

STUDENT MANAGEMENT SYSTEM



Shaheed Zulfikar Ali Bhutto
Institute of Science & Technology

DOCUMENTATION

DATABASE SYSTEM

GROUP MEMBERS

NAME	CLASS	REGISTRATION NUMBER
ARISH AMIN	BSCS-4B	2112140
MUBEEN NAUSHAD	BSCS-4B	2112151

ACKNOWLEDGMENT

We are really grateful because we managed to complete our project within the time given by our teacher [**Sir Abid Ali & Sir Faraz Afsar**]. This assignment cannot be completed without the effort and co-operation from our group members, Group members:
[Mubeen Naushad] & [Arish Amin].

We also sincerely thank our teacher [**Sir Abid Ali & Sir Faraz Afsar**] for the guidance and encouragement in finishing this project and also for teaching us in this course.

Last but not the least, we would like to express gratitude to our friends and respondents for the support and willingness directly or indirectly to spend some times with us to fill in the questionnaires.

I would also like to thank my group member for their support, cooperation, and the motivation they have given me to complete this report. I am honestly overwhelmed in all humbleness and gratefulness to acknowledge my dept to all those who have helped me to put these ideas, well above the level of simplicity and into something concrete.

STUDENT MANAGEMENT SYSTEM

Table of Contents

Introduction.....	4
Objectives of the project.....	5
Topic of our Project.....	6
Hardware/ Software Requirements	7
WORK ANALYSIS	8
CODE SNIPPETS.....	9
EXPLANATION OF CODE.....	22

STUDENT MANAGEMENT SYSTEM

PROJECT INTRODUCTION

The Student Management System (SMS) project is an advanced software application built on the Java programming language and powered by the MySQL database. This comprehensive system offers a wide range of features designed to streamline the management of student-related information within educational institutions. With its intuitive user interface and robust functionality, the SMS project aims to revolutionize the way student data is stored, accessed, and analyzed.

The SMS project encompasses a plethora of essential features, starting with a visually appealing Loading Screen that provides a seamless and engaging user experience. This loading screen ensures that users have a smooth transition into the system while necessary resources are loaded.

The Login Panel is a crucial component of the SMS project, ensuring that only authorized users can access and manipulate student data. Through secure authentication mechanisms, such as username and password, the login panel safeguards the system from unauthorized access and maintains data privacy.

One of the core functionalities of the SMS project is the ability to efficiently manage student information. This includes features such as Student Adding, which allows administrators to easily register new students into the system. Additionally, the system provides options for Deleting and Updating student records, ensuring that the database remains up-to-date and accurate.

The SMS project also offers comprehensive course management capabilities. Administrators can effortlessly Add, Delete, and Update courses within the system. This feature facilitates easy organization and maintenance of the courses offered by the institution, enabling seamless integration with the student records.

To enhance academic assessment, the SMS project incorporates the Scores Adding and Updating functionality. This feature allows teachers to record and update student scores according to specific courses. By linking scores to courses, the system ensures a streamlined and organized approach to grade management.

A notable feature of the SMS project is the ability to generate comprehensive reports. The system facilitates the calculation of the Cumulative Grade Point Average (CGPA) of students within a particular semester. This report provides valuable insights into student performance and allows for effective evaluation. Moreover, the system offers the option to export the database in PDF format, allowing for convenient sharing and printing of reports.

In conclusion, the Student Management System project offers a robust and efficient solution for managing student-related information. With features including a Loading Screen, Login Panel, Student Adding, Deleting, and Updating, Course Adding, Deleting, and Updating, Scores Adding and Updating, CGPA calculation, Bar chart for tracking progress of Student and PDF report generation, the SMS project revolutionizes student data management in educational institutions. By harnessing the power of Java and MySQL, this system provides a user-friendly interface and advanced functionalities that streamline administrative processes and enhance data analysis, ensuring an optimized and comprehensive approach to student management.

STUDENT MANAGEMENT SYSTEM

Objectives of the Project

The primary objectives of the Student Management System project are as follows:

a) **Automation and Efficiency:** The project aims to automate manual processes involved in student management, such as registration, attendance tracking, grade management, and report generation. By eliminating tedious paperwork and streamlining administrative tasks, the system enhances efficiency and saves time for both students and administrators.

b) **Centralized Database:** The project focuses on creating a centralized database to store student-related information. This ensures easy access, retrieval, and manipulation of data, reducing redundancy and data inconsistencies. A well-organized database enhances data integrity and facilitates seamless data management.

c) **User-Friendly Interface:** The project aims to provide a user-friendly interface that is intuitive and easy to navigate. Students, teachers, and administrators should be able to interact with the system effortlessly, ensuring a positive user experience. This includes features like clear menus, informative prompts, and logical workflows.

d) **Data Security and Privacy:** The project emphasizes implementing robust security measures to protect student data. User authentication and authorization mechanisms are incorporated to ensure that only authorized individuals have access to sensitive information. By maintaining data confidentiality and integrity, the system instills trust and ensures compliance with data protection regulations.

e) **Accurate Reports and Analysis:** The project focuses on generating accurate and comprehensive reports related to student performance, attendance, and grades. These reports provide valuable insights for teachers, administrators, and parents, enabling informed decision-making and effective evaluation of student progress. The system aims to generate reports in a format that is easily understandable and shareable.

e) **Progress Tracking:** It also includes a bar chart which displays the current student semester score against it's courses and help student to identify their weak points and make progress.

STUDENT MANAGEMENT SYSTEM

Topic of the Project

The topic of our project is the "Student Management System." This project encompasses the development of a software application that facilitates efficient management of student-related information in educational institutions. It covers various aspects such as student registration which includes Adding, Updating, Deleting and Reading student's information. Moreover, it also provide the facility to add courses against student's ID. In Contrast with, it also help keeping the track of score by adding GPA of respective subjects, And a Report generating system which calculates the CGPA of semesters and a bar chart which help to keep track of student current weakness and strong points.

Hardware/Software Requirements

Hardware Requirements:

Computer or server with sufficient processing power and memory to handle the application and database operations smoothly.

Adequate storage capacity to store the database and application files.

Network infrastructure, if the system needs to be accessed by multiple users across different locations.

Software Requirements:

Java Development Kit (JDK) to develop and run the Java-based application.

Integrated Development Environment (IDE) such as NetBeans or IntelliJ IDEA for coding, debugging, and project management.










MySQL database management system for storing and managing the student data (MySQL WorkBench).

JDBC (Java Database Connectivity) driver to establish a connection between the Java application and the MySQL database.

Operating system compatible with the chosen software stack (e.g., Windows, Linux, macOS).

Ensuring that the hardware and software requirements are met will facilitate the successful development and deployment of the Student Management System project, enabling efficient student data management within educational institutions.

Work Analysis

Task	Mubeen Naushad	ARISH AMIN
Analysis		
Design		
Coding		
Testing		
Documentation		

STUDENT MANAGEMENT SYSTEM

Code Snippets

- Student Class

```
public class Students {

    Connection con = MyConnection.getConnection();
    PreparedStatement ps;

    public int getMax() {
        int id = 0;
        Statement st;
        try {
            st = con.createStatement();
            ResultSet rs = st.executeQuery(string: "select max(id) from student");
            while (rs.next()) {

                id = rs.getInt(i: 1);
            }
        } catch (SQLException ex) {

            Logger.getLogger(name: Students.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
        }
        return id + 1;
    }
}
```

```
public void insert(int id, String name, String date, String gender, String email,
    String phone, String father, String mother, String address1, String address2, String image) {
    String sql = "insert into student Values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
    try {
        ps = con.prepareStatement(string: sql);
        ps.setInt(i: 1, il: id);
        ps.setString(i: 2, string: name);
        ps.setString(i: 3, string: date);
        ps.setString(i: 4, string: gender);
        ps.setString(i: 5, string: email);
        ps.setString(i: 6, string: phone);
        ps.setString(i: 7, string: father);
        ps.setString(i: 8, string: mother);
        ps.setString(i: 9, string: address1);
        ps.setString(i: 10, string: address2);
        ps.setString(i: 11, string: image);

        if (ps.executeUpdate() > 0) {

            JOptionPane.showMessageDialog(parentComponent:null, message:"New Student Added Successfully");
        }

    } catch (SQLException ex) {
        Logger.getLogger(name: Students.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

STUDENT MANAGEMENT SYSTEM

```
public boolean isEmailExist(String email) {
    try {
        ps = con.prepareStatement(string: "select * from student where email = ?");
        ps.setString(i: 1, string: email);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return true;
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Students.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    return false;
}

public boolean isPhoneExist(String phone) {
    try {
        ps = con.prepareStatement(string: "select * from student where phone = ?");
        ps.setString(i: 1, string: phone);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return true;
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Students.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    return false;
}

public boolean isIdExist(int id) {
    try {
        ps = con.prepareStatement(string: "select * from student where id = ?");
        ps.setInt(i: 1, i1: id);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return true;
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Students.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    return false;
}
```

STUDENT MANAGEMENT SYSTEM

```
public void getStudentValue(JTable table, String searchValue) {
    String sql = "select * from student where concat(name, email, phone) like ? order by id desc";
    try {
        ps = con.prepareStatement(string: sql);
        ps.setString(i: 1, "%" + searchValue + "%");
        ResultSet rs = ps.executeQuery();
        DefaultTableModel model = (DefaultTableModel) table.getModel();
        Object[] row;
        while (rs.next()) {
            row = new Object[11];
            row[0] = rs.getInt(i: 1);
            row[1] = rs.getString(i: 2);
            row[2] = rs.getString(i: 3);
            row[3] = rs.getString(i: 4);
            row[4] = rs.getString(i: 5);
            row[5] = rs.getString(i: 6);
            row[6] = rs.getString(i: 7);
            row[7] = rs.getString(i: 8);
            row[8] = rs.getString(i: 9);
            row[9] = rs.getString(i: 10);
            row[10] = rs.getString(i: 11);
            model.addRow(rowData: row);
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Students.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    }
}
```

STUDENT MANAGEMENT SYSTEM

```
public void update(int id, String name, String date, String gender, String email, String phone,
    String father, String mother, String address1, String address2, String image) {

    String sql = "update student set name = ?, date_of_birth = ?, gender = ?, email = ?, phone = ?,"
        + " father_name = ?, mother_name = ?, address1 = ?, address2 = ?, image_path = ? where id = ?";
    try {
        ps = con.prepareStatement(string: sql);
        ps.setString(i: 1, string: name);
        ps.setString(i: 2, string: date);
        ps.setString(i: 3, string: gender);
        ps.setString(i: 4, string: email);
        ps.setString(i: 5, string: phone);
        ps.setString(i: 6, string: father);
        ps.setString(i: 7, string: mother);
        ps.setString(i: 8, string: address1);
        ps.setString(i: 9, string: address2);
        ps.setString(i: 10, string: image);
        ps.setInt(i: 11, i1: id);

        if (ps.executeUpdate() > 0) {
            JOptionPane.showMessageDialog(parentComponent:null, message:"Student Data Updated Successfully.");
        }

    } catch (SQLException ex) {
        Logger.getLogger(name: Students.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

```
public void delete(int id) {
    int yesOrNo = JOptionPane.showConfirmDialog(parentComponent:null, message:"Course and score records will also be deleted", title:"Student Delete",
        optionType: JOptionPane.OK_CANCEL_OPTION, messageType:0);

    if (yesOrNo == JOptionPane.OK_OPTION) {
        try {
            ps = con.prepareStatement(string: "delete from student where id = ?");
            ps.setInt(i: 1, i1: id);
            if (ps.executeUpdate() > 0) {
                JOptionPane.showMessageDialog(parentComponent:null, message:"Student deleted successfully.");
            }
        } catch (SQLException ex) {
            Logger.getLogger(name: Students.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
        }
    }
}
```

STUDENT MANAGEMENT SYSTEM

● Course

```
public class Course {
    Connection con = MyConnection.getConnection();
    PreparedStatement ps;

    public int getMax() {
        int id = 0;
        Statement st;
        try {
            st = con.createStatement();
            ResultSet rs = st.executeQuery(string: "select max(id) from course");
            while (rs.next()) {

                id = rs.getInt(i: 1);
            }
        } catch (SQLException ex) {

            Logger.getLogger(name: Course.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);

        }
        return id + 1;
    }

    public boolean getId(int id){
        try {
            ps = con.prepareStatement(string: "select * from student where id = ?");
            ps.setInt(i: 1, i1: id);
            ResultSet rs = ps.executeQuery();
            if(rs.next()){
                Home.jTextField39.setText(v: String.valueOf(i: rs.getInt(i: 1)));
                return true;
            }else {
                JOptionPane.showMessageDialog(parentComponent:null, message:"Student id doesn't exist");
            }
        } catch (SQLException ex) {
            Logger.getLogger(name: Course.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
        }
        return false;
    }
}
```

STUDENT MANAGEMENT SYSTEM

```
public int countSemester(int id){
    int total = 0;
    try {
        ps = con.prepareStatement(string: "select count(*) as 'Total' from course where student_id = ?");
        ps.setInt(i: 1, il: id);
        ResultSet rs = ps.executeQuery();
        while(rs.next()){
            total = rs.getInt(i: 1);
        }
        if(total == 8){
            JOptionPane.showMessageDialog(parentComponent:null, message:"This student has completed all the courses");
            return -1;
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Course.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    return total;
}

public boolean isSemesterExist(int sid, int semesterNo) {
    try {
        ps = con.prepareStatement(string: "select * from course where student_id = ? and semester = ?");
        ps.setInt(i: 1, il: sid);
        ps.setInt(i: 2, il: semesterNo);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return true;
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Course.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    return false;
}
```

```
public boolean isCourseExist(int sid, String courseNo, String course) {
    String sql = "select * from course where student_id = ? and "+courseNo+" = ?";
    try {
        ps = con.prepareStatement(string: sql);
        ps.setInt(i: 1, il: sid);
        ps.setString(i: 2, string: course);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return true;
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Course.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    return false;
}

public void insert(int id, int sid, int semester, String course1, String course2, String course3, String course4, String course5) {
    String sql = "insert into course Values(?, ?, ?, ?, ?, ?, ?)";
    try {
        ps = con.prepareStatement(string: sql);
        ps.setInt(i: 1, il: id);
        ps.setInt(i: 2, il: sid);
        ps.setInt(i: 3, il: semester);
        ps.setString(i: 4, string: course1);
        ps.setString(i: 5, string: course2);
        ps.setString(i: 6, string: course3);
        ps.setString(i: 7, string: course4);
        ps.setString(i: 8, string: course5);
        if (ps.executeUpdate() > 0) {
            JOptionPane.showMessageDialog(parentComponent:null, message:"New Course Added Successfully");
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Course.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

STUDENT MANAGEMENT SYSTEM

```
public boolean isEmailExist(String email) {
    try {
        ps = con.prepareStatement(string: "select * from student where email = ?");
        ps.setString(i: 1, string: email);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return true;
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Students.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    return false;
}

public void getCourseValue(JTable table, String searchValue) {
    String sql = "select * from course where concat(id, student_id, semester) like ? order by id desc";
    try {
        ps = con.prepareStatement(string: sql);
        ps.setString(i: 1, "%" + searchValue + "%");
        ResultSet rs = ps.executeQuery();
        DefaultTableModel model = (DefaultTableModel) table.getModel();
        Object[] row;
        while (rs.next()) {
            row = new Object[8];
            row[0] = rs.getInt(i: 1);
            row[1] = rs.getString(i: 2);
            row[2] = rs.getString(i: 3);
            row[3] = rs.getString(i: 4);
            row[4] = rs.getString(i: 5);
            row[5] = rs.getString(i: 6);
            row[6] = rs.getString(i: 7);
            row[7] = rs.getString(i: 8);

            model.addRow(rowData: row);
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Course.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
}
```

```
public void delete(int id, int semNo) {
    int yesOrNo = JOptionPane.showConfirmDialog(parentComponent:null, message:"Score records will also be deleted", title:"Course Delete",
        optionType: JOptionPane.OK_CANCEL_OPTION, messageType:0);
    if (yesOrNo == JOptionPane.OK_OPTION) {
        try {
            ps = con.prepareStatement(string: "delete from course where student_id = ? and semester = ?");
            ps.setInt(i: 1, i1: id);
            ps.setInt(i: 2, i1: semNo);
            if (ps.executeUpdate() > 0) {
                JOptionPane.showMessageDialog(parentComponent:null, message:"Student deleted successfully.");
            }
        } catch (SQLException ex) {
            Logger.getLogger(name: Course.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
        }
    }
}
```

STUDENT MANAGEMENT SYSTEM

● Score

```
public class Score {
    Connection con = MyConnection.getConnection();
    PreparedStatement ps;

    public int getMax() {
        int id = 0;
        Statement st;
        try {
            st = con.createStatement();
            ResultSet rs = st.executeQuery("select max(id) from Score");
            while (rs.next()) {
                id = rs.getInt(1);
            }
        } catch (SQLException ex) {
            Logger.getLogger(name: Score.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
        }
        return id + 1;
    }
}
```


STUDENT MANAGEMENT SYSTEM

```
public boolean getDetails(int id,int semesterNo){
    try {
        ps = con.prepareStatement(string: "select * from course where student_id = ? and semester = ?");
        ps.setInt(i: 1, il: id);
        ps.setInt(i: 2, il: semesterNo);
        ResultSet rs = ps.executeQuery();
        if(rs.next()){
            Home.jTextField41.setText(v: String.valueOf(i: rs.getInt(i: 2)));
            Home.jTextField10.setText(v: String.valueOf(i: rs.getInt(i: 3)));
            Home.jTextField11.setText(v: rs.getString(i: 4));
            Home.jTextField12.setText(v: rs.getString(i: 5));
            Home.jTextField13.setText(v: rs.getString(i: 6));
            Home.jTextField14.setText(v: rs.getString(i: 7));
            Home.jTextField15.setText(v: rs.getString(i: 8));

            return true;
        }else {
            JOptionPane.showMessageDialog(parentComponent:null, message:"Student id or Semester ID doesn't exist");
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Score.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    return false;
}

public boolean isIdExist(int id) {
    try {
        ps = con.prepareStatement(string: "select * from Score where id = ?");
        ps.setInt(i: 1, il: id);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return true;
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Score.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    return false;
}
```

```
public boolean isSidExist(int sid, int semesterNo) {
    try {
        ps = con.prepareStatement(string: "select * from Score where student_id = ? and semester =?");
        ps.setInt(i: 1, il: sid);
        ps.setInt(i: 2, il: semesterNo);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            return true;
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Score.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }
    return false;
}
```

STUDENT MANAGEMENT SYSTEM

```
public void getScoreValue(JTable table, String searchValue) {
    String sql = "select * from score where concat(id, student_id, semester) like ? order by id desc";
    try {
        ps = con.prepareStatement(string: sql);
        ps.setString(i: 1, "%" + searchValue + "%");
        ResultSet rs = ps.executeQuery();
        DefaultTableModel model = (DefaultTableModel) table.getModel();
        Object[] row;
        while (rs.next()) {
            row = new Object[14];
            row[0] = rs.getInt(i: 1);
            row[1] = rs.getInt(i: 2);
            row[2] = rs.getInt(i: 3);
            row[3] = rs.getString(i: 4);
            row[4] = rs.getDouble(i: 5);
            row[5] = rs.getString(i: 6);
            row[6] = rs.getDouble(i: 7);
            row[7] = rs.getString(i: 8);
            row[8] = rs.getDouble(i: 9);
            row[9] = rs.getString(i: 10);
            row[10] = rs.getDouble(i: 11);
            row[11] = rs.getString(i: 12);
            row[12] = rs.getDouble(i: 13);

            row[13] = rs.getDouble(i: 14);

            model.addRow(rowData: row);
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Score.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    }
}
```

```
public void insert(int id, int sid, int semester, String course1, String course2, String course3, String course4, String course5, double score1, double score2, double score3, double score4, double score5, double average) {
    String sql = "insert into score Values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
    try {
        ps = con.prepareStatement(string: sql);
        ps.setInt(i: 1, i1: id);
        ps.setInt(i: 2, i1: sid);
        ps.setInt(i: 3, i1: semester);
        ps.setString(i: 4, string: course1);
        ps.setDouble(i: 5, d: score1);
        ps.setString(i: 6, string: course2);
        ps.setDouble(i: 7, d: score2);
        ps.setString(i: 8, string: course3);
        ps.setDouble(i: 9, d: score3);
        ps.setString(i: 10, string: course4);
        ps.setDouble(i: 11, d: score4);
        ps.setString(i: 12, string: course5);
        ps.setDouble(i: 13, d: score5);
        ps.setDouble(i: 14, d: average);
        if (ps.executeUpdate() > 0) {
            JOptionPane.showMessageDialog(parentComponent: null, message: "Scores Added Successfully");
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Score.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    }
}
```

STUDENT MANAGEMENT SYSTEM

```
public void update(int id, double score1, double score2, double score3, double score4, double score5, double average) {
String sql = "update score set score1 = ?, score2 = ?, score3 = ?, score4 = ?, score5 = ?, average = ? where id = ?";
try {
    ps = con.prepareStatement(string: sql);

    ps.setDouble(i: 1, d: score1);
    ps.setDouble(i: 2, d: score2);
    ps.setDouble(i: 3, d: score3);
    ps.setDouble(i: 4, d: score4);
    ps.setDouble(i: 5, d: score5);
    ps.setDouble(i: 6, d: average);
    ps.setInt(i: 7, i1: id);

    if (ps.executeUpdate() > 0) {
        JOptionPane.showMessageDialog(parentComponent:null, message:"Score Updated Successfully.");
    }

} catch (SQLException ex) {
    Logger.getLogger(name: Score.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
}

}

public void delete(int id, int semNo) {
int yesOrNo = JOptionPane.showConfirmDialog(parentComponent:null, message:"Score records will be deleted", title: "Score Delete",
optionType: JOptionPane.OK_CANCEL_OPTION, messageType:0);
if (yesOrNo == JOptionPane.OK_OPTION) {

    try {
        ps = con.prepareStatement(string: "delete from score where student_id = ? and semester = ?");
        ps.setInt(i: 1, i1: id);
        ps.setInt(i: 2, i1: semNo);
        if (ps.executeUpdate() > 0) {
            JOptionPane.showMessageDialog(parentComponent:null, message:"Student deleted successfully.");
        }
    } catch (SQLException ex) {
        Logger.getLogger(name: Course.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
    }

}

}

}
```

STUDENT MANAGEMENT SYSTEM

● Report

```
public class Report {

    Connection con = MyConnection.getConnection();
    PreparedStatement ps;

    public boolean isIdExist(int sid) {
        try {
            ps = con.prepareStatement(string: "select * from Score where Student_id = ?");
            ps.setInt(i: 1, i1: sid);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                return true;
            }
        } catch (SQLException ex) {
            Logger.getLogger(name: Report.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
        }
        return false;
    }

    public void getScoreValue(JTable table,int sid) {
        String sql = "select * from score where student_id=?";
        try {
            ps = con.prepareStatement(string: sql);
            ps.setInt(i: 1, i1: sid);
            ResultSet rs = ps.executeQuery();
            DefaultTableModel model = (DefaultTableModel) table.getModel();
            Object[] row;
            while (rs.next()) {
                row = new Object[14];
                row[0] = rs.getInt(i: 1);
                row[1] = rs.getInt(i: 2);
                row[2] = rs.getInt(i: 3);
                row[3] = rs.getString(i: 4);
                row[4] = rs.getDouble(i: 5);
                row[5] = rs.getString(i: 6);
                row[6] = rs.getDouble(i: 7);
                row[7] = rs.getString(i: 8);
                row[8] = rs.getDouble(i: 9);
                row[9] = rs.getString(i: 10);
                row[10] = rs.getDouble(i: 11);
                row[11] = rs.getString(i: 12);
                row[12] = rs.getDouble(i: 13);

                row[13] = rs.getDouble(i: 14);

                model.addRow(rowData:row);
            }
        } catch (SQLException ex) {
            Logger.getLogger(name: Report.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
        }
    }
}
```

STUDENT MANAGEMENT SYSTEM

```
public double getCGPA(int sid){  
    double cgpa = 0.0;  
    Statement st;  
  
    try{  
        st=con.createStatement();  
        ResultSet rs =st.executeQuery("select avg(average) from score where student_id =" +sid + "");  
        if(rs.next()){  
            cgpa=rs.getDouble(1);  
        }  
    }catch(SQLException ex){  
        Logger.getLogger(name: Report.class.getName()).log(level: Level.SEVERE,msg:null,thrown: ex);  
    }  
    return cgpa;  
}
```

STUDENT MANAGEMENT SYSTEM

- Connection Sql

```
package db;

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

public class MyConnection {

    public static final String username="root";
    public static final String password="Pass123456789";
    private static final String dataConn ="jdbc:mysql://localhost:3306/student_management";
    public static Connection con=null;

    public static Connection getConnection(){
        try {
            Class.forName(className: "com.mysql.cj.jdbc.Driver");
            con = DriverManager.getConnection(url:dataConn,user: username,password);
        } catch (Exception ex) {
            System.out.println(x: ex.getMessage());
        }

        return con;
    }

}
```

Explanation of the code

Course Functions:

getMax():

This function retrieves the maximum value of the id column from the course table. It executes a SQL query to select the maximum value using the max() function. The retrieved value is returned as an integer, incremented by 1
getId(int id):

getId(int id):

This function checks if a student with the given id exists in the student table.
It prepares a SQL statement with a parameterized query to select a student by id.
The student's information is retrieved from the database and displayed in a text field (Home.JTextField39) if the student exists. Returns true if the student exists, otherwise displays a message and returns false.

countSemester(int id):

This function counts the number of courses (semesters) associated with a particular student.
It prepares a SQL statement with a parameterized query to count the number of courses using count(*).
The total count is returned as an integer.
If the total count is equal to 8, it displays a message indicating that the student has completed all courses and returns -1.

isSemesterExist(int sid, int semesterNo):

This function checks if a specific semester exists for a given student.
It prepares a SQL statement with parameterized queries to select a course by student_id and semester.
If a row is found in the result set, indicating the semester exists, it returns true; otherwise, it returns false.

isCourseExist(int sid, String courseNo, String course):

This function checks if a specific course exists for a given student.
It prepares a SQL statement with parameterized queries to select a course by student_id and the provided courseNo.
If a row is found in the result set, indicating the course exists, it returns true; otherwise, it returns false.

insert(int id, int sid, int semester, String course1, String course2, String course3, String course4, String course5):

This function inserts a new course record into the course table.
It prepares an SQL statement with parameterized queries to insert values into the respective columns of the table.
If the execution of the statement is successful (i.e., executeUpdate() returns a value greater than 0), a success message is displayed.

STUDENT MANAGEMENT SYSTEM

isEmailExist(String email):

This function checks if a student with the given email already exists in the student table.

It prepares a SQL statement with a parameterized query to select a student by email.

If a row is found in the result set, indicating the email exists, it returns true; otherwise, it returns false.

getCourseValue(JTable table, String searchValue):

This function retrieves course records from the course table and populates them into a JTable.

It prepares a SQL statement with a parameterized query to select courses based on a search value using the like operator.

The retrieved data is added to the DefaultTableModel of the provided JTable.

Each row of the result set is represented as an object array (row) and added to the model.

Score Functions:

getMax():

This function retrieves the maximum value of the id column from the Score table. It executes a SQL query to select the maximum value using the max() function. The retrieved value is returned as an integer, incremented by 1.

getDetails(int id, int semesterNo):

This function retrieves the details of a course for a specific student and semester from the course table. It prepares a SQL statement with parameterized queries to select a course by student_id and semester. The course details are retrieved from the database and displayed in different text fields (Home(jTextField41, Home(jTextField10, etc.) if the course exists. Returns true if the course exists, otherwise displays a message and returns false.

isIdExist(int id):

This function checks if a specific ID exists in the Score table. It prepares a SQL statement with a parameterized query to select a score by id. If a row is found in the result set, indicating the ID exists, it returns true; otherwise, it returns false.

isSidExist(int sid, int semesterNo):

This function checks if a specific student ID and semester exist in the Score table. It prepares a SQL statement with parameterized queries to select a score by student_id and semester. If a row is found in the result set, indicating the student ID and semester exist, it returns true; otherwise, it returns false.

getScoreValue(JTable table, String searchValue):

This function retrieves score records from the score table and populates them into a JTable. It prepares a SQL statement with a parameterized query to select scores based on a search value using the like operator. The retrieved data is added to the DefaultTableModel of the provided JTable. Each row of the result set is represented as an object array (row) and added to the model.

STUDENT MANAGEMENT SYSTEM

insert(int id, int sid, int semester, String course1, String course2, String course3, String course4, String course5, double score1, double score2, double score3, double score4, double score5, double average):

This function inserts a new score record into the score table.

It prepares an SQL statement with parameterized queries to insert values into the respective columns of the table.

If the execution of the statement is successful (i.e., executeUpdate() returns a value greater than 0), a success message is displayed.

update(int id, double score1, double score2, double score3, double score4, double score5, double average):

This function updates the scores of a specific record in the score table.

It prepares an SQL statement with parameterized queries to update score values based on the provided id.

If the execution of the statement is successful (i.e., executeUpdate() returns a value greater than 0), a success message is displayed.

Report Functions:

isIdExist(int sid):

This function checks if a specific student ID exists in the Score table.

It prepares a SQL statement with a parameterized query to select a score by Student_id.

If a row is found in the result set, indicating the student ID exists, it returns true; otherwise, it returns false.

getScoreValue(JTable table, int sid):

This function retrieves score records from the score table for a specific student and populates them into a JTable.

It prepares a SQL statement with a parameterized query to select scores based on the provided student_id.

The retrieved data is added to the DefaultTableModel of the provided JTable.

Each row of the result set is represented as an object array (row) and added to the model.

getCGPA(int sid):

This function calculates the CGPA (Cumulative Grade Point Average) for a specific student based on their scores.

It prepares a SQL statement with an aggregate function avg() to calculate the average of the average column in the score table for the provided student_id.

The calculated CGPA value is retrieved from the database and returned as a double.

Student Functions:

getMax():

This function retrieves the maximum ID value from the student table.
It prepares a SQL statement with an aggregate function max() to get the maximum value of the id column.
The retrieved ID is returned, incremented by 1, to generate a new ID for a new student.

insert():

This function inserts a new student record into the student table.
It prepares a SQL INSERT statement with parameterized values for all the fields in the student table.
The provided values are set using the prepared statement and executed.
If the execution is successful, a success message is displayed using JOptionPane.

isEmailExist(String email):

This function checks if a given email exists in the student table.
It prepares a SQL statement with a parameterized query to select a student by email.
If a row is found in the result set, indicating the email exists, it returns true; otherwise, it returns false.

isPhoneExist(String phone):

This function checks if a given phone number exists in the student table.
It prepares a SQL statement with a parameterized query to select a student by phone number.
If a row is found in the result set, indicating the phone number exists, it returns true; otherwise, it returns false.

isIdExist(int id):

This function checks if a given ID exists in the student table.
It prepares a SQL statement with a parameterized query to select a student by ID.
If a row is found in the result set, indicating the ID exists, it returns true; otherwise, it returns false.

getStudentValue(JTable table, String searchValue):

This function retrieves student records from the student table based on a search value (which can be a name, email, or phone number).
It prepares a SQL statement with a parameterized query using the like operator to search for records that match the provided search value.
The retrieved data is added to the DefaultTableModel of the provided JTable.
Each row of the result set is represented as an object array (row) and added to the model.

STUDENT MANAGEMENT SYSTEM

update():

This function updates an existing student record in the student table.

It prepares a SQL UPDATE statement with parameterized values for all the fields in the student table, except the ID.

The provided values and the ID are set using the prepared statement and executed.

If the execution is successful, a success message is displayed using JOptionPane.

delete():

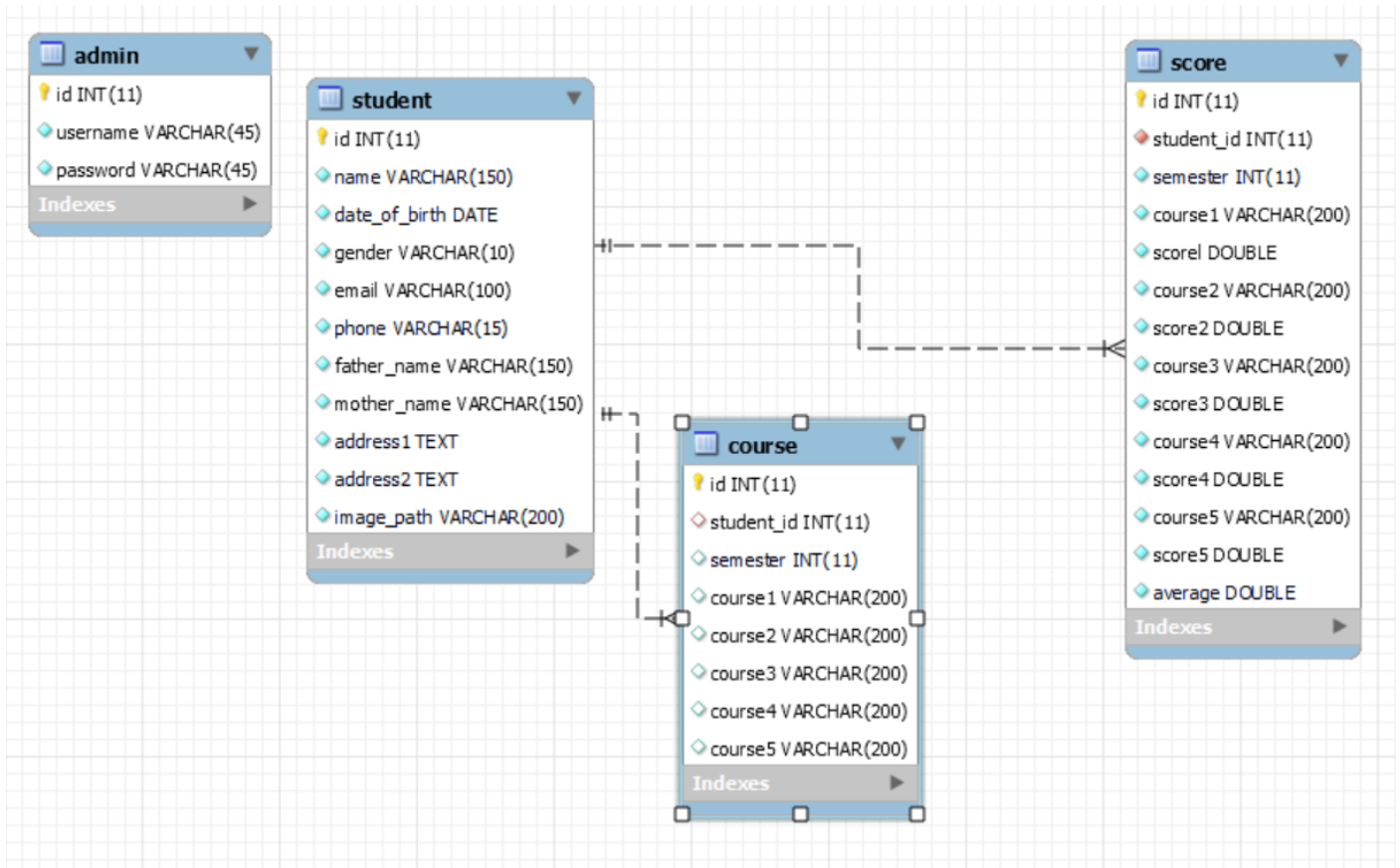
This function deletes a student record from the student table.

It prompts the user with a confirmation dialog to ensure they want to delete the student.

If the user confirms, a SQL DELETE statement is executed to delete the student record from the table.

STUDENT MANAGEMENT SYSTEM

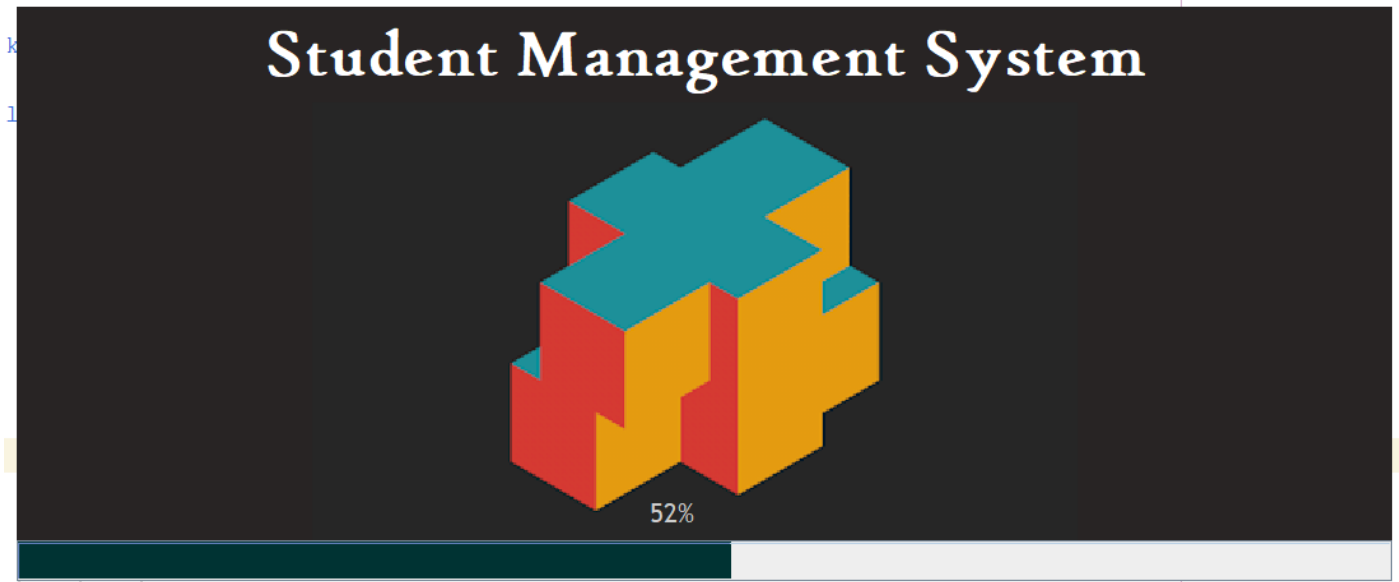
ER - Diagram:



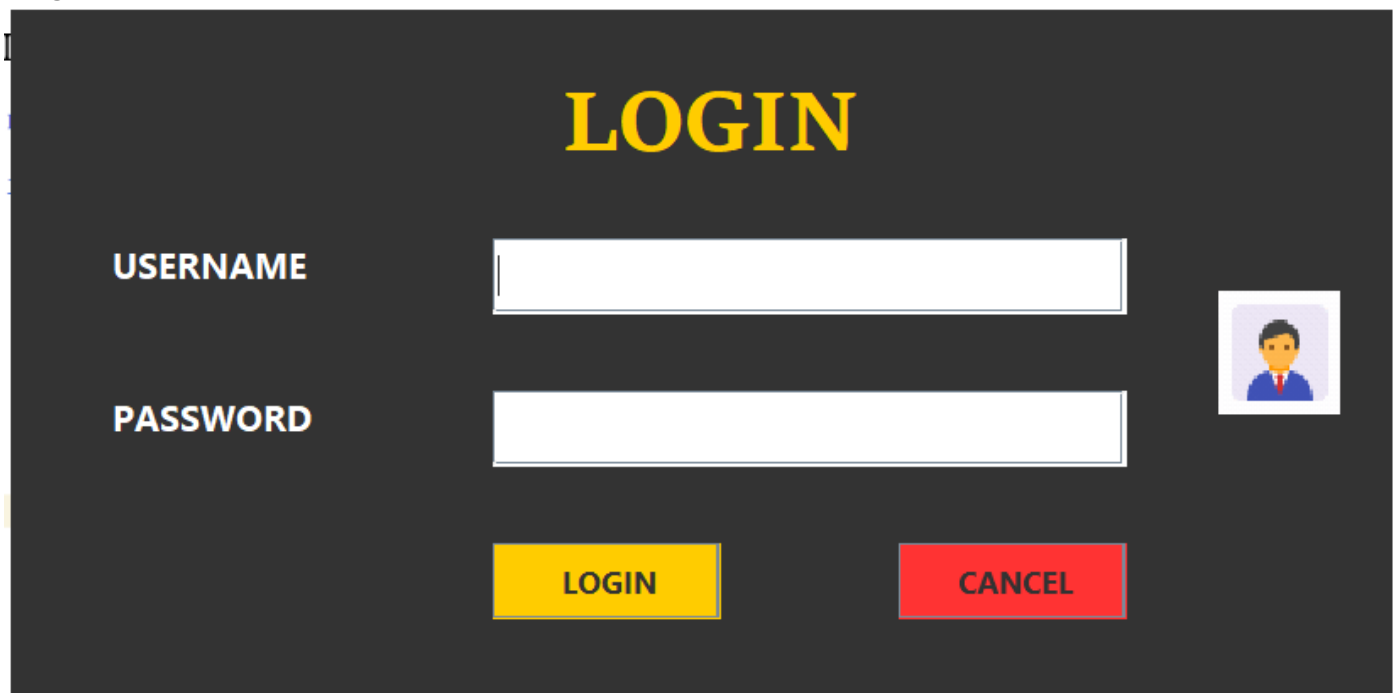
STUDENT MANAGEMENT SYSTEM

Output:

Loading Screen:



Login Panel:

The login panel has a dark gray background. At the top, the word "LOGIN" is written in a large, bold, yellow, serif font. Below this, there are two white input fields. The first field is labeled "USERNAME" in white, uppercase, sans-serif font. The second field is labeled "PASSWORD" in white, uppercase, sans-serif font. To the right of the password field is a small, square icon of a person with dark hair, wearing a blue suit and a red tie. At the bottom of the panel, there are two buttons: a yellow button labeled "LOGIN" in black, uppercase, sans-serif font, and a red button labeled "CANCEL" in black, uppercase, sans-serif font.

STUDENT MANAGEMENT SYSTEM

Student:

STUDENT MANAGEMENT SYSTEM

Student

Course

Score

Report

Student ID

3

Student Name

Date Of Birth

Gender

Male

Email Address

Phone Number

Father's Name

Mother's Name

Address Line 1

Address line 2

Image

Browse

Search Student

Search

Refresh

Student ID	Student Na...	Date of Birth	Gender	Email Addr...	Phone Nu...	Father's N...	Mother's N...	Address Li...	Address Li...	Image P...
2	Sahir	2023-06-10	Male	aa@gmail....	2412421512	Amin	Naseem	Garden	East	C:\Users\
1	Mubeen	2023-06-15	Male	mn@gmail...	1231241244	Naushad	Mumtaz	Al Azhar	Garden	C:\Users\

Add

Update

Delete

Print

Clear

Logout

Course:

STUDENT MANAGEMENT SYSTEM

Student

Course

Score

Report

Student ID

Search

ID

5

Student ID

Semester

Course 1

Fundamental of Programming

Course 2

Principle of Computers

Course 3

Application Development

Course 4

Differential Equation

Course 5

Cloud Computer

Search Student

Search

Refresh

ID	Student ID	Semester	Course 1	Course 2	Course 3	Course 4	Course 5
4	2	2	Data Structure ...	Compiler Const...	UI/UX Design	Pakistan Studies	C++ Programr
3	2	1	Fundamental of ...	Principle of Co...	Application Dev...	Differential Equ...	Machine Learn
2	1	2	Fundamental of ...	Principle of Co...	Application Dev...	Differential Equ...	Machine Learn
1	1	1	Object Oriented...	Compiler Const...	Software Testin...	Humanities	Deep Learning

Save

Delete

Print

Clear

Logout

STUDENT MANAGEMENT SYSTEM

Score:

STUDENT MANAGEMENT SYSTEM

Student

Course

Score

Report

Student's ID

Semester

Sear...

ID

1

Student ID

1

Semester

2

Course 1

Fundamental of Programming

4.0

Course 2

Principle of Computers

3.0

Course 3

Application Development

3.0

Course 4

Differential Equation

4.0

Course 5

Machine Learning

3.0

Search Student

Search

Refresh

ID	Student ID	Semester	Course 1	Score 1	Course 2	Score 2	Course 3	Score 3	Course 4	Score 4
2	1	1	Object Ori...	4.0	Compiler C...	2.0	Software T...	3.0	Humanities	2.0
1	1	2	Fundament...	4.0	Principle of...	3.0	Application...	3.0	Differential ...	4.0

Save

Update

Delete

Print

Clear

Logout

Report:

STUDENT MANAGEMENT SYSTEM

Student

Course

Score

Report

Student ID

1

Search

0.0

0.0

0.0

0.0

0.0

Bar Chart

CGPA : 3.20

ID	Student ID	Semester	Course1	Score1	Course2	Score2	Course3	Score3	Course4	Score4
1	1	2	Fundament...	4.0	Principle of...	3.0	Application...	3.0	Differential ...	4.0
2	1	1	Object Ori...	4.0	Compiler C...	2.0	Software T...	3.0	Humanities	2.0

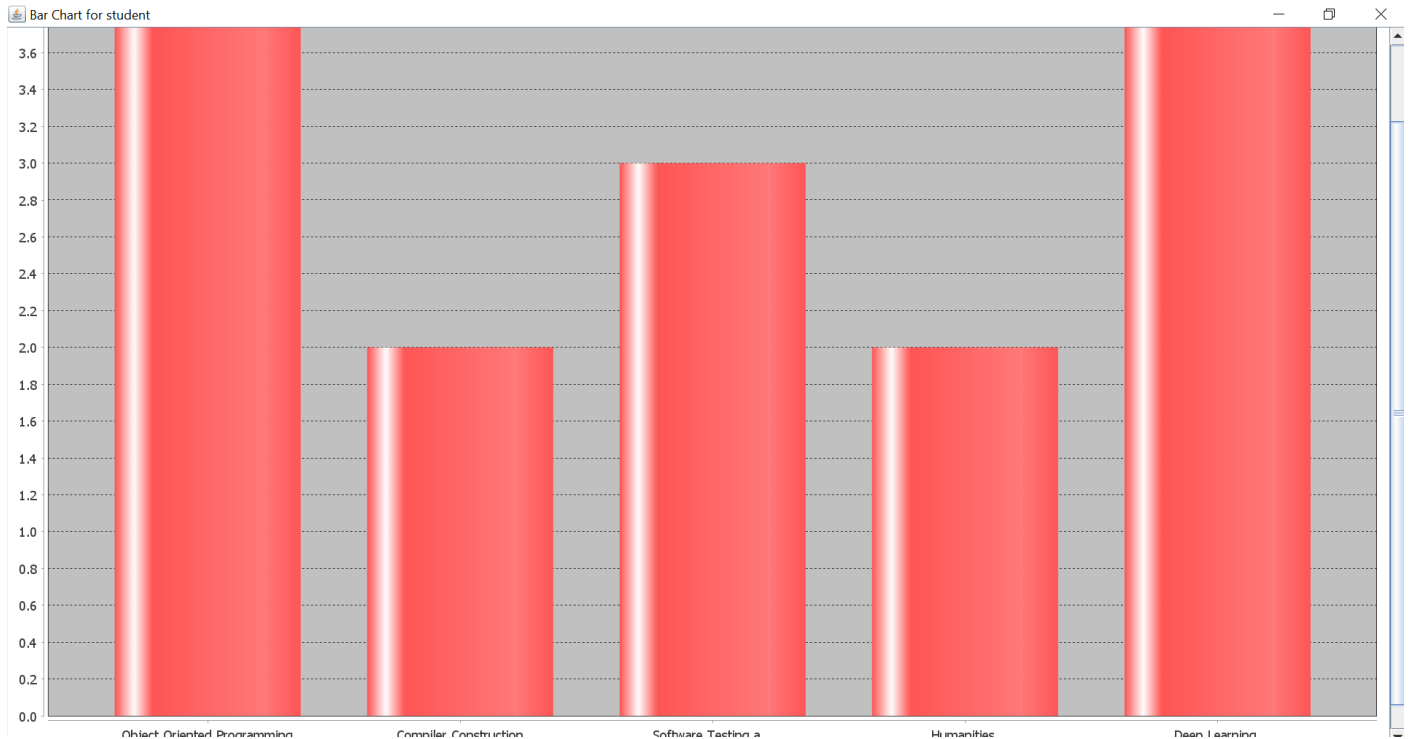
Print

Clear

Logout

STUDENT MANAGEMENT SYSTEM

BarChart:



Print Feature:

The screenshot shows the Student Management System interface. A 'Print' dialog box is open, displaying the 'General' tab. The 'Print Service' is set to 'Microsoft Print to PDF'. The 'Status' is 'Accepting jobs'. The 'Type' is 'Printer'. The 'Info' section shows 'Print To File' is checked. The 'Print Range' is set to 'All'. The 'Copies' section shows 'Number of copies' is 1 and 'Collate' is checked. The 'Print' button is highlighted.

The background interface shows a table of student scores:

Student ID	Semester	Course1	Score1	Course2	Score2	Course3
2		Fundament...	4.0	Principle of...	3.0	Application...
1		Object Ori...	4.0	Compiler C...	2.0	Software T...

Below the table, there is a 'Bar Chart' section showing the CGPA: 3.20. At the bottom right, there are buttons for 'Print', 'Clear', and 'Logout'.