1. **File Login.java**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.sql.\*;  
  
public class Login extends JFrame implements ActionListener {  
 JTextField textFieldName;  
 JPasswordField passwordField;  
 JButton login, back;  
  
 Login() {  
 setSize(1540, 850);  
 setLayout(null);  
 setLocationRelativeTo(null); // Center frame  
  
 // Background image  
 ImageIcon bgIcon = new ImageIcon(ClassLoader.*getSystemResource*("icon/bg-login.jpg"));  
 if (bgIcon.getImage() == null) {  
 System.out.println("Không tìm thấy hình nền tại icon/bg-login.jpg");  
 }  
 Image bgImage = bgIcon.getImage().getScaledInstance(1540, 850, Image.SCALE\_DEFAULT);  
 JLabel background = new JLabel(new ImageIcon(bgImage));  
 background.setBounds(0, 0, 1540, 850);  
 add(background);  
  
 // Login panel with gradient  
 JPanel loginPanel = new JPanel() {  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 Graphics2D g2d = (Graphics2D) g;  
 g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);  
 GradientPaint gp = new GradientPaint(  
 0, 0, Color.*decode*("#f0f3fd"),  
 0, getHeight(), Color.*decode*("#b2c2f0")  
 );  
 g2d.setPaint(gp);  
 g2d.fillRect(0, 0, getWidth(), getHeight());  
 }  
 };  
 loginPanel.setBounds(460, 225, 620, 520);  
 loginPanel.setLayout(null);  
 background.add(loginPanel);  
  
 // Logo  
 ImageIcon i1 = new ImageIcon(ClassLoader.*getSystemResource*("icon/logo1.png"));  
 if (i1.getImage() == null) {  
 System.out.println("Không tìm thấy logo tại icon/logo1.png");  
 }  
 Image i2 = i1.getImage().getScaledInstance(366, 82, Image.SCALE\_DEFAULT);  
 ImageIcon i3 = new ImageIcon(i2);  
 JLabel img = new JLabel(i3);  
 img.setBounds(602, 77, 366, 82);  
 background.add(img);  
  
 // Username label  
 JLabel username = new JLabel("Tên đăng nhập:");  
 username.setBounds(59, 41, 230, 30);  
 username.setFont(new Font("Serif", Font.BOLD, 30));  
 loginPanel.add(username);  
  
 // Username field  
 textFieldName = new JTextField();  
 textFieldName.setBounds(59, 99, 500, 60);  
 textFieldName.setFont(new Font("SansSerif", Font.PLAIN, 30));  
 loginPanel.add(textFieldName);  
  
 // Password label  
 JLabel password = new JLabel("Mật khẩu");  
 password.setBounds(59, 214, 144, 30);  
 password.setFont(new Font("Serif", Font.BOLD, 30));  
 loginPanel.add(password);  
  
 // Password field  
 passwordField = new JPasswordField();  
 passwordField.setBounds(59, 260, 500, 60);  
 passwordField.setFont(new Font("SansSerif", Font.PLAIN, 30));  
 loginPanel.add(passwordField);  
  
 // Login button  
 login = new JButton("Đăng nhập");  
 login.setBounds(59, 393, 224, 60);  
 login.setBackground(Color.BLACK);  
 login.setForeground(Color.WHITE);  
 login.setFont(new Font("Arial", Font.BOLD, 25));  
 login.addActionListener(this);  
 loginPanel.add(login);  
  
 // Back button  
 back = new JButton("Hủy");  
 back.setBounds(335, 394, 224, 60);  
 back.setBackground(Color.BLACK);  
 back.setForeground(Color.WHITE);  
 back.setFont(new Font("Arial", Font.BOLD, 25));  
 back.addActionListener(this);  
 loginPanel.add(back);  
  
 setVisible(true);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (e.getSource() == login) {  
 String username = textFieldName.getText();  
 String password = new String(passwordField.getPassword());  
  
 String query = "SELECT \* FROM login WHERE tendangnhap = ? AND matkhau = ?";  
 try {  
 Connect db = new Connect();  
 Connection conn = db.getConnection();  
  
 PreparedStatement pstmt = conn.prepareStatement(query);  
 pstmt.setString(1, username);  
 pstmt.setString(2, password);  
  
 ResultSet resultSet = pstmt.executeQuery();  
 if (resultSet.next()) {  
 setVisible(false);  
 new Home(); // assuming you have Home.java  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Invalid Username or Password");  
 }  
  
 // Đóng tài nguyên  
 resultSet.close();  
 pstmt.close();  
 conn.close();  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 }  
 } else if (e.getSource() == back) {  
 setVisible(false);  
 }  
 }  
  
 public static void main(String[] args) {  
 new Login();  
 }  
}

1. **Home.java**

import javax.swing.\*;

import javax.swing.border.EmptyBorder;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

public class Home extends JFrame implements ActionListener {

private JLabel studentCountLabel;

private JLabel teacherCountLabel;

private JLabel classCountLabel;

Home() {

// Black menu panel

JPanel menuPanel = new JPanel();

menuPanel.setBounds(0, 0, 254, 850);

menuPanel.setBackground(Color.BLACK);

menuPanel.setLayout(null);

add(menuPanel);

// Logo

ImageIcon logoIcon = new ImageIcon(ClassLoader.getSystemResource("icon/logo.png"));

Image logoImage = logoIcon.getImage().getScaledInstance(190, 140, Image.SCALE\_DEFAULT);

ImageIcon scaledLogoIcon = new ImageIcon(logoImage);

JLabel logoLabel = new JLabel(scaledLogoIcon);

logoLabel.setBounds(32, 10, 190, 140);

menuPanel.add(logoLabel);

// Red panel

JPanel redPanel = new JPanel();

redPanel.setBounds(254, 0, 15, 850);

redPanel.setBackground(new Color(209, 105, 0));

add(redPanel);

// Menu items with labels

String[] menuTitles = {

"Tin tức", "Thêm thông tin", "Quản lý thông tin", "Lịch trình",

"Đăng ký học", "Thanh toán", "Nhật ký", "Cài đặt", "Đăng xuất"

};

String[][] subMenuItems = {

{"Thêm tin tức", "Quản lý tin tức"},

{"Thêm sinh viên", "Thêm giảng viên", "Thêm khoa", "Thêm lớp", "Thêm học phần"},

{"Sinh viên", "Giảng viên", "Khoa", "Lớp học", "Học phần"},

{}, // Lịch trình

{}, // Đăng ký học

{}, // Thanh toán

{"Hoạt động gần đây", "Lịch sử hóa đơn"}, // Nhật ký

{}, // Cài đặt

{} // Đăng xuất

};

int yPosition = 200;

for (int i = 0; i < menuTitles.length; i++) {

// Menu button

JButton menuButton = new JButton(menuTitles[i]);

menuButton.setBounds(45, yPosition, 209, 50);

menuButton.setForeground(Color.WHITE);

menuButton.setBackground(Color.BLACK);

menuButton.setBorderPainted(false);

menuButton.setHorizontalAlignment(SwingConstants.LEFT);

menuButton.setFont(new Font("Arial", Font.BOLD, 20));

menuPanel.add(menuButton);

// Popup menu for sub-items

JPopupMenu popupMenu = new JPopupMenu();

popupMenu.setBackground(Color.BLACK);

popupMenu.setPreferredSize(new Dimension(300, subMenuItems[i].length \* 66));

for (String subItem : subMenuItems[i]) {

JMenuItem menuItem = new JMenuItem(subItem);

menuItem.setBackground(Color.BLACK);

menuItem.setForeground(Color.WHITE);

menuItem.setFont(new Font("Arial", Font.BOLD, 18));

menuItem.setPreferredSize(new Dimension(300, 50));

menuItem.setBorder(new EmptyBorder(0, 15, 0, 0)); // Add x=15 offset

menuItem.addActionListener(this);

popupMenu.add(menuItem);

}

// Handle menu button click

String menuTitle = menuTitles[i];

if (subMenuItems[i].length == 0) {

// No sub-menu items: Perform action directly

menuButton.addActionListener(e -> {

handleMenuAction(menuTitle);

});

} else {

// Has sub-menu items: Show popup menu

menuButton.addActionListener(e -> popupMenu.show(menuButton, 257 - menuButton.getX(), menuButton.getHeight()));

}

yPosition += 60; // Adjusted spacing

}

// Current Students

ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icon/students.png"));

Image i2 = i1.getImage().getScaledInstance(120, 120, Image.SCALE\_DEFAULT);

ImageIcon i3 = new ImageIcon(i2);

JLabel img = new JLabel(i3);

img.setBounds(1368, 150, 120, 120);

add(img);

studentCountLabel = new JLabel();

studentCountLabel.setBounds(1320, 305, 200, 25);

studentCountLabel.setForeground(Color.BLACK);

studentCountLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(studentCountLabel);

// Current Teachers

ImageIcon teachersIcon = new ImageIcon(ClassLoader.getSystemResource("icon/female.png"));

Image teachersImage = teachersIcon.getImage().getScaledInstance(120, 120, Image.SCALE\_DEFAULT);

ImageIcon scaledTeachersIcon = new ImageIcon(teachersImage);

JLabel teachersIconLabel = new JLabel(scaledTeachersIcon);

teachersIconLabel.setBounds(1368, 363, 120, 120);

add(teachersIconLabel);

teacherCountLabel = new JLabel();

teacherCountLabel.setBounds(1320, 518, 200, 23);

teacherCountLabel.setForeground(Color.BLACK);

teacherCountLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(teacherCountLabel);

// Current Class

ImageIcon iii1 = new ImageIcon(ClassLoader.getSystemResource("icon/class.png"));

Image iii2 = iii1.getImage().getScaledInstance(120, 120, Image.SCALE\_DEFAULT);

ImageIcon iii3 = new ImageIcon(iii2);

JLabel iiimg = new JLabel(iii3);

iiimg.setBounds(1368, 576, 120, 120);

add(iiimg);

classCountLabel = new JLabel();

classCountLabel.setBounds(1320, 731, 200, 25);

classCountLabel.setForeground(Color.BLACK);

classCountLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(classCountLabel);

// Fetch counts from MySQL

updateCounts();

setSize(1540, 850);

setLayout(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

}

private void updateCounts() {

Connect c = new Connect();

Connection conn = c.getConnection();

Statement stmt = null;

ResultSet rs = null;

try {

if (conn == null) {

System.err.println("Database connection is null!");

return;

}

stmt = conn.createStatement();

// Fetch student count

rs = stmt.executeQuery("SELECT COUNT(\*) FROM student");

if (rs.next()) {

studentCountLabel.setText("Số sinh viên = " + rs.getInt(1));

}

rs = stmt.executeQuery("SELECT COUNT(\*) FROM lecturer");

if (rs.next()) {

teacherCountLabel.setText("Số giảng viên = " + rs.getInt(1));

}

rs = stmt.executeQuery("SELECT COUNT(\*) FROM class");

if (rs.next()) {

classCountLabel.setText("Số lớp học = " + rs.getInt(1));

}

} catch (Exception e) {

System.err.println("Lỗi số lượng sinh viên: " + e.getMessage());

e.printStackTrace();

} finally {

try {

if (rs != null) rs.close();

if (stmt != null) stmt.close();

if (conn != null) conn.close();

} catch (Exception e) {

System.err.println("Lỗi khi đóng tài nguyên cơ sở dữ liệu: " + e.getMessage());

}

}

}

private void handleMenuAction(String command) {

switch (command) {

case "Thêm thông tin":

new AddNews();

dispose();

break;

case "Quản lý thông tin":

new NewsManagement();

dispose();

break;

case "Lịch trình":

// Placeholder for Lịch trình action

JOptionPane.showMessageDialog(this, "Chức năng Lịch trình chưa được triển khai!");

break;

case "Đăng ký học":

// Placeholder for Đăng ký học action

JOptionPane.showMessageDialog(this, "Chức năng Đăng ký học chưa được triển khai!");

break;

case "Thanh toán":

// Placeholder for Thanh toán action

JOptionPane.showMessageDialog(this, "Chức năng Thanh toán chưa được triển khai!");

break;

case "Nhật ký":

// Placeholder for Nhật ký action

JOptionPane.showMessageDialog(this, "Vui lòng chọn một mục con trong Nhật ký!");

break;

case "Cài đặt":

// Placeholder for Cài đặt action

JOptionPane.showMessageDialog(this, "Chức năng Cài đặt chưa được triển khai!");

break;

case "Đăng xuất":

new Login();

dispose(); // Close the current Home frame

break;

// Sub-menu items

case "Thêm sinh viên":

new AddStudent();

break;

case "Thêm giảng viên":

new AddLecturer();

break;

case "Thêm khoa":

new AddDepartment();

dispose();

break;

case "Thêm lớp":

new AddClass();

break;

case "Thêm học phần":

new AddCourse();

break;

case "Sinh viên":

new StudentManagement();

dispose();

break;

case "Giảng viên":

new LecturerManagement();

dispose();

break;

case "Khoa":

new DepartmentManagement();

dispose();

break;

case "Lớp học":

new ClassManagement();

dispose();

break;

case "Học phần":

new CourseManagement();

dispose();

break;

case "Thêm tin tức":

// Placeholder for Thêm tin tức

JOptionPane.showMessageDialog(this, "Chức năng Thêm tin tức chưa được triển khai!");

break;

case "Quản lý tin tức":

// Placeholder for Quản lý tin tức

JOptionPane.showMessageDialog(this, "Chức năng Quản lý tin tức chưa được triển khai!");

break;

case "Hoạt động gần đây":

// Placeholder for Hoạt động gần đây

JOptionPane.showMessageDialog(this, "Chức năng Hoạt động gần đây chưa được triển khai!");

break;

case "Lịch sử hóa đơn":

// Placeholder for Lịch sử hóa đơn

JOptionPane.showMessageDialog(this, "Chức năng Lịch sử hóa đơn chưa được triển khai!");

break;

}

}

@Override

public void actionPerformed(ActionEvent e) {

String command = e.getActionCommand();

handleMenuAction(command);

}

public static void main(String[] args) {

new Home();

}

}

1. **Connect.java**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class Connect {

private static final String URL = "jdbc:mysql://localhost:3306/universitysystem";

private static final String USER = "root";

private static final String PASSWORD = "123456";

private Connection connection;

public Connect() {

try {

connection = DriverManager.getConnection(URL, USER, PASSWORD);

System.out.println("✅ Connected to database successfully.");

} catch (SQLException e) {

System.out.println("❌ Connection failed!");

e.printStackTrace();

}

}

public Connection getConnection() {

return connection;

}

}

1. **AddDepartment.java**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class AddDepartment extends JFrame implements ActionListener {

private JTextField tenkhoaField, truongkhoaField;

private JLabel makhoaLabel;

private JButton submitButton, cancelButton;

private String generatedMakhoa;

AddDepartment() {

getContentPane().setBackground(new Color(227, 237, 249));

// Title Label

JLabel titleLabel = new JLabel("Thêm khoa");

titleLabel.setBounds(146, 60, 240, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

add(titleLabel);

// Department Code (makhoa) - Now a JLabel

JLabel makhoaText = new JLabel("Mã khoa:");

makhoaText.setBounds(60, 130, 200, 30);

makhoaText.setFont(new Font("Serif", Font.BOLD, 20));

add(makhoaText);

generatedMakhoa = generateMakhoa();

makhoaLabel = new JLabel(generatedMakhoa);

makhoaLabel.setBounds(190, 130, 100, 30);

makhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(makhoaLabel);

// Department Name (tenkhoa)

JLabel tenkhoaLabel = new JLabel("Tên khoa:");

tenkhoaLabel.setBounds(60, 190, 200, 30);

tenkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(tenkhoaLabel);

tenkhoaField = new JTextField();

tenkhoaField.setBounds(60, 235, 360, 50);

tenkhoaField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(tenkhoaField);

// Head of Department (matruongkhoa)

JLabel truongkhoaLabel = new JLabel("Mã trưởng khoa:");

truongkhoaLabel.setBounds(60, 315, 200, 30);

truongkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(truongkhoaLabel);

truongkhoaField = new JTextField();

truongkhoaField.setBounds(60, 360, 360, 50);

truongkhoaField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(truongkhoaField);

// Submit Button

submitButton = new JButton("Tạo");

submitButton.setBounds(60, 450, 165, 60);

submitButton.setBackground(new Color(63, 134, 30));

submitButton.setForeground(Color.WHITE);

submitButton.setFont(new Font("Arial", Font.BOLD, 20));

submitButton.addActionListener(this);

add(submitButton);

// Cancel Button

cancelButton = new JButton("Hủy");

cancelButton.setBounds(255, 450, 165, 60);

cancelButton.setBackground(new Color(193, 14, 14));

cancelButton.setForeground(Color.WHITE);

cancelButton.setFont(new Font("Arial", Font.BOLD, 20));

cancelButton.addActionListener(this);

add(cancelButton);

setSize(480, 570);

setLocation(530, 80);

setLayout(null);

setUndecorated(true);

setVisible(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

private String generateMakhoa() {

Connect c = new Connect();

Connection conn = c.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "1000"; // Fallback

}

try {

String sql = "SELECT MAX(CAST(makhoa AS UNSIGNED)) AS max\_makhoa FROM department";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

int nextMakhoa = 1000; // Start at 1000 if no records exist

if (rs.next() && rs.getString("max\_makhoa") != null) {

nextMakhoa = rs.getInt("max\_makhoa") + 1;

}

rs.close();

stmt.close();

return String.format("%04d", nextMakhoa); // Format as 4-digit string

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tạo mã khoa: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "1000"; // Fallback

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private boolean isValidMatruongkhoa(String matruongkhoa) {

if (matruongkhoa.isEmpty()) {

return true; // matruongkhoa is nullable

}

Connect c = new Connect();

Connection conn = c.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return false;

}

try {

String sql = "SELECT COUNT(\*) FROM lecturer WHERE maGV = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, matruongkhoa);

ResultSet rs = stmt.executeQuery();

boolean exists = rs.next() && rs.getInt(1) > 0;

rs.close();

stmt.close();

return exists;

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi kiểm tra mã trưởng khoa: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

return false;

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == submitButton) {

String tenkhoa = tenkhoaField.getText().trim();

String matruongkhoa = truongkhoaField.getText().trim();

// Validation

if (tenkhoa.isEmpty()) {

JOptionPane.showMessageDialog(this, "Không được để trống tên khoa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenkhoa.length() > 60) {

JOptionPane.showMessageDialog(this, "Tên khoa không quá 60 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!matruongkhoa.isEmpty() && !matruongkhoa.matches("\\d{6}")) {

JOptionPane.showMessageDialog(this, "Mã trưởng khoa phải có 6 chữ số!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!matruongkhoa.isEmpty() && !isValidMatruongkhoa(matruongkhoa)) {

JOptionPane.showMessageDialog(this, "Mã trưởng khoa không tồn tại trong bảng giảng viên!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

// Save to database

Connect c = new Connect();

Connection conn = c.getConnection();

PreparedStatement pstmt = null;

try {

if (conn == null) {

JOptionPane.showMessageDialog(this, "Kết nối cơ sở dữ liệu không thành công", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String sql = "INSERT INTO department (makhoa, tenkhoa, matruongkhoa) VALUES (?, ?, ?)";

pstmt = conn.prepareStatement(sql);

pstmt.setString(1, generatedMakhoa);

pstmt.setString(2, tenkhoa);

pstmt.setString(3, matruongkhoa.isEmpty() ? null : matruongkhoa);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Thêm khoa thành công", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

dispose(); // Close the frame

} catch (SQLException ex) {

String errorMsg = "Lỗi thêm khoa: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

if (ex.getMessage().contains("makhoa")) {

errorMsg = "Mã khoa đã tồn tại!";

} else if (ex.getMessage().contains("tenkhoa")) {

errorMsg = "Tên khoa đã tồn tại!";

}

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

} finally {

try {

if (pstmt != null) pstmt.close();

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng tài nguyên cơ sở dữ liệu: " + ex.getMessage());

}

}

} else if (e.getSource() == cancelButton) {

tenkhoaField.setText("");

truongkhoaField.setText("");

dispose(); // Close the frame

}

}

public static void main(String[] args) {

new AddDepartment();

}

}

1. **AddLecturer.java**

package university.managerment.system;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Vector;

import com.toedter.calendar.JDateChooser;

public class LecturerManagement extends JFrame implements ActionListener {

private JTextField maGVField, hotenField, sdtField, diachiField, cccdField, searchField;

private JComboBox<String> tenkhoaCombo, gioitinhCombo, hocviCombo, chucvuCombo;

private JDateChooser ngaysinhChooser;

private JButton searchButtonRight, reloadButton, editButton, printButton, deleteButton, cancelButton;

private JTable lecturerTable;

private DefaultTableModel tableModel;

private Connect dbConnect;

private SimpleDateFormat dateFormat;

public LecturerManagement() {

setSize(1540, 850);

setLocationRelativeTo(null);

setUndecorated(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Initialize date format

dateFormat = new SimpleDateFormat("yyyy-MM-dd");

// Main panel

JPanel mainPanel = new JPanel(new BorderLayout());

mainPanel.setBackground(new Color(227, 237, 249));

// Left panel (input) - make it scrollable

JPanel leftPanelContent = new JPanel(null);

leftPanelContent.setBackground(new Color(255, 255, 255));

leftPanelContent.setPreferredSize(new Dimension(565, 900)); // Increased height to accommodate all fields

// Wrap leftPanelContent in a JScrollPane

JScrollPane leftPanelScroll = new JScrollPane(leftPanelContent);

leftPanelScroll.setPreferredSize(new Dimension(565, 850));

leftPanelScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

leftPanelScroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER);

// Title

JLabel titleLabel = new JLabel("Quản lý giảng viên", SwingConstants.CENTER);

titleLabel.setBounds(147, 30, 300, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

leftPanelContent.add(titleLabel);

// Input fields

// Mã GV

JLabel maGVLabel = new JLabel("Mã GV:");

maGVLabel.setBounds(30, 105, 175, 50);

maGVLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(maGVLabel);

maGVField = new JTextField();

maGVField.setBounds(205, 105, 330, 50);

maGVField.setFont(new Font("SanSerif", Font.PLAIN, 20));

maGVField.setBackground(new Color(227, 237, 249));

maGVField.setEditable(false); // Non-editable

leftPanelContent.add(maGVField);

// Tên khoa

JLabel tenkhoaLabel = new JLabel("Tên khoa:");

tenkhoaLabel.setBounds(30, 185, 175, 50);

tenkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(tenkhoaLabel);

tenkhoaCombo = new JComboBox<>();

tenkhoaCombo.setBounds(205, 185, 330, 50);

tenkhoaCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

tenkhoaCombo.setBackground(new Color(227, 237, 249));

leftPanelContent.add(tenkhoaCombo);

// Họ tên

JLabel hotenLabel = new JLabel("Họ tên:");

hotenLabel.setBounds(30, 265, 175, 50);

hotenLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(hotenLabel);

hotenField = new JTextField();

hotenField.setBounds(205, 265, 330, 50);

hotenField.setFont(new Font("SanSerif", Font.PLAIN, 20));

hotenField.setBackground(new Color(227, 237, 249));

leftPanelContent.add(hotenField);

// Giới tính

JLabel gioitinhLabel = new JLabel("Giới tính:");

gioitinhLabel.setBounds(30, 345, 175, 50);

gioitinhLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(gioitinhLabel);

gioitinhCombo = new JComboBox<>(new String[]{"Nam", "Nữ"});

gioitinhCombo.setBounds(205, 345, 330, 50);

gioitinhCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

gioitinhCombo.setBackground(new Color(227, 237, 249));

leftPanelContent.add(gioitinhCombo);

// Ngày sinh

JLabel ngaysinhLabel = new JLabel("Ngày sinh:");

ngaysinhLabel.setBounds(30, 425, 175, 50);

ngaysinhLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(ngaysinhLabel);

ngaysinhChooser = new JDateChooser();

ngaysinhChooser.setBounds(205, 425, 330, 50);

ngaysinhChooser.setFont(new Font("SanSerif", Font.PLAIN, 20));

ngaysinhChooser.setBackground(new Color(227, 237, 249));

ngaysinhChooser.setDateFormatString("yyyy-MM-dd");

leftPanelContent.add(ngaysinhChooser);

// SĐT

JLabel sdtLabel = new JLabel("SĐT:");

sdtLabel.setBounds(30, 505, 175, 50);

sdtLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(sdtLabel);

sdtField = new JTextField();

sdtField.setBounds(205, 505, 330, 50);

sdtField.setFont(new Font("SanSerif", Font.PLAIN, 20));

sdtField.setBackground(new Color(227, 237, 249));

leftPanelContent.add(sdtField);

// Địa chỉ

JLabel diachiLabel = new JLabel("Địa chỉ:");

diachiLabel.setBounds(30, 585, 175, 50);

diachiLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(diachiLabel);

diachiField = new JTextField();

diachiField.setBounds(205, 585, 330, 50);

diachiField.setFont(new Font("SanSerif", Font.PLAIN, 20));

diachiField.setBackground(new Color(227, 237, 249));

leftPanelContent.add(diachiField);

// CCCD

JLabel cccdLabel = new JLabel("CCCD:");

cccdLabel.setBounds(30, 665, 175, 50);

cccdLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(cccdLabel);

cccdField = new JTextField();

cccdField.setBounds(205, 665, 330, 50);

cccdField.setFont(new Font("SanSerif", Font.PLAIN, 20));

cccdField.setBackground(new Color(227, 237, 249));

leftPanelContent.add(cccdField);

// Học vị

JLabel hocviLabel = new JLabel("Học vị:");

hocviLabel.setBounds(30, 745, 175, 50);

hocviLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(hocviLabel);

hocviCombo = new JComboBox<>(new String[]{"Cử nhân", "Thạc sĩ", "Tiến sĩ", "PGS.TS", "GS.TS", "GS.TS.BS", "PGS.TS.BS", "TS.BS"});

hocviCombo.setBounds(205, 745, 330, 50);

hocviCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

hocviCombo.setBackground(new Color(227, 237, 249));

leftPanelContent.add(hocviCombo);

// Chức vụ

JLabel chucvuLabel = new JLabel("Chức vụ:");

chucvuLabel.setBounds(30, 825, 175, 50);

chucvuLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(chucvuLabel);

chucvuCombo = new JComboBox<>(new String[]{"Trưởng khoa", "Giảng viên", "Trợ giảng", "Phó trưởng khoa"});

chucvuCombo.setBounds(205, 825, 330, 50);

chucvuCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

chucvuCombo.setBackground(new Color(227, 237, 249));

leftPanelContent.add(chucvuCombo);

// Right panel (table, search field, buttons)

JPanel rightPanel = new JPanel(null);

rightPanel.setBackground(new Color(227, 237, 249));

rightPanel.setPreferredSize(new Dimension(975, 850));

// Search field

searchField = new JTextField();

searchField.setBounds(30, 40, 525, 60);

searchField.setFont(new Font("SanSerif", Font.PLAIN, 20));

rightPanel.add(searchField);

// Search and Reload buttons

searchButtonRight = createButton("Tìm kiếm", new Color(37, 61, 144), 585, 40, 165, 60);

reloadButton = createButton("Tải lại", new Color(37, 61, 144), 780, 40, 165, 60);

rightPanel.add(searchButtonRight);

rightPanel.add(reloadButton);

// Action buttons

editButton = createButton("Chỉnh sửa", new Color(37, 61, 144), 30, 760, 198, 60);

printButton = createButton("In", new Color(37, 61, 144), 270, 760, 198, 60);

deleteButton = createButton("Xóa", new Color(37, 61, 144), 509, 760, 198, 60);

cancelButton = createButton("Hủy", new Color(193, 14, 14), 748, 760, 198, 60);

rightPanel.add(editButton);

rightPanel.add(printButton);

rightPanel.add(deleteButton);

rightPanel.add(cancelButton);

// Table

String[] columns = {"ID", "Mã GV", "Tên khoa", "Họ tên", "Giới tính", "Ngày sinh", "SĐT", "Địa chỉ", "CCCD", "Học vị", "Chức vụ"};

tableModel = new DefaultTableModel(columns, 0);

lecturerTable = new JTable(tableModel);

lecturerTable.setFont(new Font("SanSerif", Font.PLAIN, 16));

lecturerTable.getTableHeader().setFont(new Font("Serif", Font.BOLD, 18));

lecturerTable.setRowHeight(30);

lecturerTable.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

// Table row selection

lecturerTable.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

int row = lecturerTable.getSelectedRow();

if (row >= 0) {

maGVField.setText((String) tableModel.getValueAt(row, 1));

maGVField.setEditable(false); // Ensure maGVField is non-editable

String tenkhoa = (String) tableModel.getValueAt(row, 2);

tenkhoaCombo.setSelectedItem(tenkhoa != null ? tenkhoa : "");

hotenField.setText((String) tableModel.getValueAt(row, 3));

gioitinhCombo.setSelectedItem((String) tableModel.getValueAt(row, 4));

try {

String dateStr = (String) tableModel.getValueAt(row, 5);

if (dateStr != null && !dateStr.isEmpty()) {

Date date = dateFormat.parse(dateStr);

ngaysinhChooser.setDate(date);

} else {

ngaysinhChooser.setDate(null);

}

} catch (Exception ex) {

ngaysinhChooser.setDate(null);

}

sdtField.setText((String) tableModel.getValueAt(row, 6));

diachiField.setText((String) tableModel.getValueAt(row, 7));

cccdField.setText((String) tableModel.getValueAt(row, 8));

hocviCombo.setSelectedItem((String) tableModel.getValueAt(row, 9));

chucvuCombo.setSelectedItem((String) tableModel.getValueAt(row, 10));

}

}

});

JScrollPane scrollPane = new JScrollPane(lecturerTable);

scrollPane.setBounds(30, 130, 915, 600);

rightPanel.add(scrollPane);

// Add panels to main

mainPanel.add(leftPanelScroll, BorderLayout.WEST);

mainPanel.add(rightPanel, BorderLayout.CENTER);

add(mainPanel);

// Populate tenkhoaCombo

populateDepartmentCombo();

// Load initial data

loadTableData();

setVisible(true);

}

private JButton createButton(String text, Color bg, int x, int y, int w, int h) {

JButton button = new JButton(text) {

@Override

protected void paintComponent(Graphics g) {

Graphics2D g2 = (Graphics2D) g.create();

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g2.setColor(getBackground());

g2.fillRoundRect(0, 0, getWidth(), getHeight(), 15, 15);

super.paintComponent(g2);

g2.dispose();

}

};

button.setBounds(x, y, w, h);

button.setBackground(bg);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 18));

button.setContentAreaFilled(false);

button.setBorderPainted(false);

button.addActionListener(this);

return button;

}

private void populateDepartmentCombo() {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT tenkhoa FROM department ORDER BY tenkhoa";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

tenkhoaCombo.removeAllItems();

tenkhoaCombo.addItem(""); // Allow null selection

while (rs.next()) {

tenkhoaCombo.addItem(rs.getString("tenkhoa"));

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải danh sách khoa: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void loadTableData() {

tableModel.setRowCount(0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, maGV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd, hocvi, chucvu " +

"FROM lecturer ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("maGV"));

row.add(rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "");

row.add(rs.getString("hoten"));

row.add(rs.getString("gioitinh"));

row.add(rs.getString("ngaysinh"));

row.add(rs.getString("sdt") != null ? rs.getString("sdt") : "");

row.add(rs.getString("diachi") != null ? rs.getString("diachi") : "");

row.add(rs.getString("cccd"));

row.add(rs.getString("hocvi"));

row.add(rs.getString("chucvu"));

tableModel.addRow(row);

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void clearFields() {

maGVField.setText("");

tenkhoaCombo.setSelectedIndex(0);

hotenField.setText("");

gioitinhCombo.setSelectedIndex(0);

ngaysinhChooser.setDate(null);

sdtField.setText("");

diachiField.setText("");

cccdField.setText("");

hocviCombo.setSelectedIndex(0);

chucvuCombo.setSelectedIndex(0);

searchField.setText("");

lecturerTable.clearSelection();

maGVField.setEditable(false); // Ensure maGVField remains non-editable

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == searchButtonRight) {

String searchText = searchField.getText().trim();

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, maGV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd, hocvi, chucvu " +

"FROM lecturer WHERE maGV LIKE ? OR hoten LIKE ? OR cccd LIKE ? ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

String param = "%" + searchText + "%";

stmt.setString(1, param);

stmt.setString(2, param);

stmt.setString(3, param);

ResultSet rs = stmt.executeQuery();

tableModel.setRowCount(0);

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("maGV"));

row.add(rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "");

row.add(rs.getString("hoten"));

row.add(rs.getString("gioitinh"));

row.add(rs.getString("ngaysinh"));

row.add(rs.getString("sdt") != null ? rs.getString("sdt") : "");

row.add(rs.getString("diachi") != null ? rs.getString("diachi") : "");

row.add(rs.getString("cccd"));

row.add(rs.getString("hocvi"));

row.add(rs.getString("chucvu"));

tableModel.addRow(row);

}

rs.close();

stmt.close();

clearFields();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tìm kiếm: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == reloadButton) {

clearFields();

loadTableData();

} else if (e.getSource() == editButton) {

int selectedRow = lecturerTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một giảng viên để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String id = (String) tableModel.getValueAt(selectedRow, 0);

String tenkhoa = (String) tenkhoaCombo.getSelectedItem();

String hoten = hotenField.getText().trim();

String gioitinh = (String) gioitinhCombo.getSelectedItem();

Date ngaysinhDate = ngaysinhChooser.getDate();

String sdt = sdtField.getText().trim();

String diachi = diachiField.getText().trim();

String cccd = cccdField.getText().trim();

String hocvi = (String) hocviCombo.getSelectedItem();

String chucvu = (String) chucvuCombo.getSelectedItem();

// Validation

if (hoten.isEmpty()) {

JOptionPane.showMessageDialog(this, "Họ tên không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (hoten.length() > 50) {

JOptionPane.showMessageDialog(this, "Họ tên không được vượt quá 50 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (ngaysinhDate == null) {

JOptionPane.showMessageDialog(this, "Ngày sinh không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String ngaysinh = dateFormat.format(ngaysinhDate);

if (!sdt.isEmpty() && !sdt.matches("\\d{10,15}")) {

JOptionPane.showMessageDialog(this, "SĐT phải có từ 10 đến 15 chữ số hoặc để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (diachi.length() > 100) {

JOptionPane.showMessageDialog(this, "Địa chỉ không được vượt quá 100 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!cccd.matches("\\d{12}")) {

JOptionPane.showMessageDialog(this, "CCCD phải có đúng 12 chữ số!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (hocvi == null || hocvi.isEmpty()) {

JOptionPane.showMessageDialog(this, "Học vị không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (chucvu == null || chucvu.isEmpty()) {

JOptionPane.showMessageDialog(this, "Chức vụ không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "UPDATE lecturer SET tenkhoa = ?, hoten = ?, gioitinh = ?, ngaysinh = ?, sdt = ?, diachi = ?, cccd = ?, hocvi = ?, chucvu = ? WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, tenkhoa.isEmpty() ? null : tenkhoa);

stmt.setString(2, hoten);

stmt.setString(3, gioitinh);

stmt.setString(4, ngaysinh);

stmt.setString(5, sdt.isEmpty() ? null : sdt);

stmt.setString(6, diachi.isEmpty() ? null : diachi);

stmt.setString(7, cccd);

stmt.setString(8, hocvi);

stmt.setString(9, chucvu);

stmt.setInt(10, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Chỉnh sửa giảng viên thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy giảng viên để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi chỉnh sửa giảng viên: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

if (ex.getMessage().contains("cccd")) {

errorMsg = "CCCD đã tồn tại!";

} else if (ex.getMessage().contains("sdt")) {

errorMsg = "SĐT đã tồn tại!";

} else if (ex.getMessage().contains("tenkhoa")) {

errorMsg = "Tên khoa không hợp lệ!";

}

} else if (ex.getSQLState().equals("HY000") && ex.getMessage().contains("CHECK")) {

errorMsg = "Dữ liệu không hợp lệ (học vị hoặc chức vụ)!";

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == deleteButton) {

int selectedRow = lecturerTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một giảng viên để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

int confirm = JOptionPane.showConfirmDialog(this, "Bạn có chắc chắn muốn xóa giảng viên này?", "Xác nhận xóa", JOptionPane.YES\_NO\_OPTION);

if (confirm != JOptionPane.YES\_OPTION) return;

String id = (String) tableModel.getValueAt(selectedRow, 0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "DELETE FROM lecturer WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Xóa giảng viên thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy giảng viên để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi xóa giảng viên: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

errorMsg = "Không thể xóa giảng viên vì có dữ liệu liên quan (ví dụ: lớp học)!";

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == printButton) {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try (PrintWriter writer = new PrintWriter("lecturers.csv")) {

writer.println("ID,Mã GV,Tên khoa,Họ tên,Giới tính,Ngày sinh,SĐT,Địa chỉ,CCCD,Học vị,Chức vụ");

String sql = "SELECT id, maGV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd, hocvi, chucvu " +

"FROM lecturer ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

String id = String.valueOf(rs.getInt("id"));

String maGV = rs.getString("maGV");

String tenkhoa = rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "";

String hoten = rs.getString("hoten");

String gioitinh = rs.getString("gioitinh");

String ngaysinh = rs.getString("ngaysinh");

String sdt = rs.getString("sdt") != null ? rs.getString("sdt") : "";

String diachi = rs.getString("diachi") != null ? rs.getString("diachi") : "";

String cccd = rs.getString("cccd");

String hocvi = rs.getString("hocvi");

String chucvu = rs.getString("chucvu");

writer.println(String.format("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s",

id, maGV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd, hocvi, chucvu));

}

rs.close();

stmt.close();

JOptionPane.showMessageDialog(this, "Dữ liệu đã được xuất ra lecturers.csv!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

} catch (FileNotFoundException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi xuất file: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi truy vấn dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == cancelButton) {

setVisible(false);

dispose();

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new LecturerManagement());

}

}

1. **AddStudent.java**

package university.managerment.system;

import com.toedter.calendar.JDateChooser;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.Random;

public class AddStudent extends JFrame implements ActionListener {

private JTextField hotenField, sdtField, diachiField, cccdField, khoaField;

private JComboBox<String> tenkhoaCombo;

private JDateChooser ngaysinhChooser;

private JRadioButton maleRadio, femaleRadio;

private JButton submitButton, cancelButton;

private Connect dbConnect;

private String generatedMaSV, generatedMatkhau;

private ButtonGroup gioitinhGroup;

public AddStudent() {

getContentPane().setBackground(new Color(227, 237, 249));

// Title Label

JLabel titleLabel = new JLabel("Thêm sinh viên");

titleLabel.setBounds(324, 60, 250, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

add(titleLabel);

// Khóa

JLabel khoaText = new JLabel("Khóa:");

khoaText.setBounds(60, 130, 150, 30);

khoaText.setFont(new Font("Serif", Font.BOLD, 20));

add(khoaText);

khoaField = new JTextField();

khoaField.setBounds(60, 175, 360, 50);

khoaField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(khoaField);

// Gender (gioitinh)

JLabel gioitinhLabel = new JLabel("Giới tính:");

gioitinhLabel.setBounds(450, 185, 100, 30);

gioitinhLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(gioitinhLabel);

maleRadio = new JRadioButton("Nam");

maleRadio.setBounds(605, 185, 80, 30);

maleRadio.setFont(new Font("Serif", Font.BOLD, 20));

maleRadio.setBackground(new Color(227, 237, 249));

add(maleRadio);

femaleRadio = new JRadioButton("Nữ");

femaleRadio.setBounds(745, 185, 80, 30);

femaleRadio.setFont(new Font("Serif", Font.BOLD, 20));

femaleRadio.setBackground(new Color(227, 237, 249));

add(femaleRadio);

gioitinhGroup = new ButtonGroup();

gioitinhGroup.add(maleRadio);

gioitinhGroup.add(femaleRadio);

// Full Name (hoten)

JLabel hotenLabel = new JLabel("Họ và tên:");

hotenLabel.setBounds(60, 255, 200, 30);

hotenLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(hotenLabel);

hotenField = new JTextField();

hotenField.setBounds(60, 300, 360, 50);

hotenField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(hotenField);

// Department Name (tenkhoa)

JLabel tenkhoaLabel = new JLabel("Tên khoa:");

tenkhoaLabel.setBounds(450, 255, 200, 30);

tenkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(tenkhoaLabel);

tenkhoaCombo = new JComboBox<>(loadDepartmentNames());

tenkhoaCombo.setBounds(450, 300, 360, 50);

tenkhoaCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(tenkhoaCombo);

// Phone Number (sdt)

JLabel sdtLabel = new JLabel("Số điện thoại:");

sdtLabel.setBounds(60, 380, 200, 30);

sdtLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(sdtLabel);

sdtField = new JTextField();

sdtField.setBounds(60, 425, 360, 50);

sdtField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(sdtField);

// Date of Birth (ngaysinh)

JLabel ngaysinhLabel = new JLabel("Ngày sinh:");

ngaysinhLabel.setBounds(450, 380, 200, 30);

ngaysinhLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(ngaysinhLabel);

ngaysinhChooser = new JDateChooser();

ngaysinhChooser.setBounds(450, 425, 360, 50);

ngaysinhChooser.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(ngaysinhChooser);

// Citizen ID (cccd)

JLabel cccdLabel = new JLabel("Số CCCD:");

cccdLabel.setBounds(60, 505, 200, 30);

cccdLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(cccdLabel);

cccdField = new JTextField();

cccdField.setBounds(60, 550, 360, 50);

cccdField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(cccdField);

// Address (diachi)

JLabel diachiLabel = new JLabel("Địa chỉ:");

diachiLabel.setBounds(450, 505, 200, 30);

diachiLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(diachiLabel);

diachiField = new JTextField();

diachiField.setBounds(450, 550, 360, 50);

diachiField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(diachiField);

// Submit Button

submitButton = new JButton("Tạo");

submitButton.setBounds(90, 640, 300, 60);

submitButton.setBackground(new Color(63, 134, 30));

submitButton.setForeground(Color.WHITE);

submitButton.setFont(new Font("Arial", Font.BOLD, 20));

submitButton.addActionListener(this);

add(submitButton);

// Cancel Button

cancelButton = new JButton("Hủy");

cancelButton.setBounds(490, 640, 300, 60);

cancelButton.setBackground(new Color(193, 14, 14));

cancelButton.setForeground(Color.WHITE);

cancelButton.setFont(new Font("Arial", Font.BOLD, 20));

cancelButton.addActionListener(this);

add(cancelButton);

setSize(870, 760);

setLocation(335, 20);

setLayout(null);

setUndecorated(true);

setVisible(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

private String[] loadDepartmentNames() {

ArrayList<String> departments = new ArrayList<>();

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn != null) {

try {

String sql = "SELECT tenkhoa FROM department ORDER BY tenkhoa";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

departments.add(rs.getString("tenkhoa"));

}

rs.close();

stmt.close();

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải danh sách khoa: " + e.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

if (departments.isEmpty()) {

departments.add("Không có khoa nào");

}

return departments.toArray(new String[0]);

}

private String generateMaSV(String khoa) {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

}

PreparedStatement stmt = null;

ResultSet rs = null;

try {

// Validate khoa

int khoaNum;

try {

khoaNum = Integer.parseInt(khoa);

if (khoaNum < 10 || khoaNum > 99) {

JOptionPane.showMessageDialog(this, "Khóa phải là số 2 chữ số hợp lệ (10-99)!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

}

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(this, "Khóa phải là số 2 chữ số!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

}

// Calculate year prefix (e.g., khóa 14 → 20, khóa 15 → 21)

int yearPrefix = 20 + (khoaNum - 14);

if (yearPrefix < 0 || yearPrefix > 99) {

JOptionPane.showMessageDialog(this, "Khóa không hợp lệ để tạo mã sinh viên!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

}

// Find the highest sequential number for this khoa

String sql = "SELECT MAX(CAST(SUBSTRING(maSV, 3, 4) AS UNSIGNED)) AS max\_num " +

"FROM student WHERE SUBSTRING(maSV, 1, 2) = ? AND LENGTH(maSV) = 6";

stmt = conn.prepareStatement(sql);

stmt.setString(1, String.format("%02d", yearPrefix));

rs = stmt.executeQuery();

int nextNum = 0; // Start from 0000

if (rs.next() && rs.getString("max\_num") != null) {

nextNum = rs.getInt("max\_num") + 1; // Increment the highest number

}

rs.close();

stmt.close();

if (nextNum > 9999) {

JOptionPane.showMessageDialog(this, "Đã đạt giới hạn mã sinh viên cho khóa này!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

}

// Generate maSV

String maSV = String.format("%02d%04d", yearPrefix, nextNum);

// Verify uniqueness

sql = "SELECT COUNT(\*) FROM student WHERE maSV = ?";

stmt = conn.prepareStatement(sql);

stmt.setString(1, maSV);

rs = stmt.executeQuery();

if (rs.next() && rs.getInt(1) > 0) {

JOptionPane.showMessageDialog(this, "Mã sinh viên đã tồn tại!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

}

return maSV;

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Lỗi cơ sở dữ liệu khi tạo mã sinh viên: " + e.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

} finally {

try {

if (rs != null) rs.close();

if (stmt != null) stmt.close();

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng tài nguyên: " + ex.getMessage());

}

}

}

private String generatePassword() {

String chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&\*.,";

Random random = new Random();

char[] password = new char[8];

// Ensure at least one digit, one uppercase, one lowercase, one special character

password[0] = "0123456789".charAt(random.nextInt(10));

password[1] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".charAt(random.nextInt(26));

password[2] = "abcdefghijklmnopqrstuvwxyz".charAt(random.nextInt(26));

password[3] = "!@#$%^&\*.,".charAt(random.nextInt(10));

// Fill remaining characters

for (int i = 4; i < 8; i++) {

password[i] = chars.charAt(random.nextInt(chars.length()));

}

// Shuffle password

for (int i = 7; i > 0; i--) {

int j = random.nextInt(i + 1);

char temp = password[i];

password[i] = password[j];

password[j] = temp;

}

return new String(password);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == submitButton) {

String hoten = hotenField.getText().trim();

String tenkhoa = (String) tenkhoaCombo.getSelectedItem();

String khoa = khoaField.getText().trim();

String sdt = sdtField.getText().trim();

String diachi = diachiField.getText().trim();

String cccd = cccdField.getText().trim();

String gioitinh = maleRadio.isSelected() ? "Male" : femaleRadio.isSelected() ? "Female" : "";

Date ngaysinh = ngaysinhChooser.getDate();

// Input validation

if (hoten.isEmpty() || cccd.isEmpty() || gioitinh.isEmpty() || khoa.isEmpty() || ngaysinh == null) {

JOptionPane.showMessageDialog(this, "Vui lòng điền đầy đủ các trường bắt buộc (Họ tên, CCCD, Giới tính, Khóa, Ngày sinh)!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenkhoa == null || tenkhoa.equals("Không có khoa nào")) {

JOptionPane.showMessageDialog(this, "Không có khoa nào để chọn! Vui lòng tạo khoa trước.", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (hoten.length() > 50) {

JOptionPane.showMessageDialog(this, "Họ và tên không được vượt quá 50 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!sdt.isEmpty() && !sdt.matches("\\d{10,15}")) {

JOptionPane.showMessageDialog(this, "Số điện thoại phải có từ 10-15 chữ số hoặc để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (diachi.length() > 100) {

JOptionPane.showMessageDialog(this, "Địa chỉ không được vượt quá 100 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!cccd.matches("\\d{12}")) {

JOptionPane.showMessageDialog(this, "Số CCCD phải có 12 chữ số!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!khoa.matches("\\d{2}")) {

JOptionPane.showMessageDialog(this, "Khóa phải là số có 2 chữ số!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

// Generate maSV based on khoa

generatedMaSV = generateMaSV(khoa);

if (generatedMaSV.isEmpty()) {

return; // Error message already shown in generateMaSV

}

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

String ngaysinhStr = sdf.format(ngaysinh);

// Generate password

generatedMatkhau = generatePassword();

// Database insertion

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

PreparedStatement stmt = null;

try {

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối với cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String sql = "INSERT INTO student (maSV, hoten, gioitinh, ngaysinh, sdt, diachi, cccd, matkhau, tenkhoa, khoa) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

stmt = conn.prepareStatement(sql);

stmt.setString(1, generatedMaSV);

stmt.setString(2, hoten);

stmt.setString(3, gioitinh);

stmt.setString(4, ngaysinhStr);

stmt.setString(5, sdt.isEmpty() ? null : sdt);

stmt.setString(6, diachi.isEmpty() ? null : diachi);

stmt.setString(7, cccd);

stmt.setString(8, generatedMatkhau);

stmt.setString(9, tenkhoa);

stmt.setString(10, khoa);

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Thêm sinh viên thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

// Reset form

hotenField.setText("");

sdtField.setText("");

diachiField.setText("");

cccdField.setText("");

khoaField.setText("");

ngaysinhChooser.setDate(null);

gioitinhGroup.clearSelection();

tenkhoaCombo.setSelectedIndex(0);

}

} catch (SQLException ex) {

String errorMsg = "Lỗi cơ sở dữ liệu: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

if (ex.getMessage().contains("sdt")) {

errorMsg = "Số điện thoại đã tồn tại!";

} else if (ex.getMessage().contains("cccd")) {

errorMsg = "Số CCCD đã tồn tại!";

} else if (ex.getMessage().contains("maSV")) {

errorMsg = "Mã sinh viên đã tồn tại!";

} else if (ex.getMessage().contains("tenkhoa")) {

errorMsg = "Tên khoa không hợp lệ!";

} else if (ex.getMessage().contains("email")) {

errorMsg = "Email đã tồn tại!";

}

} else if (ex.getSQLState().equals("HY000") && ex.getMessage().contains("CHECK")) {

errorMsg = "Dữ liệu không hợp lệ (giới tính hoặc khóa)!";

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

} finally {

try {

if (stmt != null) stmt.close();

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng tài nguyên: " + ex.getMessage());

}

}

} else if (e.getSource() == cancelButton) {

// Clear all fields and close the window

hotenField.setText("");

sdtField.setText("");

diachiField.setText("");

cccdField.setText("");

khoaField.setText("");

ngaysinhChooser.setDate(null);

gioitinhGroup.clearSelection();

tenkhoaCombo.setSelectedIndex(0);

dispose();

}

}

public static void main(String[] args) {

new AddStudent();

}

}

1. **AddCourse.java**

package university.managerment.system;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class AddCourse extends JFrame implements ActionListener {

private JTextField tenmonField, sotinchiField;

private JComboBox<String> tenkhoaComboBox;

private JLabel mamonLabel;

private JButton submitButton, cancelButton;

private Connect dbConnect;

AddCourse() {

getContentPane().setBackground(new Color(227, 237, 249));

// Title Label

JLabel titleLabel = new JLabel("Thêm học phần");

titleLabel.setBounds(120, 45, 240, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

add(titleLabel);

// Course ID (mamon) - JLabel

JLabel mamonTitleLabel = new JLabel("Mã học phần:");

mamonTitleLabel.setBounds(60, 115, 150, 30);

mamonTitleLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(mamonTitleLabel);

mamonLabel = new JLabel(generateMamon());

mamonLabel.setBounds(225, 115, 90, 30);

mamonLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(mamonLabel);

// Course Name (tenmon)

JLabel tenmonLabel = new JLabel("Tên học phần:");

tenmonLabel.setBounds(60, 175, 200, 30);

tenmonLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(tenmonLabel);

tenmonField = new JTextField();

tenmonField.setBounds(60, 220, 360, 50);

tenmonField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(tenmonField);

// Department Name (tenkhoa)

JLabel tenkhoaLabel = new JLabel("Tên khoa:");

tenkhoaLabel.setBounds(60, 300, 200, 30);

tenkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(tenkhoaLabel);

tenkhoaComboBox = new JComboBox<>();

tenkhoaComboBox.setBounds(60, 345, 360, 50);

tenkhoaComboBox.setFont(new Font("SanSerif", Font.PLAIN, 20));

populateDepartmentComboBox();

add(tenkhoaComboBox);

// Credits (sotinchi)

JLabel sotinchiLabel = new JLabel("Số tín chỉ:");

sotinchiLabel.setBounds(60, 425, 200, 30);

sotinchiLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(sotinchiLabel);

sotinchiField = new JTextField();

sotinchiField.setBounds(60, 470, 360, 50);

sotinchiField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(sotinchiField);

// Submit Button

submitButton = new JButton("Tạo");

submitButton.setBounds(60, 560, 165, 60);

submitButton.setBackground(new Color(63, 134, 30));

submitButton.setForeground(Color.WHITE);

submitButton.setFont(new Font("Arial", Font.BOLD, 20));

submitButton.addActionListener(this);

add(submitButton);

// Cancel Button

cancelButton = new JButton("Hủy");

cancelButton.setBounds(255, 560, 165, 60);

cancelButton.setBackground(new Color(193, 14, 14));

cancelButton.setForeground(Color.WHITE);

cancelButton.setFont(new Font("Arial", Font.BOLD, 20));

cancelButton.addActionListener(this);

add(cancelButton);

setSize(480, 665);

setLocation(530, 80);

setLayout(null);

setUndecorated(true);

setVisible(true);

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

}

private String generateMamon() {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "00001"; // Fallback

}

try {

String sql = "SELECT MAX(CAST(mamon AS UNSIGNED)) AS max\_mamon FROM course";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

int nextMamon = 1; // Default to 1 if no records exist

if (rs.next() && rs.getString("max\_mamon") != null) {

nextMamon = rs.getInt("max\_mamon") + 1;

}

rs.close();

stmt.close();

return String.format("%05d", nextMamon); // Format as 5-digit string

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tạo mã học phần: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

return "00001"; // Fallback

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void populateDepartmentComboBox() {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT tenkhoa FROM department ORDER BY tenkhoa";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

tenkhoaComboBox.removeAllItems();

while (rs.next()) {

tenkhoaComboBox.addItem(rs.getString("tenkhoa"));

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải danh sách khoa: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void resetForm() {

tenmonField.setText("");

sotinchiField.setText("");

tenkhoaComboBox.setSelectedIndex(0);

mamonLabel.setText(generateMamon());

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == submitButton) {

String tenmon = tenmonField.getText().trim();

String tinchiStr = sotinchiField.getText().trim();

String tenkhoa = (String) tenkhoaComboBox.getSelectedItem();

String mamon = mamonLabel.getText();

// Input validation

if (tenmon.isEmpty() || tinchiStr.isEmpty() || tenkhoa == null) {

JOptionPane.showMessageDialog(this, "Nhập thiếu dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenmon.length() > 60) {

JOptionPane.showMessageDialog(this, "Tên học phần không được quá 60 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

int tinchi;

try {

tinchi = Integer.parseInt(tinchiStr);

if (tinchi < 1 || tinchi > 10) {

JOptionPane.showMessageDialog(this, "Số tín chỉ phải trong khoảng 1 - 10!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Số tín chỉ phải là một con số hợp lệ!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

// Database insertion

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "INSERT INTO course (mamon, tenmon, tinchi, tenkhoa) VALUES (?, ?, ?, ?)";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, mamon);

stmt.setString(2, tenmon);

stmt.setInt(3, tinchi);

stmt.setString(4, tenkhoa);

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Học phần đã thêm thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

resetForm();

} else {

JOptionPane.showMessageDialog(this, "Không thể thêm học phần!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi thêm học phần: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

if (ex.getMessage().contains("mamon")) {

errorMsg = "Mã học phần đã tồn tại!";

} else if (ex.getMessage().contains("tenkhoa")) {

errorMsg = "Tên khoa không hợp lệ!";

}

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == cancelButton) {

tenmonField.setText("");

sotinchiField.setText("");

tenkhoaComboBox.setSelectedIndex(-1);

dispose();

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new AddCourse());

}

}

1. **AddClass.java**

package university.managerment.system;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class AddClass extends JFrame implements ActionListener {

private JTextField tenlopField, maGVField, namhocField, soluongSVField;

private JComboBox<String> hockyCombo, tenkhoaCombo, tenmonCombo;

private JButton submitButton, cancelButton;

private Connect dbConnect;

private JLabel malopLabel;

private String generatedMalop;

private Map<String, String> tenkhoaToMakhoaMap; // Maps tenkhoa to makhoa

public AddClass() {

// Initialize database connection and map

tenkhoaToMakhoaMap = new HashMap<>();

getContentPane().setBackground(new Color(227, 237, 249));

// Title Label

JLabel titleLabel = new JLabel("Thêm lớp học");

titleLabel.setBounds(334, 60, 250, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

add(titleLabel);

// Class ID (malop)

JLabel malopTitleLabel = new JLabel("Mã lớp:");

malopTitleLabel.setBounds(60, 130, 135, 30);

malopTitleLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(malopTitleLabel);

malopLabel = new JLabel();

malopLabel.setBounds(180, 130, 120, 30);

malopLabel.setFont(new Font("Serif", Font.BOLD, 20));

updateMalopLabel(); // Set initial malop

add(malopLabel);

// Department Name (tenkhoa)

JLabel tenkhoaLabel = new JLabel("Tên khoa:");

tenkhoaLabel.setBounds(450, 130, 200, 30);

tenkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(tenkhoaLabel);

// Populate tenkhoaCombo with department names

String[] tenkhoaOptions = loadDepartmentNames();

tenkhoaCombo = new JComboBox<>(tenkhoaOptions);

tenkhoaCombo.setBounds(450, 175, 360, 50);

tenkhoaCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

tenkhoaCombo.insertItemAt("", 0); // Add empty option for nullable tenkhoa

tenkhoaCombo.setSelectedIndex(0); // Select empty option by default

add(tenkhoaCombo);

// Add ActionListener to tenkhoaCombo to update tenmonCombo

tenkhoaCombo.addActionListener(e -> {

String selectedTenkhoa = (String) tenkhoaCombo.getSelectedItem();

updateTenmonCombo(selectedTenkhoa);

});

// Class Name (tenlop)

JLabel tenlopLabel = new JLabel("Tên lớp:");

tenlopLabel.setBounds(60, 255, 200, 30);

tenlopLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(tenlopLabel);

tenlopField = new JTextField();

tenlopField.setBounds(60, 300, 360, 50);

tenlopField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(tenlopField);

// Course Name (tenmon)

JLabel tenmonLabel = new JLabel("Tên học phần:");

tenmonLabel.setBounds(450, 255, 200, 30);

tenmonLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(tenmonLabel);

tenmonCombo = new JComboBox<>(new String[]{"Vui lòng chọn khoa trước"});

tenmonCombo.setBounds(450, 300, 360, 50);

tenmonCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

tenmonCombo.setEnabled(false); // Disabled until tenkhoa is selected

add(tenmonCombo);

// Lecturer Code (maGV)

JLabel maGVLabel = new JLabel("Mã giảng viên:");

maGVLabel.setBounds(60, 380, 200, 30);

maGVLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(maGVLabel);

maGVField = new JTextField();

maGVField.setBounds(60, 425, 360, 50);

maGVField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(maGVField);

// Student Count (soluongSV)

JLabel soluongSVLabel = new JLabel("Số lượng SV:");

soluongSVLabel.setBounds(450, 380, 215, 30);

soluongSVLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(soluongSVLabel);

soluongSVField = new JTextField();

soluongSVField.setBounds(450, 425, 360, 50);

soluongSVField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(soluongSVField);

// Semester (hocky)

JLabel hockyLabel = new JLabel("Học kỳ:");

hockyLabel.setBounds(60, 505, 200, 30);

hockyLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(hockyLabel);

String[] hockyOptions = {"1", "2", "3", "4"};

hockyCombo = new JComboBox<>(hockyOptions);

hockyCombo.setBounds(60, 550, 360, 50);

hockyCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(hockyCombo);

// Academic Year (namhoc)

JLabel namhocLabel = new JLabel("Năm học:");

namhocLabel.setBounds(450, 505, 200, 30);

namhocLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(namhocLabel);

namhocField = new JTextField();

namhocField.setBounds(450, 550, 360, 50);

namhocField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(namhocField);

// Submit Button

submitButton = new JButton("Tạo");

submitButton.setBounds(90, 640, 300, 60);

submitButton.setBackground(new Color(63, 134, 30));

submitButton.setForeground(Color.WHITE);

submitButton.setFont(new Font("Arial", Font.BOLD, 20));

submitButton.addActionListener(this);

add(submitButton);

// Cancel Button

cancelButton = new JButton("Hủy");

cancelButton.setBounds(490, 640, 300, 60);

cancelButton.setBackground(new Color(193, 14, 14));

cancelButton.setForeground(Color.WHITE);

cancelButton.setFont(new Font("Arial", Font.BOLD, 20));

cancelButton.addActionListener(this);

add(cancelButton);

setSize(870, 760);

setLocation(381, 10);

setLayout(null);

setUndecorated(true);

setVisible(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

private String[] loadDepartmentNames() {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

tenkhoaToMakhoaMap.clear();

ArrayList<String> tenkhoaList = new ArrayList<>();

try {

String sql = "SELECT makhoa, tenkhoa FROM department ORDER BY tenkhoa";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

String makhoa = rs.getString("makhoa");

String tenkhoa = rs.getString("tenkhoa");

tenkhoaList.add(tenkhoa);

tenkhoaToMakhoaMap.put(tenkhoa, makhoa);

}

rs.close();

stmt.close();

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải danh sách khoa: " + e.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

return tenkhoaList.toArray(new String[0]);

}

private void updateTenmonCombo(String tenkhoa) {

ArrayList<String> tenmonList = new ArrayList<>();

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

tenmonCombo.setModel(new DefaultComboBoxModel<>(new String[]{"Không thể tải môn học"}));

tenmonCombo.setEnabled(false);

return;

}

try {

String sql;

PreparedStatement stmt;

if (tenkhoa == null || tenkhoa.isEmpty()) {

sql = "SELECT tenmon FROM course WHERE tenkhoa IS NULL ORDER BY tenmon";

stmt = conn.prepareStatement(sql);

} else {

sql = "SELECT tenmon FROM course WHERE tenkhoa = ? ORDER BY tenmon";

stmt = conn.prepareStatement(sql);

stmt.setString(1, tenkhoa);

}

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

String tenmon = rs.getString("tenmon");

tenmonList.add(tenmon);

}

rs.close();

stmt.close();

if (tenmonList.isEmpty()) {

tenmonCombo.setModel(new DefaultComboBoxModel<>(new String[]{"Không có môn học"}));

tenmonCombo.setEnabled(false);

} else {

tenmonCombo.setModel(new DefaultComboBoxModel<>(tenmonList.toArray(new String[0])));

tenmonCombo.setEnabled(true);

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải danh sách môn học: " + e.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

tenmonCombo.setModel(new DefaultComboBoxModel<>(new String[]{"Không thể tải môn học"}));

tenmonCombo.setEnabled(false);

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private String generateMalop() {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

return "";

}

PreparedStatement stmt = null;

ResultSet rs = null;

String malop = "00001"; // Default if no classes exist

try {

String sql = "SELECT MAX(CAST(malop AS UNSIGNED)) AS max\_malop FROM class WHERE LENGTH(malop) = 5 AND malop REGEXP '^[0-9]+$'";

stmt = conn.prepareStatement(sql);

rs = stmt.executeQuery();

if (rs.next() && rs.getString("max\_malop") != null) {

int nextNum = rs.getInt("max\_malop") + 1;

if (nextNum > 99999) {

JOptionPane.showMessageDialog(this, "Đã đạt giới hạn mã lớp!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

}

malop = String.format("%05d", nextNum); // e.g., 00002, 00003

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Lỗi khi tạo mã lớp: " + e.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

} finally {

try {

if (rs != null) rs.close();

if (stmt != null) stmt.close();

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng tài nguyên: " + ex.getMessage());

}

}

return malop;

}

private void updateMalopLabel() {

generatedMalop = generateMalop();

malopLabel.setText(generatedMalop.isEmpty() ? "Lỗi tạo mã" : generatedMalop);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == submitButton) {

String tenlop = tenlopField.getText().trim();

String tenmon = (String) tenmonCombo.getSelectedItem();

String tenkhoa = (String) tenkhoaCombo.getSelectedItem();

String makhoa = tenkhoaToMakhoaMap.getOrDefault(tenkhoa, null);

String maGV = maGVField.getText().trim();

String namhoc = namhocField.getText().trim();

String hocky = (String) hockyCombo.getSelectedItem();

String soluongSVStr = soluongSVField.getText().trim();

// Input validation

if (tenlop.isEmpty() || tenmon == null || maGV.isEmpty() || namhoc.isEmpty() || hocky == null) {

JOptionPane.showMessageDialog(this, "Vui lòng điền đầy đủ các trường bắt buộc (Tên lớp, Tên học phần, Mã giảng viên, Năm học, Học kỳ)!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenmon.equals("Không có môn học") || tenmon.equals("Không thể tải môn học")) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một học phần hợp lệ!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenlop.length() > 100) {

JOptionPane.showMessageDialog(this, "Tên lớp không được vượt quá 100 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenmon.length() > 100) {

JOptionPane.showMessageDialog(this, "Tên học phần không được vượt quá 100 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenkhoa != null && !tenkhoa.isEmpty() && !tenkhoaToMakhoaMap.containsKey(tenkhoa)) {

JOptionPane.showMessageDialog(this, "Tên khoa không hợp lệ!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!maGV.matches("\\d{6}")) {

JOptionPane.showMessageDialog(this, "Mã giảng viên phải có đúng 6 chữ số!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!namhoc.matches("\\d{4}-\\d{4}")) {

JOptionPane.showMessageDialog(this, "Năm học phải có định dạng YYYY-YYYY (ví dụ: 2024-2025)!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

int soluongSV = 0;

if (!soluongSVStr.isEmpty()) {

try {

soluongSV = Integer.parseInt(soluongSVStr);

if (soluongSV < 0 || soluongSV > 500) {

JOptionPane.showMessageDialog(this, "Số lượng sinh viên phải từ 0 đến 500!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Số lượng sinh viên phải là số nguyên!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

}

// Regenerate malop to ensure uniqueness

updateMalopLabel();

if (generatedMalop.isEmpty()) {

return; // Error message already shown in generateMalop

}

// Database insertion

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

PreparedStatement stmt = null;

try {

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối với cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String sql = "INSERT INTO class (malop, tenlop, tenmon, tenkhoa, soluongSV, maGV, namhoc, hocky) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

stmt = conn.prepareStatement(sql);

stmt.setString(1, generatedMalop);

stmt.setString(2, tenlop);

stmt.setString(3, tenmon);

stmt.setString(4, tenkhoa.isEmpty() ? null : tenkhoa);

stmt.setInt(5, soluongSV);

stmt.setString(6, maGV);

stmt.setString(7, namhoc);

stmt.setInt(8, Integer.parseInt(hocky));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Thêm lớp học thành công!","Thành công", JOptionPane.INFORMATION\_MESSAGE);

// Reset form

updateMalopLabel(); // Generate new malop for next entry

tenlopField.setText("");

tenmonCombo.setModel(new DefaultComboBoxModel<>(new String[]{"Vui lòng chọn khoa trước"}));

tenmonCombo.setEnabled(false);

tenkhoaCombo.setSelectedIndex(0);

maGVField.setText("");

namhocField.setText("");

soluongSVField.setText("");

hockyCombo.setSelectedIndex(0);

}

} catch (SQLException ex) {

String errorMsg = "Lỗi cơ sở dữ liệu: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

if (ex.getMessage().contains("tenmon")) {

errorMsg = "Tên học phần không hợp lệ!";

} else if (ex.getMessage().contains("maGV")) {

errorMsg = "Mã giảng viên không hợp lệ!";

} else if (ex.getMessage().contains("tenkhoa")) {

errorMsg = "Tên khoa không hợp lệ!";

} else if (ex.getMessage().contains("malop")) {

errorMsg = "Mã lớp đã tồn tại!";

}

} else if (ex.getSQLState().equals("HY000") && ex.getMessage().contains("CHECK")) {

errorMsg = "Dữ liệu không hợp lệ (số lượng SV, năm học, hoặc học kỳ)!";

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

} finally {

try {

if (stmt != null) stmt.close();

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng tài nguyên: " + ex.getMessage());

}

}

} else if (e.getSource() == cancelButton) {

// Clear all fields and close the window

tenlopField.setText("");

tenmonCombo.setModel(new DefaultComboBoxModel<>(new String[]{"Vui lòng chọn khoa trước"}));

tenmonCombo.setEnabled(false);

tenkhoaCombo.setSelectedIndex(0);

maGVField.setText("");

namhocField.setText("");

soluongSVField.setText("");

hockyCombo.setSelectedIndex(0);

dispose();

}

}

public static void main(String[] args) {

new AddClass();

}

}

1. **AddNews.java**

package university.managerment.system;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

public class AddNews extends JFrame implements ActionListener {

private JTextField matinField, tieudeField, nguoitaoField, ngaytaoField, tailieuField;

private JTextArea noidungArea;

private JButton submitButton, cancelButton, uploadButton;

private Connect dbConnect;

private SimpleDateFormat dateFormat;

public AddNews() {

getContentPane().setBackground(new Color(227, 237, 249));

// Initialize date format

dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

// Title Label

JLabel titleLabel = new JLabel("Tin tức mới");

titleLabel.setBounds(348, 60, 250, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

add(titleLabel);

// Mã tin

JLabel matinLabel = new JLabel("Mã tin:");

matinLabel.setBounds(60, 130, 150, 30);

matinLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(matinLabel);

matinField = new JTextField();

matinField.setBounds(60, 175, 360, 50);

matinField.setFont(new Font("SanSerif", Font.PLAIN, 20));

matinField.setEditable(false); // Non-editable

matinField.setText(generateMatin()); // Auto-generate matin

add(matinField);

// Ngày tạo

JLabel ngaytaoLabel = new JLabel("Ngày tạo:");

ngaytaoLabel.setBounds(450, 130, 150, 30);

ngaytaoLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(ngaytaoLabel);

ngaytaoField = new JTextField();

ngaytaoField.setBounds(450, 175, 360, 50);

ngaytaoField.setFont(new Font("SanSerif", Font.PLAIN, 20));

ngaytaoField.setEditable(false); // Non-editable

ngaytaoField.setText(dateFormat.format(new Date())); // Set current timestamp

add(ngaytaoField);

// Tiêu đề

JLabel tieudeLabel = new JLabel("Tiêu đề:");

tieudeLabel.setBounds(60, 255, 150, 30);

tieudeLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(tieudeLabel);

tieudeField = new JTextField();

tieudeField.setBounds(60, 300, 360, 50);

tieudeField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(tieudeField);

// Người tạo

JLabel nguoitaoLabel = new JLabel("Tên người tạo:");

nguoitaoLabel.setBounds(450, 255, 150, 30);

nguoitaoLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(nguoitaoLabel);

nguoitaoField = new JTextField();

nguoitaoField.setBounds(450, 300, 360, 50);

nguoitaoField.setFont(new Font("SanSerif", Font.PLAIN, 20));

add(nguoitaoField);

// Tài liệu

JLabel tailieuLabel = new JLabel("Tài liệu:");

tailieuLabel.setBounds(60, 380, 150, 30);

tailieuLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(tailieuLabel);

tailieuField = new JTextField();

tailieuField.setBounds(60, 425, 590, 50);

tailieuField.setFont(new Font("SanSerif", Font.PLAIN, 20));

tailieuField.setEditable(false); // Non-editable

add(tailieuField);

uploadButton = new JButton("File");

uploadButton.setBounds(680, 415, 130, 60);

uploadButton.setBackground(new Color(63, 134, 30));

uploadButton.setForeground(Color.WHITE);

uploadButton.setFont(new Font("Arial", Font.BOLD, 16));

uploadButton.addActionListener(this);

add(uploadButton);

// Nội dung

JLabel noidungLabel = new JLabel("Nội dung:");

noidungLabel.setBounds(60, 505, 150, 30);

noidungLabel.setFont(new Font("Serif", Font.BOLD, 20));

add(noidungLabel);

noidungArea = new JTextArea();

noidungArea.setFont(new Font("SanSerif", Font.PLAIN, 20));

noidungArea.setLineWrap(true);

noidungArea.setWrapStyleWord(true);

JScrollPane noidungScroll = new JScrollPane(noidungArea);

noidungScroll.setBounds(60, 550, 760, 180);

add(noidungScroll);

// Submit Button

submitButton = new JButton("Tạo");

submitButton.setBounds(90, 770, 300, 60);

submitButton.setBackground(new Color(63, 134, 30));

submitButton.setForeground(Color.WHITE);

submitButton.setFont(new Font("Arial", Font.BOLD, 20));

submitButton.addActionListener(this);

add(submitButton);

// Cancel Button

cancelButton = new JButton("Hủy");

cancelButton.setBounds(490, 770, 300, 60);

cancelButton.setBackground(new Color(193, 14, 14));

cancelButton.setForeground(Color.WHITE);

cancelButton.setFont(new Font("Arial", Font.BOLD, 20));

cancelButton.addActionListener(this);

add(cancelButton);

setSize(870, 890);

setLocation(335, 0);

setLayout(null);

setUndecorated(true);

setVisible(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

private String generateMatin() {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

}

PreparedStatement stmt = null;

ResultSet rs = null;

try {

// Find the highest sequential number

String sql = "SELECT MAX(CAST(SUBSTRING(matin, 4, 5) AS UNSIGNED)) AS max\_num " +

"FROM news WHERE matin LIKE 'TIN%' AND LENGTH(matin) = 8";

stmt = conn.prepareStatement(sql);

rs = stmt.executeQuery();

int nextNum = 1; // Start from 00001

if (rs.next() && rs.getString("max\_num") != null) {

nextNum = rs.getInt("max\_num") + 1; // Increment the highest number

}

rs.close();

stmt.close();

if (nextNum > 99999) {

JOptionPane.showMessageDialog(this, "Đã đạt giới hạn mã tin tức!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

}

// Generate matin

String matin = String.format("TIN%05d", nextNum);

// Verify uniqueness

sql = "SELECT COUNT(\*) FROM news WHERE matin = ?";

stmt = conn.prepareStatement(sql);

stmt.setString(1, matin);

rs = stmt.executeQuery();

if (rs.next() && rs.getInt(1) > 0) {

JOptionPane.showMessageDialog(this, "Mã tin tức đã tồn tại!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

}

return matin;

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Lỗi cơ sở dữ liệu khi tạo mã tin tức: " + e.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

return "";

} finally {

try {

if (rs != null) rs.close();

if (stmt != null) stmt.close();

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng tài nguyên: " + ex.getMessage());

}

}

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == uploadButton) {

JFileChooser fileChooser = new JFileChooser();

int result = fileChooser.showOpenDialog(this);

if (result == JFileChooser.APPROVE\_OPTION) {

File selectedFile = fileChooser.getSelectedFile();

tailieuField.setText(selectedFile.getAbsolutePath());

}

} else if (e.getSource() == submitButton) {

String matin = matinField.getText().trim();

String tieude = tieudeField.getText().trim();

String noidung = noidungArea.getText().trim();

String tailieu = tailieuField.getText().trim();

String nguoitao = nguoitaoField.getText().trim();

String ngaytao = ngaytaoField.getText().trim();

// Input validation

if (matin.isEmpty()) {

JOptionPane.showMessageDialog(this, "Mã tin tức không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tieude.isEmpty()) {

JOptionPane.showMessageDialog(this, "Tiêu đề không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tieude.length() > 200) {

JOptionPane.showMessageDialog(this, "Tiêu đề không được vượt quá 200 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (noidung.isEmpty()) {

JOptionPane.showMessageDialog(this, "Nội dung không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (nguoitao.length() > 100) {

JOptionPane.showMessageDialog(this, "Tên người tạo không được vượt quá 100 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!tailieu.isEmpty()) {

File file = new File(tailieu);

if (!file.exists()) {

JOptionPane.showMessageDialog(this, "Tệp tài liệu không tồn tại!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tailieu.length() > 255) {

JOptionPane.showMessageDialog(this, "Đường dẫn tài liệu không được vượt quá 255 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

}

// Database insertion

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

PreparedStatement stmt = null;

try {

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối với cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String sql = "INSERT INTO news (matin, tieude, noidung, tailieu\_url, nguoitao) VALUES (?, ?, ?, ?, ?)";

stmt = conn.prepareStatement(sql);

stmt.setString(1, matin);

stmt.setString(2, tieude);

stmt.setString(3, noidung);

stmt.setString(4, tailieu.isEmpty() ? null : tailieu);

stmt.setString(5, nguoitao.isEmpty() ? null : nguoitao);

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Thêm tin tức thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

// Reset form

matinField.setText(generateMatin());

tieudeField.setText("");

noidungArea.setText("");

tailieuField.setText("");

nguoitaoField.setText("");

ngaytaoField.setText(dateFormat.format(new Date()));

}

} catch (SQLException ex) {

String errorMsg = "Lỗi cơ sở dữ liệu: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

if (ex.getMessage().contains("matin")) {

errorMsg = "Mã tin tức đã tồn tại!";

}

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (stmt != null) stmt.close();

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng tài nguyên: " + ex.getMessage());

}

}

} else if (e.getSource() == cancelButton) {

// Clear all fields and close the window

matinField.setText("");

tieudeField.setText("");

noidungArea.setText("");

tailieuField.setText("");

nguoitaoField.setText("");

ngaytaoField.setText("");

dispose();

}

}

public static void main(String[] args) {

new AddNews();

}

}

1. **ClassManegement.java**

package university.managerment.system;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.Vector;

public class ClassManagement extends JFrame implements ActionListener {

private JTextField malopField, tenlopField, soluongSVField, maGVField, namhocField, searchField;

private JComboBox<String> tenkhoaCombo, tenmonCombo, hockyCombo;

private JButton searchButtonRight, reloadButton, editButton, printButton, deleteButton, cancelButton;

private JTable classTable;

private DefaultTableModel tableModel;

private Connect dbConnect;

public ClassManagement() {

setSize(1540, 850);

setLocationRelativeTo(null);

setUndecorated(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Main panel

JPanel mainPanel = new JPanel(new BorderLayout());

mainPanel.setBackground(new Color(227, 237, 249));

// Left panel (input)

JPanel leftPanel = new JPanel(null);

leftPanel.setBackground(new Color(255, 255, 255));

leftPanel.setPreferredSize(new Dimension(565, 850));

// Title

JLabel titleLabel = new JLabel("Quản Lý Lớp Học", SwingConstants.CENTER);

titleLabel.setBounds(160, 30, 250, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

leftPanel.add(titleLabel);

// Input fields

// Mã lớp (malop)

JLabel malopLabel = new JLabel("Mã lớp:");

malopLabel.setBounds(30, 105, 175, 50);

malopLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(malopLabel);

malopField = new JTextField();

malopField.setBounds(205, 105, 330, 50);

malopField.setFont(new Font("SanSerif", Font.PLAIN, 20));

malopField.setBackground(new Color(227, 237, 249));

malopField.setEditable(false); // Non-editable

leftPanel.add(malopField);

// Tên khoa (tenkhoa)

JLabel tenkhoaLabel = new JLabel("Tên khoa:");

tenkhoaLabel.setBounds(30, 185, 175, 50);

tenkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(tenkhoaLabel);

tenkhoaCombo = new JComboBox<>();

tenkhoaCombo.setBounds(205, 185, 330, 50);

tenkhoaCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

tenkhoaCombo.setBackground(new Color(227, 237, 249));

tenkhoaCombo.addActionListener(e -> updateTenmonCombo((String) tenkhoaCombo.getSelectedItem()));

leftPanel.add(tenkhoaCombo);

// Tên lớp (tenlop)

JLabel tenlopLabel = new JLabel("Tên lớp:");

tenlopLabel.setBounds(30, 265, 175, 50);

tenlopLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(tenlopLabel);

tenlopField = new JTextField();

tenlopField.setBounds(205, 265, 330, 50);

tenlopField.setFont(new Font("SanSerif", Font.PLAIN, 20));

tenlopField.setBackground(new Color(227, 237, 249));

leftPanel.add(tenlopField);

// Tên môn (tenmon)

JLabel tenmonLabel = new JLabel("Tên môn:");

tenmonLabel.setBounds(30, 345, 175, 50);

tenmonLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(tenmonLabel);

tenmonCombo = new JComboBox<>(new String[]{"Vui lòng chọn khoa trước"});

tenmonCombo.setBounds(205, 345, 330, 50);

tenmonCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

tenmonCombo.setBackground(new Color(227, 237, 249));

tenmonCombo.setEnabled(false); // Disabled until tenkhoa is selected

leftPanel.add(tenmonCombo);

// Số lượng SV (soluongSV)

JLabel soluongSVLabel = new JLabel("Số lượng SV:");

soluongSVLabel.setBounds(30, 425, 175, 50);

soluongSVLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(soluongSVLabel);

soluongSVField = new JTextField();

soluongSVField.setBounds(205, 425, 330, 50);

soluongSVField.setFont(new Font("SanSerif", Font.PLAIN, 20));

soluongSVField.setBackground(new Color(227, 237, 249));

leftPanel.add(soluongSVField);

// Mã giảng viên (maGV)

JLabel maGVLabel = new JLabel("Mã giảng viên:");

maGVLabel.setBounds(30, 505, 175, 50);

maGVLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(maGVLabel);

maGVField = new JTextField();

maGVField.setBounds(205, 505, 330, 50);

maGVField.setFont(new Font("SanSerif", Font.PLAIN, 20));

maGVField.setBackground(new Color(227, 237, 249));

leftPanel.add(maGVField);

// Năm học (namhoc)

JLabel namhocLabel = new JLabel("Năm học:");

namhocLabel.setBounds(30, 585, 175, 50);

namhocLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(namhocLabel);

namhocField = new JTextField();

namhocField.setBounds(205, 585, 330, 50);

namhocField.setFont(new Font("SanSerif", Font.PLAIN, 20));

namhocField.setBackground(new Color(227, 237, 249));

leftPanel.add(namhocField);

// Học kỳ (hocky)

JLabel hockyLabel = new JLabel("Học kỳ:");

hockyLabel.setBounds(30, 665, 175, 50);

hockyLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(hockyLabel);

hockyCombo = new JComboBox<>(new String[]{"1", "2", "3", "4"});

hockyCombo.setBounds(205, 665, 330, 50);

hockyCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

hockyCombo.setBackground(new Color(227, 237, 249));

leftPanel.add(hockyCombo);

// Right panel (table, search field, buttons)

JPanel rightPanel = new JPanel(null);

rightPanel.setBackground(new Color(227, 237, 249));

rightPanel.setPreferredSize(new Dimension(975, 850));

// Search field

searchField = new JTextField();

searchField.setBounds(30, 40, 525, 60);

searchField.setFont(new Font("SanSerif", Font.PLAIN, 20));

rightPanel.add(searchField);

// Search and Reload buttons

searchButtonRight = createButton("Tìm kiếm", new Color(37, 61, 144), 585, 40, 165, 60);

reloadButton = createButton("Tải lại", new Color(37, 61, 144), 780, 40, 165, 60);

rightPanel.add(searchButtonRight);

rightPanel.add(reloadButton);

// Action buttons

editButton = createButton("Chỉnh sửa", new Color(37, 61, 144), 30, 760, 198, 60);

printButton = createButton("In", new Color(37, 61, 144), 270, 760, 198, 60);

deleteButton = createButton("Xóa", new Color(37, 61, 144), 509, 760, 198, 60);

cancelButton = createButton("Hủy", new Color(193, 14, 14), 748, 760, 198, 60);

rightPanel.add(editButton);

rightPanel.add(printButton);

rightPanel.add(deleteButton);

rightPanel.add(cancelButton);

// Table

String[] columns = {"ID", "Mã lớp", "Tên lớp", "Tên môn", "Tên khoa", "Số lượng SV", "Mã GV", "Năm học", "Học kỳ"};

tableModel = new DefaultTableModel(columns, 0);

classTable = new JTable(tableModel);

classTable.setFont(new Font("SanSerif", Font.PLAIN, 16));

classTable.getTableHeader().setFont(new Font("Serif", Font.BOLD, 18));

classTable.setRowHeight(30);

classTable.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

// Table row selection

classTable.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

int row = classTable.getSelectedRow();

if (row >= 0) {

malopField.setText((String) tableModel.getValueAt(row, 1));

malopField.setEditable(false); // Ensure malopField is non-editable

tenlopField.setText((String) tableModel.getValueAt(row, 2));

String tenkhoa = (String) tableModel.getValueAt(row, 4);

tenkhoaCombo.setSelectedItem(tenkhoa != null ? tenkhoa : "");

updateTenmonCombo(tenkhoa); // Update tenmonCombo based on tenkhoa

tenmonCombo.setSelectedItem((String) tableModel.getValueAt(row, 3));

soluongSVField.setText(String.valueOf(tableModel.getValueAt(row, 5)));

maGVField.setText((String) tableModel.getValueAt(row, 6));

namhocField.setText((String) tableModel.getValueAt(row, 7));

hockyCombo.setSelectedItem(String.valueOf(tableModel.getValueAt(row, 8)));

}

}

});

JScrollPane scrollPane = new JScrollPane(classTable);

scrollPane.setBounds(30, 130, 915, 600);

rightPanel.add(scrollPane);

// Add panels to main

mainPanel.add(leftPanel, BorderLayout.WEST);

mainPanel.add(rightPanel, BorderLayout.CENTER);

add(mainPanel);

// Populate tenkhoaCombo

populateDepartmentCombo();

// Load initial data

loadTableData();

setVisible(true);

}

private JButton createButton(String text, Color bg, int x, int y, int w, int h) {

JButton button = new JButton(text) {

@Override

protected void paintComponent(Graphics g) {

Graphics2D g2 = (Graphics2D) g.create();

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g2.setColor(getBackground());

g2.fillRoundRect(0, 0, getWidth(), getHeight(), 15, 15);

super.paintComponent(g2);

g2.dispose();

}

};

button.setBounds(x, y, w, h);

button.setBackground(bg);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 18));

button.setContentAreaFilled(false);

button.setBorderPainted(false);

button.addActionListener(this);

return button;

}

private void populateDepartmentCombo() {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT tenkhoa FROM department ORDER BY tenkhoa";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

tenkhoaCombo.removeAllItems();

tenkhoaCombo.addItem(""); // Allow null selection

while (rs.next()) {

tenkhoaCombo.addItem(rs.getString("tenkhoa"));

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải danh sách khoa: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void updateTenmonCombo(String tenkhoa) {

ArrayList<String> tenmonList = new ArrayList<>();

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

tenmonCombo.setModel(new DefaultComboBoxModel<>(new String[]{"Không thể tải môn học"}));

tenmonCombo.setEnabled(false);

return;

}

try {

String sql;

PreparedStatement stmt;

if (tenkhoa == null || tenkhoa.isEmpty()) {

sql = "SELECT tenmon FROM course WHERE tenkhoa IS NULL ORDER BY tenmon";

stmt = conn.prepareStatement(sql);

} else {

sql = "SELECT tenmon FROM course WHERE tenkhoa = ? ORDER BY tenmon";

stmt = conn.prepareStatement(sql);

stmt.setString(1, tenkhoa);

}

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

tenmonList.add(rs.getString("tenmon"));

}

rs.close();

stmt.close();

if (tenmonList.isEmpty()) {

tenmonCombo.setModel(new DefaultComboBoxModel<>(new String[]{"Không có môn học"}));

tenmonCombo.setEnabled(false);

} else {

tenmonCombo.setModel(new DefaultComboBoxModel<>(tenmonList.toArray(new String[0])));

tenmonCombo.setEnabled(true);

}

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải danh sách môn học: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

tenmonCombo.setModel(new DefaultComboBoxModel<>(new String[]{"Không thể tải môn học"}));

tenmonCombo.setEnabled(false);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void loadTableData() {

tableModel.setRowCount(0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, malop, tenlop, tenmon, tenkhoa, soluongSV, maGV, namhoc, hocky FROM class ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("malop"));

row.add(rs.getString("tenlop"));

row.add(rs.getString("tenmon") != null ? rs.getString("tenmon") : "");

row.add(rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "");

row.add(String.valueOf(rs.getInt("soluongSV")));

row.add(rs.getString("maGV"));

row.add(rs.getString("namhoc"));

row.add(String.valueOf(rs.getInt("hocky")));

tableModel.addRow(row);

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void clearFields() {

malopField.setText("");

tenlopField.setText("");

tenkhoaCombo.setSelectedIndex(0);

tenmonCombo.setModel(new DefaultComboBoxModel<>(new String[]{"Vui lòng chọn khoa trước"}));

tenmonCombo.setEnabled(false);

soluongSVField.setText("");

maGVField.setText("");

namhocField.setText("");

hockyCombo.setSelectedIndex(0);

searchField.setText("");

classTable.clearSelection();

malopField.setEditable(false); // Ensure malopField remains non-editable

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == searchButtonRight) {

String searchText = searchField.getText().trim();

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, malop, tenlop, tenmon, tenkhoa, soluongSV, maGV, namhoc, hocky " +

"FROM class WHERE malop LIKE ? OR tenlop LIKE ? OR maGV LIKE ? ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

String param = "%" + searchText + "%";

stmt.setString(1, param);

stmt.setString(2, param);

stmt.setString(3, param);

ResultSet rs = stmt.executeQuery();

tableModel.setRowCount(0);

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("malop"));

row.add(rs.getString("tenlop"));

row.add(rs.getString("tenmon") != null ? rs.getString("tenmon") : "");

row.add(rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "");

row.add(String.valueOf(rs.getInt("soluongSV")));

row.add(rs.getString("maGV"));

row.add(rs.getString("namhoc"));

row.add(String.valueOf(rs.getInt("hocky")));

tableModel.addRow(row);

}

rs.close();

stmt.close();

clearFields();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tìm kiếm: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == reloadButton) {

clearFields();

loadTableData();

} else if (e.getSource() == editButton) {

int selectedRow = classTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một lớp học để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String id = (String) tableModel.getValueAt(selectedRow, 0);

String malop = malopField.getText().trim();

String tenlop = tenlopField.getText().trim();

String tenmon = (String) tenmonCombo.getSelectedItem();

String tenkhoa = (String) tenkhoaCombo.getSelectedItem();

String soluongSVStr = soluongSVField.getText().trim();

String maGV = maGVField.getText().trim();

String namhoc = namhocField.getText().trim();

String hocky = (String) hockyCombo.getSelectedItem();

// Validation

if (tenlop.isEmpty()) {

JOptionPane.showMessageDialog(this, "Tên lớp không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenlop.length() > 100) {

JOptionPane.showMessageDialog(this, "Tên lớp không được vượt quá 100 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenmon == null || tenmon.isEmpty() || tenmon.equals("Vui lòng chọn khoa trước") || tenmon.equals("Không có môn học") || tenmon.equals("Không thể tải môn học")) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một môn học hợp lệ!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenmon.length() > 100) {

JOptionPane.showMessageDialog(this, "Tên môn không được vượt quá 100 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

int soluongSV = 0;

if (!soluongSVStr.isEmpty()) {

try {

soluongSV = Integer.parseInt(soluongSVStr);

if (soluongSV < 0 || soluongSV > 500) {

JOptionPane.showMessageDialog(this, "Số lượng sinh viên phải từ 0 đến 500!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Số lượng sinh viên phải là số nguyên!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

}

if (!maGV.matches("\\d{6}")) {

JOptionPane.showMessageDialog(this, "Mã giảng viên phải có đúng 6 chữ số!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!namhoc.matches("\\d{4}-\\d{4}")) {

JOptionPane.showMessageDialog(this, "Năm học phải có định dạng YYYY-YYYY (ví dụ: 2024-2025)!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (hocky == null || hocky.isEmpty()) {

JOptionPane.showMessageDialog(this, "Học kỳ không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "UPDATE class SET tenlop = ?, tenmon = ?, tenkhoa = ?, soluongSV = ?, maGV = ?, namhoc = ?, hocky = ? WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, tenlop);

stmt.setString(2, tenmon);

stmt.setString(3, tenkhoa.isEmpty() ? null : tenkhoa);

stmt.setInt(4, soluongSV);

stmt.setString(5, maGV);

stmt.setString(6, namhoc);

stmt.setInt(7, Integer.parseInt(hocky));

stmt.setInt(8, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Chỉnh sửa lớp học thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy lớp học để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi chỉnh sửa lớp học: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

if (ex.getMessage().contains("tenmon")) {

errorMsg = "Tên môn không hợp lệ!";

} else if (ex.getMessage().contains("maGV")) {

errorMsg = "Mã giảng viên không hợp lệ!";

} else if (ex.getMessage().contains("tenkhoa")) {

errorMsg = "Tên khoa không hợp lệ!";

}

} else if (ex.getSQLState().equals("HY000") && ex.getMessage().contains("CHECK")) {

errorMsg = "Dữ liệu không hợp lệ (số lượng SV, năm học, hoặc học kỳ)!";

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == deleteButton) {

int selectedRow = classTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một lớp học để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

int confirm = JOptionPane.showConfirmDialog(this, "Bạn có chắc chắn muốn xóa lớp học này?", "Xác nhận xóa", JOptionPane.YES\_NO\_OPTION);

if (confirm != JOptionPane.YES\_OPTION) return;

String id = (String) tableModel.getValueAt(selectedRow, 0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "DELETE FROM class WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Xóa lớp học thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy lớp học để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi xóa lớp học: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

errorMsg = "Không thể xóa lớp học vì có dữ liệu liên quan!";

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == printButton) {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try (PrintWriter writer = new PrintWriter("classes.csv")) {

writer.println("ID,Mã lớp,Tên lớp,Tên môn,Tên khoa,Số lượng SV,Mã GV,Năm học,Học kỳ");

String sql = "SELECT id, malop, tenlop, tenmon, tenkhoa, soluongSV, maGV, namhoc, hocky FROM class ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

String id = String.valueOf(rs.getInt("id"));

String malop = rs.getString("malop");

String tenlop = rs.getString("tenlop");

String tenmon = rs.getString("tenmon") != null ? rs.getString("tenmon") : "";

String tenkhoa = rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "";

String soluongSV = String.valueOf(rs.getInt("soluongSV"));

String maGV = rs.getString("maGV");

String namhoc = rs.getString("namhoc");

String hocky = String.valueOf(rs.getInt("hocky"));

writer.println(String.format("%s,%s,%s,%s,%s,%s,%s,%s,%s", id, malop, tenlop, tenmon, tenkhoa, soluongSV, maGV, namhoc, hocky));

}

rs.close();

stmt.close();

JOptionPane.showMessageDialog(this, "Dữ liệu đã được xuất ra classes.csv!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

} catch (FileNotFoundException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi xuất file: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi truy vấn dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == cancelButton) {

setVisible(false);

dispose();

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new ClassManagement());

}

}

1. **DepartmentManegement.java**

package university.managerment.system;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.sql.\*;

import java.util.Vector;

public class DepartmentManagement extends JFrame implements ActionListener {

private JTextField makhoaField, tenkhoaField, matruongkhoaField, searchField;

private JButton searchButtonRight, reloadButton, printButton, deleteButton, editButton, cancelButton;

private JTable departmentTable;

private DefaultTableModel tableModel;

private Connect dbConnect;

public DepartmentManagement() {

setSize(1540, 850);

setLocationRelativeTo(null);

setUndecorated(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Main panel

JPanel mainPanel = new JPanel(new BorderLayout());

mainPanel.setBackground(new Color(227, 237, 249));

// Left panel (input)

JPanel leftPanel = new JPanel(null);

leftPanel.setBackground(new Color(255, 255, 255));

leftPanel.setPreferredSize(new Dimension(565, 850));

// Title

JLabel titleLabel = new JLabel("Quản lý khoa", SwingConstants.CENTER);

titleLabel.setBounds(160, 30, 200, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

leftPanel.add(titleLabel);

// Input fields (explicitly declared without loop)

// Mã khoa

JLabel makhoaLabel = new JLabel("Mã khoa:");

makhoaLabel.setBounds(30, 105, 175, 50);

makhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(makhoaLabel);

makhoaField = new JTextField();

makhoaField.setBounds(205, 105, 330, 50);

makhoaField.setFont(new Font("SanSerif", Font.PLAIN, 20));

makhoaField.setBackground(new Color(227, 237, 249));

makhoaField.setEditable(false); // Non-editable

leftPanel.add(makhoaField);

// Tên khoa

JLabel tenkhoaLabel = new JLabel("Tên khoa:");

tenkhoaLabel.setBounds(30, 185, 175, 50);

tenkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(tenkhoaLabel);

tenkhoaField = new JTextField();

tenkhoaField.setBounds(205, 185, 330, 50);

tenkhoaField.setFont(new Font("SanSerif", Font.PLAIN, 20));

tenkhoaField.setBackground(new Color(227, 237, 249));

leftPanel.add(tenkhoaField);

// Mã trưởng khoa

JLabel matruongkhoaLabel = new JLabel("Mã trưởng khoa:");

matruongkhoaLabel.setBounds(30, 265, 175, 50);

matruongkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(matruongkhoaLabel);

matruongkhoaField = new JTextField();

matruongkhoaField.setBounds(205, 265, 330, 50);

matruongkhoaField.setFont(new Font("SanSerif", Font.PLAIN, 20));

matruongkhoaField.setBackground(new Color(227, 237, 249));

leftPanel.add(matruongkhoaField);

// Right panel (table, search field, buttons)

JPanel rightPanel = new JPanel(null);

rightPanel.setBackground(new Color(227, 237, 249));

rightPanel.setPreferredSize(new Dimension(975, 850));

// Search field

searchField = new JTextField();

searchField.setBounds(30, 40, 525, 60);

searchField.setFont(new Font("SanSerif", Font.PLAIN, 20));

rightPanel.add(searchField);

// Search and Reload buttons

searchButtonRight = createButton("Tìm kiếm", new Color(37, 61, 144), 585, 40, 165, 60);

reloadButton = createButton("Tải lại", new Color(37, 61, 144), 780, 40, 165, 60);

rightPanel.add(searchButtonRight);

rightPanel.add(reloadButton);

// Action buttons (explicitly declared without loop)

editButton = createButton("Chỉnh sửa", new Color(37, 61, 144), 30, 760, 198, 60);

printButton = createButton("In", new Color(37, 61, 144), 270, 760, 198, 60);

deleteButton = createButton("Xóa", new Color(37, 61, 144), 509, 760, 198, 60);

cancelButton = createButton("Hủy", new Color(193, 14, 14), 748, 760, 198, 60);

rightPanel.add(editButton);

rightPanel.add(printButton);

rightPanel.add(deleteButton);

rightPanel.add(cancelButton);

// Table

String[] columns = {"ID", "Mã khoa", "Tên khoa", "Mã trưởng khoa"};

tableModel = new DefaultTableModel(columns, 0);

departmentTable = new JTable(tableModel);

departmentTable.setFont(new Font("SanSerif", Font.PLAIN, 16));

departmentTable.getTableHeader().setFont(new Font("Serif", Font.BOLD, 18));

departmentTable.setRowHeight(30);

departmentTable.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

// Table row selection

departmentTable.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

int row = departmentTable.getSelectedRow();

if (row >= 0) {

makhoaField.setText((String) tableModel.getValueAt(row, 1));

makhoaField.setEditable(false); // Ensure makhoaField is non-editable

tenkhoaField.setText((String) tableModel.getValueAt(row, 2));

String matruongkhoa = (String) tableModel.getValueAt(row, 3);

matruongkhoaField.setText(matruongkhoa != null ? matruongkhoa : "");

}

}

});

JScrollPane scrollPane = new JScrollPane(departmentTable);

scrollPane.setBounds(30, 130, 915, 600);

rightPanel.add(scrollPane);

// Add panels to main

mainPanel.add(leftPanel, BorderLayout.WEST);

mainPanel.add(rightPanel, BorderLayout.CENTER);

add(mainPanel);

// Load initial data

loadTableData();

setVisible(true);

}

private JButton createButton(String text, Color bg, int x, int y, int w, int h) {

JButton button = new JButton(text) {

@Override

protected void paintComponent(Graphics g) {

Graphics2D g2 = (Graphics2D) g.create();

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g2.setColor(getBackground());

g2.fillRoundRect(0, 0, getWidth(), getHeight(), 15, 15);

super.paintComponent(g2);

g2.dispose();

}

};

button.setBounds(x, y, w, h);

button.setBackground(bg);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 18));

button.setContentAreaFilled(false);

button.setBorderPainted(false);

button.addActionListener(this);

return button;

}

private void loadTableData() {

tableModel.setRowCount(0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, makhoa, tenkhoa, matruongkhoa FROM department ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("makhoa"));

row.add(rs.getString("tenkhoa"));

row.add(rs.getString("matruongkhoa") != null ? rs.getString("matruongkhoa") : "");

tableModel.addRow(row);

}

rs.close();

stmt.close();

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải dữ liệu: " + e.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

e.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void clearFields() {

makhoaField.setText("");

tenkhoaField.setText("");

matruongkhoaField.setText("");

searchField.setText("");

departmentTable.clearSelection();

makhoaField.setEditable(false); // Ensure makhoaField remains non-editable

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == searchButtonRight) {

String searchText = searchField.getText().trim();

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, makhoa, tenkhoa, matruongkhoa FROM department WHERE makhoa LIKE ? OR tenkhoa LIKE ? OR matruongkhoa LIKE ? ORDER BY makhoa";

PreparedStatement stmt = conn.prepareStatement(sql);

String param = "%" + searchText + "%";

stmt.setString(1, param);

stmt.setString(2, param);

stmt.setString(3, param);

ResultSet rs = stmt.executeQuery();

tableModel.setRowCount(0);

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("makhoa"));

row.add(rs.getString("tenkhoa"));

row.add(rs.getString("matruongkhoa") != null ? rs.getString("matruongkhoa") : "");

tableModel.addRow(row);

}

rs.close();

stmt.close();

clearFields();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tìm kiếm: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == reloadButton) {

clearFields();

loadTableData();

} else if (e.getSource() == editButton) {

int selectedRow = departmentTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một khoa để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String id = (String) tableModel.getValueAt(selectedRow, 0);

String tenkhoa = tenkhoaField.getText().trim();

String matruongkhoa = matruongkhoaField.getText().trim();

// Validation

if (tenkhoa.isEmpty()) {

JOptionPane.showMessageDialog(this, "Tên khoa không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenkhoa.length() > 60) {

JOptionPane.showMessageDialog(this, "Tên khoa không được vượt quá 60 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!matruongkhoa.isEmpty() && !matruongkhoa.matches("\\d{6}")) {

JOptionPane.showMessageDialog(this, "Mã trưởng khoa phải có đúng 6 chữ số hoặc để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "UPDATE department SET tenkhoa = ?, matruongkhoa = ? WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, tenkhoa);

stmt.setString(2, matruongkhoa.isEmpty() ? null : matruongkhoa);

stmt.setInt(3, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Chỉnh sửa khoa thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy khoa để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi chỉnh sửa khoa: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

if (ex.getMessage().contains("tenkhoa")) {

errorMsg = "Tên khoa đã tồn tại!";

} else if (ex.getMessage().contains("matruongkhoa")) {

errorMsg = "Mã trưởng khoa không hợp lệ!";

}

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == deleteButton) {

int selectedRow = departmentTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một khoa để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

int confirm = JOptionPane.showConfirmDialog(this, "Bạn có chắc chắn muốn xóa khoa này?", "Xác nhận xóa", JOptionPane.YES\_NO\_OPTION);

if (confirm != JOptionPane.YES\_OPTION) return;

String id = (String) tableModel.getValueAt(selectedRow, 0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "DELETE FROM department WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Xóa khoa thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy khoa để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi xóa khoa: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

errorMsg = "Không thể xóa khoa vì có dữ liệu liên quan!";

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == printButton) {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try (PrintWriter writer = new PrintWriter("departments.csv")) {

writer.println("ID,Mã khoa,Tên khoa,Mã trưởng khoa");

String sql = "SELECT id, makhoa, tenkhoa, matruongkhoa FROM department ORDER BY makhoa";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

String id = String.valueOf(rs.getInt("id"));

String makhoa = rs.getString("makhoa");

String tenkhoa = rs.getString("tenkhoa");

String matruongkhoa = rs.getString("matruongkhoa") != null ? rs.getString("matruongkhoa") : "";

writer.println(String.format("%s,%s,%s,%s", id, makhoa, tenkhoa, matruongkhoa));

}

rs.close();

stmt.close();

JOptionPane.showMessageDialog(this, "Dữ liệu đã được xuất ra departments.csv!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

} catch (FileNotFoundException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi xuất file: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi truy vấn dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == cancelButton) {

setVisible(false);

dispose();

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new DepartmentManagement());

}

}

1. **CourseManegement.java**

package university.managerment.system;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.sql.\*;

import java.util.Vector;

public class CourseManagement extends JFrame implements ActionListener {

private JTextField mamonField, tenmonField, searchField;

private JComboBox<String> tinchiCombo, tenkhoaCombo;

private JButton searchButtonRight, reloadButton, editButton, printButton, deleteButton, cancelButton;

private JTable courseTable;

private DefaultTableModel tableModel;

private Connect dbConnect;

public CourseManagement() {

setSize(1540, 850);

setLocationRelativeTo(null);

setUndecorated(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Main panel

JPanel mainPanel = new JPanel(new BorderLayout());

mainPanel.setBackground(new Color(227, 237, 249));

// Left panel (input)

JPanel leftPanel = new JPanel(null);

leftPanel.setBackground(new Color(255, 255, 255));

leftPanel.setPreferredSize(new Dimension(565, 850));

// Title

JLabel titleLabel = new JLabel("Quản Lý Học Phần", SwingConstants.CENTER);

titleLabel.setBounds(160, 30, 250, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

leftPanel.add(titleLabel);

// Input fields

// Mã môn (mamon)

JLabel mamonLabel = new JLabel("Mã môn:");

mamonLabel.setBounds(30, 105, 175, 50);

mamonLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(mamonLabel);

mamonField = new JTextField();

mamonField.setBounds(205, 105, 330, 50);

mamonField.setFont(new Font("SanSerif", Font.PLAIN, 20));

mamonField.setBackground(new Color(227, 237, 249));

mamonField.setEditable(false); // Non-editable

leftPanel.add(mamonField);

// Tên môn (tenmon)

JLabel tenmonLabel = new JLabel("Tên môn:");

tenmonLabel.setBounds(30, 185, 175, 50);

tenmonLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(tenmonLabel);

tenmonField = new JTextField();

tenmonField.setBounds(205, 185, 330, 50);

tenmonField.setFont(new Font("SanSerif", Font.PLAIN, 20));

tenmonField.setBackground(new Color(227, 237, 249));

leftPanel.add(tenmonField);

// Số tín chỉ (tinchi)

JLabel tinchiLabel = new JLabel("Số tín chỉ:");

tinchiLabel.setBounds(30, 265, 175, 50);

tinchiLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(tinchiLabel);

String[] tinchiOptions = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10"};

tinchiCombo = new JComboBox<>(tinchiOptions);

tinchiCombo.setBounds(205, 265, 330, 50);

tinchiCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

tinchiCombo.setBackground(new Color(227, 237, 249));

leftPanel.add(tinchiCombo);

// Tên khoa (tenkhoa)

JLabel tenkhoaLabel = new JLabel("Tên khoa:");

tenkhoaLabel.setBounds(30, 345, 175, 50);

tenkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(tenkhoaLabel);

tenkhoaCombo = new JComboBox<>();

tenkhoaCombo.setBounds(205, 345, 330, 50);

tenkhoaCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

tenkhoaCombo.setBackground(new Color(227, 237, 249));

leftPanel.add(tenkhoaCombo);

// Right panel (table, search field, buttons)

JPanel rightPanel = new JPanel(null);

rightPanel.setBackground(new Color(227, 237, 249));

rightPanel.setPreferredSize(new Dimension(975, 850));

// Search field

searchField = new JTextField();

searchField.setBounds(30, 40, 525, 60);

searchField.setFont(new Font("SanSerif", Font.PLAIN, 20));

rightPanel.add(searchField);

// Search and Reload buttons

searchButtonRight = createButton("Tìm kiếm", new Color(37, 61, 144), 585, 40, 165, 60);

reloadButton = createButton("Tải lại", new Color(37, 61, 144), 780, 40, 165, 60);

rightPanel.add(searchButtonRight);

rightPanel.add(reloadButton);

// Action buttons

editButton = createButton("Chỉnh sửa", new Color(37, 61, 144), 30, 760, 198, 60);

printButton = createButton("In", new Color(37, 61, 144), 270, 760, 198, 60);

deleteButton = createButton("Xóa", new Color(37, 61, 144), 509, 760, 198, 60);

cancelButton = createButton("Hủy", new Color(193, 14, 14), 748, 760, 198, 60);

rightPanel.add(editButton);

rightPanel.add(printButton);

rightPanel.add(deleteButton);

rightPanel.add(cancelButton);

// Table

String[] columns = {"ID", "Mã môn", "Tên môn", "Số tín chỉ", "Tên khoa"};

tableModel = new DefaultTableModel(columns, 0);

courseTable = new JTable(tableModel);

courseTable.setFont(new Font("SanSerif", Font.PLAIN, 16));

courseTable.getTableHeader().setFont(new Font("Serif", Font.BOLD, 18));

courseTable.setRowHeight(30);

courseTable.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

// Table row selection

courseTable.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

int row = courseTable.getSelectedRow();

if (row >= 0) {

mamonField.setText((String) tableModel.getValueAt(row, 1));

mamonField.setEditable(false); // Ensure mamonField is non-editable

tenmonField.setText((String) tableModel.getValueAt(row, 2));

tinchiCombo.setSelectedItem(String.valueOf(tableModel.getValueAt(row, 3)));

String tenkhoa = (String) tableModel.getValueAt(row, 4);

tenkhoaCombo.setSelectedItem(tenkhoa != null ? tenkhoa : "");

}

}

});

JScrollPane scrollPane = new JScrollPane(courseTable);

scrollPane.setBounds(30, 130, 915, 600);

rightPanel.add(scrollPane);

// Add panels to main

mainPanel.add(leftPanel, BorderLayout.WEST);

mainPanel.add(rightPanel, BorderLayout.CENTER);

add(mainPanel);

// Populate tenkhoaCombo

populateDepartmentCombo();

// Load initial data

loadTableData();

setVisible(true);

}

private JButton createButton(String text, Color bg, int x, int y, int w, int h) {

JButton button = new JButton(text) {

@Override

protected void paintComponent(Graphics g) {

Graphics2D g2 = (Graphics2D) g.create();

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g2.setColor(getBackground());

g2.fillRoundRect(0, 0, getWidth(), getHeight(), 15, 15);

super.paintComponent(g2);

g2.dispose();

}

};

button.setBounds(x, y, w, h);

button.setBackground(bg);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 18));

button.setContentAreaFilled(false);

button.setBorderPainted(false);

button.addActionListener(this);

return button;

}

private void populateDepartmentCombo() {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT tenkhoa FROM department ORDER BY tenkhoa";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

tenkhoaCombo.removeAllItems();

tenkhoaCombo.addItem(""); // Allow null selection

while (rs.next()) {

tenkhoaCombo.addItem(rs.getString("tenkhoa"));

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải danh sách khoa: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void loadTableData() {

tableModel.setRowCount(0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, mamon, tenmon, tinchi, tenkhoa FROM course ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("mamon"));

row.add(rs.getString("tenmon"));

row.add(String.valueOf(rs.getInt("tinchi")));

row.add(rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "");

tableModel.addRow(row);

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void clearFields() {

mamonField.setText("");

tenmonField.setText("");

tinchiCombo.setSelectedIndex(0);

tenkhoaCombo.setSelectedIndex(0);

searchField.setText("");

courseTable.clearSelection();

mamonField.setEditable(false); // Ensure mamonField remains non-editable

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == searchButtonRight) {

String searchText = searchField.getText().trim();

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, mamon, tenmon, tinchi, tenkhoa " +

"FROM course WHERE mamon LIKE ? OR tenmon LIKE ? ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

String param = "%" + searchText + "%";

stmt.setString(1, param);

stmt.setString(2, param);

ResultSet rs = stmt.executeQuery();

tableModel.setRowCount(0);

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("mamon"));

row.add(rs.getString("tenmon"));

row.add(String.valueOf(rs.getInt("tinchi")));

row.add(rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "");

tableModel.addRow(row);

}

rs.close();

stmt.close();

clearFields();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tìm kiếm: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == reloadButton) {

clearFields();

loadTableData();

} else if (e.getSource() == editButton) {

int selectedRow = courseTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một học phần để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String id = (String) tableModel.getValueAt(selectedRow, 0);

String tenmon = tenmonField.getText().trim();

String tinchi = (String) tinchiCombo.getSelectedItem();

String tenkhoa = (String) tenkhoaCombo.getSelectedItem();

// Validation

if (tenmon.isEmpty()) {

JOptionPane.showMessageDialog(this, "Tên môn không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tenmon.length() > 100) {

JOptionPane.showMessageDialog(this, "Tên môn không được vượt quá 100 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tinchi == null || tinchi.isEmpty()) {

JOptionPane.showMessageDialog(this, "Số tín chỉ không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "UPDATE course SET tenmon = ?, tinchi = ?, tenkhoa = ? WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, tenmon);

stmt.setInt(2, Integer.parseInt(tinchi));

stmt.setString(3, tenkhoa.isEmpty() ? null : tenkhoa);

stmt.setInt(4, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Chỉnh sửa học phần thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy học phần để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi chỉnh sửa học phần: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

if (ex.getMessage().contains("tenmon")) {

errorMsg = "Tên môn đã tồn tại!";

} else if (ex.getMessage().contains("tenkhoa")) {

errorMsg = "Tên khoa không hợp lệ!";

}

} else if (ex.getSQLState().equals("HY000") && ex.getMessage().contains("CHECK")) {

errorMsg = "Số tín chỉ không hợp lệ!";

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == deleteButton) {

int selectedRow = courseTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một học phần để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

int confirm = JOptionPane.showConfirmDialog(this, "Bạn có chắc chắn muốn xóa học phần này?", "Xác nhận xóa", JOptionPane.YES\_NO\_OPTION);

if (confirm != JOptionPane.YES\_OPTION) return;

String id = (String) tableModel.getValueAt(selectedRow, 0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "DELETE FROM course WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Xóa học phần thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy học phần để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi xóa học phần: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

errorMsg = "Không thể xóa học phần vì có dữ liệu liên quan (ví dụ: lớp học)!";

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == printButton) {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try (PrintWriter writer = new PrintWriter("courses.csv")) {

writer.println("ID,Mã môn,Tên môn,Số tín chỉ,Tên khoa");

String sql = "SELECT id, mamon, tenmon, tinchi, tenkhoa FROM course ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

String id = String.valueOf(rs.getInt("id"));

String mamon = rs.getString("mamon");

String tenmon = rs.getString("tenmon");

String tinchi = String.valueOf(rs.getInt("tinchi"));

String tenkhoa = rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "";

writer.println(String.format("%s,%s,%s,%s,%s", id, mamon, tenmon, tinchi, tenkhoa));

}

rs.close();

stmt.close();

JOptionPane.showMessageDialog(this, "Dữ liệu đã được xuất ra courses.csv!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

} catch (FileNotFoundException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi xuất file: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi truy vấn dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == cancelButton) {

setVisible(false);

dispose();

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new CourseManagement());

}

}

1. **LecturerManegement.java**

package university.managerment.system;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Vector;

import com.toedter.calendar.JDateChooser;

public class LecturerManagement extends JFrame implements ActionListener {

private JTextField maGVField, hotenField, sdtField, diachiField, cccdField, searchField;

private JComboBox<String> tenkhoaCombo, gioitinhCombo, hocviCombo, chucvuCombo;

private JDateChooser ngaysinhChooser;

private JButton searchButtonRight, reloadButton, editButton, printButton, deleteButton, cancelButton;

private JTable lecturerTable;

private DefaultTableModel tableModel;

private Connect dbConnect;

private SimpleDateFormat dateFormat;

public LecturerManagement() {

setSize(1540, 850);

setLocationRelativeTo(null);

setUndecorated(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Initialize date format

dateFormat = new SimpleDateFormat("yyyy-MM-dd");

// Main panel

JPanel mainPanel = new JPanel(new BorderLayout());

mainPanel.setBackground(new Color(227, 237, 249));

// Left panel (input) - make it scrollable

JPanel leftPanelContent = new JPanel(null);

leftPanelContent.setBackground(new Color(255, 255, 255));

leftPanelContent.setPreferredSize(new Dimension(565, 900)); // Increased height to accommodate all fields

// Wrap leftPanelContent in a JScrollPane

JScrollPane leftPanelScroll = new JScrollPane(leftPanelContent);

leftPanelScroll.setPreferredSize(new Dimension(565, 850));

leftPanelScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

leftPanelScroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER);

// Title

JLabel titleLabel = new JLabel("Quản lý giảng viên", SwingConstants.CENTER);

titleLabel.setBounds(147, 30, 300, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

leftPanelContent.add(titleLabel);

// Input fields

// Mã GV

JLabel maGVLabel = new JLabel("Mã GV:");

maGVLabel.setBounds(30, 105, 175, 50);

maGVLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(maGVLabel);

maGVField = new JTextField();

maGVField.setBounds(205, 105, 330, 50);

maGVField.setFont(new Font("SanSerif", Font.PLAIN, 20));

maGVField.setBackground(new Color(227, 237, 249));

maGVField.setEditable(false); // Non-editable

leftPanelContent.add(maGVField);

// Tên khoa

JLabel tenkhoaLabel = new JLabel("Tên khoa:");

tenkhoaLabel.setBounds(30, 185, 175, 50);

tenkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(tenkhoaLabel);

tenkhoaCombo = new JComboBox<>();

tenkhoaCombo.setBounds(205, 185, 330, 50);

tenkhoaCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

tenkhoaCombo.setBackground(new Color(227, 237, 249));

leftPanelContent.add(tenkhoaCombo);

// Họ tên

JLabel hotenLabel = new JLabel("Họ tên:");

hotenLabel.setBounds(30, 265, 175, 50);

hotenLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(hotenLabel);

hotenField = new JTextField();

hotenField.setBounds(205, 265, 330, 50);

hotenField.setFont(new Font("SanSerif", Font.PLAIN, 20));

hotenField.setBackground(new Color(227, 237, 249));

leftPanelContent.add(hotenField);

// Giới tính

JLabel gioitinhLabel = new JLabel("Giới tính:");

gioitinhLabel.setBounds(30, 345, 175, 50);

gioitinhLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(gioitinhLabel);

gioitinhCombo = new JComboBox<>(new String[]{"Nam", "Nữ"});

gioitinhCombo.setBounds(205, 345, 330, 50);

gioitinhCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

gioitinhCombo.setBackground(new Color(227, 237, 249));

leftPanelContent.add(gioitinhCombo);

// Ngày sinh

JLabel ngaysinhLabel = new JLabel("Ngày sinh:");

ngaysinhLabel.setBounds(30, 425, 175, 50);

ngaysinhLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(ngaysinhLabel);

ngaysinhChooser = new JDateChooser();

ngaysinhChooser.setBounds(205, 425, 330, 50);

ngaysinhChooser.setFont(new Font("SanSerif", Font.PLAIN, 20));

ngaysinhChooser.setBackground(new Color(227, 237, 249));

ngaysinhChooser.setDateFormatString("yyyy-MM-dd");

leftPanelContent.add(ngaysinhChooser);

// SĐT

JLabel sdtLabel = new JLabel("SĐT:");

sdtLabel.setBounds(30, 505, 175, 50);

sdtLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(sdtLabel);

sdtField = new JTextField();

sdtField.setBounds(205, 505, 330, 50);

sdtField.setFont(new Font("SanSerif", Font.PLAIN, 20));

sdtField.setBackground(new Color(227, 237, 249));

leftPanelContent.add(sdtField);

// Địa chỉ

JLabel diachiLabel = new JLabel("Địa chỉ:");

diachiLabel.setBounds(30, 585, 175, 50);

diachiLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(diachiLabel);

diachiField = new JTextField();

diachiField.setBounds(205, 585, 330, 50);

diachiField.setFont(new Font("SanSerif", Font.PLAIN, 20));

diachiField.setBackground(new Color(227, 237, 249));

leftPanelContent.add(diachiField);

// CCCD

JLabel cccdLabel = new JLabel("CCCD:");

cccdLabel.setBounds(30, 665, 175, 50);

cccdLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(cccdLabel);

cccdField = new JTextField();

cccdField.setBounds(205, 665, 330, 50);

cccdField.setFont(new Font("SanSerif", Font.PLAIN, 20));

cccdField.setBackground(new Color(227, 237, 249));

leftPanelContent.add(cccdField);

// Học vị

JLabel hocviLabel = new JLabel("Học vị:");

hocviLabel.setBounds(30, 745, 175, 50);

hocviLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(hocviLabel);

hocviCombo = new JComboBox<>(new String[]{"Cử nhân", "Thạc sĩ", "Tiến sĩ", "PGS.TS", "GS.TS", "GS.TS.BS", "PGS.TS.BS", "TS.BS"});

hocviCombo.setBounds(205, 745, 330, 50);

hocviCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

hocviCombo.setBackground(new Color(227, 237, 249));

leftPanelContent.add(hocviCombo);

// Chức vụ

JLabel chucvuLabel = new JLabel("Chức vụ:");

chucvuLabel.setBounds(30, 825, 175, 50);

chucvuLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(chucvuLabel);

chucvuCombo = new JComboBox<>(new String[]{"Trưởng khoa", "Giảng viên", "Trợ giảng", "Phó trưởng khoa"});

chucvuCombo.setBounds(205, 825, 330, 50);

chucvuCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

chucvuCombo.setBackground(new Color(227, 237, 249));

leftPanelContent.add(chucvuCombo);

// Right panel (table, search field, buttons)

JPanel rightPanel = new JPanel(null);

rightPanel.setBackground(new Color(227, 237, 249));

rightPanel.setPreferredSize(new Dimension(975, 850));

// Search field

searchField = new JTextField();

searchField.setBounds(30, 40, 525, 60);

searchField.setFont(new Font("SanSerif", Font.PLAIN, 20));

rightPanel.add(searchField);

// Search and Reload buttons

searchButtonRight = createButton("Tìm kiếm", new Color(37, 61, 144), 585, 40, 165, 60);

reloadButton = createButton("Tải lại", new Color(37, 61, 144), 780, 40, 165, 60);

rightPanel.add(searchButtonRight);

rightPanel.add(reloadButton);

// Action buttons

editButton = createButton("Chỉnh sửa", new Color(37, 61, 144), 30, 760, 198, 60);

printButton = createButton("In", new Color(37, 61, 144), 270, 760, 198, 60);

deleteButton = createButton("Xóa", new Color(37, 61, 144), 509, 760, 198, 60);

cancelButton = createButton("Hủy", new Color(193, 14, 14), 748, 760, 198, 60);

rightPanel.add(editButton);

rightPanel.add(printButton);

rightPanel.add(deleteButton);

rightPanel.add(cancelButton);

// Table

String[] columns = {"ID", "Mã GV", "Tên khoa", "Họ tên", "Giới tính", "Ngày sinh", "SĐT", "Địa chỉ", "CCCD", "Học vị", "Chức vụ"};

tableModel = new DefaultTableModel(columns, 0);

lecturerTable = new JTable(tableModel);

lecturerTable.setFont(new Font("SanSerif", Font.PLAIN, 16));

lecturerTable.getTableHeader().setFont(new Font("Serif", Font.BOLD, 18));

lecturerTable.setRowHeight(30);

lecturerTable.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

// Table row selection

lecturerTable.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

int row = lecturerTable.getSelectedRow();

if (row >= 0) {

maGVField.setText((String) tableModel.getValueAt(row, 1));

maGVField.setEditable(false); // Ensure maGVField is non-editable

String tenkhoa = (String) tableModel.getValueAt(row, 2);

tenkhoaCombo.setSelectedItem(tenkhoa != null ? tenkhoa : "");

hotenField.setText((String) tableModel.getValueAt(row, 3));

gioitinhCombo.setSelectedItem((String) tableModel.getValueAt(row, 4));

try {

String dateStr = (String) tableModel.getValueAt(row, 5);

if (dateStr != null && !dateStr.isEmpty()) {

Date date = dateFormat.parse(dateStr);

ngaysinhChooser.setDate(date);

} else {

ngaysinhChooser.setDate(null);

}

} catch (Exception ex) {

ngaysinhChooser.setDate(null);

}

sdtField.setText((String) tableModel.getValueAt(row, 6));

diachiField.setText((String) tableModel.getValueAt(row, 7));

cccdField.setText((String) tableModel.getValueAt(row, 8));

hocviCombo.setSelectedItem((String) tableModel.getValueAt(row, 9));

chucvuCombo.setSelectedItem((String) tableModel.getValueAt(row, 10));

}

}

});

JScrollPane scrollPane = new JScrollPane(lecturerTable);

scrollPane.setBounds(30, 130, 915, 600);

rightPanel.add(scrollPane);

// Add panels to main

mainPanel.add(leftPanelScroll, BorderLayout.WEST);

mainPanel.add(rightPanel, BorderLayout.CENTER);

add(mainPanel);

// Populate tenkhoaCombo

populateDepartmentCombo();

// Load initial data

loadTableData();

setVisible(true);

}

private JButton createButton(String text, Color bg, int x, int y, int w, int h) {

JButton button = new JButton(text) {

@Override

protected void paintComponent(Graphics g) {

Graphics2D g2 = (Graphics2D) g.create();

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g2.setColor(getBackground());

g2.fillRoundRect(0, 0, getWidth(), getHeight(), 15, 15);

super.paintComponent(g2);

g2.dispose();

}

};

button.setBounds(x, y, w, h);

button.setBackground(bg);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 18));

button.setContentAreaFilled(false);

button.setBorderPainted(false);

button.addActionListener(this);

return button;

}

private void populateDepartmentCombo() {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT tenkhoa FROM department ORDER BY tenkhoa";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

tenkhoaCombo.removeAllItems();

tenkhoaCombo.addItem(""); // Allow null selection

while (rs.next()) {

tenkhoaCombo.addItem(rs.getString("tenkhoa"));

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải danh sách khoa: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void loadTableData() {

tableModel.setRowCount(0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, maGV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd, hocvi, chucvu " +

"FROM lecturer ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("maGV"));

row.add(rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "");

row.add(rs.getString("hoten"));

row.add(rs.getString("gioitinh"));

row.add(rs.getString("ngaysinh"));

row.add(rs.getString("sdt") != null ? rs.getString("sdt") : "");

row.add(rs.getString("diachi") != null ? rs.getString("diachi") : "");

row.add(rs.getString("cccd"));

row.add(rs.getString("hocvi"));

row.add(rs.getString("chucvu"));

tableModel.addRow(row);

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void clearFields() {

maGVField.setText("");

tenkhoaCombo.setSelectedIndex(0);

hotenField.setText("");

gioitinhCombo.setSelectedIndex(0);

ngaysinhChooser.setDate(null);

sdtField.setText("");

diachiField.setText("");

cccdField.setText("");

hocviCombo.setSelectedIndex(0);

chucvuCombo.setSelectedIndex(0);

searchField.setText("");

lecturerTable.clearSelection();

maGVField.setEditable(false); // Ensure maGVField remains non-editable

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == searchButtonRight) {

String searchText = searchField.getText().trim();

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, maGV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd, hocvi, chucvu " +

"FROM lecturer WHERE maGV LIKE ? OR hoten LIKE ? OR cccd LIKE ? ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

String param = "%" + searchText + "%";

stmt.setString(1, param);

stmt.setString(2, param);

stmt.setString(3, param);

ResultSet rs = stmt.executeQuery();

tableModel.setRowCount(0);

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("maGV"));

row.add(rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "");

row.add(rs.getString("hoten"));

row.add(rs.getString("gioitinh"));

row.add(rs.getString("ngaysinh"));

row.add(rs.getString("sdt") != null ? rs.getString("sdt") : "");

row.add(rs.getString("diachi") != null ? rs.getString("diachi") : "");

row.add(rs.getString("cccd"));

row.add(rs.getString("hocvi"));

row.add(rs.getString("chucvu"));

tableModel.addRow(row);

}

rs.close();

stmt.close();

clearFields();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tìm kiếm: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == reloadButton) {

clearFields();

loadTableData();

} else if (e.getSource() == editButton) {

int selectedRow = lecturerTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một giảng viên để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String id = (String) tableModel.getValueAt(selectedRow, 0);

String tenkhoa = (String) tenkhoaCombo.getSelectedItem();

String hoten = hotenField.getText().trim();

String gioitinh = (String) gioitinhCombo.getSelectedItem();

Date ngaysinhDate = ngaysinhChooser.getDate();

String sdt = sdtField.getText().trim();

String diachi = diachiField.getText().trim();

String cccd = cccdField.getText().trim();

String hocvi = (String) hocviCombo.getSelectedItem();

String chucvu = (String) chucvuCombo.getSelectedItem();

// Validation

if (hoten.isEmpty()) {

JOptionPane.showMessageDialog(this, "Họ tên không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (hoten.length() > 50) {

JOptionPane.showMessageDialog(this, "Họ tên không được vượt quá 50 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (ngaysinhDate == null) {

JOptionPane.showMessageDialog(this, "Ngày sinh không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String ngaysinh = dateFormat.format(ngaysinhDate);

if (!sdt.isEmpty() && !sdt.matches("\\d{10,15}")) {

JOptionPane.showMessageDialog(this, "SĐT phải có từ 10 đến 15 chữ số hoặc để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (diachi.length() > 100) {

JOptionPane.showMessageDialog(this, "Địa chỉ không được vượt quá 100 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!cccd.matches("\\d{12}")) {

JOptionPane.showMessageDialog(this, "CCCD phải có đúng 12 chữ số!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (hocvi == null || hocvi.isEmpty()) {

JOptionPane.showMessageDialog(this, "Học vị không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (chucvu == null || chucvu.isEmpty()) {

JOptionPane.showMessageDialog(this, "Chức vụ không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "UPDATE lecturer SET tenkhoa = ?, hoten = ?, gioitinh = ?, ngaysinh = ?, sdt = ?, diachi = ?, cccd = ?, hocvi = ?, chucvu = ? WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, tenkhoa.isEmpty() ? null : tenkhoa);

stmt.setString(2, hoten);

stmt.setString(3, gioitinh);

stmt.setString(4, ngaysinh);

stmt.setString(5, sdt.isEmpty() ? null : sdt);

stmt.setString(6, diachi.isEmpty() ? null : diachi);

stmt.setString(7, cccd);

stmt.setString(8, hocvi);

stmt.setString(9, chucvu);

stmt.setInt(10, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Chỉnh sửa giảng viên thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy giảng viên để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi chỉnh sửa giảng viên: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

if (ex.getMessage().contains("cccd")) {

errorMsg = "CCCD đã tồn tại!";

} else if (ex.getMessage().contains("sdt")) {

errorMsg = "SĐT đã tồn tại!";

} else if (ex.getMessage().contains("tenkhoa")) {

errorMsg = "Tên khoa không hợp lệ!";

}

} else if (ex.getSQLState().equals("HY000") && ex.getMessage().contains("CHECK")) {

errorMsg = "Dữ liệu không hợp lệ (học vị hoặc chức vụ)!";

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == deleteButton) {

int selectedRow = lecturerTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một giảng viên để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

int confirm = JOptionPane.showConfirmDialog(this, "Bạn có chắc chắn muốn xóa giảng viên này?", "Xác nhận xóa", JOptionPane.YES\_NO\_OPTION);

if (confirm != JOptionPane.YES\_OPTION) return;

String id = (String) tableModel.getValueAt(selectedRow, 0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "DELETE FROM lecturer WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Xóa giảng viên thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy giảng viên để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi xóa giảng viên: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

errorMsg = "Không thể xóa giảng viên vì có dữ liệu liên quan (ví dụ: lớp học)!";

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == printButton) {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try (PrintWriter writer = new PrintWriter("lecturers.csv")) {

writer.println("ID,Mã GV,Tên khoa,Họ tên,Giới tính,Ngày sinh,SĐT,Địa chỉ,CCCD,Học vị,Chức vụ");

String sql = "SELECT id, maGV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd, hocvi, chucvu " +

"FROM lecturer ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

String id = String.valueOf(rs.getInt("id"));

String maGV = rs.getString("maGV");

String tenkhoa = rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "";

String hoten = rs.getString("hoten");

String gioitinh = rs.getString("gioitinh");

String ngaysinh = rs.getString("ngaysinh");

String sdt = rs.getString("sdt") != null ? rs.getString("sdt") : "";

String diachi = rs.getString("diachi") != null ? rs.getString("diachi") : "";

String cccd = rs.getString("cccd");

String hocvi = rs.getString("hocvi");

String chucvu = rs.getString("chucvu");

writer.println(String.format("%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s",

id, maGV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd, hocvi, chucvu));

}

rs.close();

stmt.close();

JOptionPane.showMessageDialog(this, "Dữ liệu đã được xuất ra lecturers.csv!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

} catch (FileNotFoundException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi xuất file: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi truy vấn dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == cancelButton) {

setVisible(false);

dispose();

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new LecturerManagement());

}

}

1. **NewsManegement.java**

package university.managerment.system;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Vector;

public class NewsManagement extends JFrame implements ActionListener {

private JTextField matinField, tieudeField, nguoitaoField, ngaytaoField, tailieuField, searchField;

private JTextArea noidungArea;

private JButton searchButtonRight, reloadButton, editButton, printButton, deleteButton, cancelButton, uploadButton;

private JTable newsTable;

private DefaultTableModel tableModel;

private Connect dbConnect;

private SimpleDateFormat dateFormat;

public NewsManagement() {

setSize(1540, 850);

setLocationRelativeTo(null);

setUndecorated(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Initialize date format and database connection

dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

dbConnect = new Connect();

// Main panel

JPanel mainPanel = new JPanel(new BorderLayout());

mainPanel.setBackground(new Color(227, 237, 249));

// Left panel (input) - make it scrollable

JPanel leftPanelContent = new JPanel(null);

leftPanelContent.setBackground(new Color(255, 255, 255));

leftPanelContent.setPreferredSize(new Dimension(565, 900)); // Increased height for all fields

// Wrap leftPanelContent in a JScrollPane

JScrollPane leftPanelScroll = new JScrollPane(leftPanelContent);

leftPanelScroll.setPreferredSize(new Dimension(565, 850));

leftPanelScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);

leftPanelScroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER);

// Title

JLabel titleLabel = new JLabel("Quản lý tin tức", SwingConstants.CENTER);

titleLabel.setBounds(147, 30, 300, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

leftPanelContent.add(titleLabel);

// Input fields

// Mã tin

JLabel matinLabel = new JLabel("Mã tin:");

matinLabel.setBounds(30, 105, 175, 50);

matinLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(matinLabel);

matinField = new JTextField();

matinField.setBounds(205, 105, 330, 50);

matinField.setFont(new Font("SanSerif", Font.PLAIN, 20));

matinField.setBackground(new Color(227, 237, 249));

matinField.setEditable(false); // Non-editable

leftPanelContent.add(matinField);

// Tiêu đề

JLabel tieudeLabel = new JLabel("Tiêu đề:");

tieudeLabel.setBounds(30, 185, 175, 50);

tieudeLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(tieudeLabel);

tieudeField = new JTextField();

tieudeField.setBounds(205, 185, 330, 50);

tieudeField.setFont(new Font("SanSerif", Font.PLAIN, 20));

tieudeField.setBackground(new Color(227, 237, 249));

leftPanelContent.add(tieudeField);

// Nội dung

JLabel noidungLabel = new JLabel("Nội dung:");

noidungLabel.setBounds(30, 265, 175, 50);

noidungLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(noidungLabel);

noidungArea = new JTextArea();

noidungArea.setFont(new Font("SanSerif", Font.PLAIN, 20));

noidungArea.setLineWrap(true);

noidungArea.setWrapStyleWord(true);

JScrollPane noidungScroll = new JScrollPane(noidungArea);

noidungScroll.setBounds(205, 265, 330, 150);

leftPanelContent.add(noidungScroll);

// Tài liệu

JLabel tailieuLabel = new JLabel("Tài liệu:");

tailieuLabel.setBounds(30, 425, 175, 50);

tailieuLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(tailieuLabel);

tailieuField = new JTextField();

tailieuField.setBounds(205, 425, 250, 50);

tailieuField.setFont(new Font("SanSerif", Font.PLAIN, 20));

tailieuField.setBackground(new Color(227, 237, 249));

tailieuField.setEditable(false); // Non-editable

leftPanelContent.add(tailieuField);

uploadButton = new JButton("Chọn file");

uploadButton.setBounds(465, 425, 70, 50);

uploadButton.setBackground(new Color(37, 61, 144));

uploadButton.setForeground(Color.WHITE);

uploadButton.setFont(new Font("Arial", Font.BOLD, 14));

uploadButton.addActionListener(this);

leftPanelContent.add(uploadButton);

// Người tạo

JLabel nguoitaoLabel = new JLabel("Người tạo:");

nguoitaoLabel.setBounds(30, 505, 175, 50);

nguoitaoLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(nguoitaoLabel);

nguoitaoField = new JTextField();

nguoitaoField.setBounds(205, 505, 330, 50);

nguoitaoField.setFont(new Font("SanSerif", Font.PLAIN, 20));

nguoitaoField.setBackground(new Color(227, 237, 249));

leftPanelContent.add(nguoitaoField);

// Ngày tạo

JLabel ngaytaoLabel = new JLabel("Ngày tạo:");

ngaytaoLabel.setBounds(30, 585, 175, 50);

ngaytaoLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanelContent.add(ngaytaoLabel);

ngaytaoField = new JTextField();

ngaytaoField.setBounds(205, 585, 330, 50);

ngaytaoField.setFont(new Font("SanSerif", Font.PLAIN, 20));

ngaytaoField.setBackground(new Color(227, 237, 249));

ngaytaoField.setEditable(false); // Non-editable

leftPanelContent.add(ngaytaoField);

// Right panel (table, search field, buttons)

JPanel rightPanel = new JPanel(null);

rightPanel.setBackground(new Color(227, 237, 249));

rightPanel.setPreferredSize(new Dimension(975, 850));

// Search field

searchField = new JTextField();

searchField.setBounds(30, 40, 525, 60);

searchField.setFont(new Font("SanSerif", Font.PLAIN, 20));

rightPanel.add(searchField);

// Search and Reload buttons

searchButtonRight = createButton("Tìm kiếm", new Color(37, 61, 144), 585, 40, 165, 60);

reloadButton = createButton("Tải lại", new Color(37, 61, 144), 780, 40, 165, 60);

rightPanel.add(searchButtonRight);

rightPanel.add(reloadButton);

// Action buttons

editButton = createButton("Chỉnh sửa", new Color(37, 61, 144), 30, 760, 198, 60);

printButton = createButton("In", new Color(37, 61, 144), 270, 760, 198, 60);

deleteButton = createButton("Xóa", new Color(37, 61, 144), 509, 760, 198, 60);

cancelButton = createButton("Hủy", new Color(193, 14, 14), 748, 760, 198, 60);

rightPanel.add(editButton);

rightPanel.add(printButton);

rightPanel.add(deleteButton);

rightPanel.add(cancelButton);

// Table

String[] columns = {"ID", "Mã tin", "Tiêu đề", "Nội dung", "Tài liệu", "Ngày tạo", "Người tạo"};

tableModel = new DefaultTableModel(columns, 0);

newsTable = new JTable(tableModel);

newsTable.setFont(new Font("SanSerif", Font.PLAIN, 16));

newsTable.getTableHeader().setFont(new Font("Serif", Font.BOLD, 18));

newsTable.setRowHeight(30);

newsTable.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

// Table row selection

newsTable.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

int row = newsTable.getSelectedRow();

if (row >= 0) {

matinField.setText((String) tableModel.getValueAt(row, 1));

matinField.setEditable(false); // Ensure non-editable

tieudeField.setText((String) tableModel.getValueAt(row, 2));

noidungArea.setText((String) tableModel.getValueAt(row, 3));

tailieuField.setText((String) tableModel.getValueAt(row, 4));

ngaytaoField.setText((String) tableModel.getValueAt(row, 5));

nguoitaoField.setText((String) tableModel.getValueAt(row, 6));

}

}

});

JScrollPane scrollPane = new JScrollPane(newsTable);

scrollPane.setBounds(30, 130, 915, 600);

rightPanel.add(scrollPane);

// Add panels to main

mainPanel.add(leftPanelScroll, BorderLayout.WEST);

mainPanel.add(rightPanel, BorderLayout.CENTER);

add(mainPanel);

// Load initial data

loadTableData();

setVisible(true);

}

private JButton createButton(String text, Color bg, int x, int y, int w, int h) {

JButton button = new JButton(text) {

@Override

protected void paintComponent(Graphics g) {

Graphics2D g2 = (Graphics2D) g.create();

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g2.setColor(getBackground());

g2.fillRoundRect(0, 0, getWidth(), getHeight(), 15, 15);

super.paintComponent(g2);

g2.dispose();

}

};

button.setBounds(x, y, w, h);

button.setBackground(bg);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 18));

button.setContentAreaFilled(false);

button.setBorderPainted(false);

button.addActionListener(this);

return button;

}

private void loadTableData() {

tableModel.setRowCount(0);

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, matin, tieude, noidung, tailieu\_url, ngaytao, nguoitao FROM news ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("matin"));

row.add(rs.getString("tieude"));

row.add(rs.getString("noidung"));

row.add(rs.getString("tailieu\_url") != null ? rs.getString("tailieu\_url") : "");

row.add(rs.getString("ngaytao"));

row.add(rs.getString("nguoitao") != null ? rs.getString("nguoitao") : "");

tableModel.addRow(row);

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void clearFields() {

matinField.setText("");

tieudeField.setText("");

noidungArea.setText("");

tailieuField.setText("");

nguoitaoField.setText("");

ngaytaoField.setText("");

searchField.setText("");

newsTable.clearSelection();

matinField.setEditable(false); // Ensure non-editable

}

private void exportToCSV(File file) {

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try (PrintWriter writer = new PrintWriter(file)) {

writer.println("ID,Mã tin,Tiêu đề,Nội dung,Tài liệu,Ngày tạo,Người tạo");

String sql = "SELECT id, matin, tieude, noidung, tailieu\_url, ngaytao, nguoitao FROM news ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

String id = String.valueOf(rs.getInt("id"));

String matin = rs.getString("matin");

String tieude = rs.getString("tieude");

String noidung = rs.getString("noidung");

String tailieu = rs.getString("tailieu\_url") != null ? rs.getString("tailieu\_url") : "";

String ngaytao = rs.getString("ngaytao");

String nguoitao = rs.getString("nguoitao") != null ? rs.getString("nguoitao") : "";

writer.println(String.format("%s,%s,%s,%s,%s,%s,%s", id, matin, tieude, noidung, tailieu, ngaytao, nguoitao));

}

rs.close();

stmt.close();

JOptionPane.showMessageDialog(this, "Dữ liệu đã được xuất ra " + file.getAbsolutePath(), "Thành công", JOptionPane.INFORMATION\_MESSAGE);

} catch (FileNotFoundException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi xuất file: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi truy vấn dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == uploadButton) {

JFileChooser fileChooser = new JFileChooser();

int result = fileChooser.showOpenDialog(this);

if (result == JFileChooser.APPROVE\_OPTION) {

File selectedFile = fileChooser.getSelectedFile();

tailieuField.setText(selectedFile.getAbsolutePath());

}

} else if (e.getSource() == searchButtonRight) {

String searchText = searchField.getText().trim();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, matin, tieude, noidung, tailieu\_url, ngaytao, nguoitao " +

"FROM news WHERE matin LIKE ? OR tieude LIKE ? OR nguoitao LIKE ? ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

String param = "%" + searchText + "%";

stmt.setString(1, param);

stmt.setString(2, param);

stmt.setString(3, param);

ResultSet rs = stmt.executeQuery();

tableModel.setRowCount(0);

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("matin"));

row.add(rs.getString("tieude"));

row.add(rs.getString("noidung"));

row.add(rs.getString("tailieu\_url") != null ? rs.getString("tailieu\_url") : "");

row.add(rs.getString("ngaytao"));

row.add(rs.getString("nguoitao") != null ? rs.getString("nguoitao") : "");

tableModel.addRow(row);

}

rs.close();

stmt.close();

clearFields();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tìm kiếm: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == reloadButton) {

clearFields();

loadTableData();

} else if (e.getSource() == editButton) {

int selectedRow = newsTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một tin tức để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String id = (String) tableModel.getValueAt(selectedRow, 0);

String tieude = tieudeField.getText().trim();

String noidung = noidungArea.getText().trim();

String tailieu = tailieuField.getText().trim();

String nguoitao = nguoitaoField.getText().trim();

// Validation

if (tieude.isEmpty()) {

JOptionPane.showMessageDialog(this, "Tiêu đề không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tieude.length() > 200) {

JOptionPane.showMessageDialog(this, "Tiêu đề không được vượt quá 200 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (noidung.isEmpty()) {

JOptionPane.showMessageDialog(this, "Nội dung không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!tailieu.isEmpty()) {

File file = new File(tailieu);

if (!file.exists()) {

JOptionPane.showMessageDialog(this, "Tệp tài liệu không tồn tại!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (tailieu.length() > 255) {

JOptionPane.showMessageDialog(this, "Đường dẫn tài liệu không được vượt quá 255 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

}

if (nguoitao.length() > 100) {

JOptionPane.showMessageDialog(this, "Tên người tạo không được vượt quá 100 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "UPDATE news SET tieude = ?, noidung = ?, tailieu\_url = ?, nguoitao = ? WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, tieude);

stmt.setString(2, noidung);

stmt.setString(3, tailieu.isEmpty() ? null : tailieu);

stmt.setString(4, nguoitao.isEmpty() ? null : nguoitao);

stmt.setInt(5, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Chỉnh sửa tin tức thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy tin tức để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi chỉnh sửa tin tức: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == deleteButton) {

int selectedRow = newsTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một tin tức để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

int confirm = JOptionPane.showConfirmDialog(this, "Bạn có chắc chắn muốn xóa tin tức này?", "Xác nhận xóa", JOptionPane.YES\_NO\_OPTION);

if (confirm != JOptionPane.YES\_OPTION) return;

String id = (String) tableModel.getValueAt(selectedRow, 0);

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "DELETE FROM news WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Xóa tin tức thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy tin tức để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi xóa tin tức: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == printButton) {

JFileChooser fileChooser = new JFileChooser();

fileChooser.setDialogTitle("Lưu file CSV");

FileNameExtensionFilter filter = new FileNameExtensionFilter("CSV Files (\*.csv)", "csv");

fileChooser.setFileFilter(filter);

fileChooser.setSelectedFile(new File("news.csv"));

int result = fileChooser.showSaveDialog(this);

if (result == JFileChooser.APPROVE\_OPTION) {

File selectedFile = fileChooser.getSelectedFile();

String filePath = selectedFile.getAbsolutePath();

if (!filePath.toLowerCase().endsWith(".csv")) {

selectedFile = new File(filePath + ".csv");

}

exportToCSV(selectedFile);

}

} else if (e.getSource() == cancelButton) {

setVisible(false);

dispose();

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new NewsManagement());

}

}

1. **StudentManegement.java**

package university.managerment.system;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Vector;

import com.toedter.calendar.JDateChooser;

public class StudentManagement extends JFrame implements ActionListener {

private JTextField maSVField, hotenField, sdtField, diachiField, cccdField, searchField;

private JComboBox<String> tenkhoaCombo, gioitinhCombo;

private JDateChooser ngaysinhChooser;

private JButton searchButtonRight, reloadButton, editButton, printButton, deleteButton, cancelButton;

private JTable studentTable;

private DefaultTableModel tableModel;

private Connect dbConnect;

private SimpleDateFormat dateFormat;

public StudentManagement() {

setSize(1540, 850);

setLocationRelativeTo(null);

setUndecorated(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Initialize date format

dateFormat = new SimpleDateFormat("yyyy-MM-dd");

// Main panel

JPanel mainPanel = new JPanel(new BorderLayout());

mainPanel.setBackground(new Color(227, 237, 249));

// Left panel (input)

JPanel leftPanel = new JPanel(null);

leftPanel.setBackground(new Color(255, 255, 255));

leftPanel.setPreferredSize(new Dimension(565, 850));

// Title

JLabel titleLabel = new JLabel("Quản lý sinh viên", SwingConstants.CENTER);

titleLabel.setBounds(147, 30, 300, 40);

titleLabel.setFont(new Font("Serif", Font.BOLD, 30));

titleLabel.setForeground(Color.BLACK);

leftPanel.add(titleLabel);

// Input fields (explicitly declared without loop)

// Mã SV

JLabel maSVLabel = new JLabel("Mã SV:");

maSVLabel.setBounds(30, 105, 175, 50);

maSVLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(maSVLabel);

maSVField = new JTextField();

maSVField.setBounds(205, 105, 330, 50);

maSVField.setFont(new Font("SanSerif", Font.PLAIN, 20));

maSVField.setBackground(new Color(227, 237, 249));

maSVField.setEditable(false); // Non-editable

leftPanel.add(maSVField);

// Tên khoa

JLabel tenkhoaLabel = new JLabel("Tên khoa:");

tenkhoaLabel.setBounds(30, 185, 175, 50);

tenkhoaLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(tenkhoaLabel);

tenkhoaCombo = new JComboBox<>();

tenkhoaCombo.setBounds(205, 185, 330, 50);

tenkhoaCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

tenkhoaCombo.setBackground(new Color(227, 237, 249));

leftPanel.add(tenkhoaCombo);

// Họ tên

JLabel hotenLabel = new JLabel("Họ tên:");

hotenLabel.setBounds(30, 265, 175, 50);

hotenLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(hotenLabel);

hotenField = new JTextField();

hotenField.setBounds(205, 265, 330, 50);

hotenField.setFont(new Font("SanSerif", Font.PLAIN, 20));

hotenField.setBackground(new Color(227, 237, 249));

leftPanel.add(hotenField);

// Giới tính

JLabel gioitinhLabel = new JLabel("Giới tính:");

gioitinhLabel.setBounds(30, 345, 175, 50);

gioitinhLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(gioitinhLabel);

gioitinhCombo = new JComboBox<>(new String[]{"Nam", "Nữ"});

gioitinhCombo.setBounds(205, 345, 330, 50);

gioitinhCombo.setFont(new Font("SanSerif", Font.PLAIN, 20));

gioitinhCombo.setBackground(new Color(227, 237, 249));

leftPanel.add(gioitinhCombo);

// Ngày sinh

JLabel ngaysinhLabel = new JLabel("Ngày sinh:");

ngaysinhLabel.setBounds(30, 425, 175, 50);

ngaysinhLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(ngaysinhLabel);

ngaysinhChooser = new JDateChooser();

ngaysinhChooser.setBounds(205, 425, 330, 50);

ngaysinhChooser.setFont(new Font("SanSerif", Font.PLAIN, 20));

ngaysinhChooser.setBackground(new Color(227, 237, 249));

ngaysinhChooser.setDateFormatString("yyyy-MM-dd");

leftPanel.add(ngaysinhChooser);

// SĐT

JLabel sdtLabel = new JLabel("SĐT:");

sdtLabel.setBounds(30, 505, 175, 50);

sdtLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(sdtLabel);

sdtField = new JTextField();

sdtField.setBounds(205, 505, 330, 50);

sdtField.setFont(new Font("SanSerif", Font.PLAIN, 20));

sdtField.setBackground(new Color(227, 237, 249));

leftPanel.add(sdtField);

// Địa chỉ

JLabel diachiLabel = new JLabel("Địa chỉ:");

diachiLabel.setBounds(30, 585, 175, 50);

diachiLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(diachiLabel);

diachiField = new JTextField();

diachiField.setBounds(205, 585, 330, 50);

diachiField.setFont(new Font("SanSerif", Font.PLAIN, 20));

diachiField.setBackground(new Color(227, 237, 249));

leftPanel.add(diachiField);

// CCCD

JLabel cccdLabel = new JLabel("CCCD:");

cccdLabel.setBounds(30, 665, 175, 50);

cccdLabel.setFont(new Font("Serif", Font.BOLD, 20));

leftPanel.add(cccdLabel);

cccdField = new JTextField();

cccdField.setBounds(205, 665, 330, 50);

cccdField.setFont(new Font("SanSerif", Font.PLAIN, 20));

cccdField.setBackground(new Color(227, 237, 249));

leftPanel.add(cccdField);

// Right panel (table, search field, buttons)

JPanel rightPanel = new JPanel(null);

rightPanel.setBackground(new Color(227, 237, 249));

rightPanel.setPreferredSize(new Dimension(975, 850));

// Search field

searchField = new JTextField();

searchField.setBounds(30, 40, 525, 60);

searchField.setFont(new Font("SanSerif", Font.PLAIN, 20));

rightPanel.add(searchField);

// Search and Reload buttons

searchButtonRight = createButton("Tìm kiếm", new Color(37, 61, 144), 585, 40, 165, 60);

reloadButton = createButton("Tải lại", new Color(37, 61, 144), 780, 40, 165, 60);

rightPanel.add(searchButtonRight);

rightPanel.add(reloadButton);

// Action buttons (explicitly declared without loop)

editButton = createButton("Chỉnh sửa", new Color(37, 61, 144), 30, 760, 198, 60);

printButton = createButton("In", new Color(37, 61, 144), 270, 760, 198, 60);

deleteButton = createButton("Xóa", new Color(37, 61, 144), 509, 760, 198, 60);

cancelButton = createButton("Hủy", new Color(193, 14, 14), 748, 760, 198, 60);

rightPanel.add(editButton);

rightPanel.add(printButton);

rightPanel.add(deleteButton);

rightPanel.add(cancelButton);

// Table

String[] columns = {"ID", "Mã SV", "Tên khoa", "Họ tên", "Giới tính", "Ngày sinh", "SĐT", "Địa chỉ", "CCCD"};

tableModel = new DefaultTableModel(columns, 0);

studentTable = new JTable(tableModel);

studentTable.setFont(new Font("SanSerif", Font.PLAIN, 16));

studentTable.getTableHeader().setFont(new Font("Serif", Font.BOLD, 18));

studentTable.setRowHeight(30);

studentTable.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

// Table row selection

studentTable.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

int row = studentTable.getSelectedRow();

if (row >= 0) {

maSVField.setText((String) tableModel.getValueAt(row, 1));

maSVField.setEditable(false); // Ensure maSVField is non-editable

tenkhoaCombo.setSelectedItem((String) tableModel.getValueAt(row, 2));

hotenField.setText((String) tableModel.getValueAt(row, 3));

gioitinhCombo.setSelectedItem((String) tableModel.getValueAt(row, 4));

try {

String dateStr = (String) tableModel.getValueAt(row, 5);

if (dateStr != null && !dateStr.isEmpty()) {

Date date = dateFormat.parse(dateStr);

ngaysinhChooser.setDate(date);

} else {

ngaysinhChooser.setDate(null);

}

} catch (Exception ex) {

ngaysinhChooser.setDate(null);

}

sdtField.setText((String) tableModel.getValueAt(row, 6));

diachiField.setText((String) tableModel.getValueAt(row, 7));

cccdField.setText((String) tableModel.getValueAt(row, 8));

}

}

});

JScrollPane scrollPane = new JScrollPane(studentTable);

scrollPane.setBounds(30, 130, 915, 600);

rightPanel.add(scrollPane);

// Add panels to main

mainPanel.add(leftPanel, BorderLayout.WEST);

mainPanel.add(rightPanel, BorderLayout.CENTER);

add(mainPanel);

// Initialize database connection

dbConnect = new Connect();

// Populate tenkhoaCombo

populateDepartmentCombo();

// Load initial data

loadTableData();

setVisible(true);

}

private JButton createButton(String text, Color bg, int x, int y, int w, int h) {

JButton button = new JButton(text) {

@Override

protected void paintComponent(Graphics g) {

Graphics2D g2 = (Graphics2D) g.create();

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g2.setColor(getBackground());

g2.fillRoundRect(0, 0, getWidth(), getHeight(), 15, 15);

super.paintComponent(g2);

g2.dispose();

}

};

button.setBounds(x, y, w, h);

button.setBackground(bg);

button.setForeground(Color.WHITE);

button.setFont(new Font("Arial", Font.BOLD, 18));

button.setContentAreaFilled(false);

button.setBorderPainted(false);

button.addActionListener(this);

return button;

}

private void populateDepartmentCombo() {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT tenkhoa FROM department ORDER BY tenkhoa";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

tenkhoaCombo.removeAllItems();

tenkhoaCombo.addItem(""); // Allow null selection

while (rs.next()) {

tenkhoaCombo.addItem(rs.getString("tenkhoa"));

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải danh sách khoa: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void loadTableData() {

tableModel.setRowCount(0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, maSV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd FROM student ORDER BY id DESC";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("maSV"));

row.add(rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "");

row.add(rs.getString("hoten"));

row.add(rs.getString("gioitinh"));

row.add(rs.getString("ngaysinh"));

row.add(rs.getString("sdt") != null ? rs.getString("sdt") : "");

row.add(rs.getString("diachi") != null ? rs.getString("diachi") : "");

row.add(rs.getString("cccd"));

tableModel.addRow(row);

}

rs.close();

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tải dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

}

private void clearFields() {

maSVField.setText("");

tenkhoaCombo.setSelectedIndex(0);

hotenField.setText("");

gioitinhCombo.setSelectedIndex(0);

ngaysinhChooser.setDate(null);

sdtField.setText("");

diachiField.setText("");

cccdField.setText("");

searchField.setText("");

studentTable.clearSelection();

maSVField.setEditable(false); // Ensure maSVField remains non-editable

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == searchButtonRight) {

String searchText = searchField.getText().trim();

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "SELECT id, maSV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd " +

"FROM student WHERE maSV LIKE ? OR hoten LIKE ? OR cccd LIKE ? ORDER BY maSV";

PreparedStatement stmt = conn.prepareStatement(sql);

String param = "%" + searchText + "%";

stmt.setString(1, param);

stmt.setString(2, param);

stmt.setString(3, param);

ResultSet rs = stmt.executeQuery();

tableModel.setRowCount(0);

while (rs.next()) {

Vector<String> row = new Vector<>();

row.add(String.valueOf(rs.getInt("id")));

row.add(rs.getString("maSV"));

row.add(rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "");

row.add(rs.getString("hoten"));

row.add(rs.getString("gioitinh"));

row.add(rs.getString("ngaysinh"));

row.add(rs.getString("sdt") != null ? rs.getString("sdt") : "");

row.add(rs.getString("diachi") != null ? rs.getString("diachi") : "");

row.add(rs.getString("cccd"));

tableModel.addRow(row);

}

rs.close();

stmt.close();

clearFields();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi tìm kiếm: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == reloadButton) {

clearFields();

loadTableData();

} else if (e.getSource() == editButton) {

int selectedRow = studentTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một sinh viên để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String id = (String) tableModel.getValueAt(selectedRow, 0);

String tenkhoa = (String) tenkhoaCombo.getSelectedItem();

String hoten = hotenField.getText().trim();

String gioitinh = (String) gioitinhCombo.getSelectedItem();

Date ngaysinhDate = ngaysinhChooser.getDate();

String sdt = sdtField.getText().trim();

String diachi = diachiField.getText().trim();

String cccd = cccdField.getText().trim();

// Validation

if (hoten.isEmpty()) {

JOptionPane.showMessageDialog(this, "Họ tên không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (hoten.length() > 50) {

JOptionPane.showMessageDialog(this, "Họ tên không được vượt quá 50 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (ngaysinhDate == null) {

JOptionPane.showMessageDialog(this, "Ngày sinh không được để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

String ngaysinh = dateFormat.format(ngaysinhDate);

if (!sdt.isEmpty() && !sdt.matches("\\d{10,15}")) {

JOptionPane.showMessageDialog(this, "SĐT phải có từ 10 đến 15 chữ số hoặc để trống!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (diachi.length() > 100) {

JOptionPane.showMessageDialog(this, "Địa chỉ không được vượt quá 100 ký tự!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!cccd.matches("\\d{12}")) {

JOptionPane.showMessageDialog(this, "CCCD phải có đúng 12 chữ số!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "UPDATE student SET tenkhoa = ?, hoten = ?, gioitinh = ?, ngaysinh = ?, sdt = ?, diachi = ?, cccd = ? WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, tenkhoa.isEmpty() ? null : tenkhoa);

stmt.setString(2, hoten);

stmt.setString(3, gioitinh);

stmt.setString(4, ngaysinh);

stmt.setString(5, sdt.isEmpty() ? null : sdt);

stmt.setString(6, diachi.isEmpty() ? null : diachi);

stmt.setString(7, cccd);

stmt.setInt(8, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Chỉnh sửa sinh viên thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy sinh viên để chỉnh sửa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

String errorMsg = "Lỗi khi chỉnh sửa sinh viên: " + ex.getMessage();

if (ex.getSQLState().equals("23000")) {

if (ex.getMessage().contains("cccd")) {

errorMsg = "CCCD đã tồn tại!";

} else if (ex.getMessage().contains("sdt")) {

errorMsg = "SĐT đã tồn tại!";

} else if (ex.getMessage().contains("tenkhoa")) {

errorMsg = "Tên khoa không hợp lệ!";

}

}

JOptionPane.showMessageDialog(this, errorMsg, "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == deleteButton) {

int selectedRow = studentTable.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Vui lòng chọn một sinh viên để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

int confirm = JOptionPane.showConfirmDialog(this, "Bạn có chắc chắn muốn xóa sinh viên này?", "Xác nhận xóa", JOptionPane.YES\_NO\_OPTION);

if (confirm != JOptionPane.YES\_OPTION) return;

String id = (String) tableModel.getValueAt(selectedRow, 0);

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

String sql = "DELETE FROM student WHERE id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, Integer.parseInt(id));

int rowsAffected = stmt.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(this, "Xóa sinh viên thành công!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

loadTableData();

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Không tìm thấy sinh viên để xóa!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

}

stmt.close();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi xóa sinh viên: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == printButton) {

Connect dbConnect = new Connect();

Connection conn = dbConnect.getConnection();

if (conn == null) {

JOptionPane.showMessageDialog(this, "Không thể kết nối tới cơ sở dữ liệu!", "Lỗi", JOptionPane.ERROR\_MESSAGE);

return;

}

try (PrintWriter writer = new PrintWriter("students.csv")) {

writer.println("ID,Mã SV,Tên khoa,Họ tên,Giới tính,Ngày sinh,SĐT,Địa chỉ,CCCD");

String sql = "SELECT id, maSV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd FROM student ORDER BY maSV";

PreparedStatement stmt = conn.prepareStatement(sql);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

String id = String.valueOf(rs.getInt("id"));

String maSV = rs.getString("maSV");

String tenkhoa = rs.getString("tenkhoa") != null ? rs.getString("tenkhoa") : "";

String hoten = rs.getString("hoten");

String gioitinh = rs.getString("gioitinh");

String ngaysinh = rs.getString("ngaysinh");

String sdt = rs.getString("sdt") != null ? rs.getString("sdt") : "";

String diachi = rs.getString("diachi") != null ? rs.getString("diachi") : "";

String cccd = rs.getString("cccd");

writer.println(String.format("%s,%s,%s,%s,%s,%s,%s,%s,%s", id, maSV, tenkhoa, hoten, gioitinh, ngaysinh, sdt, diachi, cccd));

}

rs.close();

stmt.close();

JOptionPane.showMessageDialog(this, "Dữ liệu đã được xuất ra students.csv!", "Thành công", JOptionPane.INFORMATION\_MESSAGE);

} catch (FileNotFoundException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi xuất file: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Lỗi khi truy vấn dữ liệu: " + ex.getMessage(), "Lỗi", JOptionPane.ERROR\_MESSAGE);

ex.printStackTrace();

} finally {

try {

if (conn != null) conn.close();

} catch (SQLException ex) {

System.err.println("Lỗi khi đóng kết nối: " + ex.getMessage());

}

}

} else if (e.getSource() == cancelButton) {

setVisible(false);

dispose();

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new StudentManagement());

}

}

1. **Các thư viên tôi sử dụng là jcalender.jar, ResultSet2xml.jar, jdk17.**