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 java.text.SimpleDateFormat;

import java.util.Date;

public final class Suppliers implements ActionListener {

private JFrame frmSuppliersManagement;

private JTable suppliersTable;

private DefaultTableModel model;

private JButton btnAdd, btnEdit, btnDelete, btnRefresh;

private JLabel lblDate;

public Suppliers() {

init();

}

public void init() {

frmSuppliersManagement = new JFrame("Suppliers Management");

ImageIcon frameicon = new ImageIcon("src\\Images\\jframeicon.jpg");

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

frmSuppliersManagement.setIconImage(frame);

frmSuppliersManagement.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

frmSuppliersManagement.setSize(1100, 700);

frmSuppliersManagement.setLocationRelativeTo(null);

// Create a panel for the date label

JPanel datePanel = new JPanel(new BorderLayout());

lblDate = new JLabel(getCurrentDate(), SwingConstants.RIGHT);

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

lblDate.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 20)); // Add margin

datePanel.add(lblDate, BorderLayout.EAST);

frmSuppliersManagement.add(datePanel, BorderLayout.NORTH);

// Table model setup

model = new DefaultTableModel(new String[]{"ID", "Supplier Name", "Description", "Address", "Contact Number", "Store Link"}, 0);

suppliersTable = new JTable(model);

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

suppliersTable.setRowHeight(25);

// Set header styling

JTableHeader tableHeader = suppliersTable.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));

// Center-align table values

DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();

centerRenderer.setHorizontalAlignment(SwingConstants.CENTER);

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

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

}

JScrollPane scrollPane = new JScrollPane(suppliersTable);

frmSuppliersManagement.add(scrollPane, BorderLayout.CENTER);

// Button panel setup

JPanel btnPanel = new JPanel();

btnAdd = new JButton("Add Supplier");

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 Supplier");

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 Supplier");

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);

frmSuppliersManagement.add(btnPanel, BorderLayout.SOUTH);

loadSuppliersTable(); // Load suppliers into the table

frmSuppliersManagement.setVisible(true); // Show frame

}

private String getCurrentDate() {

SimpleDateFormat formatter = new SimpleDateFormat("MMMM dd, yyyy");

return formatter.format(new Date());

}

@Override

public void actionPerformed(ActionEvent e) {

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

addSupplier();

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

editSupplier();

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

deleteSupplier();

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

loadSuppliersTable();

}

}

private void loadSuppliersTable() {

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

try (Connection connection = new Database().getConnection()) {

String query = "SELECT SupplierID, SupplierName, Description, Address, ContactNumber, StoreLink FROM suppliers";

try (PreparedStatement ps = connection.prepareStatement(query);

ResultSet rs = ps.executeQuery()) {

while (rs.next()) {

model.addRow(new Object[]{

rs.getInt("supplierID"),

rs.getString("supplierName"),

rs.getString("description"),

rs.getString("address"),

rs.getString("contactNumber"),

rs.getString("storeLink")

});

}

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(frmSuppliersManagement, "Error loading suppliers: " + e.getMessage());

}

}

private void addSupplier() {

JTextField txtSupplierName = new JTextField();

JTextField txtDescription = new JTextField();

JTextField txtAddress = new JTextField();

JTextField txtContactNumber = new JTextField();

JTextField txtStoreLink = new JTextField();

Object[] message = {

"Supplier Name:", txtSupplierName,

"Description:", txtDescription,

"Address:", txtAddress,

"Contact Number:", txtContactNumber,

"Store Link:", txtStoreLink

};

int option = JOptionPane.showConfirmDialog(frmSuppliersManagement, message, "Add Supplier", JOptionPane.OK\_CANCEL\_OPTION);

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

if (txtSupplierName.getText().trim().isEmpty() || txtAddress.getText().trim().isEmpty() || txtContactNumber.getText().trim().isEmpty()) {

JOptionPane.showMessageDialog(frmSuppliersManagement, "Required fields cannot be empty.", "Validation Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection connection = new Database().getConnection()) {

String query = "INSERT INTO suppliers (SupplierName, Description, Address, ContactNumber, StoreLink) VALUES (?, ?, ?, ?, ?)";

try (PreparedStatement ps = connection.prepareStatement(query)) {

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

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

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

ps.setString(4, txtContactNumber.getText());

ps.setString(5, txtStoreLink.getText());

ps.executeUpdate();

JOptionPane.showMessageDialog(frmSuppliersManagement, "Supplier added successfully.");

loadSuppliersTable();

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(frmSuppliersManagement, "Error adding supplier: " + e.getMessage());

}

}

}

private void editSupplier() {

int selectedRow = suppliersTable.getSelectedRow();

if (selectedRow >= 0) {

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

JTextField txtSupplierName = new JTextField((String) model.getValueAt(selectedRow, 1));

JTextField txtDescription = new JTextField((String) model.getValueAt(selectedRow, 2));

JTextField txtAddress = new JTextField((String) model.getValueAt(selectedRow, 3));

JTextField txtContactNumber = new JTextField((String) model.getValueAt(selectedRow, 4));

JTextField txtStoreLink = new JTextField((String) model.getValueAt(selectedRow, 5));

Object[] message = {

"Supplier Name:", txtSupplierName,

"Description:", txtDescription,

"Address:", txtAddress,

"Contact Number:", txtContactNumber,

"Store Link:", txtStoreLink

};

int option = JOptionPane.showConfirmDialog(frmSuppliersManagement, message, "Edit Supplier", JOptionPane.OK\_CANCEL\_OPTION);

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

if (txtSupplierName.getText().trim().isEmpty() || txtAddress.getText().trim().isEmpty() || txtContactNumber.getText().trim().isEmpty()) {

JOptionPane.showMessageDialog(frmSuppliersManagement, "Required fields cannot be empty.", "Validation Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection connection = new Database().getConnection()) {

String query = "UPDATE suppliers SET SupplierName = ?, Description = ?, Address = ?, ContactNumber = ?, StoreLink = ? WHERE SupplierID = ?";

try (PreparedStatement ps = connection.prepareStatement(query)) {

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

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

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

ps.setString(4, txtContactNumber.getText());

ps.setString(5, txtStoreLink.getText());

ps.setInt(6, supplierId);

ps.executeUpdate();

JOptionPane.showMessageDialog(frmSuppliersManagement, "Supplier updated successfully.");

loadSuppliersTable();

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(frmSuppliersManagement, "Error updating supplier: " + e.getMessage());

}

}

} else {

JOptionPane.showMessageDialog(frmSuppliersManagement, "Please select a supplier to edit.");

}

}

private void deleteSupplier() {

int selectedRow = suppliersTable.getSelectedRow();

if (selectedRow >= 0) {

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

int option = JOptionPane.showConfirmDialog(frmSuppliersManagement, "Are you sure you want to delete this supplier?", "Delete Supplier", JOptionPane.YES\_NO\_OPTION);

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

try (Connection connection = new Database().getConnection()) {

String query = "DELETE FROM suppliers WHERE SupplierID = ?";

try (PreparedStatement ps = connection.prepareStatement(query)) {

ps.setInt(1, supplierId);

ps.executeUpdate();

JOptionPane.showMessageDialog(frmSuppliersManagement, "Supplier deleted successfully.");

loadSuppliersTable();

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(frmSuppliersManagement, "Error deleting supplier: " + e.getMessage());

}

}

} else {

JOptionPane.showMessageDialog(frmSuppliersManagement, "Please select a supplier to delete.");

}

}

public static void main(String[] args) {

Suppliers suppliers = new Suppliers();

}

}