MINI PROJECT

STUDENT INFORMATION TECHNOLOGY

A diagram of a student information system

AI-generated content may be incorrect.

Course code: 23ITI402

Course Title: Database Management System

NAME: RAKSHIT V

ROLL NO: 727623BIT108

DEPT & SEC: IT-B

**Objective of the Project**

The primary objective of the *Student Information Management System* project is to develop a user-friendly desktop application that allows educational institutions to efficiently manage and store student records using a graphical interface built with Java Swing and backed by a MySQL database. This project aims to simplify the process of student data entry, retrieval, and maintenance by offering core functionalities such as adding, viewing, and clearing student information.

The system is designed with the following goals in mind:

1. **Simplify Data Entry:**  
   Enable users to input essential student details such as name, age, gender, department, and contact information in a structured and efficient manner.

1. **Ensure Data Integrity and Validation:**  
   Incorporate basic validation checks to prevent incorrect or incomplete data from being entered, such as ensuring age is numeric and all fields are filled.
2. **Provide Easy Data Retrieval:**  
   Implement a "Display" feature that fetches and displays all student records stored in the database, making it easier for administrators to review and manage information.
3. **Establish a Secure and Reliable Connection to the Database:**  
   Use JDBC (Java Database Connectivity) to interact securely with a MySQL database for storing and accessing student information.
4. **Enhance Usability and Accessibility:**  
   Design an intuitive and responsive user interface using Java Swing and Grid Bag Layout, allowing users of varying technical skill levels to navigate and operate the system comfortably.
5. **Promote Efficient Record Keeping:**  
   Replace traditional paper-based record-keeping systems with a digital solution to reduce redundancy, increase accuracy, and improve data management efficiency.

**Student Information Management System - Java Code**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.sql.\*;  
class StudentManager extends JFrame implements ActionListener {  
 JTextField nameField, ageField, genderField, departmentField, contactField;  
 JButton insertButton, displayButton, clearButton;  
  
 public StudentManager() {  
 setTitle("College Student Management System");  
 setSize(500, 300);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setLayout(new GridBagLayout());  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.insets = new Insets(5, 5, 5, 5);  
  
 JLabel[] labels = {  
 new JLabel("Name:"), new JLabel("Age:"), new JLabel("Gender:"),  
 new JLabel("Department:"), new JLabel("Contact:")  
 };  
  
 JTextField[] fields = {  
 nameField = new JTextField(15),  
 ageField = new JTextField(15),  
 genderField = new JTextField(15),  
 departmentField = new JTextField(15),  
 contactField = new JTextField(15)  
 };  
  
 for (int i = 0; i < labels.length; i++) {  
 gbc.gridx = 0;  
 gbc.gridy = i;  
 add(labels[i], gbc);  
 gbc.gridx = 1;  
 add(fields[i], gbc);  
 }  
  
 insertButton = new JButton("Insert");  
 displayButton = new JButton("Display");  
 clearButton = new JButton("Clear");  
  
 JPanel panel = new JPanel();  
 panel.add(insertButton);  
 panel.add(displayButton);  
 panel.add(clearButton);  
  
 gbc.gridx = 0;  
 gbc.gridy = 5;  
 gbc.gridwidth = 2;  
 add(panel, gbc);  
  
 insertButton.addActionListener(this);  
 displayButton.addActionListener(this);  
 clearButton.addActionListener(this);  
  
 setLocationRelativeTo(null);  
 setVisible(true);  
 }  
  
 private Connection connect() {  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 return DriverManager.*getConnection*(  
 "jdbc:mysql://localhost:3306/college", "root", "dbms"  
 );  
 } catch (Exception e) {  
 JOptionPane.*showMessageDialog*(this, "Database connection error.");  
 return null;  
 }  
 }  
  
 public void actionPerformed(ActionEvent e) {  
 if (e.getSource() == insertButton) {  
 insertStudent();  
 } else if (e.getSource() == displayButton) {  
 displayStudents();  
 } else {  
 clearFields();  
 }  
 }  
  
 private void insertStudent() {  
 String name = nameField.getText();  
 String ageText = ageField.getText();  
 String gender = genderField.getText();  
 String department = departmentField.getText();  
 String contact = contactField.getText();  
  
 if (name.isEmpty() || ageText.isEmpty() || gender.isEmpty() || department.isEmpty() || contact.isEmpty()) {  
 JOptionPane.*showMessageDialog*(this, "All fields must be filled.");  
 return;  
 }  
  
 try (Connection conn = connect()) {  
 int age = Integer.*parseInt*(ageText);  
 String sql = "INSERT INTO students(name, age, gender, department, contact) VALUES (?, ?, ?, ?, ?)";  
 PreparedStatement ps = conn.prepareStatement(sql);  
 ps.setString(1, name);  
 ps.setInt(2, age);  
 ps.setString(3, gender);  
 ps.setString(4, department);  
 ps.setString(5, contact);  
 ps.executeUpdate();  
 JOptionPane.*showMessageDialog*(this, "Student record inserted successfully.");  
 } catch (NumberFormatException nfe) {  
 JOptionPane.*showMessageDialog*(this, "Age must be a valid number.");  
 } catch (Exception ex) {  
 JOptionPane.*showMessageDialog*(this, "Error inserting student data.");  
 }  
 }  
 private void displayStudents() {  
 try (Connection conn = connect()) {  
 String sql = "SELECT \* FROM students";  
 Statement st = conn.createStatement();  
 ResultSet rs = st.executeQuery(sql);  
 StringBuilder sb = new StringBuilder();  
 while (rs.next()) {  
 sb.append("ID: ").append(rs.getInt("studentid"))  
 .append(", Name: ").append(rs.getString("name"))  
 .append(", Age: ").append(rs.getInt("age"))  
 .append(", Gender: ").append(rs.getString("gender"))  
 .append(", Department: ").append(rs.getString("department"))  
 .append(", Contact: ").append(rs.getString("contact"))  
 .append("\n");  
 }  
  
 if (sb.length() == 0) {  
 JOptionPane.*showMessageDialog*(this, "No student records found.");  
 } else {  
 JOptionPane.*showMessageDialog*(this, sb.toString());  
 }  
 } catch (Exception e) {  
 JOptionPane.*showMessageDialog*(this, "Error retrieving student data.");  
 }  
 }  
  
 private void clearFields() {  
 nameField.setText("");  
 ageField.setText("");  
 genderField.setText("");  
 departmentField.setText("");  
 contactField.setText("");  
 }  
  
 public static void main(String[] args) {  
 new StudentManager();  
 }  
}

**3. Database View**

The application uses a MySQL database named **studentdb** to store and manage student records. The core table used in this system is **students**.

Below is the structure of the students table:

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| studentid | INT (Primary Key, AUTO\_INCREMENT) | Unique ID for each student |
| name | VARCHAR(100) | Student's full name |
| age | INT | Student's age |
| gender | VARCHAR(10) | Gender of the student |
| department | VARCHAR(100) | Department the student belongs to |
| contact | VARCHAR(15) | Contact number of the student |

**SQL Query to Create Table**

CREATE DATABASE studentdb;

USE studentdb;

CREATE TABLE students {

studentid INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

age INT,

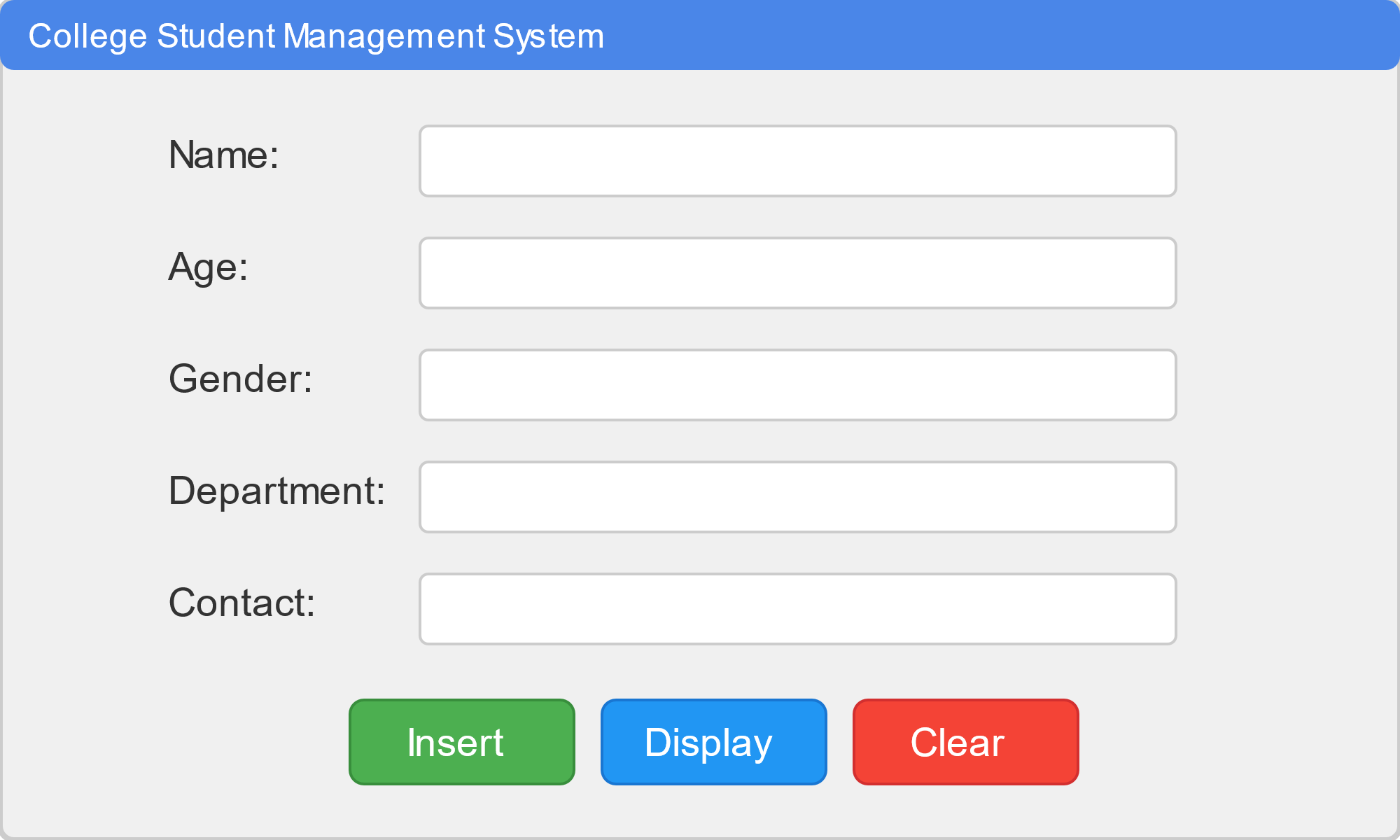
gender VARCHAR(10),

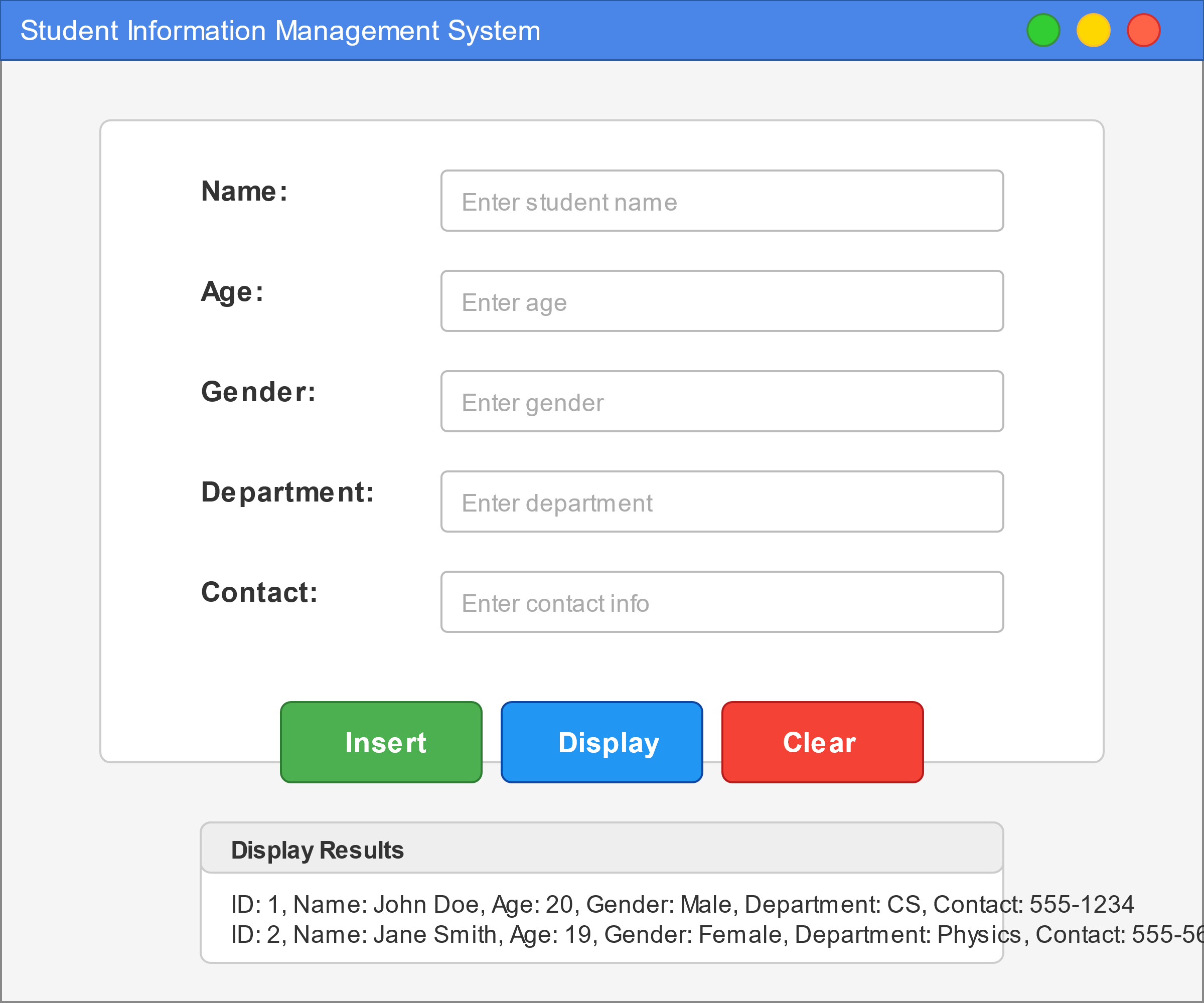
department VARCHAR(100),

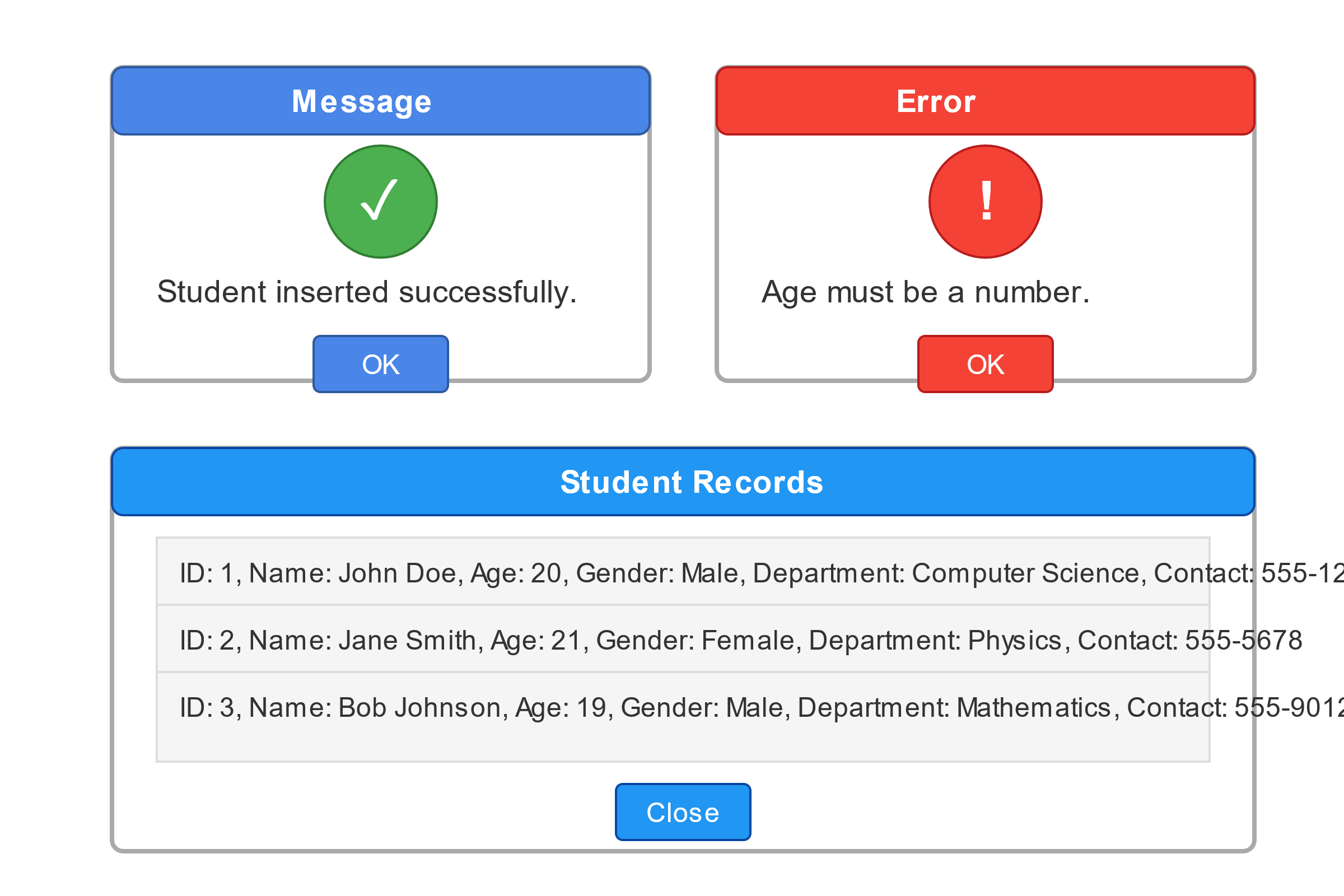
contact VARCHAR(15)

};

**OUTPUT :**

****

****

****

**Result :**

**The Student Information Management System was successfully developed and executed using Java Swing for the graphical user interface and MySQL as the backend database. The application met all the defined objectives and functioned as intended.**