SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:ABHIJITH R	**USN: 1NT23IS005**	**Date:09/05/2025**
Lab Activity # / Task #: TASK 5	**Document name: software design**	**Submitted details:**

SOFTWARE DESIGN

1. Introduction

This document details the design, implementation, and testing approach for an **Electronic Health Records (EHR) System** catering to **healthcare organizations**. The software follows **object-oriented principles**, ensuring **scalability, modularity, security, and data integrity**.

2. System Overview

The EHR system consists of the following core modules:

- **Patient Management** – Handles patient registration, demographic information, and updating records.
- **Medical History** – Tracks diagnoses, treatments, prescriptions, and doctor assessments.
- **Billing System** – Manages invoices, payments, insurance claims, and financial transactions.
- **Security & Access Control** – Implements authentication, role-based access (RBAC), and data encryption.
- **Reporting & Analytics** – Provides insights into patient history and financial reports.

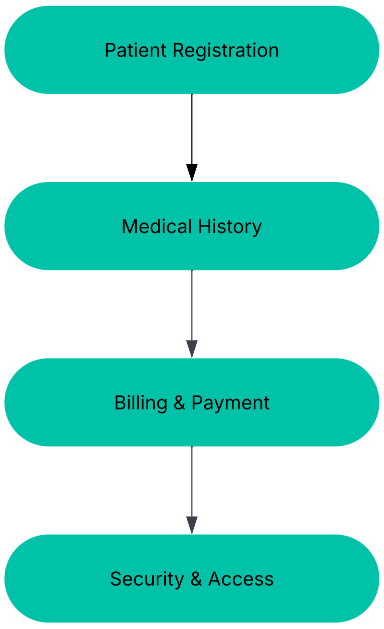
3. System Design

3.1 Data Flow Diagram (DFD - Level 1)

Illustrates how data moves through the system.

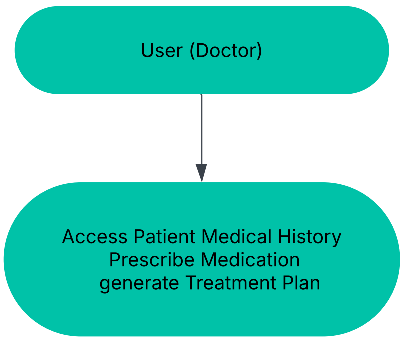
SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:ABHIJITH R	**USN: 1NT23IS005**	**Date:09/05/2025**
Lab Activity # / Task #: TASK 5	**Document name: software design**	**Submitted details:**



3.2 Use Case Diagram (UML Representation)

Shows interactions between actors and the system.



SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:ABHIJITH R	**USN: 1NT23IS005**	**Date:09/05/2025**
Lab Activity # / Task #: TASK 5	**Document name: software design**	**Submitted details:**

4. Database Design

The system uses a **relational database model** (MySQL or PostgreSQL).

4.1 Patient Table

```
CREATE TABLE Patients (

  Patient_ID INT PRIMARY KEY,

  Name VARCHAR(100),

  Age INT,

  Gender VARCHAR(10),

  Contact_Info VARCHAR(150)

);
```

4.2 Medical History Table

```
CREATE TABLE Medical_History (

  Record_ID INT PRIMARY KEY,

  Patient_ID INT,

  Diagnosis TEXT,

  Treatment TEXT,

  Prescription TEXT,

  Date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  FOREIGN KEY (Patient_ID) REFERENCES Patients(Patient_ID)

);
```

4.3 Billing Table

```
CREATE TABLE Billing (

  Bill_ID INT PRIMARY KEY,
```

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:ABHIJITH R	**USN: 1NT23IS005**	**Date:09/05/2025**
Lab Activity # / Task #: TASK 5	**Document name: software design**	**Submitted details:**

```
  Patient_ID INT,
  Amount DECIMAL(10,2),
  Status VARCHAR(50),
  FOREIGN KEY (Patient_ID) REFERENCES Patients(Patient_ID)
);
```

5. API Design (RESTful Architecture)

Provides endpoints for different functionalities.

5.1 Patient Registration API

```
POST /api/patient/register
{
  "name": "John Doe",
  "age": 30,
  "gender": "Male",
  "contact_info": "john.doe@example.com"
}
```

RESPONSE :

```
{
  "message": "Patient registered successfully",
  "patient_id": 101
}
```

5.2 Fetch Medical History API

```
GET /api/medical-history/{patient_id}
```

RESPONSE :

```
{
  "patient_id": 101,
  "diagnosis": "Flu",
  "treatment": "Rest and hydration",
  "prescription": "Paracetamol",
  "date": "2025-05-09"
}
```

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:ABHIJITH R	**USN: 1NT23IS005**	**Date:09/05/2025**
Lab Activity # / Task #: TASK 5	**Document name: software design**	**Submitted details:**

6. Code Implementation

6.1 Object-Oriented Python Classes

The Patient class represents a core entity in the EHR system, encapsulating attributes and behaviors related to patient data. It includes properties such as patient_id, name, age, gender, and contact_info, and provides a method to display the patient's information in a structured format.

```
class Patient:
    def __init__(self, patient_id, name, age, gender, contact_info):
        self.patient_id = patient_id
        self.name = name
        self.age = age
        self.gender = gender
        self.contact_info = contact_info

    def display_info(self):
        return f"Patient ID: {self.patient_id}, Name: {self.name}, Age: {self.age}, Gender: {self.gender}, Contact:
{self.contact_info}"
```

6.2 Billing Class Implementation

The `Billing` class handles all billing-related operations, such as storing billing details and generating patient invoices. This ensures a clear separation of financial logic from clinical records, enhancing modularity.

```
class Billing:
    def __init__(self, bill_id, patient_id, amount, status):
        self.bill_id = bill_id
        self.patient_id = patient_id
        self.amount = amount
        self.status = status  # Paid, Unpaid, Pending

    def generate_invoice(self):
        return f"Invoice - ID: {self.bill_id}, Patient ID: {self.patient_id}, Amount: ${self.amount}, Status:
{self.status}"
```

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:ABHIJITH R	USN: 1NT23IS005	Date:09/05/2025
Lab Activity # / Task #: TASK 5	Document name: software design	Submitted details:

7. Testing and Validation

7.1 Unit Test for Billing Module

```
import unittest

class TestBilling(unittest.TestCase):
    def test_invoice_generation(self):
        invoice = Billing(5001, 101, 150.75, "Paid")
        self.assertIn("Invoice - ID: 5001", invoice.generate_invoice())\
```

7.2 Security Testing

Ensuring secure **role-based access** and **authentication**.

The User class demonstrates role-based access control (RBAC), which restricts access to modules based on user roles. This is critical for complying with healthcare regulations such as HIPAA, where access to sensitive patient data must be strictly controlled.

```
if __name__ == "__main__":
    unittest.main()


class User:
    def __init__(self, username, role):
        self.username = username
        self.role = role  # Admin, Doctor, Nurse, Billing Staff

    def has_access(self, module):
        access_control = {
            "Admin": ["Patient", "MedicalHistory", "Billing", "Security"],
            "Doctor": ["MedicalHistory"],
            "Billing Staff": ["Billing"],
        }
        return module in access_control.get(self.role, [])

# Example Usage
user1 = User("alice_admin", "Admin")
print(user1.has_access("Billing"))  # Output: True
```

SOFTWARE ENGINEERING LABORATORY (22IS43) TASK EXECUTION SHEET

Name:ABHIJITH R	**USN: 1NT23IS005**	**Date:09/05/2025**
Lab Activity # / Task #: TASK 5	**Document name: software design**	**Submitted details:**

8. Conclusion

This **EHR System** integrates **object-oriented architecture, modular database design, API structures, and robust security mechanisms** to provide a scalable solution for **healthcare data management**. The software ensures **efficient patient data storage**, **billing management**, and **secure access control**.