AN INTERNSHIP REPORT

ON

WEB DEVELOPMENT WITH JAVA

Submitted by

RATHOD MANAV

(200130107042)

Student of

Bachelors of Engineering

in

Computer Engineering Department

Government Engineering College Gandhinagar, Sector-28

Summer Internship (3170001)

DECLARATION

I do hereby solemnly declare that the work presented in this Internship Report has been carried out by me and has not been previously submitted to any other University, College, and Organization for any academic Qualification and Certificate.

I hereby warrant that the work I have presented does not breach any existing copyright acts.

Rathod Manav

GOVERNMENT ENGINEERING COLLEGE GANDHINAGAR
SECTOR – 28

CERTIFICATE

This is to certify that the "**Internship report**" submitted by **RATHOD MANAV (Enroll. No: 200130107042)** is work done by her and submitted during 2023 - 2024 academic year, in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF ENGINEERING in COMPUTER ENGINEERING, at TECHMICRA IT SOLUTION, AHMEDABAD**

Internal Guide
Prof. N K Prajapati

Head of The Department
Dr. D A Parikh

CONFIRMATION LETTER

techmicra

Date: 19th July 2023

To,
Head of Computer Engineering Department.
Govt. Engineering College Gandhinagar,
Sector 28 GIDC, Sector 28, Gandhinagar,
Gujarat 382028

Subject: Confirmation Letter of Internship at Techmicra IT Solutions.

Respected Sir,

We are glad to offer an internship to your student **Manav Rathod (Semester 7th) Enrollment No. 200130107042** of **Computer Engineering.** He will be working as an intern at our organization for Summer Internship from **27th July 2023 till 10th August 2023 in Java Web Development** ; along with that we will be helping him to clear his basics.

We wish best of luck for career and congratulation for becoming a member of our company.

Sincerely,

For, TECHMICRA IT SOLUTIONS
PROPRIETOR

Pallav Mamtora
CEO Techmicra IT Solutions
pallav@techmicra.co.in

CERTIFIED ISO 9001:2015 COMPANY
www.techmicra.co.in
info@techmicra.co.in | pallav@techmicra.co.in
+91-9727835207 | +91-9427617574
Office No. 12, Sanidhya Building, Oppo. UCO Bank, Ashram Road, Ahmedabad - 380009

CERTIFICATE OF COMPLETION

CERTIFICATE techmicra

This is to certify that **Mr. Manav Rathod (Enrollment No. 200130107042)** 7th Semester **Govt. Engineering College Gandhinagar**, Department of Computer Engineering has successfully completed his internship and task work at "Techmicra IT Solutions", Ahmedabad, during **27th July 2023 till 10th August 2023.**

As a part of the internship till now he has completed the **Web Development with Java** tasks under our companies' guidance

Getting started with different types of file and project structure anatomy in Java & J2EE, Environment setup & Core Programming Concepts in Java Mutability Execution framework, Thread pool, Callable, Future, Reentrant lock ,Collections(Arrays, List, Que) Collections(HASH MAP, HASH TABLE, HASH SET) hashing features,JAVA 8 New features : Interfaces Optional class JAVA 8 New features , Servlet, JSP, Session Management with Authentication, Working with J2EE & JDBC Crud operations (insert, retrieve, update, delete, sorting, searching, pagination, chart generation),Overview of Hibernate basics configuration and ORM implementation, Usage in Spring or Spring Boot, Spring environment setup and configurations. Bean its usage and scope ,Spring IOC, DI, Setter injections, Autowirig ,Maven(POM), DI and Autowiring, AOP, Event Handling ,Spring Boot configuration and required annotation ,Controller and RESTful controller, CRUD operation with hibernate in spring boot - 1 and CRUD operation with hibernate in spring boot - 2.

His understanding of problem context and technical knowledge for the tools used was up to the mark. During the project work, we found him sincere and the work done by him was commendable.

We wish him all the best in his future endeavors and hope that he will have a successful career.

Sincerely,

For, TECHMICRA IT SOLUTIONS

Techmicra IT Solutions CEO.

techmicra IT Solutions

CERTIFIED ISO 9001:2015 COMPANY

www.techmicra.co.in

info@techmicra.co.in | pallav@techmicra.co.in

+91-9727835207 | +91-9427617574

Office No. 12, Sanidhya Building, oppo. UCO Bank, Ashram Road, Ahmedabad - 380009

COMPANY PROFILE

We are India's top organization for furnishing the best assets expansion administrations with 100 percent guaranteed results.

Since our foundation, we have assisted 100+ little organizations and 50+ large undertakings with accomplishing their business objectives by offering excellent IT ability.

We are tied in with tracking down the perfect individuals, with the right abilities, for the right work.

Having the right-sized group is basic to any fruitful undertaking and, as they enter new periods of development, once in a while important to add new individuals are perfect for the gig.

Our IT staffing administrations give the speed, proficient sharpness and adaptability your group needs proportional rapidly and complete improvement on time.

Our objective is to sustain with exponential growth in IT industry. Our mission is to deliver the best with top notch quality every quarter and vision is to develop a product with one of its kind concepts which could be used by millions of people.

ACKNOWLEDGEMENT

The internship opportunity I had with Techmicra was a great chance for me to learning and professional development. Therefore, I am very thankful to Techmicra for giving this wonderful opportunity.I am very thankful to Mr. Pallav Mamtora sir and all faculty members of Information Technology department for this amazing learning experience and all this help in completing internship.It was great experience at Techmicra, learned a lot and professional experience with amazing friendly team of company. Thanks to company for this all.

ABSTRACT

At Techmicra I have learned web development (frontend and backend) using JAVA. In this firstly learned concepts of core JAVA(J2SE) then J2EE's concepts of Java Database Connectivity, Java Server Pages, Servlets, Java Standard Tag Library, Hibernate and Spring and also included HTML and CSS. They taught all this topics with examples.

TABLE OF CONTENT

	- Collections(HASH MAP,HASH TABLE, HASH SET)	
WEEK 1	**02 AUGUST 2023**	**13**
	- Java Networking	
	- Assignment Task 1	
WEEK 2	**03 AUGUST 2023**	**17**
	- Servlet	
	- Web Discriptor file	
WEEK 2	**04 AUGUST 2023**	**20**
	- JSP	
	- JSP Core Tag Library	
WEEK 2	**07 AUGUST 2023**	**24**
	- Session Management with Authentication	
	- Working with J2EE & JDBC Crud operation (insert, retrieve, update, delete, sorting, searching, pagination, chart generation)	
WEEK 2	**08 AUGUST 2023**	**26**
	- Overview of Hibernate basics configuration and ORM implementation	
	- Spring (Spring Boot , Spring IOC)	
WEEK 2	**09 AUGUST 2023**	**29**
	- Spring Dependency Injection	

	- Types of Spring Dependency Injection	
WEEK 2	**10 AUGUST 2023**	**31**
	- Conclusion	

Week 1: 27 July 2023

➢ Basic of Java:

Java program is an object-oriented programming language, that means java is the collection of objects, and these objects communicate through method calls to each other to work together.

➢ Java Variables:

Variables are containers for storing data values.

In Java, there are different **types** of variables, for example:

- **String** - stores text, such as "Hello". String values are surrounded by double quotes
- **int** - stores integers (whole numbers), without decimals, such as 123 or -123
- **float** - stores floating point numbers, with decimals, such as 19.99 or -19.99
- **char** - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- **boolean** - stores values with two states: true or false

➢ Java Datatype

Data types are divided into two groups:

- Primitive data types - includes byte, short, int, long, float, double, boolean and char
- Non-primitive data types - such as String, Arrays and Classes (you will learn more about these in a later chapter)

➢ Java Operators

Operators are used to perform operations on variables and values.

Java divides the operators into the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Bitwise operators

➢ Java Conditional Statements

You already know that Java supports the usual logical conditions from mathematics:

- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b
- Equal to a == b
- Not Equal to: a != b

You can use these conditions to perform different actions for different decisions.

Java has the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

➢ Java Loops

In computer programming, loops are used to repeat a block of code. For example, if you want to show a message 100 times, then rather than typing the same code 100 times, you can use a loop.

In Java, there are three types of loops.

- for loop
- while loop
- do...while loop

Week 1: 28 July 2023

➢ OOPs Concept in Java

Object-Oriented Programming System (OOPs) is a programming concept that works on the principles of abstraction, encapsulation, inheritance, and polymorphism. It allows users to create objects they want and create methods to handle those objects. The basic concept of OOPs is to create objects, re-use them throughout the program, and manipulate these objects to get results.

The following are general OOPs concepts in Java:

1) Class

The class is one of the Basic concepts of OOPs which is a group of similar entities. It is only a logical component and not the physical entity. Lets understand this one of the OOPs Concepts with example, if you had a class called "Expensive Cars" it could have objects like Mercedes, BMW, Toyota, etc. Its properties(data) can be price or speed of these cars. While the methods may be performed with these cars are driving, reverse, braking etc.

2) Object

An object can be defined as an instance of a class, and there can be multiple instances of a class in a program. An Object is one of the Java OOPs concepts which contains both the data and the function, which operates on the data. For example – chair, bike, marker, pen, table, car, etc.

3) Inheritance

Inheritance is one of the Basic Concepts of OOPs in which one object acquires the properties and behaviors of the parent object. It's creating a parent-child relationship between two classes. It offers robust and natural mechanism for organizing and structure of any software.

4) Polymorphism

Polymorphism refers to one of the OOPs concepts in Java which is the ability of a variable, object or function to take on multiple forms. For example, in English, the verb *run* has a different meaning if you use it with *a laptop*, *a foot race*, and *business*. Here, we understand the meaning of *run* based on the other words used along with it. The same also applied to Polymorphism.

5) Abstraction

Abstraction is one of the OOP Concepts in Java which is an act of representing essential features without including background details. It is a technique of creating a new data type that is suited for a specific application. Lets understand this one of the OOPs Concepts with example, while driving a car, you do not have to be concerned with its internal working. Here you just need to concern about parts like steering wheel, Gears, accelerator, etc.

6) Encapsulation

Encapsulation is one of the best Java OOPs concepts of wrapping the data and code. In this OOPs concept, the variables of a class are always hidden from other classes. It can only be accessed using the methods of their current class. For example – in school, a student cannot exist without a class.

7) Association

Association is a relationship between two objects. It is one of the OOP Concepts in Java which defines the diversity between objects. In this OOP concept, all objects have their separate lifecycle, and there is no owner. For example, many students can associate with one teacher while one student can also associate with multiple teachers.

8) Aggregation

In this technique, all objects have their separate lifecycle. However, there is ownership such that child object can't belong to another parent object. For example consider class/objects department and teacher. Here, a single teacher can't belong to multiple departments, but even if we delete the department, the teacher object will never be destroyed.

9) Composition

Composition is a specialized form of Aggregation. It is also called "death" relationship. Child objects do not have their lifecycle so when parent object deletes all child object will also delete automatically. For that, let's take an example of House and rooms. Any house can have several rooms. One room can't become part of two different houses. So, if you delete the house room will also be deleted.

Week 1 :31 July 2023:

➢ Exception Handling

- When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.
- When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an **exception** (throw an error).

➢ Java Try and Catch

- The try statement allows you to define a block of code to be tested for errors while it is being executed.
- The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

Example :

```
public class Main {
  public static void main(String[] args) {
    try {
      int[ ] myNumbers = {1, 2, 3};
      System.out.println(myNumbers[10]);
    } catch (Exception e) {
      System.out.println("Something went wrong.");
    }
  }
}
```

➢ String Builder

- StringBuffer is a class in Java that represents a mutable sequence of characters. It provides an alternative to the immutable String class, allowing you to modify the contents of a string without creating a new object every time.

Here are some important features and methods of the StringBuffer class:

1. StringBuffer objects are mutable, meaning that you can change the contents of the buffer without creating a new object.
2. The initial capacity of a StringBuffer can be specified when it is created, or it can be set later with the ensureCapacity() method.
3. The append() method is used to add characters, strings, or other objects to the end of the buffer.
4. The insert() method is used to insert characters, strings, or other objects at a specified position in the buffer.
5. The delete() method is used to remove characters from the buffer.
6. The reverse() method is used to reverse the order of the characters in the buffer.

Example:

```
public class StringBufferExample {
public static void main(String[] args)
{
StringBuffer sb = new StringBuffer();
sb.append("Hello");
sb.append(" ");
sb.append("world");
String message = sb.toString();
System.out.println(message);
}
}
```

➢ String Builder

- StringBuilder in Java is a class used to create a mutable, or in other words, a modifiable succession of characters. Like StringBuffer, the StringBuilder class is an alternative to the Java Strings Class, as the Strings class provides an immutable succession of characters. However, there is one significant difference between StringBuffer and StringBuilder, and it is that the latter is non-synchronized. It means that StringBuilder in Java is a more suited choice while working with a single thread, as it will be quicker than StringBuffer.

Example:

```
public class Main {

  public static void main(String[] args) {

    StringBuilder sb1 = new StringBuilder();
    System.out.println("Initial Capacity of sb1: " + sb1.capacity());

    StringBuilder sb2 = new StringBuilder(10);
    System.out.println("Initial Capacity of sb2: " + sb2.capacity());

    CharSequence seq = "Scaler";
    StringBuilder sb3 = new StringBuilder(seq);
    System.out.println("sb3: " + sb3);

    StringBuilder sb4 = new StringBuilder("Scaler");
    System.out.println("sb4: " + sb4);
  }
}
```

➢ File Operation in Java

The following are the several operations that can be performed on a file in Java :

- **Create a File**
- **Read from a File**
- **Write to a File**
- **Delete a File**

Now let us study each of the above operations in detail.

1. Create a File

- In order to create a file in Java, you can use the createNewFile() method.
- If the file is successfully created, it will return a Boolean value true and false if the file already exists.

2. Read from a File: We will use the Scanner class in order to read contents from a file.

3. Write to a File: We use the FileWriter class along with its write() method in order to write some text to the file.

4. Delete a File: We use the delete() method in order to delete a file.

Week 1: 01 August 2023:

➢ Collection Framework in Java

Collections in programming refer to data structures or classes that allow you to store, organize, and manipulate a collection of objects. In Java, for example, there are several collections frameworks and classes, including Arrays, Lists, and Queues. Let's briefly discuss each of them:

1. Arrays:

- An array is a fixed-size, ordered collection of elements of the same data type.
- Once you define the size of an array, you cannot change it.
- Elements in an array are accessed by their index, which starts from 0.
- Arrays are suitable when you know the exact number of elements you need to store and when you don't need to dynamically resize the collection.

Example :

```
int[] numbers = new int[5];
numbers[0] = 1;
```

2. Lists:

- Lists are dynamic collections that can grow or shrink in size.
- In Java, common implementations of lists include ArrayList and LinkedList.
- Lists allow you to store elements of different data types and provide methods for adding, removing, and accessing elements.

Example :

```
import java.util.ArrayList;
ArrayList<String> names = new ArrayList<>();
names.add("Alice");
names.add("Bob");
names.remove("Alice");
```

3. Queue:

- A queue is a data structure that follows the FIFO (First-In-First-Out) principle.
- It's often used for tasks like managing tasks or events in a specific order.
- In Java, you can use implementations like LinkedList or PriorityQueue to create queues.

Example :

```
import java.util.LinkedList;
import java.util.Queue;
Queue<String> queue = new LinkedList<>();
queue.offer("Task 1");
queue.offer("Task 2");
String task = queue.poll(); // Retrieves and removes "Task 1"
```

4. HashMap:

- HashMap is a data structure that implements the Map interface in Java.
- It stores key-value pairs and allows for quick retrieval of values based on their keys.
- Keys are unique within a HashMap, and they are used to index and locate corresponding values.
- HashMap does not guarantee order, and the order of elements may change over time, but it offers fast O(1) average time complexity for basic operations like adding, retrieving, and deleting elements.

Example :

```
import java.util.HashMap;
HashMap<String, Integer> scores = new HashMap<>();
scores.put("Alice", 95);
scores.put("Bob", 88);
int aliceScore = scores.get("Alice"); // Retrieves the value associated with the key "Alice"
```

5. HashTable:

- HashTable is similar to HashMap but is considered legacy, and it is synchronized, which means it's thread-safe.
- Like HashMap, it stores key-value pairs with unique keys.
- While it provides synchronization, making it safe for concurrent use, this can introduce some performance overhead.
- It's recommended to use HashMap or other modern alternatives if thread safety is not a primary concern.

Example :

```
import java.util.Hashtable;

Hashtable<String, Integer> scores = new Hashtable<>();

scores.put("Alice", 95);

scores.put("Bob", 88);

int aliceScore = scores.get("Alice"); // Retrieves the value associated with the key "Alice"
```

6. HashSet:

- HashSet is an implementation of the Set interface in Java.
- It stores a collection of unique elements, similar to a mathematical set.
- Unlike HashMap and HashTable, it doesn't store key-value pairs; it only stores individual elements.
- HashSet uses the hash code of elements to efficiently determine uniqueness and ensure that duplicate elements are not allowed.

Example :

```
import java.util.HashSet;

HashSet<String> names = new HashSet<>();

names.add("Alice");

names.add("Bob");

names.add("Alice"); // This won't add a duplicate "Alice" to the set
```

Week 1: 02 August 2023:

➢ Java Networking

The term *network programming* refers to writing programs that execute across multiple devices (computers), in which the devices are all connected to each other using a network.

The java.net package of the J2SE APIs contains a collection of classes and interfaces that provide the low-level communication details, allowing you to write programs that focus on solving the problem at hand.

The java.net package provides support for the two common network protocols −

- **TCP** − TCP stands for Transmission Control Protocol, which allows for reliable communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.
- **UDP** − UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitted between applications.

Assignment Task 1:

AIM: Create chat application using either TCP protocol.

Code:

 # TCP_CLIENT

```
import java.io.*;
import java.net.Socket;
import java.util.Scanner;
public class P2_TCPClient {
	public static void main(String[] args) throws Exception
	{
		Socket s=new Socket("localhost",7888);
		if(s.isConnected())
		{
			System.out.println("Connected to server");
		}
		DataInputStream msg=new DataInputStream(System.in);
		String str="Start
		DataOutputStream dout=new DataOutputStream(s.getOutputStream());
		dout.writeUTF(str);
		System.out.println(str);
		DataInputStream din=new DataInputStream(s.getInputStream());
		while(true)
		{
			System.out.print("Client:\t");
			str=msg.readLine();
			dout.writeUTF(str+"\n");
			str=din.readUTF();
			System.out.println("Server:\t"+str);
		}
	}
}
```

 # TCP_SERVER

```
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class P2_TCPServer {



	public static void main(String[] args) throws Exception
	{
		ServerSocket ss=new ServerSocket(7888);
		Socket s=ss.accept();
		DataInputStream din=new DataInputStream(s.getInputStream());
		String str;
		str=din.readUTF();
		System.out.println("Client:\t"+str);
		DataOutputStream dout=new DataOutputStream(s.getOutputStream());
		DataInputStream msg=new DataInputStream(System.in);
		while(true)
		{
			str=din.readUTF();
			System.out.print("Client:\t"+str);
			System.out.print("Server:\t");
			str=msg.readLine();
			dout.writeUTF(str);
		}
	}
}
```

Output:

```
---------------------< com.mycompany:TCPPractice >---------------------
Building TCPPractice 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.1.0:exec (default-cli) @ TCPPractice ---
Connected to server
Start Chat.....................................................
Client: hii
Server: hello
Client: how are you?
Server: i am fine. Thank you
```

Week 2: 3 August 2023:

➢ Servlet

A servlet is a Java-based, platform-independent component that extends the capabilities of a web server. It's designed to handle dynamic content creation and management within web applications. Servlets are a fundamental part of Java Enterprise Edition (Java EE) and are commonly used for building web-based applications, including websites and web services.

Here are some key points and characteristics of servlets:

1. **Server-Side Java Components**: Servlets are Java classes that are executed on the server-side. They run in a web container (e.g., Apache Tomcat, Jetty) within a web server (e.g., Apache HTTP Server) and handle HTTP requests and responses.
2. **Dynamic Content:** Servlets are primarily used for generating dynamic content. They can process client requests, interact with databases, perform business logic, and generate HTML or other data formats to be sent back to the client's web browser.
3. **Lifecycle**: Servlets have a well-defined lifecycle that includes methods for initialization, handling requests, and destruction. Commonly used methods include init(), service(), and destroy().
4. **Thread Safety:** Servlets are designed to be thread-safe, meaning they can handle multiple client requests simultaneously without issues. Each request to a servlet typically runs in its own thread.
5. **Mapping:** Servlets are mapped to specific URLs or URL patterns in the web application's deployment descriptor (web.xml). When a client sends an HTTP request matching a URL pattern, the corresponding servlet is invoked.

Example:

Code:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(urlPatterns = {"/table"})
public class table extends HttpServlet {
   public void doGet(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException{

       res.setContentType("text/html");
      PrintWriter pw = res.getWriter();
       pw.println("<html>");
      pw.println("<head><title>Employee Database</title></head>");
      pw.println("<body>");
         pw.println("<table border=1>");
         pw.println("<tr><td>emp_id</td><td>Ename</td><td>Email</td><td>Age</td></tr>");
   }
   }
```

Output:

➢ WEB Discriptor

- Java web applications use a deployment descriptor file to determine how URLs map to servlets, which URLs require authentication, and other information. This file is named web.xml, and resides in the app's WAR under the WEB-INF/ directory.

- A web application's deployment descriptor describes the classes, resources and configuration of the application and how the web server uses them to serve web requests. When the web server receives a request for the application, it uses the deployment descriptor to map the URL of the request to the code that ought to handle the request.

➢ The deployment descriptor is a file named web.xml. It resides in the app's WAR under the WEB-INF/ directory. The file is an XML file whose root element is <web-app>.

web.xml File :-

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee">
   <servlet>
      <servlet-name>WebServlet</servlet-name>
      <servlet-class>WebServlet</servlet-class>
   </servlet>
   <welecome-file-list>
      <welcome-file>MainPage.html</welcome-file>
        <welcome-file>index.html</welcome-file>
   </welecome-file-list>

   <servlet-mapping>
      <servlet-name>WebServlet</servlet-name>
      <url-pattern>/WebServlet</url-pattern>
   </servlet-mapping>
</web-app>
```

Week 2: 4 August 2023:

➢ JSP

- JSP, or JavaServer Pages, is a technology used for developing web-based applications in Java. It allows you to embed Java code within HTML web pages, making it easier to create dynamic and interactive web content.

Example:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page language="java" import="java.sql.*" %>
<%@page import="java.io.*,java.util.*" %>

<%
   Class.forName("com.mysql.cj.jdbc.Driver");%>
   <%
   Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/student1",
"root", "");

  out.write("Connected to mysql!!!");
   Statement stmt = con.createStatement();
   ResultSet rs = stmt.executeQuery("select *from student");
%>

<html>
   <body>
   <center>
      <h2>Student subject list..</h2>
      <tr>
         <th>SUB_CODE</th>
         <th>SUBJECT NAME</th>
         <th>MIN_MARKS</th>
         <th>MAX_MARKS</th>
         <th>MARKS OBTAINED</th>
      </tr>
```

```
    <%      while (rs.next()) {

    %>
    <tr>
        <td><%=rs.getInt("Sub_code")%></td>
        <td><%=rs.getString("Sub_name")%></td>
        <td><%=rs.getInt("Min_marks")%></td>
        <td><%=rs.getInt("Max_marks")%></td>
        <td><%=rs.getInt("Marks_Obtained")%></td>

    </tr>
    <%
        }
        rs.close();
        stmt.close();
        con.close();
    %>
  </table>
</center>
</body>
</html>
```

Output:

Sub_code	Sub_name	Min_marks	Max_marks	Marks_Obtained
101	TOC	23	70	45
102	AJ	23	70	50
103	IOT	23	70	57
104	MPI	23	70	35
105	CPDP	23	70	61

➢ Java Core Tag Library

- The core tag is used nearly in all the web application. This tag is useful for performing basic input and output, for looping operation or for evaluating expressions.
- Various tags used in core library are-

1. <c:out>
2. <c:set>
3. <c:if>
4. <c:choose>
5. <c:when>
6. <c:otherwise>
7. <c:forEach>
8. <c:forTokens>
9. <c:url>
10. <c:redirect>

Example:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
  <title>JSP Core Tag Library Example</title>
</head>
<body>
  <c:set var="name" value="John" />
  <c:if test="${name eq 'John'}">
    <p>Welcome, ${name}!</p>
  </c:if>
  <ul>
    <c:forEach var="i" begin="1" end="5">
      <li>Item ${i}</li>
    </c:forEach>
  </ul>
</body>
</html>
```

Output:

Welcome, John!

Item 1

Item 2

Item 3

Item 4

Item 5

Week 2: 07 August 2023:

➢ Session Management with Authentication

- Session management is the process of maintaining user-specific data across multiple HTTP requests in a stateless HTTP protocol. In Java web applications, you can implement session management using HttpSession, which allows you to store and retrieve user-related data on the server.

  1. Create a Servlet for Authentication: You can create a servlet that handles user login and authentication. In the doPost method of this servlet, you can validate the user's credentials (e.g., username and password) against a database or any other authentication mechanism.

  2. Start a Session: Upon successful authentication, create an HttpSession object using request.getSession(true). This creates a new session or retrieves an existing one if available.

  3. Store User Information: Store relevant user information in the session, such as the user's ID or username. For example:

  HttpSession session = request.getSession();
  session.setAttribute("userId", userId);

  4. Session Timeout: Configure the session timeout based on your application's requirements. You can do this in your web.xml configuration or programmatically.

Authentication:

- Authentication is the process of verifying the identity of a user. In an advanced Java web application, you typically use a combination of techniques to authenticate users:

  1. Database Authentication: Verify user credentials (username and password) against a database. You can use JDBC to connect to the database and check if the provided credentials match those stored in the database.

  2. Session Tracking: After successful authentication, store a user's identity in the session, as mentioned earlier. Use this information to identify users and manage their access rights.

  3. Authorization: Implement authorization to control what authenticated users can access. Define roles and permissions, and check these permissions when processing requests. You can use annotations or programmatic checks for this purpose.

  4. Secure Password Storage: Ensure that user passwords are securely hashed and salted before storing them in the database. Use strong encryption algorithms and follow best practices for password security**.**

➢ CRUD Operation with JDBC

- Creating, reading, updating, and deleting data in a database is a common task in many applications, and JDBC (Java Database Connectivity) is a Java API that allows you to connect to a database and perform these operations. In this blog post, we will walk through the steps of setting up a simple CRUD (create, read, update, delete) operation using JDBC.

1. Connect to the database

The first step is to establish a connection to the database. You can do this by loading the JDBC driver and creating a connection object.

2. Create a new record

Once you have a connection to the database, you can use the connection object to create a new record in the database. To do this, you will need to use an SQL INSERT statement and execute it using the connection object.

3. Read a record

To read a record from the database, you will need to use an SQL SELECT statement and execute it using the connection object. The result of the query will be a ResultSet object that you can use to access the data in the record.

4. Update a record

To update a record in the database, you will need to use an SQL UPDATE statement and execute it using the connection object.

5. Delete a record

To delete a record from the database, you will need to use an SQL DELETE statement and execute it using the connection object.

Week 2: 08 August 2023:

➢ Hibernate Framework

- Hibernate is a framework which provides some abstraction layer, meaning that the programmer does not have to worry about the implementations, Hibernate does the implementations for you internally like Establishing a connection with the database, writing query to perform CRUD operations etc.
- It is a java framework which is used to develop persistence logic. Persistence logic means to store and process the data for long use. More precisely Hibernate is an open-source, non-invasive, light-weight java ORM(Object-relational mapping) framework to develop objects which are independent of the database software and make independent persistence logic in all JAVA, JEE.

- Framework means it is special install-able software that provides an abstraction layer on one or more technologies like JDBC, Servlet, etc to simplify or reduce the complexity for the development process.

Functionalities supported by Hibernate framework

- Hibernate framework support Auto DDL operations. In JDBC manually we have to create table and declare the data-type for each and every column. But Hibernate can do DDL operations for you internally like creation of table,drop a table,alter a table etc.
- Hibernate supports Auto Primary key generation. It means in JDBC we have to manually set a primary key for a table. But Hibernate can this task for you.
- Hibernate framework is independent of Database because it supports HQL (Hibernate Query Language) which is not specific to any database, whereas JDBC is database dependent.
- In Hibernate, Exception Handling is not mandatory, whereas In JDBC exception handling is mandatory.
- Hibernate supports Cache Memory whereas JDBC does not support cache memory.
- Hibernate is a ORM tool means it support Object relational mapping. Whereas JDBC is not object oriented moreover we are dealing with values means primitive data. In hibernate each record is represented as a Object but in JDBC each record is nothing but a data which is nothing but primitive values.

➢ Spring

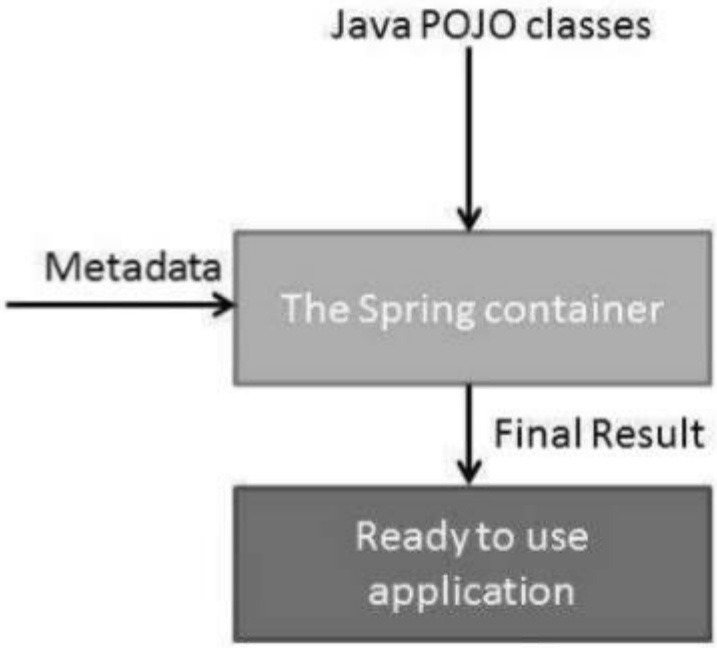
- Spring is an open-source lightweight framework that allows Java EE 7 developers to build simple, reliable, and scalable enterprise applications. This framework mainly focuses on providing various ways to help you manage your business objects. It made the development of Web applications much easier than compared to classic Java frameworks and Application Programming Interfaces (APIs), such as Java database connectivity (JDBC), JavaServer Pages(JSP), and Java Servlet. This framework uses various new techniques such as Aspect-Oriented Programming (AOP), Plain Old Java Object (POJO), and dependency injection (DI), to develop enterprise applications. The Spring framework can be considered as a collection of sub-frameworks, also called layers, such as Spring AOP. Spring Object-Relational Mapping (Spring ORM). Spring Web Flow, and Spring Web MVC. You can use any of these modules separately while constructing a Web application. The modules may also be grouped together to provide better functionalities in a Web application.

1. Spring Boot

- Spring Boot is built on top of the conventional spring framework. So, it provides all the features of spring and is yet easier to use than spring. Spring Boot is a microservice-based framework and making a production-ready application in very less time. In Spring Boot everything is auto-configured. We just need to use proper configuration for utilizing a particular functionality. Spring Boot is very useful if we want to develop REST API.

2. Spring IOC

- The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete life cycle from creation till destruction. The Spring container uses DI to manage the components that make up an application. These objects are called Spring Beans, which we will discuss in the next chapter.
- The container gets its instructions on what objects to instantiate, configure, and assemble by reading the configuration metadata provided. The configuration metadata can be represented either by XML, Java annotations, or Java code. The following diagram represents a high-level view of how Spring works. The Spring IoC container makes use of Java POJO classes and configuration metadata to produce a fully configured and executable system or application.



Spring provides the following two distinct types of containers.

Sr.No.	Container & Description
1	**Spring BeanFactory Container** This is the simplest container providing the basic support for DI and is defined by the *org.springframework.beans.factory.BeanFactory* interface. The BeanFactory and related interfaces, such as BeanFactoryAware, InitializingBean, DisposableBean, are still present in Spring for the purpose of backward compatibility with a large number of third-party frameworks that integrate with Spring.
2	**Spring ApplicationContext Container** This container adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by the *org.springframework.context.ApplicationContext* interface.

Week 2: 09 August 2023:

➢ Spring Dependency Injection

- Dependency Injection is the main functionality provided by Spring IOC(Inversion of Control). The Spring-Core module is responsible for injecting dependencies through either Constructor or Setter methods. The design principle of Inversion of Control emphasizes keeping the Java classes independent of each other and the container frees them from object creation and maintenance. These classes, managed by Spring, must adhere to the standard definition of Java-Bean. Dependency Injection in Spring also ensures loose-coupling between the classes.

Need for Dependency Injection:

- Suppose class One needs the object of class Two to instantiate or operate a method, then class One is said to be dependent on class Two. Now though it might appear okay to depend a module on the other but, in the real world, this could lead to a lot of problems, including system failure. Hence such dependencies need to be avoided.

- Spring IOC resolves such dependencies with Dependency Injection, which makes the code easier to test and reuse. Loose coupling between classes can be possible by defining interfaces for common functionality and the injector will instantiate the objects of required implementation. The task of instantiating objects is done by the container according to the configurations specified by the developer.

Types of Spring Dependency Injection:

There are two types of Spring Dependency Injection. They are:

1. **Setter Dependency Injection (SDI)**: This is the simpler of the two DI methods. In this, the DI will be injected with the help of setter and/or getter methods. Now to set the DI as SDI in the bean, it is done through the bean-configuration file For this, the property to be set with the SDI is declared under the <property> tag in the bean-config file.

2. **Constructor Dependency Injection (CDI)**: In this, the DI will be injected with the help of constructors. Now to set the DI as CDI in bean, it is done through the bean-configuration file For this, the property to be set with the CDI is declared under the <constructor-arg> tag in the bean-config file.

Week 2: 10 August 2023:

Conclusion:

"In conclusion, my web development internship with Java has been an invaluable experience that has significantly enhanced my knowledge and skills in the field of web development. Throughout this internship, I have had the opportunity to work on a variety of projects, each presenting unique challenges and learning opportunities.

One of the most significant takeaways from this internship is the practical experience I gained in Java web development. I have become proficient in using Java technologies such as Servlets and JSP to build dynamic web applications. I have also gained hands-on experience with popular web development frameworks like Spring and Hibernate, which have broadened my understanding of modern web development practices.

Overall, my web development internship with Java has been a transformative experience. It has prepared me for a career in web development and provided me with a strong foundation in Java-based technologies. I am excited to continue building on this foundation and furthering my career in the ever-evolving field of web development.

In conclusion, this internship has been a significant step in my journey as a web developer, and I look forward to the exciting opportunities and challenges that lie ahead."