Hospital Management Database

Mini Project Report -Database Lab (DSE 2260)

Department of Data Science & Computer Applications



B. Tech Data Science

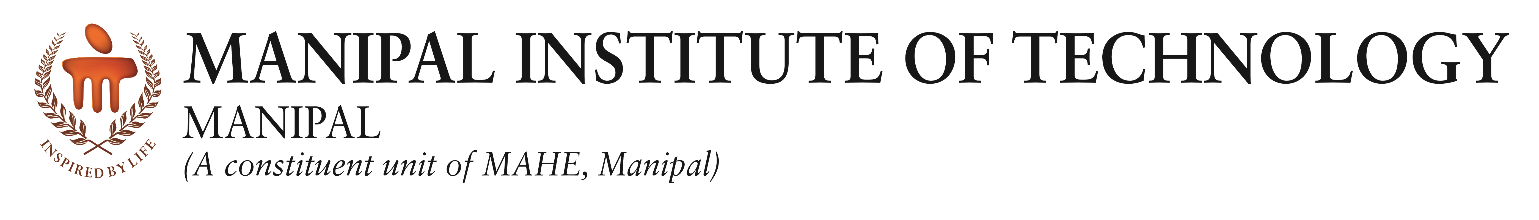
4th Semester – Batch: A1/A2/B1/B2

Submitted By

|  |  |
| --- | --- |
| Sonakshi Badlani | 210968055 |
| Bhavna Aggarwal | 210968060 |
| Alankrith Addala | 210968064 |
| Bolampalli Vivek | 210968056 |
| Ambati Aarush | 210968066 |

**Mentored By**

|  |  |
| --- | --- |
| Vinayak M | Archana H/ Dr. Natesh |
| Assistant Professor-Senior | Assistant Professor-Senior |
| DSCA, MIT | DSCA, MIT |



Date:

**CERTIFICATE**

This is to certify that the students, Sonakshi Badlani, Bhavna Aggarwal, Alankrith Addala, Ambati Aarush and Bollampalli Vivek have successfully executed a mini project titled “…….” rightly brining fore the competencies and skill sets they have gained during the course- Database Lab (DSE 2262 & DSE), thereby resulting in the culmination of this project.

|  |  |
| --- | --- |
| **Vinayak M** | **Archana H / Dr. Natesh** |
| **Assistant Professor-Senior** | **Assistant Professor-Senior** |
| **DSCA, MIT** | **DSCA, MIT** |

**ABSTRACT**

Hospital Management System is an organised system designed to make the day to day operations of a hospital more efficient and error free. It takes care of updated records, database treatments, status of a patient, fee transactions. Along with this, it also stores information regarding doctors in charge and accepts feedback for further consultations.

Contents

**1. Introduction 4**

**2. Synopsis 4**

**2.1 Proposed System 4**

**2.2 Objectives 4**

**3. Data Requirements 5**

**4. Functional Requirements 6**

**5. Detailed Design 8**

**5.1 ER Diagram 8**

**5.2 Schema Diagram 9**

**5.3 Data Dictionary 10**

**5.4 Relational Model Implementation 11**

**5.5 Queries 13**

**6. Implementation Functional Requirements**

**7. Result**

**8. Conclusion and Future Work 18**

**Chapter 1**

**Introduction**

We are working with Hospital Management Database, the database is expected to be sorted and easy to read so that in case of an event where data is needed urgently, it should be provided. Hospitals are the most important part of our lives, trying to provide the best medical facilities to people suffering from various type of illness, which may be due to change in climate conditions, increased work-load, emotional trauma stress etc. It is very much difficult for the hospital to maintain its day-to-day activities and records manually. That is why a database is required to keep records of all type of activities of a hospital.

The Hospital Management system stores full history along with the details of a patient, including the doctors who treated them previously . It provides a unique id for every staff and patient. Users can look up availability of doctors using the id. The data can be retrieved easily and is well protected.

It is usually a long process to find out the old medical records of a patient which are often necessary in the current treatment hence, the main aim of the Hostel Management System is to have the history of a patient at finger tips for future references. It reduces manual errors and ensures the data is updated regularly and allow ease and speed of information storage and retrieval.  

We will have the following data in this database:

* Patients information: name, surname, date of birth, insurance number, age, etc.
* We can have records for the patient's chronic diseases
* Doctor's information: name, surname, date of birth, employment date, specialty, salary, etc.
* Patient's treatments and doctor consultations - date, patient Id, doctor Id, notes from the consultation, diagnosis

**Chapter 2**

**Synopsis**

**2.1 Proposed System**

Hospital Data needs to be stored in an organised manner. It should be kept up to date and all patient current as well as past records must be available in minimum amount of time.

**2.2 Objectives**

The main objectives of the Hospital Management Database are:

* Maintain the medical records of the patient
* Maintain the contact details of the patient
* Keep track of the appointment dates
* Save the insurance information for later reference
* Tracking the bill payments.
* Maintain all records related to the doctors, includes: the day they visit, specialty.
* Access of the medical records of a patient to the doctors.

**Chapter 3**

**Data Requirements**

**3.1** **Purpose, scope and overview**

The application requires data describing the Patients, medical history, doctors, appointment availability and data for the fee transaction operation.

**3.2 Requirements**

**3.2.1 Patient Demographics**

|  |  |  |
| --- | --- | --- |
| **Data Name** | **Description** | **Example** |
| Patient\_Id | An unique alphanumeric value to identify each patient. | 12345 |
| Name | Name of the patient | Alankrith |
| DOB | Date of birth of the patient | 1 Oct 1999 |
| Gender | Gender of  the patient | M |
| Last\_date\_of\_visit | The date that the patiend visited the hospital last | 12 Oct 2021 |

**3.2.2 Patient Medical History**

|  |  |  |
| --- | --- | --- |
| Patient\_id | An unique alphanumeric value to identify each patient. | 12345 |
| Latest\_Past\_disease | A record of the recent or current illness or disease of the patient | Anemia |
| Latest\_Medication | Prescribed medication that the patient is taking so as to ensure that no medication contradicting it is prescribed | Iron Supplement |
| Doctor1 | Id of the primary doctor handling the patient’s latest disease | 3458 |
| Doctor2 | Id number of any other doctor handling the patient | 1365 |
| Date\_diagnosed | Date of the diagnoses of the latest disease | 13 Nov 2022 |
| Surgeries | Past surgeries info (if any) | Cataract |
| Allergies | A record of the list of things the patient is allergic to so as to prevent any mishap | eczema |

**3.2.3 Transaction**

|  |  |  |
| --- | --- | --- |
| Trans\_no | A unique id to identify each transaction | 2222 |
| Date | Date of transaction | 13 Nov 2022 |
| Amount\_paid | Amount paid by each individual patient | 23458 |
| Dr\_id | A unique numeric id to identify primary doctor | S23456 |
| Amt\_insured | Amount of the bill paid by insurance |  |

**3.2.4 Appointments**

|  |  |  |
| --- | --- | --- |
| Patient\_id | An unique alphanumeric value to identify each patient. | 34632 |
| Doctor\_id | A unique id to identify each doctor | S12347 |
| Date | Date of the particular appointment | 5 Nov 2022 |
| Time | Time of the appointment | 6:00 |
| Payment\_Status | Status of the payment – done or not | Done |
| Dept\_id | Department in which the doctor belongs | D3 |
| Trans\_id | A unique id to identify each transaction |  |

**3.2.5 Doctor**

|  |  |  |
| --- | --- | --- |
| Id | A unique id to identify each doctor | S23465 |
| Dept\_id | Department in which the doctor belongs | D3 |
| Name | Name of the doctor | Ramesh |
| Years of experience | Years of practicing medicine | 7 |

**3.2.6 Department**

|  |  |  |
| --- | --- | --- |
| Dept\_id | A unique id to identify each department | D3 |
| Dept\_Name | Name of department | OPD |

**3.2.7 Mobile Numbers**

|  |  |  |
| --- | --- | --- |
| Patient\_id | An unique alphanumeric value to identify each patient. | P1 |
| Mobile number | Patient mobile number | 9876543210 |

**Chapter 4**

**Functional Requirements**

What is a Functional Requirement?

A Functional Requirement (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behaviour, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. ]

Briefly write overview of functionalities provided by the application in terms of different modules.

**4.1 Reception Module**

**4.1.1 New Patient Registration**

Registering a new Patient

|  |  |
| --- | --- |
| INPUT | Patient\_id,Name,DOB,Gender, Mobile |
| Processing | If patient is not registered  Check for validity of phone number by prompting to enter OTP |
| OUTPUT | Patient Successfully registered message or highlight the information entered which is wrong or contradicting and allow to renter. |

**4.1.2 New Appointment**

Scheduling a new Appointment

|  |  |
| --- | --- |
| INPUT | Patient\_id, Doctor\_id,Date,Time,Amount |
| Processing | Check if Doctor available on Date Time,  If available  Schedule appointment and update appointments table  Else  Display message |
| OUTPUT | Appointment scheduled or Time slot unavailable.  Payment status |

**4.1.3 Patient Medical History Update**

The receptionist/Doctor can manually update Medical History of patients from other Hospitals/

|  |  |
| --- | --- |
| INPUT | Patient\_id, Past\_dis, Medication,Doctor1,Doctor2, Date\_diagnosed, Surgeries, Allergies |
| Processing | Update PMH table |
| OUTPUT | PMH updated |

**4.2 Patient Medical History Module**

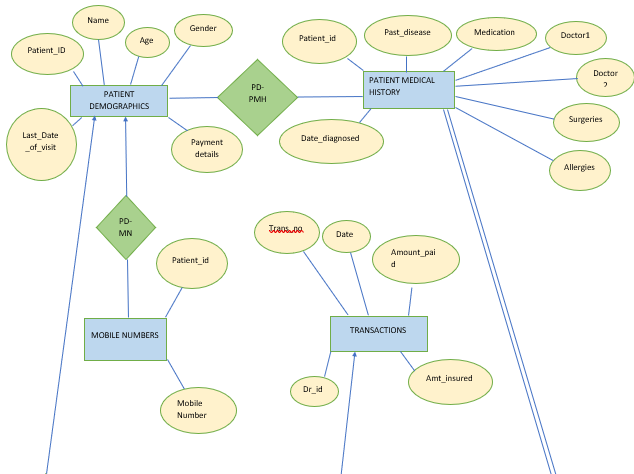
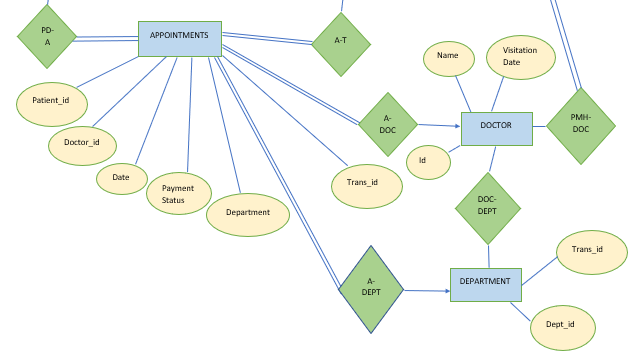
Will be used by the doctors to review patient medical records

|  |  |
| --- | --- |
| INPUT | Patient\_Id |
| Processing | Fetch Patient Records |
| OUTPUT | Patient Medical Records |

**Chapter 5:**

**Detailed Design**

**5.1 ER Diagram**

****

**5.2 Schema Diagram**

**Patient Demographics**(Patient\_Id,Name,Age,Gender,Last\_date\_of\_visit)

**Patient Medical History**(Patient id,Past disease,Medication,Doctor1,Doctor2,Date\_diagnosed,Surgeries,Allergies)

Patient\_id references Patient Demographics

Doctor1,Doctor2 references Doctor

**Transaction**(Trans\_no,Date,Amount\_paid,Dr\_id,Amt\_insured)

Dr\_id references Doctor

**Appointments**(Patient\_id,Doctor\_id,Date,Time,Payment\_Status,Department\_id,Trans\_id)

Patient\_id references Patient Demographics

Doctor\_id references Doctor

Department\_id references Department

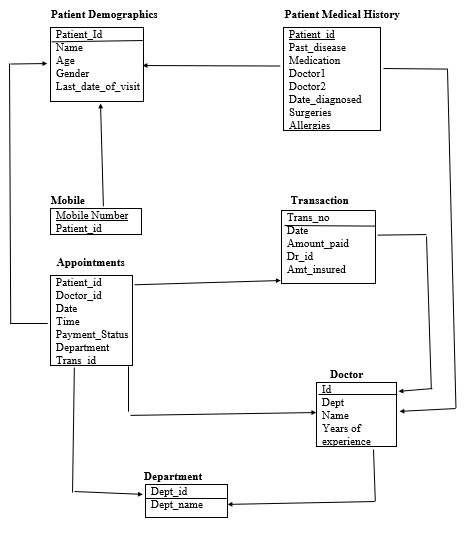
Trans\_id references transaction

**Doctor**(Id,Dept\_id,Name,Years of Experience)

Dept references Department

**Department**(Dept\_id,Dept\_name)

**Mobile\_Number**(Patient\_id, Mobile Number)

****

**5.3 Data Dictionary**

**Patient Demographics**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data type (size) | Constraint | Constraint Name |
| Patient\_id | Varchar2(10) | Primary Key | Pkey\_id |
| Name | Varchar2(40) |  |  |
| Dob | Date | Less than or equal to current date | Dob\_valid |
| Gender | Varchar(2) | M,F or O | Gender\_const |
| Last date of visit | Date | Less than or equal to current date | Visit\_Date\_valid |

**Dept**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data type (size) | Constraint | Constraint Name |
| Dept\_id | Varchar2(10) | Primary key | Pkey\_dept\_id |
| Dept\_name | Varchar2(10) | ‘OPD’,’Cardiac’,’Opthal’ | Dept\_valid |

**Doctor**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data type (size) | Constraint | Constraint Name |
| Doctor\_id | Varchar2(10) | Primary key | Pkey\_doctor\_id |
| Dept\_id | Varchar2(10) | References Dept | Rkey\_dept\_doc |
| Name | Varchar2(40) |  |  |
| Years\_exp | Number(2,2) | >=0 | Years\_exp\_positive |

**Transaction**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data type (size) | Constraint | Constraint Name |
| Transaction\_id | Varchar2(10) | Primary Key | Pkey\_trans |
| Date | Date | Less than or equal to current date | Transaction\_date\_valid |
| Amt\_paid | Number(10) | >0 | Amount\_positive |
| Doctor\_id | Varchar(10) | References Doctor | Rkey\_doctor\_trans |
| Amt\_insured | Number(10) | >0, <amt paid | Insured\_amt\_positive. Insurance\_paid |

**Patient Medical History**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data type (size) | Constraint | Constraint Name |
| Patient\_id | Varchar2(10) | References Patient Demographics | Rkey\_patient\_id |
| Past\_dis | Varchar2(10) |  |  |
| Medication | Varchar2(10) |  |  |
| Doctor1\_id | Varchar2(10) | References Doctor | Rkey\_doctor1 |
| Doctor2\_id | Varchar2(10) | References Doctor,not equal to Doctor1\_id | Rkey\_doctor2,diff\_doc\_ids |
| Date\_diagnosed | Date | Less than or equal to current date | Diagnosis\_valid\_Date |
| Surgeries | Varchar2(40) |  |  |
| Allergies | Varchar2(40) |  |  |

**Appointments**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data type (size) | Constraint | Constraint Name |
| Patient\_id | Varchar2(10) | References Patient Demographics | Rkey\_patient\_id\_apt |
| Doctor\_id | Varchar2(10) | References Doctor | Rkey\_doctor\_apt |
| Date | Date |  |  |
| Time | Varchar2(10) |  |  |
| Payment\_status | Varchar2(10) | C-completed, F-failed | Status\_c\_f |
| Dept\_id | Varchar2(10) | References Dept | Rkey\_dept\_apt |
| Trans\_id | Varchar2(10) | References transaction | Rkey\_trans\_apt |
|  |  |  |  |

**Mobile Numbers**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data type (size) | Constraint | Constraint Name |
| Patient\_id | Varchar2(10) | References Patient Demographics | Rkey\_id\_pcv |
| Mobile Number | Number(10) | Primary key | P\_key\_number |

**5.4 Relational Model Implementation**

CREATE TABLE Patient Demographics (

Patient\_id VARCHAR2(10) CONSTRAINT Pkey\_id PRIMARY KEY,

Name VARCHAR2(40),

Age NUMBER(3) CONSTRAINT Age\_positive CHECK (Age > 0),

Gender VARCHAR2(2) CONSTRAINT Gender\_const CHECK (Gender IN ('M', 'F', 'O')),

Last\_date\_of\_visit DATE CONSTRAINT Visit\_Date\_valid CHECK (Last\_date\_of\_visit <= SYSDATE),

Payment\_details VARCHAR2(2) CONSTRAINT Pay\_det CHECK (Payment\_details IN ('Y', 'N'))

);

CREATE TABLE Dept (

Dept\_id VARCHAR2(10) CONSTRAINT Pkey\_dept\_id PRIMARY KEY CONSTRAINT Dept\_valid CHECK (Dept\_id IN ('OPD', 'Cardiac', 'Opthal')),

Dept\_name VARCHAR2(10)

);

CREATE TABLE Doctor (

Doctor\_id VARCHAR2(10) CONSTRAINT Pkey\_doctor\_id PRIMARY KEY,

Dept\_id VARCHAR2(10) CONSTRAINT Rkey\_dept\_doc REFERENCES Department(Dept\_id),

Name VARCHAR2(40) CONSTRAINT Name\_valid CHECK (Name IS NOT NULL),

Years\_exp NUMBER(2) CONSTRAINT Years\_exp\_positive CHECK (Years\_exp > 0)

);

CREATE TABLE Transaction (

Transaction\_id VARCHAR2(10) CONSTRAINT Pkey\_trans PRIMARY KEY,

Date VARCHAR2(40),

Amt\_paid NUMBER(10) CONSTRAINT Amount\_positive CHECK (Amt\_paid > 0),

Doctor\_id VARCHAR2(10) CONSTRAINT Rkey\_doctor\_trans REFERENCES Doctor(Doctor\_id),

Amt\_insured NUMBER(10) CONSTRAINT Insured\_amt\_positive CHECK (Amt\_insured > 0)

);

CREATE TABLE Patient Medical\_History (

Patient\_id VARCHAR2(10) CONSTRAINT Rkey\_patient\_id REFERENCES Patient(Patient\_id),

Past\_dis VARCHAR2(10),

Medication VARCHAR2(10),

Doctor1\_id VARCHAR2(10) CONSTRAINT Rkey\_doctor1 REFERENCES Doctor(Doctor\_id),

Doctor2\_id VARCHAR2(10) CONSTRAINT Rkey\_doctor2 REFERENCES Doctor(Doctor\_id) CHECK (Doctor2\_id <> Doctor1\_id),

Date\_diagnosed DATE CONSTRAINT Date\_valid CHECK (Date\_diagnosed <= SYSDATE),

Surgeries VARCHAR2(40),

Allergies VARCHAR2(40),

Other\_details VARCHAR2(40)

);

CREATE TABLE Appointments (

Patient\_id VARCHAR2(10) CONSTRAINT Rkey\_patient\_id\_apt REFERENCES Patient(Patient\_id),

Doctor\_id VARCHAR2(10) CONSTRAINT Rkey\_doctor\_apt REFERENCES Doctor(Doctor\_id),

Date VARCHAR2(10) CONSTRAINT Date\_valid\_apt CHECK (Date <= SYSDATE),

Time VARCHAR2(10),

Payment\_status VARCHAR2(2) CONSTRAINT Status\_c\_f CHECK (Payment\_status IN ('C', 'F')),

Dept\_id VARCHAR2(10) CONSTRAINT Rkey\_dept\_apt REFERENCES Department(Dept\_id),

Trans\_id VARCHAR2(10) CONSTRAINT Rkey\_trans\_apt REFERENCES Transactions(Transaction\_id)

);

CREATE TABLE mobile(

Mobile NUMBER(10) CONSTRAINT Pkey\_mob PRIMARY KEY,

Patient\_id VARCHAR2(10) CONSTRAINT Rkey\_patient\_id\_apt REFERENCES Patient(Patient\_id),

);

**5.5 Queries**

SELECT \* FROM Appointment where date=CURDATE();

Select \* from doctors where(Year\_exp>10);

Select pd.patient\_id, pd.mobile\_number from patient \_demographics pd JOIN appointments a ON pd.patient\_id=a.patient\_id WHERE a.payment\_status=’F’;

**6 Functional Requirement Implementation**

New patient Registration

package hms;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import java.util.\*;

public class patient\_registeration extends JFrame implements ActionListener {

private JLabel titleLabel, nameLabel, dobLabel, genderLabel, mobile1Label, mobile2Label;

private JTextField nameField, dobField, mobile1Field, mobile2Field;

private JComboBox genderComboBox;

private JButton registerButton, clearButton;

Connection conn;

PreparedStatement pst;

public patient\_registeration() {

JFrame frame;

frame=new JFrame();

frame.setTitle("New Patient Registration");

frame.setSize(800, 400);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setSize(400, 300);

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.setLocationRelativeTo(null);

titleLabel = new JLabel("New Patient Registration");

titleLabel.setFont(new Font("Tahoma", Font.***BOLD***, 18));

titleLabel.setHorizontalAlignment(JLabel.***CENTER***);

nameLabel = new JLabel("Name:");

dobLabel = new JLabel("Date of Birth:");

genderLabel = new JLabel("Gender:");

mobile1Label = new JLabel("Mobile Number 1:");

mobile2Label = new JLabel("Mobile Number 2:");

nameField = new JTextField();

dobField = new JTextField();

mobile1Field = new JTextField();

mobile2Field = new JTextField();

genderComboBox = new JComboBox(new String[]{"Male", "Female", "Other"});

registerButton = new JButton("Register");

registerButton.addActionListener(this);

clearButton = new JButton("Clear");

clearButton.addActionListener(this);

JPanel inputPanel = new JPanel(new GridLayout(5, 2));

inputPanel.add(nameLabel);

inputPanel.add(nameField);

inputPanel.add(dobLabel);

inputPanel.add(dobField);

inputPanel.add(genderLabel);

inputPanel.add(genderComboBox);

inputPanel.add(mobile1Label);

inputPanel.add(mobile1Field);

inputPanel.add(mobile2Label);

inputPanel.add(mobile2Field);

JPanel buttonPanel = new JPanel(new GridLayout(1, 2));

buttonPanel.add(registerButton);

buttonPanel.add(clearButton);

JPanel mainPanel = new JPanel(new BorderLayout());

mainPanel.add(titleLabel, BorderLayout.***NORTH***);

mainPanel.add(inputPanel, BorderLayout.***CENTER***);

mainPanel.add(buttonPanel, BorderLayout.***SOUTH***);

setLayout(new BorderLayout());

add(mainPanel, BorderLayout.***CENTER***);

setVisible(true);

try {

Class.*forName*("com.mysql.cj.jdbc.Driver");

conn=DriverManager.*getConnection*("jdbc:mysql://localhost:3306/hospital?zeroDateTimeBehavior=CONVERT\_TO\_NULL","root","xamp@root");

} catch (Exception e) {

e.printStackTrace();

}

}

public void actionPerformed(ActionEvent e) {

if (e.getSource() == registerButton) {

registerPatient();

}

}

private void registerPatient() {

String name = nameField.getText();

String dob = dobField.getText();

String gender = (String) genderComboBox.getSelectedItem();

String mobile1 = mobile1Field.getText();

String mobile2 = mobile2Field.getText();

int mob1=Integer.*parseInt*(mobile1);

int mob2=Integer.*parseInt*(mobile2);

try {

pst=conn.prepareStatement("INSERT INTO patient\_demo (name, Date\_Of\_Birth, gender)values(?,?,?)");

pst.setString(1, name);

pst.setString(2, dob);

pst.setString(3, gender);

} catch (SQLException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

new patient\_registeration();

}

}

Patient Medical History Update

package hms;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.sql.\*;

public class pmh\_update extends JFrame implements ActionListener {

private JTextField patientIDTextField, pastDiseasesTextField, medicationTextField, doctor1TextField, doctor2TextField, dateDiagnosedTextField, surgeriesTextField, allergiesTextField;

private JButton submitButton;

private final String DB\_URL = "jdbc:mysql://localhost:3306/your\_database\_name";

private final String USERNAME = "your\_username";

private final String PASSWORD = "your\_password";

public pmh\_update() {

setTitle("Enter Patient Medical History");

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setSize(400, 300);

setLocationRelativeTo(null);

JPanel panel = new JPanel(new GridLayout(9, 2, 10, 10));

panel.setBorder(BorderFactory.*createEmptyBorder*(10, 10, 10, 10));

panel.add(new JLabel("Patient ID:"));

patientIDTextField = new JTextField();

panel.add(patientIDTextField);

panel.add(new JLabel("Past Diseases:"));

pastDiseasesTextField = new JTextField();

panel.add(pastDiseasesTextField);

panel.add(new JLabel("Medication:"));

medicationTextField = new JTextField();

panel.add(medicationTextField);

panel.add(new JLabel("Doctor 1:"));

doctor1TextField = new JTextField();

panel.add(doctor1TextField);

panel.add(new JLabel("Doctor 2:"));

doctor2TextField = new JTextField();

panel.add(doctor2TextField);

panel.add(new JLabel("Date Diagnosed (yyyy-mm-dd):"));

dateDiagnosedTextField = new JTextField();

panel.add(dateDiagnosedTextField);

panel.add(new JLabel("Surgeries:"));

surgeriesTextField = new JTextField();

panel.add(surgeriesTextField);

panel.add(new JLabel("Allergies:"));

allergiesTextField = new JTextField();

panel.add(allergiesTextField);

submitButton = new JButton("Submit");

submitButton.addActionListener(this);

panel.add(submitButton);

add(panel);

setVisible(true);

}

public void actionPerformed(ActionEvent e) {

if (e.getSource() == submitButton) {

String patientID = patientIDTextField.getText();

String pastDiseases = pastDiseasesTextField.getText();

String medication = medicationTextField.getText();

String doctor1 = doctor1TextField.getText();

String doctor2 = doctor2TextField.getText();

String dateDiagnosed = dateDiagnosedTextField.getText();

String surgeries = surgeriesTextField.getText();

String allergies = allergiesTextField.getText();

try {

Connection conn = DriverManager.*getConnection*(DB\_URL, USERNAME, PASSWORD);

String query = "INSERT INTO patient\_medical\_history (patient\_id, past\_diseases, medication, doctor1, doctor2, date\_diagnosed, surgeries, allergies) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, patientID);

pstmt.setString(2, pastDiseases);

pstmt.setString(3, medication);

pstmt.setString(4, doctor1);

pstmt.setString(5, doctor2);

pstmt.setString(6, dateDiagnosed);

pstmt.setString(7, surgeries);

pstmt.setString(8, allergies);

pstmt.executeUpdate();

pstmt.close();

conn.close();

JOptionPane.*showMessageDialog*(this, "Patient medical history has been successfully saved to the database.");

} catch (SQLException ex) {

JOptionPane.*showMessageDialog*(this, "Error: " + ex.getMessage(), "Error", JOptionPane.***ERROR\_MESSAGE***);

}

}

}

public static void main(String[] args) {

new pmh\_update();

}

}

Patient Medical History view

package hms;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

public class display\_pmh {

JFrame frame;

JTable table;

private JLabel lblNewLabel;

private JButton btnNewButton;

private JTextField id;

public display\_pmh() {

frame = new JFrame("Patient Medical History");

frame.setSize(800, 400);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

String[] columnHeaders = {"Patient ID", "Past Disease", "Medication", "Doctor 1 ID", "Doctor 2 ID", "Date Diagnosed", "Surgeries", "Allergies"};

DefaultTableModel tableModel = new DefaultTableModel(columnHeaders, 0);

frame.getContentPane().setLayout(null);

table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

scrollPane.setBounds(0, 50, 784, 315);

frame.getContentPane().add(scrollPane);

lblNewLabel = new JLabel("Patient Id");

lblNewLabel.setBounds(134, 25, 59, 14);

frame.getContentPane().add(lblNewLabel);

btnNewButton = new JButton("Submit");

btnNewButton.setBounds(403, 21, 89, 23);

frame.getContentPane().add(btnNewButton);

id = new JTextField();

id.setBounds(264, 22, 86, 20);

frame.getContentPane().add(id);

id.setColumns(10);

Connection conn;

try {

Class.forName("com.mysql.cj.jdbc.Driver");

conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/hospital?zeroDateTimeBehavior=CONVERT\_TO\_NULL","root","xamp@root");

String query = "SELECT \* FROM patient\_medical\_history";

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query);

while (rs.next()) {

int patientID = rs.getInt("patient\_id");

String pastDisease = rs.getString("past\_dis");

String medication = rs.getString("medication");

String doctor1ID = rs.getString("doctor1\_id");

String doctor2ID = rs.getString("doctor2\_id");

String dateDiagnosed = rs.getString("date\_diagnosed");

String surgeries = rs.getString("surgeries");

String allergies = rs.getString("allergies");

Object[] row = {patientID, pastDisease, medication, doctor1ID, doctor2ID, dateDiagnosed, surgeries, allergies};

tableModel.addRow(row);

}

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String patient\_id=id.getText();

int pid=Integer.parseInt(patient\_id);

try {

String query = "SELECT \* FROM patient\_medical\_history";

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query);

int flag=0;

while(rs.next())

{

if(rs.getInt("patient\_id")==pid)

{

tableModel.setRowCount(0);

int patientID = rs.getInt("patient\_id");

String pastDisease = rs.getString("past\_dis");

String medication = rs.getString("medication");

String doctor1ID = rs.getString("doctor1\_id");

String doctor2ID = rs.getString("doctor2\_id");

String dateDiagnosed = rs.getString("date\_diagnosed");

String surgeries = rs.getString("surgeries");

String allergies = rs.getString("allergies");

Object[] row = {patientID, pastDisease, medication, doctor1ID, doctor2ID, dateDiagnosed, surgeries, allergies};

tableModel.addRow(row);

flag=1;

break;

}

}

if(flag==0)

{

JOptionPane.showMessageDialog(null,"Patient history not found");

}

} catch (SQLException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

}

});

} catch (Exception e) {

e.printStackTrace();

}

frame.setVisible(true);

}

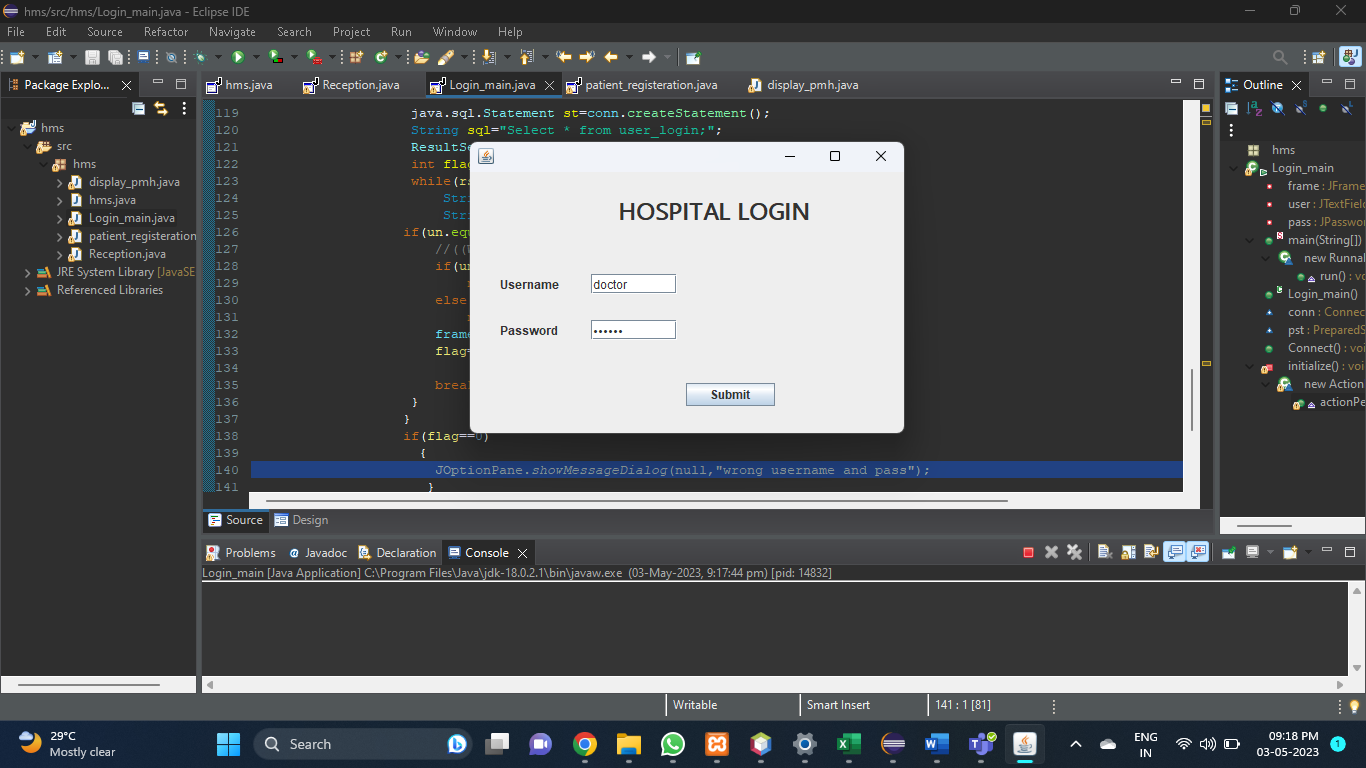
public static void main(String[] args) {

new display\_pmh();

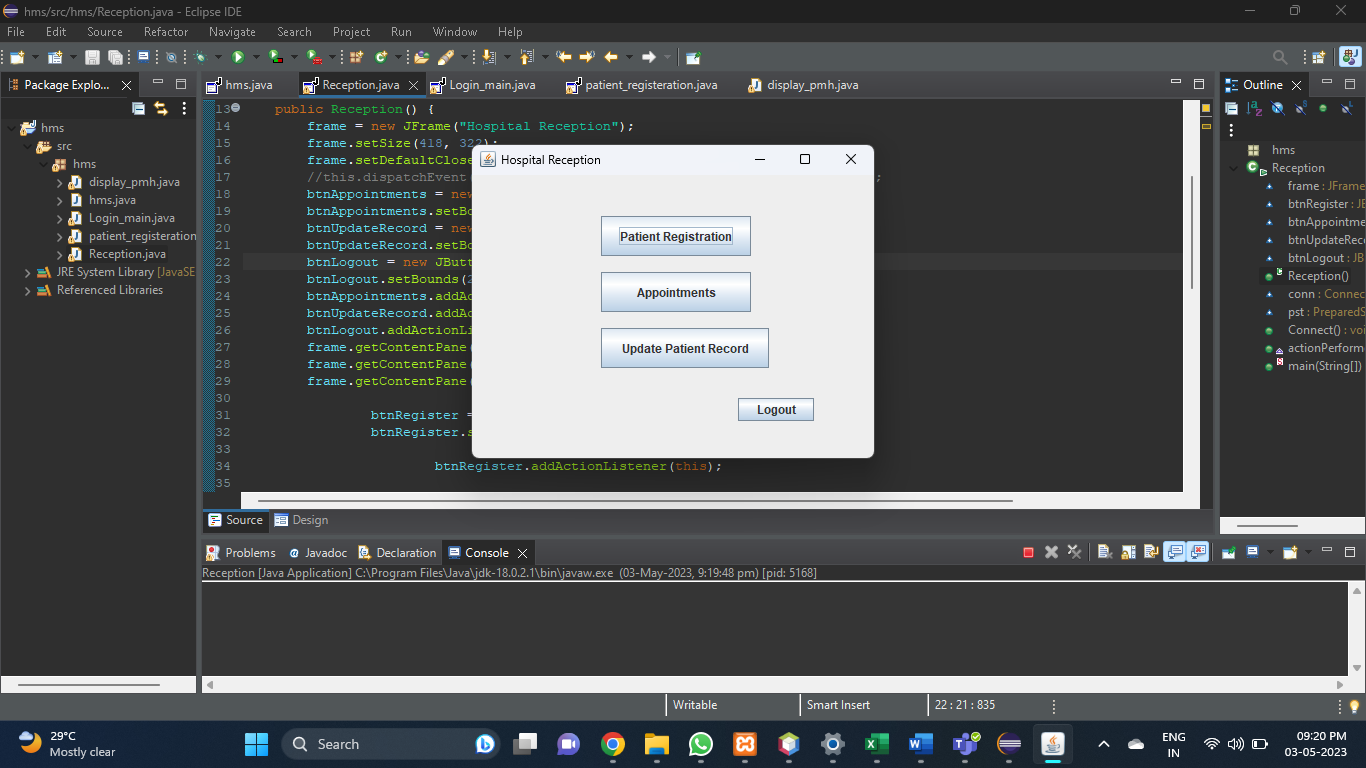
}

}

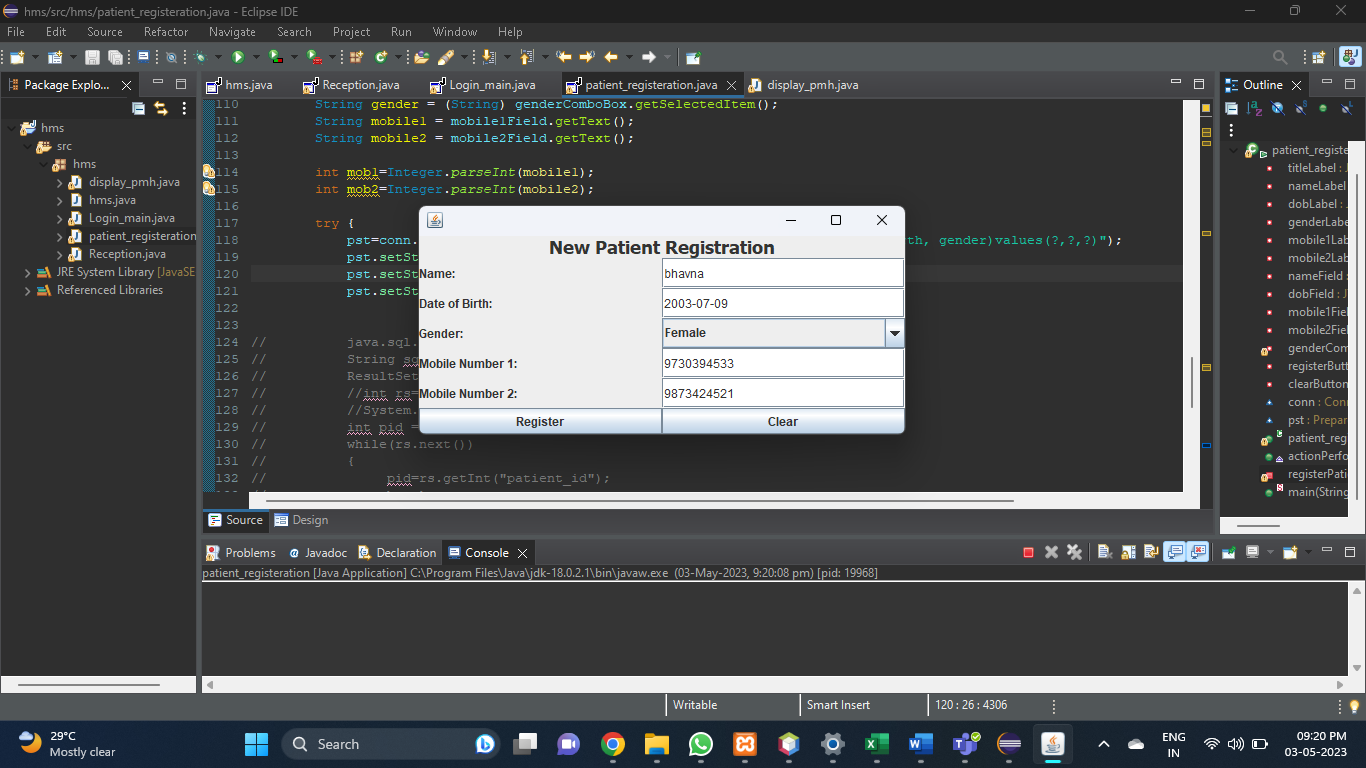
**7. Result screenshot**



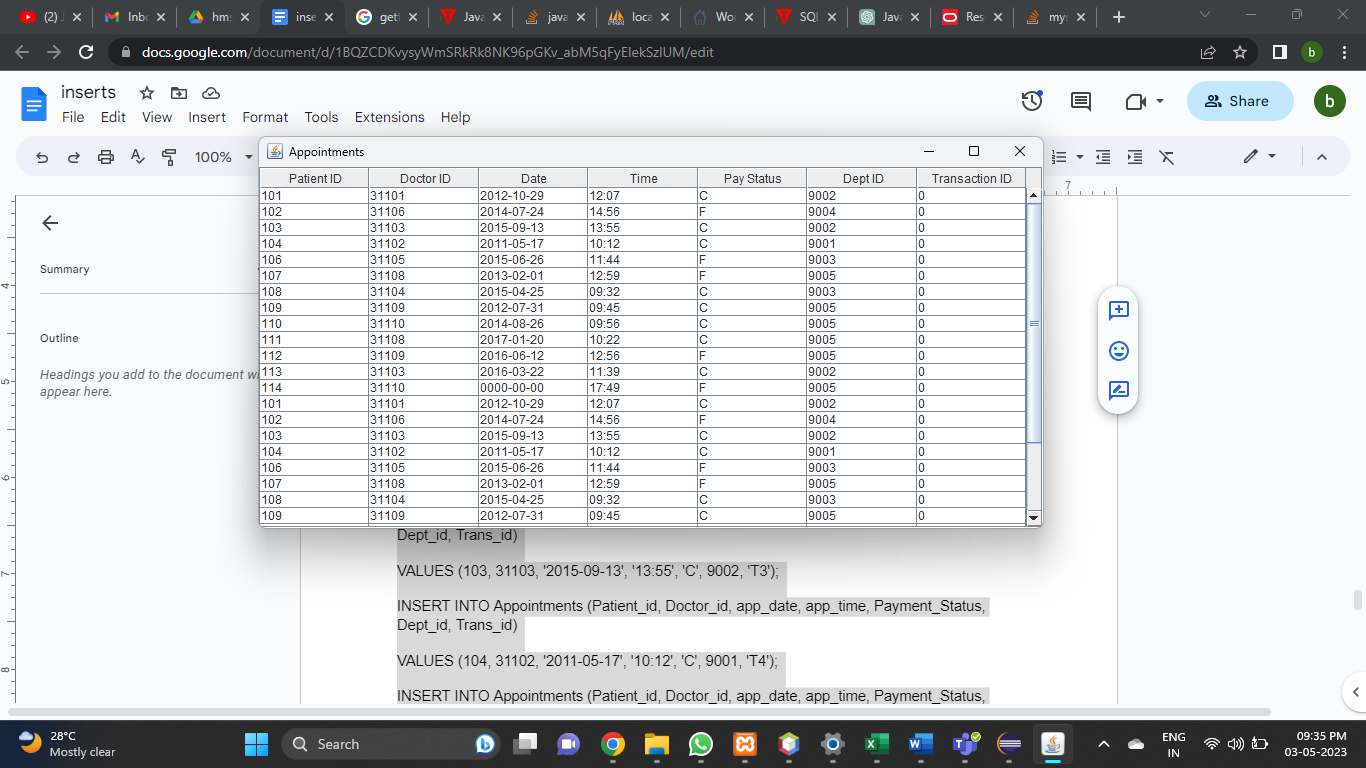
Login for Reception Or doctor module, Reception Module goes to reception, Doctor module open Patient\_medical\_history.



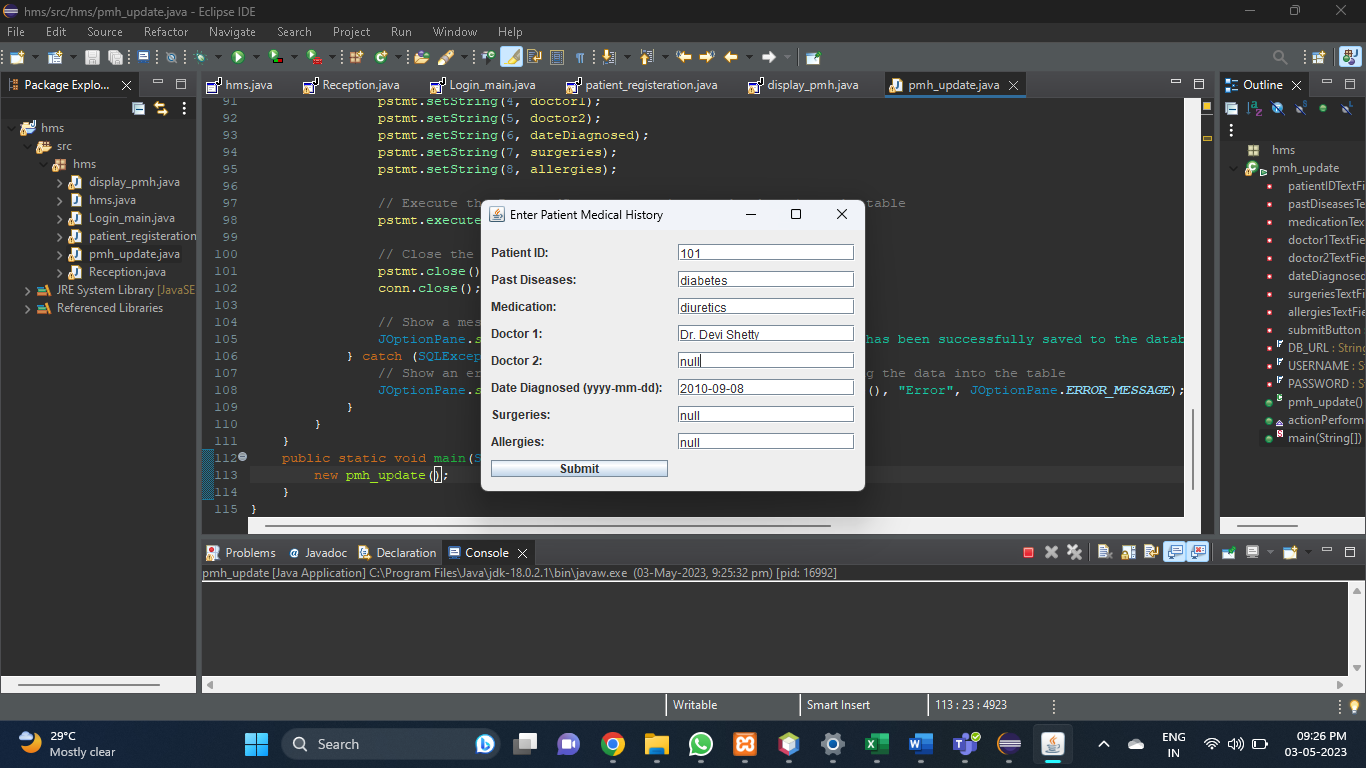
Patient Registration, opens new patient\_registration, appointments gives list of scheduled appointments, update patient record allows updation of patient medical history

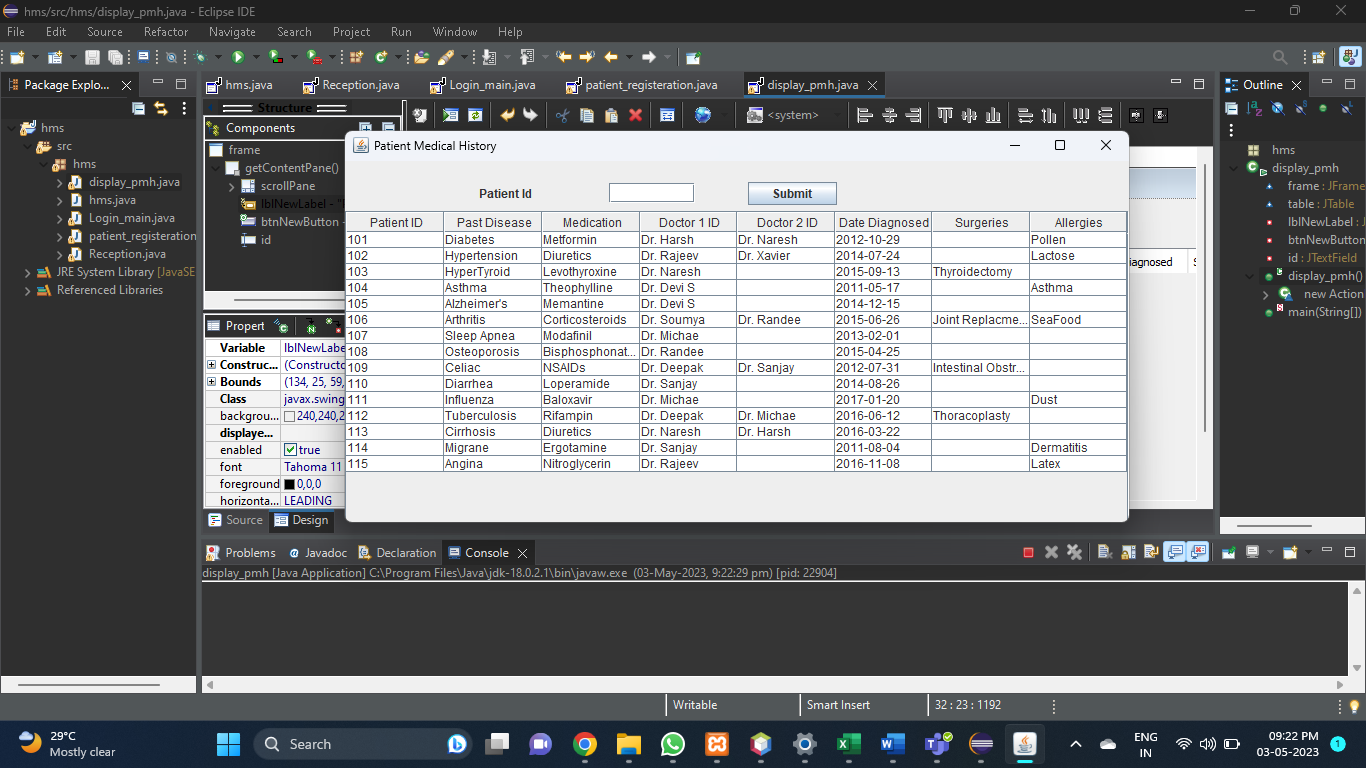


New patient Registration

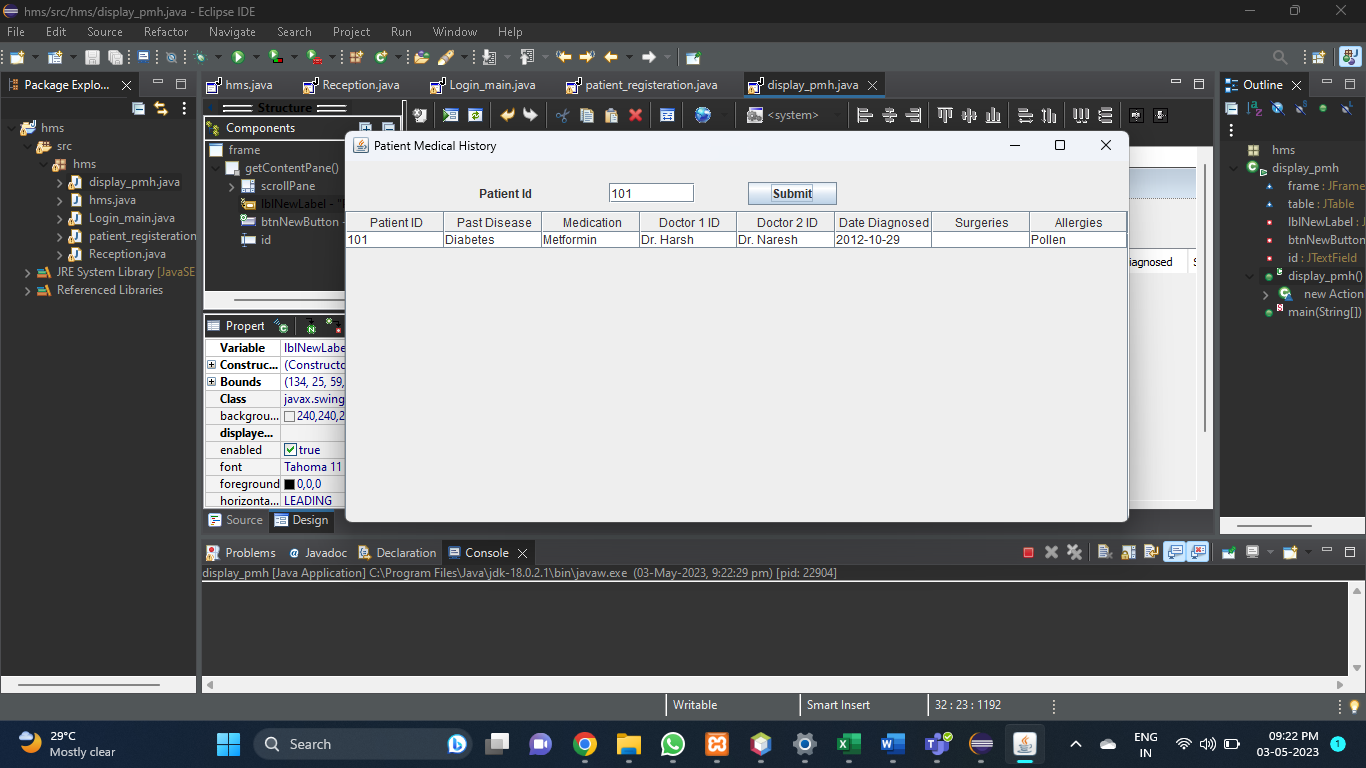


List of scheduled appointments

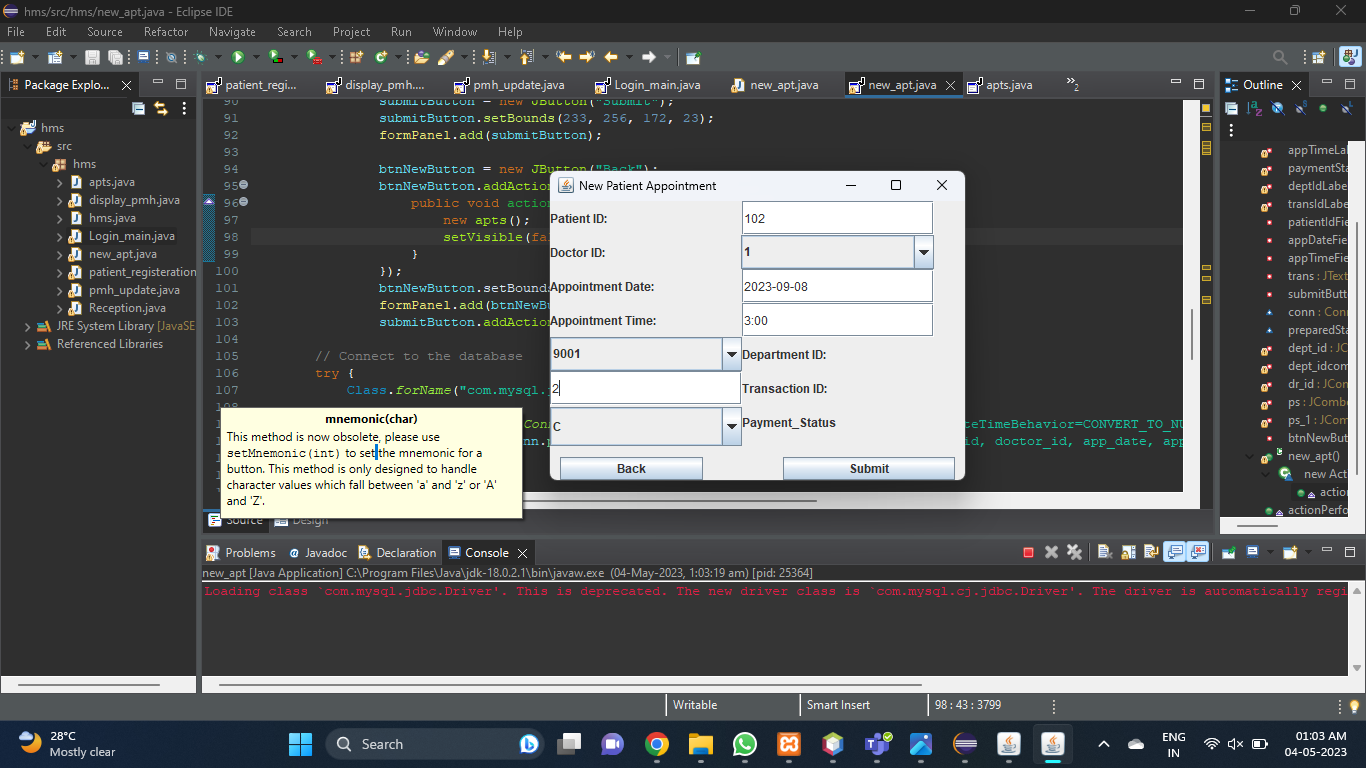
Updating Patient Medical History



Patients Medical History view



Selecting medical history of particular patient

 New Appointment schedule

**8. Conclusion and Future Work**

**8.1 Conclusion**

Hospital Database Management, digitalises all records in hospitals for its day to day functionality. It eases the process of appointments and ensure a smooth working hospital. Helps in maintaining records and ease diagnosis and medications for future complications. The current system was thoroughly tested with dummy data.

**8.2 Scope for future work**

More details about the patients and doctors maybe stored in the database. Predefined tables for various types of surgeries offered in the medical stream maybe stored, to allow checking of all patients undergoing a particular surgery. Cost per medical practice can be stored in another table to ease calculation of bills. Bill calculation function could be created. In short, there is a lot of scope and use of the current hospital database.