Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that ~~Mr./Miss~~ ADITYA SHARMA

of IT Department, Semester V with

Roll No. 116 has completed a course of the necessary

experiments in the subject IP - LAB under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2025 - 2026

Teacher In- Charge

09/10/25

Head of the Department

Date 9/10/2025

Principal

THADOMAL SHAHANI
TSEC
ENGINEERING SOLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1	Develop a webpage using HTML & HTML 5 tags	1-7	16/7/25	
2	Using CSS & CSS 3 enhance webpage in Assg. 1.	8-17	23/7/25	
3	Design web page with following topics.	18-22	30/7/25	
4	Develop a web page using Bootstrap framework	23-29	6/8/25	
5	WAP in JS to study conditional statement, loops, functions, inheritance.	30-35	13/8/25	
6	WAP in JS to study arrow function, DOM manipulation and CSS manipulation.	36-44	25/8/25	
7	Design an HTML 5 form and validate using JS.	45-52	3/9/25	
8	WAP to implement concept of react hooks	53-54	10/9/25	
9	WAP in react to implement concept of asynchronous programming using promises.	55-62	17/9/25	
10	WAP in Node.JS to - a] Create a file b] Write data to file c] Read data of file d] Rename a file e] Delete a file	63-67	24/9/25	
11	Written Assignment 1	68-73	12/8/25	
12	Written Assignment 2	74-81	30/9/25	

09/10/25

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>Shlok'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>Shlok 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> Shlok 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>

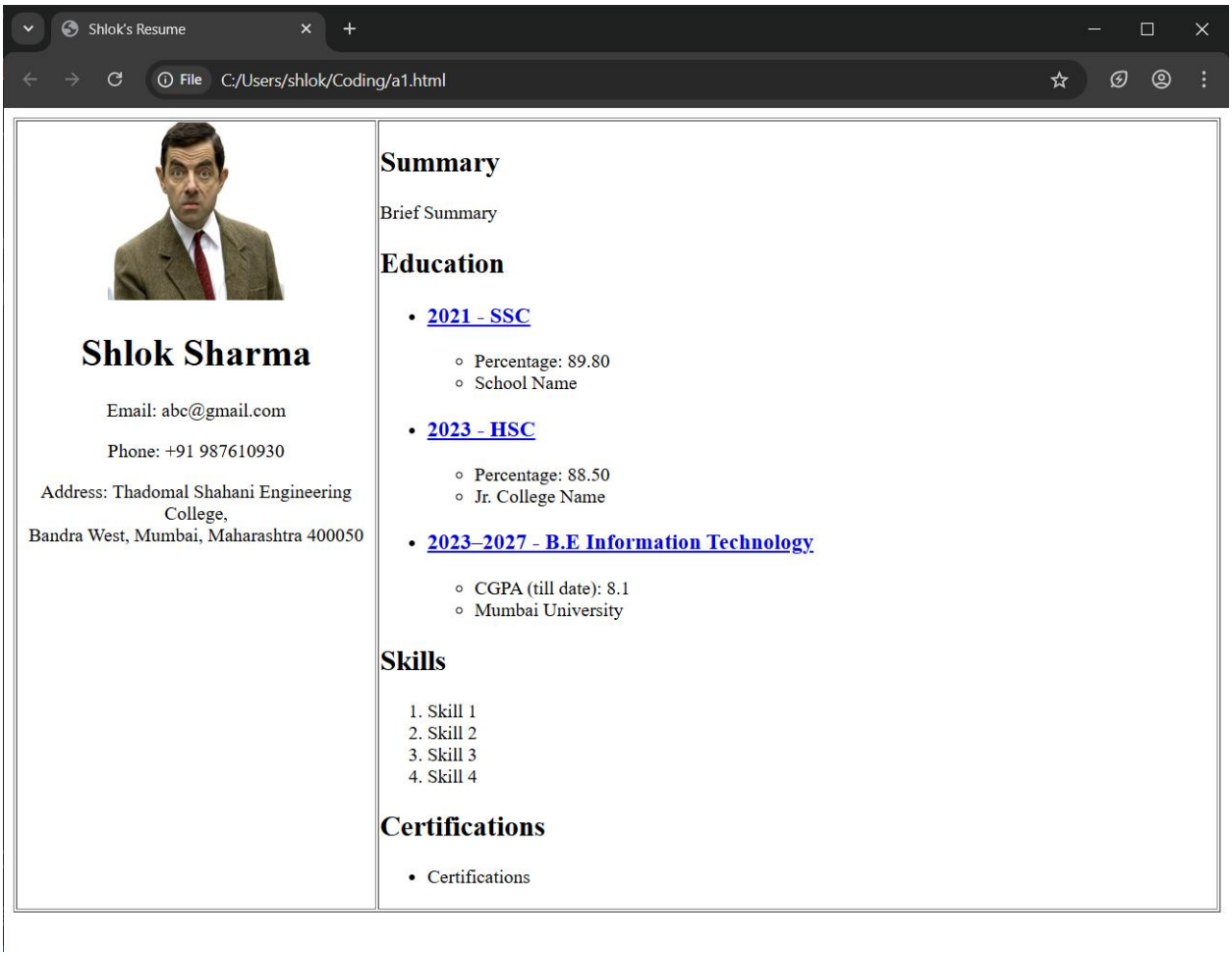
```
  <tr>
    <td>ITC404</td>
    <td>Automata Theory</td>
    <td>100</td>
    <td>91</td>
  </tr>
  <tr>
    <td>ITC405</td>
    <td>Computer Organization and Architecture</td>
    <td>100</td>
    <td>95</td>
  </tr>
</table>

<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:


Shlok's Resume
File
C:/Users/shlok/Coding/a1.html
Shlok Sharma
Email: abc@gmail.com
Phone: +91 987610930
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

Marksheet

File C:/Users/shlok/Coding/a1.html

University Of Mumbai

Marksheet

Name: Shlok Sharma

Seat Number: 241109

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

Get in Touch

Name:
Ayush

Email:
hello@xyz.com

Message:
Hello from this side !

Send

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>Shlok'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>Shlok 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>
```

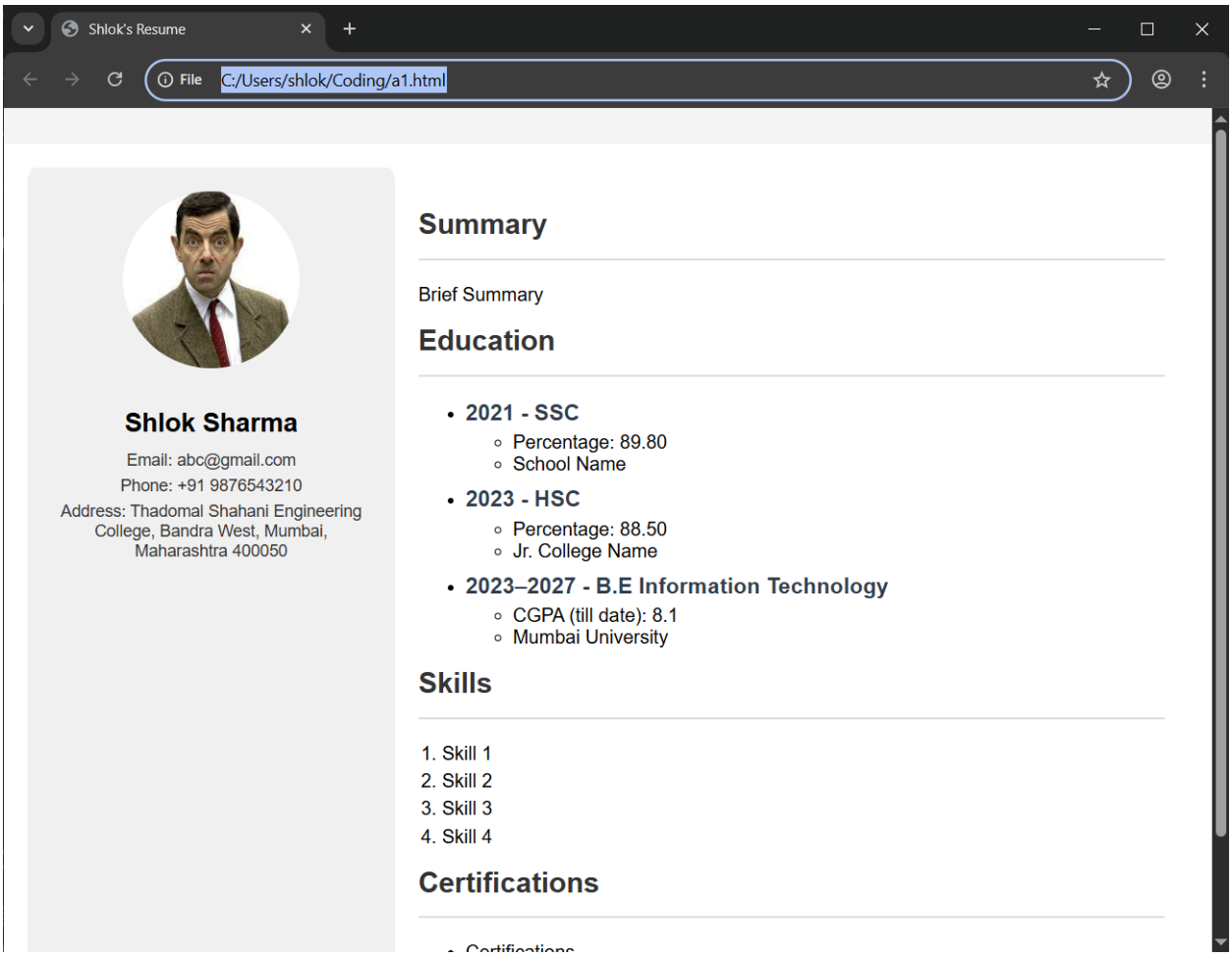
```
    body {
      font-family: Arial, sans-serif;
    }
    .marksheet {
      max-width: 600px;
      margin: auto;
      border:solid ;
      padding: 20px;
    }
    h2, h3 {
      text-align: center;
      font-family: 'Old English Text MT', serif;
    }
    table {
      width: 100%;
      margin-top: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

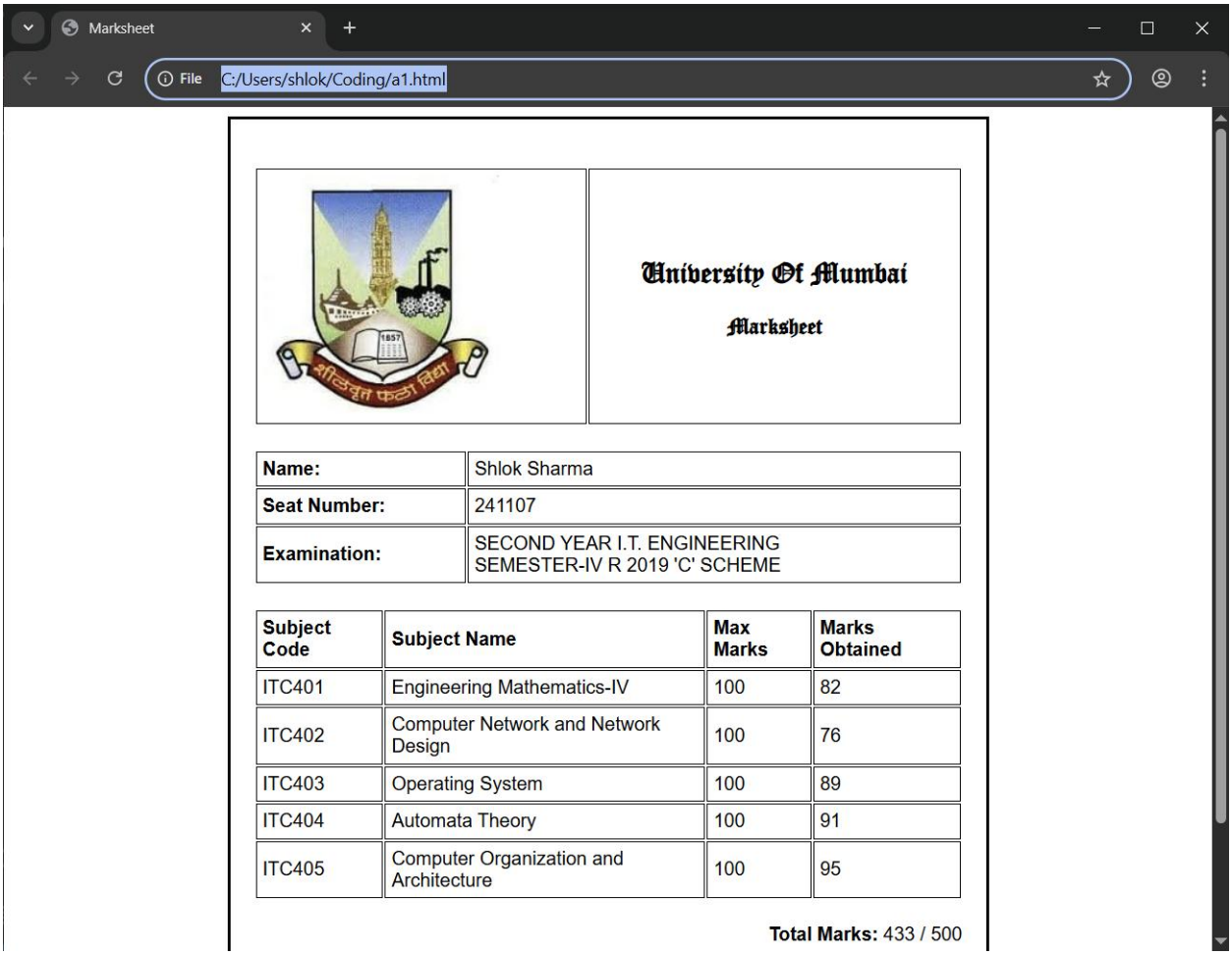
```
  <div class="marksheet">
<table >
    <tr>
    <td><img src="logo.jpeg"></td>
    <td><h2>University Of Mumbai</h2>
    <h3>Marksheet</h3></td>
    </tr>
</table>

    <table>
      <tr>
        <td><strong>Name:</strong></td>
        <td>Shlok 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:


Shlok's Resume
File
C:/Users/shlok/Coding/a1.html
Shlok 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



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

LO MAPPING: LO2

ASSIGNMENT 3 - LO2

AIM : Implement typography, color modes, transitions and animations

THEORY :

CSS: Typography, Color Modes, Transitions, and Animations

1. Typography in CSS

Typography in CSS refers to the styling and formatting of text on a webpage. It includes properties such as:

- font-family – Defines the font used (e.g., sans-serif, serif, monospace, or custom fonts like 'Roboto').
- font-size – Sets the size of the text (px, em, rem, %).
- font-weight – Controls the boldness (e.g., normal, bold, 400, 700).
- line-height – Adjusts the space between lines of text.
- letter-spacing and word-spacing – Manage spacing between letters and words.
- text-align – Aligns text (left, center, right, justify).
- text-transform – Controls casing (uppercase, lowercase, capitalize).

2. Color Modes in CSS

CSS supports various color formats to style elements:

- Named Colors – e.g., red, blue, black
- Hexadecimal – e.g., #ff0000 for red
- RGB – rgb(255, 0, 0)
- RGBA – rgba(255, 0, 0, 0.5) (adds transparency)
- HSL – hsl(0, 100%, 50%) (hue, saturation, lightness)
- HSLA – hsla(0, 100%, 50%, 0.5)

Each mode allows for different control—RGBA and HSLA support transparency, while HSL is great for adjusting themes programmatically.

3. Transitions in CSS

CSS transitions allow smooth changes between property values over time.

Basic properties:

- transition-property – The property to animate
- transition-duration – How long it takes
- transition-timing-function – The pace (ease, linear, ease-in-out)
- transition-delay – Delay before transition starts

4. Animations in CSS

CSS animations let you animate elements using keyframes.

- @keyframes – Defines the animation steps
- animation-name, animation-duration, animation-iteration-count, etc., control the animation behavior

CODE :

```
<!DOCTYPE html>

<html lang="en">

<head>

    <title>CSS Animations</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+Single:wght@100..900&display=
swap" rel="stylesheet">

        <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Audiowide">



        <style>

        .new_font {

        font-family: "Audiowide";

        color: hsl(240, 100%, 30%);

        }



        .outer-box {

        width: 500px;

        height: 100px;

        display: flex;

        background-color: aqua;

        }



        .box {

        width: 100px;

        height: 100px;

        background-color: red;

        display: flex;

        position: relative;

        animation: moveAndColor 3s ease-in-out infinite;
```

```
}

@keyframes moveAndColor {
0% {
        left: 0;
        background-color: red;
}
50% {
        left: 400px;
        background-color: blue;
}
100% {
        left: 0;
        background-color: red;
}
}

/* Button Styling */
.btn {
margin-top: 20px;
padding: 12px 24px;
font-size: 16px;
background-color: #008CBA; /* Blue */
color: white;
```

```
        border: none;
        border-radius: 5px;
        cursor: pointer;

        /* Transition */
        transition: background-color 0.4s ease, transform 0.3s ease, color 0.3s ease;
        }

        .btn:hover {
        background-color: #005f73; /* Darker blue */
        color: #ffcc00; /* Highlighted text color */
        transform: scale(1.1); /* Slight zoom on hover */
        }
        </style>
</head>
<body>

        <div class="outer-box">
        <div class="box"></div>
        </div>

        <div class="new_font">
        <h1>Trying out</h1>
        <p>Hello There</p>
```

```
        <p>Using Audiowide font</p>


        <button class="btn">Hover Me</button>

        </div>


</body>

</html>
```

OUTPUT :

CONCLUSION :

CSS provides powerful tools for creating visually appealing and interactive web content. Typography controls how text appears, color modes offer flexibility in styling, and transitions and animations bring interactivity and life to static designs—making modern web experiences smooth and engaging.

ASSIGNMENT 4 - LO3

AIM : Develop an e-commerce webpage using BootStrap elements.

THEORY :

Bootstrap is a popular open-source **front-end framework** used for developing responsive and mobile-first websites and web applications. It provides a collection of **HTML, CSS, and JavaScript** components and utilities that make it easier to design web pages quickly and consistently.

CODE :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>My Shop</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.cs
s" rel="stylesheet" />
</head>
<body>

  <!-- Navbar -->
  <nav class="navbar bg-primary" data-bs-theme="dark">
        <div class="container">
        <a class="navbar-brand" href="#">ShopPure</a>
        <button class="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target="#navbarNav">
        <span class="navbar-toggler-icon"></span>
        </button>
```

```
        <div class="collapse navbar-collapse justify-content-end"
id="navbarNav">
        <ul class="navbar-nav">
        <li class="nav-item"><a class="nav-link active"
href="#">Home</a></li>
        <li class="nav-item"><a class="nav-link" href="#">Shop</a></li>
        <li class="nav-item"><a class="nav-link" href="#">Cart</a></li>
        </ul>
        </div>
        </div>
  </nav>

  <nav aria-label="breadcrumb" class="bg-light">
        <div class="container py-2">
        <ol class="breadcrumb mb-0">
        <li class="breadcrumb-item"><a href="#">Home</a></li>
        <li class="breadcrumb-item"><a href="#">Shop</a></li>
        <li class="breadcrumb-item active" aria-current="page">Featured
Products</li>
        </ol>
        </div>
  </nav>


  <!-- Hero Banner -->
  <div class="bg-light p-5 text-center">
        <h1>Welcome to ShopPure!</h1>
        <p>Shop pure - support the village industries!</p>
        <a href="#products" class="btn btn-primary">Shop Now</a>
  </div>

  <div class="bg-light p-5">
        <h3>Sign Up for our Newsletter!!!</h3>
        <br>
        <form>
```

```
        <div class="mb-3">
        <label for="exampleInputEmail1" class="form-label">Email
address</label>
        <input type="email" class="form-control" id="exampleInputEmail1"
aria-describedby="emailHelp">
        <div id="emailHelp" class="form-text">We'll never share your email
with anyone else.</div>
        </div>
        <div class="mb-3">
        <label for="exampleInputPassword1"
class="form-label">Password</label>
        <input type="password" class="form-control"
id="exampleInputPassword1">
        </div>
        <div class="mb-3 form-check">
        <input type="checkbox" class="form-check-input"
id="exampleCheck1">
        <label class="form-check-label" for="exampleCheck1">Check me
out</label>
        </div>
        <button type="submit" class="btn btn-primary">Submit</button>
        </form>
  </div>


  <!-- Product Section -->
  <section id="products" class="py-5">
        <div class="container">
        <h2 class="text-center mb-4">Featured Products</h2>
        <div class="row g-4">

        <!-- Jumbotron -->
<div class="p-5 mb-4 bg-light rounded-3 text-center">
        <div class="container py-5">
        <h1 class="display-4 fw-bold">Welcome to MyShop</h1>
```

```
        <p class="fs-5">Discover amazing products and unbeatable
deals.</p>
        <button type="button" class="btn btn-success"
id="goToFirstProduct">Go to First Product</button>
        </div>
  </div>

        <!-- Product 1 -->
        <div>
        <div class="card" id="firstproduct">
        <img src="PLACEHOLDER_URL" class="card-img-top" alt="Product
1" />
        <div class="card-body">
                <h5 class="card-title">Essential Oil</h5>
                <p class="card-text">Rs 1000</p>
                <a href="#" class="btn btn-outline-primary">Add to Cart</a>
        </div>
        </div>
        </div>
        <!-- Product 2 -->
        <div class="col-md-4">
        <div class="card">
        <img src="https://via.placeholder.com/300x200" class="card-img-top"
alt="Product 2" />
        <div class="card-body">
                <h5 class="card-title">Product 2</h5>
                <p class="card-text">$49.99</p>
                <a href="#" class="btn btn-outline-primary">Add to Cart</a>
        </div>
        </div>
        </div>
        <!-- Product 3 -->
        <div class="col-md-4">
        <div class="card">
```

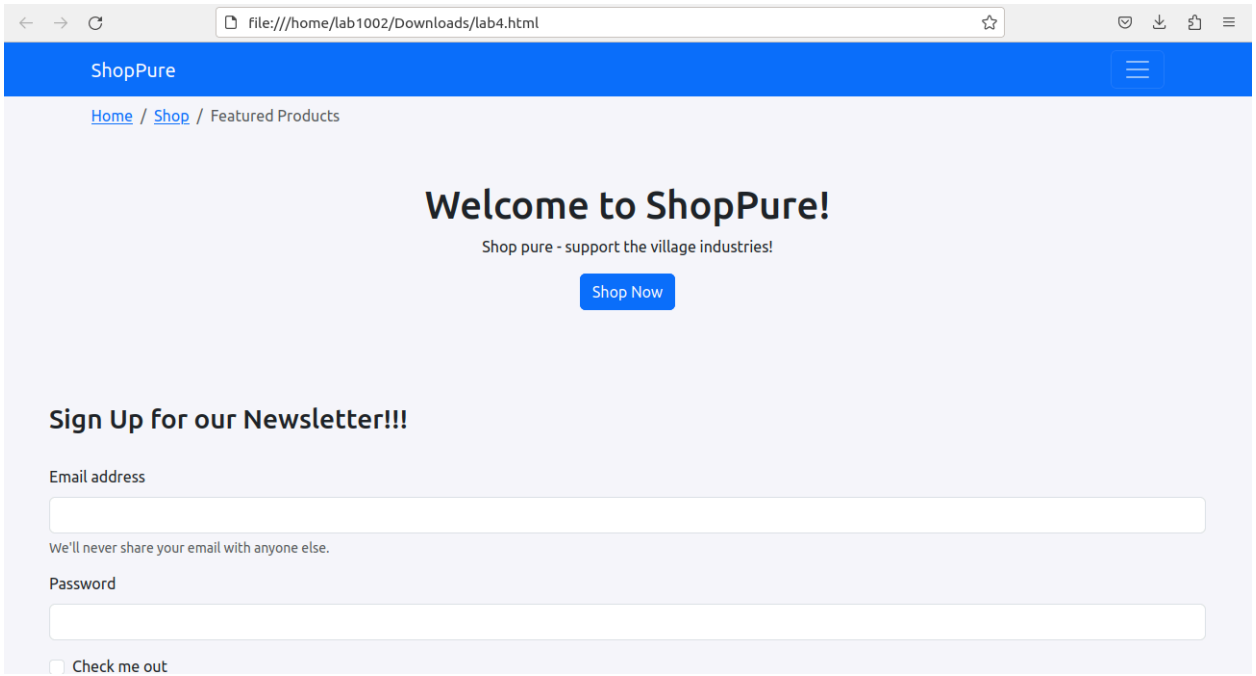
```
        <img src="https://via.placeholder.com/300x200" class="card-img-top"
alt="Product 3" />
        <div class="card-body">
                <h5 class="card-title">Product 3</h5>
                <p class="card-text">$19.99</p>
                <a href="#" class="btn btn-outline-primary">Add to Cart</a>
        </div>
        </div>
        </div>
        <div class="col-md-4">
        <div class="card">
                <img src="https://via.placeholder.com/300x200"
class="card-img-top" alt="Product 3" />
                <div class="card-body">
                <h5 class="card-title">Product 4</h5>
                <p class="card-text">$19.99</p>
                <a href="#" class="btn btn-outline-primary">Add to Cart</a>
                </div>
        </div>
        </div>

        </div>
        </div>
  </section>

  <!-- Footer -->
  <footer class="bg-dark text-white text-center p-3 mt-4">
        <p class="mb-0">&copy; 2025 MyShop. All rights reserved.</p>
  </footer>

  <!-- Bootstrap JS -->
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.
min.js"></script>
```

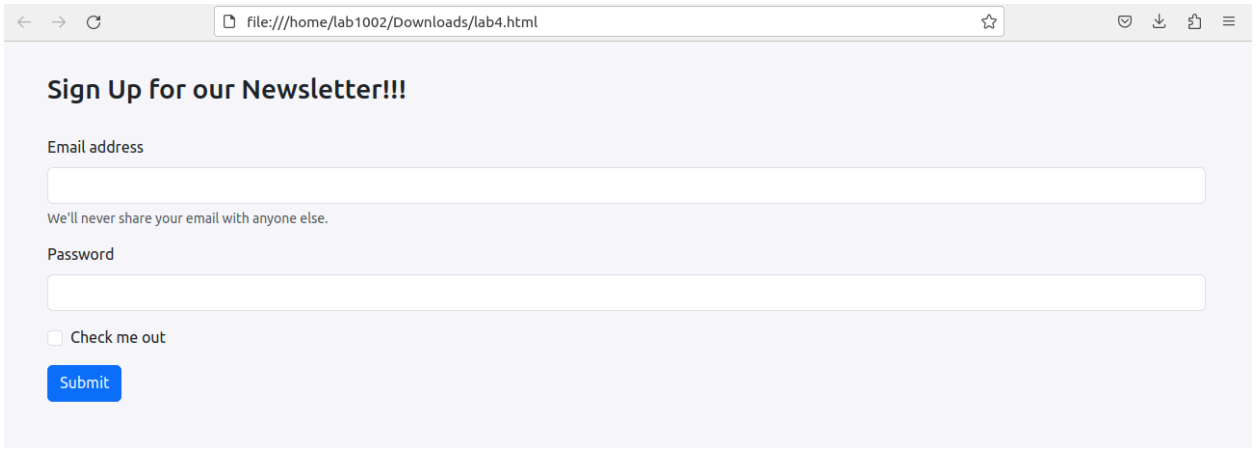
```
<script>
  document.getElementById('goToFirstProduct').addEventListener('click', ()
=> {
        document.getElementById('firstproduct').scrollIntoView({ behavior:
'smooth' });
  });
</script>

</body>
</html>
```

ELEMENTS USED :

1] Navbar
2] Breadcrumbs
3] Grid Layout using cards
4] Form
5] Jumbotron using card
6] Collapsable menu

OUTPUT :





Featured Products

file:///home/lab1002/Downloads/lab4.html

Featured Products

Welcome to MyShop

Discover amazing products and unbeatable deals.

Go to First Product

file:///home/lab1002/Downloads/lab4.html

Essential Oil

Rs 1000

Add to Cart

file:///home/lab1002/Downloads/lab4.html

Essential Oil

Rs 1000

Add to Cart

Product 2

Product 2

$49.99

Add to Cart

Product 3

Product 3

$19.99

Add to Cart

Product 3

Product 4

$19.99

Add to Cart

ASSIGNMENT 5 -

AIM -

Write a Javascript program to implement the following constructs -
Conditional statements
Loops
Functions
Inheritance
Iterator
Generator

THEORY -

1) Conditional Statements

- **Definition**: Conditional statements allow you to execute different blocks of code depending on whether a condition evaluates to `true` or `false`.
- **Types**:
    - `if…else`: Checks a condition and executes code accordingly.
    - `switch`: Useful when comparing one value against multiple possible cases.

Example:

```
if (age >= 18) console.log("Adult");
else console.log("Minor");
```

-

---

2) Loops

- **Definition**: Loops allow repeated execution of a block of code until a condition is met.
- **Types**:
    - `for`: Runs a block for a specific number of iterations.
    - `while`: Runs as long as a condition is true.
    - `for…of`: Iterates directly over iterable objects (like arrays, strings).

Example:

```
for (let i = 0; i < 3; i++) console.log(i);
```

-

---

3) Functions

- **Definition**: A function is a block of reusable code designed to perform a specific task.
- **Types**:
    - Function declaration: `function add(a,b){ return a+b; }`
    - Arrow function: `const square = x => x*x;`
    - Higher-order function: A function that takes another function as input.
- **Why useful**: Helps break code into smaller, reusable, modular parts.

---

4) Inheritance (Object-Oriented Programming)

- **Definition**: Inheritance allows one class (child) to acquire properties and methods of another class (parent).

- **How in JS**: Using `class` and `extends` keywords. `super()` is used to call the parent constructor.

Example:

```
class Dog extends Animal {
  speak() { console.log("Barks"); }
}
```

-
- **Why useful**: Promotes code reusability and avoids duplication.

---

5) Iterator

- **Definition**: An iterator is an object that defines a sequence and can be used to step through it one element at a time.

- **In JS**: Objects become iterable when they implement the `Symbol.iterator` method, which returns an object with a `next()` method.

Example:

```
const it = [1,2,3][Symbol.iterator]();
console.log(it.next()); // {value:1, done:false}
```

-
- **Why useful**: Customizes how data structures can be looped with `for…of`.

---

6) Generator

- **Definition**: Generators are special functions that can pause execution and resume later.
- **Syntax**: Defined using `function*` and `yield` keyword.
- **Behavior**: Each call to `.next()` executes until the next `yield`.

Example:

```
function* gen() {
  yield 1;
  yield 2;
}
```

-
- **Why useful**: Helps manage infinite sequences, asynchronous operations, and custom iterators.

CODE -

```
/**********************
 * 1) CONDITIONAL STATEMENTS
 **********************/
const age = 19;

if (age >= 18) {
  console.log("You are an adult.");
} else {
  console.log("You are a minor.");
}

// Switch example
const day = "Sun";
switch (day) {
  case "Sat":
  case "Sun":
    console.log("Weekend!");
    break;
```

```
  default:
    console.log("Weekday!");
    break;
}

/************************
 * 2) LOOPS
 ************************/
// Classic for loop
for (let i = 1; i <= 3; i++) {
  console.log("for loop i:", i);
}

// while loop
let count = 3;
while (count > 0) {
  console.log("while loop count:", count);
  count--;
}

// for...of over an array
const nums = [10, 20, 30];
for (const n of nums) {
  console.log("for...of:", n);
}

/************************
 * 3) FUNCTIONS
 ************************/
// Function declaration
function add(a, b) {
  return a + b;
}
console.log("add(2, 3):", add(2, 3));

// Arrow function
const square = (x) => x * x;
console.log("square(5):", square(5));

// Higher-order function (takes a function)
```

```
function applyToArray(arr, fn) {
  const out = [];
  for (const item of arr) out.push(fn(item));
  return out;
}
console.log("applyToArray(nums, square):", applyToArray(nums, square));

/************************
 * 4) INHERITANCE (ES6 Classes)
 ************************/
class Animal {
  constructor(name) {
    this.name = name;
  }
  speak() {
    console.log(`${this.name} makes a noise.`);
  }
}

class Dog extends Animal {
  constructor(name, breed) {
    super(name); // call parent constructor
    this.breed = breed;
  }
  speak() {
    // method override
    console.log(`${this.name} barks. Breed: ${this.breed}`);
  }
}

const a = new Animal("Creature");
a.speak();

const d = new Dog("Bruno", "Labrador");
d.speak();

/************************
 * 5) ITERATOR (Custom Iterable)
 * Create an iterable range: for (const x of range(5, 8)) -> 5,6,7,8
 ************************/
```

```
function range(start, end) {
  return {
    start,
    end,
    [Symbol.iterator]() {
      let current = this.start;
      const last = this.end;
      return {
        next() {
          if (current <= last) {
            return { value: current++, done: false };
          }
          return { value: undefined, done: true };
        },
      };
    },
  };
}

for (const x of range(5, 8)) {
  console.log("iterable range value:", x);
}

/**********************
 * 6) GENERATOR
 * A generator that yields the first n Fibonacci numbers
 **********************/
function* fibonacci(n) {
  let a = 0,
    b = 1,
    i = 0;
  while (i < n) {
    yield a;
    [a, b] = [b, a + b];
    i++;
  }
}

for (const f of fibonacci(7)) {
  console.log("fibonacci:", f);
```

```
}

/************************
 * Bonus: Using a generator to create an infinite ID stream (consumed safely)
 ************************/
function* idGenerator(start = 1) {
  let id = start;
  while (true) {
    yield id++;
  }
}
const ids = idGenerator(100);
console.log("id1:", ids.next().value);
console.log("id2:", ids.next().value);
console.log("id3:", ids.next().value);
```

OUTPUT -

```
You are an adult.
Weekend!
for loop i: 1
for loop i: 2
for loop i: 3
while loop count: 3
while loop count: 2
while loop count: 1
for...of: 10
for...of: 20
for...of: 30
add(2, 3): 5
square(5): 25
applyToArray(nums, square): [ 100, 400, 900 ]
Creature makes a noise.
Bruno barks. Breed: Labrador
iterable range value: 5
iterable range value: 6
iterable range value: 7
iterable range value: 8
fibonacci: 0
fibonacci: 1
```

fibonacci: 1
fibonacci: 2
fibonacci: 3
fibonacci: 5
fibonacci: 8
id1: 100
id2: 101
id3: 102

```
You are an adult.
Weekend!
for loop i: 1
for loop i: 2
for loop i: 3
while loop count: 3
while loop count: 2
while loop count: 1
for...of: 10
for...of: 20
for...of: 30
add(2, 3): 5
square(5): 25
applyToArray(nums, square): [ 100, 400, 900 ]
Creature makes a noise.
Bruno barks. Breed: Labrador
iterable range value: 5
iterable range value: 6
iterable range value: 7
iterable range value: 8
fibonacci: 0
fibonacci: 1
fibonacci: 1
fibonacci: 2
fibonacci: 3
fibonacci: 5
fibonacci: 8
id1: 100
id2: 101
id3: 102
```

ASSIGNMENT 6 -

AIM : Write a Javascript program to implement -
- Arrow Function
- DOM Manipulation
- CSS Manipulation

THEORY :

1) Arrow Functions

Definition: A shorter syntax to write functions, introduced in ES6.

Syntax:

```
const add = (a, b) => a + b;
```

Key Features:

. More concise than function keyword.

. Do not have their own this, arguments, or super (inherit them from the surrounding scope).

. Often used in callbacks, event handlers, and array methods.

2) DOM Manipulation

DOM (Document Object Model): A tree-like structure representing all elements of a webpage.

DOM Manipulation: Using JavaScript to dynamically add, modify, or remove elements and content.

Common Methods:

getElementById("id") → select an element by ID.

textContent or innerHTML → change text or HTML inside an element.

appendChild() or removeChild() → add/remove elements.

Example:

```
document.getElementById("msg").textContent = "Hello World!";
```

3) CSS Manipulation

Definition: Changing the style of HTML elements using JavaScript.

Ways to Manipulate CSS:

Inline styles:

```
element.style.color = "red";
```

Class manipulation:

```
element.classList.add("highlight");
element.classList.remove("highlight");
element.classList.toggle("highlight");
```

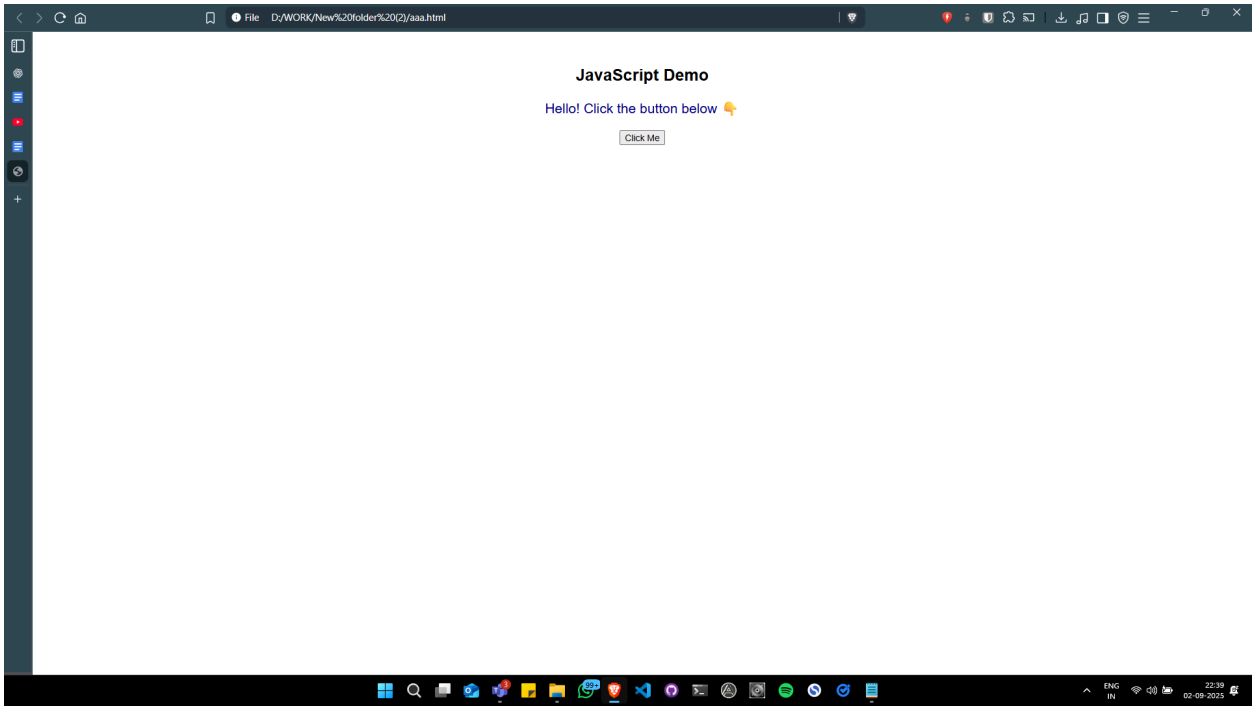
Why useful: Enables dynamic styling like hover effects, animations, dark mode, etc.

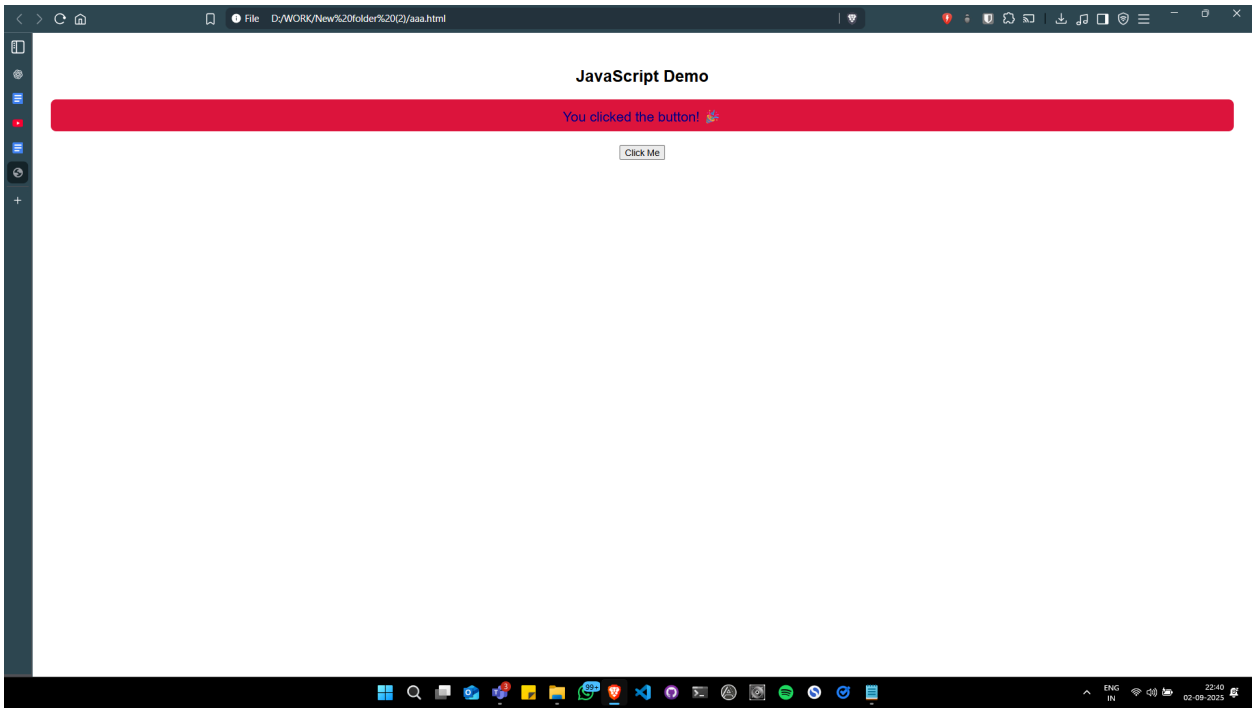
CODE :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>JS Demo</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin-top: 50px;
    }
    #message {
      font-size: 20px;
      margin: 20px;
      color: darkblue;
    }
    .highlight {
      color: white;
      background-color: crimson;
      padding: 10px;
      border-radius: 8px;
    }
  </style>
</head>
<body>
  <h2>JavaScript Demo</h2>
  <p id="message">Hello! Click the button below 👇</p>
  <button id="btn">Click Me</button>
```

```
  <script>
    // 1) Arrow Function
    const changeMessage = () => {
      // 2) DOM Manipulation: change content of <p>
      const msg = document.getElementById("message");
      msg.textContent = "You clicked the button! 🎉";

      // 3) CSS Manipulation: toggle a CSS class
      msg.classList.toggle("highlight");
    };

    // Attach event listener
    document.getElementById("btn").addEventListener("click",
changeMessage);
  </script>
</body>
</html>
```

OUTPUT :


File D:/WORK/New%20folder%20(2)/aaa.html
JavaScript Demo
Hello! Click the button below 👇
Click Me


File D:/WORK/New%20folder%20(2)/aaa.html
JavaScript Demo
You clicked the button! 🎉
Click Me

ASSIGNMENT 7 -

AIM : Write a Javascript program to implement form validation.

THEORY :

Form validation is the process of checking whether the data entered into a form meets the required criteria. It helps prevent errors, ensures data integrity, and improves user experience by providing immediate feedback.

Types of Form Validation

1. Client-Side Validation

- Done in the browser using JavaScript or HTML5 attributes.
- Provides instant feedback without needing to contact the server.
- Examples:
    - Checking if a field is empty.
    - Ensuring an email address follows the correct format.
    - Restricting input to numbers only.

2. Server-Side Validation

- Performed after the form is submitted, on the server.
- Essential for security, as client-side validation can be bypassed.
- Examples:
    - Checking if a username already exists in the database.
    - Validating authentication credentials.

Common Validation Rules

Field Type	Validation Rule Example
Text	Must not be empty, no special characters allowed

Email	Must follow `user@example.com` format
Phone Number	Must be exactly 10 digits
Age	Must be a positive number, within a valid range
Password	Minimum length, includes letters and numbers
Dropdown	Must select a valid option

Why Is Validation Important?

- **Data Integrity**: Prevents invalid or incomplete data from entering the system.
- **Security**: Protects against malicious input like SQL injection or XSS.
- **User Experience**: Helps users correct mistakes before submission.
- **Efficiency**: Reduces server load by catching errors early.

CODE :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Hospital Patient Form</title>

  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 30px;
      background-color: #f4f4f4;
```

```
    }
    form {
      background: #fff;
      padding: 20px;
      border-radius: 8px;
      max-width: 500px;
      margin: auto;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
    }
    label {
      display: block;
      margin-top: 15px;
    }
    input, select {
      width: 100%;
      padding: 8px;
      margin-top: 5px;
      box-sizing: border-box;
    }
    button {
      margin-top: 20px;
      padding: 10px;
      width: 100%;
      background-color: #007bff;
      color: white;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }
    .error {
      color: red;
      font-size: 0.9em;
    }
  </style>
</head>
```

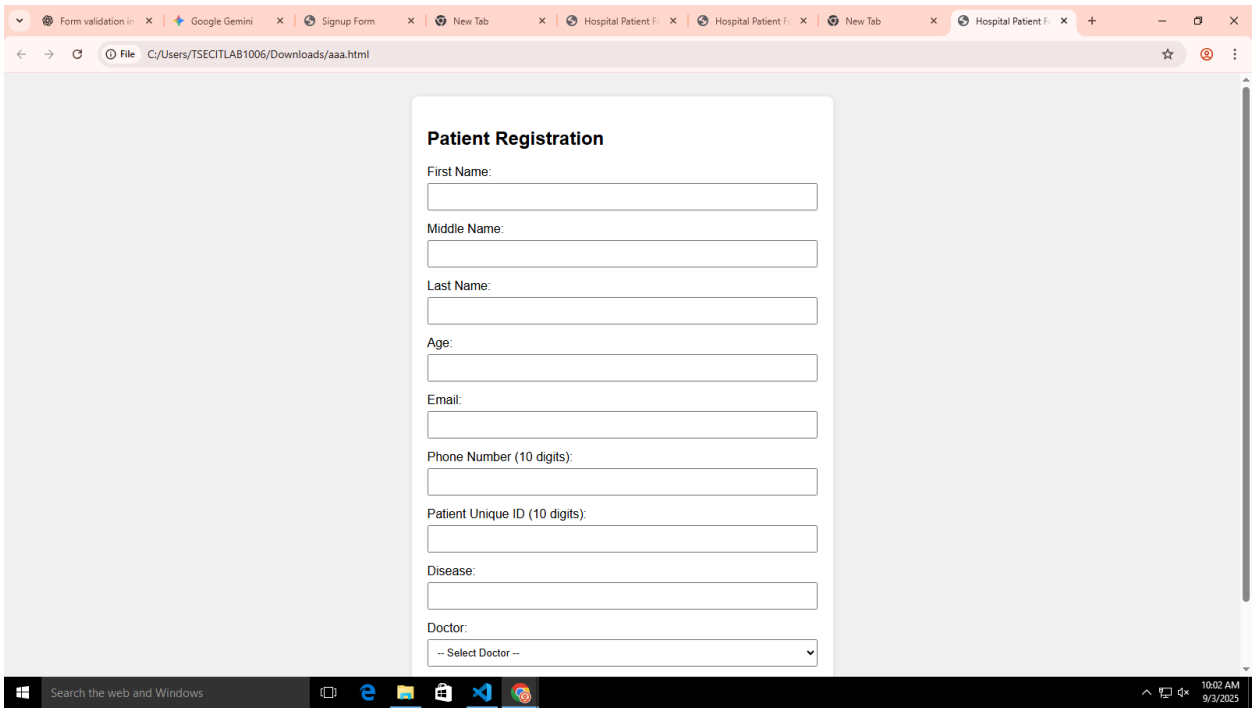
```
<body>

<form id="patientForm">
  <h2>Patient Registration</h2>

  <label>First Name:</label>
  <input type="text" id="firstName" required>

  <label>Middle Name:</label>
  <input type="text" id="middleName" required>

  <label>Last Name:</label>
  <input type="text" id="lastName" required>

  <label>Age:</label>
  <input type="number" id="age" min="0" required>

  <label>Email:</label>
  <input type="email" id="email" required>

  <label>Phone Number (10 digits):</label>
  <input type="text" id="phone" maxlength="10" required>

  <label>Patient Unique ID (10 digits):</label>
  <input type="text" id="patientId" maxlength="10" required>

  <label>Disease:</label>
  <input type="text" id="disease" required>

  <label>Doctor:</label>
  <select id="doctor" required>
    <option value="">-- Select Doctor --</option>
    <option value="Dr. Ahuja - Cardiologist">Dr. Ahuja -
Cardiologist</option>
```

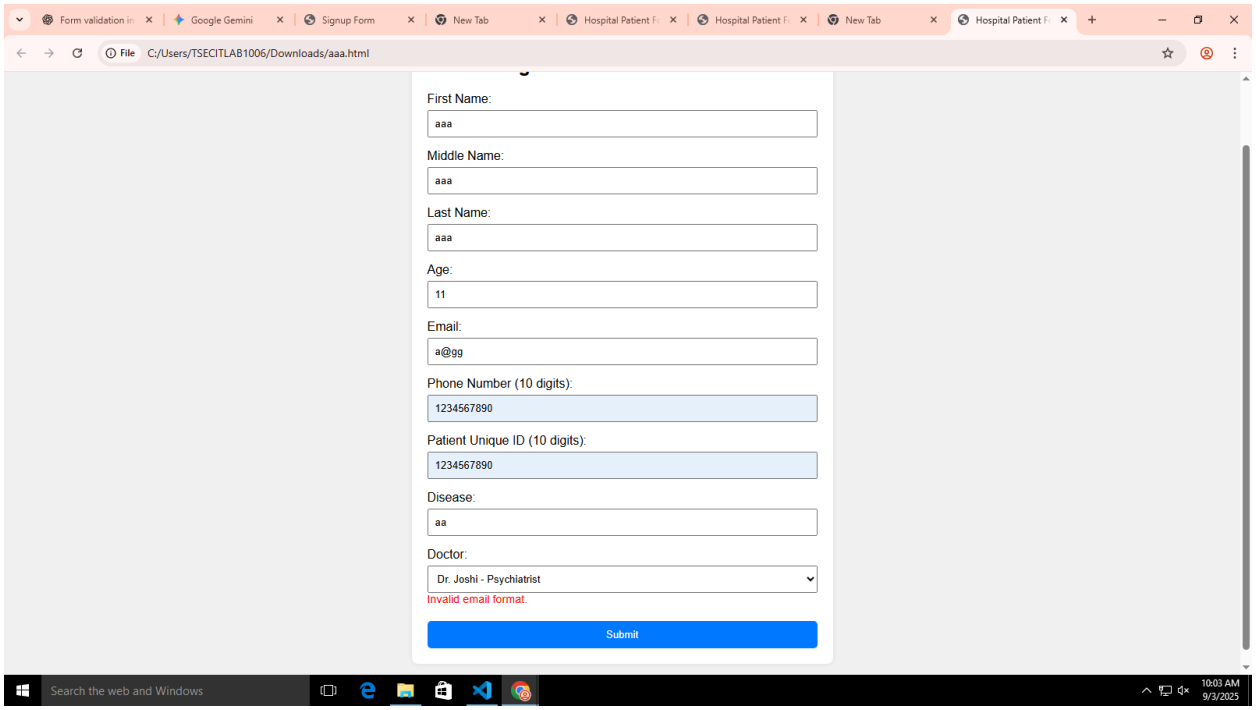
```
    <option value="Dr. Mehta - Neurologist">Dr. Mehta - Neurologist</option>
    <option value="Dr. Kapoor - Orthopedic">Dr. Kapoor - Orthopedic</option>
    <option value="Dr. Sharma - Dermatologist">Dr. Sharma - Dermatologist</option>
    <option value="Dr. Reddy - Pediatrician">Dr. Reddy - Pediatrician</option>
    <option value="Dr. Joshi - Psychiatrist">Dr. Joshi - Psychiatrist</option>
    <option value="Dr. Verma - Oncologist">Dr. Verma - Oncologist</option>
    <option value="Dr. Singh - ENT Specialist">Dr. Singh - ENT Specialist</option>
    <option value="Dr. Desai - Gastroenterologist">Dr. Desai - Gastroenterologist</option>
    <option value="Dr. Khan - Pulmonologist">Dr. Khan - Pulmonologist</option>
  </select>

  <div class="error" id="errorMsg"></div>

  <button type="submit">Submit</button>
</form>

<script>
  document.getElementById("patientForm").addEventListener("submit", function(e) {
    e.preventDefault();
    const errorMsg = document.getElementById("errorMsg");
    errorMsg.textContent = "";

    const fields = [
      "firstName", "middleName", "lastName", "age", "email",
      "phone", "patientId", "disease", "doctor"
    ];
```

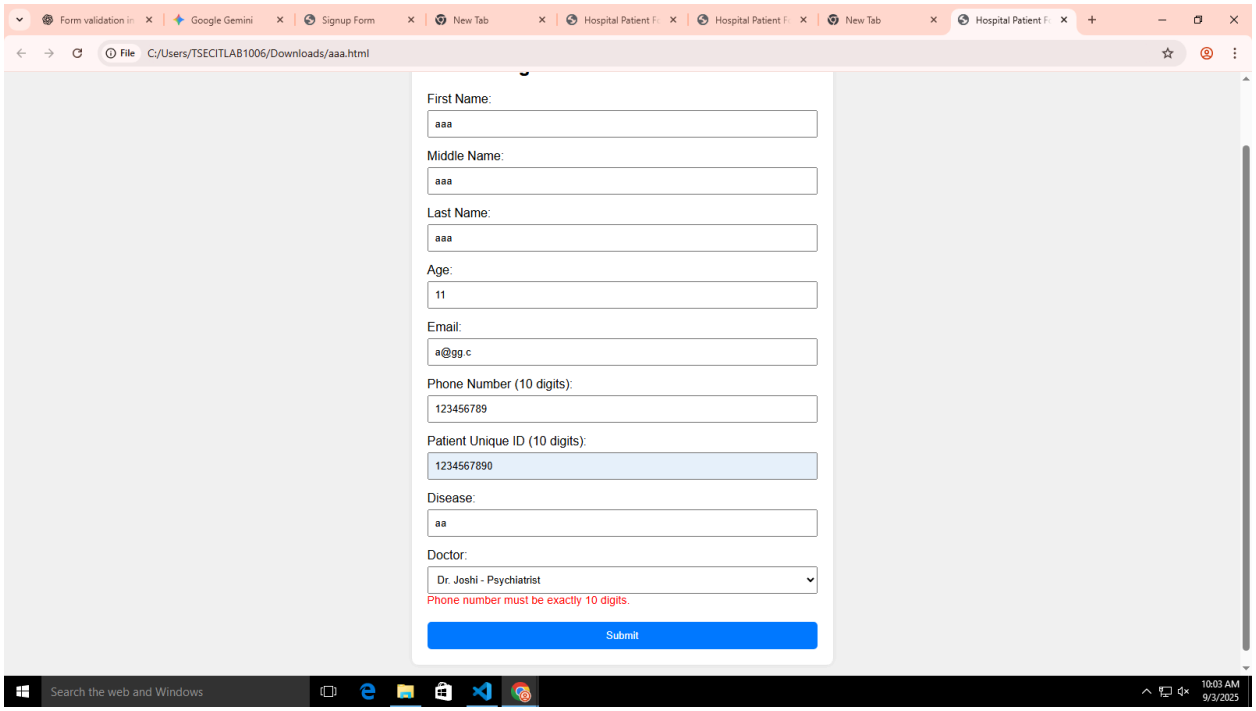
```
    for (let field of fields) {
      const value = document.getElementById(field).value.trim();
      if (!value) {
        errorMsg.textContent = "All fields must be filled.";
        return;
      }
    }

    const phone = document.getElementById("phone").value.trim();
    const patientId = document.getElementById("patientId").value.trim();
    const email = document.getElementById("email").value.trim();

    const phoneRegex = /^\d{10}$/;
    const idRegex = /^\d{10}$/;
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    if (!phoneRegex.test(phone)) {
      errorMsg.textContent = "Phone number must be exactly 10 digits.";
      return;
    }

    if (!idRegex.test(patientId)) {
      errorMsg.textContent = "Patient ID must be exactly 10 digits.";
      return;
    }

    if (!emailRegex.test(email)) {
      errorMsg.textContent = "Invalid email format.";
      return;
    }

    alert("Form submitted successfully!");
    // You can now send the data to a server or process it further
  });
</script>
```

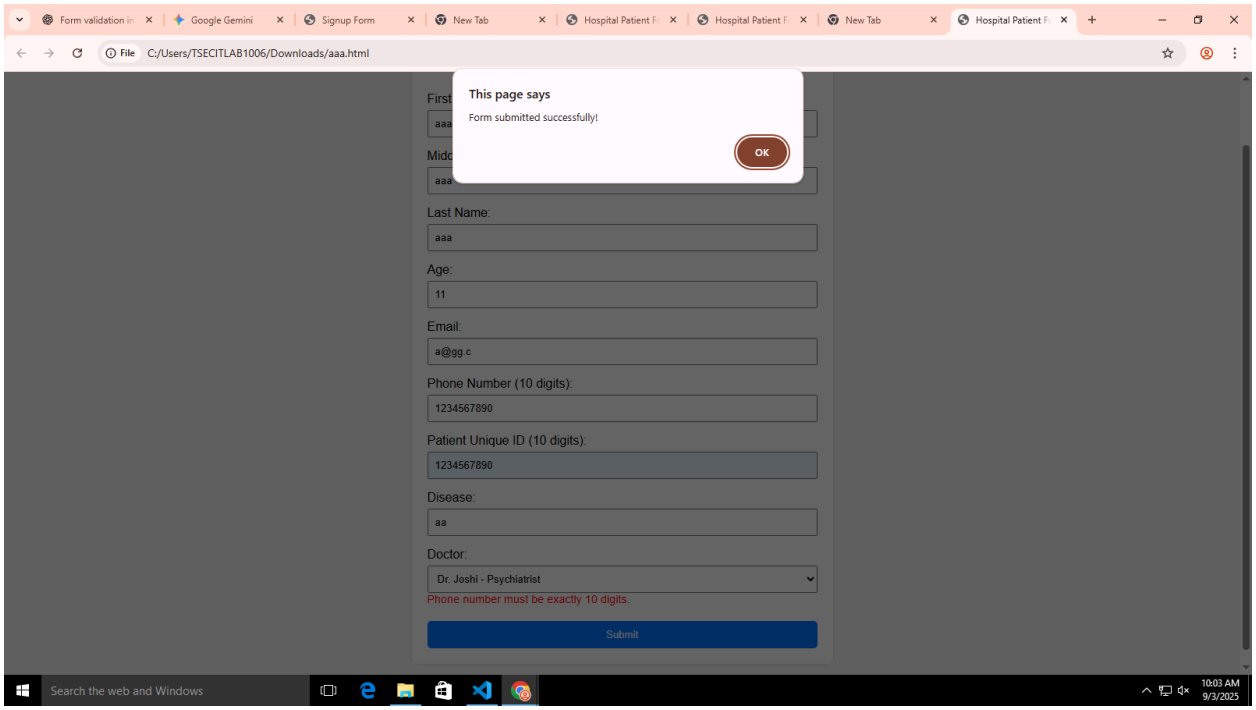
```
</body>
</html>
```

OUTPUT :




Form validation in
Google Gemini
Signup Form
New Tab
Hospital Patient F
Hospital Patient F
New Tab
Hospital Patient F
File
C:/Users/TSECITLAB1006/Downloads/aaa.html
First Name:
aaa
Middle Name:
aaa
Last Name:
aaa
Age:
11
Email:
a@gg
Phone Number (10 digits):
1234567890
Patient Unique ID (10 digits):
1234567890
Disease:
aa
Doctor:
Dr. Joshi - Psychiatrist
Invalid email format.
Submit
Search the web and Windows
10:03 AM
9/3/2025


Form validation in
Google Gemini
Signup Form
New Tab
Hospital Patient F
Hospital Patient F
New Tab
Hospital Patient F
File
C:/Users/TSECITLAB1006/Downloads/aaa.html
First Name:
aaa
Middle Name:
aaa
Last Name:
aaa
Age:
11
Email:
a@gg.c
Phone Number (10 digits):
123456789
Patient Unique ID (10 digits):
1234567890
Disease:
aa
Doctor:
Dr. Joshi - Psychiatrist
Phone number must be exactly 10 digits.
Submit
Search the web and Windows
10:03 AM
9/3/2025


Form validation in
Google Gemini
Signup Form
New Tab
Hospital Patient F
Hospital Patient F
New Tab
Hospital Patient F
File
C:/Users/TSECITLAB1006/Downloads/aaa.html
This page says
Form submitted successfully!
OK
First
aaa
Midc
aaa
Last Name:
aaa
Age:
11
Email:
a@gg.c
Phone Number (10 digits):
1234567890
Patient Unique ID (10 digits):
1234567890
Disease:
aa
Doctor:
Dr. Joshi - Psychiatrist
Phone number must be exactly 10 digits.
Submit
Search the web and Windows
10:03 AM
9/3/2025

ASSIGNMENT 8 –

AIM: Write a program to implement the concept of React Hooks (use States, use effect).

THEORY: React Hooks are functions that let you use React features like state and lifecycle methods in functional components. Two commonly used hooks are `useState` and `useEffect`.

- `useState` allows functional components to manage state. It returns a state variable and a function to update it. This makes it easy to track and modify dynamic data, such as a counter value.
- `useEffect` enables performing side effects in functional components, like data fetching, subscriptions, or manipulating the DOM. It runs after every render by default but can be controlled to run only when specific dependencies change or once on component mount.

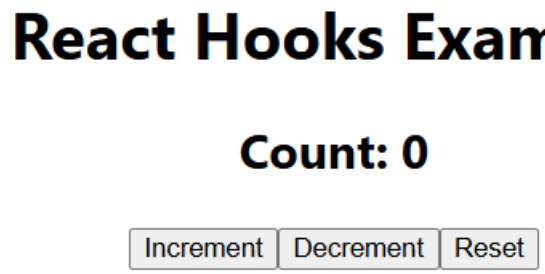
In this program, `useState` manages the counter value, enabling increment, decrement, and reset functionality. The `useEffect` hook is used twice: first, to load a saved counter value from `localStorage` when the component mounts, ensuring persistence across page reloads; and second, to save the current counter value to `localStorage` whenever it changes. This demonstrates effective use of React Hooks to maintain stateful behavior with side effects, improving user experience by persisting data locally.

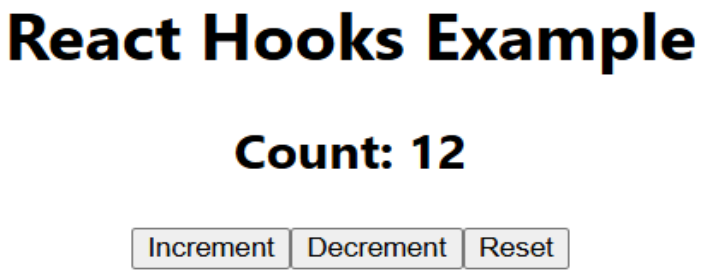
CODE:

```
import React, { useState, useEffect } from "react";

function App() {
  // useState Hook
  const [count, setCount] = useState(0);

  // useEffect Hook
  useEffect(() => {
    document.title = `You clicked ${count} times`; // updates tab title
    console.log("useEffect executed");

    return () => {
      console.log("Cleanup executed before next effect");
    };
  }, [count]); // runs only when count changes

  return (
    <div style={{ textAlign: "center", marginTop: "50px" }}>
```

```
      <h1>React Hooks Example</h1>
      <h2>Count: {count}</h2>
      <button onClick={() => setCount(count + 1)}>Increment</button>
      <button onClick={() => setCount(count - 1)}>Decrement</button>
      <button onClick={() => setCount(0)}>Reset</button>
    </div>
  );
}

export default App;
```

OUTPUT:





CONCLUSION: This program demonstrates how React Hooks (useState and useEffect) manage state and side effects, enabling persistent and interactive UI behavior efficiently.

LO mapping: LO5

ASSIGNMENT 9 -

AIM: Write the program to implement the concept of the Asynchronous Programming using promises

THEORY:

Asynchronous Programming

Asynchronous programming allows us to perform tasks (like data fetching, file reading, etc.) without blocking the main thread, meaning the application can continue executing while waiting for the task to complete. In web development, this is essential to avoid freezing the user interface while performing operations that may take time (e.g., fetching data from a server).

What is a Promise?

1. A Promise is a JavaScript object that represents the eventual completion (or failure) of an asynchronous operation and its resulting value.
2. It acts as a placeholder for a value that will be available in the future.

Promise States

➔ Pending: The operation is still ongoing.
➔ Resolved: The operation was successful.
➔ Rejected: The operation failed.

Using Promises in React:

React uses useState and useEffect to manage state and side effects like fetching data:

● useState stores data (like loading state, fetched data, or errors).
● useEffect handles side effects, like fetching data when the component mounts.

How it Works in React:

1. A Promise simulates fetching data.
2. useEffect triggers the asynchronous operation when the component mounts.
3. useState manages loading, data, and error states.
4. The component renders based on the fetched data or error.

Creating and Using Promises

You create a Promise using the new Promise() constructor, which takes a function with two arguments: resolve and reject.

SYNTAX :-

```
const myPromise = new Promise((resolve, reject) => {
// asynchronous operation if
(success) {
resolve(result); // fulfill the promise
} else {
reject(error); // reject the promise
}
```

});
Key Benefits of Promises
● Improved readability over nested callbacks.
● Error handling using .catch() that catches errors anywhere in the promise chain.
● Chaining multiple async operations easily with .then() returning new promises.
● They represent the future result of an operation clearly.
APIs (Application Programming Interfaces) allow apps to communicate with external services to fetch or send data. In this project, we use the Fetch API to asynchronously request data from the public joke API at https://official-joke-api.appspot.com/random_joke. The Fetch API returns a promise that resolves when the response is received, allowing the app to handle data or errors smoothly without blocking the user interface.
Code Structure:
/ joke-app
│
├── /src
│ ├── App.js
│ ├── index.js # Entry point for the React app
└── /public
└── index.html # HTML template for the app

CODE:
/src/App.js
return (

<h1 style={styles.title}>Random Joke Generator</h1>
<button onClick={fetchJoke} style={styles.button}> Get a Joke
</button>
{loading && Loading... }
{error && {error} }
{joke && (

{joke.setup}
{joke.punchline}

)}

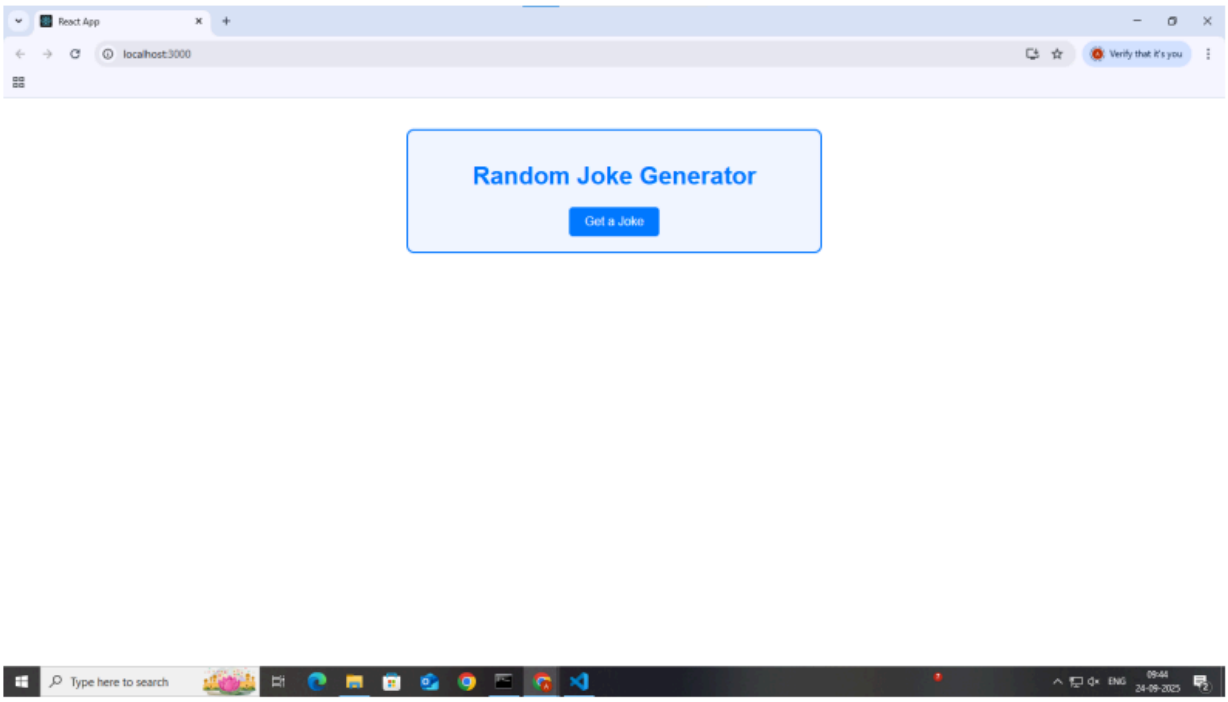
);
}
const styles = { container: {
maxWidth: 500,
margin: '40px auto', textAlign: 'center',

```
fontFamily: 'Arial, sans-serif', padding: 20,
border: '2px solid #007bff', borderRadius: 10,
backgroundColor: '#f0f8ff',
},
title: {
color: '#007bff',
},
button: {
padding: '10px 20px', fontSize: 16,
borderRadius: 5, border:
'none',
backgroundColor: '#007bff', color: 'white',
cursor: 'pointer',
/src/index.js
/public/index.html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
<meta name="viewport" content="width=device-width, initial-scale=1"
/>
<meta name="theme-color" content="#000000" />
<meta
name="description"
content="Web site created using create-react-app"
/>
<link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
<!--
manifest.json provides metadata used when your web app is
installed on a
user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
-->
<link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
<!--
Notice the use of %PUBLIC_URL% in the tags above.
It will be replaced with the URL of the `public` folder during the build.
Only files inside the `public` folder can be referenced from the
HTML.
Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
work correctly both with client-side routing and a non-root public URL.
Learn how to configure a non-root public URL by running `npm run build`.
-->
```

<title>React App</title>
</head>
<body>
<noscript>You need to enable JavaScript to run this app.</noscript>

<!--
This HTML file is a template.

OUTPUT:



CONCLUSION:

Promises enable efficient asynchronous programming by handling future results cleanly and preventing blocking. They improve code readability and simplify managing async tasks in JavaScript.
LO mapping: LO5

ASSIGNMENT 10 -

Aim: Write a program in NodeJS to:
(a) Create a file
(b) Read the data from the file
(c) Write the data to a file
(d) Rename a file
(e) Delete a file

Theory: Node.js provides a built-in module called `fs` (File System) to perform file-related operations such as creating, reading, writing, renaming, and deleting files. These operations are essential for handling persistent data storage in server-side applications. File handling is one of the most important features of any programming language, as it allows data to be stored permanently for future use. In Node.js, file handling is achieved using the File System (fs) module, which provides various functions to interact with the file system in an efficient and asynchronous way.
The major file operations supported in Node.js include:
1. Creating a File
A new file can be created, and initial data can be written into it. If the file already exists, its content may be replaced.

2. Reading from a File
The contents of a file can be accessed and displayed. This is useful for retrieving stored data and processing it further.

3. Writing/Appending to a File
Data can be added to an existing file. Instead of replacing old content, new data is appended at the end, thus preserving earlier information.

4. Renaming a File
A file can be given a new name without affecting its contents. This is helpful for organizing or updating file names.

5. Deleting a File
Unnecessary or outdated files can be removed permanently from the storage system.

Node.js performs these file operations in an asynchronous, non-blocking manner, meaning the program continues execution while the file operation is processed in the background. This makes it highly efficient for server-side applications where multiple tasks must run simultaneously.

Code:

```
const fs = require('fs');
function createFile(filename, data) {
 fs.writeFile(filename, data, (err) => {
 if (err) {
 console.error('Error creating file:', err);
 } else {
 console.log(`File '${filename}' created successfully.`);
 }
 });
}
function readFile(filename) {
 fs.readFile(filename, 'utf8', (err, data) => {
 if (err) {
 console.error('Error reading file:', err);
 } else {
 console.log(`Content of '${filename}':\n${data}`);
 }
 });
}
function writeFile(filename, data) {
 fs.appendFile(filename, data, (err) => {
 if (err) {
 console.error('Error writing to file:', err);
 } else {
 console.log(`Data appended to '${filename}' successfully.`);
 }
 });
}
function renameFile(oldName, newName) {
 fs.rename(oldName, newName, (err) => {
 if (err) {
 console.error('Error renaming file:', err);
 } else {
 console.log(`File renamed from '${oldName}' to '${newName}'.`);
 }
 });
}
function deleteFile(filename) {
 fs.unlink(filename, (err) => {
 if (err) {
 console.error('Error deleting file:', err);
 } else {
 console.log(`File '${filename}' deleted successfully.`);
```

```
 }
 });
}
// createFile('sample.txt', 'This is the initial content.\n');
// readFile('sample.txt');
// writeFile('sample.txt', 'Adding more data to the file.\n');
// renameFile('sample.txt', 'renamed_sample.txt');
// deleteFile('renamed_sample.txt');
```

Output:
1) Create:
2) Read:
3)Append:
Dakshika Sidhwani
T22-124
3) Rename:
4) Delete:
Conclusion: This program demonstrates essential Node.js file system operations, enabling efficient file creation, reading, writing, renaming, and deletion for managing persistent data effectively.

LO Mapping: LO 6

IP

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CLASS ASSIGNMENT - 1

ADITYA SHARMA 7-22 IT 116

Q- What is JSON?

JSON (JavaScript Object Notation) is a lightweight, text-based format for storing and exchanging data between systems, especially between a server and client.

JSON stores data as key-value pairs and arrays. It is readable and easy for machines to parse.

Eg-

```
{
  "name" : "Aditya",
  "age" : "21"
}
```

Q- Compare XML and HTML

	XML	HTML
Acronym :	Extensible Markup Language	HyperText Markup Language.
Tags :	User-defined/custom tags allowed.	Predefined, fixed set of tags.
Structure :	Strict - must be well formed and follow rules.	Flexible - some tags can be unclosed or improperly nested.

	XML	HTML
Data v/s Display	Focuses on describing the data	Focuses on ho the data is displayed
Case Sensitivity	Case-sensitive	Not Case-Sensitive
Parsing	Designed for easy data parsing by programs.	Designed for render by browsers.
Use in Web.	Often used with API's, configuration files and data exchange	Used to create web page layouts and user interfaces.

Q- What is DNS?

DNS: Domain Name System is a distributed, hierarchical naming system that translated human-readable domain names into numerical IP addresses (eg 93.184.216.34) that computers use to locate and communicate with each other over the internet.

DNS acts like the phonebook of the internet, enabling users to access websites and online services without memorizing complex IP addresses. When a user enters a domain name into a browser, the DNS resolver queries multiple DNS servers to

find the matching IP address

Key components of a DNS -

- Domain Names : Human - friendly names for websites
- IP - Addresses : Numeric addresses used by computers to identify each other
- DNS - Resolver : A client-side service that initiates DNS queries.
- DNS - Servers : Store and provide domain name to IP-mappings.

Advantages -

- Simplifies access to websites by using names instead of numbers.
- Distributes data across multiple servers for speed and reliability.
- Easily updates domain-to-IP mappings without affecting users.

Q- What is DOM?

DOM: Document Object Model is a programming interface for HTML, XML and SVG documents. It represents the page as a tree-like structure where each element, attribute and piece of text is a node.
Through the DOM, languages like JavaScript can access and manipulate the structure, style and content of a webpage

Q- Differentiate b/w ES5 and ES6 -

	ES5	ES6-
Definition	Fifth edition of ECMA script	Sixth edition of ECMA sc
Release	Introduced in 2009	Introduced in 2015
Support	Supports primitive datatypes ex- string, etc.	Some additions to JS datatypes.
Defining Variables	Could only define the variables by using var keyword.	Two new ways to define → let and const.
Time comming	More time comming	less time comming.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

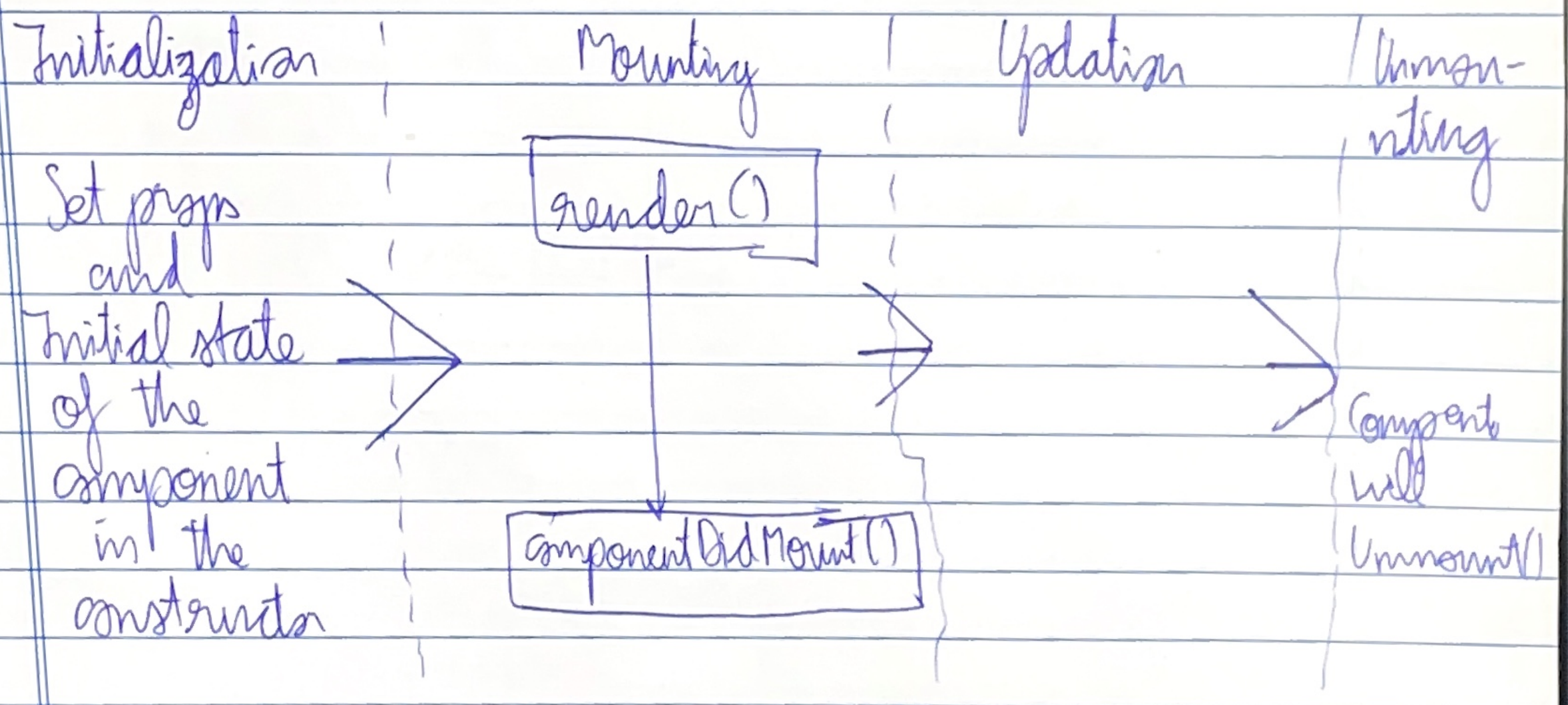
Q- Explain React component lifecycle -

The lifecycle of a react component can be divided into three main phases.

1] Mounting → Creation of components into DOM.

2] Updating → Occurs when a component is re-rendered due to changes in props or state!

3] Unmounting → Involves removal of component from DOM.



ADITYA SHARMA
T22 116

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

IP ASSIGNMENT (LO4, LO5, LO6)

CLASS ASSIGNMENT - 2

1] What is React Ref? When to use and when not to use Refs?

A- 1. A Ref (Short for Reference) in React is a way to directly access & manipulate DOM elements or a react component instance.

2. They are like "reference pointer" to a DOM node.

3. Normally React encourages a declarative style but sometimes we need to iteratively work with DOM and that's where Ref helps.

4. In functional component use `useRef()` hook.

5. Eg-

```
import {useRef} from "react";
function InputFocus(){
    const inputRef = useRef(null);
    const handleFocus = () => {
        inputRef.current.focus(); // Directly accesses DOM input
    };
    return (
        <div>
        <input ref={inputRef} type="text"
            placeholder="Type here..." />
        <button onClick={handleFocus}>
            Focus Input </button>
        </div>
    ); }
```

2] Compare MVC v/s Flux v/s Redux

ASPECT	MVC	FLUX	REDUX
Architecture	Triangular (Model, Controller ↔ View ↔, 2-way data binding)	Unidirectional Action → Dispatcher → View ← Store ←	Strict uni-directional cycle Action → Reducer → Store → View.
Data Flow	2-way	1-way	1-way
State Management	Distributed across models	Multiple stores	Single store (Single source of truth)
Dispatcher	Not required	Central dispatcher	Not required.

3] Explain features of Node.JS

a] Asynchronous & Event driven: Non-blocking, handles many requests at once.
b] Single Threaded & scalable: Uses event loop, support multiple client requests efficiently.
c] Built on google Chrome's V8 engine
d] NPM support → npm is node package manager, built-in.
e] Realtime data handling: Supports realtime apps like chat appl^n, online games or anything that requires instant updates.
f] JSON friendly: works smoothly with API's & no-sql databases.

4) Diff b/w Express JS & Node JS-

PARAMETER	EXPRESS JS	NODE JS
Definition	A web application framework built on top of Node.JS	A runtime environment for executing JS on server side.
Level	High Level	Low Level.
Routing	Built-in routing system like app.get() app.put(), etc.	Manual using HTTP module
Dependencies	Requires Node.JS to run	Independent
Use Case	Web Apps, REST API's, etc.	General purpose server side scripting
Error Handling	Standard way to define error handling middleware provided.	Handles manually for each request.

5] What is Rest API? Principles-

REST API: Representational State Transfer. An architecture style for designing networked applications.

An interface that allows different software applications to communicat over HTTP using REST principles.

Basically -

- Server exposes resources (like users, posts, products)
- Clients interact with these resources using HTTP methods.

Data usually exchanged in JSON format.

Principles of REST API's -

1] Stateless : Each request from client to server must contain all informatio needed
2] Client - Server Architecture : Client handles user interface and user exp. Server side handes data storage & business logic.
3] Uniform Interface
4] Cacheable : Responses define wether they're cacheable or not.
5] Layered system.
6] Resource - based.

6] Explain Event loops in Node.JS -

1. Node.JS is single threaded, non-blocking I/O model.
2. It can handle many requests at once without executing creation of multiple threads.
3. The secret of this efficiency is event-loop.
4. Event loop is the core mechanism in node.js that allow it to perform non-blocking asynchronous operations while still being single threaded.
5. It continuously checks call stack & call back queue processing tasks efficiently.
6. How it works -

Call Stack → Execute functions one by one
Node APIs → Handle async operations.
Callback queue → Stores completed async callbacks
Event loop → Moves callbacks from queue to stack when stack is free.

7. Phases -

Timers → Executes setTimeout/setInterval
poll → handles I/O events.
check → Executes setImmediate() callbacks
close → handles close events.

8. Event loop makes Node.js asynchronous, fast and scalable allowing it to handle multiple requests efficiently on a single thread.

9. Eg- console.log("Start"); setTimeout(() => {console.log("Timeout");}, 2000); console.log("End");

OUTPUT: Start
End
Timeout.