Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./~~Miss~~ AYUSH SANJAY SHARMA

of I.T Department, Semester V with

Roll No. 117 has completed a course of the necessary

experiments in the subject INTERNET PROGRAMMING LAB under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2025 - 2026

Teacher In- Charge

08/10/2025

Head of the Department

Date 08/10/2025

Principal

THADOMAL SHAHANI TSEC ENGINEERING SOLLEGE

CONTENTS

ASSIGNMENT- 1

(HTML)

AIM: To create an HTML webpage using all its elements.

THEORY:

HTML (HyperText Markup Language) is the standard language used to create web pages. It is a markup language, not a programming language, and it structures content using "tags". Each tag tells the browser how to display different types of content like text, links, images, or input forms.

Basic Structure of an HTML Page:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    Page content goes here...
  </body>
</html>
```

Common HTML Elements:

1. Headings and Text:

* <h1> to <h6> : Headings
* <p> : Paragraph
*
 : Line break
* <hr> : Horizontal line
* or <b> : Bold text
* or <i> : Italic text

2. Links and Images:
* <a href="url"> : Anchor tag for hyperlinks
* <img src="image.jpg" alt="description"> : To display an image

3. Lists:
* : Unordered (bulleted) list
* : Ordered (numbered) list
* : List item (used inside ul or ol)

4. Tables:
Used to display structured tabular data.

Tags used in tables:
* <table> : The table container
* <tr> : Table row
* <th> : Table header cell
* <td> : Table data cell

5. Forms:
Used to take input from the user.

Form elements include:

* <form> : The form container
* <input> : For text fields, emails, passwords, and buttons
* <textarea> : For multi-line text input
* <label> : Describes each input

Attributes:

Tags can have attributes that define additional behavior or information.

Examples:

* href="url" in <a> for the destination link
* src="image.jpg" in <img> for the image path
* type="text" or type="submit" in <input>
* name="fieldname" to identify form data
* alt="text" in <img> for alternate text

Semantic Elements:

Used for better structure and readability:

* <header>, <footer>
* <main>, <section>, <article>
* <nav>, <aside>

CODE:

- **Resume**

```
<!DOCTYPE html>
```

```
<head>
   <title>Aayush's Resume</title>
</head>
<body>
<table border="1"  width="100%">
   <tr>
      <!-- Left Column -->
      <td width="30%" valign="top" align="center">
         <img src="profile_pic.jpeg" alt="Profile Picture" width="150"
height="150"><br>
         <h1>Aayush Sharma</h1>
         <p>Email: abc@gmail.com</p>
         <p>Phone: +91 9876543210</p>
         <p>Address: Thadomal Shahani Engineering College,<br>Bandra West, Mumbai,
Maharashtra 400050</p>
      </td>

      <!-- Right Column -->
      <td width="70%" valign="top">
         <h2>Summary</h2>
         <p>Brief Summary</p>

         <h2>Education</h2>
         <ul>
            <li>
               <h3><a href="ssc_marksheet.html">2021 - SSC</a></h3>
               <ul>
                  <li>Percentage: 89.80</li>
                  <li>School Name</li>
```

```
            </ul>
        </li>
        <li>
            <h3><a href="marksheet2.html">2023 - HSC</a></h3>
            <ul>
                <li>Percentage: 88.50</li>
                <li>Jr. College Name</li>
            </ul>
        </li>
        <li>
            <h3><a href="marksheet3.html">2023–2027 - B.E Information
Technology</a></h3>
            <ul>
                <li>CGPA (till date): 8.1</li>
                <li>Mumbai University</li>
            </ul>
        </li>
    </ul>

    <h2>Skills</h2>
    <ol>
        <li>Skill 1</li>
        <li>Skill 2</li>
        <li>Skill 3</li>
        <li>Skill 4</li>
    </ol>

    <h2>Certifications</h2>
    <ul>
```

```
            <li>Certifications</li>
          </ul>
        </td>
    </tr>
</table>

</body>
</html>
```

- **Marksheet**

```
<!DOCTYPE html>
<html>
<head>
  <title>Marksheet</title>
</head>
<body>

<table>
  <tr>
    <td><img src="logo.jpeg" alt="Logo"></td>
    <td>
      <h2>University Of Mumbai</h2>
      <h3>Marksheet</h3>
    </td>
  </tr>
</table>

<p><strong>Name:</strong> Ayush Sanjay Sharma</p>
```

Seat Number: 241107
Examination: SECOND YEAR I.T. ENGINEERING
SEMESTER-IV R 2019 'C' SCHEME

<table border="1">
  <tr>
    <th>Subject Code</th>
    <th>Subject Name</th>
    <th>Max Marks</th>
    <th>Marks Obtained</th>
  </tr>
  <tr>
    <td>ITC401</td>
    <td>Engineering Mathematics-IV</td>
    <td>100</td>
    <td>82</td>
  </tr>
  <tr>
    <td>ITC402</td>
    <td>Computer Network and Network Design</td>
    <td>100</td>
    <td>76</td>
  </tr>
  <tr>
    <td>ITC403</td>
    <td>Operating System</td>
    <td>100</td>
    <td>89</td>
  </tr>

```
  <tr>
    <td>ITC404</td>
    <td>Automata Theory</td>
    <td>100</td>
    <td>91</td>
  </tr>
  <tr>
    <td>ITC405</td>
    <td>Computer Organization and Architecture</td>
    <td>100</td>
    <td>95</td>
  </tr>
</table>

<p><strong>Total Marks:</strong> 433 / 500</p>
<p><strong>Result:</strong> Pass</p>
<p><strong>Date:</strong> 23/07/2025</p>
<a href="html_resume.html">Back to Resume</a>
</body>
</html>
```

- **Form**

```
<!DOCTYPE html>
<html>
<head>
  <title>Get in Touch</title>
</head>
<body>
```

```
<h2>Get in Touch</h2>

<form action="/submit" method="post">
  <p>
    <label>Name:</label><br>
    <input type="text" name="name">
  </p>
  <p>
    <label>Email:</label><br>
    <input type="email" name="email">
  </p>
  <p>
    <label>Message:</label><br>
    <textarea name="message" rows="5" cols="30"></textarea>
  </p>
  <p>
    <input type="submit" value="Send">
  </p>
</form>
</body>
</html>
```

OUTPUT:

Aayush Sharma

Email: abc@gmail.com

Phone: +91 9876543210

Address: Thadomal Shahani Engineering College, Bandra West, Mumbai, Maharashtra 400050

Summary

Brief Summary

Education

- **2021 - SSC**
  - Percentage: 89.80
  - School Name
- **2023 - HSC**
  - Percentage: 88.50
  - Jr. College Name
- **2023–2027 - B.E Information Technology**
  - CGPA (till date): 8.1
  - Mumbai University

Skills

1. Skill 1
2. Skill 2
3. Skill 3
4. Skill 4

Certifications

- Certifications

University Of Mumbai

Marksheet

Name: Ayush Sanjay Sharma

Seat Number: 241107

Examination: SECOND YEAR I.T. ENGINEERING
SEMESTER-IV R 2019 'C' SCHEME

Subject Code	Subject Name	Max Marks	Marks Obtained
ITC401	Engineering Mathematics-IV	100	82
ITC402	Computer Network and Network Design	100	76
ITC403	Operating System	100	89
ITC404	Automata Theory	100	91
ITC405	Computer Organization and Architecture	100	95

Total Marks: 433 / 500

Result: Pass

Date: 23/07/2025

Back to Resume

CONCLUSION: This experiment successfully demonstrates the creation of HTML webpages using its various elements.

LO MAPPING: LO1

ASSIGNMENT- 2

(CSS)

AIM: To apply CSS styles to the previously created HTML webpages.

THEORY: CSS (Cascading Style Sheets) is the language used to style and format HTML content. While HTML structures the webpage, CSS defines how it looks- colors, fonts, spacing, layout, and responsiveness. CSS works by selecting HTML elements and applying styles to them. It can be added in three ways:

1. Inline: Directly inside an HTML tag
2. Internal: Inside a tag in the HTML
3. External: In a separate .css file linked to HTML using

CSS Syntax:

```
selector {
property: value;
}
```

Example:

```
p {
color: blue;
font-size: 16px;
}
```

Selectors:

- Element selector: targets all instances of an element
  Example: h1 { color: red; }

- Class selector: uses .classname to target elements with a class
  Example: .box { border: 1px solid black; }
- ID selector: uses #idname to target an element with an ID
  Example: #header { background-color: grey; }

Common Properties:

1. Text and Fonts:

- color
- font-size
- font-family
- text-align
- font-weight
- line-height

2. Box Model:
   Every HTML element is treated as a box consisting of

- content
- padding: space inside the box
- border: the outline of the box
- margin: space outside the box

Example:

```
div {
padding: 10px;
border: 1px solid black;
margin: 20px;
}
```

3. Background:

- background-color
- background-image
- background-repeat
- background-size

4. Display and Layout:

- display: block | inline | inline-block | none | flex | grid
- position: static | relative | absolute | fixed | sticky
- top, bottom, left, right (used with position)
- float: left | right
- clear: both
- z-index: stacking order

5. Width and Height:

- width
- height
- max-width
- min-height

6. Flexbox (for layout):

Example:

```
.container {
display: flex;
justify-content: space-between;
align-items: center;
}
```

7. Pseudo-classes:

   Special states or effects

   Example:

   a:hover {

   text-decoration: underline;

   }

8. Media Queries:

   Used to make pages responsive

   Example:

   @media (max-width: 600px) {

   body {

   background-color: lightgray;

   }

   }

Cascading Order:

When multiple styles apply, CSS resolves conflicts by:

- Importance (inline > internal > external)
- Specificity (ID > class > element)
- Order of appearance (last rule wins)

CODE:

- **Resume**

```
<!DOCTYPE html>
<head>
   <title>Aayush's Resume</title>
```

```
<style>
    body {
        font-family: Arial, sans-serif;
        margin: 0;
        background-color: #f4f4f4;
    }

    .container {
        display: flex;
        max-width: 1000px;
        margin: 30px auto;
        background-color: #ffffff;

        padding: 20px;
    }

    .left-side {
        border-radius: 2%;
        width: 30%;
        padding: 20px;
        background-color: #f0f0f0;
        text-align: center;
    }

    .left-side img {
        border-radius: 50%;
        margin-bottom: 15px;
    }
```

```
.left-side h1 {
    font-size: 22px;
    margin-bottom: 10px;
}

.left-side p {
    font-size: 14px;
    color: #333;
    margin: 5px 0;
}

.right-side {
    width: 70%;
    padding: 20px;
}

.right-side h2 {
    color: #333;

    border-bottom: 2px solid #ddd; /*horizontal line*/
    padding-bottom: 15px;
    margin-top: 15px;
}

.right-side h3 {
    margin: 10px 0 5px;
}

.right-side ol {
```

```
            padding-left: 20px;
        }

        .right-side ol li {
            margin-bottom: 5px;
        }

        a {
            text-decoration: none;
            color: #2c3e50;
        }

        a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>

<div class="container">
    <!-- Left Side -->
    <div class="left-side">
        <img src="profile_pic.jpeg" alt="Profile Picture" width="150" height="150">
        <h1>Aayush Sharma</h1>
        <p>Email: abc@gmail.com</p>
        <p>Phone: +91 9876543210</p>
        <p>Address: Thadomal Shahani Engineering College, Bandra West, Mumbai,
Maharashtra 400050</p>
    </div>
```

```
  <!-- Right Side -->
  <div class="right-side">
    <h2>Summary</h2>
    <p>Brief Summary</p>

    <h2>Education</h2>
    <ul>
      <li>
        <h3><a href="ssc_marksheet.html">2021 - SSC</a></h3>
        <ul>
          <li>Percentage: 89.80</li>
          <li>School Name</li>
        </ul>
      </li>
      <li>
        <h3><a href="marksheet2.html">2023 - HSC</a></h3>
        <ul>
          <li>Percentage: 88.50</li>
          <li>Jr. College Name</li>
        </ul>
      </li>
      <li>
        <h3><a href="marksheet3.html">2023–2027 - B.E Information
Technology</a></h3>
        <ul>
          <li>CGPA (till date): 8.1</li>
          <li>Mumbai University</li>
        </ul>
```

```
            </li>
        </ul>

        <h2>Skills</h2>
        <ol>
            <li>Skill 1</li>
            <li>Skill 2</li>
            <li>Skill 3</li>
            <li>Skill 4</li>
        </ol>

        <h2>Certifications</h2>
        <ul>
            <li>Certifications</li>
        </ul>
    </div>
</div>

</body>
</html>
```

- **Marksheet**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Marksheet</title>
  <style>
```

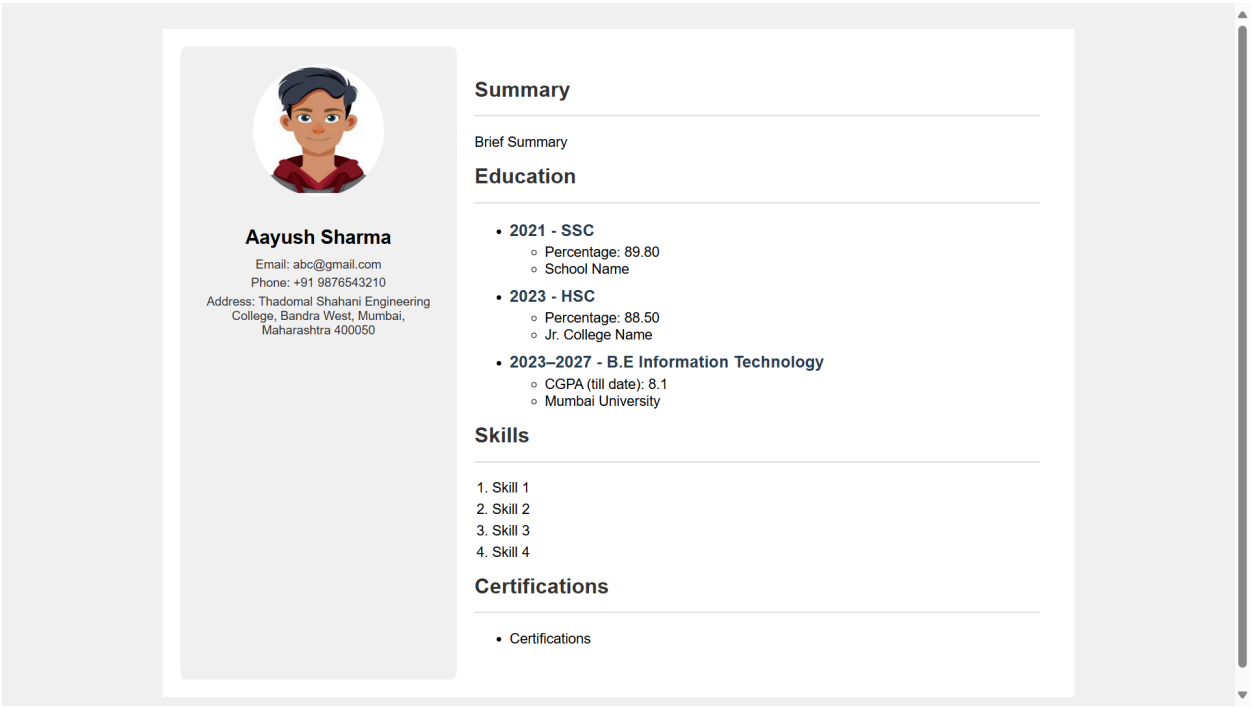
```
    body {
      font-family: Arial, sans-serif;
    }
    .marksheet {
      max-width: 600px;
      margin: auto;
      border:solid ;
      padding: 20px;
    }
    h2, h3 {
      text-align: center;
      font-family: 'Old English Text MT', serif;
    }
    table {
      width: 100%;
      margin-top: 20px;
    }
    th, td {
      border:1px solid ;
      padding: 5px;
      text-align: left;
    }
    .footer {
      margin-top: 20px;
      text-align: right;
    }
  </style>
</head>
<body>
```

```
  <div class="marksheet">
<table >
    <tr>
    <td><img src="logo.jpeg"></td>
    <td><h2>University Of Mumbai</h2>
    <h3>Marksheet</h3></td>
    </tr>
</table>

    <table>
      <tr>
        <td><strong>Name:</strong></td>
        <td>Ayush Sanjay Sharma</td>
      </tr>
      <tr>
        <td><strong>Seat Number:</strong></td>
        <td>241107</td>
      </tr>
      <tr>
        <td><strong>Examination:</strong></td>
        <td>SECOND YEAR I.T. ENGINEERING<br>SEMESTER-IV R 2019 'C'
SCHEME</td>
      </tr>
    </table>

    <table>
      <tr>
        <th>Subject Code</th>
```

```
  <th>Subject Name</th>
  <th>Max Marks</th>
  <th>Marks Obtained</th>
</tr>
<tr>
  <td>ITC401</td>
  <td>Engineering Mathematics-IV</td>
  <td>100</td>
  <td>82</td>
</tr>
<tr>
  <td>ITC402</td>
  <td>Computer Network and Network Design</td>
  <td>100</td>
  <td>76</td>
</tr>
<tr>
  <td>ITC403</td>
  <td>Operating System</td>
  <td>100</td>
  <td>89</td>
</tr>
<tr>
  <td>ITC404</td>
  <td>Automata Theory</td>
  <td>100</td>
  <td>91</td>
</tr>
<tr>
```

```
      <td>ITC405</td>
      <td>Computer Organization and Architecture</td>
      <td>100</td>
      <td>95</td>
    </tr>
  </table>

  <div class="footer">
    <p><strong>Total Marks:</strong> 433 / 500</p>
    <p><strong>Result:</strong> Pass</p>
    <p><strong>Date:</strong> 23/07/2025</p>
  </div>
 </div>

</body>
</html>
```

OUTPUT:



Aayush Sharma

Email: abc@gmail.com
Phone: +91 9876543210
Address: Thadomal Shahani Engineering College, Bandra West, Mumbai, Maharashtra 400050

Summary

Brief Summary

Education

- **2021 - SSC**
  - Percentage: 89.80
  - School Name
- **2023 - HSC**
  - Percentage: 88.50
  - Jr. College Name
- **2023–2027 - B.E Information Technology**
  - CGPA (till date): 8.1
  - Mumbai University

Skills

1. Skill 1
2. Skill 2
3. Skill 3
4. Skill 4

Certifications

- Certifications

University Of Mumbai

Marksheet

Name:	Ayush Sanjay Sharma
Seat Number:	241107
Examination:	SECOND YEAR I.T. ENGINEERING SEMESTER-IV R 2019 'C' SCHEME

Subject Code	Subject Name	Max Marks	Marks Obtained
ITC401	Engineering Mathematics-IV	100	82
ITC402	Computer Network and Network Design	100	76
ITC403	Operating System	100	89
ITC404	Automata Theory	100	91
ITC405	Computer Organization and Architecture	100	95

Total Marks: 433 / 500

Result: Pass

Date: 23/07/2025

CONCLUSION: This experiment successfully demonstrates the beautification of the previously created HTML webpages using CSS.

LO MAPPING: LO2

ASSIGNMENT- 3

(Typography, Color Modes, Transitions & Animations)

AIM: To demonstrate typography, various color modes, transitions and animations using HTML & CSS.

THEORY:

- **Typography:** Typography refers to the style, appearance, and layout of text on a webpage. It includes properties like font family (typeface), font size (text size), font weight (boldness), font style (italic/normal), text alignment (left, right, center, justify), letter spacing (space between characters), line height (vertical space between lines), and text transformation (uppercase, lowercase, capitalize). Typography enhances readability and visual hierarchy.

- **Color Modes:** Color modes define how colors are represented in CSS. These modes offer flexibility in defining solid and transparent colors for various UI elements. Common formats include:
    1. Named colors (predefined color names)
    2. Hexadecimal codes (using # followed by digits/letters)
    3. RGB (Red, Green, Blue values)
    4. RGBA (RGB with alpha/transparency value)
    5. HSL (Hue, Saturation, Lightness)
    6. HSLA (HSL with alpha/transparency value)

- **Transitions:** Transitions allow smooth changes from one property value to another over a defined duration. They are triggered by events like hover, focus, or click. Only animatable properties like background, color, width, height, transform, and opacity can be transitioned. Transition properties include the target CSS property, duration, timing function (ease, linear, ease-in, ease-out), and delay.

- **Animations:** Animations enable the gradual change of CSS properties over time using keyframes. Unlike transitions, animations define multiple stages of changes. Animation properties include the animation name (linked to keyframes), duration, timing function, iteration count (how many times it runs), direction (normal, reverse), and delay. Animations are more versatile and allow looping, reversing, or complex sequences.

CODE:

- ## Typography

```
<!DOCTYPE html>
<head>
  <title>Typography Expt</title>

<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Bitcount+Prop+Double:wght@100..900
&display=swap" rel="stylesheet">

<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
```

```
<link
href="https://fonts.googleapis.com/css2?family=Playwrite+AU+QLD:wght@100..400&display=swap" rel="stylesheet">

  <style>
    .font1 {
      font-family: 'Bitcount Prop Double';
    }
    .font2 {
      font-family: 'Playwrite AU QLD'    ;
    }
  </style>
</head>
<body>

  <p class="font1">This is Bitcount Prop Double font</p>
  <p class="font2">This is Playwrite Australia QLD font</p>

</body>
</html>
```

- **Color Modes**

```
<!DOCTYPE html>

<head>

  <title>Color Modes</title>
  <style>
```

```
    .shape {
      width: 200px;
      height: 100px;


      color: white;
      text-align: center;
      line-height: 100px; /* vertically centers text */


    }
    .rgb {
      background-color: rgb(255, 0, 0);
    }
    .hex {
      background-color: #008000;
    }
    .name {
      background-color: blue;
    }
  </style>
</head>
<body>
  <div class="shape rgb">Red rgb(255, 0, 0)</div>
  <div class="shape hex">Green #008000</div>
  <div class="shape name">Blue (color by name)</div>
</body>
</html>
```

- **Transitions**

```
<!DOCTYPE html>
<html>
<head>
 <title>CSS Transitions</title>
 <style>
   .box {
     width: 100px;
     height: 100px;
     margin: 10px;
     display: inline-block;
     transition: 0.5s;
     text-align: center;
     background: black;
   }
 </style>
</head>
<body>

<h2>CSS Transitions</h2>

<!-- Width -->
<div class="box" onmouseover="this.style.width='150px'"
onmouseout="this.style.width='100px'"></div>
<p>Width Change</p>

<!-- Scale -->
<div class="box" onmouseover="this.style.transform='scale(1.3)'"
onmouseout="this.style.transform='scale(1)'"></div>
```

```
<p>Scale Change</p>

<!-- Rotate -->
<div class="box" onmouseover="this.style.transform='rotate(360deg)'"
onmouseout="this.style.transform='rotate(0deg)'"></div>
<p>Rotate</p>

</body>
</html>
```

- **Animations**

```
<!DOCTYPE html>
<html>
<head>
  <style>
    /* Container box */
    .f {
      width: 320px;
      height: 320px;
      border: 2px solid #000;
      position: relative;
      font-size: 12px;
      color: #fff;
      text-align: center;
      line-height: 50px;
      font-family: sans-serif;
    }
```

```
/* Common box style */
.b {
  width: 50px;
  height: 50px;
  position: absolute;
  top: 0;
  left: 0;
  overflow: hidden;
}

/* Translate animation */
.p {
  animation: path 4s linear infinite;
  background: red;
}

@keyframes path {
  0%   { top: 0;    left: 0; }
  25%  { top: 0;    left: 270px; }
  50%  { top: 270px; left: 270px; }
  75%  { top: 270px; left: 0; }
  100% { top: 0;    left: 0; }
}

/* Rotate animation */
.r {
  top: 135px;
  left: 135px;
  background: green;
```

```
  animation: r 3s linear infinite;
  transform-origin: center;
}

@keyframes r {
  0%   { transform: rotate(0); }
  100% { transform: rotate(360deg); }
}

/* Color change animation */
.c {
  top: 0;
  left: 135px;
  background: blue;
  animation: c 3s linear infinite;
}

@keyframes c {
  0%, 100% { background: yellow; }
  50%      { background: purple; }
}

/* Scale animation */
.s {
  top: 135px;
  left: 0;
  background: orange;
  animation: s 2s ease-in-out infinite;
}
```

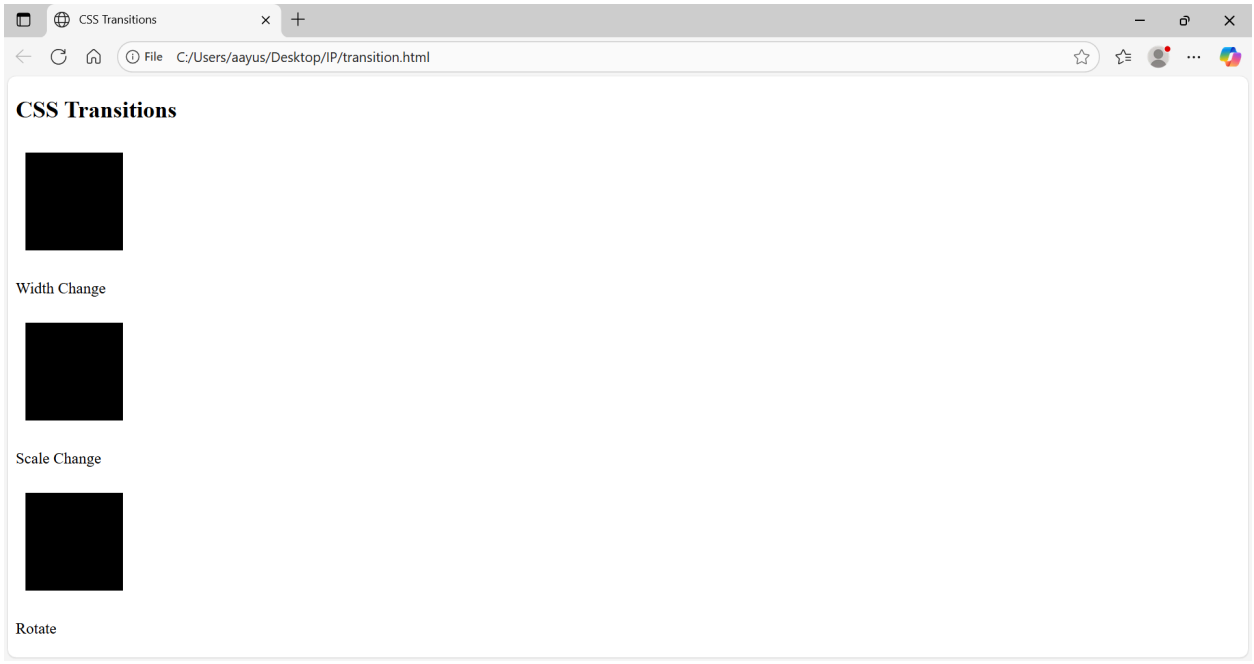
```
    @keyframes s {
      0%, 100% { transform: scale(1); }
      50%      { transform: scale(1.5); }
    }
  </style>
</head>
<body>

  <div class="f">
    <div class="b p">Translate</div>
    <div class="b r">Rotate</div>
    <div class="b c">Color</div>
    <div class="b s">Scale</div>
  </div>

</body>
</html>
```

OUTPUT:

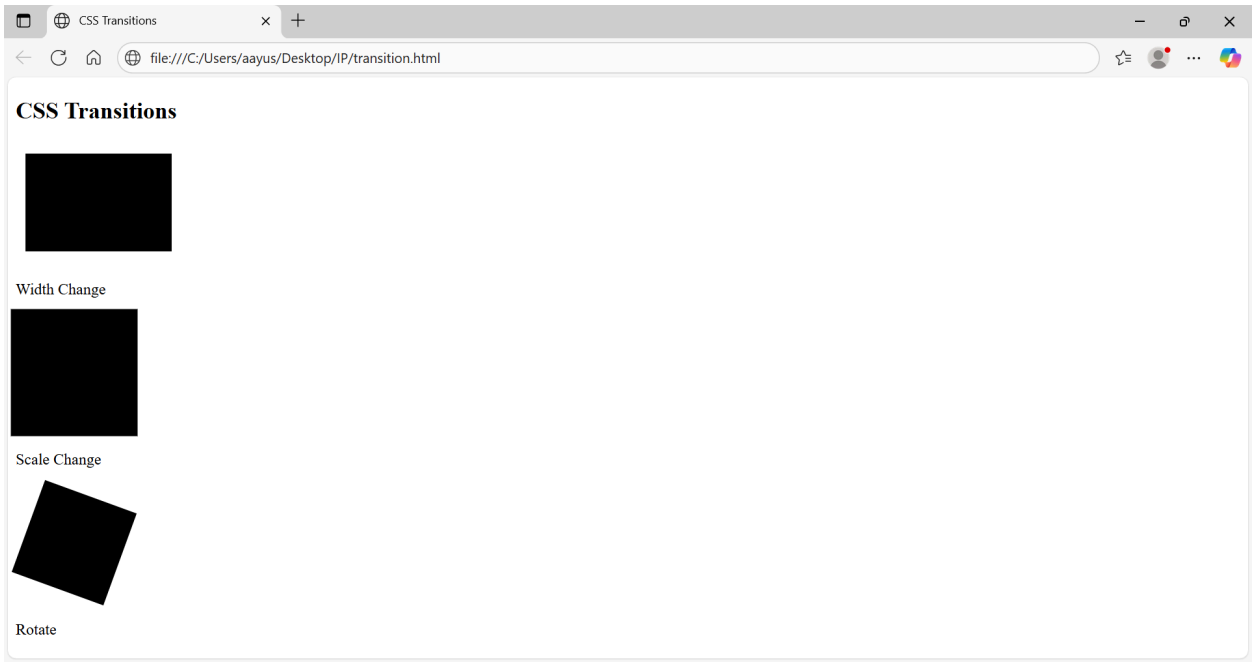
- **Typography:**

- **Color Modes:**

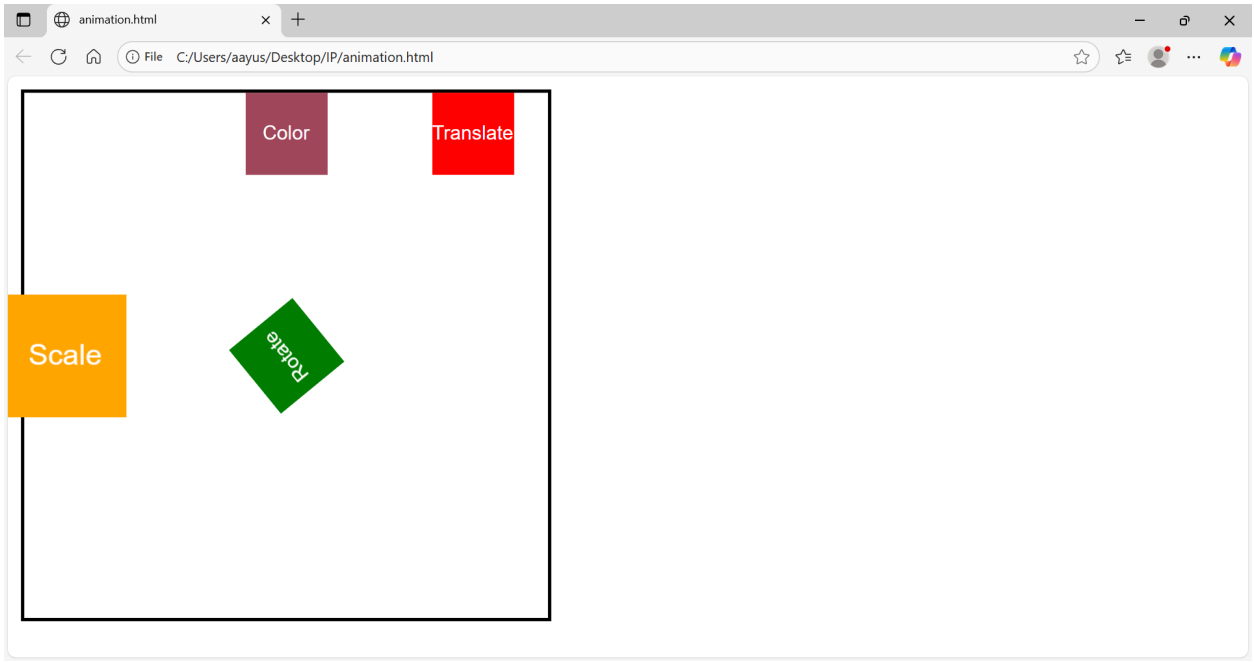
- **Before Transition:**



- **During Transition (on hover):**



- **Animation (running infinitely):**



CONCLUSION: This experiment successfully demonstrates typography, various color modes, transitions and animations using HTML & CSS.

LO MAPPING: LO2

ASSIGNMENT- 4

(Bootstrap)

AIM: To develop a webpage using the Bootstrap framework.

THEORY: Bootstrap is a popular open-source front-end framework that simplifies the development of responsive, mobile-first websites. It offers a set of pre-designed HTML, CSS, and JavaScript components that help developers build consistent and visually appealing interfaces quickly.

One of Bootstrap's core features is its **grid system**, which is based on a 12-column layout. Using rows and columns (.row, .col-*), developers can create responsive layouts that adapt smoothly to any screen size. The **navbar** is another essential component, allowing for responsive navigation menus that collapse on smaller screens and can include links, branding, and dropdowns.

Breadcrumbs are used to show users their navigation path and are styled with the .breadcrumb class. They're useful for improving site navigation and user experience. The **jumbotron** (used in older versions) is a large header section designed to highlight key messages or content with extra padding and standout styling.

Bootstrap also makes building **forms** easy with clean input styles using .form-control and layout tools like .form-group and .form-check. For user interaction, **buttons** are styled with .btn classes and come in various types like .btn-primary or .btn-danger, making it easy to indicate different actions.

CODE:

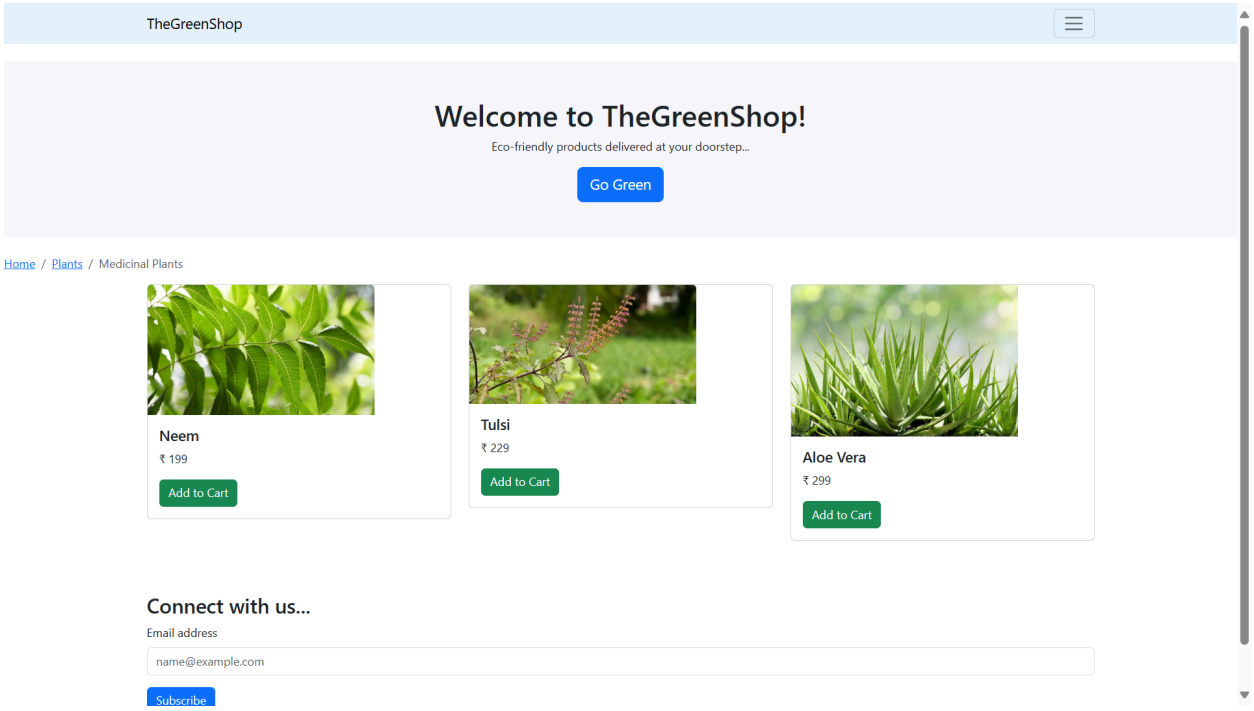
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>E-commerce Page</title>
    <!-- Bootstrap CSS CDN -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.7/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-LN+7fdVzj6u52u30Kp6M/trliBMCMKTyK833zpbD+pXdCLuTusPj6
97FH4R/5mcr" crossorigin="anonymous">
  </head>
  <body>
    <nav class="navbar" style="background-color: #e3f2fd;" data-bs-theme="light">
      <div class="container">
        <a class="navbar-brand" href="#">TheGreenShop</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav"
          aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
          <ul class="navbar-nav ms-auto">
            <li class="nav-item"><a class="nav-link active" href="#">Home</a></li>
            <li class="nav-item"><a class="nav-link" href="#">Products</a></li>
            <li class="nav-item"><a class="nav-link" href="#">Cart</a></li>
            <li class="nav-item"><a class="nav-link" href="#">Contact</a></li>
```

```
          </ul>
        </div>
      </div>
    </nav>
    <div class= "bg-light p-5 rounded my-4 text-center">
      <h1>Welcome to TheGreenShop!</h1>
      <p>Eco-friendly products delivered at your doorstep...</p>
      <a href="#" class="btn btn-primary btn-lg">Go Green</a>
    </div>
    <nav aria-label="breadcrumb">
      <ol class="breadcrumb">
        <li class="breadcrumb-item"><a href="#">Home</a></li>
        <li class="breadcrumb-item"><a href="plants.html">Plants</a></li>
        <li class="breadcrumb-item active" aria-current="page">Medicinal Plants</li>
      </ol>
    </nav>
    <div class="container">
      <div class="row">
        <!-- 1 -->
        <div class="col-md-4 mb-4">
          <div class="card">
            <img
src="https://i0.wp.com/detechter.com/wp-content/uploads/2016/12/neem-plant.jpg?fit=1042%2C596&ssl=1" class="card-img-top img-fluid w-75" alt="Image 1"  />
            <div class="card-body">
              <h5 class="card-title">Neem</h5>
              <p class="card-text">₹ 199</p>
              <a href="#" class="btn btn-success">Add to Cart</a>
            </div>
```

```
        </div>
      </div>
      <!-- 2 -->
      <div class="col-md-4 mb-4">
        <div class="card">
          <img
src="https://tse2.mm.bing.net/th/id/OIP.LtTEDukTH9npTNEyNEkbwQHaD4?r=0&cb=thfc1&rs=1&pid=ImgDetMain&o=7&rm=3" class="card-img-top img-fluid w-75"
alt="Product 2" />
          <div class="card-body">
            <h5 class="card-title">Tulsi</h5>
            <p class="card-text">₹ 229</p>
            <a href="#" class="btn btn-success">Add to Cart</a>
          </div>
        </div>
      </div>
      <!--  3 -->
      <div class="col-md-4 mb-4">
        <div class="card">
          <img
src="https://www.almanac.com/sites/default/files/users/Almanac%20Staff/aloe-vera-cgdeaw_ss_full_width.jpg" class="card-img-top img-fluid w-75" alt="Product 3" />
          <div class="card-body">
            <h5 class="card-title">Aloe Vera</h5>
            <p class="card-text">₹ 299</p>
            <a href="#" class="btn btn-success">Add to Cart</a>
          </div>
        </div>
      </div>
```

```
        </div>
      </div>
      <div class="container my-5">
        <h3>Connect with us...</h3>
        <form>
          <div class="mb-3">
            <label for="emailInput" class="form-label">Email address</label>
            <input type="email" class="form-control" id="emailInput"
placeholder="name@example.com" />
          </div>
          <button type="submit" class="btn btn-primary">Subscribe</button>
        </form>
      </div>
    </body>
</html>
```

OUTPUT:



CONCLUSION: This experiment successfully demonstrates the creation of an ecommerce web page using various bootstrap framework components.

LO MAPPING: LO3

ASSIGNMENT- 5

(JavaScript Basics)

AIM: To study the basic elements of javascript- conditional statements, loops, functions, inheritance, iterators & generators.

THEORY:

1. Conditional Statements: Conditional statements let the program make decisions.

- **if**: runs code if a condition is true.
- **else if / else**: run different code if the first condition is false.
- **switch**: checks one value against many cases.

2. Loops: Loops repeat code multiple times.

- **for**: runs code a set number of times.
- **while**: runs while a condition is true.
- **do...while**: runs at least once, then repeats while condition is true.
- **for...of**: goes through array/iterable values.
- **for...in**: goes through object keys.

3. Functions: Functions are reusable blocks of code.

- Can take inputs (**parameters**) and return outputs (**return value**).
- Declared with function or using arrow functions () => {}.

4. Inheritance: Inheritance allows one class to use properties/methods of another class.

- Achieved with **extends** in ES6 classes.

- Helps reuse code and build hierarchies (child class inherits from parent).

Types of Inheritance in JS:

- **Single**: one class inherits from another.
- **Multilevel**: a chain of inheritance (Grandparent → Parent → Child).
- **Hierarchical**: multiple classes inherit from the same parent.
- **Multiple:** JS does not support multiple inheritance directly.
- **Hybrid:** combines two or more different types of inheritance.

5. Iterators: Iterators provide a way to access elements of a collection one by one.

- Must have a **next()** method that returns { value, done }.
- Arrays, Sets, Maps already have built-in iterators.

6. Generators: Generators are special functions that can pause and resume.

- Declared with function*.
- Use yield to return values step by step.
- Useful for handling sequences or asynchronous tasks.

CODE:

- **Conditional Statements**

```
let input = prompt("Enter a number:");
let number = Number(input);

 if (number > 0) {
  console.log("The entered number is positive.");
} else if (number < 0) {
  console.log("The entered number is negative.");
```

```
} else {
  console.log("The entered number is zero.");
}
```

- **Loops**

```
for (let i = 0; i < 3; i++) {
  console.log("for loop:", i);
}

let j = 0;
while (j < 3) {
  console.log("while loop:", j);
  j++;
}

let k = 0;
do {
  console.log("do...while loop:", k);
  k++;
} while (k < 3);

const fruits = ["apple", "banana", "cherry"];
for (const fruit of fruits) {
  console.log("for...of loop:", fruit);
}

const person = { name: "Ayush", age: 20, city: "BOM" };
for (const key in person) {
  console.log("for...in loop:", key, "=", person[key]);
```

```
}
```

- **Functions**

```
function printTable() {
  let num = prompt("Enter a number:");
  console.log(`Multiplication table of ${num}:`);
  for (let i = 1; i <= 10; i++) {
    console.log(`${num} x ${i} = ${num * i}`);
  }
}
printTable();
```

- **Inheritance (hybrid combining single, multilevel & hierarchical inheritance)**

```
class Vehicle {
  constructor(name) {
    this.name = name;
  }
  show() {
    console.log("Vehicle: " + this.name);
  }
}

class Car extends Vehicle {
  drive() {
    console.log(this.name + " drives.");
  }
}

class Bike extends Vehicle {
```

```
  ride() {
    console.log(this.name + " rides.");
  }
}

class SportsCar extends Car {
  turbo() {
    console.log(this.name + " uses turbo!");
  }
}

let c = new Car("Generic Car");
let b = new Bike("Mountain Bike");
let s = new SportsCar("Ferrari");

c.show();
c.drive();

b.show();
b.ride();

s.show();
s.drive();
s.turbo();
```

- **Iterators**

```
const myArray = [10, 20, 30];
const iterator = myArray[Symbol.iterator]();
```

```
console.log(iterator.next());
console.log(iterator.next());
console.log(iterator.next());
console.log(iterator.next());
```

- **Generators**

```
function* myGenerator() {
  yield "Hello";
  yield "World";
}

const gen = myGenerator();
console.log(gen.next()); // { value: "Hello", done: false }
console.log(gen.next()); // { value: "World", done: false }
console.log(gen.next()); // { value: undefined, done: true }
```

OUTPUT:

- **Conditional Statements**

```
Output

Enter a number:117
The entered number is positive.

=== Code Execution Successful ===
```

- **Loops**

```
Output
for loop: 0
for loop: 1
for loop: 2
while loop: 0
while loop: 1
while loop: 2
do...while loop: 0
do...while loop: 1
do...while loop: 2
for...of loop: apple
for...of loop: banana
for...of loop: cherry
for...in loop: name = Ayush
for...in loop: age = 20
for...in loop: city = BOM

=== Code Execution Successful ===
```

- **Functions**

```
Output
Enter a number:117
Multiplication table of 117:
117 x 1 = 117
117 x 2 = 234
117 x 3 = 351
117 x 4 = 468
117 x 5 = 585
117 x 6 = 702
117 x 7 = 819
117 x 8 = 936
117 x 9 = 1053
117 x 10 = 1170

=== Code Execution Successful ===
```

- **Inheritance**

```
Output

Vehicle: Generic Car
Generic Car drives.
Vehicle: Mountain Bike
Mountain Bike rides.
Vehicle: Ferrari
Ferrari drives.
Ferrari uses turbo!

=== Code Execution Successful ===
```

- **Iterators**

```
Output

{ value: 10, done: false }
{ value: 20, done: false }
{ value: 30, done: false }
{ value: undefined, done: true }

=== Code Execution Successful ===
```

- **Generators**

```
Output

{ value: 'Hello', done: false }
{ value: 'World', done: false }
{ value: undefined, done: true }

=== Code Execution Successful ===
```

CONCLUSION: This experiment successfully demonstrates conditional statements, loops, functions, inheritance, iterators & generators in JavaScript.

LO MAPPING: LO4

ASSIGNMENT- 6

(Arrow Functions, DOM & CSS Manipulation)

AIM: To study arrow functions, dom manipulation and css manipulation in javascript.

THEORY: Arrow Functions in JavaScript are a shorter & simpler way to write functions. They make code cleaner and easier to read, especially when using small functions. For example, instead of writing function add(a, b) { return a+b; }, we can write const add = (a, b) => a+b;. Arrow functions are often used in modern JavaScript because they are concise and useful in events, callbacks, and simple calculations.

DOM and CSS Manipulation allow JavaScript to interact with the web page. DOM (Document Object Model) manipulation means changing the structure or content of a webpage, like updating text, adding new elements, or removing them. CSS manipulation means changing the styles of elements, such as colors, sizes, or shapes, directly from JavaScript. Together, these let you make webpages more interactive and dynamic, for example: clicking a button to change text, add content, or style elements.

CODE:

- **Arrow Functions:**

```
const add = (a, b) => a + b;
const multiply = (a, b) => a * b;
const square = (n) => n * n;

console.log("Add:", add(5, 3));
console.log("Multiply:", multiply(4, 2));
```

```
console.log("Square:", square(6));
```

- **DOM Manipulation:**

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Manipulation</title>
</head>
<body>
  <h2 id="heading">Welcome!</h2>
  <button onclick="changeText()">Change Heading</button>
  <button onclick="addParagraph()">Add Paragraph</button>

  <script>
    const changeText = () => {
      document.getElementById("heading").innerText = "Heading Changed!";
    }

    const addParagraph = () => {
      const para = document.createElement("p");
      para.innerText = "This is a new paragraph.";
      document.body.appendChild(para);
    }
  </script>
</body>
</html>
```

- **CSS Manipulation:**

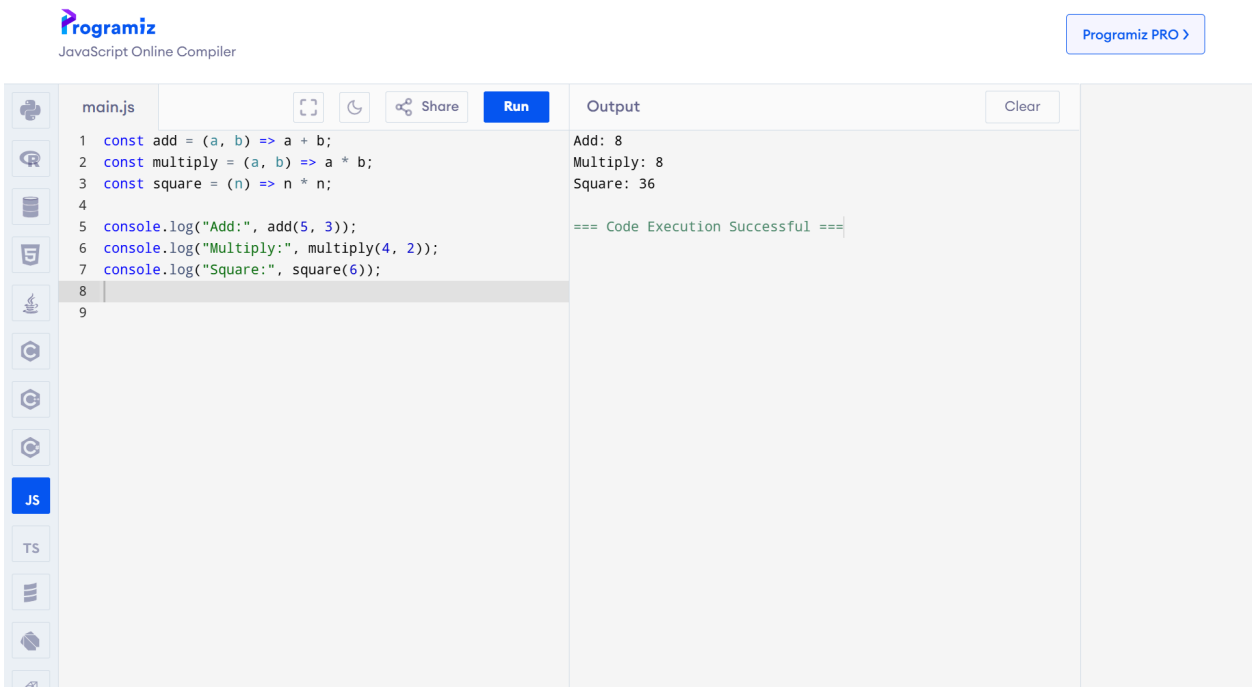
```
<!DOCTYPE html>
```

```
<html>
<head>
  <title>CSS Manipulation</title>
  <style>
    #box {
      width: 120px;
      height: 120px;
      background-color: lightblue;
      transition: all 0.5s;
    }
  </style>
</head>
<body>
  <div id="box"></div>
  <button onclick="makeCircle()">Make Circle</button>
  <button onclick="resetBox()">Reset</button>
  <button onclick="changeColor()">Change Color</button>

  <script>
    const makeCircle = () => {
      document.getElementById("box").style.borderRadius = "50%";
    }

    const resetBox = () => {
      const box = document.getElementById("box");
      box.style.borderRadius = "0";
      box.style.backgroundColor = "lightblue";
    }
```

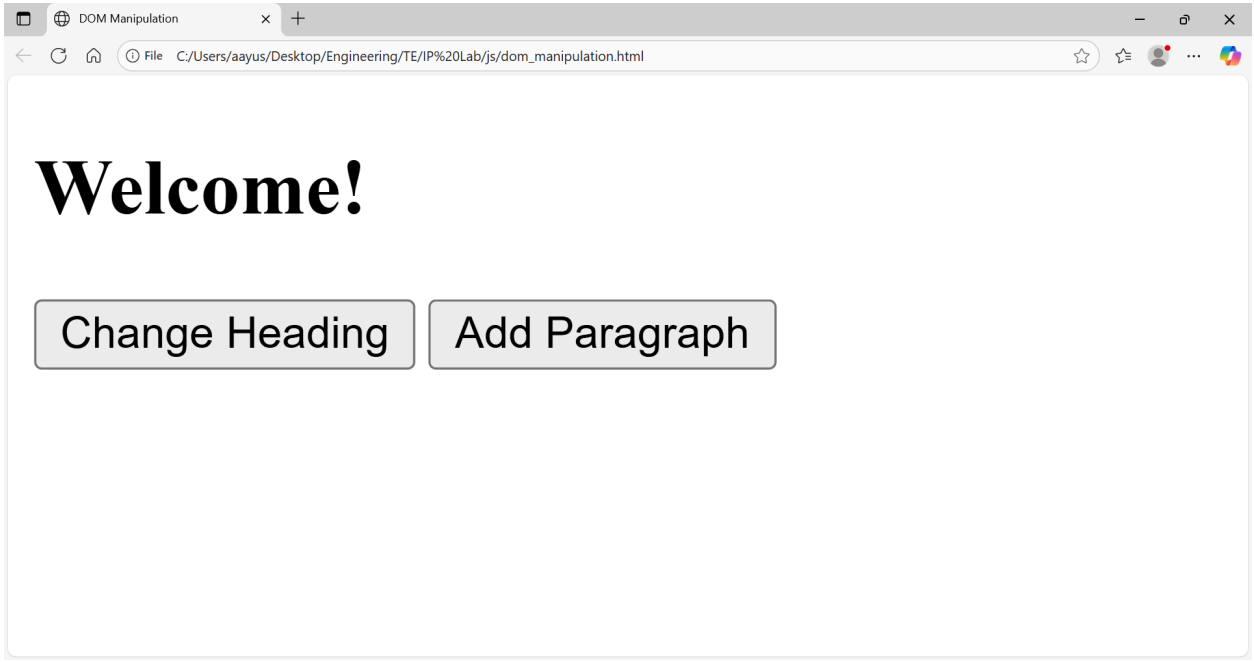
```
  const changeColor = () => {
    const colors = ["tomato", "limegreen", "gold", "violet", "skyblue"];
    const box = document.getElementById("box");
    box.style.backgroundColor = colors[Math.floor(Math.random() * colors.length)];
  }
 </script>
</body>
</html>
```

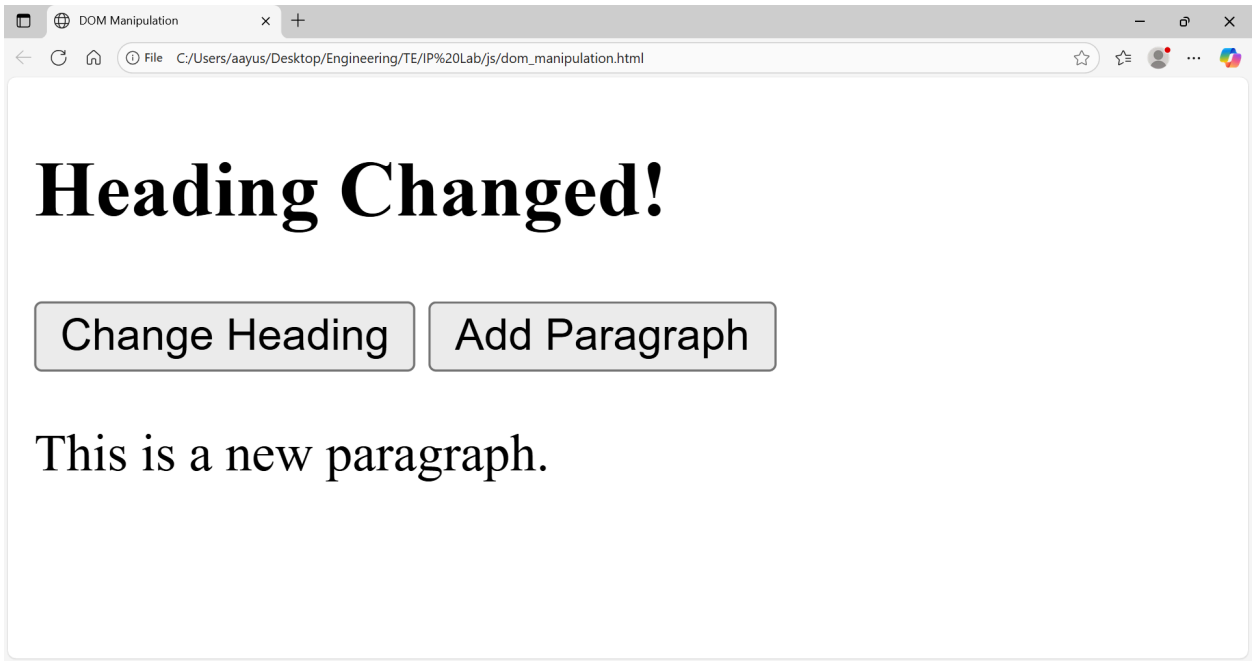
OUTPUT:

- **Arrow Function**

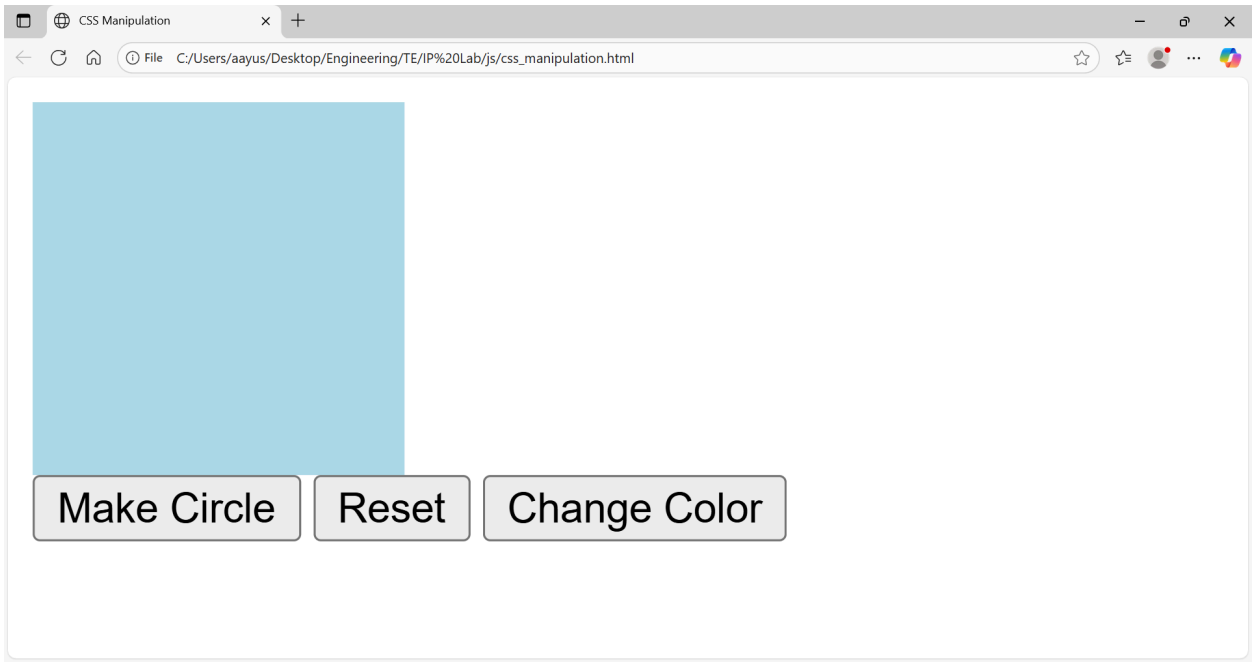


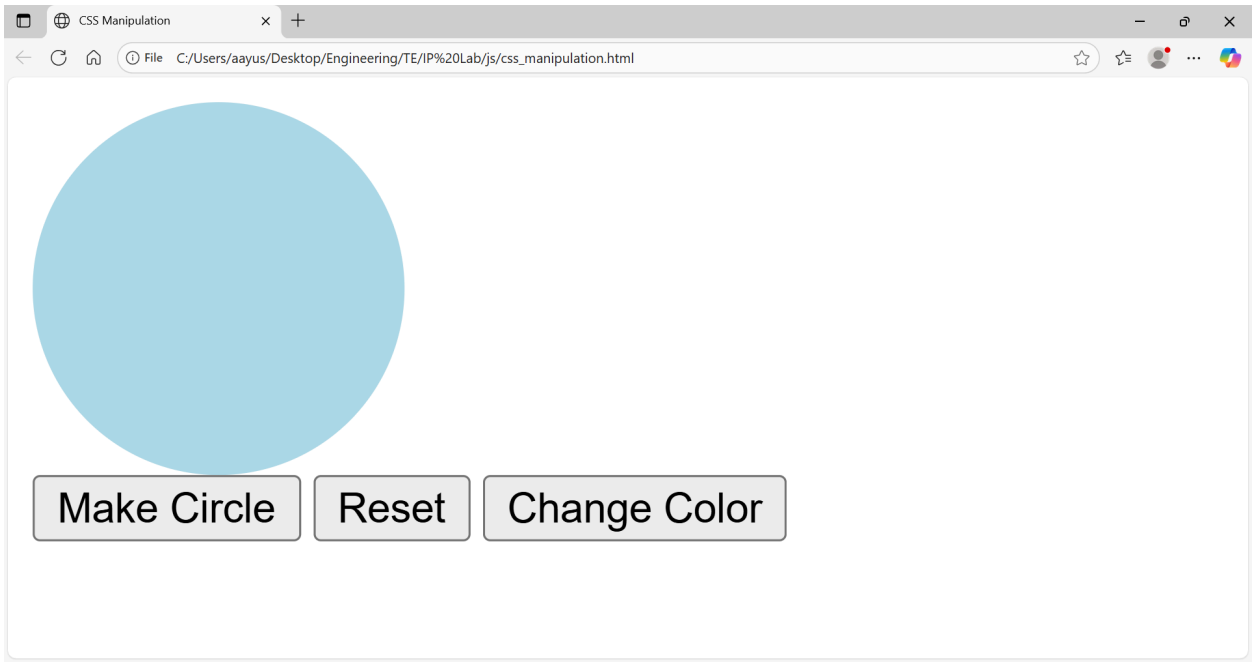
- **DOM Manipulation**

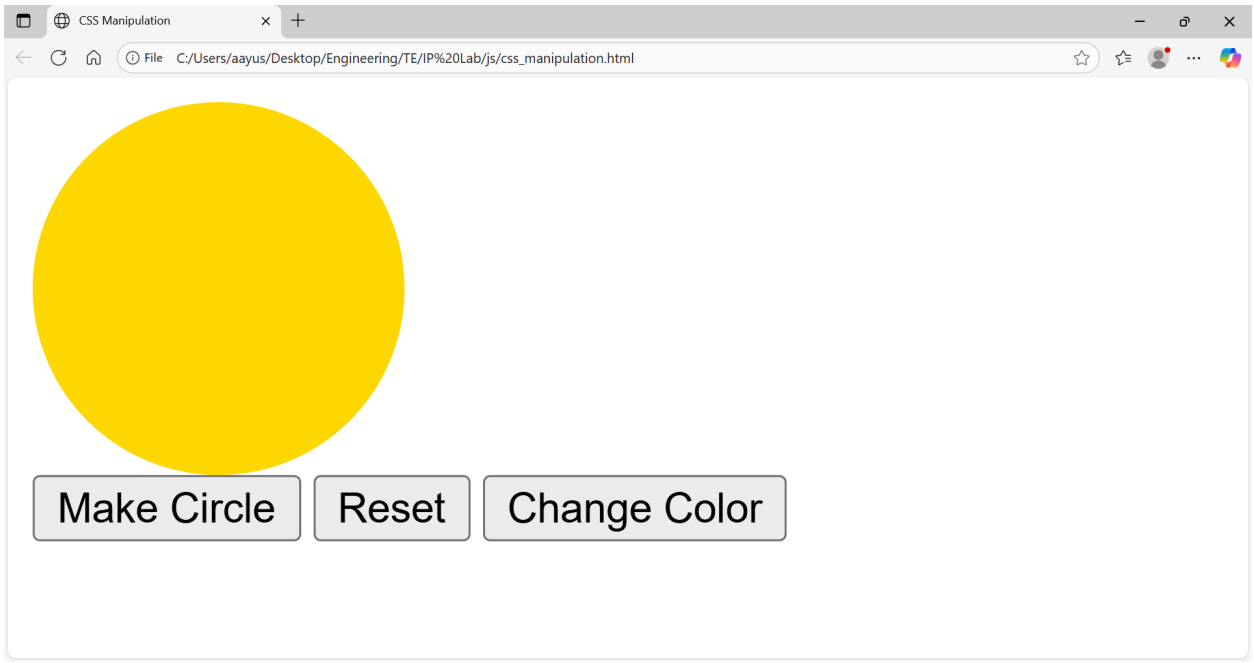




- **CSS Manipulation**







CONCLUSION: This experiment successfully demonstrates arrow functions, dom manipulation and css manipulation in javascript.

LO MAPPING: LO4

ASSIGNMENT- 7

(Form Validation using JavaScript)

AIM: To design an HTML5 form and validate it using JavaScript.

THEORY: Form Validation is the process of checking the information entered by users in a web form before it is sent to the server. It ensures that the data is correct, complete, and follows the required rules like format, length, or type.

Advantages of form validation:

- Improves user experience by giving instant feedback.
- Reduces server load by preventing incorrect data from being submitted.
- Helps maintain data quality and avoid errors.
- Enhances security by ensuring only proper data is processed.

Types of Validation:

1. Client-side validation – Runs in the browser using JavaScript; shows errors immediately.
2. Server-side validation – Runs on the server to double-check the data because client-side validation can be bypassed.

JavaScript Form Validation Working:

- Access form elements: Use document.getElementById() or querySelector() to get input values.

- Check conditions: Validate if fields are filled, match patterns, or have correct length.
- Use regular expressions (regex): Validate formats like email, phone numbers, or passwords.
- Prevent form submission: Use return false or event.preventDefault() when validation fails.
- Show error messages: Display messages near the fields to guide users.

CODE:

```
<!DOCTYPE html>
<head>
    <title>Student Registration</title>
</head>
<body>
   <h2>Student Registration</h2>
   <form id="registrationForm" onsubmit="return validateForm()">
      <p>
         <label for="name">Name:</label><br>
         <input type="text" id="name" name="name">
         <span id="nameError"></span>
      </p>
      <p>
         <label>Gender:</label><br>
         <input type="radio" id="male" name="gender" value="M"> <label
for="male">Male</label>
         <input type="radio" id="female" name="gender" value="F"> <label
for="female">Female</label>
```

```
    <span id="genderError"></span>
</p>
<p>
    <label for="branch">Branch:</label><br>
    <select id="branch" name="branch">
        <option value="">--Select Branch--</option>
        <option value="Comps">Comps</option>
        <option value="IT">IT</option>
        <option value="Aids">AIDS</option>
        <option value="Extc">EXTC</option>
        <option value="Chem">Chem</option>
    </select>
    <span id="branchError"></span>
</p>
<p>
    <label for="division">Division:</label><br>
    <input type="text" id="division" name="division">
    <span id="divisionError"></span>
</p>
<p>
    <label for="prn">PRN (16 digits):</label><br>
    <input type="text" id="prn" name="prn" maxlength="16">
    <span id="prnError"></span>
</p>
<p>
    <label for="year">Year:</label><br>
    <select id="year" name="year">
        <option value="">--Select Year--</option>
        <option value="FE">FE</option>
```

```
            <option value="SE">SE</option>
            <option value="TE">TE</option>
            <option value="BE">BE</option>
        </select>
        <span id="yearError"></span>
    </p>
    <p>
        <label for="phoneNo">Phone Number (10 digits):</label><br>
        <input type="text" id="phoneNo" name="phoneNo" maxlength="10">
        <span id="phoneNoError"></span>
    </p>
    <p>
        <label for="emailId">Email ID:</label><br>
        <input type="email" id="emailId" name="emailId">
        <span id="emailIdError"></span>
    </p>
    <p>
        <label for="dob">Date of Birth (YYYY-MM-DD):</label><br>
        <input type="text" id="dob" name="dob" placeholder="YYYY-MM-DD">
        <span id="dobError"></span>
    </p>

    <p>
        <label for="password">Password (min 8 chars, 1 uppercase, 1 lowercase, 1
number, 1 special char):</label><br>
        <input type="password" id="password" name="password">
        <span id="passwordError"></span>
    </p>
    <p>
```

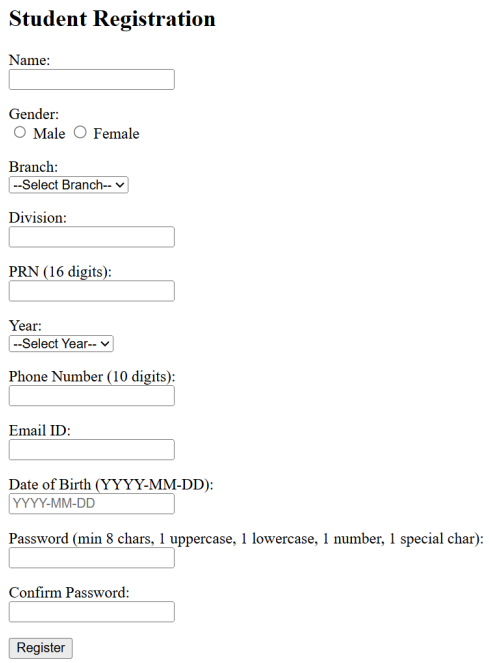
```
        <label for="confirmPassword">Confirm Password:</label><br>
        <input type="password" id="confirmPassword" name="confirmPassword">
        <span id="confirmPasswordError"></span>
    </p>
    <button type="submit">Register</button>
</form>

<script>
    function validateForm() {
        let isValid = true;
        // Clear all previous error messages
        document.querySelectorAll('span').forEach(el => el.textContent = '');

        // Get form values
        const name = document.getElementById('name').value.trim();
        const gender = document.querySelector('input[name="gender"]:checked');
        const branch = document.getElementById('branch').value;
        const division = document.getElementById('division').value.trim();
        const prn = document.getElementById('prn').value.trim();
        const year = document.getElementById('year').value;
        const phoneNo = document.getElementById('phoneNo').value.trim();
        const emailId = document.getElementById('emailId').value.trim();
        const dob = document.getElementById('dob').value.trim();

        const password = document.getElementById('password').value;
        const confirmPassword = document.getElementById('confirmPassword').value;

        // Simple validation
        if (name === '') {
```

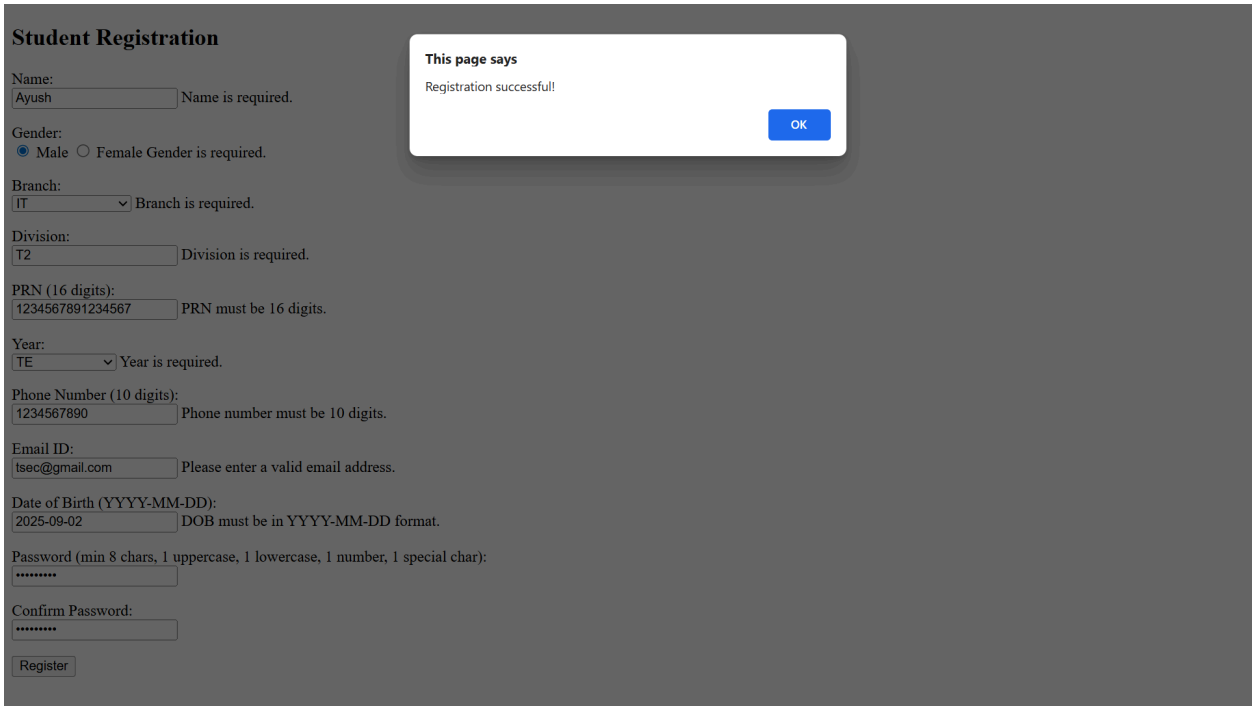
```
        document.getElementById('nameError').textContent = 'Name is required.';
        isValid = false;
    }
    if (!gender) {
        document.getElementById('genderError').textContent = 'Gender is required.';
        isValid = false;
    }
    if (branch === '') {
        document.getElementById('branchError').textContent = 'Branch is required.';
        isValid = false;
    }
    if (division === '') {
        document.getElementById('divisionError').textContent = 'Division is required.';
        isValid = false;
    }
    if (prn.length !== 16 || isNaN(prn)) {
        document.getElementById('prnError').textContent = 'PRN must be 16 digits.';
        isValid = false;
    }
    if (year === '') {
        document.getElementById('yearError').textContent = 'Year is required.';
        isValid = false;
    }
    if (phoneNo.length !== 10 || isNaN(phoneNo)) {
        document.getElementById('phoneNoError').textContent = 'Phone number must
be 10 digits.';
        isValid = false;
    }
    if (emailId === '' || !emailId.includes('@') || !emailId.includes('.')) {
```

```
            document.getElementById('emailIdError').textContent = 'Please enter a valid email address.';
            isValid = false;
        }

        // DOB validation (YYYY-MM-DD format)
        const dobRegex = /^\d{4}-\d{2}-\d{2}$/;
        if (!dobRegex.test(dob)) {
            document.getElementById('dobError').textContent = 'DOB must be in YYYY-MM-DD format.';
            isValid = false;
        }




        // Password validation using regex
        const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%?&])[A-Za-z\d@$!%?&]{8,}$/;

        if (!passwordRegex.test(password)) {
            document.getElementById('passwordError').textContent = 'Password must be at least 8 characters long, including 1 uppercase, 1 lowercase, 1 number, and 1 special character.';
            isValid = false;
        }
        if (password !== confirmPassword) {
            document.getElementById('confirmPasswordError').textContent = 'Passwords do not match.';
```

```
            isValid = false;
        }

        if (isValid) {
            alert('Registration successful!');
            return true;
        } else {
            return false;
        }
    }
  </script>
</body>
</html>
```

OUTPUT:





CONCLUSION: This experiment successfully demonstrates form validation using JavaScript on an HTML5 form.

LO MAPPING: LO4

ASSIGNMENT- 8

(React Hooks)

AIM: To demonstrate the concept of React hooks (useState, useEffect).

THEORY: React Hooks are special functions that let you use state and other React features without writing a class. The two most commonly used hooks are useState and useEffect. The useState hook allows a functional component to store and update local state variables, while useEffect lets you perform side effects — such as updating the document title, fetching data, or interacting with the browser — whenever certain values change.

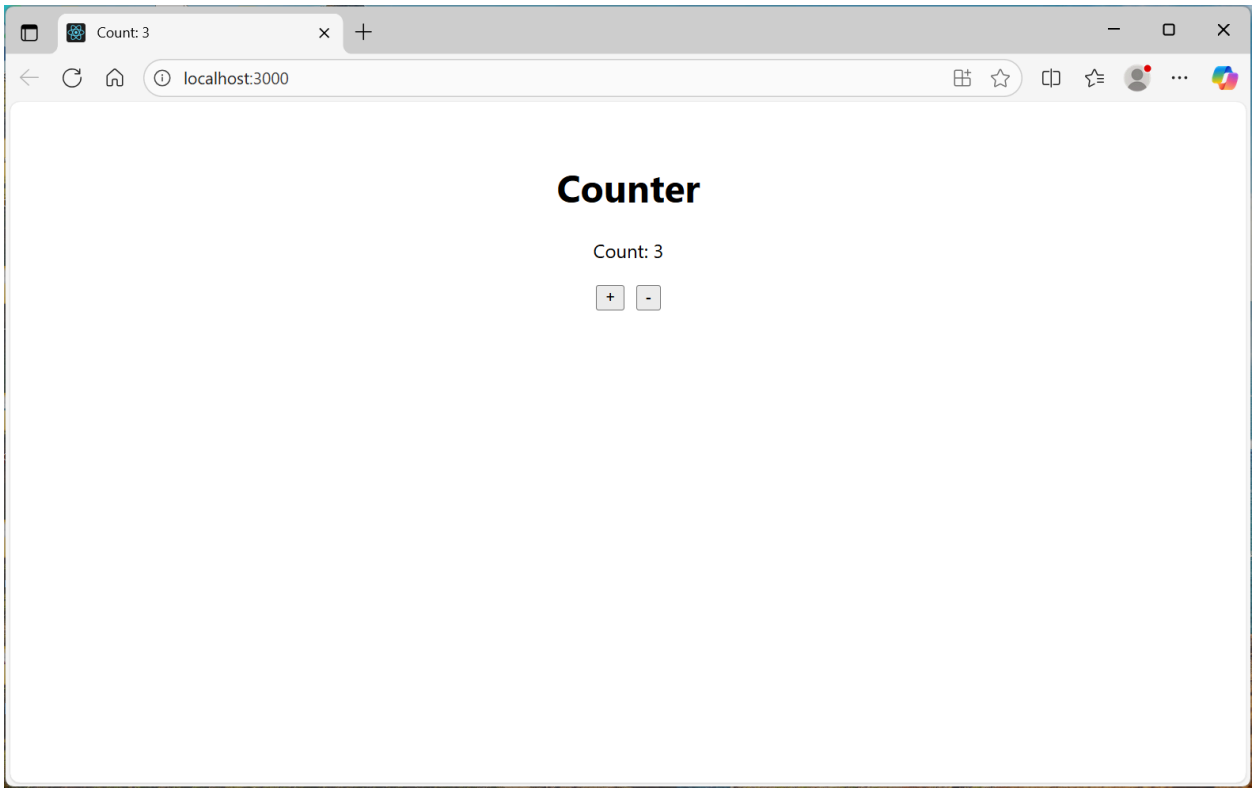
In this counter program, useState is used to create a count variable that keeps track of the current counter value and a function setCount to update it whenever the user clicks the buttons. The useEffect hook runs every time the count changes and updates the browser's document title to reflect the new count.

CODE:

```
import React, { useState, useEffect } from "react";
function Counter() {
  const [count, setCount] = useState(0);

  // useEffect runs every time 'count' changes
  useEffect(() => {
    document.title = `Count: ${count}`;
  }, [count]);

  return (
```

```
    <div style={{ textAlign: "center", marginTop: "50px" }}>
      <h1>Counter</h1>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>+</button>
      <button onClick={() => setCount(count - 1)} style={{ marginLeft: "10px" }}>
        -
      </button>
    </div>
  );
}

export default Counter;
```

OUTPUT:



CONCLUSION: This experiment successfully demonstrates the concept of React hooks (useState, useEffect) with the help of a counter program.

LO MAPPING: LO5

ASSIGNMENT- 9

(React Promises)

AIM: To implement the concept of asynchronous programming using promises in React

THEORY: Asynchronous programming allows a program to perform tasks without blocking the main thread, enabling other operations to continue while waiting for long-running tasks like API calls. In JavaScript, a Promise is an object that represents the eventual completion or failure of an asynchronous operation and can be in one of three states: pending, fulfilled, or rejected. Promises provide methods such as .then() to handle successful completion, .catch() to handle errors, and .finally() to run code after the operation finishes regardless of the outcome. In the code below, the fetch() function returns a Promise when requesting a random dog image from the API. The .then() methods process the response and update the component state with the image when the request succeeds, while .catch() captures any network or response errors and sets the error state. React automatically re-renders the UI based on these state changes, displaying the image on success, a loading message while waiting, or an error message if the request fails.

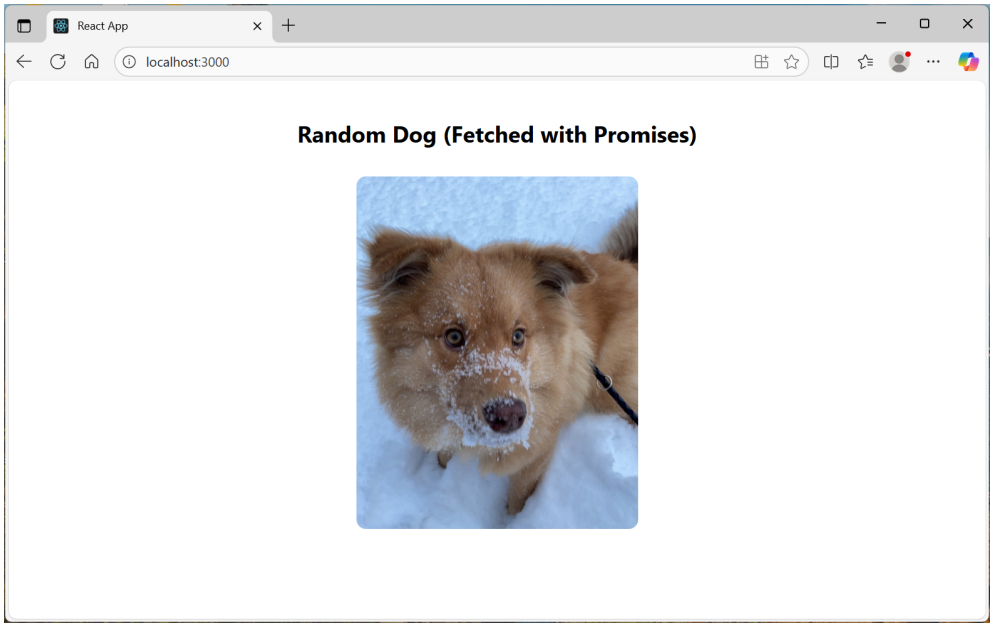
CODE:

```
import React, { useState, useEffect } from "react";
function RandomDog() {
  const [dogImage, setDogImage] = useState(null);
  const [error, setError] = useState("");
  const [loading, setLoading] = useState(true);

  useEffect(() => {
```

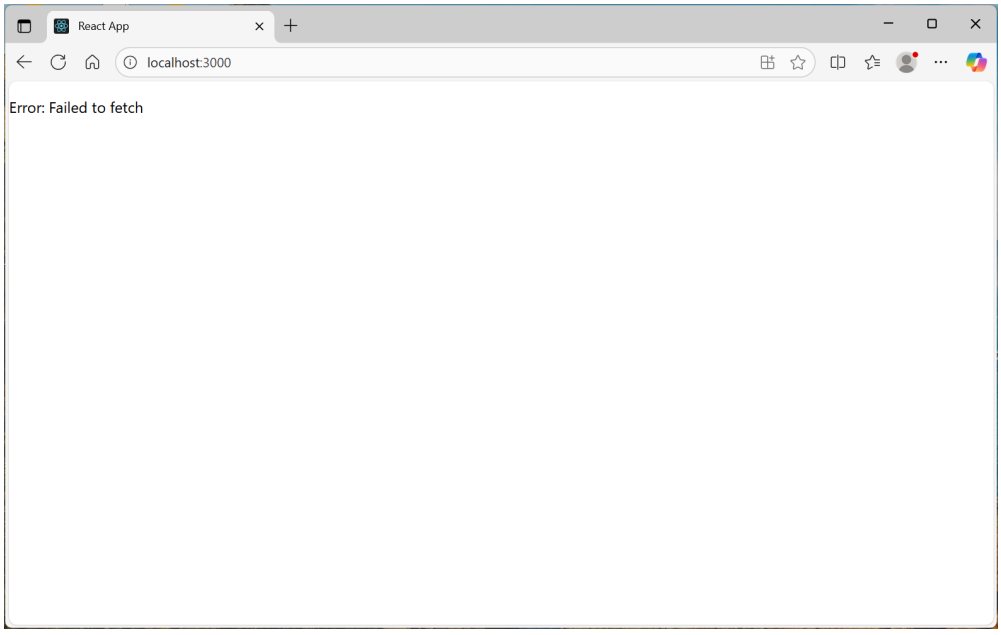
```
    // Fetch a random dog image from the Dog CEO API
    fetch("https://dog.ceo/api/breeds/image/random")
      .then((response) => {
        if (!response.ok) {
          throw new Error("Failed to fetch dog image");
        }
        return response.json();
      })
      .then((data) => {
        setDogImage(data.message); // API returns image URL in "message"
        setLoading(false);
      })
      .catch((err) => {
        setError(err.message);
        setLoading(false);
      });
  }, []);

  if (loading) return <p>Loading random dog...</p>;
  if (error) return <p>Error: {error}</p>;

  return (
    <div style={{ textAlign: "center", padding: "20px" }}>
      <h2>Random Dog (Fetched with Promises)</h2>
      <img
        src={dogImage}
        alt="Random Dog"
        style={{ width: "300px", borderRadius: "10px", marginTop: "10px" }}
      />
    </div>
  ); }
export default RandomDog;
```

OUTPUT:

- Successful fetch



- Failed to fetch (when the internet is turned off)



CONCLUSION: This experiment successfully demonstrates the concept of asynchronous programming using promises in React.

LO MAPPING: LO5

ASSIGNMENT- 10

(Node js)

AIM: To perform file operations in Node js (create, read, write, rename, delete).

THEORY: File handling is an essential part of programming that allows data to be stored permanently on disk. In Node.js, file operations are performed using the built-in File System (fs) module, which provides methods to manage files easily. The method fs.writeFileSync() is used to create a new file and write data into it, while fs.readFileSync() reads the contents of a file. To modify or add more data without deleting existing content, fs.appendFileSync() can be used. Files can also be renamed using fs.renameSync(), and deleted from the system with fs.unlinkSync(). Together, these functions demonstrate the fundamental file operations—creation, reading, writing, renaming, and deletion, showing how Node.js interacts with the file system to handle persistent data efficiently.

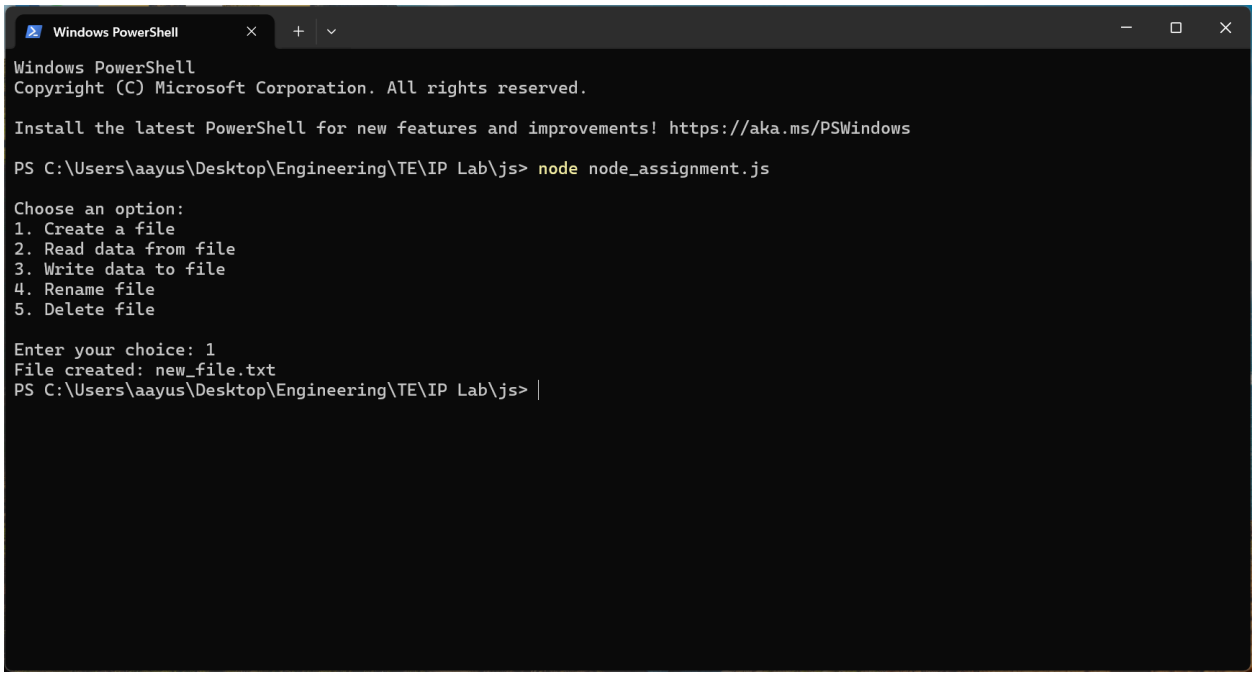
CODE:

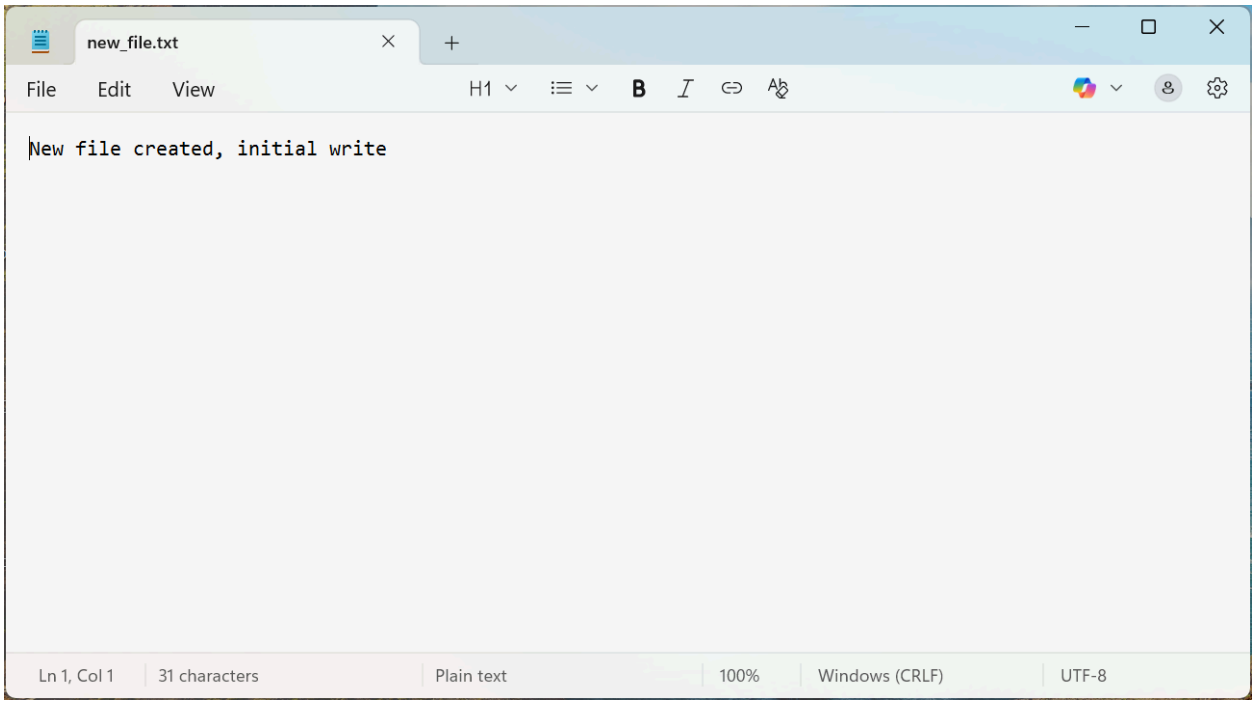
```
// import fs
const fs = require('fs');
const readline = require('readline');

// Createinterface
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});
```

```
console.log(`
Choose an option:
1. Create a file
2. Read data from file
3. Write data to file
4. Rename file
5. Delete file
`);

rl.question("Enter your choice: ", function(choice) {
  switch (choice) {
    case '1':
      fs.writeFileSync('new_file.txt', 'New file created, initial write');
      console.log('File created: new_file.txt');
      break;
    case '2':
      if (fs.existsSync('new_file.txt')) {
        const data = fs.readFileSync('new_file.txt', 'utf8');
        console.log('File content:', data);
      } else {
        console.log('File does not exist.');
      }
      break;
    case '3':
      if (fs.existsSync('new_file.txt')) {
        fs.appendFileSync('new_file.txt', 'New content written');
        console.log('\nFile updated.');
      } else {
        console.log('File does not exist.');
      }
      break;
    case '4':
```

```
    if (fs.existsSync('new_file.txt')) {
      fs.renameSync('new_file.txt', 'renamed.txt');
      console.log('File renamed to renamed.txt');
    } else {
      console.log('new_file.txt does not exist.');
    }
    break;
  case '5':
    if (fs.existsSync('renamed.txt')) {
      fs.unlinkSync('renamed.txt');
      console.log('renamed.txt deleted.');
    } else if (fs.existsSync('new_file.txt')) {
      fs.unlinkSync('new_file.txt');
      console.log('new_file.txt deleted.');
    } else {
      console.log('No file found to delete.');
    }
    break;

  default:
    console.log('Invalid choice');
 }
 rl.close();
});
```

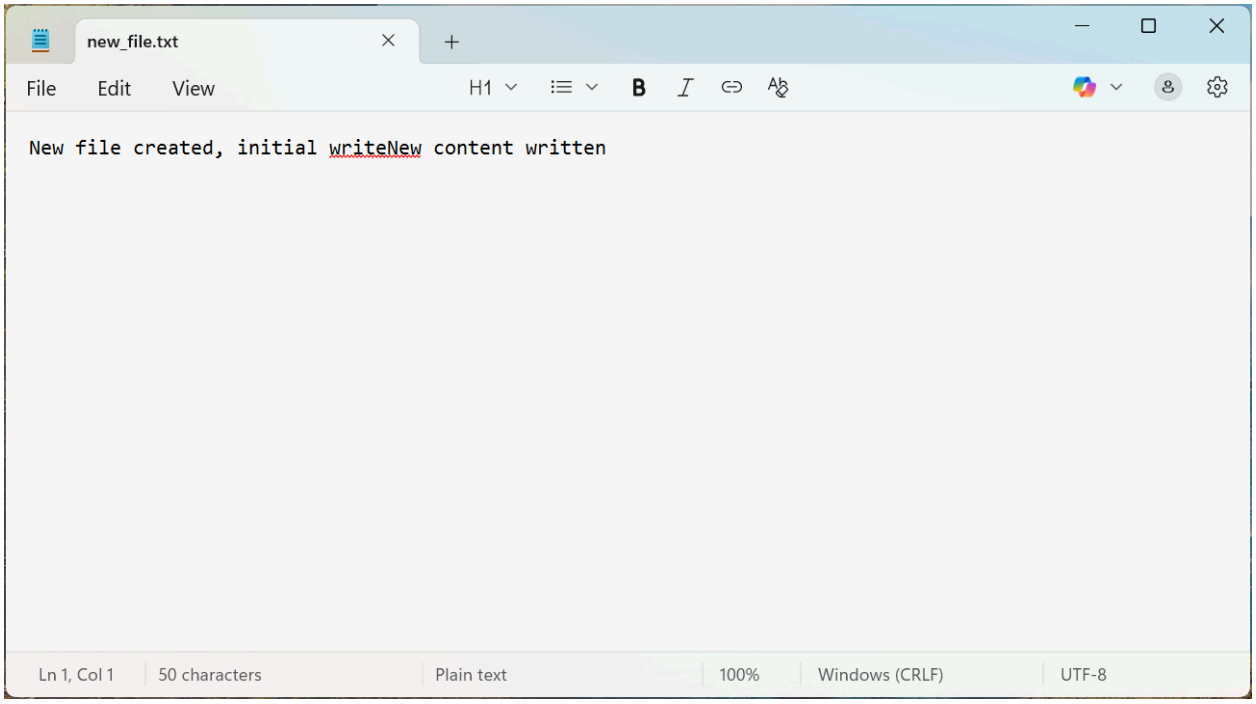
OUTPUT:


Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\aayus\Desktop\Engineering\TE\IP Lab\js> node node_assignment.js
Choose an option:
1. Create a file
2. Read data from file
3. Write data to file
4. Rename file
5. Delete file
Enter your choice: 1
File created: new_file.txt
PS C:\Users\aayus\Desktop\Engineering\TE\IP Lab\js>

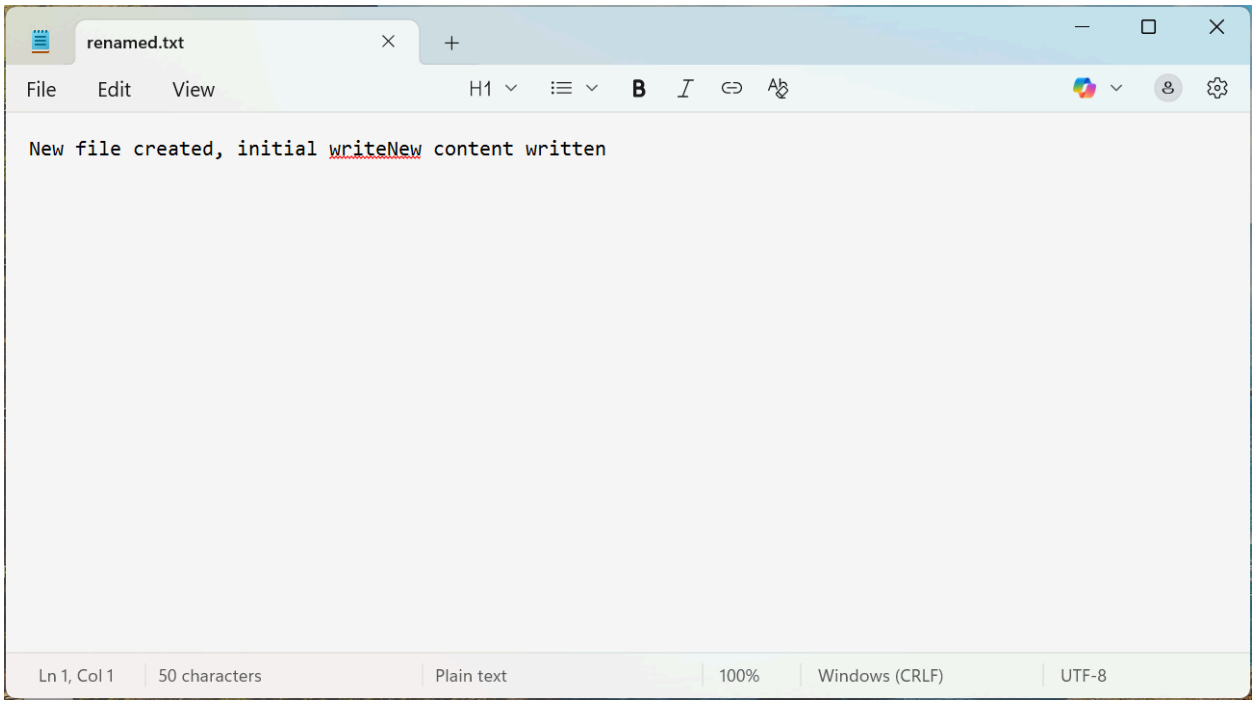

new_file.txt
File
Edit
View
H1
New file created, initial write
Ln 1, Col 1
31 characters
Plain text
100%
Windows (CRLF)
UTF-8

```
PS C:\Users\aayus\Desktop\Engineering\TE\IP Lab\js> node node_assignment.js

Choose an option:
1. Create a file
2. Read data from file
3. Write data to file
4. Rename file
5. Delete file

Enter your choice: 2
File content: New file created, initial write
PS C:\Users\aayus\Desktop\Engineering\TE\IP Lab\js>
```

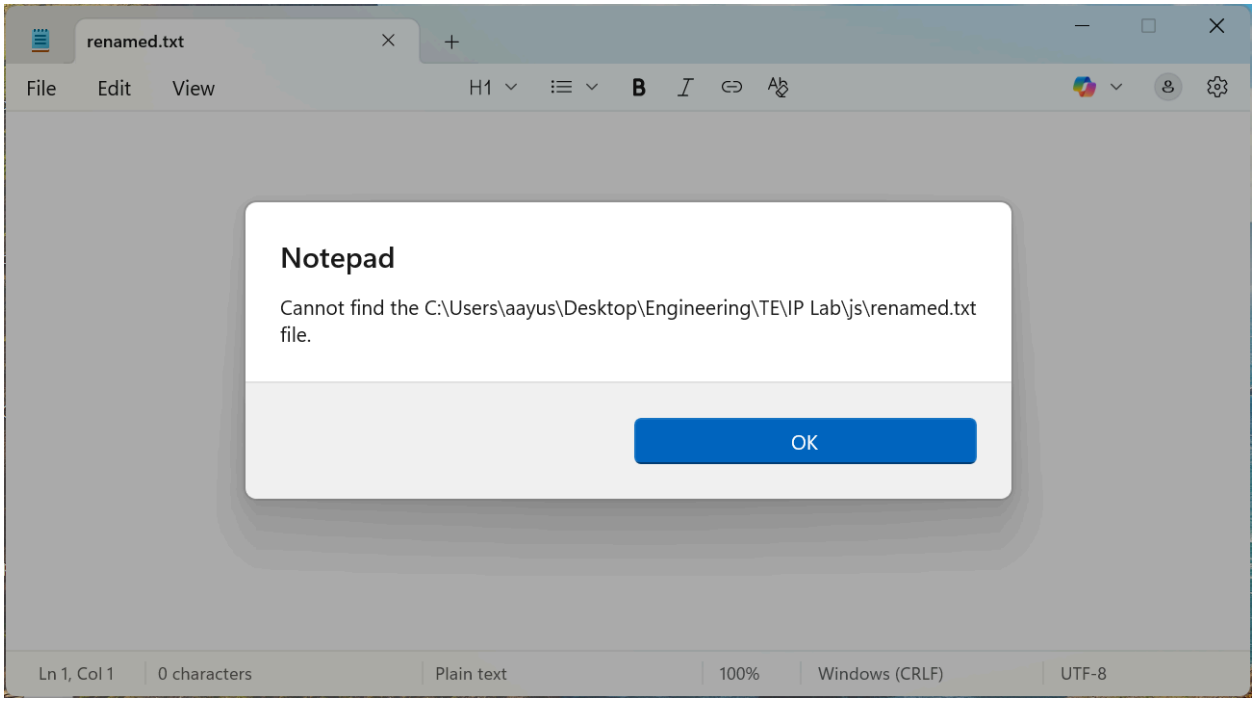
```
PS C:\Users\aayus\Desktop\Engineering\TE\IP Lab\js> node node_assignment.js

Choose an option:
1. Create a file
2. Read data from file
3. Write data to file
4. Rename file
5. Delete file

Enter your choice: 3

File updated.
PS C:\Users\aayus\Desktop\Engineering\TE\IP Lab\js>
```



```
PS C:\Users\aayus\Desktop\Engineering\TE\IP Lab\js> node node_assignment.js

Choose an option:
1. Create a file
2. Read data from file
3. Write data to file
4. Rename file
5. Delete file

Enter your choice: 4
File renamed to renamed.txt
PS C:\Users\aayus\Desktop\Engineering\TE\IP Lab\js>
```




```
PS C:\Users\aayus\Desktop\Engineering\TE\IP Lab\js> node node_assignment.js

Choose an option:
1. Create a file
2. Read data from file
3. Write data to file
4. Rename file
5. Delete file

Enter your choice: 5
renamed.txt deleted.
PS C:\Users\aayus\Desktop\Engineering\TE\IP Lab\js>
```



CONCLUSION: This experiment successfully demonstrates the various file operations in Node js.

LO MAPPING: LO6

AYUSH SHARMA
T2-117

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CLASS ASSIGNMENT-1

Q1. Compare XML and HTML.

Ans

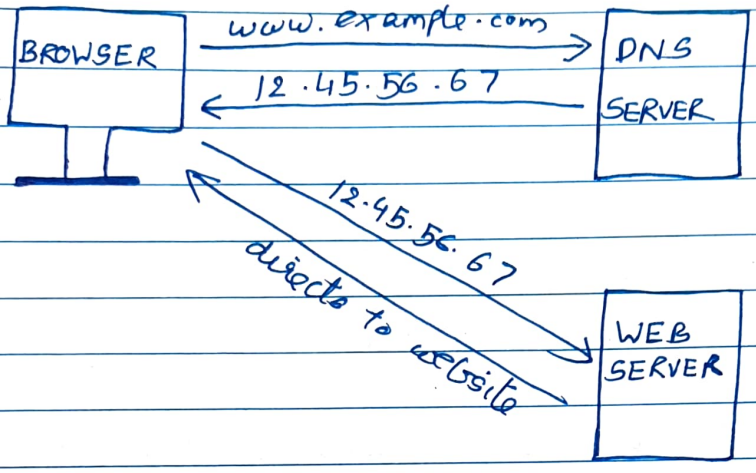
HTML	XML
• It was written in 1933.	It was released in 1996.
• It stands for 'hyper text markup language'.	It stands for 'extensible markup language'.
• HTML is static in nature.	XML is dynamic in nature.
• It was developed by WHATWG.	It was developed by 'Worldwide Web Consortium'.
• It is termed as a presentation language.	It is neither termed as a presentation nor a programming language.
• HTML is a markup language.	XML provides a framework to define markup languages.
• HTML can ignore small errors.	XML doesn't allow errors
• It has an extension of .html and .htm.	It has an extension of .xml.
• It is not case sensitive	It is case sensitive.
• Here tags are predefined.	Here tags are user-defined.

• It doesn't ~~support~~ preserve white spaces.	White space can be preserved in XML.
• HTML tags are used for displaying data.	XML tags are used for describing the data not for displaying
• Here closing tags are not necessary.	Here closing tags are necessary.
• HTML cannot carry data, it just displays it.	XML carries data to and from the database.
• HTML offers native object support.	In XML the objects are expressed by conventions using attributes.
• HTML document size is relatively small.	XML document size is relatively large as the approach of formatting & the codes both are lengthy.
• No additional application is needed for passing Java script code into HTML.	DOM (document object model) is required for parsing Java script codes & mapping of text.
• Example: <html> <body> John Roll No: 117 </body> </html>	Example: <student> <name> John </name> <roll_no> 117 </roll_no> </student>

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Q2. What is DNS?

Ans



→ DNS (domain name system) is a distributed and hierarchical naming system that translates human readable domain names (eg. www.example.com) into machine readable IP address (eg. 93.184.126.34). It acts like the 'phonebook' of the internet allowing users to access websites using names instead of numeric IP addresses.

- Need for DNS.

(1) Computers communicate using IP addresses.
(2) Remembering IP addresses is difficult for users.
(3) DNS resolves domain names to IP addresses automatically making browsing user friendly.

- DNS working steps:

(1) User enters a domain name in the address bar of browser.
(2) Browser sends query to DNS Resolver (usually ISP's server).
(3) Resolver contacts Root DNS To find the TLD server.

④ TLD server gives address of authoritative DNS.
⑤ Authoritative DNS returns the IP address to resolver.
⑥ Browser uses the IP address to connect to the website.

- Types of DNS servers:
  ① Root DNS servers - Handles requests for Top level domains.
  ② TLD DNS servers - Handle domains like .com, .org, .net, etc...
  ③ Authoritative DNS servers - Provides final IP address mapping.

- Advantages:
  ① Ease of use & user friendly
  ② Highly scalable.
  ③ High fault tolerance with redundant DNS servers
  ④ Fast & efficient
  ⑤ Load distribution through multiple IPs.

- Disadvantages
  ① Dependence on DNS servers.
  ② DNS servers may suffer from performance issues.
  ③ DNS spoofing ~~which~~
  ④ Privacy concerns as DNS queries can reveal a user's browsing history.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q3. Differentiate between ES5 & ES6

Ans

PARAMETER	ES5	ES6
Definition	5th edition of ECMA script which is a trademarked scripting language developed by ECMA international.	6th edition of ECMA script - a trademark scripting language defined by ECMA international.
Year of introduction	2009	2015
Data types supported	Primitive data types - string, number, boolean, null, undefined.	More additions to Java Script data types. New primitive data type 'symbol' for supporting unique values.
Variable definition	Only one way i.e. using 'var' keyword.	Two new ways i.e. ~~let~~ using 'let' & 'const'.
Performance	Relatively lower.	Relatively higher.
Object manipulation	~~Less~~ more time consuming	Less time consuming.

Function definition	Both function and return keywords are used.	A new feature - arrow function is used where the function keyword isn't required for function definition.
Community support.	Relatively larger.	Relatively smaller.

Q4. Explain arrow functions with suitable example.

Ans An arrow function is a shorter syntax for writing functions in Java Script. Arrow functions were introduced in ES6. They allow for a more concise & readable code especially in cases of small functions. Unlike regular functions, arrow functions don't have their own 'this'. But instead inherit from the surrounding context.

- Properties

→ Arrow functions use => for a compact syntax.
→ They inherit from the surrounding context.
→ Single expression functions have an implicit return.
→ They don't have access to the arguments object.
→ Best to declare with const, unless reassignment is needed.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

- Example

```
const add = (a, b) => a + b;
console.log(add(5, 3));
```

- Output : 8

→ 'add' is an arrow function that takes 2 parameters a & b and returns their sum.
→ The arrow function's concise syntax eliminates the need for the function keyword and curly braces for single-line expressions.

- Advantages :
→ Concise syntax
→ Lexical 'this' binding → inherits 'this' from surrounding scope avoiding common issues in callbacks
→ No need for function keyword.
→ Good for callbacks.

- Disadvantages :
→ No prototype property
→ Cannot be used with 'new'.
→ Cannot be generators
→ Cannot directly access arguments inside arrow function.

Q5. Explain the features of React js.

Ans React Js is an open source javascript library developed by Facebook for building fast, interactive and reusable user interfaces.

- Features of React Js.

① JSX (javascript extension)

→ It allows writing HTML like syntax inside javascript code. and makes the code more readable & easier to write.

eg. const element = <h1> Hello </h1>;

② Virtual DOM

→ React creates a virtual copy of the real DOM in memory. When changes occur, it updates only the changed elements in the real DOM improving speed.

③ One way data binding

→ Data flows in a single direction from parent to child components. This makes debugging & managing data easier.

④ Performance

→ Virtual DOM & efficient diffing algorithms result in fast UI updates.

→ Reduces unnecessary rendering for better performance.

⑤ Extension

→ React can be extended with libraries like React routes (for routing) & redux (for state management).

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

⑥ Conditional statements
→ JSX supports conditional rendering using javascript conditions

④ Components
→ UI is built using small reusable and independent pieces of code called components.

⑧ Simplicity
→ The combination of JSX, components and unidirectional data flow makes React code easy to understand & debug.

⑨ Strong community support
→ Large community, numerous libraries and continuous updates.

Q6. Explain different types of components in React Js with example.

Ans In react, components are reusable, independent blocks that define the structure and behaviour of the UI. They accept inputs (props) and return elements that describe what should appear on the screen.

→ There are 2 primary types of React components
- Functional Components

→ Functional components are simple and preferred for most used cases. They are Javascript functions that return React elements.

With the introduction of React Hooks, functional components can also manage state & lifecycle events.

- Stateless or stateful - Can manage state using React Hooks.
- Simples syntax - ideal for small & reusable components
- Performance - generally faster as no 'this' keyword is required.
- Example:

```
function Greet (props) {
    return <h1> Hello, {props.name} </h1>;
}
```

- Class Components :

→ Class components are ES6 classes that extend React.component. They include additional features like state management & lifecycle methods.

- State management: State is managed using this.state property
- Lifecycle methods - includes methods like componentDidMount, componentDidUpdate, etc.

- Example:

```
class Greet extends React.component {
    render () {
        return <h1> Hello {this.props.name}
                </h1> ;
    }
}
```

AYUSH SHARMA
T22-117

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

IP WRITTEN ASSIGNMENT-2

Q1. What is React Ref? When to use Refs and when not to use them?

Ans In React, a Ref (short for reference) is an object that provides a way to directly access and interact with the DOM elements or React component instances created during rendering. Normally react promotes a ~~declar~~ declarative approach, where the UI is controlled through state and props. However in certain cases direct access to a DOM node or component is required which is achieved using Refs.

- Ref creation

→ React.createRef() => in class based components
→ useRef() => hook in functional components

Each Ref object has a 'current' property, which stores the reference to the corresponding DOM node or React element.

- Example:

```
import React, {useRef} from "react";
function InputFocusExample()
{ const inputRef = useRef(null);

  function handleFocus()
{ inputRef.current.focus();
  }
  return
{ <div>
<input ref={inputRef} type="text" placeholder=
    "Enter text here" />
  <button onClick={handleFocus}>Focus Input
  </button></div>); }
```

→ In this example, the InputRef provides direct access to the <input> element, allowing the button to programatically focus the I/P field.

- When to use Refs:
  1. Managing focus, text selection or media playback
  2. Triggering or controlling animations
  3. Storing mutable values without causing re-renders
  4. Integrating with third-party libraries

- When not to use Refs:
  1. Managing data that should be in state
  2. Communication between components
  3. Replacing controlled components
  4. Overcomplicating component logic

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q 2. Compare. MVC vs Flux vs Redux

Ans

Parameters	MVC	Flux	Redux
Architecture type	Traditional design pattern with 3 components- Model, view, controller	Application architecture by Facebook for React apps.	State management library inspired by Flux.
Data Flow	Bidirectional between components	Unidirectional between components	Unidirectional between components
Components	Model, view, controller	Action, dispatcher, store, view	Action, Reducer, Store view.
Store / State management	Multiple models hold state	Multiple stores (each for specific parts of state)	Single centralized store (single source of truth)
Complexity	Simple(small apps) Complex(large apps)	Moderate	More boilerplate, but highly structured
Debugging	Difficult	Easier compared to MVC	Very easy.
Scalability	Poor for large apps	Better than MVC	Highly scalable
Performance	Degrades as app grows large	Better than MVC	Best performance

Q3. Explain features of Node.js

Ans Following are the features of Node.js:

(1) Open source :- Node.js is free and open source maintained by a large community of developers making it easily accessible & continuously supported.

(2) Simple and fast
→ Built on V8 engine, executes code quickly.

(3) Asynchronous
→ Uses non-blocking I/O for faster execution

(4) High scalability
→ Handles many client requests efficiently

(5) Single threaded
→ Uses event loop instead of multiple threads

(6) No buffering:
→ Outputs data in chunks, no data buffering

(7) Cross platform
→ Runs on Windows, Linux, macOS, etc.

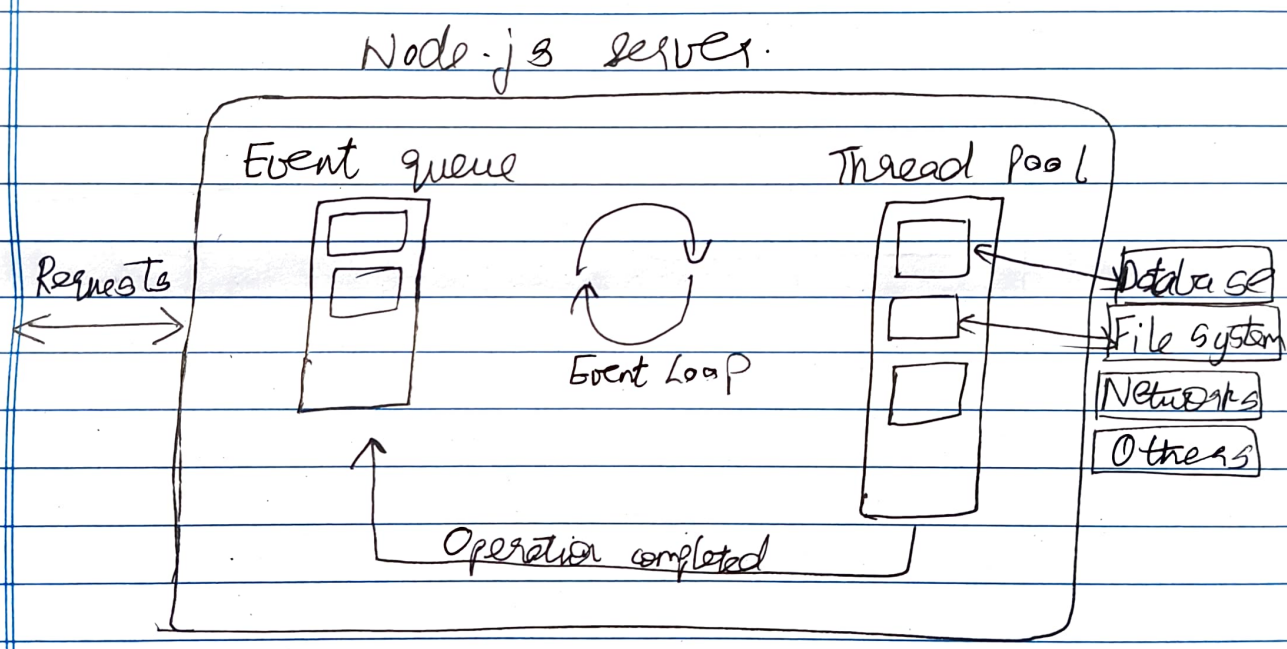
(8) License
→ Distributed under permissive MIT license.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q4. Explain event loops in node.js

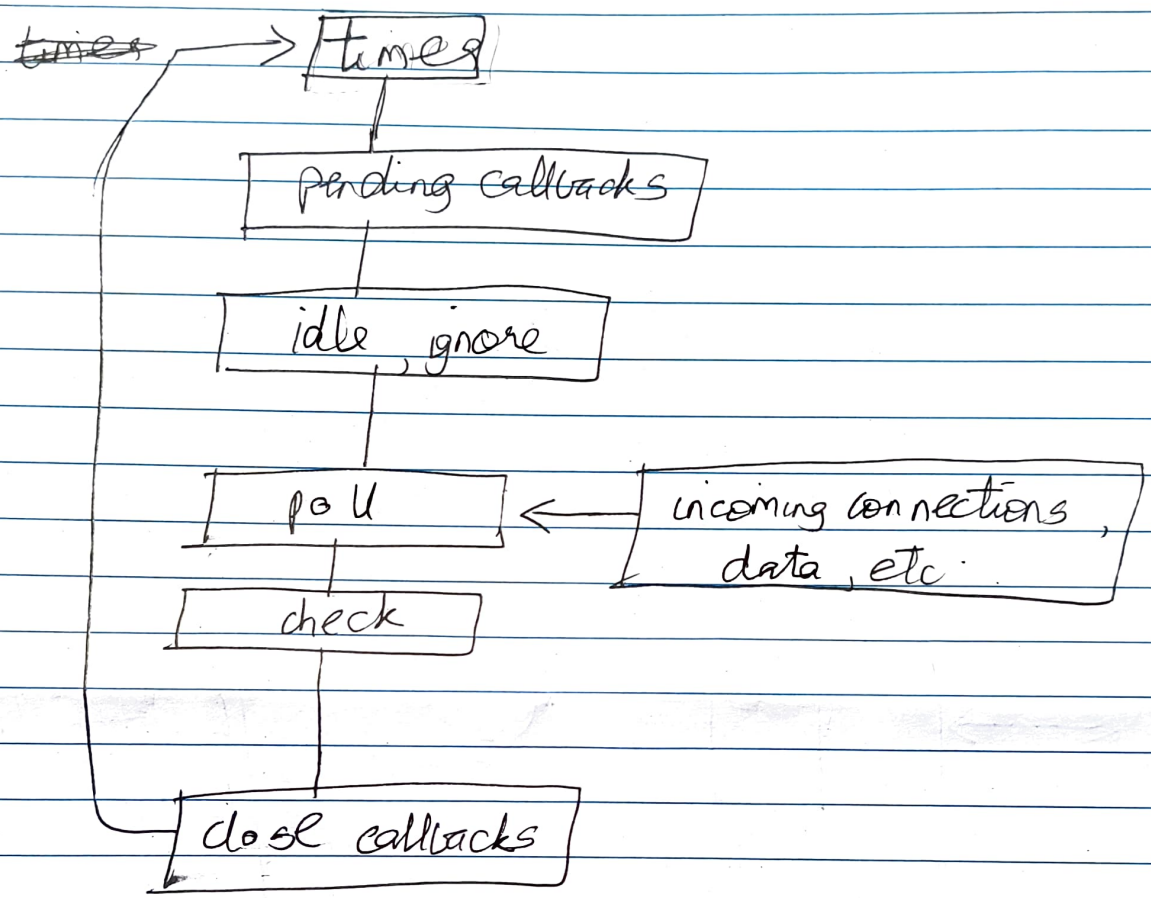
Ans The event loop allows Node to perform non blocking I/O operations despite the fact that JavaScript is single threaded. It is done by assigning operations to the operating system whenever & wherever possible

- Working of Event Loop.



(1) When node.js starts, it initializes event loop.
(2) The event loop continuously checks the call stack & callback queue.
(3) If the all stack is empty, it takes functions from the callback queue and pushes them to the call stack for execution.
(4) This process repeats enabling Node.js to handle multiple requests without blocking.

- Phases of event loop.



① Timers phase - Executes callbacks scheduled by setTimeout() & setInterval()
② Pending callbacks - Executes I/O callbacks deferred from the previous cycle.
③ Idle, prepare - Internal o/p
④ Poll - Retrieves new I/O events & executes I/O callbacks
⑤ check phase - executes setImmediate() callbacks
⑥ close callbacks - Executes close events

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q5. ~~Explain~~ What is Rest API, explain its principles.

Ans REST (representational state transfer) is an architectural style for designing web services.

It allows communication between client and server using standard HTTP methods like GET, POST, PUT and delete.

- 6 guiding principles of REST:

(1) Statelessness :- Each request carries all needed info; server doesn't store session state.

(2) Client - server :- Client handles UI, server handles data / logic, separation involves scalability.

(3) Uniform Interface :- Standard URI's, HTTP methods and formats (like JSON) ensure consistency

(4) Cacheable :- Responses define if they can be cached, improving performance & reducing load.

(5) Layered system :- API can use layers (proxies, load balancers, security) without client awareness.

(6) Code on demand :- Server can send executable code (eg. Javascript) to extend client features.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q6. Differentiate between express.js & node.js

Ans

Parameter	Node.js	Express.js
• Definition	Cross platform runtime environment developed on Google's V8 JS engine	Node.js framework for building backend web applications.
• Purpose.	Developing server side & network apps.	Building server side apps on node.js
• Programming language.	Written in JavaScript, C, C++	Written in JavaScript
• Web framework	Not a web framework	Web framework for node.js
• Use case	Can be used both on client & server side	Can be used only on server side.
• Features.	Provides many features to build apps & write database queries.	Provides routing components & supports middleware for easier app development.
• Language Support	Supports other languages like Typescript, Coffee-script, Ruby.	Supports JavaScript only.
• Companies using	Paypal, Walmart, LinkedIn, Uber, etc.	PayPal IBM, Fox sports, etc.