**JAVA SWING BASED –VACCINATION MANAGEMENT SYSTEM– SQL CONNECTIVITY USING JDBC**

*A Report*

*Submitted in partial fulfillment of the Requirements*

*for the COURSE*

**DATABASE MANAGEMENT SYSTEMS**

**By**

### SADHU ASHISH <1602-21-737-011>

**Under the guidance of Ms B. Leelavathy**



### Department of Information Technology Vasavi College of Engineering (Autonomous) (Affiliated to Osmania University) Ibrahimbagh, Hyderabad-31

### 2022-2023

BONAFIDE CERTIFICATE

This is to certify that this project report titled

***‘VACCINATION MANAGEMENT SYSTEM’***

is a project work of **SADHU ASHISH** bearing roll no. 1602-21-737-011 who carried out this project under my supervision in the IV semester for the academic year 2022- 2023

Signature Signature

External Examiner Internal Examiner

# ABSTRACT

The Vaccination Management System is an automated platform designed to streamline and optimize the process of administering and tracking vaccinations. This system aims to enhance the efficiency, accuracy, and accessibility of vaccination services by utilizing technology to manage patient records, vaccine inventory, appointment scheduling, and data analysis. The abstract highlights the system's ability to centralize and digitize vaccination-related information, enabling healthcare providers to effectively manage vaccine distribution, monitor immunization coverage, and generate real-time reports. By automating various administrative tasks, reducing manual errors, and improving data visibility, the Vaccination Management System contributes to the overall improvement of public health initiatives, ensuring timely and targeted vaccination campaigns for individuals and communities.

# Requirement Analysis

## List of Tables:

* users
* vaccinationcenter
* vaccinationrecord
* vaccine

## List of Attributes with their Domain Types:

**USERS**

ID NOT NULL NUMBER

NAME VARCHAR2(20)

EMAIL VARCHAR2(30)

PASSWORD VARCHAR2(20)

**vaccinationcenter**

ID NOT NULL NUMBER

NAME NOT NULL VARCHAR2(20)

ADDRESS NOT NULL VARCHAR2(20)

CONTACT\_INFO VARCHAR2(20)

OPERATING\_HOURS VARCHAR2(20)

CAPACITY NUMBER

**vaccinationrecord**

ID NOT NULL NUMBER

USER\_ID NOT NULL NUMBER

VACCINE\_ID NOT NULL NUMBER

VACCINATION\_DATE NOT NULL DATE

HEALTHCARE\_PROVIDER\_ID NUMBER

**vaccine**

ID NOT NULL NUMBER

NAME NOT NULL VARCHAR2(255)

MANUFACTURER VARCHAR2(255)

LOT\_NUMBER VARCHAR2(255)

EXPIRATION\_DATE DATE

DOSAGE VARCHAR2(255)

# AIM AND PRIORITY OF THE PROJECT

The aim of the Vaccination Management System project is to provide an efficient and organized system for managing the administration, tracking, and monitoring of vaccinations. The priority of the project is to streamline the vaccination process, ensuring accurate record-keeping, efficient vaccine distribution, and timely reminders for follow-up doses. The system aims to facilitate effective communication between healthcare providers, patients, and vaccination centers, enabling better coordination and ensuring that individuals receive the appropriate vaccines at the right time. By automating various tasks and centralizing data management, the project aims to enhance public health efforts by ensuring that vaccinations are administered effectively and efficiently, ultimately contributing to the prevention and control of infectious diseases.

# ARCHITECTURE AND TECHNOLOGY

### Software used:

Java, Oracle 11g Database, Java SE version 14, Run SQL.

### Java SWING:

**Java SWING** is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs.

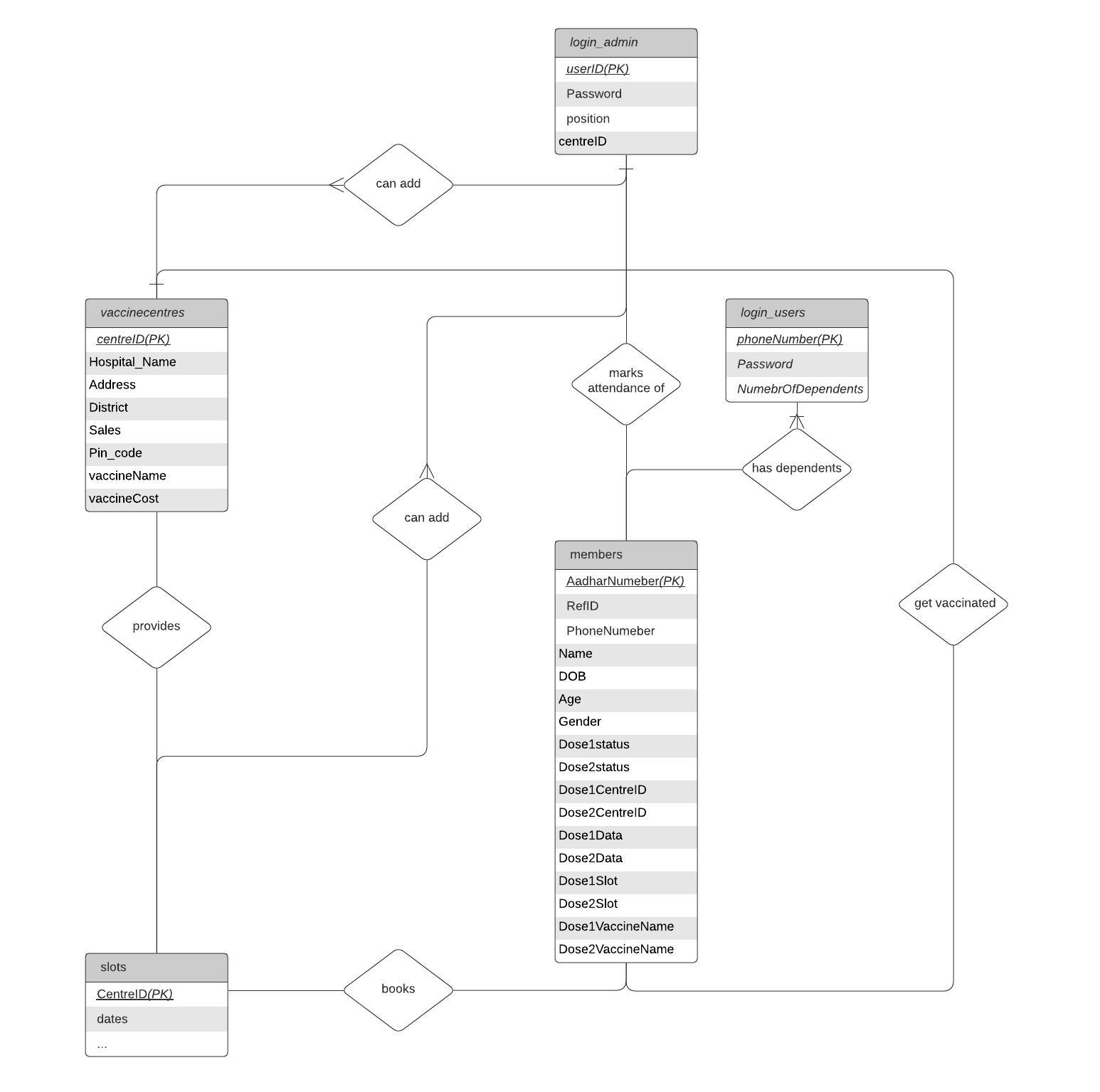
Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT. Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

### SQL:

Structure Query Language(SQL) is a database query language used for storing and managing data in **Relational** DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

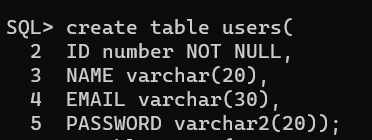
**DESIGN**

### Entity Relationship Diagram

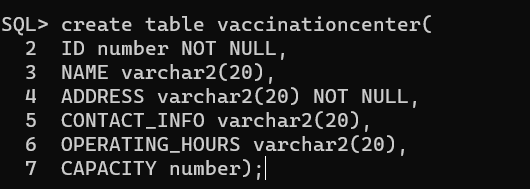


**TABLE CREATED IN SQL:**

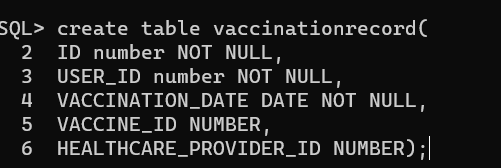
1. Users Table



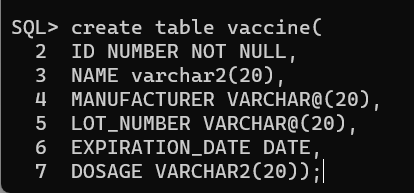
1. Vaccinationcenter



1. Vaccinationrecord



1. Vaccine



### DATABASE DESIGN:

SQL> desc users;

Name Null? Type

----------------------------------------- -------- ----------------------------

ID NOT NULL NUMBER

NAME VARCHAR2(20)

EMAIL VARCHAR2(30)

SQL> desc vaccinationcenter;

Name Null? Type

----------------------------------------- -------- ----------------------------

ID NOT NULL NUMBER

NAME NOT NULL VARCHAR2(20)

ADDRESS NOT NULL VARCHAR2(20)

CONTACT\_INFO VARCHAR2(20)

OPERATING\_HOURS VARCHAR2(20)

CAPACITY NUMBER

SQL> desc vaccinationrecord

Name Null? Type

----------------------------------------- -------- ---------------------------

ID NOT NULL NUMBER

USER\_ID NOT NULL NUMBER

VACCINE\_ID NOT NULL NUMBER

VACCINATION\_DATE NOT NULL DATE

HEALTHCARE\_PROVIDER\_ID NUMBER

SQL> desc vaccine

Name Null? Type

----------------------------------------- -------- ----------------------------

ID NOT NULL NUMBER

NAME NOT NULL VARCHAR2(255)

MANUFACTURER VARCHAR2(255)

LOT\_NUMBER VARCHAR2(255)

EXPIRATION\_DATE DATE

DOSAGE VARCHAR2(255)

# IMPLEMENTATION

### JAVA-SQL Connectivity using JDBC:

**Java Database Connectivity (JDBC)** is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

The connection to the database can be performed using Java programming (JDBC API) as:

**import** javax.swing.\*;

**import** javax.swing.event.ListSelectionEvent;

**import** javax.swing.event.ListSelectionListener;

**import** javax.swing.table.DefaultTableModel;

**import** java.awt.\*;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.sql.\*;

**import** java.util.Vector;

**public** **class** VaccinationSystemGUI **extends** JFrame {

**private** JTable usersTable;

**private** JTable vaccinationCenterTable;

**private** JTable vaccinationRecordTable;

**private** JTable vaccineTable;

**private** JTable vaccineInventoryTable;

**private** Connection connection;

**private** Statement statement;

// Fields for adding new user data

**private** JTextField userIdField;

**private** JTextField userNameField;

**private** JTextField userEmailField;

**private** JPasswordField userPasswordField;

// Fields for adding new vaccination center data

**private** JTextField centerIdField;

**private** JTextField centerNameField;

**private** JTextField centerAddressField;

**private** JTextField centerContactInfoField;

**private** JTextField centerOperatingHoursField;

**private** JTextField centerCapacityField;

**private** JTextField recordIdField;

**private** JTextField vaccineIdField;

**private** JTextField dateField;

**private** JTextField providerIdField;

**public** VaccinationSystemGUI() {

initializeGUI();

establishConnection();

populateTables();

usersTable.getSelectionModel().addListSelectionListener(**new** ListSelectionListener() {

@Override

**public** **void** valueChanged(ListSelectionEvent e) {

// Retrieve the selected row

**int** selectedRow = usersTable.getSelectedRow();

// Check if a row is selected

**if** (selectedRow != -1) {

// Retrieve the data from the selected row

String id = usersTable.getValueAt(selectedRow, 0).toString();

String name = usersTable.getValueAt(selectedRow, 1).toString();

String email = usersTable.getValueAt(selectedRow, 2).toString();

String password = usersTable.getValueAt(selectedRow, 3).toString();

// Update the fields in the users panel

userIdField.setText(id);

userNameField.setText(name);

userEmailField.setText(email);

userPasswordField.setText(password);

}

}

});

vaccinationCenterTable.getSelectionModel().addListSelectionListener(**new** ListSelectionListener() {

@Override

**public** **void** valueChanged(ListSelectionEvent e) {

// Retrieve the selected row

**int** selectedRow = vaccinationCenterTable.getSelectedRow();

// Check if a row is selected

**if** (selectedRow != -1) {

// Retrieve the data from the selected row

String id = vaccinationCenterTable.getValueAt(selectedRow, 0).toString();

String name = vaccinationCenterTable.getValueAt(selectedRow, 1).toString();

String address = vaccinationCenterTable.getValueAt(selectedRow, 2).toString();

String contactInfo = vaccinationCenterTable.getValueAt(selectedRow, 3).toString();

String operatingHours = vaccinationCenterTable.getValueAt(selectedRow, 4).toString();

String capacity = vaccinationCenterTable.getValueAt(selectedRow, 5).toString();

// Update the fields in the vaccination center panel

centerIdField.setText(id);

centerNameField.setText(name);

centerAddressField.setText(address);

centerContactInfoField.setText(contactInfo);

centerOperatingHoursField.setText(operatingHours);

centerCapacityField.setText(capacity);

}

}

});

vaccinationRecordTable.getSelectionModel().addListSelectionListener(**new** ListSelectionListener() {

@Override

**public** **void** valueChanged(ListSelectionEvent e) {

// Retrieve the selected row

**int** selectedRow = vaccinationRecordTable.getSelectedRow();

// Check if a row is selected

**if** (selectedRow != -1) {

// Retrieve the data from the selected row

String id = vaccinationRecordTable.getValueAt(selectedRow, 0).toString();

String userId = vaccinationRecordTable.getValueAt(selectedRow, 1).toString();

String vaccineId = vaccinationRecordTable.getValueAt(selectedRow, 2).toString();

String vaccinationDate = vaccinationRecordTable.getValueAt(selectedRow, 3).toString();

String healthcareProviderId = vaccinationRecordTable.getValueAt(selectedRow, 4).toString();

// Update the fields in the vaccination record panel

JTextField recordIdField = **new** JTextField();

// Declare and define the vaccineIdField, vaccinationDateField, and healthcareProviderIdField variables

JTextField vaccineIdField = **new** JTextField();

JTextField vaccinationDateField = **new** JTextField();

JTextField healthcareProviderIdField = **new** JTextField();

recordIdField.setText(id);

userIdField.setText(userId);

vaccineIdField.setText(vaccineId);

vaccinationDateField.setText(vaccinationDate);

healthcareProviderIdField.setText(healthcareProviderId);

}

}

});

}

**private** **void** initializeGUI() {

setTitle("Vaccination System");

setSize(800, 600);

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

setLocationRelativeTo(**null**);

// Create a tabbed pane to hold multiple tables

JTabbedPane tabbedPane = **new** JTabbedPane();

// Create panels for each table11111

usersTable = **new** JTable();

JPanel usersPanel = createTablePanel("Users", usersTable);

vaccinationCenterTable = **new** JTable();

JPanel vaccinationCenterPanel = createTablePanel("Vaccination Centers", vaccinationCenterTable);

// Add the panels to the tabbed pane

tabbedPane.addTab("Users", usersPanel);

tabbedPane.addTab("Vaccination Centers", vaccinationCenterPanel);

// Add the tabbed pane to the main frame

getContentPane().add(tabbedPane, BorderLayout.***CENTER***);

// Create a panel for adding new data

JPanel addDataPanel = createAddDataPanel();

getContentPane().add(addDataPanel, BorderLayout.***SOUTH***);

vaccinationRecordTable = **new** JTable();

JPanel vaccinationRecordPanel = createTablePanel("Vaccination Records", vaccinationRecordTable);

vaccineTable = **new** JTable();

JPanel vaccinePanel = createTablePanel("Vaccine", vaccineTable);

// Add the panels to the tabbed pane

tabbedPane.addTab("Users", usersPanel);

tabbedPane.addTab("Vaccination Centers", vaccinationCenterPanel);

tabbedPane.addTab("Vaccination Records", vaccinationRecordPanel);

tabbedPane.addTab("Vaccine", vaccinePanel);

}

**private** JPanel createTablePanel(String title, JTable table) {

JPanel panel = **new** JPanel(**new** BorderLayout());

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

JLabel titleLabel = **new** JLabel(title);

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

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

JScrollPane scrollPane = **new** JScrollPane(table);

panel.add(scrollPane, BorderLayout.***CENTER***);

**return** panel;

}

**private** JPanel createAddDataPanel() {

JPanel panel = **new** JPanel();

panel.setLayout(**new** BorderLayout());

JTabbedPane tabbedPane = **new** JTabbedPane();

JPanel usersPanel = createUserAddDataPanel();

JPanel centersPanel = createVaccinationCenterAddDataPanel();

JPanel recordPanel = createVaccinationRecordAddDataPanel();

tabbedPane.addTab("Users", usersPanel);

tabbedPane.addTab("Vaccination Centers", centersPanel);

tabbedPane.addTab("Vaccination Records", recordPanel);

panel.add(tabbedPane, BorderLayout.***CENTER***);

**return** panel;

}

**private** JPanel createUserAddDataPanel() {

JPanel panel = **new** JPanel();

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

panel.setLayout(**new** GridLayout(5, 2, 10, 5));

JLabel idLabel = **new** JLabel("ID:");

userIdField = **new** JTextField();

JLabel nameLabel = **new** JLabel("Name:");

userNameField = **new** JTextField();

JLabel emailLabel = **new** JLabel("Email:");

userEmailField = **new** JTextField();

JLabel passwordLabel = **new** JLabel("Password:");

userPasswordField = **new** JPasswordField();

JButton addButton = **new** JButton("Add");

addButton.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

addUser();

}

});

JButton modifyButton = **new** JButton("Modify");

modifyButton.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

modifyUser();

}

});

JButton deleteButton = **new** JButton("Delete");

deleteButton.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

deleteUser();

}

});

panel.add(idLabel);

panel.add(userIdField);

panel.add(nameLabel);

panel.add(userNameField);

panel.add(emailLabel);

panel.add(userEmailField);

panel.add(passwordLabel);

panel.add(userPasswordField);

panel.add(addButton);

panel.add(modifyButton);

panel.add(deleteButton);

**return** panel;

}

**private** JPanel createVaccinationRecordAddDataPanel() {

recordIdField = **new** JTextField();

vaccineIdField = **new** JTextField();

dateField = **new** JTextField();

providerIdField = **new** JTextField();

JPanel panel = **new** JPanel();

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

panel.setLayout(**new** GridLayout(5, 2, 10, 5));

JLabel idLabel = **new** JLabel("ID:");

JTextField recordIdField = **new** JTextField();

JLabel userIdLabel = **new** JLabel("User ID:");

JTextField userIdField = **new** JTextField();

JLabel vaccineIdLabel = **new** JLabel("Vaccine ID:");

JTextField vaccineIdField = **new** JTextField();

JLabel dateLabel = **new** JLabel("Vaccination Date:");

JTextField dateField = **new** JTextField();

JLabel providerIdLabel = **new** JLabel("Healthcare Provider ID:");

JTextField providerIdField = **new** JTextField();

JButton addButton = **new** JButton("Add");

addButton.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

addVaccinationRecord();

}

});

JButton modifyButton = **new** JButton("Modify");

modifyButton.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

//modifyVaccinationRecord();

}

});

JButton deleteButton = **new** JButton("Delete");

deleteButton.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

//deleteVaccinationRecord();

}

});

panel.add(idLabel);

panel.add(recordIdField);

panel.add(userIdLabel);

panel.add(userIdField);

panel.add(vaccineIdLabel);

panel.add(vaccineIdField);

panel.add(dateLabel);

panel.add(dateField);

panel.add(providerIdLabel);

panel.add(providerIdField);

panel.add(addButton);

panel.add(modifyButton);

panel.add(deleteButton);

**return** panel;

}

**private** JPanel createVaccinationCenterAddDataPanel() {

JPanel panel = **new** JPanel();

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

panel.setLayout(**new** GridLayout(7, 2, 10, 5));

JLabel idLabel = **new** JLabel("ID:");

centerIdField = **new** JTextField();

JLabel nameLabel = **new** JLabel("Name:");

centerNameField = **new** JTextField();

JLabel addressLabel = **new** JLabel("Address:");

centerAddressField = **new** JTextField();

JLabel contactInfoLabel = **new** JLabel("Contact Info:");

centerContactInfoField = **new** JTextField();

JLabel operatingHoursLabel = **new** JLabel("Operating Hours:");

centerOperatingHoursField = **new** JTextField();

JLabel capacityLabel = **new** JLabel("Capacity:");

centerCapacityField = **new** JTextField();

JButton addButton = **new** JButton("Add");

addButton.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

addVaccinationCenter();

}

});

JButton modifyButton = **new** JButton("Modify");

modifyButton.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

modifyVaccinationCenter();

}

});

JButton deleteButton = **new** JButton("Delete");

deleteButton.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

deleteVaccinationCenter();

}

});

panel.add(idLabel);

panel.add(centerIdField);

panel.add(nameLabel);

panel.add(centerNameField);

panel.add(addressLabel);

panel.add(centerAddressField);

panel.add(contactInfoLabel);

panel.add(centerContactInfoField);

panel.add(operatingHoursLabel);

panel.add(centerOperatingHoursField);

panel.add(capacityLabel);

panel.add(centerCapacityField);

panel.add(addButton);

panel.add(modifyButton);

panel.add(deleteButton);

**return** panel;

}

**private** **void** populateTables() {

// Example data for demonstration

String[] usersColumns = {"ID", "Name", "Email","Password"};

DefaultTableModel usersModel = **new** DefaultTableModel(usersColumns, 0);

**try** {

ResultSet resultSet = statement.executeQuery("SELECT \* FROM users");

**while** (resultSet.next()) {

String id = resultSet.getString("ID");

String name = resultSet.getString("Name");

String email = resultSet.getString("Email");

String password=resultSet.getString("Password");

Object[] rowData = {id, name, email,password};

usersModel.addRow(rowData);

}

} **catch** (SQLException e) {

e.printStackTrace();

}

usersTable.setModel(usersModel);

// Retrieve data from the database for Vaccination Centers

String[] centerColumns = {"ID", "Name", "Address", "Contact Info", "Operating Hours", "Capacity"};

DefaultTableModel centerModel = **new** DefaultTableModel(centerColumns, 0);

**try** {

ResultSet resultSet = statement.executeQuery("SELECT \* FROM vaccinationcenter");

**while** (resultSet.next()) {

String id = resultSet.getString("ID");

String name = resultSet.getString("Name");

String address = resultSet.getString("Address");

String contactInfo = resultSet.getString("Contact\_Info");

String operatingHours = resultSet.getString("Operating\_Hours");

String capacity = resultSet.getString("Capacity");

Object[] rowData = {id, name, address, contactInfo, operatingHours, capacity};

centerModel.addRow(rowData);

}

} **catch** (SQLException e) {

e.printStackTrace();

}

vaccinationCenterTable.setModel(centerModel);

// Set preferred column widths

usersTable.getColumnModel().getColumn(0).setPreferredWidth(50);

vaccinationCenterTable.getColumnModel().getColumn(0).setPreferredWidth(50);

// Refresh the table models

usersModel.fireTableDataChanged();

centerModel.fireTableDataChanged();

// Retrieve data from the database for Vaccination Records

String[] recordColumns = {"ID", "User ID", "Vaccine ID", "Vaccination Date", "Healthcare Provider ID"};

DefaultTableModel recordModel = **new** DefaultTableModel(recordColumns, 0);

**try** {

ResultSet resultSet = statement.executeQuery("SELECT \* FROM vaccinationrecord");

**while** (resultSet.next()) {

String id = resultSet.getString("ID");

String userID = resultSet.getString("USER\_ID");

String vaccineID = resultSet.getString("VACCINE\_ID");

String vaccinationDate = resultSet.getString("VACCINATION\_DATE");

String healthcareProviderID = resultSet.getString("HEALTHCARE\_PROVIDER\_ID");

Object[] rowData = {id, userID, vaccineID, vaccinationDate, healthcareProviderID};

recordModel.addRow(rowData);

}

} **catch** (SQLException e) {

e.printStackTrace();

}

vaccinationRecordTable.setModel(recordModel);

// Set preferred column widths

vaccinationRecordTable.getColumnModel().getColumn(0).setPreferredWidth(50);

// Refresh the table model

String[] vaccineColumns = {"ID", "NAME", "MANUFACTURER", "LOT\_NUMBER", "EXPIRATION\_DATE", "DOSAGE"};

DefaultTableModel vaccineModel = **new** DefaultTableModel(vaccineColumns, 0);

**try** {

ResultSet resultSet = statement.executeQuery("SELECT \* FROM vaccine");

**while** (resultSet.next()) {

String id = resultSet.getString("ID");

String name = resultSet.getString("NAME");

String manufacturer = resultSet.getString("MANUFACTURER");

String lotNumber = resultSet.getString("LOT\_NUMBER");

String expirationDate = resultSet.getString("EXPIRATION\_DATE");

String dosage = resultSet.getString("DOSAGE");

Object[] rowData = {id, name, manufacturer, lotNumber, expirationDate, dosage};

vaccineModel.addRow(rowData);

}

} **catch** (SQLException e) {

e.printStackTrace();

}

vaccineTable.setModel(vaccineModel);

// Set preferred column widths

usersTable.getColumnModel().getColumn(0).setPreferredWidth(50);

vaccinationCenterTable.getColumnModel().getColumn(0).setPreferredWidth(50);

vaccineTable.getColumnModel().getColumn(0).setPreferredWidth(50);

// Refresh the table models

usersModel.fireTableDataChanged();

centerModel.fireTableDataChanged();

vaccineModel.fireTableDataChanged();

}

**private** **void** establishConnection() {

**try** {

// Update the connection details with your database credentials

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "ashish";

String password = "ashish";

connection = DriverManager.*getConnection*(url, username, password);

statement = connection.createStatement();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** addVaccinationCenter() {

String id = centerIdField.getText();

String name = centerNameField.getText();

String address = centerAddressField.getText();

String contactInfo = centerContactInfoField.getText();

String operatingHours = centerOperatingHoursField.getText();

String capacity = centerCapacityField.getText();

// Add data to the table model

DefaultTableModel model = (DefaultTableModel) vaccinationCenterTable.getModel();

model.addRow(**new** Object[]{id, name, address, contactInfo, operatingHours, capacity});

// Add data to the database

Vector<Object> data = **new** Vector<>();

data.add(id);

data.add(name);

data.add(address);

data.add(contactInfo);

data.add(operatingHours);

data.add(capacity);

addDataToDatabase("vaccinationcenter", data);

}

**private** **void** addVaccinationRecord() {

String id = recordIdField.getText();

String userId = userIdField.getText();

String vaccineId = vaccineIdField.getText();

String date = dateField.getText();

String providerId = providerIdField.getText();

// Add data to the table model

DefaultTableModel model = (DefaultTableModel) vaccinationRecordTable.getModel();

model.addRow(**new** Object[]{id, userId, vaccineId, date, providerId});

// Add data to the database

Vector<Object> data = **new** Vector<>();

data.add(id);

data.add(userId);

data.add(vaccineId);

data.add(date);

data.add(providerId);

addDataToDatabase("vaccinationrecord", data);

// Clear the input fields

recordIdField.setText("");

userIdField.setText("");

vaccineIdField.setText("");

dateField.setText("");

providerIdField.setText("");

}

**private** JPanel createVaccinationRecordTablePanel() {

JPanel panel = **new** JPanel(**new** BorderLayout());

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

JLabel titleLabel = **new** JLabel("Vaccination Records");

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

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

JScrollPane scrollPane = **new** JScrollPane(vaccinationRecordTable);

panel.add(scrollPane, BorderLayout.***CENTER***);

**return** panel;

}

**private** **void** modifyVaccinationCenter() {

**int** selectedRow = vaccinationCenterTable.getSelectedRow();

**if** (selectedRow != -1) {

// Retrieve data from the table model

DefaultTableModel model = (DefaultTableModel) vaccinationCenterTable.getModel();

String id = model.getValueAt(selectedRow, 0).toString();

String name = model.getValueAt(selectedRow, 1).toString();

String address = model.getValueAt(selectedRow, 2).toString();

String contactInfo = model.getValueAt(selectedRow, 3).toString();

String operatingHours = model.getValueAt(selectedRow, 4).toString();

String capacity = model.getValueAt(selectedRow, 5).toString();

// Populate the fields in the vaccination center panel

centerIdField.setText(id);

centerNameField.setText(name);

centerAddressField.setText(address);

centerContactInfoField.setText(contactInfo);

centerOperatingHoursField.setText(operatingHours);

centerCapacityField.setText(capacity);

}

}

**private** **void** deleteVaccinationCenter() {

**int** selectedRow = vaccinationCenterTable.getSelectedRow();

**if** (selectedRow != -1) {

String id = centerIdField.getText();

// Delete data from the table model

DefaultTableModel model = (DefaultTableModel) vaccinationCenterTable.getModel();

model.removeRow(selectedRow);

// Delete data from the database

deleteDataFromDatabase("vaccinationcenter", "ID", id);

// Clear the input fields

centerIdField.setText("");

centerNameField.setText("");

centerAddressField.setText("");

centerContactInfoField.setText("");

centerOperatingHoursField.setText("");

centerCapacityField.setText("");

}

}

**private** **void** addDataToDatabase(String tableName, Vector<Object> data) {

StringBuilder query = **new** StringBuilder("INSERT INTO " + tableName + " VALUES (");

**int** dataSize = data.size();

**for** (**int** i = 0; i < dataSize; i++) {

**if** (i > 0) {

query.append(", ");

}

query.append("?");

}

query.append(")");

**try** (PreparedStatement preparedStatement = connection.prepareStatement(query.toString())) {

**for** (**int** i = 0; i < dataSize; i++) {

preparedStatement.setObject(i + 1, data.get(i));

}

**int** insertedRows = preparedStatement.executeUpdate();

**if** (insertedRows > 0) {

System.***out***.println("Data inserted successfully.");

}

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** addUser() {

String id = userIdField.getText();

String name = userNameField.getText();

String email = userEmailField.getText();

String password = String.*valueOf*(userPasswordField.getPassword());

// Add data to the table model

DefaultTableModel model = (DefaultTableModel) usersTable.getModel();

model.addRow(**new** Object[]{id, name, email});

// Add data to the database

Vector<Object> data = **new** Vector<>();

data.add(id);

data.add(name);

data.add(email);

data.add(password);

addDataToDatabase("users", data);

// Clear the input fields

userIdField.setText("");

userNameField.setText("");

userEmailField.setText("");

userPasswordField.setText("");

}

**private** **void** modifyUser() {

**int** selectedRow = usersTable.getSelectedRow();

**if** (selectedRow != -1) {

String id = userIdField.getText();

String name = userNameField.getText();

String email = userEmailField.getText();

String password = userPasswordField.~~getText~~();

// Modify data in the table model

DefaultTableModel model = (DefaultTableModel) usersTable.getModel();

model.setValueAt(id, selectedRow, 0);

model.setValueAt(name, selectedRow, 1);

model.setValueAt(email, selectedRow, 2);

model.setValueAt(password, selectedRow, 3);

userIdField.setText(id);

userNameField.setText(name);

userEmailField.setText(email);

userPasswordField.setText(password);

// Modify data in the database

modifyUserDataInDatabase(id, name, email, password);

// Clear the input fields

}

}

**private** **void** deleteUser() {

**int** selectedRow = usersTable.getSelectedRow();

**if** (selectedRow != -1) {

// Delete row from the table model

DefaultTableModel model = (DefaultTableModel) usersTable.getModel();

model.removeRow(selectedRow);

// Delete row from the database

String id = userIdField.getText();

deleteDataFromDatabase("users", "ID", id);

// Clear the input fields

userIdField.setText("");

userNameField.setText("");

userEmailField.setText("");

userPasswordField.setText("");

}

}

**private** **void** modifyDataInDatabase(String tableName, **int** row, Vector<Object> newData) {

StringBuilder query = **new** StringBuilder("UPDATE " + tableName + " SET ");

**int** dataSize = newData.size();

**for** (**int** i = 0; i < dataSize; i++) {

**if** (i > 0) {

query.append(", ");

}

query.append(tableName).append(i).append(" = ?");

}

query.append(" WHERE ROWID = (SELECT rid FROM (SELECT ROWID rid FROM ").append(tableName).append(") WHERE ROWNUM <= ? AND ROWNUM >= ?)");

**try** (PreparedStatement preparedStatement = connection.prepareStatement(query.toString())) {

**for** (**int** i = 0; i < dataSize; i++) {

preparedStatement.setObject(i + 1, newData.get(i));

}

preparedStatement.setInt(dataSize + 1, row + 1);

preparedStatement.setInt(dataSize + 2, row + 1);

**int** updatedRows = preparedStatement.executeUpdate();

**if** (updatedRows > 0) {

System.***out***.println("Data modified successfully.");

}

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** deleteDataFromDatabase(String tableName, String columnName, String value) {

String sql = "DELETE FROM " + tableName + " WHERE " + columnName + " = ?";

**try** (PreparedStatement preparedStatement = connection.prepareStatement(sql)) {

preparedStatement.setString(1, value);

preparedStatement.executeUpdate();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**private** **void** modifyUserDataInDatabase(String id, String name, String email, String password) {

**try** {

// Update the "users" table in the database with the modified data

String query = "UPDATE users SET Name=?, Email=?, Password=? WHERE ID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, name);

statement.setString(2, email);

statement.setString(3, password);

statement.setString(4, id);

statement.executeUpdate();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**public** **static** **void** main(String[] args) {

SwingUtilities.*invokeLater*(**new** Runnable() {

@Override

**public** **void** run() {

**new** VaccinationSystemGUI().setVisible(**true**);

}

});

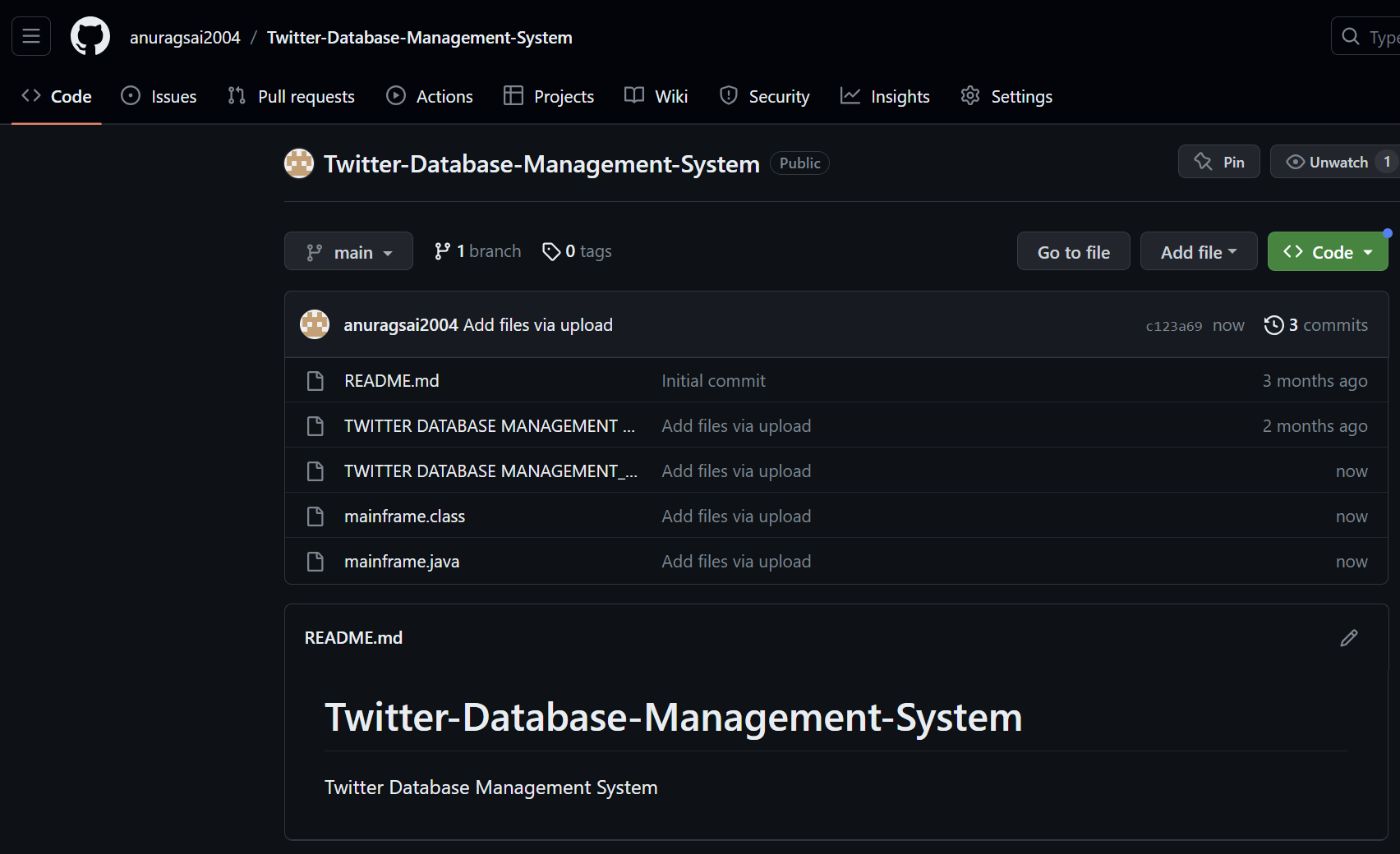
}

}

**GitHub Links and Folder Structure**

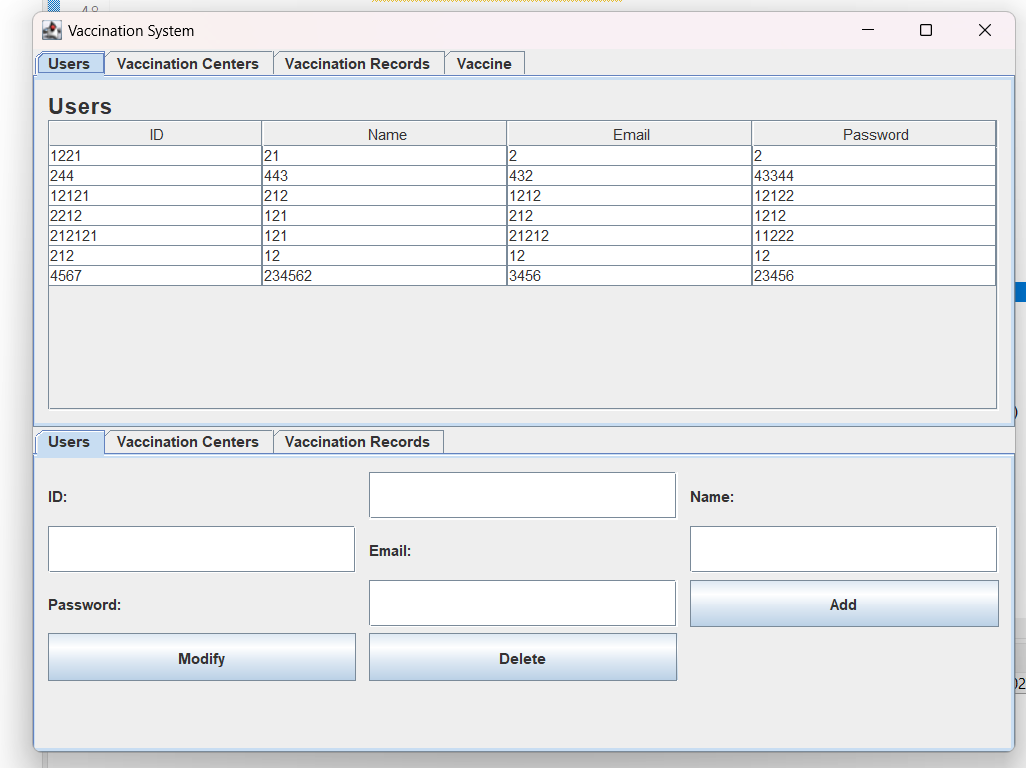
**Link:** <https://github.com/ashishsadhu1/VACCINE-MANAGEMENT-SYSTEM>

**Folder Structure:**

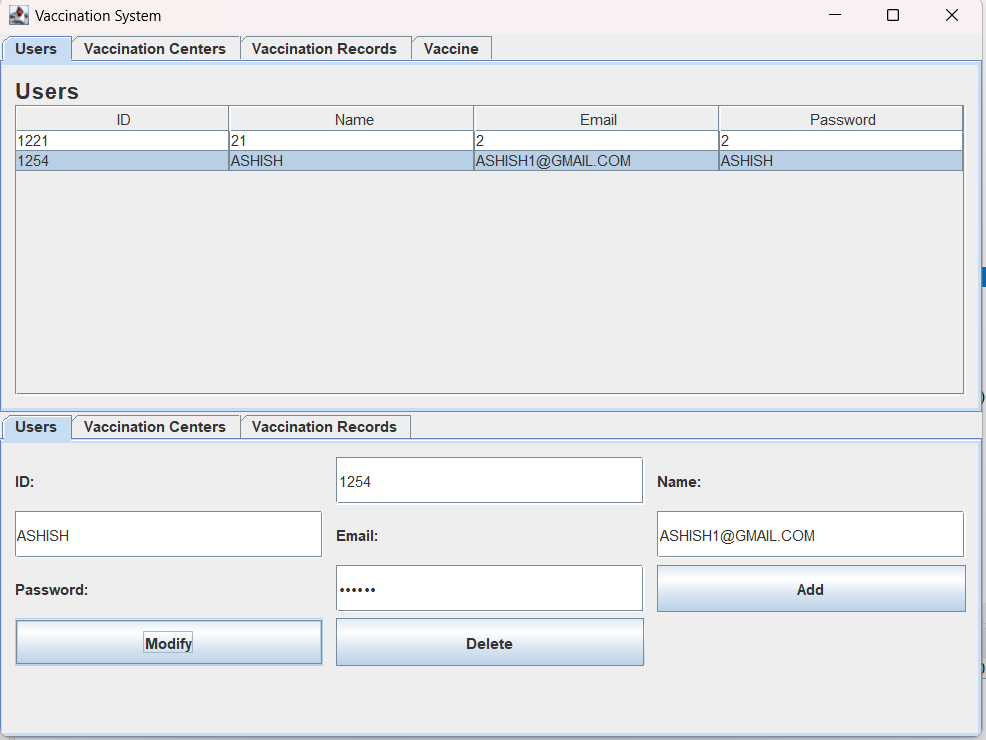
****

# TESTING

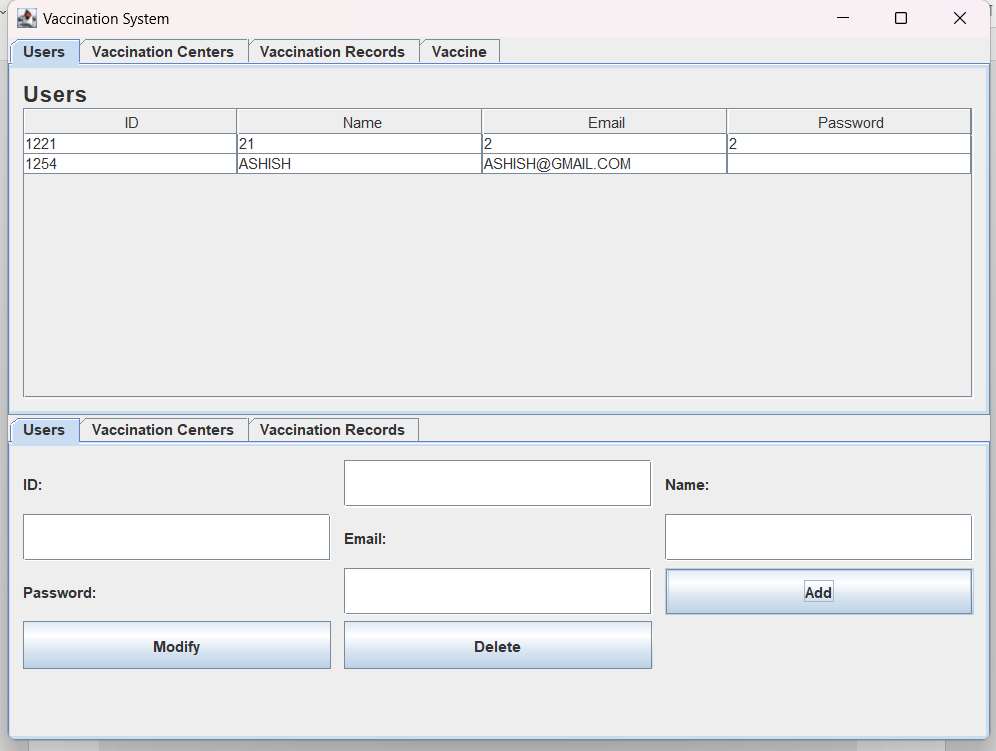
**TWITTER DATBASE MANAGEMENT:**

****

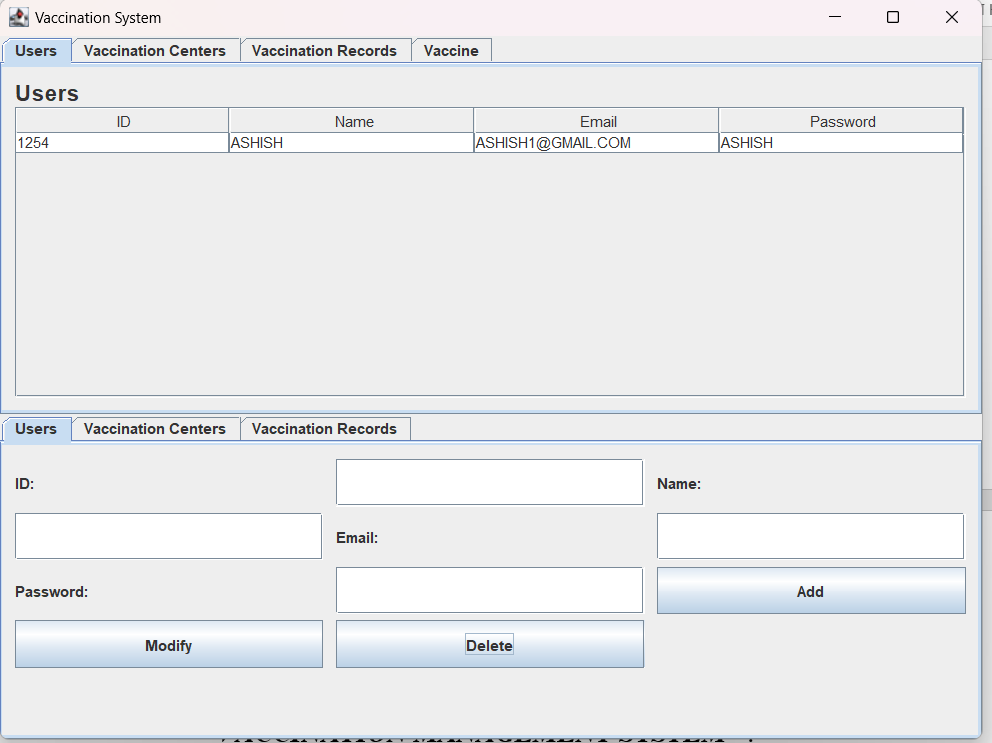
**I.UPDATE PAGE:**

****

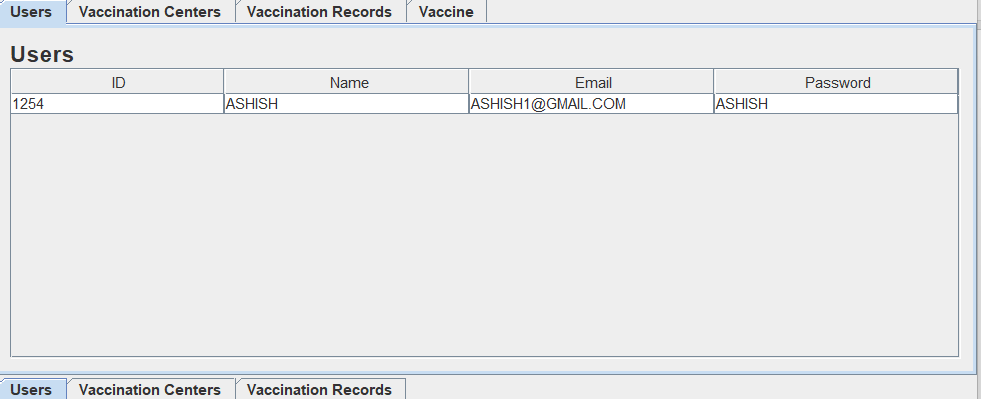
**II.INSERT PAGE**

****

1. **DELETE Page**



1. **VIEW PAGE**

****

**RESULTS**

I have successfully completed the mini-project ***“VACCINATION MANAGEMENT SYSTEM”*** .

## DISCUSSION AND FUTURE WORK

Discussion:

* The Vaccination Management System streamlines operations and improves efficiency by automating tasks such as record-keeping and appointment scheduling.
* Real-time reports and analytics help healthcare authorities monitor immunization coverage and make data-driven decisions for resource allocation.
* The system promotes accessibility for patients through online appointment scheduling and personalized reminders.

Future Work:

* Integration with immunization registries or health information exchanges would enable seamless data sharing and a comprehensive view of a patient's immunization history.
* Enhancements in data analytics capabilities can provide deeper insights into vaccination trends and effectiveness.
* Integration with mobile applications can enable easy access to vaccination information and services for individuals on-the-go.
* Implementation of vaccine passport features to support verification and documentation of vaccination status for travel or other purposes.
* Integration with electronic health records systems to ensure seamless communication and information exchange between healthcare providers.
* Development of an immunization tracking feature to monitor vaccine inventory, expiration dates, and usage patterns for efficient supply chain management.

## REFERENCES

* https://docs.oracle.com/javase/7/docs/api/
* [https://www.javatpoint.com/java-swing](http://www.javatpoint.com/java-swing)
* https://stackoverflow.com/