Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Internet Programming

Name: Parth Shikhare

Class: TE IT

Roll no.: 53

Date: 25/07/2025

Experiment no. 1

Title: Design a basic web page using HTML

Problem Definition:

Design a Basic Web page using HTML 5 tags Head, Body, Hyperlink, Formatting styles, Images and Table.

Pre-requisites:

Basic knowledge of HTML, CSS, and web design principles.

Theory:

HTML (HyperText Markup Language) is used to structure the content of web pages. CSS (Cascading Style Sheets) is used to style the visual presentation of the content. Together, they allow for creating well-structured and visually appealing websites. In this experiment, we design a student profile page with sections for personal details, achievements, and academic links. Flexbox or CSS Grid can be used to arrange the layout.

Procedure:

1. Create an HTML file and structure the page into sections: Header, Profile Information, Achievements, and Academics.
2. Use CSS to style each section, applying margins, paddings, colors, and font styles.
3. Add profile image and personal details in the profile section.
4. Add a bullet list of achievements.
5. Provide academic links for SE and TE.
6. Save the HTML and CSS files.
7. Test the webpage in a browser.

Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Program:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8"/>

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

<title>DBIT</title>

<style>

*{box-sizing:border-box}body{font-family:"Segoe UI",Tahoma,Geneva,Verdana,sans-serif;margin:0;padding:0;background:linear-gradient(to bottom right,#e0f2fe,#f8fafc);color:#1f2937;line-height:1.6}header{background:linear-gradient(135deg,#1e3a8a,#2563eb);color:#fff;padding:2rem 1rem;text-align:center;box-shadow:0 4px 12px rgba(0,0,0,.15)}header h1{margin:0;font-size:3rem;letter-spacing:1.2px;animation:fadeIn 1.5s ease-in-out}.banner{background:#1e40af;color:#fff;padding:10px 0;text-align:center;font-weight:500;overflow:hidden;position:relative}.banner span{display:inline-block;white-space:nowrap;padding-left:100%;animation:slide 12s linear infinite}@keyframes slide{0%{transform:translateX(0)}100%{transform:translateX(-100%)}}main{padding:40px 20px;max-width:1200px;margin:auto}.intro-section{display:flex;flex-wrap:wrap;gap:30px;margin-bottom:50px;background:rgba(255,255,255,.85);border-radius:14px;box-shadow:0 8px 20px rgba(0,0,0,.1);padding:30px;backdrop-filter:blur(8px);animation:fadeInUp 1s ease-in-out}.image-container{flex:1 1 300px;max-width:300px}.image-container img{width:100%;border-radius:12px;box-shadow:0 6px 16px rgba(0,0,0,.2)}.info-text{flex:2 1 400px;display:flex;flex-direction:column;justify-content:center}.info-text h2{margin:0 0 10px;color:#1d4ed8;font-size:2.2rem}section{background:#fff;margin-bottom:40px;padding:30px;border-radius:14px;box-shadow:0 8px 20px rgba(0,0,0,.06);animation:fadeInUp 1s ease}section h2{color:#1e3a8a;border-bottom:2px solid #e2e8f0;padding-bottom:10px;margin-bottom:20px;font-size:1.8rem}ul{list-style:disc;margin-
```

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

left:20px;padding:0}ul li{margin-bottom:12px;font-size:1.05rem}a{color:#2563eb;font-weight:500;text-decoration:none;transition:color .2s ease}a:hover{color:#1e40af;text-decoration:underline}footer{background:#1f2937;color:#fff;text-align:center;padding:2rem 1rem;margin-top:40px;font-size:.95rem}footer p{margin:6px 0}footer a{color:#93c5fd;text-decoration:none;transition:color .2s ease}footer a:hover{color:#bfdbfe;text-decoration:underline}@media(max-width:768px){.intro-section{flex-direction:column;align-items:center;text-align:center}.image-container{max-width:100%}.info-text{text-align:center}}@media(max-width:480px){header h1{font-size:2.2rem}section{padding:20px}.info-text h2{font-size:1.7rem}}@keyframes fadeInUp{from{opacity:0;transform:translateY(20px)}to{opacity:1;transform:translateY(0)}}@keyframes fadeIn{from{opacity:0}to{opacity:1}}

</style>

</head>

<body>

<header><h1>Don Bosco Institute of Technology</h1></header>

Department of Information Technology

<main>

<section class="intro-section">

<h2>Parth Shikhare</h2> I am Parth Shikhare, currently pursuing a B.Tech degree at Don Bosco Institute of Technology in the Information Technology Department.

</section>

<section class="achievements-section">

<h2>Achievements</h2>

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Awarded "Best Engineering College in Innovation" for three consecutive years.

Our students consistently secure top ranks in university examinations.

Numerous alumni hold leadership positions in renowned multinational corporations.

Successful completion of various research projects with industry partners.

Strong track record in inter-collegiate technical competitions.

</section>

<section class="academics-section">

<h2>Academics</h2>

SE

TE

</section>

</main>

<footer>

Email: info@dbit.edu

LinkedIn: DBIT LinkedIn Page

Phone: +91 12345 67890

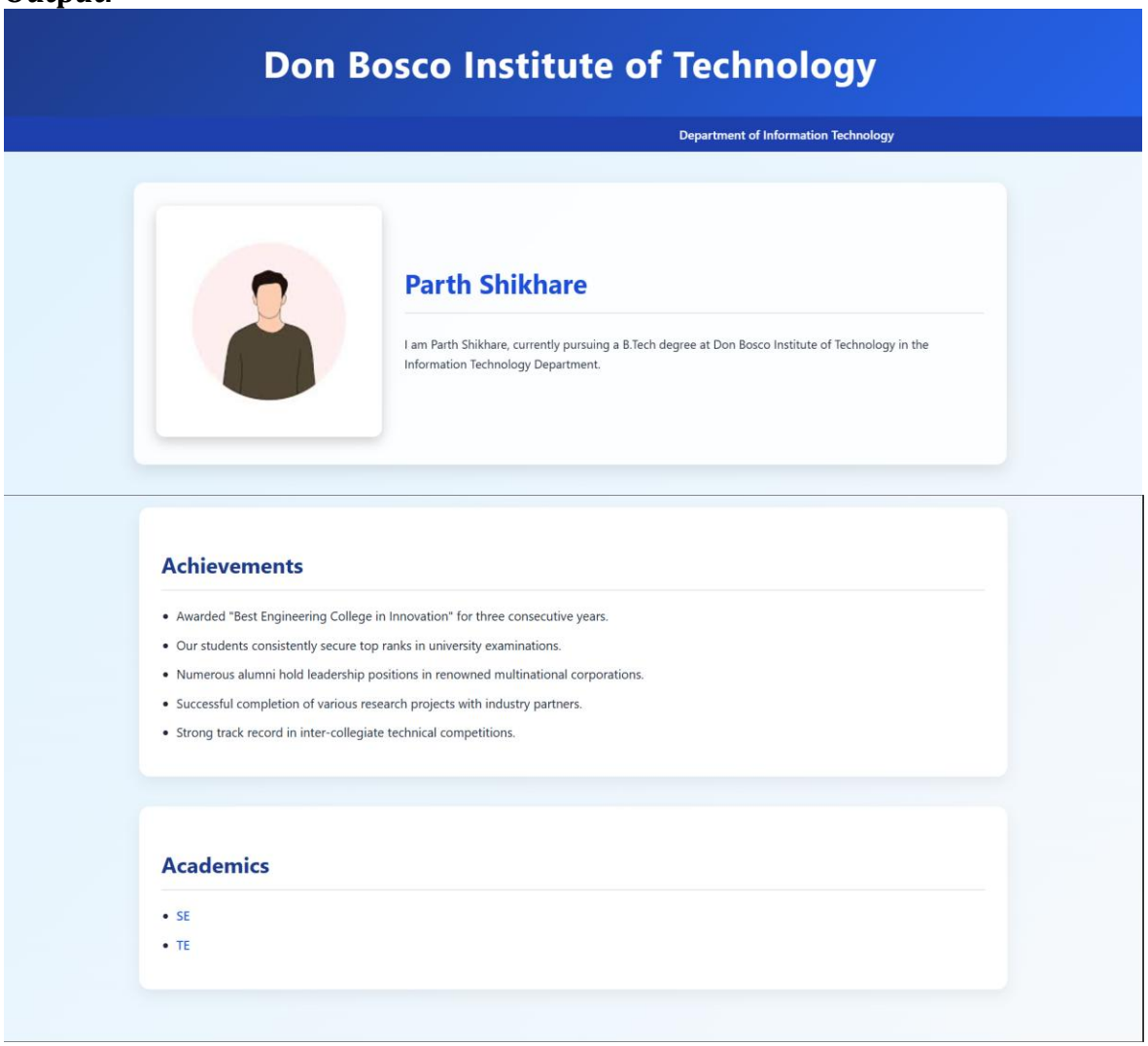
© 2025 DBIT. All rights reserved.

</footer>

</body>

</html>

Output:



Results:

The student profile webpage was successfully created and displayed personal details, achievements, and academic links in a clean, organized format.

References:

1. https://www.w3schools.com/html/
2. MDN Web Docs (https://developer.mozilla.org/)

Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Internet Programming

Name: Parth Pravin Shikhare

Class: TE-IT

Roll No.: 53

Date: 1/8/2025

Experiment No.: 2

Title:
Design a basic web page using advanced HTML5

Problem Definition:
Design web pages using advanced HTML5 features including List, Frames, Forms, and Multimedia tags.

Pre-requisites:
1. Understanding of basic HTML5 structure and syntax
2. Familiarity with HTML5 form elements, multimedia, and lists
3. Basic knowledge of HTML frames and layout structure
4. A text editor or IDE and a web browser for testing

Theory:
Advanced HTML5 features allow developers to create more interactive and media-rich webpages.

Key elements include:
- Lists: <ul>, <ol>, and <li> are used to create unordered and ordered lists.
- Frames: Though <frameset> is deprecated in HTML5, iframes (<iframe>) can be used to embed external web pages.
- Forms: Include form controls like <input>, <textarea>, <select>, <button>, etc., allowing user interaction.
- Multimedia: Tags like <audio> and <video> are used to embed media files with controls for play, pause, etc.

Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Procedure:

1. Open a text editor or IDE.
2. Create a new HTML file and set up the basic structure.
3. Add list elements (ordered/unordered) to demonstrate list usage.
4. Use <iframe> to embed another webpage or document.
5. Create a form using various input controls (text, radio, checkbox, etc.).
6. Embed audio and video elements using <audio> and <video> tags.
7. Save the file with a .html extension and test it in a browser.
8. Modify and re-test as necessary.

Program:

First Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8"/>

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

<title>SE Marksheet</title>

<style>

body{font-family:Arial,sans-serif;margin:0;padding:20px}

h1{text-align:center;margin-bottom:20px}

.marksheet-container{max-width:1000px;margin:auto}

table{width:100%;border-collapse:collapse}

th,td{border:1px solid #000;padding:8px;text-align:center}

td table{width:100%;border:none}

td table td{border:none;padding:4px}

tr:nth-child(even){background-color:#f5f5f5}
```

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

.footer-row td{border:none;font-weight:bold;text-align:left}

button{display:block;margin:20px auto 0;padding:8px 15px;border:1px solid #000;background:none;cursor:pointer}

a{text-decoration:none;color:inherit}

</style>

</head>

<body>

<h1>Second Year Marksheet</h1>

<table>

<tr><th>Sr No</th><th>Course Name</th><th>Credits</th><th>ESE<table><tr><th>Min</th><th>Obtained</th></tr></table></th><th>IA<table><tr><th>Min</th><th>Obtained</th></tr></table></th><th>Total<table><tr><th>Max</th><th>Obtained</th></tr></table></th><th>Grades</th><th>Pointers</th></tr>

<tr><td>1</td><td>Maths IV</td><td>4</td><td><table><tr><td>32</td><td>73</td></tr></table></td><td><table><tr><td>8</td><td>19</td></tr></table></td><td><table><tr><td>100</td><td>92</td></tr></table></td><td>O</td><td>10</td></tr>

<tr><td>2</td><td>Computer Networks</td><td>3</td><td><table><tr><td>32</td><td>66</td></tr></table></td><td><table><tr><td>8</td><td>19</td></tr></table></td><td><table><tr><td>100</td><td>85</td></tr></table></td><td>O</td><td>10</td></tr>

<tr><td>3</td><td>Operating System</td><td>4</td><td><table><tr><td>32</td><td>57</td></tr></table></td><td><table><tr><td>8</td><td>16</td></tr></table></td><td><table><tr><td>100</td><td>73</td></tr></table></td><td>B</td><td>8</td></tr>

<tr><td>4</td><td>Automata Theory</td><td>3</td><td><table><tr><td>32</td><td>44</td></tr></table><

```
/td><td><table><tr><td>8</td><td>20</td></tr></table></td><td><table><tr><
td>100</td><td>64</td></tr></table></td><td>D</td><td>7</td></tr>

<tr><td>5</td><td>Computer Organisation and
Architecture</td><td>3</td><td><table><tr><td>32</td><td>64</td></tr></tab
le></td><td><table><tr><td>8</td><td>19</td></tr></table></td><td><table><
tr><td>100</td><td>83</td></tr></table></td><td>O</td><td>10</td></tr>

<tr class="footer-row"><td colspan="3">Remark: SUCCESSFUL</td><td
colspan="3">SGPI: 9.35</td><td colspan="2">Marks Obtained: 657 /
775</td></tr>

</table>

</div>

<a href="./profile.html"><button>Go Back</button></a>

</body>

</html>
```

Second Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Forms</title>

<style>

body{font-family:Arial,sans-serif;margin:0;padding:20px}

h1{text-align:center;margin-bottom:20px}

form{max-width:600px;margin:auto;padding:20px;border:1px solid #000}
```

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

```
label{font-weight:bold;display:block;margin-bottom:5px}

input[type="text"],input[type="password"],input[type="date"],select{width:100%;p
adding:8px;margin-bottom:15px;border:1px solid #000}

.radio-group{margin-bottom:15px}

.radio-group label{font-weight:normal;margin-right:10px}

button{padding:8px 15px;border:1px solid #000;background:none;cursor:pointer}

.back-link{text-align:center;margin-top:15px}

a{text-decoration:none;color:inherit}

</style>

</head>

<body>

<h1>Student Details Form</h1>

<form action="submit">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required>

<label>Department:</label>

<div class="radio-group">

<input type="radio" id="comps" name="fav_language" value="COMPS"><label
for="comps">COMPS</label>

<input type="radio" id="it" name="fav_language" value="IT"><label
for="it">IT</label>

<input type="radio" id="extc" name="fav_language" value="EXTC"><label
for="extc">EXTC</label>

<input type="radio" id="mech" name="fav_language" value="MECH"><label
for="mech">MECH</label>
```

```
</div>
<label for="password">Roll No:</label>
<input type="text" id="password" name="password" required>
<label for="dob">Date of Birth:</label>
<input type="date" id="dob" name="dob">
<label for="gender">Gender</label>
<select name="gender_opt" id="gender">
<option value="Male">Male</option>
<option value="Female">Female</option>
<option value="Other">Other</option>
</select>
<button type="submit">Submit</button>
</form>
<div class="back-link">
<a href="./profile.html"><button>Go Back</button></a>
</div>
</body>
</html>
```

Output:

Second Year Marksheet

Sr No	Course Name	Credits	ESE		IA		Total		Grades	Pointers
			Min	Obtained	Min	Obtained	Max	Obtained		
1	Maths IV	4	32	73	8	19	100	92	O	10
2	Computer Networks	3	32	66	8	19	100	85	O	10
3	Operating System	4	32	57	8	16	100	73	B	8
4	Automata Theory	3	32	44	8	20	100	64	D	7
5	Computer Organisation and Architecture	3	32	64	8	19	100	83	O	10

Remark: SUCCESSFUL **SGPI: 9.35** **Marks Obtained: 657 / 775**

Go Back

Student Details Form

Name:

Department:

COMPS
IT
EXTC
MECH

Roll No:

Date of Birth:

dd-mm-yyyy

Gender

Male

Submit

Go Back

Results:

Successfully created an advanced HTML5 webpage using lists, frames, forms, and multimedia elements.

References:

1. https://developer.mozilla.org/en-US/docs/Web/HTML
2. https://www.w3schools.com/html/

Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Internet Programming

Name: Parth Pravin Shikhare

Class: TE-IT

Roll No.: 53

Date: 08/08/2025

Experiment No.: 3

Title:
Design a web pages using CSS3 styles

Problem Definition:
Design web pages using CSS3 styles focusing on Color, Background, Fonts, Tables, and Lists.

Pre-requisites:
1. Understanding of basic HTML5 structure and elements
2. Basic knowledge of CSS syntax and selectors
3. A text editor or IDE and a modern web browser for testing
4. Familiarity with linking external stylesheets and using inline/internal styles

Theory:
CSS3 (Cascading Style Sheets Level 3) provides powerful styling capabilities for web pages, enabling separation of content from presentation. CSS3 introduces modules that enhance design flexibility and maintainability.

Key Concepts covered in this experiment:
- Color: Using color names, HEX, RGB, RGBA, HSL, and HSLA to set text and element colors.
- Background: Applying background-color, background-image, background-repeat, background-position, background-size, and layered backgrounds.
- Fonts & Text: Using @font-face, font-family, font-size, font-weight, line-height, text-transform, letter-spacing, and Google Fonts integration.
- Tables: Styling <table>, <th>, <td> with borders, border-collapse, zebra striping (nth-child), padding, and alignment.

- Lists: Customizing list-style-type, list-style-image, list-style-position, and creating horizontal/inline navigation lists.

Procedure:

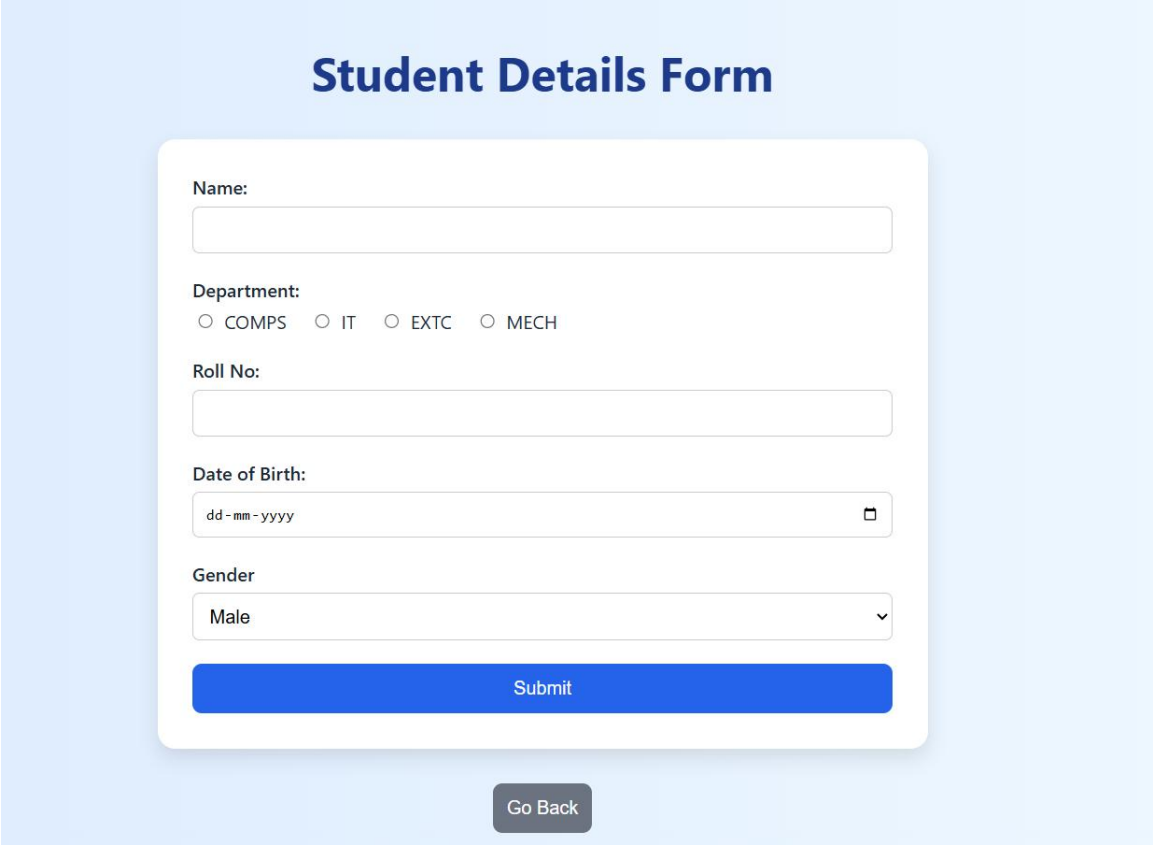
1. Create a basic HTML5 page structure (index.html).
2. Link an external CSS file (style.css) using the <link> tag in the <head> section.
3. Define base styles for the body including fonts and colors.
4. Add sections to demonstrate:
   a) Color: Apply different color formats (HEX, RGB, HSL) to headings and paragraphs.
   b) Background: Set solid colors, images, and gradients; adjust size and positioning.
   c) Fonts: Use web-safe fonts and an imported web font; adjust weight, size, and line-height.
   d) Tables: Create a sample table and style borders, header row, and zebra stripes using :nth-child.
   e) Lists: Create ordered/unordered lists and style bullets, spacing, and display inline for a menu.
5. Test the page in a web browser and verify styles render correctly across sections.
6. Validate CSS using browser DevTools; refine selectors and properties as needed.
7. Save final files and capture screenshots of styled sections.

Program:

```
<!DOCTYPE html><html lang="en"><head><meta charset="UTF-8"><meta
name="viewport" content="width=device-width,initial-
scale=1.0"><title>Forms</title><style>body{font-family:'Segoe
UI',Tahoma,Geneva,Verdana,sans-serif;background:linear-gradient(to
right,#dbeafe,#f0f9ff);margin:0;padding:0}h1{text-align:center;color:#1e3a8a;margin-
top:40px;font-size:2.5em}form{max-width:600px;margin:30px
auto;padding:30px;background-color:#fff;border-radius:15px;box-shadow:0 8px 16px
rgba(0,0,0,.1);display:flex;flex-direction:column}label{font-weight:600;margin-
bottom:6px;color:#1f2937}input[type="text"],input[type="password"],input[type="date"],s
elect{padding:10px;border-radius:6px;border:1px solid #ccc;margin-bottom:20px;font-
size:1em;transition:border .3s ease}input:focus,select:focus{border-
color:#2563eb;outline:0}.radio-group{margin-bottom:20px}.radio-group label{font-
weight:400;margin-right:15px;cursor:pointer}.radio-group input{margin-
right:5px}button{padding:12px;background-color:#2563eb;color:#fff;border:0;border-
radius:8px;font-size:1em;cursor:pointer;transition:background-color .3s
ease}button:hover{background-color:#1e40af}.back-link{text-align:center;margin-
top:25px}.back-link a{text-decoration:none}.back-link button{background-
color:#6b7280}.back-link button:hover{background-
color:#4b5563}</style></head><body><h1>Student Details Form</h1><form
action="submit"><label for="name">Name:</label><input type="text" id="name"
name="name" required><label>Department:</label><div class="radio-group"><input
type="radio" id="comps" name="fav_language" value="COMPS"><label
for="comps">COMPS</label><input type="radio" id="it" name="fav_language"
```

value="IT"><label for="it">IT</label><input type="radio" id="extc" name="fav_language" value="EXTC"><label for="extc">EXTC</label><input type="radio" id="mech" name="fav_language" value="MECH"><label for="mech">MECH</label> <label for="password">Roll No:</label><input type="text" id="password" name="password" required><label for="dob">Date of Birth:</label><input type="date" id="dob" name="dob"><label for="gender">Gender</label><select name="gender_opt" id="gender"><option value="Male">Male</option><option value="Female">Female</option><option value="Other">Other</option></select><button type="submit">Submit</button></form> <button>Go Back</button> <script src="./te-validation.js"></script></body></html>

Output:



Results:

Successfully created web pages styled using CSS3 features for color, background, fonts, tables, and lists.

References:

1. https://developer.mozilla.org/en-US/docs/Web/CSS
2. https://www.w3schools.com/css/

Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Internet Programming

Name: Parth Pravin Shikhare

Class: TE-IT

Roll No.: 53

Date: 08/08/2025

Experiment No.: 4

Title:
Design a Web pages using Advance CSS3

Problem Definition:
Design web pages using advanced CSS3 features such as selectors, pseudo-classes, and pseudo-elements.

Pre-requisites:
1. Understanding of HTML5 structure and elements
2. Knowledge of basic CSS selectors and properties
3. Familiarity with linking external CSS files
4. A text editor or IDE and a modern web browser for testing

Theory:
CSS3 advanced features provide greater control over the styling and interactivity of web pages.

Key concepts covered:
- Selectors: Used to target HTML elements for styling. Types include universal (*), element (p, h1), class (.classname), ID (#idname), attribute selectors ([type='text']), and combinators.
- Pseudo-classes: Define the special state of an element. Examples include :hover, :focus, :active, :first-child, :nth-child().
- Pseudo-elements: Allow styling of specific parts of elements. Examples include ::before, ::after, ::first-letter, and ::first-line.

These tools allow developers to add interactivity (hover effects), highlight specific parts of elements (first letter of a paragraph), and create complex layouts with less code.

Procedure:

1. Create a basic HTML page structure with headings, paragraphs, lists, links, and forms.
2. Link an external CSS3 file to the HTML document.
3. Apply different types of CSS selectors to style multiple elements uniquely.
4. Add pseudo-classes like :hover to links and buttons, :focus to input fields, and :nth-child to list items.
5. Use pseudo-elements like ::before and ::after to insert decorative content, and ::first-letter or ::first-line for typography enhancements.
6. Test the webpage in a modern browser and observe the applied effects.
7. Debug and refine CSS styles as necessary.
8. Save final files and capture screenshots of the outputs.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>SE Marksheet</title>
  <style>
    body {
      font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
      background: linear-gradient(to right, #f0f4ff, #dbeafe);
      margin: 0;
      padding: 20px;
    }

    h1 {
      text-align: center;
      color: #1e3a8a;
      font-size: 2.5em;
      margin-bottom: 30px;
    }

    .marksheet-container {
      max-width: 1000px;
      margin: auto;
      background: white;
      padding: 20px;
      border-radius: 12px;
      box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
```

```
    overflow-x: auto;
  }

  table {
    width: 100%;
    border-collapse: collapse;
    border: 2px solid #1e3a8a;
  }

  th, td {
    border: 1px solid #888;
    padding: 10px;
    text-align: center;
  }

  th {
    background-color: #1e3a8a;
    color: white;
  }

  td table {
    width: 100%;
    border: none;
  }

  td table td {
    border: none;
    padding: 6px;
  }

  tr:nth-child(even) {
    background-color: #f9fafe;
  }

  .footer-row {
    background-color: #e0e7ff;
    font-weight: bold;
    text-align: left;
  }

  .footer-row td {
    border: none;
    font-size: 1.1em;
```

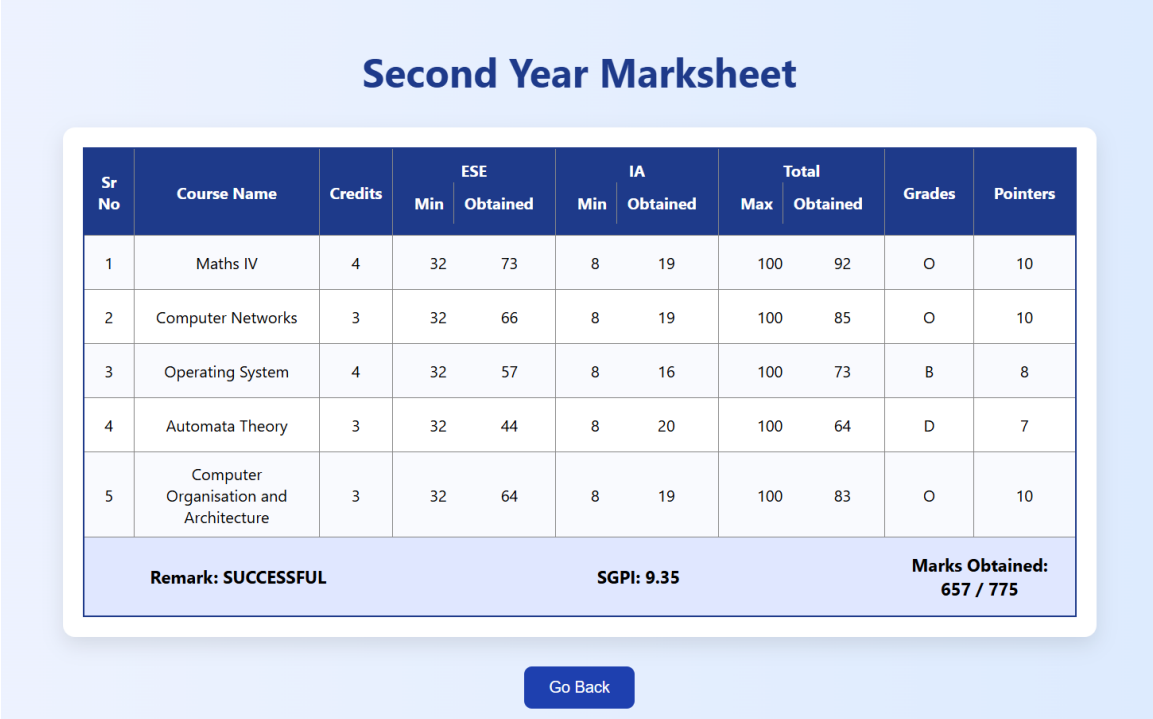
```
    padding: 15px;
  }

  button {
    display: block;
    margin: 30px auto 0;
    padding: 12px 25px;
    background-color: #2563eb;
    color: white;
    border: none;
    border-radius: 8px;
    font-size: 1em;
    cursor: pointer;
    transition: background-color 0.3s ease;
  }

  button:hover {
    background-color: #1e40af;
  }
  a {
    text-decoration: none;
    color: #2563eb;
  }
 </style>
</head>
<body>
 <h1>Second Year Marksheet</h1>
 <div class="marksheet-container">
  <table>
   <tr>
    <th>Sr No</th>
    <th>Course Name</th>
    <th>Credits</th>
    <th>
     ESE
     <table>
      <tr>
       <th>Min</th>
       <th>Obtained</th>
      </tr>
     </table>
    </th>
    <th>
```

```
    IA
    <table>
     <tr>
      <th>Min</th>
      <th>Obtained</th>
     </tr>
    </table>
   </th>
   <th>
    Total
    <table>
     <tr>
      <th>Max</th>
      <th>Obtained</th>
     </tr>
    </table>
   </th>
   <th>Grades</th>
   <th>Pointers</th>
  </tr>

  <!-- Course Rows -->
  <tr>
   <td>1</td>
   <td>Maths IV</td>
   <td>4</td>
   <td><table><tr><td>32</td><td>73</td></tr></table></td>
   <td><table><tr><td>8</td><td>19</td></tr></table></td>
   <td><table><tr><td>100</td><td>92</td></tr></table></td>
   <td>O</td>
   <td>10</td>
  </tr>

  <tr>
   <td>2</td>
   <td>Computer Networks</td>
   <td>3</td>
   <td><table><tr><td>32</td><td>66</td></tr></table></td>
   <td><table><tr><td>8</td><td>19</td></tr></table></td>
   <td><table><tr><td>100</td><td>85</td></tr></table></td>
   <td>O</td>
   <td>10</td>
  </tr>
```

```
    <tr>
      <td>3</td>
      <td>Operating System</td>
      <td>4</td>
      <td><table><tr><td>32</td><td>57</td></tr></table></td>
      <td><table><tr><td>8</td><td>16</td></tr></table></td>
      <td><table><tr><td>100</td><td>73</td></tr></table></td>
      <td>B</td>
      <td>8</td>
    </tr>

    <tr>
      <td>4</td>
      <td>Automata Theory</td>
      <td>3</td>
      <td><table><tr><td>32</td><td>44</td></tr></table></td>
      <td><table><tr><td>8</td><td>20</td></tr></table></td>
      <td><table><tr><td>100</td><td>64</td></tr></table></td>
      <td>D</td>
      <td>7</td>
    </tr>

    <tr>
      <td>5</td>
      <td>Computer Organisation and Architecture</td>
      <td>3</td>
      <td><table><tr><td>32</td><td>64</td></tr></table></td>
      <td><table><tr><td>8</td><td>19</td></tr></table></td>
      <td><table><tr><td>100</td><td>83</td></tr></table></td>
      <td>O</td>
      <td>10</td>
    </tr>

    <!-- Footer Row -->
    <tr class="footer-row">
      <td colspan="3">Remark: SUCCESSFUL</td>
      <td colspan="3">SGPI: 9.35</td>
      <td colspan="2">Marks Obtained: 657 / 775</td>
    </tr>
  </table>
 </div>
```

```
  <a href="./profile.html"><button>Go Back</button></a>
</body>
</html>
```

Output:



Second Year Marksheet

Sr No	Course Name	Credits	ESE		IA		Total		Grades	Pointers
			Min	Obtained	Min	Obtained	Max	Obtained		
1	Maths IV	4	32	73	8	19	100	92	O	10
2	Computer Networks	3	32	66	8	19	100	85	O	10
3	Operating System	4	32	57	8	16	100	73	B	8
4	Automata Theory	3	32	44	8	20	100	64	D	7
5	Computer Organisation and Architecture	3	32	64	8	19	100	83	O	10
Remark: SUCCESSFUL					**SGPI: 9.35**				**Marks Obtained: 657 / 775**	

Results:

Successfully designed advanced CSS3 styled web pages using selectors, pseudo-classes, and pseudo-elements.

References:

1. https://developer.mozilla.org/en-US/docs/Web/CSS
2. https://www.w3schools.com/css/

Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Internet Programming

Name: Parth Pravin Shikhare

Class: TEIT

Roll No.: 53

Date:22/08/25

Experiment No.: 5

Title:

Design a web pages using Bootstrap

Problem Definition:

Design web pages using Bootstrap framework implementing Grid system, Forms, Buttons, and Navbar.

Pre-requisites:

1. Understanding of HTML5 and CSS3 basics
2. Basic knowledge of responsive design concepts
3. Familiarity with Bootstrap framework structure
4. A text editor or IDE and a modern web browser
5. Internet connection to include Bootstrap CDN or downloaded Bootstrap files

Theory:

Bootstrap is a popular front-end open-source framework used for developing responsive and mobile-first web pages. It provides pre-styled components and a grid system for easier design and faster development.

Key Bootstrap Features used in this experiment:
- Grid System: Bootstrap uses a 12-column grid system with responsive breakpoints (xs, sm, md, lg, xl) to create layouts.
- Forms: Pre-styled form elements such as input fields, checkboxes, radio buttons, and validation classes.
- Buttons: Various button classes like btn, btn-primary, btn-success, btn-danger, and customization options.

- Navbar: Responsive navigation bar with support for branding, links, dropdowns, and toggler for smaller screens.

Procedure:

1. Create a new HTML file and include Bootstrap via CDN in the <head> section.
   2. Implement the Bootstrap grid system using , , and classes.
   3. Add form elements styled with Bootstrap classes (form-control, form-group, etc.).
   4. Insert buttons using predefined classes like btn btn-primary and customize their appearance.
   5. Create a responsive navigation bar using <nav class='navbar'> and related classes.
   6. Test the webpage in a browser and resize the window to verify responsiveness.
   7. Refine the design and ensure all components are functioning correctly.
   8. Save final files and capture screenshots of the results.

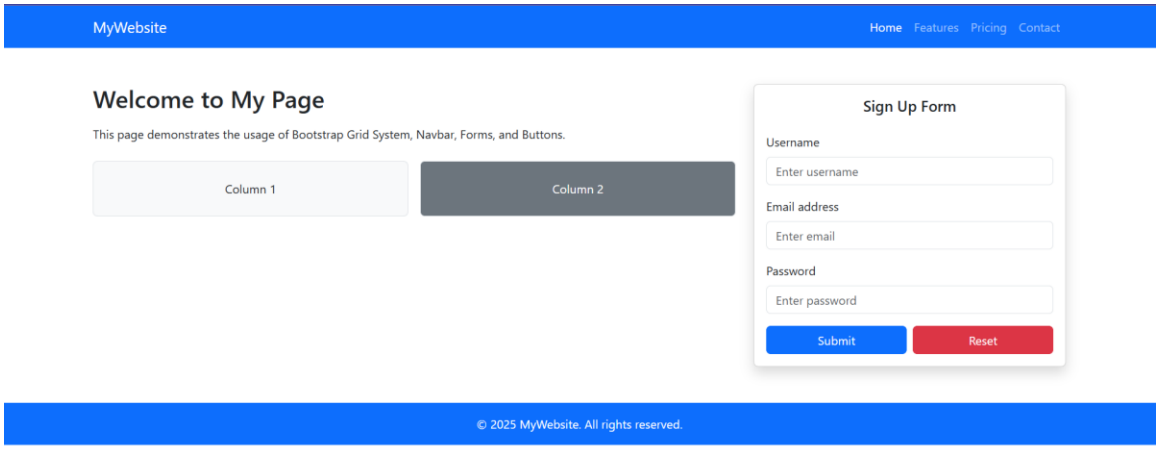
Program:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Bootstrap Example Page</title>
  <link
   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.c
ss"
   rel="stylesheet"
  />
 </head>
 <body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
   <div class="container">
    <a class="navbar-brand" href="#">MyWebsite</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="#">Features</a></li>
      <li class="nav-item"><a class="nav-link" href="#">Pricing</a></li>
      <li class="nav-item"><a class="nav-link" href="#">Contact</a></li>
     </ul>
    </div>
   </div>
  </nav>

  <div class="container mt-5">
   <div class="row g-4">
    <div class="col-md-8">
     <h2 class="mb-3">Welcome to My Page</h2>
```

```
  <p class="mb-4">
    This page demonstrates the usage of Bootstrap Grid System, Navbar,
    Forms, and Buttons.
  </p>
  <div class="row g-3">
    <div class="col-md-6">
      <div class="p-4 bg-light border text-center rounded">Column 1</div>
    </div>
    <div class="col-md-6">
      <div class="p-4 bg-secondary text-white border text-center rounded">
        Column 2
      </div>
    </div>
  </div>
</div>
<div class="col-md-4">
  <div class="card shadow">
    <div class="card-body">
      <h5 class="card-title text-center mb-4">Sign Up Form</h5>
      <form>
        <div class="mb-3">
          <label for="username" class="form-label">Username</label>
          <input
            type="text"
            class="form-control"
            id="username"
            placeholder="Enter username"
          />
        </div>
        <div class="mb-3">
          <label for="email" class="form-label">Email address</label>
          <input
            type="email"
            class="form-control"
            id="email"
            placeholder="Enter email"
          />
        </div>
        <div class="mb-3">
          <label for="password" class="form-label">Password</label>
```

```
              <input
                type="password"
                class="form-control"
                id="password"
                placeholder="Enter password"
              />
            </div>
            <div class="d-flex justify-content-between">
              <button type="submit" class="btn btn-primary w-50 me-
2">Submit</button>
              <button type="reset" class="btn btn-danger w-50">Reset</button>
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>

  <footer class="bg-primary text-white text-center py-3 mt-5">
    <p class="mb-0">&copy; 2025 MyWebsite. All rights reserved.</p>
  </footer>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min
.js"></script>
 </body>
</html>
```

Output:



Results:

Successfully created responsive web pages using Bootstrap Grid system, Forms, Buttons, and Navbar.

References:

1. https://getbootstrap.com/
2. https://www.w3schools.com/bootstrap/
3. https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design

Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Internet Programming

Name: Parth Pravin Shikhare

Class: TEIT

Roll No.: 53

Date: 17/09/25

Experiment No.: 6

Title:

Design a web page using JavaScript Validation

Problem Definition:

Design a web page using JavaScript concepts such as Variables, Operators, Conditions, Loops, Functions, Events, Classes, Objects, and implement form validation.

Pre-requisites:

1. Basic knowledge of HTML5 and CSS3
2. Understanding of JavaScript syntax and basic programming concepts
3. Familiarity with web browsers and developer tools
4. A text editor or IDE for coding and testing

Theory:

JavaScript is a powerful client-side scripting language used to enhance the interactivity of web pages. It supports programming constructs such as variables, operators, conditions, loops, functions, events, and object-oriented features like classes and objects.

Key JavaScript Concepts used in this experiment:
- Variables: Used to store data values (var, let, const).
- Operators: Arithmetic, relational, logical, and assignment operators to perform computations.
- Conditions: if, if-else, and switch statements to implement decision-making.
- Loops: for, while, do-while loops for iteration.
- Functions: Block of reusable code that performs a specific task.
- Events: Actions like onClick, onChange, onSubmit that trigger JavaScript functions.
- Classes & Objects: Used for object-oriented programming, encapsulating properties and

methods.
- Validation: Ensuring user input meets certain criteria (e.g., email format, required fields).

Procedure:

1. Create a basic HTML form with input fields such as name, email, password, etc.
2. Write JavaScript code to declare variables and use operators for simple calculations.
3. Implement conditions (if, switch) to validate input values.
4. Use loops to process multiple inputs or repeated actions.
5. Define functions to modularize the validation logic.
6. Attach event listeners to form fields and buttons to trigger validation functions.
7. Create a simple class and object to demonstrate object-oriented concepts.
8. Test the form by entering various inputs and verifying the validation messages.
9. Debug errors using browser developer console and refine the script.
10. Save final files and capture screenshots of the outputs.

Program:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Form</title>
  <style>
   body {
    font-family: Arial, sans-serif;
    background: linear-gradient(135deg, #e3f2fd, #bbdefb);
    margin: 0;
    padding: 0;
   }

   h1,
   h2,
   h3 {
    text-align: center;
    padding: 5px;
    margin: 5px 0;
   }

   form {
    max-width: 500px;
    margin: 30px auto;
    padding: 20px;
    background: #ffffff;
    border-radius: 12px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);
   }

   input[type="text"],
```

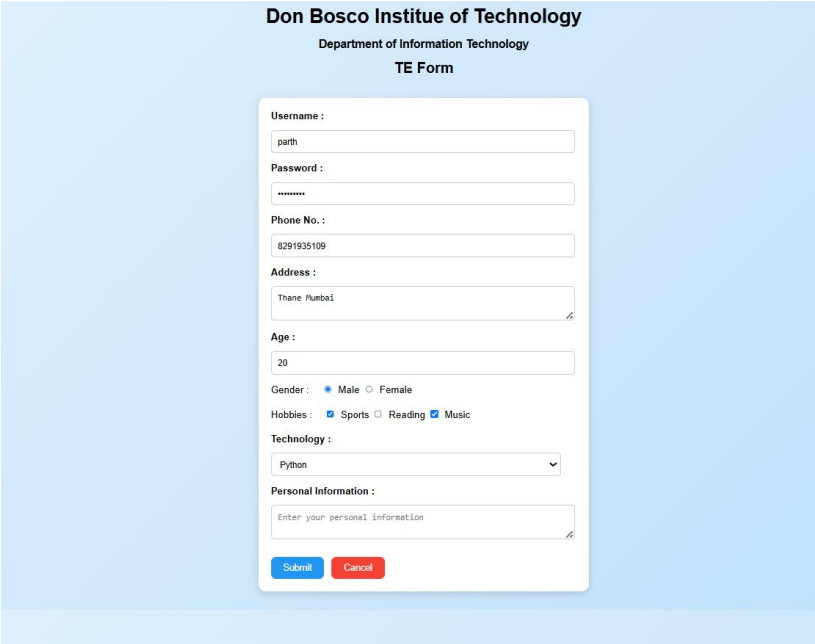
```
input[type="email"],
input[type="number"],
input[type="tel"],
input[type="password"],
textarea,
select {
  width: 95%;
  padding: 10px;
  margin: 8px 0 15px 0;
  border: 1px solid #ccc;
  border-radius: 6px;
  font-size: 14px;
  transition: 0.3s;
}

input:hover,
select:hover {
  border-color: #2196f3;
}
input:focus,
select:focus {
  outline: none;
  border-color: #1976d2;
  box-shadow: 0 0 6px rgba(25, 118, 210, 0.4);
}

label {
  display: inline-block;
  margin-bottom: 6px;
  font-weight: bold;
}

.inline-options {
  margin-bottom: 15px;
}
.inline-options input {
  margin-right: 6px;
}
.inline-options label {
  margin-right: 15px;
  font-weight: normal;
}

button {
  padding: 10px 20px;
  margin: 10px 8px 0 0;
  border: none;
  border-radius: 8px;
  font-size: 15px;
  cursor: pointer;
```

```
    transition: 0.3s;
   }

   button[type="submit"] {
    background: #2196f3;
    color: white;
   }

   button[type="submit"]:hover {
    background: #1976d2;
   }

   button[type="reset"] {
    background: #f44336;
    color: white;
   }

   button[type="reset"]:hover {
    background: #d32f2f;
   }
  </style>

 </head>
 <body>
  <h1>Don Bosco Institue of Technology</h1>
  <h3>Department of Information Technology</h3>
  <h2>TE Form</h2>

  <form id="myForm">
   <label>Username :</label>
   <input
    type="text"
    name="name"
    placeholder="Enter your username"
    required
   /><br />

   <label>Password :</label>
   <input
    type="password"
    name="password"
    placeholder="Enter your password"
    required
   /><br />

   <label>Phone No. :</label>
   <input
    type="tel"
    name="tel"
    placeholder="Enter your phone number"
```

```
      required
    /><br />

    <label>Address :</label>
    <textarea name="Address" id="Address" placeholder="Enter your address"></textarea>

    <label>Age :</label>
    <input type="number" name="age" required /><br />

    <div class="inline-options">
      <label>Gender :</label>
      <input type="radio" name="Gender" /> Male
      <input type="radio" name="Gender" /> Female
    </div>

    <div class="inline-options">
      <label>Hobbies :</label>
      <input type="checkbox" name="hobby" /> Sports
      <input type="checkbox" name="hobby" /> Reading
      <input type="checkbox" name="hobby" /> Music
    </div>

    <label>Technology :</label>
    <select name="Technology">
      <option value="Java">Java</option>
      <option value="C">C</option>
      <option value="Python">Python</option></select
    ><br />

    <label>Personal Information :</label>
    <textarea name="info" id="info" placeholder="Enter your personal
information"></textarea>

    <button type="submit">Submit</button>
    <button type="reset" id="cancelBtn">Cancel</button>
  </form>

  <script>
    document.getElementById("myForm").addEventListener("submit", function (e) {
      e.preventDefault();
      let username = document.getElementsByName("name")[0].value;
      let password = document.getElementsByName("password")[0].value;
      let phone = document.getElementsByName("tel")[0].value;
      let age = document.getElementsByName("age")[0].value;
      let nameRegex = /^[A-Za-z]+$/;
      if (!nameRegex.test(username)) {
        alert("Username should not contain numbers");
        return;
      }
      if (password.length < 8) {
```

```
      alert("Password should be at least 8 characters long");
      return;
    }
    if (age < 8 || age > 80) {
      alert("Age should be between 8 and 80");
      return;
    }
    let phoneRegex = /^[0-9]{10}$/;
    if (!phoneRegex.test(phone)) {
      alert("Phone number should contain exactly 10 digits");
      return;
    }
    let pname = prompt("Enter your name");
    if (pname !== null) {
      let conf = confirm("Do you want to submit?");
      if (conf) {
        alert("Confirm Successful");
      }
    }
  });

  document.getElementById("cancelBtn").addEventListener("click", function (e) {
    e.preventDefault();
    let pname = prompt("Enter your name");
    if (pname !== null) {
      let conf = confirm("Do you want to cancel?");
      if (conf) {
        alert("Cancel Successful");
      }
    }
  });
 </script>
</body>
</html>
```

Output:



Results:

Successfully created a web page using JavaScript that demonstrates variables, operators, conditions, loops, functions, events, classes, objects, and input validation.

References:

1. https://developer.mozilla.org/en-US/docs/Web/JavaScript
2. https://www.w3schools.com/js/
3. Eloquent JavaScript by Marijn Haverbeke

Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Internet Programming

Name: Parth Pravin SHikhare

Class: TE-IT

Roll No.: 53

Date: 17/09/25

Experiment No.: 7

Title:
To Create a web pages using React Form Validations

Problem Definition:
To create web pages that demonstrate form validations using React along with JavaScript concepts such as Validations, Arrays, String manipulations, and Date functions.

Pre-requisites:
1. Basic knowledge of HTML5, CSS3, and JavaScript
2. Understanding of React.js framework and JSX syntax
3. Familiarity with React components, props, and state
4. A text editor or IDE (like VS Code) with Node.js installed
5. Knowledge of JavaScript arrays, strings, and date handling

Theory:
React is a JavaScript library for building user interfaces, particularly single-page applications. React components allow for modular, reusable, and efficient UI design. Form validation in React can be achieved using controlled components, event handling, and JavaScript functions.

Key Concepts covered in this experiment:
- JavaScript Validations: Ensuring input fields follow specific criteria (e.g., email format, password length).
- Arrays: Storing and processing collections of data using array methods like push, pop, map, filter.
- String Functions: Manipulating user input using methods like length, toUpperCase,

toLowerCase, substring.
- Date Functions: Working with JavaScript Date object for validation (e.g., age calculation, current date checks).
- React Form Handling: Using state hooks (useState) to manage input values and error messages.
- Event Handling: Validating on form submission or onChange events.

Procedure:

1. Set up a new React project using Create React App (CRA) or Vite.
2. Create a form component with input fields such as Name, Email, Password, and Date of Birth.
3. Use useState hooks to manage input values and errors.
4. Implement JavaScript validation functions to check email format, password rules, and date validation.
5. Use array methods to process multiple entries or form data lists.
6. Apply string methods to validate and transform input values (e.g., trimming spaces, converting cases).
7. Use JavaScript Date functions to validate date fields (e.g., ensuring user age is above 18).
8. Attach event handlers (onChange, onSubmit) to trigger validation functions.
9. Display validation error messages dynamically using conditional rendering.
10. Test the React app in a web browser and refine code as necessary.

Program:

Form.jsx

```
import React, { useState } from "react";
import "./Form.css"; // External CSS file (styles below)

const Form = () => {
  const [formData, setFormData] = useState({
    name: "",
    password: "",
    tel: "",
    address: "",
    age: "",
    gender: "",
    hobbies: [],
    technology: "Java",
    info: "",
  });

  const handleChange = (e) => {
    const { name, value, type, checked } = e.target;

    if (type === "checkbox") {
```

```
    setFormData((prev) => {
      const hobbies = checked
        ? [...prev.hobbies, value]
        : prev.hobbies.filter((hobby) => hobby !== value);
      return { ...prev, hobbies };
    });
  } else if (type === "radio") {
    setFormData((prev) => ({ ...prev, [name]: value }));
  } else {
    setFormData((prev) => ({ ...prev, [name]: value }));
  }
};

const handleSubmit = (e) => {
  e.preventDefault();
  const { name, password, tel, age } = formData;

  const nameRegex = /^[A-Za-z]+$/;
  if (!nameRegex.test(name)) {
    alert("Username should not contain numbers");
    return;
  }

  if (password.length < 8) {
    alert("Password should be at least 8 characters long");
    return;
  }

  if (age < 8 || age > 80) {
    alert("Age should be between 8 and 80");
    return;
  }

  const phoneRegex = /^[0-9]{10}$/;
  if (!phoneRegex.test(tel)) {
    alert("Phone number should contain exactly 10 digits");
    return;
  }

  const pname = prompt("Enter your name");
  if (pname !== null) {
    const conf = confirm("Do you want to submit?");
    if (conf) {
      alert("Confirm Successful");
    }
  }
};

const handleCancel = (e) => {
  e.preventDefault();
```

```
  const pname = prompt("Enter your name");
  if (pname !== null) {
    const conf = confirm("Do you want to cancel?");
    if (conf) {
      alert("Cancel Successful");
      setFormData({
        name: "",
        password: "",
        tel: "",
        address: "",
        age: "",
        gender: "",
        hobbies: [],
        technology: "Java",
        info: "",
      });
    }
  }
};

return (
  <div>
    <h1>Don Bosco Institute of Technology</h1>
    <h3>Department of Information Technology</h3>
    <h2>TE Form</h2>

    <form onSubmit={handleSubmit}>
      <label>Username :</label>
      <input
        type="text"
        name="name"
        placeholder="Enter your username"
        value={formData.name}
        onChange={handleChange}
        required
      />

      <label>Password :</label>
      <input
        type="password"
        name="password"
        placeholder="Enter your password"
        value={formData.password}
        onChange={handleChange}
        required
      />

      <label>Phone No. :</label>
      <input
        type="tel"
```

```
  name="tel"
  placeholder="Enter your phone number"
  value={formData.tel}
  onChange={handleChange}
  required
/>

<label>Address :</label>
<textarea
  name="address"
  placeholder="Enter your address"
  value={formData.address}
  onChange={handleChange}
/>

<label>Age :</label>
<input
  type="number"
  name="age"
  value={formData.age}
  onChange={handleChange}
  required
/>

<div className="inline-options">
  <label>Gender :</label>
  <input
    type="radio"
    name="gender"
    value="Male"
    checked={formData.gender === "Male"}
    onChange={handleChange}
  />{" "}
  Male
  <input
    type="radio"
    name="gender"
    value="Female"
    checked={formData.gender === "Female"}
    onChange={handleChange}
  />{" "}
  Female
</div>

<div className="inline-options">
  <label>Hobbies :</label>
  <input
    type="checkbox"
    name="hobby"
    value="Sports"
```

```
          checked={formData.hobbies.includes("Sports")}
          onChange={handleChange}
        />{" "}
        Sports
        <input
          type="checkbox"
          name="hobby"
          value="Reading"
          checked={formData.hobbies.includes("Reading")}
          onChange={handleChange}
        />{" "}
        Reading
        <input
          type="checkbox"
          name="hobby"
          value="Music"
          checked={formData.hobbies.includes("Music")}
          onChange={handleChange}
        />{" "}
        Music
      </div>

      <label>Technology :</label>
      <select
        name="technology"
        value={formData.technology}
        onChange={handleChange}
      >
        <option value="Java">Java</option>
        <option value="C">C</option>
        <option value="Python">Python</option>
      </select>

      <label>Personal Information :</label>
      <textarea
        name="info"
        placeholder="Enter your personal information"
        value={formData.info}
        onChange={handleChange}
      />

      <button type="submit">Submit</button>
      <button type="reset" onClick={handleCancel}>
        Cancel
      </button>
    </form>
  </div>
 );
};
```

export default Form;

Form.css

```
/* Make the entire page a centered flex container */
body {
  font-family: "Segoe UI", Arial, sans-serif;
  background: linear-gradient(135deg, #e3f2fd, #90caf9);
  margin: 0;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}

/* Wrapper to hold headings and form together */
.container {
  text-align: center;
  width: 100%;
  max-width: 600px;
}

/* Headings styling */
h1,
h2,
h3 {
  margin: 5px 0;
  color: #0d47a1;
}

h1 {
  font-size: 28px;
  font-weight: 700;
}

h2 {
  font-size: 22px;
  font-weight: 600;
}

h3 {
  font-size: 18px;
  font-weight: 500;
}

/* Form container */
form {
  margin-top: 15px;
```

```
  padding: 30px;
  background: #ffffff;
  border-radius: 16px;
  box-shadow: 0 6px 20px rgba(0, 0, 0, 0.1);
  text-align: left;
  transition: all 0.3s ease;
}

form:hover {
  transform: translateY(-3px);
}

/* Labels */
label {
  display: block;
  margin-bottom: 6px;
  font-weight: 600;
  color: #0d47a1;
}

/* Inputs, Textareas, Selects */
input[type="text"],
input[type="email"],
input[type="number"],
input[type="tel"],
input[type="password"],
textarea,
select {
  width: 100%;
  padding: 10px 12px;
  margin-bottom: 18px;
  border: 1px solid #cfd8dc;
  border-radius: 8px;
  font-size: 15px;
  color: #333;
  background-color: #fafafa;
  transition: 0.3s;
}

input:focus,
textarea:focus,
select:focus {
  outline: none;
  border-color: #2196f3;
  box-shadow: 0 0 6px rgba(33, 150, 243, 0.3);
  background-color: #fff;
}

/* Inline radio and checkbox groups */
.inline-options {
```

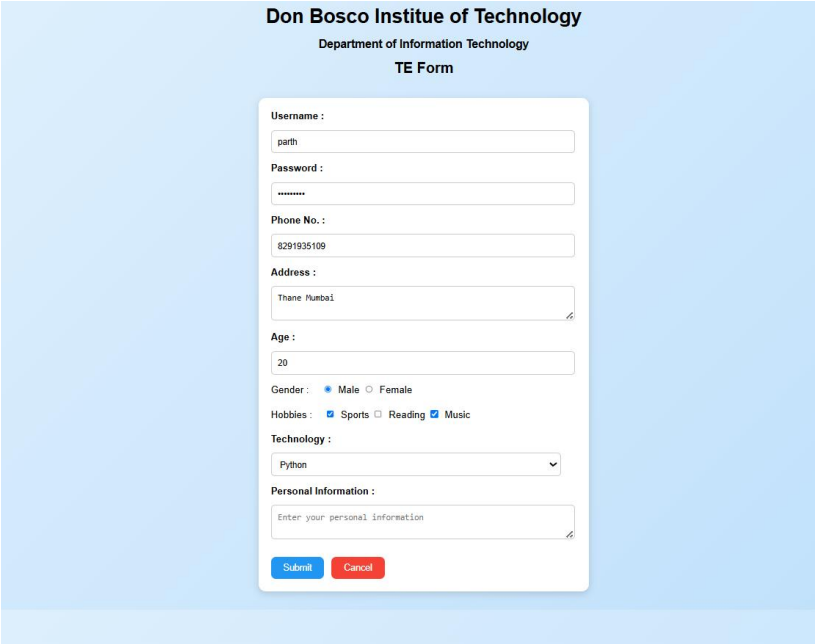
```
  margin-bottom: 18px;
}

.inline-options label {
  margin-right: 15px;
  font-weight: normal;
  color: #333;
}

.inline-options input {
  margin-right: 6px;
}

/* Buttons */
button {
  padding: 10px 22px;
  margin: 10px 10px 0 0;
  border: none;
  border-radius: 8px;
  font-size: 15px;
  cursor: pointer;
  transition: all 0.3s ease;
}

/* Submit button */
button[type="submit"] {
  background: #2196f3;
  color: white;
  font-weight: 600;
}

button[type="submit"]:hover {
  background: #1976d2;
  box-shadow: 0 3px 8px rgba(25, 118, 210, 0.3);
}

/* Reset button */
button[type="reset"] {
  background: #f44336;
  color: white;
  font-weight: 600;
}

button[type="reset"]:hover {
  background: #d32f2f;
  box-shadow: 0 3px 8px rgba(211, 47, 47, 0.3);
}

/* Textareas */
textarea {
```

```
  resize: vertical;
  min-height: 80px;
}

/* Responsive design */
@media (max-width: 600px) {
  .container {
    padding: 0 15px;
  }

  form {
    padding: 20px;
  }

  button {
    width: 100%;
    margin-bottom: 10px;
  }
}
```

App.jsx

```
import React from "react";
import Form from "./components/Form";

const App = () => {
  return <Form />;
};

export default App;
```

Output:



Results:

Successfully created a React-based web page implementing form validations using JavaScript validations, arrays, string methods, and date functions.

References:

1. https://react.dev/
2. https://developer.mozilla.org/en-US/docs/Web/JavaScript
3. https://www.w3schools.com/react/

Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Internet Programming

Name: Parth Pravin Shikhare

Class: TE-IT

Roll No.: 53

Date: 19/09/25

Experiment No.: 8

Title:

To Design a web page using React Props and State

Problem Definition:

To design a web page using React by implementing Props and State after installation and configuration of React JS, including JSX, Components, Props, and State concepts.

Pre-requisites:

1. Basic understanding of HTML, CSS, and JavaScript
2. Familiarity with ES6 features (let/const, arrow functions, import/export)
3. Node.js and npm installed
4. Basic knowledge of React project structure and JSX syntax

Theory:

React is a JavaScript library for building user interfaces using reusable components. It allows developers to create dynamic web applications efficiently. Key concepts involved in this experiment include:

- Installation & Configuration: React can be set up using Create React App (CRA) or Vite, enabling a ready development environment.
- JSX: JavaScript XML syntax allows HTML-like structures to be written inside JavaScript code.
- Components: The building blocks of React applications that can be functional or class-based.
- Props: Used to pass data from parent to child components for reusability.

- State: Represents mutable data within components. In functional components, it is managed using the useState hook.

Procedure:

1. Install Node.js and verify npm installation.
2. Create a new React project using Create React App or Vite.
3. Start the development server and open the project in a code editor.
4. Create functional or class-based components.
5. Pass data between components using Props.
6. Use the useState hook to manage and update State.
7. Render data dynamically using JSX expressions.
8. Use event handlers to update State and trigger re-renders.
9. Test the web page in a browser and verify the functionality.

Program:

UserCard.jsx

```
import "./UserCard.css";

function UserCard({ name, age, location }) {
 return (
  <div className="user-card">
   <div className="avatar">
    <span>{name.charAt(0)}</span>
   </div>
   <h2 className="user-name">{name}</h2>
   <p className="user-detail">
    <strong>Age:</strong> {age}
   </p>
   <p className="user-detail">
    <strong>Location:</strong> {location}
   </p>
  </div>
 );
}

export default UserCard;
```

UserCard.css

```
.user-card {
 background: linear-gradient(145deg, #ffffff, #f3f3f3);
 border-radius: 16px;
 width: 230px;
 padding: 25px 20px;
 box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
 text-align: center;
```

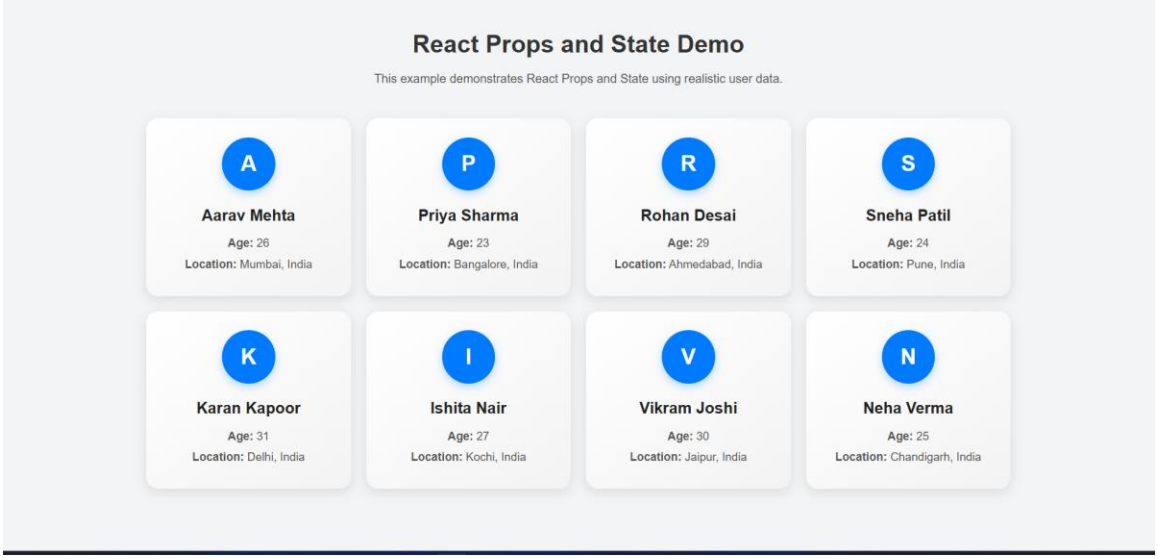
```
  transition: all 0.3s ease;
  cursor: pointer;
}

.user-card:hover {
  transform: translateY(-8px);
  box-shadow: 0 10px 25px rgba(0, 0, 0, 0.15);
  background: linear-gradient(145deg, #f0f4ff, #ffffff);
}

.avatar {
  background-color: #007bff;
  color: white;
  font-size: 1.8rem;
  font-weight: bold;
  width: 70px;
  height: 70px;
  margin: 0 auto 15px auto;
  border-radius: 50%;
  display: flex;
  align-items: center;
  justify-content: center;
  box-shadow: 0 4px 10px rgba(0, 123, 255, 0.3);
}

.user-name {
  font-size: 1.3rem;
  color: #222;
  margin-bottom: 10px;
  font-weight: 600;
}

.user-detail {
  color: #555;
  font-size: 0.95rem;
  margin: 5px 0;
}

@media (max-width: 600px) {
  .user-card {
    width: 85%;
  }
}
```

App.jsx

```
import { useState } from "react";
import UserCard from "./components/UserCard";
import "./App.css";
```

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

  const users = [
    { name: "Aarav Mehta", age: 26, location: "Mumbai, India" },
    { name: "Priya Sharma", age: 23, location: "Bangalore, India" },
    { name: "Rohan Desai", age: 29, location: "Ahmedabad, India" },
    { name: "Sneha Patil", age: 24, location: "Pune, India" },
    { name: "Karan Kapoor", age: 31, location: "Delhi, India" },
    { name: "Ishita Nair", age: 27, location: "Kochi, India" },
    { name: "Vikram Joshi", age: 30, location: "Jaipur, India" },
    { name: "Neha Verma", age: 25, location: "Chandigarh, India" },
  ];

  return (
    <div className="app-container">
      <h1>React Props and State Demo</h1>
      <p className="description">
        This example demonstrates React Props and State using realistic user data.
      </p>

      <div className="cards-container">
        {users.map((user, index) => (
          <UserCard
            key={index}
            name={user.name}
            age={user.age}
            location={user.location}
          />
        ))}
      </div>
    </div>
  );
}

export default App;
```

Output:



Results:

Successfully designed a web page using React JS by implementing Props and State with proper component structuring.

References:

1. https://react.dev/
2. https://www.w3schools.com/react/
3. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

Don Bosco Institute of Technology
Kurla (W), Mumbai 400 070

Internet Programming

Name: Parth Pravin Shikhare

Class: TE-IT

Roll No.: 53

Date: 26/09/25

Experiment No.: 9

Title:

To Design a web page using ReactJS Hooks

Problem Definition:

Create a web page using ReactJS implementing Hooks with functionalities involving ReactJS Forms, Events, Routers, Refs, and Keys.

Pre-requisites:

1. Basic knowledge of ReactJS components, JSX, and props
2. Understanding of functional components in React
3. Node.js and npm installed for React environment
4. Familiarity with ES6 features and basic DOM manipulation
5. Basic knowledge of React Router and React Hooks

Theory:

React Hooks are special functions that let you 'hook into' React features such as state and lifecycle methods from function components. Hooks allow writing stateful logic without using classes. Common hooks include useState, useEffect, useRef, useContext, and useReducer.

• useState: Manages state in functional components.
• useEffect: Handles side effects such as fetching data, subscriptions, or manual DOM manipulations.
• useRef: Creates a reference to DOM elements or mutable variables.
• useContext: Accesses context values directly without using props.
• useReducer: Manages complex state logic similar to Redux.

React Router allows for navigation between different components or pages. Refs are used to directly access DOM nodes, while Keys help React identify elements during rendering for optimized updates.

Procedure:

1. Install Node.js and verify npm installation.
2. Create a new React project using Create React App.
3. Install React Router DOM using npm.
4. Create different components for pages and navigation.
5. Implement useState and useEffect for managing state and side effects.
6. Use useRef to access form elements directly.
7. Implement form validation and event handling.
8. Use Router, Route, and Link for navigation between components.
9. Test the web page in a browser to ensure correct functionality.
10. Document the output and results.

Program:

About.jsx

```
export default function About() {
  return (
    <div className="p-6">
      <h2 className="text-2xl font-semibold">About</h2>
      <p className="mt-2 text-gray-700">
        Small demo app to teach common React patterns.
      </p>
    </div>
  );
}
```

Nav.jsx

```
import { Link } from "react-router-dom";

export default function Nav() {
  return (
    <nav className="p-4 bg-gradient-to-r from-sky-500 to-indigo-600 text-white flex gap-
4">
      <Link to="/" className="font-semibold hover:underline">Home</Link>
      <Link to="/form" className="font-semibold hover:underline">Form</Link>
      <Link to="/list" className="font-semibold hover:underline">List (Keys)</Link>
      <Link to="/refs" className="font-semibold hover:underline">Refs</Link>
      <Link to="/about" className="font-semibold hover:underline">About</Link>
    </nav>
  );
}
```

Home.jsx

```
export default function Home() {
 return (
  <div className="p-6">
   <h1 className="text-3xl font-bold mb-2">React Hooks & Features Demo</h1>
   <p className="text-gray-700">
    This sample app demonstrates hooks, forms, events, router, refs, and list keys.
   </p>
  </div>
 );
}
```

FormPage.jsx

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";

export default function FormPage() {
 const [form, setForm] = useState({ name: "", email: "", age: "" });
 const [errors, setErrors] = useState({});
 const [submittedData, setSubmittedData] = useState(null);
 const navigate = useNavigate();

 function validate(values) {
  const e = {};
  if (!values.name.trim()) e.name = "Name is required";
  if (!values.email.includes("@")) e.email = "Must be a valid email";
  const ageNum = Number(values.age);
  if (!values.age || Number.isNaN(ageNum) || ageNum <= 0)
   e.age = "Enter a valid age";
  return e;
 }

 function handleChange(e) {
  const { name, value } = e.target;
  setForm((prev) => ({ ...prev, [name]: value }));
 }

 function handleSubmit(e) {
  e.preventDefault();
  const v = validate(form);
  setErrors(v);
  if (Object.keys(v).length === 0) {
   setSubmittedData(form);
   setTimeout(() => navigate("/list"), 500);
  }
 }
```

```
function handleReset() {
  setForm({ name: "", email: "", age: "" });
  setErrors({});
  setSubmittedData(null);
}

return (
  <div className="p-6 max-w-xl">
    <h2 className="text-2xl font-semibold mb-4">Controlled Form (Hooks + Events)</h2>
    <form onSubmit={handleSubmit} onReset={handleReset} className="space-y-4">
      <div>
        <label className="block font-medium">Name</label>
        <input
          name="name"
          value={form.name}
          onChange={handleChange}
          className="mt-1 p-2 border rounded w-full"
          placeholder="Your name"
        />
        {errors.name && <div className="text-red-600 mt-1">{errors.name}</div>}
      </div>

      <div>
        <label className="block font-medium">Email</label>
        <input
          name="email"
          value={form.email}
          onChange={handleChange}
          className="mt-1 p-2 border rounded w-full"
          placeholder="you@example.com"
        />
        {errors.email && <div className="text-red-600 mt-1">{errors.email}</div>}
      </div>

      <div>
        <label className="block font-medium">Age</label>
        <input
          name="age"
          value={form.age}
          onChange={handleChange}
          className="mt-1 p-2 border rounded w-full"
          placeholder="Age"
          type="number"
        />
        {errors.age && <div className="text-red-600 mt-1">{errors.age}</div>}
      </div>

      <div className="flex gap-2">
        <button type="submit" className="px-4 py-2 bg-indigo-600 text-white rounded">
          Submit
```

```
        </button>
        <button type="reset" className="px-4 py-2 bg-gray-200 rounded">
          Reset
        </button>
      </div>
    </form>

    {submittedData && (
      <div className="mt-6 p-4 border rounded bg-green-50">
        <h3 className="font-semibold">Submitted</h3>
        <pre className="text-sm">{JSON.stringify(submittedData, null, 2)}</pre>
      </div>
    )}
  </div>
 );
}
```

ListPage.jsx

```
import { useState } from "react";
import { Link, Routes, Route } from "react-router-dom";
import ListItemDetail from "./ListItemDetail";

let idCounter = 2;

export default function ListPage() {
 const [items, setItems] = useState([
  { id: 1, text: "Learn React" },
  { id: 2, text: "Build Projects" },
 ]);
 const [text, setText] = useState("");

 function addItem() {
  if (!text.trim()) return;
  setItems([{ id: ++idCounter, text }, ...items]);
  setText("");
 }

 function removeItem(id) {
  setItems(items.filter((i) => i.id !== id));
 }

 return (
  <div className="p-6 max-w-2xl">
   <h2 className="text-2xl font-semibold mb-4">List & Keys</h2>
   <div className="flex gap-2">
    <input
     value={text}
     onChange={(e) => setText(e.target.value)}
     className="p-2 border rounded flex-1"
```

```
        placeholder="Add new item"
       />
       <button onClick={addItem} className="px-4 py-2 bg-sky-600 text-white rounded">
        Add
       </button>
      </div>

      <ul className="mt-4 space-y-2">
       {items.map((item) => (
        <li key={item.id} className="flex justify-between items-center p-3 border rounded">
         <Link to={`/list/${item.id}`} className="font-medium hover:underline">
          {item.text}
         </Link>
         <button
          onClick={() => removeItem(item.id)}
          className="px-2 py-1 bg-red-500 text-white rounded"
         >
          Remove
         </button>
        </li>
       ))}
      </ul>

      <Routes>
       <Route path=":id" element={<ListItemDetail items={items} />} />
      </Routes>
    </div>
  );
}
```

ListItemDetail.jsx

```
import { useParams } from "react-router-dom";

export default function ListItemDetail({ items }) {
  const { id } = useParams();
  const item = items.find((it) => String(it.id) === id);

  if (!item) return <div className="p-4">Item not found.</div>;

  return (
    <div className="mt-4 p-4 border rounded bg-gray-50">
      <h3 className="font-semibold">Item detail</h3>
      <p>ID: {item.id}</p>
      <p>Text: {item.text}</p>
    </div>
  );
}
```

RefsPage.jsx

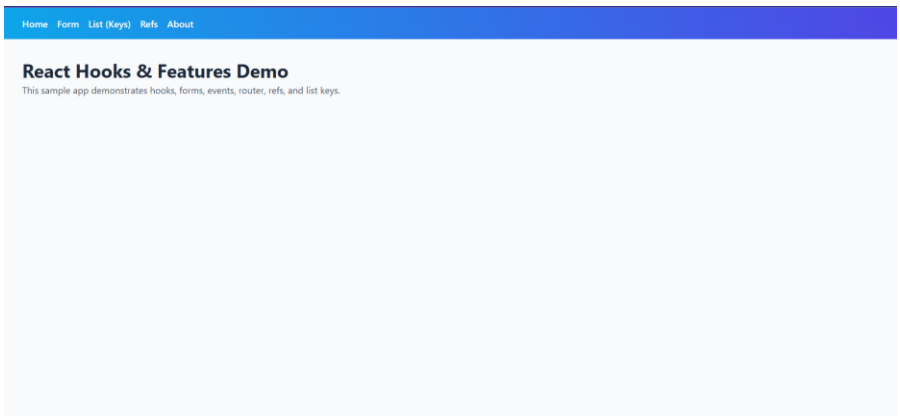
```
import { useRef, useState, useEffect } from "react";

export default function RefsPage() {
 const inputRef = useRef(null);
 const [value, setValue] = useState("");
 const [log, setLog] = useState([]);

 useEffect(() => {
  inputRef.current?.focus();
 }, []);

 function handleRead() {
  const v = inputRef.current?.value || "";
  setLog((prev) => [`Read via ref: "${v}"`, ...prev].slice(0, 10));
 }

 function handleUpdateState() {
  setValue(inputRef.current?.value || "");
 }

 return (
  <div className="p-6 max-w-xl">
   <h2 className="text-2xl font-semibold mb-4">Refs Demo</h2>
   <input
    ref={inputRef}
    className="p-2 border rounded w-full"
    placeholder="Type something..."
    defaultValue={value}
   />
   <div className="mt-3 flex gap-2">
    <button onClick={handleRead} className="px-4 py-2 bg-indigo-600 text-white
rounded">
     Read via ref
    </button>
    <button onClick={handleUpdateState} className="px-4 py-2 bg-gray-200 rounded">
     Sync to state
    </button>
   </div>

   <div className="mt-4">
    <h4 className="font-medium">State value</h4>
    <div className="p-2 border rounded mt-1">{value || <span className="text-gray-
400">(empty)</span>}</div>
   </div>

   <div className="mt-4">
    <h4 className="font-medium">Action log</h4>
```

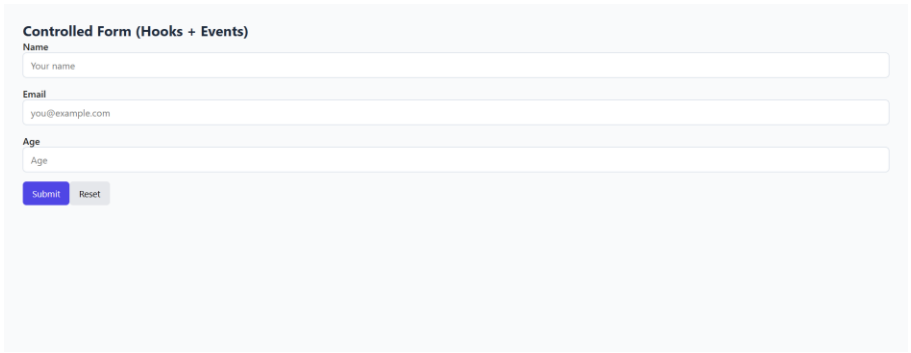
```
      <ul className="mt-2 list-disc pl-5 space-y-1">
       {log.map((l, i) => (
        <li key={i}>{l}</li>
       ))}
      </ul>
     </div>
    </div>
  );
}
```

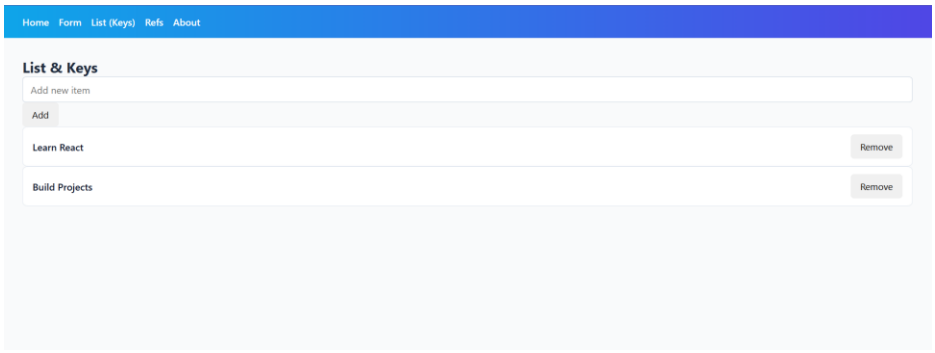
App.jsx

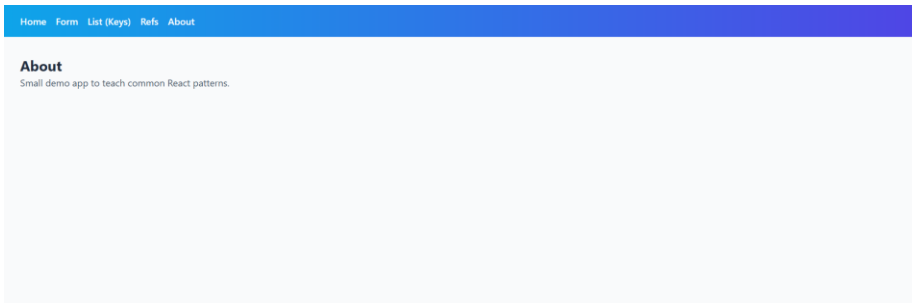
```
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Nav from "./components/Nav";
import Home from "./components/Home";
import About from "./components/About";
import FormPage from "./components/FormPage";
import ListPage from "./components/ListPage";
import RefsPage from "./components/RefsPage";

export default function App() {
  return (
    <Router>
     <div className="min-h-screen bg-gray-50">
      <Nav />
      <main>
       <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/form" element={<FormPage />} />
        <Route path="/list/*" element={<ListPage />} />
        <Route path="/refs" element={<RefsPage />} />
        <Route path="/about" element={<About />} />
        <Route path="*" element={<div className="p-6">Page not found</div>} />
       </Routes>
      </main>
     </div>
    </Router>
  );
}
```

Output:


Home Form List (Keys) Refs About
React Hooks & Features Demo
This sample app demonstrates hooks, forms, events, router, refs, and list keys.


Controlled Form (Hooks + Events)
Name
Your name
Email
you@example.com
Age
Age
Submit
Reset


Home Form List (Keys) Refs About
List & Keys
Add new item
Add
Learn React
Remove
Build Projects
Remove



Result:

Successfully designed and implemented a web page using ReactJS Hooks, including Forms, Events, Routers, Refs, and Keys.

References:

1. https://react.dev/
2. https://reactrouter.com/
3. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Internet Programming

Name: Parth Pravin Shikhare

Class: TE-IT

Roll No.: 53

Date: 03/10/25

Experiment No.: 10

Title:

Case Study - Express JS

Problem Definition:

To study Express JS and understand its core concepts including Express Router, REST API, Generator, Authentication, Session, and Integration.

Pre-requisites:

1. Basic understanding of JavaScript and Node.js
2. Knowledge of web servers and HTTP protocols
3. Familiarity with JSON and REST architecture
4. Node.js and npm installed on the system

Theory:

Express JS is a fast, unopinionated, and minimalist web framework for Node.js that simplifies building web and API servers. It provides a robust set of features for handling routes, middleware, sessions, and more.

Key Concepts:
- Express JS: Framework for building web applications and APIs in Node.js using simple, flexible routing.
- Express Router: A modular way to handle multiple routes within an application, promoting cleaner code organization.
- REST API: A standardized method of designing web services using HTTP methods like GET, POST, PUT, and DELETE.
- Express Generator: A command-line tool used to quickly scaffold an Express project structure.
- Authentication: A security process for verifying user credentials using libraries like Passport.js or JWT.
- Session: Used to maintain user data across different requests using packages like express-

session.
- Integration: Express easily integrates with databases (MongoDB, MySQL) and frontend frameworks like React, Vue, or Angular.

Procedure:

1. Install Node.js and npm.
2. Install Express globally using `npm install -g express-generator`.
3. Create a new project using `npx express-generator` and install dependencies.
4. Set up Express Router to define routes for different endpoints.
5. Develop RESTful APIs to handle CRUD operations.
6. Implement authentication using JWT or Passport.js.
7. Configure session management using express-session.
8. Integrate Express with a database such as MongoDB or MySQL.
9. Test routes and APIs using Postman or a similar tool.
10. Deploy the final project on a cloud platform like Render, Vercel, or Heroku.

Program:

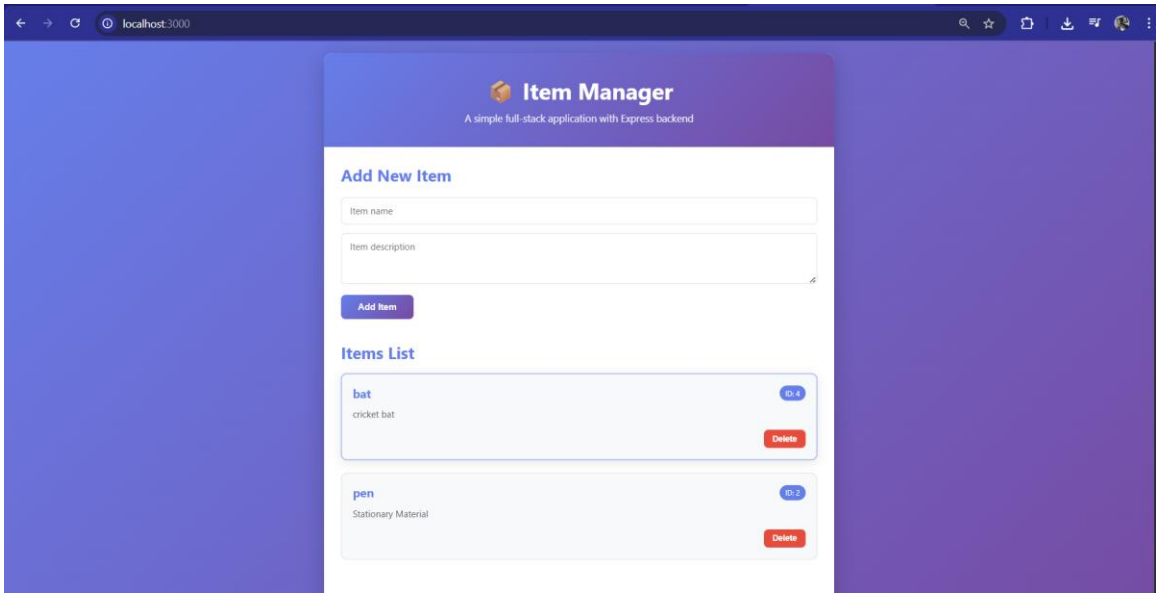
```
const API_URL = 'http://localhost:3000/api';

// Load items when page loads
document.addEventListener('DOMContentLoaded', () => {
  loadItems();
  setupFormHandler();
});

// Load all items from API
async function loadItems() {
  const itemsList = document.getElementById('itemsList');

  try {
    const response = await fetch(`${API_URL}/items`);
    const result = await response.json();

    if (result.success && result.data.length > 0) {
      displayItems(result.data);
    } else {
      itemsList.innerHTML = '<p class="loading">No items found. Add one above!</p>';
    }
  } catch (error) {
    console.error('Error loading items:', error);
    itemsList.innerHTML = '<p class="error">Failed to load items. Make sure the server is
running.</p>';
  }
}

// Display items in the UI
function displayItems(items) {
  const itemsList = document.getElementById('itemsList');
```

```
    itemsList.innerHTML = items.map(item => `
        <div class="item-card" data-id="${item.id}">
            <div class="item-header">
                <span class="item-name">${escapeHtml(item.name)}</span>
                <span class="item-id">ID: ${item.id}</span>
            </div>
            <p class="item-description">${escapeHtml(item.description)}</p>
            <div class="item-actions">
                <button class="btn btn-danger" onclick="deleteItem(${item.id})">Delete</button>
            </div>
        </div>
    `).join('');
}

// Setup form submission handler
function setupFormHandler() {
    const form = document.getElementById('addItemForm');

    form.addEventListener('submit', async (e) => {
        e.preventDefault();

        const name = document.getElementById('itemName').value;
        const description = document.getElementById('itemDescription').value;

        await addItem(name, description);

        // Clear form
        form.reset();
    });
}

// Add new item via API
async function addItem(name, description) {
    try {
        const response = await fetch(`${API_URL}/items`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ name, description })
        });

        const result = await response.json();

        if (result.success) {
            showMessage('Item added successfully!', 'success');
            loadItems(); // Reload items list
        } else {
            showMessage('Failed to add item', 'error');
```

```
        }
    } catch (error) {
        console.error('Error adding item:', error);
        showMessage('Error adding item. Check console for details.', 'error');
    }
}

// Delete item via API
async function deleteItem(id) {
    if (!confirm('Are you sure you want to delete this item?')) {
        return;
    }

    try {
        const response = await fetch(`${API_URL}/items/${id}`, {
            method: 'DELETE'
        });

        const result = await response.json();

        if (result.success) {
            showMessage('Item deleted successfully!', 'success');
            loadItems(); // Reload items list
        } else {
            showMessage('Failed to delete item', 'error');
        }
    } catch (error) {
        console.error('Error deleting item:', error);
        showMessage('Error deleting item. Check console for details.', 'error');
    }
}

// Show temporary message
function showMessage(message, type) {
    const itemsList = document.getElementById('itemsList');
    const messageDiv = document.createElement('div');
    messageDiv.className = type;
    messageDiv.textContent = message;

    itemsList.insertBefore(messageDiv, itemsList.firstChild);

    setTimeout(() => {
        messageDiv.remove();
    }, 3000);
}

// Escape HTML to prevent XSS
function escapeHtml(text) {
    const div = document.createElement('div');
    div.textContent = text;
```

```
    return div.innerHTML;
}
```

Output:



Results:

Successfully completed a case study on Express JS covering concepts such as Express Router, REST API, Generator, Authentication, Session, and Integration.

References:

1. https://expressjs.com/
2. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs
3. https://www.w3schools.com/nodejs/nodejs_express.asp
4. https://www.npmjs.com/package/express