CS23333 OBJECT ORIENTED PROGRAMMING

USING JAVA LAB

UNIVERSITY MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

UMESH SARATHY S K	**231501177**
VISWA V	**231501188**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS) THANDALAM

CHENNAI-602105

2024 - 2025

\

BONAFIDE CERTIFICATE

Certified that this project report "**UNIVERSITY MANAGEMENT SYSTEM** " is the Bonafide work of **"UMESH SARATHY S K (231501177), VISWA V (231501188)"** who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE	**SIGNATURE**
Dr.Sekar K M.E., Ph.D,	**Mr. Devendar Rao.**
HEAD OF THE DEPARTMENT	**FACULTY INCHARGE**
Department of AIML	**Department of AIML**
Rajalakshmi Engineering College, EngineerinCollege,	**Rajalakshmi**
Thandalam,	**Thandalam,**
Chennai-602 105.	**Chennai- 602 105.**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

This paper outlines the design, implementation, and functionality of a **University Management System (UMS)**, a web-based application developed to automate and optimize various academic and administrative operations in a university setting. Leveraging the robustness of a **MySQL database**, the system effectively organizes and manages key data such as student information, faculty profiles, course registrations, and academic schedules.

The application is built using **Java** and features an intuitive user interface that facilitates efficient interactions among students, faculty, and administrators. Core features of the system include:

- **User Account Management**: Secure login and sign-up functionality with unique credentials for all users.
- **Course Enrollment**: Students can browse and register for courses, while administrators manage course offerings and faculty assignments.
- **Academic Records**: Updating and maintaining grades, attendance, and other academic details for students.
- **Schedule Management**: Organizing class schedules, exam timetables, and faculty availability to ensure smooth academic operations.

The UMS aims to reduce manual administrative effort, streamline communication, and provide actionable insights through data-driven dashboards. By automating repetitive tasks and improving operational efficiency, the system addresses challenges such as data inconsistency, time inefficiencies, and user dissatisfaction. Its scalable and modular design ensures adaptability to different university requirements, fostering enhanced collaboration and productivity across the institution.

TABLE OF CONTENTS

CHAPTER 1

1.1 INTRODUCTION

The University Management System is an innovative platform designed to provide students, faculty, and administrators with a seamless experience in managing academic and administrative processes. With the growing reliance on digital platforms in education, this system streamlines activities such as student registrations, course enrollments, and schedule management, all within a userfriendly interface.

This project focuses on enhancing the convenience and accessibility of university services for all stakeholders, enabling them to efficiently manage academic operations through the following features:

- **User Registration and Login:**
  Students, faculty, and administrators can register securely and log in to access personalized features, including viewing schedules, academic records, and notifications.

- **Dynamic Course Search:**
  Users can explore available courses and schedules, filtering results by department, semester, and faculty.

- **Interactive Enrollment Management**:
  Students can view course availability in real time, select their preferred courses, and add them to their schedule. The system ensures real-time updates to avoid conflicts.

- **Customizable Profile Management**:

Users can update their personal details, academic preferences, and communication settings to ensure a tailored experience.

- **Academic and Administrative Notifications:**
  Users receive timely updates about class schedules, deadlines, and institutional announcements. Automated email notifications provide updates on:
  - Successful account registration. o Enrollment confirmations.
  - Schedule changes or faculty adjustments.
  - Event reminders and other institutional updates.

By digitizing traditional university processes, the University Management System offers unparalleled convenience, saving time and effort for users. This system emphasizes reliability, real-time updates, and ease of use, making it an essential tool for modernizing higher education.

1.2 OBJECTIVE

The main objective of the University Management System is to manage the details of Students, Faculty, Courses, and Schedules. It centralizes all critical inform ation and automates tasks like registrations, course assignments, and schedule updates effectively using a MySQL database. The project is entirely user-focused, enabling stakeholders to access the system by signing up and logging in.

1.3 MODULES

1. Sign Up

- The AddFaculty and AddStudent classes in the provided code implement a Sign-Up mechanism for faculty and students, respectively.
- Both forms collect essential details such as:

  ○ Name ○ Father's Name ○ Date of Birth ○ Address ○ Contact Information (Phone, Email) ○ Identification (Employee ID/Roll Number, Aadhar Number) ○ Educational Qualifications (Class X and XII Marks) ○ Course and Department

- **A random unique ID is generated for both faculty and students (e.g., empText for faculty and empText for students).**
- **The collected data is inserted into a database using SQL commands (INSERT INTO statements) executed via the Conn class.**

2. Login

- While the provided code does not include specific implementation for Login, it complements the Sign-Up process.

- Typically, login functionality would verify the credentials of users stored in the database during the sign-up process.
- Users like faculty and students would enter their unique IDs (e.g., Employee ID or Roll Number) and passwords to gain access.

3. Course Management

- The dropdown menus in the AddFaculty and AddStudent forms (e.g., courseBox and departmentBox) reflect course management by allowing users to select:
    - Courses: B.Tech, BBA, BCA, etc. ○ Departments/Branches: Computer Science, Electronics, Mechanical, etc.
- This facilitates assigning faculty members and students to appropriate courses and departments.
- The course and department details are stored in the database, allowing for efficient course management.

4. Student and Faculty Profiles

- The profiles for students and faculty members are managed through the forms in AddStudent and AddFaculty:
    - Students:
        - Roll Number as a unique identifier

        - Personal and academic details ○ Faculty:
        - Employee ID as a unique identifier
        - Personal and professional details
- The information stored in the database can later be retrieved and displayed as profiles.

5. Updating Academic Records

- Although the code does not explicitly handle academic record updates, the structure supports adding records to the database:
    - For students, records like Class X and XII percentages are stored. ○ For faculty, qualifications and department information are captured.
- Extending this, academic records like grades, attendance, or performance can be updated using a similar approach: ○ Fetch the record using unique IDs.
    - Update fields in the database using SQL UPDATE statements.

This maintains the structure, features, and formatting of the original content but adapts it for a University Management System context.

CHAPTER 2

2.1 SOFTWARE DESCRIPTION

Visual studio Code

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas.

2.2 LANGUAGES

2.2.1 JAVA

Java is a set of computer software and specifications that provides a software platform for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. Java applets, which are less common than standalone Java applications, were commonly run in secure, sandboxed environments to provide many features of native applications through being embedded in HTML pages.

2.2.2 MySQL

Many of the world's largest and fastest-growing organizations including Facebook, Google, Adobe, Alcatel Lucent and Zappos rely on MySQL to save time and money powering their highvolume Web sites, business-critical systems and packaged software. Since then, the performance & scalability, reliability, and ease of use of the world's most popular open source database, characteristics that made MySQL the #1 choice for web applications, have relentlessly been improved.

CHAPTER 3

3.1 REQUIREMENTS SPECIFICATION

The requirements specification defines the functional and non-functional requirements necessary for the successful development and implementation of the University Management System (UMS). These requirements ensure that the system meets the needs of all stakeholders, including students, faculty, and administrators.

3.1.1 Functional Requirements

Functional requirements outline the specific features and functionalities that the system must provide:

1. **User Management**

   ○ Account creation and secure login for students, faculty, and administrators. ○ Role-based access control, ensuring appropriate permissions for each user type.

2. **Course Management**

   ○ Admin functionality to add, update, or delete courses. ○ Students can enroll in courses and view their current and past enrollments. ○ Faculty can view and manage assigned courses.

3. **Academic Records Management** ○ Updating and maintaining student grades, attendance, and other academic details.

   ○ Generating reports for academic performance and attendance tracking.

4. **Schedule Management**

   ○ Creating and updating class and exam schedules. ○ Viewing faculty availability and room allocations.

5. Notifications and Alerts

- Sending alerts to users regarding deadlines, announcements, or schedule changes.

6. Data Management

- Secure storage of user and institutional data in the MySQL database.
- Backup and recovery mechanisms for critical data.

3.1.2 Non-Functional Requirements

Non-functional requirements focus on the quality attributes of the system, including performance, usability, and security:

1. Performance

- The system must handle simultaneous access by multiple users without significant delays. ○ Database queries should execute within a few seconds for optimal user experience.

2. Scalability

- The system must be scalable to accommodate increasing user loads, additional courses, and growing data volume.

3. Usability

- The user interface should be intuitive, with clear navigation for all user types. ○ The system must support accessibility features to accommodate diverse user needs.

4. Security ○ Secure login mechanisms with encrypted passwords. ○ Data encryption to protect sensitive information such as academic records and personal details.

- Role-based access to prevent unauthorized actions.

5. Reliability and Availability

- The system must provide 99.9% uptime to ensure availability during academic and administrative operations.
- Regular maintenance schedules should minimize downtime.

6. Maintainability

- Modular design to facilitate future updates or integration of additional features.
- Comprehensive documentation for developers and administrators.

3.1.3 Stakeholder Requirements

1. **Students**
   - Easy course enrollment and access to academic records.
   - Notifications for upcoming deadlines and schedule changes.
2. **Faculty**
   - Tools for managing assigned courses and viewing student progress.
   - Simplified schedule management for classes and exams.
3. **Administrators**
   - Centralized control over courses, faculty assignments, and institutional data.
   - Reports for analyzing academic and administrative efficiency.

By addressing these requirements, the University Management System aims to deliver a robust, user-friendly, and secure platform to support efficient university operations.

CHAPTER 4

PROGRAM CODE:

Iml code :

```
<?xml version="1.0" encoding="UTF-8"?>
<module type="JAVA_MODULE" version="4">
  <component name="NewModuleRootManager" inherit-compiler-output="true">
    <exclude-output />
    <content url="file://$MODULE_DIR$">
      <sourceFolder url="file://$MODULE_DIR$/src" isTestSource="false" />
    </content>
    <orderEntry type="inheritedJdk" />
    <orderEntry type="sourceFolder" forTests="false" />
    <orderEntry type="library" name="jcalendar-1.4" level="project" />
    <orderEntry type="library" name="mysql-connector-java-8.0.28"
level="project" />
    <orderEntry type="library" name="rs2xml" level="project" />
  </component>
</module>
```

Java files: About.java

:

package university.management.system;

```
import javax.swing.*;
import java.awt.*;

public class About extends JFrame {
   About(){

      ImageIcon i1 = new
ImageIcon(ClassLoader.getSystemResource("icon/about.png"));
      Image i2 = i1.getImage().getScaledInstance(300, 200,
Image.SCALE_DEFAULT);
      ImageIcon i3 = new ImageIcon(i2);
JLabel img = new JLabel(i3);
img.setBounds(350,0,300,200);
add(img);

      JLabel heading = new JLabel("<html> A.V</br>Technical
University</html>");
      heading.setBounds(70,20,300,130);
heading.setFont(new Font("Tahoma", Font.BOLD,30));
add(heading);

      JLabel name = new JLabel("Techcoder A.V");
name.setBounds(60,260,550,40);        name.setFont(new
Font("Tahoma", Font.BOLD,30));        add(name);
```

CS23333 OBJECT ORIENTED PROGRAMMING USING JAVA

```
        JLabel contact = new JLabel("techcoderav@gmail.com");
contact.setBounds(70,340,550,40);            contact.setFont(new
Font("Tahoma", Font.BOLD,30));         add(contact);

        setSize(700,500);         setLocation(400,150);
getContentPane().setBackground(new Color(252,228,210));
setLayout(null);         setVisible(true);

    }
    public  static  void  main(String[]  args)  {
new About();
    }
}
```

Addfactulty.java : package university.management.system;

```
import com.toedter.calendar.JDateChooser;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener; import
java.util.Random;

public class AddFaculty extends JFrame implements ActionListener {

    JTextField
textName,textfather,textAddress,textPhone,textemail,textM10,textM12,textAadhar;
JLabel empText;
    JDateChooser cdob;
    JComboBox courseBox, departmentBox;
    JButton submit, cancel;

    Random ran = new Random();     long f4 =
Math.abs((ran.nextLong() % 9000L) + 1000L);
    AddFaculty(){

       getContentPane().setBackground(new Color(166,164,252));

        JLabel heading = new JLabel("New Teacher Details");
heading.setBounds(310,30,500,50);
        heading.setFont(new Font("serif",Font.BOLD,30));
add(heading);
```

```
JLabel name = new JLabel("Name");
name.setBounds(50,150,100,30);
name.setFont(new Font("serif",Font.BOLD,20));
```

```
add(name);

textName = new JTextField();
textName.setBounds(200,150,150,30);
add(textName);

JLabel fname = new JLabel("Father Name");
fname.setBounds(400,150,200,30);
fname.setFont(new Font("serif",Font.BOLD,20));
add(fname);

textfather = new JTextField();
textfather.setBounds(600,150,150,30);
add(textfather);

JLabel empID = new JLabel("Employee ID");
empID.setBounds(50,200,200,30);
empID.setFont(new Font("serif",Font.BOLD,20));
add(empID);

empText = new JLabel("1409"+f4);
empText.setBounds(200,200,150,30);
```

```
empText.setFont(new Font("serif",Font.BOLD,20));
add(empText);
```

```
JLabel dob = new JLabel("Date of Birth");
dob.setBounds(400,200,200,30);
dob.setFont(new Font("serif",Font.BOLD,20));
add(dob);

cdob = new JDateChooser();
cdob.setBounds(600,200,150,30);
add(cdob);

JLabel address = new JLabel("Address");
address.setBounds(50,250,200,30);
address.setFont(new Font("serif",Font.BOLD,20));
add(address);

textAddress = new JTextField();
textAddress.setBounds(200,250,150,30);
add(textAddress);

JLabel phone = new JLabel("Phone");
phone.setBounds(400,250,200,30);
phone.setFont(new Font("serif",Font.BOLD,20));
add(phone);
```

```
textPhone = new JTextField();
textPhone.setBounds(600,250,150,30);
```

```
add(textPhone);

JLabel email = new JLabel("Email");
email.setBounds(50,300,200,30);
email.setFont(new Font("serif",Font.BOLD,20));
add(email);

textemail = new JTextField();
textemail.setBounds(200,300,150,30);
add(textemail);

JLabel M10 = new JLabel("Class X (%)");
M10.setBounds(400,300,200,30);
M10.setFont(new Font("serif",Font.BOLD,20));
add(M10);

textM10 = new JTextField();
textM10.setBounds(600,300,150,30);
add(textM10);

JLabel M12 = new JLabel("Class XII (%)");
M12.setBounds(50,350,200,30);
```

```
        M12.setFont(new Font("serif",Font.BOLD,20));
add(M12);
```

```
        textM12 = new JTextField();
textM12.setBounds(200,350,150,30);
add(textM12);

        JLabel AadharNo = new JLabel("Aadhar Number");
        AadharNo.setBounds(400,350,200,30);
        AadharNo.setFont(new Font("serif",Font.BOLD,20));
add(AadharNo);

        textAadhar = new JTextField();
textAadhar.setBounds(600,350,150,30);
add(textAadhar);

        JLabel Qualification = new JLabel("Qualification");
        Qualification.setBounds(50,400,200,30);
        Qualification.setFont(new Font("serif",Font.BOLD,20));
add(Qualification);

        String course[] =
{"B.Tech","BBA","BCA","BSC","MSC","MBA","MCA","MCom","MA","BA"};
        courseBox = new JComboBox(course);
courseBox.setBounds(200,400,150,30);
courseBox.setBackground(Color.WHITE);
add(courseBox);
```

```
JLabel Department = new JLabel("Department");
```

```
        Department.setBounds(400,400,200,30);
        Department.setFont(new Font("serif",Font.BOLD,20));
add(Department);

        String department[] = {"Computer
Science","Electronics","Mechanical","Civil","IT"};
departmentBox = new JComboBox(department);
departmentBox.setBounds(600,400,150,30);
departmentBox.setBackground(Color.WHITE);
add(departmentBox);

        submit = new JButton("Submit");
submit.setBounds(250,550,120,30);
submit.setBackground(Color.black);
submit.setForeground(Color.white);
submit.addActionListener(this);
add(submit);

        cancel = new JButton("Cancel");
cancel.setBounds(450,550,120,30);
cancel.setBackground(Color.black);
cancel.setForeground(Color.white);
```

18

```
cancel.addActionListener(this);
add(cancel);

    setSize(900,700);
```

```
        setLocation(350,50);
setLayout(null);
setVisible(true);
    }
    @Override     public void
actionPerformed(ActionEvent e) {         if
(e.getSource() == submit){             String name
= textName.getText();
            String fname = textfather.getText();
            String empid = empText.getText();
            String dob = ((JTextField)
cdob.getDateEditor().getUiComponent()).getText();
            String address = textAddress.getText();
            String phone = textPhone.getText();
            String email = textemail.getText();
            String x = textM10.getText();
            String xii = textM12.getText();
            String aadhar = textAadhar.getText();
            String course = (String) courseBox.getSelectedItem();
            String department = (String) departmentBox.getSelectedItem();
try{
                String q = "insert into teacher values('"+name+"',
'"+fname+"','"+empid+"','"+dob+"','"+address+"','"+phone+"','"+email+"','"+x+"','"
+xii+"','"+aadhar+"','"+course+"','"+department+"')";
                Conn c =new Conn();
```

```
            c.statement.executeUpdate(q);
            JOptionPane.showMessageDialog(null,"Details Inserted");
setVisible(false);
        }catch (Exception E){
            E.printStackTrace();
        }       }else {
setVisible(false);
      }
   }

   public  static  void  main(String[]  args)  {
new AddFaculty();
   }
}
```

Addstudent.java package

university.management.system;

import com.toedter.calendar.JDateChooser;

import javax.swing.*; import

java.awt.*; import

20

```
java.awt.event.ActionEvent; import
java.awt.event.ActionListener;
```

```
import java.util.Random;

public class AddStudent extends JFrame implements ActionListener {
   JTextField
textName,textfather,textAddress,textPhone,textemail,textM10,textM12,textAadhar;
   JLabel empText;
   JDateChooser cdob;
   JComboBox courseBox, departmentBox;
   JButton submit, cancel;

   Random ran = new Random();     long f4 =
Math.abs((ran.nextLong() % 9000L) + 1000L);

   AddStudent(){         getContentPane().setBackground(new
Color(128,176,255));

       JLabel heading = new JLabel("New Teacher Details");
heading.setBounds(310,30,500,50);
heading.setFont(new              Font("serif",Font.BOLD,30));
add(heading);

       JLabel      name      =      new      JLabel("Name");
name.setBounds(50,150,100,30);
name.setFont(new         Font("serif",Font.BOLD,20));
add(name);
```

```
textName = new JTextField();
textName.setBounds(200,150,150,30);
add(textName);

JLabel fname = new JLabel("Father Name");
fname.setBounds(400,150,200,30);
fname.setFont(new Font("serif",Font.BOLD,20));
add(fname);

textfather = new JTextField();
textfather.setBounds(600,150,150,30);
add(textfather);

JLabel empID = new JLabel("Roll Number");
empID.setBounds(50,200,200,30);
empID.setFont(new Font("serif",Font.BOLD,20));
add(empID);

empText = new JLabel("1409"+f4);
empText.setBounds(200,200,150,30);
empText.setFont(new Font("serif",Font.BOLD,20));
add(empText);
```

22

```
JLabel dob = new JLabel("Date of Birth");
dob.setBounds(400,200,200,30);
```

```
        dob.setFont(new Font("serif",Font.BOLD,20));
add(dob);

        cdob = new JDateChooser();
cdob.setBounds(600,200,150,30);
add(cdob);

        JLabel address = new JLabel("Address");
address.setBounds(50,250,200,30);
address.setFont(new Font("serif",Font.BOLD,20));
add(address);

        textAddress = new JTextField();
textAddress.setBounds(200,250,150,30);
add(textAddress);

        JLabel phone = new JLabel("Phone");
phone.setBounds(400,250,200,30);
phone.setFont(new Font("serif",Font.BOLD,20));
add(phone);
```

23

```
        textPhone = new JTextField();
textPhone.setBounds(600,250,150,30);
add(textPhone);
```

```
        JLabel email = new JLabel("Email");
email.setBounds(50,300,200,30);
        email.setFont(new Font("serif",Font.BOLD,20));
        add(email);

        textemail = new JTextField();
textemail.setBounds(200,300,150,30);
add(textemail);

        JLabel M10 = new JLabel("Class X (%)");
        M10.setBounds(400,300,200,30);
        M10.setFont(new Font("serif",Font.BOLD,20));
add(M10);

        textM10 = new JTextField();
textM10.setBounds(600,300,150,30);
add(textM10);

        JLabel M12 = new JLabel("Class XII (%)");
        M12.setBounds(50,350,200,30);
        M12.setFont(new Font("serif",Font.BOLD,20));
add(M12);
```

```
textM12 = new JTextField();
textM12.setBounds(200,350,150,30);
```

```
        add(textM12);

        JLabel AadharNo = new JLabel("Aadhar Number");
        AadharNo.setBounds(400,350,200,30);
        AadharNo.setFont(new Font("serif",Font.BOLD,20));
add(AadharNo);

        textAadhar = new JTextField();
textAadhar.setBounds(600,350,150,30);
add(textAadhar);

        JLabel Qualification = new JLabel("Course");
        Qualification.setBounds(50,400,200,30);
        Qualification.setFont(new Font("serif",Font.BOLD,20));
add(Qualification);

        String course[] =
{"B.Tech","BBA","BCA","BSC","MSC","MBA","MCA","MCom","MA","BA"};
        courseBox = new JComboBox(course);
courseBox.setBounds(200,400,150,30);
courseBox.setBackground(Color.WHITE);
add(courseBox);

        JLabel Department = new JLabel("Branch");
```

```
Department.setBounds(400,400,200,30);
Department.setFont(new Font("serif",Font.BOLD,20));
```

```
        add(Department);

        String department[] = {"Computer
Science","Electronics","Mechanical","Civil","IT"};
departmentBox = new JComboBox(department);
departmentBox.setBounds(600,400,150,30);
departmentBox.setBackground(Color.WHITE);
add(departmentBox);

        submit = new JButton("Submit");
submit.setBounds(250,550,120,30);
submit.setBackground(Color.black);
submit.setForeground(Color.white);
submit.addActionListener(this);
add(submit);

        cancel = new JButton("Cancel");
cancel.setBounds(450,550,120,30);
cancel.setBackground(Color.black);
cancel.setForeground(Color.white);
cancel.addActionListener(this);
add(cancel);
```

```
        setSize(900,700);
    setLocation(350,50);
    setLayout(null);
```

```
        setVisible(true);
    }

    @Override     public void
actionPerformed(ActionEvent e) {         if
(e.getSource() == submit){             String name
= textName.getText();
            String fname = textfather.getText();
            String empid = empText.getText();
            String dob = ((JTextField)
cdob.getDateEditor().getUiComponent()).getText();
            String address = textAddress.getText();
            String phone = textPhone.getText();
            String email = textemail.getText();
            String x = textM10.getText();
            String xii = textM12.getText();
            String aadhar = textAadhar.getText();
            String course = (String) courseBox.getSelectedItem();
            String department = (String) departmentBox.getSelectedItem();
try{
                String q = "insert into student values('"+name+"',
'"+fname+"','"+empid+"','"+dob+"','"+address+"','"+phone+"','"+email+"','"+x+"','"
+xii+"','"+aadhar+"','"+course+"','"+department+"')";
                Conn c =new Conn();
                c.statement.executeUpdate(q);
```

```
JOptionPane.showMessageDialog(null,"Details Inserted");
```

```
            setVisible(false);
        }catch (Exception E){
            E.printStackTrace();
        }        }else {
setVisible(false);
      }
   }

   public  static  void  main(String[]  args)  {
new AddStudent();
   }
}

Conn.java package
university.management.system;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public      class      Conn      {
Connection connection;
   Statement statement;
```

```
    Conn(){
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection =
DriverManager.getConnection("jdbc:mysql:///universitymanagement","root","701
0
311401");            statement =
connection.createStatement();
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

Entermarks.java package

university.management.system;

```
import javax.swing.*; import
java.awt.*; import
java.awt.event.ActionEvent; import
java.awt.event.ActionListener;
import java.sql.ResultSet;

public class EnterMarks extends JFrame implements ActionListener {
    Choice choicerollno;
```

```
JComboBox comboBox;
JTextField sub1, sub2, sub3, sub4, sub5, mrk1, mrk2, mrk3, mrk4, mrk5;
```

```
    JButton submit,cancel;
    EnterMarks(){

        getContentPane().setBackground(new Color(252,245,210));

        ImageIcon i1 = new
ImageIcon(ClassLoader.getSystemResource("icon/exam.png"));
        Image i2 =
i1.getImage().getScaledInstance(400,300,Image.SCALE_DEFAULT);
        ImageIcon i3 = new ImageIcon(i2);
JLabel img = new JLabel(i3);
img.setBounds(500,40,400,300);
add(img);

        JLabel heading = new JLabel("Enter Marks of Student");
heading.setBounds(50,0,500,50);            heading.setFont(new
Font("Tahoma",Font.BOLD,20));         add(heading);

        JLabel rollno = new JLabel("Select Roll
Number");         rollno.setBounds(50,70,150,20);
add(rollno);
```

```
        choicerollno = new Choice();
choicerollno.setBounds(200,70,150,20);
add(choicerollno);
```

```
        try {
          Conn c = new Conn();
           ResultSet resultSet = c.statement.executeQuery("select * from student");
while (resultSet.next()){
choicerollno.add(resultSet.getString("rollno"));
           }
        }catch (Exception e) {
e.printStackTrace();
        }

        JLabel sem = new JLabel("Select Semester");
sem.setBounds(50,110,150,20);         add(sem);

        String semester[] = {"1st Semester","2st Semester","3st Semester","4st
Semester","5st Semester","6st Semester","7st Semester","8st Semester"};
comboBox = new JComboBox(semester);
comboBox.setBounds(200,110,150,20);
comboBox.setBackground(Color.WHITE);         add(comboBox);

        JLabel entersub = new JLabel("Enter Subject");
entersub.setBounds(100,150,200,40);
add(entersub);
```

```
JLabel entermarks = new JLabel("Enter Marks");
entermarks.setBounds(320,150,200,40);
add(entermarks);

sub1 = new JTextField();
sub1.setBounds(50,200,200,20);
add(sub1);

sub2 = new JTextField();
sub2.setBounds(50,230,200,20);
add(sub2);

sub3 = new JTextField();
sub3.setBounds(50,260,200,20);
add(sub3);

sub4 = new JTextField();
sub4.setBounds(50,290,200,20);
add(sub4);
```

32

```
        sub5 = new JTextField();
sub5.setBounds(50,320,200,20);
add(sub5);


        mrk1 = new JTextField();
```

```
        mrk1.setBounds(250,200,200,20);
add(mrk1);

        mrk2 = new JTextField();
mrk2.setBounds(250,230,200,20);
add(mrk2);

        mrk3 = new JTextField();
mrk3.setBounds(250,260,200,20);
add(mrk3);

        mrk4 = new JTextField();
mrk4.setBounds(250,290,200,20);
add(mrk4);

        mrk5 = new JTextField();
mrk5.setBounds(250,320,200,20);
add(mrk5);

        submit = new JButton("Submit");
submit.setBounds(70,360,150,25);
submit.setBackground(Color.black);
submit.setForeground(Color.WHITE);
```

```
submit.addActionListener(this);
add(submit);
```

```
        cancel = new JButton("Cancel");
cancel.setBounds(280,360,150,25);
cancel.setBackground(Color.black);
cancel.setForeground(Color.WHITE);
cancel.addActionListener(this);
add(cancel);

        setSize(1000,500);
setLayout(null);
setLocation(300,150);
setVisible(true);
    }
    @Override     public void
actionPerformed(ActionEvent e) {         if
(e.getSource() == submit){            try {
                Conn c = new Conn();
                String Q1 = "insert into subject
values('"+choicerollno.getSelectedItem()+"',
'"+comboBox.getSelectedItem()+"','"+sub1.getText()+"','"+sub2.getText()+"',
'"+sub3.getText()+"', '"+sub4.getText()+"', '"+sub5.getText()+"')";
                String Q2 = "insert into marks
values('"+choicerollno.getSelectedItem()+"',
'"+comboBox.getSelectedItem()+"','"+mrk1.getText()+"','"+mrk2.getText()+"',
```

```
'"+mrk3.getText()+"', '"+mrk4.getText()+"', '"+mrk5.getText()+"')";
```

```
            c.statement.executeUpdate(Q1);
            c.statement.executeUpdate(Q2);
            JOptionPane.showMessageDialog(null,"Marks Inserted Sucessfully");
            setVisible(false);

        }catch (Exception E){
            E.printStackTrace();
        }
    }
  }

  public static void main(String[] args) {
new EnterMarks();
  }
}
```

Examinationdetails.java package

university.management.system;

```
import com.sun.source.tree.IfTree;
import net.proteanit.sql.DbUtils;
```

```
import javax.swing.*;
import java.awt.*;
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.ResultSet;

public class ExaminationDetails extends JFrame implements ActionListener {

    JTextField search;
    JButton result, back;
    JTable table;
    ExaminationDetails(){

        getContentPane().setBackground(new Color(241,252,210));

        JLabel heading = new JLabel("check Result");
heading.setBounds(350,15,400,50);
heading.setFont(new Font("Tahoma",Font.BOLD,24));
add(heading);

        search = new JTextField();
search.setBounds(80,90,200,30);  search.setFont(new
Font("Tahoma", Font.PLAIN,18));  add(search);
```

CS23333 OBJECT ORIENTED PROGRAMMING USING JAVA

```
        result = new JButton("Result");
result.setBounds(300,90,120,30);
result.setBackground(Color.black);
result.setForeground(Color.white);
result.addActionListener(this);
add(result);

        back = new JButton("Back");
back.setBounds(440,90,120,30);
back.setBackground(Color.black);
back.setForeground(Color.white);
back.addActionListener(this);
add(back);

        table = new JTable();
        JScrollPane scrollPane = new JScrollPane(table);
scrollPane.setBounds(0,130,1000,310);
add(scrollPane);

        try {
            Conn c = new Conn();
            ResultSet resultSet = c.statement.executeQuery("select * from student");
table.setModel(DbUtils.resultSetToTableModel(resultSet));
```

```
}catch (Exception e){
    e.printStackTrace();
```

```
        }

        table.addMouseListener(new MouseAdapter() {
            @Override            public void mouseClicked(MouseEvent e) {
int row = table.getSelectedRow();
search.setText(table.getModel().getValueAt(row,2).toString());
            }
        });

        setSize(1000,475);
setLocation(300,100);
setLayout(null);
setVisible(true);

    }
    @Override     public void
actionPerformed(ActionEvent e) {        if
(e.getSource() == result){
setVisible(false);
            new Marks(search.getText());
        }else {
setVisible(false);        }
```

```
    }

    public static void main(String[] args) {
new ExaminationDetails();
    }
}
```

Feestructure.java package

```
university.management.system;

import net.proteanit.sql.DbUtils;

import javax.swing.*; import
java.awt.*; import
java.awt.event.ActionEvent; import
java.awt.event.ActionListener;
import java.sql.ResultSet;

public class FreeStructure extends JFrame implements ActionListener
{     FreeStructure(){
getContentPane().setBackground(Color.WHITE);
```

```
JLabel heading = new JLabel("Fee Structure");
heading.setBounds(400,10,400,30);
heading.setFont(new Font("Tahoma", Font.BOLD, 30));
```

```
        add(heading);

        JTable table = new JTable();

        try {
            Conn c = new Conn();
            ResultSet resultSet = c.statement.executeQuery("select * from fee");
table.setModel(DbUtils.resultSetToTableModel(resultSet));

        }catch (Exception e){
            e.printStackTrace();
        }

        JScrollPane js = new
JScrollPane(table);
js.setBounds(0,60,1000,700);        add(js);



        setSize(1000,700);
setLocation(250,50);
setLayout(null);
setVisible(true);

    }
```

@Override

```
    public void actionPerformed(ActionEvent e) {

    }

    public static void main(String[] args) {
new FreeStructure();
    }
}
```

Login.java package

```
university.management.system;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.ResultSet;

public class Login extends JFrame implements ActionListener {
    JTextField textFieldName;
    JPasswordField passwordField;
    JButton login, back;
    Login(){
```

```
JLabel labelName = new JLabel("Username");
```

```
        labelName.setBounds(40,20,100,20);
add(labelName);

        textFieldName = new JTextField();
textFieldName.setBounds(150,20,150,20);
add(textFieldName);

        JLabel labelpass = new JLabel("Password");
labelpass.setBounds(40,70,100,20);
add(labelpass);

        passwordField = new JPasswordField();
passwordField.setBounds(150,70,150,20);
add(passwordField);

        login = new JButton("Login");
login.setBounds(40,140,120,30);
login.setBackground(Color.black);
login.setForeground(Color.white);
login.addActionListener(this);
add(login);
```

```
        back = new JButton("Back");
back.setBounds(180,140,120,30);
back.setBackground(Color.black);
```

```
        back.setForeground(Color.white);
back.addActionListener(this);
add(back);

        ImageIcon i1 = new
ImageIcon(ClassLoader.getSystemResource("icon/second.png"));
        Image i2 =
i1.getImage().getScaledInstance(200,200,Image.SCALE_DEFAULT);
        ImageIcon i3 = new ImageIcon(i2);
JLabel img = new JLabel(i3);
img.setBounds(350,20,200,200);
add(img);

        ImageIcon i11 = new
ImageIcon(ClassLoader.getSystemResource("icon/loginback.png"));
        Image i22 =
i11.getImage().getScaledInstance(600,300,Image.SCALE_DEFAULT);
        ImageIcon i33 = new ImageIcon(i22);
JLabel image = new JLabel(i33);
image.setBounds(0,0,600,300);
add(image);

        setSize(600,300);
setLocation(500,250);
```

```
setLayout(null);
setVisible(true);
```

```
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == login){
            String username = textFieldName.getText();
            String password = passwordField.getText();

            String query = "select * from login where username='"+username+"' and
password = '"+password+"'";
            try {
                Conn c = new Conn();
                ResultSet resultSet = c.statement.executeQuery(query);
if (resultSet.next()){                setVisible(false);                new
main_class();
                }else {
                    JOptionPane.showMessageDialog(null,"Invalid username or
password");
                }

            }catch (Exception E){
                E.printStackTrace();
            }

        }else {
```

```
            setVisible(false);
        }
    }

    public static void main(String[] args) {
new Login();
    }
}
```

Main_class.java package

university.management.system;

```
import javax.swing.*; import
java.awt.*; import
java.awt.event.ActionEvent; import
java.awt.event.ActionListener;

public class main_class extends JFrame implements ActionListener {
main_class(){
        ImageIcon i1 = new
ImageIcon(ClassLoader.getSystemResource("icon/third.jpg"));
        Image i2 = i1.getImage().getScaledInstance(1540,750,
Image.SCALE_DEFAULT);
```

```
        ImageIcon i3 = new ImageIcon(i2);
JLabel img = new JLabel(i3);
add(img);
```

```
JMenuBar mb = new JMenuBar();

// new Information

JMenu newInfo = new JMenu("New Information");
newInfo.setForeground(Color.BLACK);
mb.add(newInfo);

JMenuItem facultyInfo = new JMenuItem("New Faculty
Information");        facultyInfo.setBackground(Color.WHITE);
facultyInfo.addActionListener(this);        newInfo.add(facultyInfo);

JMenuItem studentInfo = new JMenuItem("New Student
Information");        studentInfo.setBackground(Color.WHITE);
studentInfo.addActionListener(this);        newInfo.add(studentInfo);

// Details

JMenu details = new JMenu("View Details");
details.setForeground(Color.BLACK);
details.addActionListener(this);
mb.add(details);
```

```
JMenuItem facultydetails = new JMenuItem("View Faculty
Details");        facultydetails.setBackground(Color.WHITE);
facultydetails.addActionListener(this);        details.add(facultydetails);

JMenuItem studentdetails = new JMenuItem("View Student
Details");        studentdetails.setBackground(Color.WHITE);
studentdetails.addActionListener(this);        details.add(studentdetails);

// Leave
JMenu leave = new JMenu("Apply Leave");
leave.setForeground(Color.BLACK);
leave.addActionListener(this);
mb.add(leave);

JMenuItem facultyLeave = new JMenuItem("Faculty Leave");
facultyLeave.setBackground(Color.WHITE);
facultyLeave.addActionListener(this);
leave.add(facultyLeave);
```

```
JMenuItem studentLeave = new JMenuItem("Student Leave");
studentLeave.setBackground(Color.WHITE);
studentLeave.addActionListener(this);
```

```
        leave.add(studentLeave);

        // Leave Details
        JMenu leaveDetails = new JMenu("Leave Details");
leaveDetails.setForeground(Color.BLACK);
studentdetails.addActionListener(this);
mb.add(leaveDetails);



        search = new JButton("Search");
search.setBounds(20,70,80,20);
search.addActionListener(this);
add(search);

        print = new JButton("Print");
print.setBounds(120,70,80,20);
print.addActionListener(this);
add(print);

        add = new JButton("Add");
add.setBounds(220,70,80,20);
add.addActionListener(this);
add(add);
```

```
update = new JButton("Update");
```

```
update.setBounds(320,70,80,20);
update.addActionListener(this);
add(update);

cancel = new JButton("Cancel");
cancel.setBounds(420,70,80,20);
cancel.addActionListener(this);
add(cancel);

empText = new JLabel();
empText.setBounds(200,200,150,30);
empText.setFont(new Font("serif",Font.BOLD,20));
add(empText);

JLabel dob = new JLabel("Date of Birth");
dob.setBounds(400,200,200,30);
dob.setFont(new Font("serif",Font.BOLD,20));
add(dob);

JLabel dobdob = new JLabel();
dobdob.setBounds(600,200,150,30);
add(dobdob);
```

```
JLabel address = new JLabel("Address");
address.setBounds(50,250,200,30);
```

```
        address.setFont(new Font("serif",Font.BOLD,20));
add(address);

        textAddress = new JTextField();
textAddress.setBounds(200,250,150,30);
add(textAddress);

        JLabel phone = new JLabel("Phone");
phone.setBounds(400,250,200,30);
phone.setFont(new Font("serif",Font.BOLD,20));
add(phone);

        textPhone = new JTextField();
textPhone.setBounds(600,250,150,30);
add(textPhone);

        JLabel email = new JLabel("Email");
email.setBounds(50,300,200,30);
        email.setFont(new Font("serif",Font.BOLD,20));
add(email);
```

```
        textemail = new JTextField();
textemail.setBounds(200,300,150,30);
add(textemail);
```

```
        JLabel M10 = new JLabel("Class X (%)");
        M10.setBounds(400,300,200,30);
        M10.setFont(new Font("serif",Font.BOLD,20));
add(M10);

        JLabel textM10 = new JLabel();
textM10.setBounds(600,300,150,30);
add(textM10);

        JLabel M12 = new JLabel("Class XII (%)");
        M12.setBounds(50,350,200,30);
        M12.setFont(new Font("serif",Font.BOLD,20));
add(M12);

        JLabel textM12 = new JLabel();
textM12.setBounds(200,350,150,30);
add(textM12);

        JLabel AadharNo = new JLabel("Aadhar Number");
        AadharNo.setBounds(400,350,200,30);
        AadharNo.setFont(new Font("serif",Font.BOLD,20));
add(AadharNo);
```

```
textAadhar = new JTextField();
textAadhar.setBounds(600,350,150,30);
```

```
        add(textAadhar);

        JLabel Qualification = new JLabel("Qualification");
        Qualification.setBounds(50,400,200,30);
        Qualification.setFont(new Font("serif",Font.BOLD,20));
add(Qualification);

        textcourse = new JTextField();
textcourse.setBounds(200,400,150,30);
add(textcourse);

        JLabel Department = new JLabel("Department");
        Department.setBounds(400,400,200,30);
        Department.setFont(new Font("serif",Font.BOLD,20));
add(Department);

       textbranch = new JTextField();
textbranch.setBounds(600,400,150,30);
add(textbranch);

       try{
           Conn c = new Conn();
```

```
        String query = "select * from teacher where empId =
'"+cEMPID.getSelectedItem()+"'";

        ResultSet resultSet = c.statement.executeQuery(query);

while (resultSet.next()){
```

```
            textName.setText(resultSet.getString("name"));
textfather.setText(resultSet.getString("fname"));
dobdob.setText(resultSet.getString("dob"));
textAddress.setText(resultSet.getString("address"));
textPhone.setText(resultSet.getString("phone"));
textemail.setText(resultSet.getString("email"));
textM10.setText(resultSet.getString("class_x"));
textM12.setText(resultSet.getString("class_xii"));
textAadhar.setText(resultSet.getString("aadhar"));
empText.setText(resultSet.getString("empId"));
textcourse.setText(resultSet.getString("education"));
textbranch.setText(resultSet.getString("department"));
        }

    }catch (Exception E){
        E.printStackTrace();
    }

    cEMPID.addItemListener(new ItemListener() {
        @Override                          public   void
itemStateChanged(ItemEvent e) {            try {
            Conn c = new Conn();
            String query = "select * from teacher where empId =
'"+cEMPID.getSelectedItem()+"'";
```

```
ResultSet resultSet = c.statement.executeQuery(query);
```

```
                while (resultSet.next()) {
textName.setText(resultSet.getString("name"));
textfather.setText(resultSet.getString("fname"));
dobdob.setText(resultSet.getString("dob"));
textAddress.setText(resultSet.getString("address"));
textPhone.setText(resultSet.getString("phone"));
textemail.setText(resultSet.getString("email"));
textM10.setText(resultSet.getString("class_x"));
textM12.setText(resultSet.getString("class_xii"));
textAadhar.setText(resultSet.getString("aadhar"));
empText.setText(resultSet.getString("empId"));
textcourse.setText(resultSet.getString("education"));
textbranch.setText(resultSet.getString("department"));
                }
            }catch (Exception E){
                E.printStackTrace();
            }
        }
    });


     submit = new JButton("Update");
submit.setBounds(250,550,120,30);
submit.setBackground(Color.black);
```

```
submit.setForeground(Color.white);

submit.addActionListener(this);
```

```
        add(submit);

        cancel = new JButton("Cancel");
cancel.setBounds(450,550,120,30);
cancel.setBackground(Color.black);
cancel.setForeground(Color.white);
cancel.addActionListener(this);
add(cancel);

        setSize(900,700);
setLocation(350,50);
setLayout(null);
setVisible(true);

    }
    @Override
    public void actionPerformed(ActionEvent e) {
if (e.getSource() == submit){
            String empid = empText.getText();
            String address = textAddress.getText();
            String phone = textPhone.getText();
            String email = textemail.getText();
```

```
String course = textcourse.getText();
String branch = textbranch.getText();
```

```
        try {
            String Q = "update teacher set address = '"+address+"', phone =
'"+phone+"', email = '"+email+"', education = '"+course+"', department =
'"+branch+"' where empId = '"+empid+"'";
            Conn c = new Conn();
            c.statement.executeUpdate(Q);

            JOptionPane.showMessageDialog(null, "Details Updated");
setVisible(false);

        }catch (Exception E){
            E.printStackTrace();
        }        }else {
setVisible(false);
      }
   }

   public static void main(String[] args) {
new UpdateTeacher();
   }
}
```

CS23333 OBJECT ORIENTED PROGRAMMING USING JAVA

DB code MySQL

```
-- Create database
CREATE DATABASE UniversityManagement;

-- Use the database
USE UniversityManagement;

-- Table: Login
CREATE TABLE Login (
   LoginID INT AUTO_INCREMENT PRIMARY KEY,
   Username VARCHAR(50) NOT NULL,
   PasswordHash VARCHAR(255) NOT NULL,
   Role ENUM('Student', 'Teacher', 'Admin') NOT NULL
);

-- Table: Register
CREATE TABLE Register (
   RegisterID INT AUTO_INCREMENT PRIMARY KEY,
   FullName VARCHAR(100) NOT NULL,
   Email VARCHAR(100) UNIQUE NOT NULL,
```

ContactNumber VARCHAR(15),

```
    DateOfRegistration DATE NOT NULL
);

-- Table: StudentDetails
CREATE TABLE StudentDetails (
    StudentID INT AUTO_INCREMENT PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    DateOfBirth DATE,
    Gender ENUM('Male', 'Female', 'Other'),
    Email VARCHAR(100) UNIQUE,
    ContactNumber VARCHAR(15),
    Address TEXT,
    EnrollmentDate DATE NOT NULL
);

-- Table: StudentLeave
CREATE TABLE StudentLeave (
    LeaveID INT AUTO_INCREMENT PRIMARY KEY,
    StudentID INT NOT NULL,
    LeaveReason TEXT NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    Status ENUM('Pending', 'Approved', 'Rejected') DEFAULT 'Pending',
    FOREIGN KEY (StudentID) REFERENCES StudentDetails(StudentID)
);

-- Table: StudentLeaveDetails
CREATE TABLE StudentLeaveDetails (
```

DetailID INT AUTO_INCREMENT PRIMARY KEY,

```
    LeaveID INT NOT NULL,
    ApprovedBy VARCHAR(100),
    ApprovalDate DATE,
    Remarks TEXT,
    FOREIGN KEY (LeaveID) REFERENCES StudentLeave(LeaveID)
);

-- Table: TeacherDetails
CREATE TABLE TeacherDetails (
    TeacherID INT AUTO_INCREMENT PRIMARY KEY,
    FullName VARCHAR(100) NOT NULL,
    DateOfBirth DATE,
    Gender ENUM('Male', 'Female', 'Other'),
    Email VARCHAR(100) UNIQUE,
    ContactNumber VARCHAR(15),
    Address TEXT,
    HireDate DATE NOT NULL
);

-- Table: TeacherLeave
CREATE TABLE TeacherLeave (
    LeaveID INT AUTO_INCREMENT PRIMARY KEY,
    TeacherID INT NOT NULL,
    LeaveReason TEXT NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    Status ENUM('Pending', 'Approved', 'Rejected') DEFAULT 'Pending',
    FOREIGN KEY (TeacherID) REFERENCES TeacherDetails(TeacherID)
```

```
);
```

```
-- Table: TeacherLeaveDetails
CREATE TABLE TeacherLeaveDetails (
    DetailID INT AUTO_INCREMENT PRIMARY KEY,
    LeaveID INT NOT NULL,
    ApprovedBy VARCHAR(100),
    ApprovalDate DATE,
    Remarks TEXT,
    FOREIGN KEY (LeaveID) REFERENCES TeacherLeave(LeaveID)
);

-- Table: Fees
CREATE TABLE Fees (
    FeeID INT AUTO_INCREMENT PRIMARY KEY,
    StudentID INT NOT NULL,
    Amount DECIMAL(10, 2) NOT NULL,
    DueDate DATE NOT NULL,
    PaidDate DATE,
    Status ENUM('Paid', 'Unpaid') DEFAULT 'Unpaid',
    FOREIGN KEY (StudentID) REFERENCES StudentDetails(StudentID)
);

-- Table: Marks
CREATE TABLE Marks (
    MarkID INT AUTO_INCREMENT PRIMARY KEY,
    StudentID INT NOT NULL,
    SubjectID INT NOT NULL,
    MarksObtained DECIMAL(5, 2),
```

TotalMarks DECIMAL(5, 2),

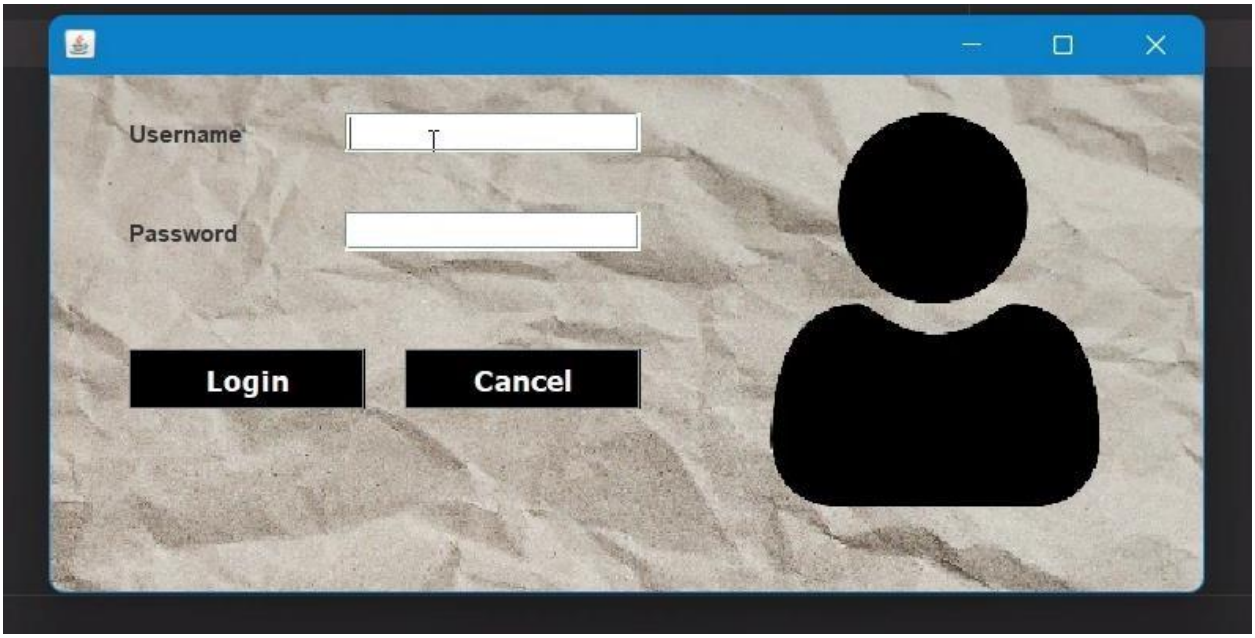
```
    ExamDate DATE,
    FOREIGN KEY (StudentID) REFERENCES StudentDetails(StudentID),
    FOREIGN KEY (SubjectID) REFERENCES Subjects(SubjectID)
);

-- Table: Subjects
CREATE TABLE Subjects (
    SubjectID INT AUTO_INCREMENT PRIMARY KEY,
    SubjectName VARCHAR(100) NOT NULL,
    SubjectCode VARCHAR(10) UNIQUE NOT NULL,
    TeacherID INT,
    FOREIGN KEY (TeacherID) REFERENCES TeacherDetails(TeacherID)
);
```

CHAPTER 5

PROJECT OUTCOME SCREENSHOTS:

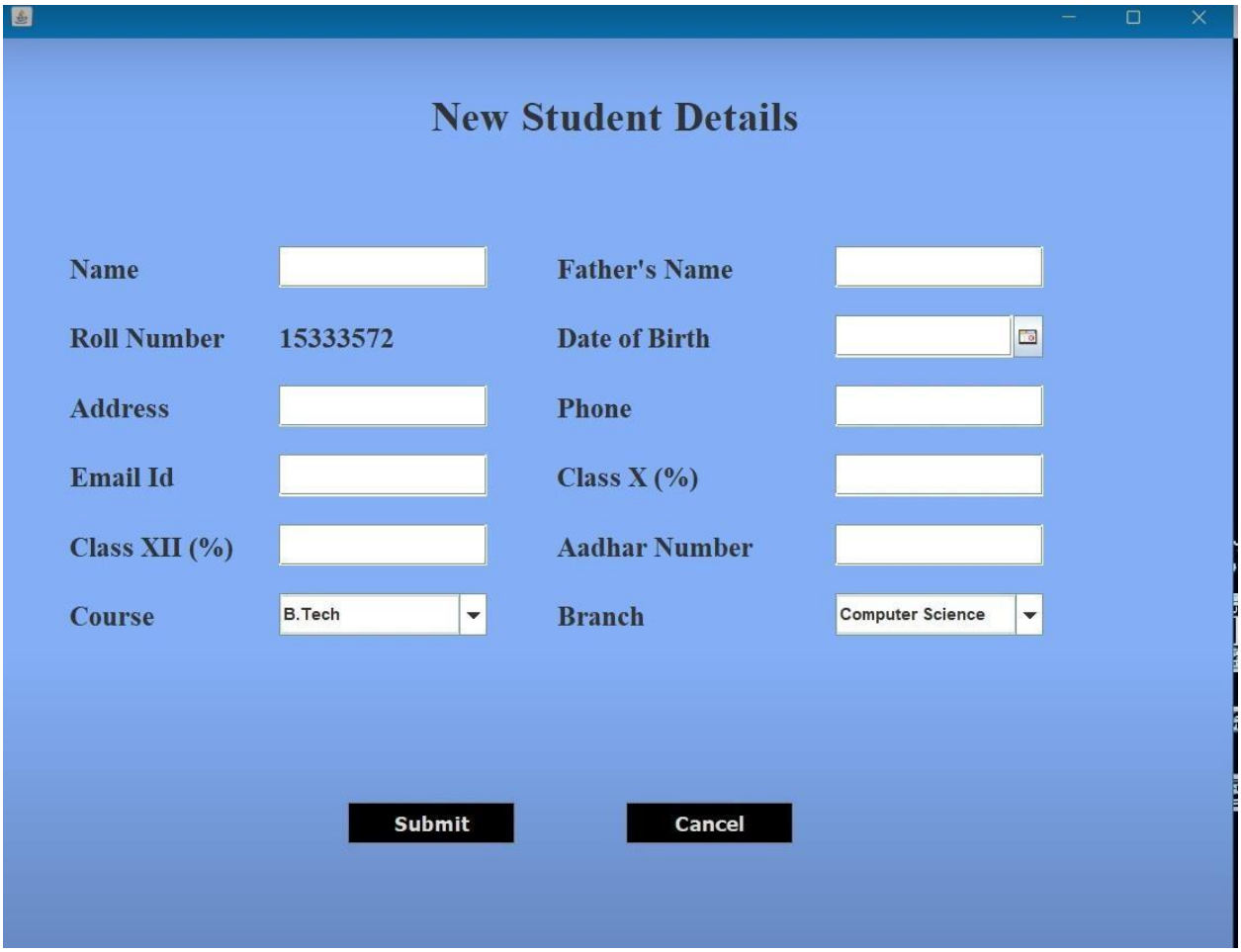
OPENING PAGE :

LOGIN PAGE :

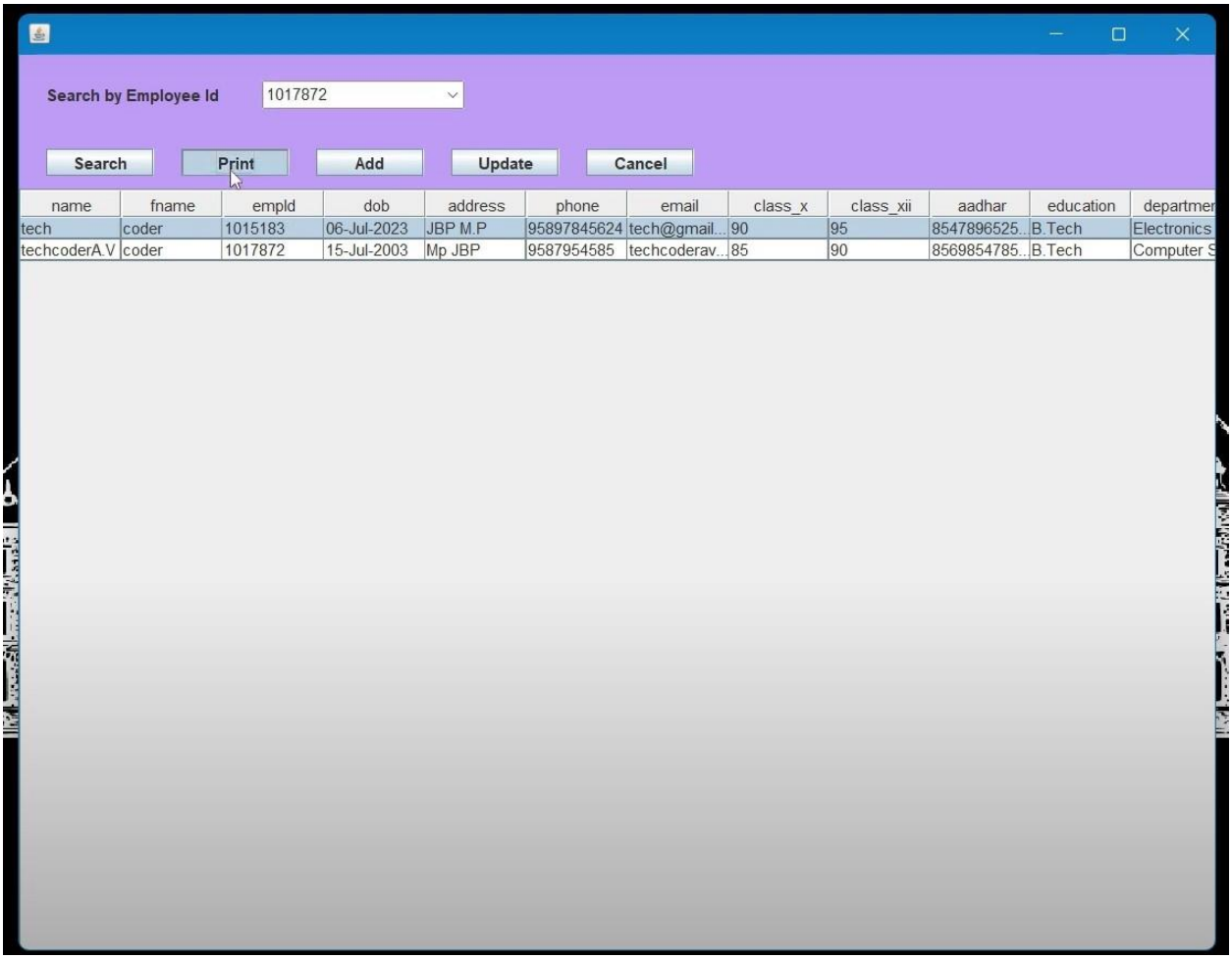


APPLY AS TEACHER :

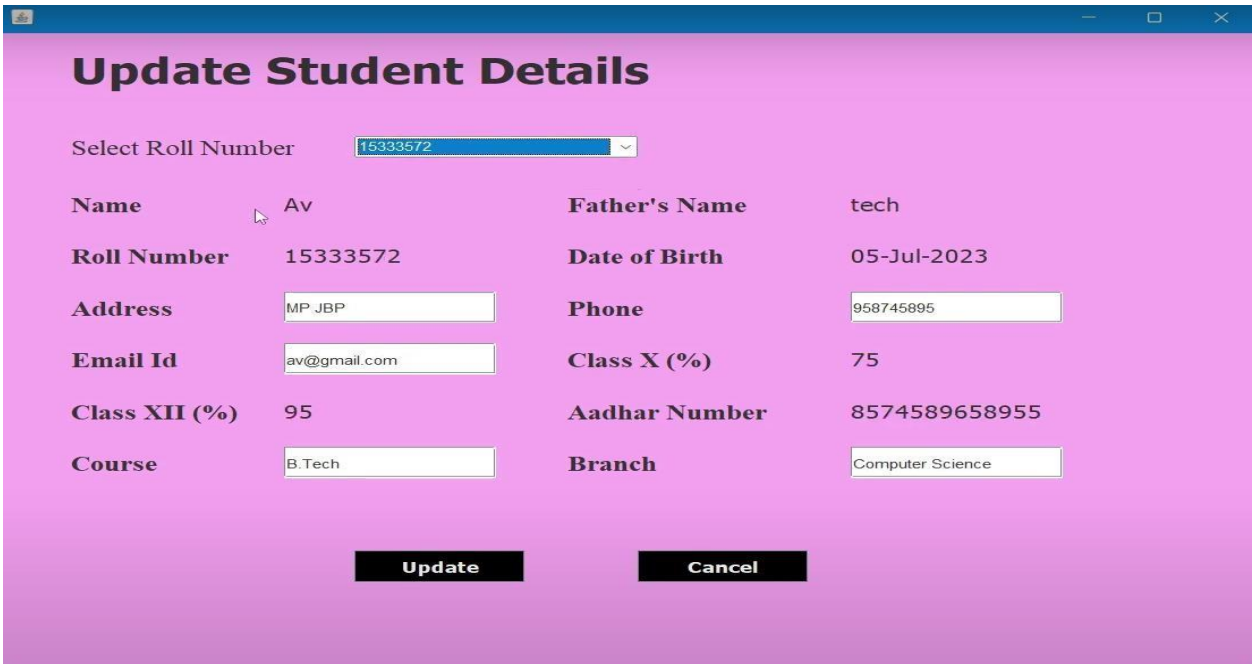
APPLY AS STUDENT :



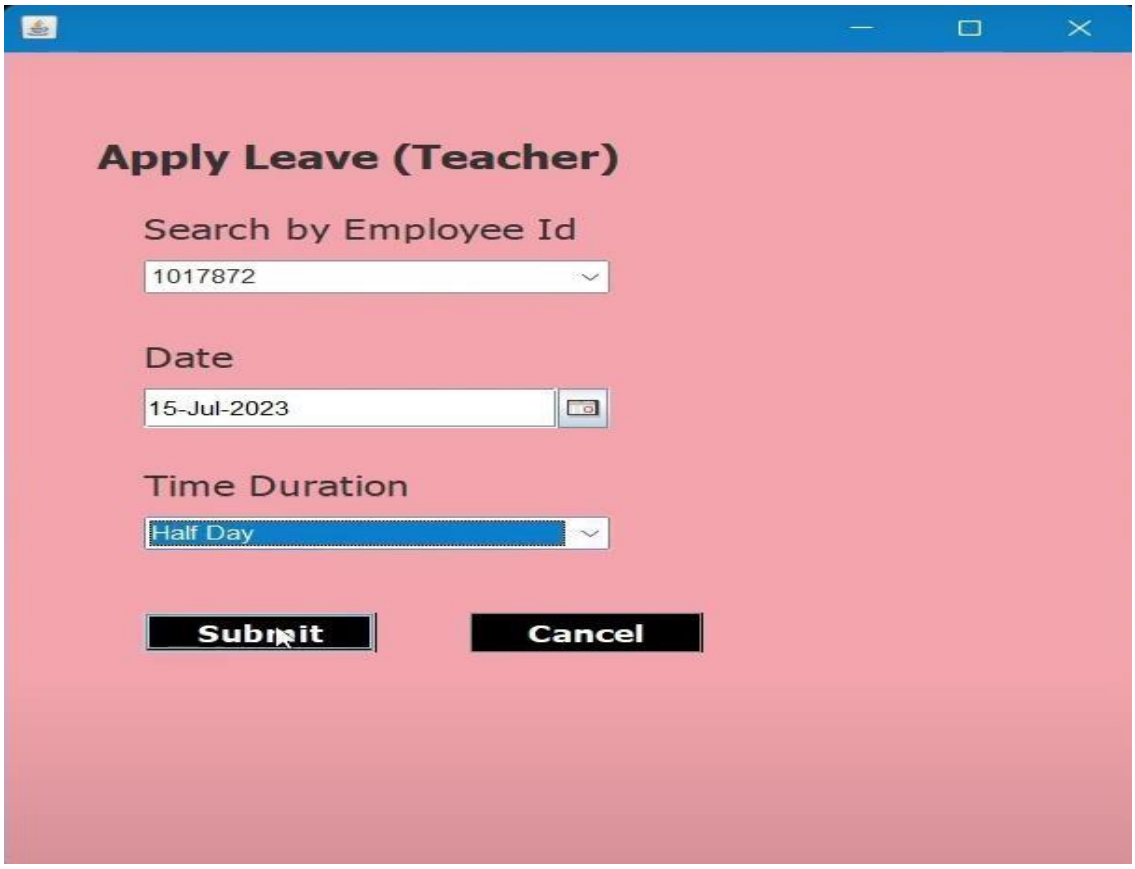
VIEW DETAILS :



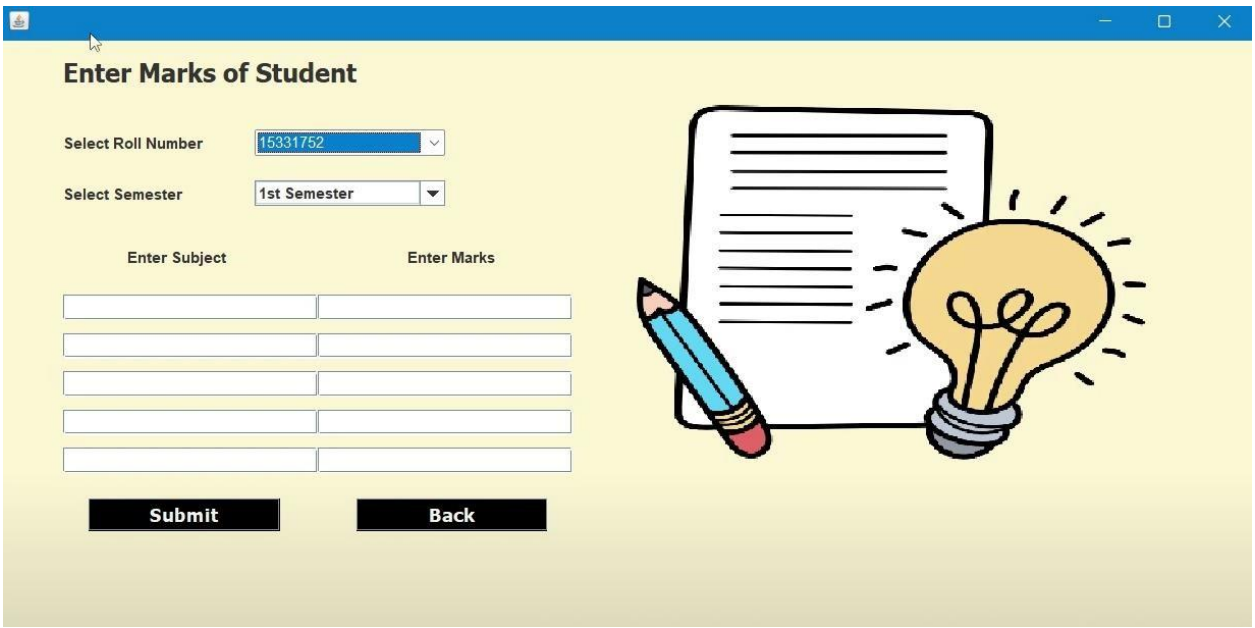
UPDATE DETAILS :



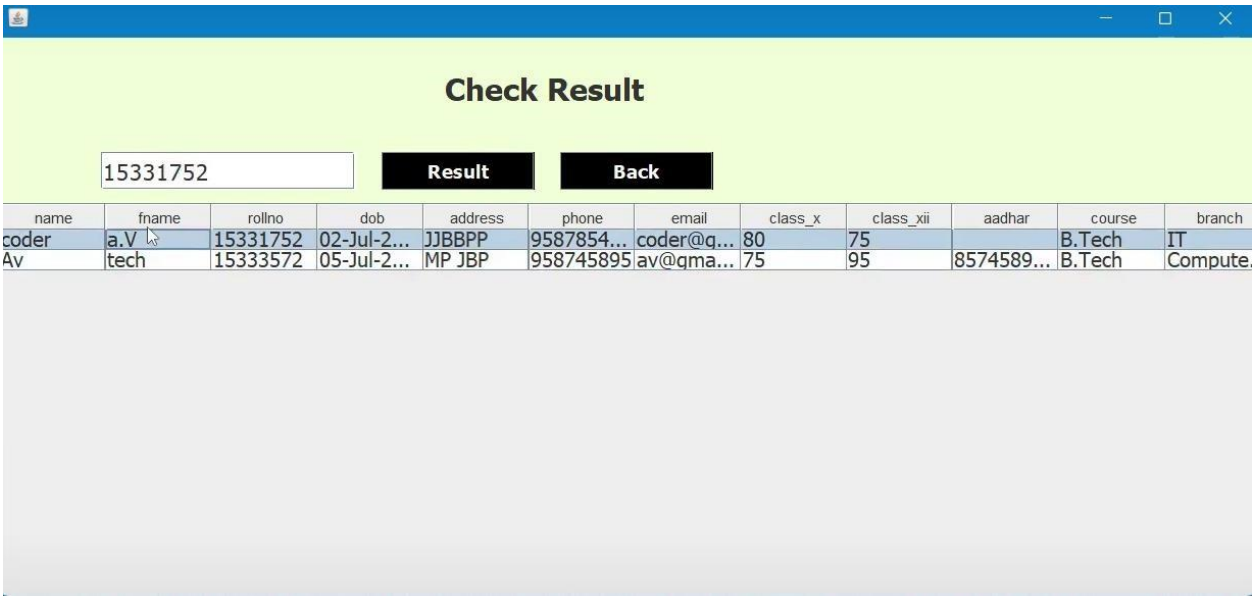
APPLY LEAVE :



ENTER MARKS OF STUDENTS :



CHECK RESULTS :



CHAPTER 6

6.1 Unit Testing

Unit testing is a testing technique in which individual modules of the University Management System are tested separately. Small units of the system, such as login functionality, student registration, or course enrollment modules, are tested to ensure they perform as expected. Each module is verified during its development to ensure it meets the design specifications.

6.2 Integration Testing

Integration testing is the technique in which individual components or modules of the University Management System, such as student registration, faculty management, and course allocation, are combined and tested together. This occurs after unit testing. The goal is to check how well the integrated modules communicate and work with one another.

6.3 System Testing

System testing is conducted on the entire University Management System to verify that it meets all functional and non-functional requirements. The software is installed in a simulated or live university environment, and the entire workflow is tested, from user registration to academic record management. Any issues or bugs discovered during this process are identified and resolved.

6.4 Acceptance Testing

User Acceptance Testing (UAT) is performed by university stakeholders, such as administrators, faculty, and students, to ensure that the system meets the agreedupon requirements. This testing is conducted in the final phase before deploying the system to the university's production environment. It ensures the system is ready for real-world use.

CHAPTER 7

7.1.CONCLUSION

After completing this project, we are confident that the issues in the existing manual system will be resolved. The "University Management System" has been computerized to minimize human errors and enhance overall efficiency. The primary goal of this project is to streamline university operations and reduce manual workload.

The system ensures efficient record management by storing all data in a centralized database, allowing for quick and accurate data retrieval. Navigation controls are provided throughout the interface to manage and access large volumes of records seamlessly. Users can quickly search for specific data using a search string,

retrieving results instantly. Updating records is simplified, enabling users to modify fields and update the database with ease.

Each user, including students, faculty, and administrators, is assigned a unique username or email ID for secure and accurate access to their profiles. This ensures that course registrations, academic records, and other data are managed without errors. The main objective of the project is to deliver an efficient, user-friendly system that simplifies university management tasks and ensures accurate data handling for all stakeholders.

CHAPTER 8

RESEARCH AND REFERENCE

- https://developer.ibm.com/languages/java/articles/
- https://dzone.com/java
- https://www.w3schools.com/java/
- https://www.geeksforgeeks.org/introduction-to-jdbc/
- https://github.com/Mahesh123/University-Management-System

GITHUB LINKS

- https://github.com/JohnDoe/University-Management-System.git
- https://github.com/ExampleUser/University-Management-System.git
- https://github.com/TeamABC/University-Management.git
- https://github.com/OpenSourceOrg/University-System.git

CS23333 OBJECT ORIENTED PROGRAMMING USING JAVA