

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Internet Programming

Name:

Class: IT SEM V

Roll no.:

Submission Date:

Experiment no. 1

Title: Design a Basic Web page using HTML 5

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

Pre-requisites: Knowledge of basic HTML and programming.

Theory:

What is HTML?

- HTML is a language for describing web pages
- HTML stands for Hyper Text Markup Language
- HTML is a markup language
- A markup language is a set of markup tags
- The tags describe document content
- HTML documents contain HTML tags and plain text
- HTML documents are also called web pages

HTML tags

- HTML markup tags are usually called HTML tags
- HTML tags are keywords (tag names) surrounded by angle brackets like <html>
- HTML tags normally come in pairs like and
- The first tag in a pair is the start tag, the second tag is the end tag
- The end tag is written like the start tag, with a forward slash before the tag name Start and end tags are also called opening tags and closing tags

HTML Tags

- The text between <html> and </html> describes the web page
- The text between <body> and </body> is the visible page content
- The text between <h1> and </h1> is displayed as a heading
- The text between <p> and </p> is displayed as a paragraph

HTML Headings

- HTML headings are defined with the <h1> to <h6> tags.
- Examples:
`<h1>This is a heading</h1>`
`<h2>This is a heading</h2>`
`<h3>This is a heading</h3>`

HTML Paragraphs

- HTML paragraphs are defined with the <p> tag.
- Examples:
`<p>This is a paragraph.</p>`
`<p>This is another paragraph.</p>`

HTML Links

- HTML links are defined with the <a> tag.
- Example:
`This is a link`

HTML Images

- HTML images are defined with the tag.
- Example
``

HTML Lines

- The <hr>tag creates a horizontal line in an HTML page.
- The hr element can be used to separate content:
- Examples:
`<p>This is a paragraph.</p>`

`<hr>`

`<p>This is a paragraph.</p>`

`<hr>`

`<p>This is a paragraph.</p>`

HTML Comments

- Comments can be inserted into the HTML code to make it more readable and understandable. Comments are ignored by the browser and are not displayed.
- Comments are written like this: `<!-- This is a comment -->`

HTML Text Formatting

`<html>`

`<body>`

`<p>This text is bold</p>`

`<p>This text is strong</p>`

`<p><i>This text is italic</i></p>`

`<p>This text is emphasized</p>`

`<p><code>This is computer output</code></p>`

`<p>This is_{subscript} and ^{superscript}</p>`

`</body>`

`</html>`

Output:

This text is bold

This text is italic

This is computer output

This is subscript and superscript

Program:

portfolio.html file

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Portfolio Site</title>

  <link href="normalize.css" rel="stylesheet">

  <link
href="https://fonts.googleapis.com/css2?family=Days+One&family=Work+Sans&display=swap"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
  <link rel="stylesheet" type="text/css" href="portfolio.css">

</head>
<body>

  <nav>
    <ul>
      <li><a href="#about">About</a></li>
      <li><a href="#services">Services</a></li>
      <li><a href="#sem">Sem 5</a></li>

    </ul>
  </nav>

  <section class="hero">
    <div class="container">
      
      <h1>Hi! I'm Srini!</h1>
    </div>
  </section>

  <section id="aboutme" class="aboutme">
    <h2>About me</h2>
```

```

<div class="aboutme-section">
  <div class="img">
    <div class="aboutme-image"></div>
  </div>
  <div class="aboutme-text">
    <p>I am an enthusiastic, self-motivated, reliable, responsible and hard
working person. I am a mature team worker and adaptable to all challenging situations. I
am able to work well both in a team environment as well
as using own initiative. I am able to work well under pressure and adhere
to strict deadlines.</p>

    <p>I am familiar with HTML5, CSS3, JavaScript, PHP7 and MySQL.</p>

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

<section id="abt">
  <div class="aboutme">
    <h2>More about me</h2>
    <div class="abt">

      <div class="acads">
        <div class="text">
          <h3 class="acads-name">Academics</h3>
          <div class="content">
            <p>I am an undergrad in Information Technology Department</p>
          </div>

          <a class="button" href="Academics/index.html"
target="_bank">View</a>
        </div>

      </div>

      <div class="sports">
        <div class="text">
          <h3 class="sports-name">Sports</h3>
          <div class="content">
            <p>I love playing sports</p>
          </div>

          <a class="button" href="Sports/index.html" target="_bank">View</a>
        </div>

```

```

    </div>

    <div>
      <iframe id= "hobby"
        title= "Hobby"
        width= "500"
        height= "500"
        src= "https://www.w3schools.com " title= "W 3Schools Free O nline
Web Tutorials">
      </iframe>
    </div>

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

<section id= "contact" m e" class= "contact">
  <h2> Contact</h2>
  <p> Use this fom  to get in touch .I would love to hear from you!</p>

  <form class= "contact-form ">

    <div class= "contact-details">

      <label for= "name"> Nam e</label>
      <input type= "text" id= "youname" placeholder= "Yourname">

      <label for= "subject"> Subject</label>
      <input type= "text" id= "subject" placeholder= "Reason for your message">

      <label for= "email"> Em ailAddress</label>
      <input type= "email" id= "email" placeholder= "Your email address">

    </div>

    <div class= "message">
      <label for= "msg"> M essage</label>
      <textarea id= "msg " rows= "15"></textarea>
      <button type= "submit"> Subm it</button>
    </div>

  </form>

```

```
< /section>

< footer>
  < div class= "container">
    < div class= "copyright">
      This site & copy; 2023 Sraïa Panchangam
    < /div>

  < /div>

< /footer>

< /body>
< /html>
```

Output:

References:

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/html/?ref=lbp>

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Internet Programming

Name:

Class: IT SEM V

Roll no.:

Submission Date:

Experiment no. 2

Title: Design a Basic Web page using advance HTML 5

Problem Definition: Design a web pages using advance HTML5, List, Frames, Forms, Multimedia tags of HTML5

Pre-requisites: Knowledge of basic HTML and programming.

Theory:

- HTML List:

HTML Unordered Lists: An unordered list starts with the tag. Each list item starts with the tag. The list items are marked with bullets (typically small black circles).

Eg: Coffee Milk

How the HTML code above looks in a browser: Coffee Milk

HTML Ordered Lists: An ordered list starts with the tag. Each list item starts with the tag. The list items are marked with numbers.

Example: Coffee Milk

How the HTML code above looks in a browser:

1. Coffee

2. Milk

HTML Definition Lists: A definition list is a list of items, with a description of each item.

The <dl> tag defines a definition list.

The <dl> tag is used in conjunction with <dt> (defines the item in the list) and <dd> (describes the item in the list):

```
<dl> <dt>Coffee</dt> <dd>- black hot drink</dd> <dt>Milk</dt> <dd>- white cold drink</dd> </dl>
```

How the HTML code above looks in a browser:

Coffee

- black hot drink

Milk

- white cold drink

HTML Forms

HTML forms are used to pass data to a server.

An HTML form can contain input elements like text fields, checkboxes, radio-buttons, submit

buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label

elements.

The <form> tag is used to create an HTML form:

```
<form>
```

.

input elements

.

```
</form>
```

- HTML Forms - The Input Element

The most important form element is the `<input>` element. The `<input>` element is used to select user information. An `<input>` element can vary in many ways, depending on the type attribute. An `<input>` element can be of type text field, checkbox, password, radio button, submit button, and more. The most common input types are described below.

1. Text Fields

`<input type="text">` defines a one-line input field that a user can enter text into:

`<form>`

First name: `<input type="text" name="firstname">``
`

Last name: `<input type="text" name="lastname">`

`</form>`

How the HTML code above looks in a browser:

First name:

Last name:

The form itself is not visible. Also note that the default width of a text field is 20 characters.

2. Password Field

`<input type="password">` defines a password field:

`<form>`

Password: `<input type="password" name="pwd">`

`</form>`

How the HTML code above looks in a browser:

Password: The characters in a password field are masked (shown as asterisks or circles).

3. Radio Buttons

<input type="radio"> defines a radio button. Radio buttons let a user select ONLY ONE of a limited number of choices:

```
<form>
```

```
<input type="radio" name="sex" value="male">Male<br>
```

```
<input type="radio" name="sex" value="female">Female
```

```
</form>
```

How the HTML code above looks in a browser:

Male

Female

4. Checkboxes

<input type="checkbox"> defines a checkbox. Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<form>
```

```
<input type="checkbox" name="vehicle" value="Bike">I have a bike<br>
```

```
<input type="checkbox" name="vehicle" value="Car">I have a car
```

```
</form>
```

How the HTML code above looks in a browser:

I have a bike

I have a car

5. Submit Button

<input type="submit"> defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

```
<form name="input" action="html_form_action.asp" method="get">
```

```
Username: <input type="text" name="user">
```

```
<input type="submit" value="Submit">
```

```
</form>
```

How the HTML code above looks in a browser:

Username:

If you type some characters in the text field above, and click the "Submit" button, the browser will send your input to a page called "html_form_action.asp". The page will show you the received input.

- HTML Iframes

An iframe is used to display a web page within a web page.

Syntax for adding an iframe:

```
<iframe src="URL"></iframe>
```

The URL points to the location of the separate page.

Iframe - Set Height and Width

The height and width attributes are used to specify the height and width of the iframe. The attribute values are specified in pixels by default, but they can also be in percent (like "80%").

Example

```
<iframe src="demo_iframe.htm" width="200" height="200"></iframe>
```

Iframe - Remove the Border

The frameborder attribute specifies whether to display a border around the iframe.

Set the attribute value to "0" to remove the border:

Example

```
<iframe src="demo_iframe.htm" frameborder="0"></iframe>
```

Use iframe as a Target for a Link

An iframe can be used as the target frame for a link.

The target attribute of a link must refer to the name attribute of the iframe:

Example

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe> <p><a  
href="http://www.w3schools.com"  
target="iframe_a">W3Schools.com</a></p>
```

- HTML Multi-Media Tags

HTML allows adding different multimedia files on your website by various multimedia tags. These tags include:

<audio> for displaying a audio file on the web page,

<video> for displaying a video file on the web page,

<embed> for embedding multimedia files on the web page,

<object> for embedding multimedia files on the web page.

<iframe> for embedding other web pages.

Program:

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta http-equiv="X-UA-Compatible" content="IE=edge">  
  <meta name="viewport" content="width=device-width, initial-  
scale=1.0">  
  <title>Portfolio Site</title>  
  
  <link href="normalize.css" rel="stylesheet">  
  
  <link  
href="https://fonts.googleapis.com/css2?family=Days+One&family=Work+Sans&display=swap" rel="stylesheet">  
  <link rel="stylesheet"  
href="https://cdnjs.cloudflare.com/ajax/libs/font-  
awesome/4.7.0/css/font-awesome.min.css">
```

```

< link rel= "stylesheet" type= "text/css" href= "portfolio.css">

< /head>

< body>

    < nav>
        < ul>
            < li> < a href= "# db_it"> DB II< /a> < /li>
            < li> < a href= "# te_it"> TE II< /a> < /li>
            < li> < a href= "# sem "> Sem 5< /a> < /li>

        < /ul>
    < /nav>

    < section class= "hero">
        < div class= "container">
            < in g src= "images/pfp.jpg">
            < h1> Hi! I'm Srina!< /h1>
        < /div>
    < /section>

    < section id= "aboutme" class= "aboutme">
        < h2> Aboutme< /h2>
        < div class= "aboutme-section">
            < div class= "in g">
                < div class= "aboutme-in age"> < /div>
            < /div>
            < div class= "aboutme-text">
                < p> Iam an enthusiastic, self-m otivated, reliable,
                responsible and hard working person. Iam a mature team worker and
                adaptable to all challenging situations. Iam able to work well both
                in a team environment as well
                as using own initiative. Iam able to work well under
                pressure and adhere to strict deadlines.< /p>

                < p> Im fam iliar w ith HTML5 , CSS3 , JavaScript, PHP7
                and MySQL .< /p>

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

```



```
< /footer>
```

```
< /body>
```

```
< /html>
```

Output:

References:

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/html/?ref=lp>

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Internet Programming

Name:

Class: IT SEM V

Roll no.:

Submission Date:

Experiment no. 3

Title: Design a Basic Web page using advance HTML 5

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

Pre-requisites: Knowledge of basic HTML and programming.

Theory:

➤ What is CSS?

- CSS stands for Cascading Style Sheets Styles define how to display
- HTML elements Styles were added to HTML 4.0 to solve a problem
- External Style Sheets can save a lot of work External Style Sheets are stored in CSS files

➤ CSS Background

- CSS background properties are used to define the background effects of element.
- CSS properties used for background effects: background-color background-image background-repeat background-attachment background-position

- Background Color:

- The background-color property specifies the background color of an element.
- The background color of a page is defined in the body selector:

Example:

```
body {background-color:#b0c4de;}
```

With CSS, a color is most often specified by: a HEX value - like "#ff0000" an RGB value - like "rgb(255,0,0)" a color name - like "red"

In the example below, the h1, p, and div elements have different background colors:

Example:

```
h1 {background-color:#6495ed;} p {background-color:#e0ffff;} div  
{background-color:#b0c4de;}
```

- Background Image
 - The background-image property specifies an image to use as the background of an element.
 - By default, the image is repeated so it covers the entire element.
 - The background image for a page can be set like this:
 - Example:

```
body {background-image:url('paper.gif');}
```
 - Below is an example of a bad combination of text and background image. The text is almost not readable:
 - Example:

```
body {background-image:url('bgdesert.jpg');}
```

➤ CSS Font

- CSS Font Families
 - In CSS, there are two types of font family names:
 - generic family - a group of font families with a similar look (like "Serif" or "Monospace")
 - font family - a specific font family (like "Times New Roman" or "Arial")
- Font Family
 - The font family of a text is set with the font-family property.
 - The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font.
 - Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.
 - If the name of a font family is more than one word, it must be in quotation marks, like font-family: "Times New Roman".
 - More than one font family is specified in a comma-separated list:
Example:

```
p{font-family:"Times New Roman", Times, serif;}
```
- Font Style
 - The font-style property is mostly used to specify italic text.
 - This property has three values: normal - The text is shown normally
italic - The text is shown in italics
oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Example:

```
p.normal {font-style:normal;} p.italic {font-style:italic;}  
p.oblique {font-style:oblique;}
```

- Font Size

- The font-size property sets the size of the text.
- Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.
- Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
- The font-size value can be an absolute, or relative size.
- Absolute size: Sets the text to a specified size Does not allow a user to change the text size in all browsers (bad for accessibility reasons) Absolute size is useful when the physical size of the output is known
- Relative size: Sets the size relative to surrounding elements Allows a user to change the text size in browsers
- If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).
- Set Font Size With Pixels
- Setting the text size with pixels gives you full control over the text size:

Example:

```
h1 {font-size:40px;} h2 {font-size:30px;} p {font-size:14px;}
```

- CSS Lists

- The CSS list properties allow you to: Set different list item markers for ordered lists Set different list item markers for unordered lists Set an image as the list item marker
- List:
- In HTML, there are two types of lists: unordered lists - the list items are marked with bullets ordered lists - the list items are marked with numbers or letters
- With CSS, lists can be styled further, and images can be used as the list item marker.
- Different List Item Markers
- The type of list item marker is specified with the list-style-type property:

Example

```
ul.a {list-style-type: circle;} ul.b {list-style-type: square;} ol.c {list-style-type: upper-roman;} ol.d {list-style-type: lower-alpha;}
```

- An Image as The List Item Marker
- To specify an image as the list item marker, use the list-style-image property:

Example:


```
ul { list-style-image: url('sqpurple.gif'); }
```

➤ CSS Tables

- Table Borders:
- To specify table borders in CSS, use the border property.
- The example below specifies a black border for table, th, and td elements:
Example:

```
table, th, td { border: 1px solid black; }
```
- Table Width and Height:
- Width and height of a table is defined by the width and height properties.
- The example below sets the width of the table to 100%, and the height of the th elements to 50px:
Example:

```
table { width:100%; } th { height:50px; }
```
- Table Text Alignment:
- The text in a table is aligned with the text-align and vertical-align properties.
- The text-align property sets the horizontal alignment, like left, right, or center:
Example

```
td { text-align:right; }  
td { height:50px; vertical-align:bottom; }
```
- Table Color:
- The example below specifies the color of the borders, and the text and background color of th elements:
Example:

```
table, td, th { border:1px solid green; } th { background-color:green; color:white; }
```

Code:

```
body {  
  font-family: Arial, sans-serif;  
  margin: 0;  
  padding: 0;  
  background-image: linear-gradient(120deg, #e0c3fc 0%, #8ec5fc  
100% );  
}  
  
header {  
  background-color: #333;  
  color: #ff;  
  padding: 20px;
```

```
    text-align: center;
}

main {
    max-width: 800px;
    margin: 20px auto;
    padding: 20px;
}

h1 {
    margin: 0;
}

h2 {
    margin-top: 20px;
}

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

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

th {
    background-color: #f2f2f2;
}

ul {
    list-style-type: disc;
    padding-left: 20px;
}

.container {
    position: relative;
    width: 100%;
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 15px;
}
```

Output:

References:

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/html/?ref=lp>

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Internet Programming

Name:

Class:

Roll no.:

Submission Date:

Experiment no. 4

Title: Design a Basic Web page using advance HTML 5

Problem Definition: Design a Web pages using CSS3 selectors, Pseudo classes, Pseudo elements.

Pre-requisites: Knowledge of basic HTML and programming.

Theory:

- CSS3 Selectors:
 - Element Selector: Selects elements based on their tag name (e.g., p selects all <p> elements).
 - Class Selector: Selects elements with a specific class attribute (e.g., .btn selects elements with class="btn").
 - ID Selector: Selects a single element with a specific id attribute (e.g., #header selects an element with id="header").
 - Universal Selector: Selects all elements in the document (e.g., * selects all elements).
- Pseudo-Classes:
 - :hover: Applies styles when an element is hovered over by the cursor.
 - :active: Applies styles when an element is being clicked or activated.
- Pseudo-Elements:
 - ::before: Inserts content before the content of the selected element.
 - ::after: Inserts content after the content of the selected element.
 - ::first-line: Selects the first line of text within a block-level element.
 - ::first-letter: Selects the first letter of the text within a block-level element.
 - ::selection: Targets the portion of text selected by the user.

Code:

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-image: linear-gradient(120deg, #e0c3fc 0%, #8ec5fc
100% );
}

header {
  background-color: #333;
  color: #fff;
  padding: 20px;
  text-align: center;
}

main {
  max-width: 800px;
  margin: 20px auto;
  padding: 20px;
}

h1 {
  margin: 0;
}

h2 {
  margin-top: 20px;
}

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

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

th {
  background-color: #f2f2f2;
}
```

```
ul{
  list-style-type: disc;
  padding-left: 20px;
}

.container{
  position: relative;
  width: 100%;
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 15px;
}
```

Output:

References:

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/html/?ref=lp>

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Internet Programming

Name:

Class: IT SEM V

Roll no.:

Submission Date:

Experiment no. 5

Title: Design a web page using Bootstrap.

Problem Definition: Design a web page using Bootstrap Grid system, Forms, Button, Navbar.

Pre-requisites: Knowledge of basic HTML and programming.

Theory:

- Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development.
- It contains HTML, CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.
- Front-end Framework: Bootstrap is an open-source front-end framework developed by Twitter (now maintained by the Bootstrap community). It provides a collection of pre-designed HTML, CSS, and JavaScript components for building responsive web applications.
- Responsive Design: Bootstrap is known for its responsive design capabilities, ensuring that websites and web apps look and work well on various devices and screen sizes, from mobile phones to desktops.
- Customization: Bootstrap can be customized to match the branding and design requirements of a project. Developers can modify default styles using Sass variables and build tools.

Elements used in Bootstrap:

- **Grid System:** Bootstrap's grid system is a responsive, mobile-first layout system. It uses a 12-column grid to create flexible and responsive layouts. Grid classes like col-md-6 can be used to define the column width for different screen sizes (e.g., medium-sized screens).

- **Tables:** Bootstrap offers styles for tables, making them responsive and visually appealing. Classes like `table` and `table-striped` can be used to style tables.
- **Forms:** Bootstrap provides styles for form elements like inputs, buttons, and labels. It includes various form control styles, including validation states.
- **Buttons:** Bootstrap offers styles for buttons with various sizes and styles (e.g., primary, secondary, success, danger). Button groups and dropdowns are also supported.
- **Navbar:** The Bootstrap navbar component creates a responsive navigation bar. It supports branding, navigation links, buttons, and dropdown menus.
- **Cards:** Cards are flexible and customizable containers for displaying content. They can include images, headings, text, buttons, and more.
- **Alerts:** Bootstrap provides alert styles for displaying various types of messages. Alerts can be used to convey success, warning, error, or information messages.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Bootstrap Web Page</title>
  <!-- Include Bootstrap CSS -->
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap/dist/css/bootstrap.min.c
ss">
</head>
<body>
  <!-- Navbar -->
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">My Website</a>
    <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav"
aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav ml-auto">
        <li class="nav-item">
          <a class="nav-link" href="#">Home</a>
        </li>
```

```

    < li class= "nav-item ">
      < a class= "nav-link" href= "# "> About< /a>
    < /li>
    < li class= "nav-item ">
      < a class= "nav-link" href= "# "> Services< /a>
    < /li>
    < li class= "nav-item ">
      < a class= "nav-link" href= "# "> Contact< /a>
    < /li>
  < /ul>
< /div>
< /nav>

<!-- Container with Grid System -->
<div class= "container m t-5">
  <div class= "row ">
    <div class= "col-m d-6">
      <!-- Form -->
      <form >
        <h2> Contact Us< /h2>
        <div class= "form -group">
          <label for= "nam e"> Nam e< /label>
          <input type= "text" class= "form -control"
id= "nam e" placeholder= "Enter your nam e">
        < /div>
        <div class= "form -group">
          <label for= "em ail"> Em ail< /label>
          <input type= "em ail" class= "form -control"
id= "em ail" placeholder= "Enter your em ail">
        < /div>
        <div class= "form -group">
          <label for= "m essage"> Message< /label>
          <textarea class= "form -control" id= "m essage"
rows= "4" placeholder= "Enter your m essage">< /textarea>
        < /div>
        <button type= "subm it" class= "btn btn -
primary"> Subm it< /button>
      < /form >
    < /div>
    <div class= "col-m d-6">
      <!-- Content -->
      <h2> Welcom e to Our W ebsite< /h2>
      <p> Lorem ipsum dolor sit am et, consectetur ad ipiscing
elit. Nulla non diam eu libero consectetur laoreet. Curabiturnec
libero in elit sagittis tem por.< /p>

```

```

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

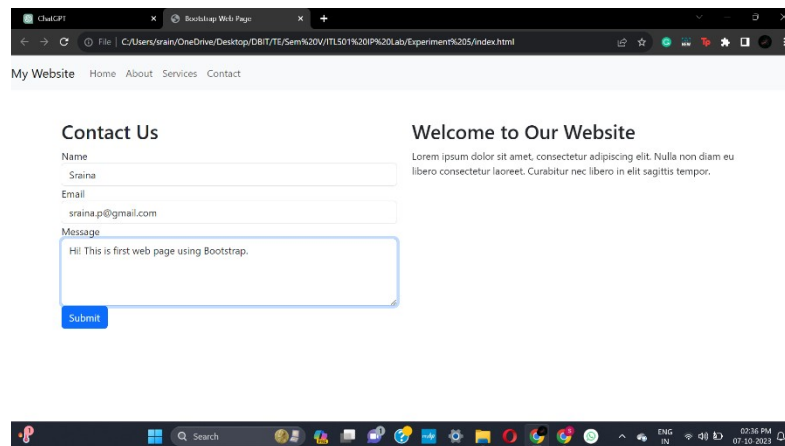
```

```

<!-- Include Bootstrap JS and JQuery -->
src= "https://cdn.jsdelivr.net/npm/bootstrap/dist/js/bootstrap.min.js"
<script src= "https://code.jquery.com/jquery-
3.6.0.min.js"> </script>
<script
> </script>
</body>
</html>

```

Output:



References:

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/html/?ref=lp>

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Internet Programming

Name:

Class: IT SEM V

Roll no.:

Submission Date:

Experiment no. 6

Title: Design a web page using JavaScript Variables

Problem Definition: Design a web page using JavaScript Variables, Operators, Conditions, Loops, Functions, Events, Classes, and Objects.

Pre-requisites: Knowledge of basic HTML and programming.

Theory:

- What is JavaScript?
 - JavaScript is a programming language that enables you to make your web pages interactive. It's like the glue that holds websites together and allows you to create dynamic and responsive content
 - JavaScript is a vital tool for web developers to make websites more interactive, user-friendly, and dynamic.
 - It's the language that brings websites to life and allows them to respond to user actions.
- Fundamental JavaScript Concepts
 - Variables: They are used to store data. We use variables like *name* and *age* to store a person's name and age.
 - Operators: Operators perform operations on variables. We use the + operator to add age and 5 and store the result in *sum*.
 - Conditions: Conditions (if-else statements) are used to make decisions in your code. We check if a person is an adult based on their *age* and display a message accordingly.
 - Loops: Loops are used to execute code repeatedly. We use a for loop to log a message to the console five times.
 - Functions: Functions are reusable blocks of code. The greet function takes a *personName* parameter and returns a greeting message.
 - Events: We use the *addEventListener* method to attach a click event to the paragraph element. When clicked, it triggers an alert.

- **Classes and Objects:** Classes are used to create objects with shared properties and methods. We define a `Person` class and create two instances, `person1` and `person2`, with individual properties and methods.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Message System </title>
</head>
<body>
  <h1>Message System </h1>
  <button id="sendMessage">Send Message</button>

  <script>
    // JavaScript Class
    class Message {
      constructor(sender, content) {
        this.sender = sender;
        this.content = content;
      }

      display() {
        alert(`Message from
${this.sender}: \n${this.content}`);
      }
    }

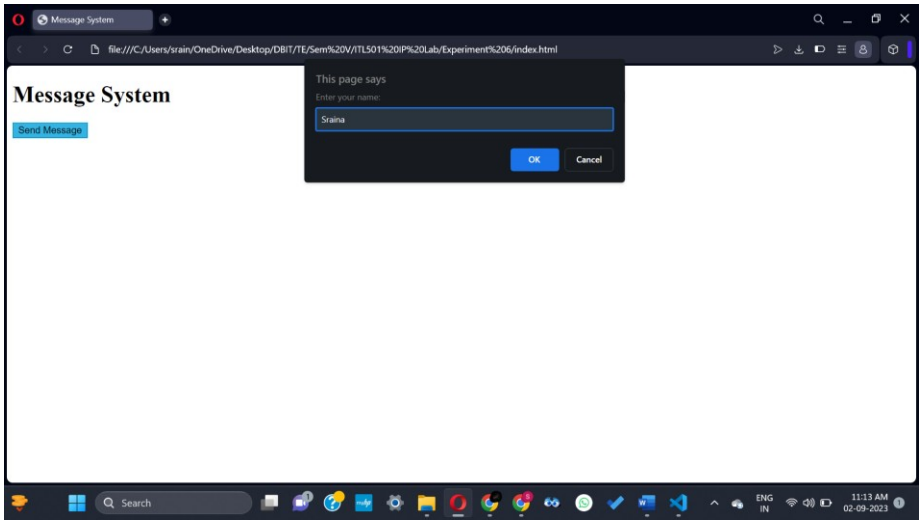
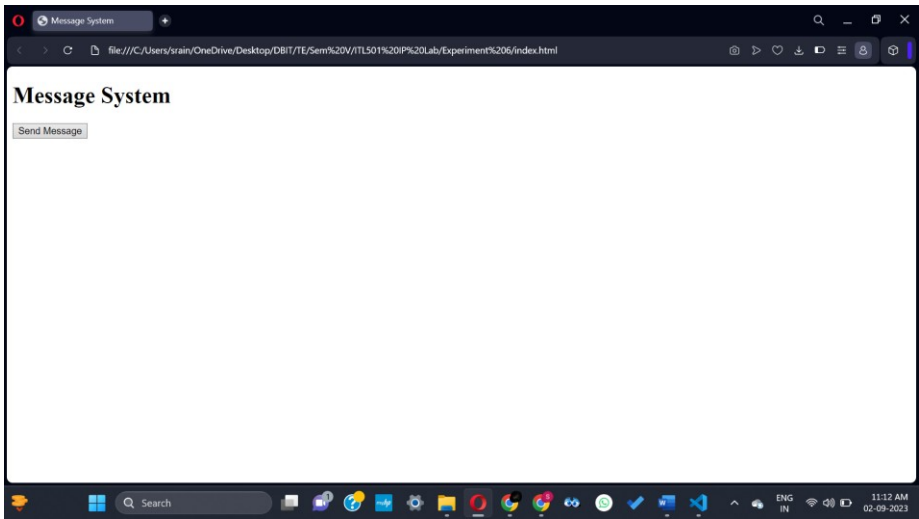
    // JavaScript Event
    document.getElementById("sendMessage").addEventListener
("click", function() {
      const sender = prompt("Enter your name:");
      const content = prompt("Enter your message:");

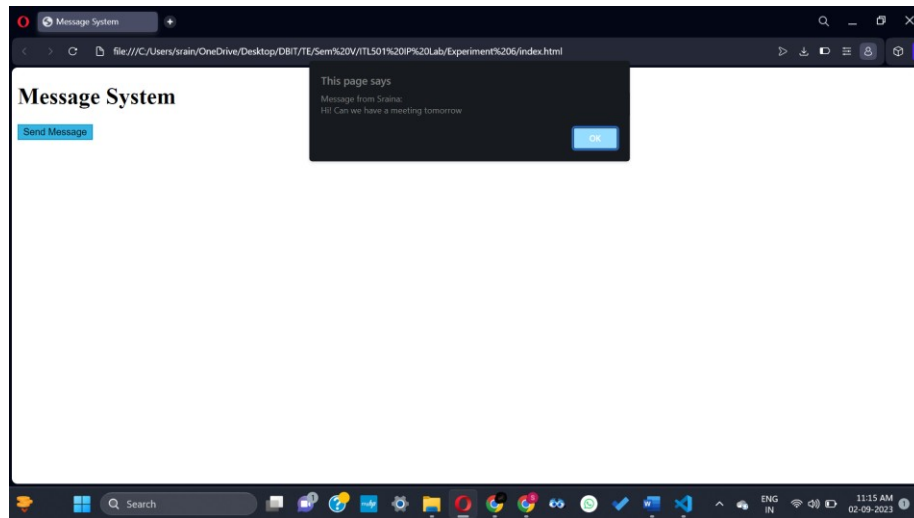
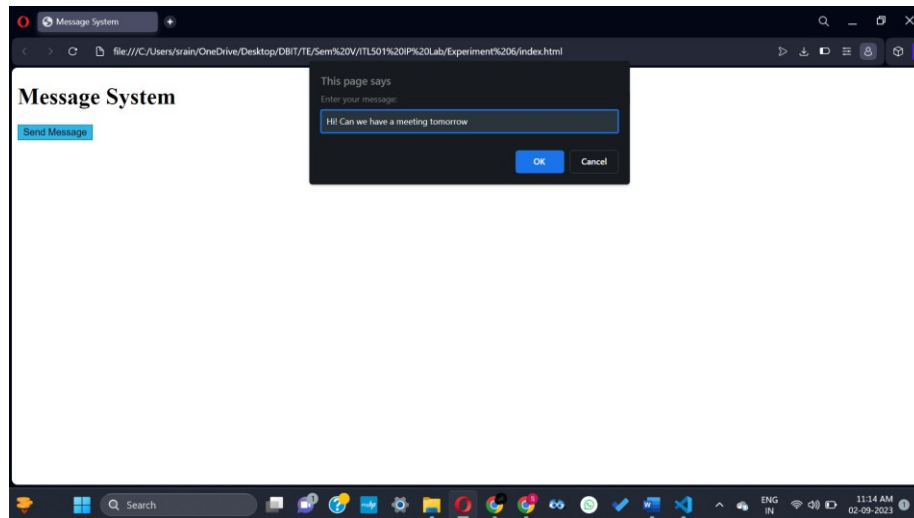
      if (sender !== null && content !== null) {
        const message = new Message(sender, content);
        message.display();
      } else {
        alert("Please enter both your name and
message.");
      }
    });
  </script>
</body>
</html>
```



```
    }  
  });  
</script>  
</body>  
</html>
```

Output:





References:

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/html/?ref=lp>

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Internet Programming

Name:

Class: IT SEM V

Roll no.:

Submission Date:

Experiment no. 7

Title: To create a web page using JavaScript Variables

Problem Definition: To Create a web page using JavaScript Validations, Arrays, String and Date function.

Pre-requisites: Knowledge of basic HTML and programming.

Theory:

- JavaScript Validations
 - JavaScript is often used for client-side form validations.
 - Validations ensure that user input is correct and meets specific criteria before it is sent to the server.
 - Common validation techniques include:
 - Field Check: Ensure that required fields are not left empty.
 - Data Type Validation: Verify that the input matches the expected data type (e.g., numbers, dates).
 - Range Check: Ensure values fall within a specified range.
 - Pattern Matching: Use regular expressions to validate data formats (e.g., email, phone number).
 - Length Validation: Check if input length meets requirements (e.g., password length).
 - Custom Validations: Implement custom validation logic based on your application's needs.
- JavaScript Arrays
 - Arrays in JavaScript are ordered collections of values. Key concepts about arrays include:
 - Declaration: Create an array using square brackets, e.g., `var fruits = ["apple", "banana", "cherry"]`.
 - Accessing Elements: Access elements by their index, starting from 0, e.g., `fruits[0]` returns "apple".

- Array Methods: JavaScript provides a variety of methods for manipulating arrays, such as push, pop, shift, unshift, splice, concat, slice, forEach, map, and filter.
 - Length Property: Get the length of an array using the length property.
 - Multidimensional Arrays: Create arrays within arrays to represent matrices or multi-level data structures.
- JavaScript Strings
- Strings in JavaScript represent sequences of characters. Key features of strings include:
 - String Declaration: Create strings using single or double quotes, e.g., var greeting = "Hello, world!".
 - String Methods: JavaScript provides various methods for working with strings, such as concat, length, indexOf, substring, split, replace, toUpperCase, toLowerCase, and more.
 - String Interpolation: Use backticks (`) for template literals, allowing variable insertion and multiline strings, e.g., `\${name} is \${age} years old.
- JavaScript Date Functions
- JavaScript has built-in functionality for working with dates and times. Key aspects include:
 - Date Object: Create a date object using new Date() or by specifying a date string.
 - Date Methods: Use methods like getFullYear, getMonth, getDate, getHours, getMinutes, getSeconds, and getDay to extract components of a date.
 - Date Arithmetic: Perform date arithmetic to calculate differences between dates, add/subtract time intervals, or format dates.
 - Date Formatting: Format dates and times for display using a combination of methods and string concatenation.

Code:

```
<!DOCTYPE html>
<html>

<head>
  <title> Simple Form </title>
  <script>
    function validateForm () {
      var name = document.forms["myForm"]["name"].value;
      var email = document.forms["myForm"]["email"].value;
      var dob = document.forms["myForm"]["dob"].value;
```

```

    if (name === "" || email === "" || dob === "") {
        alert("All fields must be filled out");
        return false;
    }

    var dobDate = new Date(dob);
    var currentDate = new Date();
    var maxDate = new Date("2023-12-31");

    if (dobDate > maxDate) {
        alert("Date of Birth must be on or before 2023");
        return false;
    }

    var emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;
    if (!emailPattern.test(email)) {
        alert("Invalid email format");
        return false;
    }

    alert("Form submitted!" +
        "\nName: " + name +
        "\nEmail: " + email +
        "\nDate of Birth: " + dob);

    return true;
}
</script>
</head>

<body>
    <h1>Simple Registration Form </h1>
    <form name="myForm" onsubmit="return validateForm()">
        <label>Name:</label>
        <input type="text" name="name"> <br> <br>

        <label>Email:</label>
        <input type="text" name="email"> <br> <br>

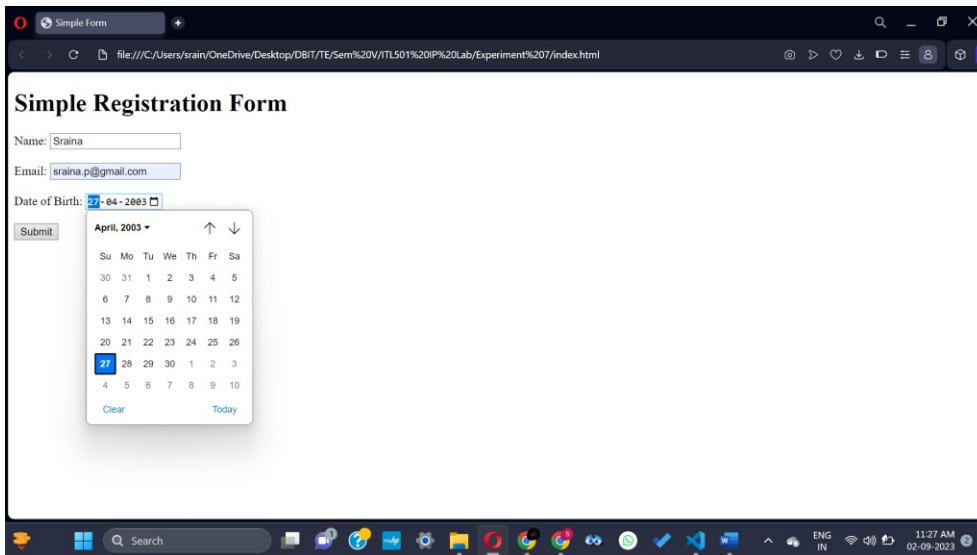
        <label>Date of Birth:</label>
        <input type="date" name="dob" max="2023-12-31"> <br> <br>

```

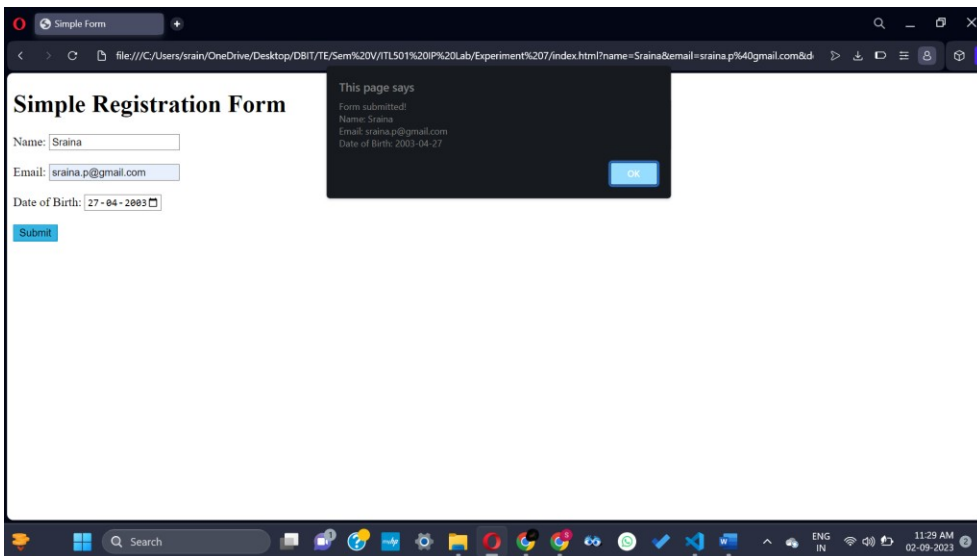
```
<input type="submit" value="Submit">
</form>
</body>

</html>
```

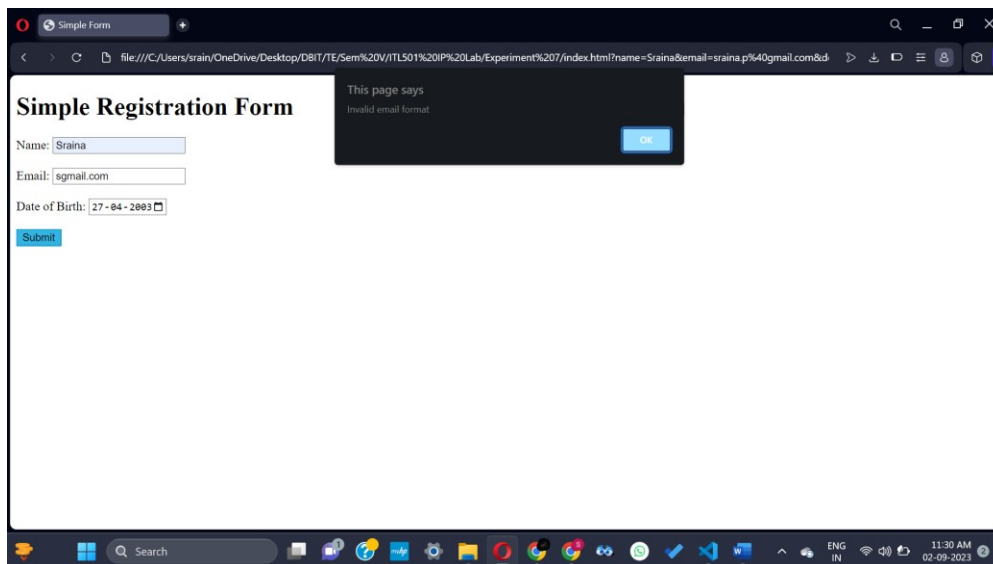
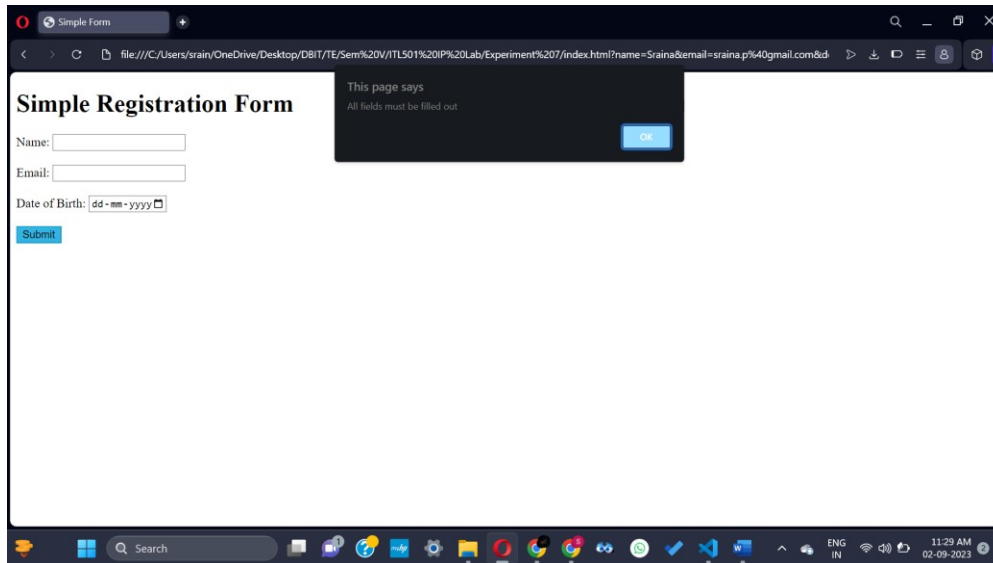
Output:



The screenshot shows a web browser window titled 'Simple Form' with the address bar displaying the file path: file:///C:/Users/srain/OneDrive/Desktop/DBIT/TE/Sem%20V/ITL501%20IP%20Lab/Experiment%207/index.html. The page content is titled 'Simple Registration Form' and includes three input fields: 'Name: Sraina', 'Email: sraina.p@gmail.com', and 'Date of Birth: 27-04-2003'. A 'Submit' button is located below the date field. A date picker calendar is open, showing the month of April 2003, with the 27th selected. The Windows taskbar at the bottom shows the time as 11:27 AM on 02-09-2023.



The screenshot shows the same web browser window after the form has been submitted. The 'Submit' button is now disabled and highlighted in blue. A dark grey confirmation box is overlaid on the right side of the form, containing the text: 'This page says', 'Form submitted!', 'Name: Sraina', 'Email: sraina.p@gmail.com', and 'Date of Birth: 2003-04-27'. An 'OK' button is at the bottom right of the confirmation box. The address bar now includes the query string: ?name=Sraina&email=sraina.p%40gmail.com&d. The Windows taskbar at the bottom shows the time as 11:29 AM on 02-09-2023.



References:

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/html/?ref=lp>

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Internet Programming

Name:

Class: IT SEM V

Roll no.:

Submission Date:

Experiment no. 8

Title: To design a web page using React Props and State

Problem Definition: Installation and Configuration of React JS, JSX, Components, Props, State.

Pre-requisites: Knowledge of basic React and programming.

Theory:

- React
 - React is an open-source JavaScript library for building user interfaces, primarily for single-page applications (SPAs).
 - It was developed by Facebook and is now maintained by a community of developers.
 - React focuses on the declarative approach to UI development, allowing developers to describe how the UI should look based on its current state.
- Components
 - In React, a component is a self-contained, reusable piece of user interface. It can be thought of as a building block for constructing UI elements.
 - Components can be either functional components (defined as functions) or class components (defined as ES6 classes). Functional components are more commonly used, especially with React hooks.
 - Components are designed to be modular and can be composed together to create complex user interfaces.
- Props (Properties)
 - Props are a mechanism for passing data from a parent component to a child component in React.
 - They are read-only and cannot be modified by the child component.

- Props make it easy to create reusable components by allowing dynamic data to be passed into components.

➤ State:

- State is a feature in React that allows a component to maintain and manage its own data.
- Unlike props, which are passed from parent to child, state is internal to a component and can be updated using the `setState` method (for class components) or the `useState` hook (for functional components).
- State is used to store and manage data that can change over time and affect the component's behaviour or rendering.
- When state changes in a component, React automatically re-renders the component to reflect the updated state.

Code:

```
import React, { Component } from "react";
import "./App.css";

//CounterComponent
class Counter extends Component {
  constructor(props) {
    super(props);
    this.state = {
      count: this.props.initialCount, // Initialize count from props
    };
  }

  //Increment the count
  incrementCount = () => {
    this.setState({ count: this.state.count + 1 });
  };

  //Decrement the count
  decrementCount = () => {
    this.setState({ count: this.state.count - 1 });
  };

  render() {
    return (
      <div>
        <h2>Counter: {this.state.count}</h2>
        <button onClick={this.incrementCount}>Increment</button>
        <button onClick={this.decrementCount}>Decrement</button>
      </div>
    );
  }
}
```

```

    });
  }
}

//App Component
function App() {
  return (
    <div className="App">
      <h1>React CounterApp</h1>
      <Counter initialCount={0} /> { /* Pass initialCount as a prop */}
      <Counter initialCount={10} /> { " "}
      { /* Pass a different initialCount as a prop */}
    </div>
  );
}

export default App;

```

Output:

The screenshot shows the VS Code interface with the terminal open. The command `npx create-react-app my-react-app` has been executed. The terminal output shows the progress of creating the app, including installing packages and dependencies. The final message is "Success! Created my-react-app at C:\Users\srain\OneDrive\Desktop\081717E\Sem V\ITL581 IP Lab\Experiment 8\my-react-app".

```

PS C:\Users\srain\OneDrive\Desktop\081717E\Sem V\ITL581 IP Lab\Experiment 8> npx create-react-app my-react-app
Creating a new React app in C:\Users\srain\OneDrive\Desktop\081717E\Sem V\ITL581 IP Lab\Experiment 8\my-react-app.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1458 packages in 1s
241 packages are looking for funding
  run `npm fund` for details

Installing template dependencies using npm...

added 69 packages, and changed 1 package in 38s
245 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...

removed 1 package, and audited 1527 packages in 4s
245 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Success! Created my-react-app at C:\Users\srain\OneDrive\Desktop\081717E\Sem V\ITL581 IP Lab\Experiment 8\my-react-app
Inside that directory, you can run several commands:

```

The screenshot shows the VS Code interface with the terminal open. The command `react-scripts start` has been executed. The terminal output shows the progress of starting the development server, including the deprecation warning for `onAfterSetupMiddleware` and the successful compilation of the app. The final message is "webpack compiled successfully".

```

> react-scripts start
(node:21896) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' is deprecated. Use 'onAfterSetup' instead.
Starting the development server...

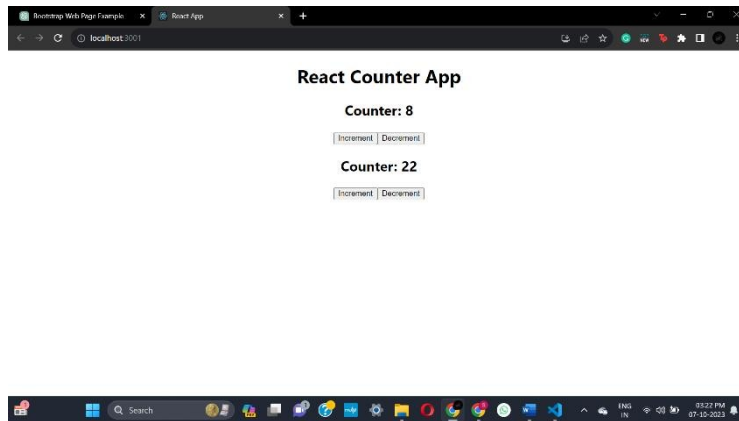
One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.

babel-preset-react-app is part of the create-react-app project, which
Compiled successfully!
t1:Done
on Your Network: http://192.168.1.175:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully

```



This web page shows a simple counter app with two counters, each managing its own state and receiving initial count values as props from the App component. Clicking the "Increment" and "Decrement" buttons will update the counts individually.

References:

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/html/?ref=lp>

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Internet Programming

Name:

Class: IT SEM V

Roll no.:

Submission Date:

Experiment no. 9

Title: To design a web page using ReactJS Hooks.

Problem Definition: Create a web page using ReactJS Forms, Events, Routers, Refs, Keys.

Pre-requisites: Knowledge of basic JavaScript, HTML, CSS and programming.

Theory:

- > Forms:
 - In React, forms are used to collect and manage user input.
 - React form elements maintain their state in the component's state.
 - You can use controlled components to link the form elements to the component's state, allowing React to control the form data.
 - Common form elements like `<input>`, `<textarea>`, and `<select>` can be controlled components.
- > Events:
 - React components can respond to user interactions (events) like clicks, keypresses, and mouse movements.
 - Event handlers are functions that handle specific events and are typically defined within components.
 - Event handlers can be attached to DOM elements using event attributes like `onClick`, `onSubmit`, etc.
 - In React, event objects are automatically passed to event handlers, providing information about the event.
- > Routers:
 - React Router is a library for handling client-side routing in React applications.
 - It allows you to define routes and navigate between different components without full page reloads.
 - React Router provides components like `<BrowserRouter>`, `<Route>`, and `<Link>` for routing.
 - Nested routes can be used to organize your application's structure.
- > Refs:

- Refs (short for references) are a way to access and interact with DOM elements directly in React.
 - They are useful for managing focus, text selection, media playback, and more.
 - Refs can be created using the `React.createRef()` method and attached to React elements via the `ref` attribute.
 - While refs can be powerful, it's often recommended to use them sparingly and favor controlled components.
- Keys:
- In React, the `key` prop is used to uniquely identify elements within a list of elements (like an array of components).
 - Keys help React efficiently update the virtual DOM and reconcile it with the actual DOM.
 - Each key should be unique within a set of sibling elements.
 - Avoid using the index of the array as keys if the order of elements may change, as it can lead to unexpected behaviour.

Code:

App.js

```
import React from "react";
import { BrowserRouter as Router, Routes, Route, Link } from
"react-router-dom ";
import "./App.css";

// Import your components here
import RegistrationForm from "../components/RegistrationForm ";
import TodoList from "../components/TodoList";

function App() {
  return (
    <Router>
      <div>
        <nav>
          <ul>
            <li>
              <Link to= "/registration"> Registration </Link>
            </li>
            <li>
              <Link to= "/todo list"> To-D o List< /Link>
            </li>
          </ul>
        </nav>
        <Routes>
          <Route path= "/reg istration"
element= {<RegistrationForm /> } />

```

```

    <Route path= "/todolist" element= {<TodoList /> } />
  </Routes>
</div>
</Router>
);
}

export default App;

```

RegistrationForm.js

```

import React, { useState } from "react";

function RegistrationForm () {
  const [formData, setFormData] = useState({
    firstName: "",
    lastName: "",
    email: "",
    password: "",
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData({
      ...formData,
      [name]: value,
    });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Registration Data:", formData);
    //You can implement API calls or validation here
  };

  return (
    <div>
      <h2>Registration Form </h2>
      <form onSubmit={handleSubmit}>
        <div>
          <input
            type= "text"
            name= "firstName"
            placeholder= "First Name"

```

```

        value= { form Data .firstNam e}
        onChange= {handleChange}
      />
    </div>
    <div>
      <input
        type= "text"
        nam e= "lastNam e"
        placeholder= "Last Nam e"
        value= { form Data .lastNam e}
        onChange= {handleChange}
      />
    </div>
    <div>
      <input
        type= "em ail"
        nam e= "em ail"
        placeholder= "Em ail"
        value= { form Data .em ail}
        onChange= {handleChange}
      />
    </div>
    <div>
      <input
        type= "passw ord"
        nam e= "passw ord"
        placeholder= "Passw ord"
        value= { form Data .passw ord}
        onChange= {handleChange}
      />
    </div>
    <div>
      <button type= "subm it"> Register</button>
    </div>
  </form >
</div>
);
}

export default RegistrationForm ;

```

TodoList.js

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

```

function TodoList() {
  const [tasks, setTasks] = useState([]);
  const [newTask, setNewTask] = useState("");

  const handleAddTask = () => {
    if (newTask.trim() !== "") {
      setTasks([...tasks, { id: Date.now(), text: newTask }]);
      setNewTask("");
    }
  };

  const handleDeleteTask = (taskId) => {
    setTasks(tasks.filter((task) => task.id !== taskId));
  };

  return (
    <div>
      <h2> To-Do List</h2>
      <input
        type= "text"
        value= {newTask}
        onChange= {(e) => setNewTask(e.target.value)}
        placeholder= "Add a new task"
      />
      <button onClick= {handleAddTask}> Add Task</button>
      <ul>
        {tasks.map((task) => (
          <li key= {task.id}>
            {task.text}
            <button onClick= {(e) =>
handleDeleteTask(task.id)}> Delete</button>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default TodoList;

```

App.css

```
body {
```



```
font-family: Arial, sans-serif;
background-color: #f0f0f0;
margin: 0;
padding: 0;
}

nav ul{
  list-style: none;
  padding: 0;
  display: flex;
  background-color: #d6e9c6; /* Light green background */
}

nav ul li{
  margin: 0;
  padding: 0;
}

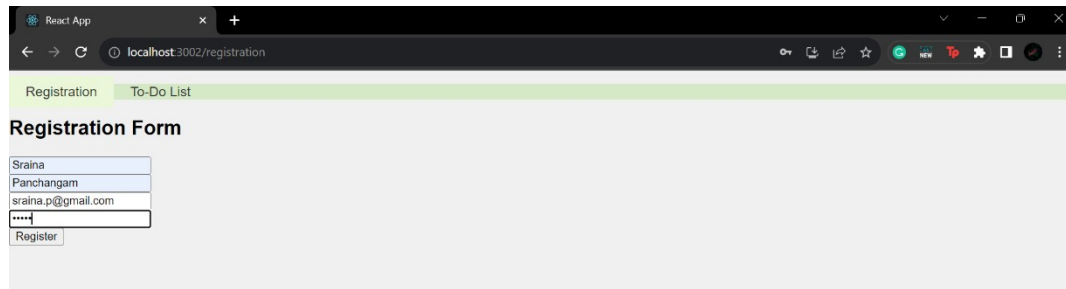
nav ul li a {
  text-decoration: none;
  color: #333; /* Dark text color */
  padding: 10px 20px;
}

nav ul li a:hover{
  background-color: #eaf7d8; /* Light yellow on hover */
  color: #333; /* Dark text color on hover */
}

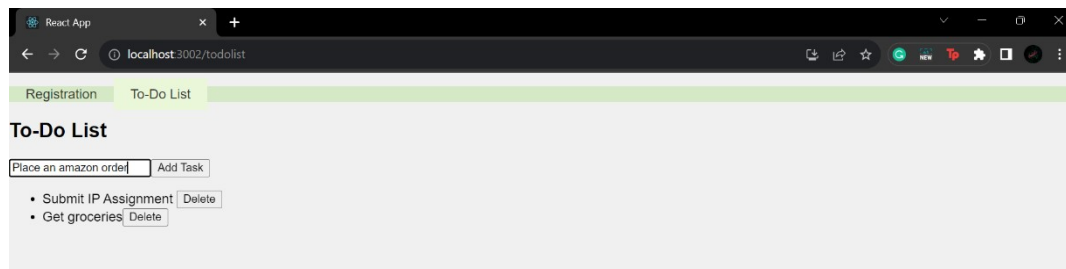
.registration-form {
  background-color: #eaf7d8; /* Light yellow background */
  border: 1px solid #c5e1a5; /* Light green border */
  padding: 20px;
  margin: 10px;
  border-radius: 5px;
}

.todo-list {
  background-color: #eaf7d8; /* Light yellow background */
  border: 1px solid #c5e1a5; /* Light green border */
  padding: 20px;
  margin: 10px;
  border-radius: 5px;
}
```

Output:



A screenshot of a web browser window titled "React App" with the address bar showing "localhost:3002/registration". The page has a green header bar with two tabs: "Registration" (active) and "To-Do List". Below the header, the title "Registration Form" is displayed. The form contains four input fields: a text field with "Sraina", a text field with "Panchangam", a text field with "sraina.p@gmail.com", and a password field with four dots. A "Register" button is located at the bottom of the form.



A screenshot of a web browser window titled "React App" with the address bar showing "localhost:3002/todolist". The page has a green header bar with two tabs: "Registration" and "To-Do List" (active). Below the header, the title "To-Do List" is displayed. The form contains a text input field with "Place an amazon order" and an "Add Task" button. Below this, there is a list of tasks: "Submit IP Assignment" and "Get groceries". Each task has a "Delete" button next to it.

References:

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/html/?ref=lp>

Don Bosco Institute of Technology

Kurla (W), Mumbai 400 070

Internet Programming

Name:

Class: IT SEM V

Roll no.:

Submission Date:

Experiment no. 10

Title: case Study – Express JS

Problem Definition: What is Express JS, Express router, REST API, Generator, Authentication, session, Integration.

Pre-requisites: Knowledge of basic JavaScript and programming.

Theory:

- What is Express.js?
 - Node.js web application framework.
 - Simplifies web application and API development.
 - Handles routing, middleware, and HTTP requests.
 - **Middleware Support:** Express.js offers a robust middleware system for easily integrating third-party middleware to enhance your application (e.g., body parsing, authentication, logging).
 - **Template Engines:** It supports various template engines (like EJS, Pug, and Handlebars) for rendering dynamic content on the server-side, making it versatile for generating HTML.
 - **Error Handling:** Express provides built-in error handling and a flexible way to define error-handling middleware, making it easier to manage and handle errors in your application.
 - **Scalability:** Express.js is highly scalable, and you can utilize clustering or other strategies to handle heavy loads and ensure your application remains responsive.

- Express Router:
 - Part of Express.js. It organizes and modularizes route handlers.
 - Enhances code structure and maintainability.
 - **Middleware Chaining:** You can apply middleware at both the router and route level, allowing for fine-grained control over request processing.
 - **Nested Routers:** Routers can be nested within other routers, enabling the creation of complex routing structures, making your code more organized and maintainable.
 - **Reusability:** Routers promote code reusability, as you can define common routes and middleware in separate router modules and reuse them across your application.
 - **Mounting:** Routers can be mounted at different paths within your application, making it easy to handle routes for different parts of your website or API.
 - **Code Snippet:**

```
//const express = require('express');
const app = express();

app.get('/', (req, res) => res.send('Hello, Express!'));

app.listen(3000, () => console.log('Server is running on port 3000'));
```
- REST API (Representational State Transfer):
 - Architectural style for networked applications.
 - Uses HTTP methods for CRUD operations on resources.
 - Common for web services, implemented with Express.js.
 - **Stateless:** RESTful APIs are stateless, meaning each request from a client to the server must contain all the information needed to understand and fulfil the request. This simplifies server-side implementation and scales well.

- **Status Codes:** They use HTTP status codes to indicate the outcome of API requests (e.g., 200 for success, 404 for not found, 401 for unauthorized).
- **Resource Naming:** Resources in REST APIs are typically named using plural nouns (e.g., /users, /products), and individual resource items are identified by unique IDs (e.g., /users/123).
- **Idempotent Methods:** Some HTTP methods in REST (e.g., GET, PUT, DELETE) are idempotent, meaning they have the same effect whether called once or multiple times.

- **Code Snippet:**

```
//REST API CRUD operations
const express = require('express');
const app = express();
const users = [];

app.get('/users', (req, res) => res.json(users));
app.get('/users/:id', (req, res) => {
  const user = users.find(user => user.id === parseInt(req.params.id));
  user ? res.json(user) : res.status(404).send('User not found');
});

// Implement POST, PUT, and DELETE routes

app.listen(3000, () => console.log('Server is running on port 3000'));
```

- **Generator:**

- **Generator Functions:** Introduced in ECMAScript 6 (ES6), generator functions are defined with an asterisk (*) and allow you to use the yield keyword to pause and resume their execution.
- **Iterators:** Generator functions create iterators, which can be used to iterate through a series of values, allowing you to work with asynchronous code in a more synchronous style.
- **Limited Use:** Generators were widely used for handling asynchronous code before the introduction of async/await and promises. They are less commonly used in modern JavaScript development.

- **Coroutine Patterns:** Generators can be employed to implement coroutine patterns, which are useful for managing asynchronous tasks in a sequential manner.
- Authentication:
 - Verifies user or system identity. Key for secure web application access.
 - Implemented using login systems, OAuth, JWT.
 - **Strategies:** Authentication in Express.js often involves authentication strategies, such as Passport.js, which support various methods like local authentication, OAuth, and more.
 - **Middleware:** Authentication middleware is used to check whether a user is authenticated before granting access to specific routes, enhancing the security of your application.
 - **User Management:** You'll need to manage user data, such as storing passwords securely (e.g., through hashing) and managing user sessions and tokens (e.g., JWT).
 - **Social Authentication:** Many web applications integrate with social media platforms for user registration and login, making authentication more convenient for users.
 - **Code Snippet:**

```
//Passport.js
const express = require('express');
const passport = require('passport');
const LocalStrategy = require('passport-local').Strategy;
const app = express();

passport.use(new LocalStrategy((username, password, done) => {/*
Auth logic */}));

app.post('/login', passport.authenticate('local'), (req, res) =>
res.send('Logged in'));

app.listen(3000, () => console.log('Server is running on port 3000'));
```

- Session:

- Tracks user-specific data during interactions. Created when users log in. Managed with middleware like express-session.
- **Server-Side Storage:** Session data is typically stored on the server, often in memory or databases, and identified by a session ID stored as a cookie on the client's side.
- **Security Concerns:** Managing session data requires attention to security, as vulnerabilities can lead to session hijacking or other attacks. Developers must take precautions to secure sessions.
- **Session Expiry:** Sessions can have a predefined duration and may be automatically destroyed after a period of inactivity to improve security.
- **Persistent Sessions:** Some applications require persistent sessions, such as e-commerce websites that need to maintain shopping carts across user visits.

- **Code Snippet:**

```
//Using Middlewear
```

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

```
const session = require('express-session');
```

```
const app = express();
```

```
app.use(session({ secret: 'your-secret-key', resave: false,  
saveUninitialized: true }));
```

```
app.get('/', (req, res) => { req.session.username = 'user123';  
res.send('Session set'); });
```

```
app.get('/profile', (req, res) => res.send(`Welcome,  
${req.session.username}!`));
```

```
app.listen(3000, () => console.log('Server is running on port 3000'));
```

- Integration:

- Connects various components and services. Central to building complete web applications. Includes database, third-party API, and frontend integration.

- **API Integration:** Often involves connecting to third-party APIs, like payment gateways (e.g., Stripe), geolocation services (e.g., Google Maps), or social media APIs (e.g., Facebook or Twitter).
- **Database Integration:** Involves linking your application to databases (e.g., SQL databases like MySQL, NoSQL databases like MongoDB) to store and retrieve data.
- **Real-time Integration:** Can include real-time features through technologies like WebSockets or integrating with platforms like Firebase for live updates.
- **Frontend Integration:** Combines the backend with frontend technologies, such as React, Angular, or Vue.js, to create a seamless user experience. Frontend and backend integration can be achieved through RESTful APIs or GraphQL.

References:

<https://www.geeksforgeeks.org/html/?ref=lp>