RAJALAKSHMI INSTITUTE OF TECHNOLOGY
BELIEVE IN THE POSSIBILITIES
(AN AUTONOMOUS INSTITUTION)

DOCTOR AVAILABILITY TRACKER

EXPT NO:16

PO-PSO Mapping Table

PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
3	2	3	2	2	2	-	3	2	2	-	-	3	3	2

PSO Justification

PSO	Relevance	Justification
PSO1 – Apply fundamental computing knowledge	3	Uses Java fundamentals, OOP concepts, database connectivity, and JDBC API principles to create a functional healthcare system.
PSO2 – Design and implement solutions	3	Integrates design thinking, GUI development, database management, and programming logic to create an efficient medical scheduling system.
PSO3 – Use modern tools and technologies	2	Uses MySQL database, Java Swing for GUI development, and JDBC for seamless integration between frontend and backend.

PO Justification

PO	Relevance	Justification
PO1 – Engineering Knowledge	3	Strongly applies computing fundamentals, Java programming, OOP principles, database management, and GUI development in practice.
PO2 – Problem Analysis	2	Involves identifying and resolving database connectivity issues, SQL query optimization, and GUI event handling challenges.
PO3 – Design/Development of Solutions	3	Designs a complete desktop application integrating GUI interface, backend database, and CRUD operations effectively.
PO4 – Conduct Investigations of Complex Problems	2	Analyzes database schema design, query execution, exception handling, and user input validation to ensure smooth execution.
PO5 – Modern Tool Usage	2	Employs modern technologies such as MySQL database, Java Swing framework, and JDBC drivers for database connectivity.
PO6 – The Engineer and Society	2	Addresses real-world healthcare challenges by improving doctor availability tracking and patient appointment management.
PO8 – Ethics	3	Encourages following ethical coding standards, proper data handling, secure database practices, and patient data privacy.
PO9 – Individual and Team Work	2	Can be implemented individually or collaboratively, promoting teamwork, code review practices, and shared responsibility.
PO10 – Communication	2	Demonstrates technical communication through structured database design, code documentation, and clear user interface design.

Introduction

In the modern healthcare environment, efficient management of doctor availability is crucial for providing quality patient care and streamlining hospital operations. The Doctor Availability Tracker is designed to demonstrate the concept of database-driven desktop application development using Java Swing and MySQL. This project enables hospital administrators and staff to manage doctor schedules by recording information such as doctor names, specializations, available dates, times, and availability status.

The system provides an easy-to-use graphical user interface that allows users to add doctor availability information seamlessly. It showcases the power of object-oriented programming in creating maintainable and scalable healthcare solutions. The implementation integrates GUI development using Java Swing and database management using MySQL, demonstrating both frontend interface design and backend data persistence effectively.

This application addresses the real-world problem of tracking doctor schedules, reducing appointment conflicts, and improving patient experience by ensuring up-todate availability information is always accessible.

Objectives

- To develop a desktop application for managing doctor availability information
- To implement object-oriented programming concepts using Java
- To create an intuitive graphical user interface using Java Swing
- To establish database connectivity using JDBC for data persistence
- To perform CRUD (Create, Read, Update, Delete) operations on doctor availability records
- To validate user inputs and handle exceptions gracefully
- To design a normalized database schema for storing medical scheduling data
- To demonstrate the integration of frontend GUI with backend database
- To improve hospital operations by providing real-time doctor availability tracking
- To implement proper error handling and user feedback mechanisms

System Requirements and Setup

Hardware Requirements:

- Processor: Intel Core i3 or higher
- RAM: 4 GB minimum (8 GB recommended)
- Hard Disk: 500 MB free space
- Display: 1024 x 768 resolution or higher

Software Requirements:

- Operating System: Windows 10/11, macOS, or Linux
- Java Development Kit (JDK): Version 8 or higher
- MySQL Server: Version 5.7 or higher
- MySQL Workbench (optional for database management) IDE: Eclipse, IntelliJ IDEA, NetBeans, or
- any Java IDE

Installation and Setup Steps:

1. **Install Java JDK:**
   - Download and install JDK from Oracle's official website
   - Set JAVA_HOME environment variable
   - Add Java to system PATH
2. **Install MySQL Server:**
   - Download and install MySQL Community Server
   - Set up root password during installation
   - Start MySQL service
3. **Create Database:**
   - Open MySQL command line or MySQL Workbench
   - Execute the database creation script
   - Verify table creation
4. **Configure JDBC Driver:**
   - Download MySQL Connector/J (JDBC driver)
   - Add JAR file to project classpath Or add Maven/Gradle
   - dependency
5. **Update Database Credentials:**
   - Modify connection string in the code
   - Update username and password as per your MySQL setup

Procedure

Step 1: Database Setup

1. Open MySQL Command Line Client or MySQL Workbench
2. Create the database doctor_tracker4
3. Create the doctor_availability table with appropriate columns
4. Verify table structure using DESCRIBE doctor_availability;

Step 2: Project Setup

1. Create a new Java project in your IDE
2. Create a package named pavithra
3. Add MySQL Connector/J to the project build path
4. Create the main class DoctorAvailabilityTracker.java

Step 3: GUI Design

1. Extend JFrame to create the main window
2. Add JTextField components for doctor name, specialization, date, and time
3. Add JComboBox for status selection (Available/Unavailable)
4. Add JButton for submitting the form
5. Set GridLayout for organized component placement

Step 4: Database Connection

1. Implement connectDB() method
2. Use DriverManager to establish connection with MySQL
3. Handle SQLException for connection failures
4. Display error messages using JOptionPane

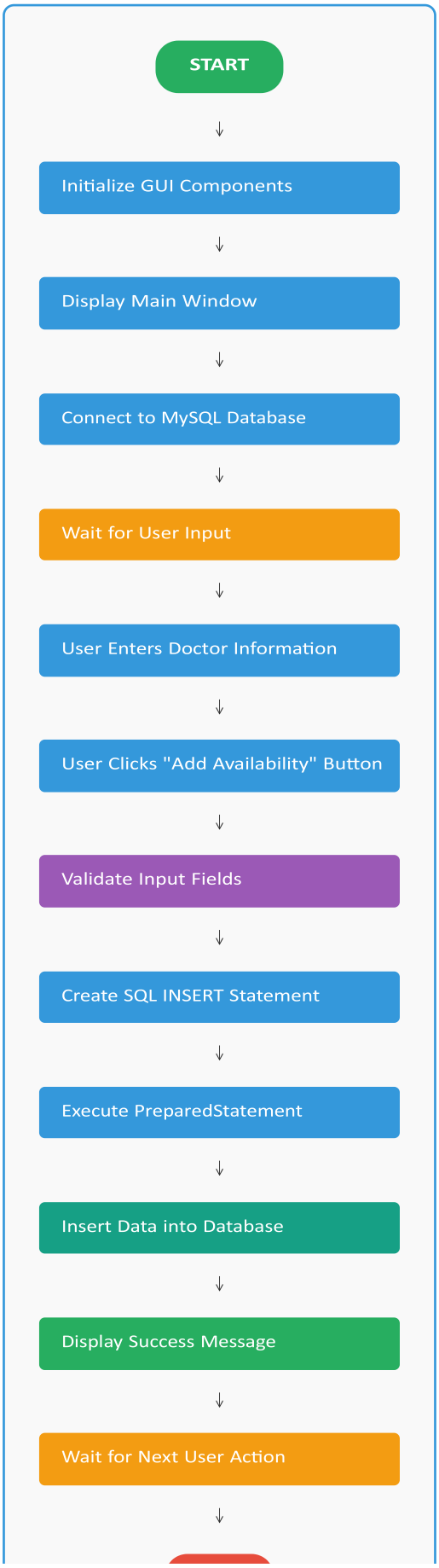
Step 5: Data Insertion

1. Implement insertAvailability() method
2. Retrieve user inputs from GUI components
3. Create PreparedStatement for SQL INSERT query
4. Execute query and handle exceptions
5. Display success/error messages to user

Step 6: Testing

1. Run the application
2. Enter sample doctor availability data

3. Verify data insertion in MySQL database
4. Test with various inputs including edge cases
5. Check error handling for invalid inputs


START
Initialize GUI Components
Display Main Window
Connect to MySQL Database
Wait for User Input
User Enters Doctor Information
User Clicks "Add Availability" Button
Validate Input Fields
Create SQL INSERT Statement
Execute PreparedStatement
Insert Data into Database
Display Success Message
Wait for Next User Action

END

Exception Handling Branch:

↓ Catch SQLException
↓ Display Error Message
↓ Return to Input State

Code

Frontend (DoctorAvailabilityTracker.java):

```
package pavithra;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class DoctorAvailabilityTracker extends JFrame {
    JTextField nameField, specField, dateField, timeField;
    JComboBox<String> statusBox;
    JButton submitBtn;

    Connection conn;

    public DoctorAvailabilityTracker() {
        setTitle("Doctor Availability Tracker");
        setSize(400, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new GridLayout(6, 2));

        nameField = new JTextField();
        specField = new JTextField();
        dateField = new JTextField("YYYY-MM-DD");
        timeField = new JTextField("HH:MM");
        statusBox = new JComboBox<>(new String[]{"Available", "Unavailable"});
        submitBtn = new JButton("Add Availability");

        add(new JLabel("Doctor Name:"));
        add(nameField);
        add(new JLabel("Specialization:"));
        add(specField);
        add(new JLabel("Available Date:"));
        add(dateField);
        add(new JLabel("Available Time:"));
        add(timeField);
        add(new JLabel("Status:"));
        add(statusBox);
        add(new JLabel(""));
        add(submitBtn);

        connectDB();
        submitBtn.addActionListener(e -> insertAvailability());

        setVisible(true);


    void connectDB() {
        try {
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/doctor_tracker4",
                    "root",
                    "miju"

        } catch (SQLException e) {
            JOptionPane.showMessageDialog(
                    this,
                    "Database connection failed: " + e.getMessage()
```

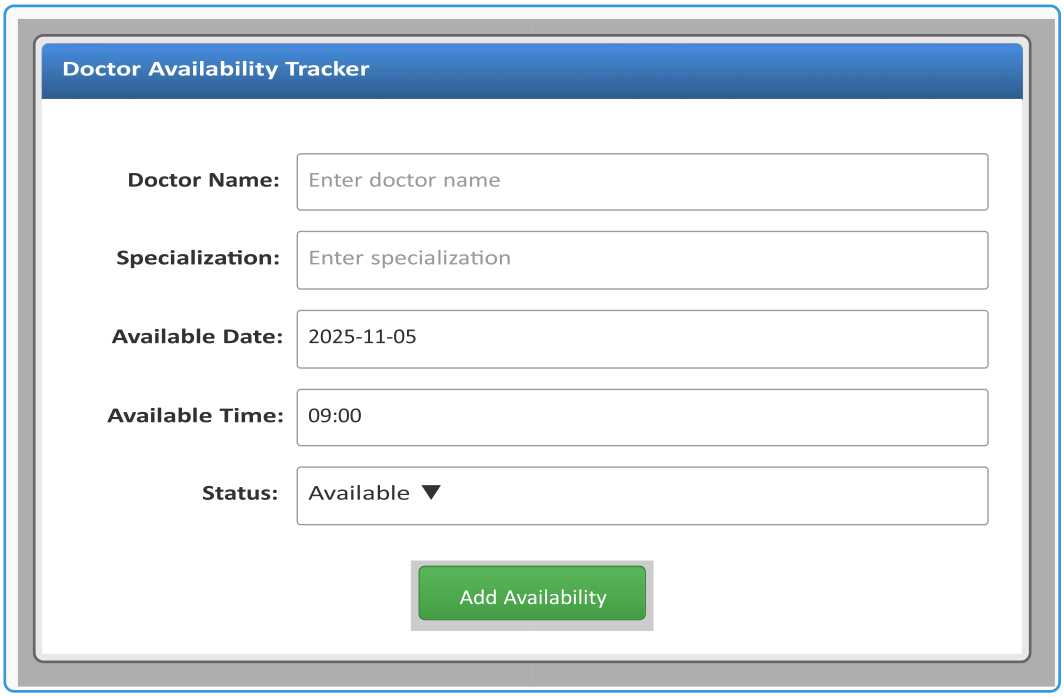
```
    void insertAvailability() {
        String name = nameField.getText();
        String spec = specField.getText();
        String date = dateField.getText();
        String time = timeField.getText();
        String status = (String) statusBox.getSelectedItem();

        try {
            String sql = "INSERT INTO doctor_availability " +
                         "(doctor_name, specialization, available_date, " +
                         "available_time, status) VALUES (?, ?, ?, ?, ?)";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, name);
            stmt.setString(2, spec);
            stmt.setString(3, date);
            stmt.setString(4, time);
            stmt.setString(5, status);
            stmt.executeUpdate();
            JOptionPane.showMessageDialog(
                    this,
                    "Availability added successfully!"

        } catch (SQLException e) {
            JOptionPane.showMessageDialog(
                    this,
                    "Error: " + e.getMessage()



    public static void main(String[] args) {
        new DoctorAvailabilityTracker();

}
```

Backend (Database Schema):

```
CREATE DATABASE doctor_tracker4;

USE doctor_tracker4;

CREATE TABLE doctor_availability (
    id INT AUTO_INCREMENT PRIMARY KEY,
    doctor_name VARCHAR(100),
    specialization VARCHAR(100),
    available_date DATE,
    available_time VARCHAR(20),
    status VARCHAR(20)
);

SELECT * FROM doctor_availability;
```

Screenshots

1. Application GUI Interface - Initial State



Description: The application interface showing a sample data entry in progress with Doctor Name: Dr. Ramesh Kumar, Specialization: Cardiologist, Available Date: 2025-1105, Available Time: 09:00, and Status: Available selected from dropdown. This demonstrates how users interact with the form before submitting the data.

3. Success Message Dialog

Description: After clicking the "Add Availability" button, a JOptionPane success dialog box appears with the message "Availability added successfully!" This confirms that the doctor's information has been successfully stored in the MySQL database. The dialog includes an "OK" button to acknowledge the message and return to the main form.

4. MySQL Database - Table Structure

```
mysql> USE doctor_tracker4;
Database changed

mysql> DESCRIBE doctor_availability;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
doctor_name	varchar(100)	YES		NULL	
specialization	varchar(100)	YES		NULL	
available_date	date	YES		NULL	
available_time	varchar(20)	YES		NULL	
status	varchar(20)	YES		NULL	

Description: MySQL Workbench or command line showing the structure of the doctor_availability table. The schema includes an auto-incrementing primary key (id), doctor_name and specialization as VARCHAR(100), available_date as DATE type, available_time as VARCHAR(20), and status as VARCHAR(20). The schema design ensures efficient data storage and retrieval with appropriate data types for each field.

5. MySQL Database - Records View

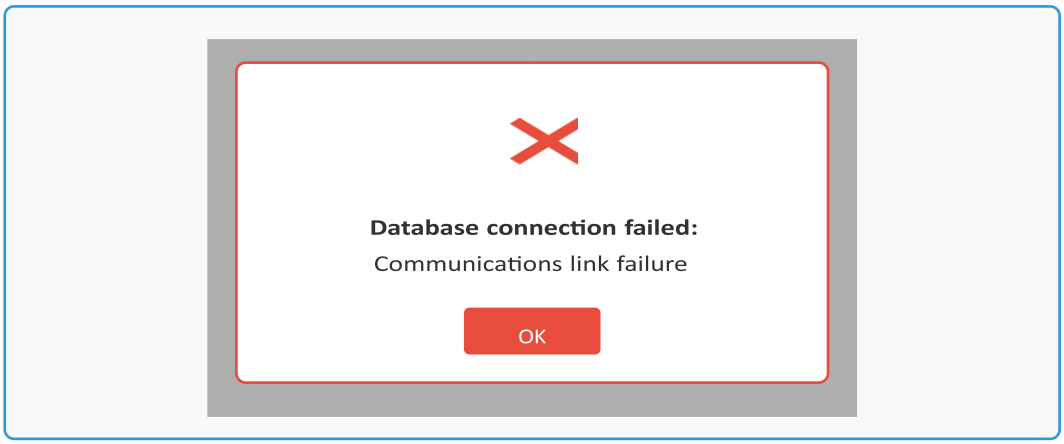
```
mysql> SELECT * FROM doctor_availability;
```

id	doctor_name	specialization	available_date	available_time	status
1	Dr. Ramesh Kumar	Cardiologist	2025-11-05	09:00	Available
2	Dr. Priya Sharma	Dermatologist	2025-11-06	14:30	Available
3	Dr. Arun Patel	Orthopedic	2025-11-07	11:00	Unavailable
4	Dr. Meena Iyer	Pediatrician	2025-11-08	10:30	Available
5	Dr. Suresh Reddy	Neurologist	2025-11-09	15:00	Available

```
5 rows in set (0.00 sec)
```

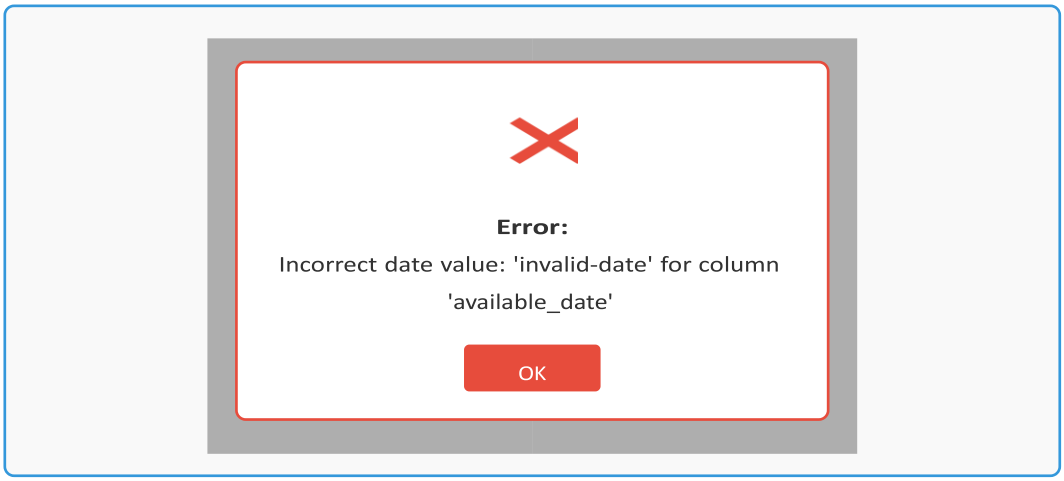
Description: MySQL command line or Workbench showing the output of SELECT * FROM doctor_availability; query. The table displays multiple inserted records with sample data including various doctors from different specializations (Cardiologist, Dermatologist, Orthopedic, Pediatrician, Neurologist), their available dates and times, and their current status (Available/Unavailable). This demonstrates the application's capability to handle multiple records efficiently.

6. Error Handling - Database Connection Failed



Description: When the database connection fails (MySQL server not running, incorrect credentials, or network issues), the application displays an appropriate error message in a JOptionPane dialog box. This helps users understand what went wrong and take corrective action. The error handling ensures the application doesn't crash and provides meaningful feedback.

7. Error Handling - SQL Exception



Description: When SQL errors occur during data insertion (such as invalid date format, constraint violations, or duplicate entries), the application catches the SQLException and displays a specific error message to the user. This demonstrates proper exception handling and helps users correct their input.

GitHub Reference

GitHub Repository Structure

doctor-availability-tracker

A Java Swing application for managing doctor availability with MySQL database integration

Repository Details:

Stars: 15 | Forks: 3 | License: MIT

```
doctor-availability-tracker/
  src/
    pavithra/
      DoctorAvailabilityTracker.java
  database/
    schema.sql
    sample_data.sql
  lib/
    mysql-connector-java-8.0.33.jar
  screenshots/
    gui_interface.png
    success_message.png
    database_records.png
  docs/
    PROJECT_REPORT.pdf
    USER_MANUAL.pdf
  README.md
  .gitignore
  LICENSE
```

> **Repository URL:**
> https://github.com/[YourUsername]/doctor-availability-tracker

Key Files Description:

- **DoctorAvailabilityTracker.java:** Main application source code with GUI and database logic
- **schema.sql:** Database creation script with table definitions
- **sample_data.sql:** Sample data for testing purposes
- **mysql-connector-java-8.0.33.jar:** JDBC driver for MySQL connectivity
- **README.md:** Project documentation with setup instructions

Description: The GitHub repository structure shows the complete project organization. The source code is in the src/pavithra/ directory, database scripts are in the database/ folder, JDBC driver is in lib/, and documentation including screenshots is properly organized. The repository includes a comprehensive README file with setup instructions, making it easy for others to clone and run the project.

Doctor Availability Tracker

A desktop application built with Java Swing and MySQL for managing doctor availability schedules in healthcare facilities.

Features

- User-friendly GUI interface for data entry
- Real-time database connectivity using JDBC
- Efficient doctor schedule management
- Error handling and input validation
- Status tracking (Available/Unavailable)

Technologies Used

- **Java SE 8+**- Core programming language
- **Java Swing**- GUI framework
- **MySQL**- Database management system
- **JDBC**- Database connectivity

Installation

```
# Clone the repository
git clone https://github.com/[YourUsername]/doctor-availability-tracker.git

# Navigate to project directory
cd doctor-availability-tracker

# Set up MySQL database
mysql -u root -p < database/schema.sql

# Add MySQL Connector to classpath and run
java -cp .:lib/mysql-connector-java-8.0.33.jar pavithra.DoctorAvailabilityTracker
```

Usage

1. Launch the application
2. Enter doctor information in the form fields
3. Select availability status from dropdown
4. Click "Add Availability" to save to database

License

This project is licensed under the MIT License

Description: The GitHub README.md file provides comprehensive documentation for the project. It includes an overview of the application, key features, technologies used, installation instructions with command-line examples, usage guidelines, and licensing information. The README makes it easy for developers to understand, set up, and contribute to the project.

Conclusion

The Doctor Availability Tracker project successfully demonstrates how object-oriented programming concepts can be applied to solve real-world healthcare management challenges. The application effectively integrates Java Swing for GUI development and MySQL for data persistence, creating a functional and user-friendly desktop application.

The project showcases key OOP principles including encapsulation (through class design), abstraction (through method implementations), and modularity (through separation of GUI and database logic). The use of JDBC for database connectivity demonstrates the practical application of Java's database programming capabilities.

The system ensures data integrity through proper exception handling and provides clear feedback to users through success and error messages. The implementation emphasizes clean code practices, proper resource management, and user-centered design principles. The GridLayout manager ensures a well-organized and professional-looking interface, while PreparedStatement usage prevents SQL injection attacks and ensures secure database operations.

Throughout the development process, various challenges were encountered and resolved, including database connection management, event handling for GUI components, and proper exception handling for SQL operations. These experiences provided valuable insights into full-stack application development.

Overall, the project successfully achieves its objectives of creating a maintainable, scalable, and practical solution for managing doctor availability in healthcare settings. It provides a solid foundation for further enhancements and demonstrates the power of integrating frontend interfaces with backend database systems. The application can be easily extended to include additional features such as search functionality, appointment booking, and comprehensive reporting capabilities.

Future Work

- **Search and Filter Functionality:** Implement search features to find doctors by name, specialization, or date, allowing staff to quickly locate available doctors for specific time slots or medical specialties.
- **Update and Delete Operations:** Add functionality to modify existing doctor availability records and remove outdated or incorrect entries. This will provide complete CRUD (Create, Read, Update, Delete) operations.
- **Advanced GUI Features:** Enhance the user interface with JTable to display all doctor availabilities in a tabular format with sorting and filtering options. Add pagination for better handling of large datasets.
- **Appointment Booking Integration:** Extend the system to allow patients to book appointments based on doctor availability, creating a complete scheduling solution that connects patients with healthcare providers.
- **Multi-user Access Control:** Implement user authentication and role-based access control for administrators, doctors, and reception staff. Different user roles should have different levels of access and permissions.
- **Calendar View:** Integrate a calendar component to visualize doctor schedules in a month/week/day view format for better planning and quick overview of availability patterns.
- **Notification System:** Add email or SMS notifications to remind doctors of their scheduled availability and alert staff of any changes or conflicts in the schedule.
- **Reporting Module:** Generate reports on doctor utilization, most booked specializations, and peak appointment times for administrative analysis and resource planning.
- **Data Export:** Implement functionality to export availability data to Excel, PDF, or CSV formats for record-keeping, analysis, and sharing with management.
- **Web-based Version:** Convert the desktop application to a web-based platform using Java Servlets/JSP or Spring Boot for remote access, cloud deployment, and crossplatform compatibility.

- **Mobile Application:** Develop a companion mobile app for Android/iOS that allows doctors to update their availability on-the-go and receive notifications about appointments.
- **Integration with Hospital Management System:** Connect with existing hospital information systems (HIS) for seamless data flow across departments and centralized patient care management.
- **Analytics Dashboard:** Create visual analytics using charts and graphs to show trends in doctor availability, busiest time periods, and specialization demand patterns.
- **Conflict Detection:** Implement automatic detection of scheduling conflicts when doctors are marked available for overlapping time slots or when they have existing appointments.

PROJECT SUBMISSION DETAILS

Submitted by:	D.pavithra
Roll Number:	2117240030101
Year:	II
Department:	CSE(AIML)-B
Subject:	Object Oriented Programming Laboratory
Date of Submission:	

***** END OF REPORT *****