package chefdonburi;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.Image;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

import javax.swing.JOptionPane;

import java.text.SimpleDateFormat;

import javax.swing.ImageIcon;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.JTableHeader;

public class Logs {

private JFrame frame;

private JTable table;

private DefaultTableModel model;

private JButton btnRefresh, btnDelete;

// Database connection variables

private Connection connection;

private PreparedStatement ps;

private ResultSet rs;

public Logs() {

// Initialize the Logs window

init();

}

private void init() {

// Set up frame icon

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

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

frame = new JFrame("User Activity Logs");

frame.setIconImage(jframe);

frame.setSize(1100, 700);

frame.setLocationRelativeTo(null);

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

// Create a panel with BorderLayout to manage the layout more easily

JPanel panel = new JPanel(new BorderLayout());

// Create the table to display logs

String[] columns = {"Log ID", "Username", "Login Time", "Logout Time"};

model = new DefaultTableModel(columns, 0); // Initialize table with no data

table = new JTable(model);

// Set table font and row height

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

table.setRowHeight(30);

// Center align table values

DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();

centerRenderer.setHorizontalAlignment(DefaultTableCellRenderer.CENTER);

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

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

}

// Set header font

JTableHeader tableHeader = table.getTableHeader();

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

tableHeader.setFont(new Font("Arial", Font.BOLD, 14)); // Bold Arial for header

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

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

JScrollPane scrollPane = new JScrollPane(table);

panel.add(scrollPane, BorderLayout.CENTER); // Add table to the center of the panel

// Create buttons: Refresh and Delete

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));

btnDelete = new JButton("Delete");

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

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

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

// Add buttons to a button panel

JPanel btnPanel = new JPanel();

btnPanel.add(btnRefresh);

btnPanel.add(btnDelete);

panel.add(btnPanel, BorderLayout.SOUTH); // Add the button panel to the bottom of the main panel

// Add the panel to the frame and make the frame visible

frame.add(panel);

frame.setVisible(true);

// Add action listeners for the buttons

btnRefresh.addActionListener(e -> loadLogs(model));

btnDelete.addActionListener(e -> deleteLog());

// Load logs initially

loadLogs(model);

}

// Load the logs from the database

private void loadLogs(DefaultTableModel model) {

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

try {

connection = new Database().getConnection(); // Assume Database is a class that handles DB connection

String query = """

SELECT logs.logID, users.username, logs.loginTime, logs.logoutTime

FROM logs

JOIN users ON logs.userID = users.userID

""";

ps = connection.prepareStatement(query);

rs = ps.executeQuery();

SimpleDateFormat dateFormat = new SimpleDateFormat("MMMM dd, yyyy HH:mm:ss");

while (rs.next()) {

// Format the login and logout times

String loginTime = rs.getTimestamp("loginTime") != null

? dateFormat.format(rs.getTimestamp("loginTime"))

: "N/A";

String logoutTime = rs.getTimestamp("logoutTime") != null

? dateFormat.format(rs.getTimestamp("logoutTime"))

: "N/A";

model.addRow(new Object[]{

rs.getInt("logID"),

rs.getString("username"), // Display the username

loginTime, // Formatted login time

logoutTime // Formatted logout time

});

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(frame, "Error loading logs: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

} finally {

closeConnections();

}

}

// Delete the selected log from the database

private void deleteLog() {

int selectedRow = table.getSelectedRow(); // Get the selected row

if (selectedRow >= 0) {

int logId = (int) model.getValueAt(selectedRow, 0); // Get the Log ID from the selected row

int confirm = JOptionPane.showConfirmDialog(frame, "Are you sure you want to delete this log?", "Confirm Delete", JOptionPane.YES\_NO\_OPTION);

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

try {

connection = new Database().getConnection(); // Get database connection

String query = "DELETE FROM logs WHERE logID = ?";

ps = connection.prepareStatement(query);

ps.setInt(1, logId); // Set the Log ID in the query

int rowsAffected = ps.executeUpdate();

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(frame, "Log deleted successfully.");

loadLogs(model); // Refresh the table

} else {

JOptionPane.showMessageDialog(frame, "Failed to delete the log.", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException e) {

JOptionPane.showMessageDialog(frame, "Error deleting log: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

} finally {

closeConnections();

}

}

} else {

JOptionPane.showMessageDialog(frame, "Please select a log to delete.", "No Selection", JOptionPane.WARNING\_MESSAGE);

}

}

private void closeConnections() {

try {

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

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

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

} catch (SQLException e) {

JOptionPane.showMessageDialog(frame, "Error closing database connection: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

public static void main(String[] args) {

Logs logs = new Logs();

}

}