JAVA AWT BASED- Proctorial and personal counselling management system

A

Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

T.Hyndavi <1602-18-737-074>

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2020

BONAFIDE CERTIFICATE

This to Certify that the project report titled "**Proctorial and personal counselling management system**" project work of **T.Hyndavi** bearing Roll.no:**1602-18-737-074**who carried out this project under my supervision in the IV semester for the academic year 2019-2020.

Signature

external examiner

Signature

internal examiner

B.LEELAVATHY

Assistant Professor

Department of Information Technology

Abstract

Proctorial and personal counselling system is a system which manages the details regarding the proctors allocated to studends and students who require personal counselling are identified and allocated with a personal counseller. It is a student level data connection system. This Proctorial and personal counselling system has a major role in academics. This system helps students to know their position better and it helps to enhance their skills and perform well in their academics. To ensure quality teaching and learning process it is important to have interaction sessions between students and faculty and to know the problems faced by students so that the faculty can help and guide them.With this view and to resolve generic problems which students face in their regular academics, this project helps in maintaining this data and monitoring the students.

INTRODUCTION

Requirement Analysis

List of tables:

- Proctor
- proctor_personalCounseller
- personalCounseller
- personalCounseller_students
- students
- proctor_students
- department
- department_students

List of attributes and their domain types:

Proctor:

pid Number(10)

pname varchar2(15)

experience Number(2)

age Number(2)

proctor_personalCounseller:

pid Number(10)

cid Number (10)

personalCounseller:

cid Number(10)

cname varchar2 (15)

experience Number(2)

age Number(2)

personalCounseller_students:

cid Number(10)

sid Number(10)

students:

sid Number(10)

sname varchar2(15)

dob varchar2(15)

grade Number(3,1)

gender varchar2(5)

proctor_students:

pid Number(10)

sid Number(10)

department:

did Number(10)

dname varchar2(20)

hod varchar2(20)

department_students:

did Number(10)

sid Number(10)

Specific goal of the project:

The special goal of this project is to provide a online Proctorial and personal counselling management system to the universities so that they can monitor the students details and keep a check on their academics.

This system helps students to know their position better and it helps to enhance their skills and perform well in their academics. To ensure quality teaching and learning process it is important to have interaction sessions between students and faculty and to know the problems faced by students so that the faculty can help and guide them.With this view and to resolve generic problems which students face in their regular academics, this project helps in maintaining this data and monitoring the students.

Architecture and technology used:

SQL Plus is the most basic Oracle Database utility with a basic command-line interface, commonly used by users, administrators and programmers.

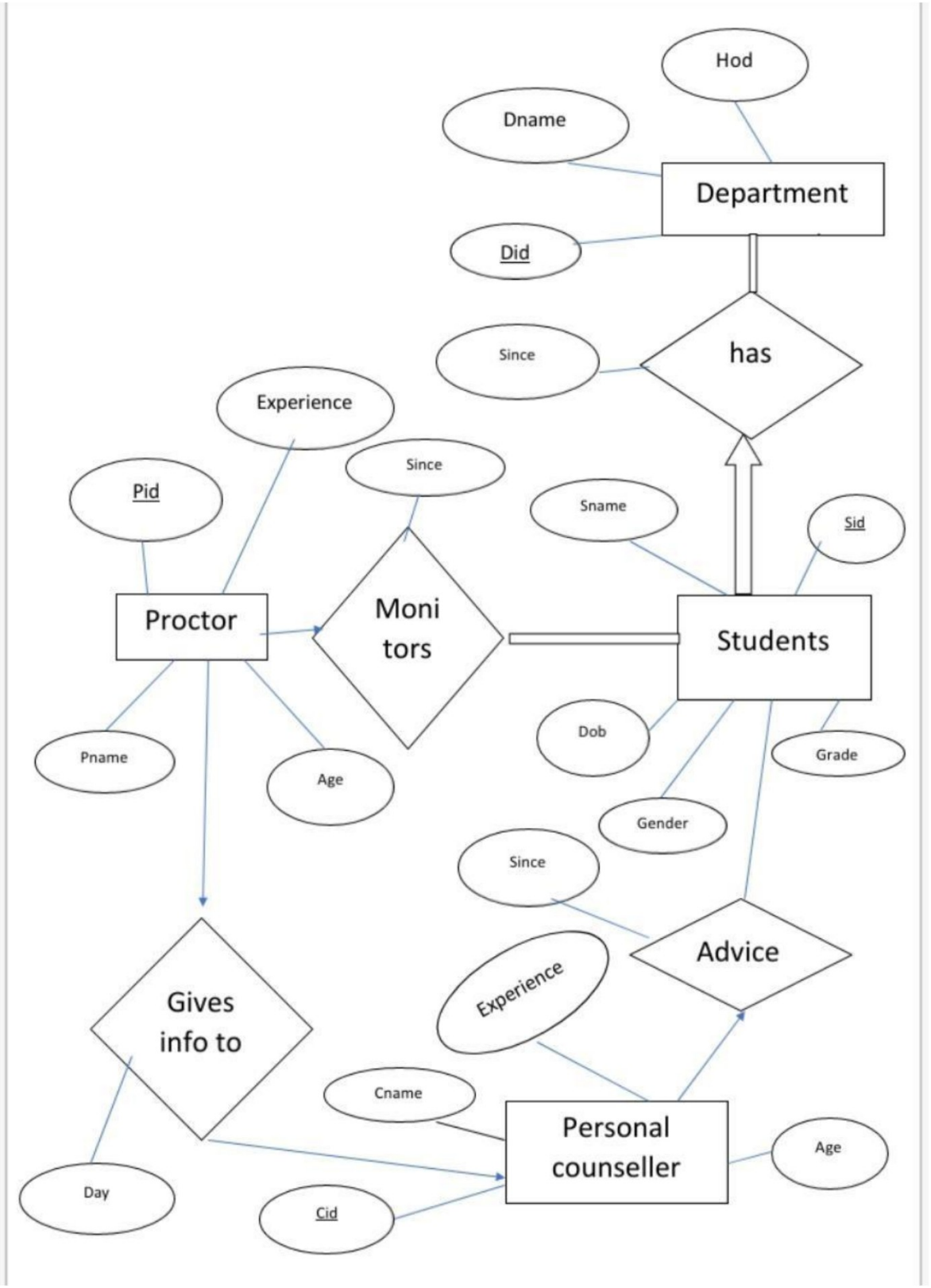
The interface of SQL Plus is used for creating the database. DDL and DML commands are implemented for operations being executed. The details of various Online MOOC's provider, courses, student, assignments, and results are stored in the form of tables in the database.

Eclipse 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. Eclipse is written mostly in java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Erlang, JavaScripts etc.

The front end application code is written in "**Java**" using Eclipse. The portal for front end application is designed through Eclipse, runs and has the capacity to connect with the database which has data inserted using SQL.

Design:

ER diagram



Mapping Cardinalities and Participation Constraints:

One proctor can monitor many students, but one student can have only one proctor. So it is one to many mapping.

All students must have a Proctor, so students are participating completely,it is complete participation.

One proctor gives information to only one personal counseller. Therefore it is one to one mapping.

One personal counseller can advice many students, but one student has only one personal counseller. Hence it is one to many mapping.Not all students need a personal counseller only students with less grade will have a personal counseller. So it is partial participation.

A department can have many students,but one student belongs to only one department. Therefore it is a many to one mapping.

Every student must belong to a department and also every department should have a student. So both are participating totally.

DDL Commands

SQL> create table proctor(pid Number(10) primary key,

Pname varchar2(15)

experience Number(2),

age Number(2)

);

Table created.

SQL> create table proctor_personalCounseller(pid Number(10),

cid Number (10),day varchar2(20),

foreign key (pid) references proctor,

foreign key (cid) references personalCounseller,

primarykey(pid,cid));

Table created.

SQL> create table personalCounseller(cid Number(10) primary key,

cnamevarchar2(15),

experience Number(2),

age Number(2));

Table created.

SQL> create table personalCounseller_students(sid Number(10) ,

cid Number (10),since varchar2(5),

```
foreign key (sid) references students,
foreign key (cid) references personalCounseller,
primarykey(sid,cid));
Table created.
SQL> create table students(sid Number(10) primary key,
sname varchar2(15 ),
dob varchar2(15),
grade Number(3,1),
gender varchar2(5));
Table created.
SQL> create table proctor_students(sid Number(10),
pid Number (10),since varchar2(5),
foreign key (sid) references students,
foreign key (pid) references proctor,
primarykey(sid,pid));
Table created.


SQL> create table department(did Number(10) primary key,
dname varchar2(20 )
hod varchar2(20));
Table created.
SQL> create table department_students(sid Number(10),
```

did Number (10),

foreign key (sid) references students,

foreign key (did) references department,primarykey(sid,cid));

Table created.

```
SQL> desc proctor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
---------
 PID                                       NOT NULL NUMBER(10)
 PNAME                                              VARCHAR2(15)
 EXPERIENCE                                         NUMBER(2)
 AGE                                                NUMBER(2)

SQL> desc proctor_personalCounseller
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
---------
 PID                                       NOT NULL NUMBER(10)
 CID                                       NOT NULL NUMBER(10)

SQL> desc personalCounseller;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
---------
 CID                                       NOT NULL NUMBER(10)
 CNAME                                              VARCHAR2(20)
 EXPERIENCE                                         NUMBER(2)
 AGE                                                NUMBER(2)



SQL> desc personalCouseller_students;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
---------
 CID                                       NOT NULL NUMBER(10)
 SID                                       NOT NULL NUMBER(10)


SQL> desc students;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
---------
 SID                                       NOT NULL NUMBER(10)
 SNAME                                              VARCHAR2(15)
 DOB                                                VARCHAR2(15)
 GRADE                                              NUMBER(3,1)
 GENDER                                             VARCHAR2(5)

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

```
 ---------------------------------------- -------- ------------------
---------
 PID                                      NOT NULL NUMBER(10)
 SID                                      NOT NULL NUMBER(10)


SQL> desc department;
 Name                                     Null?    Type
 ---------------------------------------- -------- ------------------
---------
 DID                                      NOT NULL NUMBER(10)
 DNAME                                             VARCHAR2(20)
 HOD                                               VARCHAR2(20)

SQL> desc departmen_students;
 Name                                     Null?    Type
 ---------------------------------------- -------- ------------------
---------
 DID                                      NOT NULL NUMBER(10)
 SID                                      NOT NULL NUMBER(10
```

Implementation:

Front end programs:

1.INSERT STUDENT:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class InsertStudent extends Frame
{
Button insertStudentButton;
TextField sidText, snameText, dobText, genderText,gradeText;
TextArea errorText;
Connection connection;
Statement statement;
public InsertStudent()
{
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","hyndavi","hyndhu3612");
 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 buildGUI()
{
//Handle Insert Account Button
insertStudentButton = new Button("Insert Student");
insertStudentButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
try
{
 //String query = "INSERT INTO sailors (SID,SNAME, RATING, AGE) VALUES (2,'Divya',7,20)";
 String query= "INSERT INTO students VALUES(" + sidText.getText() + ", " + "'" + snameText.getText()+"'"
+ "," + "'"+dobText.getText() + "'"+","+gradeText.getText() +  "," + "'"+genderText.getText()+"'"+")";
 int i = statement.executeUpdate(query);
 errorText.append("\nInserted " + i + " rows successfully");
}
catch (SQLException insertException)
{
 displaySQLErrors(insertException);
}
}
});


sidText = new TextField(30);
snameText = new TextField(30);
genderText = new TextField(30);
dobText = new TextField(30);
gradeText = new TextField(30);


errorText = new TextArea(20, 40);
errorText.setEditable(false);

Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Student ID:"));
first.add(sidText);
first.add(new Label("Name:"));
first.add(snameText);
```

```
first.add(new Label("DOB:"));
first.add(dobText);
first.add(new Label("Grade:"));
            first.add(gradeText);
first.add(new Label("Gender:"));
            first.add(genderText);
first.setBounds(125,100,200,100);

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

setTitle("New Student Creation");
setSize(500, 600);
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");
}



public static void main(String[] args)
{
InsertStudent stu = new InsertStudent();

stu.addWindowListener(new WindowAdapter(){
 public void windowClosing(WindowEvent e)
 {
System.exit(0);
 }
});

stu.buildGUI();
}
}
```

2.DELETE STUDENT

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class DeleteStudent extends Frame
```

```
{
Button deleteStudentButton;
List studentIDList;
TextField sidText, snameText, gradeText,dobText ,genderText;
TextArea errorText;
Connection connection;
Statement statement;
ResultSet rs;

public DeleteStudent()
{
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","hyndavi","hyndhu3612");
 statement = connection.createStatement();

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

private void loadStudents()
{
try
{
 rs = statement.executeQuery("SELECT * FROM students");
 while (rs.next())
 {
studentIDList.add(rs.getString("SID"));
 }
}
catch (SQLException e)
{
 displaySQLErrors(e);
}
}

public void buildGUI()
```

```
{
   studentIDList = new List(10);
loadStudents();
add(studentIDList);

//When a list item is selected populate the text fields
studentIDList.addItemListener(new ItemListener()
{
public void itemStateChanged(ItemEvent e)
{
try
{
rs = statement.executeQuery("SELECT * FROM students");
while (rs.next())
{
if (rs.getString("SID").equals(studentIDList.getSelectedItem()))
break;
}
if (!rs.isAfterLast())
{
sidText.setText(rs.getString("SID"));
snameText.setText(rs.getString("SNAME"));
dobText.setText(rs.getString("DOB"));
gradeText.setText(rs.getString("GRADE"));
genderText.setText(rs.getString("GENDER"));
}
}
catch (SQLException selectException)
{
displaySQLErrors(selectException);
}
}
});


//Handle Delete Sailor Button
deleteStudentButton = new Button("Delete Student");
deleteStudentButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
try
{
Statement statement = connection.createStatement();
int i = statement.executeUpdate("DELETE FROM students WHERE SID = "
+ studentIDList.getSelectedItem());
errorText.append("\nDeleted " + i+1 + " rows successfully");
sidText.setText(null);
snameText.setText(null);
dobText.setText(null);
gradeText.setText(null);
genderText.setText(null);
studentIDList.removeAll();
loadStudents();
}
catch (SQLException insertException)
{
displaySQLErrors(insertException);
```

```
}
}
});

sidText = new TextField(15);
snameText = new TextField(15);
dobText = new TextField(15);
gradeText = new TextField(15);
genderText = 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("Student ID:"));
first.add(sidText);
first.add(new Label("Name:"));
first.add(snameText);
first.add(new Label("DOB:"));
first.add(dobText);
first.add(new Label("Grade:"));
first.add(gradeText);
first.add(new Label("Gender:"));
            first.add(genderText);

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

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

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

setTitle("Remove Student");
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");
}



public static void main(String[] args)
{
DeleteStudent dels = new DeleteStudent();
```

```
dels.addWindowListener(new WindowAdapter(){
 public void windowClosing(WindowEvent e)
 {
System.exit(0);
 }
});

dels.buildGUI();
}
}
```

3.UPDATE STUDENT

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class UpdateStudent extends Frame
{
Button updateStudentButton;
List studentIDList;
TextField sidText, snameText, dobText, gradeText,genderText;
TextArea errorText;
Connection connection;
Statement statement;
ResultSet rs;

public UpdateStudent()
{
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","hyndavi","hyndhu3612");
 statement = connection.createStatement();

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

```
private void loadStudents()
{
try
{
 rs = statement.executeQuery("SELECT SID FROM students");
 while (rs.next())
 {
studentIDList.add(rs.getString("SID"));
 }
}
catch (SQLException e)
{
 displaySQLErrors(e);
}
}

public void buildGUI()
{
   studentIDList = new List(10);
loadStudents();
add(studentIDList);

//When a list item is selected populate the text fields
studentIDList.addItemListener(new ItemListener()
{
public void itemStateChanged(ItemEvent e)
{
try
{
rs = statement.executeQuery("SELECT * FROM students where SID ="+studentIDList.getSelectedItem());
rs.next();
sidText.setText(rs.getString("SID"));
snameText.setText(rs.getString("SNAME"));
dobText.setText(rs.getString("DOB"));
gradeText.setText(rs.getString("GRADE"));
genderText.setText(rs.getString("GENDER"));

}
catch (SQLException selectException)
{
displaySQLErrors(selectException);
}
}
});


//Handle Update Sailor Button
updateStudentButton = new Button("Update Student");
updateStudentButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
try
{
Statement statement = connection.createStatement();
int i = statement.executeUpdate("UPDATE students "
+ "SET sname='" + snameText.getText() + "', "
+ "dob='" + dobText.getText() + "', "
```

```
+ "grade ="+ gradeText.getText() +", "
                                + "gender ='"+ genderText.getText() + "' WHERE sid = "
+ studentIDList.getSelectedItem());
errorText.append("\nUpdated " + i + " rows successfully");
studentIDList.removeAll();
loadStudents();
}
catch (SQLException insertException)
{
displaySQLErrors(insertException);
}
}
});

sidText = new TextField(15);
sidText.setEditable(false);
snameText = new TextField(15);
dobText = new TextField(15);
gradeText = new TextField(15);
genderText = 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("Sailor ID:"));
first.add(sidText);
first.add(new Label("Name:"));
first.add(snameText);
first.add(new Label("Dob:"));
first.add(dobText);
first.add(new Label("Grade:"));
first.add(gradeText);
first.add(new Label("Gender:"));
            first.add(genderText);

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

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

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

setTitle("Update Student");
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");
}

public static void main(String[] args)
{
UpdateStudent upst = new UpdateStudent();

upst.addWindowListener(new WindowAdapter(){
 public void windowClosing(WindowEvent e)
 {
System.exit(0);
 }
});

upst.buildGUI();
}
```

4. Connectivity with the Database:

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.

Block of code for JAVA- SQL connectivity with JDBC:

```
/*import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException; */

import java.sql.*;

class OracleCon
{
public static void main(String args[]){
try{
//step1 load the driver class
Class.forName("oracle.jdbc.driver.OracleDriver");   // or Class.forName("oracle.jdbc.OracleDriver");

//step2 create  the connection object
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","hyndavi","hyndhu3612");

//step3 create the statement object
Statement stmt=con.createStatement();

//step4 execute query
ResultSet rs=stmt.executeQuery("select * from students");
while(rs.next())
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));

//step5 close the connection object
```

```
con.close();

}catch(Exception e){ System.out.println(e);}

}
}
```

5.Main program-PROCTORIALSYSTEM

```
import java.awt.*;
import java.awt.event.*;

class proctorialSystem extends Frame implements ActionListener
{
 String msg = "";
 Label ll;
 InsertStudent stu;
 UpdateStudent upst;
 DeleteStudent dels;
 InsertProctor proc;
        UpdateProctor uppro;
        DeleteProctor delpro;
 InsertPersonalCounseller perc;
        UpdatePersonalCounseller uppc;
        DeletePersonalCounseller delpc;
 InsertDepartment dep;
        UpdateDepartment upd;
        DeleteDepartment deld;
 //MakeReservation mks;
 personalCounseller_student pcs;
 proctor_student ps;
 proctor_personalCounseller ppc;
 department_student ds;


 proctorialSystem()
 {
ll = new Label();
ll.setAlignment(Label.CENTER);
ll.setBounds(100,300,300,150);
ll.setText("Welcome to proctorial system");
add(ll);

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

// create the menu items and add it to Menu
Menu student = new Menu("Student");
MenuItem item1, item2, item3;
student.add(item1 = new MenuItem("Submit Student"));
student.add(item2 = new MenuItem("Modify Student"));
student.add(item3 = new MenuItem("Delete Student"));
mbar.add(student);

Menu proctor = new Menu("proctor");
MenuItem item4, item5, item6;
```

```
proctor.add(item4 = new MenuItem("Submit proctor"));
proctor.add(item5 = new MenuItem("Modify proctor"));
proctor.add(item6 = new MenuItem("Delete  proctor"));
mbar.add(proctor);
    Menu personalCounseller = new Menu("personalCounseller");
                MenuItem item7, item8, item9;
                 personalCounseller.add(item7 = new MenuItem("Submit personalCounseller"));
                 personalCounseller.add(item8 = new MenuItem("Modify personalCounseller"));
                 personalCounseller.add(item9= new MenuItem("Delete  personalCounseller"));
                mbar.add(personalCounseller);
Menu department = new Menu("department");
                MenuItem item10, item11, item12;
                department.add(item10 = new MenuItem("Submit department"));
                department.add(item11= new MenuItem("Modify department"));
                department.add(item12= new MenuItem("Delete department"));
                mbar.add(department);


Menu proctor_student = new Menu("proctor_student");
MenuItem item13, item14, item15;
proctor_student.add(item13 = new MenuItem("Submit proctor_student"));
//proctor_student.add(item14= new MenuItem("Modify student_proctor"));
//proctor_student.add(item15= new MenuItem("Delete student_proctor"));
mbar.add(proctor_student);
    Menu  department_student = new Menu(" department_student");
                MenuItem item16, item17, item18;
                 department_student.add(item16 = new MenuItem("Submit department_student"));
                 //department_student.add(item17= new MenuItem("Modify  department_student"));
                 //department_student.add(item18= new MenuItem("Delete  department_student"));
                mbar.add( department_student);
Menu  personalCounseller_student= new Menu(" personalCounseller_student");
                MenuItem item19, item20, item21;
                 personalCounseller_student.add(item19 = new MenuItem("Submit
personalCounseller_student"));
                 //personalCounseller_student.add(item20= new
MenuItem("Modify  personalCounseller_student"));
                 //personalCounseller_student.add(item21= new
MenuItem("Delete  personalCounseller_student"));
                mbar.add( personalCounseller_student);
Menu proctor_personalCounseller = new Menu("proctor_personalCounseller");
                MenuItem item22, item23, item24;
                proctor_personalCounseller.add(item22 = new MenuItem("Submit
proctor_personalCounseller"));
                //proctor_personalCounseller.add(item23= new MenuItem("Modify
proctor_personalCounseller"));
                //proctor_personalCounseller.add(item24 = new MenuItem("Delete
proctor_personalCounseller"));
                mbar.add(proctor_personalCounseller);
// 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);
                    item16.addActionListener(this);
                    //item17.addActionListener(this);
                    //item18.addActionListener(this);
                    item19.addActionListener(this);
//tem20.addActionListener(this);
                    //item21.addActionListener(this);
                    item22.addActionListener(this);
                    //item23.addActionListener(this);
                    //item24.addActionListener(this);

// Anonymous inner class which extends WindowAdaptor to handle the Window event: windowClosing
addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent we)
{
System.exit(0);
}
});

//Frame properties
setTitle("Proctorial system");
Color clr = new Color(200, 100, 150);
setBackground(clr);
setFont(new Font("SansSerif", Font.BOLD, 14));
setLayout(null);
setSize(500, 600);
setVisible(true);

 }

 public void actionPerformed(ActionEvent ae)
 {

 String arg = ae.getActionCommand();
 if(arg.equals("Submit Student"))
 {
stu = new InsertStudent();

stu.addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent e)
{
stu.dispose();
}
});
stu.buildGUI();
        }

else if(arg.equals("Modify Student"))
{
upst = new UpdateStudent();
upst.addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent e)
  {
```

```
upst.dispose();
}
});
upst.buildGUI();
}

else if(arg.equals("Delete Student"))
{
dels = new DeleteStudent();

dels.addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent e)
{
dels.dispose();
}
});
dels.buildGUI();
}
else if(arg.equals("Submit proctor"))
              {
                   proc = new InsertProctor();

                   proc.addWindowListener(new WindowAdapter(){
                   public void windowClosing(WindowEvent e)
                   {
                         proc.dispose();
                   }
                   });
                   proc.buildGUI();
       }

             else if(arg.equals("Modify proctor"))
             {
                   uppro = new UpdateProctor();
                   uppro.addWindowListener(new WindowAdapter(){
                   public void windowClosing(WindowEvent e)
                 {
                         uppro.dispose();
                   }
                   });
                   uppro.buildGUI();
             }
 else if(arg.equals("Delete proctor"))
             {
                   delpro = new DeleteProctor();

                   delpro.addWindowListener(new WindowAdapter(){
                   public void windowClosing(WindowEvent e)
                   {
                         delpro.dispose();
                   }
                   });
                   delpro.buildGUI();
             }
else if(arg.equals("Submit personalCounseller"))
              {
                   perc = new InsertPersonalCounseller();
```

```
                perc.addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent e)
                {
                        perc.dispose();
                }
                });
                perc.buildGUI();
        }

                else if(arg.equals("Modify personalCounseller"))
                {
                        uppc = new UpdatePersonalCounseller();
                        uppc.addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent e)
                  {
                                uppc.dispose();
                        }
                        });
                        uppc.buildGUI();
                }
 else if(arg.equals("Delete personalCounseller"))
                {
                        delpc = new DeletePersonalCounseller();

                        delpc.addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent e)
                        {
                                delpc.dispose();
                        }
                        });
                        delpc.buildGUI();
                }
else if(arg.equals("Submit department"))
                 {
                        dep = new InsertDepartment();

                        dep.addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent e)
                        {
                                dep.dispose();
                        }
                        });
                        dep.buildGUI();
        }

       else if(arg.equals("Modify department"){
                      upd = new UpdateDepartment();
                        upd.addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent e)
                  {
                                upd.dispose();
                        }
                        });
                        upd.buildGUI();
                }
else if(arg.equals("Delete department"))
                {
                        deld = new DeleteDepartment();
```

```
                deld.addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent e)
                {
                        deld.dispose();
                }
                });
                deld.buildGUI();
        }
else if(arg.equals("Submit proctor_student"))
{
ps = new proctor_student();
setVisible(false);
ps.addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent e)
{
ps.dispose();
setVisible(true);
}
});
ps.buildGUI();
}
else if(arg.equals("Submit department_student"))
        {
                ds = new department_student();
                setVisible(false);
                ds.addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent e)
                {
                        ds.dispose();
                        setVisible(true);
                }
                });
                ds.buildGUI();
        }
else if(arg.equals("Submit personalCounseller_student"))
        {
                pcs = new personalCounseller_student();
                setVisible(false);
                pcs.addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent e)
                {
                        pcs.dispose();
                        setVisible(true);
                }
                });
                pcs.buildGUI();
        }
else if(arg.equals("Submit proctor_personalCounseller"))
        {
                ppc = new proctor_personalCounseller();
                setVisible(false);
                ppc.addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent e)
                {
                        ppc.dispose();
                        setVisible(true);
                }
```

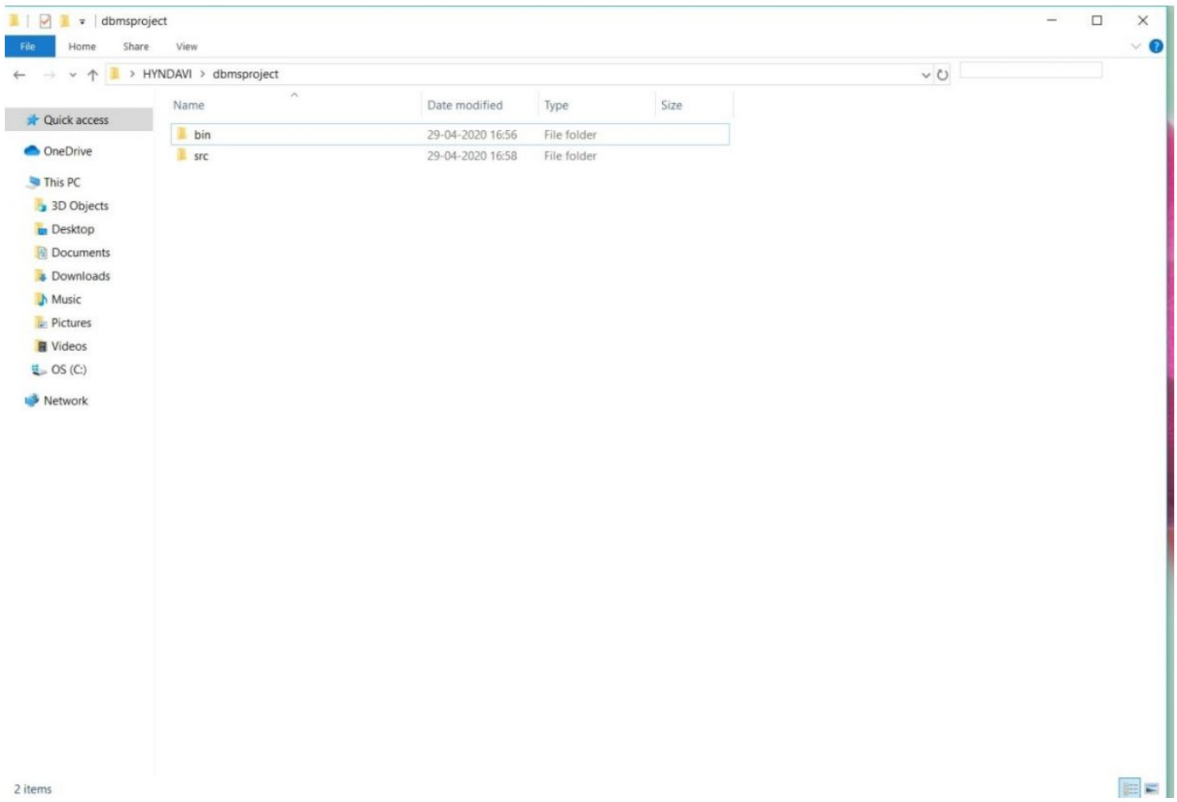
```
                    });
                    ppc.buildGUI();
                }
 }
  public static void main(String ... args)
 {
new proctorialSystem();
 }
}
```

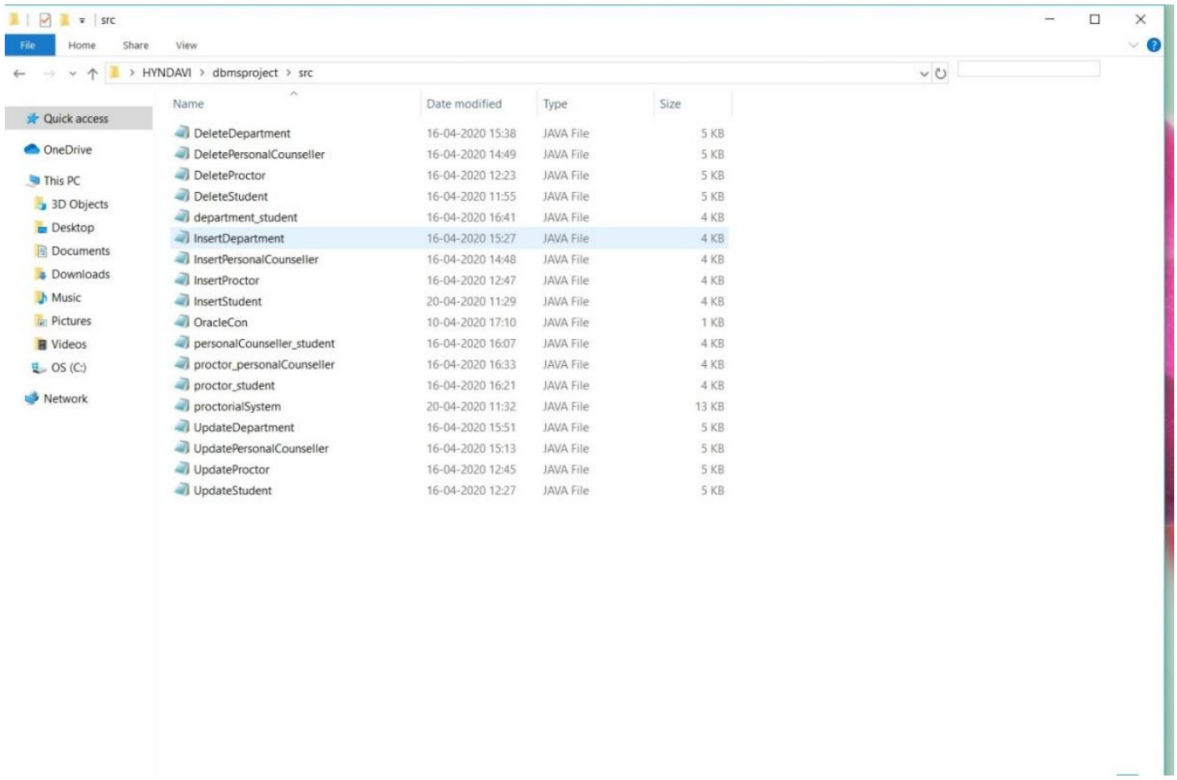
Github link:

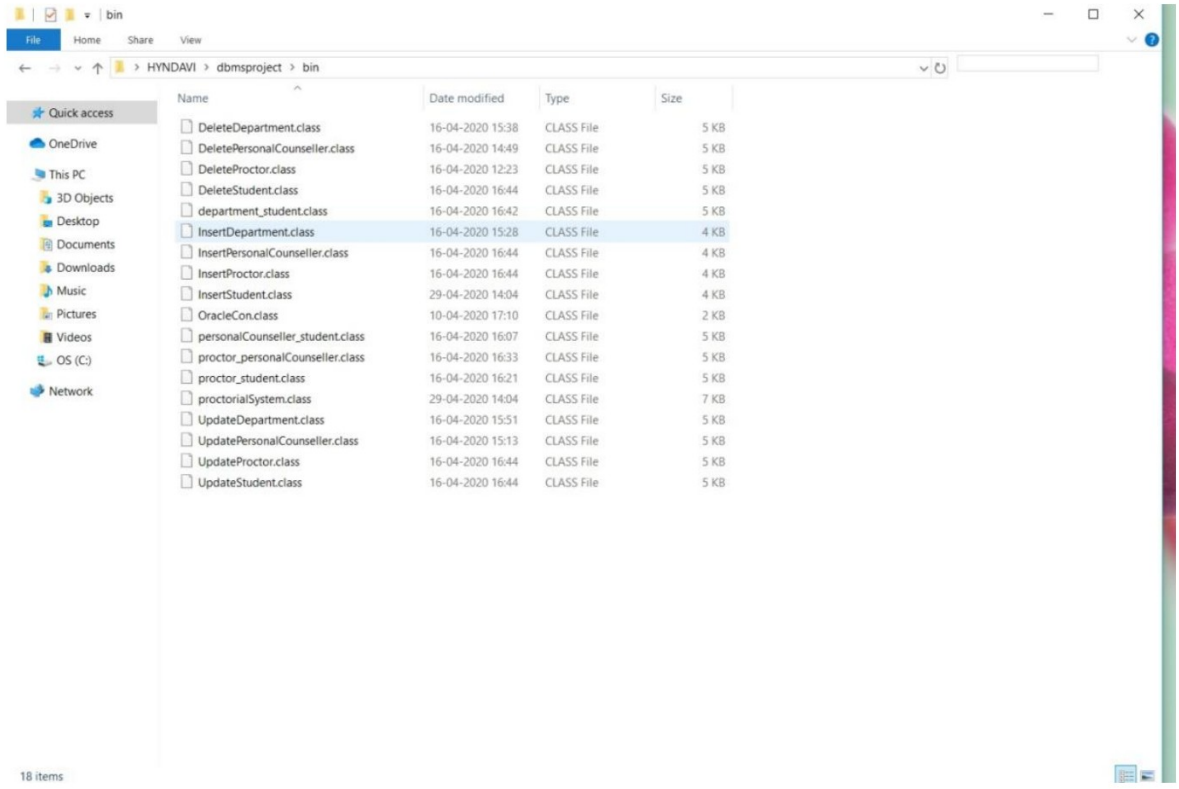
https://github.com/hyndavi-ui

Folder Structure :

The folder **dbmsproject** consists of two sub folders-**src** and **bin**. src folder consists of all .java files and bin folder consists of all .class files.







Testing:

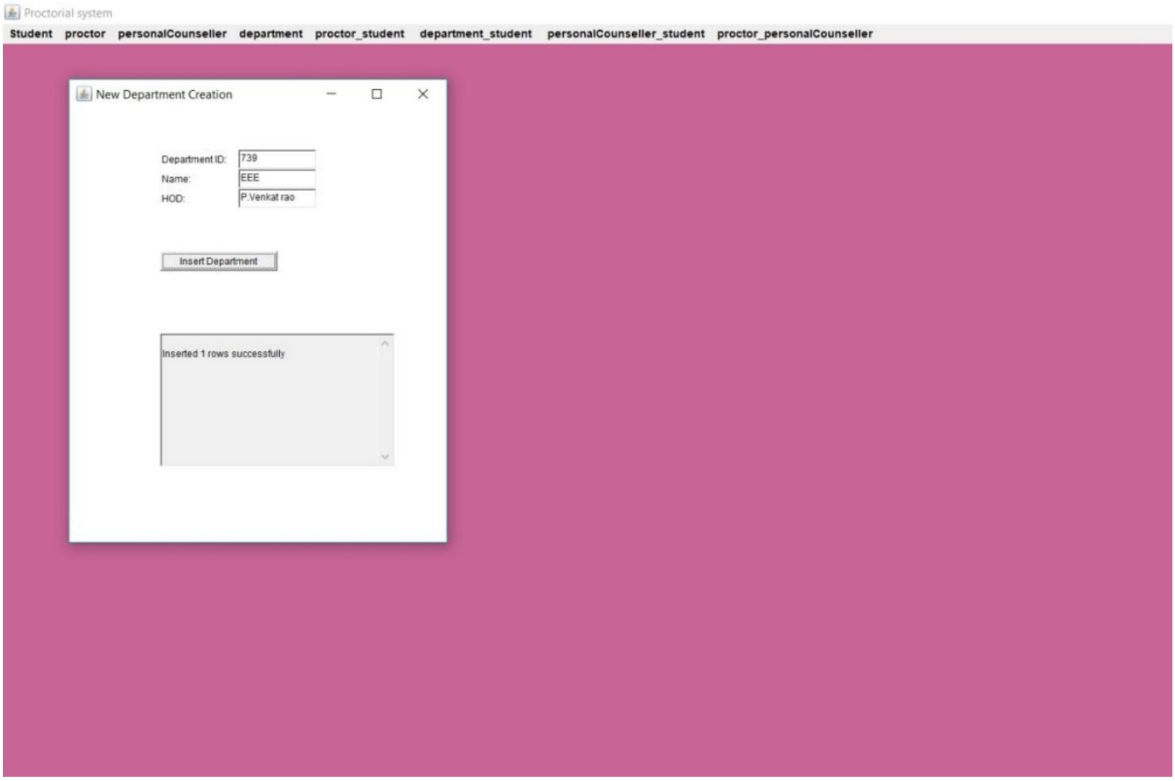
The program runs for execution of three basic operations of insertion, update and delete on 8 different table. Along with this, it also has a output column which gives the information about how many rows have been edited. Errors, syntactical or exceptional will be shown if occurred.

The code written for building GUI and connecting with database ensures that the values entered by the user are of correct data types. It prompts an error message in the text message box.

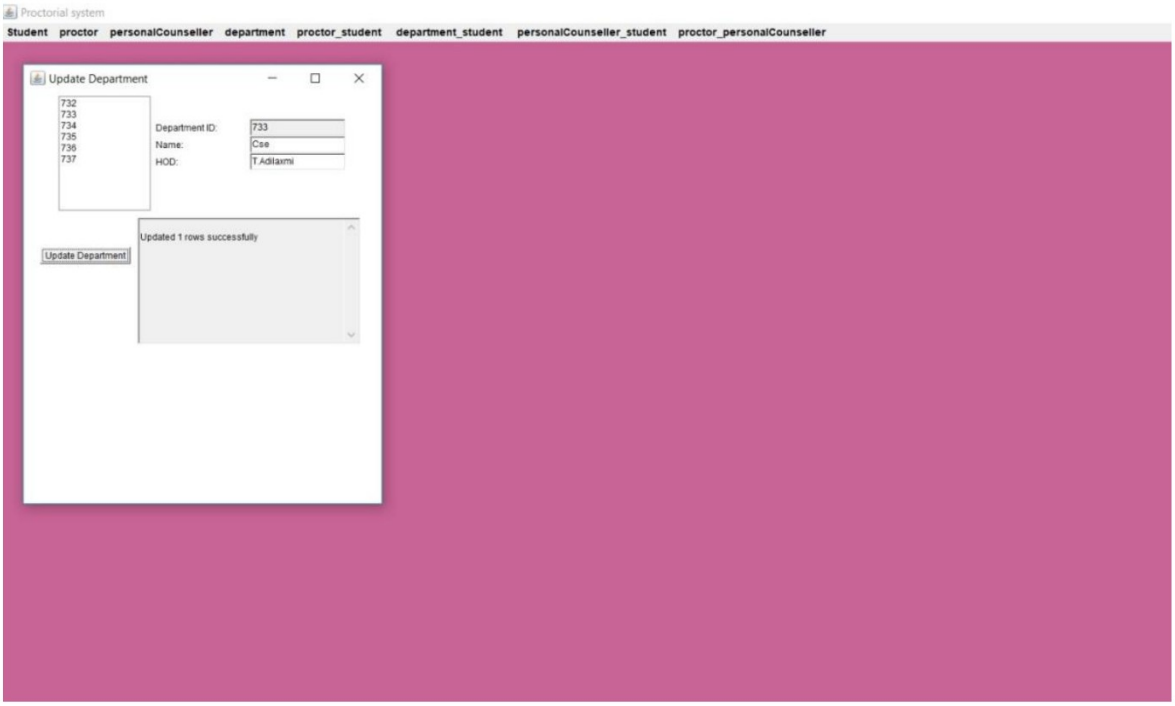
Main frame:

Output screenshots:

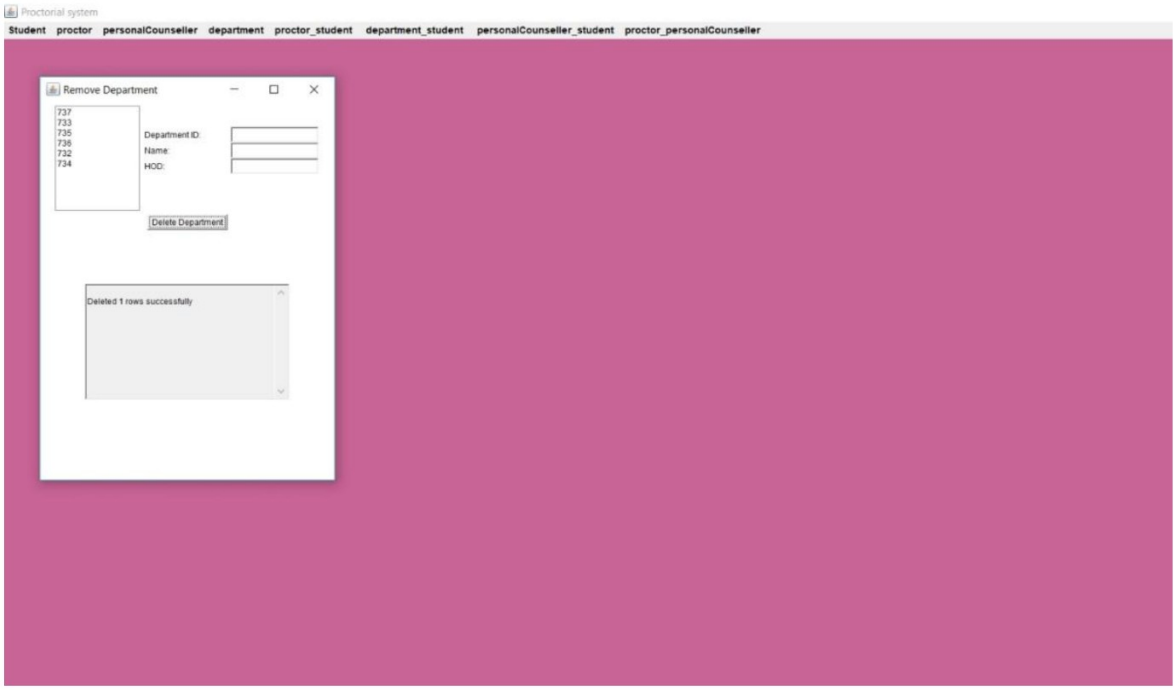
Insert



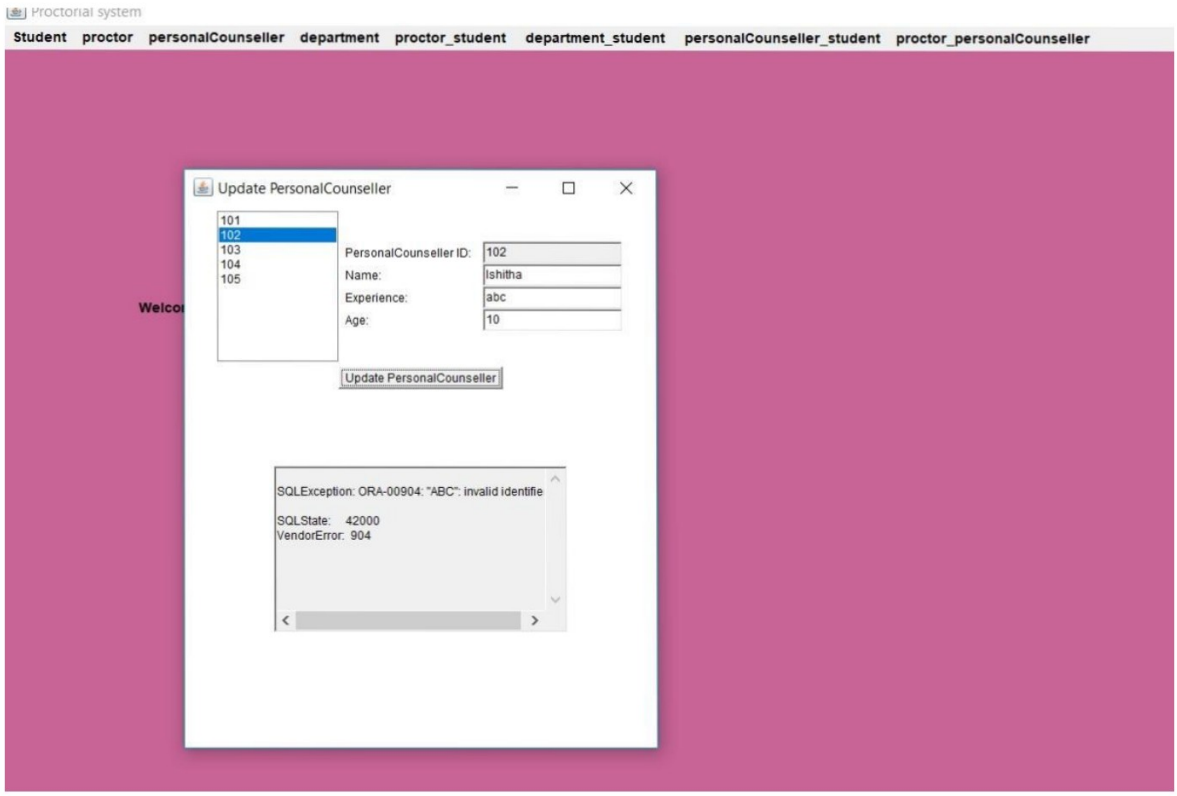
Update



Delete



If we give wrong input ,it will display an error message:



Results:

DML Commands

SQL> insert into proctor values(&pid,'&pname',&experience, &age);
Enter value for pid: 1
Enter value for pname: Deepa
Enter value for experience: 8
Enter value for age: 30
old 1: insert into proctor values(&pid,'&pname',&experience,&age)
new 1: insert into proctor values(1,Deepa,8,30)
SQL> /
5 rows created

```
SQL> insert into proctor_personalCounseller values(&pid, &cid);
Enter value for pid: 1
Enter value for cid: 101
old   1: insert into proctor values(&pid,&cid)
new   1: insert into proctor values(1,101)
SQL> /
5 rows created
SQL> insert into personalCounseller values(&cid,'&cname',&experience, &age);
Enter value for cid: 101
Enter value for cname: Laxmi
Enter value for experience:20
Enter value for age: 45
old   1: insert into proctor values(&cid,'&cname',&experience,&age)
new   1: insert into proctor values(101,Laxmi,20,45)
SQL> /
5 rows created
SQL> insert into personalCounseller_students values(&sid, &cid);
Enter value for sid: 14
Enter value for cid: 102
old   1: insert into proctor values(&sid,&cid)
new   1: insert into proctor values(14,102)
SQL> /
5 rows created
```

```
SQL> insert into students values(&sid,'&sname','&dob',&grade, '&gender');
Enter value for sid: 10
Enter value for sname: Ruhi
Enter value for dob: 01-Apr-2001
Enter value for grade: 9.3
Enter value for gender:f
old   1: insert into proctor values(&pid,'&pname',&experience,&age)
new   1: insert into proctor values(1,Deepa,8,30)
SQL> /
5 rows created

SQL> insert into proctor_students values(&pid, &sid);
Enter value for pid: 1
```

Enter value for sid: 101
old 1: insert into proctor values(&pid,&sid)
new 1: insert into proctor values(1,12)
SQL> /
5 rows created

SQL> insert into department values(&did,'&dname',’&hod');
Enter value for did: 737
Enter value for dname: It
Enter value for hod: K.Ram Mohan
old 1: insert into proctor values(&did,'&dname',’&hod ‘)
new 1: insert into proctor values(737,It,K.Ram Mohan
SQL> /
5 rows created

SQL> insert into department_students values(&did, &sid);
Enter value for did: 737
Enter value for sid: 10
old 1: insert into proctor values(&did,&sid)
new 1: insert into proctor values(737,10)
SQL> /
5 rows created

```
SQL> select * from proctor;

       PID PNAME            EXPERIENCE        AGE
---------- --------------- ---------- ----------
         1 Deepa                    8         30
         2 Karthik                  7         32
         3 Soundarya               10         46
         4 Sravya                   6         29
         5 Anand                   12         49

SQL> select * from proctor_personalCounseller;

       PID        CID
---------- ----------
         1        101
         2        102
         3        103
         4        104
         5        105

SQL> select * from personalCounseller;

       CID CNAME                EXPERIENCE        AGE
```

```
---------- -------------------- ---------- ----------
       101 Laxmi                        45         20
       102 Ishitha                      25         10
       103 Raman                        30         12
       104 Aditya                       28         10
       105 Arun                         32         13

SQL> select * from personalCounseller_students;

       CID        SID
---------- ----------
       102         14
       103         18
       101         19

SQL> select * from students;

       SID SNAME           DOB                  GRADE GENDE
---------- --------------- --------------- ---------- -----
        10 Ruhi            01-Apr-2001            9.3 f
        12 Pihu            12-Mar-2000            8.6 f
        14 Siddarth        05-Feb-2000            7.3 m
        18 Manik           17-Mar-2000            6.5 m
        19 Nandini         09-Aug-2000            6.3 f
        11 Druv            22-May-2001            8.1 m


6 rows selected.



SQL> select * from proctor_students;

       PID        SID
---------- ----------
         1         12
         2         18
         3         10
         4         11
         5         19
         3         14

6 rows selected.

SQL> select * from department;

       DID DNAME                HOD
---------- -------------------- --------------------
       737 It                   K.Ram Mohan
       733 Cse                  T.Adilaxmi
       735 Ece                  E.Sreenivasa Rao
       736 Mechanical           T.Ram Mohan
       732 Civil                B.Sridhar
       734 Eee                  M.Chakravathy

6 rows selected.
```

```
SQL> select * from department_students;

       DID        SID
---------- ----------
       737         10
       733         18
       735         11
       736         14
       732         19
       734         12

6 rows selected.
```

Discussion and future work:

The application done till now is to store all the information related to the Proctorialsystem of a college . Furthermore, other programming languages can also be used along with database by connecting SQL with it. This application can be extended further more to information regarding students and their proctors and personal counsellers of many other colleges, organizations etc

CONCLUSION:

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

References:

https://www.freeprojectz.com/php-mysql-project/student-counselling-management-system

https://partheniumprojects.com/personal-counselling/