Development of **Faculty Batch Report System**

for an Organisation / Institute

**A Project Report for Summer Industrial Training**

***Submitted by***

*Rajeev Ranjan*

*Siddharth Sanghvi*

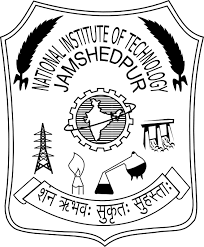
*Saurabh Tripathi*

*Prasanna Gorai*

*of*

*Computer Science and Engineering Dept.*

**NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR**



**At**

**Ardent Computech Pvt. Ltd.**



**May 2015**

**Ardent Computech Pvt. Ltd.**



**BONAFIDE CERTIFICATE**

Certified that this project work was carried out under my supervision

***“***Development of **Faculty Batch Report System** for an Organisation / Institute”

is the bonafide work of

***Name of the student: Rajeev Ranjan Signature:***

***Name of the student: Siddharth Sanghvi Signature:***

***Name of the student: Saurabh Tripathi Signature:***

***Name of the student: Prasanna Gorai Signature:***

**SIGNATURE**

Sumanta Roy

**PROJECT MENTOR**

**SIGNATURE**

**Name:**

**EXAMINERS**

**Acknowledgement**

We take this opportunity to express our deep gratitude and sincerest thank to our project mentor, Mr. Sumanta Roy, for giving most valuable suggestions, helpful guidance and encouragement in the execution of this project work. We are also grateful to all other faculty members of Ardent Computech Pvt. Ltd. for their support and encouragement.

Siddharth Sanghvi

Rajeev Ranjan

Prasanna Gorai

Saurabh Tripathi

|  |  |
| --- | --- |
| **Table of Contents** | **Page No** |
| * Abstract |  |
| * Introduction and Objectives of the Project |  |
| * Project Category |  |
| * Tools/Platform, Hardware and Software Requirement specifications |  |
| * Goals of Implementation |  |
| * Use Case Diagram |  |
| * Use Case Description |  |
| * Feasibility Study |  |
| * Data Flow Diagram (DFD) |  |
| * User Interface Design |  |
| * Coding |  |
| * Testing |  |
| * System Security measures |  |
| * Database/Data security |  |
| * Creation of User profiles and access rights |  |
| * Cost Estimation of the Project along with Cost Estimation Model |  |
| * Future scope and further enhancement of the Project |  |
| * Bibliography |  |

**ABSTRACT**

This project includes development of a database system for managing the faculty data within an institute or an organization. The system as such as it has been developed is called **Faculty Batch Report System**. It consists of functionally related GUI (application program) and database.

**Introduction and Objectives of the Project**

* A faculty should be able to
* Create a new account
* a new account
* Login to his account
* Create new batch report
* View batches under him
* Maintain batch records
* Update his details
* Log out Successfully
* An admin should be able to
* Have access to the database
* Change or delete records as and when required
* Create tables as required

**Project Category**

Stand Alone Application

**Tools/Platform, Hardware and Software Requirement**

**Tools**

* Eclipse
* Oracle
* Java

**Platform**

Microsoft Windows 8/8.1/7

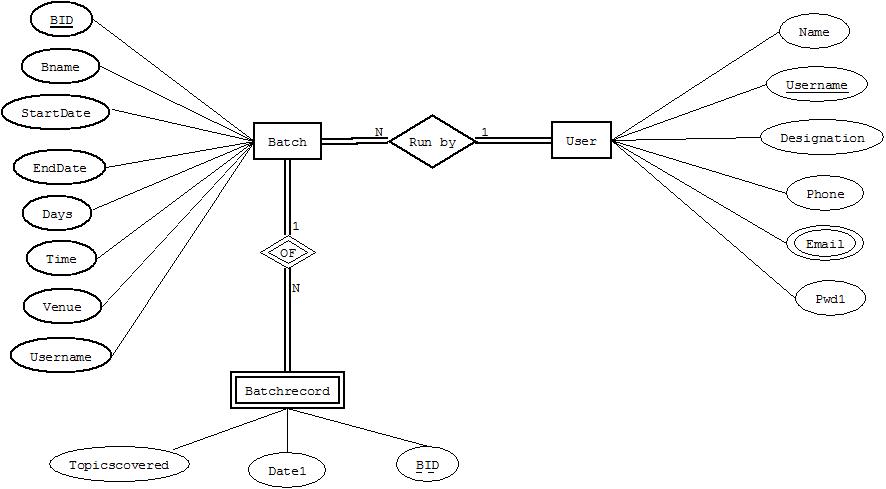
**Hardware Requirement Specification**

|  |  |
| --- | --- |
| **Client Machine** | |
| **HDD** | 200 MB |
| **Processor** | Pentium 4 or newer processor that supports SSE2 |
| **Memory** | 512 MB |

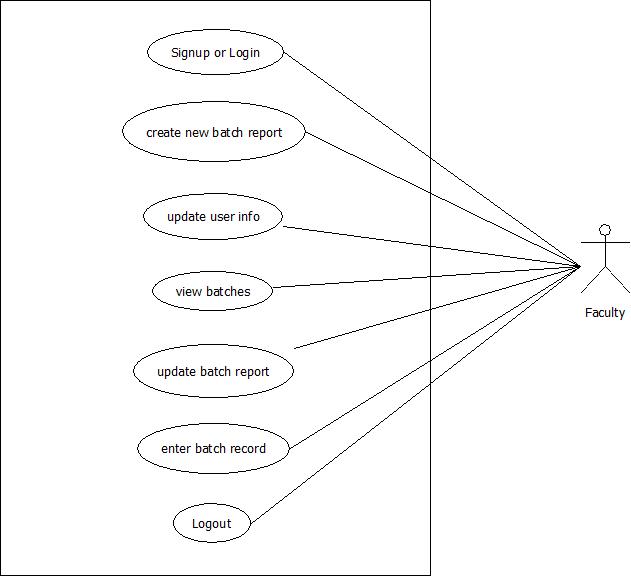
**Goals of Implementation**

The implementation aims at recording details of the batches of any registered faculty.

**ER DIAGRAM**



**USE CASE DIAGRAM**

****

**USE CASE DESCRIPTION**

|  |  |
| --- | --- |
| Use Case Name | Sign Up or Login |
| Priority | Essential |
| Precondition | User has access to the home page |
| Basic Path | * A new user fills up all the necessary details and registers * A registered user logins with his username and password * The system checks if the user is genuine and grants/denies access |
| Post Condition | The user is on the Options Page |

|  |  |
| --- | --- |
| Use Case Name | Create New Batch Report |
| Priority | Optional |
| Precondition | User must have been successfully logged in |
| Basic Path | * User fills up the necessary details for the new batch to be created * The new batch gets added in the database on pressing the ‘Add Batch’ button |
| Post Condition | New batch is added to the database |

|  |  |
| --- | --- |
| Use Case Name | Update User Info |
| Priority | Optional |
| Precondition | User must have been successfully logged in |
| Basic Path | * User updates any changes if required * Data in database gets updated on pressing ‘Update’ button |
| Post Condition | User’s data is updated in database |

|  |  |
| --- | --- |
| Use Case Name | View Batches |
| Priority | Optional |
| Precondition | User must have been successfully logged in |
| Basic Path | * On pressing the ‘View Batches’ button, all the current batches under the user are displayed |
| Post Condition | Record of the current batches is displayed |

|  |  |
| --- | --- |
| Use Case Name | Update Batch Report |
| Priority | Optional |
| Precondition | User must have been successfully logged in |
| Basic Path | * User enters the Batch ID of the batch whose record is to be updated * Required changes are made in the Batch Details * Database records the made changes when ‘Update Batch’ is pressed |
| Post Condition | All the changes made are recorded |

|  |  |
| --- | --- |
| Use Case Name | Enter Batch Record |
| Priority | Optional |
| Precondition | User must have been successfully logged in |
| Basic Path | * User enters the batch whose record is to be updated * Batch record details are changed |
| Post Condition | All the changes made are recorded in the database |

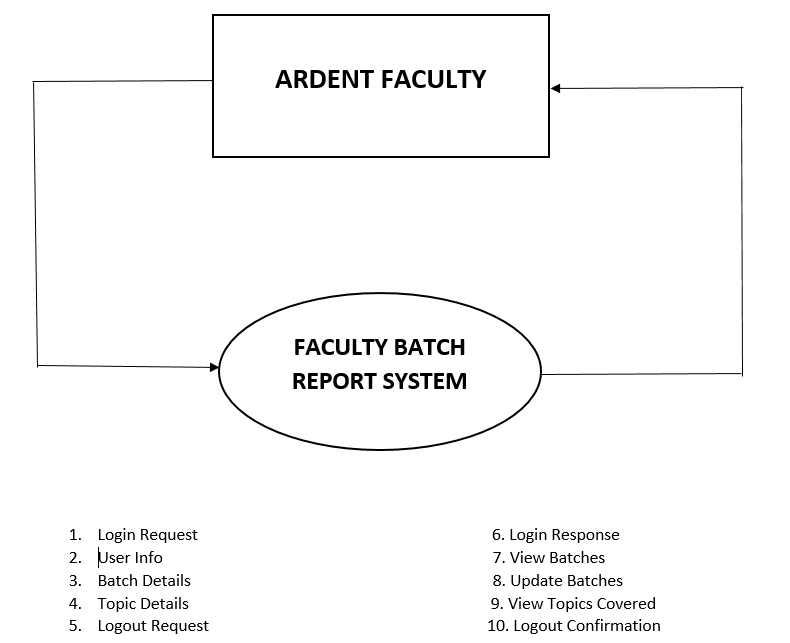
|  |  |
| --- | --- |
| Use Case Name | Log Out |
| Priority | Essential |
| Precondition | User must have been successfully logged in before |
| Basic Path | * User presses the ‘Logout’ button |
| Post Condition | The user is logged out from his account back to the login page |

**FEASIBILITY STUDY**

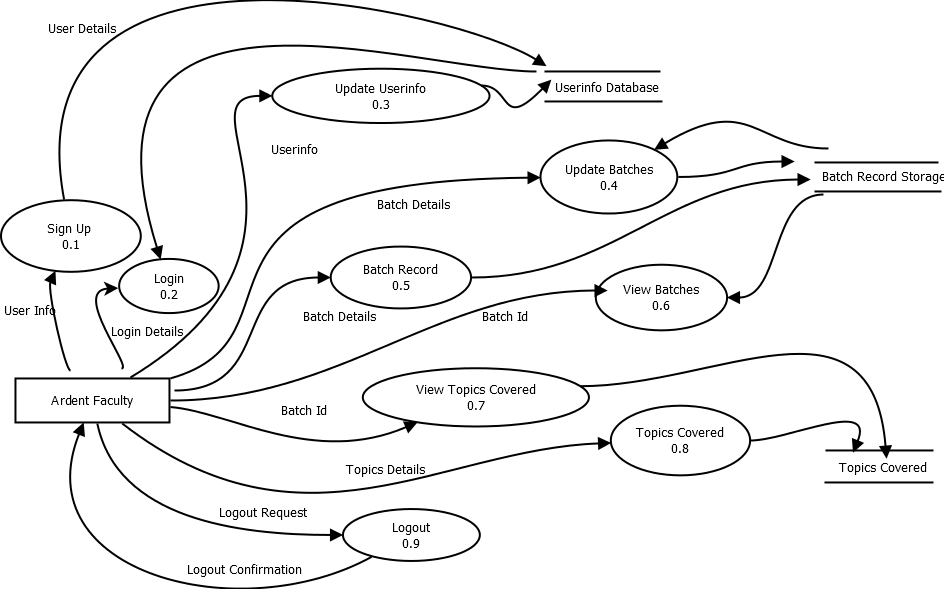
|  |  |
| --- | --- |
| **Feasibility Study** | |
| System: Record Management System | Date: 28/05/2015 |
| Author: Rajeev Ranjan / Siddharth Sanghvi | Page: 1 |
| **Product** | |
| The project requires an application to be developed that will allow any faculty of an organization to record his details as well as the details of the batches under him | |
| **Market Research** | |
| Market research says that this application would be useful for any institute which intends to record its faculty details as well as the batch details. | |
| **Economic Feasibility** | |
| The application can be developed within budget. | |
| **Alternate Solution** | |
| It may be a browser application as well, but that is not needed moreover requiring extra overheads. | |

**DATA FLOW DIAGRAMS**

Level 0

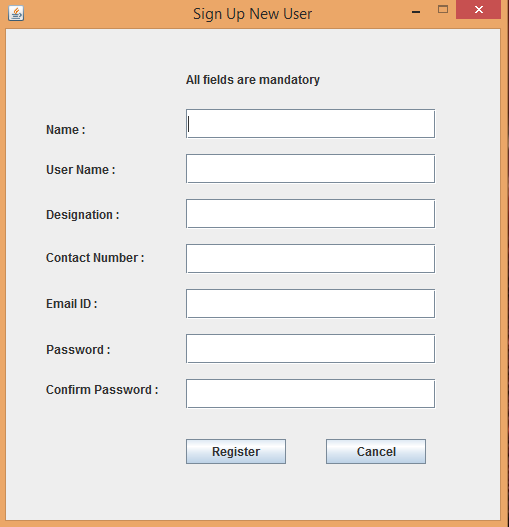
****

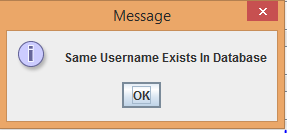
Level 1

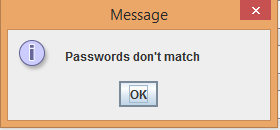
****

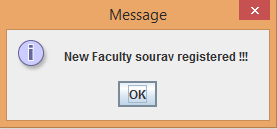
**USER INTERFACE DESIGN**

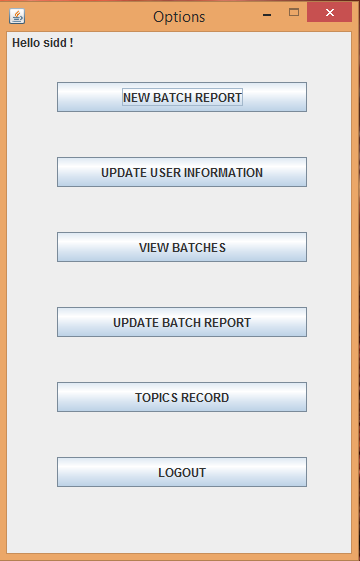
****

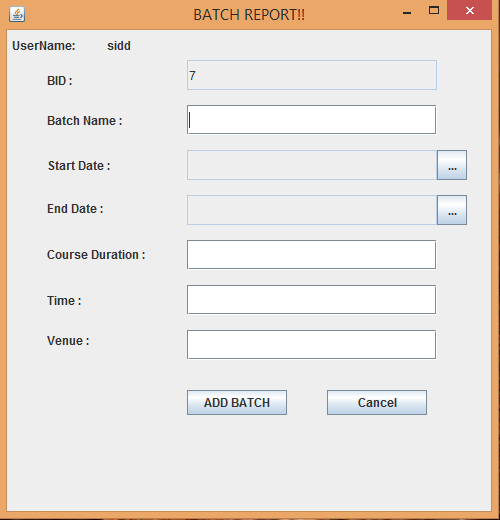
****

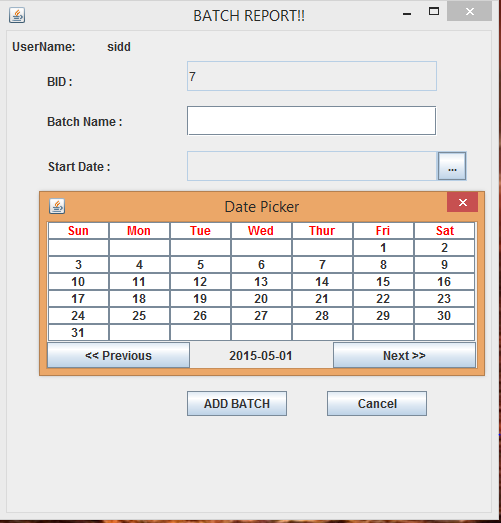
****

****

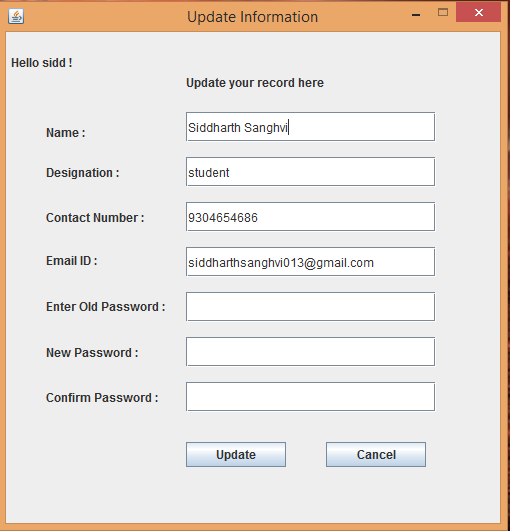
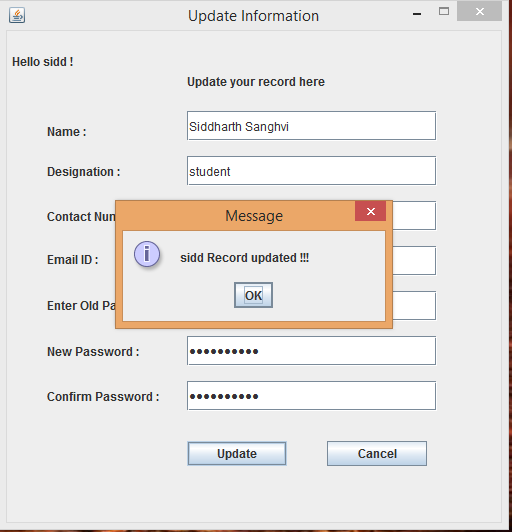
****

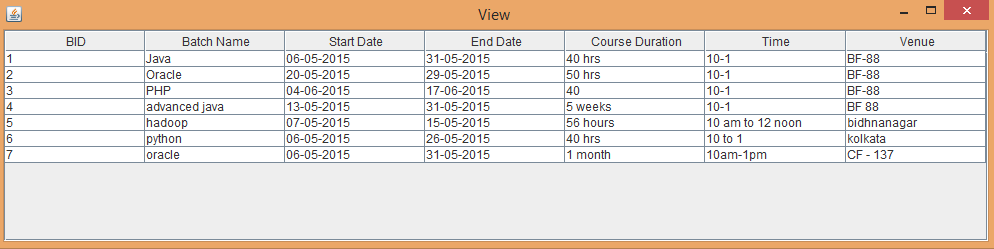
****

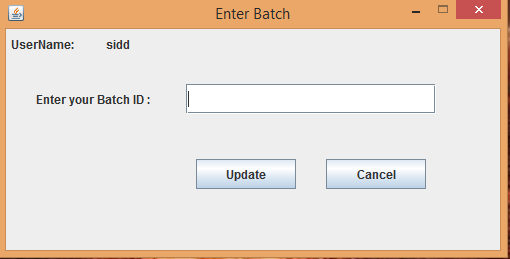
****

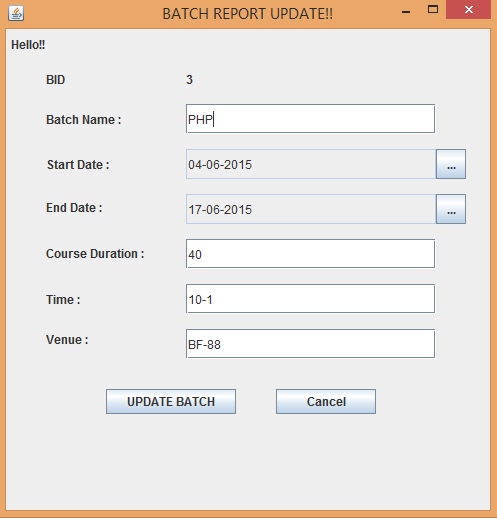
****

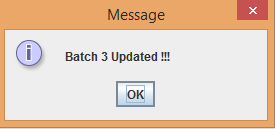
****

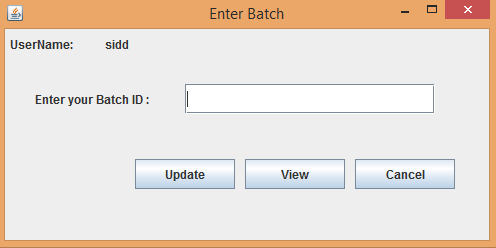
** **

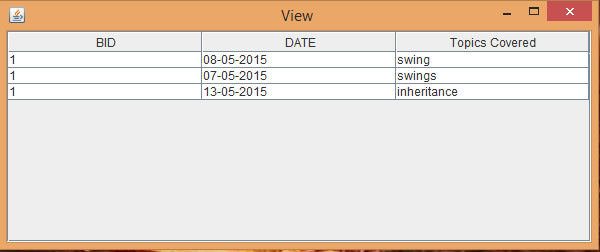
****

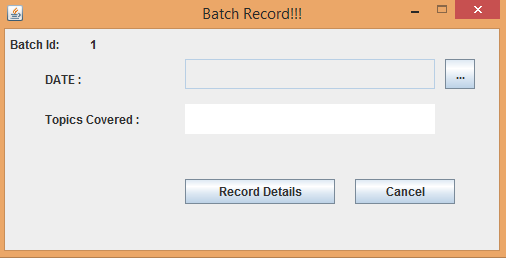
****

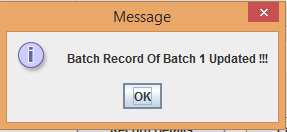
****

****

****

****

****

****

**SAMPLE CODE**

Provider.java

package Project1;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class Provider {

static Connection cn;

public static Connection getConn() throws SQLException{

cn=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","scott","tiger");

return cn;

}

static{

try {

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

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

LoginPage.java

package Project1;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

import javax.swing.\*;

public class LoginPage extends JFrame implements ActionListener {

JPanel p;

JLabel l1,l2;

JTextField t1;

JPasswordField pwd;

JButton b1,b2,b3;

public LoginPage(){

setSize(500,250);

//setResizable(false);

setTitle("Login or Register New Faculty");

setLocation(400,250);

setResizable(false);

p=new JPanel();

p.setLayout(null);

l1=new JLabel("Enter User Name : ");

l1.setBounds(40, 10, 250, 80);

p.add(l1);

t1=new JTextField();

t1.setBounds(180,30,250,30);

p.add(t1);

l2=new JLabel("Enter Password : ");

l2.setBounds(40,50,250,80);

p.add(l2);

pwd=new JPasswordField();

pwd.setBounds(180,75,250,30);

p.add(pwd);

b1=new JButton("Login");

b1.setBounds(100,140,100,25);

b1.addActionListener(this);

p.add(b1);

b2=new JButton("Sign Up");

b2.setBounds(210,140,100,25);

b2.addActionListener(this);

p.add(b2);

b3=new JButton("Exit");

b3.setBounds(320,140,100,25);

b3.addActionListener(this);

p.add(b3);

add(p);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

public void actionPerformed(ActionEvent ae) {

if(ae.getActionCommand().equals("Login")){

if((t1.getText().trim()).equals("") || pwd.getText().equals(""))

JOptionPane.showMessageDialog(null, "Field cannot be left empty");

else{

try {

Connection con=Provider.getConn();

Statement stmt=con.createStatement();

String str=t1.getText();

ResultSet rs=stmt.executeQuery("select \* from userinfo");

int count=0;

try{

while(rs.next()){

if(rs.getString("username").equals(t1.getText())){

if(rs.getString("pwd1").equals(pwd.getText())){

//set the values in faculty object

Faculty fc=new Faculty();

fc.setName(rs.getString("name"));

fc.setUsername(rs.getString("username"));

fc.setDesignation(rs.getString("designation"));

fc.setPhone(rs.getString("phone"));

fc.setEmail(rs.getString("email"));

fc.setPwd1(rs.getString("pwd1"));

Options ob=new Options(this,fc);

this.setVisible(false);

ob.setVisible(true);

count=1;

break;

}

else

{

JOptionPane.showMessageDialog(this, "Password does not match");

pwd.setText("");

count=1;

break;

}

}

}

if(count==0){

JOptionPane.showMessageDialog(this, "Access Denied");

pwd.setText("");

t1.setText("");

}

}

catch(NullPointerException np){

JOptionPane.showMessageDialog(this, "Access Denied");

pwd.setText("");

t1.setText("");

}

con.close();

}

catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

else if(ae.getActionCommand().equals("Sign Up")){

SignUp su=new SignUp(this);

this.setVisible(false);

su.setVisible(true);

}

else

{

System.exit(0);

}

}

public static void main(String[] args) {

new LoginPage();

}

}

SignUp.java

package Project1;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

public class SignUp extends JFrame implements ActionListener {

JPanel p;

JLabel l1,l2,l3,l4,l5,l6,l7,l8;

JTextField t1,t2,t3,t4,t5;

JPasswordField pwd1,pwd2;

JButton b1,b2;

LoginPage lp;

JOptionPane jop;

public SignUp(LoginPage lp){

this.lp=lp;

setLocation(420,100);

setSize(500,520);

setResizable(false);

setTitle("Sign Up New User");

p=new JPanel();

p.setLayout(null);

l8=new JLabel("All fields are mandatory");

l8.setBounds(180,40, 150,20);

p.add(l8);

l1=new JLabel("Name : ");

l1.setBounds(40, 60, 250, 80);

p.add(l1);

t1=new JTextField();

t1.setBounds(180,80,250,30);

p.add(t1);

l2=new JLabel("User Name : ");

l2.setBounds(40,100,250,80);

p.add(l2);

t2=new JTextField();

t2.setBounds(180,125,250,30);

p.add(t2);

l3=new JLabel("Designation : ");

l3.setBounds(40,145,250,80);

p.add(l3);

t3=new JTextField();

t3.setBounds(180,170,250,30);

p.add(t3);

l4=new JLabel("Contact Number : ");

l4.setBounds(40,188,250,80);

p.add(l4);

t4=new JTextField();

t4.setBounds(180,215,250,30);

p.add(t4);

l5=new JLabel("Email ID : ");

l5.setBounds(40,234,250,80);

p.add(l5);

t5=new JTextField();

t5.setBounds(180,260,250,30);

p.add(t5);

l6=new JLabel("Password : ");

l6.setBounds(40,280,250,80);

p.add(l6);

pwd1=new JPasswordField();

pwd1.setBounds(180,305,250,30);

p.add(pwd1);

l7=new JLabel("Confirm Password : ");

l7.setBounds(40,320,250,80);

p.add(l7);

pwd2=new JPasswordField();

pwd2.setBounds(180,350,250,30);

p.add(pwd2);

b1=new JButton("Register");

b1.addActionListener(this);

b1.setBounds(180,410,100,25);

p.add(b1);

b2=new JButton("Cancel");

b2.addActionListener(this);

b2.setBounds(320,410,100,25);

p.add(b2);

add(p);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

public void actionPerformed(ActionEvent e) {

String st=t5.getText();

int a=st.indexOf("@");

int b=st.lastIndexOf(".");

int len=st.length();

long n11;

if(e.getActionCommand().equals("Register")){ //if we click on register button

try

{

n11=Long.parseLong(t4.getText());

}

catch(NumberFormatException ae){

jop.showMessageDialog(null,"Enter Integer Contact no");

t4.setText("0");

}

//n11=Integer.parseInt(t4.getText());

n11=Long.parseLong(t4.getText());

if((t1.getText().trim()).equals("") || pwd1.getText().equals("") ||

(t2.getText().trim()).equals("") || (t3.getText().trim()).equals("") ||

(t4.getText().trim()).equals("") || (t5.getText().trim()).equals("") || pwd2.getText().equals("") )

{

JOptionPane.showMessageDialog(this, "Field cannot be left empty");

}

else if(!(pwd1.getText().equals(pwd2.getText()))){

JOptionPane.showMessageDialog(this, "Passwords don't match");

pwd1.setText("");

pwd2.setText("");

}

else if(pwd1.getText().length()<8)

{

JOptionPane.showMessageDialog(this, "Password Must be Atleast 8 Characters Long");

pwd1.setText("");

pwd2.setText("");

}

else if(b-a<=1||b==len-1){ // check format of email ID

JOptionPane.showMessageDialog(this, "'Enter Email ID in the format abc@xyz.com'");

t5.setText("");

}

else if(n11!=0){

try {

Connection con=Project1.Provider.getConn();

PreparedStatement ps1=con.prepareStatement("Select name from userinfo where username=?");

ps1.setString(1,t2.getText());

int n1=ps1.executeUpdate();

if(n1>0){

JOptionPane.showMessageDialog(this, "Same Username Exists In Database");

t2.setText("");

pwd1.setText("");

pwd2.setText("");

}

else{

PreparedStatement ps=con.prepareStatement("insert into userinfo(name,username,designation,phone,email,pwd1)values(?,?,?,?,?,?)");

ps.setString(1, t1.getText());

ps.setString(2, t2.getText());

ps.setString(3,t3.getText());

ps.setString(4,t4.getText());

ps.setString(5,t5.getText());

ps.setString(6,pwd1.getText());

int n=ps.executeUpdate();

if(n>0){

JOptionPane.showMessageDialog(this,"New Faculty "+t2.getText()+" registered !!!");

}

this.dispose();

lp.setVisible(true);

}

/\*

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select \* from student");

while(rs.next()){

System.out.println(rs.getString("name")+" "+rs.getString("password")+" "+rs.getString("address")+rs.getInt("phone"));

}\*/

con.close();

} catch (SQLException ae) {

// TODO Auto-generated catch block

ae.printStackTrace();

}

}

}

else{ // if we click on cancel

this.dispose();

lp.setVisible(true);

}

}

}

**TESTING**

**Team Interaction**

The following describes the level of team interaction necessary to have a successful product.

* The Test Team will work closely with the Development Team to achieve a high quality design and user interface specifications based on customer requirements. The Test Team is responsible for visualizing test cases and raising quality issues and concerns during meetings to address issues early enough in the development cycle.
* The Test Team will work closely with Development Team to determine whether or not the application meets standards for completeness. If an area is not acceptable for testing, the code complete date will be pushed out, giving the developers additional time to stabilize the area.
* Since the application interacts with a back-end system component, the Test Team will need to include a plan for integration testing. Integration testing must be executed successfully prior to system testing.

**Test Objective**

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. We will be testing a Binary Search Tree Application utilizing a pre-order traversal format. There will be eight key functions used to manage our application: load, store, clear, search, insert, delete, list in ascending order, and list in descending order. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

**Process Overview**

The following represents the overall flow of the testing process:

* The requirements to be tested are identified. All test cases shall be derived using the current Program Specification.
* Which particular test(s) will be used to test each module is/are identified.
* The test data and test cases are reviewed to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
* The expected results for each test are identified.
* The test case configuration, test data, and expected results are documented.
* The test(s) is/are performed.
* The test data, test cases, and test configuration used during the testing process are documented. This information shall be submitted via the Unit/System Test Report (STR).
* Successful unit testing is required before the unit is eligible for component integration/system testing.
* Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.
* Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

**Testing Process**

**a.** Organize Project

**b.** Design System Test

**c.** Design/Build Test Proc.

**d.** Build Test Environment

**e.** Design/Build Test Proc.

**f.** Signoff

The diagram above outlines the Test Process approach that will be followed.

1. **Organize Project** involves creating a System Test Plan, Schedule & Test Approach, and assigning responsibilities.

**b.** **Design/Build System Test** involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions/expected results will be identified by the Test Team in conjunction with the Development Team. The Test Team will then identify Test Cases and the Data required. The Test conditions are derived from the Program Specifications Document.

**c.** **Design/Build Test Procedures** includes setting up procedures such as Error Management systems and Status reporting.

**d.** **Build Test Environment** includes requesting/building hardware, software and data set-ups.

**e. Execute System Tests –** The tests identified in the Design/Build Test Procedures will be executed. All results will be documented and Bug Report Forms filled out and given to the Development Team as necessary.

**f.** **Signoff** - Signoff happens when all pre-defined exit criteria have been achieved.

**Testing Strategy**

The following outlines the types of testing that will be done for unit, integration, and system testing. While it includes what will be tested, the specific use cases that determine how the testing is done will be detailed in the Test Design Document. The test cases that will be used for designing use cases is shown in Figure 2.1 and onwards.

**Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Rajeev Ranjan | |
| **Test Type** | | Unit Testing | |
| **Test Case Number** | | 1 | |
| **Test Case Name** | | User Identification | |
| **Test Case Description** | | The user should enter his/ her accurate user name and password so that he/she can able to go for the further options. The test case will check the application for the same since a user can only login with the correct user name , password. | |
| **Item(s) to be tested** | | | |
| 1 | Verification of the user name and password with the record in the database. | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| * Correct User name and password      * Incorrect Id or Password | | | * Successful login * Failure Message |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Rajeev Ranjan | |
| **Test Type** | | Unit Testing | |
| **Test Case Number** | | 2 | |
| **Test Case Name** | | New Faculty Sign up testing | |
| **Test Case Description** | | A new Faculty must enter his /her profile in data base after clicking on sign up button | |
| **Item(s) to be tested** | | | |
| 1 | Whether the username entered is unique, is checked. | | |
| 2 | Whether the field is non empty, is checked. | | |
| 3 | Whether the password is of length 8 character or long, is checked | | |
| 4 | Whether the password and confirm password field is same, is checked | | |
| 5 | Whether the Contact no is number, is checked | | |
| 6 | Whether the Email ID is of the correct format, is checked | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| 1) Field is left empty and Sign up button is clicked, Password does not match, email id is not in correct format, Phone no. Is not number.  2) All fields are in correct format and Sign up button is clicked.  3) Cancel button is clicked. | | | 1) Respective error message is generated.  2) A message is generated informing new faculty is registered.  3) Login page appears. |

|  |  |  |
| --- | --- | --- |
| **Tested By:** | | Rajeev Ranjan |
| **Test Type** | | Unit Testing |
| **Test Case Number** | | 3 |
| **Test Case Name** | | Options page |
| **Test Case Description** | | Faculty must choose from the list of options available to proceed forward after clickin the Login button on Login Page. |
| **Item(s) to be tested** | | |
| 1 | Each button must lead to its corresponding page. | |
| 2 | Logout button will take faculty to Login/Signup page. | |
| **Specifications** | | |
| **Input** | | **Expected**  **Output/Result** |
| 1. Each button will be clicked by the user | | 1. Each button will take the user to a new page corresponding to its name. |

|  |  |  |
| --- | --- | --- |
| **Tested By:** | | Rajeev |
| **Test Type** | | Unit Testing |
| **Test Case Number** | | 4 |
| **Test Case Name** | | Update user Information |
| **Test Case Description** | | Faculty can update his/her information stored in data Base by providing the current password. |
| **Item(s) to be tested** | | |
| 1 | Whether the field is empty, is checked. | |
| 2 | Whether the password is of length 8 character or long, is checked | |
| 3 | Whether the current password matches with that in the database. | |
| 4 | Whether the password and confirm password field is same, is checked | |
| 5 | Whether the Email ID is of the correct format, is checked | |
| **Specifications** | | |
| **Input** | | **Expected Output/Result** |
| 1) Field is left empty and Sign up button is clicked, Password does not match, email id is not in correct format, Phone no. Is not number.  2) All fields are in correct format and Sign up button is clicked.  3) Cancel button is clicked. | | 1) Respective error message is generated.  2) A message is generated informing new faculty record is updated.  3) Option page appears. |

|  |  |  |
| --- | --- | --- |
| **Tested By:** | | Rajeev Ranjan |
| **Test Type** | | Unit Testing |
| **Test Case Number** | | 5 |
| **Test Case Name** | | Create Batch |
| **Test Case Description** | | A new batch is created under the username of the logged in faculty. |
| **Item(s) to be tested** | | |
| 1 | Whether all the fields are filled up. | |
| 2 | Date is selected from DatePicker. | |
| **Specifications** | | |
| **Input** | | **Expected**  **Output/Result** |
| 1) Submit button clicked without entering quantity.  2) Cancel button is clicked.  3) Click Datepicker corresponding to the textbox. | | 1) A message asks the user to enter the values.  2) Option page appears.  3) Calendar will appear from which we can select the date. |

|  |  |  |
| --- | --- | --- |
| **Tested By:** | | Rajeev Ranjan |
| **Test Type** | | Unit Testing |
| **Test Case Number** | | 6 |
| **Test Case Name** | | Update Batch |
| **Test Case Description** | | A batch can be updated by providing the BID |
| **Item(s) to be tested** | | |
| 1 | Whether all the fields are filled up, is checked. | |
| 2 | Whether the date is selected from DatePicker, is checked. | |
| 3 | Whether the Form will appear with the corresponding current values, is checked | |
| **Specifications** | | |
| **Input** | | **Expected**  **Output/Result** |
| 1) Submit button clicked without entering quantity.  2) Cancel button is clicked on page demanding BID.  3) Cancel button is clicked on page demanding updated values.  4) On successful completion of all the fields,  And submit button is clicked  5) Click Datepicker corresponding to the textbox. | | 1) A message asks the user to enter the values.  2) Option page appears.  3) Enter bid page appears.  4) A message is generated showing update is successful is generated.  5) Calendar will appear from which we can select the date |

|  |  |  |
| --- | --- | --- |
| **Tested By:** | | Rajeev Ranjan |
| **Test Type** | | Unit Testing |
| **Test Case Number** | | 7 |
| **Test Case Name** | | LOGOUT |
| **Test Case Description** | | Faculty can logout. |
| **Item(s) to be tested** | | |
| 1 | Whether Logout button leads to Login/Signup page. | |
| **Specifications** | | |
| **Input** | | **Expected**  **Output/Result** |
| 1) Submit button clicked without entering quantity.  2) Cancel button is clicked on page demanding BID.  3) Cancel button is clicked on page demanding updated values.  4) On successful completion of all the fields,  And submit button is clicked  5) Click Datepicker corresponding to the textbox. | | 1) A message asks the user to enter the values.  2) Option page appears.  3) Enter bid page appears.  4) A message is generated showing update is successful is generated.  5) Calendar will appear from which we can select the date |

**Unit Testing**

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness.

**White Box Testing**

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

Each function of the binary tree repository is executed independently; therefore, a program flow for each function has been derived from the code.

**Black Box Testing**

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

**System Testing**

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case well focus only on function validation and performance. And in both cases we will use the black-box method of testing.

**System Security measures (Implementation of security for the project developed)**

* Only authenticated users are allowed.
* The users (Faculty or the admin) are allowed to go to a page according to their respective authorizations.

**Database/Data security**

* Database is present in the client’s own machine.
* Oracle’s default securities are applied.

**Creation of User profiles and access rights**

* The registered agents and the admin is allowed to take actions accordingly.
* Passwords can be changed by the users themselves.

**Cost Estimation of the Project along with Cost Estimation Model**

**Analogous estimate of effort or cost**

Used for Early Estimate or Individual Activity Estimate

Sample example shown below is for two major deliverables of a software project. You use a previous project as a benchmark for analogous estimation. Using your experience you will estimate a multiplier.

Multipliers:

* Prototyping: 0.75.
* Testing: 0.5
* Deployment: 0.5

Finally, if you want to convert to cost, you would use current rates for the resource.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **WBS ID** | **Previous**  **Similar Project**  **Activity** | **Previous**  **Effort** | **Current**  **Project**  **Estimate** | **Multiplier** | **Effort**  **(Previous Effort \* 0.75)** | **Cost**  **(Rs. 500/hr.)** |
| 1 | Prototyping | 40 Work-Hours | Prototyping | 0.75 | 30 Work-hours | Rs. 15000/- |
| 2 | Testing | 20 Work-Hours | Testing | 0.50 | 10 Work-Hours | Rs. 5000/- |
| **Total** |  |  |  |  | **40 Work- Hours** | **Rs. 20000/-** |

Note: Effort is also called Size and unit of estimation is called either Work-Hour, person-hours.

**Future scope and further enhancement of the Project**

Inventory Management System has lot of enhancement options. In future, the total money transaction on a particular date can be found out. Various categories of the same type of product can be included. AI concepts can be applied to make the system intelligent.

**Bibliography**

* Roger S. Pressman. Software Engineering: A Practioner's Approach (Sixth Edition, International Edition). McGraw-Hill, 2005.
* Ian Sommerville. Software Engineering (Seventh Edition). Addison-Wesley, 2004.
* Edward Yourdon and Larry L. Constantine. Structured Design: Fundamentals of a Discipline of Computer Program and System Design. Prentice-Hall, 1979.
* JAVA The Complete Reference by Herbert Schildt.
* Robert Eckstein and Marc Loy and Dave Wood, Java Swing, O'Reilly, 1998.
* David Flanagan, Java in a Nutshell, second ed., O'Reilly, 1996.
* George Reese, Database Programming with JDBC and Java, O'Reilly, 1997.