

**TRƯỜNG ĐẠI HỌC**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO BÀI TẬP LỚN**

**CÔNG NGHỆ PHẦN MỀM**

**ĐỀ TÀI:**

**QUẢN LÝ BÁN HÀNG CỦA SIÊU THỊ**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn: |  |
| Sinh viên thực hiện: |  |
| Thành viên nhóm: |  |
| Lớp: |  |

Hà Nội, năm 2024

**LẬP TRÌNH NHÂN SỰ**

|  |  |
| --- | --- |
| Tên chương trình: QUẢN LÝ BÁN HÀNG CỦA SIÊU THỊ | Mã chương trình:  QLBHCST |
| Tên module: Lập trình Bộ phận nhân sự | Mã tài liệu:  LT1 |
| Tên người làm: | Mã người làm: N2 |
| Thời hạn: Ngày bắt đầu: 10/04/2024 | Ngày kết thúc:  14/05/2024 |
| Ngày nộp: 14/05/2024 | |
| Phiên bản/ File: LT1-240526.docx | |

**NGƯỜI LÀM KÝ XÁC NHẬN**

**MỤC LỤC**

I. GIAO DIỆN 3

1. FormDangNhap 3

2. FormNhanSu 17

II.THỰC THỂ 65

1. User 65

2. UserRole 65

3. Role 66

4. Employee 66

III. ĐỐI TƯỢNG TRUYỀN DỮ LIỆU 68

UserWithRoles 68

IV. CONTOLLER 70

1. UserController 70

2. EmployeeController 84

V. CONNECT 93

MyConnect 93

VI. KẾT QUẢ 94

1. Đăng nhập 94

2. Quản lý nhân viên 95

3. Quản lý tài khoản 96

# I. GIAO DIỆN

## 1. FormDangNhap

1. /\*  
    \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  
    \* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template  
    \*/  
   package org.example.views;  
     
   import org.example.controllers.UserController;  
   import org.example.models.Role;  
     
   import javax.swing.\*;  
   import java.awt.\*;  
   import java.awt.event.ActionEvent;  
   import java.awt.event.KeyAdapter;  
   import java.awt.event.KeyEvent;  
   import java.awt.event.KeyListener;  
   import java.util.List;  
     
   */\*\*  
    \*  
    \* @author   
    \*/*public class FormDangNhap extends javax.swing.JFrame {  
    private UserController userController; // Thêm biến để truy cập vào controller  
    private final int loggedInUserID; // Declare as final  
     
    public int getLoggedInUserID() {  
    return loggedInUserID;  
    }  
     
    */\*\*  
    \* Creates new form FormDangNhap  
    \*/* public FormDangNhap() {  
    initComponents();  
    userController = new UserController(); // Khởi tạo UserController  
    loggedInUserID = 0; // Initialize the final variable  
    loadRoles(); // Load danh sách quyền khi form được tạo  
    addKeyListeners();  
    addEscapeKeyBinding();  
    }  
     
    private void addEscapeKeyBinding() {  
    String escapeAction = "escapeAction";  
    KeyStroke escapeKeyStroke = KeyStroke.*getKeyStroke*(KeyEvent.*VK\_ESCAPE*, 0, false);  
     
    Action escapeActionHandler = new AbstractAction() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
    Object[] options = {"Thoát", "Không"};  
    int response = JOptionPane.*showOptionDialog*(  
    FormDangNhap.this,  
    "Bạn có muốn thoát chương trình?",  
    "Xác nhận",  
    JOptionPane.*YES\_NO\_OPTION*,  
    JOptionPane.*QUESTION\_MESSAGE*,  
    null,  
    options,  
    options[0]  
    );  
    if (response == JOptionPane.*NO\_OPTION*) {  
    System.*out*.println("không có gì");  
    } else if (response == JOptionPane.*YES\_OPTION*) {  
    // Thoát chương trình  
    System.*out*.println("đã thoát chương trình đăng nhập");  
    System.*exit*(0);  
    }  
    }  
    };  
     
    JRootPane rootPane = this.getRootPane();  
    rootPane.getInputMap(JComponent.*WHEN\_IN\_FOCUSED\_WINDOW*).put(escapeKeyStroke, escapeAction);  
    rootPane.getActionMap().put(escapeAction, escapeActionHandler);  
    }  
     
     
    private void addKeyListeners() {  
    KeyListener keyListener = new KeyAdapter() {  
    @Override  
    public void keyPressed(KeyEvent e) {  
    int keyCode = e.getKeyCode();  
    if (keyCode == KeyEvent.*VK\_ENTER*) {  
    btnDangNhap.doClick(); // Giả lập sự kiện nhấn nút Đăng Nhập  
    }  
   // else if (keyCode == KeyEvent.VK\_ESCAPE) {  
   // dispose(); // Đóng form đăng nhập  
   // }  
    }  
    };  
     
    tbUsername.addKeyListener(keyListener);  
    tbPassword.addKeyListener(keyListener);  
    cbRole.addKeyListener(keyListener);  
    btnDangNhap.addKeyListener(keyListener);  
    }  
    private void loadRoles() {  
    List<Role> roles = userController.getAllRoles();  
    for (Role role : roles) {  
    cbRole.addItem(role.getRoleName());  
    }  
    }  
    */\*\*  
    \* This method is called from within the constructor to initialize the form.  
    \* WARNING: Do NOT modify this code. The content of this method is always  
    \* regenerated by the Form Editor.  
    \*/* @SuppressWarnings("unchecked")  
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents  
    private void initComponents() {  
     
    jPanel1 = new javax.swing.JPanel();  
    btnDangNhap = new javax.swing.JButton();  
    jLabel4 = new javax.swing.JLabel();  
    jLabel1 = new javax.swing.JLabel();  
    jLabel2 = new javax.swing.JLabel();  
    jLabel3 = new javax.swing.JLabel();  
    cbRole = new javax.swing.JComboBox<>();  
    tbUsername = new javax.swing.JTextField();  
    tbPassword = new javax.swing.JTextField();  
     
    setDefaultCloseOperation(javax.swing.WindowConstants.*EXIT\_ON\_CLOSE*);  
    setBackground(new java.awt.Color(102, 153, 255));  
     
    jPanel1.setBackground(new java.awt.Color(204, 204, 255));  
     
    btnDangNhap.setBackground(new java.awt.Color(102, 102, 255));  
    btnDangNhap.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N  
    btnDangNhap.setForeground(new java.awt.Color(255, 255, 255));  
    btnDangNhap.setText("Đăng nhập");  
    btnDangNhap.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
    btnDangNhapActionPerformed(evt);  
    }  
    });  
     
    jLabel4.setFont(new java.awt.Font("Segoe UI", 1, 36)); // NOI18N  
    jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.*CENTER*);  
    jLabel4.setText("Đăng Nhập");  
     
    jLabel1.setFont(new java.awt.Font("Arial", 1, 24)); // NOI18N  
    jLabel1.setText("Tên dăng nhập");  
     
    jLabel2.setFont(new java.awt.Font("Arial", 1, 24)); // NOI18N  
    jLabel2.setText("Mật khẩu");  
     
    jLabel3.setFont(new java.awt.Font("Arial", 1, 24)); // NOI18N  
    jLabel3.setText("Quyền");  
     
    cbRole.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
    cbRoleActionPerformed(evt);  
    }  
    });  
     
    tbUsername.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
    tbUsernameActionPerformed(evt);  
    }  
    });  
     
    tbPassword.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
    tbPasswordActionPerformed(evt);  
    }  
    });  
     
    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  
    jPanel1.setLayout(jPanel1Layout);  
    jPanel1Layout.setHorizontalGroup(  
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(jPanel1Layout.createSequentialGroup()  
    .addContainerGap()  
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(jPanel1Layout.createSequentialGroup()  
    .addComponent(jLabel1)  
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*UNRELATED*)  
    .addComponent(tbUsername))  
    .addComponent(jLabel4, javax.swing.GroupLayout.Alignment.*TRAILING*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
    .addGroup(jPanel1Layout.createSequentialGroup()  
    .addGap(62, 62, 62)  
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(jPanel1Layout.createSequentialGroup()  
    .addGap(31, 31, 31)  
    .addComponent(jLabel3)  
    .addGap(18, 18, 18)  
    .addComponent(cbRole, 0, 454, Short.*MAX\_VALUE*))  
    .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel1Layout.createSequentialGroup()  
    .addComponent(jLabel2)  
    .addGap(18, 18, 18)  
    .addComponent(tbPassword))))  
    .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel1Layout.createSequentialGroup()  
    .addGap(0, 0, Short.*MAX\_VALUE*)  
    .addComponent(btnDangNhap, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 177, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
    .addContainerGap())  
    );  
    jPanel1Layout.setVerticalGroup(  
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel1Layout.createSequentialGroup()  
    .addContainerGap()  
    .addComponent(jLabel4, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 74, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
    .addGap(66, 66, 66)  
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addComponent(jLabel1)  
    .addGroup(jPanel1Layout.createSequentialGroup()  
    .addGap(3, 3, 3)  
    .addComponent(tbUsername, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 34, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
    .addGap(12, 12, 12)  
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
    .addComponent(jLabel2)  
    .addComponent(tbPassword, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 34, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*UNRELATED*)  
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
    .addComponent(cbRole, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 34, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
    .addComponent(jLabel3))  
    .addGap(86, 86, 86)  
    .addComponent(btnDangNhap, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 48, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
    .addContainerGap())  
    );  
     
    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
    getContentPane().setLayout(layout);  
    layout.setHorizontalGroup(  
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(layout.createSequentialGroup()  
    .addContainerGap()  
    .addComponent(jPanel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
    .addContainerGap(javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
    );  
    layout.setVerticalGroup(  
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
    .addGroup(layout.createSequentialGroup()  
    .addContainerGap()  
    .addComponent(jPanel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
    .addContainerGap())  
    );  
     
    pack();  
    }// </editor-fold>//GEN-END:initComponents  
     
    private void tbUsernameActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbUsernameActionPerformed  
    // *TODO add your handling code here:* }//GEN-LAST:event\_tbUsernameActionPerformed  
     
    private void tbPasswordActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbPasswordActionPerformed  
    // *TODO add your handling code here:* }//GEN-LAST:event\_tbPasswordActionPerformed  
     
    private void cbRoleActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_cbRoleActionPerformed  
    // *TODO add your handling code here:* }//GEN-LAST:event\_cbRoleActionPerformed  
     
    private void btnDangNhapActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnDangNhapActionPerformed  
    // *TODO add your handling code here:* String username = tbUsername.getText();  
    String password = tbPassword.getText();  
    String selectedRole = cbRole.getSelectedItem().toString();  
     
    // Attempt login  
    boolean loggedIn = userController.loginWithRole(username, password, selectedRole);  
     
    if (loggedIn) {  
    // Retrieve the user's ID after successful login  
    int userID = userController.getUserID(username);  
     
    // Save the user ID in the session  
    userController.setUserIDInSession(userID);  
     
    // Update the loggedInUserID  
     
    JOptionPane.*showMessageDialog*(this, "Đăng nhập thành công với quyền: " + selectedRole);  
    System.*out*.println("Đăng nhập thành công với các thông tin");  
    // Redirect to the appropriate page based on the user's role  
    switch(selectedRole) {  
    case "bộ phận nhân sự":  
    FormNhanSu formNhanSu = new FormNhanSu();  
    formNhanSu.setVisible(true);  
    System.*out*.println("Chức Vụ: bộ phận nhân sự");  
    break;  
    case "bộ phận giao dịch":  
    FormGiaoDich formCungUng = new FormGiaoDich(userID);  
    formCungUng.setVisible(true);  
    System.*out*.println("Chức Vụ: bộ phận giao dịch");  
    break;  
    case "quản lý hàng":  
    FormQuanLyHang formQuanLyHang = new FormQuanLyHang(userID);  
    formQuanLyHang.setVisible(true);  
    System.*out*.println("Chức Vụ: quản lý hàng");  
    break;  
    case "kế toán":  
    FormKeToan formKeToan = new FormKeToan(userID);  
    formKeToan.setVisible(true);  
    System.*out*.println("Chức Vụ: kế toán");  
    break;  
    case "Admin":  
    FormAdmin formAdmin = new FormAdmin();  
    formAdmin.setVisible(true);  
    break;  
    default:  
    JOptionPane.*showMessageDialog*(this, "Không tìm thấy trang tương ứng cho quyền này");  
    }  
    System.*out*.println("UserName: " + username);  
    System.*out*.println("UserID: " + userID);  
     
    this.dispose(); // Close the login form  
    } else {  
    JOptionPane.*showMessageDialog*(this, "Đăng nhập thất bại");  
    }  
    }//GEN-LAST:event\_btnDangNhapActionPerformed  
     
    */\*\*  
    \* @param args the command line arguments  
    \*/* public static void main(String args[]) {  
    /\* Set the Nimbus look and feel \*/  
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
    /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
    \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html   
    \*/  
    try {  
    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.*getInstalledLookAndFeels*()) {  
    if ("Nimbus".equals(info.getName())) {  
    javax.swing.UIManager.*setLookAndFeel*(info.getClassName());  
    break;  
    }  
    }  
    } catch (ClassNotFoundException ex) {  
    java.util.logging.Logger.*getLogger*(FormDangNhap.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
    } catch (InstantiationException ex) {  
    java.util.logging.Logger.*getLogger*(FormDangNhap.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
    } catch (IllegalAccessException ex) {  
    java.util.logging.Logger.*getLogger*(FormDangNhap.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {  
    java.util.logging.Logger.*getLogger*(FormDangNhap.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
    }  
    //</editor-fold>  
     
    /\* Create and display the form \*/  
    java.awt.EventQueue.*invokeLater*(new Runnable() {  
    public void run() {  
    new FormDangNhap().setVisible(true);  
    }  
    });  
    }  
     
    // Variables declaration - do not modify//GEN-BEGIN:variables  
    private javax.swing.JButton btnDangNhap;  
    private javax.swing.JComboBox<String> cbRole;  
    private javax.swing.JLabel jLabel1;  
    private javax.swing.JLabel jLabel2;  
    private javax.swing.JLabel jLabel3;  
    private javax.swing.JLabel jLabel4;  
    private javax.swing.JPanel jPanel1;  
    private javax.swing.JTextField tbPassword;  
    private javax.swing.JTextField tbUsername;  
    // End of variables declaration//GEN-END:variables  
   }

## 2. FormNhanSu

/\*  
 \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license  
 \* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template  
 \*/  
package org.example.views;  
  
import org.example.controllers.EmployeeController;  
import org.example.controllers.UserController;  
import org.example.models.Employee;  
import org.example.models.Role;  
import org.example.models.User;  
  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
import java.awt.event.KeyListener;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.List;  
  
*/\*\*  
 \* @author   
 \*/*public class FormNhanSu extends javax.swing.JFrame {  
 private EmployeeController employeeController;  
 private DefaultTableModel tableModel;  
 private UserController userController;  
  
  
 public FormNhanSu() {  
 initComponents(); // Initialize components first  
 showNhanVienTable(); // Then populate the table  
 showUserTable();  
 userController = new UserController(); // Assign the local instance to the class variable  
 loadRoles(); // Load danh sách quyền khi form được tạo  
 loadEmployee(); // Load danh sách quyền khi form được tạo  
 addEscapeKeyBinding();  
 }  
 private void addEscapeKeyBinding() {  
 String escapeAction = "escapeAction";  
 KeyStroke escapeKeyStroke = KeyStroke.*getKeyStroke*(KeyEvent.*VK\_ESCAPE*, 0, false);  
  
 Action escapeActionHandler = new AbstractAction() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Object[] options = {"Đăng xuất", "Thoát", "Không"};  
 int response = JOptionPane.*showOptionDialog*(  
 FormNhanSu.this,  
 "Bạn có muốn đăng xuất hay thoát chương trình?",  
 "Xác nhận",  
 JOptionPane.*YES\_NO\_CANCEL\_OPTION*,  
 JOptionPane.*QUESTION\_MESSAGE*,  
 null,  
 options,  
 options[0]  
 );  
  
 if (response == JOptionPane.*YES\_OPTION*) {  
 // Xử lý đăng xuất  
 System.*out*.println("Bộ phận nhân sự đã đăng xuất");  
 btnDangXuatActionPerformed(null);  
 } else if (response == JOptionPane.*NO\_OPTION*) {  
 System.*out*.println("Đã thoát chương trình");  
 // Thoát chương trình  
 System.*exit*(0);  
 }else if (response == JOptionPane.*CANCEL\_OPTION*) {  
 System.*out*.println("Không có gì");  
 }  
 }  
 };  
  
 JRootPane rootPane = this.getRootPane();  
 rootPane.getInputMap(JComponent.*WHEN\_IN\_FOCUSED\_WINDOW*).put(escapeKeyStroke, escapeAction);  
 rootPane.getActionMap().put(escapeAction, escapeActionHandler);  
 }  
  
  
 private void loadRoles() {  
 UserController userController = new UserController(); // Create an instance of UserController  
 List<Role> roles = userController.getAllRoles(); // Call getAllRoles method  
 for (Role role : roles) {  
 Role.addItem(role.getRoleName());  
 }  
 }  
  
  
 private void loadEmployee() {  
 List<Employee> employees = userController.getAllEmployees();  
 // Assuming you have a combo box named "Employee" to display the employee names  
 for (Employee employee : employees) {  
 Employee.addItem(employee.getFullName()); // Assuming you have a method to get the full name of an employee  
 }  
 }  
  
 */\*\*  
 \* Creates new form FormNhanSu  
 \*/* private void showNhanVienTable() {  
 employeeController = new EmployeeController();  
 // Gọi phương thức getAllNhanVienTableModel từ đối tượng EmployeeController  
 tableModel = employeeController.getAllNhanVienTableModel();  
 // Đặt mô hình dữ liệu cho bảng  
 TableQuanLyNhanVien.setModel(tableModel);  
 }  
  
 private void showUserTable() {  
 userController = new UserController();  
 // Gọi phương thức getAllNhanVienTableModel từ đối tượng EmployeeController  
 tableModel = userController.getAllUserInfoTableModel();  
 // Đặt mô hình dữ liệu cho bảng  
 TableUser.setModel(tableModel);  
 }  
  
 private Date convertToDate(String dateString) {  
 SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");  
 try {  
 return dateFormat.parse(dateString);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 return null; // Handle the parse exception appropriately  
 }  
 }  
  
 */\*\*  
 \* This method is called from within the constructor to initialize the form.  
 \* WARNING: Do NOT modify this code. The content of this method is always  
 \* regenerated by the Form Editor.  
 \*/* @SuppressWarnings("unchecked")  
 // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents  
 private void initComponents() {  
  
 jPmain = new javax.swing.JPanel();  
 jtPmain = new javax.swing.JTabbedPane();  
 jPquanlynhanvien = new javax.swing.JPanel();  
 jLabel2 = new javax.swing.JLabel();  
 jLabel3 = new javax.swing.JLabel();  
 jLabel4 = new javax.swing.JLabel();  
 jLabel5 = new javax.swing.JLabel();  
 jScrollPane1 = new javax.swing.JScrollPane();  
 TableQuanLyNhanVien = new javax.swing.JTable();  
 jLabel6 = new javax.swing.JLabel();  
 jLabel7 = new javax.swing.JLabel();  
 jLabel8 = new javax.swing.JLabel();  
 jLabel9 = new javax.swing.JLabel();  
 tbEmployeeID = new javax.swing.JTextField();  
 tbFirstName = new javax.swing.JTextField();  
 tbPosition = new javax.swing.JTextField();  
 tbLastName = new javax.swing.JTextField();  
 tbSalary = new javax.swing.JTextField();  
 tbContactInfo = new javax.swing.JTextField();  
 tbHireDate = new javax.swing.JTextField();  
 tbDateOfBirth = new javax.swing.JTextField();  
 jPanel3 = new javax.swing.JPanel();  
 btnLamRong = new javax.swing.JButton();  
 btnXoa = new javax.swing.JButton();  
 btnSua = new javax.swing.JButton();  
 btnThem = new javax.swing.JButton();  
 jPquanlytaikhoan = new javax.swing.JPanel();  
 jScrollPane2 = new javax.swing.JScrollPane();  
 TableUser = new javax.swing.JTable();  
 jPanel1 = new javax.swing.JPanel();  
 jLabel10 = new javax.swing.JLabel();  
 jLabel11 = new javax.swing.JLabel();  
 jLabel12 = new javax.swing.JLabel();  
 jLabel13 = new javax.swing.JLabel();  
 Employee = new javax.swing.JComboBox<>();  
 jLabel14 = new javax.swing.JLabel();  
 jLabel15 = new javax.swing.JLabel();  
 tbUserID = new javax.swing.JTextField();  
 tbUserName = new javax.swing.JTextField();  
 tbEmail = new javax.swing.JTextField();  
 tbPasswordHash = new javax.swing.JTextField();  
 Role = new javax.swing.JComboBox<>();  
 jPanel2 = new javax.swing.JPanel();  
 btnSua1 = new javax.swing.JButton();  
 btnLamRong1 = new javax.swing.JButton();  
 btnThem1 = new javax.swing.JButton();  
 btnXoa1 = new javax.swing.JButton();  
 jLabel1 = new javax.swing.JLabel();  
 btnDangXuat = new javax.swing.JButton();  
  
 setDefaultCloseOperation(javax.swing.WindowConstants.*EXIT\_ON\_CLOSE*);  
  
 jLabel2.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel2.setText("Position");  
  
 jLabel3.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel3.setText("EmployeeID");  
  
 jLabel4.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel4.setText("FirstName");  
  
 jLabel5.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel5.setText("LastName");  
  
 TableQuanLyNhanVien.setFont(new java.awt.Font("Arial", 1, 14)); // NOI18N  
 TableQuanLyNhanVien.setModel(new javax.swing.table.DefaultTableModel(  
 new Object [][] {  
 {},  
 {},  
 {},  
 {}  
 },  
 new String [] {  
  
 }  
 ));  
 TableQuanLyNhanVien.setRowHeight(50);  
 TableQuanLyNhanVien.setSelectionBackground(new java.awt.Color(51, 153, 255));  
 TableQuanLyNhanVien.setShowHorizontalLines(true);  
 TableQuanLyNhanVien.addMouseListener(new java.awt.event.MouseAdapter() {  
 public void mouseClicked(java.awt.event.MouseEvent evt) {  
 TableQuanLyNhanVienMouseClicked(evt);  
 }  
 });  
 jScrollPane1.setViewportView(TableQuanLyNhanVien);  
 TableQuanLyNhanVien.getColumnModel().getSelectionModel().setSelectionMode(javax.swing.ListSelectionModel.*SINGLE\_INTERVAL\_SELECTION*);  
  
 jLabel6.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel6.setText("DateOfBirth");  
  
 jLabel7.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel7.setText("ContactInfo");  
  
 jLabel8.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel8.setText("HireDate");  
  
 jLabel9.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel9.setText("Salary");  
  
 tbEmployeeID.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbEmployeeIDActionPerformed(evt);  
 }  
 });  
  
 tbFirstName.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbFirstNameActionPerformed(evt);  
 }  
 });  
  
 tbPosition.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbPositionActionPerformed(evt);  
 }  
 });  
  
 tbLastName.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbLastNameActionPerformed(evt);  
 }  
 });  
  
 tbSalary.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbSalaryActionPerformed(evt);  
 }  
 });  
  
 tbContactInfo.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbContactInfoActionPerformed(evt);  
 }  
 });  
  
 tbHireDate.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbHireDateActionPerformed(evt);  
 }  
 });  
  
 tbDateOfBirth.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbDateOfBirthActionPerformed(evt);  
 }  
 });  
  
 btnLamRong.setBackground(new java.awt.Color(0, 204, 204));  
 btnLamRong.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
 btnLamRong.setForeground(new java.awt.Color(255, 255, 255));  
 btnLamRong.setText("Làm Rỗng");  
 btnLamRong.setHorizontalTextPosition(javax.swing.SwingConstants.*CENTER*);  
 btnLamRong.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 btnLamRongActionPerformed(evt);  
 }  
 });  
  
 btnXoa.setBackground(new java.awt.Color(220, 53, 69));  
 btnXoa.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
 btnXoa.setForeground(new java.awt.Color(255, 255, 255));  
 btnXoa.setText("Xóa");  
 btnXoa.setHorizontalTextPosition(javax.swing.SwingConstants.*CENTER*);  
 btnXoa.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 btnXoaActionPerformed(evt);  
 }  
 });  
  
 btnSua.setBackground(new java.awt.Color(108, 117, 125));  
 btnSua.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
 btnSua.setForeground(new java.awt.Color(255, 255, 255));  
 btnSua.setText("Sửa");  
 btnSua.setHorizontalTextPosition(javax.swing.SwingConstants.*CENTER*);  
 btnSua.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 btnSuaActionPerformed(evt);  
 }  
 });  
  
 btnThem.setBackground(new java.awt.Color(0, 123, 255));  
 btnThem.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
 btnThem.setForeground(new java.awt.Color(255, 255, 255));  
 btnThem.setText("Thêm");  
 btnThem.setHorizontalTextPosition(javax.swing.SwingConstants.*CENTER*);  
 btnThem.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 btnThemActionPerformed(evt);  
 }  
 });  
  
 javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);  
 jPanel3.setLayout(jPanel3Layout);  
 jPanel3Layout.setHorizontalGroup(  
 jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel3Layout.createSequentialGroup()  
 .addContainerGap(93, Short.*MAX\_VALUE*)  
 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(btnXoa, javax.swing.GroupLayout.Alignment.*TRAILING*, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 96, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(btnThem, javax.swing.GroupLayout.Alignment.*TRAILING*, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 96, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(18, 18, 18)  
 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(btnSua, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(btnLamRong, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 .addContainerGap())  
 );  
 jPanel3Layout.setVerticalGroup(  
 jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel3Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(btnSua, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(btnThem, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 .addGap(30, 30, 30)  
 .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(btnLamRong, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(btnXoa, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 .addContainerGap())  
 );  
  
 javax.swing.GroupLayout jPquanlynhanvienLayout = new javax.swing.GroupLayout(jPquanlynhanvien);  
 jPquanlynhanvien.setLayout(jPquanlynhanvienLayout);  
 jPquanlynhanvienLayout.setHorizontalGroup(  
 jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.*TRAILING*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 859, Short.*MAX\_VALUE*)  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addComponent(jLabel3)  
 .addGap(18, 18, 18)  
 .addComponent(tbEmployeeID, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 195, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addComponent(jLabel4)  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(tbFirstName, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 195, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jLabel5)  
 .addComponent(jLabel2))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(tbLastName, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 195, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(tbPosition, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 195, javax.swing.GroupLayout.*PREFERRED\_SIZE*))))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addComponent(jLabel6)  
 .addGap(18, 18, 18)  
 .addComponent(tbDateOfBirth, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 195, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jLabel8)  
 .addComponent(jLabel7)  
 .addComponent(jLabel9))  
 .addGap(18, 18, 18)  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(tbContactInfo, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 195, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(tbHireDate, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 195, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(tbSalary, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 195, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))))  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPquanlynhanvienLayout.createSequentialGroup()  
 .addGap(0, 0, Short.*MAX\_VALUE*)  
 .addComponent(jPanel3, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
 .addContainerGap())  
 );  
 jPquanlynhanvienLayout.setVerticalGroup(  
 jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addGap(18, 18, 18)  
 .addComponent(jScrollPane1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(18, 18, 18)  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel3)  
 .addComponent(tbEmployeeID, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(18, 18, 18)  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel4)  
 .addComponent(tbFirstName, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(18, 18, 18)  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addComponent(jLabel5)  
 .addGap(18, 18, 18)  
 .addComponent(jLabel2))  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addComponent(tbLastName, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(18, 18, 18)  
 .addComponent(tbPosition, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))))  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jLabel6)  
 .addComponent(tbDateOfBirth, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(18, 18, 18)  
 .addGroup(jPquanlynhanvienLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addComponent(jLabel7, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 16, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(18, 18, 18)  
 .addComponent(jLabel8)  
 .addGap(18, 18, 18)  
 .addComponent(jLabel9))  
 .addGroup(jPquanlynhanvienLayout.createSequentialGroup()  
 .addComponent(tbContactInfo, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(18, 18, 18)  
 .addComponent(tbHireDate, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*)  
 .addComponent(tbSalary, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, 12, Short.*MAX\_VALUE*)  
 .addComponent(jPanel3, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap())  
 );  
  
 jtPmain.addTab("Quản Lý Nhân Viên", jPquanlynhanvien);  
  
 TableUser.setFont(new java.awt.Font("Arial", 1, 14)); // NOI18N  
 TableUser.setModel(new javax.swing.table.DefaultTableModel(  
 new Object [][] {  
 {},  
 {},  
 {},  
 {}  
 },  
 new String [] {  
  
 }  
 ));  
 TableUser.setRowHeight(50);  
 TableUser.setSelectionBackground(new java.awt.Color(51, 153, 255));  
 TableUser.setShowHorizontalLines(true);  
 TableUser.addMouseListener(new java.awt.event.MouseAdapter() {  
 public void mouseClicked(java.awt.event.MouseEvent evt) {  
 TableUserMouseClicked(evt);  
 }  
 });  
 jScrollPane2.setViewportView(TableUser);  
  
 jLabel10.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel10.setText("UserID");  
  
 jLabel11.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel11.setText("Username");  
  
 jLabel12.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel12.setText("PasswordHash");  
  
 jLabel13.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel13.setText("Email");  
  
 Employee.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 EmployeeActionPerformed(evt);  
 }  
 });  
  
 jLabel14.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel14.setText("EmployeeID");  
  
 jLabel15.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N  
 jLabel15.setText("Role");  
  
 tbUserID.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbUserIDActionPerformed(evt);  
 }  
 });  
  
 tbUserName.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbUserNameActionPerformed(evt);  
 }  
 });  
  
 tbEmail.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbEmailActionPerformed(evt);  
 }  
 });  
  
 tbPasswordHash.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 tbPasswordHashActionPerformed(evt);  
 }  
 });  
  
 Role.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 RoleActionPerformed(evt);  
 }  
 });  
  
 btnSua1.setBackground(new java.awt.Color(0, 123, 255));  
 btnSua1.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
 btnSua1.setForeground(new java.awt.Color(255, 255, 255));  
 btnSua1.setText("Sửa");  
 btnSua1.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 btnSua1ActionPerformed(evt);  
 }  
 });  
  
 btnLamRong1.setBackground(new java.awt.Color(0, 204, 204));  
 btnLamRong1.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
 btnLamRong1.setForeground(new java.awt.Color(255, 255, 255));  
 btnLamRong1.setText("Làm Rỗng");  
  
 btnThem1.setBackground(new java.awt.Color(108, 117, 125));  
 btnThem1.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
 btnThem1.setForeground(new java.awt.Color(255, 255, 255));  
 btnThem1.setText("Thêm");  
 btnThem1.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 btnThem1ActionPerformed(evt);  
 }  
 });  
  
 btnXoa1.setBackground(new java.awt.Color(220, 53, 69));  
 btnXoa1.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
 btnXoa1.setForeground(new java.awt.Color(255, 255, 255));  
 btnXoa1.setText("Xóa");  
 btnXoa1.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 btnXoa1ActionPerformed(evt);  
 }  
 });  
  
 javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);  
 jPanel2.setLayout(jPanel2Layout);  
 jPanel2Layout.setHorizontalGroup(  
 jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel2Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(btnThem1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 142, Short.*MAX\_VALUE*)  
 .addComponent(btnXoa1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, 79, Short.*MAX\_VALUE*)  
 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(btnLamRong1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 137, Short.*MAX\_VALUE*)  
 .addComponent(btnSua1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 .addContainerGap())  
 );  
 jPanel2Layout.setVerticalGroup(  
 jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel2Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(btnThem1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 41, Short.*MAX\_VALUE*)  
 .addComponent(btnSua1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 .addGap(18, 18, 18)  
 .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(btnXoa1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 42, Short.*MAX\_VALUE*)  
 .addComponent(btnLamRong1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 .addContainerGap(javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 );  
  
 javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  
 jPanel1.setLayout(jPanel1Layout);  
 jPanel1Layout.setHorizontalGroup(  
 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*TRAILING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addComponent(jLabel10, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 76, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(18, 18, 18)  
 .addComponent(tbUserID, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 255, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addComponent(jLabel13, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 65, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(tbEmail, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 226, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addComponent(jLabel12)  
 .addGap(18, 18, 18)  
 .addComponent(tbPasswordHash, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 218, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addComponent(jLabel11)  
 .addGap(18, 18, 18)  
 .addComponent(tbUserName, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 255, javax.swing.GroupLayout.*PREFERRED\_SIZE*))))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, 125, Short.*MAX\_VALUE*)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel1Layout.createSequentialGroup()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*TRAILING*)  
 .addComponent(jLabel15, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 50, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel14))  
 .addGap(31, 31, 31)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(Employee, 0, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(Role, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 205, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
 .addComponent(jPanel2, javax.swing.GroupLayout.Alignment.*TRAILING*, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addContainerGap())  
 );  
 jPanel1Layout.setVerticalGroup(  
 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel1Layout.createSequentialGroup()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(Employee, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel14)))  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGap(43, 43, 43)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel15)  
 .addComponent(Role, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, 75, Short.*MAX\_VALUE*)  
 .addComponent(jPanel2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap())  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGap(20, 20, 20)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel10)  
 .addComponent(tbUserID, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(18, 18, 18)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel11)  
 .addComponent(tbUserName, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel12)  
 .addComponent(tbPasswordHash, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(11, 11, 11)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel13)  
 .addComponent(tbEmail, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addContainerGap(javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 );  
  
 javax.swing.GroupLayout jPquanlytaikhoanLayout = new javax.swing.GroupLayout(jPquanlytaikhoan);  
 jPquanlytaikhoan.setLayout(jPquanlytaikhoanLayout);  
 jPquanlytaikhoanLayout.setHorizontalGroup(  
 jPquanlytaikhoanLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPquanlytaikhoanLayout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPquanlytaikhoanLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*TRAILING*)  
 .addComponent(jScrollPane2)  
 .addComponent(jPanel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 .addContainerGap())  
 );  
 jPquanlytaikhoanLayout.setVerticalGroup(  
 jPquanlytaikhoanLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPquanlytaikhoanLayout.createSequentialGroup()  
 .addContainerGap()  
 .addComponent(jScrollPane2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 413, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(18, 18, 18)  
 .addComponent(jPanel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 );  
  
 jtPmain.addTab("Quản Lý Tài Khoản", jPquanlytaikhoan);  
  
 jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 36)); // NOI18N  
 jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.*CENTER*);  
 jLabel1.setText("Bộ Phận Nhân Sự");  
  
 btnDangXuat.setBackground(new java.awt.Color(220, 53, 69));  
 btnDangXuat.setFont(new java.awt.Font("Arial", 0, 18)); // NOI18N  
 btnDangXuat.setForeground(new java.awt.Color(255, 255, 255));  
 btnDangXuat.setText("Đăng Xuất");  
 btnDangXuat.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 btnDangXuatActionPerformed(evt);  
 }  
 });  
  
 javax.swing.GroupLayout jPmainLayout = new javax.swing.GroupLayout(jPmain);  
 jPmain.setLayout(jPmainLayout);  
 jPmainLayout.setHorizontalGroup(  
 jPmainLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPmainLayout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPmainLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jtPmain)  
 .addGroup(jPmainLayout.createSequentialGroup()  
 .addComponent(jLabel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 669, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(39, 39, 39)  
 .addComponent(btnDangXuat)  
 .addGap(0, 0, Short.*MAX\_VALUE*)))  
 .addContainerGap())  
 );  
 jPmainLayout.setVerticalGroup(  
 jPmainLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPmainLayout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPmainLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 103, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(btnDangXuat))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*UNRELATED*)  
 .addComponent(jtPmain)  
 .addContainerGap())  
 );  
  
 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
 getContentPane().setLayout(layout);  
 layout.setHorizontalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addContainerGap()  
 .addComponent(jPmain, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addContainerGap())  
 );  
 layout.setVerticalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addContainerGap()  
 .addComponent(jPmain, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addContainerGap())  
 );  
  
 pack();  
 }// </editor-fold>//GEN-END:initComponents  
  
 private void RoleActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_RoleActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_RoleActionPerformed  
  
 private void tbPasswordHashActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbPasswordHashActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbPasswordHashActionPerformed  
  
 private void tbEmailActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbEmailActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbEmailActionPerformed  
  
 private void tbUserNameActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbUserNameActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbUserNameActionPerformed  
  
 private void tbUserIDActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbUserIDActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbUserIDActionPerformed  
  
 private void EmployeeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_EmployeeActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_EmployeeActionPerformed  
  
 private void btnThem1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnThem1ActionPerformed  
 // Lấy thông tin từ các trường dữ liệu trên giao diện  
 String username = tbUserName.getText();  
 String passwordHash = tbPasswordHash.getText();  
 String email = tbEmail.getText();  
 String employeeName = (String) Employee.getSelectedItem(); // Lấy tên nhân viên từ JComboBox  
 String roleName = (String) Role.getSelectedItem(); // Lấy tên quyền từ JComboBox  
  
 // Tạo một đối tượng User từ thông tin thu thập được  
 User user = new User(username, passwordHash, email);  
  
 // Gọi phương thức addUser của UserController để thêm người dùng mới  
 UserController userController = new UserController();  
  
 // Tìm roleID từ tên quyền  
 int roleID = userController.findRoleIDByRoleName(roleName);  
  
 // Thực hiện thêm người dùng và kiểm tra kết quả  
 String result = userController.addUser(user, employeeName, roleID);  
  
 // Kiểm tra xem thêm người dùng có thành công hay không và thông báo cho người dùng  
 switch (result) {  
 case "Success":  
 JOptionPane.*showMessageDialog*(this, "Thêm người dùng thành công!");  
 // Load lại bảng  
 showUserTable();  
 loadEmployee();  
 showNhanVienTable();  
 // Xóa các trường nhập liệu sau khi thêm thành công  
 clearInputFields();  
 break;  
 case "Username already exists.":  
 JOptionPane.*showMessageDialog*(this, "Tên người dùng đã tồn tại!");  
 break;  
 case "Email already exists.":  
 JOptionPane.*showMessageDialog*(this, "Email đã tồn tại!");  
 break;  
 case "Employee ID already exists.":  
 JOptionPane.*showMessageDialog*(this, "Employee ID đã tồn tại!");  
 break;  
 default:  
 JOptionPane.*showMessageDialog*(this, "Thêm người dùng thất bại!");  
 break;  
 }  
 }//GEN-LAST:event\_btnThem1ActionPerformed  
  
 private void btnSua1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnSua1ActionPerformed  
 // *TODO add your handling code here:* // Lấy thông tin từ các trường dữ liệu trên giao diện  
 String username = tbUserName.getText();  
 String passwordHash = tbPasswordHash.getText();  
 String email = tbEmail.getText();  
 String employeeName = (String) Employee.getSelectedItem(); // Lấy tên nhân viên từ JComboBox  
 String roleName = (String) Role.getSelectedItem(); // Lấy tên quyền từ JComboBox  
  
 // Tạo một đối tượng User từ thông tin thu thập được  
 User user = new User(username, passwordHash, email);  
  
 // Gọi phương thức updateUser của UserController để sửa thông tin người dùng  
 UserController userController = new UserController();  
  
 // Tìm roleID từ tên quyền  
 int roleID = userController.findRoleIDByRoleName(roleName);  
  
 // Thực hiện sửa thông tin người dùng và kiểm tra kết quả  
 boolean success = userController.updateUser(user, employeeName, roleID);  
  
 // Kiểm tra xem sửa thông tin người dùng có thành công hay không và thông báo cho người dùng  
 if (success) {  
 JOptionPane.*showMessageDialog*(this, "Sửa thông tin người dùng thành công!");  
 // Cập nhật lại bảng hiển thị thông tin người dùng  
 showUserTable();  
 loadEmployee();  
 showNhanVienTable();  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Sửa thông tin người dùng thất bại!");  
 }  
 }//GEN-LAST:event\_btnSua1ActionPerformed  
  
 private void btnXoa1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnXoa1ActionPerformed  
 // *TODO add your handling code here:* int selectedRow = TableUser.getSelectedRow();  
 if (selectedRow != -1) { // Ensure that a row is selected  
 int userID = (int) TableUser.getValueAt(selectedRow, 0); // Assuming the first column contains userID  
 UserController userController = new UserController();  
 boolean success = userController.deleteUser(userID);  
 if (success) {  
 JOptionPane.*showMessageDialog*(this, "Xóa người dùng thành công!");  
 // Load lại bảng người dùng sau khi xóa  
 showUserTable();  
 loadEmployee();  
 showNhanVienTable();  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Xóa người dùng thất bại!");  
 }  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Vui lòng chọn một người dùng để xóa!");  
 }  
 }//GEN-LAST:event\_btnXoa1ActionPerformed  
  
 private void TableUserMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_TableUserMouseClicked  
 int selectedRow = TableUser.getSelectedRow();  
  
 if (selectedRow != -1) {  
 int userID = (int) TableUser.getValueAt(selectedRow, 0);  
 String username = (String) TableUser.getValueAt(selectedRow, 1);  
 String passwordHash = (String) TableUser.getValueAt(selectedRow, 2);  
 String email = (String) TableUser.getValueAt(selectedRow, 3);  
  
 tbUserID.setText(String.*valueOf*(userID));  
 tbUserName.setText(username);  
 tbPasswordHash.setText(passwordHash);  
 tbEmail.setText(email);  
  
 Role role = userController.findRoleByUserId(userID);  
 if (role != null) {  
 selectRole(role.getRoleName());  
 }  
  
 // Fetch and select the employee associated with the user  
 String fullName = userController.findEmployeeByUserId(userID);  
 if (fullName != null) {  
 selectEmployee(fullName);  
 } else {  
 Employee.setSelectedItem(null); // Clear selection if employee is not found  
 }  
  
 }  
 }//GEN-LAST:event\_TableUserMouseClicked  
  
 private void btnThemActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnThemActionPerformed  
 // Get input values from text fields  
 String firstName = tbFirstName.getText();  
 String lastName = tbLastName.getText();  
 String position = tbPosition.getText();  
 String dateOfBirthString = tbDateOfBirth.getText(); // Assuming this is a string representing date of birth  
 String contactInfo = tbContactInfo.getText();  
 String hireDateString = tbHireDate.getText(); // Assuming this is a string representing hire date  
 String salaryString = tbSalary.getText(); // Assuming this is a string representing salary  
 // You can extract other fields similarly  
  
 // Check if date of birth and hire date fields are not empty  
 // Check if any required field is empty  
 if (firstName.isEmpty() || lastName.isEmpty() || position.isEmpty() || contactInfo.isEmpty() || dateOfBirthString.isEmpty() || hireDateString.isEmpty() || salaryString.isEmpty()) {  
 JOptionPane.*showMessageDialog*(this, "Please fill in all required fields", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 return; // Exit method if any required field is empty  
 }  
  
 // Convert string dates to java.util.Date  
 Date dateOfBirthUtil = convertToDate(dateOfBirthString);  
 Date hireDateUtil = convertToDate(hireDateString);  
  
 // Check if date conversion is successful  
 if (dateOfBirthUtil == null || hireDateUtil == null) {  
 JOptionPane.*showMessageDialog*(this, "Invalid date format", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 return; // Exit method if date format is invalid  
 }  
  
 // Convert java.util.Date to java.sql.Date  
 java.sql.Date dateOfBirth = new java.sql.Date(dateOfBirthUtil.getTime());  
 java.sql.Date hireDate = new java.sql.Date(hireDateUtil.getTime());  
  
 // Convert string salary to double  
 double salary = Double.*parseDouble*(salaryString);  
  
 // Now create the Employee object  
 Employee employee = new Employee();  
 employee.setFirstName(firstName);  
 employee.setLastName(lastName);  
 employee.setPosition(position);  
 employee.setDateOfBirth(dateOfBirth);  
 employee.setContactInfo(contactInfo);  
 employee.setHireDate(hireDate);  
 employee.setSalary(salary);  
 // Set other fields as needed  
  
 // Now call the controller to add the employee  
 EmployeeController controller = new EmployeeController();  
 boolean success = controller.addEmployee(employee);  
  
 if (success) {  
 // Show success message  
 JOptionPane.*showMessageDialog*(this, "Employee added successfully");  
  
 // Clear input fields after successful addition  
 clearFields();  
 showUserTable();  
 loadEmployee();  
 showNhanVienTable();  
 // Refresh employee table  
 showNhanVienTable();  
 } else {  
 // Show error message  
 JOptionPane.*showMessageDialog*(this, "Failed to add employee", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
  
 // Phương thức để làm mới các trường nhập liệu sau khi thêm nhân viên thành công  
 private void clearFields() {  
 tbEmployeeID.setText("");  
 tbFirstName.setText("");  
 tbLastName.setText("");  
 tbPosition.setText("");  
 tbDateOfBirth.setText("");  
 tbContactInfo.setText("");  
 tbHireDate.setText("");  
 tbSalary.setText("");  
 }//GEN-LAST:event\_btnThemActionPerformed  
  
 private void btnLamRongActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnLamRongActionPerformed  
 // *TODO add your handling code here:* tbEmployeeID.setText("");  
 tbFirstName.setText("");  
 tbLastName.setText("");  
 tbPosition.setText("");  
 tbDateOfBirth.setText("");  
 tbContactInfo.setText("");  
 tbHireDate.setText("");  
 tbSalary.setText("");  
 }//GEN-LAST:event\_btnLamRongActionPerformed  
  
 private void btnXoaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnXoaActionPerformed  
 // *TODO add your handling code here:* // Fetch selected row index  
 int selectedRow = TableQuanLyNhanVien.getSelectedRow();  
  
 if (selectedRow != -1) {  
 // Fetch the employee ID from the selected row  
 int employeeID = (int) TableQuanLyNhanVien.getValueAt(selectedRow, 0);  
  
 // Confirm deletion  
 int confirm = JOptionPane.*showConfirmDialog*(this, "Are you sure you want to delete this employee?", "Confirm Delete", JOptionPane.*YES\_NO\_OPTION*);  
  
 if (confirm == JOptionPane.*YES\_OPTION*) {  
 EmployeeController controller = new EmployeeController();  
 boolean success = controller.deleteEmployee(employeeID);  
  
 if (success) {  
 // Show success message  
 JOptionPane.*showMessageDialog*(this, "Employee deleted successfully");  
  
 // Refresh employee table  
 showUserTable();  
 loadEmployee();  
 showNhanVienTable();  
 } else {  
 // Show error message  
 JOptionPane.*showMessageDialog*(this, "Failed to delete employee", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Please select an employee to delete");  
 }  
 }//GEN-LAST:event\_btnXoaActionPerformed  
  
 private void btnSuaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnSuaActionPerformed  
 // Retrieve modified data from UI input fields  
 int employeeID = Integer.*parseInt*(tbEmployeeID.getText()); // Assuming tbEmployeeID is a JTextField  
 String firstName = tbFirstName.getText();  
 String lastName = tbLastName.getText();  
 String position = tbPosition.getText();  
 String dateOfBirthStr = tbDateOfBirth.getText(); // Assuming tbDateOfBirth is a JTextField  
 String contactInfo = tbContactInfo.getText();  
 String hireDateStr = tbHireDate.getText(); // Assuming tbHireDate is a JTextField  
 double salary = Double.*parseDouble*(tbSalary.getText()); // Assuming tbSalary is a JTextField  
  
 // Check if any required field is empty  
 if (firstName.isEmpty() || lastName.isEmpty() || position.isEmpty() || dateOfBirthStr.isEmpty() || hireDateStr.isEmpty()) {  
 JOptionPane.*showMessageDialog*(this, "Please fill in all required fields", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 // Parse the date strings into java.sql.Date  
 java.sql.Date dateOfBirth;  
 java.sql.Date hireDate;  
 try {  
 dateOfBirth = java.sql.Date.*valueOf*(dateOfBirthStr);  
 hireDate = java.sql.Date.*valueOf*(hireDateStr);  
 } catch (IllegalArgumentException e) {  
 JOptionPane.*showMessageDialog*(this, "Invalid date format", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 return;  
 }  
  
 // Construct Employee object with updated data  
 Employee updatedEmployee = new Employee();  
 updatedEmployee.setEmployeeID(employeeID);  
 updatedEmployee.setFirstName(firstName);  
 updatedEmployee.setLastName(lastName);  
 updatedEmployee.setPosition(position);  
 updatedEmployee.setDateOfBirth(dateOfBirth);  
 updatedEmployee.setContactInfo(contactInfo);  
 updatedEmployee.setHireDate(hireDate);  
 updatedEmployee.setSalary(salary);  
  
 // Call controller method to update employee information  
 EmployeeController controller = new EmployeeController();  
 boolean success = controller.updateEmployee(updatedEmployee);  
  
 // Provide feedback to the user  
 if (success) {  
 JOptionPane.*showMessageDialog*(this, "Employee information updated successfully", "Success", JOptionPane.*INFORMATION\_MESSAGE*);  
 // Clear input fields or perform any other necessary actions  
 clearFields(); // Assuming this method clears the input fields  
 showUserTable(); // Refresh the table or any other necessary actions  
 loadEmployee();  
 showNhanVienTable();  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Failed to update employee information", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }//GEN-LAST:event\_btnSuaActionPerformed  
  
 private void tbDateOfBirthActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbDateOfBirthActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbDateOfBirthActionPerformed  
  
 private void tbHireDateActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbHireDateActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbHireDateActionPerformed  
  
 private void tbContactInfoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbContactInfoActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbContactInfoActionPerformed  
  
 private void tbSalaryActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbSalaryActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbSalaryActionPerformed  
  
 private void tbLastNameActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbLastNameActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbLastNameActionPerformed  
  
 private void tbPositionActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbPositionActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbPositionActionPerformed  
  
 private void tbFirstNameActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbFirstNameActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbFirstNameActionPerformed  
  
 private void tbEmployeeIDActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_tbEmployeeIDActionPerformed  
 // *TODO add your handling code here:* }//GEN-LAST:event\_tbEmployeeIDActionPerformed  
  
 private void TableQuanLyNhanVienMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_TableQuanLyNhanVienMouseClicked  
 // *TODO add your handling code here:* // Get the index of the selected row  
 int selectedRow = TableQuanLyNhanVien.getSelectedRow();  
  
 // Ensure a row is actually selected  
 if (selectedRow != -1) {  
 // Get the values from the selected row and populate the form fields  
 tbEmployeeID.setText((String) TableQuanLyNhanVien.getValueAt(selectedRow, 0).toString());  
 tbFirstName.setText((String) TableQuanLyNhanVien.getValueAt(selectedRow, 1));  
 tbLastName.setText((String) TableQuanLyNhanVien.getValueAt(selectedRow, 2));  
 tbPosition.setText((String) TableQuanLyNhanVien.getValueAt(selectedRow, 3));  
 tbDateOfBirth.setText(TableQuanLyNhanVien.getValueAt(selectedRow, 4).toString());  
 tbContactInfo.setText((String) TableQuanLyNhanVien.getValueAt(selectedRow, 5));  
 tbHireDate.setText(TableQuanLyNhanVien.getValueAt(selectedRow, 6).toString());  
 tbSalary.setText(TableQuanLyNhanVien.getValueAt(selectedRow, 7).toString());  
 // Optionally set other fields if present  
 }  
 }//GEN-LAST:event\_TableQuanLyNhanVienMouseClicked  
  
 private void btnDangXuatActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_btnDangXuatActionPerformed  
 // *TODO add your handling code here:* FormDangNhap loginForm = new FormDangNhap(); // Assuming LoginForm is your login screen  
 loginForm.setVisible(true);  
 this.dispose(); // Close the current window  
 }//GEN-LAST:event\_btnDangXuatActionPerformed  
  
 private void clearInputFields() {  
 // Clear the text fields and reset combo boxes after successful addition  
 tbUserName.setText("");  
 tbPasswordHash.setText("");  
 tbEmail.setText("");  
 tbUserID.setText("");  
 Role.setSelectedIndex(0); // Assuming the first item is a default or empty option  
 }  
  
 private void selectRole(String roleName) {  
 for (int i = 0; i < Role.getItemCount(); i++) {  
 String item = (String) Role.getItemAt(i);  
 if (item.equals(roleName)) {  
 Role.setSelectedItem(item);  
 break;  
 }  
 }  
 }  
  
 private void selectEmployee(String employeeName) {  
 for (int i = 0; i < Employee.getItemCount(); i++) {  
 String item = (String) Employee.getItemAt(i);  
 if (item.equals(employeeName)) {  
 Employee.setSelectedItem(item);  
 break;  
 }  
 }  
 }  
  
  
 */\*\*  
 \* @param args the command line arguments  
 \*/* public static void main(String args[]) {  
  
 /\* Set the Nimbus look and feel \*/  
 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  
 \*/  
 try {  
 for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.*getInstalledLookAndFeels*()) {  
 if ("Nimbus".equals(info.getName())) {  
 javax.swing.UIManager.*setLookAndFeel*(info.getClassName());  
 break;  
 }  
 }  
 } catch (ClassNotFoundException ex) {  
 java.util.logging.Logger.*getLogger*(FormNhanSu.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (InstantiationException ex) {  
 java.util.logging.Logger.*getLogger*(FormNhanSu.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (IllegalAccessException ex) {  
 java.util.logging.Logger.*getLogger*(FormNhanSu.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (javax.swing.UnsupportedLookAndFeelException ex) {  
 java.util.logging.Logger.*getLogger*(FormNhanSu.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 }  
 //</editor-fold>  
  
 /\* Create and display the form \*/  
 java.awt.EventQueue.*invokeLater*(new Runnable() {  
 public void run() {  
 new FormNhanSu().setVisible(true);  
 }  
 });  
 }  
  
 // Variables declaration - do not modify//GEN-BEGIN:variables  
 private javax.swing.JComboBox<String> Employee;  
 private javax.swing.JComboBox<String> Role;  
 private javax.swing.JTable TableQuanLyNhanVien;  
 private javax.swing.JTable TableUser;  
 private javax.swing.JButton btnDangXuat;  
 private javax.swing.JButton btnLamRong;  
 private javax.swing.JButton btnLamRong1;  
 private javax.swing.JButton btnSua;  
 private javax.swing.JButton btnSua1;  
 private javax.swing.JButton btnThem;  
 private javax.swing.JButton btnThem1;  
 private javax.swing.JButton btnXoa;  
 private javax.swing.JButton btnXoa1;  
 private javax.swing.JLabel jLabel1;  
 private javax.swing.JLabel jLabel10;  
 private javax.swing.JLabel jLabel11;  
 private javax.swing.JLabel jLabel12;  
 private javax.swing.JLabel jLabel13;  
 private javax.swing.JLabel jLabel14;  
 private javax.swing.JLabel jLabel15;  
 private javax.swing.JLabel jLabel2;  
 private javax.swing.JLabel jLabel3;  
 private javax.swing.JLabel jLabel4;  
 private javax.swing.JLabel jLabel5;  
 private javax.swing.JLabel jLabel6;  
 private javax.swing.JLabel jLabel7;  
 private javax.swing.JLabel jLabel8;  
 private javax.swing.JLabel jLabel9;  
 private javax.swing.JPanel jPanel1;  
 private javax.swing.JPanel jPanel2;  
 private javax.swing.JPanel jPanel3;  
 private javax.swing.JPanel jPmain;  
 private javax.swing.JPanel jPquanlynhanvien;  
 private javax.swing.JPanel jPquanlytaikhoan;  
 private javax.swing.JScrollPane jScrollPane1;  
 private javax.swing.JScrollPane jScrollPane2;  
 private javax.swing.JTabbedPane jtPmain;  
 private javax.swing.JTextField tbContactInfo;  
 private javax.swing.JTextField tbDateOfBirth;  
 private javax.swing.JTextField tbEmail;  
 private javax.swing.JTextField tbEmployeeID;  
 private javax.swing.JTextField tbFirstName;  
 private javax.swing.JTextField tbHireDate;  
 private javax.swing.JTextField tbLastName;  
 private javax.swing.JTextField tbPasswordHash;  
 private javax.swing.JTextField tbPosition;  
 private javax.swing.JTextField tbSalary;  
 private javax.swing.JTextField tbUserID;  
 private javax.swing.JTextField tbUserName;  
 // End of variables declaration//GEN-END:variables  
}

# II.THỰC THỂ

## 1. User

package org.example.models;  
  
import jakarta.persistence.Table;  
import lombok.Data;  
  
import java.sql.Date;  
@Data  
@Table(name = "users")  
public class User {  
 private int userID;  
 private String username;  
 private String passwordHash;  
 private String email;  
 private int employeeID;  
 private Date createdAt;  
  
  
 public User(String username, String passwordHash, String email) {  
 this.username = username;  
 this.passwordHash = passwordHash;  
 this.email = email;  
 }  
 // Constructors, getters, and setters  
}

## 2. UserRole

package org.example.models;  
  
import jakarta.persistence.Table;  
import lombok.Data;  
  
@Data  
@Table(name = "userroles")  
public class UserRole {  
 private int userRoleID;  
 private int userID;  
 private int roleID;  
  
 // Constructors, getters, and setters  
}

## 3. Role

package org.example.models;  
  
import jakarta.persistence.Table;  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Table(name = "roles")  
public class Role {  
 private int roleID;  
 private String roleName;  
 private String description;  
  
 // Constructors, getters, and setters  
}

## 4. Employee

package org.example.models;  
  
import jakarta.persistence.Table;  
import lombok.Data;  
  
import java.sql.Date;  
@Data  
@Table(name = "employees")  
public class Employee {  
 private int employeeID;  
 private String firstName;  
 private String lastName;  
 private String position;  
 private Date dateOfBirth;  
 private String contactInfo;  
 private Date hireDate;  
 private double salary;  
 private int createdBy;  
 private int updatedBy;  
 private Date updatedAt;  
  
 public String getFullName() {  
 return firstName + " " + lastName;  
 }  
 // Constructors, getters, and setters  
}

# III. ĐỐI TƯỢNG TRUYỀN DỮ LIỆU

## UserWithRoles

package org.example.DTO;  
  
import lombok.Data;  
import org.example.models.Role;  
import org.example.models.User;  
  
@Data  
public class UserWithRoles {  
  
 private User user;  
 private Role role;  
  
 public UserWithRoles(User user, Role role) {  
 this.user = user;  
 this.role = role;  
 }  
  
 public User getUser() {  
 return user;  
 }  
  
 public Role getRole() {  
 return role;  
 }  
}

# IV. CONTOLLER

## 1. UserController

package org.example.controllers;  
  
import org.example.connect.MyConnection;  
import org.example.models.Employee;  
import org.example.models.Role;  
import org.example.models.User;  
  
import javax.swing.table.DefaultTableModel;  
import java.sql.\*;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class UserController {  
 private Connection connection;  
  
 public UserController() {  
 // Initialize database connection  
 this.connection = MyConnection.*getConnection*();  
 }  
 private int loggedInUserID = -1; // Default value indicating no user is logged in  
 private Map<Integer, Integer> sessionUserMap = new HashMap<>(); // Map to store user IDs in session  
  
 public int getUserID(String username) {  
 int userID = -1; // Default value if user ID is not found  
  
 // Implement your logic to retrieve the user ID from the database based on the username  
 // This could involve executing a SQL query to fetch the user ID associated with the provided username  
 String query = "SELECT userID FROM users WHERE username = ?";  
 try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
 preparedStatement.setString(1, username);  
 ResultSet resultSet = preparedStatement.executeQuery();  
 if (resultSet.next()) {  
 userID = resultSet.getInt("userID");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 return userID;  
 }  
  
 public void setUserIDInSession(int userID) {  
 loggedInUserID = userID; // Set the logged-in user ID in the controller  
  
 // Also, store the user ID in the session map  
 sessionUserMap.put(userID, userID);  
 }  
  
 public boolean loginWithRole(String username, String password, String roleName) {  
 String query = "SELECT u.\*, r.\* FROM users u " +  
 "INNER JOIN userroles ur ON u.userID = ur.userID " +  
 "INNER JOIN roles r ON ur.roleID = r.roleID " +  
 "WHERE u.username = ? AND u.passwordHash = ? AND r.roleName = ?";  
 try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
 preparedStatement.setString(1, username);  
 preparedStatement.setString(2, password);  
 preparedStatement.setString(3, roleName);  
 ResultSet resultSet = preparedStatement.executeQuery();  
  
 if (resultSet.next()) {  
 // Đăng nhập thành công với quyền đã chọn  
 return true;  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return false; // Authentication failed  
 }  
  
 public Role findRoleByUserId(int userID) {  
 Role role = null;  
 String query = "SELECT r.\* FROM roles r " +  
 "INNER JOIN userroles ur ON r.roleID = ur.roleID " +  
 "WHERE ur.userID = ?";  
 try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
 preparedStatement.setInt(1, userID);  
 ResultSet resultSet = preparedStatement.executeQuery();  
 if (resultSet.next()) {  
 role = new Role();  
 role.setRoleID(resultSet.getInt("roleID"));  
 role.setRoleName(resultSet.getString("roleName"));  
 role.setDescription(resultSet.getString("description"));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return role;  
 }  
  
 public String findEmployeeByUserId(int userID) {  
 String fullName = null;  
 String query = "SELECT CONCAT(e.firstName, ' ', e.lastName) AS fullName " +  
 "FROM employees e " +  
 "INNER JOIN users u ON e.employeeID = u.employeeID " +  
 "WHERE u.userID = ?";  
 try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
 preparedStatement.setInt(1, userID);  
 ResultSet resultSet = preparedStatement.executeQuery();  
 if (resultSet.next()) {  
 fullName = resultSet.getString("fullName");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return fullName;  
 }  
  
  
  
 public List<Role> getAllRoles() {  
 List<Role> roles = new ArrayList<>();  
 String query = "SELECT \* FROM roles";  
 try (Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery(query)) {  
  
 while (resultSet.next()) {  
 Role role = new Role();  
 role.setRoleID(resultSet.getInt("roleID"));  
 role.setRoleName(resultSet.getString("roleName"));  
 role.setDescription(resultSet.getString("description"));  
 roles.add(role);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return roles;  
 }  
 public List<Employee> getAllEmployees() {  
 List<Employee> employees = new ArrayList<>();  
 String query = "SELECT \* FROM employees"; // Assuming your table name is "employees"  
  
 try (Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery(query)) {  
  
 while (resultSet.next()) {  
 Employee employee = new Employee();  
 employee.setEmployeeID(resultSet.getInt("employeeID"));  
 employee.setFirstName(resultSet.getString("firstName"));  
 employee.setLastName(resultSet.getString("lastName"));  
 // Set other properties as needed  
 employees.add(employee);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 return employees;  
 }  
  
  
 public DefaultTableModel getAllUserInfoTableModel() {  
 // Chuỗi truy vấn SQL để lấy thông tin  
 String query = "SELECT u.userID, u.username, u.passwordHash, u.email, e.firstName, e.lastName, e.position, r.roleName " +  
 "FROM users u " +  
 "INNER JOIN employees e ON u.employeeID = e.employeeID " +  
 "INNER JOIN userroles ur ON u.userID = ur.userID " +  
 "INNER JOIN roles r ON ur.roleID = r.roleID";  
  
 // Kết nối CSDL và thực hiện truy vấn  
 try (Connection connection = MyConnection.*getConnection*();  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery(query)) {  
  
 // Tạo một DefaultTableModel để lưu trữ dữ liệu  
 DefaultTableModel model = new DefaultTableModel(new String[]{"UserID", "Username", "PasswordHash", "Email", "FirstName", "LastName", "Position", "RoleName"}, 0);  
  
 // Duyệt qua kết quả của truy vấn và thêm vào DefaultTableModel  
 while (resultSet.next()) {  
 int userID = resultSet.getInt("userID");  
 String username = resultSet.getString("username");  
 String passwordHash = resultSet.getString("passwordHash");  
 String email = resultSet.getString("email");  
 String firstName = resultSet.getString("firstName");  
 String lastName = resultSet.getString("lastName");  
 String position = resultSet.getString("position");  
 String roleName = resultSet.getString("roleName");  
  
 // Thêm hàng mới vào DefaultTableModel  
 model.addRow(new Object[]{userID, username, passwordHash, email, firstName, lastName, position, roleName});  
 }  
  
 return model; // Trả về DefaultTableModel đã được điền dữ liệu  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return null; // Trả về null nếu có lỗi xảy ra  
 }  
 }  
  
 public void assignUserRole(int userID, int roleID) {  
 String query = "INSERT INTO userroles (userID, roleID) VALUES (?, ?)";  
 try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
 preparedStatement.setInt(1, userID);  
 preparedStatement.setInt(2, roleID);  
 preparedStatement.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public int findEmployeeIDByFullName(String fullName) {  
 int employeeID = -1; // Giá trị mặc định nếu không tìm thấy  
  
 String query = "SELECT employeeID FROM employees WHERE CONCAT(firstName, ' ', lastName) = ?";  
 try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
 preparedStatement.setString(1, fullName);  
 ResultSet resultSet = preparedStatement.executeQuery();  
 if (resultSet.next()) {  
 employeeID = resultSet.getInt("employeeID");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return employeeID;  
 }  
 public int findRoleIDByRoleName(String roleName) {  
 int roleID = -1; // Giá trị mặc định nếu không tìm thấy  
  
 String query = "SELECT roleID FROM roles WHERE roleName = ?";  
 try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
 preparedStatement.setString(1, roleName);  
 ResultSet resultSet = preparedStatement.executeQuery();  
 if (resultSet.next()) {  
 roleID = resultSet.getInt("roleID");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 return roleID;  
 }  
  
 public Role findRoleByName(String roleName) {  
 Role role = null;  
 String query = "SELECT \* FROM roles WHERE roleName = ?";  
 try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
 preparedStatement.setString(1, roleName);  
 ResultSet resultSet = preparedStatement.executeQuery();  
 if (resultSet.next()) {  
 role = new Role();  
 role.setRoleID(resultSet.getInt("roleID"));  
 role.setRoleName(resultSet.getString("roleName"));  
 role.setDescription(resultSet.getString("description"));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return role;  
 }  
  
 public String addUser(User user, String employeeName, int roleID) {  
 String checkUsernameQuery = "SELECT COUNT(\*) FROM users WHERE username = ?";  
 String checkEmailQuery = "SELECT COUNT(\*) FROM users WHERE email = ?";  
 String checkEmployeeIDQuery = "SELECT COUNT(\*) FROM users WHERE employeeID = " +  
 "(SELECT employeeID FROM employees WHERE CONCAT(firstName, ' ', lastName) = ?)";  
 String insertUserQuery = "INSERT INTO users (username, passwordHash, email, employeeID) " +  
 "SELECT ?, ?, ?, e.employeeID " +  
 "FROM employees e " +  
 "WHERE CONCAT(e.firstName, ' ', e.lastName) = ?";  
 String insertUserRoleQuery = "INSERT INTO userroles (userID, roleID) " +  
 "SELECT u.userID, r.roleID " +  
 "FROM users u " +  
 "INNER JOIN employees e ON u.employeeID = e.employeeID " +  
 "INNER JOIN roles r ON r.roleID = ? " +  
 "WHERE CONCAT(e.firstName, ' ', e.lastName) = ?";  
  
 try (PreparedStatement checkUsernameStmt = connection.prepareStatement(checkUsernameQuery);  
 PreparedStatement checkEmailStmt = connection.prepareStatement(checkEmailQuery);  
 PreparedStatement checkEmployeeIDStmt = connection.prepareStatement(checkEmployeeIDQuery);  
 PreparedStatement insertUserStmt = connection.prepareStatement(insertUserQuery);  
 PreparedStatement insertUserRoleStmt = connection.prepareStatement(insertUserRoleQuery)) {  
  
 // Kiểm tra xem tên người dùng đã tồn tại chưa  
 checkUsernameStmt.setString(1, user.getUsername());  
 ResultSet resultSetUsername = checkUsernameStmt.executeQuery();  
 if (resultSetUsername.next() && resultSetUsername.getInt(1) > 0) {  
 return "Username already exists.";  
 }  
  
 // Kiểm tra xem email đã tồn tại chưa  
 checkEmailStmt.setString(1, user.getEmail());  
 ResultSet resultSetEmail = checkEmailStmt.executeQuery();  
 if (resultSetEmail.next() && resultSetEmail.getInt(1) > 0) {  
 return "Email already exists.";  
 }  
  
 // Kiểm tra xem employeeID đã tồn tại chưa  
 checkEmployeeIDStmt.setString(1, employeeName);  
 ResultSet resultSetEmployeeID = checkEmployeeIDStmt.executeQuery();  
 if (resultSetEmployeeID.next() && resultSetEmployeeID.getInt(1) > 0) {  
 return "Employee ID already exists.";  
 }  
  
 // Đặt các tham số cho việc thêm người dùng  
 insertUserStmt.setString(1, user.getUsername());  
 insertUserStmt.setString(2, user.getPasswordHash());  
 insertUserStmt.setString(3, user.getEmail());  
 insertUserStmt.setString(4, employeeName);  
  
 // Thực thi câu lệnh thêm người dùng  
 int rowsAffectedUser = insertUserStmt.executeUpdate();  
  
 // Kiểm tra xem việc thêm người dùng có thành công không  
 if (rowsAffectedUser > 0) {  
 // Đặt các tham số cho việc thêm vai trò người dùng  
 insertUserRoleStmt.setInt(1, roleID);  
 insertUserRoleStmt.setString(2, employeeName);  
  
 // Thực thi câu lệnh thêm vai trò người dùng  
 int rowsAffectedUserRole = insertUserRoleStmt.executeUpdate();  
  
 // Kiểm tra xem việc thêm vai trò người dùng có thành công không  
 if (rowsAffectedUserRole > 0) {  
 return "Success";  
 }  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return "Failed to add user.";  
 }  
  
  
  
  
 public boolean deleteUser(int userID) {  
 String deleteUserRoleQuery = "DELETE FROM userroles WHERE userID = ?";  
 String deleteUserQuery = "DELETE FROM users WHERE userID = ?";  
  
 try (PreparedStatement preparedStatementUserRole = connection.prepareStatement(deleteUserRoleQuery);  
 PreparedStatement preparedStatementUser = connection.prepareStatement(deleteUserQuery)) {  
  
 // Xóa tất cả các bản ghi trong bảng userroles liên quan đến userID  
 preparedStatementUserRole.setInt(1, userID);  
 preparedStatementUserRole.executeUpdate();  
  
 // Sau đó mới xóa người dùng từ bảng users  
 preparedStatementUser.setInt(1, userID);  
 int rowsAffected = preparedStatementUser.executeUpdate();  
  
 return rowsAffected > 0;  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 }  
 }  
  
  
 public boolean updateUser(User user, String employeeName, int roleID) {  
 String updateUserQuery = "UPDATE users u " +  
 "INNER JOIN employees e ON u.employeeID = e.employeeID " +  
 "SET u.username = ?, u.passwordHash = ?, u.email = ? " +  
 "WHERE CONCAT(e.firstName, ' ', e.lastName) = ?";  
  
 String updateUserRoleQuery = "UPDATE userroles ur " +  
 "INNER JOIN users u ON ur.userID = u.userID " +  
 "SET ur.roleID = ? " +  
 "WHERE u.username = ?";  
  
 try (PreparedStatement preparedStatementUser = connection.prepareStatement(updateUserQuery);  
 PreparedStatement preparedStatementUserRole = connection.prepareStatement(updateUserRoleQuery)) {  
  
 // Set parameters for updating user  
 preparedStatementUser.setString(1, user.getUsername());  
 preparedStatementUser.setString(2, user.getPasswordHash());  
 preparedStatementUser.setString(3, user.getEmail());  
 preparedStatementUser.setString(4, employeeName);  
  
 // Execute user update query  
 int rowsAffectedUser = preparedStatementUser.executeUpdate();  
  
 // Check if user update was successful  
 if (rowsAffectedUser > 0) {  
 // Set parameters for updating user role  
 preparedStatementUserRole.setInt(1, roleID);  
 preparedStatementUserRole.setString(2, user.getUsername());  
  
 // Execute user role update query  
 int rowsAffectedUserRole = preparedStatementUserRole.executeUpdate();  
  
 // Check if user role update was successful  
 if (rowsAffectedUserRole > 0) {  
 return true;  
 }  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return false;  
 }  
  
  
 public Employee getByEmployeeIDWithUserID(int employeeID, int userID) {  
 Employee employee = null;  
  
 // Tạo truy vấn SQL để lấy thông tin Employee dựa trên EmployeeID và UserID  
 String query = "SELECT \* FROM employees WHERE EmployeeID = ? AND CreatedBy = ?";  
  
 try (Connection conn = MyConnection.*getConnection*();  
 PreparedStatement preparedStatement = conn.prepareStatement(query)) {  
  
 // Thiết lập giá trị cho các tham số trong truy vấn  
 preparedStatement.setInt(1, employeeID);  
 preparedStatement.setInt(2, userID);  
  
 // Thực thi truy vấn và lấy kết quả  
 ResultSet resultSet = preparedStatement.executeQuery();  
  
 // Kiểm tra xem có kết quả nào không  
 if (resultSet.next()) {  
 // Tạo một đối tượng Employee từ dữ liệu trong kết quả  
 employee = new Employee();  
 employee.setEmployeeID(resultSet.getInt("EmployeeID"));  
 employee.setFirstName(resultSet.getString("FirstName"));  
 employee.setLastName(resultSet.getString("LastName"));  
 // Các trường thông tin khác của Employee  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 return employee; // Trả về đối tượng Employee hoặc null nếu không tìm thấy  
 }  
 public int getEmployeeIDByUserID(int userID) {  
 int employeeID = -1; // Giá trị mặc định nếu không tìm thấy  
  
 String query = "SELECT employeeID FROM users WHERE userID = ?";  
 try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
 preparedStatement.setInt(1, userID);  
 ResultSet resultSet = preparedStatement.executeQuery();  
 if (resultSet.next()) {  
 employeeID = resultSet.getInt("employeeID");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 return employeeID;  
 }  
  
}

## 2. EmployeeController

package org.example.controllers;  
  
import java.sql.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
import org.example.connect.MyConnection;  
import org.example.models.Employee;  
import org.example.models.Role;  
import org.example.models.User;  
  
import javax.swing.table.DefaultTableModel;  
  
public class EmployeeController {  
 private Connection connection;  
  
 public EmployeeController() {  
 // Initialize database connection  
 this.connection = MyConnection.*getConnection*();  
 }  
 public List<Employee> getAllNhanVien() {  
 List<Employee> employeeList = new ArrayList<>();  
 String sql = "SELECT e.\*, r.roleID, r.roleName " +  
 "FROM employees e " +  
 "LEFT JOIN users u ON e.employeeID = u.employeeID " +  
 "LEFT JOIN userroles ur ON u.userID = ur.userID " +  
 "LEFT JOIN roles r ON ur.roleID = r.roleID";  
  
 try (Connection conn = MyConnection.*getConnection*();  
 PreparedStatement preparedStatement = conn.prepareStatement(sql);  
 ResultSet resultSet = preparedStatement.executeQuery()) {  
  
 while (resultSet.next()) {  
 Employee employee = new Employee();  
 // Lấy dữ liệu từ ResultSet và thiết lập cho đối tượng Employee  
 employee.setEmployeeID(resultSet.getInt("employeeID"));  
 employee.setFirstName(resultSet.getString("firstName"));  
 employee.setLastName(resultSet.getString("lastName"));  
 employee.setPosition(resultSet.getString("position"));  
 employee.setDateOfBirth(resultSet.getDate("dateOfBirth"));  
 employee.setContactInfo(resultSet.getString("contactInfo"));  
 employee.setHireDate(resultSet.getDate("hireDate"));  
 employee.setSalary(resultSet.getDouble("salary"));  
  
 // Tạo đối tượng Role từ dữ liệu cột roles  
 Role role = new Role();  
 role.setRoleID(resultSet.getInt("roleID"));  
 role.setRoleName(resultSet.getString("roleName"));  
 // Thêm thông tin về vai trò vào danh sách các vai trò của nhân viên  
 employeeList.add(employee);  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 return employeeList;  
 }  
  
  
 public DefaultTableModel getAllNhanVienTableModel() {  
 // Chuỗi truy vấn SQL để lấy thông tin với LEFT JOIN  
 String query = "SELECT e.employeeID, e.firstName, e.lastName, e.position, e.dateOfBirth, e.contactInfo, e.hireDate, e.salary, u.username, r.roleName " +  
 "FROM employees e " +  
 "LEFT JOIN users u ON e.employeeID = u.employeeID " +  
 "LEFT JOIN userroles ur ON u.userID = ur.userID " +  
 "LEFT JOIN roles r ON ur.roleID = r.roleID";  
  
 // Kết nối CSDL và thực hiện truy vấn  
 try (Connection connection = MyConnection.*getConnection*();  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery(query)) {  
  
 // Tạo một DefaultTableModel để lưu trữ dữ liệu  
 DefaultTableModel model = new DefaultTableModel(new String[]{"EmployeeID", "FirstName", "LastName", "Position", "DateOfBirth", "ContactInfo", "HireDate", "Salary", "Username", "RoleName"}, 0);  
  
 // Duyệt qua kết quả của truy vấn và thêm vào DefaultTableModel  
 while (resultSet.next()) {  
 int employeeID = resultSet.getInt("employeeID");  
 String firstName = resultSet.getString("firstName");  
 String lastName = resultSet.getString("lastName");  
 String position = resultSet.getString("position");  
 Date dateOfBirth = resultSet.getDate("dateOfBirth");  
 String contactInfo = resultSet.getString("contactInfo");  
 Date hireDate = resultSet.getDate("hireDate");  
 double salary = resultSet.getDouble("salary");  
 String username = resultSet.getString("username");  
 String roleName = resultSet.getString("roleName");  
  
 // Xử lý các giá trị null  
 if (username == null) {  
 username = "";  
 }  
 if (roleName == null) {  
 roleName = "";  
 }  
  
 // Thêm hàng mới vào DefaultTableModel  
 model.addRow(new Object[]{employeeID, firstName, lastName, position, dateOfBirth, contactInfo, hireDate, salary, username, roleName});  
 }  
  
 return model; // Trả về DefaultTableModel đã được điền dữ liệu  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return null; // Trả về null nếu có lỗi xảy ra  
 }  
 }  
  
  
  
  
  
 public boolean addEmployee(Employee employee) {  
 String query = "INSERT INTO employees (firstName, lastName, position, dateOfBirth, contactInfo, hireDate, salary) VALUES (?, ?, ?, ?, ?, ?, ?)";  
 try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
 preparedStatement.setString(1, employee.getFirstName());  
 preparedStatement.setString(2, employee.getLastName());  
 preparedStatement.setString(3, employee.getPosition());  
 preparedStatement.setDate(4, employee.getDateOfBirth());  
 preparedStatement.setString(5, employee.getContactInfo());  
 preparedStatement.setDate(6, employee.getHireDate());  
 preparedStatement.setDouble(7, employee.getSalary());  
  
 int rowsAffected = preparedStatement.executeUpdate();  
 return rowsAffected > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 }  
 }  
  
 public boolean updateEmployee(Employee employee) {  
 String query = "UPDATE employees SET firstName = ?, lastName = ?, position = ?, dateOfBirth = ?, contactInfo = ?, hireDate = ?, salary = ? WHERE employeeID = ?";  
 try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {  
 preparedStatement.setString(1, employee.getFirstName());  
 preparedStatement.setString(2, employee.getLastName());  
 preparedStatement.setString(3, employee.getPosition());  
 preparedStatement.setDate(4, employee.getDateOfBirth());  
 preparedStatement.setString(5, employee.getContactInfo());  
 preparedStatement.setDate(6, employee.getHireDate());  
 preparedStatement.setDouble(7, employee.getSalary());  
 preparedStatement.setInt(8, employee.getEmployeeID());  
  
 int rowsAffected = preparedStatement.executeUpdate();  
 return rowsAffected > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return false;  
 }  
 }  
  
 public boolean deleteEmployee(int employeeID) {  
 Connection connection = null;  
 PreparedStatement deleteRolesStatement = null;  
 PreparedStatement deleteUsersStatement = null;  
 PreparedStatement deleteEmployeeStatement = null;  
  
 try {  
 connection = MyConnection.*getConnection*();  
 connection.setAutoCommit(false); // Bắt đầu transaction  
  
 // Xóa các vai trò của người dùng trong bảng userroles  
 String deleteRolesQuery = "DELETE ur FROM userroles ur " +  
 "INNER JOIN users u ON ur.userID = u.userID " +  
 "WHERE u.employeeID = ?";  
 deleteRolesStatement = connection.prepareStatement(deleteRolesQuery);  
 deleteRolesStatement.setInt(1, employeeID);  
 deleteRolesStatement.executeUpdate();  
  
 // Xóa tài khoản người dùng trong bảng users  
 String deleteUsersQuery = "DELETE FROM users WHERE employeeID = ?";  
 deleteUsersStatement = connection.prepareStatement(deleteUsersQuery);  
 deleteUsersStatement.setInt(1, employeeID);  
 deleteUsersStatement.executeUpdate();  
  
 // Xóa thông tin nhân viên trong bảng employees  
 String deleteEmployeeQuery = "DELETE FROM employees WHERE employeeID = ?";  
 deleteEmployeeStatement = connection.prepareStatement(deleteEmployeeQuery);  
 deleteEmployeeStatement.setInt(1, employeeID);  
 int rowsAffected = deleteEmployeeStatement.executeUpdate();  
  
 connection.commit(); // Commit transaction  
 return rowsAffected > 0;  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 if (connection != null) {  
 try {  
 connection.rollback(); // Rollback transaction nếu có lỗi xảy ra  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
 return false;  
 } finally {  
 // Đóng các PreparedStatement và Connection  
 try {  
 if (deleteRolesStatement != null) deleteRolesStatement.close();  
 if (deleteUsersStatement != null) deleteUsersStatement.close();  
 if (deleteEmployeeStatement != null) deleteEmployeeStatement.close();  
 if (connection != null) connection.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
  
 public List<Employee> getAllEmployees() {  
 List<Employee> employees = new ArrayList<>();  
 String query = "SELECT \* FROM employees"; // Assuming your table name is "employees"  
  
 try (Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery(query)) {  
  
 while (resultSet.next()) {  
 Employee employee = new Employee();  
 employee.setEmployeeID(resultSet.getInt("employeeID"));  
 employee.setFirstName(resultSet.getString("firstName"));  
 employee.setLastName(resultSet.getString("lastName"));  
 // Set other properties as needed  
 employees.add(employee);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
  
 return employees;  
 }  
  
}

# V. CONNECT

## MyConnect

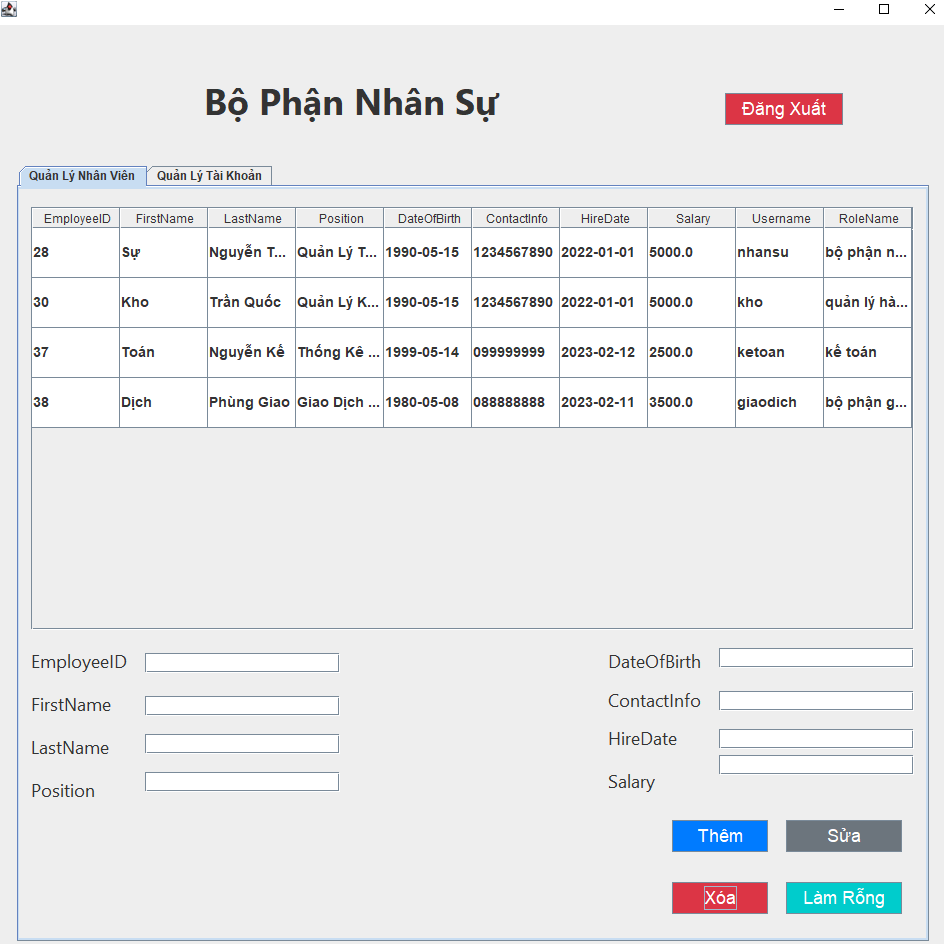
package org.example.connect;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class MyConnection {  
 private static final String *dbURL* = "jdbc:mysql://localhost:3306/mart";  
 private static final String *userName* = "root";  
 private static final String *password* = "";  
  
 public static Connection getConnection() {  
 Connection conn = null;  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 conn = DriverManager.*getConnection*(*dbURL*, *userName*, *password*);  
 } catch (ClassNotFoundException | SQLException e) {  
 e.printStackTrace();  
 }  
 return conn;  
 }  
}

# VI. KẾT QUẢ

## 1. Đăng nhập



## 2. Quản lý nhân viên



## 3. Quản lý tài khoản

