JAVA AWT BASED- REGISTRATION FORM- SQL CONNECTIVITY USING JDBC

A

Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

P Pradeep kumar reddy<1602-18-737-088>

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2019-2020

BONAFIDE CERTIFICATE

This is to certify that the project report titled

“ **FOOD AND NUTRITION SYSTEM”** project work of Mr. *P Pradeep kumar reddy* bearing roll no: 1602-18-737-088 who carried out this project under my supervision the IV semester for the academic year 2019-2020

Signature

External examine

Signature

B.leelavathy

Assistant professior

Abstract :

Inadequate and inappropriate intake of food is known to cause various health issues and diseases. Due to lack of concise information about healthy diet, people have to rely on medicines instead of taking preventive measures in food intake. Due to diversity in food components and large number of dietary sources, it is challenging to perform real-time selection of diet patterns that must fulfill one’s nutrition needs. Particularly, selection of proper diet is critical for patients suffering from various diseases.this system will intake the weight and height of the multiple users and we will also take the calorie and food intake and it will give the suggestions of food and it will give what to intake and gives the diet plan.

AIM:

To create a **Java GUI based health nutrition system** which takes the values like Name,gender,weight,daily food and excersice details and it will provide us the type and amount food we have to the user. These values are to be updated in the database using **JDBC connectivity .**

SOFTWARE USED:

Java Eclipse, Oracle 11g Database, Java SE version 7, SQL*Plus.

Java AWT:

Java AWT (Abstract Window Toolkit) 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 heavyweight i.e. its components are using the resources of OS.

The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

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.

Java-SQL Connectivity using JDBC:

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

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

```
private void connToDb(){
            try {
              Class.forName("oracle.jdbc.driver.OracleDriver");
              connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:
1522:xe","PRADEEP","v");
              statement = connection.createStatement();

            } catch (SQLException connectException) {
              System.out.println(connectException.getMessage());
              System.out.println(connectException.getSQLState());
              System.out.println(connectException.getErrorCode());
              System.exit(1);
            }
            catch (Exception e)
           {
                System.err.println("Unable to find and load driver");
                System.exit(1);
           }
```

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

Requirement analysis:

List of tables:

User_details User intake diet_plan calories_spent Gets Intake

List of attributes with their domain types:

user_details:

Name-varchar2( ); email_id-varchar2( ); Gender- varchar2( ); Weight-number( ) Height-number( ) Id-number( ) Day-number( )

user_intake:

Calorie-number( ) Id-number( ) food_quantity-number( ) food_type-varchar2( )

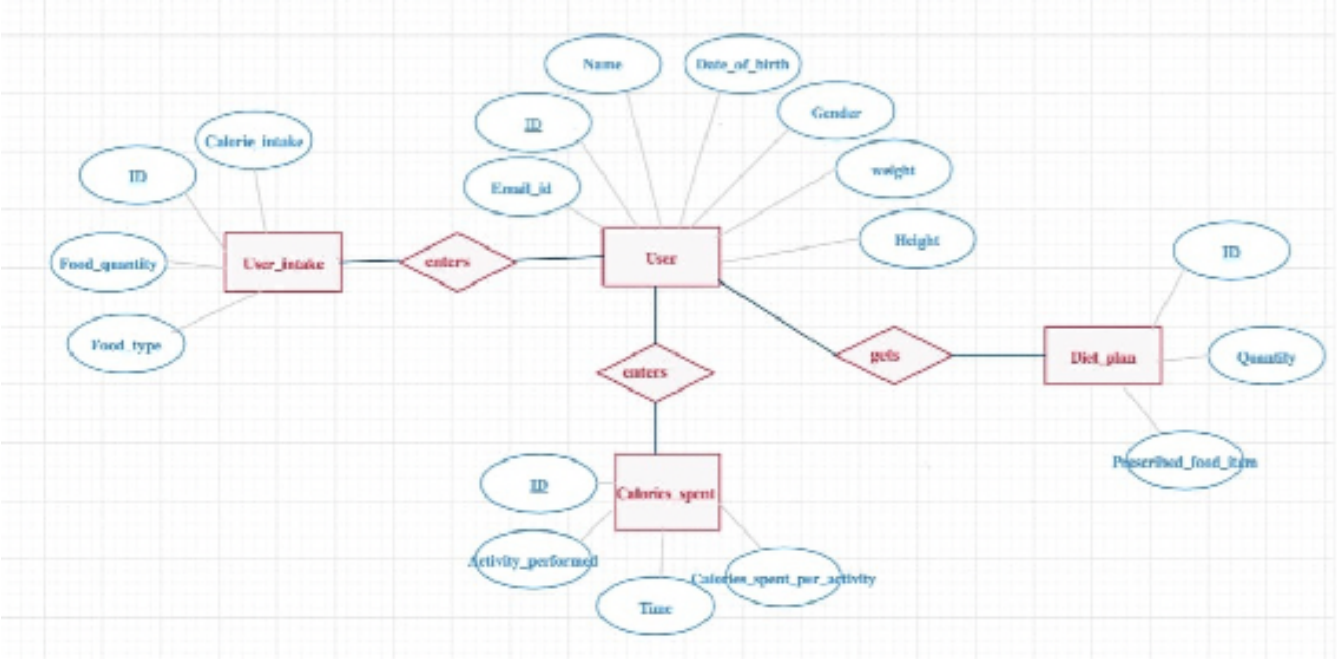
calories_spent: id-number();

Activity:

-varchar( ); time_number( ); calories_spent:-number( ); Intake:

Id- number( ) user_diet: id-number( )

E-R diagram:


Name
Date_of_birth
ID
Gender
Calorie intake
ID
weight
Email_id
Height
Food_quantity
User_intake
enters
User
ID
Food_type
gets
Diet_plan
Quantity
enters
Prescribed_food_item
ID
Calories_spent
Activity_performed
Calories_spent_per_activity
Time

DDL COMMANDS

SQL> create table user_details(2 name varchar(10),
3 email_id varchar(10),
4 gender varchar(10),

5 weight number(5,3),
6 height number(5,3),
7 id number(5) primary key, 8 day date);

Table created.

SQL> create table user_intake(2 calorie_intake number(5), 3 id number(5),
4 food_quantity number(5), 5 food_type varchar2(10),

6 foreign key(id) references user_details);

SQL> create table calories_spent(
2 id number(5),
3 activity_performed varchar2(20),
4 time number(5,3),
5 calories_spent_per_activity number(5), 6 foreign key(id) references user_details);

Table created.

```
SQL> create table diet_plan(
2 id number(5),
3 quantity number(5,3),
4 prescribed_food_item varchar2(10),
5 foreign key(id) references user_details);

Table created.

SQL> create table intake(
2 id number(5),
3 foreign key(id) references user_details);

Table created.

SQL> create table user_spent(
2 id number(5),
3 foreign key(id) references user_details);

Table created.

SQL> create table user_diet(
2 id number(5),
3 foreign key(id) references user_details);

Table created.

SQL> desc user_details;
Name Null? Type ----------------------------------------
-------- ----------------------------
```

NAME EMAIL_ID GENDER WEIGHT HEIGHT ID

DAY

VARCHAR2(10) VARCHAR2(10)

VARCHAR2(10) NUMBER(5,3)

NUMBER(5,3) NOT NULL NUMBER(5)

DATE

SQL> desc user_intake;

Name Null? Type ----------

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

CALORIE_INTAKE ID FOOD_QUANTITY
FOOD_TYPE

SQL> desc diet_plan;
Name

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

ID
QUANTITY PRESCRIBED_FOOD_ITEM

NUMBER(5) NUMBER(5,3)

VARCHAR2(10)

SQL> desc calories_spent;
Name Null? Type --
-------- ----------------------------

NUMBER(5) NUMBER(5)

NUMBER(5) VARCHAR2(10)

Null? Type

ID
ACTIVITY_PERFORMED
TIME
CALORIES_SPENT_PER_ACTIVITY NUMBER(5)

SQL> desc intake;

NUMBER(5) NUMBER(5,3)

VARCHAR2(20)

Name Null? Type --
-------- ---------------------------- ID NUMBER(5)

SQL> desc user_spent;
Name Null? Type --
-------- ---------------------------- ID NUMBER(5)

SQL> desc user_diet;
Name Null? Type --
-------- ---------------------------- ID NUMBER(5)

DML commands:

SQL> insert into user_details
values(‘&name’,’&email_id’,’&gender’,&weight,’&heigh
t’,&id,’&day’); Enter value for name:pradeep
Enter value for email_id:p@gmail.com
Enter value for gender:male
Enter value for weight:65
Enter value for height:5.7
Enter value for id:88
Enter value for day:08-jul-2000

1 row created

SQL> insert into user_details
values(‘&name’,’&email_id’,’&gender’,&weight,’&heigh
t’,&id,’&day’); Enter value for name:malli

Enter value for email_id:mgmail
Enter value for gender:male
Enter value for weight:70
Enter value for height:5.7
Enter value for id:96
Enter value for day:09-sep-2000

1 row created

SQL> insert into user_details
values(‘&name’,’&email_id’,’&gender’,&weight,’&heigh
t’,&id,’&day’); Enter value for name:ai
Enter value for email_id:saigmail
Enter value for gender:male
Enter value for weight:74
Enter value for height:5.4
Enter value for id:90
Enter value for day:08-jan-1999

1 row created

Select * from user_details;
SQL> alter table user_diet add(day date);

SQL> insert into intake values(&id,'&day');
Enter value for id: 88
Enter value for day: 12-feb-2020

old 1: insert into intake values(&id,'&day')
new 1: insert into intake values(88,'12-feb-2020')

1 row created.

SQL> /
Enter value for id: 96
Enter value for day: 12-feb-2020
old 1: insert into intake values(&id,'&day')
new 1: insert into intake values(96,'12-feb-2020')

1 row created.

SQL> /
Enter value for id: 90
Enter value for day: 12-feb-2020
old 1: insert into intake values(&id,'&day')
new 1: insert into intake values(90,'12-feb-2020')

1 row created.
SQL> select * from intake;

ID DAY ---------- ---------

88 12-FEB-20 96 12-FEB-20 90 12-FEB-20

SQL> insert into user_spent values(&id,'&day'); Enter value for id: 88
Enter value for day: 13-feb-2020

old 1: insert into user_spent values(&id,'&day')
new 1: insert into user_spent values(88,'13-feb-2020')

1 row created.

SQL> /
Enter value for id: 96
Enter value for day: 13-feb-2020
old 1: insert into user_spent values(&id,'&day')
new 1: insert into user_spent values(96,'13-feb-2020')

1 row created.

SQL> /
Enter value for id: 90
Enter value for day: 13-feb-2020
old 1: insert into user_spent values(&id,'&day')
new 1: insert into user_spent values(90,'13-feb-2020')

1 row created.
SQL> select * from user_spent;

ID DAY ---------- ---------

88 13-FEB-20 96 13-FEB-20 90 13-FEB-20

QL> insert into user_diet values(&id,'&day');
Enter value for id: 88
Enter value for day: 13-feb-2020

```
old 1: insert into user_diet values(&id,'&day')
new 1: insert into user_diet values(88,'13-feb-2020')


1 row created.

SQL> /
Enter value for id: 96
Enter value for day: 13-feb-2020
old 1: insert into user_diet values(&id,'&day')
new 1: insert into user_diet values(96,'13-feb-2020')

1 row created.

SQL>
Enter value for id: 90
Enter value for day: 13-feb-2020
old 1: insert into user_diet values(&id,'&day')
new 1: insert into user_diet values(90,'13-feb-2020')

1 row created.
SQL> select * from user_diet;

ID DAY ---------- ---------

88 13-FEB-20

96 13-FEB-20 90 13-FEB-20
```

Program:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import college.*;
import expert.*;
import hackathon.*;
import results.*;
import students.*;
@SuppressWarnings("serial")
public class FrontPage extends JFrame implements ActionListener{
        String msg = "";
         Label ll;
         CardLayout cardLO;

         //Create Panels for each of the menu items, welcome screen panel and home screen panel with CardLayout
         AddC addC;
         UpdateC upC;
         DeleteC delC;
         AddE addE;
```

```
UpdateE upE;
DeleteE delE;
AddH addH;
UpdateH upH;
DeleteH delH;
AddResults addR;
DeleteResults delR;
UpdateResults upR;
AddS addS;
UpdateS upS;
DeleteS delS;

Panel home,welcome;

public FrontPage()
{
        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 food and nutrition 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
addC=new AddC();addC.buildGUI();
upC = new UpdateC();  upC.buildGUI();
delC = new DeleteC();  delC.buildGUI();
addE = new AddE();addE.buildGUI();
upE = new UpdateE();upE.buildGUI();
delE=new DeleteE();delE.buildGUI();
addH=new AddH();addH.buildGUI();
upH=new UpdateH();upH.buildGUI();
delH=new DeleteH();delH.buildGUI();
addR=new AddR();addR.buildGUI();
delR=new DeleteR();delR.buildGUI();
```

```
upR=new UpdateR();upR.buildGUI();
addS=new AddS();addS.buildGUI();
upS = new UpdateS();upS.buildGUI();
delS = new DeleteS();delS.buildGUI();
//add all the panels to the home panel which has a cardlayout
home.add(welcome, "Welcome");
home.add(addC, "Add person");
home.add(upC, "Update person");
home.add(delC, "Delete person");
home.add(addE, "Add Expert");
home.add(upE, "Update Expert");
home.add(delE,"Delete Expert");
home.add(addH,"Add diet");
home.add(upH,"Update diet");
home.add(delH,"Delete diet");
home.add(addR,"Add Results");
home.add(upR,"Update Results");
home.add(delR,"Delete Results");
home.add(addS,"Add plan");
home.add(upS,"Update plan");
home.add(upS,"Delete plan");
// 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 C = new Menu("person");
MenuItem item1, item2, item3;
C.add(item1 = new MenuItem("Add person"));
C.add(item2 = new MenuItem("View person"));
C.add(item3 = new MenuItem("Delete person"));
mbar.add(C);

Menu E = new Menu("expert");
MenuItem item4, item5, item6;
E.add(item4 = new MenuItem("Add Expert"));
E.add(item5 = new MenuItem("View Expert"));
E.add(item6 = new MenuItem("Delete Expert"));
mbar.add(E);

Menu H = new Menu("diet");
```

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

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

Menu S = new Menu("plan");
MenuItem item13, item14, item15;
S.add(item13 = new MenuItem("Add plan"));
S.add(item14 = new MenuItem("View plan"));
S.add(item15 = new MenuItem("Delete plan"));
mbar.add(S);

// 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);
item13.addActionListener(this);
item14.addActionListener(this);
item15.addActionListener(this);


 // Anonymous inner class which extends WindowAdaptor to handle the Window event: windowClosing
addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent we)
    {
        quitApp();
```

```
                }
        });

        //Frame properties
        setTitle("food and nutrition system");
        setSize(500, 600);
        setVisible(true);

    }

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

        else if(arg.equals("View person"))
        {
            cardLO.show(home, "Update person");
            upC.loadColleges();
```

```
}

else if(arg.equals("Delete person"))
{
        cardLO.show(home, "Delete person");
        delC.loadColleges();
}

else if(arg.equals("Add Expert"))
{
        cardLO.show(home, "Add Expert");
}
else if(arg.equals("View Expert"))
{
        cardLO.show(home, "Update Expert");
        upE.loadExperts();
}
else if(arg.equals("Delete Expert"))
{
        cardLO.show(home, "Delete Expert");
        delE.loadExperts();
}
```

```
else if(arg.equals("Add diet"))
{
        cardLO.show(home, "Add diet");
}
else if(arg.equals("View diet"))
{
        cardLO.show(home, "Update diet");
        upH.loadHackathons();
}
else if(arg.equals("Delete diet"))
{
        cardLO.show(home, "Delete diet");
        delH.loadHackathons();
}
else if(arg.equals("Add Results"))
{
        cardLO.show(home, "Add Results");
}
else if(arg.equals("Delete Results"))
{
        cardLO.show(home, "Delete Results");
        delR.loadResults();
```

```
        }
        else if(arg.equals("View Results"))
        {
                cardLO.show(home, "Update Results");
                upR.loadResults();
        }
        else if(arg.equals("Add plan"))
        {
                cardLO.show(home, "Add plan");
        }
        else if(arg.equals("Delete plan"))
        {
                cardLO.show(home, "Delete plan");
                delS.loadStudents();
        }
        else if(arg.equals("View plan"))
        {
                cardLO.show(home, "Update plan");
                upS.loadStudents();
        }

}
```

```
    private void quitApp () {


            try {
                //Show a Confirmation Dialog.
                int reply = JOptionPane.showConfirmDialog (this,
                        "Are you really want to exit\nFrom food and nutrition suggestion system?",
                        "Contest - Exit", JOptionPane.YES_NO_OPTION, JOptionPane.PLAIN_MESSAGE);
                //Check the User Selection.
                if (reply == JOptionPane.YES_OPTION) {
                    setVisible (false); //Hide the Frame.
                    dispose();          //Free the System Resources.
                    System.out.println ("Thanks for Using food suggestion system\nAuthor - pradeep");
                    System.exit (0);    //Close the Application.
                }
                else if (reply == JOptionPane.NO_OPTION) {

setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
                }
            }
```

```
                catch (Exception e) {}

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



}
package c;

import java.awt.*;

import java.awt.event.*;
import java.sql.*;
public class Addperson extends Panel{
    /**
     *
     */
    private static final long serialVersionUID = 57263820961602445 64L;
```

```
Button AddCollegeButton;
TextField cidText,cnameText,addressText;
TextArea errorText;
Connection connection;
Statement statement;
public Addperson()
{
        try
        {
                Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
                System.err.println("Unable to find and l"
                                + ""
                                + ""
                                + ""
                                + "oad driver");
                System.exit(1);
        }
        connectToDB();
```

```
    }


    public void connectToDB()
  {
          try
          {
           connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:
1521:xe","pradeep","pradeep");
           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
AddCButton = new Button("Add person");
AddCButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
          //String query = "INSERT INTO company
(ID,NAME,Address,RATING) VALUES (2,'sai rohith','abc colony',20)";

          String query= "INSERT INTO
colleges(C_Address,CNAME,CID) VALUES('"+ addressText.getText() + "', " + "'"
+ cnameText.getText() +"',"+cidText.getText()+")";
          int i = statement.executeUpdate(query);
          statement.executeUpdate("commit");
          errorText.append("\nInserted " + i + " rows successfully");
        }
        catch (SQLException insertException)
        {
          displaySQLErrors(insertException);
        }
    }
```

```
        });
        cidText=new TextField(15);
        cnameText = new TextField(15);
        addressText = new TextField(15);


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

        Panel first = new Panel();
        first.setLayout(new GridLayout(4, 2));
        first.add(new Label("person weight:"));
        first.add(cidText);
        first.add(new Label("person Name:"));
        first.add(cnameText);
        first.add(new Label("person height:"));
        first.add(addressText);
        first.setBounds(125,90,200,100);

        Panel second = new Panel(new GridLayout(4, 1));
        second.add(AddCButton);
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("hello");
}



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

```
        }
}

package c;

import java.awt.*;
import java.awt.event.*;
import java.sql.*;
@SuppressWarnings("serial")
public class DeleteC extends Panel {
        //private static final List collegesIDList = null;
        Button deleteCollegeButton;
        List collegesIDList;
        TextField cidText, cnameText, c_addressText;
        TextArea errorText;
        Connection connection;
        Statement statement;
        ResultSet rs;

        public DeleteC()
        {
                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","pavan","pavan");
              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 loadC()
    {
        try
        {


            cweightList.removeAll();
          rs = statement.executeQuery("SELECT * FROM c");
          while (rs.next())
          {
            cIDList.add(rs.getString("Cweight"));
          }
        }
        catch (SQLException e)
        {
```

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

public void buildGUI()
{
   cList = new List(10);
        loadC();
        add(cList);

        //When a list item is selected populate the text fields
        collegesIDList.addItemListener(new ItemListener()
        {

                public void itemStateChanged(ItemEvent e)
                {
                        try
                        {
                                rs = statement.executeQuery("SELECT * FROM c");
```

```
                    while (rs.next())
                    {
                        if
(rs.getString("Cweight").equals(collegesIDList.getSelectedItem()))
                        break;
                    }
                    if (!rs.isAfterLast())
                    {
                        cidText.setText(rs.getString("Cweight"));
                        cnameText.setText(rs.getString("CNAME"));

c_addressText.setText(rs.getString("C_height"));
                    }
                }
                catch (SQLException selectException)
                {
                    displaySQLErrors(selectException);
                }
            }
        });


        deleteCollegeButton = new Button("Delete persons");
```

```
deleteCollegeButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("DELETE FROM
persons WHERE Cweight = "
                    + collegesIDList.getSelectedItem());
            errorText.append("\nDeleted " + i + " rows
successfully");
            cidText.setText(null);
            cnameText.setText(null);
            c_addressText.setText(null);
            statement.executeUpdate("commit");
            loadColleges();
        }
        catch (SQLException insertException)
        {
            displaySQLErrors(insertException);
        }
    }
```

```
});

cidText = new TextField(15);
cnameText = new TextField(15);
c_addressText = new TextField(15);

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

Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("person id:"));
first.add(cidText);
first.add(new Label("person Name:"));
first.add(cnameText);
first.add(new Label("person height:"));
first.add(c_addressText);

Panel second = new Panel(new GridLayout(4, 1));
second.add(deleteCollegeButton);

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");
        }


}
package c;
```

```
import java.awt.*;

import java.awt.event.*;
import java.sql.*;
@SuppressWarnings("serial")
public class UpdateC extends Panel{
	Button updateCButton;
	List cweightList;
	TextField cidText, cnameText, c_addressText;
	TextArea errorText;
	Connection connection;
	Statement statement;
	ResultSet rs;

	public UpdateCollege()
	{
		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","pradeep","pradeep");
              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 loadColleges()
      {
            //try
      //    {
                  try {
                        cweightList.removeAll();
rs = statement.executeQuery("SELECT Cweight FROM c");
while (rs.next())
{
                        collegeIDList.add(rs.getString("Cweight"));
}
                  } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        errorText.append("\nSQLException: " + e.getMessage() +
"\n");
                        errorText.append("SQLState:     " + e.getSQLState() +
"\n");
                        errorText.append("VendorError:  " + e.getErrorCode() +
"\n");
                  }
```

```
            //}
            //catch (SQLException e)
            //{
             // displaySQLErrors(e);
            //}
        }

        public void buildGUI()
        {
           cweightList = new List(10);
                loadC();
                add(cweightList);

                cweightList.addItemListener(new ItemListener()
                {
                        public void itemStateChanged(ItemEvent e)
                        {
                                try
                                {
                                        rs = statement.executeQuery("SELECT * FROM c
where cweight ="+cList.getSelectedItem());
                                        rs.next();
```

```
                    cidText.setText(rs.getString("Cweight"));
                    cnameText.setText(rs.getString("CNAME"));
                    c_addressText.setText(rs.getString("c_address"));
                }
                catch (SQLException selectException)
                {
                    displaySQLErrors(selectException);
                }
        }
});



updateCButton = new Button("Update person");
updateCButton.addActionListener(new ActionListener()
{
        public void actionPerformed(ActionEvent e)
        {
                try
                {
                    Statement statement = connection.createStatement();
                    int i = statement.executeUpdate("UPDATE persons "
```

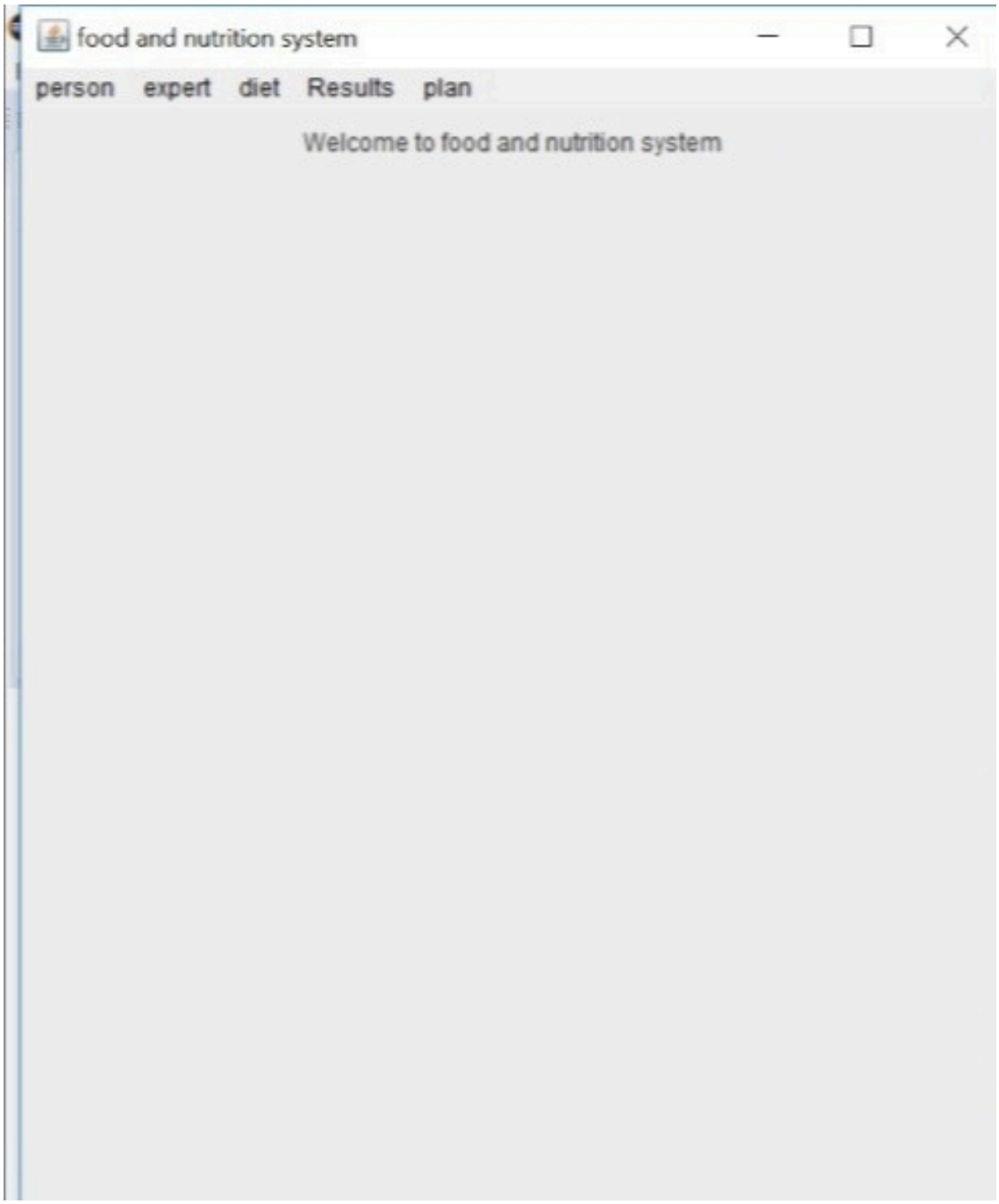
```
                        + "SET name='" + cnameText.getText() + "', "
                        + "c_address ='"+ c_addressText.getText() + "'
WHERE cid = "
                        + collegeIDList.getSelectedItem());
                        errorText.append("\nUpdated " + i + " rows
successfully");
                        i = statement.executeUpdate("commit");
                        loadColleges();
                    }
                    catch (SQLException insertException)
                    {
                        displaySQLErrors(insertException);
                    }
                }
            });

            cidText = new TextField(15);
            cidText.setEditable(false);
            cnameText = new TextField(15);
            c_addressText = new TextField(15);

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

```
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("person ID:"));
first.add(cidText);
first.add(new Label("person Name:"));
first.add(cnameText);
first.add(new Label("persons height:"));
first.add(c_addressText);

Panel second = new Panel(new GridLayout(4, 1));
second.add(updateCButton);

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

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

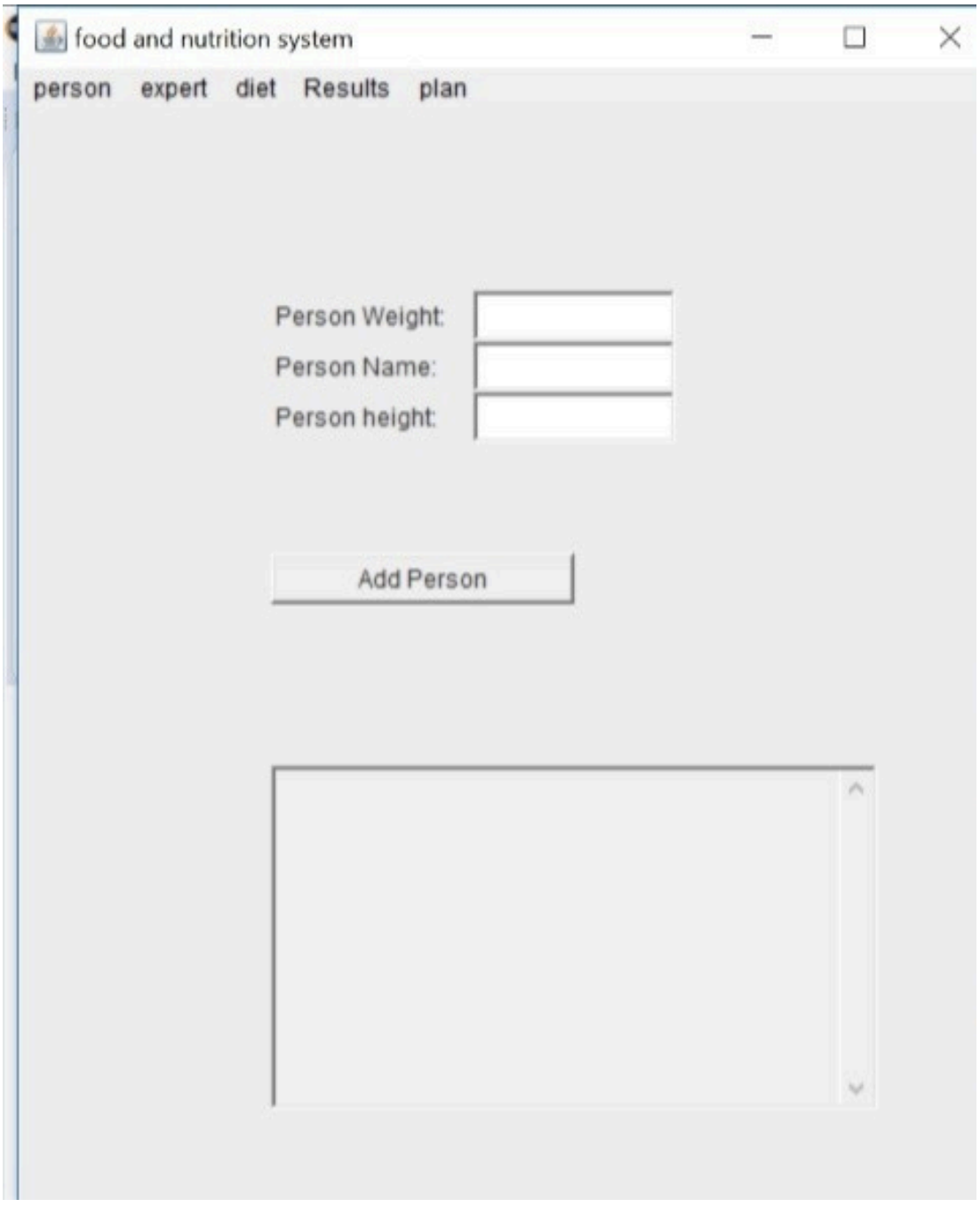
```
    }
    private void displaySQLErrors(SQLException e)
    {
        //errorText.append("\nSQLException: " + e.getMessage() + "\n");
        //errorText.append("SQLState:     " + e.getSQLState() + "\n");
        //errorText.append("VendorError:  " + e.getErrorCode() + "\n");
    }


}
```

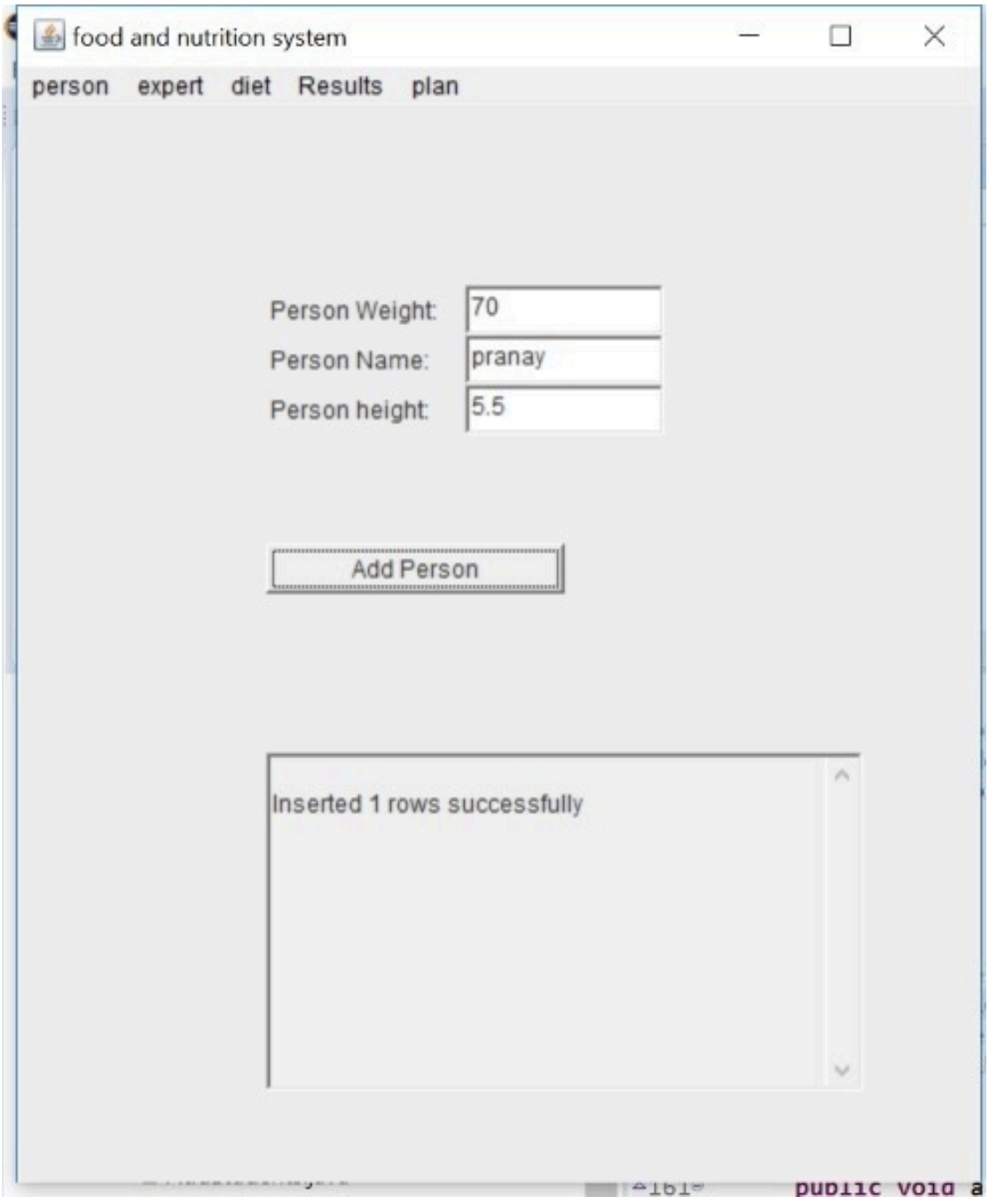
OUTPUT SCREENSHOTS:

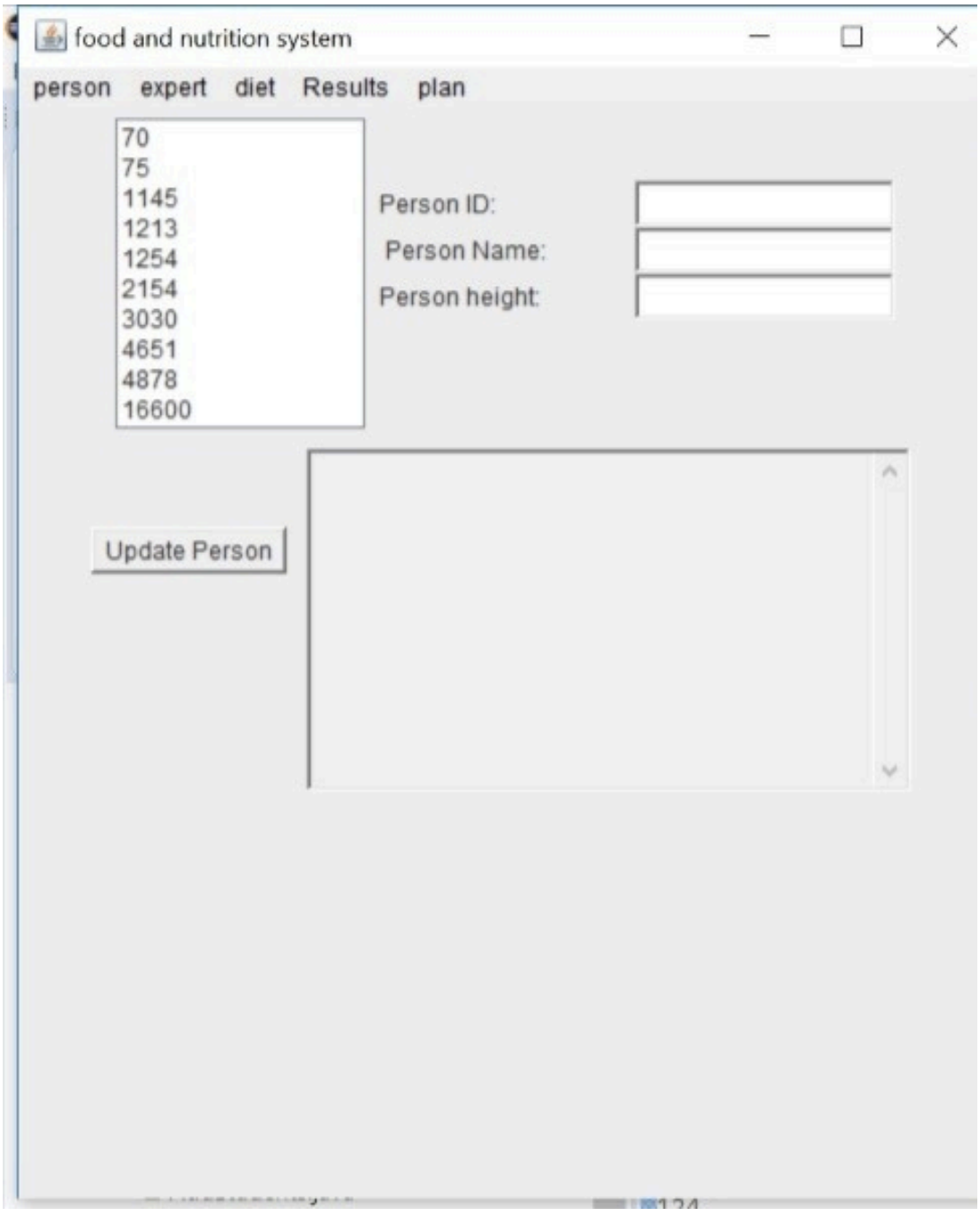
Java GUI Screenshot:

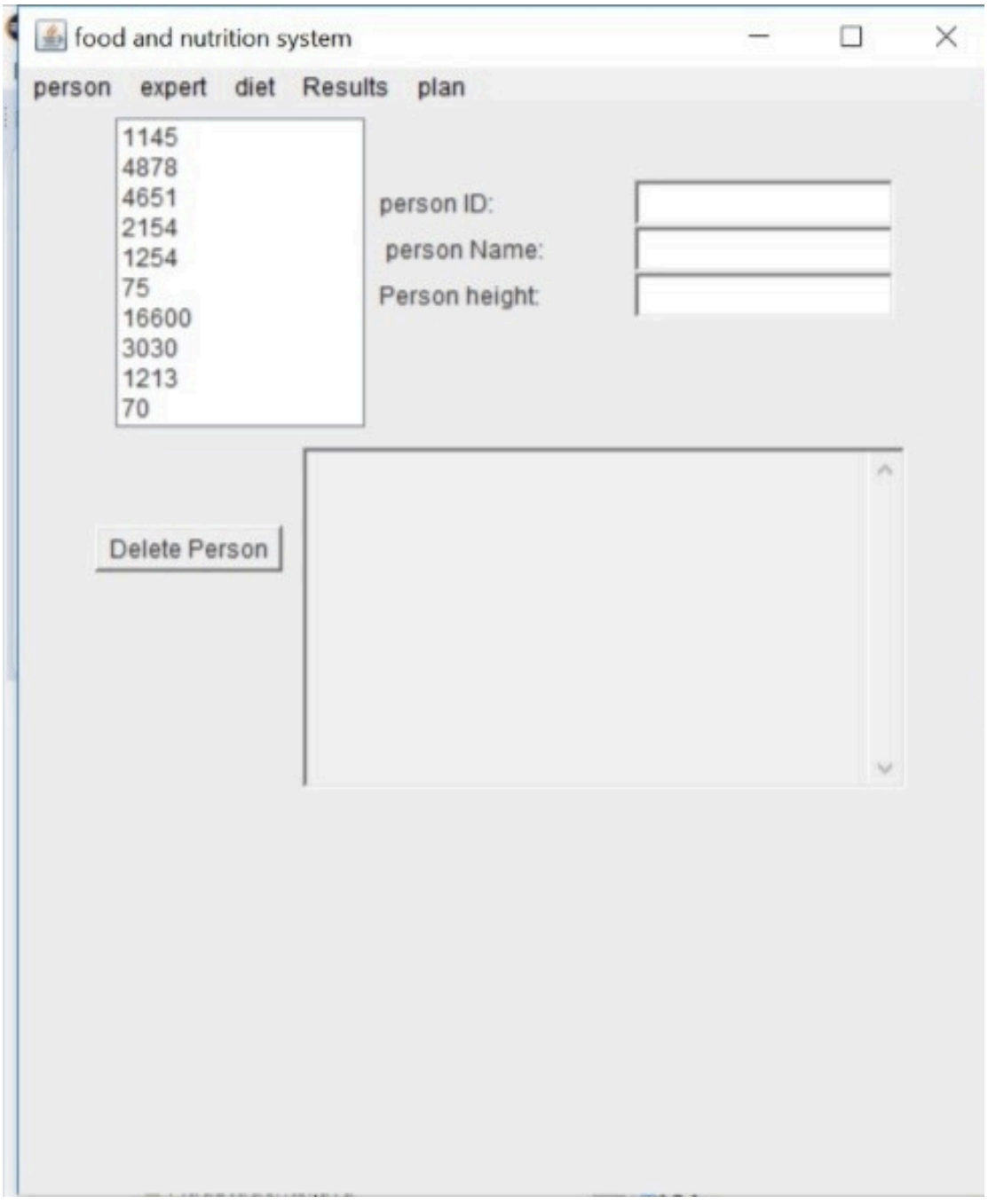



food and nutrition system
person
expert
diet
Results
plan
Add person
View person
Delete person
Welcome to food and nutrition system


food and nutrition system
person
expert
diet
Results
plan
Person Weight:
Person Name:
Person height:
Add Person


food and nutrition system
person
expert
diet
Results
plan
Person Weight:
70
Person Name:
pranay
Person height:
5.5
Add Person
Inserted 1 rows successfully


food and nutrition system
person
expert
diet
Results
plan
70
75
1145
1213
1254
2154
3030
4651
4878
16600
Person ID:
Person Name:
Person height:
Update Person


food and nutrition system
person
expert
diet
Results
plan
1145
4878
4651
2154
1254
75
16600
3030
1213
70
person ID:
person Name:
Person height:
Delete Person

CONCLUSION:

Thus, a Java AWT based registration form is created which is connected to the Oracle 11g database. Therefore, all the entries in the form are directly updated on the register table created in the database

REFERENCES:

https://www.decodejava.com/what-is-jdbc.htm

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

https://www.tutorialspoint.com/swing/index.htm