Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./~~Miss~~ Niranjan Rajesh Joshi

of IT Department, Semester 5 with

Roll No. 51 has completed a course of the necessary

experiments in the subject IP Lab under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

Teacher In-Charge

Head of the Department

Date 13/10/23

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
01.	Develop a web application using HTML Tags	1	21/7/23	
02.	using CSS and CSS3 enhance the web app of Assignment #1		28/7/23	
03.	Develop a webpage using bootstrap framework		04/8/23	Sawobed 13/10/23
04.	(a) WAP in JS to study loops and functions		11/8/23	
	(b) WAP on inheritance, iterators and generators			
05.	(a) WAP to study arrow functions, DOM and CSS manipulation		18/8/23	
	(b) WAP in JS to implement promises fetch and asynchronous JS			
06.	(a) WAP to implement state and props		25/8/23	Sawobed 13/10/23
	(b) WAP to implement forms and events			
07.	(a) WAP to implement ReactJS router and animation		01/09/23	
	(b) WAP to implement ReactHooks		08/09/23	
08.	Assignment on REPL (commandline)		15/09/23	
09.	WAP to implement ReactRefs		03/10/23	
10.	WAP in node.js to create a file, read and write data, rename the file and delete the file		26/09/23	Sawobed 13/10/23
11.	Create a web application that performs CRUD operations (database connectivity)		13/10/23	
12.	Written Assignment 1		10/8/23	
13.	written Assignment 2		23/9/23	

Roll No :- 2205051
Name:- Niranjan Joshi
Date:-21/07/2023

Assignment No-1

Aim:-

Develop a web application by using HTML Tags. Elements, Attributes, Head, Body, Hyperlinks, Formatting Images, Tables, Lists, Frames, Forms, and Multimedia elements should be used.

LO Mapped:- LO1

Theory:-

The preferred markup language for documents intended to be viewed in a web browser is HTML, or Hyper Text Markup Language. It describes the purpose and organization of web content. HTML documents are downloaded from a web server or local storage by web browsers, who then turn them into multimedia web pages. HTML originally featured cues for a web page's appearance and semantically explains the structure of a web page.

HTML Tags:-

1)Structural tags:

These tags define the structure of the web page, such as the header, body, and footer. Examples of structural tags include:

a) <html>: The root element of the HTML document.

b)<head>: The head of the HTML document, which contains metadata about the page, such as the title and the author.

c)<title>: The title of the web page.

d)<body>: The body of the HTML document, which contains the main content of the page.

e)<footer>: The footer of the HTML document, which contains information such as copyright and contact information**.**

2)Formatting tags:

These tags are used to format the text on the web page, such as making the text bold, italic, or underlined. Examples of formatting tags include:

a)<b>: Makes the text bold.

b)<i>: Makes the text italic.

c)<u>: Underlines the text.

d)<h1>: Defines a level 1 heading.

e)<h2>: Defines a level 2 heading.

f)<p>: Defines a paragraph.

3)List tags:

These tags are used to create lists of items, such as ordered lists (numbered lists) and unordered lists (bulleted lists). Examples of list tags include:

a)<ul>: Defines an unordered list.

b)<li>: Defines an item in an unordered list.

c)<ol>: Defines an ordered list.

d)<li>: Defines an item in an ordered list.

4)Image tags:

These tags are used to insert images into the web page. Example of image tags include:

a)<img>: Defines an image.

b)src: The source of the image file.

c)alt: An alternative text for the image, which is displayed if the image cannot be loaded.

5)Link tags:

These tags are used to create hyperlinks, which allow users to navigate to other web pages. Example of link tags include:

a) <a>: Defines a hyperlink.

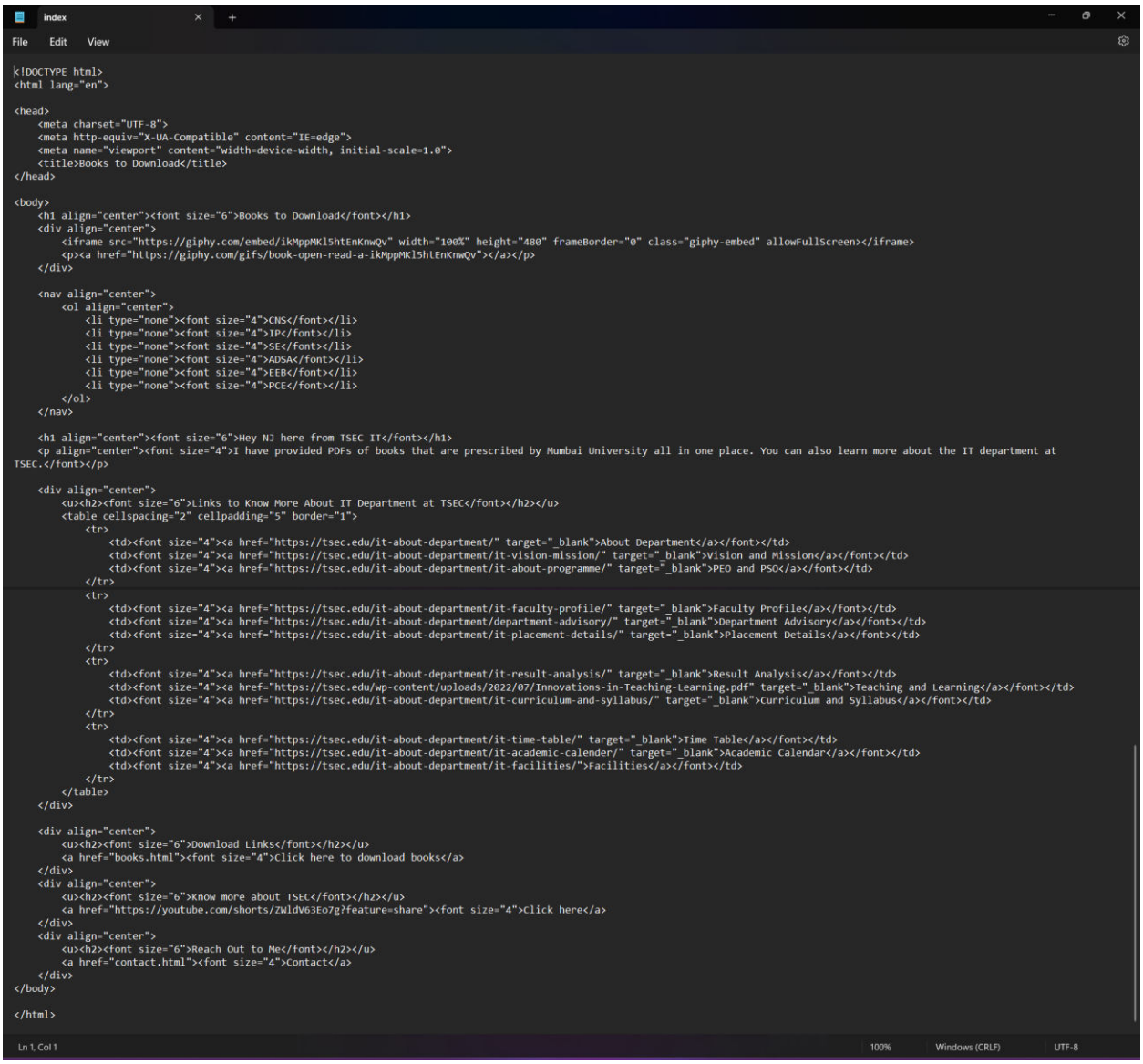
b)href: The URL of the page that the link points to.

c)title: A title for the link, which is displayed when the user hovers over the link.

There are several other types of tags but tags that are listed above are prominent ones

Code:-

1)Code for index page :-



2)Output for Index Page :-

Books to Download

CNS
IP
SE
ADSA
EEB
PCE

Hey NJ here from TSEC IT

I have provided PDFs of books that are prescribed by Mumbai University all in one place. You can also learn more about the IT department at TSEC.

Links to Know More About IT Department at TSEC

About Department	Vision and Mission	PEO and PSO
Faculty Profile	Department Advisory	Placement Details
Result Analysis	Teaching and Learning	Curriculum and Syllabus
Time Table	Academic Calendar	Facilities

Download Links

Click here to download books

Know more about TSEC

Click here

Reach Out to Me

Contact

3)Code for Contact page:-

contact

File Edit View

```
<!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>Contact-ME!</title>
</head>
<body>
<form>
<fieldset>
    <table align="center" cellpadding="5" border="1">
        <caption><strong><font size="6">Contact Me</font></strong></caption>
        <tr>
            <td>
                <label for="Name"><strong><font size="4">Name:</font></strong></label>
            </td>
            <td>
                <input type="text">
            </td>
        </tr>
        <tr>
            <td>
                <label for="email"><strong><font size="4">Email:</font></strong></label>
            </td>
            <td>
                <input type="email">
            </td>
        </tr>
        <tr>
            <td>
                <label for="subject"><strong><font size="4">Subject:</font></strong></label>
            </td>
            <td>
                <input type="text">
            </td>
        </tr>
        <tr>
            <td>
                <label for="message"><strong><font size="4">Message:</font></strong></label>
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

```
            </td>
            <td>
                <input type="message">
            </td>
        </tr>
        <tr>
            <td>
                <label for="country"><strong><font size="4">Country:</font></strong></label>
            </td>
            <td>
                <select>
                    <option value="India">India</option>
                    <option value="Canada">Canada</option>
                    <option value="USA">USA</option>
                </select>
            </td>
        </tr>
        <tr>
            <td>
                <label for="website"><strong><font size="4">Website:</font></strong></label>
            </td>
            <td>
                <input type="url">
            </td>
        </tr>
        <tr>
            <td colspan="2">
                <strong><font size="5">Social Media</font></strong>
            </td>
        </tr>
        <tr>
            <td>
                <label for="Twitter"><strong><font size="4">Twitter</font></strong></label>
            </td>
            <td>
                <a href="https://twitter.com" target="_blank"><img src="https://th.bing.com/th/id/OIP.o06snWh5G4LQ1fJbNlarpAHaGB?pid=ImgDet&rs=1" alt="" srcset="" width="25px"
height="25px"></a>
            </td>
        </tr>
        <tr>
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

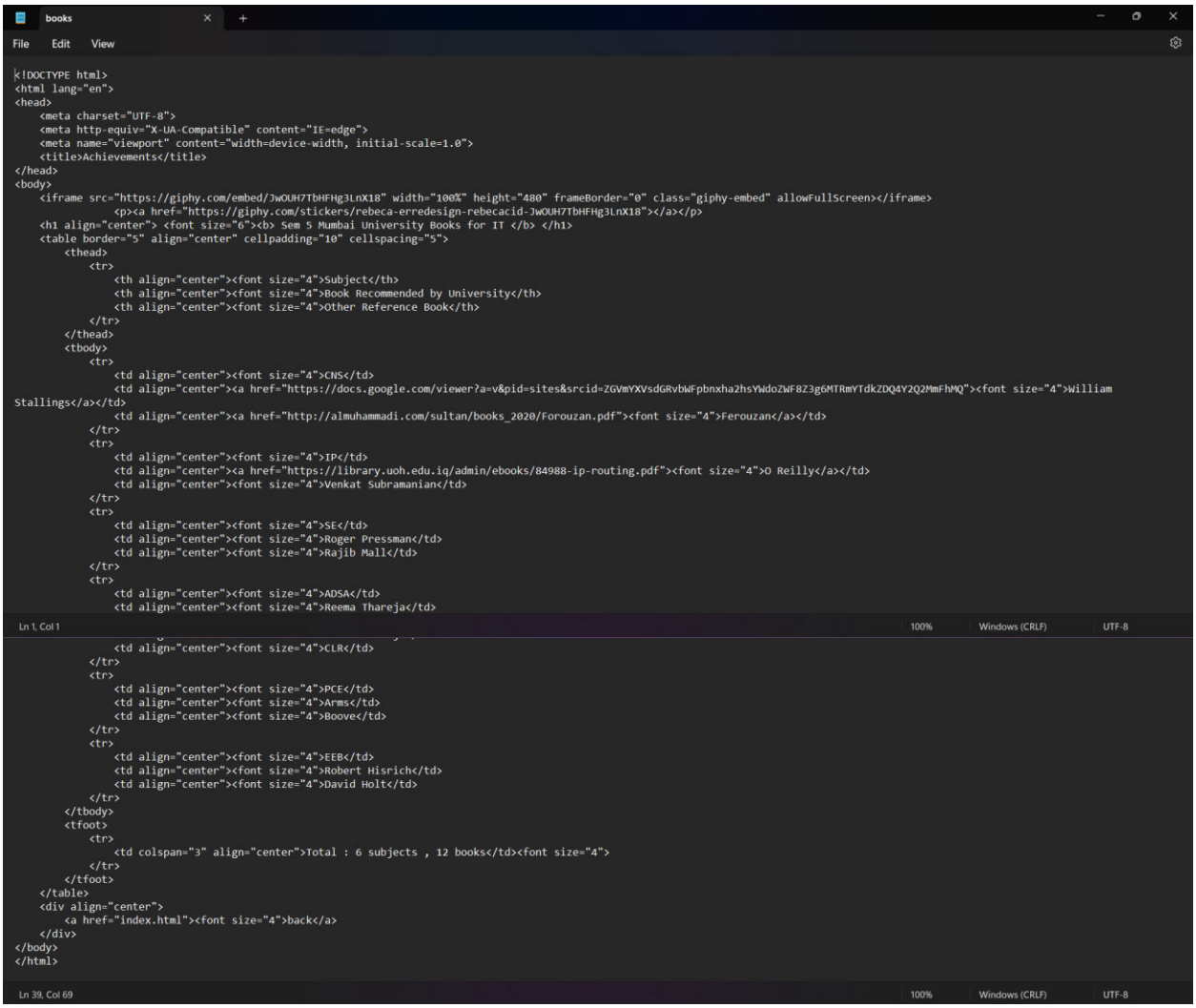
```
            <td>
                <label for="Instagram"><strong><font size="4">Instagram</font></strong></label>
            </td>
            <td>
                <a href="https://instagram.com" target="_blank"><img src="https://th.bing.com/th/id/R.26d9974a1feec9905a4e0d5e5ddf8db6?rik=Og1ujXM2C1AJHQ&riu=http%3a%2f%
2fupload.wikimedia.org%2fwikipedia%2fcommons%2fa%2fa5%2fInstagram_icon.png&ehk=1%2fZWXYn2nN%2fR80TOtcKH5SsdLkkUvMLrB%2fHUXRDHk9I%3d&risl=&pid=ImgRaw&r=0" alt="" srcset="" width="25px"
height="25px"></a>
            </td>
        </tr>
        <tr>
            <td colspan="2">
                <label for="terms"><strong><font size="4">&amp;copy;I agree to the Terms &amp; condition</font></strong></label>
                <input type="checkbox" name="" id="">
            </td>
        </tr>
        <tr>
            <td colspan="2">
                <button type="submit"><strong><font size="4">Click to Submit</font></strong></button>
            </td>
        </tr>
    </table>
</fieldset>
</form>
</body>
</html>
```

Ln 71, Col 13 100% Windows (CRLF) UTF-8

4)Output for Contact Page:-

---

5)Code for books page:-



```
books
File  Edit  View

<!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>Achievements</title>
</head>
<body>
    <iframe src="https://giphy.com/embed/JwOUH7TbHFHg3LnX18" width="100%" height="480" frameBorder="0" class="giphy-embed" allowFullScreen></iframe>
            <p><a href="https://giphy.com/stickers/rebeca-erredesign-rebecacid-JwOUH7TbHFHg3LnX18"></a></p>
    <h1 align="center"> <font size="6"><b> Sem 5 Mumbai University Books for IT </b> </h1>
    <table border="5" align="center" cellpadding="10" cellspacing="5">
        <thead>
            <tr>
                <th align="center"><font size="4">Subject</th>
                <th align="center"><font size="4">Book Recommended by University</th>
                <th align="center"><font size="4">Other Reference Book</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td align="center"><font size="4">CNS</td>
                <td align="center"><a href="https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxha2hsYWdoZWF8Z3g6MTRmYTdkZDQ4Y2Q2MmFhMQ"><font size="4">William
Stallings</a></td>
                <td align="center"><a href="http://almuhammadi.com/sultan/books_2020/Forouzan.pdf"><font size="4">Ferouzan</a></td>
            </tr>
            <tr>
                <td align="center"><font size="4">IP</td>
                <td align="center"><a href="https://library.uoh.edu.iq/admin/ebooks/84988-ip-routing.pdf"><font size="4">O Reilly</a></td>
                <td align="center"><font size="4">Venkat Subramanian</td>
            </tr>
            <tr>
                <td align="center"><font size="4">SE</td>
                <td align="center"><font size="4">Roger Pressman</td>
                <td align="center"><font size="4">Rajib Mall</td>
            </tr>
            <tr>
                <td align="center"><font size="4">ADSA</td>
                <td align="center"><font size="4">Reema Thareja</td>
Ln 1, Col 1                                                                    100%  Windows (CRLF)  UTF-8
                <td align="center"><font size="4">CLR</td>
            </tr>
            <tr>
                <td align="center"><font size="4">PCE</td>
                <td align="center"><font size="4">Arms</td>
                <td align="center"><font size="4">Boove</td>
            </tr>
            <tr>
                <td align="center"><font size="4">EEB</td>
                <td align="center"><font size="4">Robert Hisrich</td>
                <td align="center"><font size="4">David Holt</td>
            </tr>
        </tbody>
        <tfoot>
            <tr>
                <td colspan="3" align="center">Total : 6 subjects , 12 books</td><font size="4">
            </tr>
        </tfoot>
    </table>
    <div align="center">
        <a href="index.html"><font size="4">back</a>
    </div>
</body>
</html>
Ln 39, Col 69                                                                  100%  Windows (CRLF)  UTF-8
```

6)Output of Books page :-

Sem 5 Mumbai University Books for IT

Subject	Book Recommended by University	Other Reference Book
CNS	William Stallings	Ferouzan
IP	O Reilly	Venkat Subramanian

Sem 5 Mumbai University Books for IT

Subject	Book Recommended by University	Other Reference Book
CNS	William Stallings	Ferouzan
IP	O Reilly	Venkat Subramanian
SE	Roger Pressman	Rajib Mall
ADSA	Reema Thareja	CLR
PCE	Arms	Boove
EEB	Robert Hisrich	David Holt
Total : 6 subjects , 12 books		

back

Reference:-

https://www.w3schools.com/tags/ref_byfunc.asp

https://www.tutorialspoint.com/html/html_basic_tags.htm

Conclusion:-

Used several HTML tags such as list , table , formatting , structural , link tags and several elements related to these tags to create several webpages and also inserted images , did formatting to make these look more appealing to user

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-28/07/2023

Assignment No-2

Aim:-

Using CSS and CSS3 enhance the web application developed in Assignment #1
Color, Background, Font, Table, List, CSS3 selectors, Pseudo classes, and
Pseudo elements properties should be used to enhance the web pages.

LO Mapped:- LO2

Theory:-

CSS is a stylesheet language used to control the presentation and layout of HTML documents. It enables web developers to style the elements on a web page, such as fonts, colors, margins, padding, and more, making it visually appealing and well organized. CSS3 is the latest version of CSS, offering enhanced features and capabilities compared to its predecessors. It introduces new properties, animations, transformations, transitions, and more, allowing developers to create more sophisticated and interactive designs.

CSS provides various ways to specify colors for elements on a web page. You can use predefined color names, hexadecimal codes, RGB, RGBA, HSL, or HSLA values to set the color of text, backgrounds, borders, and other visual elements.

CSS allows you to define background styles for elements, including setting background colors, images, gradients, and positioning. This helps in creating visually appealing and dynamic backgrounds for web pages.

With CSS, you can control the typography of text content on a web page. It allows you to specify font families, sizes, styles (bold, italic), and other text-related properties to achieve the desired visual appearance.

CSS3 Selectors:
CSS3 introduces several new selectors, expanding the ways you can target and style specific elements on a web page. These selectors are more advanced than the basic

element and class selectors. Here are a few examples:
Attribute Selectors: You can select elements based on their attributes and attribute values. For instance, you can style all links with a specific target attribute using [target="_blank"].
Child Selectors: These selectors target elements that are direct children of a specific parent element. For example, ul > li targets all list items that are direct children of a <ul> element.
Sibling Selectors: You can target elements that are siblings of other elements. For instance, h2 + p selects the <p> element that immediately follows an <h2> element.
:not() Selector: This allows you to exclude elements from styling. For example, li:not(.special) selects all list items that do not have the class "special."
These are just a few examples of CSS3 selectors that provide more flexibility and precision when applying styles to elements on a webpage.

Pseudo-classes:
Pseudo-classes are keywords that allow you to style elements based on their state or position in relation to user interactions. They start with a colon : followed by the name of the pseudo-class. Some common pseudo-classes include:

:hover: This pseudo-class is used to apply styles when the user hovers the mouse pointer over an element. For example, you can change the color of a link when hovered
.
:active: The :active pseudo-class targets elements that are being clicked by the user. It's often used to give feedback during the click action, such as changing the color of a button
.
:focus: This pseudo-class targets elements that currently have keyboard focus. It's useful for styling form input fields when they are selected.

:nth-child(): With this pseudo-class, you can select elements based on their position within a parent element. For example, li:nth-child(odd) selects all odd-numbered list Items.

Pseudo-elements:
Pseudo-elements allow you to style parts of an element's content without the need for extra HTML markup. They start with two colons :: followed by the name of the pseudoelement. Common examples include:
::before: This pseudo-element inserts content before the selected element. It is often used for adding decorative elements or icons before specific elements.

::after: Similar to ::before, this pseudo-element inserts content after the selected element. It's often used to add content or styling after elements, like clearing floats in a layout.
::first-line: This pseudo-element targets the first line of text within a block-level element. You can use it to apply specific styles to the first line of a paragraph.
::first-letter: This pseudo-element targets the first letter of a block-level element. It's commonly used to apply drop caps or special styles to the first letter of a paragraph.
Pseudo-classes and pseudo-elements give developers more control over the appearance and behavior of elements based on various conditions, improving the interactivity and visual design of web pages

Output:-

1)Code for index page :-

```
<!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>Books to Download</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-image: url("wall.jpg");
            background-size: cover;
            background-repeat: no-repeat;
            color:#0000FF;
        }

        h1 {
            font-size: 30px;
            text-align: center;
            margin: 20px 0;
        }

        h2 {
            font-size: 30px;
```

```
    text-decoration: underline;
}

p {
    font-size: 20px;
    text-align: center;
    margin: 10px 0;
    color:black;
}

nav {
    text-align: center;
    margin: 20px 0;
}

ol {
    list-style-type: none;
    padding: 0;
}

li {
    font-size: 18px;
}

iframe {
    width: 100%;
    height: 480px;
    border: none;
}

table {
    width: 70%;
    border-collapse: collapse;
    margin: 20px auto;
    color: white; /* Dark black color */
}

th, td {
    border: 1px solid #000080; /* Dark blue border */
    padding: 8px;
```

```
            text-align: center;
            font-size: 16px;
            color:white;
        }
        table td {
        color:white; }

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

        .centered-heading {
            text-align: center;
            margin-top: 20px;
            font-size: 20px;
        }
    </style>
</head>
<body>
    <h1>Books to Download</h1>
    <div>
        <iframe src="https://giphy.com/embed/ikMppMKl5htEnKnwQv" frameborder="0"
allowfullscreen></iframe>
        <p><a href="https://giphy.com/gifs/book-open-read-a-ikMppMKl5htEnKnwQv"></a></p>
    </div>

    <h1><font size="20px">Hey NJ here from TSEC IT</h1>
    <p>I have provided PDFs of books that are prescribed by Mumbai University all in one place.
You can also learn more about the IT department at TSEC.</p>
    <div>
        <h2 align="center">Links to Know More About IT Department at TSEC</h2>
        <table align="center">
            <tr>
                <td style="color: white;"><font size="4"><a href="https://tsec.edu/it-about-
department/" target="_blank">About Department</a></td>
                <td style="color: white;"><font size="4"><a href="https://tsec.edu/it-about-
department/it-vision-mission/" target="_blank">Vision and Mission</a></td>
                <td style="color: white;"><font size="4"><a href="https://tsec.edu/it-about-
department/it-about-programme/" target="_blank">PEO and PSO</a></td>
```

```
        </tr>
        <tr>
          <td style="color: white;"><font size="4"><a href="https://tsec.edu/it-about-department/it-faculty-profile/" target="_blank">Faculty Profile</a></font></td>
          <td style="color: white;"><font size="4"><a href="https://tsec.edu/it-about-department/department-advisory/" target="_blank">Department Advisory</a></font></td>
          <td style="color: white;"><font size="4"><a href="https://tsec.edu/it-about-department/it-placement-details/" target="_blank">Placement Details</a></font></td>
        </tr>
        <tr>
          <td style="color: white;"><font size="4"><a href="https://tsec.edu/it-about-department/it-result-analysis/" target="_blank">Result Analysis</a></font></td>
          <td style="color: white;"><font size="4"><a href="https://tsec.edu/wp-content/uploads/2022/07/Innovations-in-Teaching-Learning.pdf" target="_blank">Teaching and Learning</a></font></td>
          <td style="color: white;"><font size="4"><a href="https://tsec.edu/it-about-department/it-curriculum-and-syllabus/" target="_blank">Curriculum and Syllabus</a></font></td>
        </tr>
        <tr>
          <td style="color: white;"><font size="4"><a href="https://tsec.edu/it-about-department/it-time-table/" target="_blank">Time Table</a></font></td>
          <td style="color: white;"><font size="4"><a href="https://tsec.edu/it-about-department/it-academic-calender/" target="_blank">Academic Calendar</a></font></td>
          <td style="color: white;"><font size="4"><a href="https://tsec.edu/it-about-department/it-facilities/">Facilities</a></font></td>
        </tr>
      </table>
    </div>
  <div class="centered-heading">
      <u><h2>Download Links</h2></u>
      <a href="books.html"><font size="4">Click here to download books</a>
    </div>
    <div class="centered-heading">
      <u><h2>Know more about TSEC</h2></u>
      <a href="https://youtube.com/shorts/ZWldV63Eo7g?feature=share"><font size="4">Click here</a>
    </div>
    <div class="centered-heading">
      <u><h2>Reach Out to Me</h2></u>
```

```
        <a href="contact.html"><font size="4">Contact</a>
    </div>
</body>
</html>
```

2)Code for contact page :-

```
<!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>Contact-ME!</title>
    <style>
    body {
    font-family: Comic Sans MS, sans-serif;
    background-image:url("wall2.jpg");
}

p quote {
    font-size: 25px;
    font-weight: bold;
    text-align: center; /* Center the text */
}

form {
    width: 400px;
    margin: 0 auto;
}

div {
    margin-bottom: 10px;
}

label {
    display: block;
    font-weight: bold;
    font-size: 20px; /* Increase the label font size */
}
```

```
input, select {
    width: 100%;
    padding: 5px;
}

fieldset {
    margin-bottom: 15px;
}

legend {
    font-weight: bold;
}

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

button {
    padding: 10px 20px;
    background-color: #4CAF50;
    color: white;
    border: none;
    cursor: pointer;
}
    </style>
</head>
<body>
    <p align="center"><quote>Contact Me</quote></p>
    <form action="" method="post">
        <div class="name"><label for="Name">Name:</label>
            <input type="text"></div>
            <div class="email">
                <label for="email">Email:</label>
                <input type="email">
            </div>
            <div class="subject">
                <label for="subject">Subject:</label>
                <input type="text">
            </div>
```

```
        <div class="message">
          <label for="message">Message:</label>
          <input type="message">
        </div>
        <div class="country">
          <label for="country">Country:</label>
          <select>
            <option value="India">India</option>
            <option value="Canada">Canada</option>
            <option value="USA">USA</option>
          </select>
        </div>
        <div class="website">
          <label for="website">Website:</label>
          <input type="url">
        </div>
        <fieldset>
          <legend>Social Media</legend>
          <label for="Twitter">Twitter</label>
          <a href="https://twitter.com" target="_blank"><img
src="https://th.bing.com/th/id/OIP.o06snWh5G4LQ1fJbNlarpAHaGB?pid=ImgDet&rs=1"
alt="" srcset="" width="25px" height="25px"></a>
          <label for="Instagram">Instagram</label>
          <a href="https://instagram.com" target="_blank"><img
src="https://th.bing.com/th/id/R.26d9974a1feec9905a4e0d5e5ddf8db6?rik=Og1ujXM2C1AJHQ
&riu=http%3a%2f%2fupload.wikimedia.org%2fwikipedia%2fcommons%2fa%2fa5%2fInstagra
m_icon.png&ehk=1%2fZWXYn2nN%2fR80TOtcKH5SsdLkkUvMLrB%2fHUXRDHk9I%3d&
risl=&pid=ImgRaw&r=0" alt="" srcset="" width="25px" height="25px"></a>

        </fieldset>
        <label for="terms">&copy;I agree to the Terms & condition </label>
        <input type="checkbox" name="" id="">
        <div class="submit"> <button type="submit
          ">Book</button></div>

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

3)Code for Books page :-

```
<!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>Achievements</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <h1> <b> Sem 5 Mumbai University Books for IT </b> </h1>
   <table class="books-table">
      <thead>
         <tr>
            <th>Subject</th>
            <th>Book Recommended by University</th>
            <th>Other Reference Book</th>
         </tr>
      </thead>
      <tbody>
         <tr>
            <td>CNS</td>
            <td><a
href="https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxha2hs
YWdoZWF8Z3g6MTRmYTdkZDQ4Y2Q2MmFhMQ">William Stallings</a></td>
            <td><a
href="http://almuhammadi.com/sultan/books_2020/Forouzan.pdf">Ferouzan</a></td>
         </tr>
      <tr>
         <td>IP</td>
         <td><a href="https://library.uoh.edu.iq/admin/ebooks/84988-ip-routing.pdf">O
Reilly</a></td>
         <td>Venkat Subramanian</td>
      </tr>
      <tr>
         <td>SE</td>
         <td>Roger Pressman</td>
         <td>Rajib Mall</td>
      </tr>
      <tr>
```

```
            <td>ADSA</td>
            <td>Reema Thareja</td>
            <td>CLR</td>
        </tr>
            <tr>
            <td>PCE</td>
            <td>Arms</td>
            <td>Boove</td>
        </tr>
            <tr>
            <td>EEB</td>
            <td>Robert Hisrich</td>
            <td>David Holt</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
                <td colspan="3"> Total : 6 subjects , 12 books</td>
            </tr>
        </tfoot>
    </table>
    <a href="index.html">back</a>
</body>
</html>
```

4)code for styles.css :-

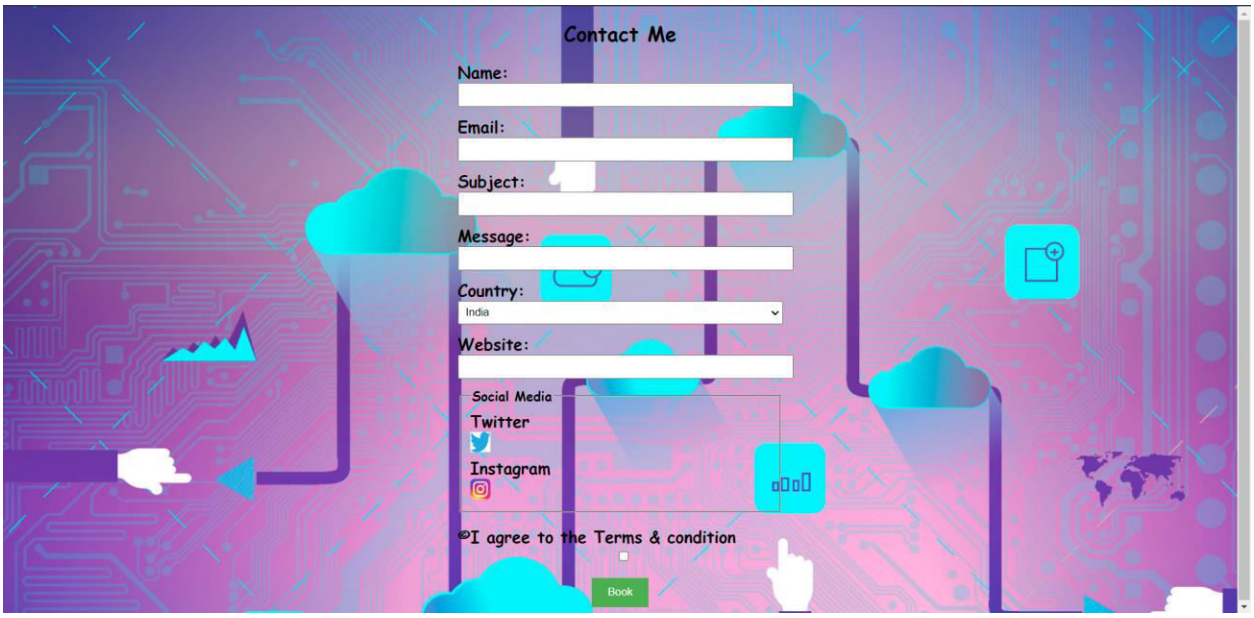
```
    body {
        background-color: lightblue;
        font-family: Arial, sans-serif;
        background-image: url("books.jpg");
            background-size: cover;
            background-repeat: no-repeat;
    }

    .books-table {
    width: 100%;
    table-layout: fixed;
    border-collapse: collapse;
    border: 1px solid blue;
```

```
    }

    .books-table th, .books-table td {
    width: 33.33%;
    text-align: center;
    border: 1px solid blue;
    padding: 10px;
    height: 25px;
    }

    .books-table tr {
    background-color:#00BFFF;
    opacity: 1;
    }

    h1 {
    text-align: center;
    color: white;
    margin-top: 50px;
    margin-bottom: 50px;
    }

    table {
    margin: 0 auto;
    border-collapse: collapse;
    }

th, td {
    padding: 10px;
    text-align: center;
    color:black;
    font-size:20px;
}

th {
    background-color: darkblue;
    color: white;
}

a {
```

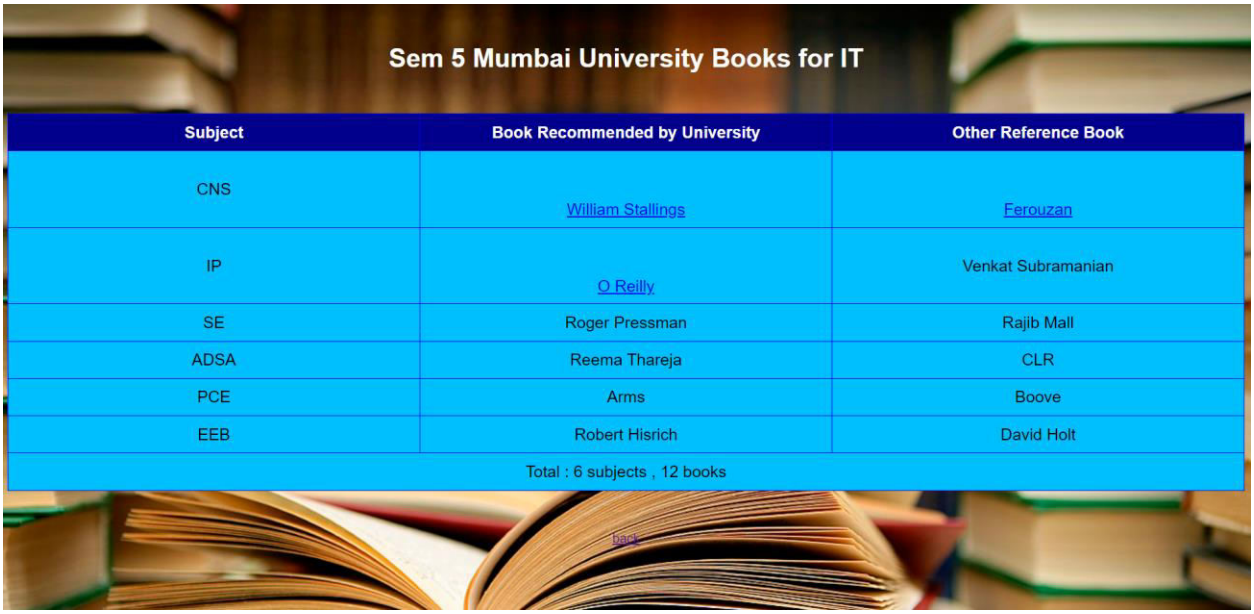
```
    display: block;
    text-align: center;
    margin-top: 50px;
}

a:hover {
    color: darkblue;
}

iframe {
    display: block;
    margin: 0 auto;
    margin-top: 50px;
}
```

5)Output of Index page :-

6)Output of contact page



7)Output for books page:-



References :-

1) https://www.w3schools.com/css/
2)https://www.w3.org/Style/CSS/Overview.en.html

Conclusion :-

Learnt about using CSS and CSS3 to enhance webpages and used several properties such as color , background, font , table , list , CSS selectors , pseudo classes , pseudo elements in developing web pages and also used other properties of CSS to enhance look of webpages.

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-04/08/2023

Assignment No-3

Aim:-

Develop a web page using the Bootstrap framework. Grid system, Forms, Button, Navbar, Breadcrumb, Jumbotron should be used..

LO Mapped:- LO3

Theory:-

Bootstrap is a popular open-source front-end framework that provides a collection of tools, styles, and components for building responsive and visually appealing websites and web applications. It was originally developed by Twitter and has since gained widespread adoption in the web development community. Bootstrap includes pre-designed HTML, CSS, and JavaScript components that make it easier to create consistent and modern user interfaces.

Some key features of Bootstrap include:

Responsive Design: Bootstrap's grid system and components are designed to adapt to different screen sizes, ensuring a seamless user experience across various devices.

Component Library: Bootstrap provides a variety of UI components like buttons, forms, navigation bars, modals, carousels, and more. These components can be customized and used to build complex interfaces quickly.

Typography and Styling: Bootstrap offers consistent typography styles and a set of CSS classes for styling elements, ensuring a coherent visual presentation.

JavaScript Plugins: Bootstrap includes JavaScript plugins for enhanced functionality, such as dropdown menus, tooltips, modals, and more.

Theming: Bootstrap can be customized using themes to match specific design requirements.

Grid system in bootstrap :-

The grid system is one of the fundamental features of the Bootstrap framework. It's designed to help developers create responsive and well-structured layouts for websites and web applications. The grid system in Bootstrap is based on a 12-column layout that can be easily customized and adapted to various screen sizes. Here's an overview of how the Bootstrap grid system works:

Container:

The layout in Bootstrap is organized within a container. There are two main types of containers: .container and .container-fluid.
The .container class provides a fixed-width container that adjusts its size based on the screen width. It's generally used for maintaining a consistent maximum width.
The .container-fluid class creates a full-width container that spans the entire viewport width.
Rows and Columns:

Within a container, you create rows using the .row class. Rows act as horizontal containers for columns.
Columns are defined using the .col-* classes, where * represents a number from 1 to 12. For example, .col-6 represents a column that spans 6 out of the 12 available columns, taking up 50% of the container's width.
Responsive Classes:

Bootstrap provides responsive classes to control the behavior of columns based on different screen sizes. These classes are suffixed with breakpoints like sm, md, lg, and xl.
For instance, you can use classes like .col-md-6 to define how columns should behave on medium-sized screens.

Form:-
Form Structure:

Start with the <form> element to create the overall form container.
Use the .form-group class to wrap each form control (input, label, etc.), providing structure and styling.
Form Controls:

Bootstrap provides styling for various form controls. For each control, use the appropriate class to apply the styling.
For example, use .form-control class for text inputs, textareas, and select elements.
Labels and Inputs:

Use the <label> element to provide a label for each form control.

Place the actual form control (input, textarea, etc.) inside the .form-control class.
Checkboxes and Radio Buttons:

Use the .form-check class to create a container for checkboxes and radio buttons.
Apply the .form-check-input class to the input element, and use the <label> element with .form-check-label class for the label text.
Form Layout and Alignment:

You can use grid classes to control the layout of form controls, aligning them within columns or rows.
Use classes like .row and .col-* to create responsive form layouts.

Jumbotron:-
In Bootstrap, a "jumbotron" is a prominent and versatile component designed to showcase key content or information on a web page. It's often used for displaying introductory content, announcements, or any content that you want to draw attention to. The jumbotron provides a visually appealing way to grab users' attention and present a clear message.

Breadcrumb :-
a "breadcrumb" is a navigation component that helps users understand the hierarchy and structure of the content on a website. Breadcrumbs are often used to show the path that leads to the current page, making it easier for users to navigate back to previous levels or sections.

Output:-

1)Code for index page :-

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Artist gallery</title>
   <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap/dist/css/bootstrap.min.css">
   <style>
      body {
         font-family: Arial, sans-serif;
      }
      .jumbotron {
         background: url('header-bg.jpg') no-repeat center center;
```

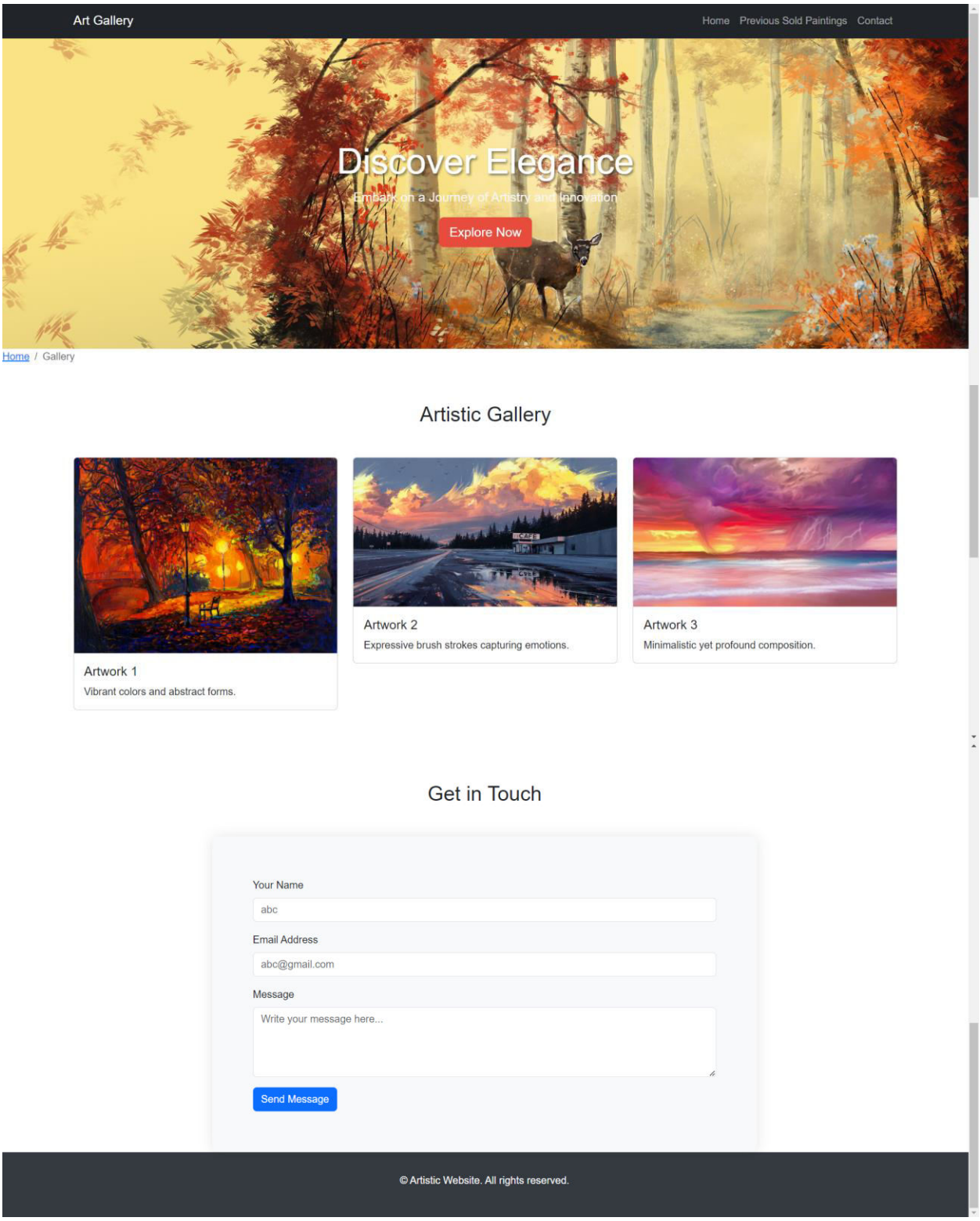
```
            background-size: cover;
            color: #fff;
            padding: 10rem 0;
        }
        .section-heading {
            margin-bottom: 3rem;
        }
        .gallery-card {
            transition: transform 0.2s;
        }
        .gallery-card:hover {
            transform: scale(1.05);
        }
        .contact-form {
            background-color: #f8f9fa;
            padding: 4rem;
            border-radius: 10px;
            box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
        }
        .footer {
            background-color: #343a40;
            color: #fff;
            padding: 2rem 0;
        }
    </style>
</head>
<body>

<!-- Navbar -->
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="container">
        <a class="navbar-brand" href="#"></font>Art Gallery</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse justify-content-end" id="navbarNav">
            <ul class="navbar-nav">
                <li class="nav-item">
```

```
                <a class="nav-link" href="#">Home</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#gallery">Previous Sold Paintings</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#contact">Contact</a>
            </li>
        </ul>
    </div>
  </div>
</nav>

<!-- Jumbotron -->
<div class="jumbotron jumbotron-fluid text-center" style="background-image: url('556069.jpg');
background-size: cover; background-position: center;">
  <div class="container">
    <h1 class="display-4" style="font-family: 'Cursive', sans-serif; color: #ffffff; text-shadow:
2px 2px 4px rgba(0, 0, 0, 0.5);">Discover Elegance</h1>
    <p class="lead" style="font-size: 20px; color: #f7f7f7;">Embark on a Journey of Artistry
and Innovation</p>
    <a href="#" class="btn btn-primary btn-lg" style="background-color: #e74c3c; border-
color: #e74c3c;">Explore Now</a>
  </div>
</div>


<!-- Breadcrumb -->
<nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="#">Home</a></li>
    <li class="breadcrumb-item active" aria-current="page">Gallery</li>
  </ol>
</nav>

<!-- Gallery -->
<section id="gallery" class="container mt-5">
  <h2 class="section-heading text-center">Artistic Gallery</h2>
  <div class="row">
    <div class="col-md-4 mb-4">
```

```
        <div class="card gallery-card">
          <img src="image1.jpg" class="card-img-top" alt="Artwork 1">
          <div class="card-body">
            <h5 class="card-title">Artwork 1</h5>
            <p class="card-text">Vibrant colors and abstract forms.</p>
          </div>
        </div>
      </div>
      <div class="col-md-4 mb-4">
        <div class="card gallery-card">
          <img src="image2.jpg" class="card-img-top" alt="Artwork 2">
          <div class="card-body">
            <h5 class="card-title">Artwork 2</h5>
            <p class="card-text">Expressive brush strokes capturing emotions.</p>
          </div>
        </div>
      </div>
      <div class="col-md-4 mb-4">
        <div class="card gallery-card">
          <img src="image3.jpg" class="card-img-top" alt="Artwork 3">
          <div class="card-body">
            <h5 class="card-title">Artwork 3</h5>
            <p class="card-text">Minimalistic yet profound composition.</p>
          </div>
        </div>
      </div>
    </div>
</section>

<!-- Contact Form -->
<section id="contact" class="container mt-5">
    <h2 class="section-heading text-center">Get in Touch</h2>
    <div class="row justify-content-center">
      <div class="col-md-8">
        <div class="contact-form">
          <form>
            <div class="mb-3">
              <label for="name" class="form-label">Your Name</label>
              <input type="text" class="form-control" id="name" placeholder="abc">
            </div>
```

```
                <div class="mb-3">
                    <label for="email" class="form-label">Email Address</label>
                    <input type="email" class="form-control" id="email"
placeholder="abc@gmail.com">
                </div>
                <div class="mb-3">
                    <label for="message" class="form-label">Message</label>
                    <textarea class="form-control" id="message" rows="4" placeholder="Write your
message here..."></textarea>
                </div>
                <button type="submit" class="btn btn-primary">Send Message</button>
            </form>
        </div>
    </div>
  </div>
</section>

<!-- Footer -->
<footer class="footer text-center">
  <div class="container">
    <p>&copy; Artistic Website. All rights reserved.</p>
  </div>
</footer>

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

2)Output of Index page:-


Art Gallery
Home
Previous Sold Paintings
Contact
Discover Elegance
Embark on a Journey of Artistry and Innovation
Explore Now
Home / Gallery
Artistic Gallery
Artwork 1
Vibrant colors and abstract forms.
Artwork 2
Expressive brush strokes capturing emotions.
Artwork 3
Minimalistic yet profound composition.
Get in Touch
Your Name
abc
Email Address
abc@gmail.com
Message
Write your message here...
Send Message
© Artistic Website. All rights reserved.

References :-

1)https://getbootstrap.com/docs/5.3/getting-started/introduction/
2)https://blog.getbootstrap.com/2023/03/24

Conclusion :-

Learnt about bootstrap and how to use bootstrap alongwith HTML and CSS , created a webpage using bootstrap and used several bootstrap components such as grid , navbar , breadcrumb , jumbotron , forms etc , the webpage looked more aesthetic than webpage which can only be created using CSS and code for all components is easily available

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-11/08/2023

Assignment No-4a

Aim:-

Write a program in JavaScript to study conditional statements , Loops and functions

LO Mapped:- LO4

Theory:-

Conditional statements in JavaScript are used to make decisions in your code based on certain conditions. They allow your program to execute different blocks of code depending on whether a given condition is true or false. The main types of conditional statements in JavaScript are if, else if, else, and the switch statement.

if statement:
The basic if statement allows you to execute a block of code if a specified condition is true.

```
if (condition) {
   // Code to be executed if the condition is true
}
```

if-else statement:
The if-else statement allows you to execute one block of code if a condition is true, and another block of code if the condition is false.

```
if (condition) {
   // Code to be executed if the condition is true
} else {
   // Code to be executed if the condition is false
}
```

else if statement:
The else if statement allows you to test multiple conditions and execute different blocks of code depending on which condition is true. You can have multiple else if blocks.

```
if (condition1) {
   // Code to be executed if condition1 is true
```

```
} else if (condition2) {
   // Code to be executed if condition2 is true
} else {
   // Code to be executed if none of the conditions are true
}
```

switch statement:

The switch statement allows you to compare a single value against multiple possible cases and execute code based on which case matches the value.

```
switch (expression) {
   case value1:
      // Code to be executed if expression === value1
      break;
   case value2:
      // Code to be executed if expression === value2
      break;
   // ... more cases ...
   default:
      // Code to be executed if none of the cases match
}
```

Here's an example of using these conditional statements:

```
var num = 10;
if (num > 0) {
   console.log("Positive number");
} else if (num < 0) {
   console.log("Negative number");
} else {
   console.log("Zero");
}
```

In this example, the if and else if statements are used to determine whether num is positive, negative, or zero, and the appropriate message is logged to the console based on the value of num.

Loops in JavaScript are used to execute a block of code repeatedly as long as a specified condition is true. There are three main types of loops in JavaScript: for loop, while loop, and do-while loop.

for loop:

The for loop is used when you know the number of iterations you want to perform. It consists of an initialization, a condition, and an iteration statement.

```
for (initialization; condition; iteration) {
   // Code to be executed in each iteration
}
```

while loop:
The while loop is used when you want to execute a block of code as long as a specific condition remains true.

```
while (condition) {
   // Code to be executed as long as the condition is true
}
```

do-while loop:
The do-while loop is similar to the while loop, but it ensures that the code inside the loop is executed at least once, even if the condition is initially false.

```
do {
   // Code to be executed at least once
} while (condition);
```

forEach loop:
The forEach loop is specifically designed to iterate over each element of an array and execute a provided function for each element.

```
array.forEach(function(element, index, array) {
   // Code to be executed for each element
});
```

for-in loop:
The for-in loop is used to iterate over the enumerable properties of an object. It is often used for iterating over the keys (properties) of an object.

```
for (var key in object) {
   // Code to be executed for each key
}
```

for-of loop:
The for-of loop is used to iterate over the values of iterable objects such as arrays, strings, maps, sets, etc.

```
for (var value of iterable) {
   // Code to be executed for each value
}
```

Return function:
Return function contains return statement that contains return statement that returns value to console

```
function add(a, b) {
   return a + b;
}

var sum = add(3, 5); // Call the function and store the result in 'sum'
console.log(sum);    // Output: 8
```

Parameterized Function:
A parameterized function is a function that accepts parameters (arguments) when it is called. These parameters allow you to pass values into the function to be used in its logic.
Example:

```
function greet(name) {
   console.log("Hello, " + name + "!");
}
greet("Alice"); // Output: Hello, Alice!
```

Arrow Function:
Arrow functions are a concise way to write functions in JavaScript. They provide a more compact syntax and automatically bind the this value to the surrounding context.
Example:

```
const multiply = (a, b) => a * b;
console.log(multiply(2, 3)); // Output: 6
```

Anonymous Function:
An anonymous function is a function without a specified name. They are often used as arguments to higher-order functions or assigned to variables.
Example:

```
var calculate = function(a, b) {
   return a * b;
};
console.log(calculate(4, 5)); // Output: 20
```

Callback Function:
A callback function is a function that is passed as an argument to another function and is executed later, typically after an asynchronous operation or an event occurs.
Example:

```
function processInput(input, callback) {
   console.log("Processing input: " + input);
   callback();
}
function complete() {
   console.log("Processing complete.");
}

processInput("data", complete);
```

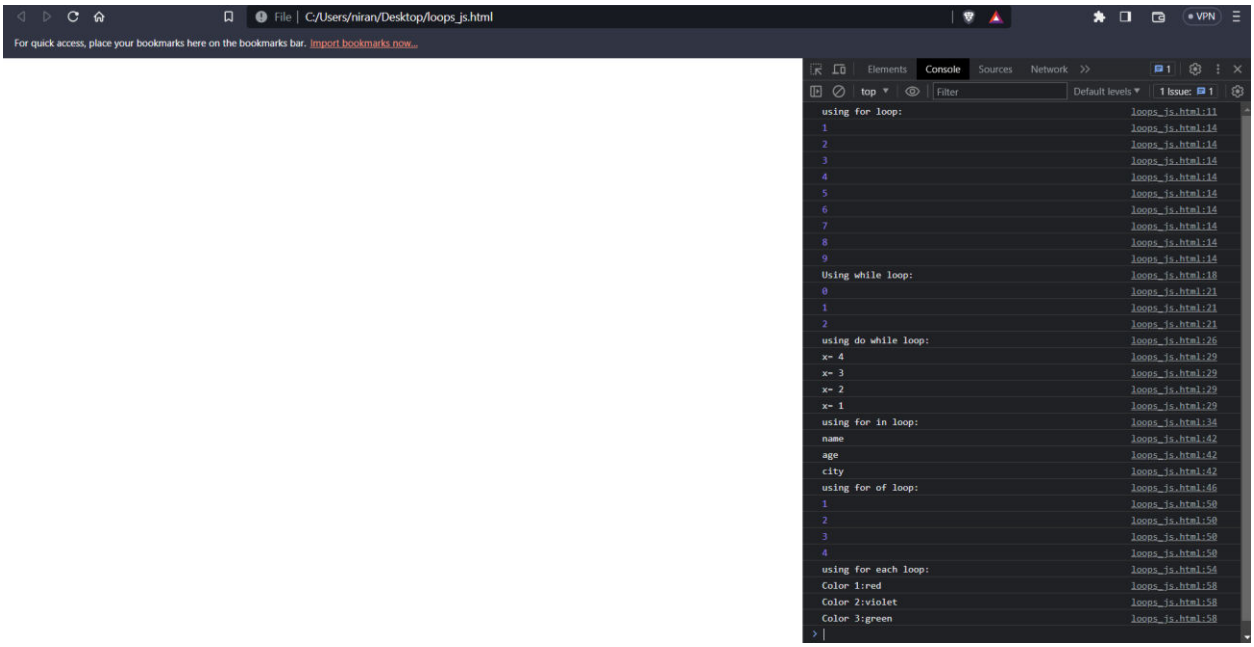
Output:-

1)Code for loops in javascript :-

```
<html>
<head>
   <title>various types of loops</title>
</head>
<body>
<script>
   loopdemo()
   function loopdemo()
   {
      //for loop
      console.log("using for loop:")
      for(let i=1;i<10;i++)
      {
         console.log(i);
      }

      //while loop
      console.log("Using while loop:");
      let count = 0;
      while (count < 3) {
         console.log(count);
         count++;
      }

      //do while loop
      console.log("using do while loop:")
      let x=4;
```

```
        do{
            console.log("x= "+x)
            x--;
        }while(x<4 && x>0);

        //for in loop
        console.log("using for in loop:")
        const identity={
            name:"Aries",
            age:20,
            city:"Mumbai"
        }
        for (const key in identity)
        {
            console.log(key);
        }

        //for of loop
        console.log("using for of loop:")
        const arr=[1,2,3,4]
        for (const x of arr)
        {
            console.log(x)
        }

        //for each loop
        console.log("using for each loop:")
        const clr=["red","violet","green"]
        clr.forEach((x,index)=>
        {
            console.log('Color '+(index+1)+':'+x);
        })
    }
</script>
</body>
</html>
```

2)Output of loops in JavaScript :-

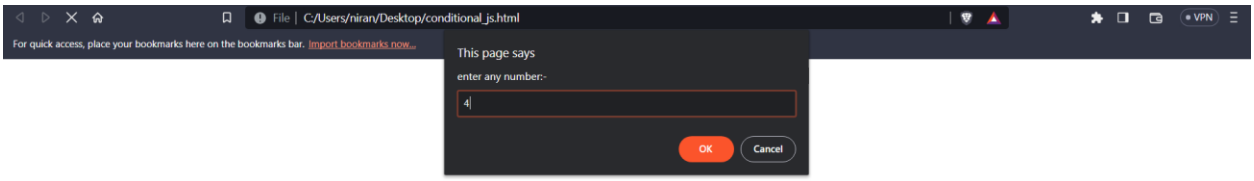


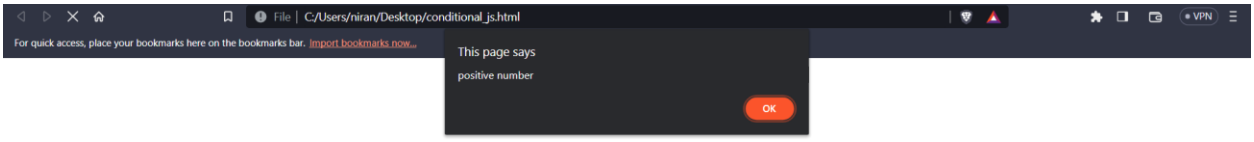
3)Code for conditional statements in JavaScript :-

```
<html>
<head>
   <title>consitional</title>
</head>
<body>
<script>
   let num=prompt("enter any number:-");
   console.log("Conditional statements:-");
   demo(num);
   function demo(num){
    if(num>0)
    {
      alert("positive number");
    }
    else if(num==0)
    {
      alert("zero is entered");
    }
    else
    {
      alert("negative number");
    }
}
```

</script>
</body>
</html>

4)Output for conditionals in JavaScript :-





---

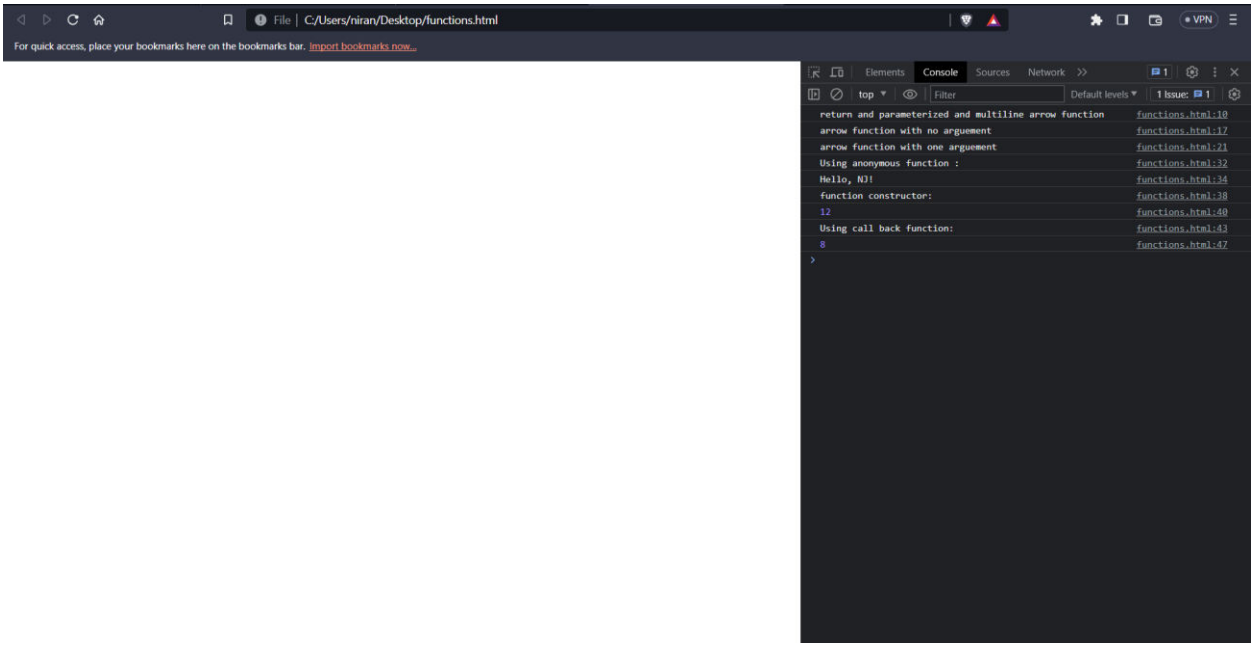
5)Code for functions in javascript :-

<html>
<head>
  <title>functions</title>

```
</head>
<body>
<script>
   //returning function and parameterized and multiline arrow function
   function multip(a,b)
   {
      console.log("return and parameterized and multiline arrow function")
      return a*b;
   }
   multip(2,3);

   //arrow function
   const myfun = () => {
      console.log("arrow function with no arguement");
   }
   myfun();
   const myfun2 = (a) => {
      console.log("arrow function with one arguement");
   }
   myfun2();

   //arrow function as expression
   let age=5;
   let welcome = (age<20) ?
   () => console.log("less than 20"):
   () => console.log("more than 20");

   //anonymous function
   console.log("Using anonymous function :")
   const greet = function(name)
   {   console.log(`Hello, ${name}!`); };
   greet("NJ");

   //function constructor
   console.log("function constructor: ")
   const multiple=new Function('a','b','return a*b');
   console.log(multiple(3,4));

   // callback function
   console.log("Using call back function: ")
```

```
function calculate(num1, num2, operation)
{   return operation(num1, num2); }
const sum = (a, b) => a + b;
console.log(calculate(5, 3, sum));
}
let gen = fun();
console.log(gen.next().value);
console.log(gen.next().value);
console.log(gen.next().value);

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

6)Output for functions in JavaScript :-



Conclusion :-

Learnt about various types of functions , loops , conditional statements in javascript and used them in various programs and observed output for each case , also learnt about new types of loops such as for..of , for…each , for…in in JavaScript .

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-11/08/2023

Assignment No - 4b

Aim:-

Write a program on inheritance , iterators , generators

LO Mapped:- LO4

Theory:-

The ES6 JavaScript supports Object-Oriented programming components such as Object, Class and Methods. Further in Classes we can implement inheritance to make child inherits all methods of Parent Class. This can be done using the extends and super keywords.

We use the extends keyword to implement the inheritance in ES6. The class to be extended is called a base class or parent class. The class that extends the base class or parent class is called the derived class or child class. The super() method in the constructor is used to access all parent's properties and methods that are used by the derived class.

ES6 introduces a new mechanism for traversing data: iteration. Two concepts are central to iteration:

An iterable is a data structure that wants to make its elements accessible to the public. It does so by implementing a method whose key is Symbol.iterator. That method is a factory for iterators.
An iterator is a pointer for traversing the elements of a data structure (think cursors in databases)
Iterable values in Javascript
The following values are iterable:

Arrays
Strings
Maps
Sets
DOM data structures (work in progress)
Plain objects are not iterable.

Symbols offer names that are unique and cannot clash with other property names. Also, Symbol.iterator will return an object called an iterator. This iterator will have a method called nextwhich will return an object with keys value and done.

Prior to ES6, functions in JavaScript followed a run-to completion model.

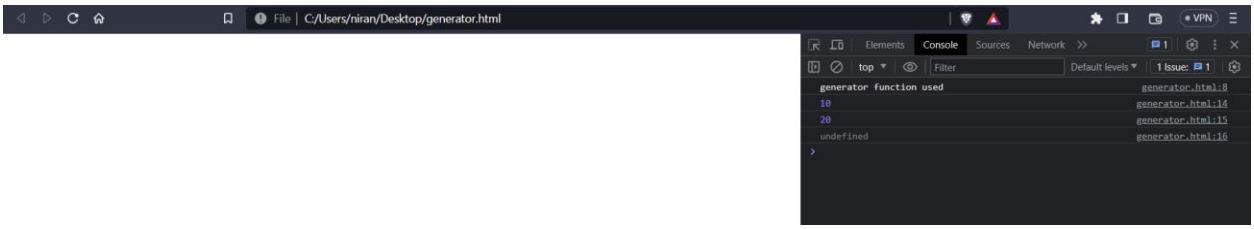
- ES6 introduces functions known as Generator which can stop midway and then continue from where it stopped.
- A generator prefixes the function name with an asterisk * character and contains one or more yield statements.
- The yield keyword returns an iterator object

Output:-

1)Code for Generator :-

```
<html>
<head>
   <title>generator</title>
</head>
<body>
<script>
   //generator
   console.log('generator function used')
   function* fun() {
   yield 10;
   yield 20;
   }
   let gen = fun();
   console.log(gen.next().value);
   console.log(gen.next().value);
   console.log(gen.next().value);

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

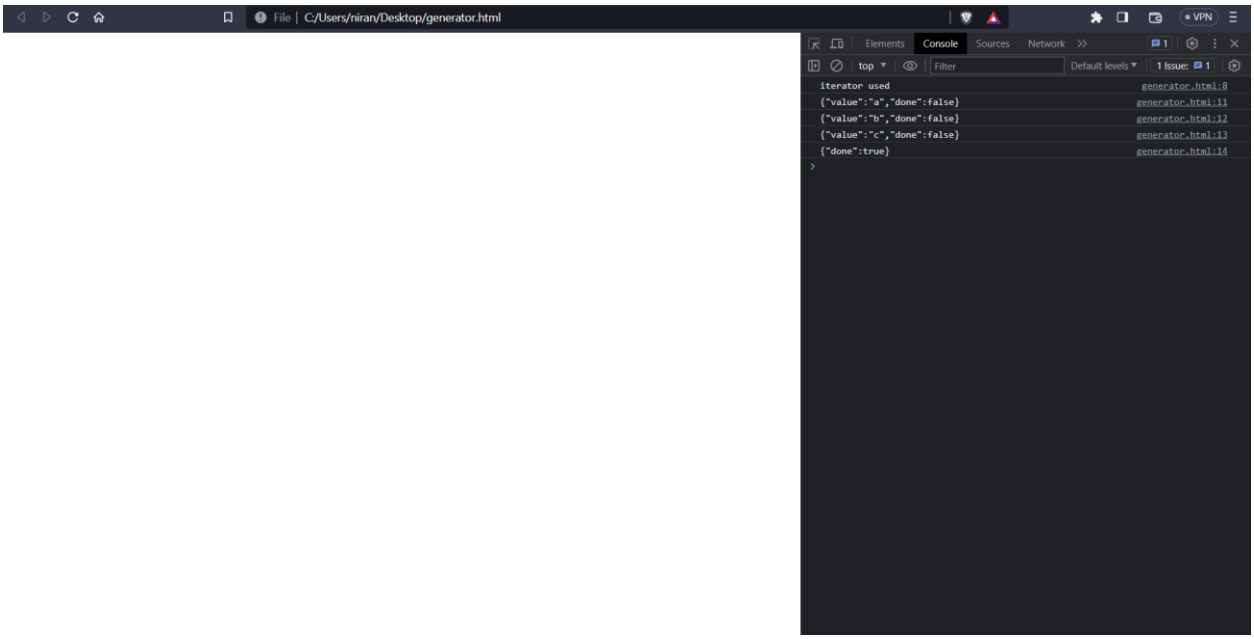
2)Output for generator code :-



3)Code for Iterator :-

```
<html>
<head>
   <title>iterator</title>
</head>
<body>
<script>
   //iterator
       console.log("iterator used")
       const array = ['a', 'b', 'c'];
       const it = array[Symbol.iterator]();
       console.log(JSON.stringify(it.next()));
       console.log(JSON.stringify(it.next()));
       console.log(JSON.stringify(it.next()));
       console.log(JSON.stringify(it.next()));

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

4)Output for Iterator :-

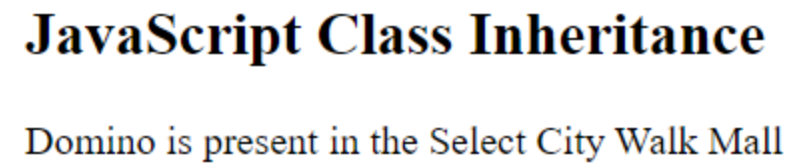


5)Code for Inheritance in JavaScript :-

```
<!DOCTYPE html>
<html>
<body>
   <h2>JavaScript Class Inheritance</h2>
   <p id="demo"></p>


   <script>
      class Mall {
         constructor(shopname) {
            this.shopname = shopname;
         }
         shopispresent() {
            return this.shopname +
              ' is present in the  ';
         }
      }

      class Shop extends Mall {
         constructor(name, mallname) {
```

```
            super(name);
            this.mallname = mallname;
        }
        showshop() {
            return this.shopispresent() +
              this.mallname;
        }
    }

    let newMall = new Shop(
      "Domino", "Select City Walk Mall"
    );
    document.getElementById("demo")
            .innerHTML = newMall.showshop();
  </script>
</body>
</html>
```

6)Output for Inheritance :-



References :-

1) https://www.geeksforgeeks.org/how-to-implement-inheritance-in-es6/
2)https://madasamy.medium.com/explanation-about-iterators-and-generators-in-javascript-es6-f7e669cbe96e

Conclusion :-

Learnt about Iterators , Generators their basic syntax and functioning in JavaScript , also learnt about inheritance in JavaScript , use of constructor , super keyword , extends in JavaScript , also compared inheritance in JavaScript with other programming languages as well.

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-18/08/2023

Assignment No-5a

Aim:-

Write a JavaScript program to study DOM manipulation and CSS manipulation

LO Mapped:- LO4

Theory:-

JavaScript CSS Manipulation: manipulate CSS properties using JavaScript. This allows for dynamic changes to styles based on user interactions or events. Use methods like getElementById, querySelector, or classList to access and modify CSS properties.

querySelector and **querySelectorAll** are JavaScript methods that allow you to select HTML elements from the DOM (Document Object Model) based on CSS selectors.
Examples:-

querySelector
The querySelector method selects the first element that matches a given CSS selector.

```
<!DOCTYPE html>
<html>
<head>
  <title>querySelector Example</title>
</head>
<body>
  <ul>
    <li class="item">Item 1</li>
    <li class="item">Item 2</li>
    <li class="item">Item 3</li>
  </ul>

  <script>
    // Select the first element with the class "item"
```

```
    const firstItem = document.querySelector(".item");
    console.log(firstItem.textContent); // Output: "Item 1"
  </script>
</body>
</html>
```

In this example, querySelector(".item") selects the first <li> element with the class "item."

querySelectorAll

The querySelectorAll method selects all elements that match a given CSS selector and returns them as a NodeList (a collection of nodes). You can iterate through this collection to access each matched element.

```
<!DOCTYPE html>
<html>
<head>
  <title>querySelectorAll Example</title>
</head>
<body>
  <ul>
    <li class="item">Item 1</li>
    <li class="item">Item 2</li>
    <li class="item">Item 3</li>
  </ul>

  <script>
    // Select all elements with the class "item"
    const itemList = document.querySelectorAll(".item");

    // Loop through the NodeList and log the text content of each element
    itemList.forEach((item) => {
      console.log(item.textContent);
    });
  </script>
</body>
</html>
```

In this example, document.querySelectorAll(".item") selects all <li> elements with the class "item" and logs their text content in the console.

DOM (Document Object Model) manipulation refers to the process of using JavaScript to interact with and modify the structure, content, and styles of a web page's Document Object Model. The DOM **represents the hierarchical structure of an HTML document**, allowing you to access and manipulate its elements. Here are some common tasks and techniques for DOM manipulation:

Accessing DOM Elements:

getElementById: You can access an element by its unique ID attribute.
javascript
const element = document.getElementById("myElement");

querySelector and querySelectorAll: These methods allow you to select elements using CSS-style selectors.
const element = document.querySelector(".myClass"); // Selects the first element with the class "myClass"
const elements = document.querySelectorAll("p"); // Selects all <p> elements

getElementsByTagName and getElementsByClassName: These methods select elements by their tag name or class name, respectively.
const paragraphs = document.getElementsByTagName("p"); // Selects all <p> elements
const items = document.getElementsByClassName("item"); // Selects elements with the class

Creating and Appending Elements:

createElement: You can create new DOM elements.
const newDiv = document.createElement("div");
appendChild: You can add new elements as children to existing elements.
parentElement.appendChild(newDiv); // Add the new div as a child of the parent element

Removing Elements:
removeChild: You can remove elements from the DOM.
const parentElement = document.getElementById("parent");
const childElement = document.getElementById("child");
parentElement.removeChild(childElement);

Traversal and Navigation:

You can navigate the DOM tree using properties like parentNode, childNodes, nextSibling, previousSibling, etc., to access adjacent elements.
DOM manipulation is a fundamental part of web development, enabling to create interactive and dynamic web applications by modifying the page's content and behavior in response to user actions or other events.

Output:-

1)Code for DOM manipulation :-

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Manipulation</title>
</head>
<body>

<div id="container">
  <h1>Hello!</h1>
  <p>simple example.</p>
  <p class="remove">This paragraph will be removed.</p>
</div>

<p id="text-element">Original Text</p>

<h1>Heading to be replaced</h1>

<p id="delete">This paragraph will be deleted.</p>

<script>
  // Create a new element , append it to the container div
  const newElement = document.createElement('p');
  newElement.textContent = 'This is a new paragraph element';
  document.getElementById('container').appendChild(newElement);
  console.log('Added a new paragraph:', newElement);

  // Remove an element by its class name
  const elementToRemove = document.getElementsByClassName('remove')[0];
  if (elementToRemove) {
    elementToRemove.parentNode.removeChild(elementToRemove);
```

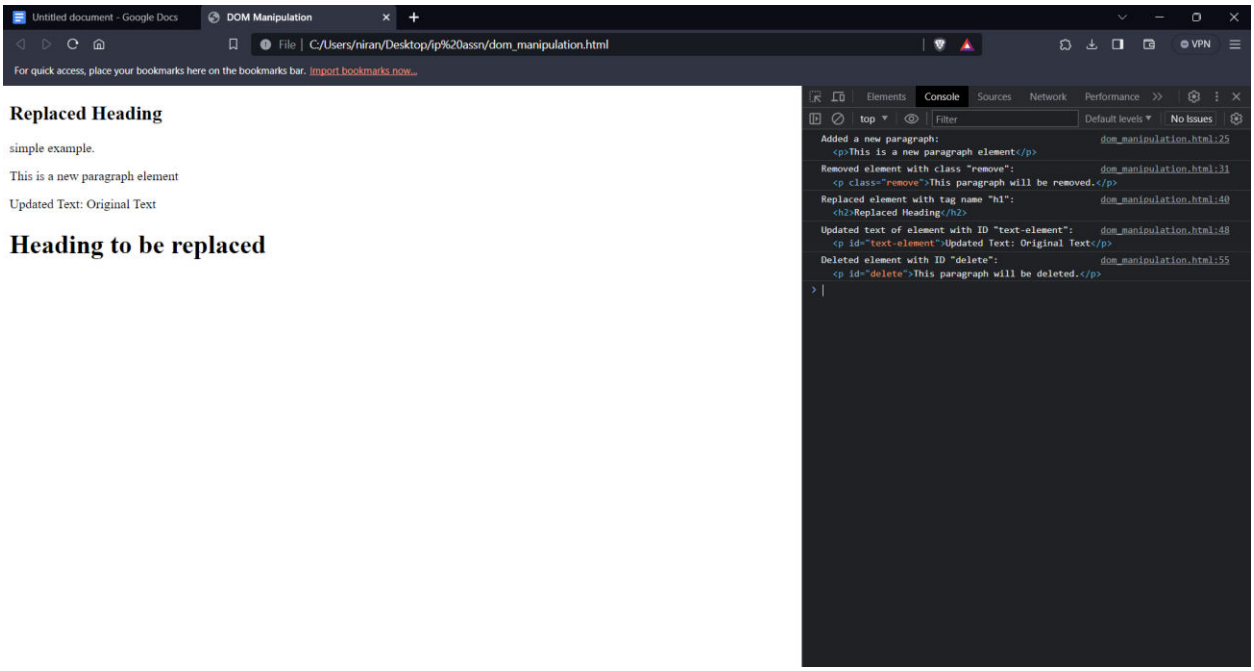
```
  console.log('Removed element with class "remove":', elementToRemove);
 }

 // Replace an element by its tag name
 const elementToReplace = document.getElementsByTagName('h1')[0];
 if (elementToReplace) {
  const newHeading = document.createElement('h2');
  newHeading.textContent = 'Replaced Heading';
  elementToReplace.parentNode.replaceChild(newHeading, elementToReplace);
  console.log('Replaced element with tag name "h1":', newHeading);
 }

 // Get and set text content of an element by its ID
 const textElement = document.getElementById('text-element');
 if (textElement) {
  const currentText = textElement.textContent;
  textElement.textContent = 'Updated Text: ' + currentText;
  console.log('Updated text of element with ID "text-element":', textElement);
 }

 // Delete an element by its ID
 const elementToDelete = document.getElementById('delete');
 if (elementToDelete) {
  elementToDelete.parentNode.removeChild(elementToDelete);
  console.log('Deleted element with ID "delete":', elementToDelete);
 }
</script>
</body>
</html>
```

2)Code for CSS manipulation :-

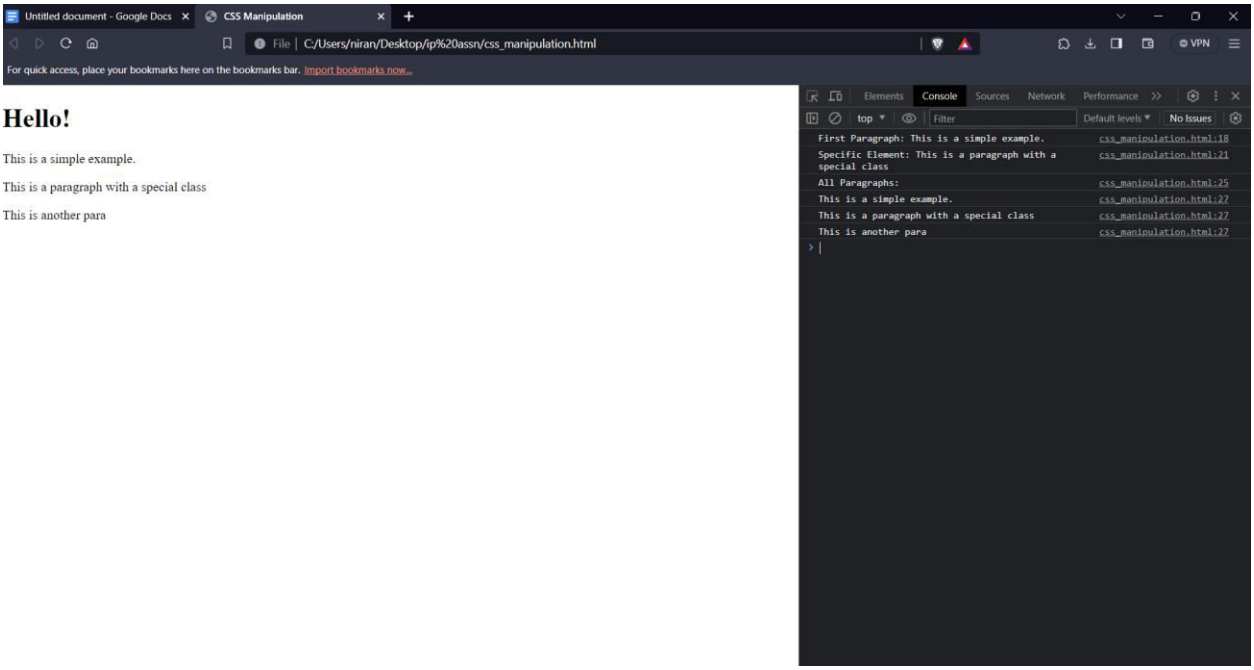
```
<!DOCTYPE html>
<html>
<head>
 <title>CSS Manipulation</title>
</head>
<body>
```

```
<div id="container">
  <h1>Hello!</h1>
  <p>This is a simple example.</p>
  <p class="newclass">This is a paragraph with a special class</p>
  <p>This is another para</p>
</div>

<script>
  // Using querySelector
  const firstParagraph = document.querySelector('p');
  console.log('First Paragraph:', firstParagraph.textContent);

  const specificElement = document.querySelector('#container .newclass');
  console.log('Specific Element:', specificElement.textContent);

  // Using querySelectorAll
  const allParagraphs = document.querySelectorAll('p');
  console.log('All Paragraphs:');
  allParagraphs.forEach(paragraph => {
    console.log(paragraph.textContent);
  });
</script>
</body>
</html>
```

3) Output for DOM manipulation :-



4)Output for CSS manipulation :-



References :-

1)https://www.w3schools.com/js/js_htmldom.asp
2)https://www.tutorialsteacher.com/jquery/jquery-css-manipulation

Conclusion :-

Learnt about DOM manipulation , creation , deletion , renaming of elements in DOM using java script and also used CSS manipulation tools such as query selector and query selector all in different programs , understood function of DOM .

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-18/08/2023

Assignment No-5b

Aim:-

Write a Javascript program.
a. Implement the concept of Promise(callback)
b. Fetch (Client Server Communication)

LO Mapped:- LO4

Theory:-

Promises in JavaScript are a mechanism for handling asynchronous operations, such as making network requests, reading files, or performing any non-blocking task. They provide a way to manage asynchronous code and make it more readable and maintainable.

A promise represents a value that may be available in the future, either as a resolved value or a reason for failure (rejected). Promises have three states:

Pending: The initial state when the promise is created, neither fulfilled nor rejected.
Fulfilled: The state when the asynchronous operation is successfully completed, and a value is available.
Rejected: The state when an error occurs during the asynchronous operation, and a reason for failure is available.

example :-

```
// Creating a simple promise
const myPromise = new Promise((resolve, reject) => {
  // Simulate an asynchronous operation (e.g., fetching data)
  setTimeout(() => {
    const success = true; // Change this to false to simulate a rejection
    if (success) {
      resolve("Promise resolved successfully!");
    } else {
      reject("Promise rejected with an error.");
```

```
    }
  }, 2000); // Simulating a 2-second delay
});

// Using the promise
myPromise
  .then((result) => {
    console.log(result); // Executed when the promise is resolved
  })
  .catch((error) => {
    console.error(error); // Executed when the promise is rejected
  });

// can also use the .finally() method to execute code regardless of whether the promise is resolved or rejected
myPromise.finally(() => {
  console.log("Promise finally block executed.");
});
```

In the example above:

create a new Promise with a function that receives resolve and reject functions as arguments.
Inside the promise, simulate an asynchronous operation with a setTimeout.
If the operation is successful, call resolve with a value; otherwise, we call reject with an error message.
use the .then() method to define what happens when the promise is resolved and the .catch() method to handle rejections.
The .finally() method can be used to execute code that should run regardless of whether the promise is resolved or rejected.
Promises help make asynchronous code more readable and easier to reason about, especially when dealing with multiple asynchronous operations or chaining promises together. They are a fundamental part of modern JavaScript and are commonly used in web development for tasks like handling AJAX requests or working with asynchronous APIs.

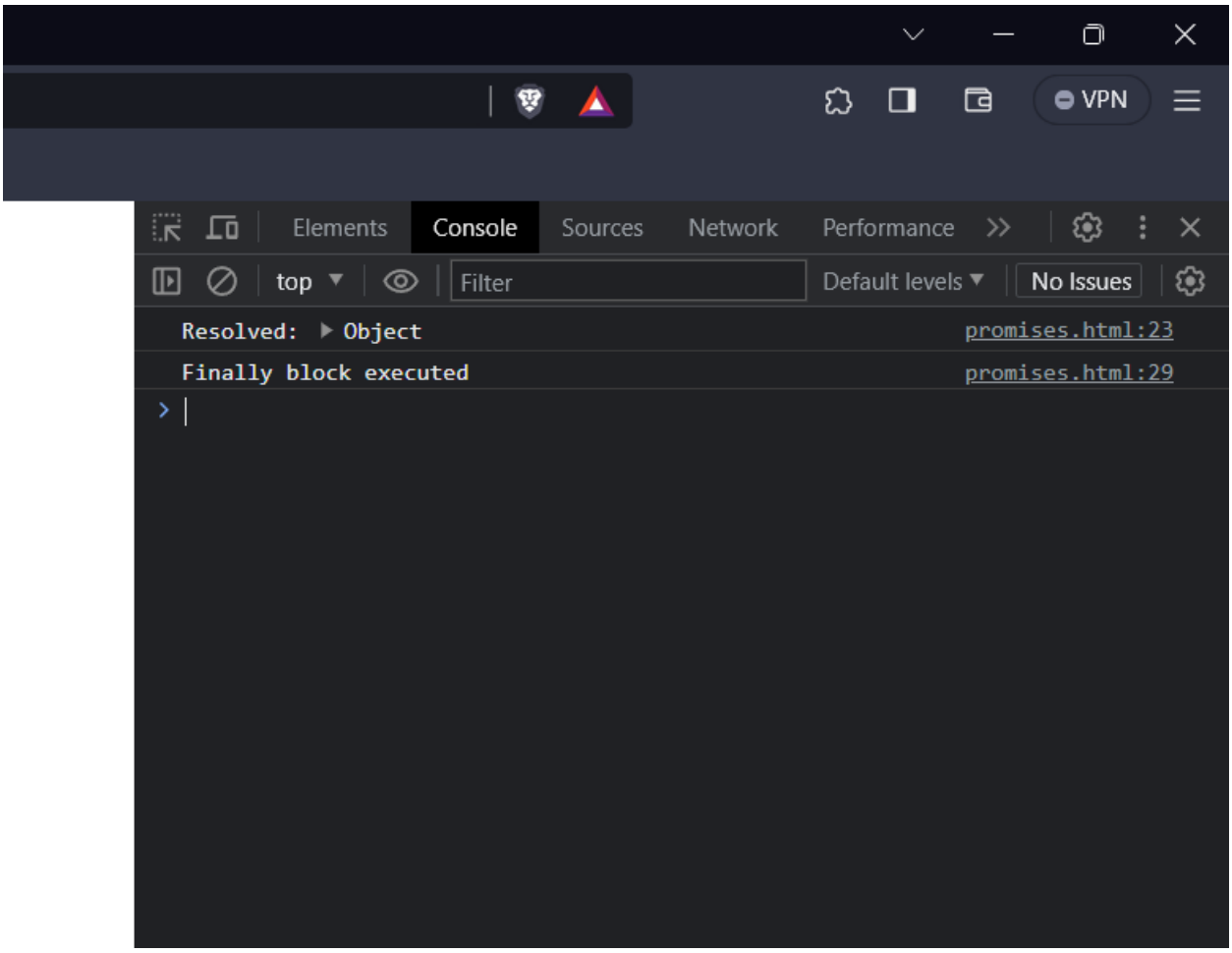
Output:-

1)Code for promises :-

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Promises in JS</title>
</head>
<body>
<script>
// Using Fetch API to make a GET request to JSONPlaceholder API for a specific post
function fetchPostById(postId) {
  return fetch(`https://jsonplaceholder.typicode.com/posts/${postId}`)
    .then(response => {
      if (!response.ok) {
        throw new Error('API call failed');
      }
      return response.json();
    });
}

// Using Promises
const postId = 1;
fetchPostById(postId)
  .then(post => {
    console.log('Resolved:', post);
  })
  .catch(error => {
    console.error('Rejected:', error);
  })
  .finally(() => {
    console.log('Finally block executed');
  });

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

2)Output for promises code :-



References :-

1)https://www.w3schools.com/js/js_promise.asp
2)https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

Conclusion :-

Learnt about promises in ES6 , also used concept of callback function in ES6 , hence learnt more about callback function and used fetch api to get request fo=rom JSON placeholder api

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-25/08/2023

Assignment No-6a

Aim:-

WAP to implement concept of props and state . Create functional component and pass current date as props and with class component , on button display both date and time

LO Mapped:- LO5

Theory:-

In React, props and state are two fundamental concepts that are used to manage and pass data around in components. They play a crucial role in building dynamic and interactive user interfaces. Let's take a look at each of them with examples:

Props (Properties):
Props are read-only properties that are passed from a parent component to a child component. They allow you to pass data from one component to another and are used to make components reusable and modular. Props cannot be modified by the child component.

Example :-

```
import React from 'react';
import ChildComponent from './ChildComponent';

function ParentComponent() {
  const greeting = 'Hello,';

  return (
    <div>
      <ChildComponent message={greeting} />
    </div>
  );
}

export default ParentComponent;
```

```
// ChildComponent.js
import React from 'react';

function ChildComponent(props) {
  return <p>{props.message} World!</p>;
}

export default ChildComponent;
```

In this example, the ParentComponent passes the message prop to the ChildComponent, which then displays "Hello, World!" using the passed prop.

State:
State is used to manage the internal state of a component. It is mutable and can be modified by the component itself using the setState function. When state changes, React re-renders the component to reflect the updated state.

Example :-

```
import React, { Component } from 'react';
class CounterComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0,
    };
  }

  incrementCount = () => {
    this.setState(prevState => ({
      count: prevState.count + 1,
    }));
  };

  render() {
    return (
      <div>
        <p>Count: {this.state.count}</p>
        <button onClick={this.incrementCount}>Increment</button>
```

```
    </div>
   );
  }
}

export default CounterComponent;
```

In this example,CounterComponent class component is created. The component's state includes a property called count which starts at 0. The incrementCount method uses setState to update the count when the "Increment" button is clicked. The component's render method displays the current count and the button.

When the button is clicked, React will automatically update the component's state, triggering a re-render of the component with the updated count value.

Output:-

1)Code for classcomp.jsx :-

```
import React, { Component } from 'react';
class ClassComponent extends Component {
  constructor() {
    super();
    this.state = {
      color: 'black',
      datetime: new Date().toLocaleString()
    };
  }

  changeColor = () => {
    const colors = ['red', 'green', 'blue'];
    const randomColor = colors[Math.floor(Math.random() * colors.length)];
    this.setState({ color: randomColor });
  }

  render() {
    const { color, datetime } = this.state;
    return (
      <div>
        <h2 style={{ color }}>Class Component</h2>
        <p>Current Date and Time: {datetime}</p>
```

```
        <button onClick={this.changeColor}>Change Color</button>
      </div>
    );
  }
}
export default ClassComponent;
```

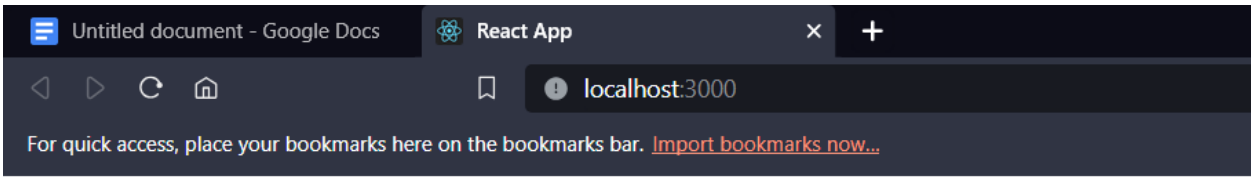
2)Code for functional.jsx :-

```
import React from 'react';

function FunctionalComponent(props) {
  const { datetime } = props;
  return (
    <div>
      <h2>Functional Component</h2>
      <p>Current Date and Time: {datetime}</p>
    </div>
  );
}

export default FunctionalComponent;
```

3)Code for app.js

```
import React from 'react';
import FunctionalComponent from './components/classcomp';
import ClassComponent from './components/functional';

function App() {
  const currentDatetime = new Date().toLocaleString();

  return (
    <div>
      <FunctionalComponent datetime={currentDatetime} />
      <ClassComponent datetime={currentDatetime} />
    </div>
  );
}

export default App;
```

4)Output :-



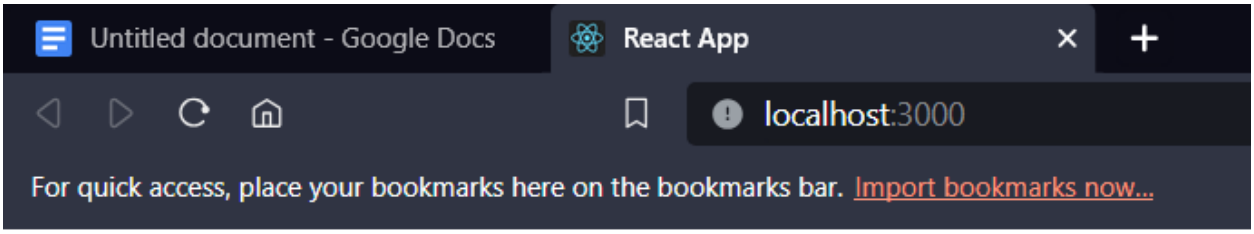
Class Component

Current Date and Time: 8/28/2023, 12:08:28 PM

Change Color

Functional Component

Current Date and Time: 8/28/2023, 12:08:28 PM



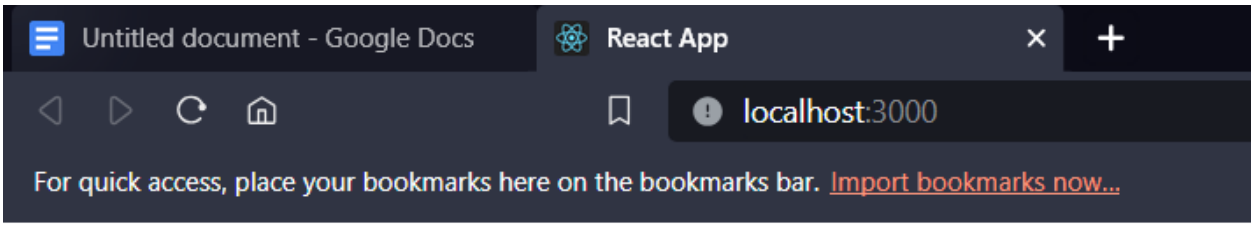
Class Component

Current Date and Time: 8/28/2023, 12:08:28 PM

Change Color

Functional Component

Current Date and Time: 8/28/2023, 12:08:28 PM



References :-

1)https://www.geeksforgeeks.org/reactjs-state-vs-props/#:~:text=Props%20are%20read%2Donly.,data%20inside%20a%20component%20itself.

2)https://www.javatpoint.com/react-state-vs-props

Conclusion :-

Learnt about usage of state and props in react , learnt about basic syntax of these and also discovered about use-case of these and implemented a program to demonstrate function of these two .

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-25/08/2023

Assignment No-6b

Aim:-

WAP to implement the concept of forms and events.
Create a React JS registration Form consisting of textbox, textarea, selection input, check box, radio button, submit button and reset button handling onSubmit, onClick and keyDown events.

LO Mapped:- LO5

Theory:-

Forms in ReactJS:
1. Controlled Components: In React, form elements like input fields, textareas, and select boxes are turned into "controlled components." This means that their values are controlled by the component's state. You store the input value in the component's state and update it as the user interacts with the form.

2. State Management: React components maintain their state, which includes form data. When the user interacts with the form, React updates the component state, and this, in turn, updates the form elements, ensuring that they always display the correct values.

3. Event Handling: React provides a consistent way to handle events, such as clicks, form submissions, and keypresses. You can attach event handlers to specific elements or components to respond to user interactions.

Event Handling in ReactJS:
1. Event Binding: You can bind event handlers to React components using the onClick, onChange, onSubmit, and similar event attributes. These event handlers are functions that get executed when a specific event occurs.

2. Event Object: When an event occurs, React passes an event object to the corresponding event handler. This object contains information about the event, such as the target element and event type. You can access this object to gather more

details about the event.

3. Preventing Default Behavior: You can use the preventDefault method of the event object to prevent the default behavior of certain events. For example, you can prevent a form from submitting when handling the onSubmit event, allowing you to perform custom validation or data processing.

4. Updating State: Event handlers often involve updating the component's state. By calling setState(), you can modify the state data, which triggers a re-render of the component and updates the user interface to reflect the new state.

5. Passing Data: You can pass data from the event handler to other parts of your application or to child components using React's props system. This enables you to share information between components efficiently.

1. Textbox (<input type="text">):
• A textbox is a standard input field where users can enter text information, such as their name.
• Event Handling: The onChange event is commonly used with textboxes to capture changes in the input value as the user types.

2. Textarea (<textarea>):
• A textarea is a multi-line input field used for longer text entries, like an address or comments.
• Event Handling: Similar to textboxes, you can use the onChange event to track changes in the textarea's content.

3. Selection Input (<select>) with Options:
• A selection input, often called a dropdown or select box, allows users to choose from a list of predefined options.
• Event Handling: The onChange event is used here as well to detect when the user selects an option.

4. Checkbox (<input type="checkbox">):
• A checkbox is used for binary choices, such as subscribing to a newsletter.
• Event Handling: The onChange event can be used to monitor whether the checkbox is checked or unchecked.

5. Radio Buttons (<input type="radio">):
• Radio buttons are used when there are multiple mutually exclusive choices

(e.g., gender).
• Event Handling: Like checkboxes, radio buttons also use the onChange event to determine which option the user has selected.

6. Submit Button (<button type="submit">):
• The submit button is used to send the form data to the server for processing.
• Event Handling: The onSubmit event is associated with the form element and is triggered when the user clicks the submit button. It's used to handle form submission logic, such as data validation and sending data to the server.

7. Reset Button (<button type="reset">):
• The reset button is used to clear or reset the form's input fields to their initial values.
• Event Handling: The onClick event is used with the reset button to trigger a function that clears the form fields, returning them to their default or empty State.

8. Keydown Event:
• The keydown event occurs when a key on the keyboard is pressed down.
• While not typically associated with form submission, you can use the keydown event to capture specific keyboard inputs (e.g., Enter key) to trigger custom actions, such as submitting the form when the Enter key is pressed.

Output:-

1)Code for loginform.js :-

```
import React, { Component } from 'react';
class LoginFormClass extends Component {
  constructor(props) {
    super(props);
    this.state = {
      username: '',
      password: '',
      errorMessage: '',
    };
  }

  handleUsernameChange = (event) => {
```

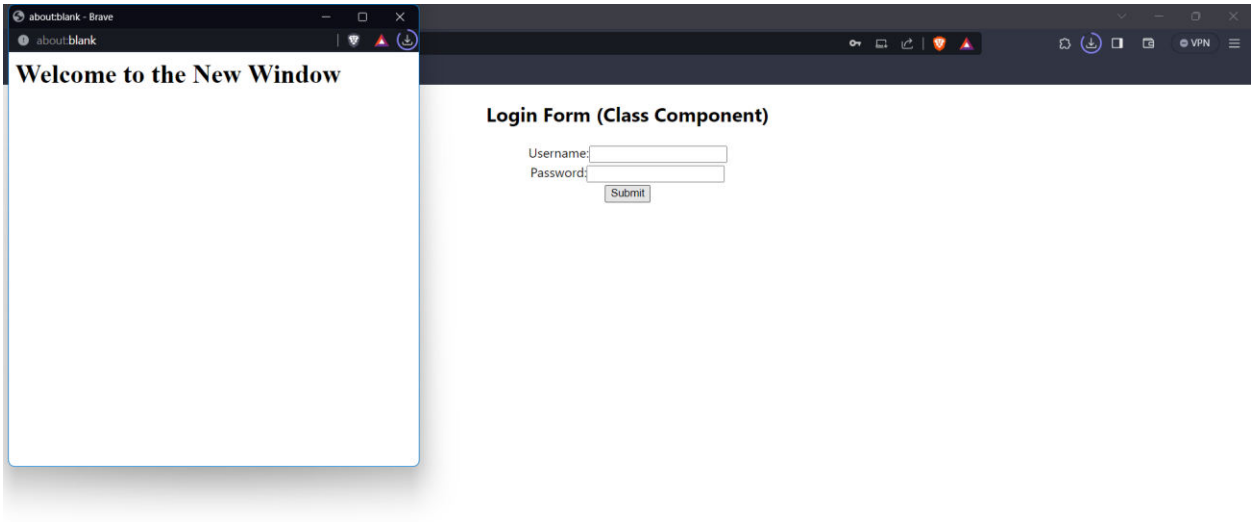
```
  const inputUsername = event.target.value;
  if (/^[a-zA-Z]*$/.test(inputUsername) || inputUsername === '') {
    this.setState({ username: inputUsername });
  }
};

handlePasswordChange = (event) => {
  this.setState({ password: event.target.value });
};

handleSubmit = (event) => {
  event.preventDefault();
  if (this.state.username !== 'user' || this.state.password !== 'password') {
    this.setState({ errorMessage: 'Invalid username or password.' });
    return;
  }

  const newWindow = window.open('', '_blank', 'width=500,height=500');
  if (newWindow) {
    newWindow.document.write('<h1>Welcome to the New Window</h1>');
  }

  this.setState({
    username: '',
    password: '',
    errorMessage: '',
  });
};

render() {
  return (
    <div>
      <form onSubmit={this.handleSubmit}>
        <div>
          <label>Username:</label>
          <input
            type="text"
            value={this.state.username}
            onChange={this.handleUsernameChange}
          />
```

```
        </div>
        <div>
          <label>Password:</label>
          <input
            type="password"
            value={this.state.password}
            onChange={this.handlePasswordChange}
          />
        </div>
        <button type="submit">Submit</button>
      </form>
      {this.state.errorMessage && <p>{this.state.errorMessage}</p>}
    </div>
  );
 }
}

export default LoginFormClass;
```

2)Code for app.js :-

```
import React from 'react';
import LoginFormClass from './components/LoginFormClass';
import './App.css';

function App() {
 return (
  <div className="App">
   <h2>Login Form (Class Component)</h2>
   <LoginFormClass />
  </div>
 );
}

export default App;
```

3)Output



References :-

1)https://legacy.reactjs.org/docs/forms.html
2)https://www.w3schools.com/react/react_forms.asp

Conclusion :-

Learnt about react JS forms and used various form elements such as textbox , textarea , input selection , checkbox , radiobutton , submit button and implemented , learnt about various react events like onkeypress , onkeydown etc.

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-01/09/2023

Assignment No-7a

Aim:-

WAP to implement ReactJS Router
Create more than two class or functional components and implement program for Routing using browserrouter.

LO Mapped:- LO5

Theory:-

React Router is a powerful library for handling routing in React applications. It allows you to create single-page applications (SPAs) where different components are displayed based on the URL, enabling navigation without refreshing the entire page. React Router uses the `BrowserRouter` component as one of its routing mechanisms. Let's explore how to use `BrowserRouter` in a React application:

1. Installation:

First, make sure you have React and React Router installed. You can install React Router using npm or yarn:
npm install react-router-dom

2. Importing BrowserRouter:

In your React application, import `BrowserRouter` from 'react-router-dom':
javascript import { BrowserRouter } from 'react-router-dom';

3. Wrapping the App Component:

Wrap your entire application or the root component (usually `App.js`) with the `BrowserRouter` component. This should be done in your main entry file :

Here's how `BrowserRouter` works in this context:

1. **`BrowserRouter` Component:**

The `<BrowserRouter>` component wraps the entire application. It provides the context for routing and is responsible for synchronizing the application's UI with the URL.

2. **`Navbar` Component:** This component is rendered outside of the `<Routes>` component, which means it will appear on every page of your application. It likely contains navigation links (e.g., Home, Contact, Brands, Registration), allowing users to navigate to different parts of your app.

3. **`Routes` Component:** Inside the `<BrowserRouter>`, the `<Routes>` component is used to define the route configuration for your application. It acts as a container for individual `<Route>` components.

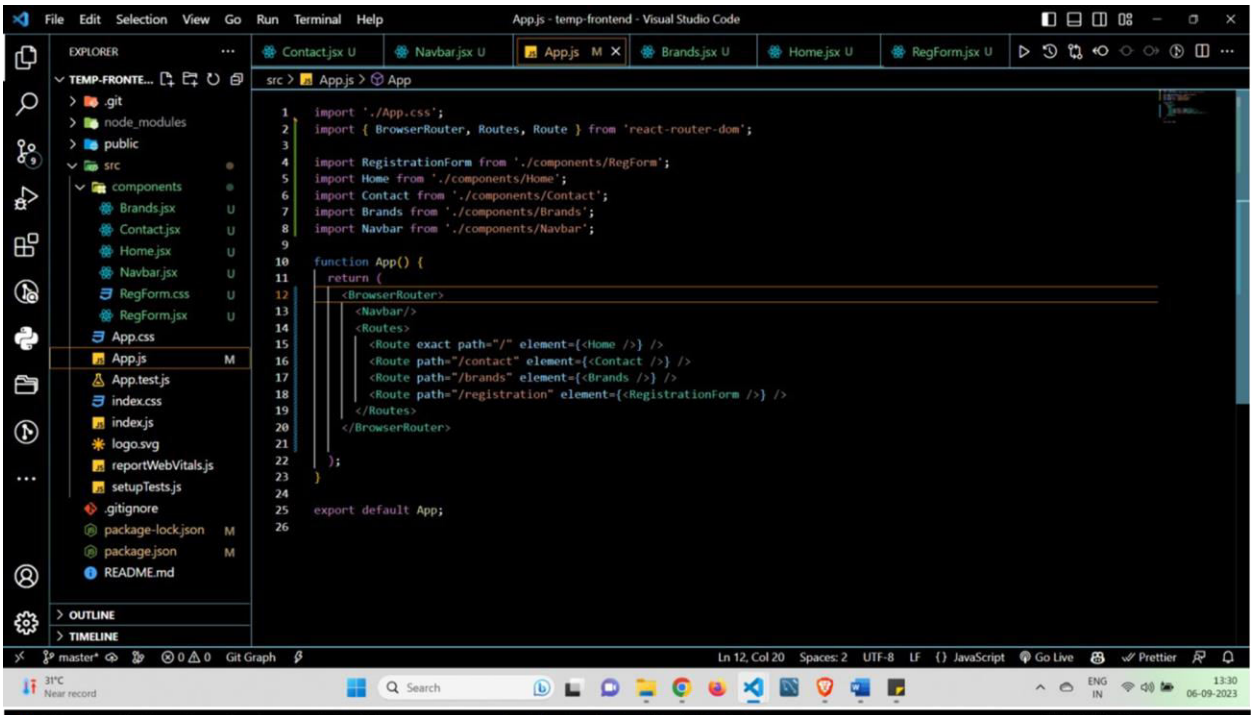
4. **`Route` Components:** Each `<Route>` component specifies how a particular URL should map to a React component. For example:

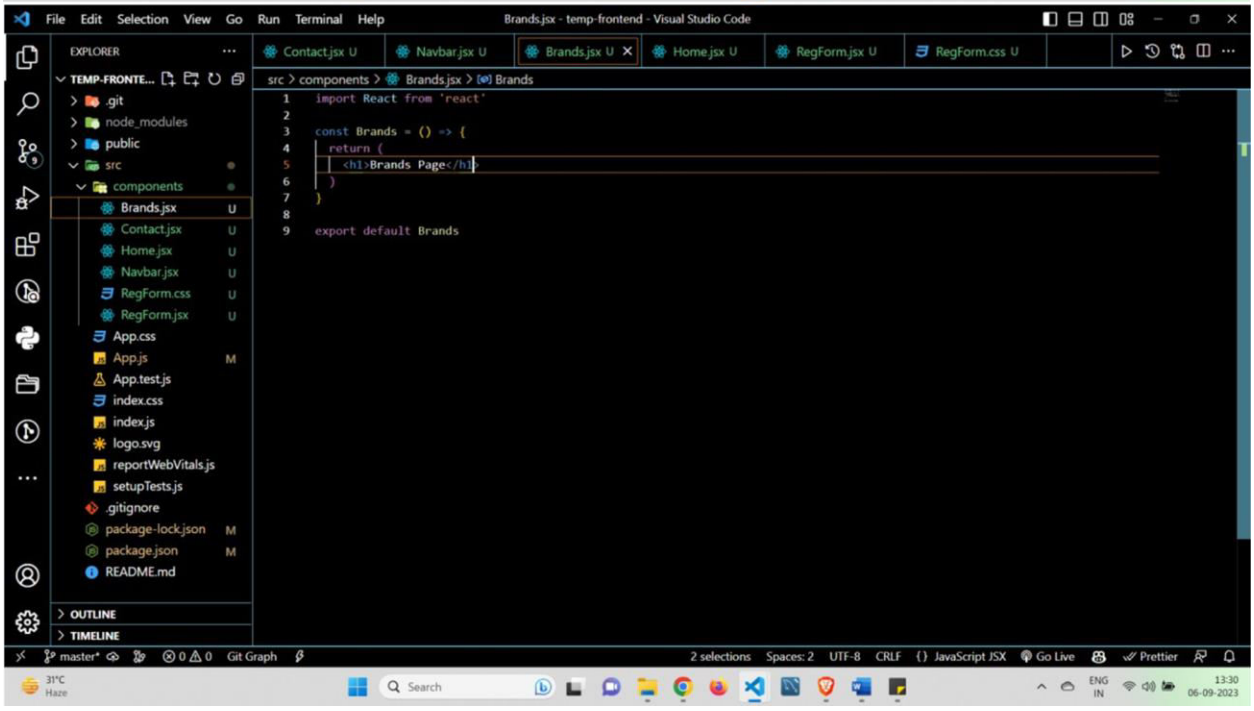
- The `<Route>` with `path="/" element={<Home />}` indicates that when the URL is exactly "/", the `Home` component should be rendered.
- The `<Route>` with `path="/contact" element={<Contact />}` maps the "/contact" URL to the `Contact` component, and so on.

By using `BrowserRouter`, you enable client-side routing. When a user clicks a link or enters a URL, React Router will determine which component to render based on the URL's path..

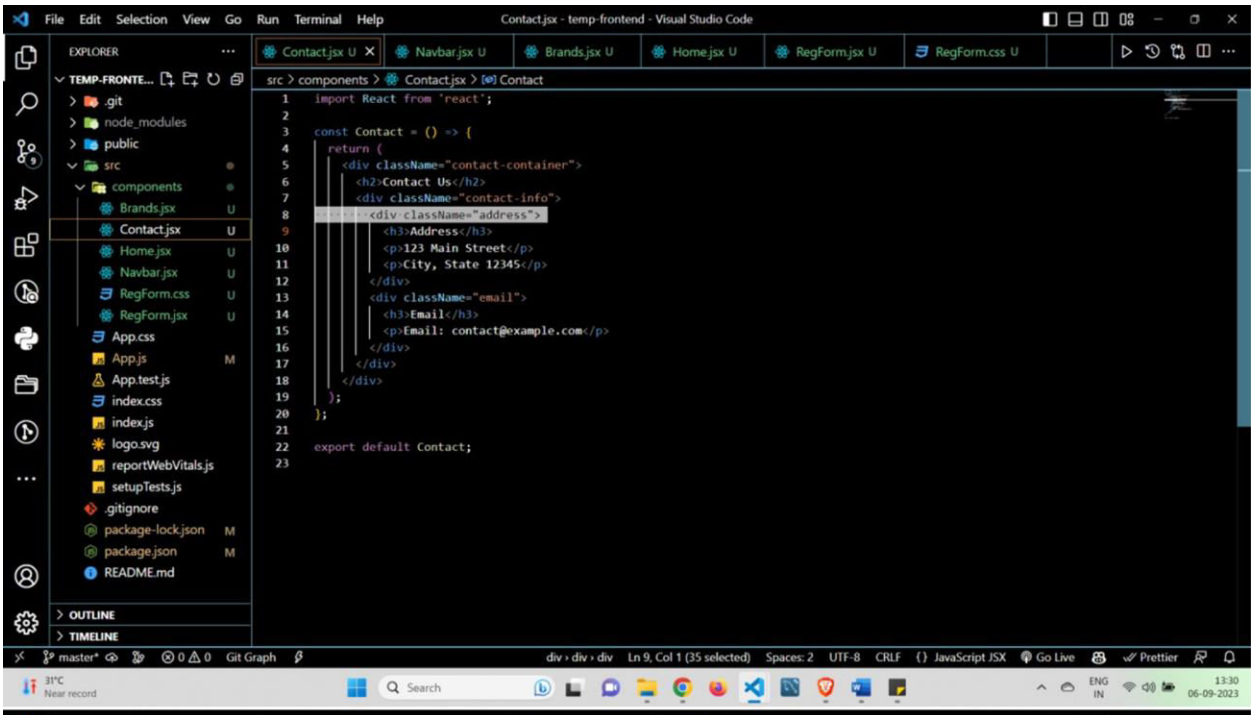
Output:-

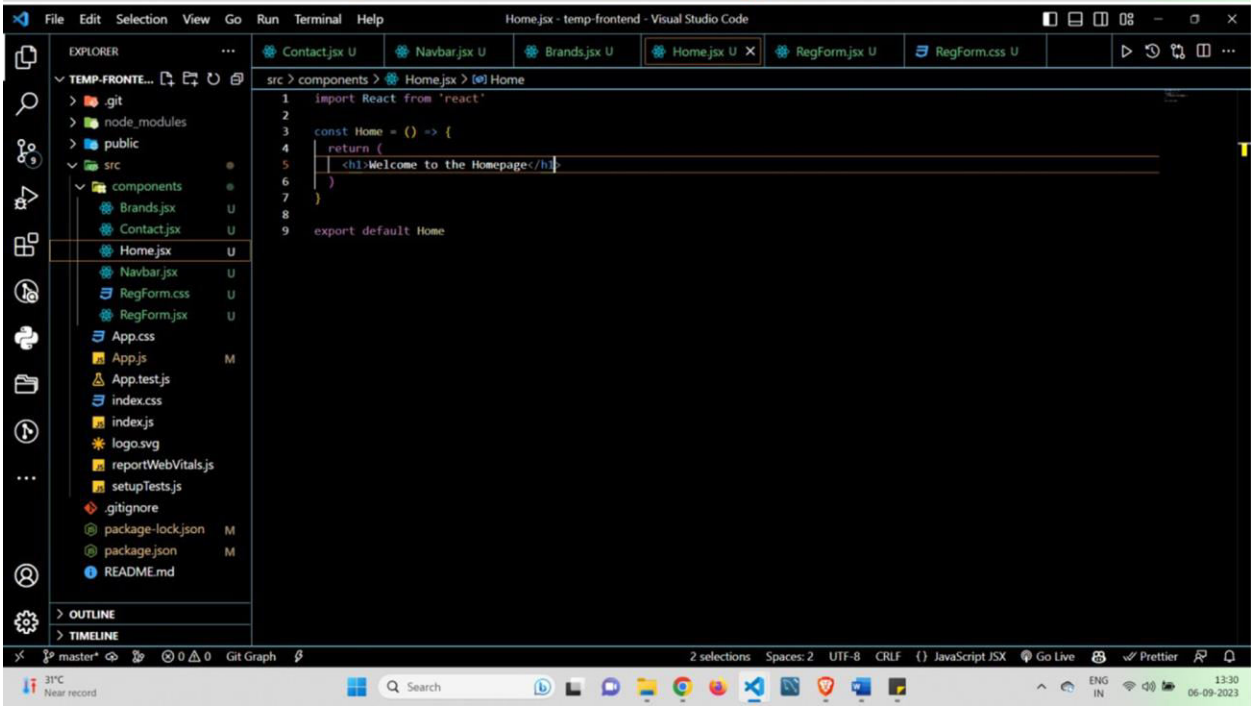
1)Code for app.js and brands.jsx



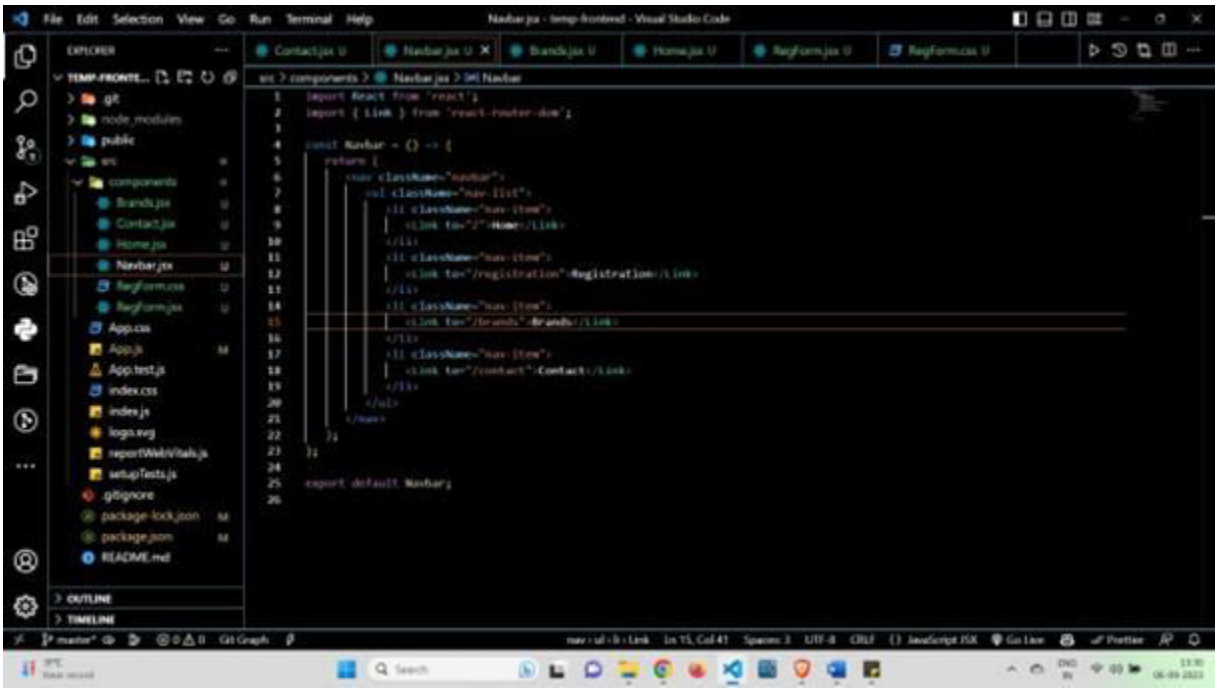


2)Code for navbar.jsx





3)Code for regform.jsx

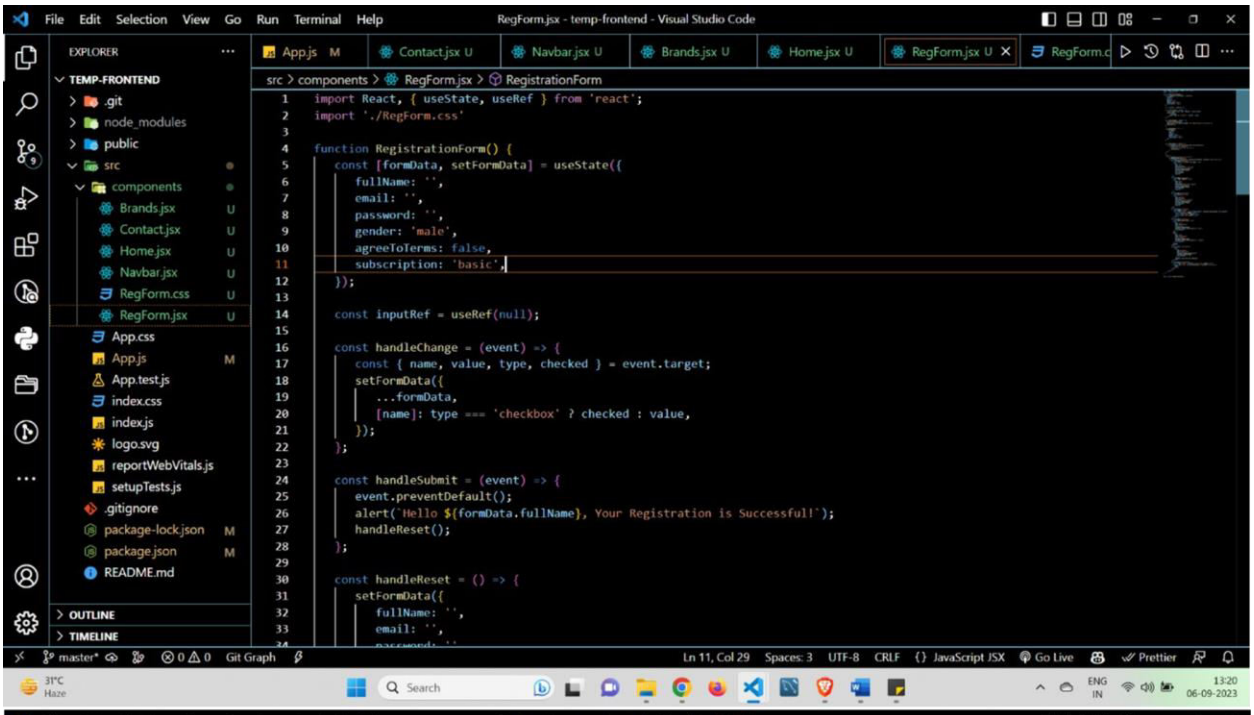

File Edit Selection View Go Run Terminal Help
Navbar.jsx - temp-frontend - Visual Studio Code
EXPLORER
TEMP-FRONTE...
git
node_modules
public
src
components
Brands.jsx
Contact.jsx
Home.jsx
Navbar.jsx
RegForm.css
RegForm.jsx
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md
OUTLINE
TIMELINE
Contact.jsx U
Navbar.jsx U
Brands.jsx U
Home.jsx U
RegForm.jsx U
RegForm.css U
src > components > Navbar.jsx > Navbar
import React from 'react';
import { Link } from 'react-router-dom';
const Navbar = () => {
return (
<nav className="navbar">
<ul className="nav-list">
<li className="nav-item">
<Link to="/">Home</Link>

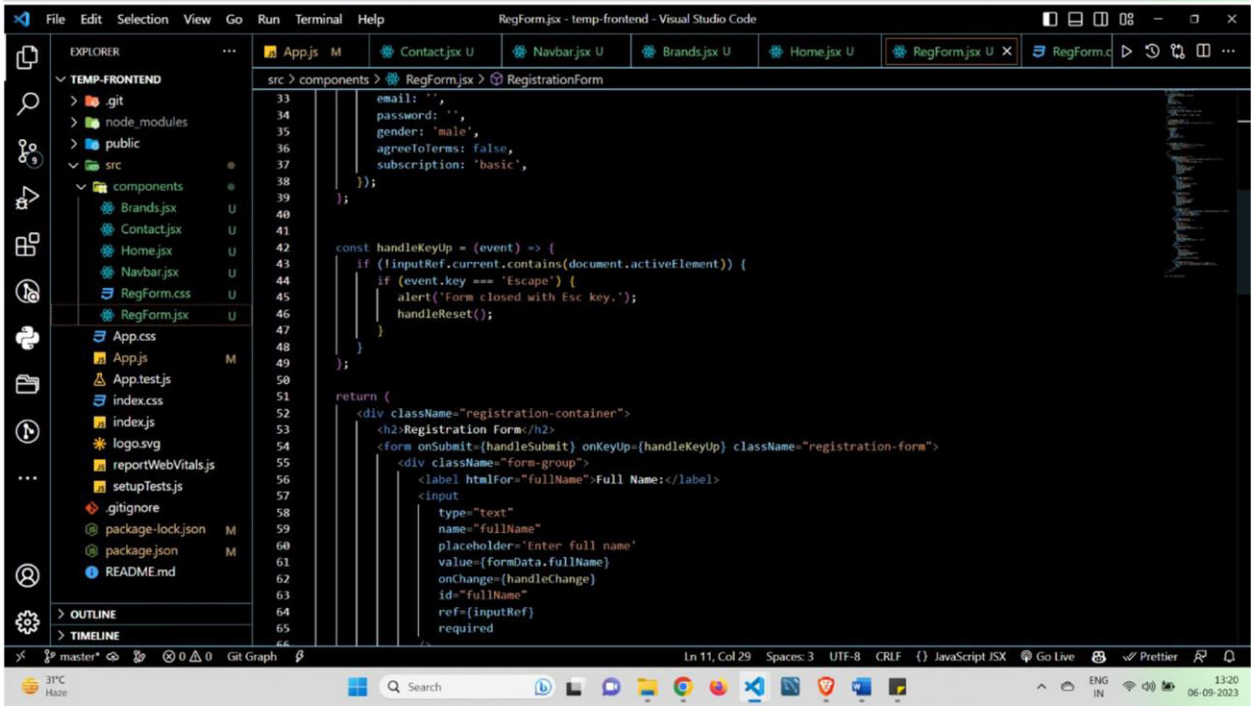
<li className="nav-item">
<Link to="/registration">Registration</Link>

<li className="nav-item">
<Link to="/brands">Brands</Link>

<li className="nav-item">
<Link to="/contact">Contact</Link>

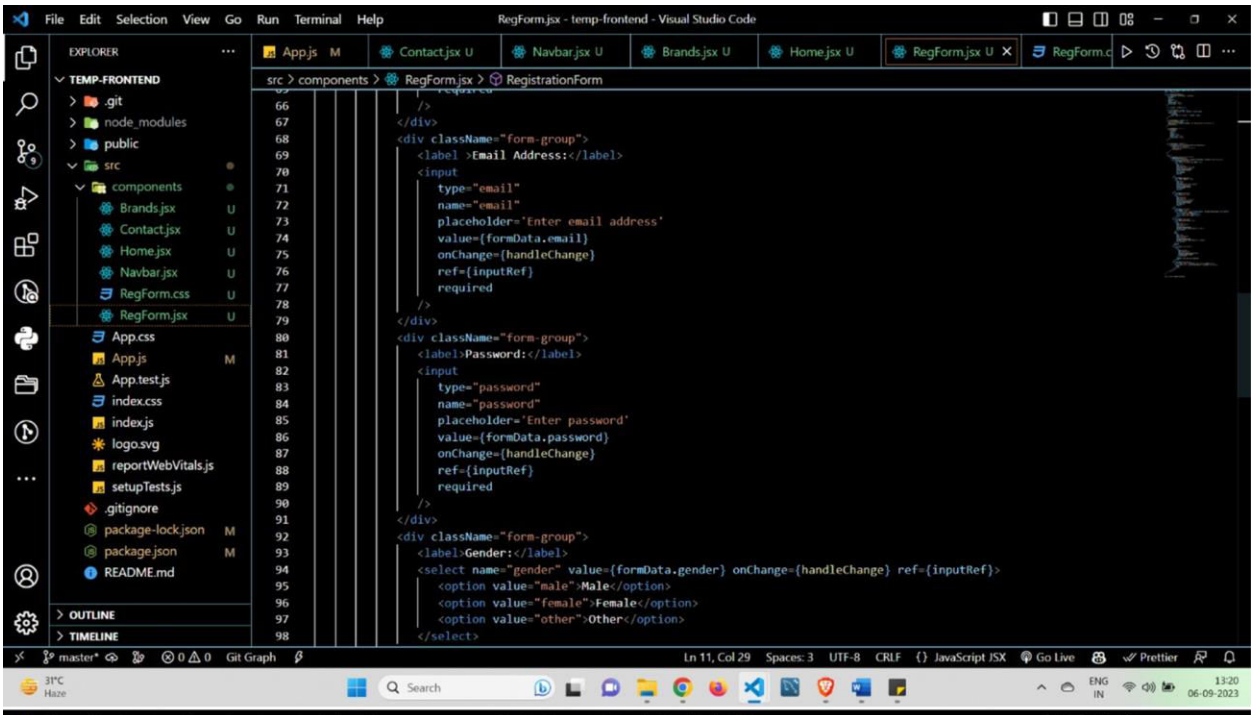
</nav>
);
};
export default Navbar;


File Edit Selection View Go Run Terminal Help
RegForm.jsx - temp-frontend - Visual Studio Code
EXPLORER
TEMP-FRONTEND
.git
node_modules
public
src
components
Brands.jsx U
Contact.jsx U
Home.jsx U
Navbar.jsx U
RegForm.css U
RegForm.jsx U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json M
package.json M
README.md
OUTLINE
TIMELINE
App.js M
Contact.jsx U
Navbar.jsx U
Brands.jsx U
Home.jsx U
RegForm.jsx U
src > components > RegForm.jsx > RegistrationForm
import React, { useState, useRef } from 'react';
import './RegForm.css'
function RegistrationForm() {
const [formData, setFormData] = useState({
fullName: '',
email: '',
password: '',
gender: 'male',
agreeToTerms: false,
subscription: 'basic',
});
const inputRef = useRef(null);
const handleChange = (event) => {
const { name, value, type, checked } = event.target;
setFormData({
...formData,
[name]: type === 'checkbox' ? checked : value,
});
};
const handleSubmit = (event) => {
event.preventDefault();
alert(`Hello ${formData.fullName}, Your Registration is Successful!`);
handleReset();
};
const handleReset = () => {
setFormData({
fullName: '',
email: '',
Ln 11, Col 29 Spaces: 3 UTF-8 CRLF {} JavaScript JSX Go Live Prettier
master* 0 0 Git Graph
31°C Haze
Search
ENG IN
13:20 06-09-2023


File Edit Selection View Go Run Terminal Help
RegForm.jsx - temp-frontend - Visual Studio Code
EXPLORER
TEMP-FRONTEND
.git
node_modules
public
src
components
Brands.jsx U
Contact.jsx U
Home.jsx U
Navbar.jsx U
RegForm.css U
RegForm.jsx U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json M
package.json M
README.md
OUTLINE
TIMELINE
App.js M
Contact.jsx U
Navbar.jsx U
Brands.jsx U
Home.jsx U
RegForm.jsx U
src > components > RegForm.jsx > RegistrationForm
email: '',
password: '',
gender: 'male',
agreeToTerms: false,
subscription: 'basic',
});
};
const handleKeyUp = (event) => {
if (!inputRef.current.contains(document.activeElement)) {
if (event.key === 'Escape') {
alert('Form closed with Esc key.');
handleReset();
}
}
};
return (

<h2>Registration Form</h2>
<form onSubmit={handleSubmit} onKeyUp={handleKeyUp} className="registration-form">

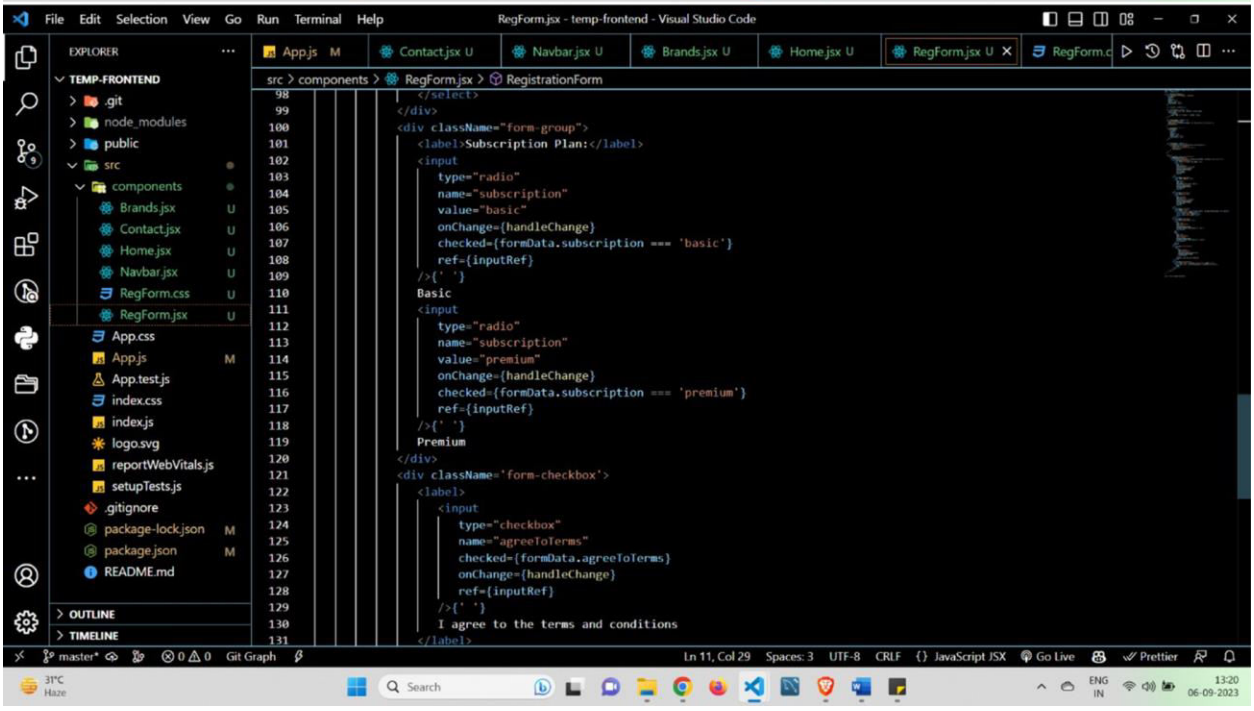
<label htmlFor="fullName">Full Name:</label>
<input
type="text"
name="fullName"
placeholder='Enter full name'
value={formData.fullName}
onChange={handleChange}
id="fullName"
ref={inputRef}
required
Ln 11, Col 29 Spaces: 3 UTF-8 CRLF {} JavaScript JSX Go Live Prettier
master* 0 0 Git Graph
31°C Haze
Search
ENG IN
13:20 06-09-2023


File Edit Selection View Go Run Terminal Help
RegForm.jsx - temp-frontend - Visual Studio Code
EXPLORER
TEMP-FRONTEND
.git
node_modules
public
src
components
Brands.jsx U
Contact.jsx U
Home.jsx U
Navbar.jsx U
RegForm.css U
RegForm.jsx U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json M
package.json M
README.md
OUTLINE
TIMELINE
App.js M
Contact.jsx U
Navbar.jsx U
Brands.jsx U
Home.jsx U
RegForm.jsx U
src > components > RegForm.jsx > RegistrationForm
/>

<label >Email Address:</label>
<input
type="email"
name="email"
placeholder='Enter email address'
value={formData.email}
onChange={handleChange}
ref={inputRef}
required
/>

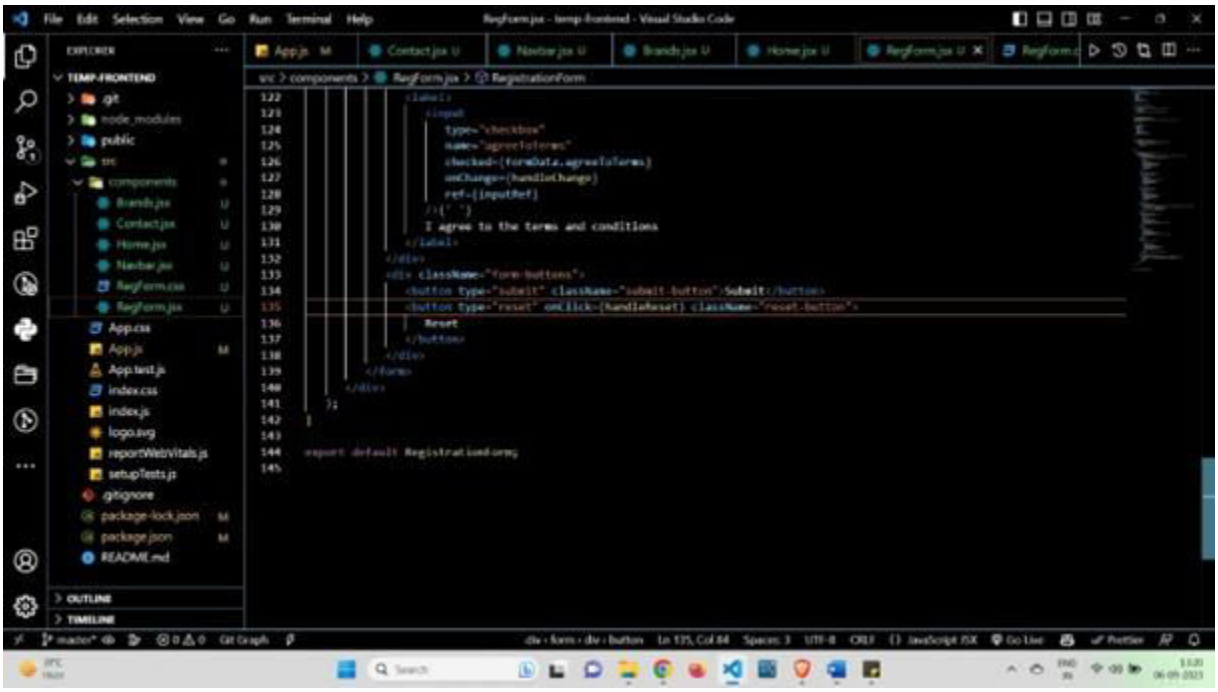
<label>Password:</label>
<input
type="password"
name="password"
placeholder='Enter password'
value={formData.password}
onChange={handleChange}
ref={inputRef}
required
/>

<label>Gender:</label>
<select name="gender" value={formData.gender} onChange={handleChange} ref={inputRef}>
<option value="male">Male</option>
<option value="female">Female</option>
<option value="other">Other</option>
</select>
master* 0 0 Git Graph
Ln 11, Col 29 Spaces: 3 UTF-8 CRLF {} JavaScript JSX Go Live Prettier
31°C Haze
Search
ENG IN
13:20 06-09-2023

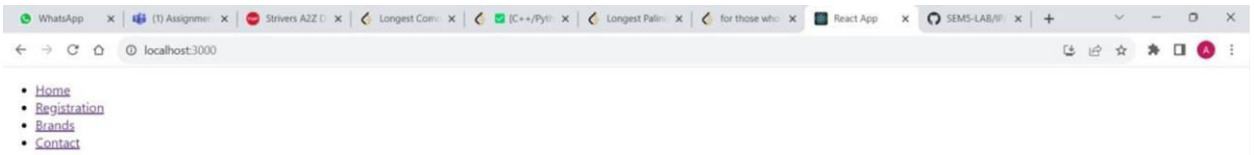

File Edit Selection View Go Run Terminal Help
RegForm.jsx - temp-frontend - Visual Studio Code
EXPLORER
TEMP-FRONTEND
.git
node_modules
public
src
components
Brands.jsx U
Contact.jsx U
Home.jsx U
Navbar.jsx U
RegForm.css U
RegForm.jsx U
App.css
App.js M
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json M
package.json M
README.md
OUTLINE
TIMELINE
App.js M
Contact.jsx U
Navbar.jsx U
Brands.jsx U
Home.jsx U
RegForm.jsx U
src > components > RegForm.jsx > RegistrationForm
</select>

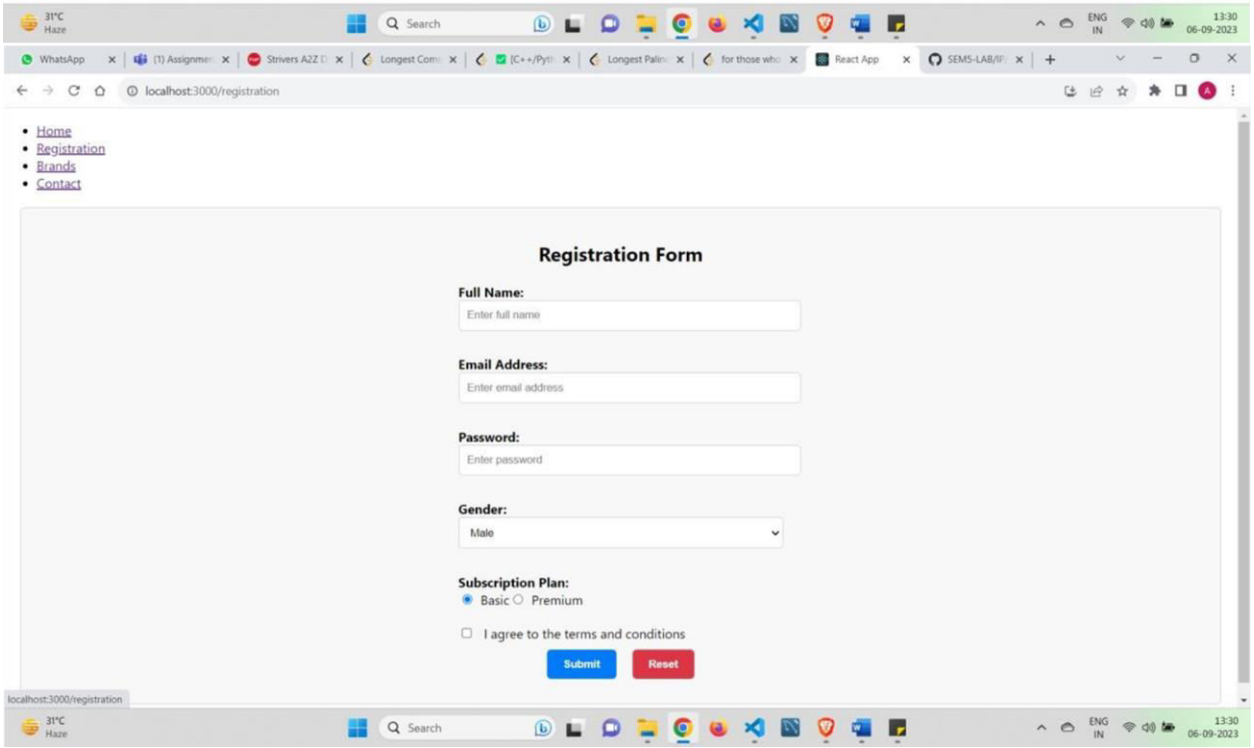
<label>Subscription Plan:</label>
<input
type="radio"
name="subscription"
value="basic"
onChange={handleChange}
checked={formData.subscription === 'basic'}
ref={inputRef}
/>{' '}
Basic
<input
type="radio"
name="subscription"
value="premium"
onChange={handleChange}
checked={formData.subscription === 'premium'}
ref={inputRef}
/>{' '}
Premium

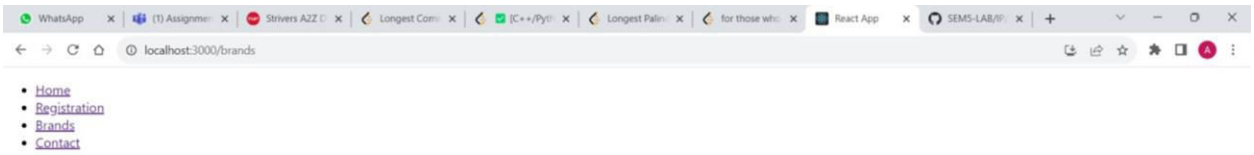
<label>
<input
type="checkbox"
name="agreeToTerms"
checked={formData.agreeToTerms}
onChange={handleChange}
ref={inputRef}
/>{' '}
I agree to the terms and conditions
</label>
master* 0 0 Git Graph
Ln 11, Col 29 Spaces: 3 UTF-8 CRLF {} JavaScript JSX Go Live Prettier
31°C Haze
Search
ENG IN
13:20 06-09-2023

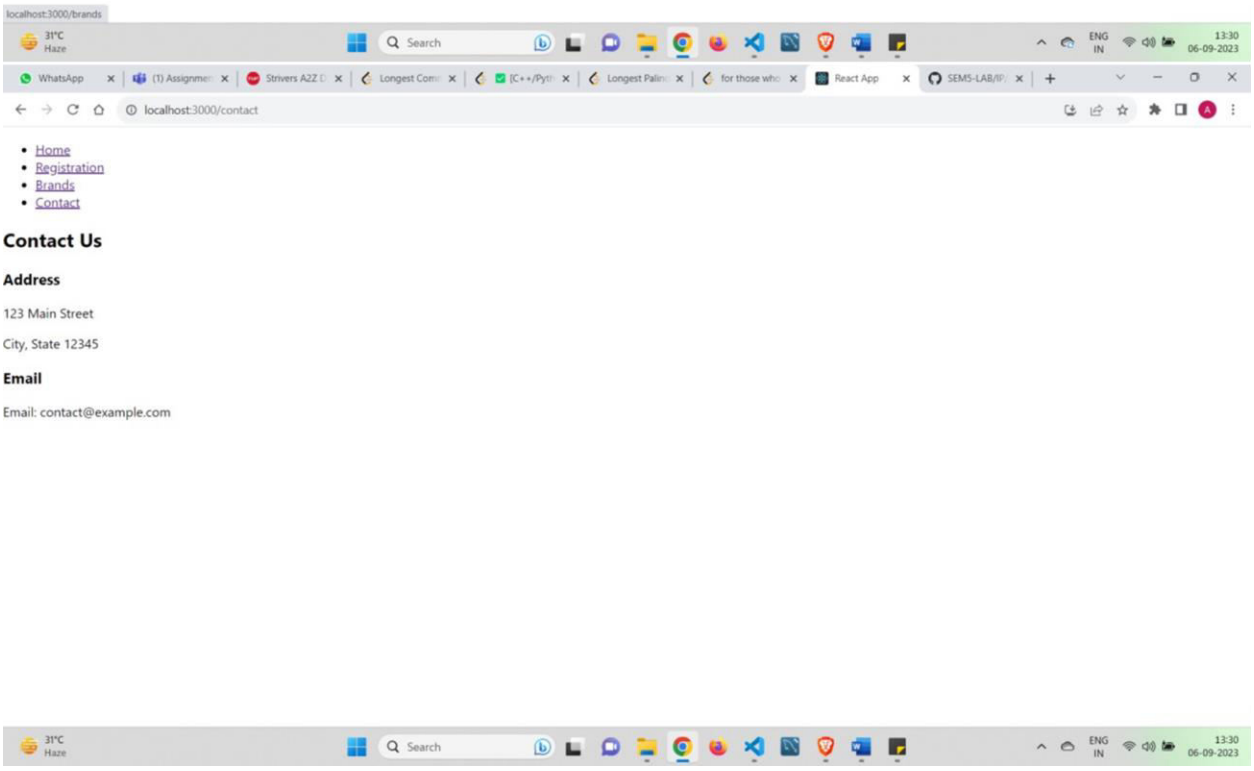


5)Output screenshots


localhost:3000
Home
Registration
Brands
Contact
Welcome to the Homepage


localhost:3000/registration
Home
Registration
Brands
Contact
Registration Form
Full Name:
Enter full name
Email Address:
Enter email address
Password:
Enter password
Gender:
Male
Subscription Plan:
Basic
Premium
I agree to the terms and conditions
Submit
Reset





References :-

1)https://www.w3schools.com/react/react_router.asp
2)https://www.tutorialspoint.com/nodejs/nodejs_repl_terminal.htm

Conclusion :-

Learnt about ReactJS router , various components related to it , explored more about its usage and created a react app by using various functional components and implemented concept of browser router in it

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-08/09/2023

Assignment No-7b

Aim:-

WAP to implement concept of React hooks

LO Mapped:- LO5

Theory:-

React Hooks are a feature introduced in React 16.8 that allow you to use state and other React features in functional components. Before Hooks, state and other React features were only available in class components.

Some of the most commonly used React Hooks are:

useState: useState allows functional components to manage state. It returns an array with the current state value and a function to update it. Here's an example:

Eg:-

```
import React, { useState } from 'react'
function Counter() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

useEffect: useEffect allows you to perform side effects in your components, such as data fetching, manually changing the DOM, or setting up subscriptions. It takes two arguments: a function to run the side effect and an array of dependencies that determine when the effect should run.

Eg:-

```
import React, { useState, useEffect } from 'react';
function Example() {
  const [count, setCount] = useState(0);
  useEffect(() => {
    document.title = `Count: ${count}`;
  }, [count]);
  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

useContext: useContext allows you to access the context provided by a Context.Provider higher up in the component tree. It is particularly useful for managing global state and avoiding prop drilling.

Eg:-

```
import React, { useContext } from 'react';
const MyContext = React.createContext();
function MyComponent() {
  const contextValue = useContext(MyContext);
  return <p>Value from context: {contextValue}</p>;
}
```

useReducer: useReducer is a hook for managing complex state logic. It is similar to useState, but it uses a reducer function to determine the new state based on the previous state and an action. It's often used when state transitions are more complex than simple updates.

Eg:-

```
import React, { useReducer } from 'react';
function counterReducer(state, action) {
  switch (action.type) {
    case 'INCREMENT':
      return { count: state.count + 1 };
    case 'DECREMENT':
      return { count: state.count - 1 };
    default:
      return state;
  }
```

```
}

function Counter() {
  const [state, dispatch] = useReducer(counterReducer, { count: 0 });
  return (
    <div>
      <p>Count: {state.count}</p>
      <button onClick={() => dispatch({ type: 'INCREMENT' })}>Increment</button>
      <button onClick={() => dispatch({ type: 'DECREMENT' })}>Decrement</button>
    </div>
  );
}
```

React provides several other built-in hooks, and you can also create your custom hooks to reuse stateful logic across different components. Hooks make it easier to manage state and side effects in functional components, making React development more flexible and expressive.

Output:-

1)Code for app.js :-

```
import React from 'react';
import NameForm from './NameForm';

function App() {
  return (
    <div className="App">
      <NameForm />
    </div>
  );
}
export default App;
```

2)Code for index.js :-

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
ReactDOM.render(
  <React.StrictMode>
```

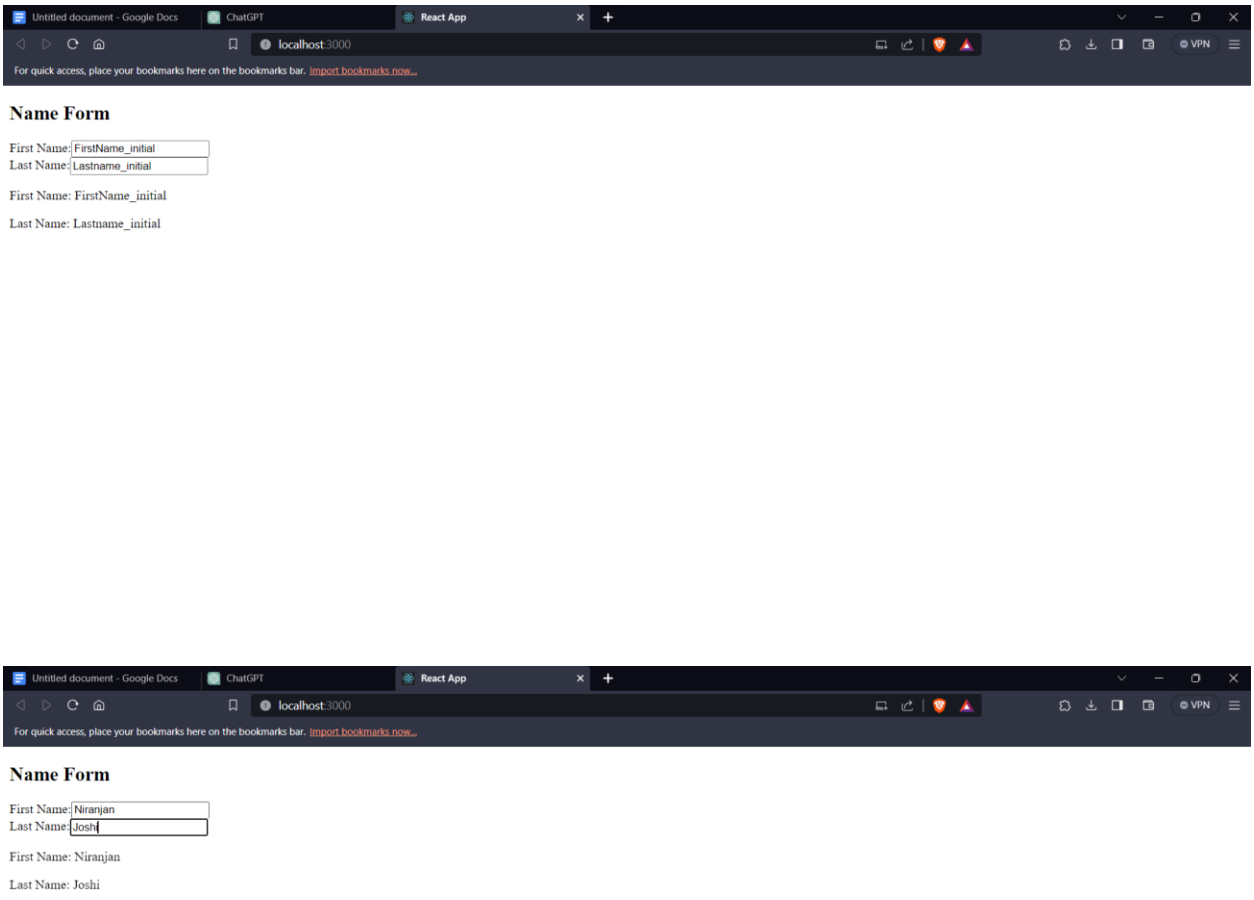
```
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

3)Code for NameForm.js

```
import React, { useState } from 'react';
function NameForm() {
  const [firstName, setFirstName] = useState('FirstName_initial');
  const [lastName, setLastName] = useState('Lastname_initial');
  const handleFirstNameChange = (e) => {
    setFirstName(e.target.value);
  };
  const handleLastNameChange = (e) => {
    setLastName(e.target.value);
  };
  return (
    <div>
      <h2>Name Form</h2>
      <div>
        <label htmlFor="firstName">First Name:</label>
        <input
          type="text"
          id="firstName"
          value={firstName}
          onChange={handleFirstNameChange}
        />
      </div>
      <div>
        <label htmlFor="lastName">Last Name:</label>
        <input
          type="text"
          id="lastName"
          value={lastName}
          onChange={handleLastNameChange}
        />
      </div>
      <div>
        <p>First Name: {firstName}</p>
```

```
        <p>Last Name: {lastName}</p>
      </div>
    </div>
  );
}
export default NameForm;
```

4)Output screenshots



References :-

1)https://legacy.reactjs.org/docs/hooks-intro.html
2)https://www.w3schools.com/react/react_hooks.asp

Conclusion :-

Learnt about react hooks , their usage and also explored more about most commonly used react hooks like usestate , useeffect and written a simple program to demonstrate usage of react hooks

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-26/09/2023

Assignment No-8

Aim:-

WAP to implement concept of REPL in commandline

LO Mapped:- LO6

Theory:-

The **node:repl** module exports the repl.REPLServer class. While running, instances of repl.REPLServer will accept individual lines of user input, evaluate those according to a user-defined evaluation function, then output the result. Input and output may be from stdin and stdout, respectively, or may be connected to any Node.js stream.

Instances of **repl.REPLServer** support automatic completion of inputs, completion preview, simplistic Emacs-style line editing, multi-line inputs, ZSH-like reverse-i-search, ZSH-like substring-based history search, ANSI-styled output, saving and restoring current REPL session state, error recovery, and customizable evaluation functions. Terminals that do not support ANSI styles and Emacs-style line editing automatically fall back to a limited feature set.

Commands and special keys
The following special commands are supported by all REPL instances:

.break: When in the process of inputting a multi-line expression, enter the .break command (or press Ctrl+C) to abort further input or processing of that expression.

.clear: Resets the REPL context to an empty object and clears any multi-line expression being input.

.exit: Close the I/O stream, causing the REPL to exit.
.help: Show this list of special commands.

.save: Save the current REPL session to a file: > .save ./file/to/save.js

.load: Load a file into the current REPL session. > .load ./file/to/load.js
.editor: Enter editor mode (Ctrl+D to finish, Ctrl+C to cancel).

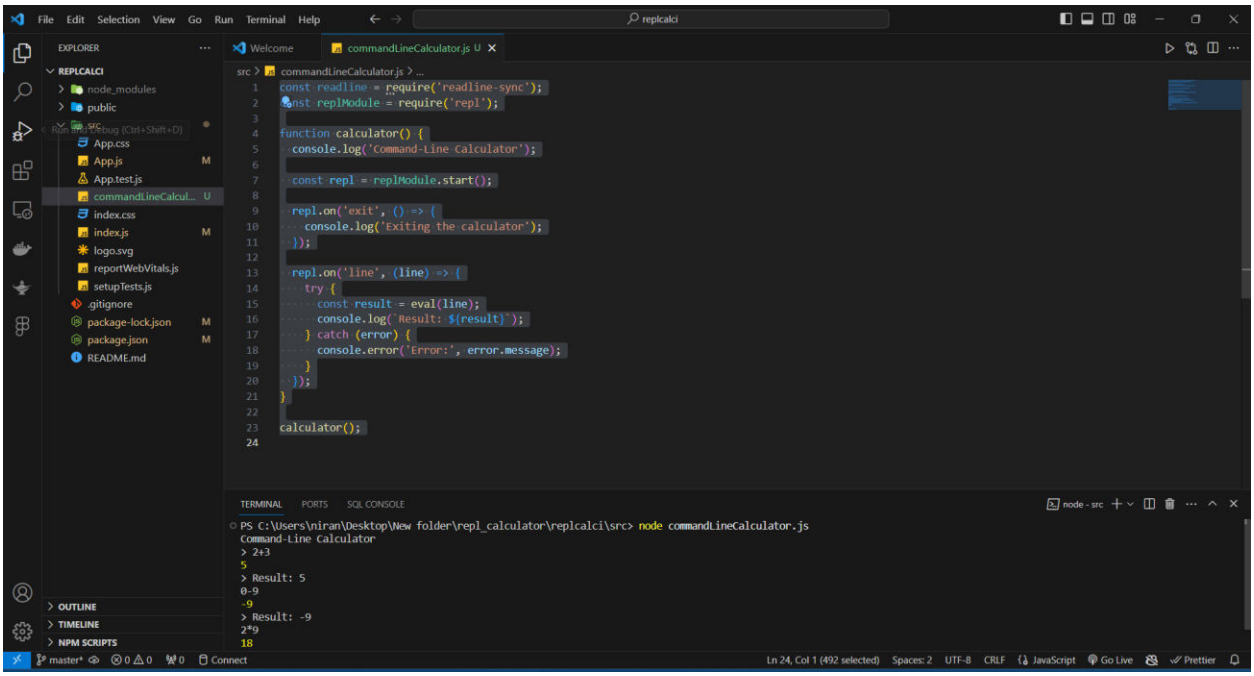
By default, all instances of repl.REPLServer use an evaluation function that evaluates JavaScript expressions and provides access to Node.js built-in modules. This default behavior can be overridden by passing in an alternative evaluation function when the repl.REPLServer instance is created.

Output:-

1)Code for commandLineCalculator.js

```
const readline = require('readline-sync');
const replModule = require('repl');

function calculator() {
  console.log('Command-Line Calculator');

  const repl = replModule.start();

  repl.on('exit', () => {
    console.log('Exiting the calculator');
  });

  repl.on('line', (line) => {
    try {
      const result = eval(line);
      console.log(`Result: ${result}`);
    } catch (error) {
      console.error('Error:', error.message);
    }
  });
}

calculator();
```

2)Output screenshots



References :-

1)https://nodejs.org/api/repl.html#design-and-features
2)https://www.tutorialspoint.com/nodejs/nodejs_repl_terminal.htm

Conclusion :-

Learnt about REPL in node.js , explored more about its command and special keys and created a calculator in commandline using REPL

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-03/10/2023

Assignment No-9

Aim:-

Write a program to implement
a. Create React refs
b. How to access Refs
c. Forward Refs
d. Callback Refs.

LO Mapped:- LO5

Theory:-

React is a popular JavaScript library for building user interfaces, and it provides various tools and patterns to efficiently manage and manipulate the Document Object Model (DOM) elements. One of these tools is React Refs, which allow developers to access and interact with DOM elements directly. With the introduction of React Hooks, working with refs has become more accessible and intuitive. In this guide, we'll explore how to create and use React refs effectively using functional components and hooks.

a. Creating React Refs:

React Refs are objects that provide a way to access and interact with DOM elements directly. In functional components, you can create refs using the `useRef` hook.

```
const myRef = useRef(null);
```

The `useRef` hook initializes the `myRef` variable with the `current` property set to `null`. You can attach this ref to a DOM element by assigning it as a `ref` attribute in the JSX.

b. How to Access Refs:

Once you've created a ref, you can access and manipulate the corresponding DOM element. To access the DOM element, you can use the `.current` property of the ref. For example:

```
const element = myRef.current;
if (element) {
```

```
// Access and manipulate the DOM element
}
```

It's important to check if the element exists before accessing and manipulating it because the `current` property may initially be `null`.

c. Forward Refs:

React allows you to forward refs from a parent component to a child component. This is particularly useful when you want to access a child's DOM element from a parent component. To create a forward ref, use the `React.forwardRef` function:

```
const ChildComponent = React.forwardRef((props, ref) => {
// JSX for child component
return <input ref={ref} />;
});
```

In this example, the `ChildComponent` accepts a `ref` parameter and attaches it to the `input` element. You can then use this forwarded ref in a parent component:

```
function ParentComponent() {
const childRef = useRef(null);
// Attach ref to the child component
return <ChildComponent ref={childRef} />;
}
```

Now, `childRef.current` refers to the `input` element inside `ChildComponent`.

d. Callback Refs:

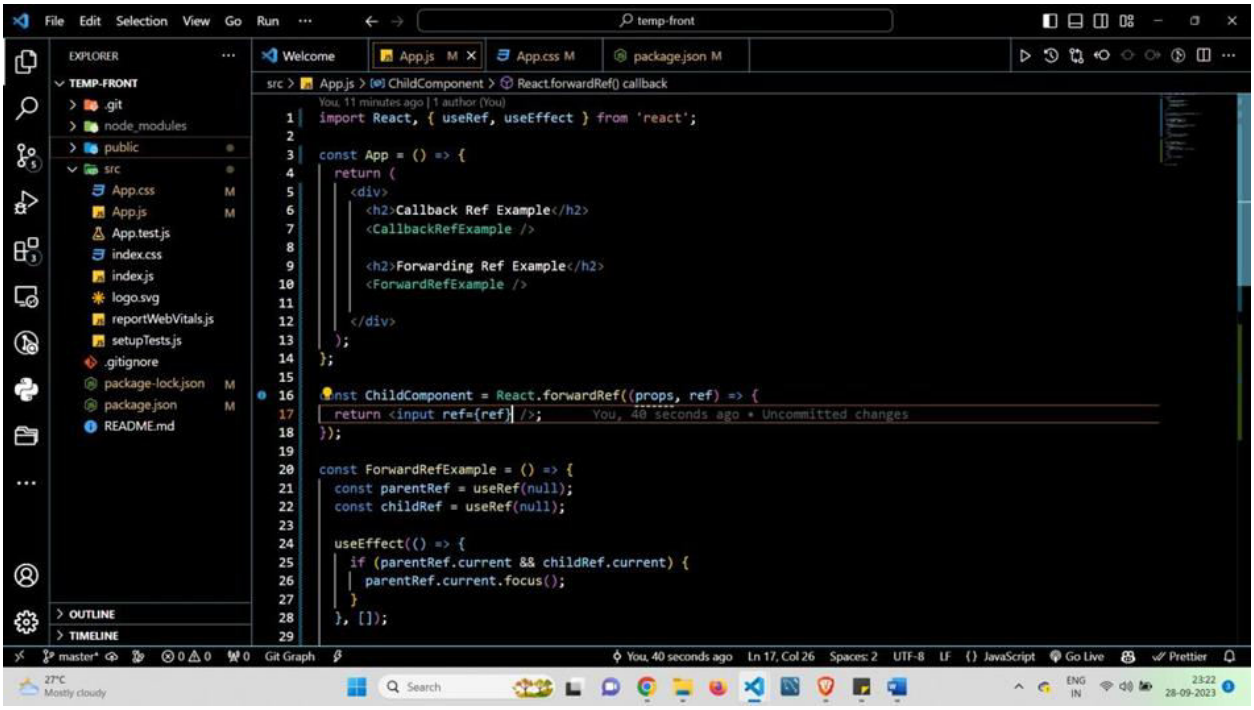
Callback refs are another way to work with refs in React. Instead of using the `useRef` hook, you define a function that receives the DOM element as an argument and stores it as a variable. Here's how to create and use a callback ref:

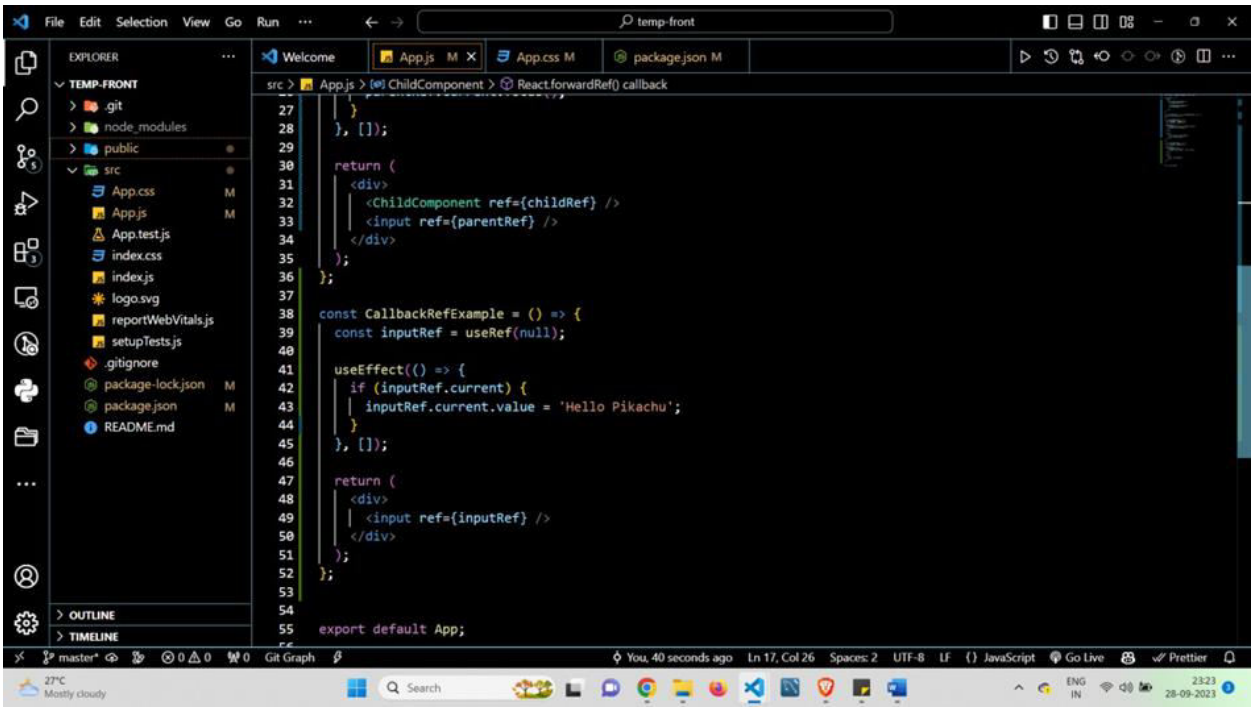
```
let myRef = null;
const setMyRef = (element) => {
myRef = element;
};
```

In this example, `setMyRef` is a callback function that receives the DOM element as an argument and assigns it to the `myRef` variable. You can then use `myRef` to access and manipulate the DOM element. Callback refs are typically defined inside the functional component.

React Hooks and functional refs provide a flexible and powerful way to work with DOM elements in your React applications. Whether you're creating refs, accessing DOM elements, forwarding refs to child components, or using callback refs, these techniques empower you to build dynamic and interactive user interfaces with ease.

Code :-




File Edit Selection View Go Run
temp-front
EXPLORER
TEMP-FRONT
.git
node_modules
public
src
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md
OUTLINE
TIMELINE
Welcome
App.js M
App.css M
package.json M
src > App.js > ChildComponent > React.forwardRef() callback
}, []);
return (

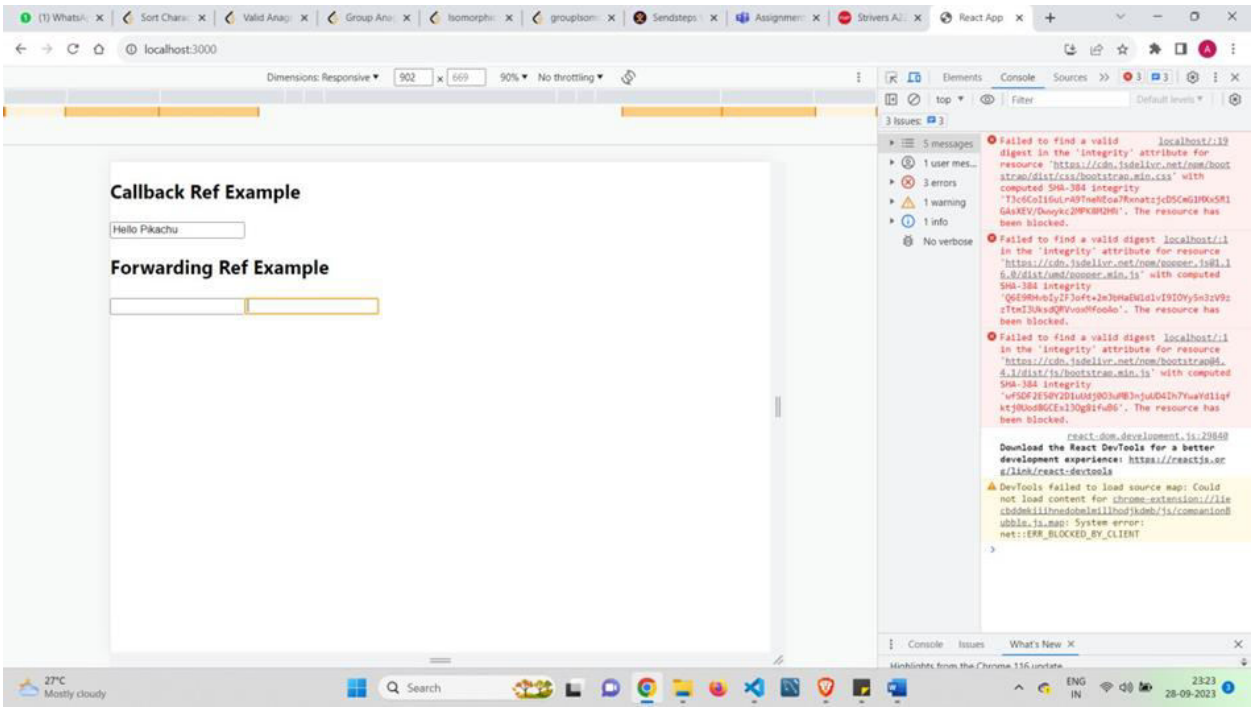
<ChildComponent ref={childRef} />
<input ref={parentRef} />

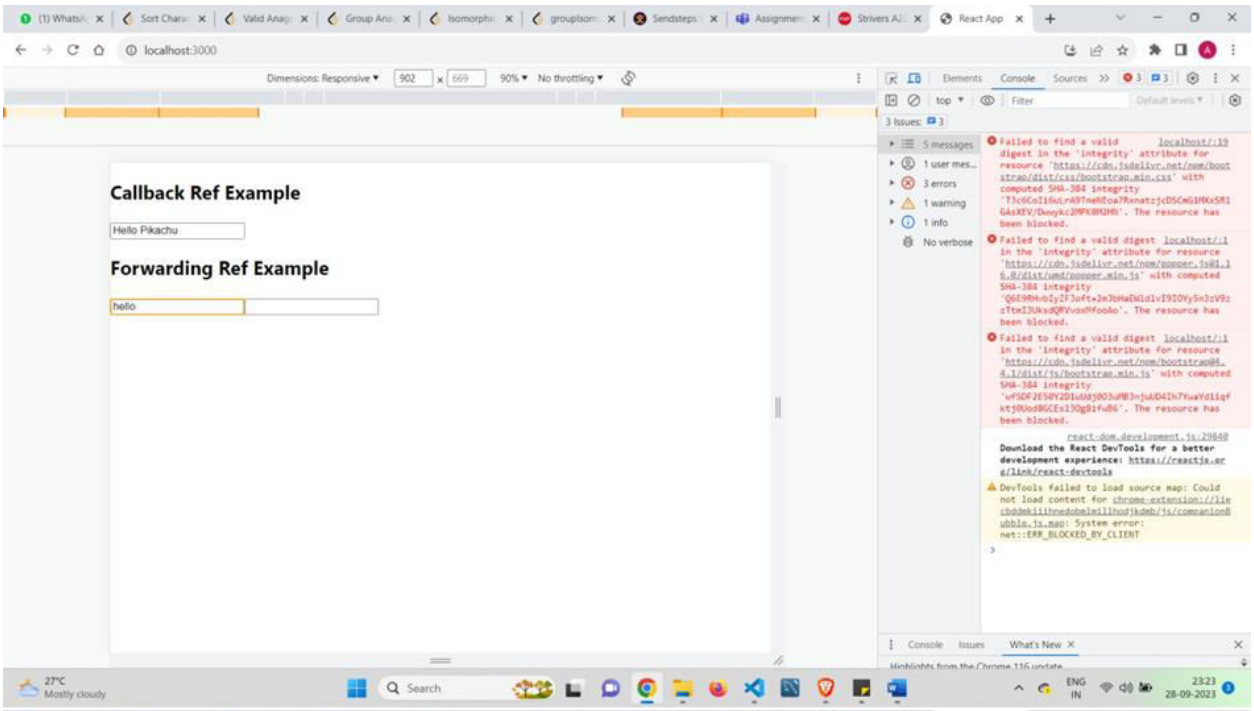
);
};
const CallbackRefExample = () => {
const inputRef = useRef(null);
useEffect(() => {
if (inputRef.current) {
inputRef.current.value = 'Hello Pikachu';
}
}, []);
return (

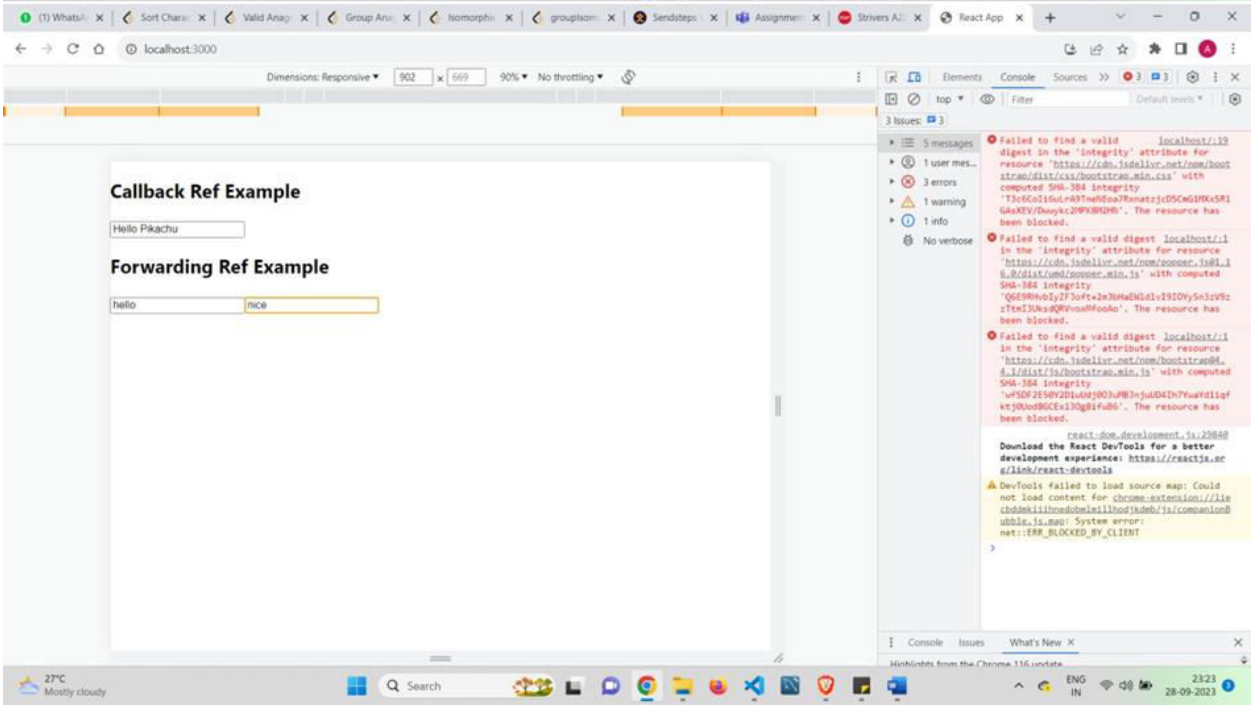
<input ref={inputRef} />

);
};
export default App;
You, 40 seconds ago Ln 17, Col 26 Spaces: 2 UTF-8 LF {} JavaScript Go Live Prettier
master* Git Graph
27°C
Mostly cloudy
Search
ENG
IN
23:23
28-09-2023

Output :-


React App
localhost:3000
Callback Ref Example
Hello Pikachu
Forwarding Ref Example
Elements Console Sources
Failed to find a valid digest in the 'integrity' attribute for resource
react-dom.development.js:29840
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
DevTools failed to load source map: Could not load content for chrome-extension://
net::ERR_BLOCKED_BY_CLIENT
Console Issues What's New
27°C
Mostly cloudy
Search
ENG
IN
23:23
28-09-2023





References :-

1)https://legacy.reactjs.org/docs/refs-and-the-dom.html
2)https://www.geeksforgeeks.org/reactjs-refs/

Conclusion :-

Learnt about react refs about how they can be created , how they can be accessed , about callback refs and about forward refs , implemented all these concepts practically and gained more knowledge about it .

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-26/09/2023

Assignment No-10

Aim:-

Write a program in Node JS to
a. Create a file
b. Read the data from file
c. Write the data to a file
d. Rename a file
e. Append data to a file
f. Delete a file

LO Mapped:- LO6

Theory:-

Node.js is a powerful runtime environment for executing JavaScript on the server side. It provides built-in modules, including the 'fs' (File System) module, that allow developers to perform various file operations. Here, we will explore how to perform common file operations in Node.js:

a. Creating a File
To create a file in Node.js, you can use the 'fs' (File System) module. You typically use the `fs.writeFile()` method, specifying the file name and content. This method will create the file if it doesn't exist or overwrite its content if it does.

b. Reading Data from a File
Reading data from a file involves using the 'fs' module's `fs.readFile()` method. You provide the file name and an encoding (e.g., 'utf8' for text files) to read the file's content. The method then asynchronously reads the content and provides it as a callback argument.

c. Writing Data to a File
To write data to an existing file, you can use the 'fs' module's `fs.writeFile()` method. Similar to creating a file, you specify the file name and the content you want to write. This method

overwrites the file's existing content with the new data.

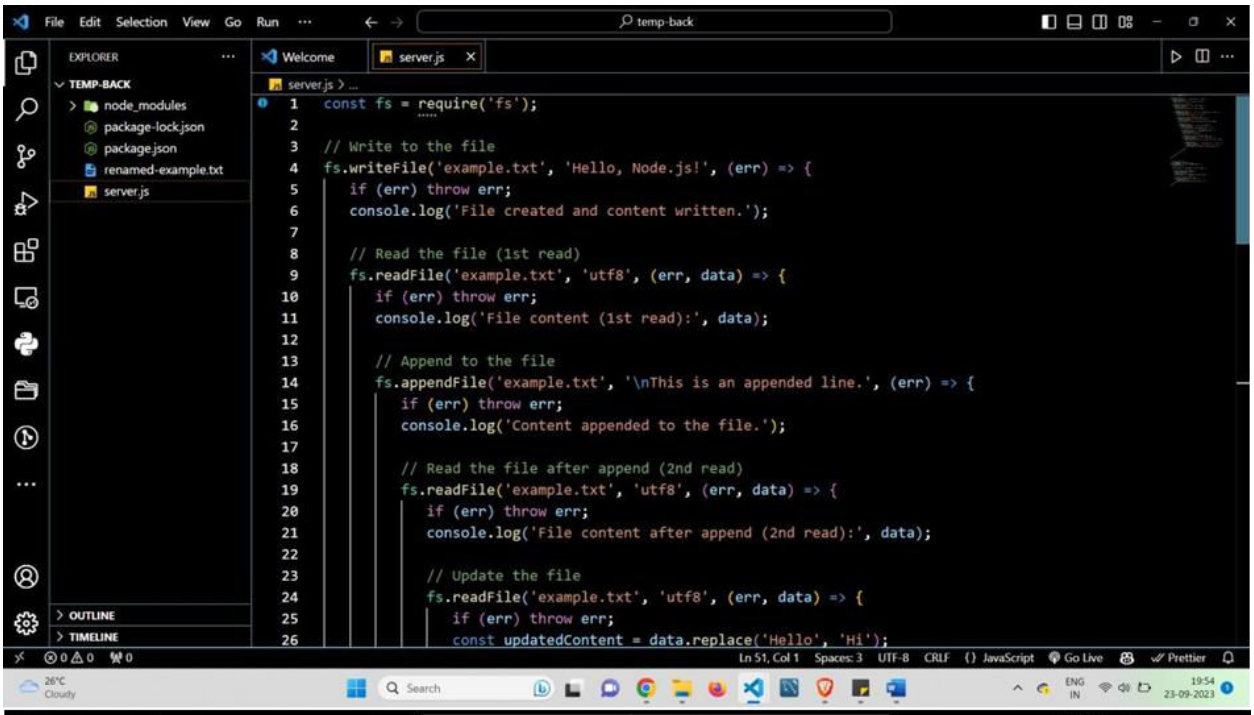
d. Renaming a File
To rename a file in Node.js, you use the 'fs' module's `fs.rename()` method. This method accepts two arguments: the current file name and the new file name. It effectively renames the file by changing its name in the file system.

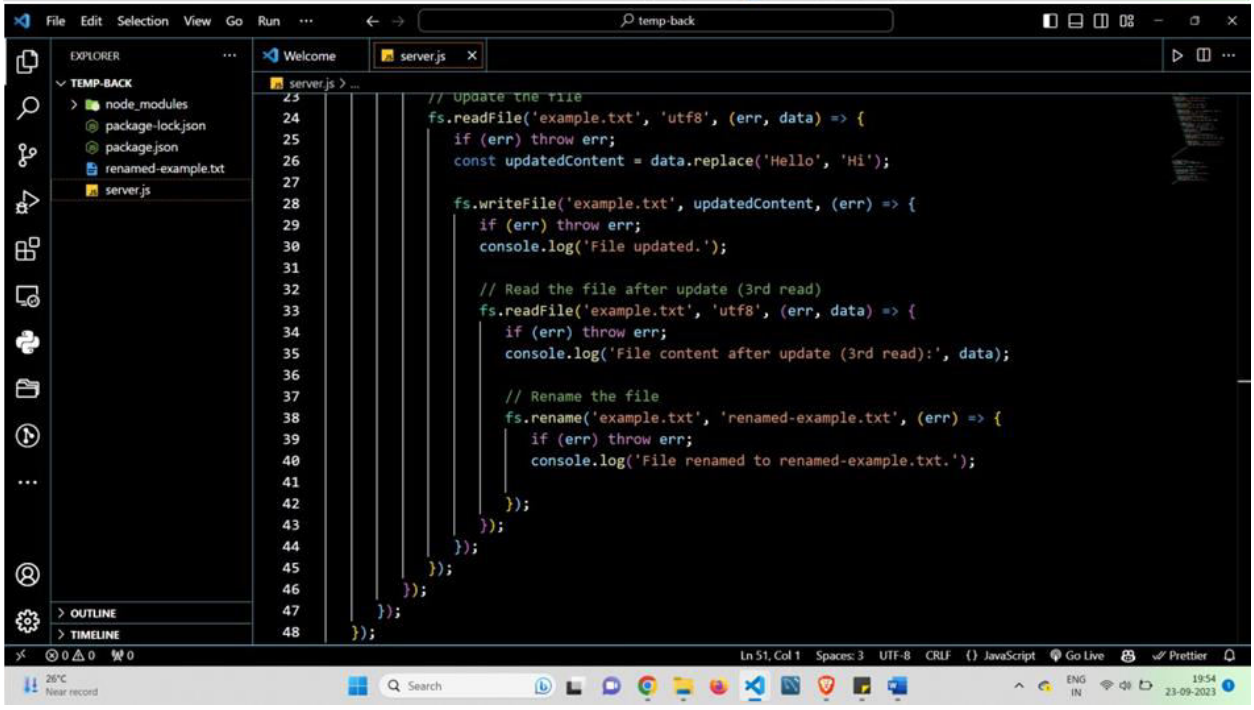
e. Appending Data to a File
Appending data to a file without overwriting its existing content can be done using the 'fs' module's `fs.appendFile()` method. You provide the file name and the data to append. This method appends the data to the end of the file, preserving the existing content.

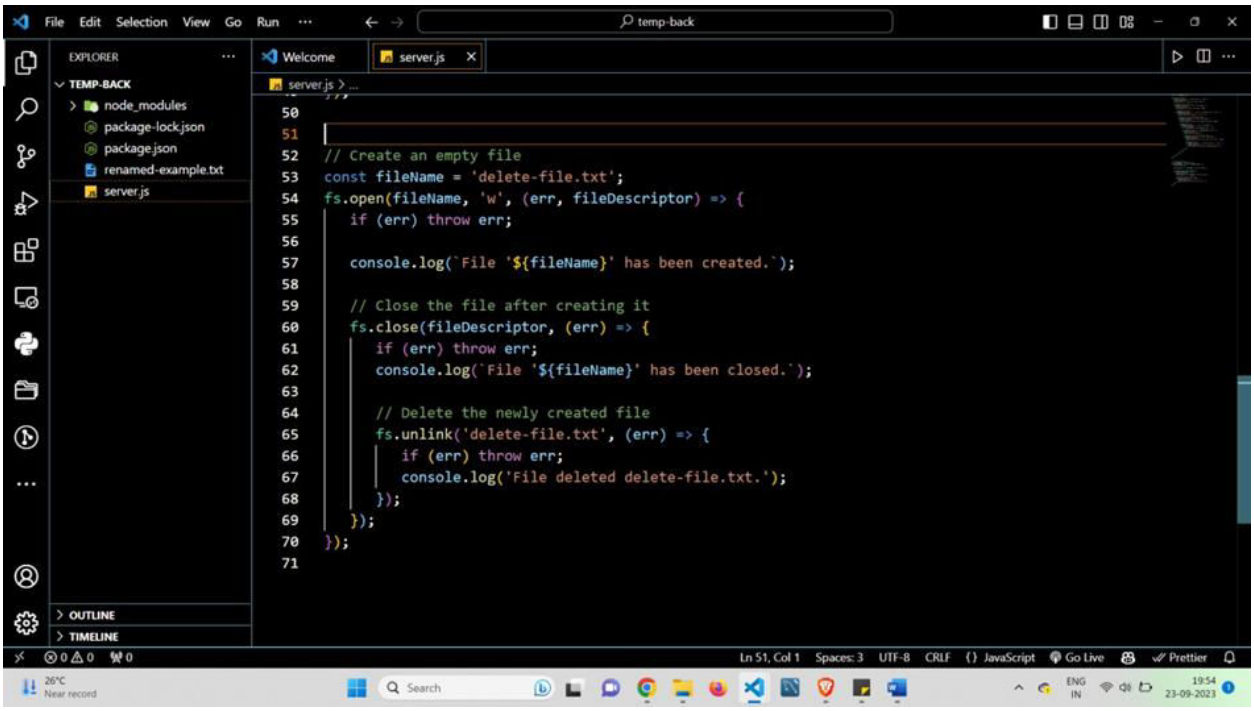
f. Deleting a File
To delete a file, you utilize the 'fs' module's `fs.unlink()` method. You specify the file name as the argument, and this method removes the file from the file system.

In Node.js, these file operations are typically performed asynchronously, allowing your application to continue executing other tasks while file operations are in progress. Proper error handling is crucial to catch and manage any errors that may occur during these operations, ensuring the robustness and reliability of your Node.js applications.

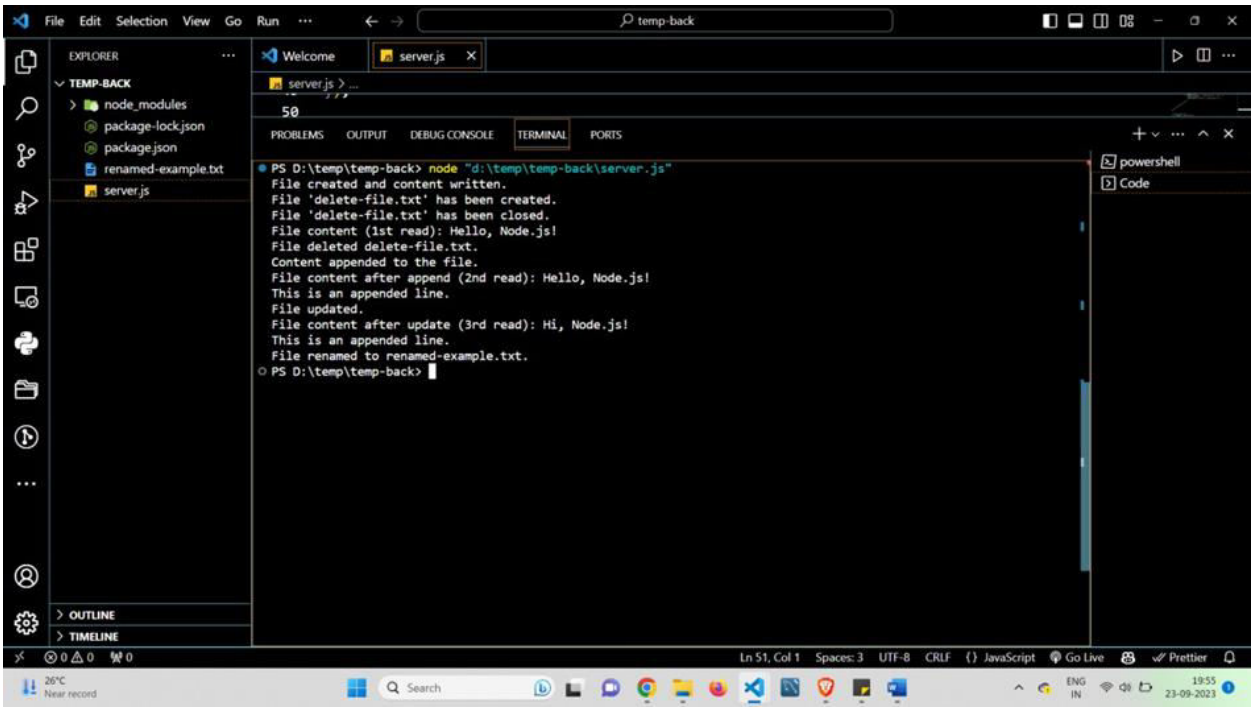
Code :-


File Edit Selection View Go Run
temp-back
EXPLORER
TEMP-BACK
node_modules
package-lock.json
package.json
renamed-example.txt
server.js
Welcome
server.js
server.js > ...
1 const fs = require('fs');
2
3 // Write to the file
4 fs.writeFile('example.txt', 'Hello, Node.js!', (err) => {
5 if (err) throw err;
6 console.log('File created and content written.');
7
8 // Read the file (1st read)
9 fs.readFile('example.txt', 'utf8', (err, data) => {
10 if (err) throw err;
11 console.log('File content (1st read):', data);
12
13 // Append to the file
14 fs.appendFile('example.txt', '\nThis is an appended line.', (err) => {
15 if (err) throw err;
16 console.log('Content appended to the file.');
17
18 // Read the file after append (2nd read)
19 fs.readFile('example.txt', 'utf8', (err, data) => {
20 if (err) throw err;
21 console.log('File content after append (2nd read):', data);
22
23 // Update the file
24 fs.readFile('example.txt', 'utf8', (err, data) => {
25 if (err) throw err;
26 const updatedContent = data.replace('Hello', 'Hi');
OUTLINE
TIMELINE
Ln 51, Col 1 Spaces: 3 UTF-8 CRLF {} JavaScript Go Live Prettier
26°C Cloudy
Search
ENG IN
19:54 23-09-2023


File Edit Selection View Go Run
temp-back
EXPLORER
TEMP-BACK
node_modules
package-lock.json
package.json
renamed-example.txt
server.js
Welcome
server.js
server.js > ...
// Update the file
fs.readFile('example.txt', 'utf8', (err, data) => {
if (err) throw err;
const updatedContent = data.replace('Hello', 'Hi');
fs.writeFile('example.txt', updatedContent, (err) => {
if (err) throw err;
console.log('File updated.');
// Read the file after update (3rd read)
fs.readFile('example.txt', 'utf8', (err, data) => {
if (err) throw err;
console.log('File content after update (3rd read):', data);
// Rename the file
fs.rename('example.txt', 'renamed-example.txt', (err) => {
if (err) throw err;
console.log('File renamed to renamed-example.txt.');
});
});
});
});
});
});
OUTLINE
TIMELINE
Ln 51, Col 1 Spaces: 3 UTF-8 CRLF {} JavaScript Go Live Prettier
26°C Near record
Search
ENG IN
19:54 23-09-2023


File Edit Selection View Go Run
temp-back
EXPLORER
TEMP-BACK
node_modules
package-lock.json
package.json
renamed-example.txt
server.js
Welcome
server.js
server.js > ...
50
51
52 // Create an empty file
53 const fileName = 'delete-file.txt';
54 fs.open(fileName, 'w', (err, fileDescriptor) => {
55 if (err) throw err;
56
57 console.log(`File '${fileName}' has been created.`);
58
59 // Close the file after creating it
60 fs.close(fileDescriptor, (err) => {
61 if (err) throw err;
62 console.log(`File '${fileName}' has been closed.`);
63
64 // Delete the newly created file
65 fs.unlink('delete-file.txt', (err) => {
66 if (err) throw err;
67 console.log('File deleted delete-file.txt.');
68 });
69 });
70 });
71
OUTLINE
TIMELINE
Ln 51, Col 1 Spaces: 3 UTF-8 CRLF {} JavaScript Go Live Prettier
26°C Near record
Search
ENG IN
19:54 23-09-2023

Output :-


File Edit Selection View Go Run
temp-back
EXPLORER
TEMP-BACK
node_modules
package-lock.json
package.json
renamed-example.txt
server.js
Welcome
server.js
server.js > ...
50
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Code
PS D:\temp\temp-back> node "d:\temp\temp-back\server.js"
File created and content written.
File 'delete-file.txt' has been created.
File 'delete-file.txt' has been closed.
File content (1st read): Hello, Node.js!
File deleted delete-file.txt.
Content appended to the file.
File content after append (2nd read): Hello, Node.js!
This is an appended line.
File updated.
File content after update (3rd read): Hi, Node.js!
This is an appended line.
File renamed to renamed-example.txt.
PS D:\temp\temp-back>
OUTLINE
TIMELINE
Ln 51, Col 1 Spaces: 3 UTF-8 CRLF {} JavaScript Go Live Prettier
26°C Near record
Search
ENG IN
19:55 23-09-2023

References :-

1)https://www.mongodb.com/docs/drivers/node/current/fundamentals/crud/
2)https://geeksforgeeks.org/node-js-crud-operations-using-mongoose-and-mongodb-atlas/

Conclusion :-

Learnt about basic CRUD operations in node.js and also learnt about different ways in which these can be implemented , practically demonstrated CRUD operations using node.js

Roll No :- 2105051
Name:- Niranjan Joshi
Date:-13/10/2023

Assignment No-11

Aim:-

Create a web application that performs CRUD operations (database connectivity).

LO Mapped:- LO1,LO2,LO3,LO4,LO5,LO6

Theory: (overview of project) :-

The Student Database Management App is a robust system designed to facilitate the four fundamental operations in database management: Create, Read, Update, and Delete (CRUD). The system comprises three main components, each serving a specific role in this process: the frontend developed using React and Bootstrap, the backend powered by Express, and the database managed through MongoDB. This architecture combines the strengths of different technologies to create a versatile and efficient student information management system.

Frontend (React and Bootstrap): React is a popular JavaScript library for building user interfaces. It allows developers to create dynamic and responsive web applications by breaking down the UI into reusable components. In this context, React is used to design the user interface for the Student Database Management App. Bootstrap, a widely used CSS framework, complements React by providing a plethora of pre-designed UI elements, ensuring a clean and user-friendly design. The combination of React and Bootstrap enables the development of an interactive and visually appealing frontend, making it easier for users to interact with the database.

Backend (Express): The backend, powered by Express, serves as the intermediary layer between the frontend and the database. Express is a minimal and flexible Node.js web application framework, designed to simplify the creation of robust and scalable APIs. In this app, Express handles HTTP requests from the frontend and communicates with the database to execute CRUD operations. It plays a crucial role in managing the logic that governs how data is accessed, modified, and deleted, ensuring the security and reliability of the application.

Database (MongoDB): MongoDB, a NoSQL database, is used as the backend data store for this application. MongoDB's document-oriented, schema-less structure is well-suited for storing student data, as it allows for easy scalability and adaptability to changing data requirements.

Students' information can be stored as JSON-like documents, making it simple to add or remove fields as needed. MongoDB also supports the indexing of data for efficient retrieval and querying, enhancing the app's ability to deliver student information quickly and accurately.

In this system, when a new student needs to be registered, the frontend provides an intuitive interface for inputting the student's details. Once submitted, the Express backend processes the request, validates the data, and then inserts it into the MongoDB database, creating a new student entry.

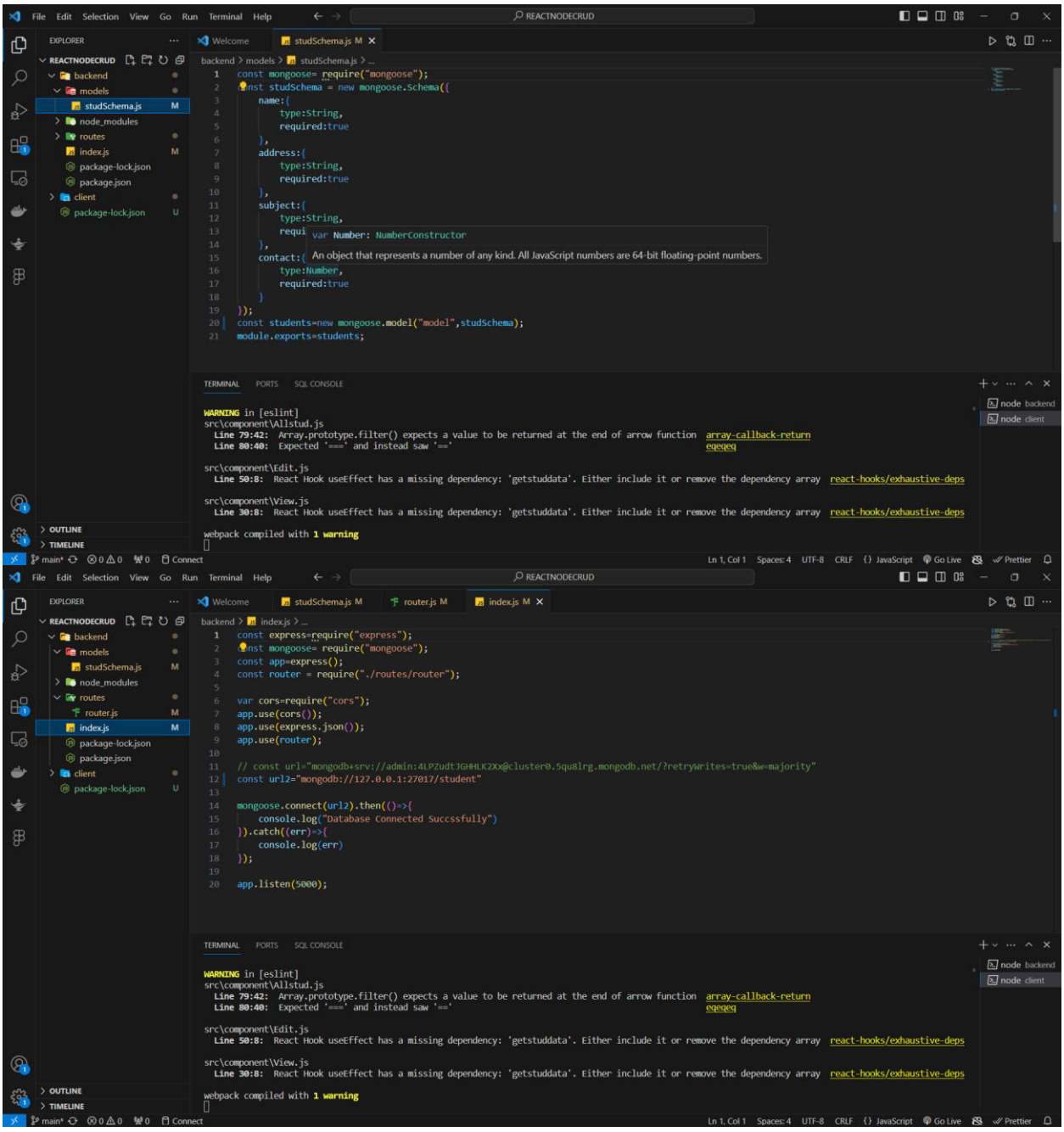
For updating information, the frontend offers a user-friendly form to modify the student's data. This change is then sent to the Express backend, which updates the corresponding record in the MongoDB database.

To view student information, the frontend communicates with the Express backend, which queries the MongoDB database, retrieves the requested data, and sends it back to the frontend for display.

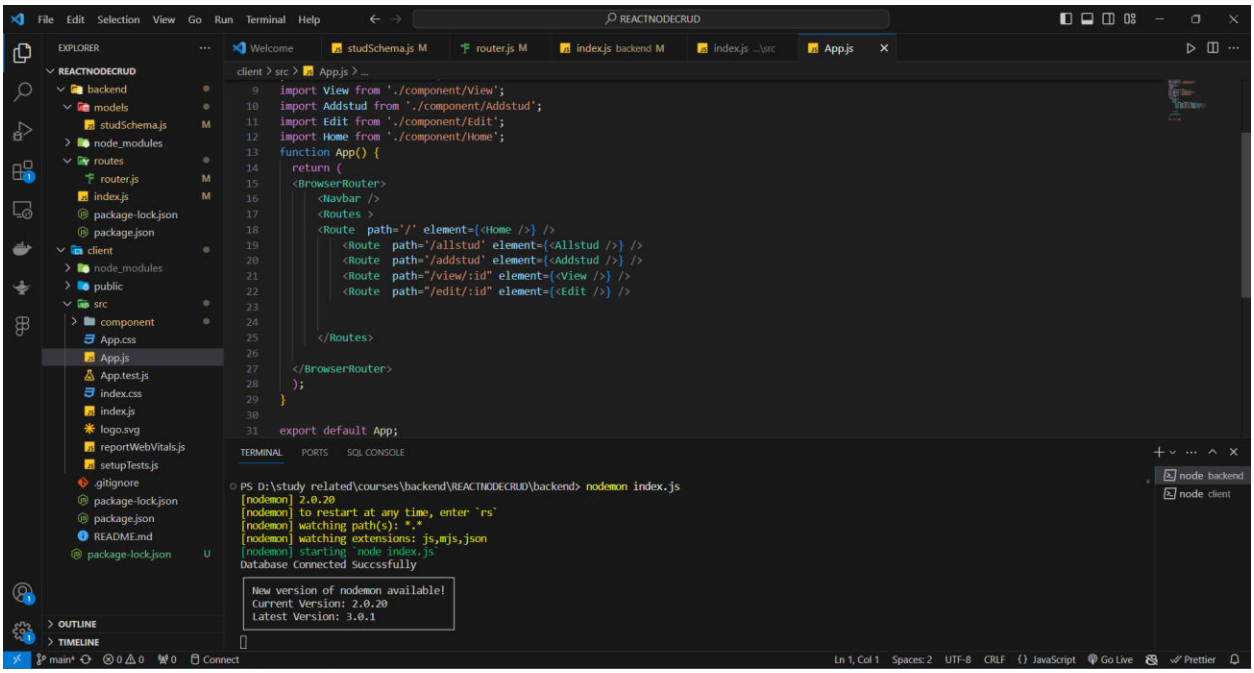
When a student's record needs to be removed, the frontend triggers a request to the backend, which locates and deletes the specified entry from the MongoDB database.

This architecture enables the app to handle the complete lifecycle of student data effectively. The combination of **React, Bootstrap, Express, and MongoDB** ensures a smooth and efficient experience for users, making student data management straightforward and intuitive.

Code :-

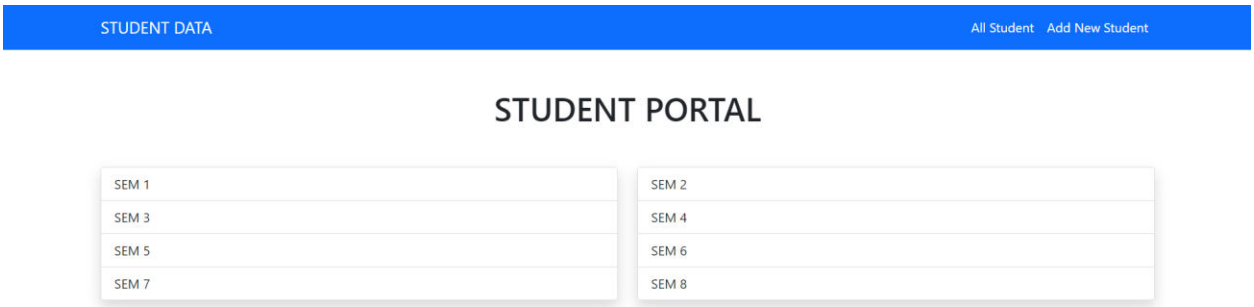

File Edit Selection View Go Run Terminal Help
REACTNODECRUD
EXPLORER
REACTNODECRUD
backend
models
studSchema.js M
node_modules
routes
index.js M
package-lock.json
package.json
client
package-lock.json U
Welcome
studSchema.js M
backend > models > studSchema.js > ...
const mongoose= require("mongoose");
const studSchema = new mongoose.Schema({
name:{
type:String,
required:true
},
address:{
type:String,
required:true
},
subject:{
type:String,
requi
var Number: NumberConstructor
An object that represents a number of any kind. All JavaScript numbers are 64-bit floating-point numbers.
},
contact:{
type:Number,
required:true
}
});
const students=new mongoose.model("model",studSchema);
module.exports=students;
TERMINAL PORTS SQL CONSOLE
node backend
node client
WARNING in [eslint]
src\component\Allstud.js
Line 79:42: Array.prototype.filter() expects a value to be returned at the end of arrow function array-callback-return
Line 80:40: Expected '===' and instead saw '==' eqeqeq
src\component\Edit.js
Line 50:8: React Hook useEffect has a missing dependency: 'getstuddata'. Either include it or remove the dependency array react-hooks/exhaustive-deps
src\component\View.js
Line 30:8: React Hook useEffect has a missing dependency: 'getstuddata'. Either include it or remove the dependency array react-hooks/exhaustive-deps
webpack compiled with 1 warning
OUTLINE
TIMELINE
main* 0 0 0 Connect
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript Go Live Prettier

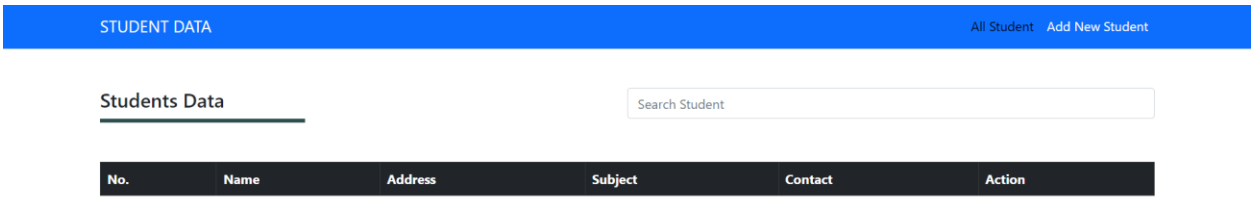
File Edit Selection View Go Run Terminal Help
REACTNODECRUD
EXPLORER
REACTNODECRUD
backend
models
studSchema.js M
node_modules
routes
router.js M
index.js M
package-lock.json
package.json
client
package-lock.json U
Welcome
studSchema.js M
router.js M
index.js M
backend > index.js > ...
const express=require("express");
const mongoose= require("mongoose");
const app=express();
const router = require("./routes/router");
var cors=require("cors");
app.use(cors());
app.use(express.json());
app.use(router);
// const url="mongodb+srv://admin:4LPZudtJGHHLK2Xx@cluster0.5qu8lrg.mongodb.net/?retryWrites=true&w=majority"
const url2="mongodb://127.0.0.1:27017/student"
mongoose.connect(url2).then(()=>{
console.log("Database Connected Succssfully")
}).catch((err)=>{
console.log(err)
});
app.listen(5000);
TERMINAL PORTS SQL CONSOLE
node backend
node client
WARNING in [eslint]
src\component\Allstud.js
Line 79:42: Array.prototype.filter() expects a value to be returned at the end of arrow function array-callback-return
Line 80:40: Expected '===' and instead saw '==' eqeqeq
src\component\Edit.js
Line 50:8: React Hook useEffect has a missing dependency: 'getstuddata'. Either include it or remove the dependency array react-hooks/exhaustive-deps
src\component\View.js
Line 30:8: React Hook useEffect has a missing dependency: 'getstuddata'. Either include it or remove the dependency array react-hooks/exhaustive-deps
webpack compiled with 1 warning
OUTLINE
TIMELINE
main* 0 0 0 Connect
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript Go Live Prettier



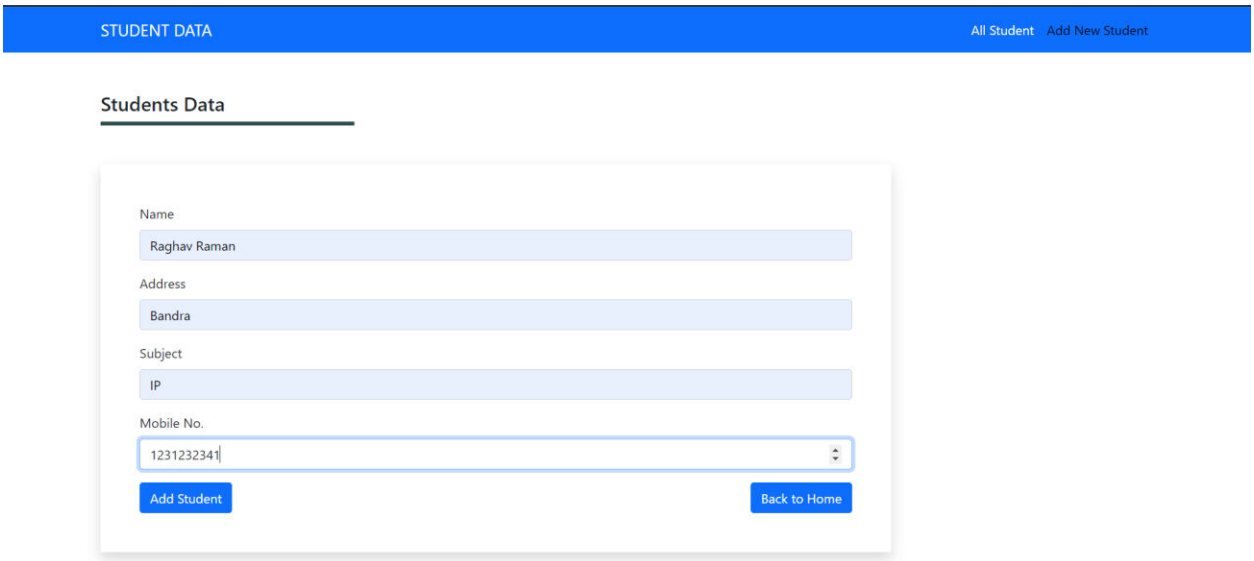
Output :-

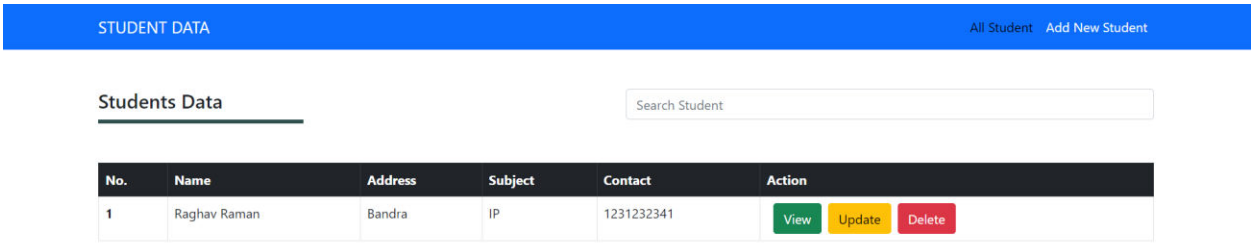
1)Frontend :-

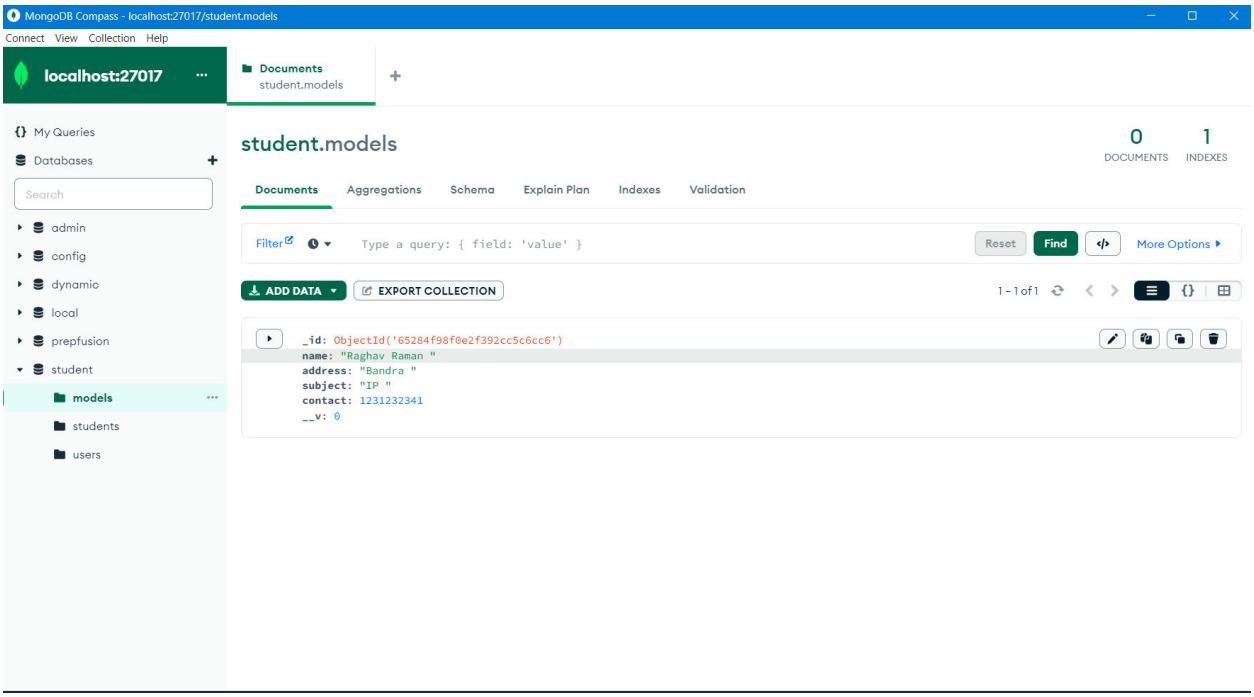




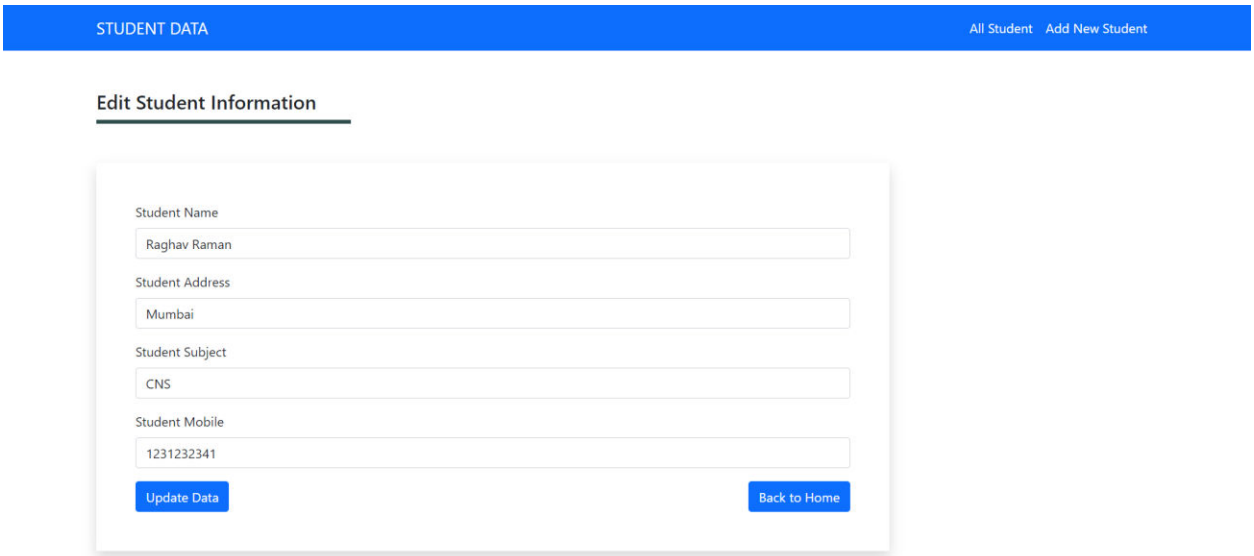
2)Create and Read operation :-

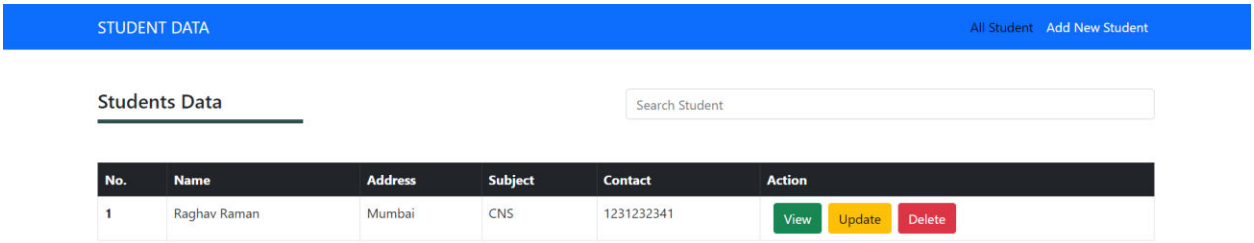

STUDENT DATA
All Student
Add New Student
Students Data
Name
Raghav Raman
Address
Bandra
Subject
IP
Mobile No.
1231232341
Add Student
Back to Home

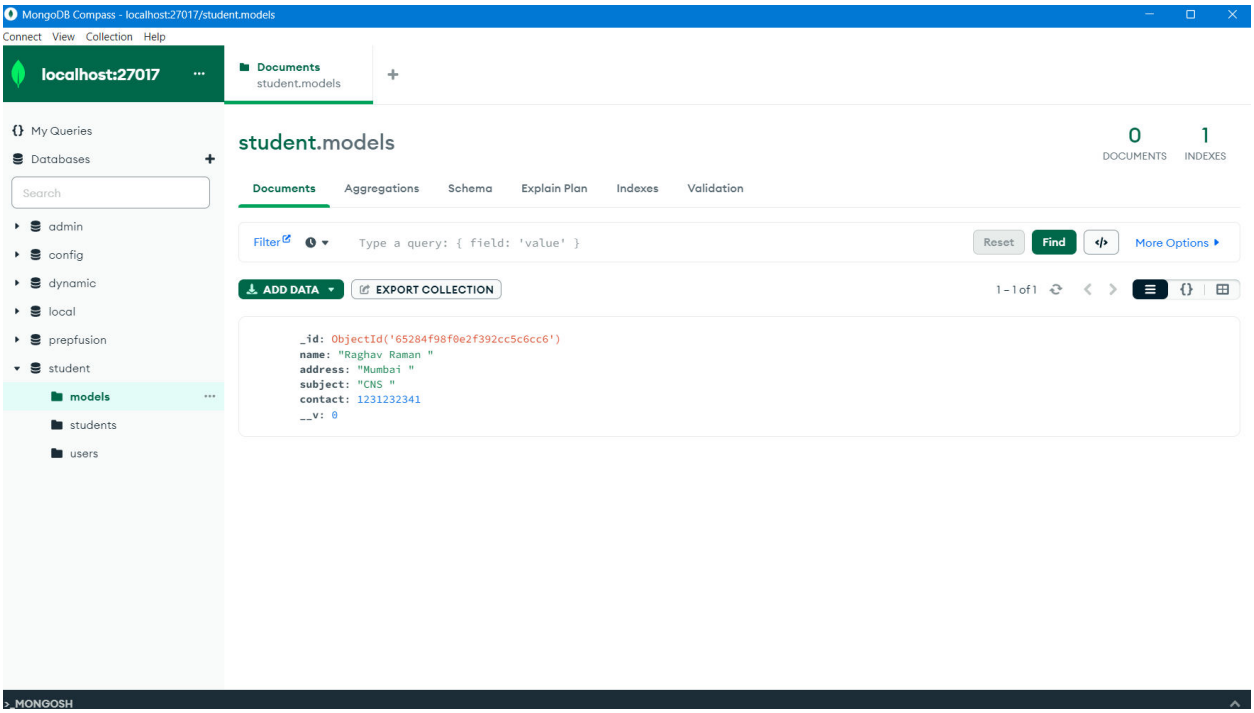

STUDENT DATA
All Student
Add New Student
Students Data
Search Student
No.
Name
Address
Subject
Contact
Action
1
Raghav Raman
Bandra
IP
1231232341
View
Update
Delete



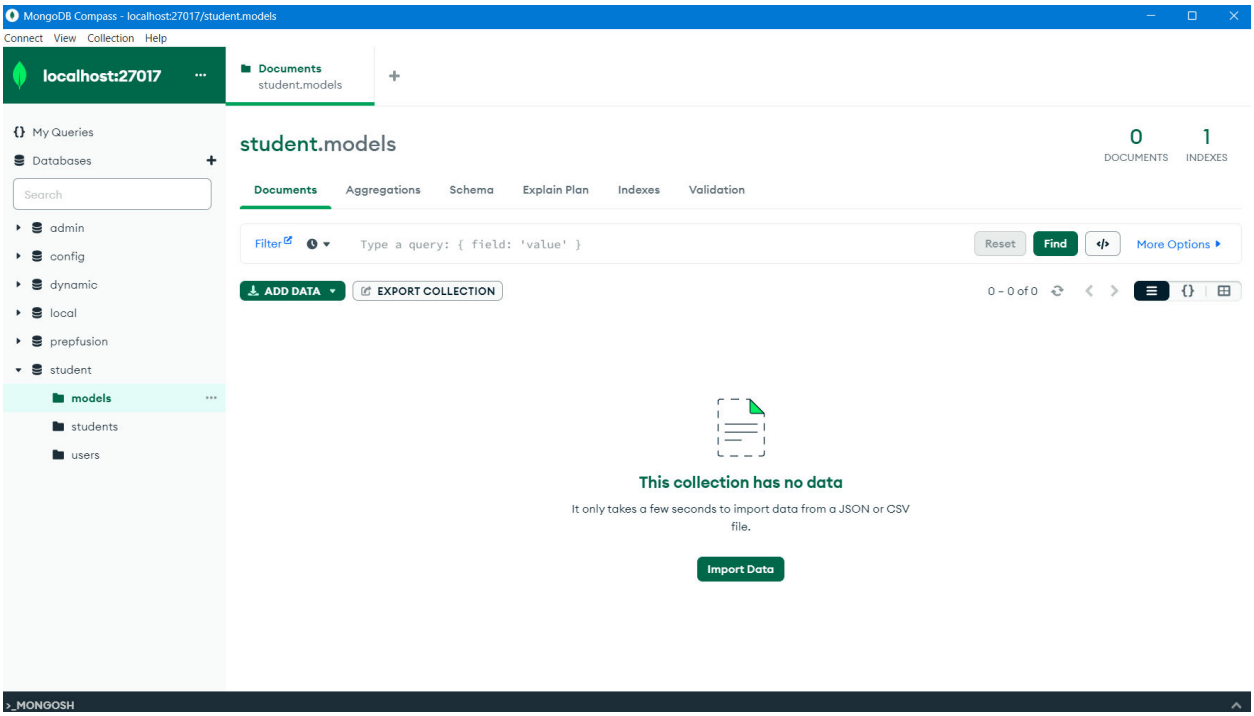
3)Update operation :-


STUDENT DATA
All Student
Add New Student
Edit Student Information
Student Name
Raghav Raman
Student Address
Mumbai
Student Subject
CNS
Student Mobile
1231232341
Update Data
Back to Home


STUDENT DATA
All Student
Add New Student
Students Data
Search Student
No.
Name
Address
Subject
Contact
Action
1
Raghav Raman
Mumbai
CNS
1231232341
View
Update
Delete



4)Delete operation :-



References :-

1https://www.mongodb.com/docs/manual/crud/
2https://www.geeksforgeeks.org/mongodb-crud-operations/

Conclusion :-

Learnt to make a fullstack application using react in the frontend, backend with express.js and node.js and database with mongodb also learned to integrate the frontend with backend and connect to database.

Subject : Internet Programming **Subject Code : ITL501**
Syllabus:R19

Q1) Compare XML and JSON

Features	XML	JSON
Definition	Extensible Markup Language	JavaScript Object Notation
Readability	More Verbose	Compact and easier to read
Data Types	Support Various data types	Support basic data types
Comments	Support Comments	Does not support comment
Extensibility	Extensible with namespace	Limited extensibility
NameSpace Support	Support namespace	No support for namespace
Security	Less Secured	More Secured than JSON
Data Representation	Represent data using tag structure	Represent data using objects

Q2) Explain Different type of arrow function

Javascript functions can be written more succinctly using the arrow function (()=>). The ES6 version introduced arrow functions. They improve the structure and readability of our code.

A function without a name or an identity is called an anonymous function, or an arrow function. Without the function keyword, arrow functions can be declared and do not return any value. They go by the name Lambda Functions as well.

Arrow Function without Parameters

```
const world = () => {
 console.log( "Hello World" );
}
world();
```
Output
Hello World

Arrow Function with Parameters
```
Const sum = ( x, y, z ) => {
 console.log( x + y + z )
}
sum ( 20, 30, 40 );
```
Output
90

Arrow Function with Default Parameters
```
Const add = ( x, y=20 ) => {
 console.log( x + " " + y );
}
add ( 10 );
```
Output
10 2

Q3)What is DNS? Explain working of DNS.

Internet domain names are identified and converted into Internet Protocol (IP) addresses using the domain name system (DNS), a naming database. The IP address that a computer uses to find a website is mapped to that website's name via the domain name system.

Types of DNS –
Authoritative DNS Servers:
These are the definitive sources of DNS data for particular domains. They preserve the linkage between IP addresses and domain names. In order to provide redundancy, there may be primary authoritative servers (master) and secondary authoritative servers (slave).

Recursive DNS –
The majority of the time, ISPs, businesses, or public DNS services like Google DNS and OpenDNS run these servers. They take in DNS requests from clients (such

your computer or router), handle the requests, and then recursively look up the IP address needed to answer the request. To enhance future answers, they store the results.

The working of DNS involves several steps, which can be summarized as follows:
DNS Resolution Request:
When a user enters a URL (Uniform Resource Locator) or clicks a link, their device initiates a DNS resolution request to translate the domain name into an IP Address.

1. Local Cache Check:
The device's DNS resolver (usually provided by the ISP or a public DNS service) first checks its local cache to see if the IP address corresponding to the domain name is already stored. If found, the resolver returns the IP address directly to the user's device. This step improves efficiency and reduces DNS lookup times.

2. Recursive Query:
If the IP address is not found in the local cache, the resolver starts a recursive query process. It begins by contacting one of the root DNS Servers.

3. Root DNS Servers:
The root DNS servers are a small set of highly reliable servers that provide referrals to Top-Level Domain (TLD) DNS servers. They do not know the IP addresses of specific domain names but guide the resolver to the appropriate authoritative DNS servers.

4. TLD DNS Servers:
The resolver contacts the TLD DNS server associated with the domain's extension (e.g., .com, .org). The TLD server provides information about the authoritative DNS server responsible for the next step.

5. Authoritative DNS Server:
The authoritative DNS server holds the specific DNS records for the domain. The resolver contacts this server to obtain the final IP address associated with the domain name.

6. Caching and Response:
The authoritative DNS server responds with the IP address, and the

resolver stores this information in its local cache. It then returns the IP address to the user's device.

7. User's Device:
With the IP address obtained, the user's device can now establish a connection to the web server hosting the requested content. This connection allows the browser to retrieve the web page and display it to the user.

Q4) Explain Promises in ES6.

Promises are a feature introduced in ECMAScript 6 (ES6) to handle asynchronous operations and improve the readability and manageability of asynchronous code. They provide a more structured way to work with asynchronous operations compared to traditional callback-based approaches. Promises simplify error handling and chaining multiple asynchronous operations together.

A promise represents the eventual completion (or failure) of an asynchronous operation, and it has three states:

1. Pending: The initial state, representing that the asynchronous operation is still ongoing and the result is not available yet.
2. Fulfilled: The state when the asynchronous operation is successfully completed, and the result is available.
3. Rejected: The state when the asynchronous operation encounters an error or fails to complete.

Here's how promises work in ES6:

1. Creating a Promise:
To create a promise, you use the `Promise` constructor. The constructor takes a function with two parameters: `resolve` and `reject`. Inside this function, you perform the asynchronous operation, and when it's done, you call either `resolve` to fulfill the promise or `reject` to reject it.

```
const myPromise = new Promise((resolve, reject) => {
// Asynchronous operation
// If successful, call resolve(result)
// If error, call reject(error)
});
```

2. Consuming a Promise:

Once you have a promise, you can use the `then()` method to define what should happen when the promise is fulfilled. You provide a callback function to `then()`, which will be executed when the promise is resolved successfully.

```
myPromise.then((result) => {
 // Handle the successful result
}).catch((error) => {
 // Handle the error
});
```

3. . Chaining Promises:

Promises can be chained together using the `then()` method, allowing you to perform multiple asynchronous operations sequentially.

```
fetch('https://api.example.com/data')
.then(response => response.json())
.then(data => {
 // Handle the fetched data
})
.catch(error => {
 // Handle errors in any of the steps
});
```

4. Promise.all() and Promise.race():

Promise.all()` takes an array of promises and returns a new promise that is fulfilled with an array of resolved values from all the input promises when they are all resolved. `Promise.race()` returns a new promise that is settled with the value of the first promise in the array to be settled (fulfilled or rejected).

```
const promises = [promise1, promise2, promise3];
Promise.all(promises)
 .then(results => {
 // Handle results when all promises are resolved
 })
 .catch(error => {
 // Handle errors if any promise rejects
 });
```

Asynchronous code is easier to manage and maintain when it is worked with in a clearer, more organized manner using promises. They serve as the basis for several additional asynchronous patterns and JavaScript APIs, such as "async/await.

Subject Teacher : Sanober Shaikh

Signature with date

Subject : Internet Programming **Subject Code : ITL501**
Syllabus:R19
Written Assignment 2

Q1) What are Refs? When to use Refs and when not to use refs.

Refs in React are a way to access the **underlying DOM element** or React component instance of a component. They are used in cases where you need to perform operations on a component, such as focusing an input element, starting an animation, or interacting with a third-party library.

When to use Refs :

Refs are used when there is need to imperatively modify a component outside of the typical React dataflow. For example, a person might use a ref to focus an input element when the component mounts, or to start an animation when a button is clicked.

Examples of when to use refs :

1) When you need to access the DOM element or React component instance of a child component.
2) When you need to interact with a third-party library that relies on DOM elements or React component instances.
3) When you need to perform imperative operations on a component, such as focusing an input element, starting an animation, or managing focus.

When not to use Refs :

Avoid using refs unless you absolutely need them. This is because refs can make code more complex and difficult to reason about.

Some examples of when not to use refs:

1) To update the state of a component. Instead, use React's state management features.
2) To communicate between components. Instead, use props and state.
3) To implement side effects. Instead, use React hooks like useEffect().

Q2) Write short note on
1)NPM
2)REPL

1) **NPM (Node Package Manager) is the default package manager for Node.js** and is written entirely in JavaScript. It is a registry of packages that can be used to add functionality to Node.js applications.

NPM packages are typically small, reusable modules of code that can be used to perform a variety of tasks, such as:

a) Communicating with APIs
b) Parsing data
c) Validating input
d) Sending emails
e) Rendering HTML
f) Creating and managing databases

NPM packages can be installed and managed using the NPM command-line utility. This utility provides a variety of commands for installing, uninstalling, updating, and managing packages.

Some of the benefits of using NPM:

a) NPM provides a large and diverse library of packages that can be used to add functionality to Node.js applications.
b) NPM makes it easy to install and manage packages.
c) NPM packages are typically well-maintained and up-to-date.
d) NPM is free and open-source.

2) **REPL stands for Read-Evaluate-Print-Loop.** It is a type of interactive programming environment that allows users to enter and execute code line by line. REPLs are typically used for rapid prototyping, debugging, and learning new programming languages.

REPLs are typically implemented as a command-line interface (CLI), but they can also be implemented as graphical user interfaces (GUIs). Some popular REPLs include:

a) Python interpreter
b) JavaScript console
c) Ruby irb
d) Node.js REPL
e) Go REPL
To use a REPL, simply start the REPL and start typing code. **The REPL will evaluate your code and print the output to the console.** You can continue to type and execute code until you are finished.

some of the benefits of using REPLs:
a) REPLs are a great way to quickly test and debug code.
b) REPLs can be used to learn new programming languages.
c) REPLs can be used to prototype new features and ideas.
d) REPLs can be used to interact with third-party libraries and tools.

Q3)Explain Routing in ExpressJS along with an example

Routing refers to how an application's endpoints (URIs) respond to client requests.
You define routing using methods of the Express app object that correspond to HTTP methods; for example, **app.get() to handle GET requests** and **app.post to handle POST requests**. You can also use **app.all()** to handle all HTTP methods and **app.use()** to specify **middleware as the callback function .**
These routing methods specify a callback function (sometimes called "handler functions") called when the application receives a request to the specified route (endpoint) and HTTP method. In other words, the application "listens" for requests that match the specified route(s) and method(s), and when it detects a match, it calls the specified callback function.

In fact, the routing methods can have more than one callback function as arguments. With multiple callback functions, it is important to provide next as an argument to the callback function and then call next() within the body of the function to hand off control to the next callback.

The following code is an example of a very basic route.

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

```
// respond with "hello world" when a GET request is made to the homepage
app.get('/', (req, res) => {
  res.send('hello world')
})
```

A route method is derived from one of the HTTP methods, and is attached to an instance of the express class.The following code is an example of routes that are defined for the GET and the POST methods to the root of the app.

```
// GET method route
app.get('/', (req, res) => {
  res.send('GET request to the homepage')
})

// POST method route
app.post('/', (req, res) => {
  res.send('POST request to the homepage')
})
```

Route parameters are named URL segments that are used to capture the values specified at their position in the URL. The captured values are populated in the req.params object, with the name of the route parameter specified in the path as their respective keys.

Route path: /users/:userId/books/:bookId
Request URL: http://localhost:3000/users/34/books/8989
req.params: { "userId": "34", "bookId": "8989" }
To define routes with route parameters, simply specify the route parameters in the path of the route as shown below.

```
app.get('/users/:userId/books/:bookId', (req, res) => {
  res.send(req.params)
})
```

express.Router class to create modular, mountable route handlers. A Router instance is a complete middleware and routing system; for this reason, it is often referred to as a **“mini-app”.**

The following example creates a router as a module, loads a middleware function in it, defines some routes, and mounts the router module on a path in the main app.

Create a router file named birds.js in the app directory, with the following content:

```
const express = require('express')
const router = express.Router()

// middleware that is specific to this router
router.use((req, res, next) => {
  console.log('Time: ', Date.now())
  next()
})
// define the home page route
router.get('/', (req, res) => {
  res.send('Birds home page')
})
// define the about route
router.get('/about', (req, res) => {
  res.send('About birds')
})

module.exports = router
```

Then, load the router module in the app:

```
const birds = require('./birds')

// ...

app.use('/birds', birds)
```

Subject Teacher : Sanober Shaikh

Signature with date