*A*

*Report*

*Submitted in partial fulfilment of the Requirements for the award of the Degree of*

***BACHELOR OF TECHNOLOGY***

***IN***

### INFORMATION TECHNOLOGY

By

**SHIVA PALLAVI R<1602-22-737-165>**

**SHARATH CHANDRA B<1602-22-737-175>**

#### Under the Guidance of

****Dr. S. Aruna****

****

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

**2023-24**

BONAFIDE CERTIFICATE

This is to certify that this project report titled

‘**CROP DATA MANAGEMENT SYSTEM**’ is a

## project work of SHIVA PALLAVI R and SHARATH CHANDRA B bearing roll no. 1602-22-737-165 and 1602-22-737-175 who carried out this project under my supervision in the III semester for the academic year 2023-24.

Signature

Internal Examiner

Signature

External Examiner

**CROP DATA MANAGEMENT SYSTEM**

### ABSTRACT

The Crop Data Management System (CDMS) is a comprehensive software solution designed to streamline the management of crop-related information for farmers and agricultural professionals. By leveraging modern technologies and database management systems, CDMS offers a user-friendly interface for efficiently storing, organizing, and analyzing crucial data related to crops. It facilitates the seamless integration of various data sources, including farmer details, farm information, pesticide usage, equipment inventory, crop details, profit and loss records, and farmer contacts. With CDMS, users can access real-time data insights, make informed decisions, optimize resource allocation, and enhance overall productivity in the agricultural sector. Whether it's tracking crop growth, monitoring pesticide usage, evaluating equipment performance, or analyzing profitability, CDMS provides a centralized platform to effectively manage and leverage crop data, ultimately contributing to sustainable farming practices and agricultural success.

#### Requirements Analysis:

**List of Tables:**

1. Farmers
2. Farm
3. Pesticides
4. Pesticides\_Farm
5. Equipment
6. Equipment\_Farmer
7. Crop
8. Profit\_Loss
9. Farmer\_F\_Contact

**Attributes list and domain types:**

**Farmers:**

F\_ID NUMBER(38)

F\_FIRSTNAME VARCHAR2(50)

F\_SURNAME VARCHAR2(50)

F\_DOB DATE

F\_DOORNO VARCHAR2(20)

F\_STREETNAME VARCHAR2(50)

F\_VILLAGE VARCHAR2(50)

F\_TYPE VARCHAR2(50)

**Farm:**

FARM\_ID NUMBER(38)

FARM\_AREA NUMBER(38)

FARM\_VILLAGE VARCHAR2(50)

FARM\_MANDAL VARCHAR2(50)

FARM\_SOILTYPE VARCHAR2(50)

F\_ID NUMBER(38)

**Pesticides:**

P\_ID NUMBER(38)

P\_NAME VARCHAR2(60)

P\_MANUFACTURE DATE

P\_EXPIRY DATE

P\_USE VARCHAR2(60)

**Pesticides\_Farm:**

P\_ID NUMBER(38)

FARM\_ID NUMBER(38)

**Equipment:**

EQUIPMENT\_ID NUMBER(38)

EQUIPMENT\_TYPE VARCHAR2(60)

EQUIPMENT\_COST NUMBER(38)

EQUIPMENT\_USE VARCHAR2(60)

**Equipment\_Farmers:**

EQUIPMENT\_ID NUMBER(38)

F\_ID NUMBER(38)

**Crop:**

C\_ID NUMBER(38)

FARM\_ID NUMBER(38)

C\_NAME VARCHAR2(60)

DURATION NUMBER(38)

SEASON\_MONTH VARCHAR2(60)

INVESTMENT NUMBER(38)

**Profit\_Loss:**

F\_ID NUMBER(38)

P\_L\_VALUE NUMBER(10,2)

**Farmer\_F\_Contact:**

F\_ID NUMBER(38)

F\_CONTACT VARCHAR2(12)

# AIM AND PRIORITY OF THE PROJECT

The aim of this project is to develop a comprehensive Crop Data Management System (CDMS) using Java. The system will provide a user-friendly interface to manage and analyze various aspects of crop-related data, including farmer details, farm information, pesticide usage, equipment inventory, crop details, profit/loss records, and farmer contact information.

# ARCHITECTURE AND TECHNOLOGY

**Software Used:**

**Java**: Java will be used for developing the GUI-based desktop application. It offers rich libraries and tools for creating interactive user interfaces.

**MySQL Database**: MySQL will serve as the backend database for storing student information and other relevant data. It provides robust data management capabilities and ensures data integrity.

**Java SE Version 14**: Java Standard Edition (SE) version 14 will be utilized for developing the application. It offers the latest features and enhancements available in the Java platform.

**JDBC (Java Database Connectivity):** JDBC will be used to establish a connection between the Java application and the MySQL database. This will allow seamless interaction between the frontend and backend components of the application.

**Run SQL**: Run SQL is a tool that provides a convenient environment for executing SQL queries and managing the MySQL database.

**Java SWING:**

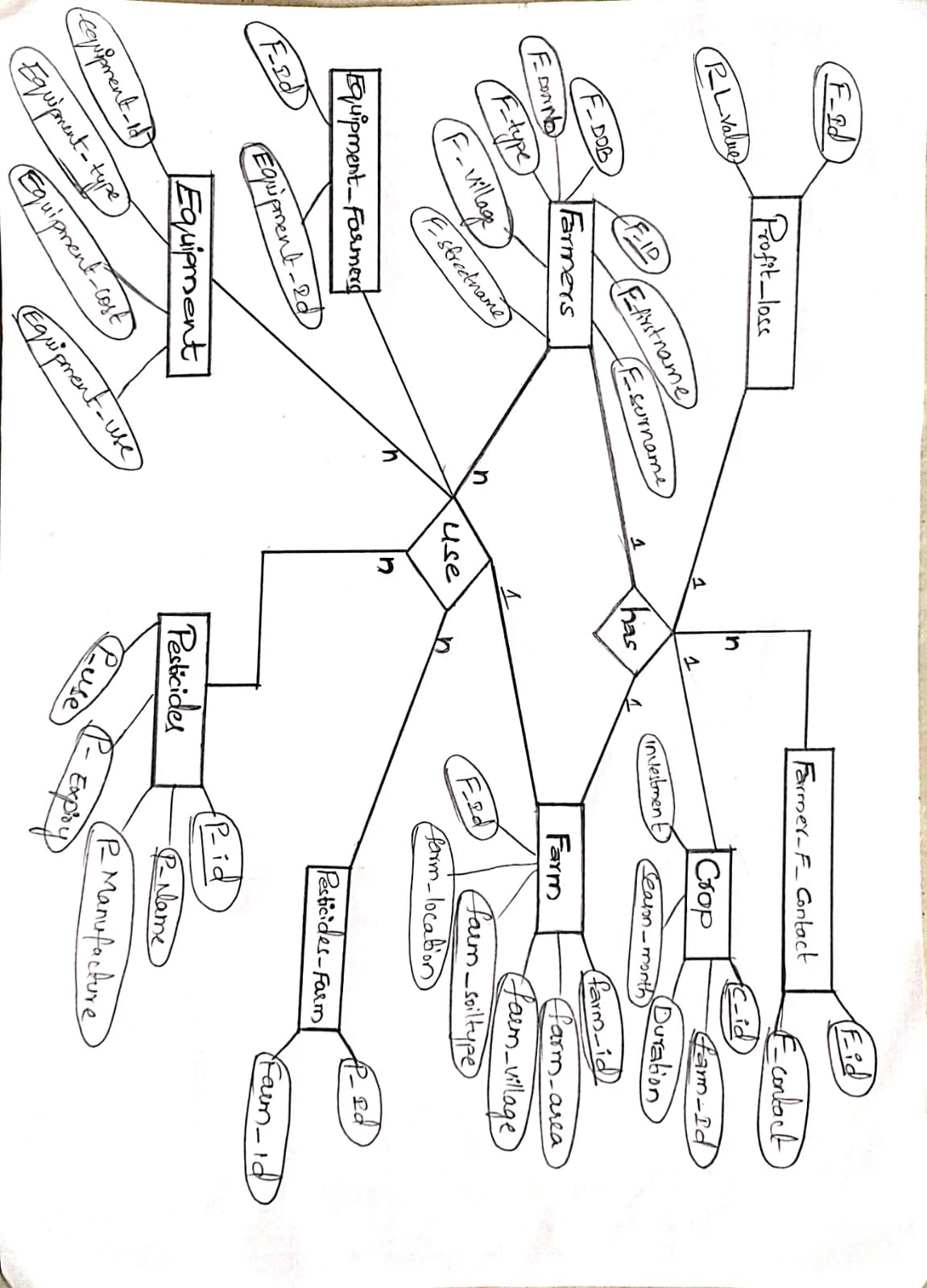
Java SWING is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs.

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

### ER MODELLING:



**Key Constraints:**

**Farmers:**

Primary Key: F\_ID

**Farm:**

Primary Key: FARM\_ID

Foreign Key: F\_ID (references Farmers(F\_ID))

**Pesticides:**

Primary Key: P\_ID

**Pesticides\_Farm**:

Composite Primary Key: (P\_ID, FARM\_ID)

Foreign Key: P\_ID (references Pesticides(P\_ID))

Foreign Key: FARM\_ID (references Farm(FARM\_ID))

**Equipment:**

Primary Key: EQUIPMENT\_ID

**Equipment\_Farmers:**

Composite Primary Key: (EQUIPMENT\_ID, F\_ID)

Foreign Key: EQUIPMENT\_ID (references Equipment(EQUIPMENT\_ID))

Foreign Key: F\_ID (references Farmers(F\_ID))

**Crop:**

Primary Key: C\_ID

Foreign Key: FARM\_ID (references Farm(FARM\_ID))

**Profit\_Loss:**

Composite Primary Key: (F\_ID, P\_L\_VALUE)

Foreign Key: F\_ID (references Farmers(F\_ID))

**Farmer\_F\_Contact:**

Composite Primary Key: (F\_ID, F\_CONTACT)

Foreign Key: F\_ID (references Farmers(F\_ID))

### DDL & DML OPERATIONS:

### Farmers:

### CREATE TABLE Farmers

### (

### F\_Id INT NOT NULL,

### F\_FirstName VARCHAR(50) NOT NULL,

### F\_SurName VARCHAR(50) NOT NULL,

### F\_DOB DATE NOT NULL,

### F\_DoorNo VARCHAR(20) NOT NULL,

### F\_StreetName VARCHAR(50) NOT NULL,

### F\_Village VARCHAR(50) NOT NULL,

### F\_Type VARCHAR(50) NOT NULL,

### PRIMARY KEY (F\_Id)

### );

### INSERT INTO Farmers (F\_ID, F\_FIRSTNAME, F\_SURNAME, F\_DOB, F\_DOORNO, F\_STREETNAME, F\_VILLAGE, F\_TYPE)

### SELECT 7451, 'Veerayya', 'Kasturi', '1962-05-21', '1/7-6F', 'Surya Nagar', 'Chepurapalli', 'Owner'

### UNION ALL SELECT 1524, 'Saidulu', 'Kadiyala', '1989-08-30', '2/5/9F', 'Ashok Nagar', 'Lakkavarapu Kota', 'Tenant'

### UNION ALL SELECT 4963, 'Subbayya', 'Gangavarapu', '1967-02-11', '91-4/A', 'Pransanth Nagar', 'Sabbavaram', 'Owner'

### UNION ALL SELECT 2582, 'Bharat', 'Ogirala', '1965-06-09', '1-2/5B', 'Chanukya Nagar', 'Srungavarapu Kota', 'Owner'

### UNION ALL SELECT 6428, 'Srinu', 'Vadlamani', '1958-03-12', '1-54/A', 'Prasanthi Nagar', 'CM Wada', 'Tenant'

### UNION ALL SELECT 6745, 'Swamy', 'Ivaturi', '1973-12-11', '8-4/1C', 'Rama Rao Colony', 'Pendurthi', 'Owner'

### UNION ALL SELECT 2983, 'Sasidhar', 'Manchiraju', '1992-05-20', '9-8-/A', 'Prakash Nagar', 'Devarapalli', 'Tenant';

### Farm Table:

### REATE TABLE Farm (

### Farm\_Id INT NOT NULL,

### Farm\_Area INT NOT NULL,

### Farm\_Village VARCHAR(50) NOT NULL,

### Farm\_Mandal VARCHAR(50) NOT NULL,

### Farm\_SoilType VARCHAR(50) NOT NULL,

### F\_Id INT NOT NULL,

### PRIMARY KEY (Farm\_Id),

### CONSTRAINT fk\_fid FOREIGN KEY (F\_Id) REFERENCES Farmers(F\_Id)

### );

### INSERT INTO Farm (Farm\_Id, Farm\_Area, Farm\_Village, Farm\_Mandal, Farm\_SoilType, F\_Id, farm\_latitude, farm\_longitude)

### VALUES

### (97482, 3, 'Sabbavaram', 'Sangivalasa', 'Red Soil', 4963, POINT(83.2185, 17.6868)),

### (48642, 2, 'Devarapalli', 'Mangalapalem', 'Black Soil', 2983, POINT(82.9831, 17.4483)),

### (34811, 4, 'Pendurthi', 'Anakapalli', 'Black Soil', 6745, POINT(82.8973, 17.8589)),

### (67418, 1, 'Lakkavarapu Kota', 'Jami', 'Red Soil', 1524, POINT(82.8284, 17.5845)),

### (87522, 1, 'CM Wada', 'Gopalapatnam', 'Black Soil', 6428, POINT(83.2908, 17.6894)),

### (75671, 2, 'Srungavarapu Kota', 'Kothavalasa', 'Red Soil', 2582, POINT(83.1461, 18.0975)),

### (73196, 1, 'Chepurapalli', 'Veera Bhadra Puram', 'Black Soil', 7451, POINT(83.0872, 18.1390));

### Pesticides Table:

### CREATE TABLE Pesticides

### (

### P\_Id INT NOT NULL,

### P\_Name VARCHAR(60) NOT NULL,

### P\_Manufacture DATE NOT NULL,

### P\_Expiry DATE NOT NULL,

### P\_Use VARCHAR(60) NOT NULL,

### PRIMARY KEY (P\_Id)

### );

### INSERT INTO Pesticides (P\_Id, P\_Name, P\_Manufacture, P\_Expiry, P\_Use)

### VALUES

### (7857, 'Sodium Arsenite', '2020-01-11', '2022-03-10', 'Purpose for Plants'),

### (9745, 'Streptomycin', '2020-03-23', '2022-02-20', 'Purpose for Bacteria'),

### (1486, 'Warfarin', '2021-02-13', '2023-05-16', 'Purpose for Rats'),

### (6068, 'Folpet', '2021-07-25', '2024-02-09', 'Purpose for Fungi'),

### (9743, 'Bactivec', '2021-09-17', '2024-03-21', 'Purpose for Larvae'),

### (1525, 'Green Beauveria', '2022-05-09', '2025-03-14', 'Purpose for Insects');

### Pesticides\_Farm Table:

### CREATE TABLE Pesticides\_Farm

### (

### P\_Id INT NOT NULL,

### Farm\_Id INT NOT NULL,

### PRIMARY KEY (P\_Id, Farm\_Id),

### FOREIGN KEY (P\_Id) REFERENCES Pesticides(P\_Id),

### FOREIGN KEY (Farm\_Id) REFERENCES Farm(Farm\_Id)

### );

### INSERT INTO Pesticides\_Farm VALUES

### (7857, 97482),

### (7857, 67418),

### (7857, 87522),

### (9745, 34811),

### (9745, 67418),

### (9745, 73196),

### (1486, 48642),

### (1486, 87522),

### (6068, 97482),

### (6068, 34811),

### (6068, 73196),

### (9743, 34811),

### (9743, 75671),

### (1525, 97482),

### (1525, 67418),

### (1525, 87522);

### Equipment Table:

### CREATE TABLE Equipment

### (

### Equipment\_Id INT NOT NULL,

### Equipment\_Type VARCHAR(60) NOT NULL,

### Equipment\_Cost INT NOT NULL,

### Equipment\_Use VARCHAR(60) NOT NULL,

### PRIMARY KEY (Equipment\_Id)

### );

### INSERT INTO Equipment VALUES

### (75, 'Water Filter Pressure', 17000, 'Watering for Crops'),

### (41, 'Turner Tool Attachment', 15000, 'Trimming Purpose'),

### (61, 'Aeronex Blower', 20000, 'Creating Airflow'),

### (93, 'Disc Harrow', 22000, 'Digging Purpose'),

### (18, 'Plough', 5000, 'Sowing Purpose'),

### (54, 'Hand Push Brush Cutter', 13000, 'Cutting Purpose'),

### (85, 'Tractor', 10000, 'Tilling Purpose');

### Equipment \_Farmers Table:

### CREATE TABLE Equipment\_Farmers

### (

### Equipment\_Id INT NOT NULL,

### F\_Id INT NOT NULL,

### PRIMARY KEY (Equipment\_Id, F\_Id),

### FOREIGN KEY (Equipment\_Id) REFERENCES Equipment(Equipment\_Id),

### FOREIGN KEY (F\_Id) REFERENCES Farmers(F\_Id)

### );

### INSERT INTO Equipment\_Farmers VALUES

### (75, 7451),

### (75, 2983),

### (41, 1524),

### (41, 2983),

### (61, 7451),

### (61, 2983),

### (93, 2582),

### (93, 2983),

### (18, 7451),

### (18, 2983),

### (54, 6428),

### (54, 2983),

### (85, 2582),

### (85, 2983);

### Crop Table:

### CREATE TABLE Crop

### (

### C\_Id INT NOT NULL,

### Farm\_Id INT NOT NULL,

### C\_Name VARCHAR(60) NOT NULL,

### Duration INT NOT NULL,

### Season\_Month VARCHAR(60) NOT NULL,

### Investment INT NOT NULL,

### PRIMARY KEY (C\_Id)

### );

### INSERT INTO Crop VALUES

### (1, 97482, 'Kharif Crops', 5, 'June', 10000),

### (2, 48642, 'Rabi Crops', 6, 'October', 10000),

### (3, 34811, 'Rice', 3, 'July', 15000),

### (4, 67418, 'Zaid Crops', 4, 'March', 12000),

### (5, 87522, 'Wheat', 5, 'November', 16000),

### (6, 75671, 'Cotton', 7, 'August', 20000),

### (7, 73196, 'Sugarcane', 10, 'September', 25000);

### Profit\_Loss Table:

### CREATE TABLE Profit\_Loss

### (

### F\_Id INT NOT NULL,

### P\_L\_Value DECIMAL(10, 2) NOT NULL,

### PRIMARY KEY (F\_Id),

### FOREIGN KEY (F\_Id) REFERENCES Farmers(F\_Id)

### );

### INSERT INTO Profit\_Loss (F\_Id, P\_L\_Value) VALUES

### (7451, 1500.50),

### (1524, -250.75),

### (4963, 1000.00),

### (2582, -1250.25),

### (6428, -458.20),

### (6745, 2000.00),

### (2983, 0.00);

### Farmer\_F\_Contact Table:

### CREATE TABLE Farmer\_F\_Contact

### (

### F\_Id INT NOT NULL,

### F\_Contact VARCHAR(12) NOT NULL,

### PRIMARY KEY (F\_Contact, F\_Id),

### FOREIGN KEY (F\_Id) REFERENCES Farmers(F\_Id)

### );

### INSERT INTO Farmer\_F\_Contact VALUES

### (7451, '7093369035'),

### (1524, '9347429568'),

### (4963, '9059144881'),

### (2582, '9348101001'),

### (6428, '9110312523'),

### (6745, '9492590911'),

### (2983, '9542765317'),

### (6745, '6303341833'),

### (6745, '8332822678'),

### (2582, '7396661751'),

### (2983, '9705563225');

### Tables:

### Farmers:

### C:\Users\MICROCHIP\Pictures\Screenshots\Screenshot (55).png

### Farm:

### C:\Users\MICROCHIP\Pictures\Screenshots\Screenshot (55).png

### Pesticides:

### C:\Users\MICROCHIP\Pictures\Screenshots\Screenshot (55).png

### Pesticides\_Farm:

### C:\Users\MICROCHIP\Pictures\Screenshots\Screenshot (55).png

### Equipment:

### C:\Users\MICROCHIP\Pictures\Screenshots\Screenshot (55).png

### Equipment\_Farmers:

### C:\Users\MICROCHIP\Pictures\Screenshots\Screenshot (55).png

**Crop:**

### C:\Users\MICROCHIP\Pictures\Screenshots\Screenshot (55).png

### Profit\_Loss:

### C:\Users\MICROCHIP\Pictures\Screenshots\Screenshot (55).png

### Farmer\_F\_Contact:

### C:\Users\MICROCHIP\Pictures\Screenshots\Screenshot (55).png

# 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 established using Java programming (JDBC API) as follows:

try (

Connection conn = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/CDMS?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC","sharath", "sharath");

Statement stmt = conn.createStatement();

**NOTE:** We have merged frontend and JDBC program into one program for each individual table.

**HomePage.java**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.IOException;

public class HomePage extends JFrame implements ActionListener {

public HomePage() {

setTitle("CROP DATA MANAGEMENT SYSTEM");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

// Maximize the frame

setExtendedState(JFrame.MAXIMIZED\_BOTH);

// Create a panel for the heading and buttons using GridBagLayout

JPanel panel = new JPanel(new GridBagLayout());

panel.setOpaque(false); // Make panel transparent

GridBagConstraints gbc = new GridBagConstraints();

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.insets = new Insets(10, 10, 10, 10);

JLabel headingLabel = new JLabel("CROP DATA MANAGEMENT SYSTEM");

headingLabel.setFont(new Font("Arial", Font.BOLD, 24));

headingLabel.setHorizontalAlignment(SwingConstants.CENTER);

gbc.gridx = 0;

gbc.gridy = 0;

gbc.gridwidth = 5; // Span across 5 columns

panel.add(headingLabel, gbc);

// Set up background image

ImageIcon backgroundImage = new ImageIcon("F:\\Team-06(1602-22-737-165 & 175)\\project\\images\\cFarm.jpg"); // Replace with your image file path

JLabel backgroundLabel = new JLabel(backgroundImage);

backgroundLabel.setLayout(new BorderLayout());

// Add panel to the background label

backgroundLabel.add(panel, BorderLayout.CENTER);

// Set background label as content pane

setContentPane(backgroundLabel);

// Add buttons

JButton[] buttons = new JButton[10];

String[] buttonLabels = {"FARMERS", "FARM", "PESTICIDES", "PESTICIDES\_FARM", "EQUIPMENT", "EQUIPMENT\_FARMERS", "CROP", "PROFIT\_LOSS", "CONTACTS", "GET FARMER'S DATA"};

String[] filePaths = {

"C:\\MyWebProjects\\FarmersManagement.java",

"C:\\MyWebProjects\\FarmManagement.java",

"C:\\MyWebProjects\\PesticidesManagement.java",

"C:\\MyWebProjects\\Pesticides\_FarmManagement.java",

"C:\\MyWebProjects\\EquipmentManagement.java",

"C:\\MyWebProjects\\Equipment\_FarmersManagement.java",

"C:\\MyWebProjects\\CropManagement.java",

"C:\\MyWebProjects\\Profit\_LossManagement.java",

"C:\\MyWebProjects\\Farmer\_F\_ContactManagement.java",

"C:\\MyWebProjects\\FarmerDataForm.java"

};

gbc.gridwidth = 1; // Reset to 1 for buttons

gbc.gridy = 1; // Start from second row

for (int i = 0; i < 10; i++) {

buttons[i] = new JButton(buttonLabels[i]);

buttons[i].setPreferredSize(new Dimension(200, 100)); // Adjusted button size

buttons[i].setActionCommand(filePaths[i]);

buttons[i].addActionListener(this);

gbc.gridx = i % 5;

if (i % 5 == 0 && i != 0) {

gbc.gridy++;

}

panel.add(buttons[i], gbc);

}

setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

String filePath = e.getActionCommand();

openFile(filePath);

}

private void openFile(String filePath) {

try {

ProcessBuilder pb = new ProcessBuilder("javac", "-cp", ".;C:\\MyWebProjects\\mysql-connector-j-8.4.0\\mysql-connector-j-8.4.0\\mysql-connector-j-8.4.0.jar", filePath);

Process process = pb.start();

int exitCode = process.waitFor();

if (exitCode == 0) {

String className = filePath.substring(filePath.lastIndexOf("\\") + 1, filePath.lastIndexOf("."));

pb = new ProcessBuilder("java", "-cp", ".;C:\\MyWebProjects\\mysql-connector-j-8.4.0\\mysql-connector-j-8.4.0\\mysql-connector-j-8.4.0.jar", className);

process = pb.start();

} else {

System.out.println("Compilation failed for file: " + filePath);

}

} catch (IOException | InterruptedException ex) {

ex.printStackTrace();

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new HomePage());

}

}

**FarmersManagement.java**

import javax.swing.\*;

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 FarmersManagement extends JFrame {

private Connection connection;

private DefaultTableModel tableModel;

private JTable table;

public FarmersManagement() {

try {

// Establish connection

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

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/CDMS?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC", "sharath", "sharath");

// Create UI

setTitle("Farmers Management");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

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

getContentPane().add(panel);

tableModel = new DefaultTableModel();

table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();

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

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

JButton updateButton = new JButton("Update Farmer");

JButton refreshButton = new JButton("Refresh");

buttonPanel.add(addButton);

buttonPanel.add(deleteButton);

buttonPanel.add(updateButton);

buttonPanel.add(refreshButton);

panel.add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addFarmer();

}

});

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deleteFarmer();

}

});

updateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updateFarmer();

}

});

refreshButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

refreshTable();

}

});

// Display table

refreshTable();

setVisible(true);

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void refreshTable() {

try {

// Clear table

tableModel.setRowCount(0);

// Fetch data from database

Statement statement = connection.createStatement();

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

ResultSetMetaData metaData = resultSet.getMetaData();

int columns = metaData.getColumnCount();

// Add column names

Vector<String> columnNames = new Vector<String>();

for (int i = 1; i <= columns; i++) {

columnNames.add(metaData.getColumnName(i));

}

tableModel.setColumnIdentifiers(columnNames);

// Add data rows

while (resultSet.next()) {

Vector<String> rowData = new Vector<String>();

for (int i = 1; i <= columns; i++) {

rowData.add(resultSet.getString(i));

}

tableModel.addRow(rowData);

}

resultSet.close();

statement.close();

} catch (Exception e) {

e.printStackTrace();

}

}

private int getNextFarmerID() {

int nextID = 0;

try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery("SELECT MAX(F\_ID) FROM Farmers");

if (rs.next()) {

nextID = rs.getInt(1) + 1;

} else {

nextID = 1; // If there are no existing farmers, start with ID 1

}

rs.close();

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

return nextID;

}

private void addFarmer() {

try {

String firstName = JOptionPane.showInputDialog("Enter First Name:");

String surname = JOptionPane.showInputDialog("Enter Surname:");

String dob = JOptionPane.showInputDialog("Enter Date of Birth (YYYY-MM-DD):");

String doorNo = JOptionPane.showInputDialog("Enter Door Number:");

String streetName = JOptionPane.showInputDialog("Enter Street Name:");

String village = JOptionPane.showInputDialog("Enter Village:");

String type = JOptionPane.showInputDialog("Enter Type:");

// Insert the new farmer into the database

PreparedStatement pstmt = connection.prepareStatement("INSERT INTO Farmers (F\_ID, F\_FIRSTNAME, F\_SURNAME, F\_DOB, F\_DOORNO, F\_STREETNAME, F\_VILLAGE, F\_TYPE) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");

pstmt.setInt(1, getNextFarmerID());

pstmt.setString(2, firstName);

pstmt.setString(3, surname);

pstmt.setDate(4, Date.valueOf(dob));

pstmt.setString(5, doorNo);

pstmt.setString(6, streetName);

pstmt.setString(7, village);

pstmt.setString(8, type);

pstmt.executeUpdate();

pstmt.close();

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void deleteFarmer() {

try {

String farmerID = JOptionPane.showInputDialog("Enter Farmer ID to delete:");

// Delete the farmer from the database

PreparedStatement pstmt = connection.prepareStatement("DELETE FROM Farmers WHERE F\_ID = ?");

pstmt.setInt(1, Integer.parseInt(farmerID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Farmer deleted successfully!");

} else {

JOptionPane.showMessageDialog(null, "Farmer deletion failed. Farmer ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void updateFarmer() {

try {

String farmerID = JOptionPane.showInputDialog("Enter Farmer ID to update:");

String field = JOptionPane.showInputDialog("Enter field to update (F\_FIRSTNAME, F\_SURNAME, F\_DOB, F\_DOORNO, F\_STREETNAME, F\_VILLAGE, F\_TYPE):");

String newValue = JOptionPane.showInputDialog("Enter new value:");

// Update the farmer in the database

PreparedStatement pstmt = connection.prepareStatement("UPDATE Farmers SET " + field + " = ? WHERE F\_ID = ?");

pstmt.setString(1, newValue);

pstmt.setInt(2, Integer.parseInt(farmerID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Farmer updated successfully!");

} else {

JOptionPane.showMessageDialog(null, "Farmer update failed. Farmer ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

FarmersManagement farmersManagement = new FarmersManagement();

}

}

**FarmManagement.java**

import javax.swing.\*;

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 FarmManagement extends JFrame {

private Connection connection;

private DefaultTableModel tableModel;

private JTable table;

public FarmManagement() {

try {

// Establish connection

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

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/CDMS?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC", "sharath", "sharath");

// Create UI

setTitle("Farm Management");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

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

getContentPane().add(panel);

tableModel = new DefaultTableModel();

table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();

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

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

JButton updateButton = new JButton("Update Farm");

JButton refreshButton = new JButton("Refresh");

buttonPanel.add(addButton);

buttonPanel.add(deleteButton);

buttonPanel.add(updateButton);

buttonPanel.add(refreshButton);

panel.add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addFarm();

}

});

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deleteFarm();

}

});

updateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updateFarm();

}

});

refreshButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

refreshTable();

}

});

// Display table

refreshTable();

setVisible(true);

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void refreshTable() {

try {

// Clear table

tableModel.setRowCount(0);

// Fetch data from database

Statement statement = connection.createStatement();

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

ResultSetMetaData metaData = resultSet.getMetaData();

int columns = metaData.getColumnCount();

// Add column names

Vector<String> columnNames = new Vector<String>();

for (int i = 1; i <= columns; i++) {

columnNames.add(metaData.getColumnName(i));

}

tableModel.setColumnIdentifiers(columnNames);

// Add data rows

while (resultSet.next()) {

Vector<String> rowData = new Vector<String>();

for (int i = 1; i <= columns; i++) {

rowData.add(resultSet.getString(i));

}

tableModel.addRow(rowData);

}

resultSet.close();

statement.close();

} catch (Exception e) {

e.printStackTrace();

}

}

private int getNextFarmID() {

int nextID = 0;

try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery("SELECT MAX(FARM\_ID) FROM Farm");

if (rs.next()) {

nextID = rs.getInt(1) + 1;

} else {

nextID = 1; // If there are no existing farms, start with ID 1

}

rs.close();

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

return nextID;

}

private void addFarm() {

try {

String area = JOptionPane.showInputDialog("Enter Area:");

String village = JOptionPane.showInputDialog("Enter Village:");

String mandal = JOptionPane.showInputDialog("Enter Mandal:");

String soilType = JOptionPane.showInputDialog("Enter Soil Type:");

String farmerID = JOptionPane.showInputDialog("Enter Farmer ID:");

// Insert the new farm into the database

PreparedStatement pstmt = connection.prepareStatement("INSERT INTO Farm (Farm\_Id, Farm\_Area, Farm\_Village, Farm\_Mandal, Farm\_SoilType, F\_Id) VALUES (?, ?, ?, ?, ?, ?)");

pstmt.setInt(1, getNextFarmID());

pstmt.setString(2, area);

pstmt.setString(3, village);

pstmt.setString(4, mandal);

pstmt.setString(5, soilType);

pstmt.setInt(6, Integer.parseInt(farmerID));

pstmt.executeUpdate();

pstmt.close();

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void deleteFarm() {

try {

String farmID = JOptionPane.showInputDialog("Enter Farm ID to delete:");

// Delete the farm from the database

PreparedStatement pstmt = connection.prepareStatement("DELETE FROM Farm WHERE FARM\_ID = ?");

pstmt.setInt(1, Integer.parseInt(farmID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Farm deleted successfully!");

} else {

JOptionPane.showMessageDialog(null, "Farm deletion failed. Farm ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void updateFarm() {

try {

String farmID = JOptionPane.showInputDialog("Enter Farm ID to update:");

String field = JOptionPane.showInputDialog("Enter field to update (FARM\_AREA, FARM\_VILLAGE, FARM\_MANDAL, FARM\_SOILTYPE, F\_ID):");

String newValue = JOptionPane.showInputDialog("Enter new value:");

// Update the farm in the database

PreparedStatement pstmt = connection.prepareStatement("UPDATE Farm SET " + field + " = ? WHERE FARM\_ID = ?");

pstmt.setString(1, newValue);

pstmt.setInt(2, Integer.parseInt(farmID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Farm updated successfully!");

} else {

JOptionPane.showMessageDialog(null, "Farm update failed. Farm ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

FarmManagement farmManagement = new FarmManagement();

}

}

**PesticidesManagement.java**

import javax.swing.\*;

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 PesticidesManagement extends JFrame {

private Connection connection;

private DefaultTableModel tableModel;

private JTable table;

public PesticidesManagement() {

try {

// Establish connection

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

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/CDMS?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC", "sharath", "sharath");

// Create UI

setTitle("Pesticides Management");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

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

getContentPane().add(panel);

tableModel = new DefaultTableModel();

table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();

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

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

JButton updateButton = new JButton("Update Pesticide");

JButton refreshButton = new JButton("Refresh");

buttonPanel.add(addButton);

buttonPanel.add(deleteButton);

buttonPanel.add(updateButton);

buttonPanel.add(refreshButton);

panel.add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addPesticide();

}

});

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deletePesticide();

}

});

updateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updatePesticide();

}

});

refreshButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

refreshTable();

}

});

// Display table

refreshTable();

setVisible(true);

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void refreshTable() {

try {

// Clear table

tableModel.setRowCount(0);

// Fetch data from database

Statement statement = connection.createStatement();

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

ResultSetMetaData metaData = resultSet.getMetaData();

int columns = metaData.getColumnCount();

// Add column names

Vector<String> columnNames = new Vector<String>();

for (int i = 1; i <= columns; i++) {

columnNames.add(metaData.getColumnName(i));

}

tableModel.setColumnIdentifiers(columnNames);

// Add data rows

while (resultSet.next()) {

Vector<String> rowData = new Vector<String>();

for (int i = 1; i <= columns; i++) {

rowData.add(resultSet.getString(i));

}

tableModel.addRow(rowData);

}

resultSet.close();

statement.close();

} catch (Exception e) {

e.printStackTrace();

}

}

private int getNextPesticideID() {

int nextID = 0;

try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery("SELECT MAX(P\_Id) FROM Pesticides");

if (rs.next()) {

nextID = rs.getInt(1) + 1;

} else {

nextID = 1; // If there are no existing pesticides, start with ID 1

}

rs.close();

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

return nextID;

}

private void addPesticide() {

try {

String name = JOptionPane.showInputDialog("Enter Pesticide Name:");

String manufactureDate = JOptionPane.showInputDialog("Enter Manufacture Date (YYYY-MM-DD):");

String expiryDate = JOptionPane.showInputDialog("Enter Expiry Date (YYYY-MM-DD):");

String use = JOptionPane.showInputDialog("Enter Use:");

// Insert the new pesticide into the database

PreparedStatement pstmt = connection.prepareStatement("INSERT INTO Pesticides (P\_Id, P\_Name, P\_Manufacture, P\_Expiry, P\_Use) VALUES (?, ?, ?, ?, ?)");

pstmt.setInt(1, getNextPesticideID());

pstmt.setString(2, name);

pstmt.setDate(3, Date.valueOf(manufactureDate));

pstmt.setDate(4, Date.valueOf(expiryDate));

pstmt.setString(5, use);

pstmt.executeUpdate();

pstmt.close();

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void deletePesticide() {

try {

String pesticideID = JOptionPane.showInputDialog("Enter Pesticide ID to delete:");

// Delete the pesticide from the database

PreparedStatement pstmt = connection.prepareStatement("DELETE FROM Pesticides WHERE P\_Id = ?");

pstmt.setInt(1, Integer.parseInt(pesticideID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Pesticide deleted successfully!");

} else {

JOptionPane.showMessageDialog(null, "Pesticide deletion failed. Pesticide ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void updatePesticide() {

try {

String pesticideID = JOptionPane.showInputDialog("Enter Pesticide ID to update:");

String field = JOptionPane.showInputDialog("Enter field to update (P\_Name, P\_Manufacture, P\_Expiry, P\_Use):");

String newValue = JOptionPane.showInputDialog("Enter new value:");

// Update the pesticide in the database

PreparedStatement pstmt = connection.prepareStatement("UPDATE Pesticides SET " + field + " = ? WHERE P\_Id = ?");

pstmt.setString(1, newValue);

pstmt.setInt(2, Integer.parseInt(pesticideID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Pesticide updated successfully!");

} else {

JOptionPane.showMessageDialog(null, "Pesticide update failed. Pesticide ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

PesticidesManagement pesticidesManagement = new PesticidesManagement();

}

}

**Pesticides\_FarmManagement.java**

import javax.swing.\*;

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 Pesticides\_FarmManagement extends JFrame {

private Connection connection;

private DefaultTableModel tableModel;

private JTable table;

public Pesticides\_FarmManagement() {

try {

// Establish connection

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

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/CDMS?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC", "sharath", "sharath");

// Create UI

setTitle("Pesticides Farm Management");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

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

getContentPane().add(panel);

tableModel = new DefaultTableModel();

table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();

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

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

JButton updateButton = new JButton("Update Pesticide");

JButton refreshButton = new JButton("Refresh");

buttonPanel.add(addButton);

buttonPanel.add(deleteButton);

buttonPanel.add(updateButton);

buttonPanel.add(refreshButton);

panel.add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addPesticide();

}

});

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deletePesticide();

}

});

updateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updatePesticide();

}

});

refreshButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

refreshTable();

}

});

// Display table

refreshTable();

setVisible(true);

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void refreshTable() {

try {

// Clear table

tableModel.setRowCount(0);

// Fetch data from database

Statement statement = connection.createStatement();

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

ResultSetMetaData metaData = resultSet.getMetaData();

int columns = metaData.getColumnCount();

// Add column names

Vector<String> columnNames = new Vector<String>();

for (int i = 1; i <= columns; i++) {

columnNames.add(metaData.getColumnName(i));

}

tableModel.setColumnIdentifiers(columnNames);

// Add data rows

while (resultSet.next()) {

Vector<String> rowData = new Vector<String>();

for (int i = 1; i <= columns; i++) {

rowData.add(resultSet.getString(i));

}

tableModel.addRow(rowData);

}

resultSet.close();

statement.close();

} catch (Exception e) {

e.printStackTrace();

}

}

private int getNextPesticideID() {

int nextID = 0;

try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery("SELECT MAX(P\_Id) FROM Pesticides\_Farm");

if (rs.next()) {

nextID = rs.getInt(1) + 1;

} else {

nextID = 1; // If there are no existing pesticides, start with ID 1

}

rs.close();

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

return nextID;

}

private void addPesticide() {

try {

String PId = JOptionPane.showInputDialog("Enter P\_Id:");

String FarmId = JOptionPane.showInputDialog("Enter Farm\_Id:");

// Convert FarmId to integer

int farmIdInt = Integer.parseInt(FarmId);

// Insert the new pesticide into the database

PreparedStatement pstmt = connection.prepareStatement("INSERT INTO Pesticides\_Farm (P\_Id, Farm\_Id) VALUES (?, ?)");

pstmt.setInt(1, getNextPesticideID());

pstmt.setInt(2, farmIdInt); // Use the integer value

pstmt.executeUpdate();

pstmt.close();

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void deletePesticide() {

try {

String pesticideID = JOptionPane.showInputDialog("Enter Pesticide ID to delete:");

// Delete the pesticide from the database

PreparedStatement pstmt = connection.prepareStatement("DELETE FROM Pesticides\_Farm WHERE P\_Id = ?");

pstmt.setInt(1, Integer.parseInt(pesticideID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Pesticide deleted successfully!");

} else {

JOptionPane.showMessageDialog(null, "Pesticide deletion failed. Pesticide ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void updatePesticide() {

try {

String pesticideID = JOptionPane.showInputDialog("Enter Pesticide ID to update:");

String field = JOptionPane.showInputDialog("Enter field to update (P\_Name, P\_Manufacture, P\_Expiry, P\_Use):");

String newValue = JOptionPane.showInputDialog("Enter new value:");

// Update the pesticide in the database

PreparedStatement pstmt = connection.prepareStatement("UPDATE Pesticides\_Farm SET " + field + " = ? WHERE P\_Id = ?");

pstmt.setString(1, newValue);

pstmt.setInt(2, Integer.parseInt(pesticideID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Pesticide updated successfully!");

} else {

JOptionPane.showMessageDialog(null, "Pesticide update failed. Pesticide ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

Pesticides\_FarmManagement pesticidesFarmManagement = new Pesticides\_FarmManagement();

}

}

**EquipmentManagement.java**

import javax.swing.\*;

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 EquipmentManagement extends JFrame {

private Connection connection;

private DefaultTableModel tableModel;

private JTable table;

public EquipmentManagement() {

try {

// Establish connection

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

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/CDMS?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC", "sharath", "sharath");

// Create UI

setTitle("Equipment Management");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

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

getContentPane().add(panel);

tableModel = new DefaultTableModel();

table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();

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

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

JButton updateButton = new JButton("Update Equipment");

JButton refreshButton = new JButton("Refresh");

buttonPanel.add(addButton);

buttonPanel.add(deleteButton);

buttonPanel.add(updateButton);

buttonPanel.add(refreshButton);

panel.add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addEquipment();

}

});

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deleteEquipment();

}

});

updateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updateEquipment();

}

});

refreshButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

refreshTable();

}

});

// Display table

refreshTable();

setVisible(true);

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void refreshTable() {

try {

// Clear table

tableModel.setRowCount(0);

// Fetch data from database

Statement statement = connection.createStatement();

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

ResultSetMetaData metaData = resultSet.getMetaData();

int columns = metaData.getColumnCount();

// Add column names

Vector<String> columnNames = new Vector<String>();

for (int i = 1; i <= columns; i++) {

columnNames.add(metaData.getColumnName(i));

}

tableModel.setColumnIdentifiers(columnNames);

// Add data rows

while (resultSet.next()) {

Vector<String> rowData = new Vector<String>();

for (int i = 1; i <= columns; i++) {

rowData.add(resultSet.getString(i));

}

tableModel.addRow(rowData);

}

resultSet.close();

statement.close();

} catch (Exception e) {

e.printStackTrace();

}

}

private int getNextEquipmentID() {

int nextID = 0;

try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery("SELECT MAX(Equipment\_Id) FROM Equipment");

if (rs.next()) {

nextID = rs.getInt(1) + 1;

} else {

nextID = 1; // If there are no existing equipment, start with ID 1

}

rs.close();

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

return nextID;

}

private void addEquipment() {

try {

String type = JOptionPane.showInputDialog("Enter Equipment Type:");

String cost = JOptionPane.showInputDialog("Enter Equipment Cost:");

String use = JOptionPane.showInputDialog("Enter Equipment Use:");

// Insert the new equipment into the database

PreparedStatement pstmt = connection.prepareStatement("INSERT INTO Equipment (Equipment\_Id, Equipment\_Type, Equipment\_Cost, Equipment\_Use) VALUES (?, ?, ?, ?)");

pstmt.setInt(1, getNextEquipmentID());

pstmt.setString(2, type);

pstmt.setInt(3, Integer.parseInt(cost));

pstmt.setString(4, use);

pstmt.executeUpdate();

pstmt.close();

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void deleteEquipment() {

try {

String equipmentID = JOptionPane.showInputDialog("Enter Equipment ID to delete:");

// Delete the equipment from the database

PreparedStatement pstmt = connection.prepareStatement("DELETE FROM Equipment WHERE Equipment\_Id = ?");

pstmt.setInt(1, Integer.parseInt(equipmentID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Equipment deleted successfully!");

} else {

JOptionPane.showMessageDialog(null, "Equipment deletion failed. Equipment ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void updateEquipment() {

try {

String equipmentID = JOptionPane.showInputDialog("Enter Equipment ID to update:");

String field = JOptionPane.showInputDialog("Enter field to update (Equipment\_Type, Equipment\_Cost, Equipment\_Use):");

String newValue = JOptionPane.showInputDialog("Enter new value:");

// Update the equipment in the database

PreparedStatement pstmt = connection.prepareStatement("UPDATE Equipment SET " + field + " = ? WHERE Equipment\_Id = ?");

if (field.equals("Equipment\_Cost")) {

pstmt.setInt(1, Integer.parseInt(newValue));

} else {

pstmt.setString(1, newValue);

}

pstmt.setInt(2, Integer.parseInt(equipmentID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Equipment updated successfully!");

} else {

JOptionPane.showMessageDialog(null, "Equipment update failed. Equipment ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

EquipmentManagement equipmentManagement = new EquipmentManagement();

}

}

**Equipment\_FarmersManagemen.java**

import javax.swing.\*;

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 Equipment\_FarmersManagement extends JFrame {

private Connection connection;

private DefaultTableModel tableModel;

private JTable table;

public Equipment\_FarmersManagement() {

try {

// Establish connection

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

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/CDMS?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC", "sharath", "sharath");

// Create UI

setTitle("Equipment Farmers Management");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

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

getContentPane().add(panel);

tableModel = new DefaultTableModel();

table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();

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

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

JButton updateButton = new JButton("Update Equipment");

JButton refreshButton = new JButton("Refresh");

buttonPanel.add(addButton);

buttonPanel.add(deleteButton);

buttonPanel.add(updateButton);

buttonPanel.add(refreshButton);

panel.add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addEquipment();

}

});

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deleteEquipment();

}

});

updateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updateEquipment();

}

});

refreshButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

refreshTable();

}

});

// Display table

refreshTable();

setVisible(true);

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void refreshTable() {

try {

// Clear table

tableModel.setRowCount(0);

// Fetch data from database

Statement statement = connection.createStatement();

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

ResultSetMetaData metaData = resultSet.getMetaData();

int columns = metaData.getColumnCount();

// Add column names

Vector<String> columnNames = new Vector<String>();

for (int i = 1; i <= columns; i++) {

columnNames.add(metaData.getColumnName(i));

}

tableModel.setColumnIdentifiers(columnNames);

// Add data rows

while (resultSet.next()) {

Vector<String> rowData = new Vector<String>();

for (int i = 1; i <= columns; i++) {

rowData.add(resultSet.getString(i));

}

tableModel.addRow(rowData);

}

resultSet.close();

statement.close();

} catch (Exception e) {

e.printStackTrace();

}

}

private int getNextEquipmentID() {

int nextID = 0;

try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery("SELECT MAX(Equipment\_Id) FROM Equipment\_Farmers");

if (rs.next()) {

nextID = rs.getInt(1) + 1;

} else {

nextID = 1; // If there are no existing equipment, start with ID 1

}

rs.close();

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

return nextID;

}

private void addEquipment() {

try {

String equipmentId = JOptionPane.showInputDialog("Enter Equipment ID:");

String farmerId = JOptionPane.showInputDialog("Enter Farmer ID:");

// Insert the new equipment into the database

PreparedStatement pstmt = connection.prepareStatement("INSERT INTO Equipment\_Farmers (Equipment\_Id, F\_Id) VALUES (?, ?)");

pstmt.setInt(1, getNextEquipmentID());

pstmt.setInt(2, Integer.parseInt(farmerId));

pstmt.executeUpdate();

pstmt.close();

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void deleteEquipment() {

try {

String equipmentID = JOptionPane.showInputDialog("Enter Equipment ID to delete:");

// Delete the equipment from the database

PreparedStatement pstmt = connection.prepareStatement("DELETE FROM Equipment\_Farmers WHERE Equipment\_Id = ?");

pstmt.setInt(1, Integer.parseInt(equipmentID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Equipment deleted successfully!");

} else {

JOptionPane.showMessageDialog(null, "Equipment deletion failed. Equipment ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void updateEquipment() {

try {

String equipmentID = JOptionPane.showInputDialog("Enter Equipment ID to update:");

String field = JOptionPane.showInputDialog("Enter field to update (F\_Id):");

String newValue = JOptionPane.showInputDialog("Enter new value:");

// Update the equipment in the database

PreparedStatement pstmt = connection.prepareStatement("UPDATE Equipment\_Farmers SET " + field + " = ? WHERE Equipment\_Id = ?");

pstmt.setString(1, newValue);

pstmt.setInt(2, Integer.parseInt(equipmentID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Equipment updated successfully!");

} else {

JOptionPane.showMessageDialog(null, "Equipment update failed. Equipment ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

Equipment\_FarmersManagement equipmentFarmersManagement = new Equipment\_FarmersManagement();

}

}

**CropManagement.java**

import javax.swing.\*;

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 CropManagement extends JFrame {

private Connection connection;

private DefaultTableModel tableModel;

private JTable table;

public CropManagement() {

try {

// Establish connection

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

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/CDMS?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC", "sharath", "sharath");

// Create UI

setTitle("Crop Management");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

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

getContentPane().add(panel);

tableModel = new DefaultTableModel();

table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();

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

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

JButton updateButton = new JButton("Update Crop");

JButton refreshButton = new JButton("Refresh");

buttonPanel.add(addButton);

buttonPanel.add(deleteButton);

buttonPanel.add(updateButton);

buttonPanel.add(refreshButton);

panel.add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addCrop();

}

});

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deleteCrop();

}

});

updateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updateCrop();

}

});

refreshButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

refreshTable();

}

});

// Display table

refreshTable();

setVisible(true);

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void refreshTable() {

try {

// Clear table

tableModel.setRowCount(0);

// Fetch data from database

Statement statement = connection.createStatement();

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

ResultSetMetaData metaData = resultSet.getMetaData();

int columns = metaData.getColumnCount();

// Add column names

Vector<String> columnNames = new Vector<String>();

for (int i = 1; i <= columns; i++) {

columnNames.add(metaData.getColumnName(i));

}

tableModel.setColumnIdentifiers(columnNames);

// Add data rows

while (resultSet.next()) {

Vector<String> rowData = new Vector<String>();

for (int i = 1; i <= columns; i++) {

rowData.add(resultSet.getString(i));

}

tableModel.addRow(rowData);

}

resultSet.close();

statement.close();

} catch (Exception e) {

e.printStackTrace();

}

}

private int getNextCropID() {

int nextID = 0;

try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery("SELECT MAX(C\_Id) FROM crop");

if (rs.next()) {

nextID = rs.getInt(1) + 1;

} else {

nextID = 1; // If there are no existing crops, start with ID 1

}

rs.close();

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

return nextID;

}

private void addCrop() {

try {

String farmId = JOptionPane.showInputDialog("Enter Farm ID:");

String name = JOptionPane.showInputDialog("Enter Crop Name:");

String duration = JOptionPane.showInputDialog("Enter Duration:");

String seasonMonth = JOptionPane.showInputDialog("Enter Season Month:");

String investment = JOptionPane.showInputDialog("Enter Investment:");

// Insert the new crop into the database

PreparedStatement pstmt = connection.prepareStatement("INSERT INTO crop (C\_Id, Farm\_Id, C\_Name, Duration, Season\_Month, Investment) VALUES (?, ?, ?, ?, ?, ?)");

pstmt.setInt(1, getNextCropID());

pstmt.setInt(2, Integer.parseInt(farmId));

pstmt.setString(3, name);

pstmt.setInt(4, Integer.parseInt(duration));

pstmt.setString(5, seasonMonth);

pstmt.setInt(6, Integer.parseInt(investment));

pstmt.executeUpdate();

pstmt.close();

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void deleteCrop() {

try {

String cropID = JOptionPane.showInputDialog("Enter Crop ID to delete:");

// Delete the crop from the database

PreparedStatement pstmt = connection.prepareStatement("DELETE FROM crop WHERE C\_Id = ?");

pstmt.setInt(1, Integer.parseInt(cropID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Crop deleted successfully!");

} else {

JOptionPane.showMessageDialog(null, "Crop deletion failed. Crop ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void updateCrop() {

try {

String cropID = JOptionPane.showInputDialog("Enter Crop ID to update:");

String field = JOptionPane.showInputDialog("Enter field to update (Farm\_Id, C\_Name, Duration, Season\_Month, Investment):");

String newValue = JOptionPane.showInputDialog("Enter new value:");

// Update the crop in the database

PreparedStatement pstmt = connection.prepareStatement("UPDATE crop SET " + field + " = ? WHERE C\_Id = ?");

pstmt.setString(1, newValue);

pstmt.setInt(2, Integer.parseInt(cropID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Crop updated successfully!");

} else {

JOptionPane.showMessageDialog(null, "Crop update failed. Crop ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

CropManagement cropManagement = new CropManagement();

}

}

**Profit\_LossManagement.java**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.math.BigDecimal;

import java.sql.\*;

import java.util.Vector;

public class Profit\_LossManagement extends JFrame {

private Connection connection;

private DefaultTableModel tableModel;

private JTable table;

public Profit\_LossManagement() {

try {

// Establish connection

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

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/CDMS?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC", "sharath", "sharath");

// Create UI

setTitle("Profit Loss Management");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

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

getContentPane().add(panel);

tableModel = new DefaultTableModel();

table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();

JButton addButton = new JButton("Add Profit/Loss");

JButton deleteButton = new JButton("Delete Profit/Loss");

JButton updateButton = new JButton("Update Profit/Loss");

JButton refreshButton = new JButton("Refresh");

buttonPanel.add(addButton);

buttonPanel.add(deleteButton);

buttonPanel.add(updateButton);

buttonPanel.add(refreshButton);

panel.add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addProfitLoss();

}

});

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deleteProfitLoss();

}

});

updateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updateProfitLoss();

}

});

refreshButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

refreshTable();

}

});

// Display table

refreshTable();

setVisible(true);

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void refreshTable() {

try {

// Clear table

tableModel.setRowCount(0);

// Fetch data from database

Statement statement = connection.createStatement();

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

ResultSetMetaData metaData = resultSet.getMetaData();

int columns = metaData.getColumnCount();

// Add column names

Vector<String> columnNames = new Vector<>();

for (int i = 1; i <= columns; i++) {

columnNames.add(metaData.getColumnName(i));

}

tableModel.setColumnIdentifiers(columnNames);

// Add data rows

while (resultSet.next()) {

Vector<String> rowData = new Vector<>();

for (int i = 1; i <= columns; i++) {

rowData.add(resultSet.getString(i));

}

tableModel.addRow(rowData);

}

resultSet.close();

statement.close();

} catch (Exception e) {

e.printStackTrace();

}

}

private void addProfitLoss() {

try {

String farmerId = JOptionPane.showInputDialog("Enter Farmer ID:");

String profitLossValue = JOptionPane.showInputDialog("Enter Profit/Loss Value:");

// Insert the new profit/loss into the database

PreparedStatement pstmt = connection.prepareStatement("INSERT INTO Profit\_Loss (F\_Id, P\_L\_Value) VALUES (?, ?)");

pstmt.setInt(1, Integer.parseInt(farmerId));

pstmt.setBigDecimal(2, new BigDecimal(profitLossValue));

pstmt.executeUpdate();

pstmt.close();

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void deleteProfitLoss() {

try {

String farmerId = JOptionPane.showInputDialog("Enter Farmer ID to delete:");

// Delete the profit/loss from the database

PreparedStatement pstmt = connection.prepareStatement("DELETE FROM Profit\_Loss WHERE F\_Id = ?");

pstmt.setInt(1, Integer.parseInt(farmerId));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Profit/Loss deleted successfully!");

} else {

JOptionPane.showMessageDialog(null, "Profit/Loss deletion failed. Farmer ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void updateProfitLoss() {

try {

String farmerId = JOptionPane.showInputDialog("Enter Farmer ID to update:");

String newValue = JOptionPane.showInputDialog("Enter new Profit/Loss Value:");

// Update the profit/loss in the database

PreparedStatement pstmt = connection.prepareStatement("UPDATE Profit\_Loss SET P\_L\_Value = ? WHERE F\_Id = ?");

pstmt.setBigDecimal(1, new BigDecimal(newValue));

pstmt.setInt(2, Integer.parseInt(farmerId));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Profit/Loss updated successfully!");

} else {

JOptionPane.showMessageDialog(null, "Profit/Loss update failed. Farmer ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

Profit\_LossManagement profitLossManagement = new Profit\_LossManagement();

}

}

**Farmer\_F\_ContactManagement.java**

import javax.swing.\*;

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 Farmer\_F\_ContactManagement extends JFrame {

private Connection connection;

private DefaultTableModel tableModel;

private JTable table;

public Farmer\_F\_ContactManagement() {

try {

// Establish connection

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

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/CDMS?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC", "sharath", "sharath");

// Create UI

setTitle("Farmer Contact Management");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

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

getContentPane().add(panel);

tableModel = new DefaultTableModel();

table = new JTable(tableModel);

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();

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

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

JButton updateButton = new JButton("Update Contact");

JButton refreshButton = new JButton("Refresh");

buttonPanel.add(addButton);

buttonPanel.add(deleteButton);

buttonPanel.add(updateButton);

buttonPanel.add(refreshButton);

panel.add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

addContact();

}

});

deleteButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

deleteContact();

}

});

updateButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

updateContact();

}

});

refreshButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

refreshTable();

}

});

// Display table

refreshTable();

setVisible(true);

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void refreshTable() {

try {

// Clear table

tableModel.setRowCount(0);

// Fetch data from database

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT \* FROM Farmer\_F\_Contact");

ResultSetMetaData metaData = resultSet.getMetaData();

int columns = metaData.getColumnCount();

// Add column names

Vector<String> columnNames = new Vector<String>();

for (int i = 1; i <= columns; i++) {

columnNames.add(metaData.getColumnName(i));

}

tableModel.setColumnIdentifiers(columnNames);

// Add data rows

while (resultSet.next()) {

Vector<String> rowData = new Vector<String>();

for (int i = 1; i <= columns; i++) {

rowData.add(resultSet.getString(i));

}

tableModel.addRow(rowData);

}

resultSet.close();

statement.close();

} catch (Exception e) {

e.printStackTrace();

}

}

private int getNextContactID() {

int nextID = 0;

try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery("SELECT MAX(F\_Id) FROM Farmer\_F\_Contact");

if (rs.next()) {

nextID = rs.getInt(1) + 1;

} else {

nextID = 1; // If there are no existing contacts, start with ID 1

}

rs.close();

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

return nextID;

}

private void addContact() {

try {

String farmerId = JOptionPane.showInputDialog("Enter Farmer ID:");

String contact = JOptionPane.showInputDialog("Enter Contact Number:");

// Insert the new contact into the database

PreparedStatement pstmt = connection.prepareStatement("INSERT INTO Farmer\_F\_Contact (F\_ID, F\_CONTACT) VALUES (?, ?)");

pstmt.setInt(1, Integer.parseInt(farmerId));

pstmt.setString(2, contact);

pstmt.executeUpdate();

pstmt.close();

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void deleteContact() {

try {

String contactID = JOptionPane.showInputDialog("Enter Contact ID to delete:");

// Delete the contact from the database

PreparedStatement pstmt = connection.prepareStatement("DELETE FROM Farmer\_F\_Contact WHERE F\_ID = ?");

pstmt.setInt(1, Integer.parseInt(contactID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Contact deleted successfully!");

} else {

JOptionPane.showMessageDialog(null, "Contact deletion failed. Contact ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

private void updateContact() {

try {

String contactID = JOptionPane.showInputDialog("Enter Farmer ID to update:");

String newContact = JOptionPane.showInputDialog("Enter new contact number:");

// Update the contact in the database

PreparedStatement pstmt = connection.prepareStatement("UPDATE Farmer\_F\_Contact SET F\_CONTACT = ? WHERE F\_ID = ?");

pstmt.setString(1, newContact);

pstmt.setInt(2, Integer.parseInt(contactID));

int rowsAffected = pstmt.executeUpdate();

pstmt.close();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(null, "Contact updated successfully!");

} else {

JOptionPane.showMessageDialog(null, "Contact update failed. Contact ID not found.");

}

refreshTable();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

Farmer\_F\_ContactManagement contactManagement = new Farmer\_F\_ContactManagement();

}

}

**FarmerDataForm.java**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class FarmerDataForm extends JFrame {

private JTextField farmerIdField;

private JTable resultTable;

private Connection connection;

public FarmerDataForm() {

// Set up the frame

setTitle("Farmer Data Form");

setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

// Initialize UI components

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

JPanel inputPanel = new JPanel(new FlowLayout());

JLabel farmerIdLabel = new JLabel("Enter Farmer ID: ");

farmerIdField = new JTextField(10);

JButton fetchButton = new JButton("Fetch Data");

JButton printButton = new JButton("Print Data");

inputPanel.add(farmerIdLabel);

inputPanel.add(farmerIdField);

inputPanel.add(fetchButton);

inputPanel.add(printButton);

// Table to display results

resultTable = new JTable();

JScrollPane scrollPane = new JScrollPane(resultTable);

panel.add(inputPanel, BorderLayout.NORTH);

panel.add(scrollPane, BorderLayout.CENTER);

getContentPane().add(panel);

// Add action listener to the button

fetchButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

fetchFarmerData();

}

});

// Add action listener to the print button

printButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

printTableData();

}

});

// Establish database connection

connectToDatabase();

}

private void connectToDatabase() {

try {

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

connection = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/CDMS?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC",

"sharath", "sharath");

} catch (Exception ex) {

ex.printStackTrace();

}

}

private void fetchFarmerData() {

String farmerId = farmerIdField.getText().trim();

if (farmerId.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please enter a Farmer ID.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String query = "SELECT F.F\_ID, F.F\_FIRSTNAME, F.F\_SURNAME, F.F\_DOB, F.F\_DOORNO, F.F\_STREETNAME, F.F\_VILLAGE, F.F\_TYPE, " +

"FA.FARM\_AREA, FA.FARM\_VILLAGE, FA.FARM\_MANDAL, FA.FARM\_SOILTYPE, " +

"P.P\_NAME, P.P\_MANUFACTURE, P.P\_EXPIRY, P.P\_USE, " +

"E.EQUIPMENT\_TYPE, E.EQUIPMENT\_COST, E.EQUIPMENT\_USE, " +

"C.C\_NAME, C.DURATION, C.SEASON\_MONTH, C.INVESTMENT, " +

"PL.P\_L\_VALUE, " +

"FFC.F\_CONTACT " +

"FROM Farmers F " +

"LEFT JOIN Farm FA ON F.F\_ID = FA.F\_ID " +

"LEFT JOIN Pesticides\_Farm PF ON FA.FARM\_ID = PF.FARM\_ID " +

"LEFT JOIN Pesticides P ON PF.P\_ID = P.P\_ID " +

"LEFT JOIN Equipment\_Farmers EF ON F.F\_ID = EF.F\_ID " +

"LEFT JOIN Equipment E ON EF.EQUIPMENT\_ID = E.EQUIPMENT\_ID " +

"LEFT JOIN Crop C ON FA.FARM\_ID = C.FARM\_ID " +

"LEFT JOIN Profit\_Loss PL ON F.F\_ID = PL.F\_ID " +

"LEFT JOIN Farmer\_F\_Contact FFC ON F.F\_ID = FFC.F\_ID " +

"WHERE F.F\_ID = ?";

PreparedStatement pstmt = connection.prepareStatement(query);

pstmt.setInt(1, Integer.parseInt(farmerId));

ResultSet resultSet = pstmt.executeQuery();

// Clear previous results

resultTable.setModel(new DefaultTableModel());

// If there is data, display it

if (resultSet.next()) {

String[] columnNames = {

"Field", "Value"

};

Object[][] data = {

{"Farmer ID", resultSet.getInt("F\_ID")},

{"First Name", resultSet.getString("F\_FIRSTNAME")},

{"Surname", resultSet.getString("F\_SURNAME")},

{"Date of Birth", resultSet.getDate("F\_DOB")},

{"Door Number", resultSet.getString("F\_DOORNO")},

{"Street Name", resultSet.getString("F\_STREETNAME")},

{"Village", resultSet.getString("F\_VILLAGE")},

{"Type", resultSet.getString("F\_TYPE")},

{"Farm Area", resultSet.getDouble("FARM\_AREA")},

{"Farm Village", resultSet.getString("FARM\_VILLAGE")},

{"Farm Mandal", resultSet.getString("FARM\_MANDAL")},

{"Farm Soil Type", resultSet.getString("FARM\_SOILTYPE")},

{"Pesticide Name", resultSet.getString("P\_NAME")},

{"Pesticide Manufacture Date", resultSet.getDate("P\_MANUFACTURE")},

{"Pesticide Expiry Date", resultSet.getDate("P\_EXPIRY")},

{"Pesticide Use", resultSet.getString("P\_USE")},

{"Equipment Type", resultSet.getString("EQUIPMENT\_TYPE")},

{"Equipment Cost", resultSet.getDouble("EQUIPMENT\_COST")},

{"Equipment Use", resultSet.getString("EQUIPMENT\_USE")},

{"Crop Name", resultSet.getString("C\_NAME")},

{"Duration", resultSet.getInt("DURATION")},

{"Season Month", resultSet.getString("SEASON\_MONTH")},

{"Investment", resultSet.getDouble("INVESTMENT")},

{"Profit/Loss Value", resultSet.getDouble("P\_L\_VALUE")},

{"Contact", resultSet.getString("F\_CONTACT")},

};

DefaultTableModel model = new DefaultTableModel(data, columnNames);

resultTable.setModel(model);

} else {

JOptionPane.showMessageDialog(this, "No data found for Farmer ID: " + farmerId);

}

resultSet.close();

pstmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void printTableData() {

try {

if (!resultTable.print()) {

System.err.println("User cancelled printing");

}

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

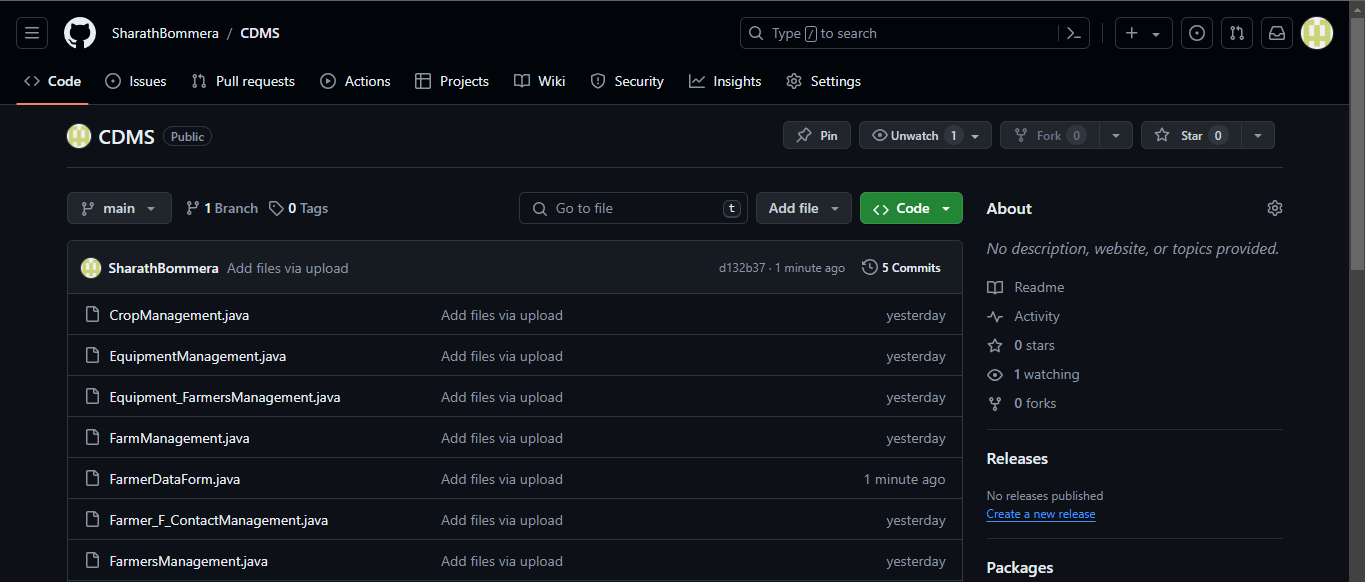
SwingUtilities.invokeLater(() -> new FarmerDataForm().setVisible(true));

}

}

# GITHUB LINK AND FOLDER STRUCTURE

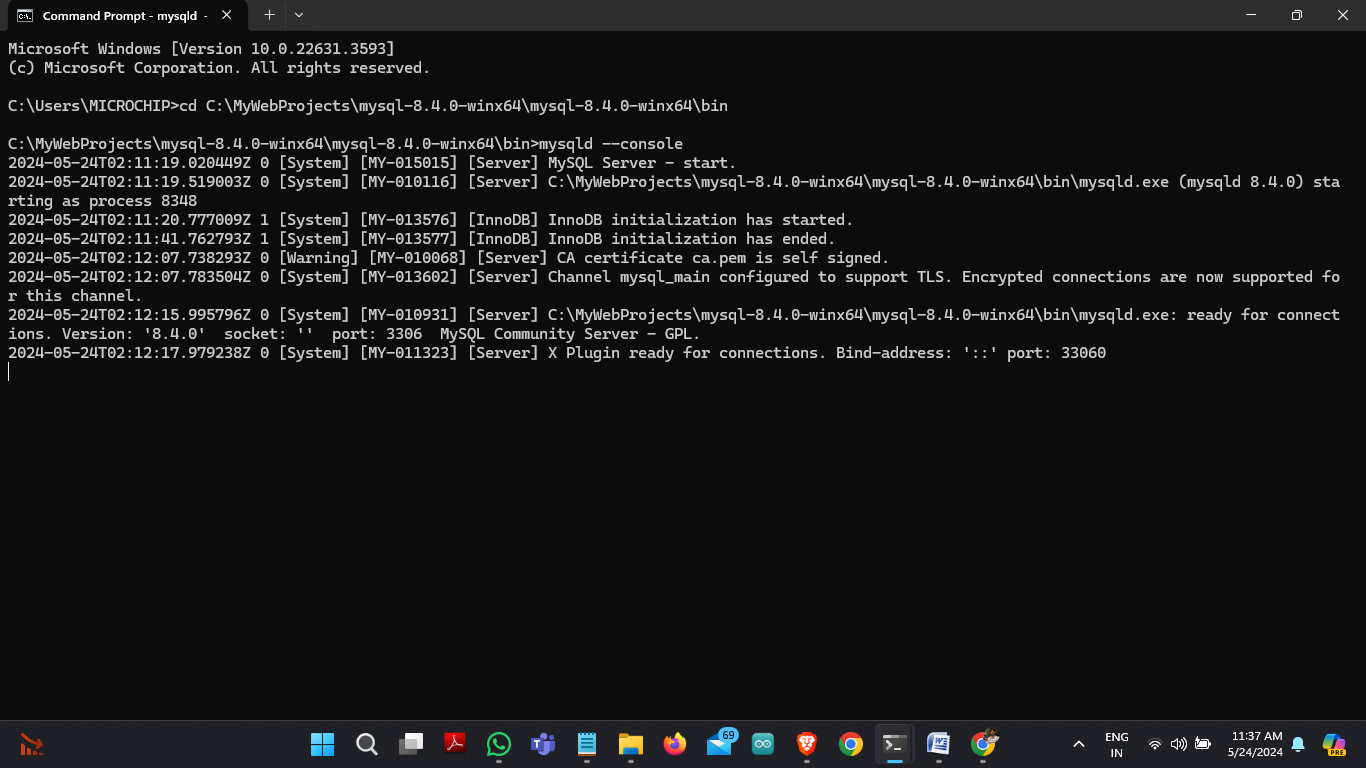
**Link:** <https://github.com/SharathBommera/CDMS>



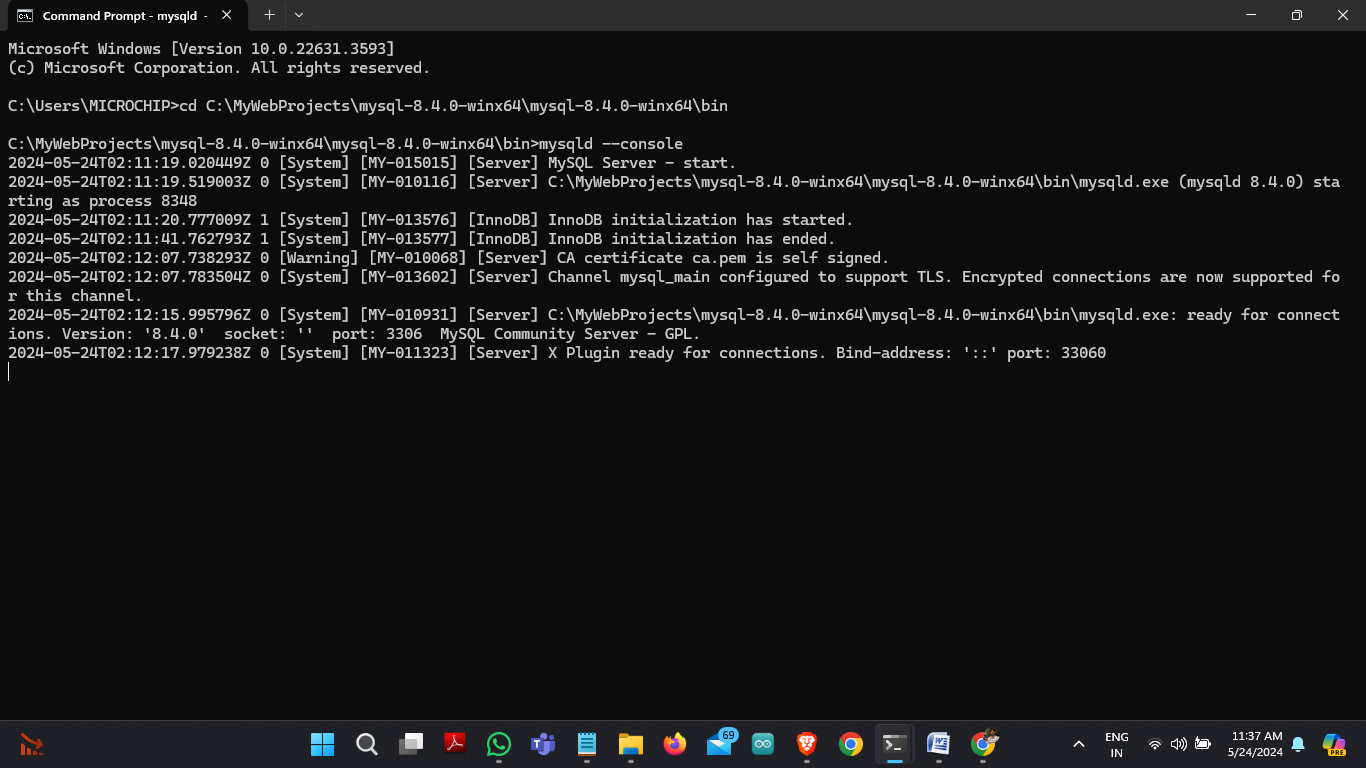


# TESTING

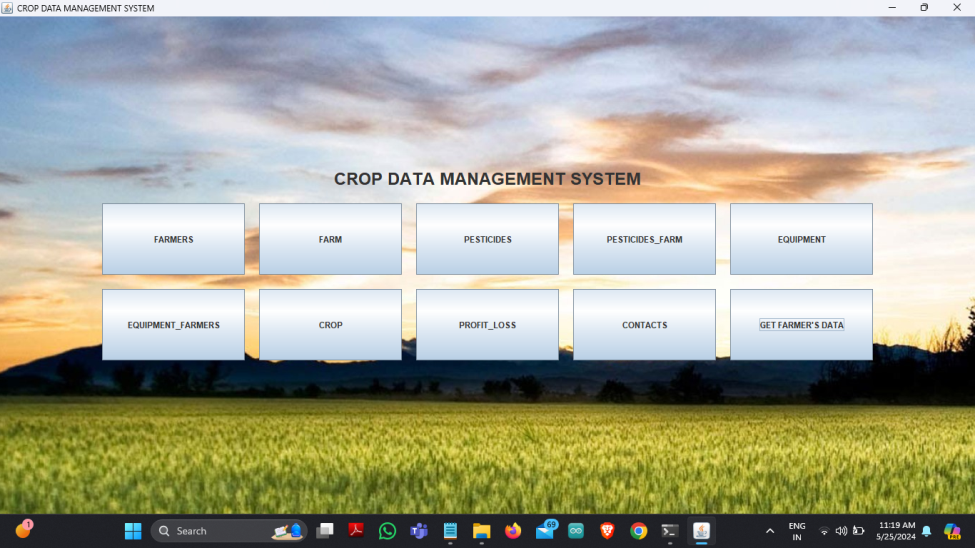
**Starting MySqld server:**



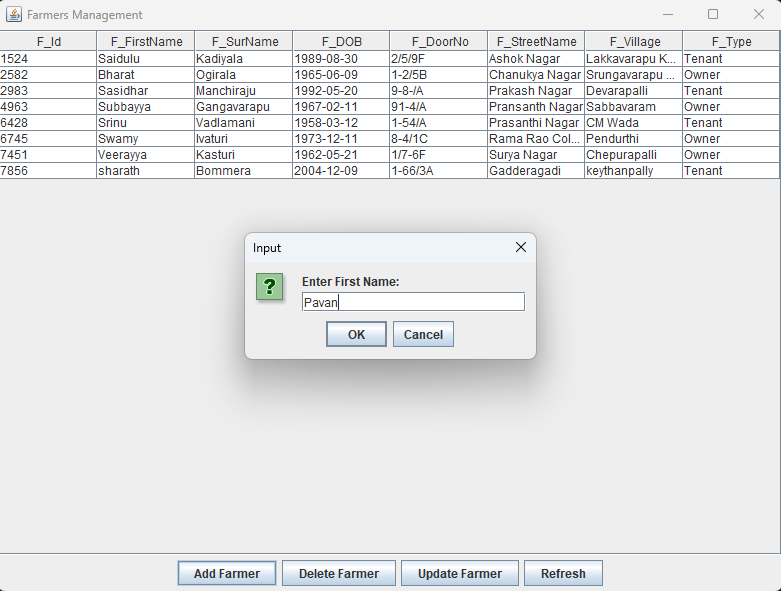
**Running MySql Monitor:**

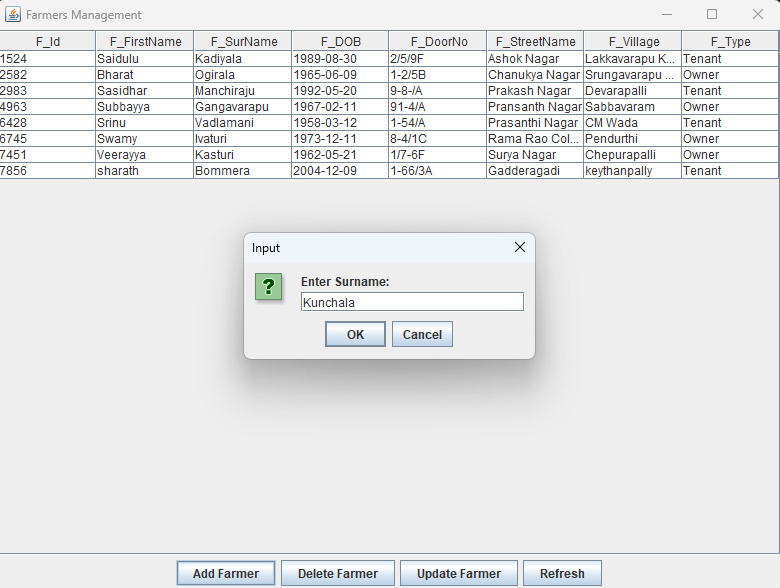


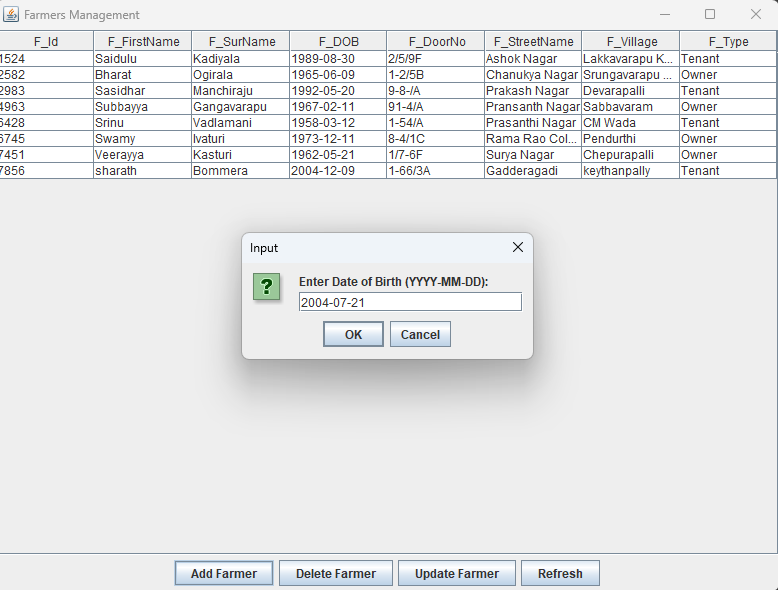
**HomePage:**

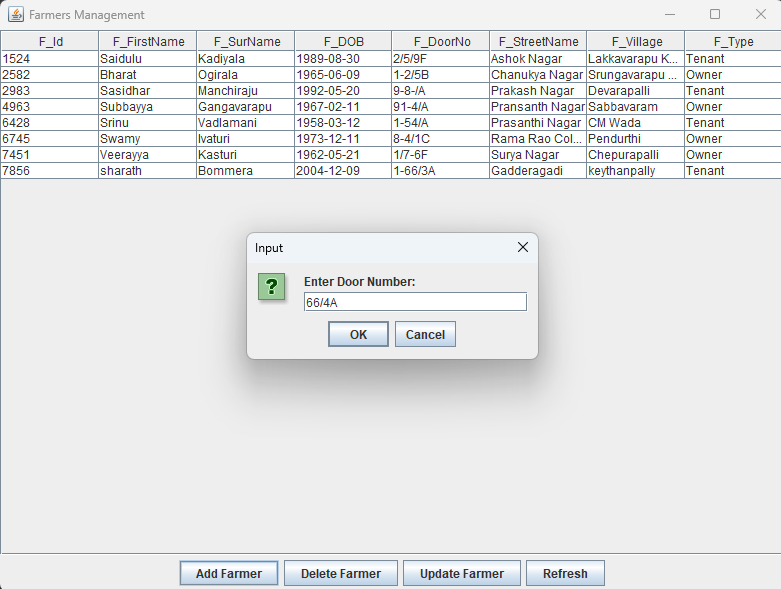


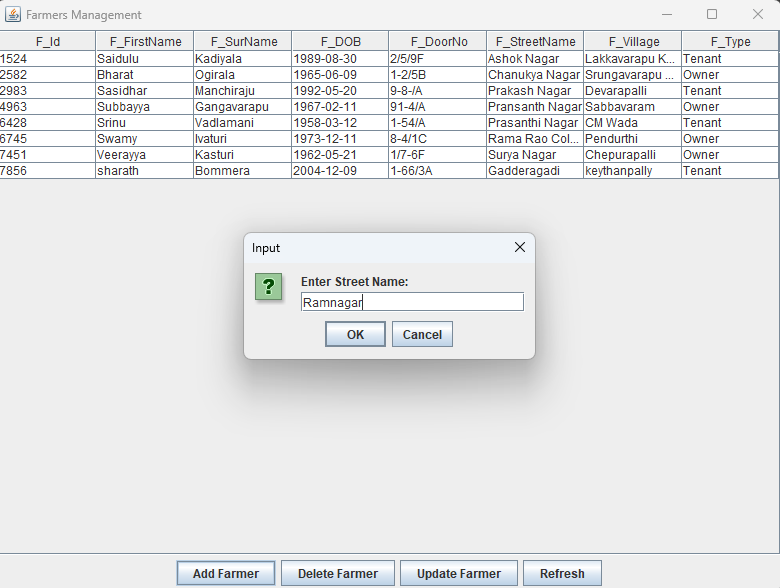
**Farmers:**

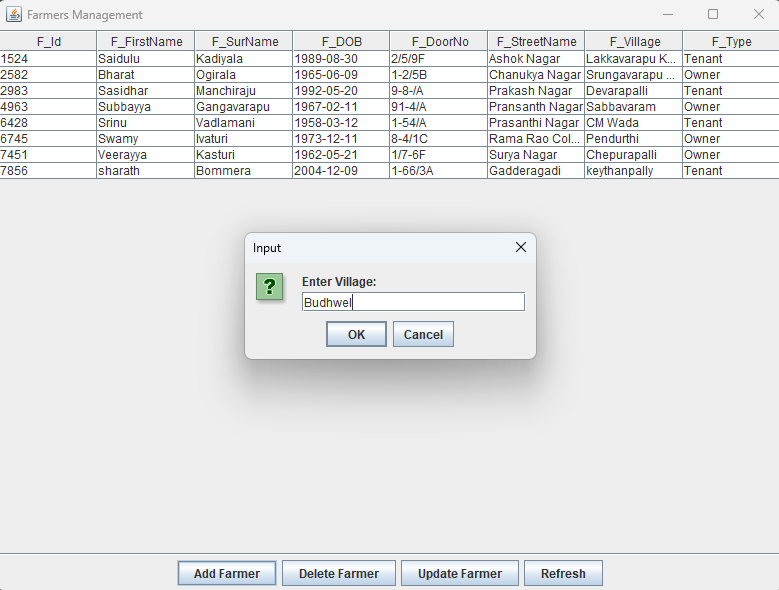


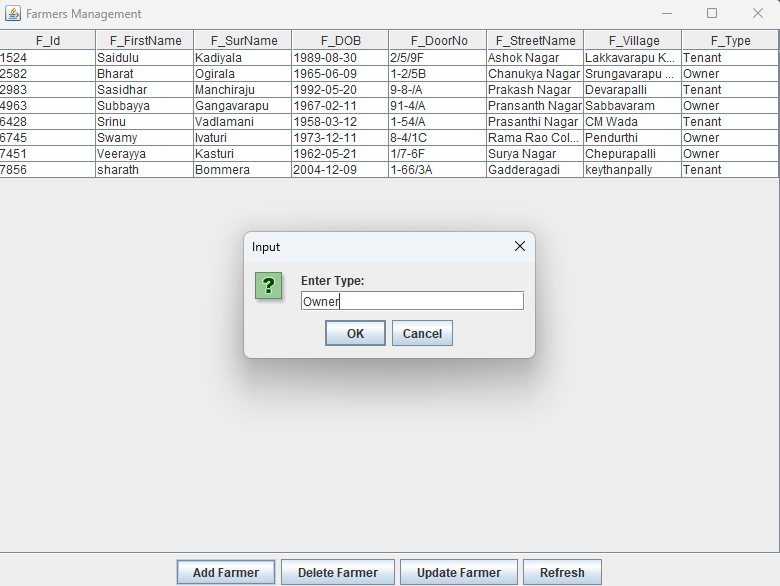




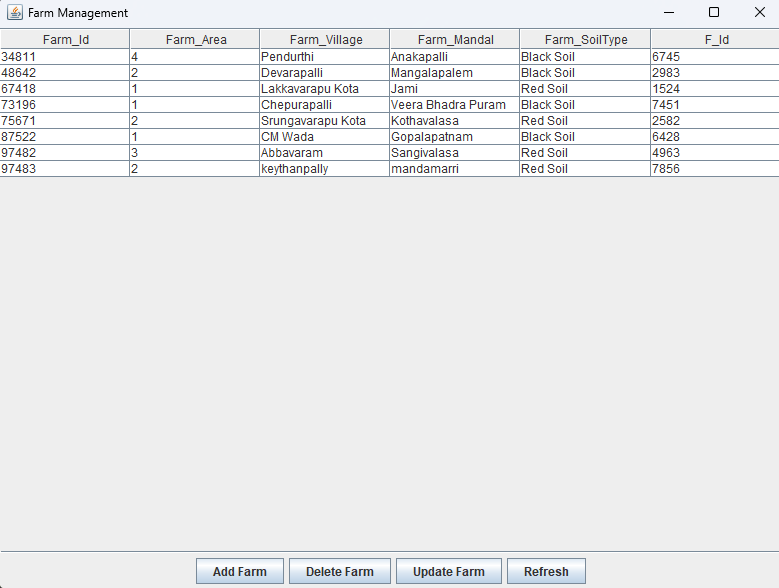




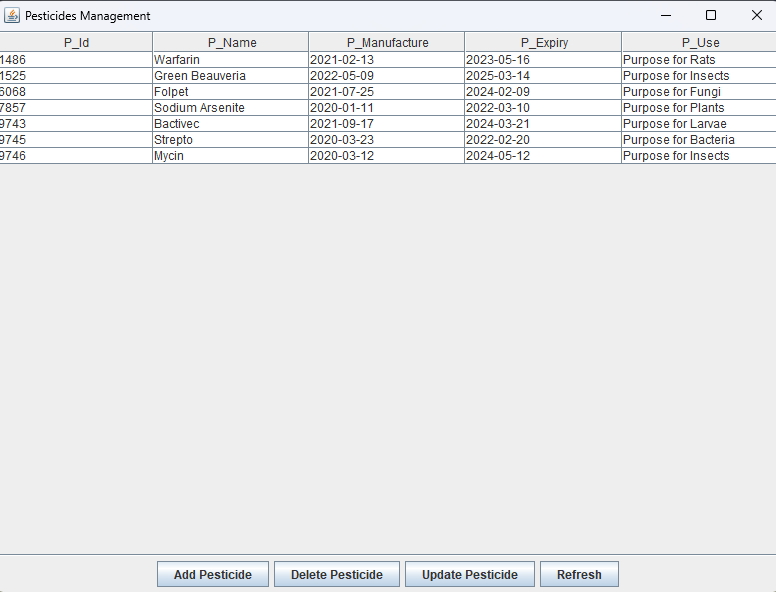




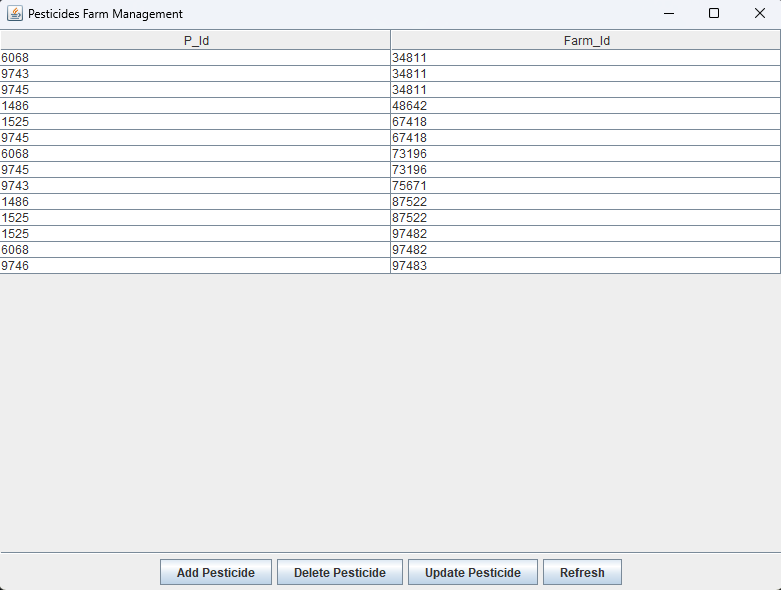
**Farm;**



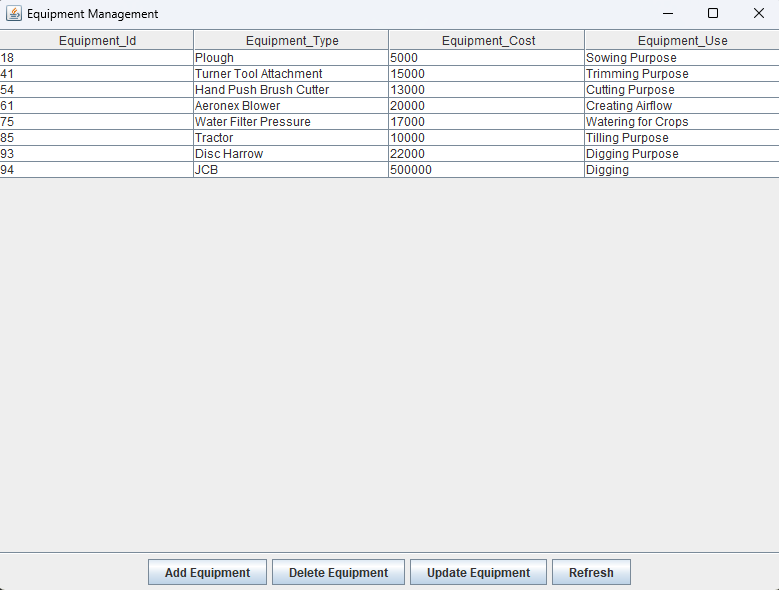
**Pesticides:**



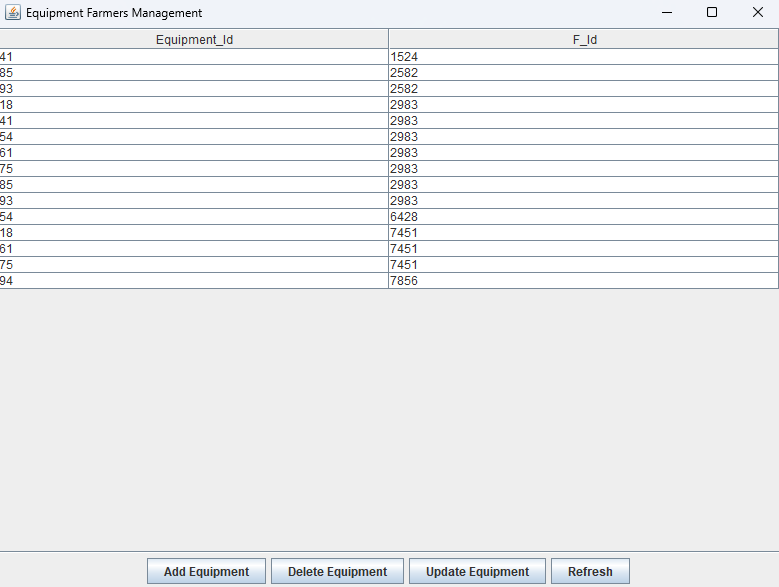
**Pesticides\_Farm:**



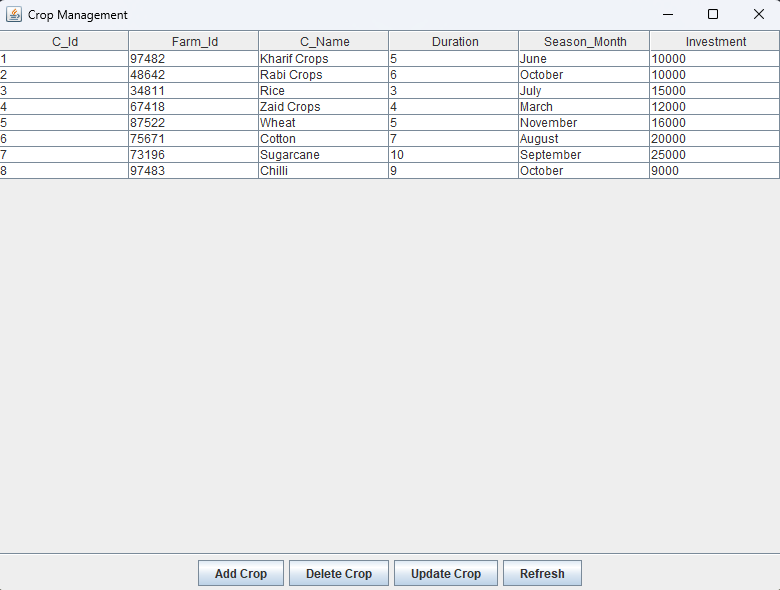
**Equipment:**



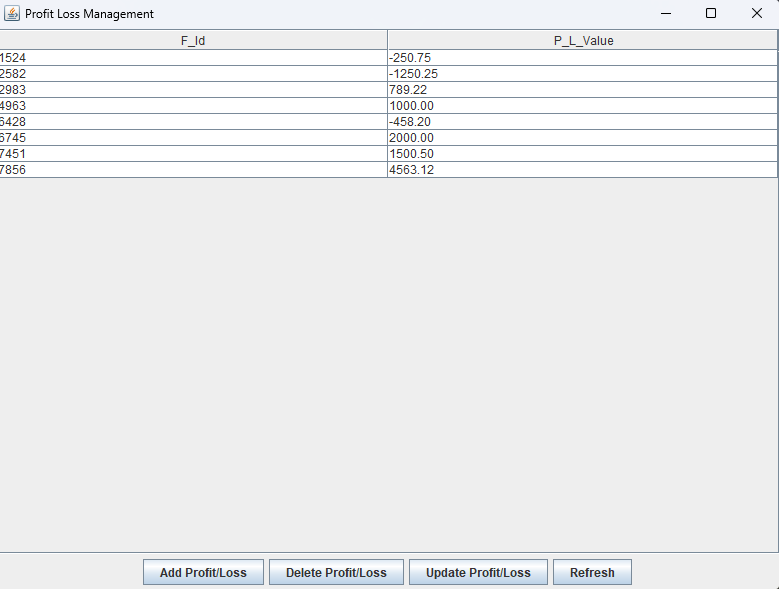
**Equipment\_Farmers:**



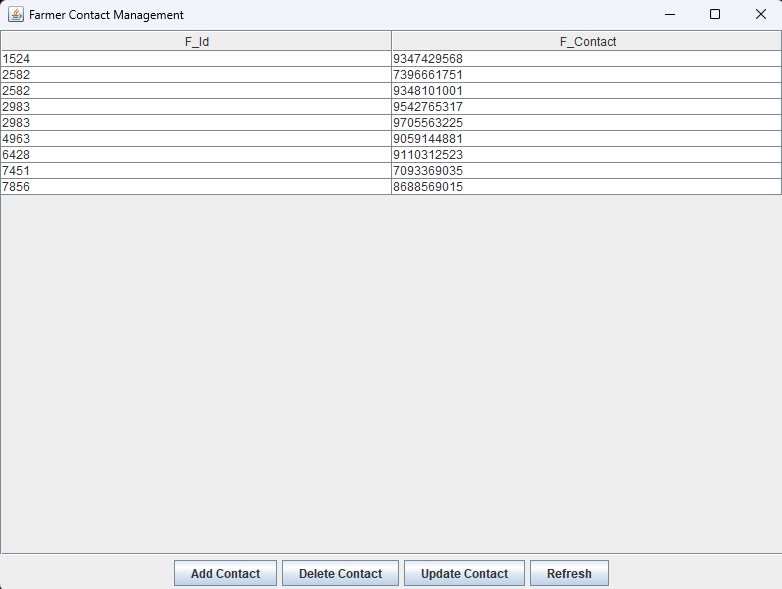
**Crop:**



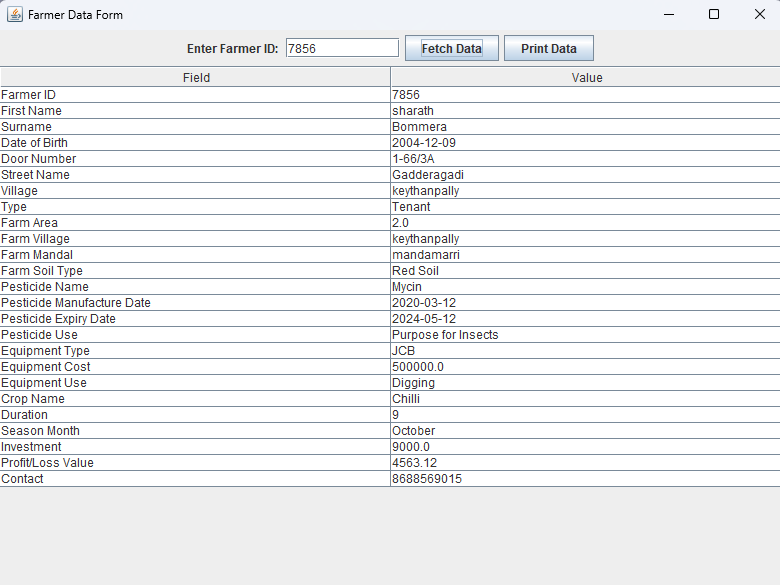
**Profit\_Loss:**

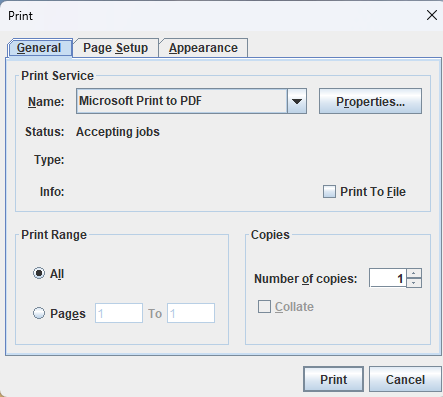


**Contacts:**



**GET FARMER’S DATA:**





# RESULTS

I have successfully completed the mini project “**CROP DATA MANAGEMENT SYSTEM**”

# DISCUSSION AND FUTURE WORK

In the future, the Crop Data Management System (CDMS) could see improvements in various areas. Enhancements might include a more user-friendly interface for easier navigation and customization options, empowering users to tailor their experience. Integrating data analysis tools and predictive analytics could offer valuable insights into crop trends and yield forecasts. Mobile app development could extend accessibility, while IoT integration could automate data collection from sensors, enhancing real-time monitoring capabilities. Strengthening security measures and optimizing performance for scalability would ensure data integrity and smooth operation. Moreover, compliance with agricultural standards and ongoing user support would guarantee the CDMS's reliability and usefulness in modern agriculture..

# REFERENCES

* <https://www3.ntu.edu.sg/home/ehchua/programming/java/JDBC_Basic.html>
* <https://www3.ntu.edu.sg/home/ehchua/programming/sql/MySQL_HowTo.html/>
* <https://www.mysql.com/>