JAVA AWT BASED- SMART FARMING SYSTEMATIC APPROACH DATA MANAGEMENT SYSTEM- SQL CONNECTIVITY USING JDBC

A report submitted in partial fulfillment of the Requirements for the award of the degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By: **A.SUSHMITHA<1602-18-737-108>**

UNDER THE GUIDANCE OF B.LEELAVATHY

2020

Department of Information Technology

VASAVI COLLEGE OF ENGINEERING AUTONOMOUS (AFFLIATED TO O.U)
IBRAHIMBAGH, HYDERABAD-500031

BONAFIDE CERTIFICATE

This to Certify that the project report titled ” SMART FARMING SYSTEMATIC APPROACH DATA MANAGEMENT SYSTEM ” project work of Miss. A.SUSHMITHA bearing Roll no: 1602-18-737-108 who carried out this project under my supervision in the IV Semester the academic year 2019-2020.

Signature . **Signature.**

External Examiner *Internal Examiner*

ABSTRACT:

This project " Smart Farming systematic approach data management system" where it reduces the ecological footprint of farming. Minimized or site-specific application of inputs, such as fertilizers and pesticides, in precision agriculture systems will mitigate leaching problems as well as the emission of greenhouse gases. With current ICT, it is possible to create a sensor network allowing almost continuous monitoring of the farm.

Smart farming can make agriculture more profitable for the farmer. Decreasing resources input will save the farmer money and lobour, and increased reliability of spatially explicit data.

a).AIM AND PRIORITY OF THE PROJECT:

To create a GUI based form for the project of **SMART FARMING SYSTEMATIC APPROACH DATA MANAGEMENT SYSTEM**

where in a smart farming checks a temperature, weather ,soil report and the farming are done by the automatic machines and reports are generated by the computers.

The values entered (insertion, updation, deletion)by the user for respective table in **GUI** should be updated in the database using **JDBC.**

b) **REQUIREMENTS ANALYSIS**

List of tables:

- Farmer
- Uses
- Auto_robotic
- Monitor

TABLE NAME	DESCRIPTION	ATTRTIBUTE	DATATYPE
Farmer	Farmer id Farmer name Farmer phone	Fid Fname fphone	Number(10) Varchar2(20) Number(11)
Uses	Farmer id Tractor Watering Seeds	Fid Tractor Auto_watering seeds	Number(10) Varchar2(20) Varchar2(20) Varchar2(30)
Auto_robotic	Auto seeding Auto Weeding	Seeding Weeding	Varchar2(30) Vaarchar2(50)
Monitor	Gid Seed results Temperature	Gid seed_results Temp	Number(10) Varchar2(30) Varchar2(30)

c)Architecture and Technology used

Java Eclipse, Oracle 11g Database, java SE version 8, SQL *plus ,java AWT

Eclipse: It is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug in system for customizing the environment. The Eclipse software

development kit (SDK), which include java development tools is meant for java developers.

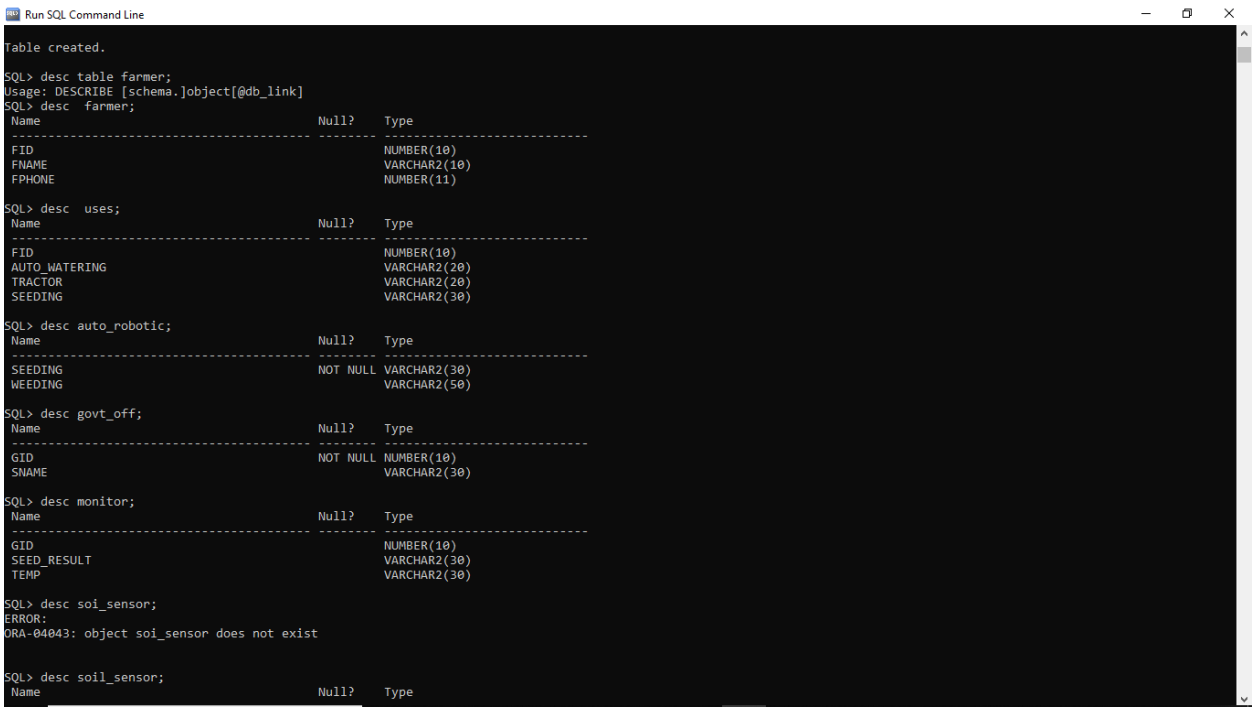
SQL *plus: SQL *plus is a command line tool proprietary to oracle. You can send SQL Queries to the server using the tool.It can also help you format the result of query. SQL is the query language that is used to communicate with the oracle server to access and modify data.

JAVA AWT: Abstract window tool kit is an API to develop GUI or Window based applications in java. Java AWT components are platform dependent i.e components are displayed according to the view of operating system. AWT is heavy weight that is components are using the resources of O.S.

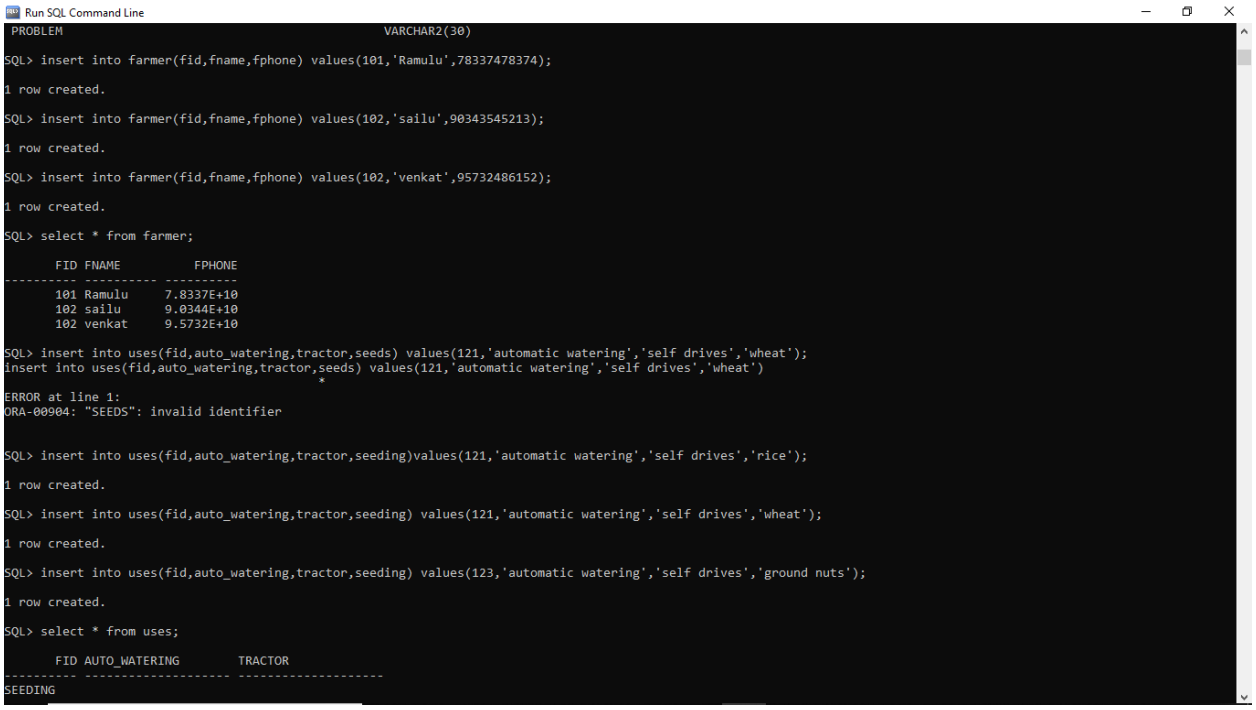
JDBC: Java Database Connectivity is an application programming interface (API) for the programming language java **,**which defines how a client may access database. It is a javabased data access technology used.

d) Data Base Design

Title : SMART FARMING SYSTEMATIC APPROACH DATA MANAGEMENT SYSTEM.



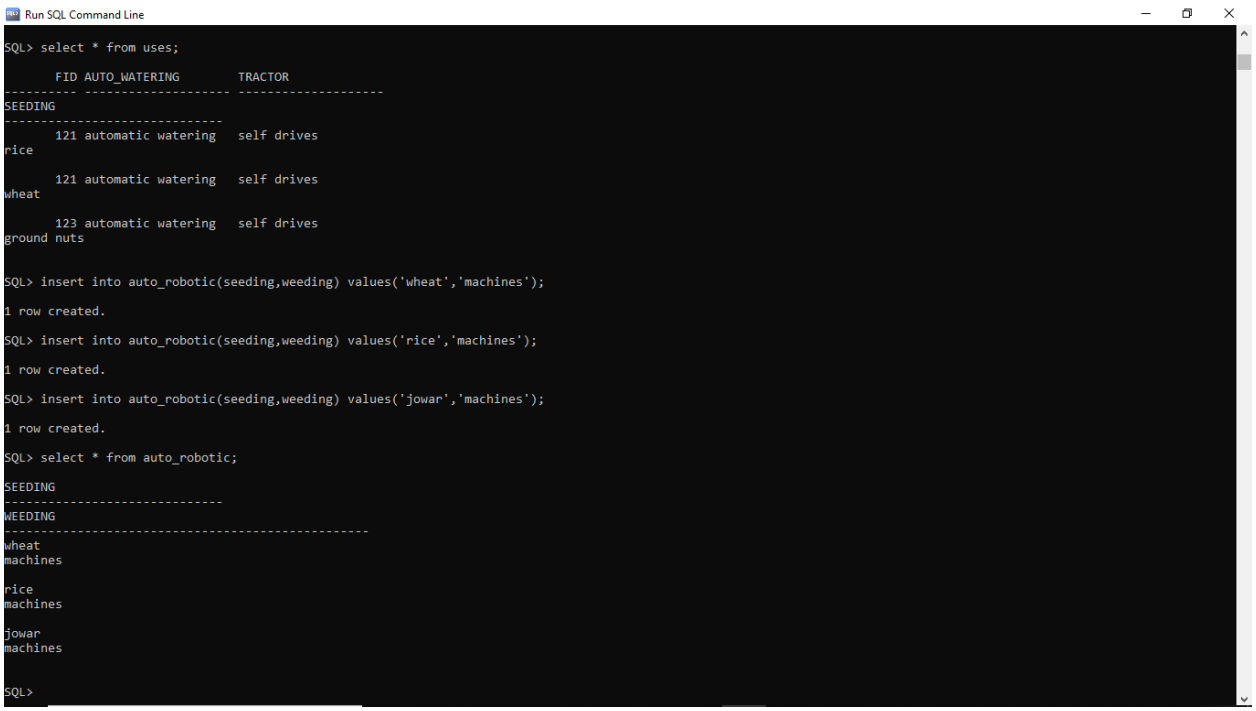
```
Run SQL Command Line

Table created.

SQL> desc table farmer;
Usage: DESCRIBE [schema.]object[@db_link]
SQL> desc  farmer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FID                                                NUMBER(10)
 FNAME                                              VARCHAR2(10)
 FPHONE                                             NUMBER(11)

SQL> desc  uses;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FID                                                NUMBER(10)
 AUTO_WATERING                                      VARCHAR2(20)
 TRACTOR                                            VARCHAR2(20)
 SEEDING                                            VARCHAR2(30)

SQL> desc auto_robotic;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SEEDING                                   NOT NULL VARCHAR2(30)
 WEEDING                                            VARCHAR2(50)

SQL> desc govt_off;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 GID                                       NOT NULL NUMBER(10)
 SNAME                                              VARCHAR2(30)

SQL> desc monitor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 GID                                                NUMBER(10)
 SEED_RESULT                                        VARCHAR2(30)
 TEMP                                               VARCHAR2(30)

SQL> desc soi_sensor;
ERROR:
ORA-04043: object soi_sensor does not exist


SQL> desc soil_sensor;
 Name                                      Null?    Type
```

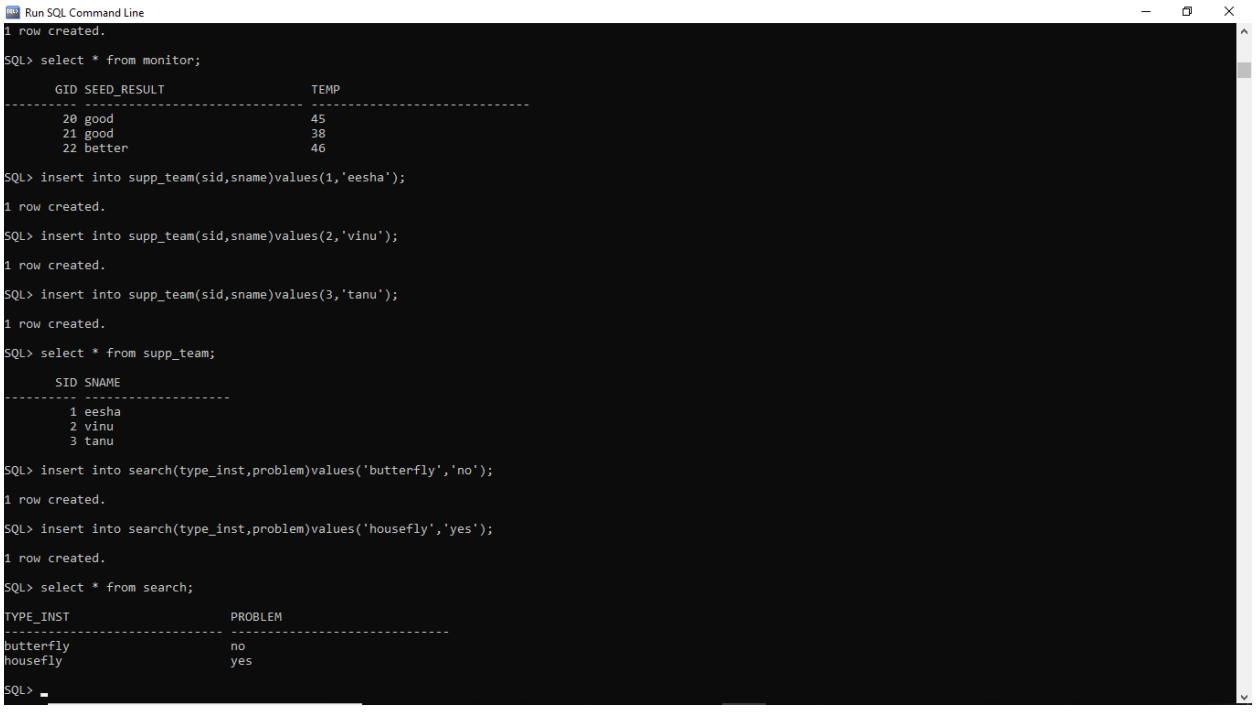


```
Run SQL Command Line

 PROBLEM                                            VARCHAR2(30)

SQL> insert into farmer(fid,fname,fphone) values(101,'Ramulu',78337478374);

1 row created.

SQL> insert into farmer(fid,fname,fphone) values(102,'sailu',90343545213);

1 row created.

SQL> insert into farmer(fid,fname,fphone) values(102,'venkat',95732486152);

1 row created.

SQL> select * from farmer;

       FID FNAME          FPHONE
---------- ---------- ----------
       101 Ramulu     7.8337E+10
       102 sailu      9.0344E+10
       102 venkat     9.5732E+10

SQL> insert into uses(fid,auto_watering,tractor,seeds) values(121,'automatic watering','self drives','wheat');
insert into uses(fid,auto_watering,tractor,seeds) values(121,'automatic watering','self drives','wheat')
                                                  *
ERROR at line 1:
ORA-00904: "SEEDS": invalid identifier


SQL> insert into uses(fid,auto_watering,tractor,seeding)values(121,'automatic watering','self drives','rice');

1 row created.

SQL> insert into uses(fid,auto_watering,tractor,seeding) values(121,'automatic watering','self drives','wheat');

1 row created.

SQL> insert into uses(fid,auto_watering,tractor,seeding) values(123,'automatic watering','self drives','ground nuts');

1 row created.

SQL> select * from uses;

       FID AUTO_WATERING        TRACTOR
---------- -------------------- --------------------
SEEDING
```

A.SUSHMITHA
ROLL NO : 1602-18-737-108

Title : SMART FARMING SYSTEMATIC APPROACH DATA MANAGEMENT SYSTEM.

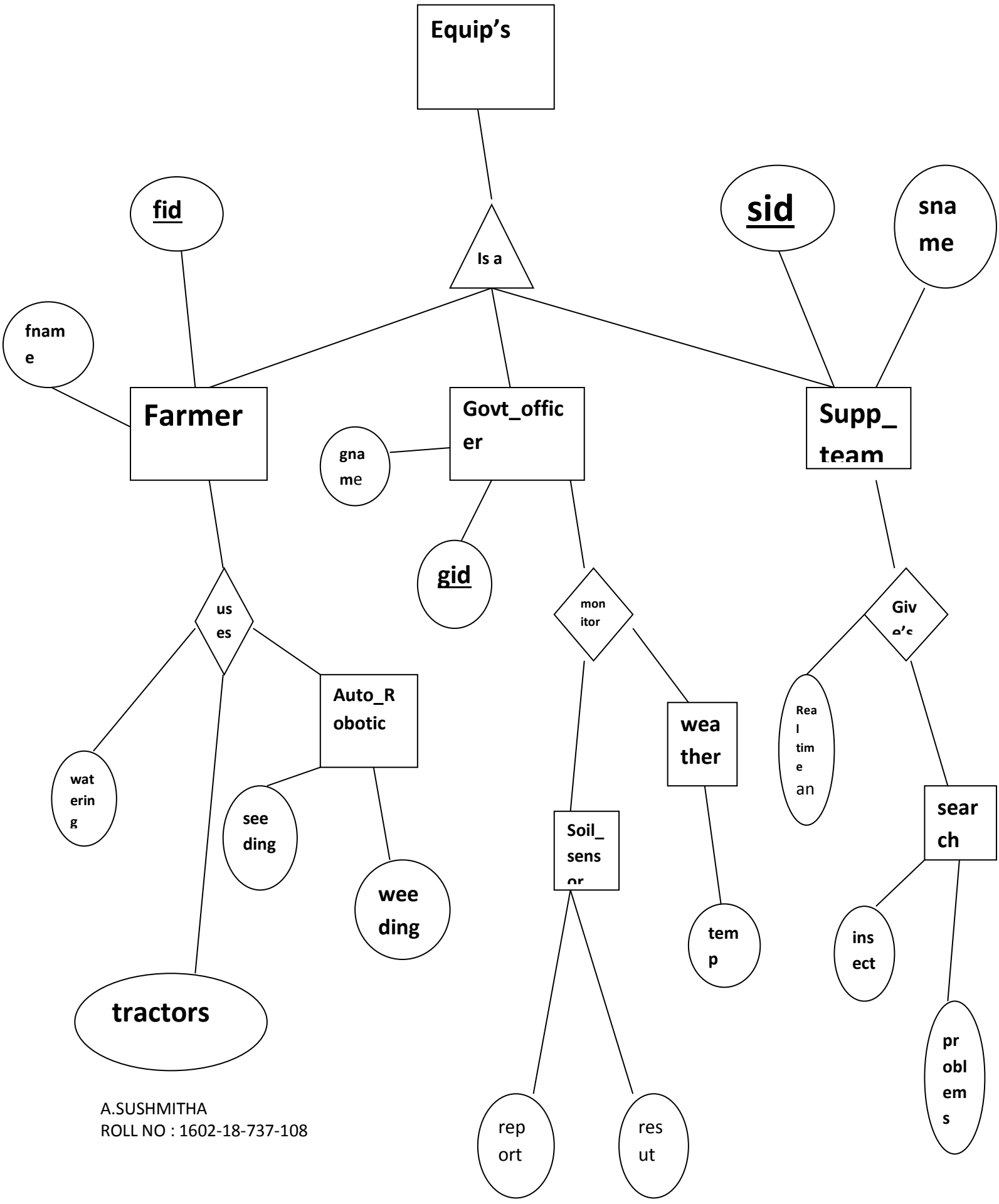




Entity Relationship Diagram

A.SUSHMITHA
ROLL NO : 1602-18-737-108

Title : SMART FARMING SYSTEMATIC APPROACH DATA MANAGEMENT SYSTEM.



A.SUSHMITHA
ROLL NO : 1602-18-737-108

e) Implementation

i) Front end programs and its connectivity.

```
public void connectToDB()
{
        try
        {
          connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","sushmitha","vasavi");
          statement = connection.createStatement();
          statement.executeUpdate("commit");

        }
        catch (SQLException connectException)
        {
          System.out.println(connectException.getMessage());
          System.out.println(connectException.getSQLState());
          System.out.println(connectException.getErrorCode());
          System.exit(1);
        }
}
```

Here,the connection from Java to Oracle database is performed and therefore, can be used for inserting, updating and deleting tables in the database directly.

Table Created in SQL for above mentioned purpose is as:

Create table farmer(

fid number(10) primary key, fname varchar(20),fphone number(11));

PROGRAM:

1)AddFarmer

```
package Farmer;

import java.awt.Button;

import java.awt.GridLayout;

import java.awt.Label;

import java.awt.Panel;

import java.awt.TextArea;

import java.awt.TextField;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import  java.sql.Statement;


public class AddFarmer extends Panel

{

        private static final long serialVersionUID = 1L;

        Button AddFarmerButton;

        TextField FID,FNAME,FPHONE;

        TextArea errorText;

        Connection connection;

        Statement statement;
```

```
	public AddFarmer()
	{
		try
		{
			Class.forName("oracle.jdbc.driver.OracleDriver");
		}
		catch (Exception e)
		{
			System.err.println("Unable to find and load driver");
			System.exit(1);
		}
		connectToDB();

	}

	public void connectToDB()
  {
		try
		{
		  connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","sushmitha","vasavi");
		  statement = connection.createStatement();
		  statement.executeUpdate("commit");

		}
		catch (SQLException connectException)
		{
```

```
                System.out.println(connectException.getMessage());
                System.out.println(connectException.getSQLState());
                System.out.println(connectException.getErrorCode());
                System.exit(1);
            }
    }
    public void buildGUI()
    {
        //Handle Insert Account Button
        AddFarmerButton = new Button("Add Address");
        AddFarmerButton.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                try
                {
                  String query= "INSERT INTO FARMER VALUES(" +
FID.getText() + ", " + FNAME.getText() + ",'" + FPHONE.getText() +")";
                  int i = statement.executeUpdate(query);
                  statement.executeUpdate("commit");
                  errorText.append("\nInserted " + i + " rows successfully");
                }
                catch (SQLException insertException)
                {
                  displaySQLErrors(insertException);
                }
            }
```

```
});
FID=new TextField(10);
FNAME= new TextField(20);
FPHONE = new TextField(11);
errorText = new TextArea(10, 40);
errorText.setEditable(false);

Panel first = new Panel();
first.setLayout(new GridLayout(6, 2));
first.add(new Label("FID:"));
first.add(FID);
first.add(new Label("FNAME"));
first.add(FNAME);
first.add(new Label("FPHONE :"));
first.add(FPHONE);
first.setBounds(125,80,200,140);

Panel second = new Panel(new GridLayout(4, 1));
second.add(AddFarmerButton);
second.setBounds(125,220,150,100);
Panel third = new Panel();
third.add(errorText);
third.setBounds(125,320,300,200);

setLayout(null);
```

```
            add(first);
            add(second);
            add(third);
            setSize(500, 600);
            setVisible(true);
            System.out.println("Hai");
        }



        private void displaySQLErrors(SQLException e)
        {
            errorText.append("\nSQLException: " + e.getMessage() + "\n");
            errorText.append("SQLState:     " + e.getSQLState() + "\n");
            errorText.append("VendorError:  " + e.getErrorCode() + "\n");
        }
}
```

2)DeleteFarmer

```
package Farmer;
import java.awt.Button;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.List;
import java.awt.Panel;
import java.awt.TextArea;
```

```
import java.awt.TextField;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;


public class DeleteFarmer extends Panel

{

        private static final long serialVersionUID = 1L;

        Button deleteFarmerButton;

        List FarmerIDList=null;

        TextField FID,FNAME,FPHONE;

        TextArea errorText;

        Connection connection;

        Statement statement;

        ResultSet rs;


        public DeleteFarmer()

        {

                try
```

```
                {
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                }
                catch (Exception e)
                {
                        System.err.println("Unable to find and load driver");
                        System.exit(1);
                }
                connectToDB();
        }

        public void connectToDB()
  {
                try
                {
                 connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","sushmitha","vasavi");
                 statement = connection.createStatement();

                }
                catch (SQLException connectException)
                {
                 System.out.println(connectException.getMessage());
                 System.out.println(connectException.getSQLState());
                 System.out.println(connectException.getErrorCode());
                 System.exit(1);
```

```
                }
  }

        public void loadFarmer()
        {
                try
                {

                 FarmerIDList.removeAll();
                 rs = statement.executeQuery("SELECT * FROM farmer");
                 while (rs.next())
                 {
                        FarmerIDList.add(rs.getString("FID"));
                 }
                }
                catch (SQLException e)
                {
                 displaySQLErrors(e);
                }
        }

        public void buildGUI()
        {
           FarmerIDList = new List(10);
                loadFarmer();
```

```
add(FarmerIDList);
FarmerIDList.addItemListener(new ItemListener()
{
    public void itemStateChanged(ItemEvent e)
    {
        try
        {
            rs = statement.executeQuery("SELECT * FROM Farmer");
            while (rs.next())
            {
                if
(rs.getString("FID").equals(FarmerIDList.getSelectedItem()))
                break;
            }
            if (!rs.isAfterLast())
            {
                FID.setText(rs.getString("FID"));
                FNAME.setText(rs.getString("FNAME"));
                FPHONE.setText(rs.getString("FPHONE"));
            }
        }
        catch (SQLException selectException)
        {
            displaySQLErrors(selectException);
        }
    }
```

```
        });
        deleteFarmerButton = new Button("Delete Farmer");
        deleteFarmerButton.addActionListener(new ActionListener()
        {
                public void actionPerformed(ActionEvent e)
                {
                        try
                        {
                                Statement statement = connection.createStatement();
                                int i = statement.executeUpdate("DELETE FROM farmer
WHERE FID = "+ FarmerIDList.getSelectedItem());
                                errorText.append("\nDeleted " + i + " rows successfully");
                                FID.setText(null);
                                FNAME.setText(null);
                                FPHONE.setText(null);
                                statement.executeUpdate("commit");
                                loadFarmer();
                        }
                        catch (SQLException insertException)
                        {
                                displaySQLErrors(insertException);
                        }
                }
        });
        FID = new TextField(10);
        FNAME = new TextField(20);
```

```
FPHONE = new TextField(11);
errorText = new TextArea(10, 40);
errorText.setEditable(false);

Panel first = new Panel();
first.setLayout(new GridLayout(6, 2));
first.add(new Label("FID:"));
first.add(FID);
first.add(new Label("FNAME:"));
first.add(FNAME);
first.add(new Label("FPHONE"));
first.add(FPHONE);
Panel second = new Panel(new GridLayout(4, 1));
second.add(deleteFarmerButton);

Panel third = new Panel();
third.add(errorText);

add(first);
add(second);
add(third);

setSize(450, 600);
setLayout(new FlowLayout());
setVisible(true);
```

```
        }



        private void displaySQLErrors(SQLException e)
        {
                errorText.append("\nSQLException: " + e.getMessage() + "\n");
                errorText.append("SQLState:     " + e.getSQLState() + "\n");
                errorText.append("VendorError:  " + e.getErrorCode() + "\n");
        }

}
```

3)UpdateFarmer

```
package Farmer;
import java.awt.Button;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.List;
import java.awt.Panel;
import java.awt.TextArea;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;


public class UpdateFarmer extends Panel


{

	private static final long serialVersionUID = 1L;

	Button updateFarmerButton;

	List FarmerIDList;

	TextField FID,FNAME,FPHONE;

	TextArea errorText;

	Connection connection;

	Statement statement;

	ResultSet rs;


	public UpdateFarmer()

	{

		try

		{

			Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
                }
                catch (Exception e)
                {
                        System.err.println("Unable to find and load driver");
                        System.exit(1);
                }
                connectToDB();
        }

        public void connectToDB()
  {
                try
                {
                 connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","sushmitha","vasavi");
                 statement = connection.createStatement();

                }
                catch (SQLException connectException)
                {
                 System.out.println(connectException.getMessage());
                 System.out.println(connectException.getSQLState());
                 System.out.println(connectException.getErrorCode());
                 System.exit(1);
                }
  }
```

```
public void loadFarmer()
{
        try
        {
                FarmerIDList.removeAll();
         rs = statement.executeQuery("SELECT FID FROM Farmer");
         while (rs.next())
         {
                 FarmerIDList.add(rs.getString("FID"));
         }
        }
        catch (SQLException e)
        {
         displaySQLErrors(e);
        }
}

public void buildGUI()
{
        FarmerIDList = new List(10);
        loadFarmer();
        add(FarmerIDList);

        FarmerIDList.addItemListener(new ItemListener()
```

```
                {
                        public void itemStateChanged(ItemEvent e)
                        {
                                try
                                {
                                        rs = statement.executeQuery("SELECT * FROM Farmer where
FID ="+FarmerIDList.getSelectedItem());
                                        rs.next();
                                        FID.setText(rs.getString("FID"));
                                        FNAME.setText(rs.getString("FNAME"));
                                        FPHONE.setText(rs.getString("FPHONE"));
                                }
                                catch (SQLException selectException)
                                {
                                        displaySQLErrors(selectException);
                                }
                        }
                });



                updateFarmerButton = new Button("Update Farmer");
                updateFarmerButton.addActionListener(new ActionListener()
                {
                        public void actionPerformed(ActionEvent e)
                        {
                                try
```

```
                    {
                        Statement statement = connection.createStatement();
                        int i = statement.executeUpdate("UPDATE Farmer"
                        + " SET FID=" + FID.getText() + ", "
                        + " FNAME = " + FNAME.getText()
                        + "  WHERE FPHONE = " + FarmerIDList.getSelectedItem());
                        errorText.append("\nUpdated " + i + " rows successfully");
                        i = statement.executeUpdate("commit");

                        loadFarmer();
                    }
                    catch (SQLException insertException)
                    {
                        displaySQLErrors(insertException);
                    }
                }
            });
            FID = new TextField(15);
            FNAME = new TextField(15);
            FPHONE= new TextField(15);
            FPHONE.setEditable(false);

            errorText = new TextArea(10, 40);
            errorText.setEditable(false);
```

```
            Panel first = new Panel();
            first.setLayout(new GridLayout(6, 2));
            first.add(new Label("FID:"));
            first.add(FID);
            first.add(new Label("FNAME:"));
            first.add(FNAME);
            first.add(new Label("FPHONE:"));
            first.add(FPHONE);
            Panel second = new Panel(new GridLayout(4, 1));
            second.add(updateFarmerButton);

            Panel third = new Panel();
            third.add(errorText);

            add(first);
            add(second);
            add(third);

            setSize(500, 600);
            setLayout(new FlowLayout());
            setVisible(true);

        }

        private void displaySQLErrors(SQLException e)
```

```
        {
                errorText.append("\nSQLException: " + e.getMessage() + "\n");
                errorText.append("SQLState:     " + e.getSQLState() + "\n");
                errorText.append("VendorError:  " + e.getErrorCode() + "\n");
        }
}
```

Main program:

```
package Main;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.JOptionPane;
import Auto_Robotic.AddAuto_Robotic;
import Auto_Robotic.DeleteAuto_Robotic;
import Auto_Robotic.UpdateAuto_Robotic;
import Farmer.DeleteFarmer;
import Farmer.UpdateFarmer;
import Farmer.AddFarmer;
import Monitor.AddMonitor;
import Monitor.DeleteMonitor;
import Monitor.UpdateMonitor;
import Uses.AddUses;
import Uses.DeleteUses;
import Uses.UpdateUses;
```

```
class FrontendPage extends JFrame implements ActionListener

{

        private static final long serialVersionUID = 1L;

        String msg = "";

         Label ll;

         CardLayout cardLO;


        //Create Panels for each of the menu items, welcome screen panel and home screen panel
with CardLayout


         AddAuto_Robotic adA;

         DeleteAuto_Robotic delA;

         UpdateAuto_Robotic upA;

         DeleteFarmer delF;

         UpdateFarmer upF;

         AddFarmer adF;

         AddMonitor adM;

         DeleteMonitor delM;

         UpdateMonitor upM;

         AddUses adU;

         DeleteUses delU;

         UpdateUses upU;

         Panel home,welcome;


         public FrontendPage()

         {
```

```
cardLO = new CardLayout();


//Create an empty home panel and set its layout to card layout
home = new Panel();
home.setLayout(cardLO);



ll = new Label();
ll.setAlignment(Label.CENTER);
ll.setText("Welcome to Smart Farming systematic approach data management
system");


//Create welcome panel and add the label to it
welcome = new Panel();
welcome.add(ll);


//create panels for each of our menu items and build them with respective
components
adA=new AddAuto_Robotic();
adA.buildGUI();
upA = new UpdateAuto_Robotic();
upA.buildGUI();
delA = new DeleteAuto_Robotic();
delA.buildGUI();
upF = new UpdateFarmer();
upF.buildGUI();
```

```
delF=new DeleteFarmer();
delF.buildGUI();
adF=new AddFarmer();
adF.buildGUI();
delM=new DeleteMonitor();
delM.buildGUI();
upM=new UpdateMonitor();
upM.buildGUI();
adM=new AddMonitor();
adM.buildGUI();
adU=new AddUses();
adU.buildGUI();
delU=new DeleteUses();
delU.buildGUI();
upU=new UpdateUses();
upU.buildGUI();
home.add(welcome, "Welcome");
home.add(adA, "Add Auto_Robotic");
home.add(upA, "Update Auto_Robotic");
home.add(delA, "Delete Auto_Robotic");
home.add(adF, "Add Farmer");
home.add(upF,"Update Farmer");
home.add(delF,"Delete Farmer");
home.add(adM,"Add Monitor");
home.add(delM,"Delete  Monitor");
```

```
home.add(upM,"Update  Monitor");
home.add(adU,"Add Uses");
home.add(delU,"Delete Uses");
home.add(upU,"Update Uses");
// add home panel to main frame
add(home);

// create menu bar and add it to frame
MenuBar mbar = new MenuBar();
setMenuBar(mbar);

// create the menu items and add it to Menu
Menu Auto_Robotic = new Menu("Auto_Robotic ");
MenuItem item1, item2, item3;
Auto_Robotic.add(item1 = new MenuItem("Add Auto_Robotic"));
Auto_Robotic.add(item2 = new MenuItem("View Auto_Robotic"));
Auto_Robotic.add(item3 = new MenuItem("Delete Auto_Robotic"));
mbar.add(Auto_Robotic);

Menu Farmer = new Menu("Farmer");
MenuItem item4, item5, item6;
Farmer.add(item4 = new MenuItem("Add Farmer"));
Farmer.add(item5 = new MenuItem("View Farmer"));
Farmer.add(item6 = new MenuItem("Delete Farmer"));
mbar.add(Farmer);
```

```
Menu Monitor= new Menu("Monitor");
MenuItem item7, item8, item9;
Monitor.add(item7 = new MenuItem("Add Monitor"));
Monitor.add(item8 = new MenuItem("View Monitor"));
Monitor.add(item9 = new MenuItem("Delete Monitor"));
mbar.add(Monitor);


Menu Uses= new Menu("Uses");
MenuItem item10, item11, item12;
Uses.add(item10 = new MenuItem("Add Uses"));
Uses.add(item11 = new MenuItem("View Uses"));
Uses.add(item12 = new MenuItem("Delete Uses"));
mbar.add(Uses);


// register listeners
item1.addActionListener(this);
item2.addActionListener(this);
item3.addActionListener(this);
item4.addActionListener(this);
item5.addActionListener(this);
item6.addActionListener(this);
item7.addActionListener(this);
```

```
                item8.addActionListener(this);
                item9.addActionListener(this);
                item10.addActionListener(this);
                item11.addActionListener(this);
                item12.addActionListener(this);
                addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent we)
                        {
                                quitApp();
                        }
                });

                //Frame properties
                setTitle("Smart Farming systematic approach data management system");
                setSize(500, 600);
                setVisible(true);

        }

        public void actionPerformed(ActionEvent ae)
        {
                String arg = ae.getActionCommand();
                if(arg.equals("Add Auto_Robotic"))
                {
                        cardLO.show(home, "Add Auto_Robotic");
```

```
    }

            else if(arg.equals("View Auto_Robotic"))
            {
                cardLO.show(home, "Update Auto_Robotic");
                upA.loadAuto_Robotic();
            }

            else if(arg.equals("Delete Auto_Robotic"))
            {
                cardLO.show(home, "DeleteAuto_Robotic");
                delA.loadAuto_Robotic();
            }
            else if(arg.equals("Add Farmer"))
            {
                cardLO.show(home, "Add Farmer");
            }
            else if(arg.equals("View Farmer"))
            {
                cardLO.show(home, "Update Farmer");
                upF.loadFarmer();
            }
            else if(arg.equals("Delete Farmer"))
            {
                cardLO.show(home, "Delete Farmer");
```

```
            delF.loadFarmer();
        }
        else if(arg.equals("Add Monitor"))
        {
            cardLO.show(home, "Add Monitor");
        }
        else if(arg.equals("Delete Monitor"))
        {
            cardLO.show(home, "Delete Monitor");
            delM.loadMonitor();
        }
        else if(arg.equals("View Monitor"))
        {
            cardLO.show(home, "Update Monitor");
            upM.loadMonitor();
        }

        else if(arg.equals("Add  Uses"))
        {
             cardLO.show(home,"Add Uses");
        }
        else if(arg.equals("Delete Uses"))
        {
             cardLO.show(home, "Delete Uses");
             delU.loadUses();
```

```
                }
                else if(arg.equals("View Uses"))
                {
                        cardLO.show(home, "Update Uses");
                        upU.loadUses();
                }

        }
        private void quitApp () {

                try {
                        int reply = JOptionPane.showConfirmDialog (this,
                                        "Are you really want to exit \n from - Smart Farming
Systematic approach database management system ?",
                                        "Smart Farming Systematic Approach Database
management system - Exit", JOptionPane.YES_NO_OPTION, JOptionPane.PLAIN_MESSAGE);
                        if (reply == JOptionPane.YES_OPTION) {
                                setVisible (false);
        dispose();
                                System.out.println ("Thanks :)");
                                System.exit (0);
                        }
                        else if (reply == JOptionPane.NO_OPTION) {
                                setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
                        }
                }
```

```
                    catch (Exception e) {}


            }
        public static void main(String ... args)
        {
                    new FrontendPage();
        }


}
```

ii) GitHub Link and folder structure:

a) Link- https://github.com/AdaboinaSushmitha/Smart-Farming

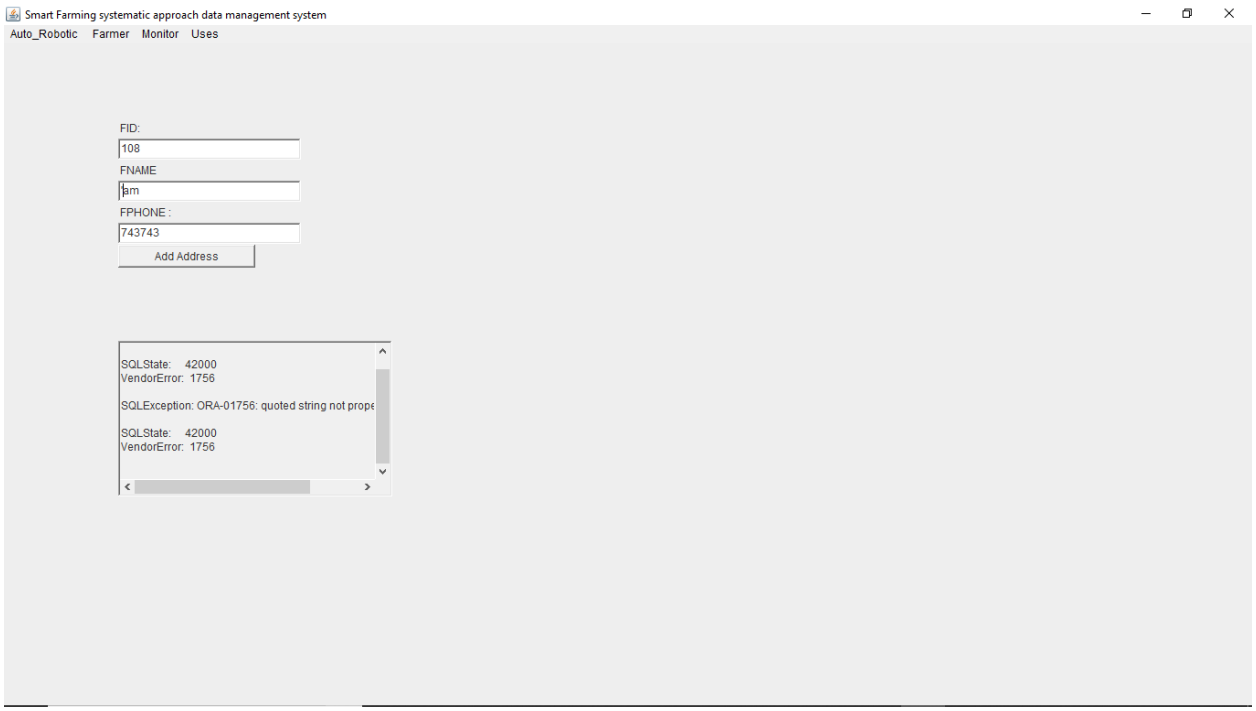
b) Folder Structure:

1. Data Base Design
2. Mini Project Report
3. Front End Programs (JAVA CODE)

Testing:

If incorrect values are entered which mismatch data types it won't allow to insert.

Title : SMART FARMING SYSTEMATIC APPROACH DATA MANAGEMENT SYSTEM.

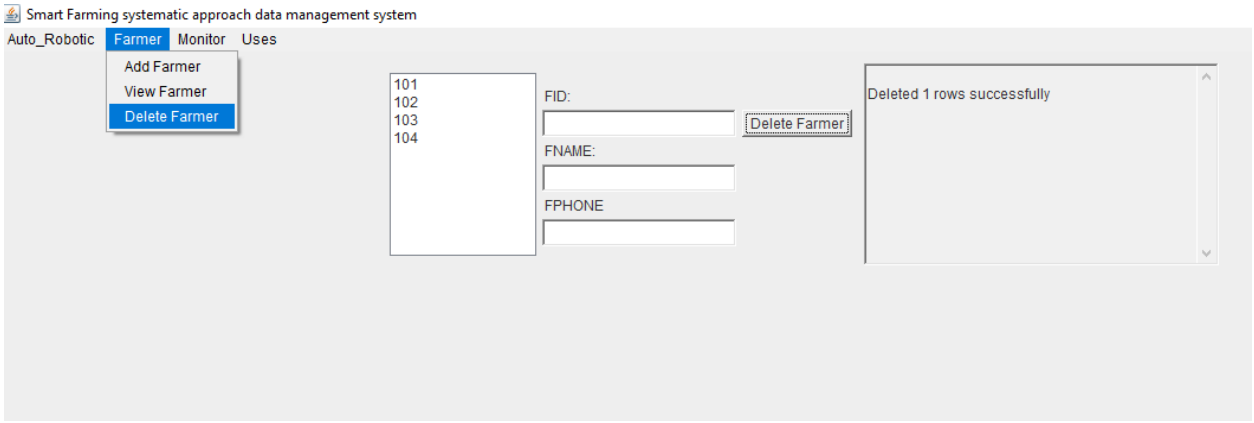


RESULT:

The process of entering information into the frame created by java code so that the data is reflected in the database using **JDBC connectivity** is done successfully.

A.SUSHMITHA
ROLL NO : 1602-18-737-108

Title : SMART FARMING SYSTEMATIC APPROACH DATA MANAGEMENT SYSTEM.

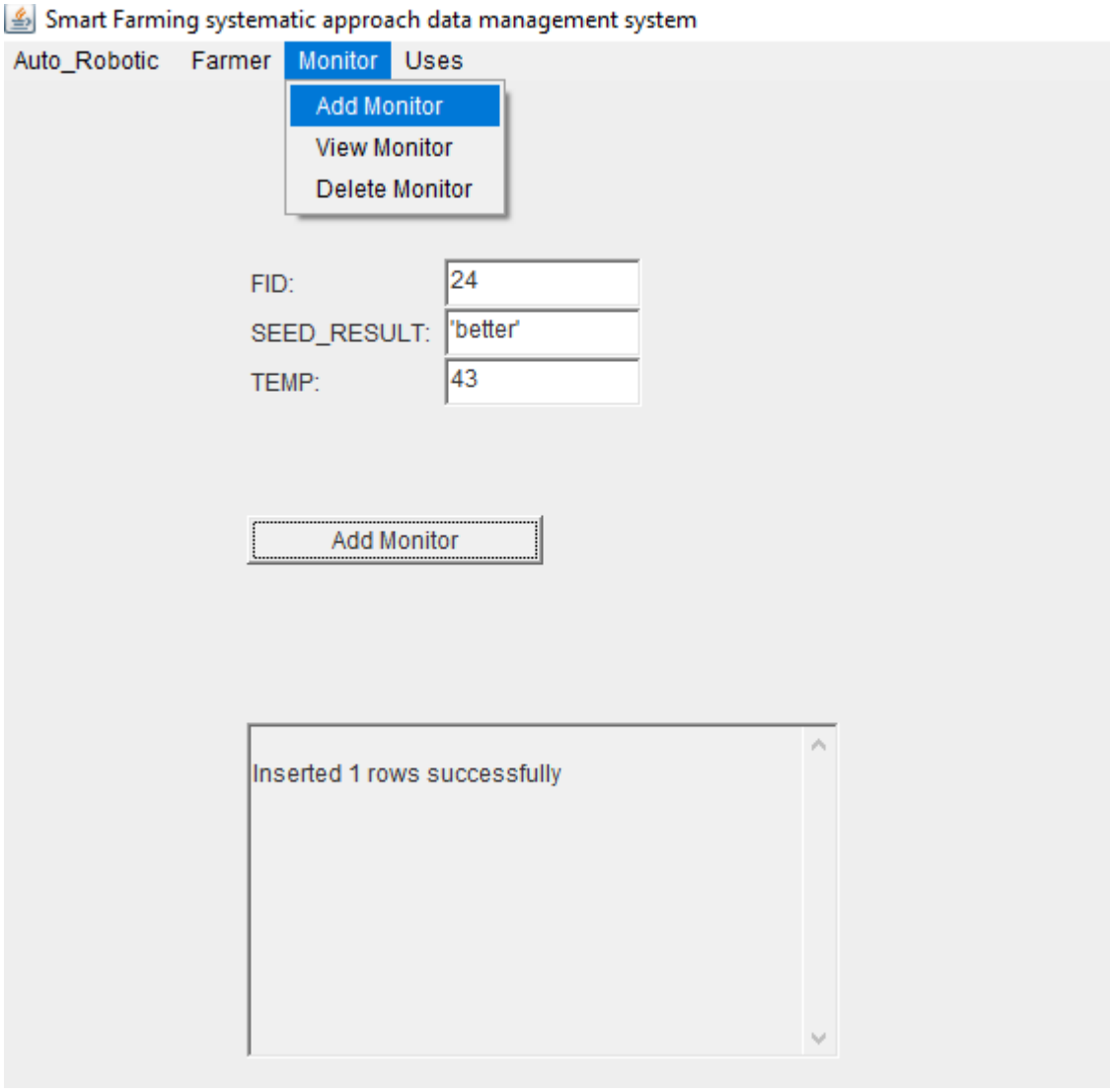


Run SQL Command Line

```
QL*Plus: Release 11.2.0.2.0 Production on Fri Apr 10 20:23:06 2020

opyright (c) 1982, 2010, Oracle.  All rights reserved.

QL> conn sushmitha/vasavi;
onnected.
QL> desc farmer;
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
FID                                       NOT NULL NUMBER(10)
FNAME                                              VARCHAR2(20)
FPHONE                                             NUMBER(11)

QL> select * from farmer;

       FID FNAME                    FPHONE
---------- -------------------- ----------
       101 Ramulu               7.8337E+10
       102 Venkat               9.5732E+10
       103 ravi                   43546655
       104 narsing                34454656
       106 ali                    34343435

QL> select * from farmer;

       FID FNAME                    FPHONE
---------- -------------------- ----------
       101 Ramulu               7.8337E+10
       102 Venkat               9.5732E+10
       103 ravi                   43546655
       104 narsing                34454656

QL>
```

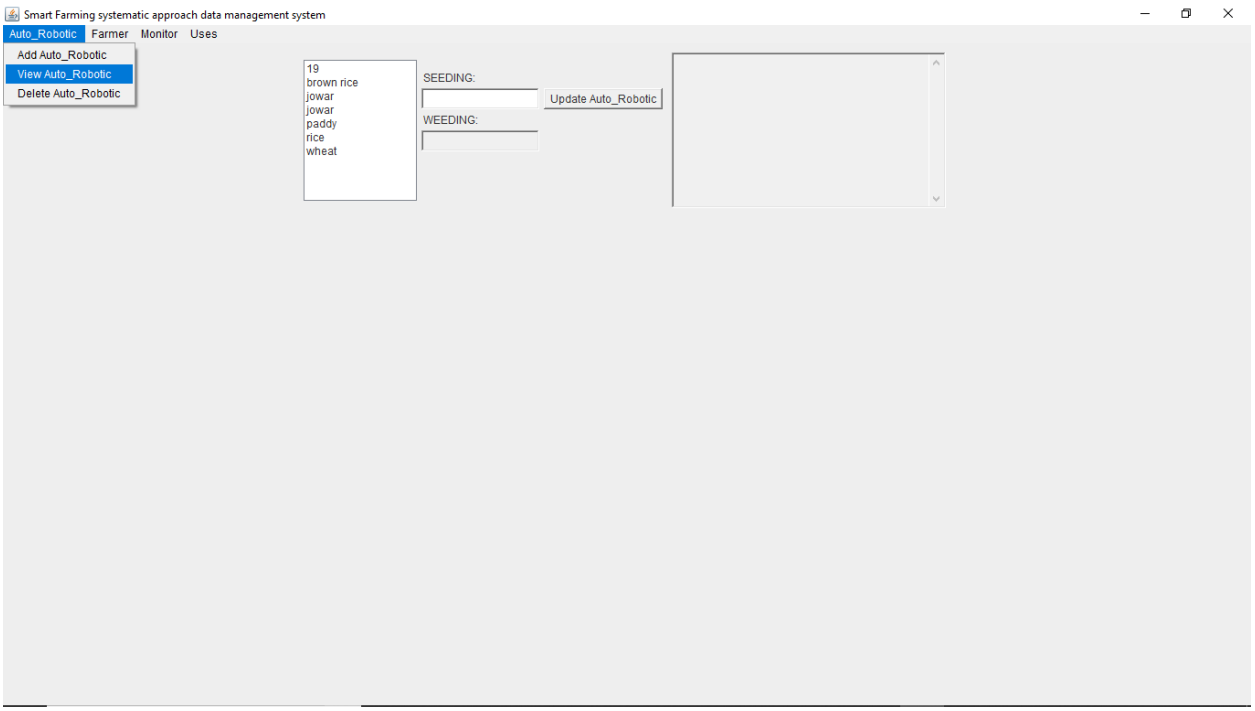
A.SUSHMITHA
ROLL NO : 1602-18-737-108

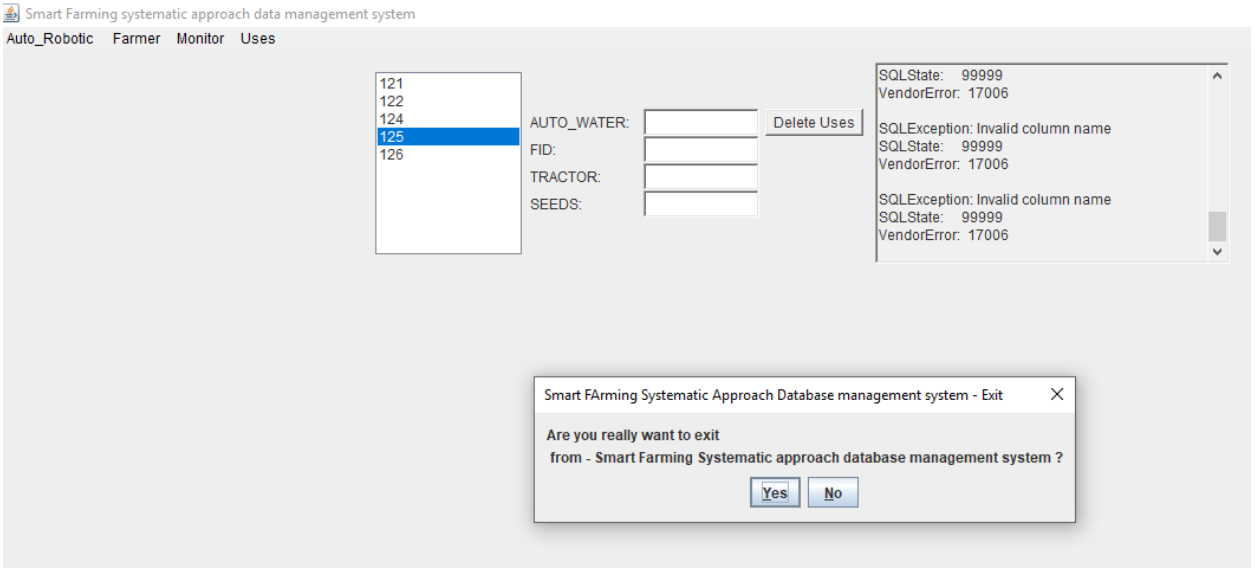
Title : SMART FARMING SYSTEMATIC APPROACH DATA MANAGEMENT SYSTEM.



A.SUSHMITHA
ROLL NO : 1602-18-737-108

Title : SMART FARMING SYSTEMATIC APPROACH DATA MANAGEMENT SYSTEM.





DISCUSSION & FUTURE WORK :

The application done till now is basically to store the details of Smart Farming like auto robotics like watering , seeding, feeding and weather like Temperature ,soil report all are stored without man power. The machines are take photo's and keep tack the growth of the plants

A.SUSHMITHA
ROLL NO : 1602-18-737-108

hence, for this number of entities and relationship between them will come into picture, which can be converted into first tables using SQL commands and then into GUI program using java code.

REFERENCES:

https://docs.oracle.com/javase/8/docs/api/

https://www.geeksforgeeks.org/establishing-jdbc-connection-in-java/