package chefdonburi;

import javax.swing.\*;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.JTableHeader;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import org.springframework.security.crypto.bcrypt.BCrypt;

public final class Users implements ActionListener {

private JFrame frmUserManagement;

private JTable userTable;

private DefaultTableModel model;

private JButton btnAdd, btnEdit, btnDelete, btnRefresh;

// Database connection variables

private Connection connection;

private PreparedStatement ps;

private ResultSet rs;

public Users() {

init();

}

void init() {

frmUserManagement = new JFrame("User Management");

ImageIcon frameicon = new ImageIcon("C:\\Users\\geramy\\Downloads\\images-20241029T152348Z-001\\images\\logochef.png");

Image frame = frameicon.getImage().getScaledInstance(100, 100, Image.SCALE\_SMOOTH);

frmUserManagement.setIconImage(frame);

frmUserManagement.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE); // Close only this window

frmUserManagement.setSize(800, 600);

frmUserManagement.setLocationRelativeTo(null);

// Table model

model = new DefaultTableModel(new String[]{"ID", "Username", "First Name", "Last Name", "Role"}, 0);

userTable = new JTable(model);

// Style the table header

JTableHeader tableHeader = userTable.getTableHeader();

tableHeader.setPreferredSize(new Dimension(tableHeader.getPreferredSize().width, 30));

tableHeader.setFont(new Font("Arial", Font.BOLD, 14));

tableHeader.setBackground(new Color(223, 49, 42));

tableHeader.setForeground(new Color(242, 245, 224));

// Style the table cells

userTable.setFont(new Font("Arial", Font.PLAIN, 14));

userTable.setRowHeight(30); // Adjust row height for better readability

DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();

centerRenderer.setHorizontalAlignment(SwingConstants.CENTER); // Center-align the text

for (int i = 0; i < userTable.getColumnCount(); i++) {

userTable.getColumnModel().getColumn(i).setCellRenderer(centerRenderer);

}

JScrollPane scrollPane = new JScrollPane(userTable);

frmUserManagement.add(scrollPane, BorderLayout.CENTER);

// Button panel

JPanel btnPanel = new JPanel();

btnAdd = new JButton("Add User");

btnAdd.setForeground(new Color(242, 245, 224));

btnAdd.setBackground(new Color(223, 49, 42));

btnAdd.setFont(new Font("Arial", Font.BOLD, 14));

btnAdd.addActionListener(this);

btnPanel.add(btnAdd);

btnEdit = new JButton("Edit User");

btnEdit.setForeground(new Color(242, 245, 224));

btnEdit.setBackground(new Color(223, 49, 42));

btnEdit.setFont(new Font("Arial", Font.BOLD, 14));

btnEdit.addActionListener(this);

btnPanel.add(btnEdit);

btnDelete = new JButton("Delete User");

btnDelete.setForeground(new Color(242, 245, 224));

btnDelete.setBackground(new Color(223, 49, 42));

btnDelete.setFont(new Font("Arial", Font.BOLD, 14));

btnDelete.addActionListener(this);

btnPanel.add(btnDelete);

btnRefresh = new JButton("Refresh");

btnRefresh.setForeground(new Color(242, 245, 224));

btnRefresh.setBackground(new Color(223, 49, 42));

btnRefresh.setFont(new Font("Arial", Font.BOLD, 14));

btnRefresh.addActionListener(this);

btnPanel.add(btnRefresh);

frmUserManagement.add(btnPanel, BorderLayout.SOUTH);

loadUserTable(); // Load users into the table

frmUserManagement.setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

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

addUser();

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

editUser();

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

deleteUser();

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

loadUserTable();

}

}

private void loadUserTable() {

model.setRowCount(0); // Clear existing rows

try {

connection = new Database().getConnection();

String query = "SELECT userID, userName, firstName, lastName, role FROM users";

ps = connection.prepareStatement(query);

rs = ps.executeQuery();

while (rs.next()) {

model.addRow(new Object[]{

rs.getInt("userID"),

rs.getString("userName"),

rs.getString("firstName"),

rs.getString("lastName"),

rs.getString("role")

});

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(frmUserManagement, "Error loading users: " + e.getMessage());

} finally {

closeConnections();

}

}

private void closeConnections() {

try {

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

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

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

} catch (SQLException e) {

JOptionPane.showMessageDialog(frmUserManagement, "Error closing resources: " + e.getMessage());

}

}

private void addUser() {

JTextField txtUserName = new JTextField();

JTextField txtFirstName = new JTextField();

JTextField txtLastName = new JTextField();

JComboBox<String> cmbRole = new JComboBox<>(new String[]{"Admin", "User"});

JPasswordField txtPassword = new JPasswordField();

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(6, 2)); // Keep the form layout simple

panel.add(new JLabel("Username:"));

panel.add(txtUserName);

panel.add(new JLabel("First Name:"));

panel.add(txtFirstName);

panel.add(new JLabel("Last Name:"));

panel.add(txtLastName);

panel.add(new JLabel("Role:"));

panel.add(cmbRole);

panel.add(new JLabel("Password:"));

panel.add(txtPassword);

int option = JOptionPane.showConfirmDialog(frmUserManagement, panel, "Add User", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION) {

if (txtUserName.getText().trim().isEmpty() ||

txtFirstName.getText().trim().isEmpty() ||

txtLastName.getText().trim().isEmpty() ||

txtPassword.getPassword().length == 0) {

JOptionPane.showMessageDialog(frmUserManagement, "All fields are required.", "Validation Error", JOptionPane.ERROR\_MESSAGE);

return;

}

// Check if the username already exists

if (isUsernameTaken(txtUserName.getText())) {

JOptionPane.showMessageDialog(frmUserManagement, "This username is already taken.", "Validation Error", JOptionPane.ERROR\_MESSAGE);

return;

}

String hashedPassword = BCrypt.hashpw(new String(txtPassword.getPassword()), BCrypt.gensalt()); // Hash the password

try {

connection = new Database().getConnection();

String query = "INSERT INTO users (userName, firstName, lastName, role, userPassword) VALUES (?, ?, ?, ?, ?)";

ps = connection.prepareStatement(query);

ps.setString(1, txtUserName.getText());

ps.setString(2, txtFirstName.getText());

ps.setString(3, txtLastName.getText());

ps.setString(4, cmbRole.getSelectedItem().toString());

ps.setString(5, hashedPassword); // Store hashed password

ps.executeUpdate();

JOptionPane.showMessageDialog(frmUserManagement, "User added successfully.");

loadUserTable();

} catch (SQLException e) {

JOptionPane.showMessageDialog(frmUserManagement, "Error adding user: " + e.getMessage());

} finally {

closeConnections();

}

}

}

private boolean isUsernameTaken(String username) {

try {

connection = new Database().getConnection();

String query = "SELECT COUNT(\*) FROM users WHERE userName = ?";

ps = connection.prepareStatement(query);

ps.setString(1, username);

rs = ps.executeQuery();

if (rs.next()) {

return rs.getInt(1) > 0; // If count > 0, username is taken

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(frmUserManagement, "Error checking username: " + e.getMessage());

} finally {

closeConnections();

}

return false;

}

private void editUser() {

int selectedRow = userTable.getSelectedRow();

if (selectedRow >= 0) {

int userId = (int) model.getValueAt(selectedRow, 0);

String currentUserName = (String) model.getValueAt(selectedRow, 1);

String currentFirstName = (String) model.getValueAt(selectedRow, 2);

String currentLastName = (String) model.getValueAt(selectedRow, 3);

String currentRole = (String) model.getValueAt(selectedRow, 4);

JTextField txtUserName = new JTextField(currentUserName);

JTextField txtFirstName = new JTextField(currentFirstName);

JTextField txtLastName = new JTextField(currentLastName);

JComboBox<String> cmbRole = new JComboBox<>(new String[]{"Admin", "User"});

cmbRole.setSelectedItem(currentRole);

JPasswordField txtCurrentPassword = new JPasswordField();

JPasswordField txtNewPassword = new JPasswordField();

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(8, 2)); // Layout for the form

panel.add(new JLabel("Username:"));

panel.add(txtUserName);

panel.add(new JLabel("First Name:"));

panel.add(txtFirstName);

panel.add(new JLabel("Last Name:"));

panel.add(txtLastName);

panel.add(new JLabel("Role:"));

panel.add(cmbRole);

panel.add(new JLabel("Current Password:"));

panel.add(txtCurrentPassword);

panel.add(new JLabel("New Password (Leave blank to keep unchanged):"));

panel.add(txtNewPassword);

int option = JOptionPane.showConfirmDialog(frmUserManagement, panel, "Edit User", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION) {

String newUserName = txtUserName.getText().trim();

// Check if the username is taken

if (!newUserName.equals(currentUserName) && isUsernameTaken(newUserName)) {

JOptionPane.showMessageDialog(frmUserManagement, "This username is already taken.", "Validation Error", JOptionPane.ERROR\_MESSAGE);

return;

}

// Handle password update and user data change logic...

}

} else {

JOptionPane.showMessageDialog(frmUserManagement, "Please select a user to edit.");

}

}

private void deleteUser() {

int selectedRow = userTable.getSelectedRow();

if (selectedRow >= 0) {

int userId = (int) model.getValueAt(selectedRow, 0);

try {

connection = new Database().getConnection();

String query = "DELETE FROM users WHERE userID = ?";

ps = connection.prepareStatement(query);

ps.setInt(1, userId);

ps.executeUpdate();

JOptionPane.showMessageDialog(frmUserManagement, "User deleted successfully.");

loadUserTable();

} catch (SQLException e) {

JOptionPane.showMessageDialog(frmUserManagement, "Error deleting user: " + e.getMessage());

} finally {

closeConnections();

}

} else {

JOptionPane.showMessageDialog(frmUserManagement, "Please select a user to delete.");

}

}

}