**EX.NO:** **ONLINE COURSE REGISTRATION SYSTEM**

**PROBLEM STATEMENT:**

Online Course Registration System is a Web-based registration software that helps you to register courses online. It is ideal for adult schools, educational camps, corporate training programs, and online training programs. Our online registration form allows you to submit your course preferences quickly and conveniently. Our mentored courses are designed for people who want to combine the convenience of online delivery with hand-marked assignments, personal support and feedback from expert instructors. The only constraint is that the course is offered only when there are a minimum number of people opting for it.

With online courses, you can learn on your own schedule and even continue working a full-time job in the meantime. And because all your materials will be in one place–in your computer–everything you need will be at your fingertips when it’s time to communicate with teachers or fellow students. Online courses are interactive, which means you will get feedback from your instructors and peers, and possibly even more individualized attention as well.

In our project, the student needs to register by giving the necessary details, for the desired course, and should also mention the mode of payment for the course. The student gives two choices and one among the two is provided to him based on the availability of mentors and on the number of students who have registered for that particular course. Once the registration is done successfully, the student can avail the course. The student is given an user id and password using which he can login and use the online system. He will then be taught via video conferencing using which he can learn from any place (all he has to do is just login with his username and password). Additional informative materials are also provided to the student.

**1.USE CASE DIAGRAM:**



**2.REGISTER**

**2.1 Brief Description**

 The use case describes how a User registers and logs in into the Course Registration System.

**2.2 Flow of Events**

**2.2.1 Basic Flow**

 This use case starts when the actor wishes to Register to the Course Registration System.

 1. The System requests the Student/Staff to give the needed details for registration.

 2. The Student/Staff enters his/her details.

 3. The System validates the details to check if all the mandatory fields are given.

**2.2.2 Alternative Flows**

**2.2.2.1 Mandatory fields not filled**

 If in the Basic Flow, the Student/Staff doesn’t enter all the required details then an error message is displayed and at this point the use case ends.

**2.2.2.2 Invalid User Name/Password**

 If in the Basic Flow, the user enters an invalid id or password, an error message is displayed and the use case terminates.

**2.3 Special Requirements**

 To login the Student/Staff should have already registered.

**2.4 Pre-Conditions**

None

**2.5 Post-Conditions**

 If the use case was successful, the Student/Staff registration details would have been saved in the database and they are successfully logged in into the System. If not, the System state is unaltered.

**2.6 Extension Points**

 None

**3.RESERVE**

**3.1 Brief Description**

 This use case allows a Student to register for any one of the courses offered. The course catalog, that is created and saved in the Database by the Administrator, provides a list of all the course offered.

**3.2 Flow of Events**

**3.2.1 Basic Flow**

 This use case starts when a Student wishes to register for a course.

1. The System prompts the Student to give an option that he/she would like to choose from the displayed catalog.
2. Based on the given choice, the Database is checked to see if seats are available for that course.
3. If it is available then it is displayed to the Student and the course is reserved.
4. Details regarding the payment is got from the Student and then on successful registration the Database is updated.
5. A message is displayed to he Student to indicate the successful registration.
6. If enough seats are not available the secondary choice is offered to the Student.

**3.2.2 Alternate Flow**

 None

**3.3** **Special Requirements**

 None

**3.4 Pre-Conditions**

 To view the catalogue and register the Student should be logged in into the System.

**3.5 Post-Conditions**

If the use case was successful, the Student would have registered for a course. Otherwise, the system state is unchanged.

**3.6 Extension Points**

 None

**4. GIVE PREFERENCE**

**4.1 Brief Description**

 This use case describes how a Staff gives his/her preference to teach a course and how he is able to view the details of all the Students he is teaching.

**4.2 Flow of Events**

**4.2.1. Basic Flow**

 This use case starts when a Staff who has registered wants to give his/her preference to teach a course to be taught from the catalog provided by the Administrator.

1. The System prompts the Staff to give his/her preference details.
2. The Staff enters his/her preference.
3. The preference details given by the Staff is stored in the database.
4. The System then displays the list of Students who have opted for that course along with their details.
5. The Staff teaches the Student the preferred course.

**4.2.2 Alternate Flow**

 None

**4.3 Special Requirements**

 None

**4.4 Pre-Conditions**

 The Staff should be logged in into the System and a minimum number of Students should have opted for that course that he prefers to teach.

**4.5 Post-Conditions**

 If, there are a minimum of 5 Students opting for the Staff’s preferred course, he/she starts teaching them. If not, the System state is unchanged.

**4.6 Extension Points**

 None

**5.5** **SOFTWARE REQUIREMENT SPECIFICATION :**

 **5.5.1 INTRODUCTION :**

 **5.5.1.1 PURPOSE:**

 The main purpose of this specification is to provide a clear view of the requirements of Online Course Registration System to the developer. This project aims at developing an Online system for students to register and learn courses online. The staffs provide online guidance to the students. This system is ideal for adult schools, educational camps, corporate training programs, and online training programs

 **5.5.1.2 SCOPE:**

 The Scope of the software product developed involves paid service to the students to select and learn a restricted number of courses that are being offered. It involves a very simple registration procedure to be done by the student while being guided by a mentor through the course. The course is not offered until a minimum number of students opt for it.

 **5.5.1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS:**

 OS- Operating System

 GUI-Graphical User Interface

 RAM-Random Access Memory

 MB-Mega Bytes

 GB-Giga Bytes

 HDD-Hard Disk Drive

 UML(Uniform Modeling Language):Standardized general purpose modeling language.

 **5.5.1.4 REFERENCES:**

 The books referred during predevelopment stages of project include:

* Software Engineering by Roger S.Pressman.
* Introduction to Object Oriented Analysis And Design by Craig Larman.
* www.wikepedia.com

 **5.5.1.5 OVERVIEW:**

 The rest of the Software Requirement Specification documents explains in detail the various software and hardware requirements and also elaborates on the constraints involved in the system.

 **5.5.2 OVERALL DESCRIPTION :**

 **5.5.2.1 PRODUCT PERSPECTIVE:**

 This system is useful to register for the course from the available courses in the catalog. It gives an overall description about the available courses that a student can opt to learn and a professor can prefer to teach.

* The system provides login facility to the students.
* The system checks whether the users have the required qualification to register and to view further details on the courses available.
* The system allows the Administrator to create the catalog and store it in the database.
* The system updates the payment details as and when the student reserves a course successfully.
* The course catalog is framed and the decision of offering the course, based on the preferences given by the staff and based on the number of students choosing that particular course, is decided by the administrator.

 **5.5.2.1.1 SYSTEM INTERFACE**

 This system is an independent application. It does not interface with any application.

 **5.5.2.1.2 User Interfaces:**

 A very interactive GUI is provided to the student and the staff. It prompts the actors to give the required input for each activity of the system.

 **5.5.2.1.3 Hardware Interfaces:**

 Not applicable.

 **5.5.2.1.4 Software Interfaces:**

* Windows XP
* Oracle 6
* MySQL
* Netbeans IDE

 **5.5.2.1.5 Communication Interfaces:**

* Webcam
* Microphone

 **5.5.2.1.6 Memory Constraints:**

 The software requires a disk space of 20 MB on each system it is loaded.

 **5.5.2.1.7 Operations:**

 This software is active and can be used to learn as per the staff’s schedule.

 **5.5.2.1.8 Site Adaption Requirements:**

 Not applicable.

 **5.5.2.2 Product Functions:**

 The major functions include:

* Entering the student details into the database.
* Offering the chosen (if available) course to the student.
* Receiving payment from the student.
* Getting the staff’s preference to teach a course.

 **5.5.2.3 User Characteristics:**

 The software product developed has a simple interface that all students can use and work on. The student requires no technical expertise to handle the product.

* + - 1. **Constraints:**
* Hardware limitations-there must be 64 MB on-based memory
* Product limitation- a course is offered only when a minimum number of students have opted it and also the number of students who have opted a particular course does not exceed a maximum limit.
* Control function-in case of errors and services problems, proper error handling and data recovery mechanism must be included.
* Interface to other application-not applicable.
* Parallel operation not applicable.
* Reliability requirements- data redundancy and use of special/blank characters must be avoided during registration

.

* Safety/ security consideration-the application must be exited always normally.
* Higher order language requirements – not applicable.

 **5.5.2.5 Assumptions and Dependencies:**

 The data provided by both the student and the staff is assumed to be valid and true.

**5.5.3 Specific Requirements:**

 **5.5.3.1 External Interface Requirements:**

 **5.5.3.1.1 User Interfaces:**

 The user interfaces is a typical form which includes text boxes, command buttons and label boxes that are appropriate. During any illegal access an error message appears using a message box on the screen. The user interface is designed using MS-Access.

 **5.5.3.1.2 Screen formats/ organizations:**

 The introductory screen will ask if the user is a student or a staff with the registration screen to follow. Once the registration is successful, login screen is provided followed by the course selection, reservation and payment.

 **5.5.3.1.3 Windows format/organizations:**

 Each function will lead to another window. The user can switch to the previous window whenever required.

 **5.5.3.1.4 Data format:**

 The data entered by the users will be alphanumeric.

 **5.5.3.1.5 End message:**

 When there are some errors on entering invalid data, correct data is asked via the error message else leads to termination of the software.

 **5.5.3.2 PERFORMANCE REQUIREMENTS:**

 Multiple users can accesses the software at any given time.

 **5.5.3.2.1 DESIGN CONSTRAINTS:**

 The software requires 1 GB RAM. It is based completely on Windows functionality platform. The software is portable and inaccessible to users with no valid id or to users who lack the basic qualifying criteria.

* + - * 1. **SOFTWARE SYSTEM ATTRIBUTES:**

 **RELIABILITY:**

 The factors establishing the software’s reliability are:

* The user cannot use the software with no valid id.
* The staff’s qualifications are tested and hence a reliable service is offered.

 **5.5.3.2.2.2 SECURITY:**

* Payment transactions made are highly secure.

 **5.5.3.2.2.3**   **PORTABILITY:**

 It can operate on all operating systems.

 **5.5.3.2.2.4 MANTAINABILITY:**

 Having implemented the concept of modularity it reduces the complexity involved in maintaining the software.

 **5.5.3.2.3 LOGICAL DATABASE:**

 The system requires the use of Oracle database table to hold the details of all the students and staffs.

**ACTIVITY DIAGRAM**

****

**SEQUENCE DIAGRAMS:**

**1.REGISTER:**

****

**2. RESERVE**

****

**3. GIVE PREFERENCE:**

****

**COMMUNICATION DIAGRAMS:**

1. **REGISTER**

****

1. **RESERVE**

****

**CLASS DIAGRAM:**

**REGISTER:** ****

**2. RESERVE**

****

**3. GIVE PREFERENCE**

****

**INTEGRATED CLASS DIAGRAM:**

****

**COMPONENT DIAGRAM:**

****

**DEPLOYMENT DIAGRAM:**

****

**STATE MACHINE DIAGRAMS:**

**1.REGISTER
**

**2.RESERVE**

****

**3. GIVE PREFERENCE**

****

**CODE:**

**preferenceUI.java:**

package Givepreference;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import Register.Staff;

import Register.Database;

import java.util.Set;

/\*\*

 \* @author user

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

public class PreferenceUI extends JFrame implements ActionListener{

 /\*\*

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 private Staff staff;

 String a1,a2,a3;

 JButton b=new JButton("Submit preference");

 JPanel p;

 JLabel l1,l2,l3;

 JTextField t1,t2,t3;

 /\*\*

 \* @return the staff

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public PreferenceUI()

 {

 p=new JPanel(new GridLayout(4,4));

 l1=new JLabel("Course1:");

 l2=new JLabel("course2:");

 l3=new JLabel("course3:");

 t1=new JTextField(40);

 t2=new JTextField(40);

 t3=new JTextField(40);

 p.add(l1);

 p.add(t1);

 p.add(l2);

 p.add(t2);

 p.add(l3);

 p.add(t3);

 add(p);

 add(b,BorderLayout.SOUTH);

 b.addActionListener(this);

 }

 public Staff getStaff() {

 return staff;

 }

 public void actionPerformed(ActionEvent ae)

 {

 String a1=t1.getText();

 String a2=t2.getText();

 String a3=t3.getText();

 if(a1.equalsIgnoreCase("Computer Science") || a2.equalsIgnoreCase("Maths")|| a3.equalsIgnoreCase("OOAD"))

 {

 JOptionPane.showMessageDialog(null,"Allocated successfully");

 }

 else

 {

 JOptionPane.showMessageDialog(this,"Not successfull");

 }

 }

 /\*\*

 \* @param theStaff the staff to set

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public void setStaff(Staff theStaff) {

 staff = theStaff;

 }

 /\*\*

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 private Database database;

 /\*\*

 \* @return the database

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public Database getDatabase() {

 return database;

 }

 /\*\*

 \* @param theDatabase the database to set

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public void setDatabase(Database theDatabase) {

 database = theDatabase;

 }

 /\*\*

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 private Set system;

 /\*\*

 \* @return the system

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public Set getSystem() {

 return system;

 }

 /\*\*

 \* @param theSystem the system to set

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public void setSystem(Set theSystem) {

 system = theSystem;

 }

 /\*\*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public void Providepreferencedetails() {// TODO Auto-generated method stub

 }

 public static void main(String[] args)throws Exception

 {

 PreferenceUI p=new PreferenceUI();

 p.setVisible(true);

 p.setSize(100,100);

 }

}

**ReserveUI.java:**

/\*\*

 \*

 \*/

package Reserve;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import Register.Student;

import Register.Database;

/\*\*

 \* @author user

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

public class ReserveUI extends JFrame implements ActionListener

{

 /\*\*

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 private Administrator administrator;

 int count1=3;

 int count2=3;

 int count3=3;

 JLabel c=new JLabel("SELECT A COURSE FROM 'COMBOBOX' TO RESERVE:");

 JLabel fees=new JLabel("Fees to be paid:");

 JTextField fees1=new JTextField(20);

 JButton b=new JButton("Confirm selection");

 JButton b1=new JButton("Pay fees");

 JLabel a=new JLabel("1.Enter your Account number:");

 JTextField ano=new JTextField(20);

 JLabel or=new JLabel("---------------(OR)---------------");

 JLabel ddno=new JLabel("1.Enter your DD No.:");

 JTextField DDno=new JTextField(20);

 JLabel ddto=new JLabel("2.Pay DD to:");

 JTextField DDto=new JTextField(20);

 JLabel ddpayat=new JLabel("3.Payable at:");

 JTextField DDpay=new JTextField(20);

 JPanel p=new JPanel();

 JFrame f=new JFrame();

 JPanel p1=new JPanel();

 JPanel p2=new JPanel();

 CheckboxGroup grp=new CheckboxGroup();

 //Checkbox c1=new Checkbox("Computer Science",grp,false);

 //Checkbox c2=new Checkbox("maths",grp,false);

 //Checkbox c3=new Checkbox("OOAD",grp,false);

 JComboBox combo=new JComboBox();

 public ReserveUI()

 {

 fees1.setText("Rs.7500");

 fees1.setEditable(false);

 combo.addItem("Computer Science");

 combo.addItem("Maths");

 combo.addItem("OOAD");

 p.add(c);

 p.add(combo);

 //p.add(c);

 //p.add(c1);

 //c1.addItemListener(this);

 //p.add(c2);

 //c2.addItemListener(this);

 //p.add(c3);

 //c3.addItemListener(this);

 add(p);

 add(BorderLayout.SOUTH,b);

 b.addActionListener(this);

 b1.addActionListener(this);

 p1.setLayout(new GridLayout(3,2,5,5));

 p1.add(fees);

 p1.add(fees1);

 p1.add(a);

 p1.add(ano);

 p1.add(or);

 p2.setLayout(new GridLayout(3,3,5,5));

 p2.add(ddno);

 p2.add(DDno);

 p2.add(ddto);

 p2.add(DDto);

 p2.add(ddpayat);

 p2.add(DDpay);

 }

 public void actionPerformed(ActionEvent e)

 {

 if(e.getSource()==b)

 {

 if(combo.getSelectedItem().equals("Computer Science"))

 {

 if(count1!=0)

 {

 JOptionPane.showMessageDialog(this, "Enter Account number (OR) DD details for payment:");

 JFrame f1=new JFrame();

 f1.add(BorderLayout.WEST,p1);

 f1.add(BorderLayout.EAST,p2);

 f1.add(BorderLayout.SOUTH,b1);

 f1.setVisible(true);

 count1--;

 }

 else

 JOptionPane.showMessageDialog(this, "Oops!! All seats for Computer Science course had been already reserved.\nTry some other course");

 }

 else if(combo.getSelectedItem().equals("Maths"))

 {

 if(count1!=0)

 {

 JOptionPane.showMessageDialog(this, "Enter Account number (OR) DD details for payment:");

 JFrame f1=new JFrame();

 f1.add(BorderLayout.WEST,p1);

 f1.add(BorderLayout.EAST,p2);

 f1.add(BorderLayout.SOUTH,b1);

 f1.setVisible(true);

 count1--;

 }

 else

 JOptionPane.showMessageDialog(this, "Oops!! All seats for Maths course had been already reserved.\nTry some other course");

 }

 if(combo.getSelectedItem().equals("OOAD"))

 {

 if(count1!=0)

 {

 JOptionPane.showMessageDialog(this, "NOTE: Enter either Account number (OR) details of DD for payment:\nDon't fill both");

 //setVisible(false);

 JFrame f1=new JFrame();

 f1.add(BorderLayout.WEST,p1);

 f1.add(BorderLayout.EAST,p2);

 f1.add(BorderLayout.SOUTH,b1);

 f1.setVisible(true);

 count1--;

 }

 else

 JOptionPane.showMessageDialog(this, "Oops!! All seats for OOAD course had been already reserved.\nTry some other course");

 }

 }

 else if(e.getSource()==b1)

 {

 if(!ano.getText().equals(""))

 JOptionPane.showMessageDialog(this,"Amount paid successfully via account number:"+ano.getText());

 else

 JOptionPane.showMessageDialog(this,"Amount paid successfully via Demand Draft number:"+DDno.getText());

 }

 }

 /\*\*

 \* @return the administrator

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public Administrator getAdministrator() {

 return administrator;

 }

 /\*\*

 \* @param theAdministrator the administrator to set

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public void setAdministrator(Administrator theAdministrator) {

 administrator = theAdministrator;

 }

 /\*\*

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 private Student student;

 /\*\*

 \* @return the student

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public Student getStudent() {

 return student;

 }

 /\*\*

 \* @param theStudent the student to set

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public void setStudent(Student theStudent) {

 student = theStudent;

 }

 /\*\*

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 private Database database;

 /\*\*

 \* @return the database

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 /\*\*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public static void main(String[] args)throws Exception

 {

 ReserveUI r=new ReserveUI();

 r.setVisible(true);

 }

}

**RegisterUI.java:**

/\*\*

 \*

 \*/

package Register;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

/\*\*

 \* @author user

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

public class RegisterUI extends JFrame implements ActionListener{

 /\*\*

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 private Student student;

 String[] studname =new String[10];int ucount=0;

 String[] studpass =new String[10];

 String[] studqualifi=new String[10];

 String[] staffname=new String[10];int u1count=0;

 String[] staffpass=new String[10];

 String[] staffqualifi=new String[10];

 JPanel button = new JPanel(new GridLayout(4,0,3,3));

 JPanel staffpan=new JPanel(new GridLayout(2,0,3,3));

 JLabel l=new JLabel("CLICK HERE FOR REGISTRATION OR LOGIN");

 JLabel l2=new JLabel("Login successful");

 JButton b1= new JButton("Students enter here!!!");

 JButton b2=new JButton("Staffs enter here!!!");

 JButton srb=new JButton("Students' register page");

 JButton strb=new JButton("Staffs' register page");

 JButton slb=new JButton("Students' login page");

 JButton stlb=new JButton("Staffs' login page");

 JPanel studreg=new JPanel(new GridLayout(2,0,5,5));

 //JLabel l1=new JLabel("how");

 JFrame f=new JFrame();

 JFrame f1=new JFrame();

 JPanel regpan=new JPanel();

 JPanel loginpan=new JPanel();

 JFrame f2=new JFrame();

 JFrame f3=new JFrame();

 JFrame f4=new JFrame();

 JFrame f5=new JFrame();

 JLabel u=new JLabel("Username:");

 JTextField u1=new JTextField(20);

 JLabel p=new JLabel("Password:");

 JPasswordField p1=new JPasswordField(20);

 JLabel q=new JLabel("Qualification:");

 JTextField q1=new JTextField(20);

 JPanel staffregpan=new JPanel();

 JButton submit=new JButton("Submit details");

 JButton submit1=new JButton("Submit details");

 JButton login=new JButton("Login");

 JButton login1=new JButton("Login");

 JLabel u2=new JLabel("Username:");

 JTextField u3=new JTextField(20);

 JLabel p2=new JLabel("Password:");

 JPasswordField p3=new JPasswordField(20);

 JPanel staffreg=new JPanel();

 JButton getdetails=new JButton("Get details");

 String Totallist=new String();

public RegisterUI()

{

 button.add(l);

 button.add(b1);

 b1.addActionListener(this);

 button.add(b2);

 b2.addActionListener(this);

 add(BorderLayout.CENTER,button);

 srb.addActionListener(this);

 strb.addActionListener(this);

 slb.addActionListener(this);

 stlb.addActionListener(this);

 regpan.setLayout(new GridLayout(3,2));

 regpan.add(u);

 regpan.add(u1);

 regpan.add(p);

 regpan.add(p1);

 regpan.add(q);

 regpan.add(q1);

 loginpan.add(u2);

 loginpan.add(u3);

 loginpan.add(p2);

 loginpan.add(p3);

 login.addActionListener(this);

 submit.addActionListener(this);

 studreg.add(srb);

 studreg.add(slb);

 staffreg.setLayout(new GridLayout(2,0));

 staffreg.add(strb);

 staffreg.add(stlb);

 staffpan.add(l2);

 staffpan.add(getdetails);

 getdetails.addActionListener(this);

 login1.addActionListener(this);

 submit1.addActionListener(this);

}

 /\*\*

 \* @return the student

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public Student getStudent() {

 return student;

 }

 /\*\*

 \* @param theStudent the student to set

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public void setStudent(Student theStudent) {

 student = theStudent;

 }

 /\*\*

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 private Database database;

 /\*\*

 \* @return the database

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public Database getDatabase() {

 return database;

 }

 /\*\*

 \* @param theDatabase the database to set

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public void setDatabase(Database theDatabase) {

 database = theDatabase;

 }

 /\*\*

 \*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 private Staff staff;

 /\*\*

 \* @return the staff

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public Staff getStaff() {

 return staff;

 }

 /\*\*

 \* @param theStaff the staff to set

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public void setStaff(Staff theStaff) {

 staff = theStaff;

 }

 /\*\*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public void Givedetails(String checkuser)//student's registration page

 {// TODO Auto-generated method stub

 if(checkuser.equalsIgnoreCase("Student"))

 {

 int flag=0;

 if(q1.getText().equalsIgnoreCase("12th") || q1.getText().equalsIgnoreCase("B.E.") || q1.getText().equalsIgnoreCase("B.Tech."))

 {

 if(!(u1.getText().equals("")|| p1.getText().equals("")))

 {

 for(int i=0;i<studname.length;i++)

 {

 if(studname.equals(u1.getText()))

 {

 flag=1;

 JOptionPane.showMessageDialog(this, "username already exists.Try another User name");

 }

 }

 if(flag==0)

 {

 studname[ucount]=u1.getText();

 studpass[ucount]=p1.getText();

 studqualifi[ucount]=q1.getText();

 JOptionPane.showMessageDialog(this, "Username created sccessfully.\nLogin ID:"+studname[ucount]+"\nPassword:"+studpass[ucount]);

 ucount++;

 //pcount++;

 //qcount++;

 f1.setVisible(false);

 setVisible(true);

 }

 }

 else

 {

 JOptionPane.showMessageDialog(this, "Error: Either Username or password is blank");

 }

 }

 else

 {

 JOptionPane.showMessageDialog(this,"Sorry!!! You should have completed atleast 12th/B.E./B.Tech. to register");

 }

 }

 }

 public void Givedetails()

 {

 // else if(checkuser.equalsIgnoreCase("Staff"))

 //{

 int flag=0;

 if(q1.getText().equalsIgnoreCase("B.E.") || q1.getText().equalsIgnoreCase("M.E.") || q1.getText().equalsIgnoreCase("B.Tech.") || q1.getText().equalsIgnoreCase("Ph.D."))

 {

 if(!(u1.getText().equals("")|| p1.getText().equals("")))

 {

 for(int i=0;i<studname.length;i++)

 {

 if(staffname.equals(u1.getText()))

 {

 flag=1;

 JOptionPane.showMessageDialog(this, "username already exists.Try another User name");

 }

 }

 if(flag==0)

 {

 staffname[u1count]=u1.getText();

 staffpass[u1count]=p1.getText();

 staffqualifi[u1count]=q1.getText();

 JOptionPane.showMessageDialog(this, "Username created sccessfully.\nLogin ID:"+studname[ucount]+"\nPassword:"+studpass[ucount]);

 u1count++;

 //pcount++;

 //qcount++;

 f1.setVisible(false);

 setVisible(true);

 }

 }

 else

 {

 JOptionPane.showMessageDialog(this, "Error: Either Username or password is blank");

 }

 }

 else

 {

 JOptionPane.showMessageDialog(this,"Sorry!!! You should have completed atleast B.E./B.Tech./M.E./Ph.D. to register");

 }

 }

 /\*\*

 \* @generated "UML to Java V1.4 (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"

 \*/

 public void Logindetails()//staff's login page

 {

 int flag=0;

 for(int i=0;i<studname.length;i++)

 {

 if(staffname[i].equals(u3.getText()) && staffpass[i].equals(p3.getText()))

 {

 JOptionPane.showMessageDialog(this, "Login successful");

 JFrame f5=new JFrame();

 f5.add(staffpan);

 f5.setVisible(true);

 flag=1;

 }

 }

 if(flag==0)

 JOptionPane.showMessageDialog(this,"OOPS.. Wrong username or password");

 }

public void actionPerformed(ActionEvent e)

{

if(e.getSource()==b1)//students page

{

 setVisible(false);

 f.add(studreg);

 //f.add(studlogin);

 f.setVisible(true);

 //JLabel l=new JLabel("how");

 //f.add(l);

}

else if(e.getSource()==b2)

{

 setVisible(false);

 f3.add(BorderLayout.CENTER,staffreg);

 f3.setVisible(true);

}

else if(e.getSource()==srb)

{

 f.setVisible(false);

 f1.add(regpan);

 f1.add(BorderLayout.SOUTH,submit);

 f1.setVisible(true);

}

else if(e.getSource()==strb)

{

 f3.setVisible(false);

 //f4.add(staffregpan);

 f4.add(regpan);

 f4.add(BorderLayout.SOUTH,submit1);

 f4.setVisible(true);

}

else if(e.getSource()==slb)

{

 f1.setVisible(false);

 f.setVisible(false);

 f2.add(loginpan);

 f2.add(BorderLayout.SOUTH,login);

 f2.setVisible(true);

}

else if(e.getSource()==submit)

{

 Givedetails("student");

 f1.setVisible(false);

 f.setVisible(true);

}

else if(e.getSource()==submit1)

{

 //Givedetails("Staff");

 Givedetails();

 f4.setVisible(false);

 setVisible(true);

}

else if(e.getSource()==login)

{

 int flag=0;

 for(int i=0;i<studname.length;i++)

 {

 if(studname[i].equals(u3.getText()) && studpass[i].equals(p3.getText()))

 {

 JOptionPane.showMessageDialog(this, "Login successful");

 flag=1;

 }

 }

if(flag==0)

{JOptionPane.showMessageDialog(this, "Oops!! the user is not registered");}

}

else if(e.getSource()==login1)

{

 JOptionPane.showMessageDialog(this, "Login successful");

 //Logindetails();

}

else if(e.getSource()==getdetails)

{

 Totallist+="List of students are:";

 for(int i=0;i<studname.length;i++)

 {

 Totallist+=studname[i]+"\t"+studqualifi[i]+"\n";

 }

 JOptionPane.showMessageDialog(this,Totallist);

}

}

public static void main(String[] args) throws Exception

{

 RegisterUI r=new RegisterUI();

 r.setVisible(true);

}

}

**OUTPUT:**

**RegisterUI.java:**

**Main frame**

****

**On clicking Students enter here!!! Button in the frame:**

****

**On clicking “students’ register page” button:**

1. **Frame showing Successful registration:**

****

1. **Frame showing Unsuccessful registration:**

****

**Login frame:**

**a)Frame showing Successful login:**

****

**b)Frame showing Unsuccessful login:**



1. **Frame showing the student details to staff after successful login of staff:**

****

**ReserveUI.java:**

**Frame asking to select a course from the combo box:**

****

****

**Message box showing a message before going to the next frame:**

****

**Frame asking to pay course fee via account number/DD.**

1. **When account number is entered:**

****

1. **When DD details are entered:**

****

**PreferenceUI.java:**

**Frame showing Successful allocation of Course:**

****

**Frame showing unsuccessful allocation of Course:**

****