Experiment 7

Student Name: Anushree Alok **UID: 22BCS16052**
Branch: CSE **Section: 625-A Cloud IOT (TPP)**
Semester: 6th **DOP: 08/03/25**
Subject: PBLJ **Subject Code: 22CSH-359**

Aim: Servlet Lifecycle, Generic Servlet, Http Servlet, Linking Servlet to HTML, HTTP Servlet Request and Response, Servlet with JDBC, configuring project using servlet, Servlet Config and Servlet Mapping JSP declaration, JSP directives, JSP Scriptlets, JSP include tag, JSP page tag.

Objective: Develop web applications using Servlets and JSP for user input handling, database integration.

Problem 1.

Write a servlet to accept user credentials through an HTML form and display a personalized welcome message if the login is successful.

Code:

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class SimpleJDBCFetch {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/company";

        String user = "Anu";

        String password = "123";

        String query = "SELECT * FROM Employee";

        try {

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

            System.out.println("Connecting to database...");

            Connection conn = DriverManager.getConnection(url, user, password);

            System.out.println("Database connection successful!");
```

```
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            System.out.println("\nEmployee Records:");
            System.out.println("--------------------------------------------------");
            System.out.printf("%-10s %-20s %-10s\n", "EmpID", "Name", "Salary");
            System.out.println("--------------------------------------------------");
            while (rs.next()) {
                int empId = rs.getInt("EmpID");
                String name = rs.getString("Name");
                double salary = rs.getDouble("Salary");
                System.out.printf("%-10d %-20s $%-10.2f\n", empId, name, salary);
            }
            System.out.println("--------------------------------------------------");
            rs.close();
            stmt.close();
            conn.close();
            System.out.println("\nDatabase connection closed.");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Output

```
Connecting to database...
Database connection successful!

Employee Records:
--------------------------------------------------
EmpID      Name                Salary
--------------------------------------------------
1          John Smith          $65000.00
2          Mary Johnson        $72000.00
3          Robert Brown        $58000.00
4          Patricia Davis      $81000.00
5          Michael Wilson      $63500.00
--------------------------------------------------

Database connection closed.
```

Problem 2:

Build a program to perform CRUD operations (Create, Read, Update, Delete) on a database table Product with columns: ProductID, ProductName, Price, and Quantity. The program should include: Menu-driven options for each operation. Transaction handling to ensure data integrity.

Code:

```
import java.sql.*;

import java.util.Scanner;

public class ProductCRUDApp {

    private static final String URL = "jdbc:mysql://localhost:3306/inventory";

    private static final String USER = "Anu";
```

```
    private static final String PASSWORD = "123";

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            do {
                System.out.println("\n===== Product Management System =====");
                System.out.println("1. View All Products");
                System.out.println("2. Add New Product");
                System.out.println("3. Update Product");
                System.out.println("4. Delete Product");
                System.out.println("5. Exit");
                System.out.print("Enter your choice: ");
                choice = scanner.nextInt();
                scanner.nextLine();
                switch (choice) {
                    case 1:
                        viewAllProducts();
                        break;
                    case 2:
                        addNewProduct(scanner);
                        break;
                    case 3:
                        updateProduct(scanner);
                        break;
                    case 4:
                        deleteProduct(scanner);
                        break;
```

```
                case 5:
                    System.out.println("Exiting application. Goodbye!");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 5);
    } catch (ClassNotFoundException e) {
        System.out.println("MySQL JDBC Driver not found.");
        e.printStackTrace();
    } finally {
        scanner.close();
    }
}
private static Connection getConnection() throws SQLException {
    return DriverManager.getConnection(URL, USER, PASSWORD);
}
private static void viewAllProducts() {
    try (Connection conn = getConnection();
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery("SELECT * FROM Product")) {

        System.out.println("\n===== Products List =====");
        System.out.println("-----------------------------------------------------------");
        System.out.printf("%-10s %-30s %-10s %-10s\n", "ProductID", "ProductName", "Price", "Quantity");
        System.out.println("-----------------------------------------------------------");
        boolean hasData = false;
        while (rs.next()) {
            hasData = true;
```

```
            int productId = rs.getInt("ProductID");
            String productName = rs.getString("ProductName");
            double price = rs.getDouble("Price");
            int quantity = rs.getInt("Quantity");
            System.out.printf("%-10d %-30s $%-10.2f %-10d\n", productId, productName, price, quantity);
        }
        if (!hasData) {
            System.out.println("No products found in the database.");
        }
        System.out.println("------------------------------------------------------------");
    } catch (SQLException e) {
        System.out.println("Error fetching products: " + e.getMessage());
    }
}
private static void addNewProduct(Scanner scanner) {
    try (Connection conn = getConnection()) {
        conn.setAutoCommit(false);
        try {
            System.out.print("Enter Product Name: ");
            String productName = scanner.nextLine();

            System.out.print("Enter Price: ");
            double price = scanner.nextDouble();
            System.out.print("Enter Quantity: ");
            int quantity = scanner.nextInt();
            scanner.nextLine();
            String sql = "INSERT INTO Product (ProductName, Price, Quantity) VALUES (?, ?, ?)";
            try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
                pstmt.setString(1, productName);
```

```
                pstmt.setDouble(2, price);
                pstmt.setInt(3, quantity);
                int rowsAffected = pstmt.executeUpdate();
                if (rowsAffected > 0) {
                    System.out.println("Product added successfully!");
                    conn.commit();
                } else {
                    System.out.println("Failed to add product.");
                    conn.rollback();
                }
            }
        } catch (Exception e) {
            System.out.println("Error occurred. Rolling back transaction.");
            conn.rollback();
            e.printStackTrace();
        } finally {
            conn.setAutoCommit(true);
        }
    } catch (SQLException e) {
        System.out.println("Database error: " + e.getMessage());
    }
}
private static void updateProduct(Scanner scanner) {
    try (Connection conn = getConnection()) {
        conn.setAutoCommit(false);
        try {
            System.out.print("Enter Product ID to update: ");
            int productId = scanner.nextInt();
            scanner.nextLine();
```

```
String checkSql = "SELECT * FROM Product WHERE ProductID = ?";
try (PreparedStatement checkStmt = conn.prepareStatement(checkSql)) {
    checkStmt.setInt(1, productId);
    ResultSet rs = checkStmt.executeQuery();
    if (!rs.next()) {
        System.out.println("Product not found with ID: " + productId);
        return;
    }
}
System.out.print("Enter new Product Name (or press Enter to keep current): ");
String productName = scanner.nextLine();
System.out.print("Enter new Price (or -1 to keep current): ");
double price = scanner.nextDouble();
System.out.print("Enter new Quantity (or -1 to keep current): ");
int quantity = scanner.nextInt();
scanner.nextLine();
StringBuilder sqlBuilder = new StringBuilder("UPDATE Product SET ");
boolean needsComma = false;
if (!productName.isEmpty()) {
    sqlBuilder.append("ProductName = ?");
    needsComma = true;
}
if (price >= 0) {
    if (needsComma) {
        sqlBuilder.append(", ");
    }
    sqlBuilder.append("Price = ?");
    needsComma = true;
}
```

```
        if (quantity >= 0) {
            if (needsComma) {
                sqlBuilder.append(", ");
            }
            sqlBuilder.append("Quantity = ?");
        }
        sqlBuilder.append(" WHERE ProductID = ?");
        try (PreparedStatement pstmt = conn.prepareStatement(sqlBuilder.toString())) {
            int parameterIndex = 1;
            if (!productName.isEmpty()) {
                pstmt.setString(parameterIndex++, productName);
            }
            if (price >= 0) {
                pstmt.setDouble(parameterIndex++, price);
            }
            if (quantity >= 0) {
                pstmt.setInt(parameterIndex++, quantity);
            }
            pstmt.setInt(parameterIndex, productId);
            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Product updated successfully!");
                conn.commit();
            } else {
                System.out.println("Failed to update product.");
                conn.rollback();
            }
        }
```

```
            } catch (Exception e) {
                System.out.println("Error occurred. Rolling back transaction.");
                conn.rollback();
                e.printStackTrace();
            } finally {
                conn.setAutoCommit(true);
            }
        } catch (SQLException e) {
            System.out.println("Database error: " + e.getMessage());
        }
    }
    private static void deleteProduct(Scanner scanner) {
        try (Connection conn = getConnection()) {
            conn.setAutoCommit(false);
            try {
                System.out.print("Enter Product ID to delete: ");
                int productId = scanner.nextInt();
                scanner.nextLine();
                String sql = "DELETE FROM Product WHERE ProductID = ?";
                try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
                    pstmt.setInt(1, productId);

                    int rowsAffected = pstmt.executeUpdate();
                    if (rowsAffected > 0) {
                        System.out.println("Product deleted successfully!");
                        conn.commit();
                    } else {
                        System.out.println("No product found with ID: " + productId);
                        conn.rollback();
```

```
                }
            }
        } catch (Exception e) {
            System.out.println("Error occurred. Rolling back transaction.");
            conn.rollback();
            e.printStackTrace();
        } finally {
            conn.setAutoCommit(true);
        }
    } catch (SQLException e) {
        System.out.println("Database error: " + e.getMessage());
    }
  }
}
```

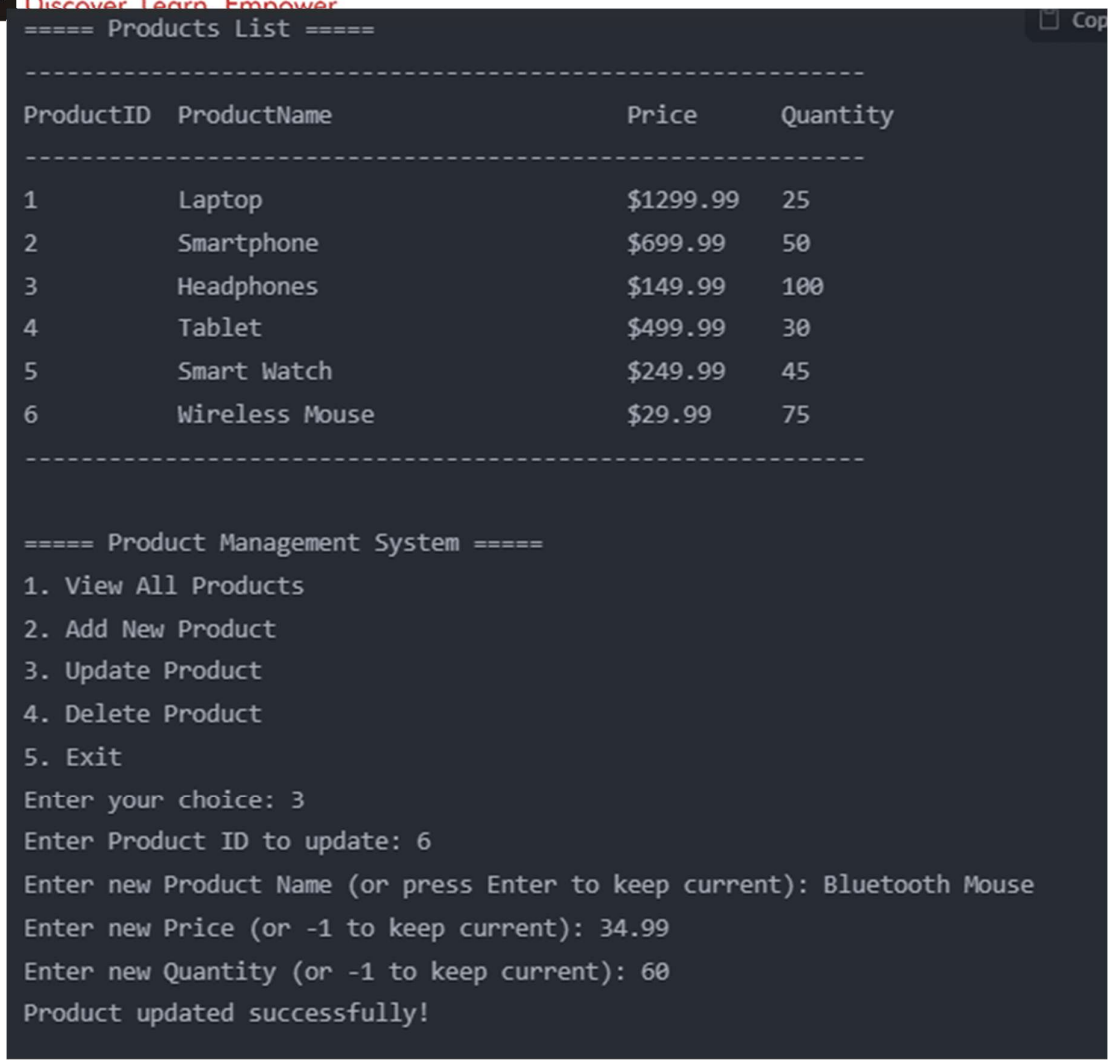
Output

```
===== Product Management System =====
1. View All Products
2. Add New Product
3. Update Product
4. Delete Product
5. Exit
Enter your choice: 1

===== Products List =====
------------------------------------------------------------
ProductID  ProductName                    Price      Quantity
------------------------------------------------------------
1          Laptop                         $1299.99   25
2          Smartphone                     $699.99    50
3          Headphones                     $149.99    100
4          Tablet                         $499.99    30
5          Smart Watch                    $249.99    45
------------------------------------------------------------
```

```
===== Product Management System =====
1. View All Products
2. Add New Product
3. Update Product
4. Delete Product
5. Exit
Enter your choice: 2
Enter Product Name: Wireless Mouse
Enter Price: 29.99
Enter Quantity: 75
Product added successfully!

===== Product Management System =====
1. View All Products
2. Add New Product
3. Update Product
4. Delete Product
5. Exit
Enter your choice: 1
```



```
===== Products List =====
------------------------------------------------------------
ProductID  ProductName                    Price      Quantity
------------------------------------------------------------
1          Laptop                         $1299.99   25
2          Smartphone                     $699.99    50
3          Headphones                     $149.99    100
4          Tablet                         $499.99    30
5          Smart Watch                    $249.99    45
6          Wireless Mouse                 $29.99     75
------------------------------------------------------------

===== Product Management System =====
1. View All Products
2. Add New Product
3. Update Product
4. Delete Product
5. Exit
Enter your choice: 3
Enter Product ID to update: 6
Enter new Product Name (or press Enter to keep current): Bluetooth Mouse
Enter new Price (or -1 to keep current): 34.99
Enter new Quantity (or -1 to keep current): 60
Product updated successfully!
```

Problem 3:

Develop a Java application using JDBC and MVC architecture to manage student data. The application should: Use a Student class as the model with fields like StudentID, Name, Department, and Marks. Include a database table to store student data. Allow the user to perform CRUD operations through a simple menu-driven view. Implement database operations in a separate controller class.

Code:

```
package com.studentmgmt.model;
```

```
public class Student {
    private int studentId;
    private String name;
    private String department;
    private double marks;
    public Student() {
    }
    public Student(int studentId, String name, String department, double marks) {
        this.studentId = studentId;
        this.name = name;
        this.department = department;
        this.marks = marks;
    }
    public int getStudentId() {
        return studentId;
    }
    public void setStudentId(int studentId) {
        this.studentId = studentId;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDepartment() {
        return department;
    }
    public void setDepartment(String department) {
        this.department = department;
```

```
    }
    public double getMarks() {
        return marks;
    }
    public void setMarks(double marks) {
        this.marks = marks;
    }
    @Override
    public String toString() {
        return "Student [ID=" + studentId + ", Name=" + name + ", Department=" + department + ", Marks=" + marks + "]";
    }
}
package com.studentmgmt.controller;
import com.studentmgmt.model.Student;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
public class StudentController {
    private static final String URL = "jdbc:mysql://localhost:3306/studentdb";
    private static final String USER = "Anu";
    private static final String PASSWORD = "123";
    private Connection getConnection() throws SQLException {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (ClassNotFoundException e) {
            throw new SQLException("MySQL JDBC Driver not found", e);
        }
    }
    public boolean addStudent(Student student) {
```

```
    String sql = "INSERT INTO Student (Name, Department, Marks) VALUES (?, ?, ?)";
    try (Connection conn = getConnection();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, student.getName());
        pstmt.setString(2, student.getDepartment());
        pstmt.setDouble(3, student.getMarks());
        int rowsAffected = pstmt.executeUpdate();
        return rowsAffected > 0;
    } catch (SQLException e) {
        System.err.println("Error adding student: " + e.getMessage());
        return false;
    }   }
public List<Student> getAllStudents() {
    List<Student> students = new ArrayList<>();
    String sql = "SELECT * FROM Student";
    try (Connection conn = getConnection();
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {
        while (rs.next()) {
            Student student = new Student();
            student.setStudentId(rs.getInt("StudentID"));
            student.setName(rs.getString("Name"));
            student.setDepartment(rs.getString("Department"));
            student.setMarks(rs.getDouble("Marks"));
            students.add(student);
        }
    } catch (SQLException e) {
        System.err.println("Error retrieving students: " + e.getMessage());
    }
    return students;
```

Output

```
===== Student Management System =====
1. View All Students
2. Add New Student
3. Update Student
4. Delete Student
5. Search Student by ID
6. Exit
Enter your choice: 1

===== Students List =====
------------------------------------------------------------
ID    Name                Department       Marks
------------------------------------------------------------
1     Alice Cooper        Computer Science 85.5
2     Bob Johnson         Electrical Eng   78.0
3     Charlie Brown       Mechanical Eng   92.3
4     Diana Ross          Civil Eng        88.7
------------------------------------------------------------

===== Student Management System =====
1. View All Students
2. Add New Student
3. Update Student
4. Delete Student
5. Search Student by ID
```

Learning Outcomes:

1. Database connectivity and CRUD operations mastery
2. Transaction management for data integrity
3. MVC architecture implementation
4. Prepared statements for security and performance
5. Proper resource management and connection handling