**LoginSystem.java**  
package handson1;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

public class LoginSystem extends JFrame {

private JTextField usernameField;

private JPasswordField passwordField;

public LoginSystem() {

setTitle("Login System");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 400, 300);

getContentPane().setLayout(null);

JLabel lblUsername = new JLabel("Username:");

lblUsername.setBounds(50, 50, 100, 25);

getContentPane().add(lblUsername);

usernameField = new JTextField();

usernameField.setBounds(150, 50, 200, 25);

getContentPane().add(usernameField);

JLabel lblPassword = new JLabel("Password:");

lblPassword.setBounds(50, 100, 100, 25);

getContentPane().add(lblPassword);

passwordField = new JPasswordField();

passwordField.setBounds(150, 100, 200, 25);

getContentPane().add(passwordField);

JButton btnLogin = new JButton("Login");

btnLogin.setBounds(150, 150, 100, 30);

getContentPane().add(btnLogin);

btnLogin.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

validateCredentials();

}

});

}

private void validateCredentials() {

String username = usernameField.getText().trim();

String password = new String(passwordField.getPassword()).trim();

if (username.isEmpty() || password.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please enter both username and password.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

Connection conn = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/user1", "root", "0000");

String query = "SELECT \* FROM users WHERE username = ? AND password = ?";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, username);

pstmt.setString(2, password);

System.out.println(pstmt.toString());

ResultSet rs = pstmt.executeQuery();

if (rs.next()) {

JOptionPane.showMessageDialog(this, "Login successful!", "Success", JOptionPane.INFORMATION\_MESSAGE);

navigateToDashboard();

} else {

JOptionPane.showMessageDialog(this, "Invalid username or password.", "Error", JOptionPane.ERROR\_MESSAGE);

}

conn.close();

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Database error: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void navigateToDashboard() {

this.dispose(); // Close the login frame

JFrame dashboard = new JFrame("Dashboard");

dashboard.setBounds(100, 100, 400, 300);

dashboard.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JLabel welcomeLabel = new JLabel("Welcome to the Dashboard!");

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

welcomeLabel.setHorizontalAlignment(SwingConstants.CENTER);

welcomeLabel.setBounds(50, 100, 300, 50);

dashboard.getContentPane().setLayout(null);

dashboard.getContentPane().add(welcomeLabel);

dashboard.setVisible(true);

}

public static void main(String[] args) {

EventQueue.invokeLater(() -> {

try {

LoginSystem frame = new LoginSystem();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

});

}

}

**StudentManagement.java**

package handson2;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class StudentManagementSystem extends JFrame {

private JTextField nameField, rollField, courseField;

private JTable studentTable;

private DefaultTableModel tableModel;

public StudentManagementSystem() {

setTitle("Student Management System");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 600, 400);

getContentPane().setLayout(null);

JLabel lblName = new JLabel("Name:");

lblName.setBounds(20, 20, 100, 25);

getContentPane().add(lblName);

nameField = new JTextField();

nameField.setBounds(100, 20, 150, 25);

getContentPane().add(nameField);

JLabel lblRoll = new JLabel("Roll No:");

lblRoll.setBounds(20, 60, 100, 25);

getContentPane().add(lblRoll);

rollField = new JTextField();

rollField.setBounds(100, 60, 150, 25);

getContentPane().add(rollField);

JLabel lblCourse = new JLabel("Course:");

lblCourse.setBounds(20, 100, 100, 25);

getContentPane().add(lblCourse);

courseField = new JTextField();

courseField.setBounds(100, 100, 150, 25);

getContentPane().add(courseField);

JButton addButton = new JButton("Add");

addButton.setBounds(20, 140, 100, 30);

getContentPane().add(addButton);

JButton updateButton = new JButton("Update");

updateButton.setBounds(130, 140, 100, 30);

getContentPane().add(updateButton);

JButton deleteButton = new JButton("Delete");

deleteButton.setBounds(240, 140, 100, 30);

getContentPane().add(deleteButton);

JButton refreshButton = new JButton("Refresh");

refreshButton.setBounds(350, 140, 100, 30);

getContentPane().add(refreshButton);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 180, 540, 150);

getContentPane().add(scrollPane);

studentTable = new JTable();

tableModel = new DefaultTableModel(new Object[]{"ID", "Name", "Roll No", "Course"}, 0);

studentTable.setModel(tableModel);

scrollPane.setViewportView(studentTable);

// Button actions

addButton.addActionListener(e -> addStudent());

updateButton.addActionListener(e -> updateStudent());

deleteButton.addActionListener(e -> deleteStudent());

refreshButton.addActionListener(e -> loadStudentData());

// Load student data on startup

loadStudentData();

}

private void addStudent() {

String name = nameField.getText().trim();

String rollNo = rollField.getText().trim();

String course = courseField.getText().trim();

if (name.isEmpty() || rollNo.isEmpty() || course.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill all fields.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = getConnection()) {

String query = "INSERT INTO students (name, roll\_no, course) VALUES (?, ?, ?)";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, name);

pstmt.setString(2, rollNo);

pstmt.setString(3, course);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Student added successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

loadStudentData();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error adding student: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void updateStudent() {

int selectedRow = studentTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a student to update.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

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

String name = nameField.getText().trim();

String rollNo = rollField.getText().trim();

String course = courseField.getText().trim();

if (name.isEmpty() || rollNo.isEmpty() || course.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill all fields.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = getConnection()) {

String query = "UPDATE students SET name = ?, roll\_no = ?, course = ? WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, name);

pstmt.setString(2, rollNo);

pstmt.setString(3, course);

pstmt.setString(4, id);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Student updated successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

loadStudentData();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error updating student: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void deleteStudent() {

int selectedRow = studentTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a student to delete.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

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

try (Connection conn = getConnection()) {

String query = "DELETE FROM students WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, id);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Student deleted successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

loadStudentData();

} catch (SQLException e) {

e.printStackTrace();

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

}

}

private void loadStudentData() {

tableModel.setRowCount(0);

try (Connection conn = getConnection()) {

String query = "SELECT \* FROM students";

PreparedStatement pstmt = conn.prepareStatement(query);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

tableModel.addRow(new Object[]{

rs.getInt("id"),

rs.getString("name"),

rs.getString("roll\_no"),

rs.getString("course")

});

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading student data: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private Connection getConnection() throws SQLException {

return DriverManager.getConnection(

"jdbc:mysql://localhost:3306/StudentManagement", "root", "0000");

}

public static void main(String[] args) {

EventQueue.invokeLater(() -> {

try {

StudentManagementSystem frame = new StudentManagementSystem();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

});

}

}

**LibraryInventorySystem.java**

package handson3;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class LibraryInventorySystem extends JFrame {

private JTextField titleField, authorField, isbnField, quantityField, searchField;

private JTable bookTable;

private DefaultTableModel tableModel;

public LibraryInventorySystem() {

setTitle("Library Book Inventory System");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 800, 500);

getContentPane().setLayout(null);

JLabel lblTitle = new JLabel("Title:");

lblTitle.setBounds(20, 20, 100, 25);

getContentPane().add(lblTitle);

titleField = new JTextField();

titleField.setBounds(80, 20, 150, 25);

getContentPane().add(titleField);

JLabel lblAuthor = new JLabel("Author:");

lblAuthor.setBounds(250, 20, 100, 25);

getContentPane().add(lblAuthor);

authorField = new JTextField();

authorField.setBounds(310, 20, 150, 25);

getContentPane().add(authorField);

JLabel lblISBN = new JLabel("ISBN:");

lblISBN.setBounds(20, 60, 100, 25);

getContentPane().add(lblISBN);

isbnField = new JTextField();

isbnField.setBounds(80, 60, 150, 25);

getContentPane().add(isbnField);

JLabel lblQuantity = new JLabel("Quantity:");

lblQuantity.setBounds(250, 60, 100, 25);

getContentPane().add(lblQuantity);

quantityField = new JTextField();

quantityField.setBounds(310, 60, 150, 25);

getContentPane().add(quantityField);

JButton addButton = new JButton("Add");

addButton.setBounds(480, 20, 100, 30);

getContentPane().add(addButton);

JButton updateButton = new JButton("Update");

updateButton.setBounds(480, 60, 100, 30);

getContentPane().add(updateButton);

JButton deleteButton = new JButton("Delete");

deleteButton.setBounds(590, 20, 100, 30);

getContentPane().add(deleteButton);

JButton searchButton = new JButton("Search");

searchButton.setBounds(590, 60, 100, 30);

getContentPane().add(searchButton);

JLabel lblSearch = new JLabel("Search:");

lblSearch.setBounds(20, 100, 100, 25);

getContentPane().add(lblSearch);

searchField = new JTextField();

searchField.setBounds(80, 100, 380, 25);

getContentPane().add(searchField);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 140, 750, 300);

getContentPane().add(scrollPane);

bookTable = new JTable();

tableModel = new DefaultTableModel(new Object[]{"ID", "Title", "Author", "ISBN", "Quantity"}, 0);

bookTable.setModel(tableModel);

scrollPane.setViewportView(bookTable);

// Button actions

addButton.addActionListener(e -> addBook());

updateButton.addActionListener(e -> updateBook());

deleteButton.addActionListener(e -> deleteBook());

searchButton.addActionListener(e -> searchBooks());

// Load book data on startup

loadBookData();

}

private void addBook() {

String title = titleField.getText().trim();

String author = authorField.getText().trim();

String isbn = isbnField.getText().trim();

String quantity = quantityField.getText().trim();

if (title.isEmpty() || author.isEmpty() || isbn.isEmpty() || quantity.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill all fields.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = getConnection()) {

String query = "INSERT INTO books (title, author, isbn, quantity) VALUES (?, ?, ?, ?)";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, title);

pstmt.setString(2, author);

pstmt.setString(3, isbn);

pstmt.setInt(4, Integer.parseInt(quantity));

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Book added successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

loadBookData();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error adding book: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void updateBook() {

int selectedRow = bookTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a book to update.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

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

String title = titleField.getText().trim();

String author = authorField.getText().trim();

String isbn = isbnField.getText().trim();

String quantity = quantityField.getText().trim();

if (title.isEmpty() || author.isEmpty() || isbn.isEmpty() || quantity.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill all fields.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = getConnection()) {

String query = "UPDATE books SET title = ?, author = ?, isbn = ?, quantity = ? WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, title);

pstmt.setString(2, author);

pstmt.setString(3, isbn);

pstmt.setInt(4, Integer.parseInt(quantity));

pstmt.setInt(5, Integer.parseInt(id));

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Book updated successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

loadBookData();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error updating book: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void deleteBook() {

int selectedRow = bookTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a book to delete.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

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

try (Connection conn = getConnection()) {

String query = "DELETE FROM books WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(query);

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

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Book deleted successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

loadBookData();

} catch (SQLException e) {

e.printStackTrace();

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

}

}

private void searchBooks() {

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

tableModel.setRowCount(0);

try (Connection conn = getConnection()) {

String query = "SELECT \* FROM books WHERE title LIKE ? OR author LIKE ?";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, "%" + searchText + "%");

pstmt.setString(2, "%" + searchText + "%");

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

tableModel.addRow(new Object[]{

rs.getInt("id"),

rs.getString("title"),

rs.getString("author"),

rs.getString("isbn"),

rs.getInt("quantity")

});

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error searching books: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void loadBookData() {

tableModel.setRowCount(0);

try (Connection conn = getConnection()) {

String query = "SELECT \* FROM books";

PreparedStatement pstmt = conn.prepareStatement(query);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

tableModel.addRow(new Object[]{

rs.getInt("id"),

rs.getString("title"),

rs.getString("author"),

rs.getString("isbn"),

rs.getInt("quantity")

});

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading book data: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private Connection getConnection() throws SQLException {

return DriverManager.getConnection(

"jdbc:mysql://localhost:3306/LibraryManagement", "root", "0000");

}

public static void main(String[] args) {

EventQueue.invokeLater(() -> {

try {

LibraryInventorySystem frame = new LibraryInventorySystem();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

});

}

}

**EmployeeManagementSystem.java**

package handson4;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class EmployeeManagementSystem extends JFrame {

private JTextField idField, nameField, departmentField, salaryField;

private JTable employeeTable;

private DefaultTableModel tableModel;

private JLabel totalSalaryLabel;

public EmployeeManagementSystem() {

setTitle("Employee Management System");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 900, 600);

getContentPane().setLayout(null);

JLabel lblId = new JLabel("ID:");

lblId.setBounds(20, 20, 100, 25);

getContentPane().add(lblId);

idField = new JTextField();

idField.setBounds(80, 20, 150, 25);

getContentPane().add(idField);

JLabel lblName = new JLabel("Name:");

lblName.setBounds(250, 20, 100, 25);

getContentPane().add(lblName);

nameField = new JTextField();

nameField.setBounds(310, 20, 150, 25);

getContentPane().add(nameField);

JLabel lblDepartment = new JLabel("Department:");

lblDepartment.setBounds(20, 60, 100, 25);

getContentPane().add(lblDepartment);

departmentField = new JTextField();

departmentField.setBounds(110, 60, 150, 25);

getContentPane().add(departmentField);

JLabel lblSalary = new JLabel("Salary:");

lblSalary.setBounds(280, 60, 100, 25);

getContentPane().add(lblSalary);

salaryField = new JTextField();

salaryField.setBounds(340, 60, 150, 25);

getContentPane().add(salaryField);

JButton addButton = new JButton("Add");

addButton.setBounds(510, 20, 100, 30);

getContentPane().add(addButton);

JButton updateButton = new JButton("Update");

updateButton.setBounds(510, 60, 100, 30);

getContentPane().add(updateButton);

JButton deleteButton = new JButton("Delete");

deleteButton.setBounds(620, 20, 100, 30);

getContentPane().add(deleteButton);

JButton calculateButton = new JButton("Calculate Total Salary");

calculateButton.setBounds(620, 60, 180, 30);

getContentPane().add(calculateButton);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 120, 850, 350);

getContentPane().add(scrollPane);

employeeTable = new JTable();

tableModel = new DefaultTableModel(new Object[]{"ID", "Name", "Department", "Salary"}, 0);

employeeTable.setModel(tableModel);

scrollPane.setViewportView(employeeTable);

totalSalaryLabel = new JLabel("Total Salary: 0");

totalSalaryLabel.setBounds(20, 490, 200, 30);

getContentPane().add(totalSalaryLabel);

// Button Actions

addButton.addActionListener(e -> addEmployee());

updateButton.addActionListener(e -> updateEmployee());

deleteButton.addActionListener(e -> deleteEmployee());

calculateButton.addActionListener(e -> calculateTotalSalary());

// Load employee data on startup

loadEmployeeData();

}

private void addEmployee() {

String id = idField.getText().trim();

String name = nameField.getText().trim();

String department = departmentField.getText().trim();

String salary = salaryField.getText().trim();

if (id.isEmpty() || name.isEmpty() || department.isEmpty() || salary.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill all fields.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = getConnection()) {

String query = "INSERT INTO employees (id, name, department, salary) VALUES (?, ?, ?, ?)";

PreparedStatement pstmt = conn.prepareStatement(query);

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

pstmt.setString(2, name);

pstmt.setString(3, department);

pstmt.setDouble(4, Double.parseDouble(salary));

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Employee added successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

loadEmployeeData();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error adding employee: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void updateEmployee() {

int selectedRow = employeeTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select an employee to update.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

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

String name = nameField.getText().trim();

String department = departmentField.getText().trim();

String salary = salaryField.getText().trim();

if (name.isEmpty() || department.isEmpty() || salary.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill all fields.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = getConnection()) {

String query = "UPDATE employees SET name = ?, department = ?, salary = ? WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, name);

pstmt.setString(2, department);

pstmt.setDouble(3, Double.parseDouble(salary));

pstmt.setInt(4, Integer.parseInt(id));

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Employee updated successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

loadEmployeeData();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error updating employee: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void deleteEmployee() {

int selectedRow = employeeTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select an employee to delete.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

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

try (Connection conn = getConnection()) {

String query = "DELETE FROM employees WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(query);

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

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Employee deleted successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

loadEmployeeData();

} catch (SQLException e) {

e.printStackTrace();

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

}

}

private void calculateTotalSalary() {

try (Connection conn = getConnection()) {

String query = "SELECT SUM(salary) AS total\_salary FROM employees";

PreparedStatement pstmt = conn.prepareStatement(query);

ResultSet rs = pstmt.executeQuery();

if (rs.next()) {

double totalSalary = rs.getDouble("total\_salary");

totalSalaryLabel.setText("Total Salary: " + totalSalary);

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error calculating total salary: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void loadEmployeeData() {

tableModel.setRowCount(0);

try (Connection conn = getConnection()) {

String query = "SELECT \* FROM employees";

PreparedStatement pstmt = conn.prepareStatement(query);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

tableModel.addRow(new Object[]{

rs.getInt("id"),

rs.getString("name"),

rs.getString("department"),

rs.getDouble("salary")

});

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading employee data: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private Connection getConnection() throws SQLException {

return DriverManager.getConnection(

"jdbc:mysql://localhost:3306/EmployeeManagement", "root", "0000");

}

public static void main(String[] args) {

EventQueue.invokeLater(() -> {

try {

EmployeeManagementSystem frame = new EmployeeManagementSystem();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

});

}

}

**InvoiceGenerationSystem.java**

package handson5;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

public class InvoiceGenerationSystem extends JFrame {

private JTextField productField, priceField, quantityField, totalField;

private JTable itemTable, invoiceTable;

private DefaultTableModel itemTableModel, invoiceTableModel;

public InvoiceGenerationSystem() {

setIconImage(Toolkit.getDefaultToolkit().getImage("D:\\Wallpaper\\20250330\_221934.jpg"));

setTitle("Invoice Generation System");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 900, 700);

getContentPane().setLayout(null);

JLabel lblProduct = new JLabel("Product:");

lblProduct.setBounds(20, 20, 100, 25);

getContentPane().add(lblProduct);

productField = new JTextField();

productField.setBounds(80, 20, 150, 25);

getContentPane().add(productField);

JLabel lblPrice = new JLabel("Price:");

lblPrice.setBounds(250, 20, 100, 25);

getContentPane().add(lblPrice);

priceField = new JTextField();

priceField.setBounds(310, 20, 100, 25);

getContentPane().add(priceField);

JLabel lblQuantity = new JLabel("Quantity:");

lblQuantity.setBounds(430, 20, 100, 25);

getContentPane().add(lblQuantity);

quantityField = new JTextField();

quantityField.setBounds(500, 20, 100, 25);

getContentPane().add(quantityField);

JButton addItemButton = new JButton("Add Item");

addItemButton.setBounds(620, 20, 100, 30);

getContentPane().add(addItemButton);

JScrollPane itemScrollPane = new JScrollPane();

itemScrollPane.setBounds(20, 70, 850, 200);

getContentPane().add(itemScrollPane);

itemTable = new JTable();

itemTableModel = new DefaultTableModel(new Object[]{"Product", "Price", "Quantity", "Total"}, 0);

itemTable.setModel(itemTableModel);

itemScrollPane.setViewportView(itemTable);

JLabel lblTotal = new JLabel("Total Cost:");

lblTotal.setBounds(20, 290, 100, 25);

getContentPane().add(lblTotal);

totalField = new JTextField();

totalField.setEditable(false);

totalField.setBounds(100, 290, 150, 25);

getContentPane().add(totalField);

JButton saveInvoiceButton = new JButton("Save Invoice");

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

saveInvoiceButton.setBounds(620, 290, 150, 30);

getContentPane().add(saveInvoiceButton);

JScrollPane invoiceScrollPane = new JScrollPane();

invoiceScrollPane.setBounds(20, 350, 850, 250);

getContentPane().add(invoiceScrollPane);

invoiceTable = new JTable();

invoiceTableModel = new DefaultTableModel(new Object[]{"Invoice ID", "Date", "Total Cost"}, 0);

invoiceTable.setModel(invoiceTableModel);

invoiceScrollPane.setViewportView(invoiceTable);

JButton viewInvoiceButton = new JButton("View Invoice");

viewInvoiceButton.setBounds(620, 620, 150, 30);

getContentPane().add(viewInvoiceButton);

// Button Actions

addItemButton.addActionListener(e -> addItem());

saveInvoiceButton.addActionListener(e -> saveInvoice());

viewInvoiceButton.addActionListener(e -> viewSelectedInvoice());

// Load invoices on startup

loadInvoices();

}

private void addItem() {

String product = productField.getText().trim();

String price = priceField.getText().trim();

String quantity = quantityField.getText().trim();

if (product.isEmpty() || price.isEmpty() || quantity.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill all fields.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try {

double priceValue = Double.parseDouble(price);

int quantityValue = Integer.parseInt(quantity);

double total = priceValue \* quantityValue;

itemTableModel.addRow(new Object[]{product, priceValue, quantityValue, total});

updateTotalCost();

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(this, "Invalid price or quantity.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void updateTotalCost() {

double totalCost = 0;

for (int i = 0; i < itemTableModel.getRowCount(); i++) {

totalCost += (double) itemTableModel.getValueAt(i, 3);

}

totalField.setText(String.valueOf(totalCost));

}

private void saveInvoice() {

if (itemTableModel.getRowCount() == 0) {

JOptionPane.showMessageDialog(this, "Add items to save the invoice.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = getConnection()) {

String insertInvoiceQuery = "INSERT INTO invoices (date, total\_cost) VALUES (?, ?)";

PreparedStatement invoiceStmt = conn.prepareStatement(insertInvoiceQuery, Statement.RETURN\_GENERATED\_KEYS);

String date = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());

double totalCost = Double.parseDouble(totalField.getText());

invoiceStmt.setString(1, date);

invoiceStmt.setDouble(2, totalCost);

invoiceStmt.executeUpdate();

ResultSet generatedKeys = invoiceStmt.getGeneratedKeys();

if (generatedKeys.next()) {

int invoiceId = generatedKeys.getInt(1);

String insertItemQuery = "INSERT INTO items (invoice\_id, product, price, quantity, total) VALUES (?, ?, ?, ?, ?)";

PreparedStatement itemStmt = conn.prepareStatement(insertItemQuery);

for (int i = 0; i < itemTableModel.getRowCount(); i++) {

itemStmt.setInt(1, invoiceId);

itemStmt.setString(2, itemTableModel.getValueAt(i, 0).toString());

itemStmt.setDouble(3, (double) itemTableModel.getValueAt(i, 1));

itemStmt.setInt(4, (int) itemTableModel.getValueAt(i, 2));

itemStmt.setDouble(5, (double) itemTableModel.getValueAt(i, 3));

itemStmt.addBatch();

}

itemStmt.executeBatch();

JOptionPane.showMessageDialog(this, "Invoice saved successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

itemTableModel.setRowCount(0);

totalField.setText("");

loadInvoices();

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error saving invoice: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void viewSelectedInvoice() {

int selectedRow = invoiceTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select an invoice to view.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

int invoiceId = (int) invoiceTableModel.getValueAt(selectedRow, 0);

try (Connection conn = getConnection()) {

String query = "SELECT \* FROM items WHERE invoice\_id = ?";

PreparedStatement stmt = conn.prepareStatement(query);

stmt.setInt(1, invoiceId);

ResultSet rs = stmt.executeQuery();

StringBuilder invoiceDetails = new StringBuilder("Invoice Details:\n");

while (rs.next()) {

invoiceDetails.append("Product: ").append(rs.getString("product"))

.append(", Price: ").append(rs.getDouble("price"))

.append(", Quantity: ").append(rs.getInt("quantity"))

.append(", Total: ").append(rs.getDouble("total"))

.append("\n");

}

JOptionPane.showMessageDialog(this, invoiceDetails.toString(), "Invoice Details", JOptionPane.INFORMATION\_MESSAGE);

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error fetching invoice details: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void loadInvoices() {

invoiceTableModel.setRowCount(0);

try (Connection conn = getConnection()) {

String query = "SELECT \* FROM invoices";

PreparedStatement stmt = conn.prepareStatement(query);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

invoiceTableModel.addRow(new Object[]{

rs.getInt("id"),

rs.getString("date"),

rs.getDouble("total\_cost")

});

}

} catch (SQLException e) {

e.printStackTrace();

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

}

}

private Connection getConnection() throws SQLException {

return DriverManager.getConnection("jdbc:mysql://localhost:3306/shop\_inventory", "root", "0000");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

InvoiceGenerationSystem frame = new InvoiceGenerationSystem();

frame.setVisible(true);

});

}

}

ProductInventorySystem.java

package handson6;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

public class ProductInventorySystem extends JFrame {

private JTextField idField, nameField, quantityField, priceField;

private JComboBox<String> categoryComboBox;

private JTable productTable;

private DefaultTableModel tableModel;

public ProductInventorySystem() {

setTitle("Product Inventory System");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 800, 500);

getContentPane().setLayout(null);

JLabel lblId = new JLabel("Product ID:");

lblId.setBounds(20, 20, 100, 25);

getContentPane().add(lblId);

idField = new JTextField();

idField.setBounds(120, 20, 150, 25);

getContentPane().add(idField);

JLabel lblName = new JLabel("Product Name:");

lblName.setBounds(20, 60, 100, 25);

getContentPane().add(lblName);

nameField = new JTextField();

nameField.setBounds(120, 60, 150, 25);

getContentPane().add(nameField);

JLabel lblQuantity = new JLabel("Quantity:");

lblQuantity.setBounds(20, 100, 100, 25);

getContentPane().add(lblQuantity);

quantityField = new JTextField();

quantityField.setBounds(120, 100, 150, 25);

getContentPane().add(quantityField);

JLabel lblPrice = new JLabel("Price:");

lblPrice.setBounds(20, 140, 100, 25);

getContentPane().add(lblPrice);

priceField = new JTextField();

priceField.setBounds(120, 140, 150, 25);

getContentPane().add(priceField);

JLabel lblCategory = new JLabel("Category:");

lblCategory.setBounds(20, 180, 100, 25);

getContentPane().add(lblCategory);

categoryComboBox = new JComboBox<>(new String[]{"Electronics", "Clothing", "Books", "Home & Kitchen", "Other"});

categoryComboBox.setBounds(120, 180, 150, 25);

getContentPane().add(categoryComboBox);

JButton addButton = new JButton("Add");

addButton.setBounds(20, 220, 100, 30);

getContentPane().add(addButton);

JButton updateButton = new JButton("Update");

updateButton.setBounds(130, 220, 100, 30);

getContentPane().add(updateButton);

JButton deleteButton = new JButton("Delete");

deleteButton.setBounds(240, 220, 100, 30);

getContentPane().add(deleteButton);

JButton refreshButton = new JButton("Refresh");

refreshButton.setBounds(350, 220, 100, 30);

getContentPane().add(refreshButton);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 260, 740, 180);

getContentPane().add(scrollPane);

productTable = new JTable();

tableModel = new DefaultTableModel(new Object[]{"ID", "Name", "Quantity", "Price", "Category"}, 0);

productTable.setModel(tableModel);

scrollPane.setViewportView(productTable);

// Button actions

addButton.addActionListener(e -> addProduct());

updateButton.addActionListener(e -> updateProduct());

deleteButton.addActionListener(e -> deleteProduct());

refreshButton.addActionListener(e -> loadProductData());

// Load product data on startup

loadProductData();

}

private void addProduct() {

String id = idField.getText().trim();

String name = nameField.getText().trim();

String quantity = quantityField.getText().trim();

String price = priceField.getText().trim();

String category = categoryComboBox.getSelectedItem().toString();

if (id.isEmpty() || name.isEmpty() || quantity.isEmpty() || price.isEmpty() || category.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill all fields.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = getConnection()) {

String query = "INSERT INTO products (id, name, quantity, price, category) VALUES (?, ?, ?, ?, ?)";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, id);

pstmt.setString(2, name);

pstmt.setInt(3, Integer.parseInt(quantity));

pstmt.setDouble(4, Double.parseDouble(price));

pstmt.setString(5, category);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Product added successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);

loadProductData();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error adding product: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void updateProduct() {

// Similar logic to your sample for updating rows

}

private void deleteProduct() {

// Similar logic to your sample for deleting rows

}

private void loadProductData() {

tableModel.setRowCount(0);

try (Connection conn = getConnection()) {

String query = "SELECT \* FROM products";

PreparedStatement pstmt = conn.prepareStatement(query);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

tableModel.addRow(new Object[]{

rs.getString("id"),

rs.getString("name"),

rs.getInt("quantity"),

rs.getDouble("price"),

rs.getString("category")

});

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error loading product data: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private Connection getConnection() throws SQLException {

return DriverManager.getConnection(

"jdbc:mysql://localhost:3306/ProductInventory", "root", "0000");

}

public static void main(String[] args) {

EventQueue.invokeLater(() -> {

try {

ProductInventorySystem frame = new ProductInventorySystem();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

});

}

}

MenuBarExample.java

package handson7;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class MenuBarExample extends JFrame {

public MenuBarExample() {

setTitle("Menu Bar Example");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

// Create MenuBar

JMenuBar menuBar = new JMenuBar();

// Create File menu

JMenu fileMenu = new JMenu("File");

JMenuItem newMenuItem = new JMenuItem("New");

JMenuItem openMenuItem = new JMenuItem("Open");

JMenuItem exitMenuItem = new JMenuItem("Exit");

// Add action listeners for File menu items

newMenuItem.addActionListener(e -> showMessage("New clicked"));

openMenuItem.addActionListener(e -> showMessage("Open clicked"));

exitMenuItem.addActionListener(e -> System.exit(0)); // Exit application

// Add items to File menu

fileMenu.add(newMenuItem);

fileMenu.add(openMenuItem);

fileMenu.addSeparator(); // Add separator between items

fileMenu.add(exitMenuItem);

// Create Edit menu

JMenu editMenu = new JMenu("Edit");

JMenuItem cutMenuItem = new JMenuItem("Cut");

JMenuItem copyMenuItem = new JMenuItem("Copy");

JMenuItem pasteMenuItem = new JMenuItem("Paste");

// Add action listeners for Edit menu items

cutMenuItem.addActionListener(e -> showMessage("Cut clicked"));

copyMenuItem.addActionListener(e -> showMessage("Copy clicked"));

pasteMenuItem.addActionListener(e -> showMessage("Paste clicked"));

// Add items to Edit menu

editMenu.add(cutMenuItem);

editMenu.add(copyMenuItem);

editMenu.add(pasteMenuItem);

// Add menus to MenuBar

menuBar.add(fileMenu);

menuBar.add(editMenu);

// Set the MenuBar to the frame

setJMenuBar(menuBar);

}

private void showMessage(String message) {

JOptionPane.showMessageDialog(this, message);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

MenuBarExample frame = new MenuBarExample();

frame.setVisible(true);

});

}

}

DepartmentSelectionForm.java

package handson8;

import javax.swing.\*;

import java.awt.\*;

import java.sql.\*;

import java.util.ArrayList;

public class DepartmentSelectionForm extends JFrame {

private JComboBox<String> departmentComboBox;

public DepartmentSelectionForm() {

setTitle("Department Selection");

setSize(400, 200);

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setLocationRelativeTo(null);

setLayout(new FlowLayout());

JLabel departmentLabel = new JLabel("Select Department:");

departmentComboBox = new JComboBox<>();

JButton showSelectionButton = new JButton("Show Selection");

add(departmentLabel);

add(departmentComboBox);

add(showSelectionButton);

// Load departments from the database

loadDepartments();

// Show selected department when the button is clicked

showSelectionButton.addActionListener(e -> {

String selectedDepartment = (String) departmentComboBox.getSelectedItem();

JOptionPane.*showMessageDialog*(this, "Selected Department: " + selectedDepartment);

});

}

private void loadDepartments() {

// Clear the combo box

departmentComboBox.removeAllItems();

try (Connection conn = getConnection()) {

String query = "SELECT name FROM departments";

PreparedStatement pstmt = conn.prepareStatement(query);

ResultSet rs = pstmt.executeQuery();

// Fetch department names and add them to the combo box

while (rs.next()) {

departmentComboBox.addItem(rs.getString("name"));

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.*showMessageDialog*(this, "Error loading departments: " + e.getMessage(), "Error", JOptionPane.***ERROR\_MESSAGE***);

}

}

private Connection getConnection() throws SQLException {

// Update database URL, username, and password as needed

return DriverManager.*getConnection*("jdbc:mysql://localhost:3306/YourDatabaseName", "root", "0000");

}

public static void main(String[] args) {

SwingUtilities.*invokeLater*(() -> {

DepartmentSelectionForm form = new DepartmentSelectionForm();

form.setVisible(true);

});

}

}

AttendanceSystem.java

package handson9;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import javax.swing.table.DefaultTableModel;

import java.sql.\*;

import java.util.ArrayList;

public class AttendanceSystem extends JFrame {

private JComboBox<String> studentComboBox;

private JTextField dateField;

private JComboBox<String> statusComboBox;

private JTable attendanceTable;

private DefaultTableModel tableModel;

public AttendanceSystem() {

setTitle("Attendance System");

setSize(600, 400);

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setLocationRelativeTo(null);

setLayout(new BorderLayout());

// Top Panel

JPanel topPanel = new JPanel(new FlowLayout());

studentComboBox = new JComboBox<>();

dateField = new JTextField(10);

statusComboBox = new JComboBox<>(new String[]{"Present", "Absent"});

JButton markButton = new JButton("Mark Attendance");

topPanel.add(new JLabel("Student:"));

topPanel.add(studentComboBox);

topPanel.add(new JLabel("Date (YYYY-MM-DD):"));

topPanel.add(dateField);

topPanel.add(new JLabel("Status:"));

topPanel.add(statusComboBox);

topPanel.add(markButton);

add(topPanel, BorderLayout.***NORTH***);

// Table for displaying attendance

tableModel = new DefaultTableModel(new Object[]{"ID", "Student", "Date", "Status"}, 0);

attendanceTable = new JTable(tableModel);

add(new JScrollPane(attendanceTable), BorderLayout.***CENTER***);

// Mark Attendance Action

markButton.addActionListener(this::markAttendance);

// Load Data

loadStudents();

loadAttendance();

}

private void markAttendance(ActionEvent e) {

String student = (String) studentComboBox.getSelectedItem();

String[] studentDetails = student.split(" - ");

String studentId = studentDetails[0];

String date = dateField.getText().trim();

String status = (String) statusComboBox.getSelectedItem();

if (date.isEmpty()) {

JOptionPane.*showMessageDialog*(this, "Please enter the date.", "Error", JOptionPane.***ERROR\_MESSAGE***);

return;

}

try (Connection conn = getConnection()) {

String query = "INSERT INTO attendance (student\_id, date, status) VALUES (?, ?, ?)";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setInt(1, Integer.*parseInt*(studentId));

pstmt.setString(2, date);

pstmt.setString(3, status);

pstmt.executeUpdate();

JOptionPane.*showMessageDialog*(this, "Attendance marked successfully!", "Success", JOptionPane.***INFORMATION\_MESSAGE***);

loadAttendance();

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.*showMessageDialog*(this, "Error marking attendance: " + ex.getMessage(), "Error", JOptionPane.***ERROR\_MESSAGE***);

}

}

private void loadStudents() {

studentComboBox.removeAllItems();

try (Connection conn = getConnection()) {

String query = "SELECT id, name FROM students";

PreparedStatement pstmt = conn.prepareStatement(query);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

studentComboBox.addItem(rs.getInt("id") + " - " + rs.getString("name"));

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.*showMessageDialog*(this, "Error loading students: " + ex.getMessage(), "Error", JOptionPane.***ERROR\_MESSAGE***);

}

}

private void loadAttendance() {

tableModel.setRowCount(0);

try (Connection conn = getConnection()) {

String query = "SELECT a.id, s.name, a.date, a.status FROM attendance a JOIN students s ON a.student\_id = s.id";

PreparedStatement pstmt = conn.prepareStatement(query);

ResultSet rs = pstmt.executeQuery();

while (rs.next()) {

tableModel.addRow(new Object[]{

rs.getInt("id"),

rs.getString("name"),

rs.getDate("date"),

rs.getString("status")

});

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.*showMessageDialog*(this, "Error loading attendance: " + ex.getMessage(), "Error", JOptionPane.***ERROR\_MESSAGE***);

}

}

private Connection getConnection() throws SQLException {

return DriverManager.*getConnection*("jdbc:mysql://localhost:3306/YourDatabaseName", "root", "0000");

}

public static void main(String[] args) {

SwingUtilities.*invokeLater*(() -> {

AttendanceSystem frame = new AttendanceSystem();

frame.setVisible(true);

});

}

}