Enterprise Programming

Lab

Manual

303105310

Department of Information technology Parul

Institute of Engineering and Technology

Faculty of Engineering and Technology

Parul University

Session 2025-26

Published by:

Faculty of Engineering and Technology

Department of Information technology

Parul University

EXPERIMENT NO. 1

Objective(s): To perform insert and retrieve operations on a database using JDBC.
Outcome: Students will be able to connect with a database and execute basic CRUD operations using JDBC.
Problem Statement: Write a Java program to insert and retrieve data from a database using JDBC.

Background Study:

1) Introduction:

- JDBC (Java Database Connectivity) is an API that allows Java programs to interact with a database.
- It helps execute SQL queries, retrieve data, and manage database connections.

2) Installation Guide:

- Load the JDBC driver.
- Establish a connection using `DriverManager`.
- Create a `Statement` or `PreparedStatement` object.
- Execute queries using `executeQuery()` or `executeUpdate()`.
- Retrieve the results using `ResultSet`.
- Close all resources.

Advantage:

- Direct control over SQL execution.
- Works with any database that supports JDBC drivers.
- Integrates well with Java EE technologies.

- **Disadvantage:**
- Verbose and error-prone for complex operations.
- No built-in ORM support.
- Manual connection/resource management required.

Algorithm (Student Work Area):

z

Code (Student Work Area):

Question Bank:

1. What is JDBC?
2. What are the steps to connect a Java application with a database using JDBC?
3. What are the types of JDBC drivers?

EXPERIMENT NO.2

Objective(s):
To demonstrate the use of PreparedStatement and ResultSet in JDBC.

Outcome:
Students will understand how to execute parameterized queries and read data using `ResultSet`.

Problem Statement:

Write a program to demonstrate the use of `PreparedStatement` and

Background Study:

`PreparedStatement`: Used to execute precompiled SQL queries.

`ResultSet`: Interface used to read data from database results.

Algorithm (Student Work Area):

Code (Student Work Area):

Question Bank:

1. What is the advantage of using `PreparedStatement` over `Statement`?
2. How does `PreparedStatement` help prevent SQL Injection?
3. What are the different types of `ResultSet`?
4. How can you update data using `ResultSet`?
5. Explain parameter binding in `PreparedStatement`.

EXPERIMENT NO. 3

Objective(s):
Servlet Execution on Apache Tomcat

Outcome:
Students will learn how to set up and deploy servlets.

Problem Statement:

Write a servlet to print “Hello World” and deploy it on Apache Tomcat.

Background Study:

Introduction:

Servlets are server-side Java programs for handling HTTP requests. Apache Tomcat is a Java-based web server and servlet container.

Steps:

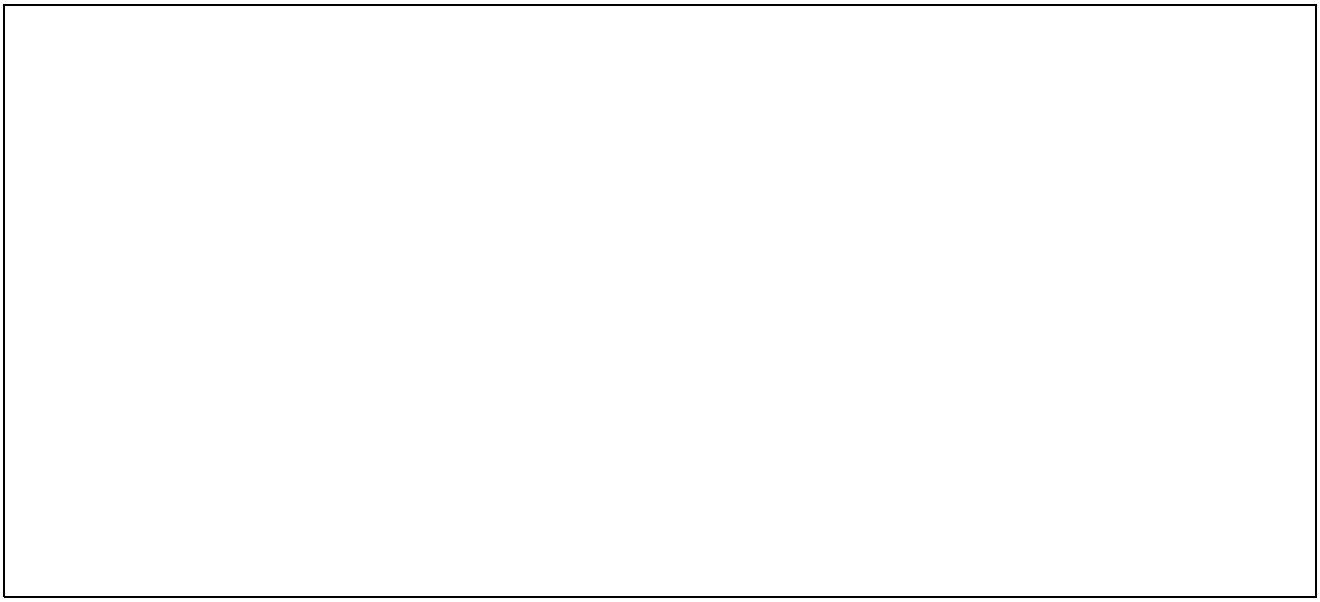
- Create a servlet by extending `HttpServlet`
- Implement `doGet()` or `doPost()` methods
- Configure the servlet in `web.xml`
- Deploy to Apache Tomcat and run via browser

Algorithm (Student Work Area):

Code (Student Work Area):

Question Bank:

1. What is a servlet?
2. What are the lifecycle methods of a servlet?
3. How do you deploy a servlet on Tomcat?
4. What is the use of `web.xml`?
5. How is `HttpServlet` different from `GenericServlet`?



EXPERIMENT NO. 4

Objective(s):
To demonstrate form handling in servlets.

Outcome:
Students will learn how to capture form data and process it via servlets

Problem Statement:

Servlets can handle form input from `GET` or `POST` methods via `request.getParameter()`.

Background Study:

Introduction:
Servlets can handle form input from `GET` or `POST` methods via `request.getParameter()`.

Steps:

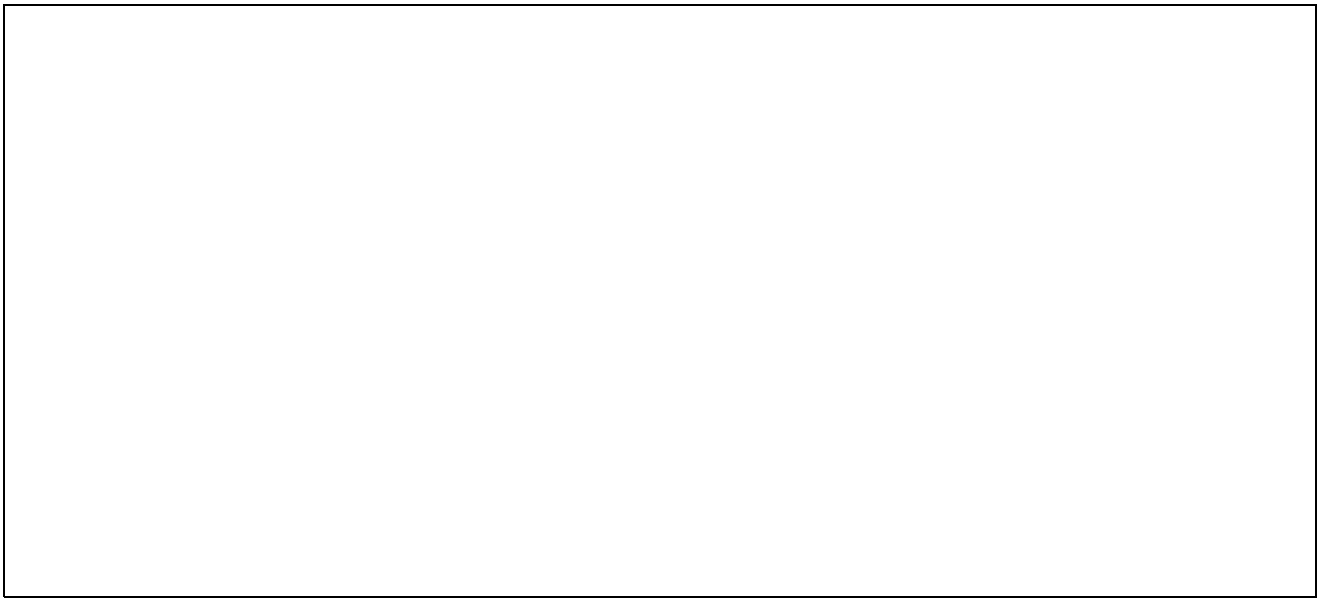
- Create HTML form
- Use servlet to fetch input using `request.getParameter()`
- Display data using `PrintWriter.`

Algorithm (Student Work Area):

Code (Student Work Area):

Question Bank:

1. How do you capture form data in a servlet?
2. What is the difference between `doGet()` and `doPost()`?
3. How do you validate user input?
4. What happens if a field is empty?
5. How can you handle multiple form fields?



EXPERIMENT NO. 5

Objective(s):
To create and read cookies in a servlet.

Outcome:
Students will understand session management using cookies.

Problem Statement:

Write a servlet to create and retrieve cookies.

Background Study:

Introduction:
Cookies are key-value pairs stored on the client's browser for tracking sessions.

Steps:

- Use `new Cookie(name, value)`
- Use `response.addCookie()`
- Use `request.getCookies()` to read them

Algorithm (Student Work Area):

Code (Student Work Area):

Question Bank:

1. What is a cookie in web programming?
2. How do you create a cookie in a servlet?
3. How are cookies retrieved?
4. What is the default lifespan of a cookie?
5. How do cookies differ from sessions?

EXPERIMENT NO. 6

Objective(s):
To implement session tracking using HttpSession.

Outcome:
Students will understand how to maintain session state using HttpSession.

Problem Statement:

Write a servlet to track session using HttpSession.

Background Study:

Introduction:

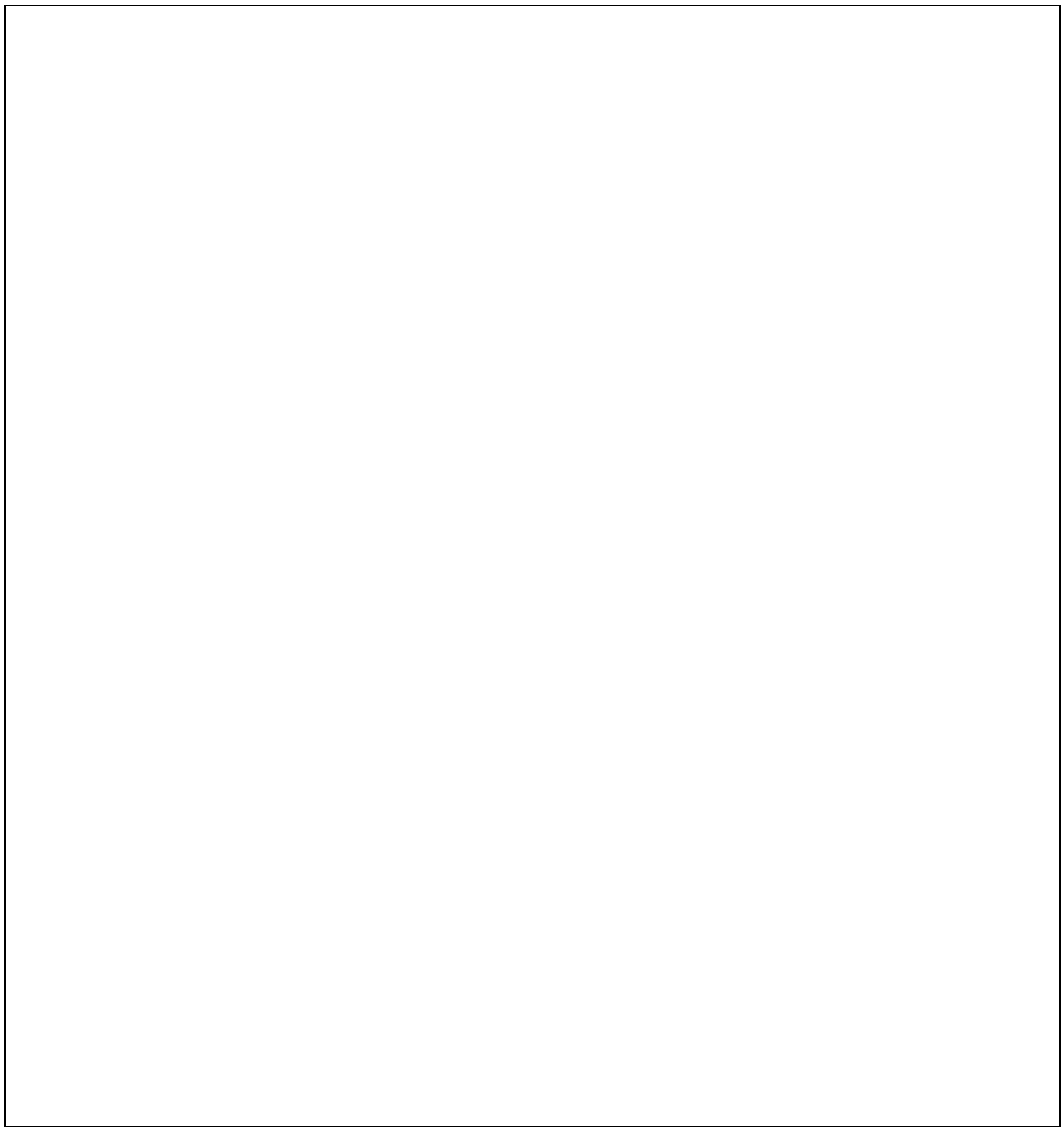
`HttpSession` stores user-specific data across multiple requests.

Steps:

- Use `request.getSession()`
- Store using `session.setAttribute()`
- Retrieve using `session.getAttribute()`
- End session using `session.invalidate()`

Algorithm (Student Work Area):

Code (Student Work Area):



Question Bank:

1. What is session tracking?
2. How is HttpSession different from cookies?
3. How do you invalidate a session?
4. What are alternative session tracking methods?
5. How secure is HttpSession?

EXPERIMENT NO. 7

Objective(s):
To implement a chat server using ServerSocket and Socket.

Outcome:
Students will understand client-server communication using Java networking.

Problem Statement:

Write a chat server program using `ServerSocket` and `Socket` classes.

Background Study:

Introduction:
Java provides `ServerSocket` for servers and `Socket` for clients to exchange data.

Steps:

- Create `ServerSocket` on a port
- Accept connections using `accept()`
- Communicate using input/output streams

Algorithm (Student Work Area):

Code (Student Work Area):

Question Bank:

- What are `ServerSocket` and `Socket`?
- How do you handle multiple clients?
- What streams are used in sockets?
- What are the advantages of TCP?
- What is blocking I/O?

EXPERIMENT NO. 8

Objective(s):
To send username and password using HTML form and authenticate using servlet.

Outcome:
Students will perform form validation and user authentication.

Problem Statement:

Write a servlet that accepts login credentials and authenticates user..

Background Study:

Introduction:

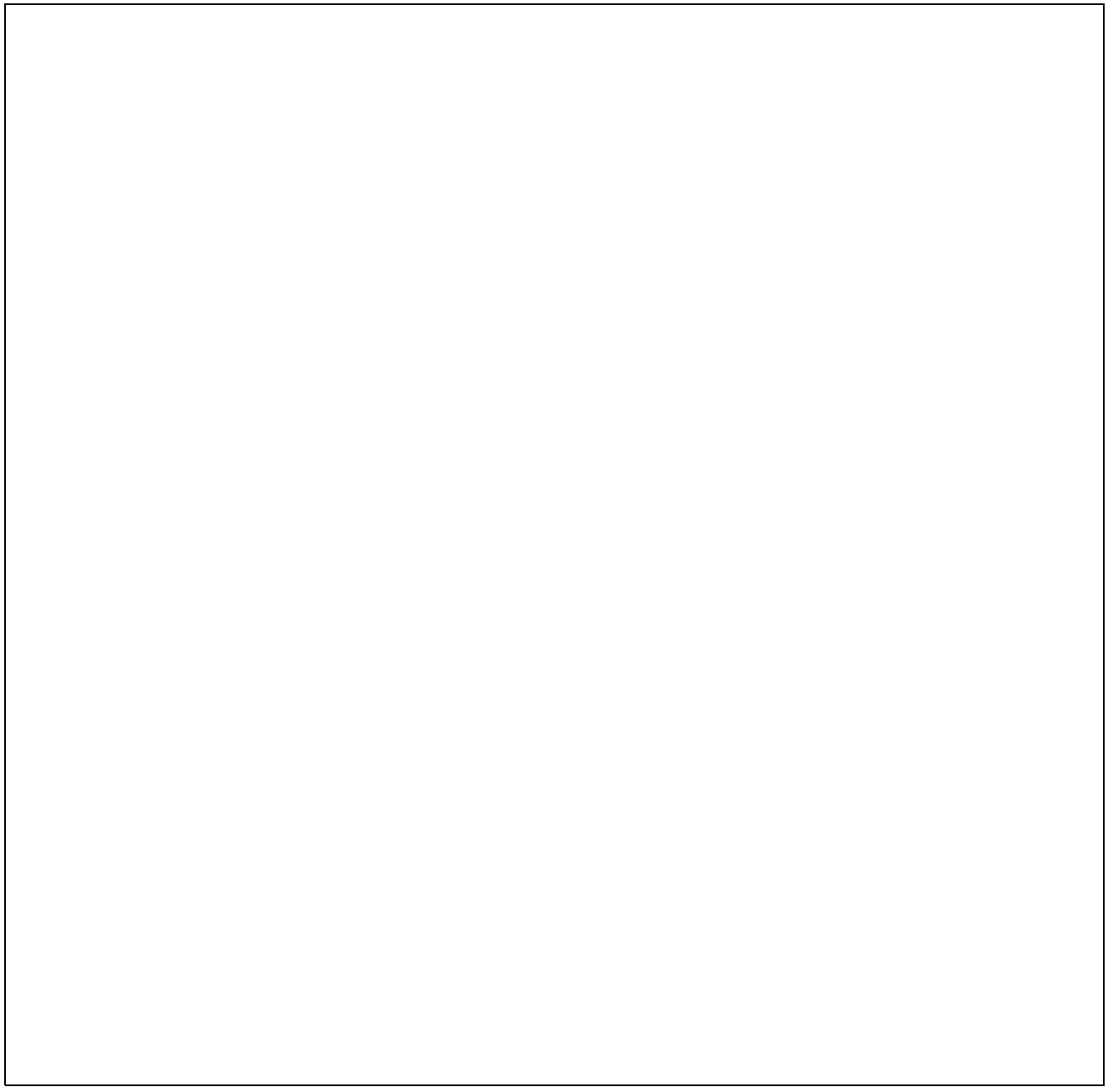
HTML form data can be verified using a servlet by checking credentials from hardcoded values or a database.

Steps:

- Read credentials using `request.getParameter()`
- Validate with stored data
- Redirect to success or error page

Algorithm (Student Work Area):

Code (Student Work Area):



Question Bank:

1. How do you validate login details in a servlet?
2. How is authentication handled securely?
3. What happens on failed login?
4. How can credentials be stored?
5. What is hashing and why is it important?

EXPERIMENT NO. 9

Objective(s):
To perform arithmetic operations in JSP.

Outcome:
Students will understand embedding Java logic in web pages using JSP.

Problem Statement:

Write a JSP program to perform arithmetic operations.

Introduction:
JSP uses scripting elements like `<% %>`, `<%= %>` for logic and expression display.

Algorithm (Student Work Area):

Code (Student Work Area):

Question Bank:

1. What is JSP?
2. How do you insert logic in JSP?
3. What is the use of scriptlet tags?
4. How are HTML forms processed in JSP?
5. What are the advantages of JSP over Servlets?

EXPERIMENT NO. 10

Objective(s):
To demonstrate use of `jsp:forward` tag and request object.

Outcome:
Students will understand request redirection and data transfer between JSPs.

Problem Statement:

Write a JSP program to use `jsp:forward` and request object.

Background Study:

Introduction:
`jsp:forward` forwards control to another resource.
`request` object carries data between pages.

Algorithm (Student Work Area):

Code (Student Work Area):

Question Bank:

1. What is the purpose of `jsp:forward`?
2. How is data passed using request?
3. What are implicit objects in JSP?
4. Difference between forward and redirect?
5. How is request used to access form data?

EXPERIMENT NO. 11

Objective(s):
CRUD Using Hibernate

Outcome:
To store and manage data using Hibernate ORM.

Problem Statement:

Students will perform CRUD operations using Hibernate framework.

Background Study:

Introduction:
Hibernate maps Java objects to database tables using annotations or XML.

Steps:

- **Configure `hibernate.cfg.xml`**
- **Create annotated POJO**
- **Use `SessionFactory` to persist data**

Algorithm (Student Work Area):

Code (Student Work Area):

Question Bank:

- What is Hibernate?
- What are POJOs and entities?
- How is configuration done in Hibernate?
- How are transactions handled?
- What are benefits of ORM?

EXPERIMENT NO. 12

Objective(s):
CRUD Using Spring MVC

Outcome:
To build a web application using Spring MVC for CRUD.

Problem Statement:

Create a Spring MVC application to perform CRUD.

Background Study:

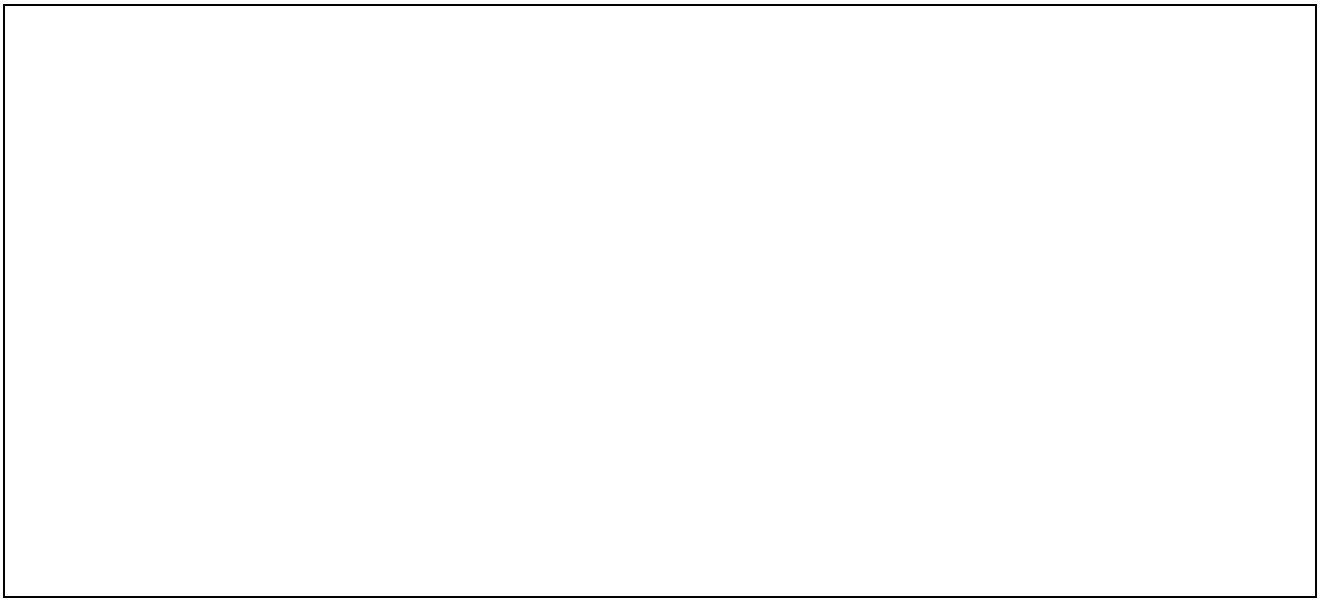
Introduction:
Spring MVC separates model, view, and controller logic for better maintainability.

Algorithm (Student Work Area):

Code (Student Work Area):

Question Bank:

1. What is Spring MVC?
2. Role of `@Controller`, `@Service`, and `@Repository`?
3. How is form data handled?
4. How do you create REST endpoints?
5. What is the dispatcher servlet?



EXPERIMENT NO. 13

Objective(s):
CRUD Using Spring Boot

Outcome:
Learn to build full-stack Java apps quickly using Spring Boot and JPA.

Problem Statement:

Create a Spring Boot web application with CRUD functionality.

Background Study:

Introduction:

Spring Boot simplifies Spring app development with embedded servers and minimal configuration.

Algorithm (Student Work Area):

Code (Student Work Area):

Question Bank:

- What is Spring Boot?
- How is database connectivity configured?
- What is JPA?
- Role of `@RestController` and `@Repository`?
- What is the use of `application.properties`?