

# STUDENT MANAGEMENT SYSTEM

## USING JAVA SWING & JDBC MYSQL

Student Name	Student ID	Student Grade	Date of Birth	Gender	Contact Name	Email
himanshu	1	80	18-11-1993	Male	8948324461	himanshusingh181...
vineet	2	80	01-01-1990	Male	9125252525	vineetsingh123@g...
ARVIND	3	89	01-02-1997	Male	9450215478	arvind123@gmail.c...

ide used: eclipse/ notepad

package name: `package sms_demo1;`

class name: `StudentManagementSystem demo`

External JAR used :mysql-connector-j-8.3.0\mysql-connector-j-8.3.0.jar

```
//mysqlatabase_name:sv_java_24_t_student
```

```
//database_table_name:students
```

```
//db_table_fields:(student_name, student_id, student_grade, dob,  
gender, contact, email)
```

**SOURCE CODE OF StudentManagementSystem\_demo.JAVA**

```
package sms_demo1;

//mysqlDATABASE_NAME:sv_java_24_t_student
//DATABASE_TABLE_NAME:students
//db_table_fields:(student_name, student_id, student_grade, dob, gender, contact, email)

//excel export import
//project ki api banana

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.table.DefaultTableModel;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.ParseException;
import java.text.SimpleDateFormat;

public class StudentManagementSystem_demo extends JFrame implements ActionListener {
    JLabel jttitle;
    JLabel studentName, studentID, studentGrade, dobLabel, genderLabel, contactLabel, emailLabel;
    JTextField jstudentName, jstudentID, jstudentGrade, dobField, contactField, emailField, searchField;
    JRadioButton maleRadio, femaleRadio;
    ButtonGroup genderGroup;
    JButton addStudent, reset, deleteRecord, searchButton;
    JTable studentTable;
    DefaultTableModel tableModel;

    private static final String DB_URL =
"jdbc:mysql://localhost:3306/sv_java_24_t_student";
```

```
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "root";
private Connection connection;

public StudentManagementSystem_demo() {
    setTitle("Student Management System @svinfotech by Java Training batch
2k24");
    setLayout(null);
    setSize(1000, 600);
    setVisible(true);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    jtitle = new JLabel("STUDENT MANAGEMENT SYSTEM");
    jtitle.setBounds(130, 6, 720, 70);
    jtitle.setFont(new Font("Comic Sans MS", Font.PLAIN, 30));
    jtitle.setBorder(new LineBorder(new Color(0, 120, 215), 5));
    jtitle.setHorizontalAlignment(SwingConstants.CENTER);

    studentName = new JLabel("Student Name");
    studentName.setBounds(50, 80, 110, 30);
    studentName.setBorder(new LineBorder(new Color(0, 120, 215), 3));

    studentID = new JLabel("Student ID");
    studentID.setBounds(50, 120, 110, 30);
    studentID.setBorder(new LineBorder(new Color(0, 120, 215), 3));

    studentGrade = new JLabel("Student Grade");
    studentGrade.setBounds(50, 160, 110, 30);
    studentGrade.setBorder(new LineBorder(new Color(0, 120, 215), 3));

    dobLabel = new JLabel("Date of Birth");
    dobLabel.setBounds(50, 200, 110, 30);
    dobLabel.setBorder(new LineBorder(new Color(0, 120, 215), 3));

    genderLabel = new JLabel("Gender");
    genderLabel.setBounds(50, 240, 110, 30);
    genderLabel.setBorder(new LineBorder(new Color(0, 120, 215), 3));

    contactLabel = new JLabel("Contact Number");
    contactLabel.setBounds(50, 280, 110, 30);
    contactLabel.setBorder(new LineBorder(new Color(0, 120, 215), 3));

    emailLabel = new JLabel("Email");
    emailLabel.setBounds(50, 320, 110, 30);
    emailLabel.setBorder(new LineBorder(new Color(0, 120, 215), 3));

    jstudentName = new JTextField();
    jstudentName.setBounds(200, 80, 200, 30);
```

```
jstudentID = new JTextField();
jstudentID.setBounds(200, 120, 200, 30);

jstudentGrade = new JTextField();
jstudentGrade.setBounds(200, 160, 200, 30);

dobField = new JTextField();
dobField.setBounds(200, 200, 200, 30);

maleRadio = new JRadioButton("Male");
maleRadio.setBounds(200, 240, 80, 30);

femaleRadio = new JRadioButton("Female");
femaleRadio.setBounds(290, 240, 100, 30);

genderGroup = new ButtonGroup();
genderGroup.add(maleRadio);
genderGroup.add(femaleRadio);

contactField = new JTextField();
contactField.setBounds(200, 280, 200, 30);

emailField = new JTextField();
emailField.setBounds(200, 320, 200, 30);

addStudent = new JButton("Add Student");
addStudent.setBounds(650, 150, 150, 30);

reset = new JButton("Reset");
reset.setBounds(650, 200, 150, 30);

deleteRecord = new JButton("Delete Record");
deleteRecord.setBounds(650, 250, 150, 30);

searchField = new JTextField();
searchField.setBounds(50, 360, 300, 30);

searchButton = new JButton("Search by ID");
searchButton.setBounds(360, 360, 150, 30);

add(jtitle);
add(studentName);
add(studentID);
add(studentGrade);
add(dobLabel);
add(genderLabel);
add(contactLabel);
add(emailLabel);
add(jstudentName);
add(jstudentID);
add(jstudentGrade);
```

```
        add(dobField);
        add(maleRadio);
        add(femaleRadio);
        add(contactField);
        add(emailField);
        add(addStudent);
        add(reset);
        add(deleteRecord);
        add(searchField);
        add(searchButton);

        String[] columnNames = {"Student Name", "Student ID", "Student Grade", "Date
of Birth", "Gender", "Contact Name", "Email"};
        tableModel = new DefaultTableModel(columnNames, 0);

        studentTable = new JTable(tableModel);
        JScrollPane scrollPane = new JScrollPane(studentTable);
        scrollPane.setBounds(50, 400, 860, 150);
        add(scrollPane);

        addStudent.addActionListener(this);
        reset.addActionListener(this);
        deleteRecord.addActionListener(this);
        searchButton.addActionListener(this);

        connectToDatabase();
        loadStudentDataFromDatabase();
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == addStudent) {
            String name = jstudentName.getText();
            String id = jstudentID.getText();
            String grade = jstudentGrade.getText();
            String dob = dobField.getText();
            String contact = contactField.getText();
            String email = emailField.getText();
            String gender = maleRadio.isSelected() ? "Male" : "Female";

            if (name.isEmpty() || grade.isEmpty() || dob.isEmpty() ||
contact.isEmpty() || email.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Please fill in all fields.",
"Error", JOptionPane.ERROR_MESSAGE);
            } else if (!isValidEmail(email)) {
                JOptionPane.showMessageDialog(this, "Invalid email address.",
"Error", JOptionPane.ERROR_MESSAGE);
            } else if (!isValidDate(dob)) {
                JOptionPane.showMessageDialog(this, "Invalid date of birth. Use the
format 'dd-MM-yyyy'.", "Error", JOptionPane.ERROR_MESSAGE);
            } else if (!isValidGrade(grade)) {
                JOptionPane.showMessageDialog(this, "Invalid student grade. It should
be a number.", "Error", JOptionPane.ERROR_MESSAGE);
            } else if (!isNumeric(id)) {
                JOptionPane.showMessageDialog(this, "Invalid student ID. It should be
a number.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}
```

```

        } else if (!isValidContactNumber(contact)) {
            JOptionPane.showMessageDialog(this, "Invalid contact number. It
should be numeric.", "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            String[] data = {name, id, grade, dob, gender, contact, email};
            tableModel.addRow(data);

            jstudentName.setText("");
            jstudentID.setText("");
            jstudentGrade.setText("");
            dobField.setText("");
            genderGroup.clearSelection();
            contactField.setText("");
            emailField.setText("");
            insertStudentData(name, id, grade, dob, gender, contact, email);
        }
    }

    if (e.getSource() == reset) {
        jstudentName.setText("");
        jstudentID.setText("");
        jstudentGrade.setText("");
        dobField.setText("");
        genderGroup.clearSelection();
        contactField.setText("");
        emailField.setText("");
    }

    if (e.getSource() == deleteRecord) {
        String studentIDToDelete = null;
        int selectedRow = studentTable.getSelectedRow();
        if (selectedRow >= 0) {
            studentIDToDelete = tableModel.getValueAt(selectedRow, 1).toString();
            tableModel.removeRow(selectedRow);

            deleteStudentData(studentIDToDelete);
        }
    }

    if (e.getSource() == searchButton) {
        String searchId = searchField.getText();
        for (int row = 0; row < tableModel.getRowCount(); row++) {
            if (tableModel.getValueAt(row, 1).equals(searchId)) {
                studentTable.setRowSelectionInterval(row, row);
                studentTable.setSelectionBackground(Color.YELLOW);
                studentTable.setSelectionForeground(Color.BLACK);
                break;
            }
        }
    }
}

private boolean isValidEmail(String email) {
    return email.matches("^[A-Za-z0-9+_.-]+@(.+)$");
}

```

```
private boolean isValidDate(String date) {
    try {
        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
        sdf.setLenient(false);
        sdf.parse(date);
        return true;
    } catch (ParseException e) {
        return false;
    }
}

private boolean isValidGrade(String grade) {
    try {
        Double.parseDouble(grade);
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}

private boolean isValidStudentID(String id) {
    return id.matches("[A-Za-z0-9]+$");
}

private boolean isValidContactNumber(String contact) {
    return contact.matches("[0-9]+$");
}

private boolean isNumeric(String str) {
    try {
        Integer.parseInt(str);
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}

private void connectToDatabase() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        connection = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
    } catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace();
    }
}

private void insertStudentData(String name, String id, String grade, String dob,
String gender, String contact, String email) {
    String insertQuery = "INSERT INTO students (student_name, student_id,
student_grade, dob, gender, contact, email) VALUES (?, ?, ?, STR_TO_DATE(?, '%d-%m-%Y'), ?, ?, ?)";
```



```
        try {
            PreparedStatement preparedStatement =
connection.prepareStatement(insertQuery);
            preparedStatement.setString(1, name);
            preparedStatement.setString(2, id);
            preparedStatement.setString(3, grade);
            preparedStatement.setString(4, dob);
            preparedStatement.setString(5, gender);
            preparedStatement.setString(6, contact);
            preparedStatement.setString(7, email);

            int rowsAffected = preparedStatement.executeUpdate();

            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(this, "Student data inserted
successfully", "Success", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "Failed to insert student data",
"Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void loadStudentDataFromDatabase() {
        try {
            String selectQuery = "SELECT student_name, student_id, student_grade,
DATE_FORMAT(dob, '%d-%m-%Y'), gender, contact, email FROM students";
            PreparedStatement preparedStatement =
connection.prepareStatement(selectQuery);
            ResultSet resultSet = preparedStatement.executeQuery();

            while (resultSet.next()) {
                String name = resultSet.getString(1);
                String id = resultSet.getString(2);
                String grade = resultSet.getString(3);
                String dob = resultSet.getString(4);
                String gender = resultSet.getString(5);
                String contact = resultSet.getString(6);
                String email = resultSet.getString(7);

                String[] data = {name, id, grade, dob, gender, contact, email};
                tableModel.addRow(data);
            }

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

    private void deleteStudentData(String studentID) {
        String deleteQuery = "DELETE FROM students WHERE student_id = ?";

        try {
```

```
        PreparedStatement preparedStatement =
connection.prepareStatement(deleteQuery);
        preparedStatement.setString(1, studentID);

        int rowsAffected = preparedStatement.executeUpdate();

        if (rowsAffected > 0) {
            JOptionPane.showMessageDialog(this, "Student data deleted
successfully", "Success", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "Failed to delete student data",
"Error", JOptionPane.ERROR_MESSAGE);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void closeDatabaseConnection() {
    try {
        if (connection != null) {
            connection.close();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        new StudentManagementSystem_demo();
    });
}
}
```

```
package sms_demo1;
```

```
//mysqlDATABASE_NAME:sv_java_24_t_student
```

```
//DATABASE_TABLE_NAME:students
```

```
//db_table_fields:(student_name, student_id, student_grade, dob,  
gender, contact, email)
```

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
import javax.swing.*;
```

```
import javax.swing.border.LineBorder;
```

```
import javax.swing.table.DefaultTableModel;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
```

```
import java.text.ParseException;
```

```
import java.text.SimpleDateFormat;
```

```
public class StudentManagementSystem_demo extends JFrame
implements ActionListener {

    JLabel jtitle;

    JLabel studentName, studentID, studentGrade, dobLabel,
genderLabel, contactLabel, emailLabel;

    JTextField jstudentName, jstudentID, jstudentGrade, dobField,
contactField, emailField, searchField;

    JRadioButton maleRadio, femaleRadio;

    ButtonGroup genderGroup;

    JButton addStudent, reset, deleteRecord, searchButton;

    JTable studentTable;

    DefaultTableModel tableModel;

    private static final String DB_URL =
"jdbc:mysql://localhost:3306/sv_java_24_t_student";

    private static final String DB_USER = "root";

    private static final String DB_PASSWORD = "root";

    private Connection connection;
```

```
public StudentManagementSystem_demo() {  
  
    setTitle("Student Management System @svinfotech by Java  
Training batch 2k24");  
  
    setLayout(null);  
  
    setSize(1000, 600);  
  
    setVisible(true);  
  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
  
    jtitle = new JLabel("STUDENT MANAGEMENT SYSTEM");  
  
    jtitle.setBounds(130, 6, 720, 70);  
  
    jtitle.setFont(new Font("Comic Sans MS", Font.PLAIN, 30));  
  
    jtitle.setBorder(new LineBorder(new Color(0, 120, 215), 5));  
  
    jtitle.setHorizontalAlignment(SwingConstants.CENTER);  
  
  
    studentName = new JLabel("Student Name");  
  
    studentName.setBounds(50, 80, 110, 30);  
  
    studentName.setBorder(new LineBorder(new Color(0, 120, 215),  
3));  
  
  

```

```
studentID = new JLabel("Student ID");
```

```
studentID.setBounds(50, 120, 110, 30);
```

```
studentID.setBorder(new LineBorder(new Color(0, 120, 215), 3));
```

```
studentGrade = new JLabel("Student Grade");
```

```
studentGrade.setBounds(50, 160, 110, 30);
```

```
studentGrade.setBorder(new LineBorder(new Color(0, 120, 215), 3));
```

```
dobLabel = new JLabel("Date of Birth");
```

```
dobLabel.setBounds(50, 200, 110, 30);
```

```
dobLabel.setBorder(new LineBorder(new Color(0, 120, 215), 3));
```

```
genderLabel = new JLabel("Gender");
```

```
genderLabel.setBounds(50, 240, 110, 30);
```

```
genderLabel.setBorder(new LineBorder(new Color(0, 120, 215),  
3));
```

```
contactLabel = new JLabel("Contact Number");
```

```
contactLabel.setBounds(50, 280, 110, 30);
```

```
contactLabel.setBorder(new LineBorder(new Color(0, 120, 215),  
3));
```

```
emailLabel = new JLabel("Email");
```

```
emailLabel.setBounds(50, 320, 110, 30);
```

```
emailLabel.setBorder(new LineBorder(new Color(0, 120, 215),  
3));
```

```
jstudentName = new JTextField();
```

```
jstudentName.setBounds(200, 80, 200, 30);
```

```
jstudentID = new JTextField();
```

```
jstudentID.setBounds(200, 120, 200, 30);
```

```
jstudentGrade = new JTextField();
```

```
jstudentGrade.setBounds(200, 160, 200, 30);
```

```
dobField = new JTextField();
```

```
dobField.setBounds(200, 200, 200, 30);
```

```
maleRadio = new JRadioButton("Male");
```

```
maleRadio.setBounds(200, 240, 80, 30);
```

```
femaleRadio = new JRadioButton("Female");
```

```
femaleRadio.setBounds(290, 240, 100, 30);
```

```
genderGroup = new ButtonGroup();
```

```
genderGroup.add(maleRadio);
```

```
genderGroup.add(femaleRadio);
```



```
contactField = new JTextField();
```

```
contactField.setBounds(200, 280, 200, 30);
```

```
emailField = new JTextField();
```

```
emailField.setBounds(200, 320, 200, 30);
```

```
addStudent = new JButton("Add Student");
```

```
addStudent.setBounds(650, 150, 150, 30);
```

```
reset = new JButton("Reset");
```

```
reset.setBounds(650, 200, 150, 30);
```

```
deleteRecord = new JButton("Delete Record");
```

```
deleteRecord.setBounds(650, 250, 150, 30);
```

```
searchField = new JTextField();
```

```
searchField.setBounds(50, 360, 300, 30);
```

```
searchButton = new JButton("Search by ID");
```

```
searchButton.setBounds(360, 360, 150, 30);
```

```
add(jtitle);
```

```
add(studentName);
```

```
add(studentID);
```

```
add(studentGrade);
```

```
add(dobLabel);
```

```
add(genderLabel);
```

```
add(contactLabel);
```

```
add(emailLabel);
```

```
add(jstudentName);
```

```
add(jstudentID);
```

```
add(jstudentGrade);
```

```
add(dobField);
```

```
add(maleRadio);
```

```
add(femaleRadio);
```

```
add(contactField);
```

```
add(emailField);
```

```
add(addStudent);
```

```
add(reset);
```

```
add(deleteRecord);
```

```
add(searchField);
```

```
add(searchButton);
```

```
String[] columnNames = {"Student Name", "Student ID", "Student  
Grade", "Date of Birth", "Gender", "Contact Name", "Email"};
```

```
tableModel = new DefaultTableModel(columnNames, 0);
```

```
studentTable = new JTable(tableModel);
```

```
JScrollPane scrollPane = new JScrollPane(studentTable);
```

```
scrollPane.setBounds(50, 400, 860, 150);
```

```
add(scrollPane);
```

```
addStudent.addActionListener(this);
```

```
reset.addActionListener(this);
```

```
deleteRecord.addActionListener(this);
```

```
searchButton.addActionListener(this);
```

```
connectToDatabase();
```

```
loadStudentDataFromDatabase();
```

```
}
```

```
public void actionPerformed(ActionEvent e) {  
    if (e.getSource() == addStudent) {  
        String name = jstudentName.getText();  
        String id = jstudentID.getText();  
        String grade = jstudentGrade.getText();  
        String dob = dobField.getText();  
        String contact = contactField.getText();  
        String email = emailField.getText();  
        String gender = maleRadio.isSelected() ? "Male" : "Female";  
  
        if (name.isEmpty() || grade.isEmpty() || dob.isEmpty() ||  
            contact.isEmpty() || email.isEmpty()) {  
            JOptionPane.showMessageDialog(this, "Please fill in all  
fields.", "Error", JOptionPane.ERROR_MESSAGE);  
        } else if (!isValidEmail(email)) {  
            JOptionPane.showMessageDialog(this, "Invalid email  
address.", "Error", JOptionPane.ERROR_MESSAGE);  
        } else if (!isValidDate(dob)) {
```

```
JOptionPane.showMessageDialog(this, "Invalid date of birth.  
Use the format 'dd-MM-yyyy'." , "Error",  
JOptionPane.ERROR_MESSAGE);  
  
    } else if (!isValidGrade(grade)) {  
  
        JOptionPane.showMessageDialog(this, "Invalid student  
grade. It should be a number." , "Error",  
JOptionPane.ERROR_MESSAGE);  
  
    } else if (!isNumeric(id)) {  
  
        JOptionPane.showMessageDialog(this, "Invalid student ID. It  
should be a number." , "Error", JOptionPane.ERROR_MESSAGE);  
  
    } else if (!isValidContactNumber(contact)) {  
  
        JOptionPane.showMessageDialog(this, "Invalid contact  
number. It should be numeric." , "Error",  
JOptionPane.ERROR_MESSAGE);  
  
    } else {  
  
        String[] data = {name, id, grade, dob, gender, contact, email};  
  
        tableModel.addRow(data);  
  
        jstudentName.setText("");  
  
        jstudentID.setText("");  
  
        jstudentGrade.setText("");  
  
        dobField.setText("");
```

```
genderGroup.clearSelection();

contactField.setText("");

emailField.setText("");

insertStudentData(name, id, grade, dob, gender, contact,
email);
}
}
```

```
if (e.getSource() == reset) {

jstudentName.setText("");

jstudentID.setText("");

jstudentGrade.setText("");

dobField.setText("");

genderGroup.clearSelection();

contactField.setText("");

emailField.setText("");
}
```

```
if (e.getSource() == deleteRecord) {

String studentIDToDelete = null;
```

```
int selectedRow = studentTable.getSelectedRow();
```

```
if (selectedRow >= 0) {
```

```
    studentIDToDelete = tableModel.getValueAt(selectedRow,  
1).toString();
```

```
    tableModel.removeRow(selectedRow);
```

```
    deleteStudentData(studentIDToDelete);
```

```
}
```

```
if (e.getSource() == searchButton) {
```

```
    String searchId = searchField.getText();
```

```
    for (int row = 0; row < tableModel.getRowCount(); row++) {
```

```
        if (tableModel.getValueAt(row, 1).equals(searchId)) {
```

```
            studentTable.setRowSelectionInterval(row, row);
```

```
            studentTable.setSelectionBackground(Color.YELLOW);
```

```
            studentTable.setSelectionForeground(Color.BLACK);
```

```
            break;
```

```
        }
```

```
    }
```

```
}
```

```
}
```

```
private boolean isValidEmail(String email) {
```

```
    return email.matches("^[A-Za-z0-9+_.-]+@(.+)$");
```

```
}
```

```
private boolean isValidDate(String date) {
```

```
    try {
```

```
        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
```

```
        sdf.setLenient(false);
```

```
        sdf.parse(date);
```

```
        return true;
```

```
    } catch (ParseException e) {
```

```
        return false;
```

```
    }
```

```
}
```

```
private boolean isValidGrade(String grade) {
```



```
try {
```

```
    Double.parseDouble(grade);
```

```
    return true;
```

```
} catch (NumberFormatException e) {
```

```
    return false;
```

```
}
```

```
private boolean isValidStudentID(String id) {
```

```
    return id.matches("^[A-Za-z0-9]+$");
```

```
}
```

```
private boolean isValidContactNumber(String contact) {
```

```
    return contact.matches("^[0-9]+$");
```

```
}
```

```
private boolean isNumeric(String str) {
```

```
    try {
```

```
        Integer.parseInt(str);
```

```
    return true;
```

```
    } catch (NumberFormatException e) {
```

```
        return false;
```

```
    }
```

```
}
```

```
private void connectToDatabase() {
```

```
    try {
```

```
        Class.forName("com.mysql.cj.jdbc.Driver");
```

```
        connection = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
```

```
    } catch (ClassNotFoundException | SQLException e) {
```

```
        e.printStackTrace();
```

```
    }
```

```
}
```

```
private void insertStudentData(String name, String id, String grade, String dob, String gender, String contact, String email) {
```

```
String insertQuery = "INSERT INTO students (student_name,  
student_id, student_grade, dob, gender, contact, email) VALUES (?, ?,  
?, STR_TO_DATE(?, '%d-%m-%Y'), ?, ?, ?)";
```

```
try {
```

```
    PreparedStatement preparedStatement =  
connection.prepareStatement(insertQuery);
```

```
    preparedStatement.setString(1, name);
```

```
    preparedStatement.setString(2, id);
```

```
    preparedStatement.setString(3, grade);
```

```
    preparedStatement.setString(4, dob);
```

```
    preparedStatement.setString(5, gender);
```

```
    preparedStatement.setString(6, contact);
```

```
    preparedStatement.setString(7, email);
```

```
    int rowsAffected = preparedStatement.executeUpdate();
```

```
    if (rowsAffected > 0) {
```

```
        JOptionPane.showMessageDialog(this, "Student data  
inserted successfully", "Success",  
JOptionPane.INFORMATION_MESSAGE);
```

```
        } else {  
            JOptionPane.showMessageDialog(this, "Failed to insert  
student data", "Error", JOptionPane.ERROR_MESSAGE);  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}  
}  
  
private void loadStudentDataFromDatabase() {  
    try {  
        String selectQuery = "SELECT student_name, student_id,  
student_grade, DATE_FORMAT(dob, '%d-%m-%Y'), gender, contact,  
email FROM students";  
        PreparedStatement preparedStatement =  
connection.prepareStatement(selectQuery);  
        ResultSet resultSet = preparedStatement.executeQuery();  
  
        while (resultSet.next()) {  
            String name = resultSet.getString(1);  
            String id = resultSet.getString(2);
```

```
String grade = resultSet.getString(3);
```

```
String dob = resultSet.getString(4);
```

```
String gender = resultSet.getString(5);
```

```
String contact = resultSet.getString(6);
```

```
String email = resultSet.getString(7);
```

```
String[] data = {name, id, grade, dob, gender, contact, email};
```

```
tableModel.addRow(data);
```

```
}
```

```
} catch (SQLException e) {
```

```
    e.printStackTrace();
```

```
}
```

```
}
```

```
private void deleteStudentData(String studentID) {
```

```
    String deleteQuery = "DELETE FROM students WHERE student_id  
= ?";
```

```
    try {
```

```
        PreparedStatement preparedStatement =  
connection.prepareStatement(deleteQuery);  
  
        preparedStatement.setString(1, studentID);  
  
        int rowsAffected = preparedStatement.executeUpdate();  
  
        if (rowsAffected > 0) {  
            JOptionPane.showMessageDialog(this, "Student data deleted  
successfully", "Success", JOptionPane.INFORMATION_MESSAGE);  
        } else {  
            JOptionPane.showMessageDialog(this, "Failed to delete  
student data", "Error", JOptionPane.ERROR_MESSAGE);  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}  
}  
  
private void closeDatabaseConnection() {  
    try {
```

```
        if (connection != null) {  
            connection.close();  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}  
  
public static void main(String[] args) {  
    SwingUtilities.invokeLater(() -> {  
        new StudentManagementSystem_demo();  
    });  
}
```