

# OOPS MINI PROJECT

## AIM:

To design and develop a Student Management System using Java, which will provide a user- friendly, efficient, and reliable solution for managing student-related data in educational institutions

## Algorithm:

1. Start the Program. Initialize database (list, file, or other storage).
2. Display Menu Options Add Student View Students Search Student Update Student Delete Student Manage Attendance Manage Grades Exit
3. Process User Input: Perform the selected action  
Add: Collect and store new student details.  
View: Display all student records.  
Search: Find and show specific student details.  
Update: Modify and save student details.  
Delete: Remove student record.  
Attendance: Record/view attendance. Grades: Add/view grades.
4. Loop Menu.Return to menu until user selects "Exit."
5. Save Data.Save all changes to persistent storage (if applicable).give correct allignment

## Program:

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

```
public class StudentManagement {  
    public static void main(String[] args) throws Exception {  
        // Step 1: Establish database connection  
        Connection c =  
DriverManager.getConnection("jdbc:mysql://localhost:3306/student_management", "root", "password");  
  
        // Step 2: Create JFrame and components  
        JFrame f = new JFrame("Student Management");
```

```

// Create input fields for Name, Age, and Course
JTextField n = new JTextField(10); // Name text field
JTextField a = new JTextField(5); // Age text field
JTextField co = new JTextField(10); // Course text field

// Create a text area to display student data
JTextArea d = new JTextArea(5, 30);
d.setEditable(false); // Make the text area non-editable

// Step 3: Set up the layout of the frame
f.setLayout(new BorderLayout(f.getContentPane(),
BoxLayout.Y_AXIS)); // Set vertical layout

// Step 4: Add components to the frame
// Adding labels and input fields for Name, Age, and Course
f.add(new JLabel("Name:"));
f.add(n);
f.add(new JLabel("Age:"));
f.add(a);
f.add(new JLabel("Course:"));
f.add(co);

// Step 5: Add 'Add' button with ActionListener to insert data into the
database
JButton addButton = new JButton("Add");
addButton.addActionListener(e -> {
    try {
        // Retrieve values from input fields
        String name = n.getText();
        String age = a.getText();
        String course = co.getText();

        // Prepare SQL query to insert data into the database
        String query = "INSERT INTO students (name, age, course)
VALUES ('" + name + "', " + age + ", '" + course + "')";

        // Execute the query to insert data

```

```

        c.createStatement().executeUpdate(query);

        // Clear the input fields after adding the student
        n.setText("");
        a.setText("");
        co.setText("");

        // Show success message
        JOptionPane.showMessageDialog(f, "Student added
successfully!");

        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(f, "Error adding student: " +
ex.getMessage());
        }
    });

    // Step 6: Add 'View' button with ActionListener to fetch and display
student data
    JButton viewButton = new JButton("View");
    viewButton.addActionListener(e -> {
        try {
            // Clear the text area before displaying new data
            d.setText("");

            // Execute SQL query to fetch all student records
            ResultSet rs = c.createStatement().executeQuery("SELECT *
FROM students");

            // Iterate through the result set and display data in the text area
            while (rs.next()) {
                int id = rs.getInt(1); // Assuming the first column is the student
ID
                String name = rs.getString(2); // Second column is the name
                int age = rs.getInt(3); // Third column is the age
                String course = rs.getString(4); // Fourth column is the course

```

```

        // Append student data to the text area
        d.append(id + " " + name + " " + age + " " + course + "\n");
    }
} catch (SQLException ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(f, "Error fetching students: " +
ex.getMessage());
}
});

// Step 7: Add the buttons and text area to the frame
f.add(addButton);
f.add(viewButton);
f.add(new JScrollPane(d)); // Wrap text area with JScrollPane for
scrolling

// Step 8: Frame settings
f.setSize(400, 300); // Set the size of the window
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Close the
application when the window is closed
f.setVisible(true); // Make the frame visible
}

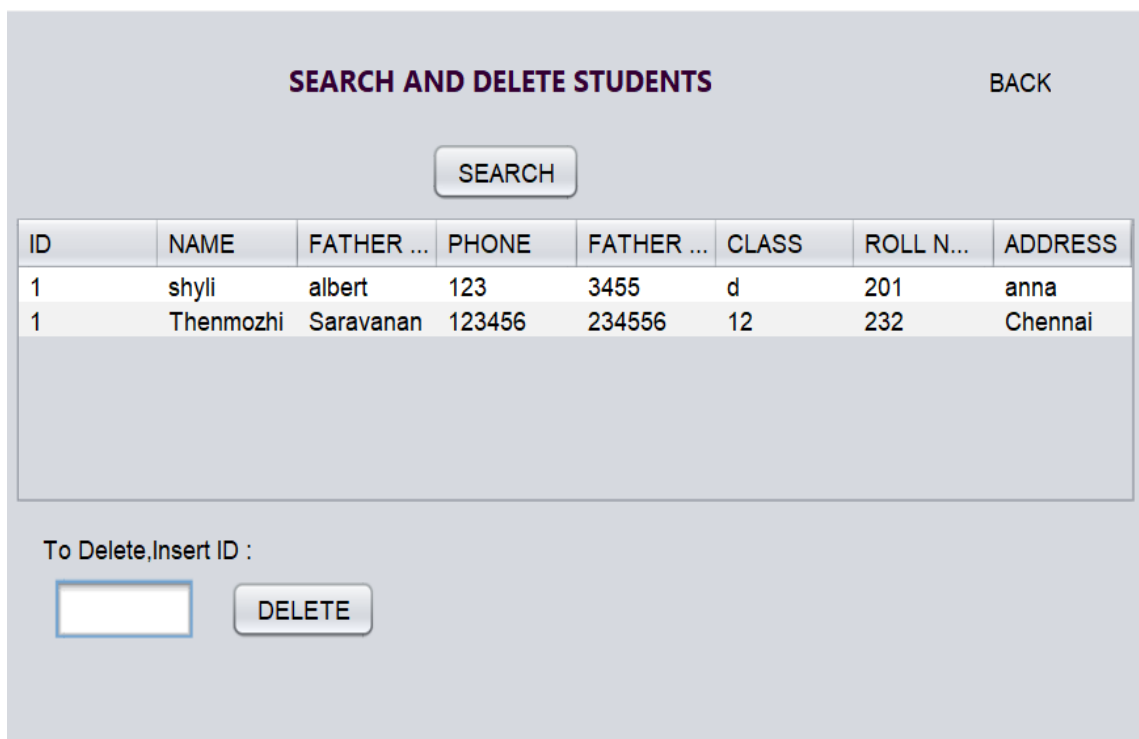
```

## RESULT:



The login page features a dark blue background with a large, stylized orange graduation cap and hands holding it. The title "STUDENT MANAGEMENT" is displayed in red, bold, uppercase letters at the top center. Below the title, there are two white input fields for "Username :" and "Password :". A red "SUBMIT" button is located at the bottom right of the form area.

Login Page



The "SEARCH AND DELETE STUDENTS" page has a light gray background. At the top, the title "SEARCH AND DELETE STUDENTS" is in bold, dark purple, uppercase letters, with a "BACK" link to its right. A "SEARCH" button is centered below the title. Below the button is a table with 8 columns: ID, NAME, FATHER ..., PHONE, FATHER ..., CLASS, ROLL N..., and ADDRESS. The table contains two rows of student data. Below the table is a section for deleting or inserting students, with the label "To Delete,Insert ID :", an input field, and a "DELETE" button.

ID	NAME	FATHER ...	PHONE	FATHER ...	CLASS	ROLL N...	ADDRESS
1	shyli	albert	123	3455	d	201	anna
1	Thenmozhi	Saravanan	123456	234556	12	232	Chennai

Insert Page

## SEARCH AND DELETE STUDENTS

BACK

ID	NAME	FATHER ...	PHONE	FATHER ...	CLASS	ROLL N...	ADDRESS
2	shyli	albert	123345	456	12	205	Chennai

To Delete,Insert ID :

Delete page

The screenshot shows the phpMyAdmin interface for a database named 'studentmanagement'. The left sidebar shows the database structure tree. The main area displays the 'Structure' tab for the 'studentmanagement' database. It lists five tables: addteacher, feesubmit, reportcard, stureg, and user\_login. Each table has a set of actions (Browse, Structure, Search, Insert, Empty, Drop) and a summary row for all tables.

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> addteacher		0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> feesubmit		5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> reportcard		3	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> stureg		1	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> user_login		1	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<b>5 tables</b>	<b>Sum</b>	<b>10</b>	<b>InnoDB</b>	<b>utf8mb4_general_ci</b>	<b>96.0 KiB</b>	<b>0 B</b>

SQL Database

## Registration Page

**STUDENT'S REGISTRATION** <BACK

ID :

NAME:

FATHER'S NAME:

PHONE :

FATHER PHONE :

CLASS :

ROLL NUMBER :

ADDRESS :

SUBMIT

## Registration Page

## RESULT:

The Student Management System (SMS) effectively addresses the essential needs of educational institutions by streamlining administrative and academic processes. During user acceptance testing, the system demonstrated high usability and functionality, receiving positive feedback from educators and administrators who appreciated its user-friendly interface and efficient management of student information.