Complete Java

----------------------------------------------------------------SWING APPLICATION----------------------------------------------------------------------------------------------------------------------------------------------------------------------

1. Create a Login System

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();

}

});

}

}

----------DataBase-----------------------

create database user1;

use user1;

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(100) NOT NULL,

password VARCHAR(100) NOT NULL

);

INSERT INTO users (username, password) VALUES

('admin', 'adminpass'),

('user1', 'password123');

select \* from users;

-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------

2. CRUD Operations for a Student Management System

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();

}

});

}

}

------------------------------------DataBase----------------------------------------------------

create database StudentManagement;

use StudentManagement;

CREATE TABLE students (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

roll\_number VARCHAR(20) NOT NULL,

course VARCHAR(100) NOT NULL

);

INSERT INTO students (name, roll\_number, course) VALUES

('John Doe', '123', 'Computer Science'),

('Jane Smith', '124', 'Electrical Engineering');

select \* from students;

-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------

3. Library Book Inventory System

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();

}

});

}

}

------------------------------------DataBase-----------------------------------------------------------------------

create database LibraryManagement;

use LibraryManagement;

CREATE TABLE books (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

author VARCHAR(100) NOT NULL,

isbn VARCHAR(50) NOT NULL,

quantity INT DEFAULT 0

);

INSERT INTO books (title, author, isbn, quantity) VALUES

('Java Programming', 'James Gosling', '978-0134685991', 10),

('Database Systems', 'C.J. Date', '978-0321306983', 5);

select \* from books;

-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------

4. Employee Management System

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();

}

});

}

}

---------------------------------------------------------DataBase-------------------------------------------------

create database EmployeeManagement;

use EmployeeManagement;

CREATE TABLE employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

department VARCHAR(100) NOT NULL,

salary DECIMAL(10, 2) NOT NULL

);

INSERT INTO employees (name, department, salary) VALUES

('Alice Johnson', 'HR', 50000),

('Bob Brown', 'Engineering', 75000);

select \* from employees;

-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------

5. Invoice Generation System

package handson5;

package Invoices;

import java.awt.BorderLayout;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

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() {

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 invoice\_items (invoice\_id, product\_name, price, quantity) 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 invoice\_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\_name"))

.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", "root");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

InvoiceGenerationSystem frame = new InvoiceGenerationSystem();

frame.setVisible(true);

});

}

}

--------------------------------------------------------DataBase-----------------------------------------------------

create database shop\_inventory;

use shop\_inventory;

CREATE TABLE invoices (

id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100) NOT NULL,

total\_cost DECIMAL(10, 2) NOT NULL,

date DATE NOT NULL

);

CREATE TABLE invoice\_items (

id INT AUTO\_INCREMENT PRIMARY KEY,

invoice\_id INT,

product\_name VARCHAR(255) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

quantity INT NOT NULL,

FOREIGN KEY (invoice\_id) REFERENCES invoices(id)

);

select \* from invoices;

select \* from invoice\_items;

-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------

6. Product Inventory System

package handson6;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

public class myframe extends JFrame {

private JTextField idField, product\_nameField, quantityField, priceField;

private JComboBox<String> categoryComboBox;

private JTable productTable;

private DefaultTableModel tableModel;

public myframe() {

setTitle("Product Inventory System");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 800, 500);

getContentPane().setLayout(null);

// Labels and Input Fields

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 lblproduct\_name = new JLabel("Product name:");

lblproduct\_name.setBounds(20, 60, 100, 25);

getContentPane().add(lblproduct\_name);

product\_nameField = new JTextField();

product\_nameField.setBounds(120, 60, 150, 25);

getContentPane().add(product\_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);

// Buttons

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);

// JTable and ScrollPane

JScrollPane scrollPane = new JScrollPane();

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

getContentPane().add(scrollPane);

productTable = new JTable();

tableModel = new DefaultTableModel(new Object[]{"ID", "product\_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 product\_name = product\_nameField.getText().trim();

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

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

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

if (id.isEmpty() || product\_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, product\_name, quantity, price, category) VALUES (?, ?, ?, ?, ?)";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, id);

pstmt.setString(2, product\_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) {

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

}

}

private void updateProduct() {

int selectedRow = productTable.getSelectedRow();

if (selectedRow == -1) {

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

return;

}

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

String product\_name = product\_nameField.getText().trim();

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

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

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

if (product\_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 = "UPDATE products SET product\_name = ?, quantity = ?, price = ?, category = ? WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, product\_name);

pstmt.setInt(2, Integer.parseInt(quantity));

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

pstmt.setString(4, category);

pstmt.setString(5, id);

pstmt.executeUpdate();

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

loadProductData();

} catch (SQLException e) {

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

}

}

private void deleteProduct() {

int selectedRow = productTable.getSelectedRow();

if (selectedRow == -1) {

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

return;

}

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

try (Connection conn = getConnection()) {

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

PreparedStatement pstmt = conn.prepareStatement(query);

pstmt.setString(1, id);

pstmt.executeUpdate();

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

loadProductData();

} catch (SQLException e) {

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

}

}

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("product\_name"),

rs.getInt("quantity"),

rs.getDouble("price"),

rs.getString("category")

});

}

} catch (SQLException e) {

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", "Pass@123");

}

public static void main(String[] args) {

EventQueue.invokeLater(() -> {

try {

myframe frame = new myframe();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

});

}

}

---------------------------DataBase--------------------------------------------------------------------------------------

create database ProductInventory;

use ProductInventory;

CREATE TABLE products (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

quantity INT DEFAULT 0,

price DECIMAL(10, 2) NOT NULL,

category VARCHAR(50)

);

select \* from products;

-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------

. 7. Basic Menu Bar Creation

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);

});

}

}

----------------- NO DataBase Required---------------------

-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------

8. ComboBox with Database Values

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);

});

}

}

------------------------------------------DataBase------------------------------------------------------------------

create database Department;

use Department;

CREATE TABLE departments (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL

);

INSERT INTO departments (name) VALUES

('Engineering'),

('Marketing'),

('HR');

select \* from departments;

-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------

9. Attendance System

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);

});

}

}

--------------------------------------------DataBase--------------------------------------------------------

create database StudentdAttendence;

use StudentdAttendence;

CREATE TABLE students (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

roll\_number VARCHAR(20) NOT NULL,

course VARCHAR(100) NOT NULL

);

CREATE TABLE attendance (

id INT AUTO\_INCREMENT PRIMARY KEY,

student\_id INT,

date DATE NOT NULL,

status ENUM('Present', 'Absent') NOT NULL,

FOREIGN KEY (student\_id) REFERENCES students(id)

);

INSERT INTO students (name, roll\_number, course) VALUES

('John Doe', '123', 'Computer Science'),

('Jane Smith', '124', 'Electrical Engineering');

select \* from students;

select \* from attendance;

----------------------------------------------------------------SERVERLET JSP APPLICATION----------------------------------------------------------------------------------------------------------------------------------------------------------------------

Question 1: Student Registration System

Objective: Create a web app to register students using JSP, Servlet, and JDBC.

Requirements:

• A JSP form to enter: Name, Email, Course, Roll No

• Servlet to handle form submission and save data in DB

• Show success or error message

Step1:

Create new dynamic web project

Step2:

Create new JSP file Registration.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Student Registration</title>

</head>

<body>

<h1>Student Registration Form</h1>

<form action="RegistrationServlet" method="post">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required><br><br>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required><br><br>

<label for="course">Course:</label>

<input type="text" id="course" name="course" required><br><br>

<label for="rollNo">Roll No:</label>

<input type="text" id="rollNo" name="rollNo" required><br><br>

<button type="submit">Register</button>

</form>

<%

// Display error message if "error" parameter is present in the request

String error = request.getParameter("error");

if (error != null && error.equals("1")) {

%>

<p style="color:red;">Registration not successful. Please try again.</p>

<% } %>

</body>

</html>

Step3:

Create new Servlet package name registrationServlet class name RegistrationServlet

package registrationServlet;

import jakarta.servlet.RequestDispatcher;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

@WebServlet("/RegistrationServlet")

public class RegistrationServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public RegistrationServlet() {

super();

}

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// Database credentials and URL

String jdbcUrl = "jdbc:mysql://localhost:3306/Registration"; // Update database name

String dbUsername = "root"; // Update database username

String dbPassword = "0000"; // Update database password

// Retrieve form data

String name = request.getParameter("name");

String email = request.getParameter("email");

String course = request.getParameter("course");

String rollNo = request.getParameter("rollNo");

try (Connection connection = DriverManager.getConnection(jdbcUrl, dbUsername, dbPassword)) {

// SQL query to insert student data

String sql = "INSERT INTO Students (name, email, course, roll\_no) VALUES (?, ?, ?, ?)";

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

ps.setString(1, name);

ps.setString(2, email);

ps.setString(3, course);

ps.setString(4, rollNo);

int rowsInserted = ps.executeUpdate();

if (rowsInserted > 0) {

// Success - forward to Welcome page

RequestDispatcher rd = request.getRequestDispatcher("Welcome.jsp");

rd.forward(request, response);

} else {

// Failure - redirect back with an error

response.sendRedirect("Registration.jsp?error=1");

}

}

} catch (SQLException e) {

e.printStackTrace();

response.sendRedirect("Registration.jsp?error=1");

}

}

}

Step4:

Create table in mysql

CREATE TABLE Students (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL UNIQUE,

course VARCHAR(255) NOT NULL,

roll\_no VARCHAR(50) NOT NULL UNIQUE

);

Step5:

Create a new jsp file Welcome.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Welcome</title>

</head>

<body>

<h1>Registration Successful!</h1>

<p>Welcome to the student portal.</p>

</body>

</html>

Step7:Run Registration.jsp on server

======================================================================================================================================

======================================================================================================================================

Question 2: Login System with JSP and Servlet

Objective: Implement a login module.

Requirements:

• JSP login form (username + password)

• Servlet to validate credentials

• Redirect to welcome.jsp on success, back to login on failure

Step1:

Create a jsp file Login.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Login</title>

<link rel="stylesheet" type="text/css" href="stylecss.css">

</head>

<body>

<div class="container">

<h1>Login</h1>

<form action="LoginServlet" method="post">

<label for="username">Username:</label>

<input type="text" id="username" name="username" required><br>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required><br>

<button type="submit">Login</button>

</form>

<p><a href="Index.html">Back to Home</a></p>

<%

String error = request.getParameter("error");

if (error != null && error.equals("1")) {

%>

<p style="color:red;">Invalid username or password. Please try again.</p>

<% } %>

</div>

</body>

</html>

Step2:

Create a new servlet package name loginServlet class name LoginServlet

package loginServlet;

import jakarta.servlet.RequestDispatcher;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import jakarta.servlet.http.HttpSession;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

@WebServlet("/LoginServlet")

public class LoginServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String jdbcUrl = "jdbc:mysql://localhost:3306/Registration";

String dbUser = "root";

String dbPassword = "0000";

String username = request.getParameter("username");

String password = request.getParameter("password");

try (Connection connection = DriverManager.getConnection(jdbcUrl, dbUser, dbPassword);

PreparedStatement ps = connection.prepareStatement("SELECT uname FROM Signdown WHERE uname=? AND password=?")) {

Class.forName("com.mysql.cj.jdbc.Driver");

ps.setString(1, username);

ps.setString(2, password);

try (ResultSet rs = ps.executeQuery()) {

if (rs.next()) {

HttpSession session = request.getSession();

session.setAttribute("username", username);

response.sendRedirect("Welcome.jsp");

} else {

response.sendRedirect("Login.jsp?error=1");

}

}

} catch (SQLException | ClassNotFoundException e) {

e.printStackTrace();

response.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR, "An error occurred while processing your request.");

}

}

}

Step3:

Create a new JSP Welcome.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Welcome</title>

</head>

<body>

<h1>Login Successful</h1>

<p>Welcome to the application, <b><%= request.getSession().getAttribute("username") %></b>!</p>

</body>

</html>

Step4:

Database table

CREATE TABLE Signdown (

id INT AUTO\_INCREMENT PRIMARY KEY,

uname VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL

);

INSERT INTO Signdown (uname, password) VALUES

('testuser', 'password123'),

('admin', 'admin123');

Step5:Run Login.jsp on server

======================================================================================================================================

======================================================================================================================================

Question 5: Develop a JSP page to display user input.

o Take name, email via a form.

o On submission, show the data on another JSP page.

Step1:

Create a new JSP file InputForm.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Input Form</title>

</head>

<body>

<h1>Enter Your Details</h1>

<form action="DisplayDetails.jsp" method="post">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required><br><br>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required><br><br>

<button type="submit">Submit</button>

</form>

</body>

</html>

Step2:

Create a new JSP file DisplayDetails.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Display Details</title>

</head>

<body>

<h1>User Details</h1>

<p><strong>Name:</strong> <%= request.getParameter("name") %></p>

<p><strong>Email:</strong> <%= request.getParameter("email") %></p>

<p><a href="InputForm.jsp">Go Back</a></p>

</body>

</html>

Step3: Run InputForm.jsp on server

======================================================================================================================================

======================================================================================================================================

Question 6: Implement a simple registration system using JSP and Servlets.

o JSP form: name, email, password, gender, etc.

o Servlet: save the data (can be stored in-memory or in a simple database).

o Display confirmation.

Step1;

Create a new JSP file register.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

</head>

<body>

<form action="RegisterServlet" method="post">

Name: <input name="name"><br>

Email: <input name="email"><br>

Password: <input type="password" name="password"><br>

Gender:

<input type="radio" name="gender" value="Male"> Male

<input type="radio" name="gender" value="Female"> Female<br>

<input type="submit">

</form>

</body>

</html>

Step2:

Create a new Servlet package name registerServlet class name RegisterServlet

package registerServlet;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

@WebServlet("/RegisterServlet")

public class RegisterServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException {

String name = req.getParameter("name");

String email = req.getParameter("email");

String password = req.getParameter("password");

String gender = req.getParameter("gender");

String jdbcUrl = "jdbc:mysql://localhost:3306/Registration"; // Replace with your database details

String dbUser = "root";

String dbPassword = "0000";

res.setContentType("text/html");

PrintWriter out = res.getWriter();

try (Connection connection = DriverManager.getConnection(jdbcUrl, dbUser, dbPassword)) {

Class.forName("com.mysql.cj.jdbc.Driver");

String query = "INSERT INTO Users (name, email, password, gender) VALUES (?, ?, ?, ?)";

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

ps.setString(1, name);

ps.setString(2, email);

ps.setString(3, password); // Ideally, hash the password before storing

ps.setString(4, gender);

int result = ps.executeUpdate();

if (result > 0) {

out.println("<h3>Registration Successful!</h3>");

out.println("<p>Welcome, " + name + " (" + gender + ")</p>");

} else {

out.println("<h3>Registration Failed. Please try again.</h3>");

}

}

} catch (SQLException | ClassNotFoundException e) {

e.printStackTrace();

out.println("<h3>Error: Unable to register. Please try again later.</h3>");

}

}

}

Step3:

Create table

CREATE TABLE Users (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

gender ENUM('Male', 'Female') NOT NULL

);

Step4:Run register.jsp on server

======================================================================================================================================

======================================================================================================================================

Question 7. Create a basic shopping cart system using JSP and Servlets.

o Use JSP to display product list.

o Servlet to manage cart (add/remove items).

o Session tracking to maintain user’s cart.

Step1:

Create a new JSP file products.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Product List</title>

</head>

<body>

<h1>Available Products</h1>

<form action="CartServlet" method="post">

<table border="1">

<tr>

<th>Product</th>

<th>Price</th>

<th>Add to Cart</th>

</tr>

<tr>

<td>Product A</td>

<td><button type="submit" name="product" value="Product A">Add</button></td>

</tr>

<tr>

<td>Product B</td>

<td>$20</td>

<td><button type="submit" name="product" value="Product B">Add</button></td>

</tr>

<tr>

<td>Product C</td>

<td>$30</td>

<td><button type="submit" name="product" value="Product C">Add</button></td>

</tr>

</table>

</form>

<p><a href="Viewcart.jsp">View Cart</a></p>

</body>

</html>

Step2:

Create a new servlet package name cartServlet class name CartServlet

package cartServlet;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import jakarta.servlet.http.HttpSession;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

@WebServlet("/CartServlet")

public class CartServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String product = request.getParameter("product");

HttpSession session = request.getSession();

@SuppressWarnings("unchecked")

List<String> cart = (List<String>) session.getAttribute("cart");

if (cart == null) {

cart = new ArrayList<>();

}

if (product != null) {

cart.add(product);

}

session.setAttribute("cart", cart);

response.sendRedirect("products.jsp");

}

}

Step3:

Create a new servlet package name cartServlet class name RemoveItemServlet

package cartServlet;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import jakarta.servlet.http.HttpSession;

import java.io.IOException;

import java.util.List;

@WebServlet("/RemoveItemServlet")

public class RemoveItemServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String product = request.getParameter("product");

HttpSession session = request.getSession();

@SuppressWarnings("unchecked")

List<String> cart = (List<String>) session.getAttribute("cart");

if (cart != null && product != null) {

cart.remove(product);

}

session.setAttribute("cart", cart);

response.sendRedirect("Viewcart.jsp");

}

}

Step4:

Create a new JSP file Viewcart.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ page import="java.util.List" %>

<%@ page import="java.util.ArrayList" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Your Cart</title>

</head>

<body>

<h1>Your Cart</h1>

<form action="RemoveItemServlet" method="post">

<table border="1">

<tr>

<th>Product</th>

<th>Remove</th>

</tr>

<%

// Retrieve the cart from the session

Object cartObj = session.getAttribute("cart");

List<String> cart = new ArrayList<>();

if (cartObj != null && cartObj instanceof List<?>) {

cart = (List<String>) cartObj; // Safe casting to List<String>

}

if (cart.isEmpty()) {

%>

<tr>

<td colspan="2">Your cart is empty.</td>

</tr>

<% } else {

for (String product : cart) {

%>

<tr>

<td><%= product %></td>

<td><button type="submit" name="product" value="<%= product %>">Remove</button></td>

</tr>

<% }

}

%>

</table>

</form>

<p><a href="products.jsp">Continue Shopping</a></p>

</body>

</html>

Step5:Run products.jsp file on server

======================================================================================================================================

======================================================================================================================================

Question 8. Build a feedback system using JSP, Servlet, and JDBC.

o Store feedback in a database.

o JSP for form display and viewing feedback.

o Servlet for data handling.

Step1:

Create a new JSP file feedback.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Feedback Form</title>

</head>

<body>

<h1>Submit Your Feedback</h1>

<form action="FeedbackServlet" method="post">

<label for="name">Name:</label><br>

<input type="text" id="name" name="name" required><br><br>

<label for="email">Email:</label><br>

<input type="email" id="email" name="email" required><br><br>

<label for="message">Message:</label><br>

<textarea id="message" name="message" rows="5" required></textarea><br><br>

<button type="submit">Submit Feedback</button>

</form>

<% String success = request.getParameter("success"); %>

<% if (success != null && success.equals("1")) { %>

<p style="color: green;">Feedback submitted successfully!</p>

<% } %>

<% String error = request.getParameter("error"); %>

<% if (error != null && error.equals("1")) { %>

<p style="color: red;">Error submitting feedback. Please try again.</p>

<% } %>

<p><a href="viewFeedback.jsp">View All Feedback</a></p>

</body>

</html>

Step2:

Create a new JSP file viewFeedback.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ page import="java.sql.\*" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>View Feedback</title>

</head>

<body>

<h1>Feedback from Users</h1>

<table border="1">

<tr>

<th>ID</th>

<th>Name</th>

<th>Email</th>

<th>Message</th>

</tr>

<%

try {

// Database connection

Class.forName("com.mysql.cj.jdbc.Driver");

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/FeedbackDB", "root", "0000");

// Retrieve feedback from database

String sql = "SELECT \* FROM feedback";

Statement statement = connection.createStatement();

ResultSet rs = statement.executeQuery(sql);

while (rs.next()) {

%>

<tr>

<td><%= rs.getInt("id") %></td>

<td><%= rs.getString("name") %></td>

<td><%= rs.getString("email") %></td>

<td><%= rs.getString("message") %></td>

</tr>

<%

}

} catch (Exception e) {

e.printStackTrace();

%>

<tr>

<td colspan="4">Error retrieving feedback.</td>

</tr>

<%

}

%>

</table>

<p><a href="feedback.jsp">Back to Feedback Form</a></p>

</body>

</html>

Step3:

Create a new Servlet package name feedbackServlet class name FeedbackServlet

package feedbackServlet;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

@WebServlet("/FeedbackServlet")

public class FeedbackServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String name = request.getParameter("name");

String email = request.getParameter("email");

String message = request.getParameter("message");

try {

// Database connection

Class.forName("com.mysql.cj.jdbc.Driver");

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/FeedbackDB", "root", "0000");

// Insert feedback into database

String sql = "INSERT INTO feedback (name, email, message) VALUES (?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, name);

statement.setString(2, email);

statement.setString(3, message);

statement.executeUpdate();

// Redirect to feedback form with a success message

response.sendRedirect("feedback.jsp?success=1");

} catch (Exception e) {

e.printStackTrace();

response.sendRedirect("feedback.jsp?error=1");

}

}

}

Step4:

Create database and table

CREATE DATABASE FeedbackDB;

USE FeedbackDB;

CREATE TABLE feedback (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

email VARCHAR(100),

message TEXT

);

Step5:Run feedback.jsp

======================================================================================================================================

======================================================================================================================================

Question 9 : Mini Blog Application

o Add/Edit/Delete blog posts using JSP/Servlet.

o Store in DB.

Show list of blogs on home page

Step1:

Create a new JSP file addBlog.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Add Blog</title>

</head>

<body>

<h1>Add Blog Post</h1>

<form action="BlogServlet" method="post">

<input type="hidden" name="action" value="add">

<label for="title">Title:</label>

<input type="text" id="title" name="title" required><br><br>

<label for="content">Content:</label><br>

<textarea id="content" name="content" rows="10" cols="30" required></textarea><br><br>

<button type="submit">Add Blog</button>

</form>

<p><a href="home.jsp">Back to Home</a></p>

</body>

</html>

Step2:

Create a new JSP file editBlog.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ page import="java.sql.\*" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Edit Blog</title>

</head>

<body>

<h1>Edit Blog Post</h1>

<%

int id = Integer.parseInt(request.getParameter("id"));

String title = "";

String content = "";

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/BlogDB", "root", "0000");

String sql = "SELECT \* FROM blog WHERE id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, id);

ResultSet rs = statement.executeQuery();

if (rs.next()) {

title = rs.getString("title");

content = rs.getString("content");

}

rs.close();

statement.close();

connection.close();

} catch (Exception e) {

e.printStackTrace();

}

%>

<form action="BlogServlet" method="post">

<input type="hidden" name="action" value="edit">

<input type="hidden" name="id" value="<%= id %>">

<label for="title">Title:</label>

<input type="text" id="title" name="title" value="<%= title %>" required><br><br>

<label for="content">Content:</label><br>

<textarea id="content" name="content" rows="10" cols="30" required><%= content %></textarea><br><br>

<button type="submit">Update Blog</button>

</form>

<p><a href="home.jsp">Back to Home</a></p>

</body>

</html>

Step3:

Create a new JSP file home.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ page import="java.sql.\*" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Blog Home</title>

</head>

<body>

<h1>Blog Posts</h1>

<a href="addBlog.jsp">Add New Blog</a>

<table border="1">

<tr>

<th>ID</th>

<th>Title</th>

<th>Content</th>

<th>Actions</th>

</tr>

<%

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/BlogDB", "root", "0000");

String sql = "SELECT \* FROM blog";

Statement statement = connection.createStatement();

ResultSet rs = statement.executeQuery(sql);

while (rs.next()) {

%>

<tr>

<td><%= rs.getInt("id") %></td>

<td><%= rs.getString("title") %></td>

<td><%= rs.getString("content") %></td>

<td>

<a href="editBlog.jsp?id=<%= rs.getInt("id") %>">Edit</a>

<a href="BlogServlet?action=delete&id=<%= rs.getInt("id") %>">Delete</a>

</td>

</tr>

<%

}

rs.close();

statement.close();

connection.close();

} catch (Exception e) {

e.printStackTrace();

}

%>

</table>

</body>

</html>

Step4:

Create a new Servlet package name blog class name BlogServlet

package blog;

import jakarta.servlet.\*;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.\*;

import java.io.\*;

import java.sql.\*;

@WebServlet("/BlogServlet")

public class BlogServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {

String action = request.getParameter("action");

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/BlogDB", "root", "0000");

if ("add".equals(action)) {

String title = request.getParameter("title");

String content = request.getParameter("content");

String sql = "INSERT INTO blog (title, content) VALUES (?, ?)";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, title);

statement.setString(2, content);

statement.executeUpdate();

statement.close();

} else if ("edit".equals(action)) {

int id = Integer.parseInt(request.getParameter("id"));

String title = request.getParameter("title");

String content = request.getParameter("content");

String sql = "UPDATE blog SET title = ?, content = ? WHERE id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, title);

statement.setString(2, content);

statement.setInt(3, id);

statement.executeUpdate();

statement.close();

}

connection.close();

} catch (Exception e) {

e.printStackTrace();

}

response.sendRedirect("home.jsp");

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {

String action = request.getParameter("action");

if ("delete".equals(action)) {

try {

int id = Integer.parseInt(request.getParameter("id"));

Class.forName("com.mysql.cj.jdbc.Driver");

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/BlogDB", "root", "0000");

String sql = "DELETE FROM blog WHERE id = ?";

PreparedStatement statement = connection.prepareStatement(sql);

statement.setInt(1, id);

statement.executeUpdate();

statement.close();

connection.close();

} catch (Exception e) {

e.printStackTrace();

}

}

response.sendRedirect("home.jsp");

}

}

Step5:

Create database and table

CREATE DATABASE IF NOT EXISTS BlogDB;

USE BlogDB;

CREATE TABLE IF NOT EXISTS blog (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

content TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

Step6: Run addBlog.jsp on server

======================================================================================================================================

======================================================================================================================================

Question 10. Library Book Search System

• JSP form to enter search criteria.

• Servlet retrieves and displays matching books from DB.

Step1:

Create a new JSP file searchBooks.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Library Book Search</title>

</head>

<body>

<h1>Search Library Books</h1>

<form action="BookSearchServlet" method="get">

<label for="title">Title:</label>

<input type="text" id="title" name="title"><br><br>

<label for="author">Author:</label>

<input type="text" id="author" name="author"><br><br>

<label for="genre">Genre:</label>

<input type="text" id="genre" name="genre"><br><br>

<label for="year">Year:</label>

<input type="text" id="year" name="year"><br><br>

<button type="submit">Search</button>

</form>

</body>

</html>

Step2:

Create a new Servlet package name library class name BookSearchServlet

package library;

import jakarta.servlet.\*;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.\*;

import java.io.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

@WebServlet("/BookSearchServlet")

public class BookSearchServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String title = request.getParameter("title");

String author = request.getParameter("author");

String genre = request.getParameter("genre");

String year = request.getParameter("year");

List<String[]> books = new ArrayList<>();

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/LibraryDB", "root", "0000");

StringBuilder sql = new StringBuilder("SELECT \* FROM books WHERE 1=1");

if (title != null && !title.isEmpty()) {

sql.append(" AND title LIKE ?");

}

if (author != null && !author.isEmpty()) {

sql.append(" AND author LIKE ?");

}

if (genre != null && !genre.isEmpty()) {

sql.append(" AND genre LIKE ?");

}

if (year != null && !year.isEmpty()) {

sql.append(" AND year = ?");

}

PreparedStatement statement = connection.prepareStatement(sql.toString());

int index = 1;

if (title != null && !title.isEmpty()) {

statement.setString(index++, "%" + title + "%");

}

if (author != null && !author.isEmpty()) {

statement.setString(index++, "%" + author + "%");

}

if (genre != null && !genre.isEmpty()) {

statement.setString(index++, "%" + genre + "%");

}

if (year != null && !year.isEmpty()) {

statement.setInt(index++, Integer.parseInt(year));

}

ResultSet rs = statement.executeQuery();

while (rs.next()) {

books.add(new String[]{

rs.getString("title"),

rs.getString("author"),

rs.getString("genre"),

String.valueOf(rs.getInt("year"))

});

}

rs.close();

statement.close();

connection.close();

} catch (Exception e) {

e.printStackTrace();

}

request.setAttribute("books", books);

RequestDispatcher dispatcher = request.getRequestDispatcher("results.jsp");

dispatcher.forward(request, response);

}

}

Step3:

Create a new JSP file results.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ page import="java.util.List" %>

<%@ page import="java.util.ArrayList" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Search Results</title>

</head>

<body>

<h1>Search Results</h1>

<%

List<String[]> books = (List<String[]>) request.getAttribute("books");

if (books == null || books.isEmpty()) {

%>

<p>No matching books found.</p>

<%

} else {

%>

<table border="1">

<tr>

<th>Title</th>

<th>Author</th>

<th>Genre</th>

<th>Year</th>

</tr>

<%

for (String[] book : books) {

%>

<tr>

<td><%= book[0] %></td>

<td><%= book[1] %></td>

<td><%= book[2] %></td>

<td><%= book[3] %></td>

</tr>

<%

}

%>

</table>

<%

}

%>

<p><a href="searchBooks.jsp">Back to Search</a></p>

</body>

</html>

Step4:

Create database and table

CREATE DATABASE IF NOT EXISTS LibraryDB;

USE LibraryDB;

CREATE TABLE IF NOT EXISTS books (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

author VARCHAR(255) NOT NULL,

genre VARCHAR(100),

year INT

);

-- Sample Data

INSERT INTO books (title, author, genre, year) VALUES

('The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', 1925),

('To Kill a Mockingbird', 'Harper Lee', 'Fiction', 1960),

('1984', 'George Orwell', 'Dystopian', 1949),

('Pride and Prejudice', 'Jane Austen', 'Romance', 1813);

Step5:Run searchBooks.jsp on server

======================================================================================================================================

======================================================================================================================================

+