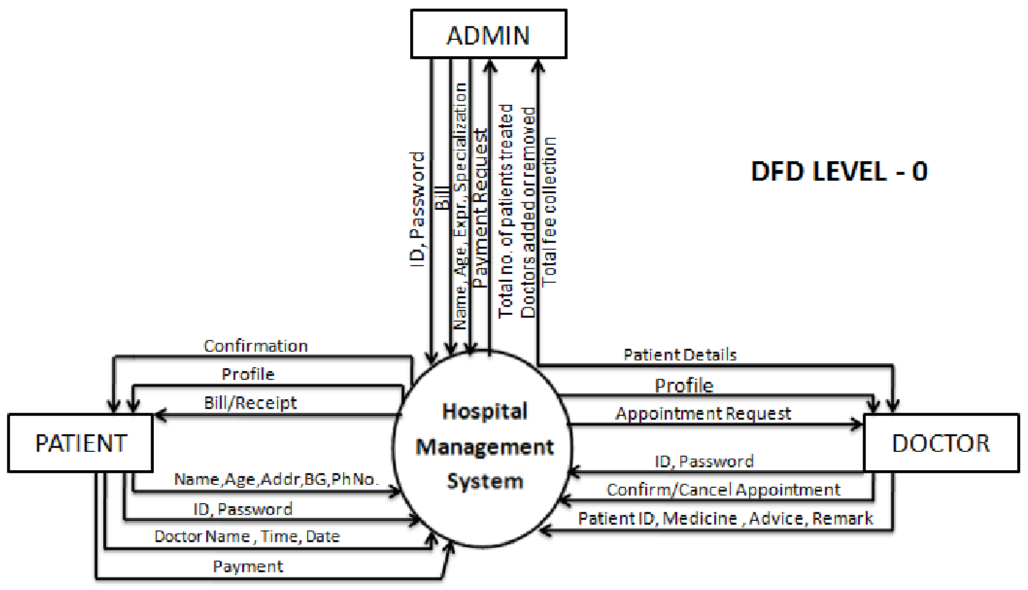
Aim: To draw DATA FLOW DIAGRAM (DFD) at Level 0 and Level 1.

Output:

HOSPITAL MANAGEMENT SYSTEM

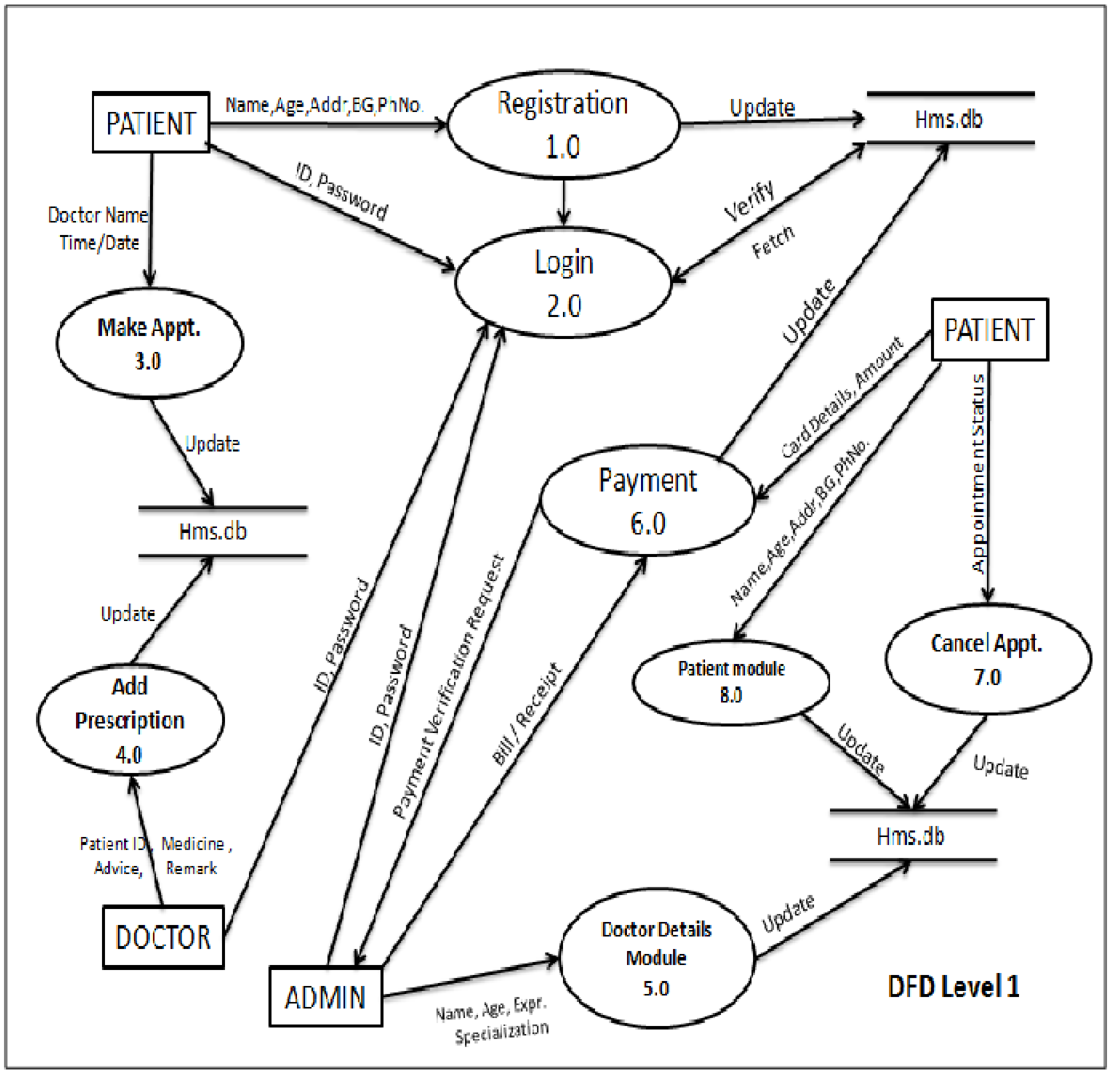
DFD LEVEL 0:

Level 0 DFDs, also known as context diagrams, are the most basic data flow diagrams. They provide a broad view that is easily digestible but offers little detail.



DFD LEVEL 1:

A level 1 DFD notates each of the main sub-processes that together form the complete system. We can think of a level 1 DFD as an “exploded view” of the context diagram.

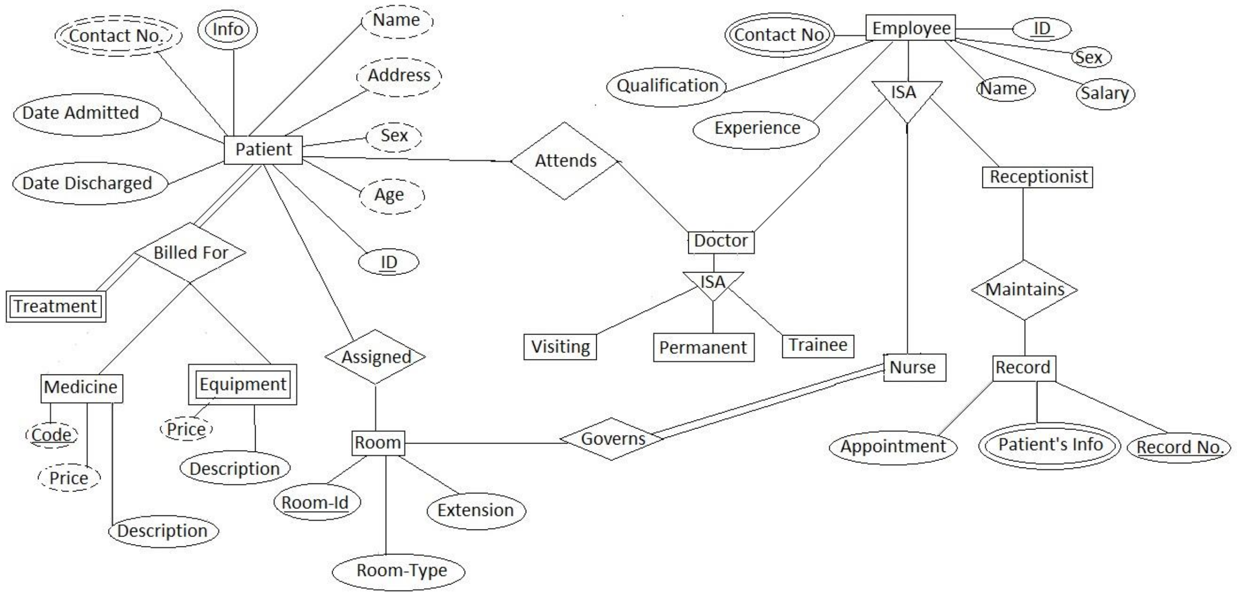


EXPERIMENT – 4

AIM: To draw ER Diagram

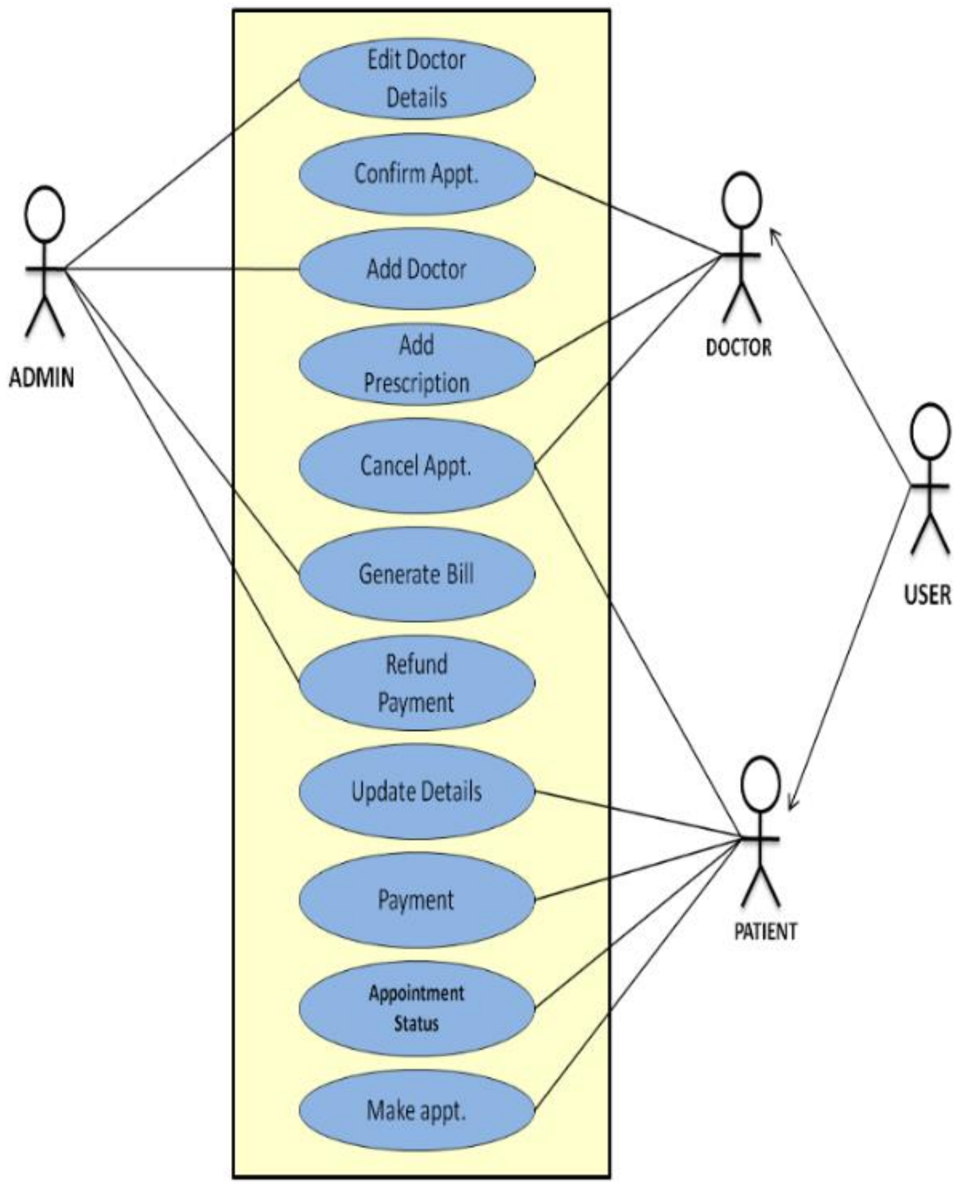
Output:

HOSPITAL MANAGEMENT SYSTEM



EXPERIMENT – 5

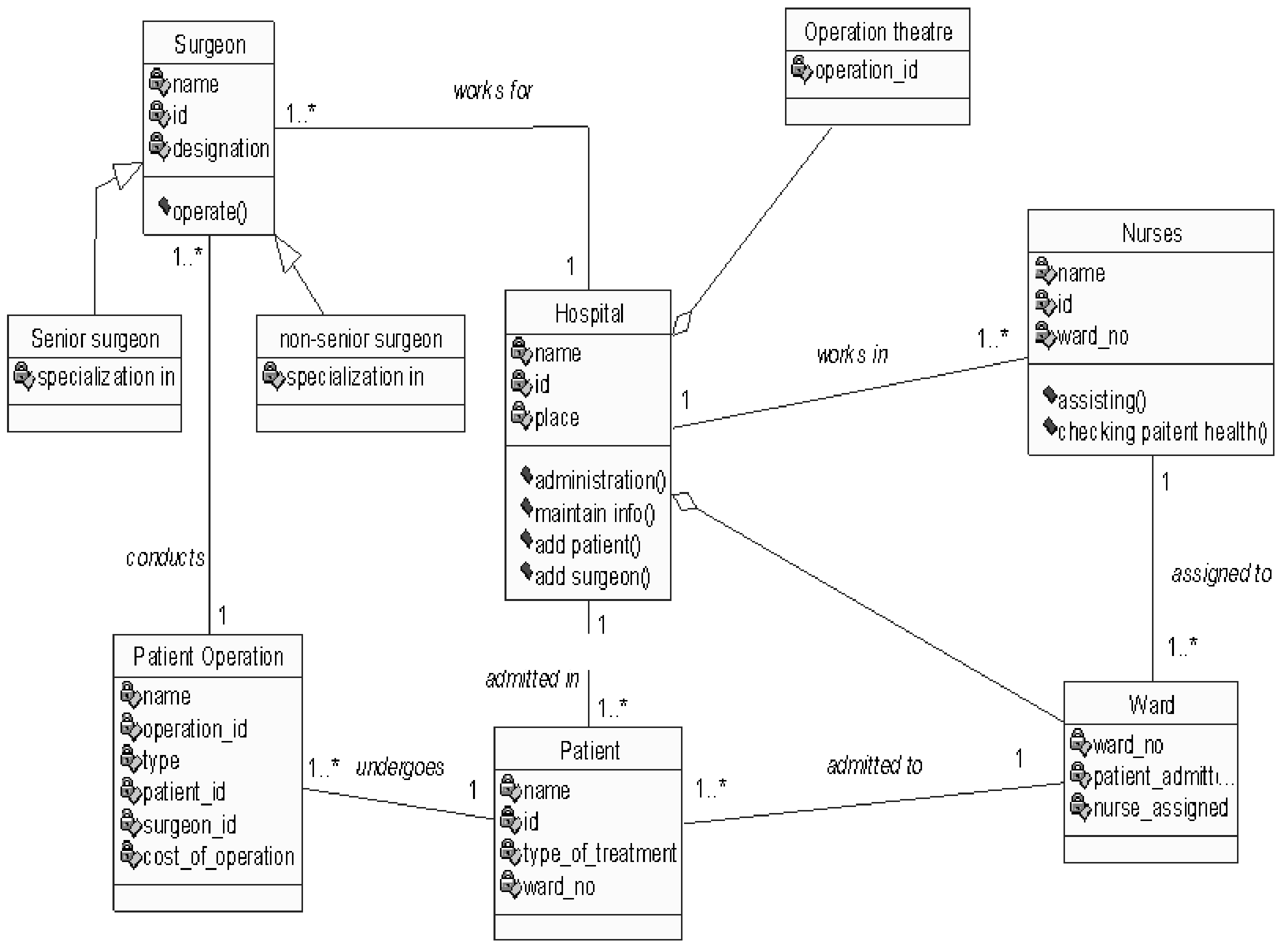
Aim: To draw the structural view diagram for the system: Class diagram, object diagram.



Experiment 6

Aim: To draw the structural view diagram for the system: Class diagram, object diagram.

OUTPUT:

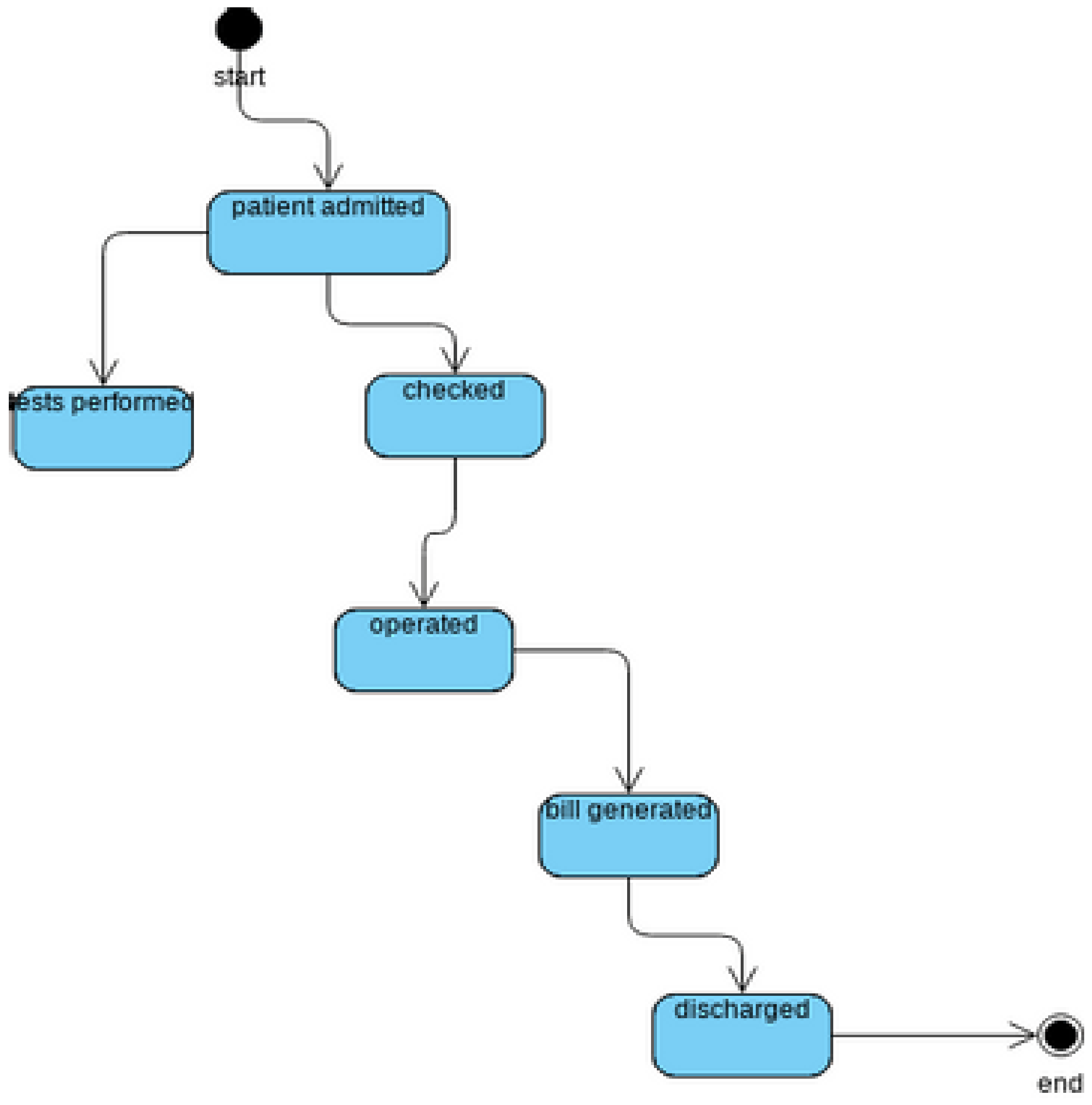


Experiment 7

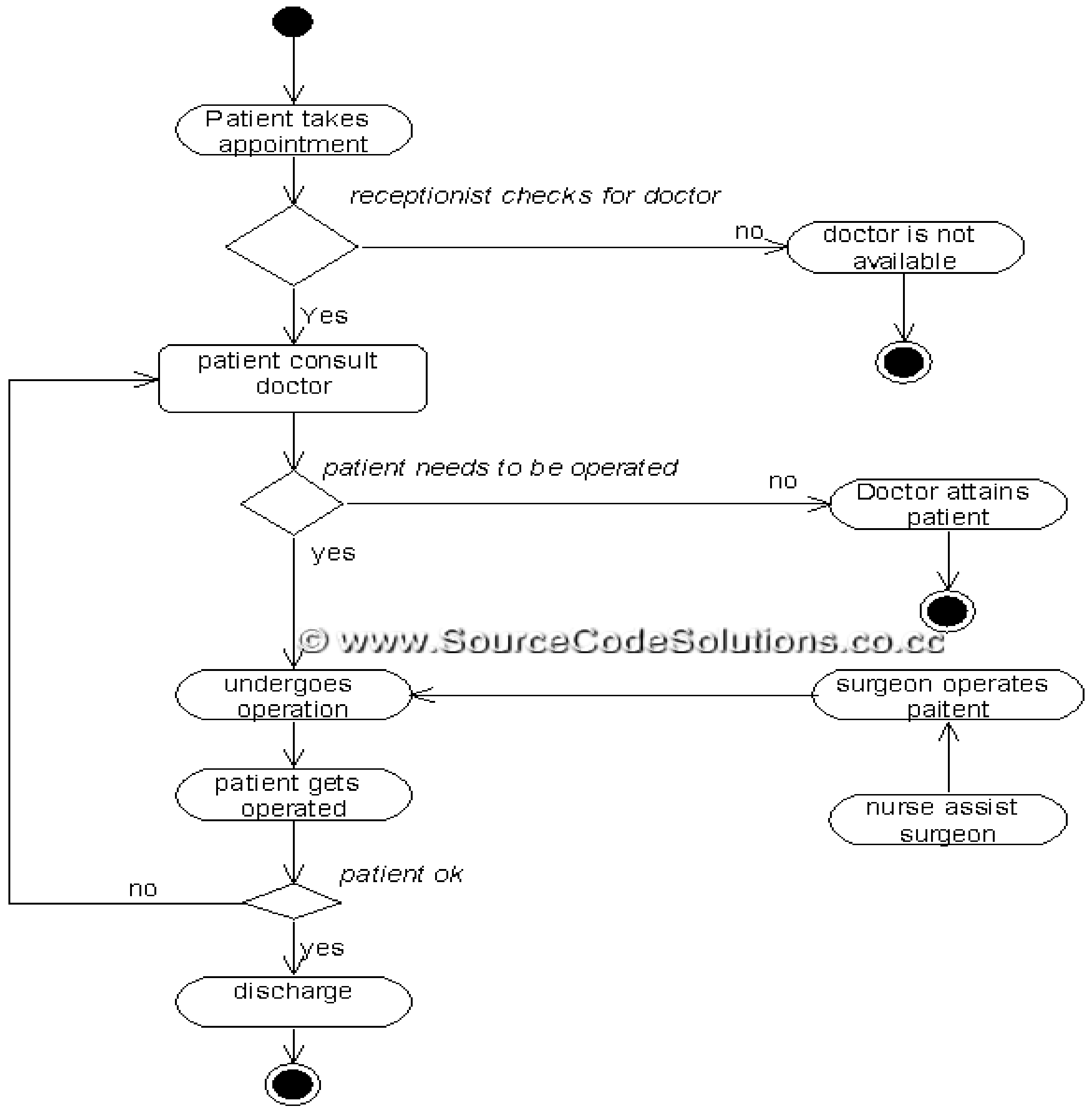
Aim: To draw the behavioral view diagram: State-chart diagram, Activity diagram.

Output:

State Chart Diagram



Activity diagram

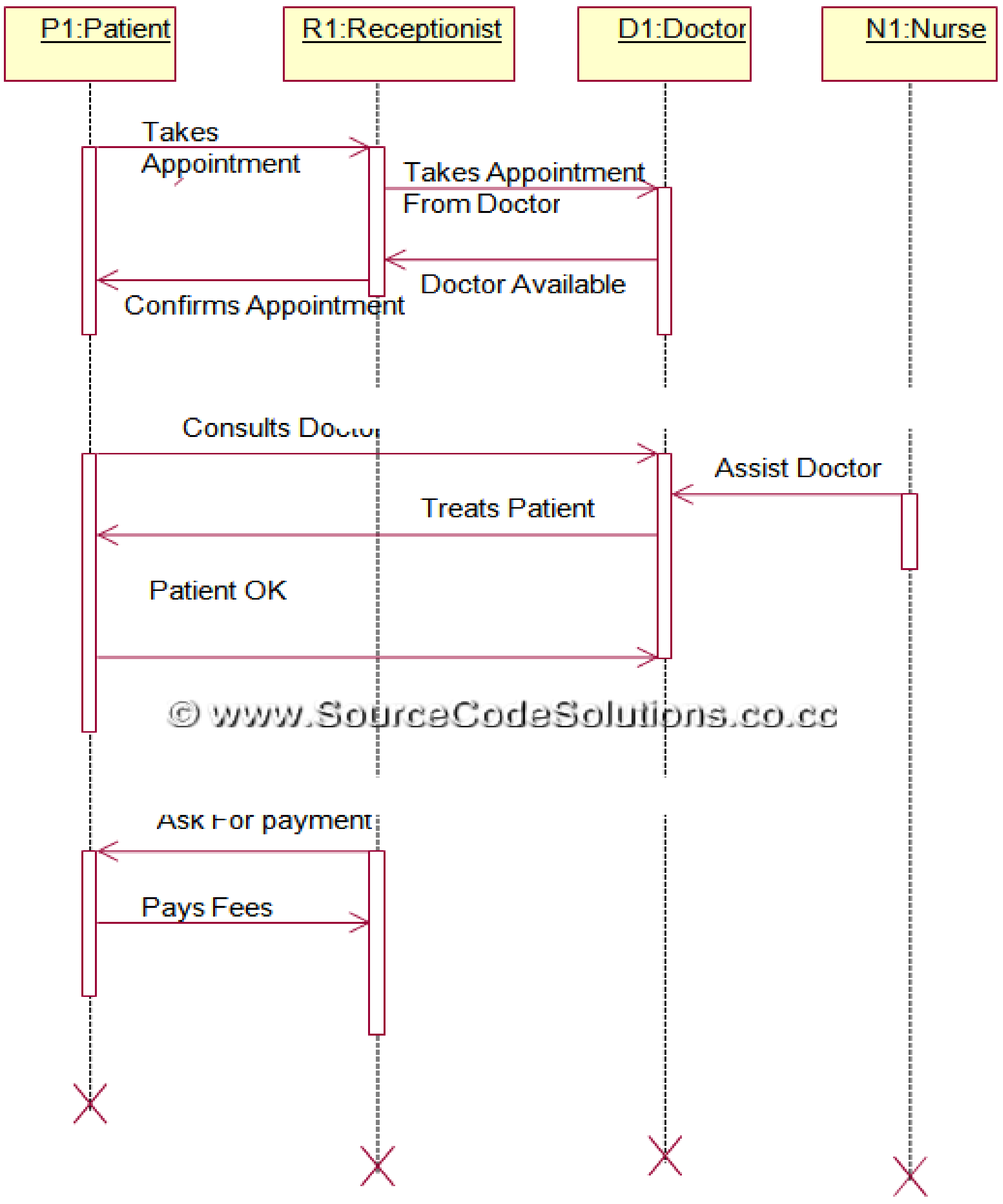


Conclusion: State Chart and Activity diagram were made successfully by following above step

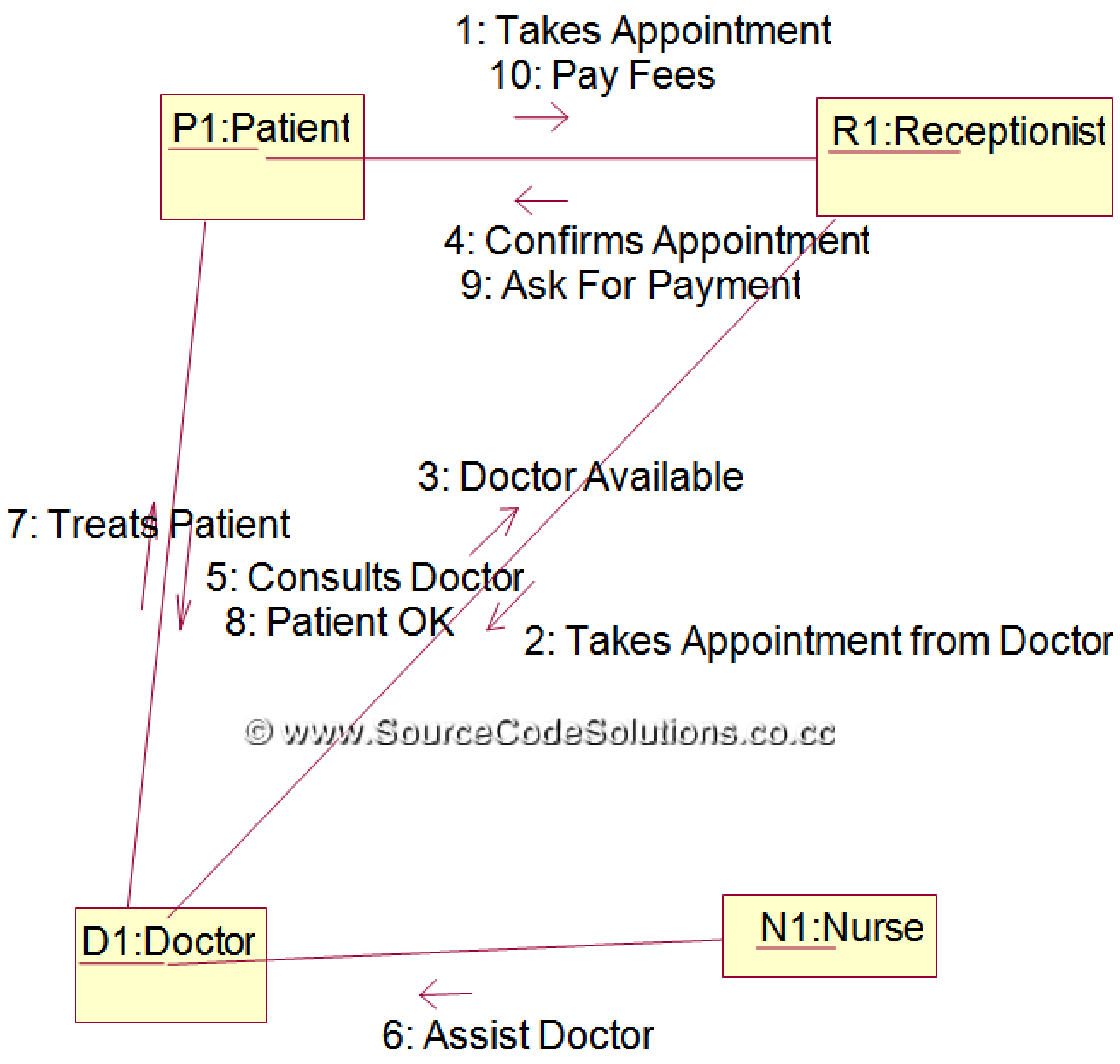
Experiment 8

Aim: To perform the behavioral view diagram for the suggested system: Sequence diagram, Collaboration diagram.

Output:



Sequence Diagram for booking ticket

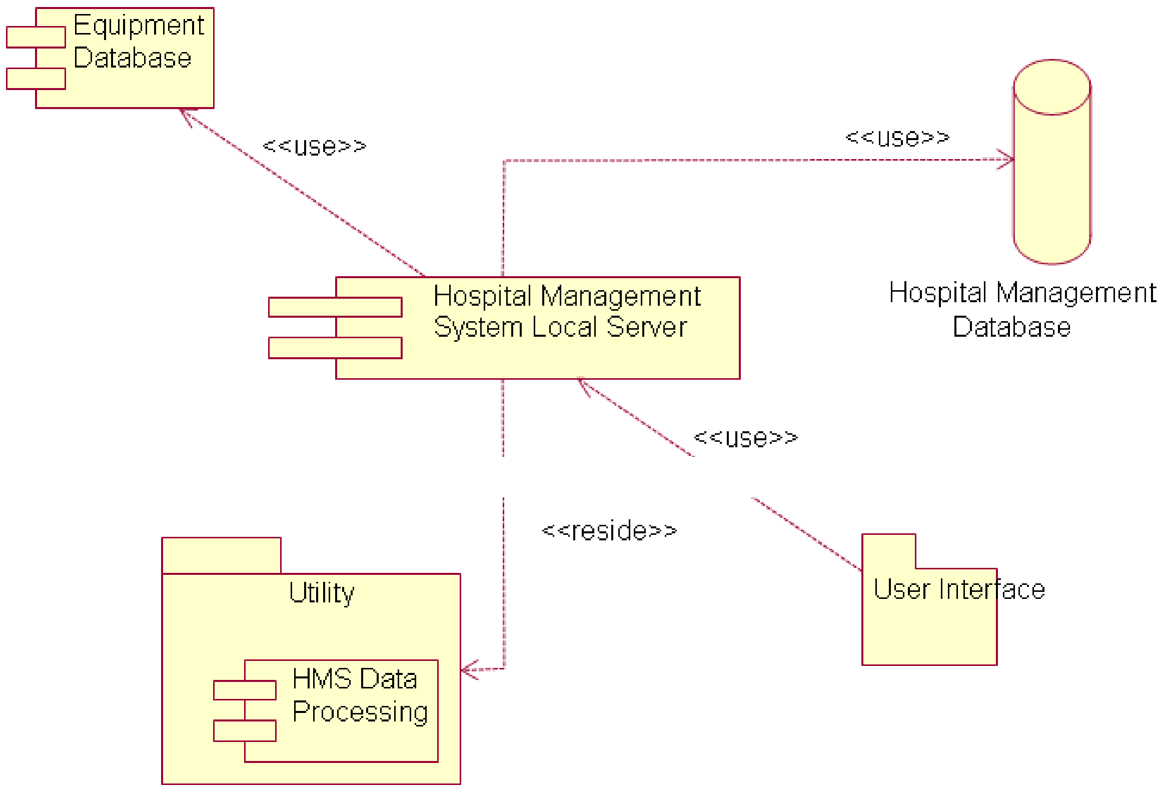


<u>Collaboration Diagram</u>

Conclusion: Sequence diagram and Collaboration Diagram were made successfully by following above steps.

Experiment 9

Aim: To perform the implementation view diagram: Component diagram for the system.

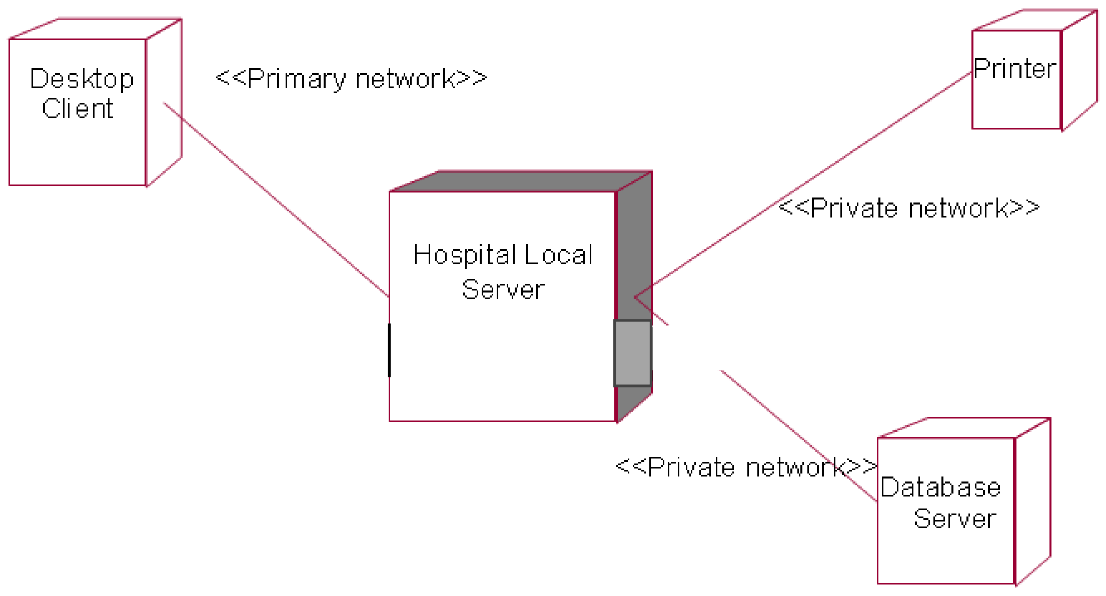


Component Diagram

Conclusion: Component diagram was made successfully by following above steps.

Experiment 10

Aim: To perform the environmental view diagram: Deployment diagram for the system.



Deployment Diagram

Conclusion: Deployment diagram was made successfully by following above steps.

Experiment 11

Aim: To perform various testing using the testing tool unit testing, integration testing for a sample code of the Hospital Management System.

Introduction: In the Hospital Management System project, testing is crucial to ensure the reliability, functionality, and performance of the system. Here is a concise exploration of unit testing and integration testing, elucidating how these testing methodologies are applied within the project context:

Unit Testing: Definition: Unit testing is the process of testing individual components or modules of a system in isolation to ensure they function as intended.

Application to the Project:

1. **Patient Registration and Authentication:**
   - **Unit Test:** Verify that the patient registration process correctly adds a new patient to the system and that authentication works as expected.
   - **Mock Dependencies:** Mock external services (like SMS services) to isolate the unit test.
2. **Medical Record Management (Create, Update, Delete):**
   - **Unit Test:** Test each operation independently. For example, check that a medical record can be successfully created, updated, and deleted.
   - **Mock Dependencies:** Simulate interactions with the database without affecting the actual patient data.
3. **Appointment Scheduling:**
   - **Unit Test:** Ensure that the appointment scheduling functionality works correctly for different types of appointments.
   - **Mock Dependencies:** Simulate interactions with the calendar system.
4. **Inventory Management:**
   - **Unit Test:** Validate that the inventory management operations such as adding, updating, and removing items are functioning correctly.
   - **Mock Dependencies:** Mock communication with the suppliers.
5. **Billing and Payment:**

- **Unit Test:** Test the billing and payment processes to ensure accurate calculation and transaction handling.
- **Mock Dependencies:** Simulate interactions with the payment gateway.

Integration Testing: Definition: Integration testing is the process of testing interactions between different components or systems to ensure they work together seamlessly.

Application to the Project:

1. **Patient-Medical Record Interaction:**
   - **Integration Test:** Test the flow from patient registration to medical record management. Verify that patient data is correctly associated with medical records.
   - **Test Cases:** Check if a patient can schedule appointments, update their records, and view relevant notifications.
2. **Appointment and Inventory Management:**
   - **Integration Test:** Validate that appointment scheduling and inventory management processes integrate smoothly.
   - **Test Cases:** Check if inventory levels are updated based on scheduled appointments.
3. **Billing and Medical Records:**
   - **Integration Test:** Test the end-to-end flow of billing, payment, and medical record updates.
   - **Test Cases:** Ensure that billing information is accurately reflected in the patient's medical records.
4. **Search and Collaboration:**
   - **Integration Test:** Verify that the search functionality integrates well with collaboration features.
   - **Test Cases:** Search for patient records and collaborate with other staff based on search results.
5. **Notification and User Interaction:**
   - **Integration Test:** Check that notifications are generated and received appropriately during various user interactions.
   - **Test Cases:** Confirm that users receive notifications for appointment reminders, medical record updates, etc.

Testing Strategies:

- **Test Automation:** Consider automating tests to ensure efficient and consistent testing.
- **Data Mocking:** Use mock data or temporary databases during testing to avoid interference with real patient data.
- **Scalability Testing:** Assess how well the system performs as the patient and medical data scales

Experiment 12

Aim: Perform Estimation of effort using FP Estimation for the Hospital Management System.

Theory: In the Hospital Management System project, accurately estimating the effort required is paramount for effective project management. The Function Point (FP) Estimation technique serves as a valuable method for quantifying the functionality provided by the system.

Steps for FP Estimation:

1. **Identifying Functionalities:** Enumerate and categorize the various functionalities offered by the Hospital Management System. This includes patient interactions, data processing, external interfaces, and more.
2. **Assigning Complexity:** Evaluate the complexity of each functionality based on predetermined criteria. Functionality complexity is usually classified as low, average, or high.
3. **Calculating Unadjusted Function Points (UFP):** Sum the complexity-adjusted counts for each functionality to obtain the Unadjusted Function Points. This reflects the overall size of the system.
4. **Determining Technical Complexity:** Assess the technical complexity of the system, considering factors like the use of distributed architecture, reusability, and performance requirements.
5. **Calculating Adjusted Function Points (AFP):** Apply technical complexity factors to the Unadjusted Function Points to derive the Adjusted Function Points. This provides a more accurate measure of the effort required.
6. **Mapping to Effort:** Utilize historical data or industry benchmarks to map Adjusted Function Points to the effort required for development. This mapping helps in estimating the person-hours or person-days needed.

By following the Function Point Estimation process, project managers can make informed decisions regarding resource allocation, project timelines, and overall project planning, contributing to the successful execution of the Hospital Management System.

Experiment 13

Aim: To prepare a Timeline Chart, Gantt Chart, and PERT Chart for the Hospital Management System project.

Theory: In the Hospital Management System project, creating a detailed project schedule is essential for effective planning, coordination, and monitoring. Timeline Chart, Gantt Chart, and PERT Chart are valuable tools to visualize project timelines, dependencies, and critical paths.

Timeline Chart:

- A Timeline Chart provides a high-level overview of project milestones and key events over time.
- Identify significant project phases, such as system design, development, testing, and deployment.
- Represent each phase on a timeline, indicating their start and end dates.
- Use this chart for a quick understanding of the project's temporal progression.

Gantt Chart:

- A Gantt Chart offers a more detailed representation of project tasks, durations, and dependencies.
- List all project tasks or activities and their estimated start and end dates.
- Visualize task durations as horizontal bars on a timeline, with dependencies between tasks indicated.
- Gantt Charts help in resource allocation, scheduling, and identifying critical paths.

PERT Chart (Program Evaluation and Review Technique):

- A PERT Chart is useful for illustrating task dependencies and determining the critical path.
- Identify all project tasks and their dependencies, representing them as nodes and arrows.
- Use optimistic, pessimistic, and most likely time estimates for each task to calculate expected durations and critical paths.

- PERT Charts provide a comprehensive view of task interdependencies and aid in risk management.

These charts will assist in effectively managing the timeline, resources, and critical aspects of the Hospital Management System project.