MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Institution of Manipal University

Department of Computer Science and Engineering

DATABASE SYSTEM LAB – CSE 2262

A mini project report on

Student Result Management System

Group members:

Name: Aditya Nerusu
Register no: 210962196
Section: A
Roll no: 62

Name: Sai Sujan Korrapati
Register no:210962194
Section: A
Roll no: 61

Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal.
May 2023

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Institution of Manipal University

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal
13/05/2023

CERTIFICATE

This is to certify that the project titled **Student Result Management System** is a record of the bonafide work done by **Student(s) (Reg. No. 210962196 & 210962194)** submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2022-2023.

Name and Signature of Examiners:

- Dinesh Acharya, Professor, CSE Dept.

Introduction:

A student result management system is a database application designed to manage and maintain academic records of students. The system allows two types of users, the administrator and the student. The administrator has the authority to insert, update, and delete the student's academic data such as attendance, grades, and personal information. On the other hand, the student can log in to the system to view their academic results, including their marks for each subject, attendance record, and overall performance. The system is essential in streamlining the academic process and reducing manual errors that are common in traditional paper-based record-keeping. With the use of technology, the system provides a convenient and efficient way of managing academic records and monitoring student progress.

Abstract:

The Student Result Management System (SRMS) is an online platform designed to manage and store academic records efficiently and securely. The system helps school administrators, teachers, and students manage and access academic data, including marks, attendance, and other performance-related data, in a user-friendly and intuitive interface. SRMS allows school administrators to manage and maintain student records, generate reports, and monitor student performance effectively. Teachers can update marks, attendance, and other data in real-time, reducing the chances of errors and ensuring data accuracy. Students and parents can access their academic records anytime, anywhere, through a secure login, reducing the need for paper-based documents and enhancing transparency. The system's features include data management, report generation, analytics, security, and accessibility. The SRMS can improve the academic process by reducing errors, saving time, enhancing transparency, and improving communication between stakeholders. This paper discusses the design, development, and implementation of the SRMS, its features, benefits, and limitations. The results of the system's evaluation show that SRMS is an effective and efficient platform for managing academic records, and it has the potential to enhance the overall academic process.

Required software:

- NOTEPAD
- MYSQL
- NETBEANS

Problem statement:

The traditional paper-based system for managing academic records is a cumbersome and time-consuming process.
School administrators and educators face challenges in managing the academic records of a large number of students efficiently, accurately, and securely.
Additionally, students and parents face difficulties in accessing their academic records, including marks, attendance, and other performance-related data, leading to confusion and anxiety.
Therefore, there is a need for an efficient and reliable student result management system that can help administrators manage student records efficiently, reduce errors, and provide easy access to academic records for students and parents.
Such a system would streamline the academic process, provide timely and accurate information to students and parents, and help them make informed decisions about their future academic and professional careers.
This report will explore the design, development, and implementation of a student result management system and discuss its features, benefits, and limitations.
It will also evaluate the system's effectiveness and identify potential areas for improvement to enhance its performance and functionality.

Data requirements:

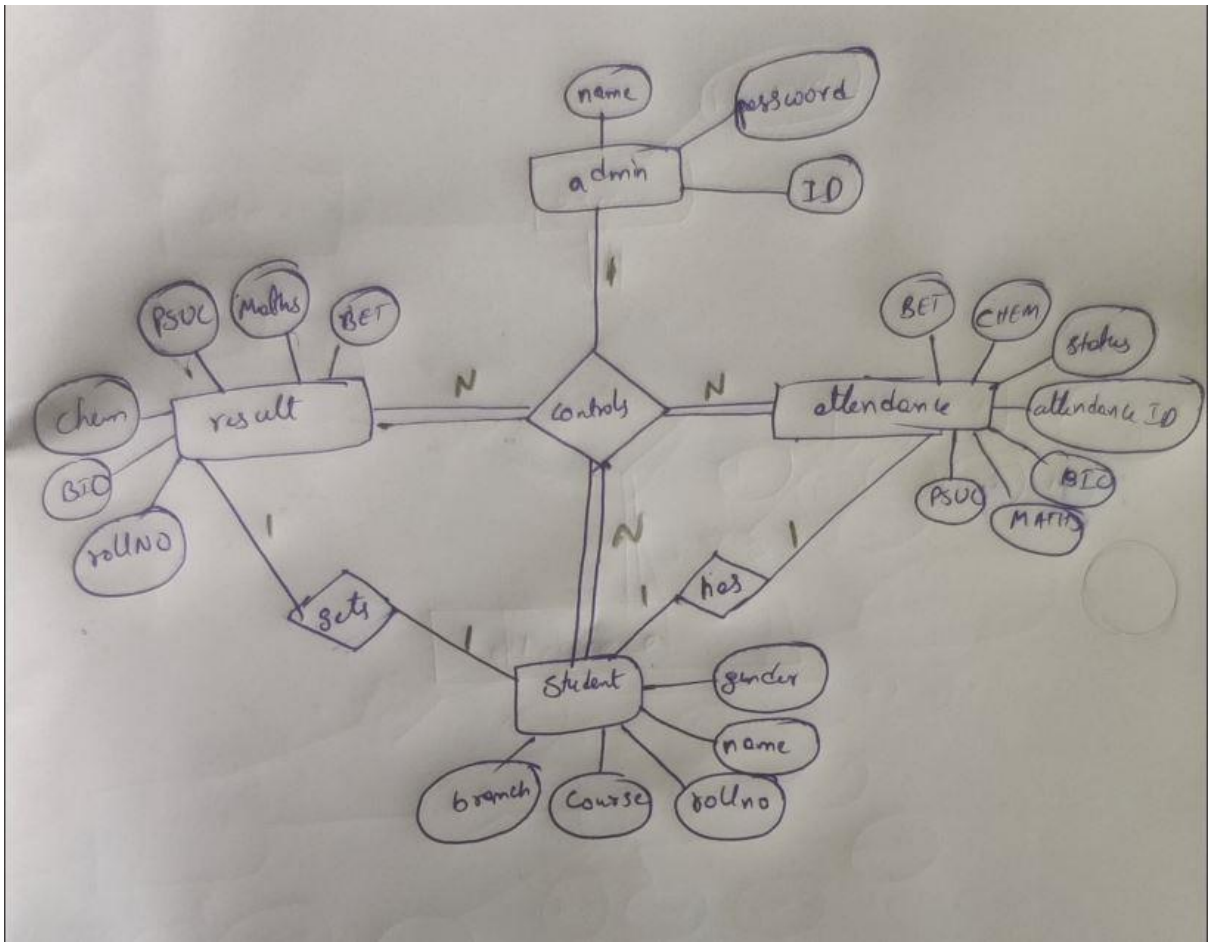
1. Admin Data: The database has information regarding login data of the admins
2. Student data: The database has information about the students who got their results.
3. Results: The database should store information about results of every student.
4. Attendance data: The database should store information about attendance of every student.
5. Log table students: The database should store information about the students whose results are updated.
6. Log table results: The database should store information about the results of students and when they are updated.

Functional requirements:

- Searching: The database allows students to search for their results and attendance.
- Updating: The database allows admin to update attendance and results of students.
- Adding: The database allows admin to add results of new students.
- Reports: The database provides reports, such as if students' attendance is alright or not.
- Security: The database should provide access control to ensure that only authorized users can access and modify the data.
- Performance: The database should be optimized for fast queries and efficient storage to handle a large amount of data and a high number of concurrent users.

Overall, the database management system should provide a comprehensive and user-friendly solution for managing a student results database, with the ability to search, update, add, and as well as generate reports and ensure security and performance.

ER Model:



Relational Tables and DDL Command:

Use srm;(database)

ADMIN TABLE:

```
desc admin;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| admin_id   | decimal(7,0) | NO   | PRI | NULL    |       |
| admin_name | varchar(20)  | YES  |     | NULL    |       |
| password   | varchar(20)  | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
```

STUDENT TABLE:

```
desc student;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| rollNo | varchar(15)  | NO   | PRI | NULL    |       |
| course | varchar(20)  | YES  |     | NULL    |       |
| branch | varchar(20)  | YES  |     | NULL    |       |
| name   | varchar(100) | YES  |     | NULL    |       |
| gender | varchar(10)  | YES  |     | NULL    |       |
| dob    | varchar(100) | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
```

RESULT TABLE:

```
desc result;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| rollNo      | varchar(15) | NO   | PRI | NULL    |       |
| biology     | varchar(10) | YES  |     | NULL    |       |
| chemistry   | varchar(10) | YES  |     | NULL    |       |
| mathematics | varchar(10) | YES  |     | NULL    |       |
| bet         | varchar(10) | YES  |     | NULL    |       |
| psuc        | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

ATTENDANCE TABLE:

```
desc attendance;
+-------------------+-------------+------+-----+---------+-------+
| Field             | Type        | Null | Key | Default | Extra |
+-------------------+-------------+------+-----+---------+-------+
| attendid          | varchar(15) | NO   | PRI | NULL    |       |
| attendance_status | varchar(40) | YES  |     | NULL    |       |
| biology           | varchar(10) | YES  |     | NULL    |       |
| chemistry         | varchar(10) | YES  |     | NULL    |       |
| mathematics       | varchar(10) | YES  |     | NULL    |       |
| bet               | varchar(10) | YES  |     | NULL    |       |
| psuc              | varchar(10) | YES  |     | NULL    |       |
+-------------------+-------------+------+-----+---------+-------+
```

SQL Queries:

- select password from admin where admin_id='"+username+"';

- INSERT INTO student(course,branch,rollNo,name,gender,dob) VALUES ('"+course+"', '"+branch+"', '"+rollno+"', '"+name+"', '"+gender+"', '"+DOB+"');
- select * from result;
- select * from student where rollno='"+rollno+"';
- INSERT INTO result(rollno,biology,chemistry,mathematics, bet,psuc) VALUES ('"+rollno+"', '"+bio+"', '"+chem+"', '"+mat+"', '"+bet+"', '"+psuc+"');
- select * from result where rollno='"+rollno+"';
- update result set result_status= 'DT' where rollno='"+rollno+"';
- update result set result_status= 'SAFE' where rollno='"+rollno+"';
- select * from log_change_takes_students;
- select * from log_change_takes_results;
- select * from student inner join result where student.rollno='"+rollno+"' and result.rollno='"+rollno+"';
- INSERT INTO attendance(attendid,biology,chemistry,mathematics, bet,psuc) VALUES ('"+rollno+"', '"+bio+"', '"+chem+"', '"+mat+"', '"+bet+"', '"+psuc+"');
- select * from attendance where attendid='"+rollno+"'

PROCEDURES:

```
CREATE PROCEDURE q2(IN rno VARCHAR(255))
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE name VARCHAR(255);
```

```
    DECLARE result_status VARCHAR(255);
    DECLARE c1 CURSOR FOR SELECT name FROM student WHERE rollno =
rno;
    DECLARE c2 CURSOR FOR SELECT result_status FROM result WHERE rollno
= rno;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    SELECT 'students';

    OPEN c1;
    read_loop1: LOOP
        FETCH c1 INTO name;
        IF done THEN
            LEAVE read_loop1;
        END IF;
        SELECT name;
    END LOOP;

    CLOSE c1;

    SELECT 'results';

    OPEN c2;
    read_loop2: LOOP
        FETCH c2 INTO result_status;
        IF done THEN
            LEAVE read_loop2;
        END IF;
        SELECT result_status;
    END LOOP;

    CLOSE c2;
END //
DELIMITER ;

CALL q2('1');
```

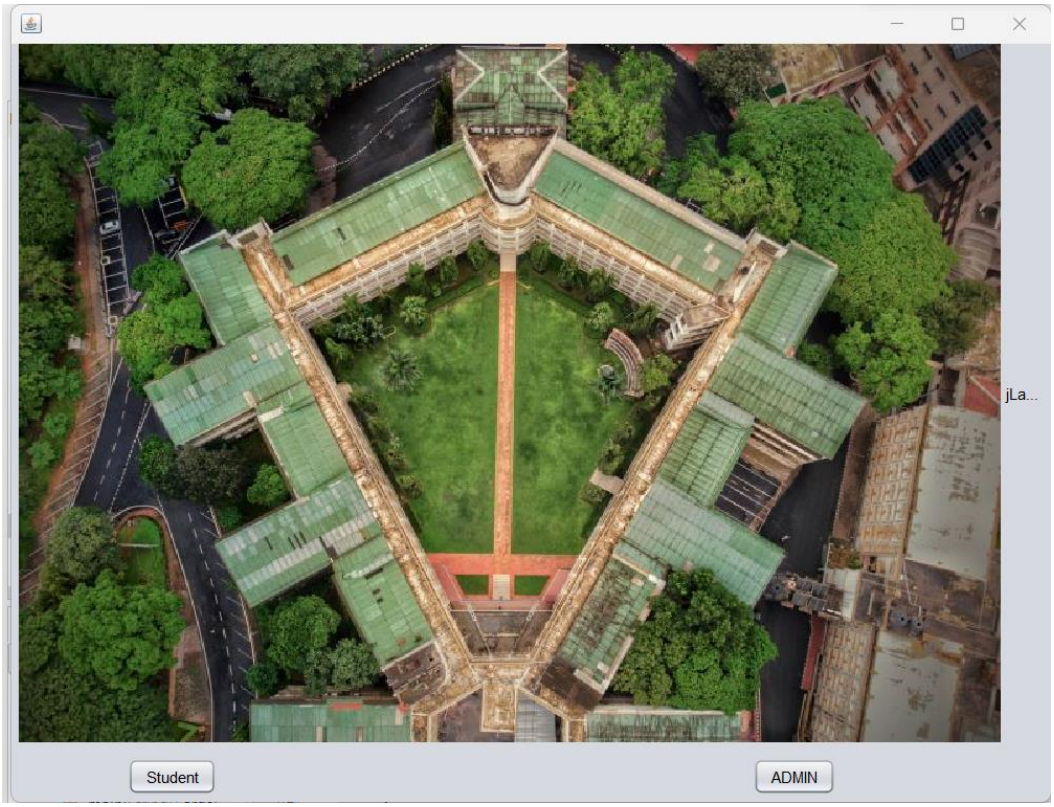
TRIGGERS:

1.CREATE TRIGGER log_change_Takes_s

```
    AFTER INSERT ON student
    FOR EACH ROW
        INSERT INTO log_change_Takes_students (date, rollno, course, branch, name,
gender, dob)
        VALUES (CURRENT_DATE(), NEW.rollno, NEW.course, NEW.branch,
NEW.name, NEW.gender, NEW.dob);

2.CREATE TRIGGER log_change_Takes_r
    AFTER INSERT ON result
    FOR EACH ROW
        INSERT INTO log_change_Takes_results (date, rollno, biology, chemistry,
mathematics, bet, psuc)
        VALUES (CURRENT_DATE(), NEW.rollno, NEW.biology, NEW.chemistry,
NEW.mathematics, NEW.bet, NEW.psuc);
```

UI DESIGN:




ADMIN LOGIN PAGE
USERNAME
PASSWORD
Login
Back


Student login portal
Roll no
Login
back

Java Functional code:

JDBC driver for connecting MYSQL Database:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
     String rollno = jTextField1.getText();
    try {
        Class.forName(className:"com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/srm",user: "root",password: "mysql");
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from student inner join result where student.rollno='"+rollno+"' and result.rollno='"+rollno+"'");
        if(rs.next()){
            setVisible(b: false);
            new studentHome1(rollno).setVisible(b: true);
        }else{
            JOptionPane.showMessageDialog(parentComponent: null, message: "Incorrect Roll Number!!");
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(parentComponent: null, message: "Connection Error!");
    }
}
```

Updating Data:

```
     String rollno = jTextField1.getText();
     String bio = jTextField2.getText();
     String chem = jTextField3.getText();
     String mat = jTextField4.getText();
     String bet = jTextField5.getText();
     String psuc = jTextField6.getText();

     try {
         Class.forName(className:"com.mysql.jdbc.Driver");
         Connection con = DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/srm",user: "root",password: "mysql");
         Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from student where rollno='"+rollno+"'");
        if(rs.next()){
            st.executeUpdate("INSERT INTO result(rollno,biology,chemistry,mathematics, bet,psuc) VALUES ('"+rollno+"', '"+bio+"', '"+chem+"', '"+mat+"',
            JOptionPane.showMessageDialog(parentComponent: null, message: "Successfully updated!!");
            setVisible(b: false);
            new insertNewResult().setVisible(b: true);

            int s1 = Integer.parseInt(s: jTextField2.getText());
             int s2 = Integer.parseInt(s: jTextField3.getText());
             int s3 = Integer.parseInt(s: jTextField4.getText());
             int s4 = Integer.parseInt(s: jTextField5.getText());
             int s5 = Integer.parseInt(s: jTextField6.getText());
            String sw="";
             ResultSet rs1 = st.executeQuery("select * from result where rollno='"+rollno+"'");
              if(s1<35 || s2<35 || s3<35 || s4<35 || s5<35){
             sw="dt";
             if(rs1.next()){
             st.executeUpdate("update result set result_status= 'DT' where rollno='"+rollno+"'" );
             }
             }
             else{
             sw="pass";
```

Sum of marks:

```
Class.forName(className:"com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/srm",user: "root",password: "mysql");
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("select * from student inner join result where student.rollno='"+rollno+"' and result.rollno='"+rollno+"'
if(rs.next()){
    jTextField1.setText(t: rs.getString(i: 2));
    jTextField2.setText(t: rs.getString(i: 4));
    jTextField3.setText(t: rs.getString(i: 5));
    jTextField4.setText(t: rs.getString(i: 3));
    jTextField6.setText(t: rs.getString(i: 6));

    jTextField7.setText(t: rs.getString(i: 8));
    jTextField8.setText(t: rs.getString(i: 9));
    jTextField9.setText(t: rs.getString(i: 10));
    jTextField10.setText(t: rs.getString(i: 11));
    jTextField11.setText(t: rs.getString(i: 12));

    int s1 = Integer.parseInt(s: jTextField7.getText());
    int s2 = Integer.parseInt(s: jTextField8.getText());
    int s3 = Integer.parseInt(s: jTextField9.getText());
    int s4 = Integer.parseInt(s: jTextField10.getText());
    int s5 = Integer.parseInt(s: jTextField11.getText());
    int total = s1+s2+s3+s4+s5;
    String s = String.valueOf(i: total);
    jTextField12.setText(t: s);
    if(s1<35 || s2<35 || s3<35 || s4<35 || s5<35)
        jTextField13.setText(t: "Fail");
    else
        jTextField13.setText(t: "Pass");


}
```

Reference:

1. JAVA The Complete Reference, Ninth Edition, Herbert Schildt, ORACLE PRESS
2. Silberschatz, Korth, Sudarshan, " Database System Concepts", McGraw-Hill, 6th Edition.
3. Iyan Bayross, "SQL, PL/SQL" 2nd/3rd Edition, BPB Publications.