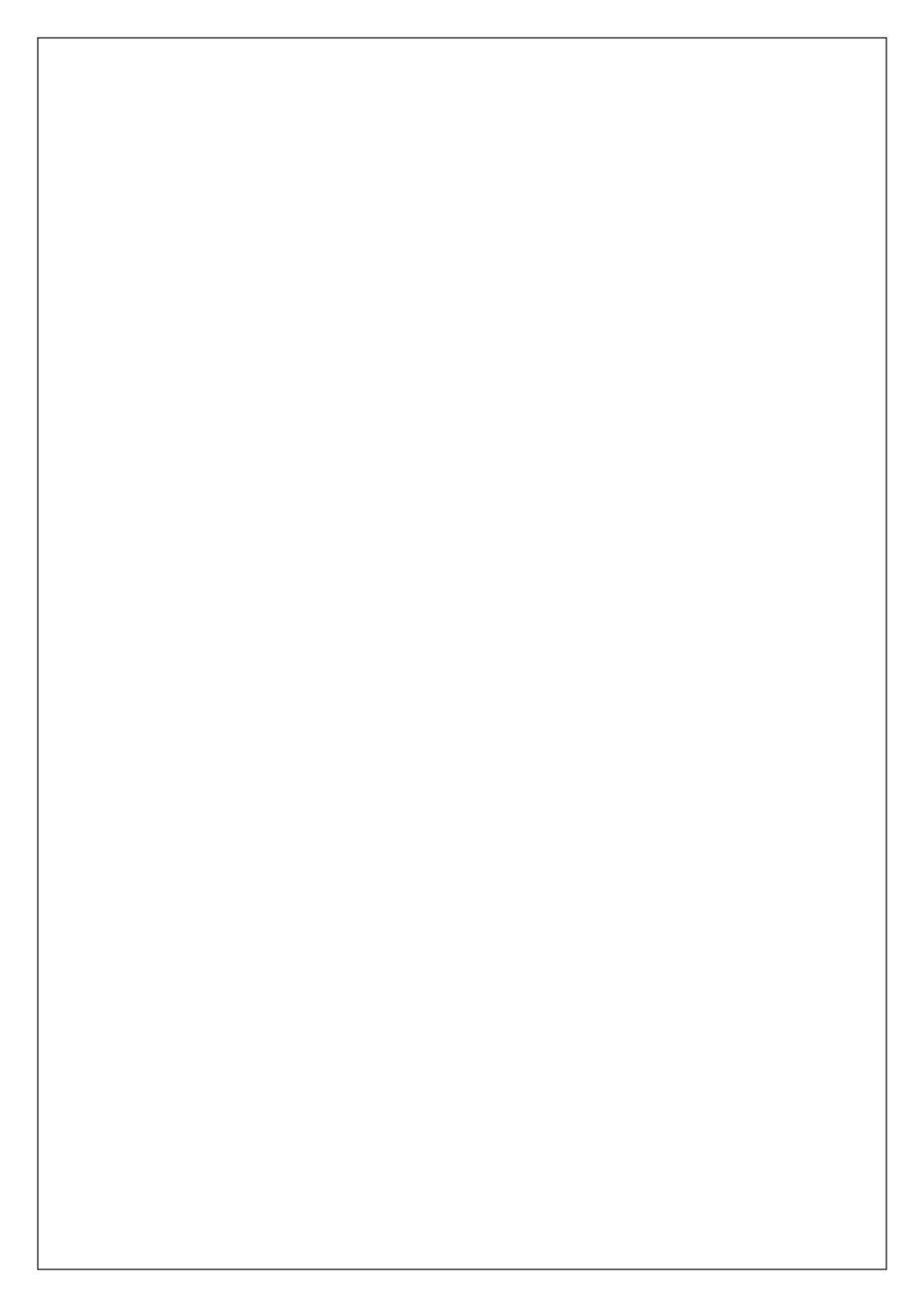
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RAJALAKSHMI ENGINEERING COLLEGE  RAJALAKSHMI NAGAR, THANDALAM — 602 105    RAJALAKSHMI  ENGINEERING COLLEGE   |  |  |  | | --- | --- | --- | | CB23332  SOFTWARE ENGINEERING LAB | | | | Laboratory Record Note Book | | | | Name: . . . . . . . . . . . . . . . . . .  Year/ Branch / Section :  Register No. :  Semester:  Academic Year: . . . . . . . . . . . . . . . . . . . . . . . |   Department of CSBS/CB23332 |



RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

RAJALAKSHMI NAGAR, THANDALAM - 602-105

# BONA FIDE CERTIFICATE

NAME:REGISTER NO.:



ACADEMIC YEAR: 2024-25 SEMESTER: 111 BRANCH: B.E/B.Tech



This Certification is the bonafide record of work done by the above student in the

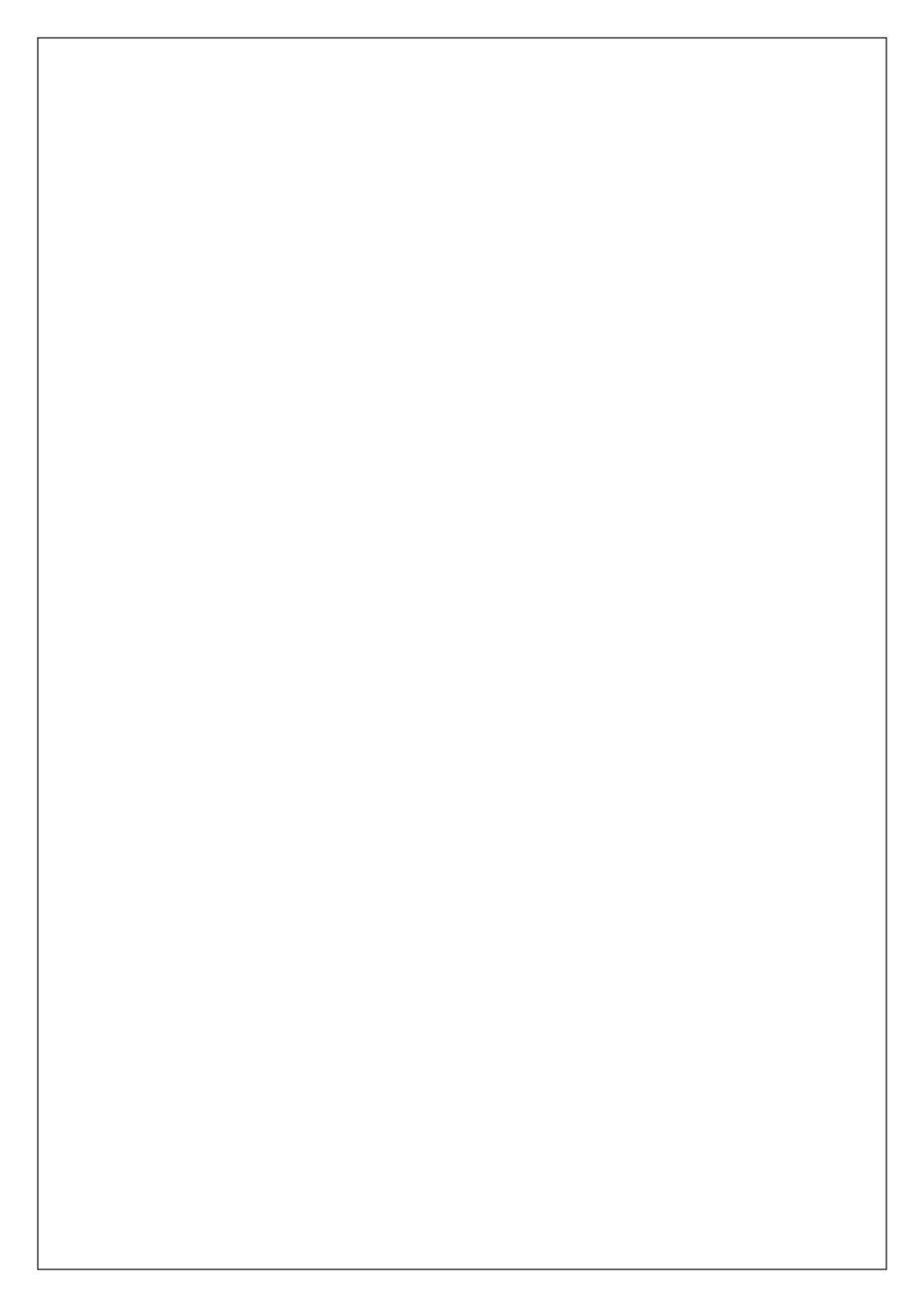
CB23332-SOFTWARE ENGINEERING - Laboratory during the year 2024 — 2025.

Signature of Faculty -in — Charge

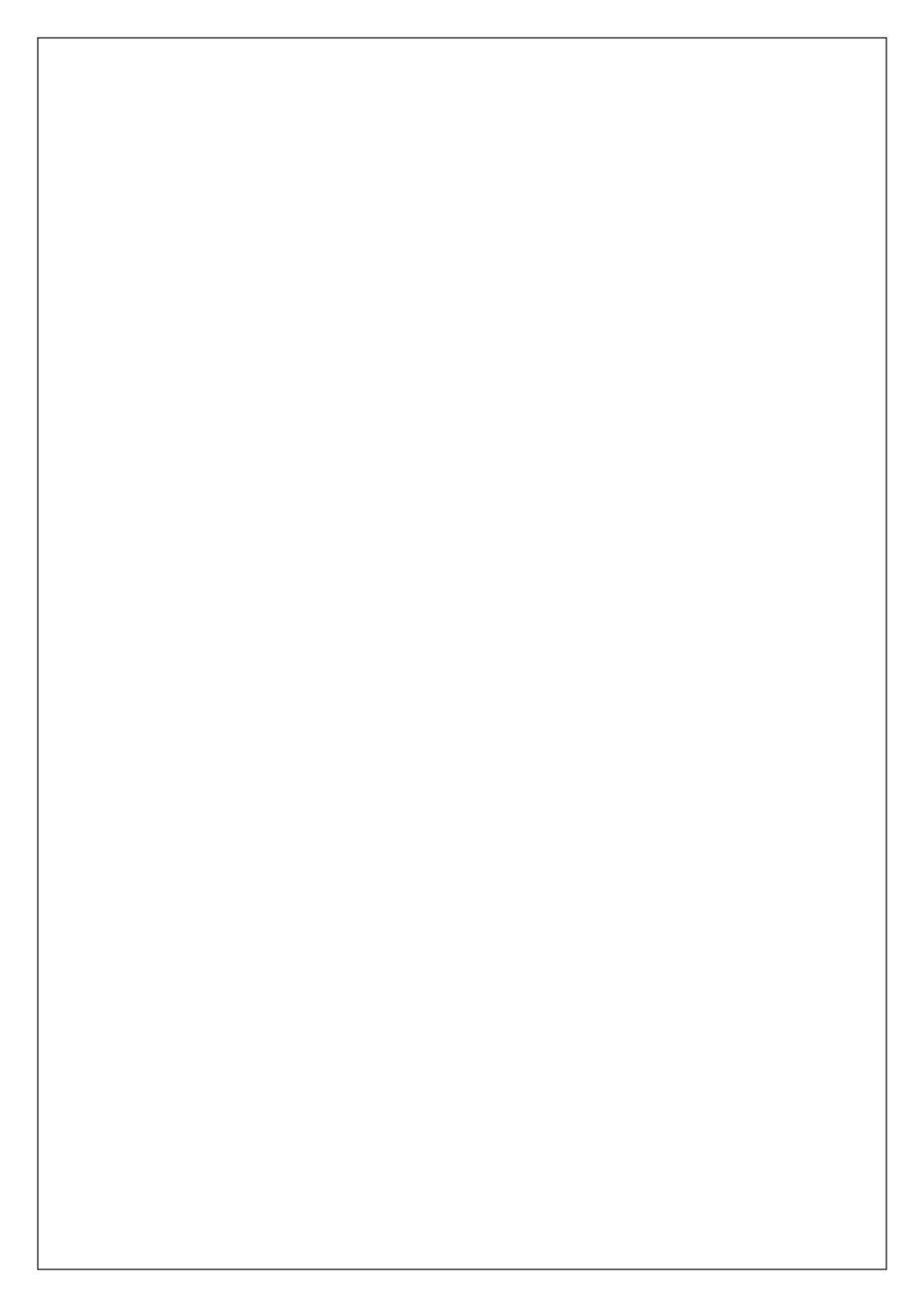
Submitted for the Practical Examination held on

Internal Examiner External Examiner

Department of CSBS/CB23332



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INDEX   |  |  |  |  | | --- | --- | --- | --- | | S No. | Name of the Experiment | Expt.  Date | Faculty Sign | | 1. | Preparing Problem Statement |  |  | | 2. | Software Requirement Specification (SRS) |  |  | | 3. | Entity-Relational Diagram |  |  | | 4. | Data Flow Diagram |  |  | | 5. | Use Case Diagram |  |  | | 6. | Activity Diagram |  |  | |  | State Chart Diagram |  |  | | 8. | Sequence Diagram |  |  | | 9. | Col laboration Diagramt |  |  | | 10. | Class Diagram |  |  |   Department of CSBS/CB23332 |



|  |  |
| --- | --- |
| EX NO:I | WRITE THE COMPLETE PROBLEM STATEMENT |
| DATE: |

AIM:

To prepare a PROBLEM STATEMENT for the project hospital management system app.

ALGORITHM:

l . The problem statement is the initial starting point for a project.

1. A problem statement describes what needs to be done without describing how.
2. It is typically a one-to-three-page document that all stakeholders agree on, describing the project goals at a high level.
3. The problem statement is intended for a broad audience and should be written in non-technical terms.
4. It helps technical and non-technical personnel communicate effectively by providing a clear description of the problem.
5. It does not describe the solution to the problem.

INPUT:

1. The input to requirement engineering is the problem statement prepared by the customer.
2. It may provide an overview of the existing system along with broad expectations from the new system.
3. The first phase of requirements engineering begins with requirements elicitation, i.e., gathering information about requirements.
4. Requirements are identified with the help of the customer and existing system processes.

PROBLEM:

Traditional hospital management systems face several challenges, including manual processes, lack of real-time information, and limited online functionality. Patients and staff find it difficult to track appointments, manage medical records, and access services efficiently. The absence of an efficient online appointment booking and medical record management system leads to delays, miscommunication, and errors. Additionally, hospital resources (e.g., beds, medical equipment) are not seamlessly integrated, making it harder for healthcare providers to coordinate care.

BACKGROUND:

A hospital management app is designed to streamline and organize the operations of healthcare institutions, improving efficiency and providing better care for patients. The background for such an app involves various facets, ranging from administrative functions to patient care, all aimed at enhancing the quality of service. Hospital inventory management ensures that essential medicines, equipment, and supplies are stocked and managed efficiently. Given the sensitive nature of healthcare data, the app must adhere to strict privacy regulations, such as HIPAA in the United States, Data Security Using encryption and secure servers to protect patient data.Compliance with Regulations**,**Ensuring the app meets legal requirements in managing patient records, billing, and medical practices.

RELEVANCE:

A hospital management app is highly relevant in today’s healthcare landscape as it streamlines operations, improves patient care, and reduces costs. By automating administrative tasks such as appointment scheduling, billing, and inventory management, it enhances efficiency and saves time for both staff and patients. The app also supports better communication, data security, and regulatory compliance, while providing insights through data analytics. With features like telemedicine integration and patient record management, hospital management apps ensure seamless care delivery, making them essential tools for modern healthcare institutions.

OBJECTIVES:

The primary objective of this project is to develop a hospital management app that leverages technology to optimize health care by improving patient experience , and promote sustainable management. Specific objectives include:

1. **Enhance Operational Efficiency:** Automate and streamline administrative tasks such as patient registration, appointment scheduling, billing, and inventory management to reduce manual effort and improve workflow efficiency.
2. **Improve Patient Care and Experience:** Provide easy access to medical records, enable seamless communication between patients and healthcare providers, and facilitate timely consultations, whether in-person or through telemedicine.
3. **Ensure Data Security and Compliance:** Safeguard patient data by adhering to healthcare regulations like HIPAA, ensuring secure storage, and managing access to sensitive information.
4. **Optimize Resource Management:** Efficiently manage hospital resources, including staff scheduling, medical supplies, and equipment, to ensure optimal utilization and reduce wastage.
5. **Support Data-Driven Decision-Making:** Provide administrators with real-time analytics and reports to track key metrics like patient flow, financial performance, and resource allocation for informed decision-making.
6. **Facilitate Financial Management:** Streamline billing, invoicing, and insurance claim processes, improving financial transparency and reducing errors.
7. **Promote Scalability and Flexibility:** Allow the app to grow with the hospital, adapting to the needs of different departments and multiple locations, ensuring long-term usability and scalability.
8. **Enhance Telemedicine Capabilities:** Integrate telemedicine features to allow remote consultations and health monitoring, improving patient access to care, particularly for follow-ups or in underserved areas.

Result:

|  |  |
| --- | --- |
| EX NO:2 | WRITE THE SOFTWARE REQUIREMENT SPECIFICATION  DOCUMENT |
| DATE: |

To conduct requirement analysis and develop a Software Requirement Specification Sheet (SRS) for the project hospital management system app.

ALGORITHM:

An SRS shall address the following:

1. Functionality: What is the software supposed to do?
2. External Interfaces: How does the software interact with users, hardware, and other systems?
3. Performance: Speed, availability, and response time for system operations.
4. Attributes: Considerations for portability, correctness, maintainability, security, etc.
5. Design Constraints: Standards, implementation language, database policies, resource limits, and operating environments.
6. Introduction
   1. Purpose

This document describes the system requirements for the Smart Hospital Management System (SHMS), designed to streamline hospital operations and improve patient care through automation and integration Document Conventions

* **IOT:** Network of connected devices for patient data.
* **UI:** Visual design of the app.
* **API:** Protocols for system integration.
* **KPI:** Metrics to measure performance.
* **RBAC:** Access control based on user roles.
* **1.**2Intended Audience and Reading Suggestions

This document is intended for:

* + - * Developers: To understand technical requirements for implementation.
      * Project Managers: To monitor timelines and deliverables.
      * Stakeholders: To verify alignment with operational goals.

Quality Assurance: To validate system functionality and performance

* 1. Project Scope

The system will support patient appointment management, consultation scheduling, medical record access, resource allocation, and automated notifications. It will cater to hospital staff, including doctors, nurses, and administrative personnel, as well as patients.

.

* 1. References:

**Setup:** Deploy 10T health sensors (e.g., temperature, heart rate, blood pressure) for patient monitoring.

**Calibration:** Ensure sensors are calibrated for accurate health data collection.

**Data Collection:** Gather real-time data from patients for remote monitoring during checkups.

**Integration:** Connect sensors to the app for automated health tracking and alerts.

**Maintenance:** Regularly test and maintain sensors for optimal functionality.

1. Overall Description
   1. Product Perspective

The SHMS is a web-based application with mobile access, designed to integrate with hospital databases, health management tools, and medical equipment tracking systems

2.2Product Features

Key features include:

Patient appointment scheduling

Real-time availability tracking for doctors, medical equipment, and beds

Integration with patient medical records

Automated reminders and notifications

Access to test results and prescriptions

Reporting and analytics tools for hospital management

2.3User Classes and Characteristics

**Patients**: Can book appointments, access their medical records, and view test results.

**Doctors**: Can access patient records, schedule consultations, and issue prescriptions.

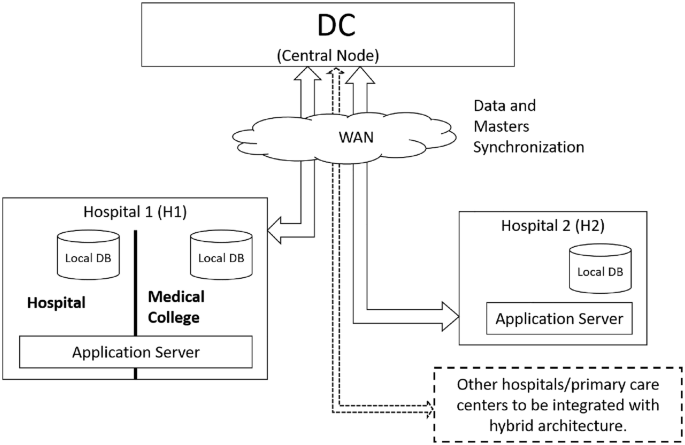
**Nurses**: Manage patient care, monitor resources, and assist in medical procedures.

**Administrative Staff**: Oversee scheduling, patient check-in, and resource management.

**Hospital Management**: Use reporting and analytics to make operational decisions and ensure

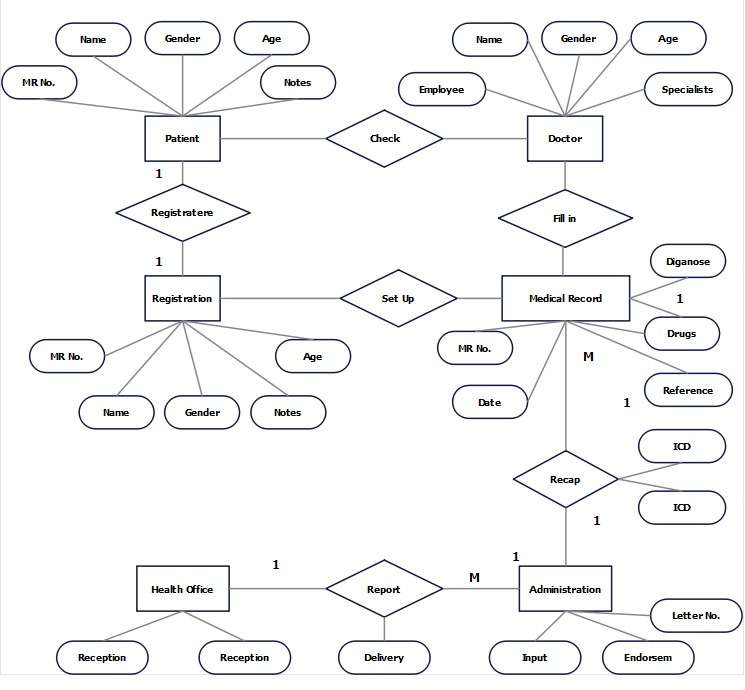
resource optimization.

* 1. The system will operate in the following environment:
* **Data Collection Devices:** Sensors (e.g., IoT-enabled medical devices such as heart rate monitors, blood pressure cuffs) for patient monitoring and data collection.
* **Distributed Database:** A central, scalable database that stores patient information, medical records, health checkup history, and operational data. This could be managed through cloud platforms like AWS, Azure, or Google Cloud.
* **Cloud-Based Architecture:** The system is hosted on the cloud, allowing access via web browsers and mobile platforms for users (patients, doctors, admins) to interact with the system from anywhere. It provides real-time updates, remote consultations, and integration with hospital management systems.



ER Diagram:

ER Diagram for hospital management system



* 1. Design and Implementation Constraints

**IoT Sensor Integration:**Support real-time data from medical sensors (e.g., heart rate, blood pressure) for patient monitoring.

**Secure APIs:**Ensure encrypted data exchange and compliance with regulations (e.g., HIPAA) using secure APIs and authentication.

**Scalability:**Design for easy scaling to support multiple hospitals, using cloud infrastructure and load balancing..

Specific Requirements

Description and Priority

The system prioritizes optimizing **Hospital Management System** maximizes patient safety, operational efficiency, and cost-effectiveness while maintaining high standards of data security and regulatory compliance.

4o mini

Stimulus/Response Sequence

1. **Patient’s vital signs exceed a threshold:**
   * **Response:** The system triggers an alert to healthcare providers for immediate intervention.
2. **Doctor updates patient’s treatment plan:**
   * **Response:** The system notifies the patient and care team about the change in treatment or medication.
3. **Patient's medication schedule is due:**
   * **Response:** Notifications are sent to the nursing staff to administer medication on time.

Functional Requirements

**Real-Time Monitoring:** The app continuously collects data from patient monitoring devices (e.g., heart rate monitors, oxygen saturation sensors) and displays real-time health metrics, enabling healthcare providers to track patient conditions.

**Automated Alerts and Notifications:** The system automatically triggers alerts for abnormal patient conditions (e.g., high blood pressure, irregular heart rate) to ensure timely medical interventions.

**Electronic Health Records (EHR) Access:** The app provides doctors, nurses, and authorized personnel with access to electronic health records, enabling efficient patient data management and collaboration on treatment plans.

**Remote Access and Communication:** Healthcare providers can monitor patient data and communicate through the app, ensuring continuous care even when staff is not physically present in the same location.

External Interface Requirements

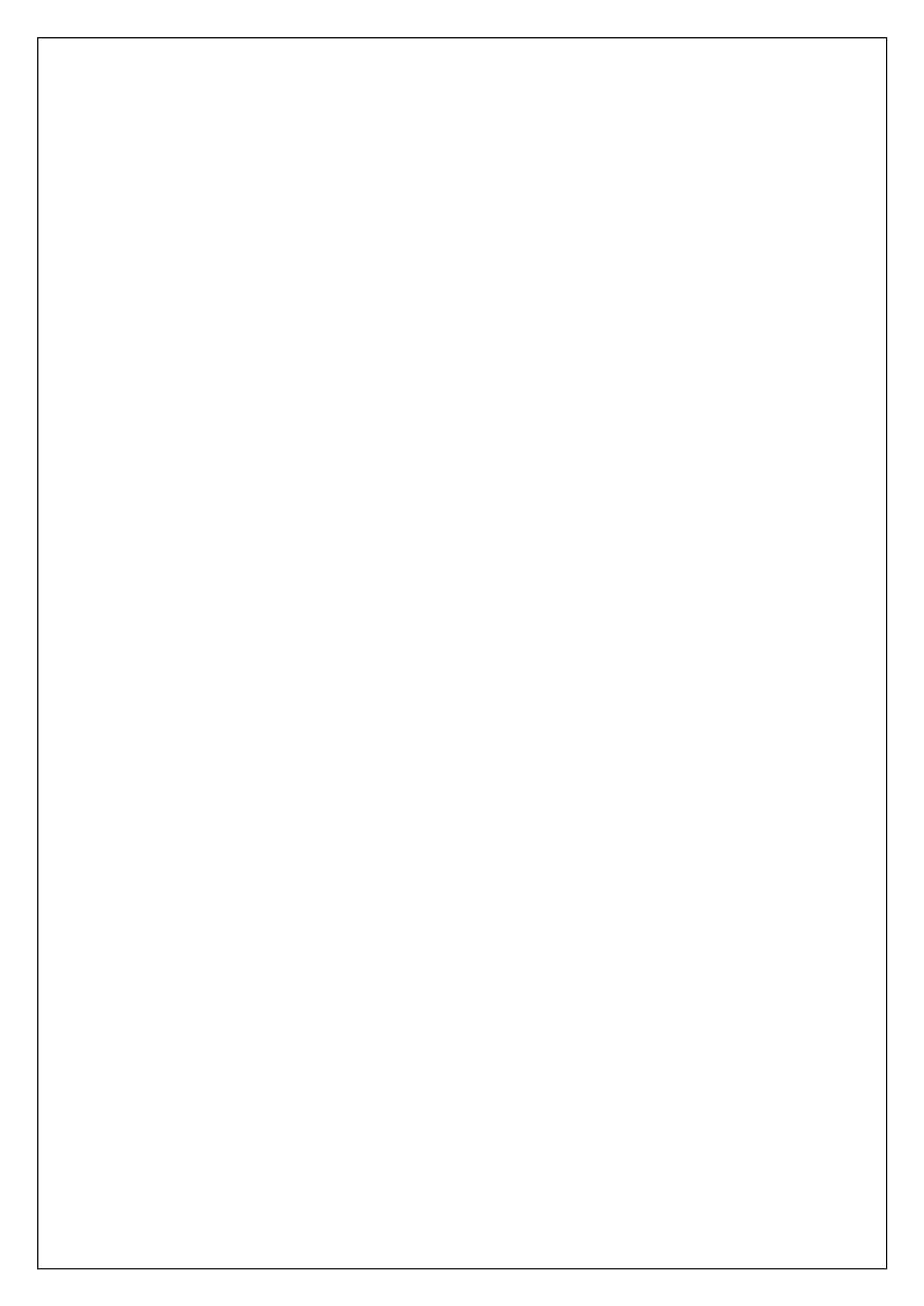
* 1. User Interfaces

 Frontend: Python with Streamlit for web-based dashboards.

 Backend: SQL database for storing sensor and irrigation data.

* 1. Hardware Interfaces

Integration with IoT-enabled medical devices for real-time patient monitoring and using Windows or Linux operating systems for server support.



* 1. Software Interfaces

1. **Programming Language:** **Python** for backend development due to its flexibility and extensive library support.
2. **Database:** **MySQL** for secure storage of patient records, medical histories, and sensor data.
3. **Weather API:** Integration with weather services to gather forecast data for hospital logistics and patient care.
   1. Communication Interfaces
   2. **Web Application:** Accessible via modern browsers (**Chrome, Firefox, Edge**) for ease of use across devices.
   3. **Secure HTTPS Protocol:** Ensures encrypted data exchange, protecting sensitive patient information..
4. Additional Requirements
   1. Performance Requirement

**ER Diagram:** Entities include **patients**, **doctors**, **appointments**, and **medical records**.

**Normalization:** Ensures efficient data storage and minimizes redundancy.

**System Response Time:** Critical actions (e.g., alerts) must trigger within **2 seconds**..

* 1. Safety Requirements

1. **Data Backups:** Regular backups to ensure data recovery.
2. **Malfunction Detection:** System detects hardware issues and notifies technicians.
   1. Security Requirements

**RBAC:** Role-based access control for **doctors**, **nurses**, and **admins**.

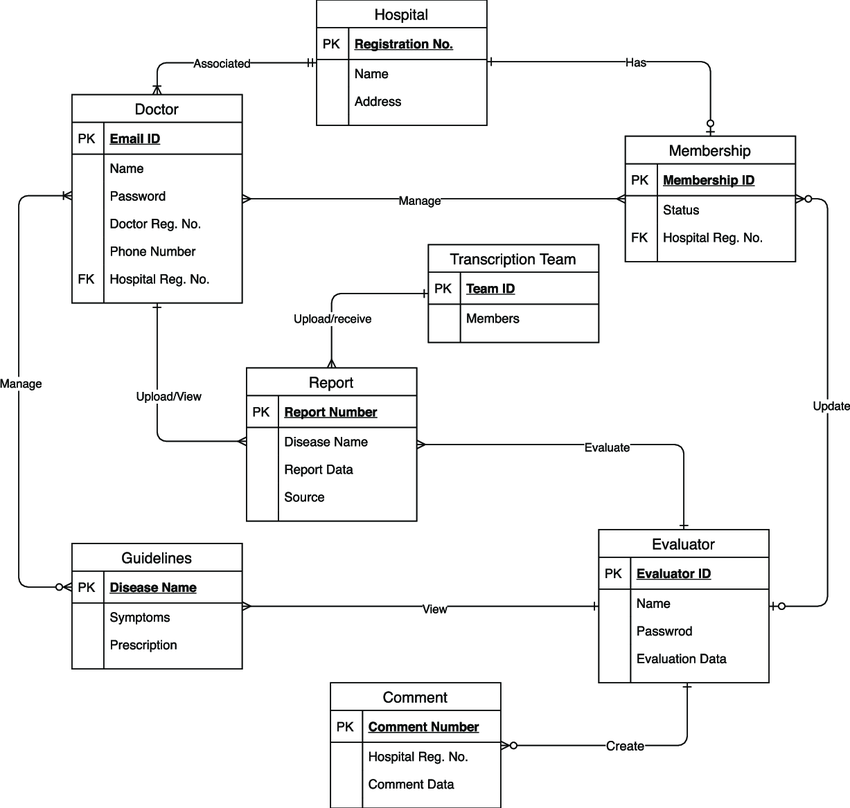
**Encryption:** Sensitive data is encrypted during **transmission** and **storage**.

* 1. Software Quality Attributes

**Availability:** 99.9% uptime.

**Correctness:** Accurate patient data and decision-making.

**Maintainability:** Easy updates and system improvements.

. ER diagram:

|  |  |
| --- | --- |
| EX NO:3 | DRAW THE ENTITY RELATIONSHIP DIAGRAM |
| DATE: |

AIM:

To Draw the Entity Relationship Diagram for hospital management system app.

ALGORITHM:

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary I Relation Types

Step 4: Mapping of Binary I :N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

INPUT:

entities

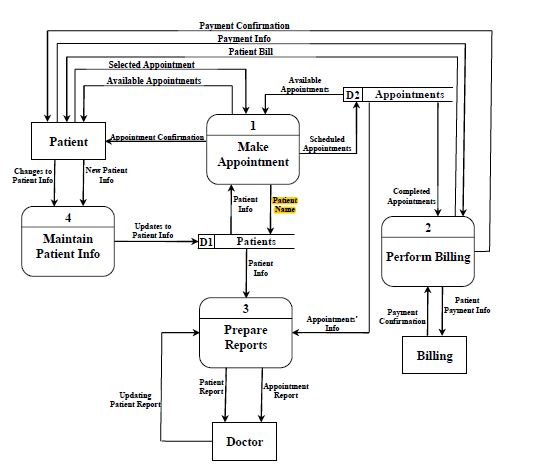
Entity Relationship Matrix

Primary Keys

Attributes

Mapping of attributes with entities

DFD diagram:



|  |  |
| --- | --- |
| EX NO:4 | DRAW THE DATA FLOW DIAGRAMS AT LEVEL O AND LEVEL 1 |
| DATE: |

AIM:

To Draw the Data Flow Diagram for hospital management system and List the Modules in the Application.

ALGORITHM:

1. Open the Visual Paradigm to draw DFD (Ex.Lucidchart)
2. Select a data flow diagram template
3. Name the data flow diagram
4. Add an external entity that starts the process
5. Add a Process to the DFD
6. Add a data store to the diagram
7. Continue to add items to the DFD
8. Add data flow to the DFD
9. Name the data flow
10. Customize the DFD with colours and fonts

I l. Add a title and share your data flow diagram

INPUT:

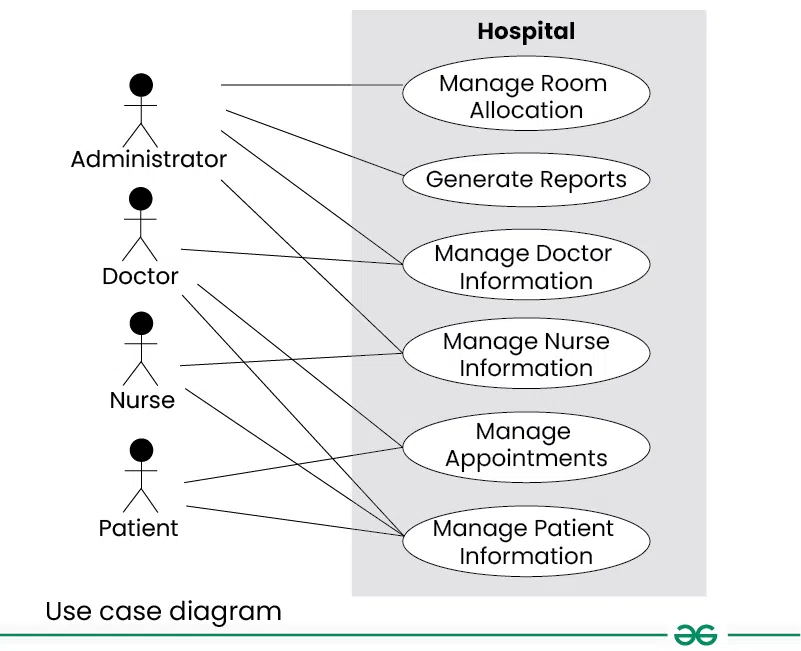
Processes

External entities

Datastores

**Result:**

Use Case diagram:



|  |
| --- |
| EX NO:5 |
| DATE: **Draw the Use Case Diagram** |

AIM: To Draw the Use Case Diagram for hospital management system

ALGORITHM:

Step l: Identify Actors

Step 2: Identify Use Cases

Step 3: Connect Actors and Use Cases

Step 4.• Add System Boundary

Step 5: Define Relationships

Step 6: Review and Refine

step 7: Validate

INPUTS:

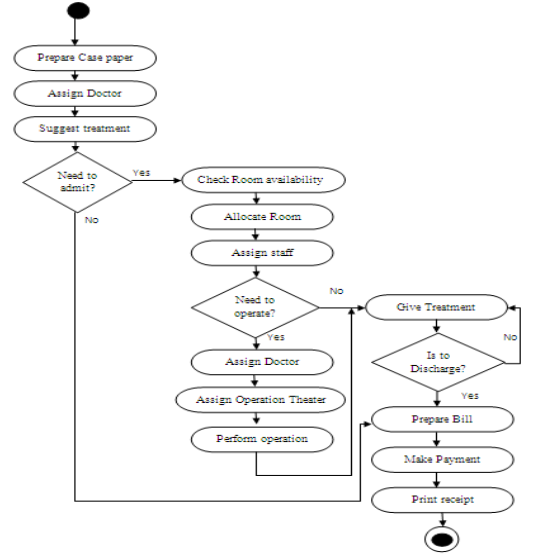
Actors

Use Cases

Relations

**Results:**

Activity diagram:



|  |  |
| --- | --- |
| EX NO:06 DRAW ACTIVITY DIAGRAM  DATE: |  |

Aim :

To Draw the activity Diagram for smart agriculture system for irrigation

* ALGORITHM:
* Step 1: Identify the Initial State and Final States
* Step 2: Identify the Intermediate Activities Needed
* Step 3: Identify the Conditions or Constraints
* Step 4: Draw the Diagram with Appropriate Notations
* INPUTS:

Activities

Decision points

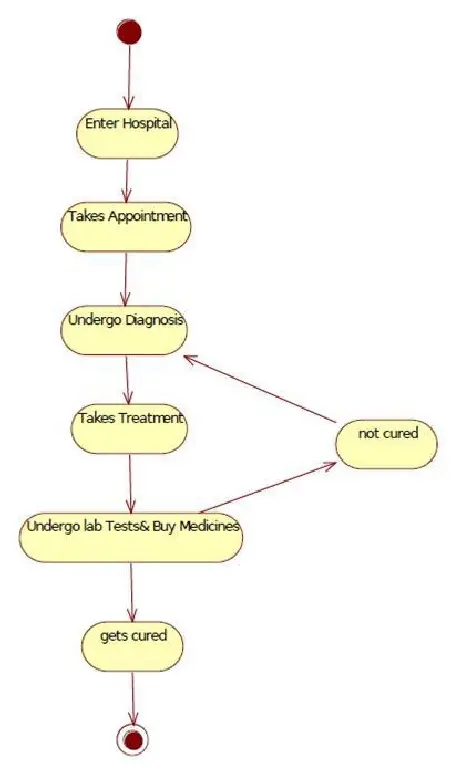
Guards

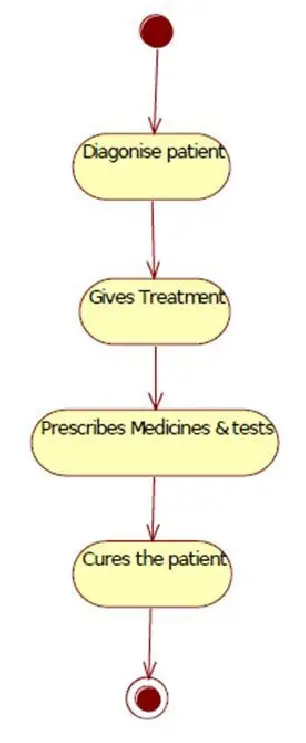
Parallel activities

Conditions

Result:

State Chart diagram:





|  |  |
| --- | --- |
| EX NO:7 | DRAW STATE CHART DIAGRAM OF ALL USE CASES. |
| DATE: |

AIM:

To Draw the State Chart Diagram for hospital management system app

ALGORITHM:

STEP-I: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

INPUTS:

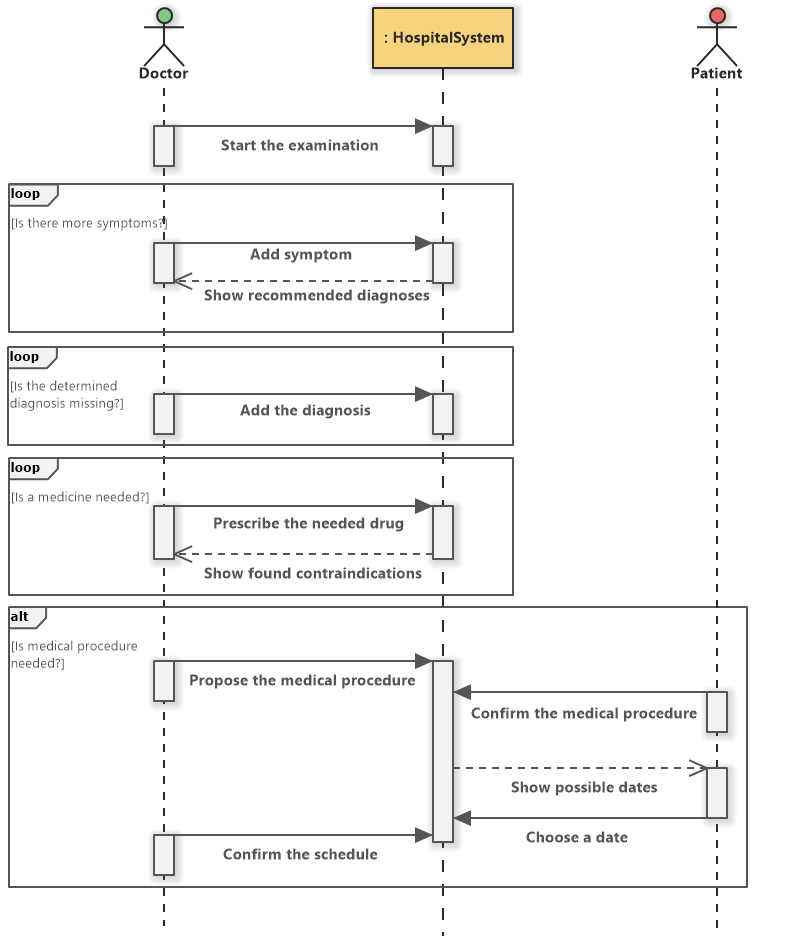
Objects

States

Events

Results:

Sequence diagram:



|  |  |
| --- | --- |
| EX NO:8 | DRAW SEQUENCE DIAGRAM OF ALL USE CASES. |
| DATE: |

AIM: To Draw the Sequence Diagram for hospital management app

ALGORITHM:

1. Identify the Scenario
2. List the Participants
3. Define Lifelines
4. Arrange Lifelines
5. Add Activation Bars
6. Draw Messages
7. Include Return Messages
8. Indicate Timing and Order
9. Include Conditions and Loops
10. Consider Parallel Execution I l. Review and Refine
11. Add Annotations and Comments
12. Document Assumptions and Constraints
13. Use a Tool to create a neat sequence diagram

INPUTS:

Objects taking part in the interaction.

Message flows among the objects.

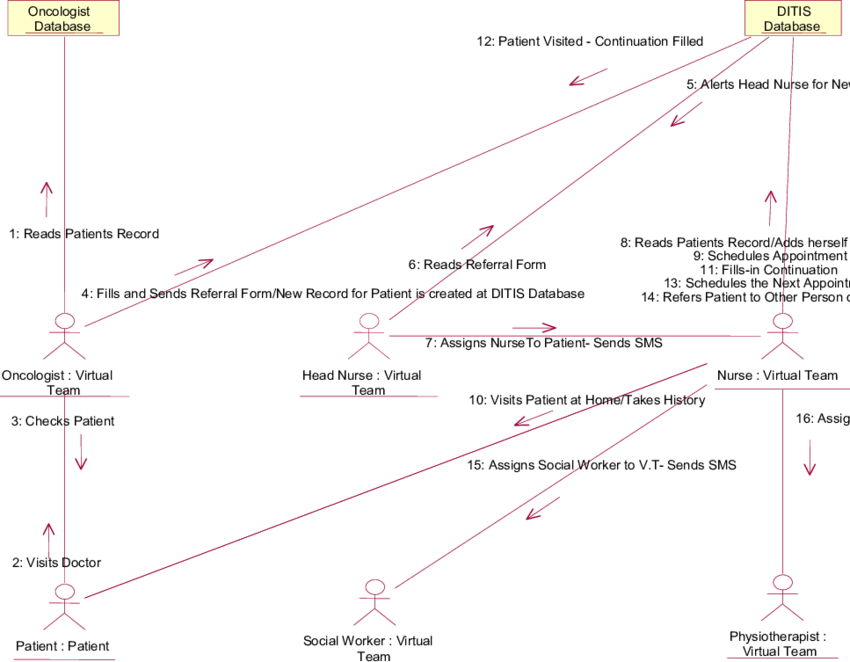
The sequence in which the messages are flowing.

Object organization.

Result:

Collaboration diagram:

Collaboration Diagram for hospital management system



|  |  |
| --- | --- |
| EX NO:9 | DRAW COLLABORATION DIAGRAM OF ALL USE CASES |
| DATE: |

To Draw the Collaboration Diagram for hospital management app

ALGORITHM:

Step l: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant explanations or annotations.

INPUTS:

Objects taking part in the interaction.

Message flows among the objects.

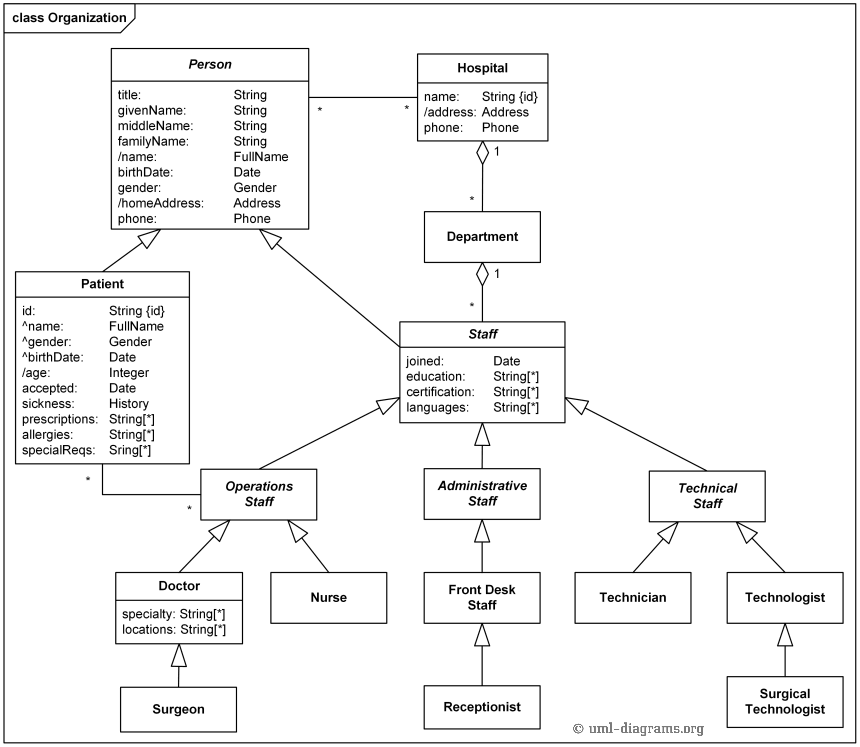
The sequence in which the messages are flowing.

Object organization

Result:

Class diagram:

cl



controls

|  |  |
| --- | --- |
| EX NO:IO | ASSIGN OBJECTS IN SEQUENCE DIAGRAM TO CLASSES AND MAKE CLASS DIAGRAM. |
| DATE: |

AIM:

To draw the class diagram for my project.

ALGORITHM:

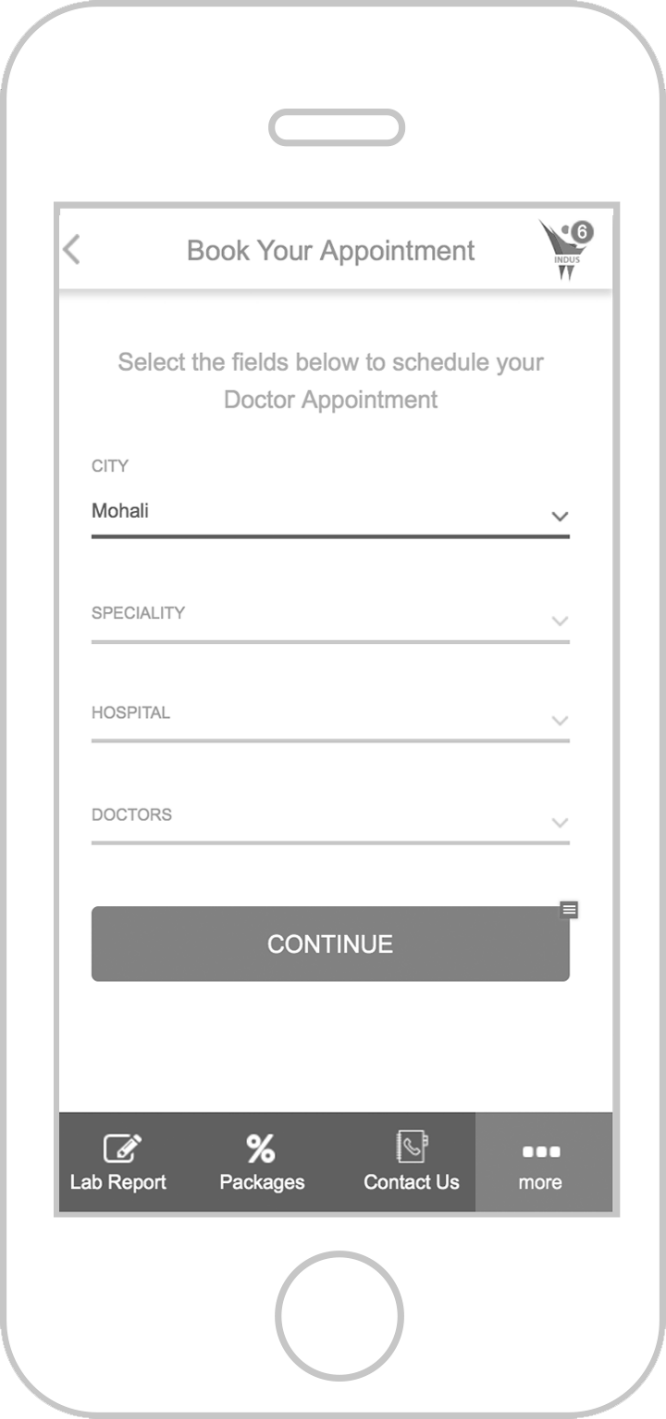
l. Identify Classes

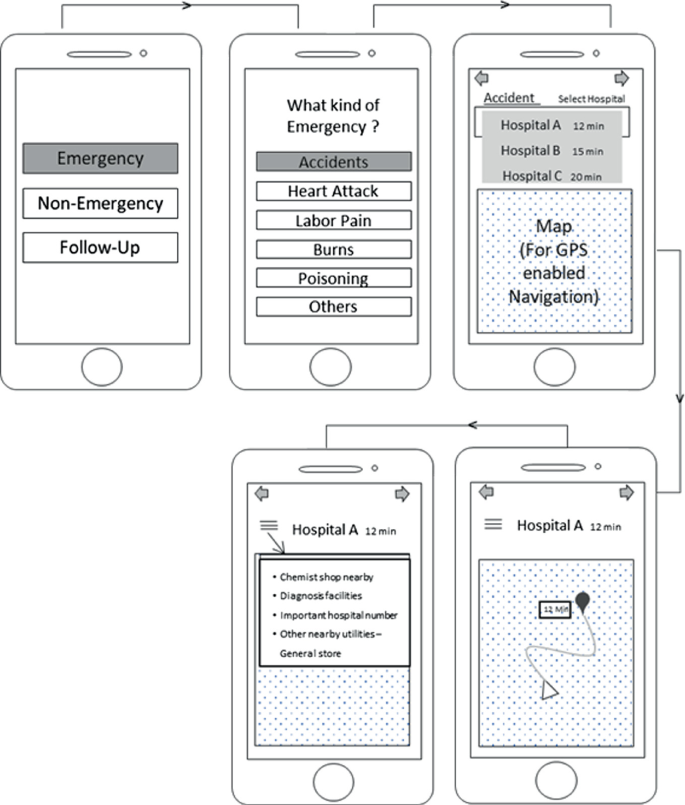
1. List Attributes and Methods
2. Identify Relationships
3. Create Class Boxes
4. Add Attributes and Methods
5. Draw Relationships
6. Label Relationships
7. Review and Refine
8. Use Tools for Digital Drawing

INPUTS:

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

RESULT:





|  |  |
| --- | --- |
| EX NO:II | MINI PROJET-hospital management system app |
| DATE: |

Aim:

The aim of the **Hospital Management System App** is to optimize hospital operations and improve patient care by providing healthcare professionals with real-time access to patient information, medical records, and treatment schedules. The app streamlines administrative tasks such as appointment scheduling, billing, and resource management, while ensuring secure communication between doctors, nurses, and staff. By automating key processes and enhancing data management, the app aims to improve efficiency, reduce operational costs, and ensure better coordination of care, ultimately leading to improved patient outcomes and a more effective healthcare environment.

Algorithm:

1. **User Authentication:**
   * User logs in; access granted on valid credentials.
2. **Patient Registration:**
   * Admin registers patient; generates unique **Patient ID**.
3. **Appointment Scheduling:**
   * Patient selects a slot; system confirms and schedules.
4. **Real-Time Monitoring:**
   * IoT devices track vitals; alerts sent if abnormal.
5. **Billing & Payment:**
   * Bill generated, patient pays, receipt issued.
6. **Data Security:**
   * **RBAC** controls access based on roles.
7. **Emergency Alerts:**
   * Abnormal vitals trigger alerts to staff.
8. **Backup & Recovery:**
   * Daily backups and recovery in case of failure.
9. **System Log:**
   * Logs user actions for auditing.
10. **System Exit:**

* User logs out, session data cleared.

**Program:** **import sqlite3**

**from datetime import datetime**

**# Connect to SQLite Database (or create it if it doesn't exist)**

**conn = sqlite3.connect('hospital\_management.db')**

**cursor = conn.cursor()**

**# Create tables**

**cursor.execute('''CREATE TABLE IF NOT EXISTS patients (**

**patient\_id INTEGER PRIMARY KEY,**

**name TEXT,**

**age INTEGER,**

**contact TEXT)''')**

**cursor.execute('''CREATE TABLE IF NOT EXISTS appointments (**

**appointment\_id INTEGER PRIMARY KEY,**

**patient\_id INTEGER,**

**doctor\_name TEXT,**

**appointment\_date TEXT,**

**FOREIGN KEY(patient\_id) REFERENCES patients(patient\_id))''')**

**cursor.execute('''CREATE TABLE IF NOT EXISTS billing (**

**bill\_id INTEGER PRIMARY KEY,**

**patient\_id INTEGER,**

**amount REAL,**

**bill\_date TEXT,**

**FOREIGN KEY(patient\_id) REFERENCES patients(patient\_id))''')**

**# Function to register a new patient**

**def register\_patient():**

**name = input("Enter patient's name: ")**

**age = int(input("Enter patient's age: "))**

**contact = input("Enter patient's contact number: ")**

**cursor.execute('''INSERT INTO patients (name, age, contact) VALUES (?, ?, ?)''', (name, age, contact))**

**conn.commit()**

**print("Patient registered successfully!")**

**# Function to schedule an appointment**

**def schedule\_appointment():**

**patient\_id = int(input("Enter patient ID: "))**

**doctor\_name = input("Enter doctor's name: ")**

**appointment\_date = input("Enter appointment date (YYYY-MM-DD): ")**

**cursor.execute('''INSERT INTO appointments (patient\_id, doctor\_name, appointment\_date) VALUES (?, ?, ?)''',**

**(patient\_id, doctor\_name, appointment\_date))**

**conn.commit()**

**print("Appointment scheduled successfully!")**

**# Function to generate a bill**

**def generate\_bill():**

**patient\_id = int(input("Enter patient ID: "))**

**amount = float(input("Enter bill amount: "))**

**bill\_date = str(datetime.now().date())**

**cursor.execute('''INSERT INTO billing (patient\_id, amount, bill\_date) VALUES (?, ?, ?)''',**

**(patient\_id, amount, bill\_date))**

**conn.commit()**

**print("Bill generated successfully!")**

**# Function to view patient details**

**def view\_patient\_details():**

**patient\_id = int(input("Enter patient ID: "))**

**cursor.execute('''SELECT \* FROM patients WHERE patient\_id = ?''', (patient\_id,))**

**patient = cursor.fetchone()**

**if patient:**

**print(f"Patient ID: {patient[0]}, Name: {patient[1]}, Age: {patient[2]}, Contact: {patient[3]}")**

**else:**

**print("Patient not found!")**

**# Function to view appointments**

**def view\_appointments():**

**patient\_id = int(input("Enter patient ID to view appointments: "))**

**cursor.execute('''SELECT \* FROM appointments WHERE patient\_id = ?''', (patient\_id,))**

**appointments = cursor.fetchall()**

**if appointments:**

**for appointment in appointments:**

**print(f"Appointment ID: {appointment[0]}, Doctor: {appointment[2]}, Date: {appointment[3]}")**

**else:**

**print("No appointments found for this patient.")**

**# Function to view bills**

**def view\_bills():**

**patient\_id = int(input("Enter patient ID to view bills: "))**

**cursor.execute('''SELECT \* FROM billing WHERE patient\_id = ?''', (patient\_id,))**

**bills = cursor.fetchall()**

**if bills:**

**for bill in bills:**

**print(f"Bill ID: {bill[0]}, Amount: {bill[2]}, Date: {bill[3]}")**

**else:**

**print("No bills found for this patient.")**

**# Main program loop**

**def main():**

**while True:**

**print("\nHospital Management System")**

**print("1. Register Patient")**

**print("2. Schedule Appointment")**

**print("3. Generate Bill")**

**print("4. View Patient Details")**

**print("5. View Appointments")**

**print("6. View Bills")**

**print("7. Exit")**

**choice = input("Enter your choice: ")**

**if choice == '1':**

**register\_patient()**

**elif choice == '2':**

**schedule\_appointment()**

**elif choice == '3':**

**generate\_bill()**

**elif choice == '4':**

**view\_patient\_details()**

**elif choice == '5':**

**view\_appointments()**

**elif choice == '6':**

**view\_bills()**

**elif choice == '7':**

**print("Exiting the system.")**

**break**

**else:**

**print("Invalid choice. Please try again.")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**# Close the database connection**

**conn.close()**

Conclusion

The **Hospital Management System App** streamlines hospital operations by automating key tasks like patient registration, appointment scheduling, and billing. It improves efficiency, enhances patient care, and ensures secure management of sensitive data with role-based access. While the current version is simplified, it provides a strong foundation for further enhancements, such as real-time monitoring and better reporting. Ultimately, this system optimizes hospital workflows, leading to better patient outcomes and more effective management.