PRACTICAL FILE

IT-455

FEDT LAB

Submitted by:
Nitin Kumar
ENROLLMENT NO - 04116403220

University School of Information, Comm & Technology Guru Gobind Singh Indraprastha University Dwarka, New Delhi

Submitted to:

Dr. Sanjay Malik, USICT, GGSIPU

INDEX

Experiment No. 1

Aim ->Create a webpage about yourself using basic HTML tags

Theory -> Some of the basic HTML tags are <html> tag which defines a file as html file, <head> tag which contains various file definition for javascript and stylesheets along with title of file, <body> tag which defines body of the page, <h1>, <h2>, …, <h6> tag which gives various size headings and <p> tag for paragraph.

Code ->

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About Me</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        header {
            text-align: center;
            background-color: #f0f0f0;
            padding: 10px;
        }
        section {
            margin-top: 20px;
        }
        h1, h2 {
            color: #333;
        }
```

```
    </style>
</head>
<body>
    <header>
        <h1>My Portfolio</h1>
    </header>

    <section>
        <h2>Introduction</h2>
        <p>Hello, I'm Lakshay Yadav, a Software developer based in New delhi</p>
    </section>

    <section>
        <h2>Education</h2>
        <p>I graduated from IP University with a degree in Computer Science. I also pursued additional courses in Programming to enhance my skills.</p>
    </section>

    <section>
        <h2>Experience</h2>
        <ul>
         <li>I have 3 years of experience working in the field of programming.</li>
         <li>Interned as a wev developer in some software based organizations.</li>
         <li>Have some projects of my own based on the skills learnt.</li>
        </ul>
    </section>

    <section>
        <h2>Contact Info</h2>
```

```
        <p>My LinkedIn Profile: <a href="#">LinkedIn</a></p>
        <p>My GitHub Profile: <a href="#">GitHub</a></p>
        <p>You can reach me via email at email@gmail.com.</p>
    </section>

    <footer>
        <p>Thank you for visiting my webpage!</p>
    </footer>
</body>
</html>
```

Result ->

Experiment No. 2

Aim -> Write a program in html using advanced HTML tags like table etc.

Theory -> Some of the advanced HTML tags are <a> tag used to add an external link to our website, <table>, <tr>, <td> tags used to create a table and <ul>, <ol>, <li> tags to create a list and <img> tag to include an image

Code ->

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Table, List, and Image Example</title>

</head>
<body>
    <h1>Table, List, and Image Example</h1>

    <!-- Table Example -->
    <table>
        <thead>
            <tr>
                <th>Name</th>
                <th>Age</th>
                <th>Occupation</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>John Doe</td>
```

```
                <td>32</td>
                <td>Software Engineer</td>
            </tr>
            <tr>
                <td>Jane Smith</td>
                <td>25</td>
                <td>Product Manager</td>
            </tr>
        </tbody>
    </table>

    <!-- List Example -->
    <h2>Unordered List Example</h2>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ul>

    <!-- Image Example -->
    <h2>Image Example</h2>
    <img
src="https://images.unsplash.com/photo-1477959858617-67f85cf4f1df?q=80&w=1
000&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxleHBsb3JlLWZlZ
WR8MXx8fGVufDB8fHx8fA%3D%3D" alt="City" height="400" width="800">


</body>
</html>
```

Result ->

Table, List, and Image Example

Name	Age	Occupation
John Doe	32	Software Engineer
Jane Smith	25	Product Manager

Unordered List Example

- Item 1
- Item 2
- Item 3

Image Example

Experiment No. 3

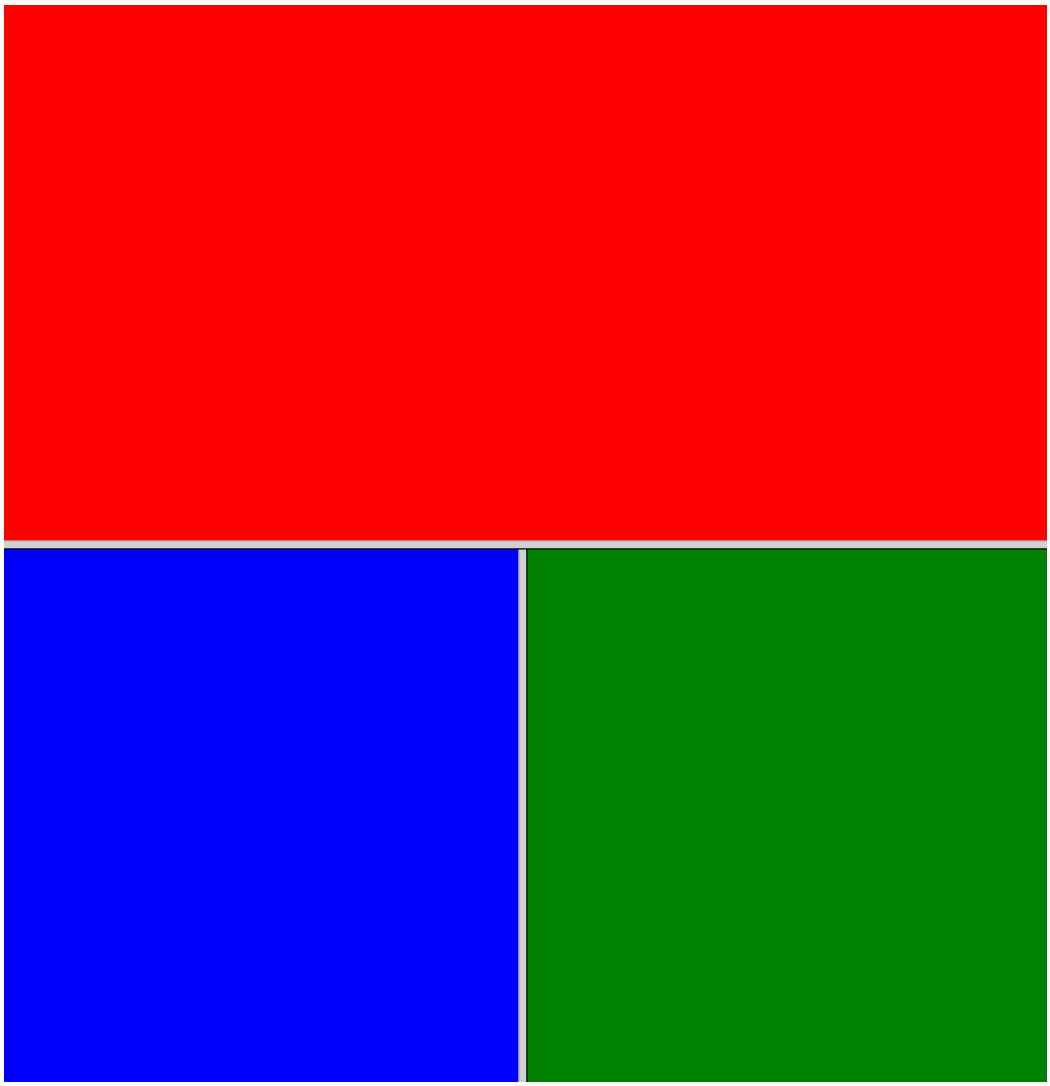
Aim -> Write an experiment for creating a frame

Theory -> The <frameset> tag is used to create a vertical split with two frames.The rowsattribute divides the vertical space into two frames (each taking 50% of the height).
Inside the first frame, a horizontal split is created using another <frameset> with the colsattribute, dividing the space into two frames (each taking 50% of the width). The srcattribute in each <frame>specifies the HTML file to be loaded into that frame.The nameattribute is used to give each frame a name, which can be targeted when creating links or forms.

Code ->

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>Frames Experiment</title>

</head>

<frameset rows="50%,50%">

    <frame src="" name="topFrame" scrolling="auto"
style="background-color: red">

    <frameset cols="50%,50%">

        <frame src="" name="leftFrame" scrolling="auto"
style="background-color: blue">

        <frame src="" name="rightFrame" scrolling="auto"
style="background-color: green">

    </frameset>

</frameset> </html>
```

Result ->



Experiment No. 4

Aim -> Write an experiment for creating forms taking input from user

Theory -> In this form:

- Text input is created using <inputtype="text">.
- Email input is created using <inputtype="email">.
- Radio buttons for gender use <inputtype="radio">.
- Checkbox for subscription uses <inputtype="checkbox">.
- A dropdown list for the country is created using <select>and <option>.
- Date input is created using <inputtype="date">.

Code ->

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Details Form</title>
</head>
<body>
    <h1>User Details Form</h1>

    <form action="#" 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>Gender:</label>
```

```
        <label for="male"><input type="radio" id="male" name="gender"
value="male"> Male</label>
        <label for="female"><input type="radio" id="female" name="gender"
value="female"> Female</label>
        <br><br>

        <label for="country">Country:</label>
        <select id="country" name="country">
            <option value="usa">India</option>
            <option value="canada">United States of America</option>
            <option value="uk">Germany</option>

        </select>
        <br><br>
        <label for="birthdate">Birthdate:</label>
        <input type="date" id="birthdate" name="birthdate">
        <br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Result ->

User Details Form

Name:

Email:

Gender: Male Female

Country: India

Birthdate: dd-mm-yyyy

Submit

Experiment No. 5

Aim -> Create a webpage using CSS styles

Theory -> CSS, or Cascading Style Sheets, is a style sheet language used for describing the presentation of a document written in HTML (or XML). In simpler terms, it's a language that controls the visual presentation of web pages. CSS allows web developers to style HTML elements, providing control over layout, colors, fonts, spacing, and more.

Code ->

```
<head>
<title>Styled Webpage</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Styled Webpage</h1>
    </header>
    <main>
        <h2>Welcome to my webpage with css design</h2>
        <p>This is a simple example of a webpage with some basic CSS styling. You can customize and enhance the styles based on your preferences. </p>
        <h2>Section 1</h2>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>

        <h2>Section 2</h2>
        <p>Curabitur non urna in velit malesuada tincidunt ac eget arcu. </p>
    </main>
```

```
    <footer>
        <p>&copy; 2023 Styled Webpage</p>
    </footer>
</body>
</html>
```

CSS->

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
    line-height: 1.6;
    color: #333;}
header {
    background-color: #333;
    color: #fff;
    padding: 10px;
    text-align: center;
}
main {
    max-width: 800px;
    margin: 20px auto;
    padding: 20px;
    background-color: #fff;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1, h2 {color: #333;}

p {margin-bottom: 20px;}
footer {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 10px;
    position: fixed;
    bottom: 0;
    width: 100%;
}
```

Result ->

Welcome to my webpage with css design

This is a simple example of a webpage with some basic CSS styling. You can customize and enhance the styles based on your preferences.

Section 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Section 2

Curabitur non urna in velit malesuada tincidunt ac eget arcu.

Experiment No. 6

Aim -> Write an experiment with JavaScript.

Theory -> JavaScript is a high-level, interpreted programming language primarily used for front-end web development. It enables the creation of dynamic and interactive web pages by allowing manipulation of the Document Object Model (DOM) and handling user events. JavaScript runs in web browsers, providing client-side scripting capabilities. It supports object-oriented, imperative, and functional programming paradigms and is an essential part of modern web development stacks.

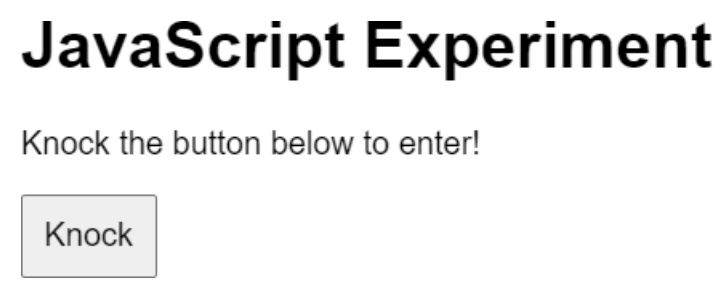
Code ->

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>JavaScript Experiment</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        button {
            padding: 10px;
            font-size: 16px;
            cursor: pointer;
        }
    </style>
</head>
```

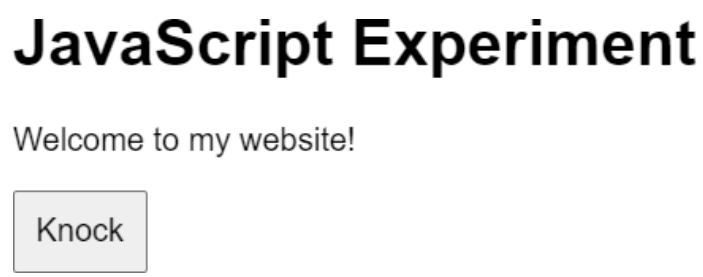
```
<body>
    <h1>JavaScript Experiment</h1>

    <p id="output">Knock the button below to enter!</p>

    <button onclick="knockDoor()">Knock</button>

    <script>
        // JavaScript function to change the text content
        function knockDoor() {
            // Get the element by its ID
            var outputElement =
document.getElementById("output");

            // Change the text content
            outputElement.textContent = "Welcome to my
website!";
        }
    </script>
</body>
</html>
```

Result ->

Before Clicking the button:



After Clicking the button:



Experiment No. 7

Aim -> Write an experiment with Dynamic HTML

Theory -> Dynamic HTML (DHTML) refers to the use of HTML, CSS, and JavaScript to create interactive and dynamic web pages. It allows web developers to change the content, structure, and style of a webpage in response to user actions or other events without requiring a full page reload.

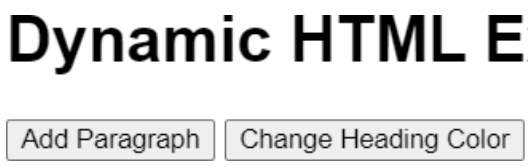
Code ->

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dynamic HTML Example</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
    </style>
</head>
<body>
    <h1 id="heading">Dynamic HTML Example</h1>

    <button onclick="addParagraph()">Add Paragraph</button>
     <button onclick="changeColor()">Change Text Color</button>

    <script>
        // JavaScript function to add a new paragraph
        function addParagraph() {
```

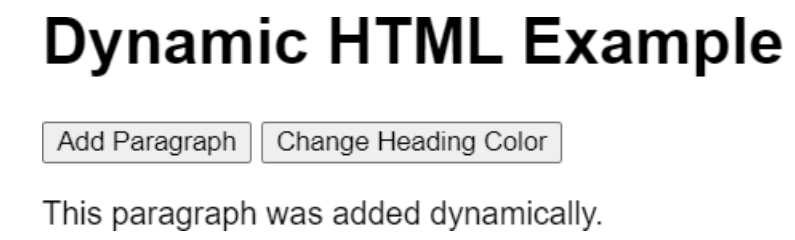
```
            // Create a new paragraph element
            var newParagraph = document.createElement("p");

            // Set the text content of the new paragraph
            newParagraph.textContent = "This paragraph was added
dynamically.";

            // Append the new paragraph to the body of the document
            document.body.appendChild(newParagraph);
        }
        function changeColor()
        {
          document.getElementById("heading").style.color = "blue";
        }
    </script>
</body>
</html>
```

Result ->

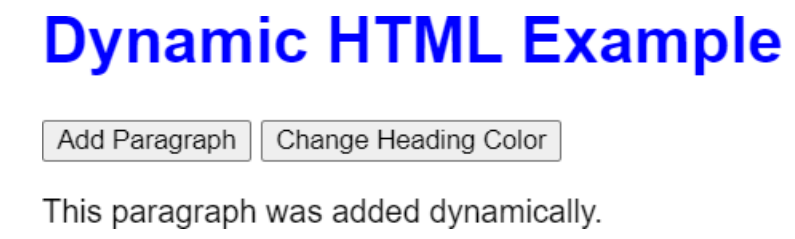
Before clicking "Add Paragraph" button:



After Clicking "Add Paragraph" Button:



After clicking “ChangeText Color” Button:



Experiment No. 8

Aim -> Develop an interactive form using ASP

Theory -> Dynamic HTML (DHTML) refers to the use of HTML, CSS, and JavaScript to create interactive and dynamic web pages. It allows web developers to change the content, structure, and style of a webpage in response to user actions or other events without requiring a full page reload.

Code ->

```
@page
@model IndexModel
@{
      ViewData["Title"] = "Home page";
}



<!DOCTYPE html>
<html lang="en">
  <head>
      <title>Add  Numbers</title>
  </head>
<body>
  <p>Enter two whole numbers and then click <strong>Add</strong>.</p>
  <form action="" method="post">
      <p><label for="text1">First Number:</label>
      <input type="text" name="text1"/>
      </p>
      <p><label for="text2">Second Number:</label>
      <input type="text" name="text2"/>
      </p>
      <p><input type="submit" value="Add" /></p>
```

```
  </form>
  <p>@ViewData["totalMessage"]</p>


</body>
</html>
```

Index.cshtml.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
namespace WebApplication1.Pages;
public class IndexModel : PageModel
{
      private readonly ILogger<IndexModel> _logger;
      public IndexModel(ILogger<IndexModel> logger)
      {
      _logger = logger;
      }
      public void OnGet()
      {
      }



      public void OnPost(int text1, int text2)
      {
      @ViewData["totalMessage"]=text1+text2;
      }
}
```

Result ->

Enter two whole numbers and then click **Add**.

First Number: 1

Second Number: 2

Add

3

Experiment No. 9

Aim -> Develop an interactive form using PHP

Theory -> PHP (Hypertext Preprocessor) is a server-side scripting language widely used for web development. It enables dynamic content generation, database connectivity, and seamless integration into HTML, making it a popular choice for building dynamic and interactive websites.

Code ->

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }
```

```
  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular expression also
allows dashes in the URL)

 if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.
;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF
"]);?>">
  Name: <input type="text" name="name" value="<?php echo $name;?>">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email" value="<?php echo $email;?>">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website" value="<?php echo $website;?>
">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
```

```
  Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment
;?></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="female") echo "checked";?> value="female">Female
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="male") echo "checked";?> value="male">Male
  <input type="radio" name="gender" <?php if (isset($gender) &&
 $gender=="other") echo "checked";?>  value="other">Other
   <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo  $name;
echo  "<br>";
echo  $email;
echo  "<br>";
echo $website;
echo  "<br>";
echo $comment;
echo  "<br>";
echo $gender;
?>

</body>
</html>
```

Result ->

PHP Form Validation Example

* required field

Name: johnny *

E-mail: johnny@gmail.com *

Website: no Invalid URL

Comment: no

Gender: ○ Female ◉ Male ○ Other *

Submit

Your Input:

johnny
johnny@gmail.com
no
no
male

Experiment No. 10

Aim -> Develop a JSP page that includes a simple form with input fields for a name and email address. Upon form submission display a 'Thank you, [Name]' message on the same page

Theory -> Java Server Pages (JSP) is a technology used in web development to create dynamic, platform-independent content. It's a Java-based technology that enables the development of web pages containing dynamic content by using Java programming language.

Code ->

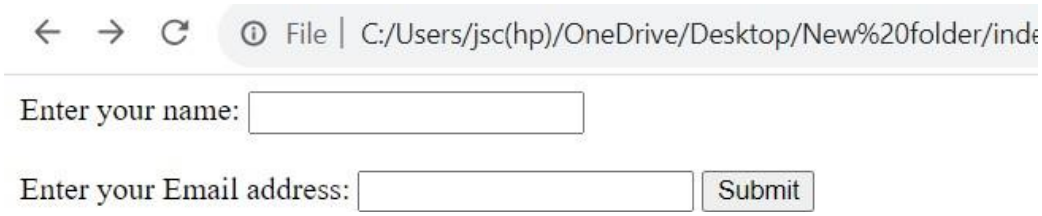
Index.html

```
<html>

<body>
<<form action="Form.jsp">
Enter your name: <input type="text" name="name">
<br>
<br>
Enter your Email address: <input type="text" name="email">
<input type="submit" value="Submit">
</form>>
</body>
</html>
```

Form.jsp

```
<html>
<body>
<%
    result = request.getParameter("name");
%>
<b>Thank you </b> <%= result %>
</body>
</html>
```

Result ->


File | C:/Users/jsc(hp)/OneDrive/Desktop/New%20folder/inde
Enter your name:
Enter your Email address:
Submit

Experiment No. 11

Aim -> Using current frontend development technologies, write an experiment

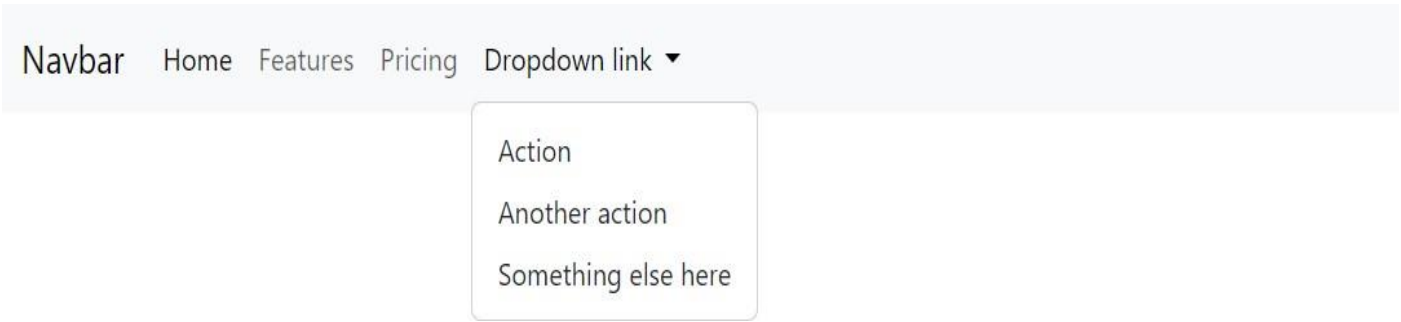
Theory -> Bootstrap is a popular open-source front-end framework that simplifies the process of designing and building responsive and mobile-friendly web pages. It was originally developed by Twitter and is now maintained by the open-source community. Bootstrap provides a collection of CSS and JavaScript components, as well as a responsive grid system, making it easy to create modern and visually appealing web applications.

Code ->

```
<nav class="navbar navbar-expand-lg bg-body-tertiary">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target="#navbarNavDropdown"
aria-controls="navbarNavDropdown" aria-expanded="false"
aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse"
id="navbarNavDropdown">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page"
href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Features</a>
        </li>
        <li class="nav-item">
```

```
          <a class="nav-link" href="#">Pricing</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#"
role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Dropdown link
          </a>
          <ul class="dropdown-menu">
            <li><a class="dropdown-item"
href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another
action</a></li>
            <li><a class="dropdown-item" href="#">Something else
here</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Result ->



Experiment No. 12

Aim -> Using current back-end development technologies, write an experiment

Theory -> The backend is the server-side of a web application. It's responsible for handling tasks that don't involve direct interaction with the user. This includes tasks such as data processing, business logic, database operations, and server-side authentication. The backend interacts with the frontend (the client-side) to provide the necessary data and functionality to the user interface.

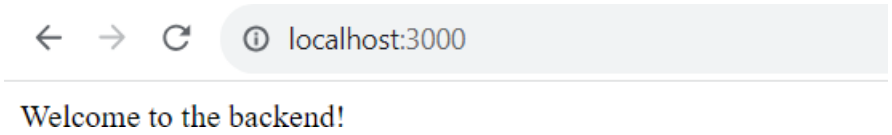
Express is a web application framework for Node.js. It simplifies the process of building robust and scalable web applications by providing a set of features and tools. Express is minimal and unopinionated, meaning it gives developers the flexibility to structure their application the way they prefer. It is widely used for creating RESTful APIs (Application Programming Interfaces) and building the backend of web applications.

Code ->

```
// Import necessary modules

const express = require('express');
const cors = require('cors');

// Create an instance of Exp

ress
const app = express();

// Use middleware to enable CORS
app.use(cors());

// Define a sample data array
const items = [
  { id: 1, name: 'Item 1' },
  { id: 2, name: 'Item 2' },
  { id: 3, name: 'Item 3' },
];

// Define a route to get the list of items
app.get('/api/items', (req, res) => {
  res.json(items);
});
```

```
// Define a default route
app.get('/', (req, res) => {
  res.send('Welcome to the backend!');
});

// Set the port for the server to listen on
const port = process.env.PORT || 3000;

// Start the server
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Result ->



Welcome to the backend!

[{"id":1,"name":"Item 1"},{"id":2,"name":"Item 2"},{"id":3,"name":"Item 3"}]

Experiment No. 13

Aim -> Create a Java program that reads and modifies an XML document containing a list of employees. Load the XML data, add a new employee, and save modified XML file

Theory -> **XML Files Introduction** XML (eXtensible Markup Language) is a versatile and human-readable markup language designed to store and transport data. It uses tags to define elements and their hierarchical relationships, making it a common choice for data exchange and configuration purposes.

Program Introduction: The Java program utilizes the Document Object Model (DOM) API to read and modify XML data. It adds a new employee to an existing list of employees in an XML file, showcasing how to manipulate XML documents programmatically. The program demonstrates key concepts such as parsing, modifying, and saving XML files, providing a foundation for handling structured data in a Java environment.

Code ->

```
import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.io.File;

import java.io.IOException;
```

```
public class ModifyXML {


    public static void main(String[] args) {
        try {
            // Load XML document
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder =
factory.newDocumentBuilder();
            Document document = builder.parse(new
File("employees.xml"));


            // Add a new employee
            addEmployee(document, "3", "Alice Johnson", "QA
Engineer");


            // Save modified XML document
            saveXML(document, "modified_employees.xml");


            System.out.println("XML modification complete. Check
'modified_employees.xml'.");
        } catch (ParserConfigurationException | IOException e) {
            e.printStackTrace();
        } catch (org.xml.sax.SAXException e) {
            e.printStackTrace();
        }
    }


    private static void addEmployee(Document document, String
id, String name, String position) {
        // Get the root element (employees)
        Element rootElement = document.getDocumentElement();
```

```
        // Create a new employee element
        Element newEmployee =
document.createElement("employee");

        // Create id, name, and position elements
        Element idElement = document.createElement("id");
        idElement.appendChild(document.createTextNode(id));

        Element nameElement = document.createElement("name");
        nameElement.appendChild(document.createTextNode(name));

        Element positionElement =
document.createElement("position");

positionElement.appendChild(document.createTextNode(position));

        // Append id, name, and position elements to the new
employee element
        newEmployee.appendChild(idElement);
        newEmployee.appendChild(nameElement);
        newEmployee.appendChild(positionElement);

        // Append the new employee element to the root element
(employees)
        rootElement.appendChild(newEmployee);
    }

    private static void saveXML(Document document, String
fileName) {
        try {
            // Use a Transformer to write the document to a file
```

```
            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer =
transformerFactory.newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT,
"yes");


            DOMSource source = new DOMSource(document);
            StreamResult result = new StreamResult(new
File(fileName));


            transformer.transform(source, result);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Result ->

employees.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<employees>
    <employee>
        <id>1</id>
        <name>John Doe</name>
        <position>Software Engineer</position>
    </employee>
    <employee>
        <id>2</id>
        <name>Jane Smith</name>
        <position>Project Manager</position>
    </employee>
</employees>
```

modified_employees.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<employees>
    <employee>
        <id>1</id>
        <name>John Doe</name>
        <position>Software Engineer</position>
    </employee>
    <employee>
        <id>2</id>
        <name>Jane Smith</name>
        <position>Project Manager</position>
    </employee>
    <employee>
        <id>3</id>
        <name>Alice Johnson</name>
        <position>QA Engineer</position>
    </employee>
</employees>
```

Experiment No. 14

Aim -> Write an experiment creating a JavaBean component

Theory -> The backend is the server-side of a web application. It's responsible for handling tasks that don't involve direct interaction with the user. This includes tasks such as data processing, business logic, database operations, and server-side authentication. The backend interacts with the frontend (the client-side) to provide the necessary data and functionality to the user interface.

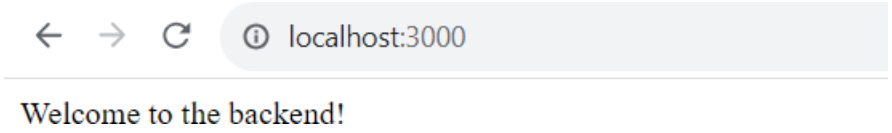
Express is a web application framework for Node.js. It simplifies the process of building robust and scalable web applications by providing a set of features and tools. Express is minimal and unopinionated, meaning it gives developers the flexibility to structure their application the way they prefer. It is widely used for creating RESTful APIs (Application Programming Interfaces) and building the backend of web applications.

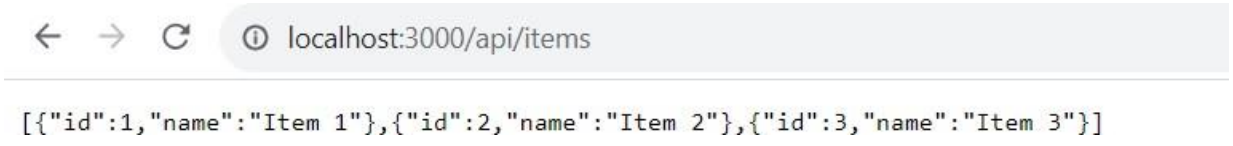
Code ->

```
// Import necessary modules
const express = require('express');
const cors = require('cors');

// Create an instance of Express
const app = express();

// Use middleware to enable CORS
app.use(cors());

// Define a sample data array
const items = [
  { id: 1, name: 'Item 1' },
  { id: 2, name: 'Item 2' },
  { id: 3, name: 'Item 3' },
];

// Define a route to get the list of items
app.get('/api/items', (req, res) => {
  res.json(items);
});
```

```
// Define a default route
app.get('/', (req, res) => {
  res.send('Welcome to the backend!');
});

// Set the port for the server to listen on
const port = process.env.PORT || 3000;

// Start the server
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Result ->





Experiment No. 15

Aim -> Develop a Java servlet that is a simple online poll system

Theory -> The backend is the server-side of a web application. It's responsible for handling tasks that don't involve direct interaction with the user. This includes tasks such as data processing, business logic, database operations, and server-side authentication. The backend interacts with the frontend (the client-side) to provide the necessary data and functionality to the user interface.

Express is a web application framework for Node.js. It simplifies the process of building robust and scalable web applications by providing a set of features and tools. Express is minimal and unopinionated, meaning it gives developers the flexibility to structure their application the way they prefer. It is widely used for creating RESTful APIs (Application Programming Interfaces) and building the backend of web applications.

Code ->

HTML

```
// Import necessary modules

<!DOCTYPE html>

<html>

<head>

    <title>Simple Poll System</title>

</head>

<body>

    <h2>What is your favorite programming language?</h2>

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

        <input type="radio" name="language" value="Java">
Java<br>

        <input type="radio" name="language" value="Python">
Python<br>

        <input type="radio" name="language" value="JavaScript">
JavaScript<br>
```

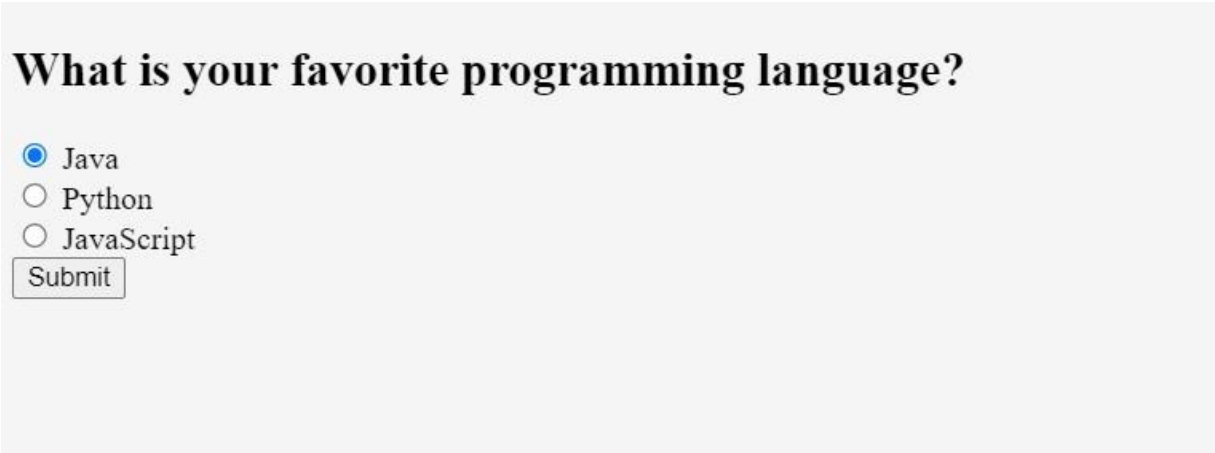
```
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.HashMap;
import java.util.Map;

public class VoteServlet extends HttpServlet {

    private Map<String, Integer> voteCount;

    public void init() throws ServletException {
        super.init();
        voteCount = new HashMap<>();
        voteCount.put("Java", 0);
        voteCount.put("Python", 0);
        voteCount.put("JavaScript", 0);
    }

    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        String languageVoted = request.getParameter("language");
```

```
        if (voteCount.containsKey(languageVoted)) {
            int count = voteCount.get(languageVoted);
            voteCount.put(languageVoted, count + 1);
        }

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head><title>Thank
You!</title></head><body>");
        out.println("<h2>Thank you for voting!</h2>");
        out.println("<p>Your vote for " + languageVoted + " has
been recorded.</p>");
        out.println("<a href=\"index.html\">Back to poll</a>");
        out.println("</body></html>");
    }

    public void destroy() {
        // You might want to save the vote counts to a database
or file before destroying the servlet
        super.destroy();
    }
}
```

Result ->


What is your favorite programming language?
Java
Python
JavaScript
Submit

Experiment No. 16

Aim -> Write an experiment using JSP Programming and applications.

Theory -> For the following JSP application, we are making a Fibonacci sequence, user is asked to Input a number, On clicking Submit button the Fibonacci sequence is displayed on the screen.

Code ->

Index.html

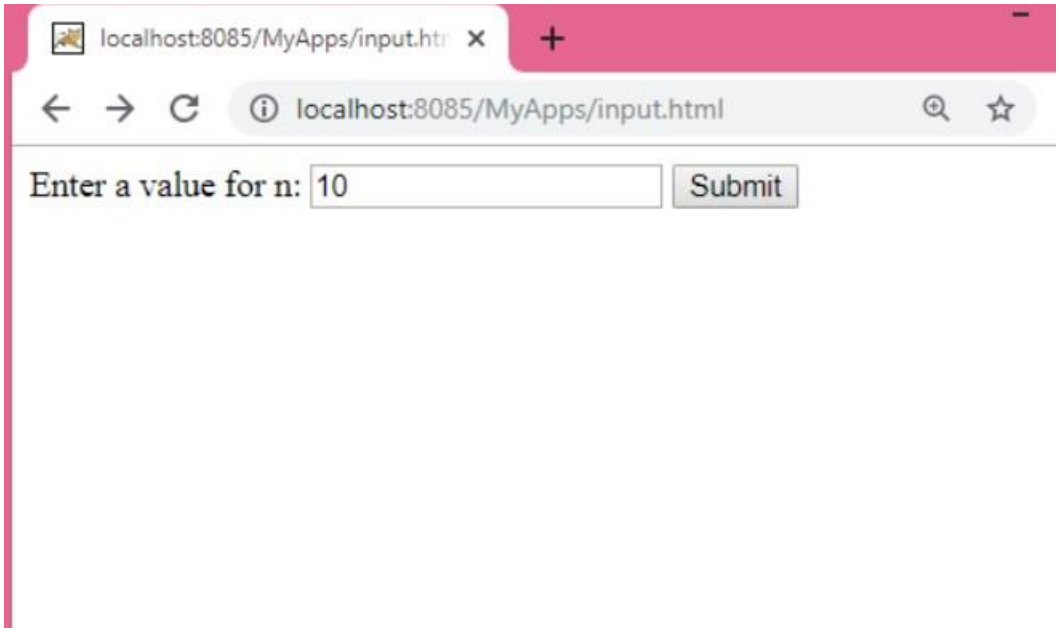
```
<html>
<body>
<form action="Fibonacci.jsp">
  Enter a value for n: <input type="text" name="val">
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

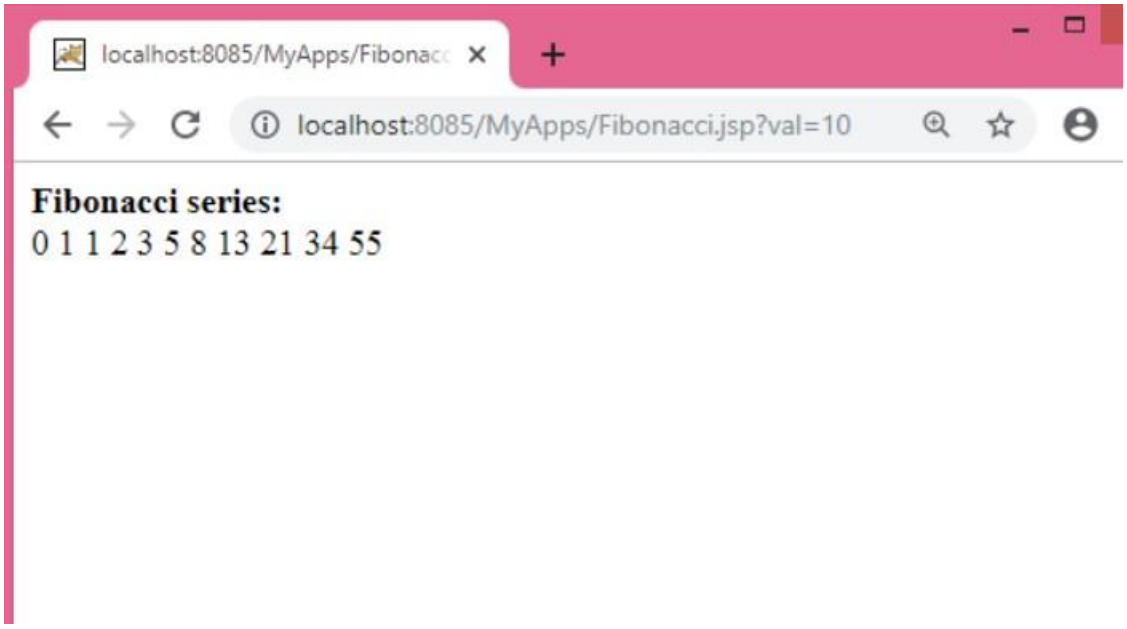
Fibonacci.jsp

```
<html>
<body>
<%!
  int n;
  String str;

  int fibo(int n) {
    if(n<2)
      return n;
    else
      return fibo(n-1) + fibo(n-2);
  }
%>
<b>Fibonacci series: </b><br>
<%
  str = request.getParameter("val");
  n = Integer.parseInt(str);

  for(int i=0; i<=n; i++) {
    out.print(fibo(i) + " ");
  }
```

```
%>
</body>
</html>
```

Result ->





Experiment No. 17

Aim -> Keeping program 10 as reference write an experiment using JDBC

Theory -> For the following JSP application, we are making a Fibonacci sequence, user is asked to Input a number, On clicking Submit button the Fibonacci sequence is displayed on the screen.

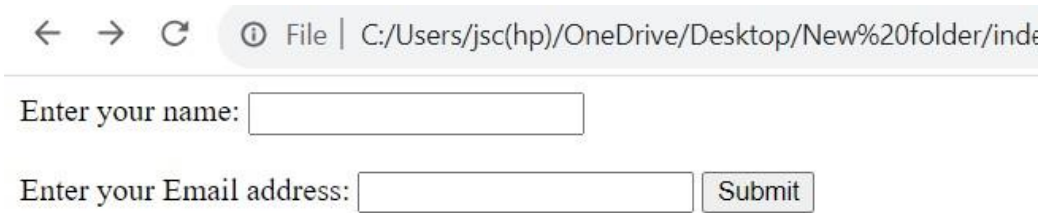
Code

```
<%@ page import="java.sql.*" %>
<%@ page import="javax.sql.DataSource" %>
<%@ page import="javax.naming.InitialContext" %>
<%@ page import="javax.naming.NamingException" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Simple Form with JDBC</title>
</head>
<body>
    <h2>Submit your name and email</h2>
    <form method="post">
        Name: <input type="text" name="name"><br>
        Email: <input type="email" name="email"><br>
        <input type="submit" value="Submit">
    </form>

    <%
        if (request.getMethod().equals("POST")) {
```

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

            if (name != null && email != null && !name.isEmpty()
&& !email.isEmpty()) {
                Connection conn = null;
                PreparedStatement pstmt = null;
                try {
                    InitialContext ctx = new InitialContext();
                    DataSource ds = (DataSource)
ctx.lookup("java:comp/env/jdbc/yourDataSource"); // Replace
'yourDataSource' with your actual data source name

                    conn = ds.getConnection();
                    String sql = "INSERT INTO UserTable (name,
email) VALUES (?, ?)";
                    pstmt = conn.prepareStatement(sql);
                    pstmt.setString(1, name);
                    pstmt.setString(2, email);
                    pstmt.executeUpdate();

                    // Display thank you message
                    out.println("<h3>Thank you, " + name +
"!</h3>");
                } catch (NamingException | SQLException e) {
                    e.printStackTrace();
                    out.println("Error: " + e.getMessage());
                } finally {
                    try {
                        if (pstmt != null) pstmt.close();
```

```
                        if (conn != null) conn.close();
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }
                }
            } else {
                out.println("<p>Please fill in both name and
email.</p>");
            }
        }
    %>
</body>
</html>
```

Result ->



Experiment No. 18

Aim -> Case study using any above program.

Theory -> For the following JSP application, we are making a Fibonacci sequence, user is asked to Input a number, On clicking Submit button the Fibonacci sequence is displayed on the screen.

Code

Index.html

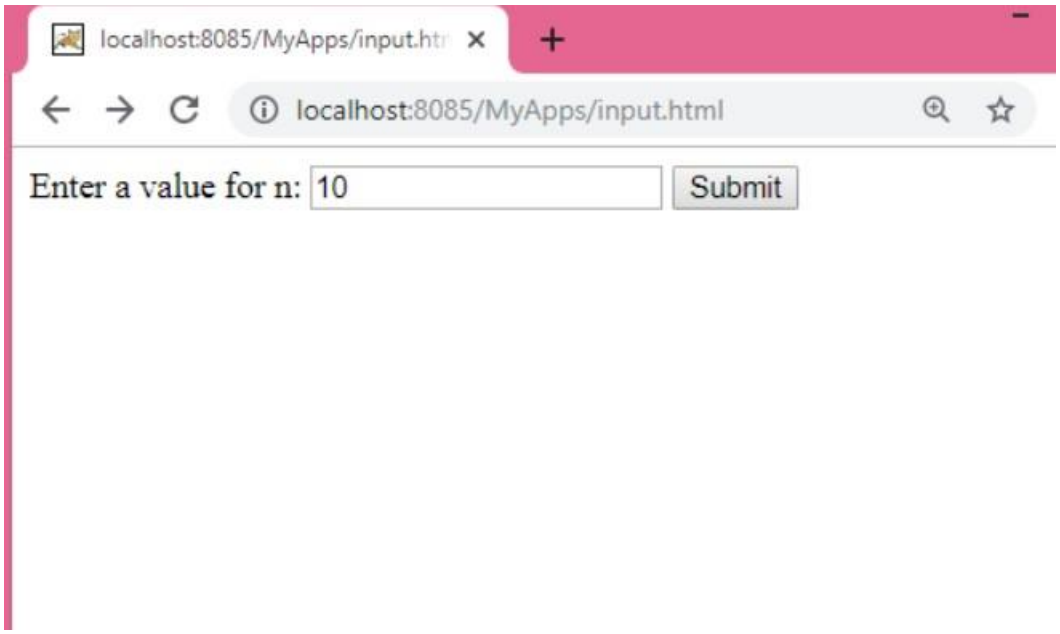
```
<html>
<body>
<form action="Fibonacci.jsp">
  Enter a value for n: <input type="text" name="val">
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

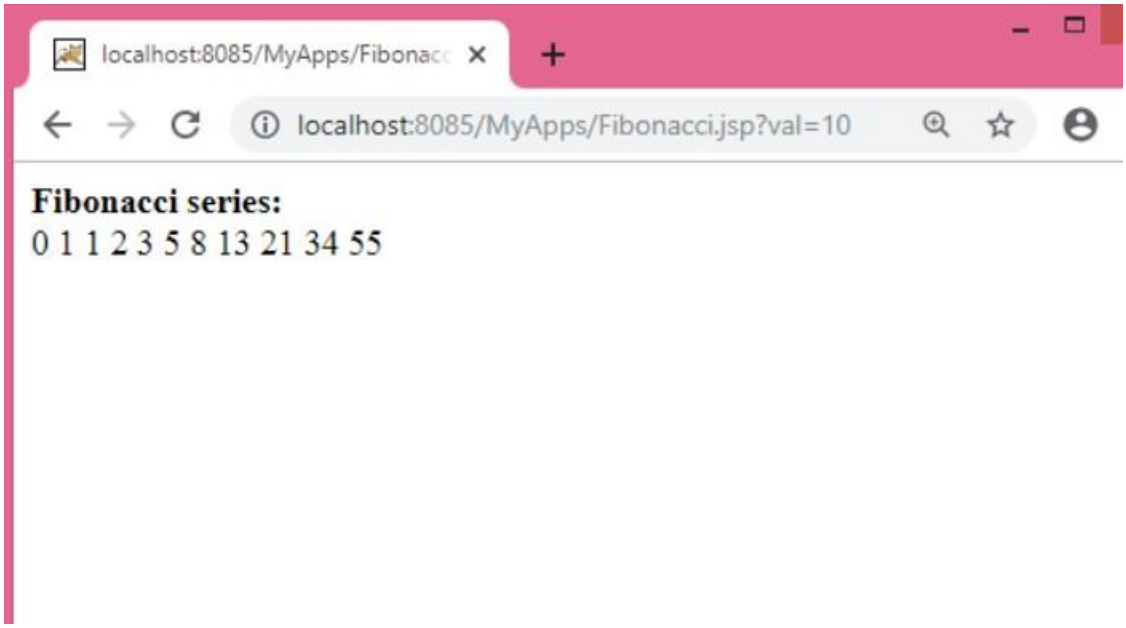
Fibonacci.jsp

```
<html>
<body>
<%!
  int n;
  String str;

  int fibo(int n) {
    if(n<2)
      return n;
    else
      return fibo(n-1) + fibo(n-2);
  }
%>
<b>Fibonacci series: </b><br>
<%
  str = request.getParameter("val");
  n = Integer.parseInt(str);

  for(int i=0; i<=n; i++) {
    out.print(fibo(i) + " ");
```

```
  }
%>
</body>
</html>
```

Result ->




localhost:8085/MyApps/Fibonac
localhost:8085/MyApps/Fibonacci.jsp?val=10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34 55

Experiment No. 19

Aim -> Write an experiment displaying database programming.

Theory -> SQL, which stands for Structured Query Language, is a standardized programming language used for managing and manipulating relational databases. It provides a set of commands to interact with databases, enabling tasks such as data retrieval, data definition, data manipulation, and data control.

I have created a table Employees, inserted some dummy data and also updated the data

Code

```
import java.sql.*;


public class DatabaseExperiment {


    static final String JDBC_DRIVER =
"com.mysql.cj.jdbc.Driver";

    static final String DB_URL =
"jdbc:mysql://localhost/your_database"; // Replace
'your_database' with your actual database name


    static final String USER = "your_username"; // Replace
'your_username' and 'your_password' with your database
credentials

    static final String PASS = "your_password";


    public static void main(String[] args) {

        Connection conn = null;

        Statement stmt = null;


        try {
```

```
            Class.forName(JDBC_DRIVER);
            conn = DriverManager.getConnection(DB_URL, USER,
PASS);
            stmt = conn.createStatement();

            // Create a table
            String createTableSql = "CREATE TABLE IF NOT EXISTS
students (id INT NOT NULL AUTO_INCREMENT, name VARCHAR(50), age
INT, PRIMARY KEY (id))";
            stmt.executeUpdate(createTableSql);

            // Insert records
            String insertRecordSql1 = "INSERT INTO students
(name, age) VALUES ('Alice', 20)";
            String insertRecordSql2 = "INSERT INTO students
(name, age) VALUES ('Bob', 22)";
            stmt.executeUpdate(insertRecordSql1);
            stmt.executeUpdate(insertRecordSql2);

            // Retrieve records
            String retrieveRecordsSql = "SELECT id, name, age
FROM students";
            ResultSet rs =
stmt.executeQuery(retrieveRecordsSql);

            // Display records
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                int age = rs.getInt("age");
```

```
                System.out.println("ID: " + id + ", Name: " +
name + ", Age: " + age);
            }

            rs.close();
            stmt.close();
            conn.close();
        } catch (SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        } finally {
            try {
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Command for data display:

```
SELECT * FROM Employee;
```

Experiment No. 20

Aim -> Develop College website

Result ->

```
<!DOCTYPE html>
<html>
<head>
<title>Indraprastha University</title>
</head>
<body bgcolor=silver>
<header>
<center>
<img src =

"https://upload.wikimedia.org/wikipedia/en/thumb/b/b8/GGSIU_logo.svg/
1200px-GGSIU_logo.svg.png" width =100>
<h1>Indraprastha University</h1>
</center>
<nav>
<ul>
<li><a href="index.html">Home</a></li>
<li><a href="about.html">About</a></li>
<li><a href="programs.html">Programs</a></li>
<li><a href="admissions.html">Admissions</a></li>
<li><a href="contact.html">Contact</a></li>
</ul>
</nav>
</header>
<main>
<section id="welcome">
<h2>Welcome to the Indraprastha University</h2>
```

Guru Gobind Singh Indraprastha University (GGSIPU) is
the first University established in 1998 by Govt. of NCT of Delhi
under the provisions of Guru Gobind Singh Indraprastha University
Act, 1998 read with its Amendment in 1999 The University is
recognized by University Grants Commission (UGC), India under section
12B of UGC Act.
</section>

<img src =

"https://i.ndtvimg.com/i/2016-08/indraprastha-university_650x400_7147
1147455.jpg" width=60%>

<section id="about">
<h2>What's new</h2>
</section>

<table width=100% bgcolor=grey border=1>
<tr>
<td>IPU Health Mela</td>
<td>Jobs and Opportunities</td>
</tr>
<tr>
<td>NAAC</td>
<td>Notices and Circulars</td>
</tr>
</table>

© 2023 Indraprastha University. All rights
reserved.
</main>

```
</body>

</html>

}
```

Result: