

Thadomal Shahani Engineering College
Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./Miss ANIMESH NARAYAN PARAB
of I.T Department, Semester V with
Roll No. 88 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 - 20 24

Sauobet
Teacher In-Charge
20/10/23

Head of the Department

Date 20/10/23

Principal

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1)	Develop a web application by using HTML Tags		27/7/23	
2)	using CSS and CSS3 enhance the web application developed in Assignments		3/8/23	
3)	Develop a web page using bootstrap framework grid, forms, Buttons, Navbar		3/8/23	Sampled 20/8/23
4)	a. write a program on conditional statement , loops b. write a programs on inheritance, Iterators and generators		10/8/23	
5)	a. write a Js program to study arrow functions , CSS and DOM manipulation b. write a Js program to implement promise (callback), fetch, Asynchronous Javascript		12/8/23	Sampled 20/8/23
6)	a. write a program to implement props & state b. write a program to implement forms and events		24/8/23	
7)	a. write a program to implement Reacts JS Router and animation. b. write a program to implement concept of React Hooks		31/8/23 4/9/23	Sampled 20/8/23

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
8)	Assignment on REPL commandline		4/9/23	
9)	write a react program to		25/9/23	
	implement Concept of Asynchronous			Approved 20/10/23
	programming using Promises			
10)	write a program in NodeJS		25/9/23	
	to : create , read , write ,			
	rename a file .			
11)	Create a web application		20/10/23	Approved 20/10/23
	that performs CRUD			
	operations			

Animesh Parab T2-T21 88

Assignment 1

Aim: Develop a Web Application by using HTML tags.

Theory: HTML is the standard markup language for creating Web pages. HTML stands for Hyper Text Markup Language

- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

Elements:

The HTML **element** is everything from the start tag to the end tag:

```
<tagname>Content goes here...</tagname>
```

Examples of some HTML elements:

```
<h1>My First Heading</h1>
<p>My first paragraph.</p>
```

The core tags of HTML are:

1. **<html>**: Denotes the beginning and end of an HTML document.
2. **<head>**: Houses meta-information about the page, such as the title and links to external resources.
3. **<body>**: Encloses the visible content of the page, such as text, images, and multimedia.
4. **<h1> - <h6>**: Define headings in descending order of importance.
5. **<p>**: Represents paragraphs of text.
6. **<a>**: Creates hyperlinks to other web pages or resources.
7. ****: Embeds images into the page.
8. **** and ****: Establish unordered and ordered lists, respectively.
9. ****: Represents individual items within lists.
10. **<div>** and ****: Generic containers for grouping and styling elements.

These tags form the building blocks for constructing the structure and content of web pages in HTML.

Implementation:

We Have Created a simple static website that consists of 5 pages. It is a Site called "Hindustani Classical Music and Its Components". We used many elements and tags such as paragraph tag, heading tag, image, audio, video, iframe and tables etc.

Listed Below are the screenshots of our website:

The screenshot shows a static website with a light blue header and footer. The header contains navigation links for 'Portfolio' and 'Contact', and social media icons for Facebook and Twitter. A placeholder image '110 x 110' is present. The main content area includes:

- Animesh Parab**
- IT Student**
- E-mail: animesh.nparab@gmail.com
 - Website: animesh.com
 - Phone: 8282919154
 - Address: Shimpoli, Mumbai, India
- Personal & Professional Informations**
- Animesh is a pre final year student at (Tsec). He has worked with HSBC for over 3 months as a Analyst, where he played the role of a data engineer. He worked as an ETL, Tableau, and SQL developer. Animesh holds a bachelor's degree in Information Technology from Thadomal Shahani Engineering College
- Worked with Company**

Company	Country
HSBC	India
BNP Paribas	India

- Education**
- Thadomal Engineering College**
- University of Mumbai
- Information Technology
- 2021-2025
- Technikal Skills**
- Programming
- Analytical
- Problem Solving
- Project Management**
- Contact**
- Name
- E-mail
- Text
- Send
- [facebook](#) [twitter](#) [Google+](#)

Conclusion:

We created a simple, static website only using vanilla HTML. Other important tools like CSS, Javascript were not used, hence the website looked bad and tacky. Therefore in our next assignment we will use CSS to style and beautify our website.

Assignment no:-2

Aim:

Enhance the web application developed using CSS and CSS3.

LO Mapped: LO2

Theory:

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

. The different properties used are:

- Background properties
- Font properties
- CSS text
- CSS link
- CSS lists
- Hover animations

The different types of selectors used are:

- tag selector • universal selector
- element id • group selector
- element classname
- descendant selector

```
* {
```

```
font-family: 'Poppins', sans-serif;  
}  
  
body{  
    background-color: lightblue;  
}  
  
  
.navbar{  
    display: flex;  
    align-items:center;  
    justify-content: space-between;  
    flex-wrap: wrap;  
    margin-left: 25px;  
    margin-right: 25px;  
}  
  
nav p{  
    margin-left: 25px;  
    font-size: 25px;  
}  
  
nav ul li {  
    display: inline-block;  
    list-style: none;  
    margin: 10px 20px;  
}  
  
nav ul li a {  
    color:black;  
    text-decoration: none;  
    font-size: 18px;  
    position: relative;  
}
```

```
nav ul li a::after {  
    content: " ";  
    width: 0;  
    height: 3px;  
    background: #5741a7;  
    position: absolute;  
    left: 0;  
    bottom: -6px;  
    transition: 0.5s;  
}  
  
nav ul li a:hover::after {  
    width: 100%;  
}  
  
/* about me */  
  
.row {  
    display: flex;  
    margin-bottom: 10%;  
    /* flex-wrap: wrap; */  
}  
.col-1{  
    margin-left: 30%;  
    margin-right: 15%;  
}  
.col-2{  
    margin-right: 20%;  
}  
  
hr{  
    border-style: dotted;  
    border-width: 7px;  
    border-bottom: 0;
```

```
    width: 10%;  
    color: #11999E;  
}  
  
/* projects */  
  
.projects{  
    display: flex;  
    gap:1rem  
}  
  
.card{  
    margin: 1rem;  
    padding: 1rem;  
    overflow: hidden;  
    border-radius: 12px;  
    border: solid .5px black;  
    width: 50%;  
}  
.card h3{  
    margin-top: 0px;  
}  
.card img{  
    display: block;  
    margin: auto;  
}  
.contact-left{  
    flex-basis: 35%;  
}  
.contact-right{  
    flex-basis: 60%;  
}  
form input,
```

```
form textarea {
    width: 80%;
    border: 0;
    outline: none;
    /* background: #b9acac4e; */
    padding: 10px;
    margin: 10px 0;
    /* color: #fff; */
    /* font-size: 18px; */
    border-radius: 6px;
}

.contact{
    padding-left: 30%;
}

.contact-left p {
    margin-left: 50%;
    margin-top: 30px;
}

.contact-left h1{
    margin-left: 50%;
}

.contact-left p i {

    margin-right: 15px;
    font-size: 25px;
}

.social-icons {
    margin-top: 30px;
    margin-left: 50%;
}
```

```
.social-icons a {  
    text-decoration: none;  
    font-size: 30px;  
    margin-right: 15px;  
    color: black;  
    display: inline-block;  
    transition: color 0.5s, transform 0.5s;  
}  
  
.social-icons a:hover {  
    color: #00000068;  
    transform: translateY(-5px);  
}
```

Output:

Hello,

Hi Animesh Parab

Software engineer



About me

Animesh is a pre final year student at Thadomal Shahani engineering college.

He has worked with HSBC as an analyst, where he played the role of a data engineer.

He worked on a software like data visualization.

He holds a degree in Information Technology.

Copyright © Designed by Animesh

Skills

Languages

- C++
- Java
- Python
- Javascript

Frameworks

- HTML
- CSS
- Django
- Node.js
- React.js
- MongoDB

Projects

Student Management System

It is python base Student Management System. It is interactive used by educational institute for computerized attendance of student.



Hospital Management System

It is java base Hospital Management System. It is interactive used by hospital for managing their hospital.



Copyright © Designed by Animesh

Contact Me animesh.nparab@gmail.com 9326211146   

First Name

Your name..

Last Name

Your last name..

Email

Your mail..

Queries/Feedback

Write something..

Copyright © Designed by Animesh

Conclusion:

In this assignment we understood the basic way of designing a webpage using various CSS properties.

Assignment 3

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

Theory: Bootstrap is a popular front-end framework developed by Twitter. It provides a set of tools, components, and styles that make it easier to create responsive and visually appealing websites and web applications. Bootstrap utilizes a combination of HTML, CSS, and JavaScript to create consistent and mobile-first designs.

Some key features of Bootstrap are:

1. **Responsive Grid System:** Bootstrap's responsive grid system allows developers to create flexible and responsive layouts that adapt to various screen sizes and devices. The grid system is based on a 12-column layout structure that can be customized for different breakpoints.
2. **Pre-designed Components:** Bootstrap comes with a wide range of predesigned UI components such as navigation bars, buttons, forms, modals, carousels, and more. These components can be easily integrated into your project, saving time and effort.
3. **Typography and Styling:** Bootstrap includes typography styles, headings, and text utilities that help maintain consistency and readability across the website. It also offers a variety of contextual classes for styling elements like buttons, alerts, and badges.
4. **CSS and JavaScript Plugins:** Bootstrap provides a collection of CSS and JavaScript plugins that enhance the functionality and interactivity of your website. Examples include dropdowns, modals, tooltips, popovers, and more.
5. **Responsive Utilities:** In addition to the grid system, Bootstrap offers responsive utility classes that allow you to control the visibility, spacing, and alignment of elements based on screen sizes.
6. **Customization:** Bootstrap is highly customizable. Developers can use Bootstrap's extensive set of variables to customize colours, typography, and other design aspects to match the project's branding and style.
7. **SASS Support:** Bootstrap can be customized using SASS (Syntactically Awesome Stylesheets), a popular CSS preprocessor. This feature allows for more advanced customization and easier management of styles.
8. **Browser Compatibility:** Bootstrap is designed to work across various modern browsers and devices, ensuring a consistent experience for users.
9. **Community and Documentation:** Bootstrap has a large and active community of developers who contribute to its development and provide support. The official documentation is comprehensive and user-friendly, making it easy to get started and find solutions to common problems.
10. **Accessibility:** Bootstrap strives to be accessible, providing features and guidelines for creating websites that can be used by people with disabilities.

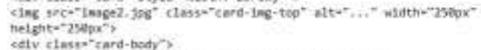
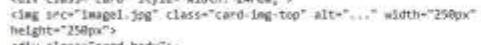
Snapshots of code and output:

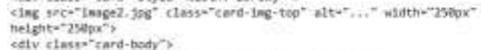
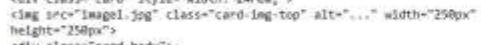
```
index - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQ0U/xprGJmRxqD8zJ4iA6+Zg5fZDy+0l4uKjZLQW9llZqFj0Vx/jI" crossorigin="anonymous">
    <title>Profile</title>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">
            <a class="navbar-brand" href="#">Profile</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav">
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="#">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#">Skills</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#">Contact Us</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>

```

```
index - Notepad
File Edit Format View Help
</div>
</nav>
<div class="container" style="margin-left:2%; margin-top:8%; margin-bottom:6em;">
    <div class="col-sm">
        <div class="row-1">
            <div>Hello,</div>
            <div><strong><font color="Red">Animesh Parab</font></strong></div>
            <div>Software Developer</div>
        </div>
        <div class="row-2">
            
        </div>
        <div>
            <div class="container" style="margin-top: 2em; margin-bottom: 4em; width:70%;">
                <div style="padding-bottom: 1.5em; text-align:center;">About me</div>
                <p>Animesh is a pre final year student at Thadomal Shahani engineering college. He has worked with HSBC as an analyst, where he played the role of a data engineer. He worked on a software like data visualization. He hold a degree in Information Technology.</p>
            </div>
            <!-- -->
            <!-- -->
            <div class="container">
                <h2>Skills</h2>
                <div class="accordion accordion-flush" id="accordionFlushExample">
                    <div class="accordion-item">
                        <h3><button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-target="#flush-collapseOne" aria-expanded="false" aria-controls="flush-collapseOne">Languages</button></h3>
                        <div id="flush-collapseOne" class="accordion-collapse collapse" aria-labelledby="flush-headingOne" data-bs-parent="#accordionFlushExample">
                            <ul>
                                <li>Java</li>
                                <li>Python</li>
                                <li>C/C++</li>
                                <li>SQL</li>
                                <li>HTML/CSS</li>
                                <li>JavaScript</li>
                            </ul>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```
index - Notepad
File Edit Format View Help
<div id="flush-collapseOne" class="accordion-collapse collapse" aria-labelledby="flush-headingOne" data-bs-parent="#accordionFlushExample">
<div class="accordion-body">
<ul>
<li>C++</li>
<li>Java</li>
<li>Python</li>
<li>JavaScript</li>
</ul>
</div>
</div>
<div class="accordion-item">
<h2 class="accordion-header" id="flush-headingTwo">
<button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-target="#flush-collapseTwo" aria-expanded="false" aria-controls="flush-collapseTwo">
Frameworks
</button>
</h2>
<div id="flush-collapseTwo" class="accordion-collapse collapse" aria-labelledby="flush-headingTwo" data-bs-parent="#accordionFlushExample">
<div class="accordion-body">
<ul>
<li>HTML</li>
<li>CSS</li>
<li>Angular</li>
<li>Node.js</li>
<li>React.js</li>
<li>MongoDB</li>
</ul>
</div>
</div>
</div>
<div class="accordion-item">
<h2 class="accordion-header" id="flush-headingThree">
<button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-target="#flush-collapseThree" aria-expanded="false" aria-controls="flush-collapseThree">
Tools
</button>
</h2>
<div id="flush-collapseThree" class="accordion-collapse collapse" aria-labelledby="flush-headingThree" data-bs-parent="#accordionFlushExample">
<div class="accordion-body">Placeholder content for this accordion, which is intended to demonstrate the <code>.accordion-flush</code> class. This is the third item's accordion body. Nothing more exciting happening here in terms of content, but just filling up the space to make it look at least at first glance, a bit more representative of how this would look in a real-world application.</div>
</div>
</div>
</div>
<!-- -->
<div class="container" style="margin-top: 5em;">
<div>
<div class="row">
<div class="col-sm-4">
<div class="card" style="width: 20rem;">

<div class="card-body">
<h5 class="card-title">Hospital Management System
<p class="card-text">It is a system used by hospital for managing their hospital in efficient way.
<a href="#" class="btn btn-primary">visit website
</div>
</div>
</div>
<div class="col-sm-4">
<div class="card" style="width: 20rem;">

<div class="card-body">
<h5 class="card-title">Student management system
</div>
</div>
</div>
</div>
<div class="card-body">
<h5 class="card-title">Student management system
<p class="card-text">It is system used by education institute for managing their institute.
<a href="#" class="btn btn-primary">visit website
</div>
</div>
</div>
</div>
<div style="margin-top: 4em;">
<center>
<p>Copyright © Designed by Animesh Parab</p>
</center>
</div>
</div>
<script>
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcWlfYizIeBhLdNfYiDqyJmEaHvqo9PldvPiPZkPAQjZ" crossorigin="anonymous"></script>
</body>
</html>
```

```
index - Notepad
File Edit Format View Help
Tools
<button>
</h2>
<div id="flush-collapseThree" class="accordion-collapse collapse" aria-labelledby="flush-headingThree" data-bs-parent="#accordionFlushExample">
<div class="accordion-body">Placeholder content for this accordion, which is intended to demonstrate the <code>.accordion-flush</code> class. This is the third item's accordion body. Nothing more exciting happening here in terms of content, but just filling up the space to make it look at least at first glance, a bit more representative of how this would look in a real-world application.</div>
</div>
</div>
</div>
<!-- -->
<div class="container" style="margin-top: 5em;">
<div>
<div class="row">
<div class="col-sm-4">
<div class="card" style="width: 20rem;">

<div class="card-body">
<h5 class="card-title">Hospital Management System
<p class="card-text">It is a system used by hospital for managing their hospital in efficient way.
<a href="#" class="btn btn-primary">visit website
</div>
</div>
</div>
<div class="col-sm-4">
<div class="card" style="width: 20rem;">

<div class="card-body">
<h5 class="card-title">Student management system
</div>
</div>
</div>
</div>
<div class="card-body">
<h5 class="card-title">Student management system
<p class="card-text">It is system used by education institute for managing their institute.
<a href="#" class="btn btn-primary">visit website
</div>
</div>
</div>
</div>
<div style="margin-top: 4em;">
<center>
<p>Copyright © Designed by Animesh Parab</p>
</center>
</div>
</div>
<script>
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcWlfYizIeBhLdNfYiDqyJmEaHvqo9PldvPiPZkPAQjZ" crossorigin="anonymous"></script>
</body>
</html>
```

```
<!-- Contact - Netpad -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQNznx95WgDngx0CaGSpQyGd6IvBf6d1Z1+1BtWz/1cB" crossorigin="anonymous">
    <title>Contact</title>
  </head>
  <body>
    <!-- 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>E-Commerce | Sign Up (title)</title>
        <link rel="stylesheet" href="dist/css/bootstrap.min.css">
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.38.5/font/bootstrap-icons.css">
      </head>
      <body>
        <!-- Navbar -->
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
          <div class="container-fluid">
            <a class="navbar-brand" href="#">Portifolio</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
              <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
              <ul class="navbar-nav">
                <li class="nav-item">
```

```
contact - Notepad
File Edit Format View Help
<ul class="nav-item">
  <li class="nav-link active" aria-current="page"
    href="#>Home/<a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Skills</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Contact Us</a>
  </li>
</ul>
</div>
</nav>
<!-- contact us page -->
<div class="container">
  <div class="row mt-5">
    <div class="col-6">
      <h3>Contact Me</h3>
      <i class="bi bi-telephone" style="display: block;"> +91 9326211146</i>
      <i class="bi bi-envelope" style="display: block;"> amresh.narwah@gmail.com</i>
      <i class="bi bi-linkedin" style="width: 12px; height:15px;"></i>
      <i class="bi bi-github" style="width: 12px; height:15px;"></i>
      <i class="bi bi-instagram" style="width: 12px; height:15px;"></i>
    </div>
    <div class="col-6">
      <form action="#">
        <div class="form-group mb-2">
          <label for="firstname" class="form-label">First Name:</label>
          <input type="text" id="firstname" class="form-control" placeholder="John">
        </div>
        <div class="form-group mb-2">
          <label for="lastname" class="form-label">Last Name:</label>
          <input type="text" id="lastname" class="form-control" placeholder="Doe">
        </div>
        <div class="form-group mb-2">
          <label for="email" class="form-label">Email Address:</label>
          <input type="text" id="email" class="form-control" placeholder="john.doe@example.com">
        </div>
        <div class="form-group mb-2">
          <label for="message" class="form-label">Message:</label>
          <input type="text" id="message" class="form-control" placeholder="Type your message here...">
        </div>
        <div class="d-grid">
          <button type="submit" class="btn btn-primary" style="width: 100%;">Send Message
        </div>
      </form>
    </div>
  </div>
</div>
```

```
contact - Notepad
File Edit Format View Help
height:15px"></i>
</div>
<div class="col-6">
<form action="">
<div class="form-group mb-2">
<label for="firstname" class="form-label">First
Name</label>
<input type="text" id="firstname" class="form-control"
placeholder="John">
</div>
<div class="form-group mb-2">
<label for="lastname" class="form-label">Last
Name</label>
<input type="text" id="lastname" class="form-control"
placeholder="Doe">
</div>
<div class="form-group mb-2">
<label for="email" class="form-label">Email</label>
<input type="email" id="email" class="form-control"
placeholder="xyz@gmail.com">
</div>
<div class="mb-3">
<label for="feedback" class="form-label">feedback</label>
<textarea class="form-control" id="feedback" rows="3"
placeholder="Write something here"></textarea>
</div>
<div class="d-grid col-6 mx-auto">
<button class="btn btn-outline-primary "
type="submit">Submit</button>
</div>
</form>
</div>
<br>
<br>
<footer>
<center>
<p>Copyright &copy; Designed by Animesh Parab</p>
</center>
</footer>
<script src="dist/js/bootstrap.min.js"></script>
</body>
</html>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min
.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UYjoMp4YLEuNSfAP+JcXn/twtIaxVXM"
crossorigin="anonymous"></script>
</body>
</html>
```

<

Profile Home Skills Contact Us

Hello,
Hii Animesh Parab
Software Developer



About me

Animesh is a pre final year student at Thadomal Shahani engineering college. He has worked with HSBC as an analyst, where he played the role of a data engineer. He worked on a software like data visualization. He holds a degree in Information Technology.

Skills

Languages:

Frameworks:

Tools:

Projects



Hospital Management System
It is a web app used by hospital for managing their hospital treatment app.

[View website](#)



Student management system
It is a system used by education institute for managing their institute.

[View website](#)

Copyright © Designed by Animesh Parab

Profile Home Skills Contact Us

Contact Me

+91 9326271146
animesh.parab@gmail.com

First Name:

Last Name:

Email:

Feedback:

[Submit](#)

Copyright © Designed by Animesh Parab

Conclusion:

Bootstrap is a powerful tool that simplifies web development by providing a ready-to-use set of components and styles. It empowers developers to create professional-looking and responsive websites efficiently. Understanding the key features of Bootstrap and its application in creating responsive layouts is essential for modern web development. With a supportive community and thorough documentation, Bootstrap remains an indispensable tool for enhancing both the efficiency of development and the overall quality of user interaction.

Name: Animesh Parab

Batch: T2- T21

Roll no:88

Aim: WAP in JS to study conditional Statements, Loops and Functions

Write a java script for simple calculation i.e. add, subtract, division, multiplication.

```
<!DOCTYPE html>
<html>
<head>
    <title>Calculator</title>
</head>
<body>
    <h1>Calculator</h1>

    <div class="calculator">
        <input type="number" id="num1" placeholder="Enter first number">
        <input type="number" id="num2" placeholder="Enter second number">

        <button id="add">Add (+)</button>
        <button id="subtract">Subtract (-)</button>
        <button id="multiply">Multiply (*)</button>
        <button id="divide">Divide (/)</button>

        <p id="result"></p>
    </div>
```

```
<script>

function performOperation(operation) {
    const num1 = parseFloat(document.getElementById("num1").value);
    const num2 = parseFloat(document.getElementById("num2").value);
    let result;

    switch (operation) {
        case "add":
            result = num1 + num2;
            break;
        case "subtract":
            result = num1 - num2;
            break;
        case "multiply":
            result = num1 * num2;
            break;
        case "divide":
            if (num2 === 0) {
                document.getElementById("result").textContent = "Cannot divide by zero";
                return;
            }
            result = num1 / num2;
            break;
    }

    document.getElementById("result").textContent = `Result: ${result}`;
}
```

```
document.getElementById("add").addEventListener("click", function() {  
    performOperation("add");  
});  
  
document.getElementById("subtract").addEventListener("click", function() {  
    performOperation("subtract");  
});  
  
document.getElementById("multiply").addEventListener("click", function() {  
    performOperation("multiply");  
});  
  
document.getElementById("divide").addEventListener("click", function() {  
    performOperation("divide");  
});  
</script>  
</body>  
</html>
```

Calculator

<input type="text" value="1"/>	<input type="text" value="1"/>	Add (+)	Subtract (-)	Multiply (*)	Divide (/)
--------------------------------	--------------------------------	---------	--------------	--------------	------------

Result: 2

Conclusion:

JavaScript's conditional statements, loops, and functions are integral components that empower developers to create dynamic and interactive web applications. By mastering these constructs, programmers can control program flow, efficiently manage repetitive tasks, and encapsulate reusable code. These foundational concepts form the basis of JavaScript programming and are essential for building sophisticated and responsive web experiences.

Assignment 4B

Aim: Write a program on Inheritance, Iterators and Generators.

Theory:

1. Inheritance-

Inheritance is a fundamental concept in object-oriented programming that allows you to create a new class (sub-class or derived class) that inherits properties and behaviours (methods) from an existing class (superclass or base class). This promotes code reusability and hierarchy in your codebase.

In JavaScript, inheritance is achieved through the **class** keyword and the **extends** keyword.

2. Iterators-

Iterators are objects that allow you to traverse through a sequence of values, often used with collections like arrays or custom data structures. They provide a way to access elements one by one in a sequential manner.

In JavaScript, the iterator protocol is implemented using the **Symbol.iterator** symbol and the **.next()** method.

3. Generators-

Generators are a special type of function in JavaScript that can be paused and resumed. They are defined using the **function*** syntax and utilize the **yield** keyword to produce a sequence of values lazily. This is particularly useful for creating iterators and handling asynchronous operations.

Code :

```

<!DOCTYPE html>
<html>
<body>
    <h1>Assignment</h1>

    <h2>Inheritance Example</h2>
    <button onclick="showInheritance()">Show Inheritance</button>
    <div id="inheritancePopup" class="popup">
        <p id="inheritanceOutput"></p>
        <button onclick="closeInheritance()">Close</button>
    </div>

    <h2>Iterator Example</h2>
    <button onclick="runIterator()">Run Iterator</button>
    <p id="iteratorOutput"></p>

    <h2>Generator Example</h2>
    <button onclick="runGenerator()">Run Generator</button>
    <p id="generatorOutput"></p>

    <script>
        // Inheritance example
        class Animal {
            constructor(name) {
                this.name = name;
            }

            makeSound() {
                return 'Some sound';
            }
        }

        class Dog extends Animal {
            makeSound() {
                return 'Woof woof!';
            }
        }

        const myDog = new Dog('Muddy');

        function showInheritance() {
            document.getElementById('inheritancePopup').style.display = 'block';
            document.getElementById('inheritanceOutput').textContent = `Dog name: ${myDog.name}, Sound: ${myDog.makeSound()}`;
        }

        function closeInheritance() {
            document.getElementById('inheritancePopup').style.display = 'none';
        }
    ... (more code here)
    
```



```

        // Iterator example
        const fruits = ['Apple', 'Banana', 'Cherry'];

        function createFruitIterator(array) {
            let index = 0;

            return {
                next: function() {
                    if (index < array.length) {
                        return { value: array[index++], done: false };
                    } else {
                        return { done: true };
                    }
                }
            };
        }

        function runIterator() {
            const fruitIterator = createFruitIterator(fruits);
            let iteratorOutput = 'Fruits: ';
            while (true) {
                const result = fruitIterator.next();
                if (result.done) break;
                iteratorOutput += result.value + ', ';
            }
            document.getElementById('iteratorOutput').textContent = iteratorOutput;
        }

        // Generator example
        function* generateNumbers() {
            let num = 1;

            while (num <= 5) {
                yield num;
                num++;
            }
        }

        function runGenerator() {
            const numberGenerator = generateNumbers();
            let generatorOutput = 'Numbers: ';
            for (const num of numberGenerator) {
                generatorOutput += num + ', ';
            }
            document.getElementById('generatorOutput').textContent = generatorOutput;
        }
    </script>
</body>
</html>

```

Output

Assignment

Inheritance Example

Dog Name: Buddy, Sound: Woof woof!

Iterator Example

Fruits: Apple, Banana, Cherry,

Generator Example

Numbers: 1, 2, 3, 4, 5,

Conclusion:

In conclusion, inheritance in JavaScript enables the creation of class hierarchies, allowing classes to inherit properties and behaviours from parent classes.

Iterators provide a systematic way to traverse through sequences of data, enhancing the control and readability of iteration processes.

Generators introduce a unique approach to creating iterable sequences, allowing functions to yield values in a paused and resumed manner, thereby facilitating efficient lazy generation and asynchronous handling.

Together, these concepts empower developers to structure code more effectively, navigate data collections seamlessly, and implement dynamic value generation with elegance and flexibility in JavaScript programming.

Assignment 5A

Aim: Write a JavaScript Program to study DOM Manipulation and CSS Manipulations.

Theory:

1. DOM Manipulation:

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of a web page as a tree of objects, where each element, attribute, and piece of text is a node in the tree. DOM manipulation involves using JavaScript to interact with and modify the content, structure, and style of a web page.

In JavaScript, you can use various methods and properties to manipulate the DOM.

Some key concepts include:

- Selecting Elements: You can select elements using methods like `document.querySelector()` or `document.getElementById()`.
- Creating Elements: You can create new DOM elements using `document.createElement()` and then append them to the DOM tree with methods like `appendChild()`.
- Modifying Content: You can change the text or HTML content of an element using properties like `textContent` or `innerHTML`.
- Modifying Attributes: You can change attributes of elements using methods like `setAttribute()`.
- Adding Event Listeners: You can attach event listeners to elements to respond to user interactions like clicks, mouseover etc.

2. CSS Manipulation

Cascading Style Sheets (CSS) is used to define the presentation and layout of web documents. CSS manipulation involves using JavaScript to change the styles of elements on a web page dynamically. In JavaScript, you can manipulate CSS styles using the `style` property of DOM elements.

Some key points include:

- Changing Styles: You can modify CSS properties like `backgroundColor`, `color`, `width`, etc using the `style` property.
- Adding/Removing Classes: You can add or remove classes from elements using methods like `classList.add()` and `classList.remove()`.
- Pseudo-Classes: You can target pseudo-classes like `:hover`, `:active`, etc., to apply styles based on user interactions.
- Transitions and Animations: You can create smooth transitions and animations by changing CSS properties with JavaScript over time.

Combining DOM manipulation and CSS manipulation allows you to create dynamic and interactive web pages. These techniques are commonly used to build web applications, enhance user experience, and create visually appealing interfaces.

It's important to note that these manipulations are often performed within an HTML context, as the DOM and CSS styles are integral parts of web development. The examples provided earlier showcase how these concepts work together to create interactive and visually appealing elements on a web page.

Code snapshot:

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

</head>
<body>
    <h1 id="heading">Animesh</h1>
    <button id="redButton">Red</button>
    <button id="greenButton">Green</button>
    <button id="blueButton">Blue</button>

    <script>
        const heading = document.getElementById('heading');
        const redButton = document.getElementById('redButton');
        const greenButton = document.getElementById('greenButton');
        const blueButton = document.getElementById('blueButton');

        redButton.addEventListener('click', () => changeColor('red'));
        greenButton.addEventListener('click', () => changeColor('green'));
        blueButton.addEventListener('click', () => changeColor('blue'));

        function changeColor(color) {
            heading.style.color = color;
        }
    </script>
</body>
</html>
```

Animesh

[Red](#) [Green](#) [Blue](#)

Conclusion:

In conclusion, DOM manipulation and CSS manipulation are two fundamental techniques in JavaScript that empower developers to dynamically interact with and enhance web pages. DOM manipulation involves using JavaScript to navigate, create, modify, and remove elements in the Document Object Model, enabling dynamic content updates and responsive user interactions. CSS manipulation, on the other hand, allows for the modification of element styles, enabling the creation of visually engaging and interactive designs. By combining these techniques, developers can create immersive web experiences, modify content on-the-fly, and tailor user interfaces to deliver seamless and engaging interactions on the modern web.

Assignment 5B

Aim: Write a JavaScript program to:

- a. Implement the concept of Promise(callback)
- b. Fetch (Client Server communication)

Theory:

a. Promises-

In JavaScript, promises are a fundamental concept used to manage asynchronous operations. They provide a clean and structured way to handle asynchronous tasks and avoid the callback hell (also known as "pyramid of doom") that can occur when dealing with multiple nested callbacks.

Promises represent a value (or an error) that might be available now, in the future, or never. They have three states:

- Pending: The initial state of a promise. It represents an ongoing or incomplete operation.
- Fulfilled (Resolved): The state of a promise when the asynchronous operation is successfully completed. The promise transitions to this state with a result value.
- Rejected: The state of a promise when the asynchronous operation encounters an error or fails. The promise transitions to this state with a reason (error) value.

Promises provide a chainable syntax for handling asynchronous operations. The **.then()** method is used to handle fulfillment, and the **.catch()** method is used to handle rejection. Additionally, you can use the **.finally()** method to attach a callback that runs regardless of whether the promise is fulfilled or rejected.

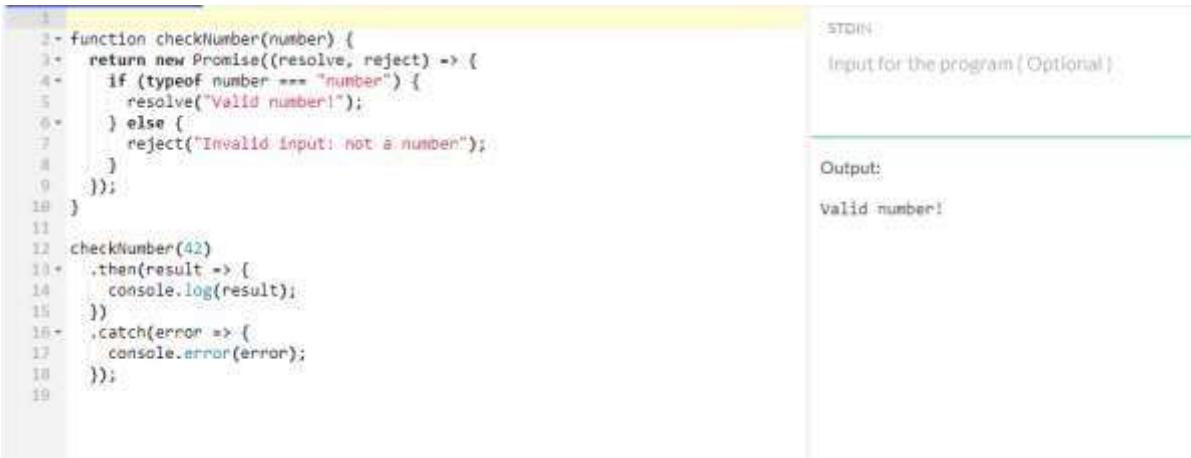
b. Weather-

In JavaScript, the Weather API is a modern and built-in mechanism for making network requests to interact with servers or APIs. It provides a more flexible and powerful alternative to the older **XMLHttpRequest** object for sending and receiving data over HTTP.

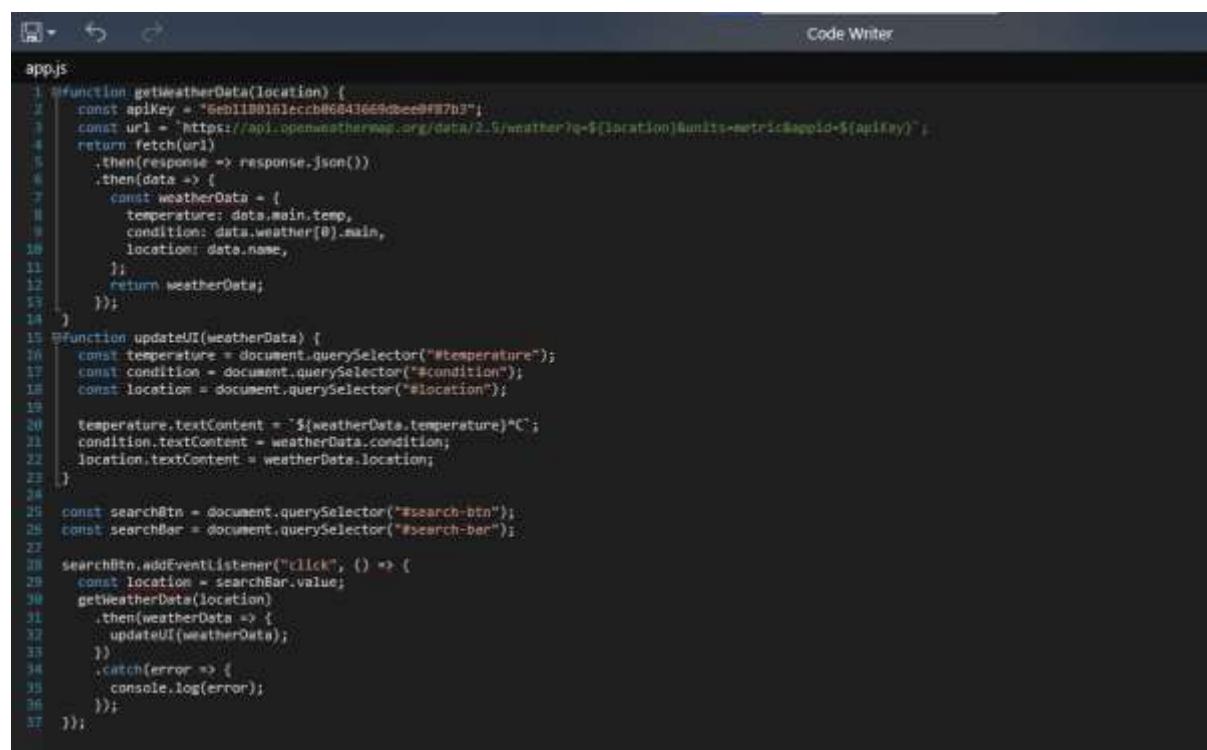
The Weather API is based on promises, which allows for a cleaner and more readable syntax when dealing with asynchronous operations.

The **weather()** function is used to make requests. It takes a URL as its argument and returns a promise that resolves to the response of the request. Methods like **response.json()** or **response.text()** to extract the actual data from the response.

Code snapshots:



```
1 - function checkNumber(number) {
2 -   return new Promise((resolve, reject) => {
3 -     if (typeof number === "number") {
4 -       resolve("Valid number!");
5 -     } else {
6 -       reject("Invalid input: not a number");
7 -     }
8 -   });
9 - }
10-
11 checkNumber(42)
12 .then(result => {
13   console.log(result);
14 })
15 .catch(error => {
16   console.error(error);
17 });
18 );
19
```



```
app.js
1 //function getWeatherData(location) {
2 //  const apiKey = "6eb1bb0161ecch068436c9dbee0ff7b3";
3 //  const url = "https://api.openweathermap.org/data/2.5/weather?q=${location}&units=metric&appid=${apiKey}";
4 //  return fetch(url)
5 //    .then(response => response.json())
6 //    .then(data => {
7 //      const weatherData = {
8 //        temperature: data.main.temp,
9 //        condition: data.weather[0].main,
10 //        location: data.name,
11 //      };
12 //      return weatherData;
13 //    });
14 //
15 //function updateUI(weatherData) {
16 //  const temperature = document.querySelector("#temperature");
17 //  const condition = document.querySelector("#condition");
18 //  const location = document.querySelector("#location");
19 //
20 //  temperature.textContent = `${weatherData.temperature}*C`;
21 //  condition.textContent = weatherData.condition;
22 //  location.textContent = weatherData.location;
23 //}
24 //
25 const searchBtn = document.querySelector("#search-btn");
26 const searchBar = document.querySelector("#search-bar");
27 //
28 searchBtn.addEventListener("click", () => {
29   const location = searchBar.value;
30   getWeatherData(location)
31     .then(weatherData => {
32       updateUI(weatherData);
33     })
34     .catch(error => {
35       console.log(error);
36     });
37 });


```

Out put :

31.05°C

Haze

Delhi

delhi
Search

Conclusion:

In this assignment, we've successfully implemented the concept of Promises to handle asynchronous operations. We also observed reject and resolve state of Promise.

We also showcased the Weather API for client-server communication. Using the **weather** function, we made a request to a sample API, retrieved JSON data from the response, and handled it using promises. This demonstrates how modern web applications can interact with servers to retrieve and process data asynchronously.

Assignment 6a

Aim :- WAP to implement the concept of props(pass date as props in functional component) and state (change the background color)

Theory :-

ReactJS

ReactJS is a popular JavaScript library used for building user interfaces in web applications. It follows the concept of a Virtual DOM (Document Object Model), which is a lightweight representation of the actual DOM in memory. This enables React to efficiently manage and update the UI without directly manipulating the real DOM, resulting in better performance.

Props

Props, short for "properties," are a way to pass data from a parent component to a child component. Props are read-only and are used to communicate information that doesn't change within the child component. They help you create reusable components that can be customized based on the data they receive.

In the parent component, you define the props and their values when rendering the child component:

Code :-

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

const DateTimeDisplay = () => {
  const [currentDateTime, setCurrentDateTime] = useState(new Date());

  useEffect(() => {
    const intervalId = setInterval(() => {
      setCurrentDateTime(new Date());
    }, 1000);

    return () => clearInterval(intervalId);
  }, []);

  const formattedDate = currentDateTime.toDateString();
  const formattedTime = currentDateTime.toLocaleTimeString();

  return (
    <div>
      <p>{formattedDate}</p>
      <p>{formattedTime}</p>
    </div>
  );
}
```

```
<div>
  <h1>Current Date and Time</h1>
  <p>Date: {formattedDate}</p>
  <p>Time: {formattedTime}</p>
</div>
);
};

export default DateTimeDisplay;
```

Current Date and Time

Date: Wed Aug 30 2023

Time: 11:05:17 PM

State:

State represents the mutable data that a component can hold. Unlike props, state is internal to a component and can be changed over time, usually due to user interactions, API calls, or other dynamic factors. When the state of a component changes, React re-renders the component to reflect those changes.

State is initialized within the constructor of a class-based component or using the useState hook in a functional component:

Code

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

```
const ColorChanger = () => {
  const [backgroundColor, setBackgroundColor] = useState('white');

  const changeBackgroundColor = (color) => {
    setBackgroundColor(color);
  };

  return (
    <div style={{ backgroundColor: backgroundColor, padding: '20px' }}>
      <h1>Changing Background Color</h1>
      <button onClick={() => changeBackgroundColor('red')}>Red</button>
      <button onClick={() => changeBackgroundColor('green')}>Green</button>
      <button onClick={() => changeBackgroundColor('blue')}>Blue</button>
      <button onClick={() => changeBackgroundColor('yellow')}>Yellow</button>
      <button onClick={() => changeBackgroundColor('purple')}>Purple</button>
    </div>
  );
};

export default ColorChanger;
```

OutPut :-

Changing Background Color

[Red](#) [Green](#) [Blue](#) [Yellow](#) [Purple](#)

Changing Background Color

[Red](#) [Green](#) [Blue](#) [Yellow](#) [Purple](#)

CONCLUSION:

In this assignment we learnt a way to pass data from a parent component to a child component using props and change the state of a component using the State.

Assignment :- 6B (React JS)

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.

Theory :- Creating a React.js registration form that incorporates various input elements (textbox, textarea, selection input, checkbox, radio button) and handles events such as `onSubmit`, `onClick`, and `onKeyDown` involves several key concepts. Below, I'll provide you with a theoretical overview of these concepts:

1. React Components : In React, user interfaces are built using components. Components are reusable, self-contained pieces of UI that can be composed to create complex applications. For this assignment, you'll need to create a registration form component that encapsulates all the form elements and their behavior.
2. State Management : React components often have state that determines their behavior and appearance. State is managed using the `useState` hook (for functional components) or `this.state` (for class components). You'll use state to keep track of the data entered into the form fields and whether the popup should be displayed.
3. Form Elements : Your registration form will include various form elements:
 - Text Input : For capturing the user's name and email address.
 - Text area : For capturing additional messages or comments.
 - Select Input : For allowing users to choose their gender.
 - Checkbox : For agreeing to terms and conditions.
 - Radio Buttons : For selecting options (e.g., gender).
4. Event Handling :
 - `onChange` : Used to capture user input as they type. For instance, you'll use it to update the state as the user types into text inputs, textarea, or selects options in the form.
 - `onClick` : Applied to buttons to trigger an action when the button is clicked. In your case, it will be used for the reset button.
 - `onSubmit` : Applied to the form itself to handle form submission. When the user clicks the submit button, this event is triggered, allowing you to process the form data.
 - `onKeyDown` : You can use this event to capture key presses on specific elements. For instance, you can use it to trigger an action when a specific key is pressed.
5. Conditional Rendering : You'll use conditional rendering to display the popup/modal when certain conditions are met. In your case, the popup will be displayed when the form is successfully submitted.
6. CSS Styling : CSS is used to style the form elements and the popup/modal. Proper styling enhances the user experience by making the form visually appealing.
7. Handling Form Submission : Inside the `onSubmit` event handler, you can perform actions such as data validation and sending the data to a server. In your example, it displays a popup message indicating successful registration.
8. Popup/Modal : The popup/modal is a UI element that appears on top of the main content, usually to provide additional information or feedback to the user. It's controlled by state and can be shown or hidden based on user actions or conditions.

By understanding these concepts, you can create a React registration form that meets the requirements of your assignment. Remember to carefully structure your React components, manage state effectively, handle events correctly, and apply appropriate CSS styling to make your form and popup visually appealing and functional.

Code :-

The screenshot shows a code editor window titled "Code Writer" with a dark theme. The left pane displays the code for "App.js". The code defines a functional component "RegistrationForm" that uses the useState hook to manage form data and a modal state. It includes event handlers for input changes, submission, and reset, and a modal message component. The right pane shows the "EXPLORER" panel with a list of open documents: "about_us.php", "app.js", "App.js", "login.php", and "registration.php". The status bar at the bottom indicates "Ln 10, Col 18" and "JavaScript".

```
App.js
1 import React, { useState } from 'react';
2 import './App.css';
3
4 function RegistrationForm() {
5   const [formData, setFormData] = useState({
6     name: '',
7     email: '',
8     gender: 'male',
9     message: '',
10    agree: false,
11  });
12
13  const [popupMessage, setPopupMessage] = useState('');
14  const [isPopupOpen, setIsPopupOpen] = useState(false);
15
16  const handleInputChange = (event) => {
17    const { name, value, type, checked } = event.target;
18    const newValue = type === 'checkbox' ? checked : value;
19    setFormData({
20      ...formData,
21      [name]: newValue,
22    });
23  };
24
25  const handleSubmit = (event) => {
26    event.preventDefault();
27    setPopupMessage('Registration successful!');
28    setIsPopupOpen(true);
29  };
30
31  const handleReset = () => {
32    setFormData({
33      name: '',
34      email: '',
35      gender: 'male',
36      message: '',
37      agree: false,
38    });
39  };
40
41  const closePopup = () => {
42    setIsPopupOpen(false);
43  };
44
45  return (
46    <div>
47      <form onSubmit={handleSubmit}>
48        <label>
49          Name:
50          <input
51            type="text"
52            name="name"
53            value={formData.name}
54            onChange={handleInputChange}
55          />
56        </label>
57        <br />
58
59        <label>
60          Email:
61          <input
62            type="email"
63            name="email"
64            value={formData.email}
65          />
66        </label>
67      </form>
68      {popupMessage}
69    </div>
70  );
71}
```

The screenshot shows the same code editor window with the "App.js" code now fully implemented. The "RegistrationForm" component contains JSX for a form with two text inputs and a modal message. The "EXPLORER" panel and status bar remain the same.

```
App.js
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1596
1597
1598
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1798
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1898
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1996
1997
1998
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
20
```

Code Writer

```
App.js
59      <label>
60        Email:
61        <input
62          type="email"
63          name="email"
64          value={formData.email}
65          onChange={handleInputChange}
66        />
67      </label>
68      <br />
69
70      <label>
71        Gender:
72        <select
73          name="gender"
74          value={formData.gender}
75          onChange={handleInputChange}
76        >
77          <option value="male">Male</option>
78          <option value="female">Female</option>
79          <option value="other">Other</option>
80        </select>
81      </label>
82      <br />
83
84      <label>
85        Message:
86        <textarea
87          name="message"
88          value={formData.message}
89          onChange={handleInputChange}
90        />
91      </label>
92      <br />
```

Ready

Ln 92, Col 15 Auto LF JavaScript

Code Writer

```
App.js
54      <form />
55
56      <label>
57        Agree to Terms:
58        <input
59          type="checkbox"
60          name="agree"
61          checked={formData.agree}
62          onChange={handleInputChange}
63        />
64      </label>
65      <br />
66
67      <button type="submit">Submit</button>
68      <button type="button" onClick={handleReset}>
69        Reset
70      </button>
71    </form>
72
73    {isPopupOpen && (
74      <div className="popup">
75        <div className="popup-content">
76          <button className="close-button" onClick={closePopup}>
77            X
78          </button>
79          <p>{popupMessage}</p>
80        </div>
81      </div>
82    );
83  }
84
85  export default RegistrationForm;
86
```

Ready

Ln 126, Col 1 Auto LF JavaScript

```
App - Notepad
File Edit Format View Help

form {
    max-width: 400px;
    margin: 0 auto;
    padding: 20px;
    border: 1px solid #ccc;
    background-color: #f9f9f9;
    border-radius: 5px;
    box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.3);
}

label {
    display: block;
    margin-bottom: 10px;
    font-weight: bold;
}

input[type="text"],
input[type="email"],
select,
textarea {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 4px;
    font-size: 16px;
}

button[type="submit"],
button[type="button"] {
    background-color: #007bff;
    color: #fff;
    border: none;
    padding: 10px 20px;
    border-radius: 4px;
    font-size: 16px;
    cursor: pointer;
    margin-right: 10px;
}

button[type="button"] {
    background-color: #ccc;
    color: #000;
}

label[for="agree"] {
    font-weight: normal;
}

form > * {
    display: flex;
    flex-direction: column;
}

form > * + * {
    margin-top: 10px;
}

form button {
    align-self: flex-start;
}


```

```
App - Notepad
File Edit Format View Help

button[type="submit"],
button[type="button"] {
    background-color: #007bff;
    color: #fff;
    border: none;
    padding: 10px 20px;
    border-radius: 4px;
    font-size: 16px;
    cursor: pointer;
    margin-right: 10px;
}

button[type="button"] {
    background-color: #ccc;
    color: #000;
}

label[for="agree"] {
    font-weight: normal;
}

form > * {
    display: flex;
    flex-direction: column;
}

form > * + * {
    margin-top: 10px;
}

form button {
    align-self: flex-start;
}


```

A screenshot of a Windows Notepad application window titled "App - Notepad". The window contains the following CSS code:

```
.popup {  
    position: fixed;  
    top: 0;  
    left: 0;  
    width: 100%;  
    height: 100%;  
    background: rgba(0, 0, 0, 0.5);  
    display: flex;  
    justify-content: center;  
    align-items: center;  
}  
  
.popup-content {  
    background: #fff;  
    padding: 20px;  
    border-radius: 5px;  
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.5);  
    text-align: center;  
    position: relative;  
}  
  
.close-button {  
    position: absolute;  
    top: 10px;  
    right: 10px;  
    background: transparent;  
    border: none;  
    cursor: pointer;  
    font-size: 20px;  
}  
  
The status bar at the bottom shows "Ln 100. Col 1" and "100% Unix (LF) UTF-8".
```

Code output :-

Name:	Animesh
Email:	animesh.nparab@gmail.com
Gender:	Male
Message:	Good 
Agree to Terms: <input checked="" type="checkbox"/>	
Submit Reset	

After clicking Submit button

The image shows a registration form with the following fields filled:

- Name:** Animesh
- Email:** animesh.nparab@gmail.com
- Gender:** Male
- Message:** Good

A modal window is overlaid on the form, displaying the message "Registration successful!" with an "X" button in the top right corner.

At the bottom of the form, there is an "Agree to Terms:" checkbox followed by a checked checkbox symbol ().

Below the terms checkbox are two buttons: a blue "Submit" button and a grey "Reset" button.

After clicking Reset Button

The image shows a registration form with the following fields:

- Name:** An empty input field.
- Email:** An empty input field.
- Gender:** A dropdown menu showing "Male".
- Message:** An empty text area with a green "G" icon in the top right corner.
- Agree to Terms:** An empty checkbox input.

At the bottom, there are two buttons: a blue "Submit" button and a grey "Reset" button.

Conclusion :- In this assignment, we implemented a React.js registration form that demonstrates several fundamental concepts of React development, including form handling, event management, and state management.

Assignment 7A (React Js)

Aim :- WAP to implement ReactJS Router

Create more than two class or functional components and implement a program for Routing using a browser router.

Theory :-

React Router is a popular library for handling routing in React applications. It allows you to build single-page applications with multiple views and manage the navigation between those views. React Router uses a component-based approach, making it easy to integrate routing into your React application.

Here are the key concepts related to React Router:

1. Router :- The `BrowserRouter` or `HashRouter` component is used to wrap your application. It provides the context for routing and renders the appropriate components based on the URL.
2. Route :- The `Route` component defines a mapping between a URL path and a React component. When the URL matches the path defined in a `Route`, the associated component is rendered.
3. Link :- The `Link` component is used to create navigation links within your application. It renders as an anchor tag (`<a>`) and allows users to navigate to different routes.
4. Switch :- The `Switch` component is used to wrap multiple `Route` components. It ensures that only the first `Route` whose path matches the current URL is rendered. This prevents multiple components from rendering simultaneously.

Code:-

App.js

```
App.js
```

```
File Edit View
import React from 'react';
import { BrowserRouter as Router, Route, Link, Routes } from 'react-router-dom';
import Home from './components/Home';
import About from './components/About';
import Contact from './components/Contact';
import './App.css';

function App() {
  return (
    <Router>
      <div className="app-container">
        <nav>
          <ul>
            <li>
              <Link to='/Home' className='nav-link'>Home</Link>
            </li>
            <li>
              <Link to='/About' className='nav-link'>About</Link>
            </li>
            <li>
              <Link to='/Contact' className='nav-link'>Contact</Link>
            </li>
          </ul>
        </nav>

        <hr />
        <Routes>

          <Route exact path="/Home" element={< Home />}></Route>
          <Route exact path="/About" element={< About />}></Route>
          <Route exact path="/Contact" element={<Contact />}></Route>
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

Ln 1, Col 1

28°C Partly cloudy

Search

File Explorer

Task View

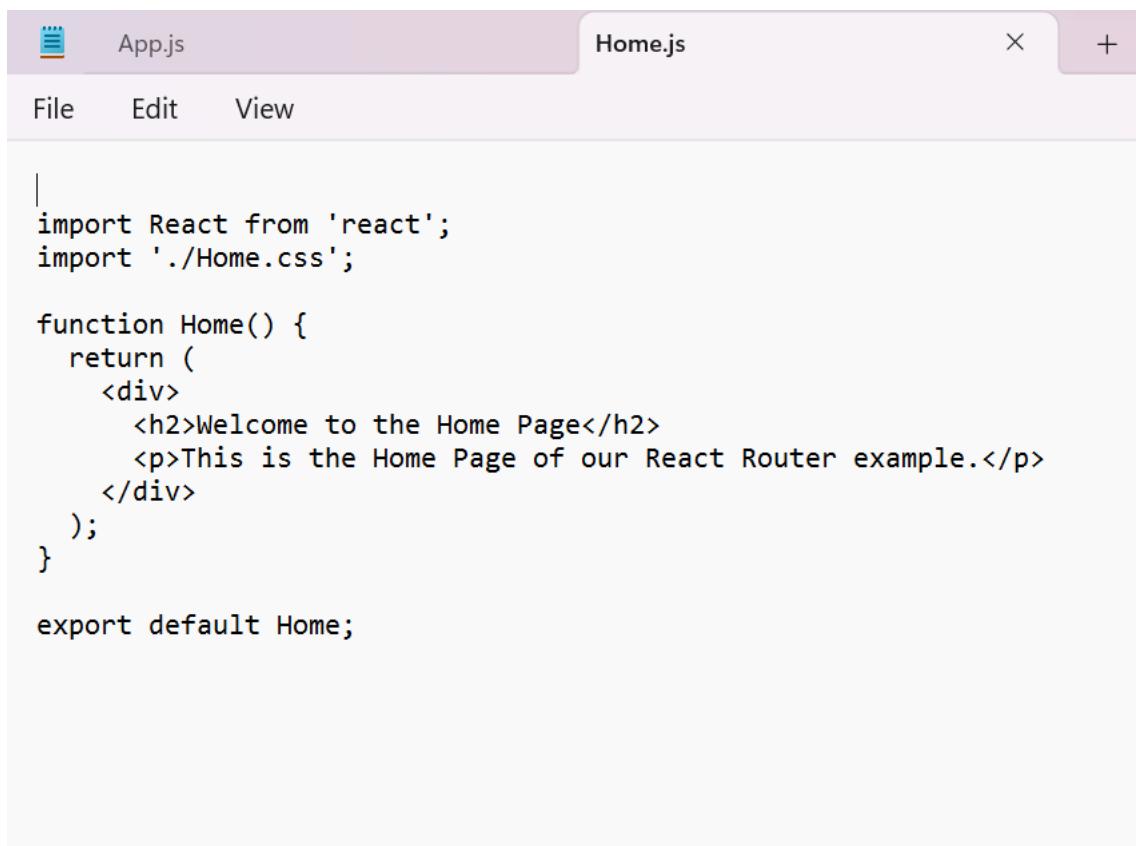
File History

File Preview

File Recovery

L

Home.js



The screenshot shows a code editor window with the following details:

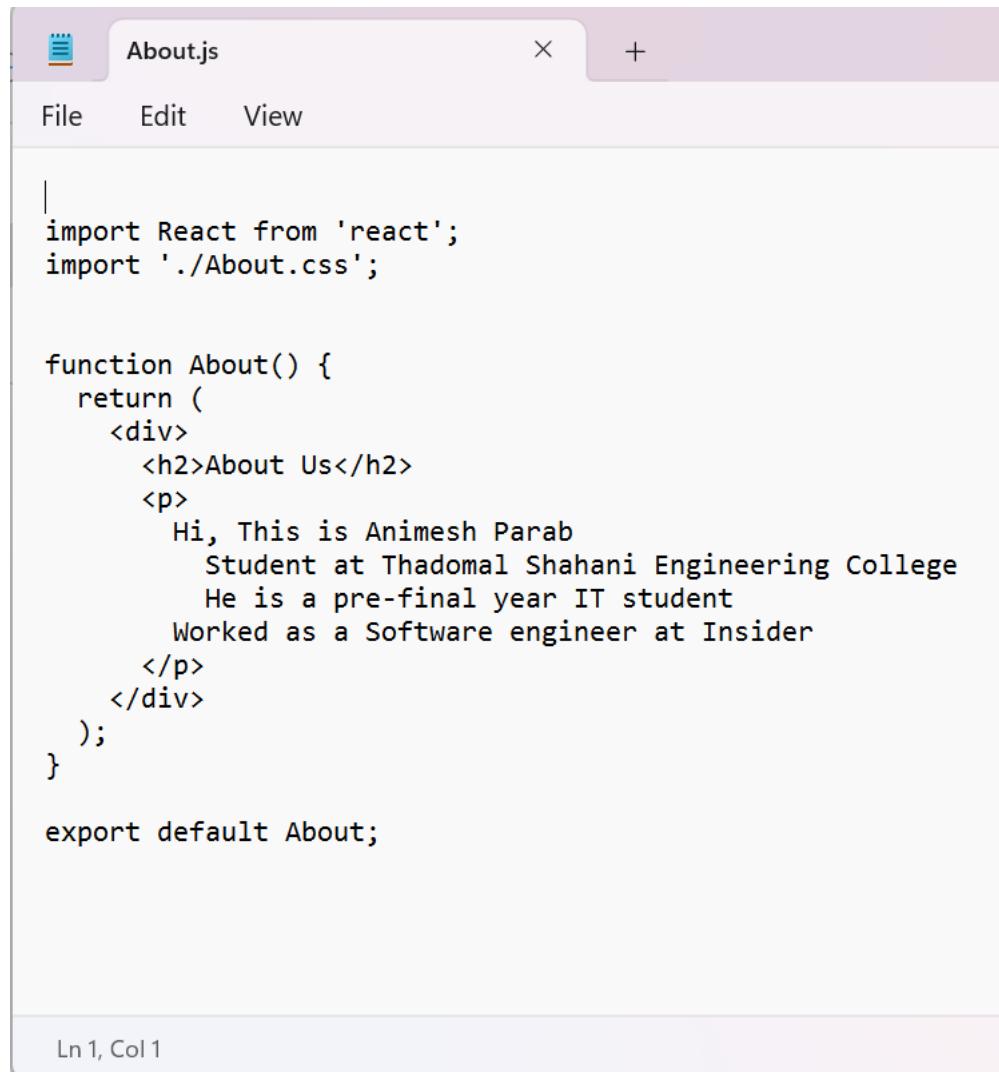
- Title Bar:** The title bar displays "Home.js" in the center, with "App.js" to its left and standard window controls (X, +) to its right.
- Menu Bar:** Below the title bar is a menu bar with "File", "Edit", and "View" options.
- Code Area:** The main area contains the following code:

```
import React from 'react';
import './Home.css';

function Home() {
  return (
    <div>
      <h2>Welcome to the Home Page</h2>
      <p>This is the Home Page of our React Router example.</p>
    </div>
  );
}

export default Home;
```

About.js



The screenshot shows a code editor window with a light purple header bar. The title bar displays the file name "About.js". Below the header is a menu bar with "File", "Edit", and "View" options. The main content area contains the following code:

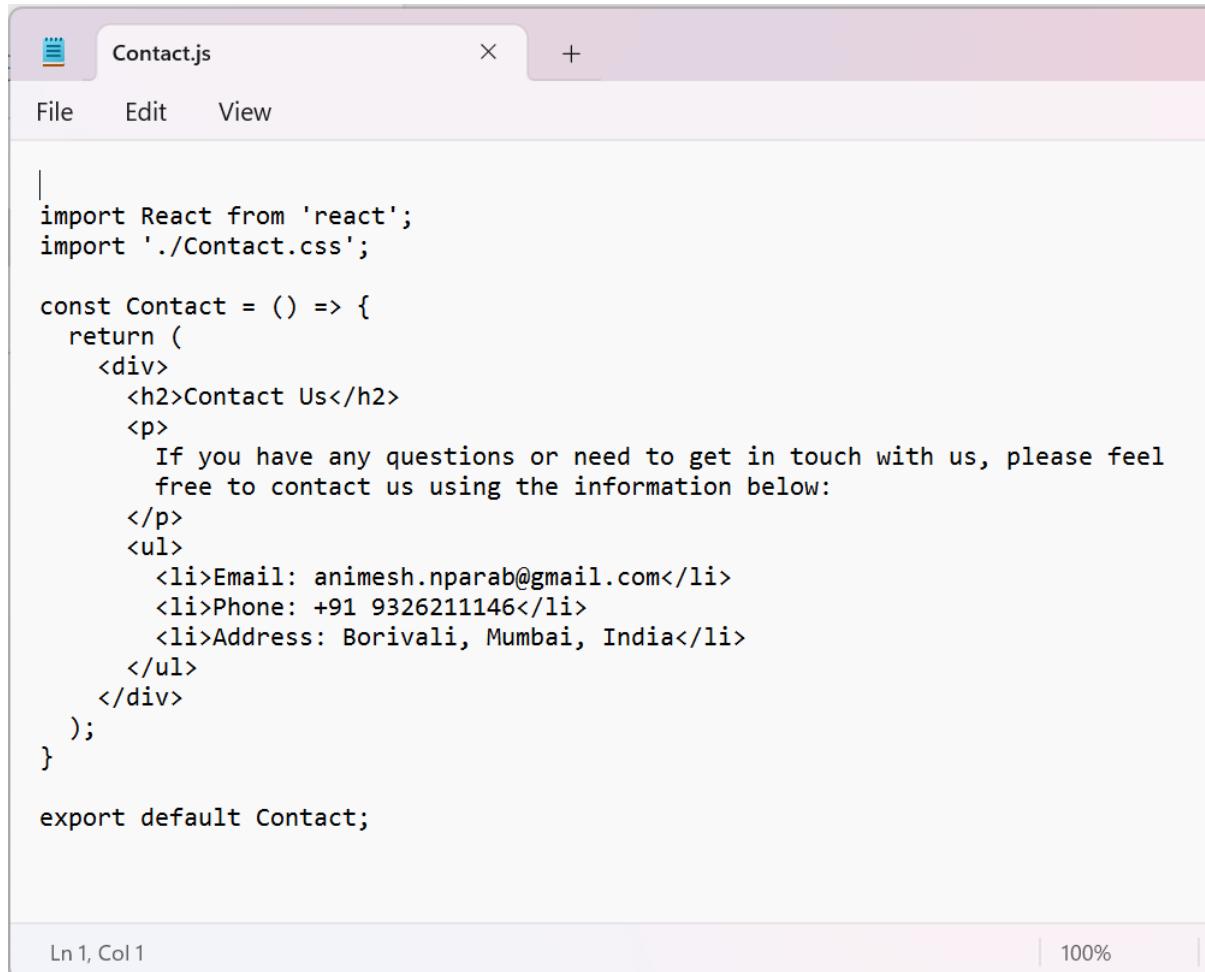
```
import React from 'react';
import './About.css';

function About() {
  return (
    <div>
      <h2>About Us</h2>
      <p>
        Hi, This is Animesh Parab
        Student at Thadomal Shahani Engineering College
        He is a pre-final year IT student
        Worked as a Software engineer at Insider
      </p>
    </div>
  );
}

export default About;
```

In the bottom left corner of the code editor, there is a status bar with the text "Ln 1, Col 1".

Contact.js



The screenshot shows a code editor window with a light gray background. At the top, there's a header bar with a small icon on the left, the title "Contact.js" in the center, and a close button ("X") and a plus sign ("+"). Below the header is a menu bar with "File", "Edit", and "View" options. The main area contains the following code:

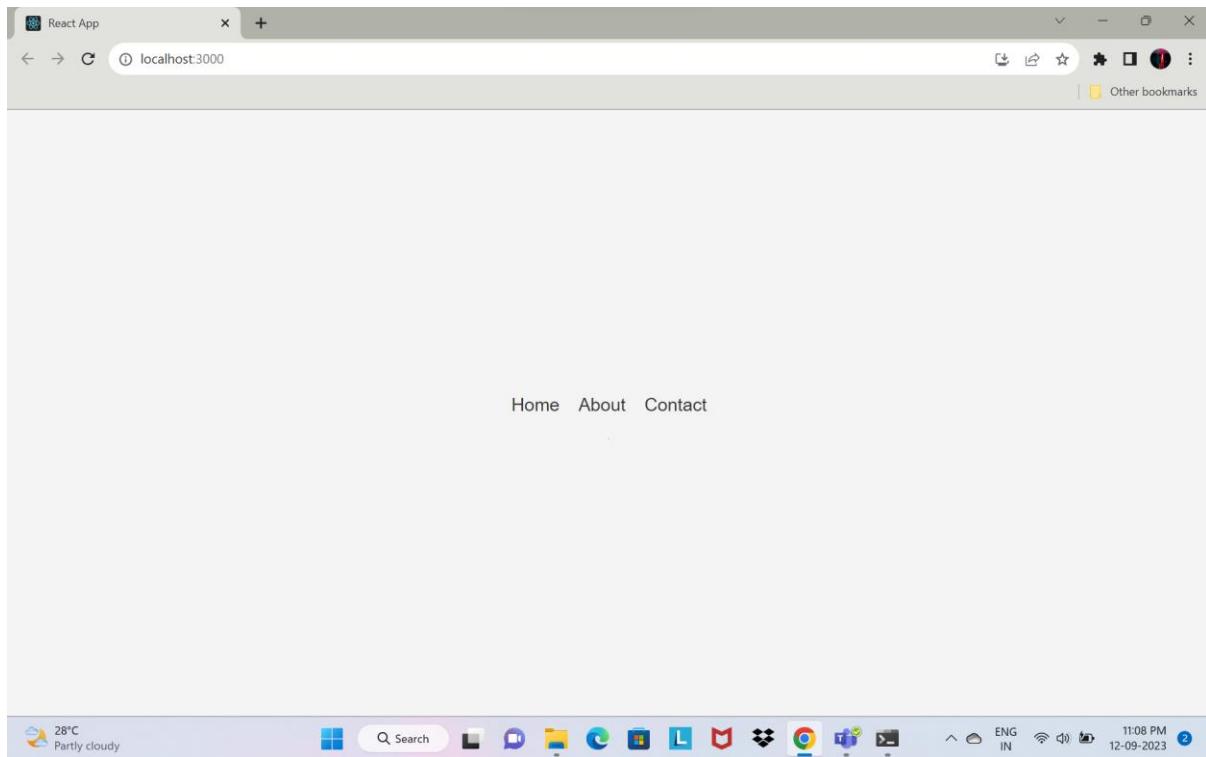
```
import React from 'react';
import './Contact.css';

const Contact = () => {
  return (
    <div>
      <h2>Contact Us</h2>
      <p>
        If you have any questions or need to get in touch with us, please feel
        free to contact us using the information below:
      </p>
      <ul>
        <li>Email: animesh.nparab@gmail.com</li>
        <li>Phone: +91 9326211146</li>
        <li>Address: Borivali, Mumbai, India</li>
      </ul>
    </div>
  );
}

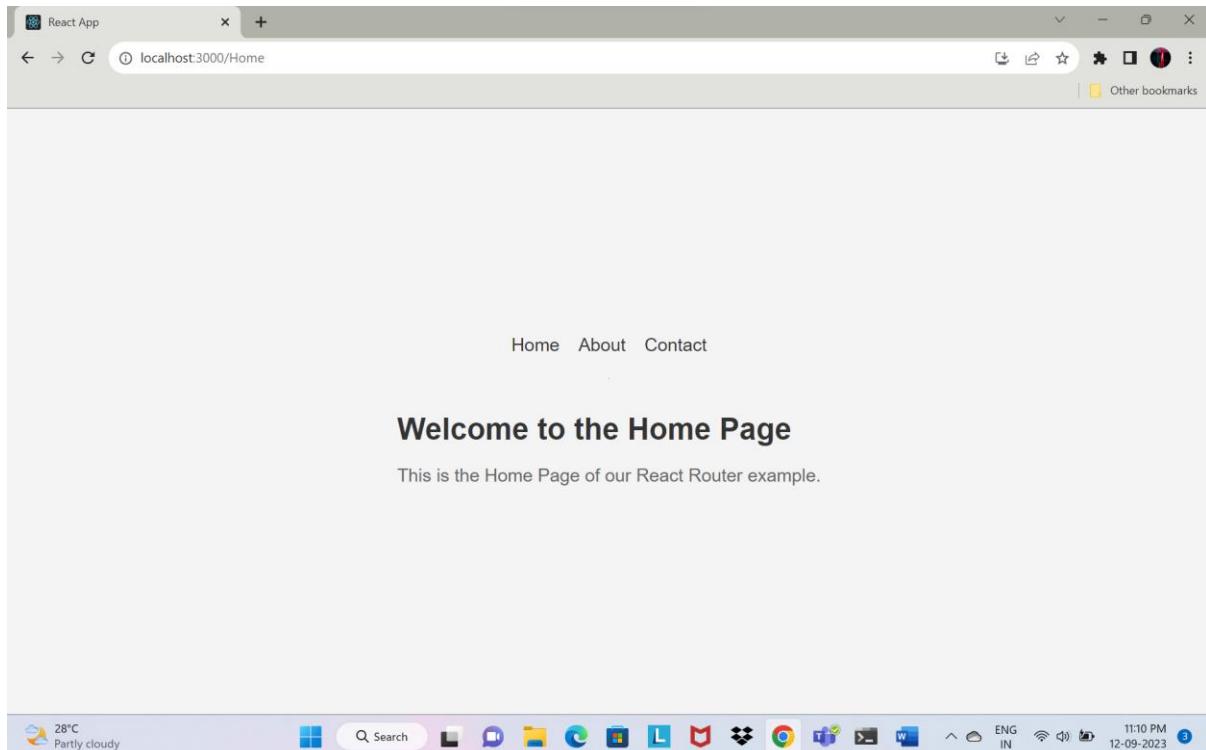
export default Contact;
```

At the bottom left of the editor, it says "Ln 1, Col 1". On the right side, there's a zoom control with "100%".

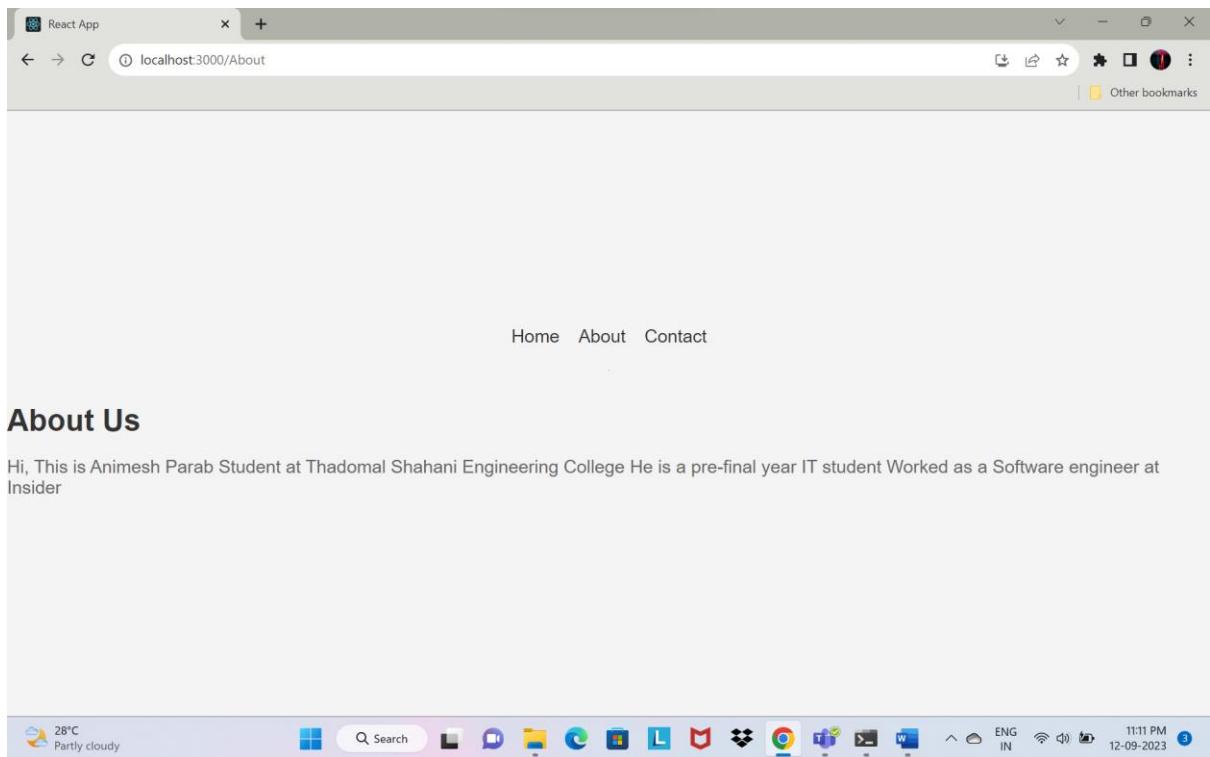
Out Put:-



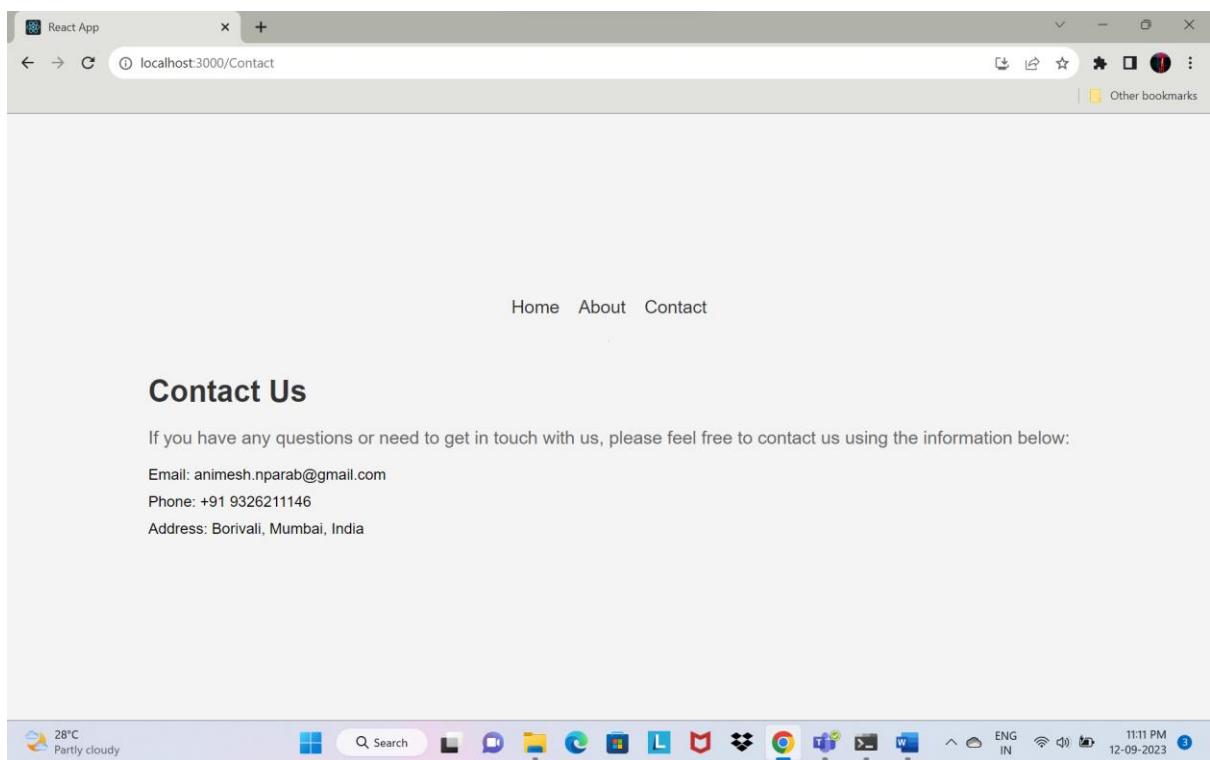
After clicking Home Button



After Clicking the About Button



After Clicking Contact Button



Conclusion:-

In this assignment, we explored the fundamentals of implementing routing in a ReactJS application using the React Router library. Routing is essential for creating single-page applications with multiple views and managing navigation seamlessly.

Animesh Parab 88 T2-T21

Assignment 7B

Aim:- WAP to implement the concept of React Hooks.

Lo Mapped :- Lo 5

Theory:-

This assignment focuses on implementing the concept of React Hooks to handle user input through a text input box. React Hooks, specifically the `useState` Hook, will be employed to create a dynamic and responsive user interface. The theory section will cover the principles of React Hooks and how they enable interactive input handling in React applications.

What are React Hooks?

- React Hooks are functions provided by the React library to allow functional components to manage state and side-effects.
 - They were introduced in React 16.8 to eliminate the need for class components in many cases, enabling state management in functional components.

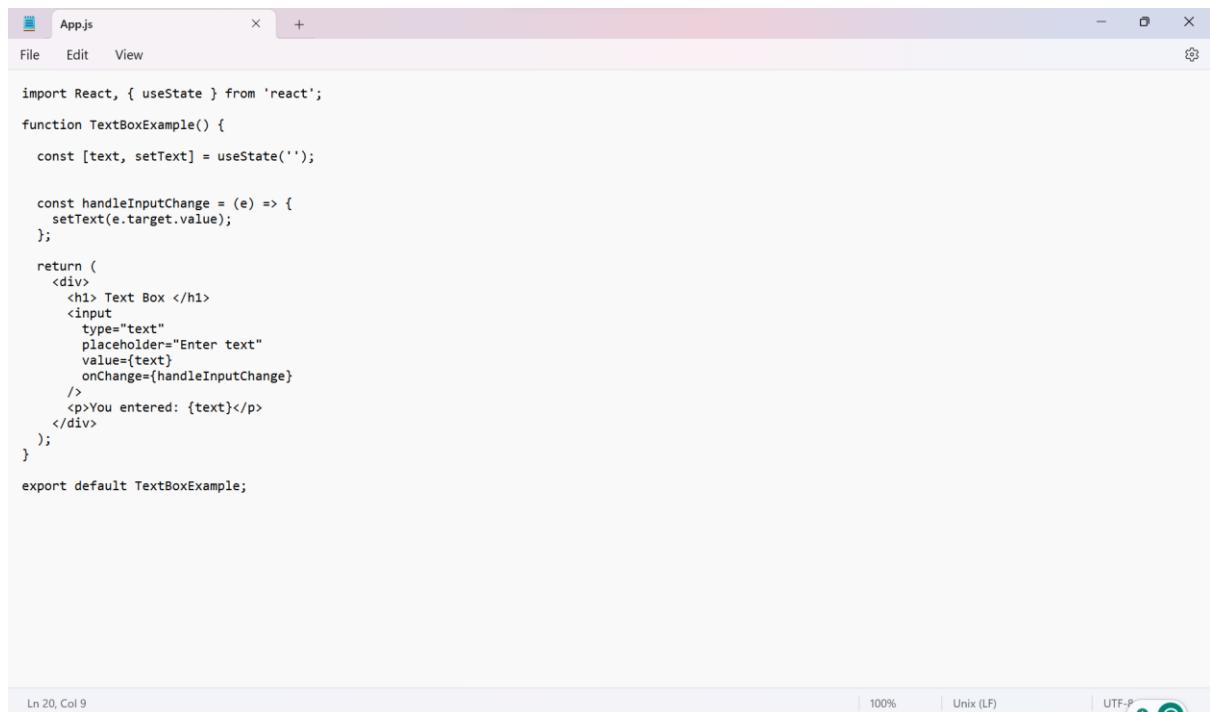
useState Hook:

- The `useState` Hook is used to create and manage state within a functional component.
 - It takes an initial state value as an argument and returns an array with two elements: the current state and a function to update it.
 - The initial state is set when the component is first rendered.

Handling User Input:

- React Hooks can be used to capture and manage user input in real-time.
 - To capture user input, create a state variable using `useState` .
 - Attach an `onChange` event handler to the input element to update the state variable as the user types.
 - The input element's `value` attribute should be set to the state variable to reflect the current input value.

Code :-

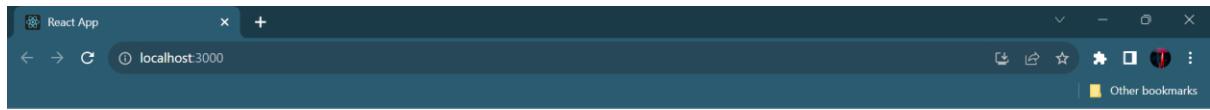


The screenshot shows a code editor window titled "App.js". The code is a React component named "TextBoxExample". It uses the useState hook to manage a text input. The component contains a heading "Text Box", a text input field with placeholder "Enter text", and a paragraph displaying the entered text. The code is as follows:

```
import React, { useState } from 'react';
function TextBoxExample() {
  const [text, setText] = useState('');
  const handleInputChange = (e) => {
    setText(e.target.value);
  };
  return (
    <div>
      <h1> Text Box </h1>
      <input
        type="text"
        placeholder="Enter text"
        value={text}
        onChange={handleInputChange}
      />
      <p>You entered: {text}</p>
    </div>
  );
}
export default TextBoxExample;
```

The status bar at the bottom indicates "Ln 20, Col 9", "100%", "Unix (LF)", and "UTF-8".

Output :-



Text Box

Animesh

You entered: Animesh

Conclusion :-

React Hooks, particularly the `useState` Hook, have transformed the way developers handle user input in React applications. They offer a clean and efficient solution to capture and respond to user interactions, making it easier to create responsive and interactive user interfaces. By incorporating React Hooks into the assignment, students can gain valuable hands-on experience with state management and user input handling, essential skills for building modern web applications using React.

Assignment 8 Node JS (REPL)

Aim:- Assignment on REPL (Commandline)

Lo Mapped :- LO-6

Theory:-

REPL, or Read-Eval-Print Loop, is a crucial tool in the world of programming and development. It offers a dynamic and interactive environment where developers can enter code, receive immediate feedback, and experiment with various programming languages. REPL is instrumental in rapid development, debugging, learning, and data exploration. It fosters a creative and iterative approach to coding, making it an invaluable resource for both beginners and experienced developers. Its ability to provide instant results and encourage experimentation contributes significantly to the efficiency and effectiveness of software development and problem-solving processes.

Components of REPL:-

- Read: In the read phase, the REPL reads user input or code and converts it into a data structure that the programming language can understand.
- Eval: The eval phase processes and evaluates the code or input provided by the user. This involves executing the code and generating results.
- Print: After evaluating the code, the print phase displays the results or output to the user.
- Loop: The loop phase repeats the process, allowing users to enter new code or input continuously.

2. Advantages of REPL:

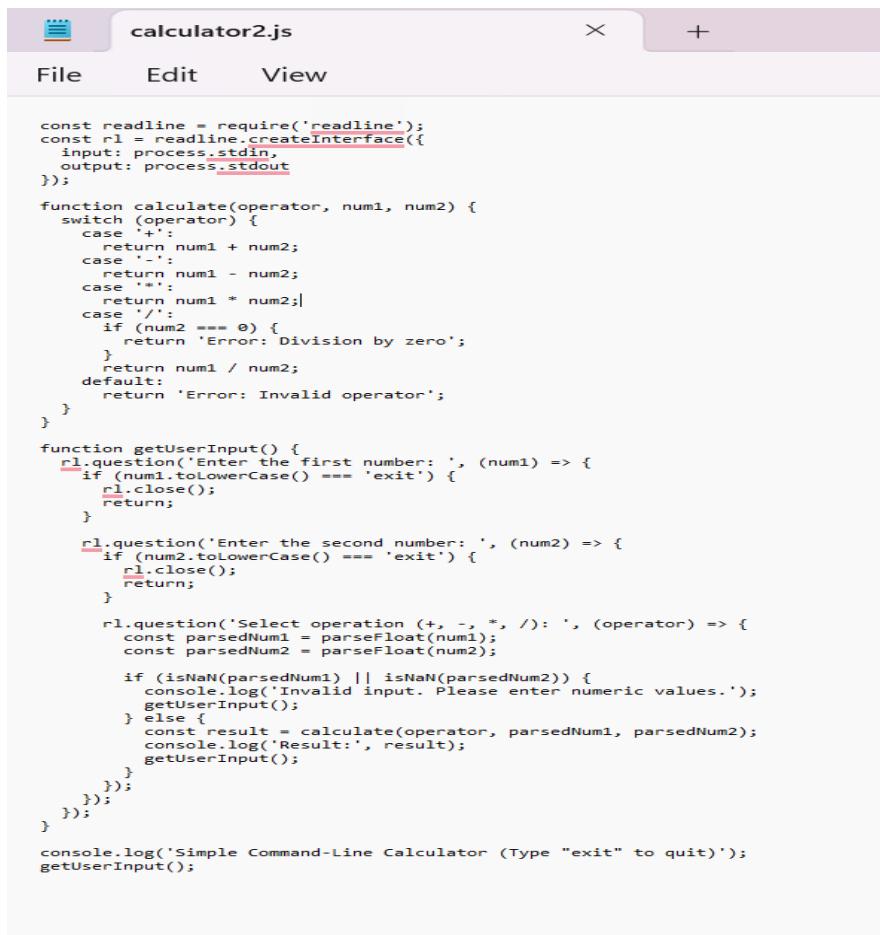
- Rapid Development: REPL enables developers to experiment with code snippets in real-time, making it easier to iterate and refine their code.
- Debugging: Developers can use REPL to test and debug small code segments, identifying and fixing issues quickly.
- Learning Tool: REPL is an excellent tool for learning programming languages as it provides immediate feedback.
- Prototyping: It is useful for quickly prototyping ideas and algorithms.

- Data Exploration: In data analysis and scientific computing, REPL can be used for exploring datasets and running data manipulation operations step by step.

Use Cases:

- Code Testing: Developers can quickly test functions and code snippets.
- Math and Calculations: REPL can be used for performing mathematical calculations and equations.
- Prototyping and Experimentation: It is valuable for experimenting with new ideas and algorithms.
- Data Analysis: Data scientists use REPL to explore datasets and test data manipulation commands.
- Debugging: Debugging small portions of code to identify and fix errors.

Code :-



```

calculator2.js

File Edit View

const readline = require('readline');
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

function calculate(operator, num1, num2) {
  switch (operator) {
    case '+':
      return num1 + num2;
    case '-':
      return num1 - num2;
    case '*':
      return num1 * num2;
    case '/':
      if (num2 === 0) {
        return 'Error: Division by zero';
      }
      return num1 / num2;
    default:
      return 'Error: Invalid operator';
  }
}

function getUserInput() {
  rl.question('Enter the first number: ', (num1) => {
    if (num1.toLowerCase() === 'exit') {
      rl.close();
      return;
    }

    rl.question('Enter the second number: ', (num2) => {
      if (num2.toLowerCase() === 'exit') {
        rl.close();
        return;
      }

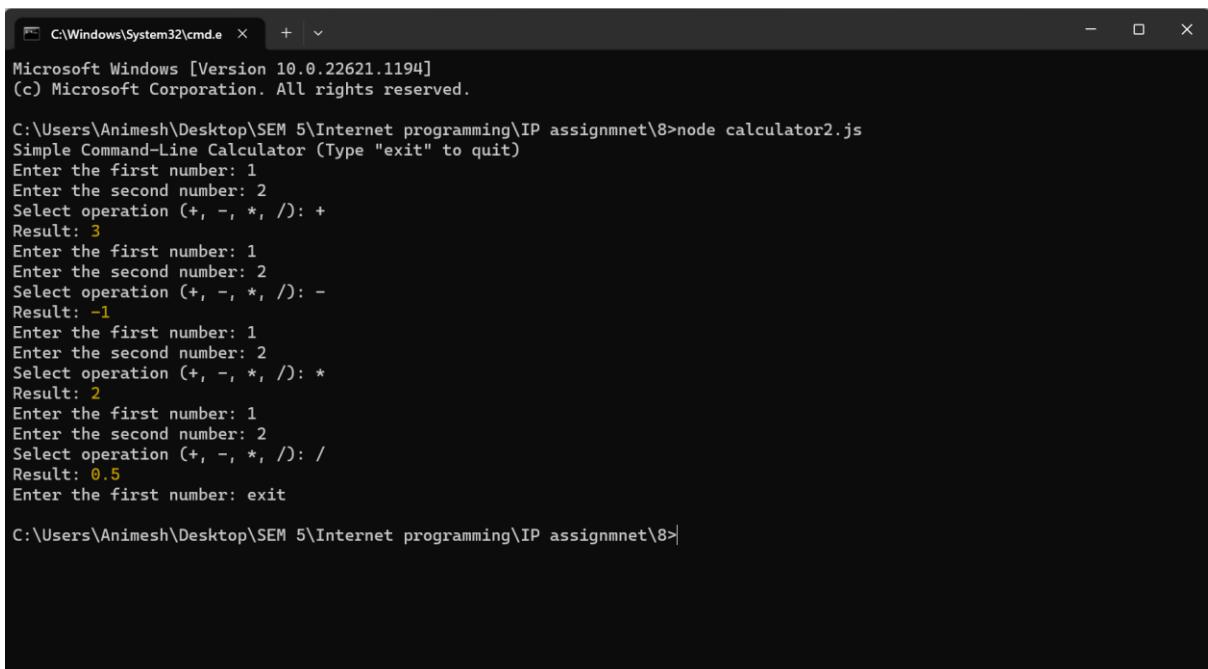
      rl.question('Select operation (+, -, *, /): ', (operator) => {
        const parsedNum1 = parseFloat(num1);
        const parsedNum2 = parseFloat(num2);

        if (isNaN(parsedNum1) || isNaN(parsedNum2)) {
          console.log('Invalid input. Please enter numeric values.');
          getUserInput();
        } else {
          const result = calculate(operator, parsedNum1, parsedNum2);
          console.log('Result:', result);
          getUserInput();
        }
      });
    });
  });
}

console.log('Simple Command-Line Calculator (Type "exit" to quit)');
getUserInput();

```

Output :-



```
C:\Windows\System32\cmd.exe + 
Microsoft Windows [Version 10.0.22621.1194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Animesh\Desktop\SEM 5\Internet programming\IP assignmnet\8>node calculator2.js
Simple Command-Line Calculator (Type "exit" to quit)
Enter the first number: 1
Enter the second number: 2
Select operation (+, -, *, /): +
Result: 3
Enter the first number: 1
Enter the second number: 2
Select operation (+, -, *, /): -
Result: -1
Enter the first number: 1
Enter the second number: 2
Select operation (+, -, *, /): *
Result: 2
Enter the first number: 1
Enter the second number: 2
Select operation (+, -, *, /): /
Result: 0.5
Enter the first number: exit

C:\Users\Animesh\Desktop\SEM 5\Internet programming\IP assignmnet\8>
```

Conclusion:- In this assignment, we have learned REPL ON command line and created program for simple arithmetic operation and output has achieved.

Assignment : 9

Aim : WAP to implement Refs in React.

Theory :

Refs in React are a way to directly access and interact with DOM elements or React components. They provide a way to reference a specific object or object in your application, so that you can read values, trigger actions, or perform other tasks that would be difficult to achieve using React's standard dataflow and state scheduling methods. Refs can help in situations such as accessibility.

Use comments for React refs:

1. DOM Access:

Often Refs are used to interact directly with DOM elements. For example, you can access input values, set focus, or trigger animations.

2. Managing Third Party Libraries:

When integrating React with third-party libraries that require direct DOM manipulation, refs are needed. For example, if you use a charting library or a video player, you can use refs to interact with their API.

1. Animating Elements:

You can use refs to trigger animations by changing CSS classes or properties. This is useful when you need precise control over animations.

2. Scrolling and Measuring Elements:

You can use refs to scroll to specific elements on the page or measure their dimensions. This is helpful for building features like smooth scrolling or lazy-loading content.

3. Integration with Non-React Code:

If you need to integrate React with non-React code, such as legacy JavaScript or libraries like D3.js, you can use refs to bridge the gap between React components and external code.

4. Forms and Input Validation:

Refs can be useful for form handling and input validation. You can access input elements directly to check and manipulate their values or apply custom validation logic.

Creating Refs:

You can create a ref using the **React.createRef()** method (for class components) or the **createRef** hook (for functional components).

Accessing Refs:

You can access the DOM element or React component associated with a ref by using the **current** property of the ref object. This property holds the reference to the underlying element or component. You typically access the ref inside lifecycle methods (for class components) or within the functional component itself.

Forwarding Refs:

Sometimes, you may need to pass a ref from a parent component to a child component. This can be achieved using the "forwarding refs" technique. React provides the **forwardRef** function for this purpose. It allows you to pass a ref down the component hierarchy and access the child component's ref from a parent component.

Callback Refs:

Callback refs are another way to work with refs, especially in cases where you need to perform additional operations when a ref is set. Instead of using the **createRef** or **useRef** functions, you define a callback function that will be called when the ref is attached to an element or component.

Code:

The screenshot shows a code editor window with a pink header bar. The title bar says "App.js". Below it is a menu bar with "File", "Edit", and "View". The main area contains the following code:

```
import React, { useRef } from 'react';

const MyComponent = () => {
    const formInput = useRef(null);

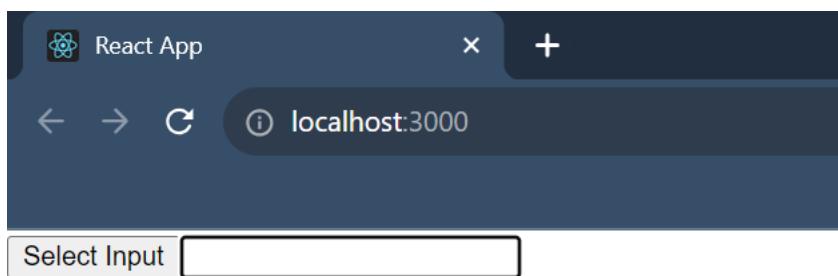
    const inputSelection = () => {
        const input = formInput.current;

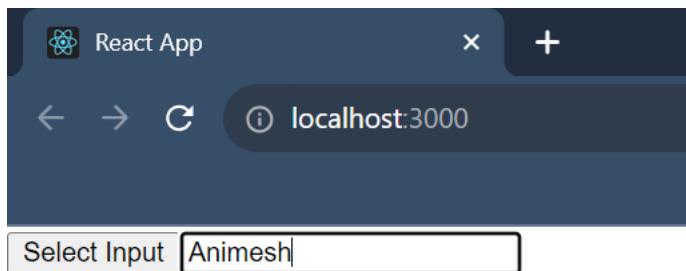
        if (input.value) {
            input.select();
        }
    };

    return (
        <div>
            <button type="button" onclick={inputSelection}>
                Select Input
            </button>
            <input ref={formInput} />
        </div>
    );
};

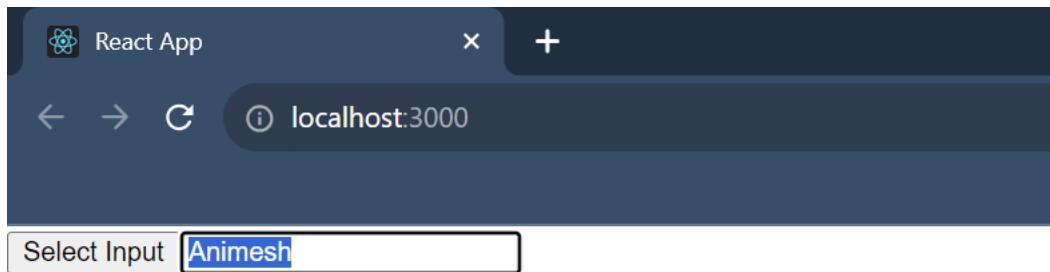
export default MyComponent;
```

Output:





After clicking the button:



Conclusion :

We successfully demonstrated how React refs can be used to create, access, and manipulate DOM elements and components, showcasing their versatility within a React application.

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

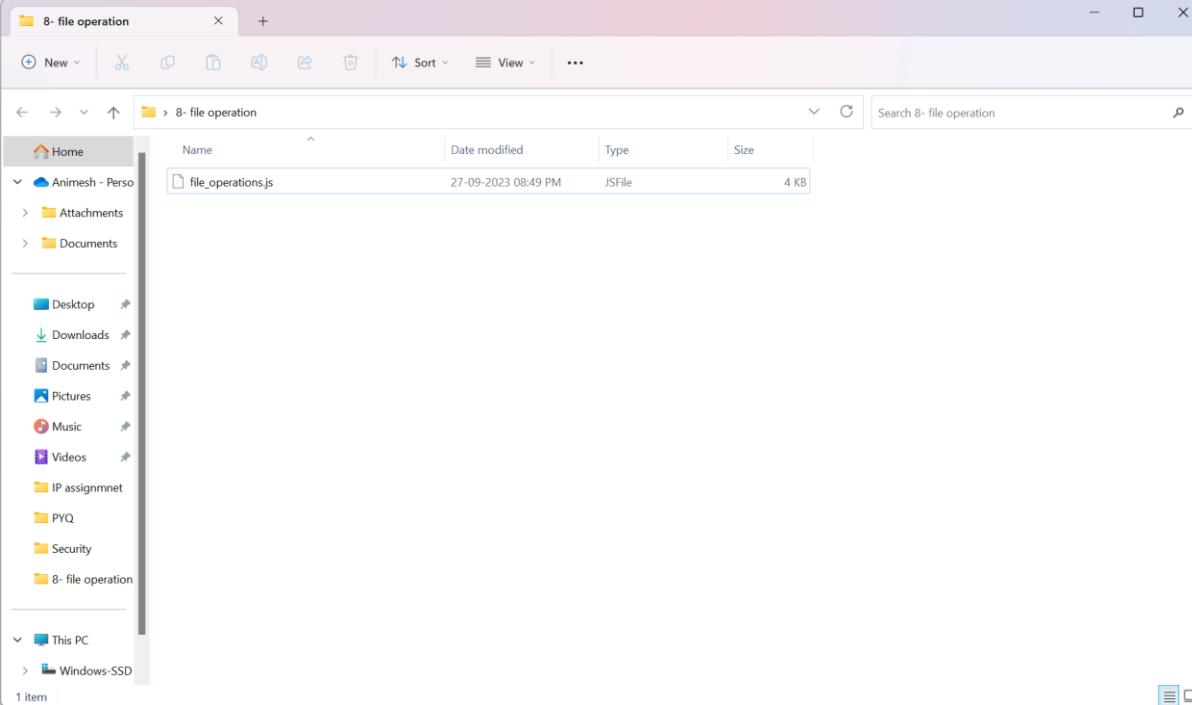
Lo mapped:- LO-6

Theory :-

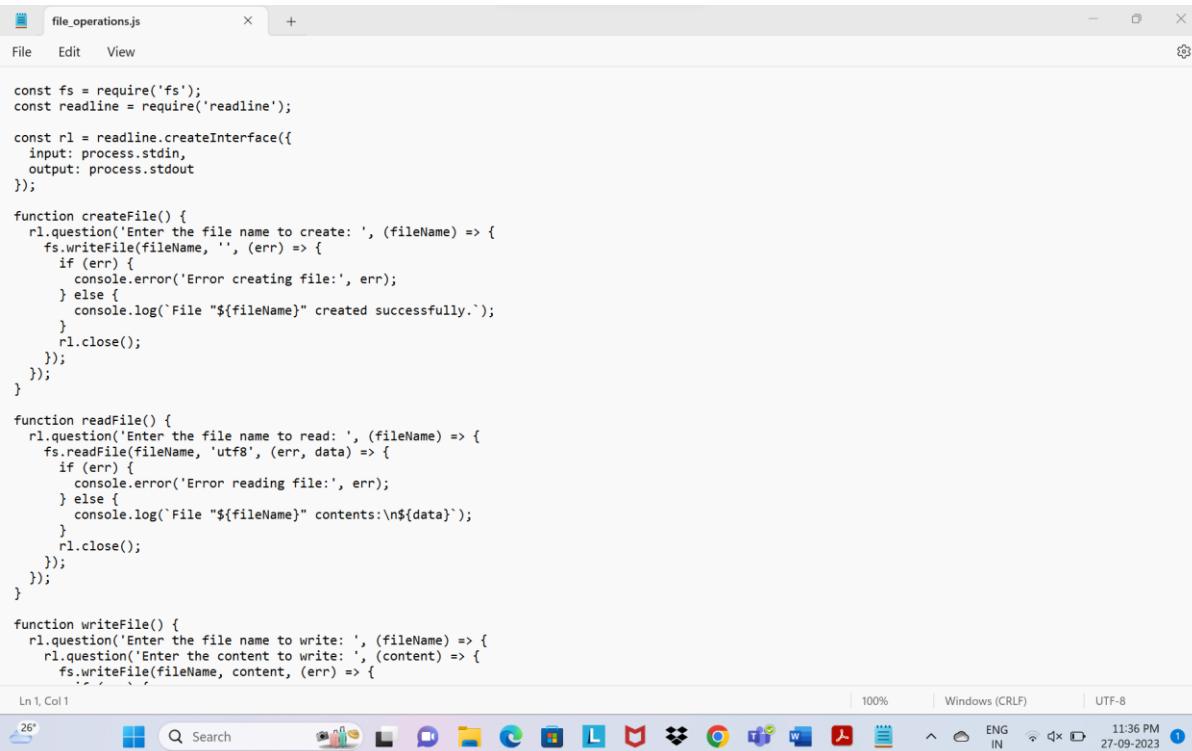
Node.js is a popular JavaScript runtime environment that allows developers to build server-side applications. The assignment leverages Node.js and its built-in `fs` module for handling file-related operations. Additionally, it uses the `readline` module to interactively collect user input from the command line.

1. Creating a File (Option 'a'): The program prompts the user to enter a filename. Upon user input, it uses the `fs.writeFile` method to create an empty file with the specified name.
2. Reading Data from a File (Option 'b'): The program prompts the user to enter the filename they want to read from. It then uses the `fs.readFile` method to read the contents of the specified file and displays the content to the user.
3. Writing Data to a File (Option 'c'): Users are asked to enter both a filename and content. The program utilizes the `fs.writeFile` method to write the provided content to the specified file.
4. Renaming a File (Option 'd'): Users must input the current filename and the new filename. The program utilizes the `fs.rename` method to rename the file.
5. Appending Data to a File (Option 'e'): Users are asked to enter a filename and content to append. The program uses the `fs.appendFile` method to append the provided content to the specified file.
6. Deleting a File (Option 'f'): The program prompts the user to enter the filename they want to delete. It then uses the `fs.unlink` method to delete the specified file.

Code :-



The screenshot shows a Windows File Explorer window titled "8- file operation". The file "file_operations.js" is listed in the folder. The file was created on 27-09-2023 at 08:49 PM and is a JSFile of size 4 KB. The left sidebar shows the user's OneDrive account and various local drives like Desktop, Downloads, Documents, Pictures, Music, Videos, IP assignmnet, PYQ, Security, and the current folder "8- file operation". The bottom status bar shows the file path as "8- file operation\file_operations.js", the file name as "file_operations.js", and the file type as "JSFile".



The screenshot shows a code editor window titled "file_operations.js". The code implements a command-line application using Node.js's fs and readline modules. It includes functions for creating, reading, and writing files. The code uses the readline.createInterface method to handle user input and output. The file operations include prompting the user for a file name, reading its contents, and writing new content to it.

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

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

function createFile() {
  rl.question('Enter the file name to create: ', (fileName) => {
    fs.writeFile(fileName, '', (err) => {
      if (err) {
        console.error('Error creating file:', err);
      } else {
        console.log(`File "${fileName}" created successfully.`);
      }
      rl.close();
    });
  });
}

function readFile() {
  rl.question('Enter the file name to read: ', (fileName) => {
    fs.readFile(fileName, 'utf8', (err, data) => {
      if (err) {
        console.error('Error reading file:', err);
      } else {
        console.log(`File "${fileName}" contents:\n${data}`);
      }
      rl.close();
    });
  });
}

function writeFile() {
  rl.question('Enter the file name to write: ', (fileName) => {
    rl.question('Enter the content to write: ', (content) => {
      fs.writeFile(fileName, content, (err) => {
        if (err) {
          console.error('Error writing file:', err);
        } else {
          console.log(`File "${fileName}" written successfully.`);
        }
      });
    });
}
```

```
function writeFile() {
    rl.question('Enter the file name to write: ', (fileName) => {
        rl.question('Enter the content to write: ', (content) => {
            fs.writeFile(fileName, content, (err) => {
                if (err) {
                    console.error('Error writing to file:', err);
                } else {
                    console.log(`Content written to "${fileName}" successfully.`);
                }
                rl.close();
            });
        });
    });
}

function renameFile() {
    rl.question('Enter the current file name: ', (currentFileName) => {
        rl.question('Enter the new file name: ', (newFileName) => {
            fs.rename(currentFileName, newFileName, (err) => {
                if (err) {
                    console.error('Error renaming file:', err);
                } else {
                    console.log(`File "${currentFileName}" renamed to "${newFileName}" successfully.`);
                }
                rl.close();
            });
        });
    });
}

function appendFile() {
    rl.question('Enter the file name to append to: ', (fileName) => {
        rl.question('Enter the content to append: ', (content) => {
            fs.appendFile(fileName, content, (err) => {
                if (err) {
                    console.error('Error appending to file:', err);
                } else {
                    console.log(`Content appended to "${fileName}" successfully.`);
                }
            });
        });
    });
}

Ln 1, Col 1
```

100% Windows (CRLF) UTF-8 28 27-09-2023

```
function appendFile() {
    rl.question('Enter the file name to append to: ', (fileName) => {
        rl.question('Enter the content to append: ', (content) => {
            fs.appendFile(fileName, content, (err) => {
                if (err) {
                    console.error('Error appending to file:', err);
                } else {
                    console.log(`Content appended to "${fileName}" successfully.`);
                }
                rl.close();
            });
        });
    });
}

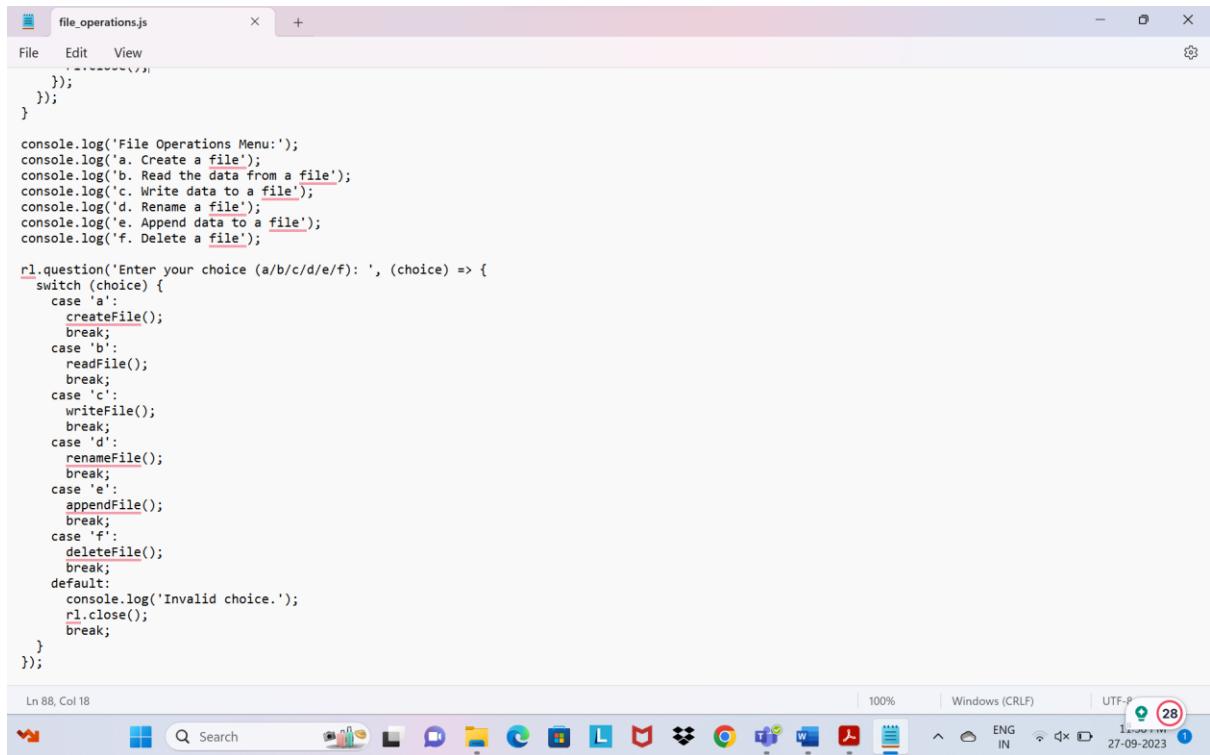
function deleteFile() {
    rl.question('Enter the file name to delete: ', (fileName) => {
        fs.unlink(fileName, (err) => {
            if (err) {
                console.error('Error deleting file:', err);
            } else {
                console.log(`File "${fileName}" deleted successfully.`);
            }
            rl.close();
        });
    });
}

console.log('File Operations Menu:');
console.log('a. Create a file');
console.log('b. Read the data from a file');
console.log('c. Write data to a file');
console.log('d. Rename a file');
console.log('e. Append data to a file');
console.log('f. Delete a file');

rl.question('Enter your choice (a/b/c/d/e/f): ', (choice) => {
    switch (choice) {
        case 'a':
            appendFile();
            break;
        case 'b':
            // Implementation for reading file
            break;
        case 'c':
            // Implementation for writing file
            break;
        case 'd':
            // Implementation for renaming file
            break;
        case 'e':
            // Implementation for appending file
            break;
        case 'f':
            deleteFile();
            break;
    }
});

Ln 88, Col 18
```

100% Windows (CRLF) UTF-8 11:37 PM 27-09-2023



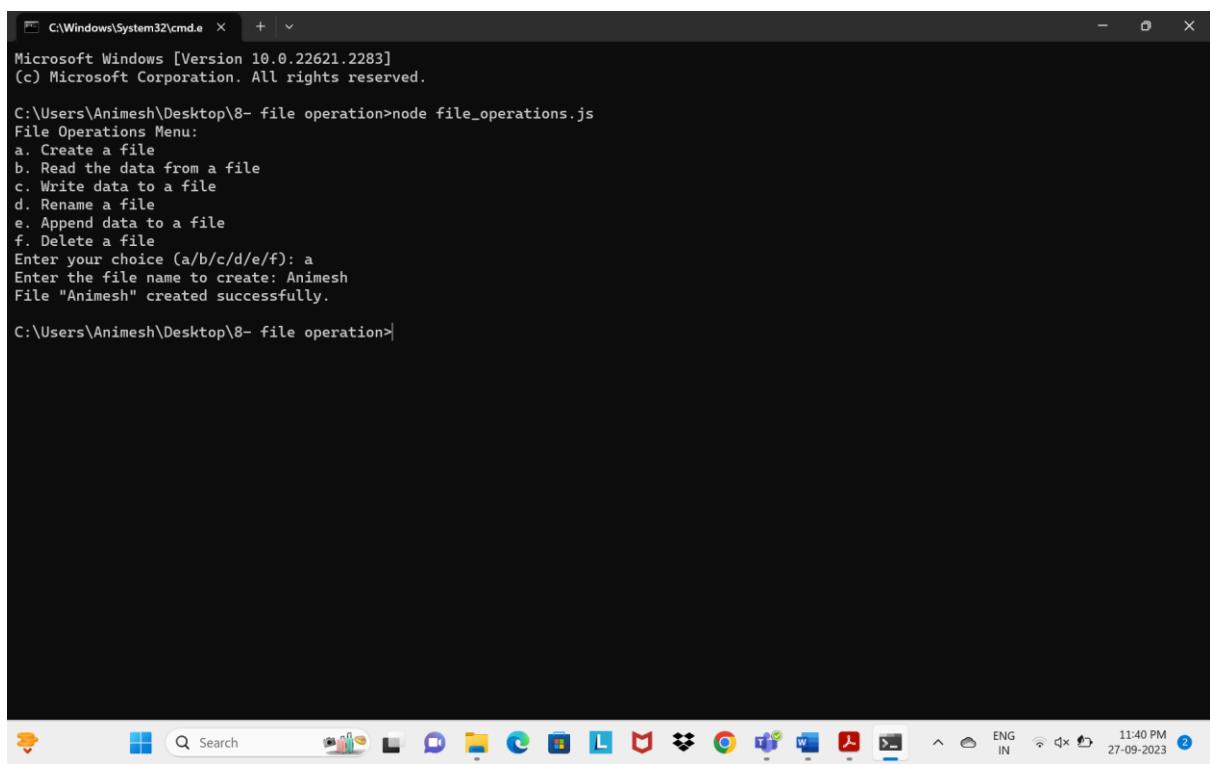
```
file_operations.js
File Edit View
};

};

console.log('File Operations Menu:');
console.log('a. Create a file');
console.log('b. Read the data from a file');
console.log('c. Write data to a file');
console.log('d. Rename a file');
console.log('e. Append data to a file');
console.log('f. Delete a file');

rl.question('Enter your choice (a/b/c/d/e/f): ', (choice) => {
  switch (choice) {
    case 'a':
      createFile();
      break;
    case 'b':
      readfile();
      break;
    case 'c':
      writefile();
      break;
    case 'd':
      renamefile();
      break;
    case 'e':
      appendfile();
      break;
    case 'f':
      deletefile();
      break;
    default:
      console.log('Invalid choice.');
      rl.close();
      break;
  }
});
Ln 88, Col 18
Windows (CRLF)
UTF-P
11:40 PM 28
27-09-2023
```

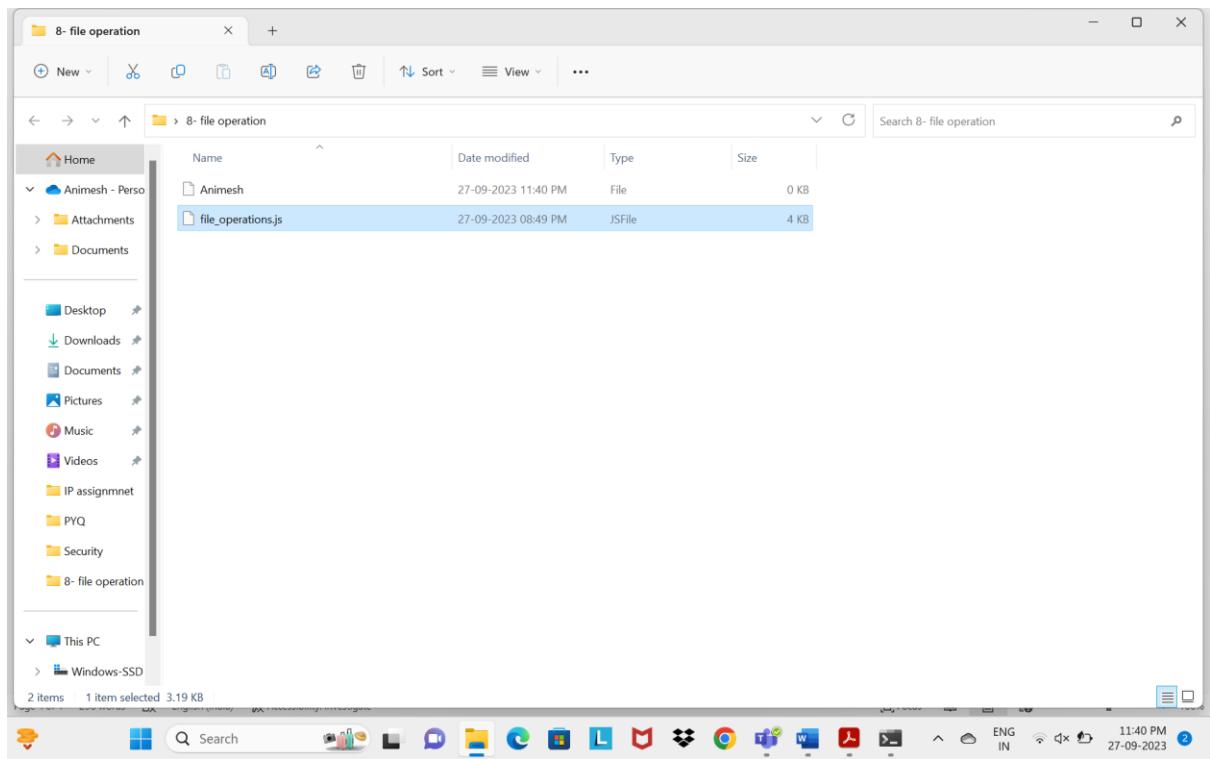
Code output :-



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Animesh\Desktop\8- file operation>node file_operations.js
File Operations Menu:
a. Create a file
b. Read the data from a file
c. Write data to a file
d. Rename a file
e. Append data to a file
f. Delete a file
Enter your choice (a/b/c/d/e/f): a
Enter the file name to create: Animesh
File "Animesh" created successfully.

C:\Users\Animesh\Desktop\8- file operation>
```



For read a data from file and write a data from file

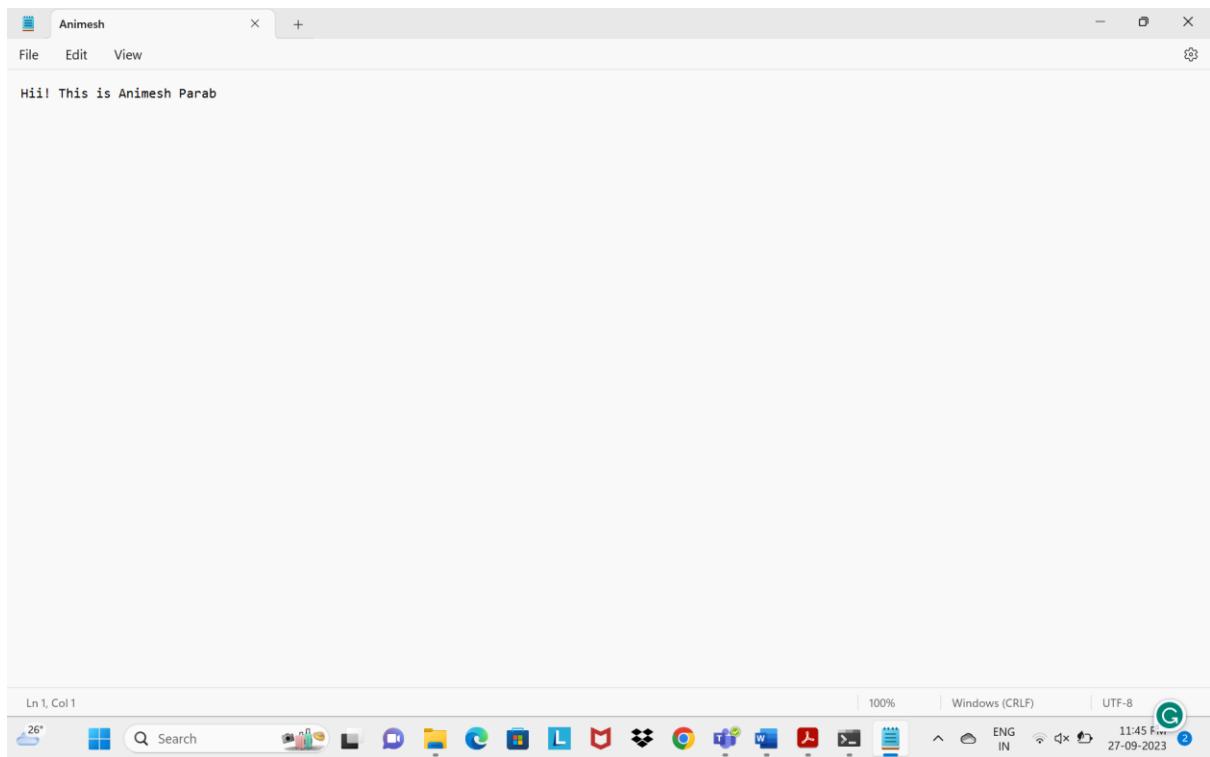
```
C:\Users\Animesh\Desktop\8- file operation>node file_operations.js
File Operations Menu:
a. Create a file
b. Read the data from a file
c. Write data to a file
d. Rename a file
e. Append data to a file
f. Delete a file
Enter your choice (a/b/c/d/e/f): b
Enter the file name to read: Animesh
File "Animesh" contents:

C:\Users\Animesh\Desktop\8- file operation>node file_operations.js
File Operations Menu:
a. Create a file
b. Read the data from a file
c. Write data to a file
d. Rename a file
e. Append data to a file
f. Delete a file
Enter your choice (a/b/c/d/e/f): c
Enter the file name to write: Animesh
Enter the content to write: Hii! This is Animesh Parab
Content written to "Animesh" successfully.

C:\Users\Animesh\Desktop\8- file operation>node file_operations.js
File Operations Menu:
a. Create a file
b. Read the data from a file
c. Write data to a file
d. Rename a file
e. Append data to a file
f. Delete a file
Enter your choice (a/b/c/d/e/f): b
```

```
C:\Users\Animesh\Desktop\8- file operation>node file_operations.js
File Operations Menu:
a. Create a file
b. Read the data from a file
c. Write data to a file
d. Rename a file
e. Append data to a file
f. Delete a file
Enter your choice (a/b/c/d/e/f): b
Enter the file name to read: Animesh
File "Animesh" contents:
Hii! This is Animesh Parab

C:\Users\Animesh\Desktop\8- file operation>
```

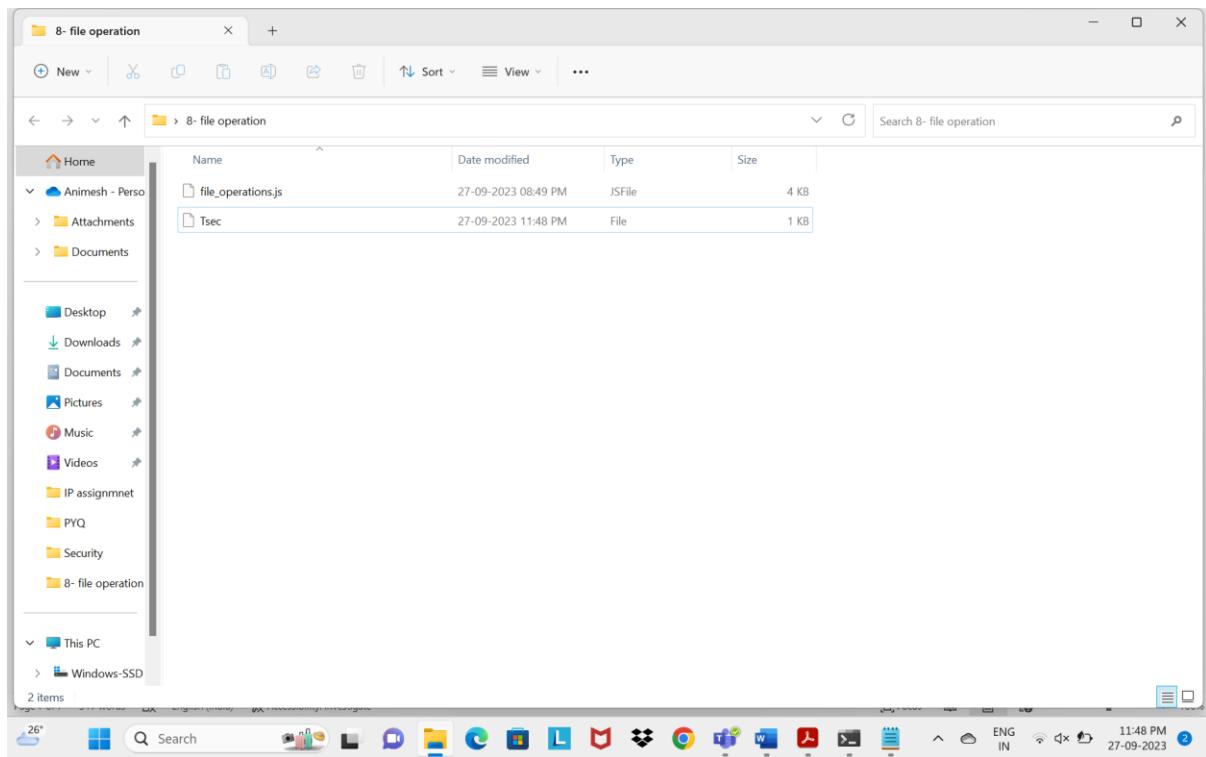


Rename file name

```
C:\Users\Animesh\Desktop\8- file operation>node file_operations.js
File Operations Menu:
a. Create a file
b. Read the data from a file
c. Write data to a file
d. Rename a file
e. Append data to a file
f. Delete a file
Enter your choice (a/b/c/d/e/f): d
Enter the current file name: Animesh
Enter the new file name: Tsec
File "Animesh" renamed to "Tsec" successfully.
```

```
C:\Users\Animesh\Desktop\8- file operation>
```

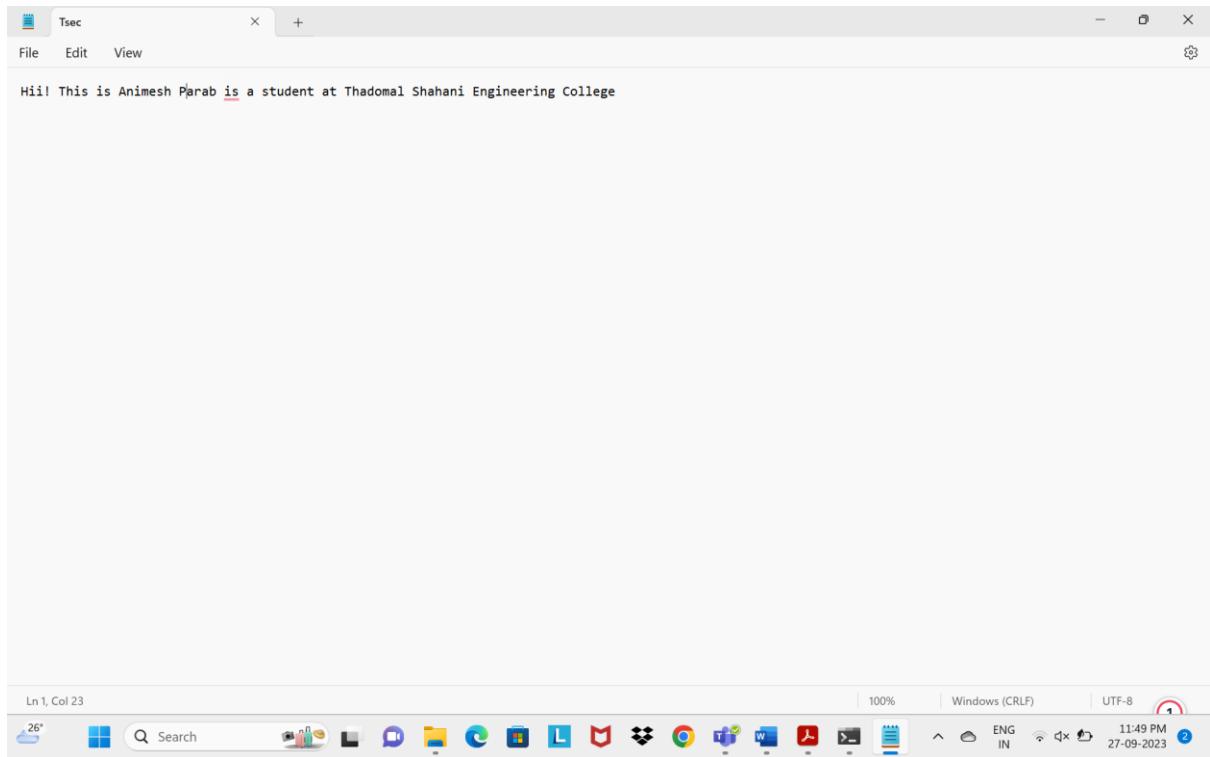




Append Data to file

```
C:\Users\Animesh\Desktop\8- file operation>node file_operations.js
File Operations Menu:
a. Create a file
b. Read the data from a file
c. Write data to a file
d. Rename a file
e. Append data to a file
f. Delete a file
Enter your choice (a/b/c/d/e/f): e
Enter the file name to append to: TSEC
Enter the content to append: is a student at Thadomal Shahani Engineering College
Content appended to "TSEC" successfully.

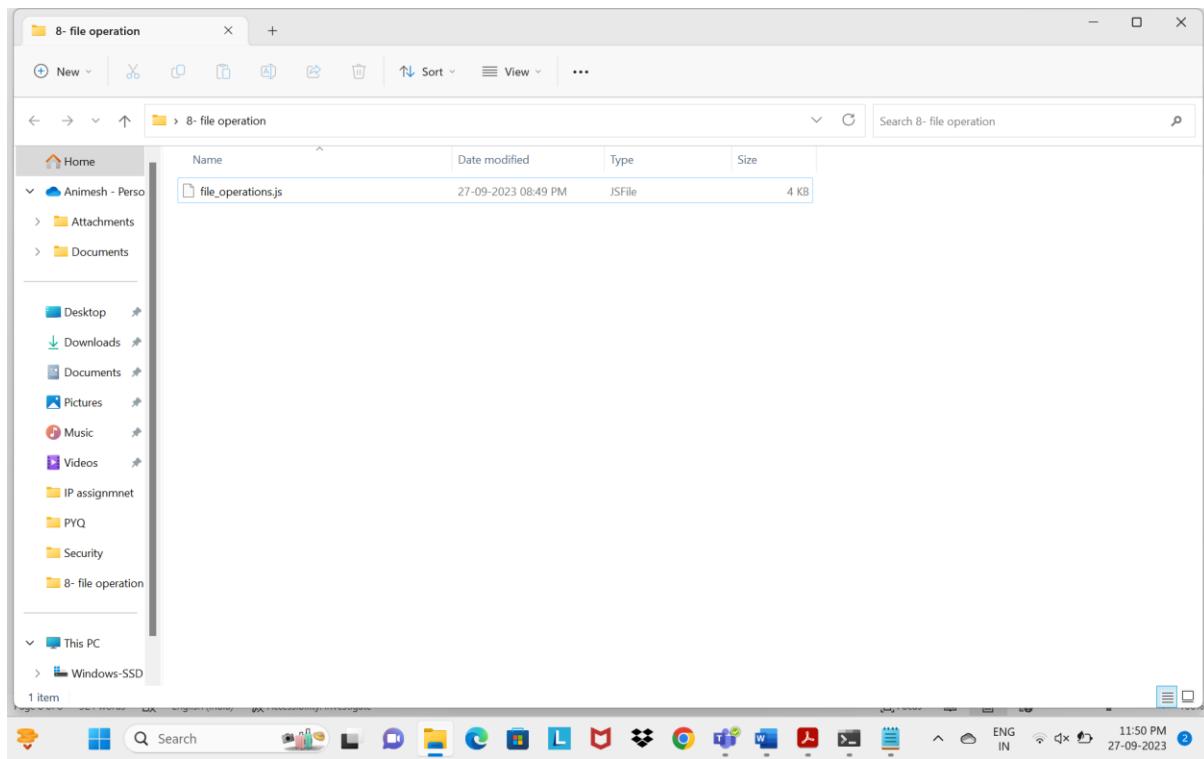
C:\Users\Animesh\Desktop\8- file operation>
```



For Delete a file

```
C:\Users\Animesh\Desktop\8- file operation>node file_operations.js
File Operations Menu:
a. Create a file
b. Read the data from a file
c. Write data to a file
d. Rename a file
e. Append data to a file
f. Delete a file
Enter your choice (a/b/c/d/e/f): f
Enter the file name to delete: Tsec
File "Tsec" deleted successfully.

C:\Users\Animesh\Desktop\8- file operation>
```



Conclusion:- In this assignment, we have learned how to create a file, how to add content to a file, how to change the file name, how to append data to existing data, how to read data for a file and how to delete a file

Assignment No. 11

Aim : Create a web application that perform CRUD operations (database connectivity).

LO Mapped : LO-6

Theory: At a high level, CRUD apps consist of three parts; the database, user interface, and APIs.

Database:

The database is where your data is stored. A database management system is used to manage the database. There are several different types of database management systems (DBMS) that can be categorized by how they store data; relational (SQL), Document (NoSQL). SQL databases consist of tables. Tables consist of records. Records consist of fields. Fields consist of data.

User Interface:

The user interface (UI) is what your users interact with. Due to the increasing popularity of applications, businesses are now prioritizing user interface design and user experience.

APIs:

Finally, the APIs are how your application informs your database of what functions to perform. These functions can be modeled in different ways but they are designed to perform four basic CRUD operations; Create, Read, Update, Delete.

The CRUD acronym is a great, memorable framework for building applications and constructing usable models. We can examine why CRUD is essential from two fronts; developers and end-users.

1. For developers, CRUD operations are essential to app development. In a similar vein, REST, a superset of CRUD, is necessary for website development.
2. For end-users, CRUD operations allow them to fill out forms, edit data, manage their admin panels and user settings.

CRUD offers many other benefits including:

- Security control
- It is more performant vs. SQL statements
- It's tried and tested
- It's a recognized pattern and recognizable by most devs
- It simplifies the process and provides an easy framework for new developers to learn

Code and Output :

Table Card

Books List



No	Title	Author	Publish Year	Operations
1	Harry Potter	J.k Rowling	1999	
2	Atomic Habit	James Clear	2018	

Table Card

Books List



6532190d5f992f33138cd913	1999
Harry Potter	
J.k Rowling	

653219465f992f33138cd918	2018
Atomic Habit	
James Clear	



Edit Book

Title

Author

Publish Year



Create Book

Title

Author

Publish Year



Delete Book

Are You Sure You want to delete this book?

Yes, Delete it



Show Book

Id 6532190d5f992f33138cd913

Title Harry Potter

Author J.K Rowling

Publish Year 1999

Create Time Fri Oct 20 2023 11:37:09 GMT+0530 (India Standard Time)

Last Update Time Fri Oct 20 2023 11:37:09 GMT+0530 (India Standard Time)

Conclusion : We successfully implemented a web application with CRUD that is database connectivity.

IP WRITTEN ASSIGNMENT 1

1. Compare XML and JSON.

Co 1

ANS-Parameter	JSON	XML
1.Full form	It stands for JavaScript Object Notation.	It stands for Extensible Markup Language.
2.Format	It is used for representing objects. It has a key-value pair format.	It is a markup language. It uses tags to represent data items.
3.Array support	It supports arrays.	It does not support arrays.
4.Encoding	It supports only UTF-8 encoding.	It supports various other encodings.
5.Comments	It does not support comments.	It supports comments.
6.Readability	It is more compact and is easier for humans to read and write.	It is comparatively more difficult to read and write.
7.Namespace support	It does not support namespaces.	It supports namespaces.

2. Explain different types of arrow functions.

Co 2

ANS-

- 1) Arrow functions, also called lambda functions are anonymous functions used in programming.
- 2) There are 3 parts in an arrow function:
 - i. Parameters
 - ii. Fat arrow notation (`=>`)
 - iii. Statements
- 3) Arrow functions remove the need to type out the 'function' keyword every time we create a function.
- 4) There are three main types of arrow functions:
 - i. Arrow function with no argument-

ii. Arrow function with parameters-

If a function that does not take any argument, then we should use empty parentheses.

For example-

```
let greet = () => console.log("Hello");
greet();
```

iii. Arrow Function with Multiple Statements-

If a function takes only one argument, then we can ignore the parentheses.

For example-

```
let cube = x => x*x*x;
greet(10);
```

However, if there are no parameters or more than one parameter, you must enclose the parameter list in parentheses.

If your arrow function needs to include multiple statements, you'll need to use curly braces and an explicit `return` statement.

For example-

```
const largerNumber = (a, b) => {
if (a > b) {
return a;
} else {
return b;
}
};
```

3. What is DNS? Explain working of DNS.

Co 1

ANS-

1) A Domain Name System (DNS) turns domain names into IP addresses, which allows browsers to access websites and other internet resources. DNS thus allows us to access an online resource without needing to know its IP address.

2) Web browsing and most other internet activities rely on DNS to quickly provide the information necessary to connect users to remote hosts. DNS mapping is distributed throughout the internet in a hierarchy of authority. They also typically run DNS servers to manage the mapping of those names to those addresses. Most Uniform Resource Locators (URLs) are built around the domain name of the web server that takes client requests.

3) The steps involved in the working of DNS are:

- The user enters a web address or domain name into a browser.

- The browser sends a message, called a recursive DNS query, to the network to find out which IP or network address the domain corresponds to.

- The query goes to a recursive DNS server, which is also called a recursive resolver, and is usually managed by the internet service provider (ISP). If the recursive resolver has the address, it will return the address to the user, and the webpage will load.

- If the recursive DNS server does not have an answer, it will query a series of other servers in the following order: DNS root name servers, top-level domain (TLD) name servers and authoritative name servers.

- v. The three server types work together and continue redirecting until they retrieve a DNS record that contains the queried IP address. It sends this information to the recursive DNS server, and the webpage the user is looking for loads. DNS root name servers and TLD servers primarily redirect queries and rarely provide the resolution themselves.
- vi. The recursive server stores, or caches, the A record for the domain name, which contains the IP address. The next time it receives a request for that domain name, it can respond directly to the user instead of querying other servers.
- vii. If the query reaches the authoritative server and it cannot find the information, it returns an error message.

4. Explain promises in ES6.

Co 2

ANS-

- 1) In JavaScript, Promises are a programming construct introduced in ECMAScript 6 (ES6) to help manage asynchronous operations in a more organized and readable manner. Asynchronous operations are tasks that may take some time to complete, such as fetching data from a server or reading a file, and they don't block the execution of other code while waiting for their completion. Prior to Promises, managing asynchronous operations often involved deeply nested callback functions, leading to a pattern known as "callback hell".
- 2) Promises provide a cleaner way to handle asynchronous code by abstracting the process of handling success and failure outcomes. Promises represent a value that may not be available yet, but will be resolved or rejected at some point in the future. They have three states:
 - i. Pending: The initial state. The Promise is neither fulfilled nor rejected; it's still in progress.
 - ii. Fulfilled: The asynchronous operation has completed successfully, and the Promise is fulfilled. It transitions to this state with a result value.
 - iii. Rejected: The asynchronous operation has encountered an error, and the Promise is rejected. It transitions to this state with a reason for the rejection.
- 3) Here's how you can use Promises in JavaScript, specifically ES6:

```
// Creating a new Promise
const myPromise = new Promise((resolve, reject) => {
  // Simulate an asynchronous operation
  setTimeout(() => {
    const randomNumber = Math.random();

    if (randomNumber > 0.5) {
      resolve(randomNumber); // Resolve the Promise with a value
    } else {
      reject("Value is too low"); // Reject the Promise with a reason
    }
  }, 1000); // Simulating a delay of 1 second
});
// Using the Promise
myPromise
then(result => {
  console.log("Fulfilled with result:", result);
})
```

```
})
.catch(error => {
  console.error("Rejected with error:", error);
});
```

In this example, `myPromise` represents an asynchronous operation that simulates generating a random number. Depending on the value of the random number, the Promise either resolves or rejects. The ` `.then()` ` method is used to handle the fulfilment case, and the ` `.catch()` ` method is used to handle the rejection case.

4) Promises also support chaining, allowing you to sequence asynchronous operations more effectively:

```
function fetchUserData() {
  return fetch('https://api.example.com/user')
    .then(response => response.json())
    .then(userData => {
      console.log('User data:', userData);
    })
    .catch(error => {
      console.error('Error fetching user data:', error);
    });
}
```

In this example, the `fetchUserData` function returns a Promise that fetches user data from an API. The ` `.then()` ` methods are chained to process the response and handle potential errors.

Assignment No. 2

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

Ans. Refs in React are a way to access and interact with the DOM (Document Object Model) or to reference a React component instance directly. They provide a means to escape the usual data flow and can be used sparingly for specific tasks that require direct interaction with the DOM or other React components. Here's an overview of when to use refs and when not to use them:

When to Use Refs:

Accessing DOM Elements: Refs are commonly used to access and manipulate DOM elements directly. For example, you might use refs to focus an input field, scroll to a specific part of a page, or trigger animations associated with a particular element.

Integrating with Third-Party Libraries: When working with third-party libraries, especially those that rely on direct DOM manipulation, refs can be useful. They allow you to bridge React with these external libraries, accessing and controlling their elements.

Managing Media Playback: Refs can be employed to manage media elements like audio or video. You can play, pause, or manipulate the media directly using the ref.

Custom Component Communication: In some advanced scenarios, you might need to reference child components to communicate or trigger specific actions. Refs can be used for this purpose.

When Not to Use Refs

State Management and Data Flow: Refs should not be used for managing application state or data flow. Instead, rely on React's built-in state management (such as `useState`) and props to pass data and trigger updates between components. Proper data flow helps maintain the predictability of your application.

Conditional Rendering: Avoid using refs to conditionally render or hide components. React's declarative nature allows you to handle conditional rendering through state or props, making your code more understandable.

Functional Components and Hooks: While you can use refs with class components as shown above, in functional components, it's recommended to use the `useRef` hook to achieve similar functionality. Be cautious with the usage of `useRef` in functional components, and consider if the same functionality can be achieved with state and props, which align better with React's functional paradigm.

Layout and Styling: Refs should not be used to directly modify the layout or styling of elements. React is designed to handle rendering efficiently, and altering styles directly through refs can lead to unexpected results. Instead, use React state and CSS styles.

2. Write short note on:

a) NPM (Node Package Manager):

NPM is the default package manager for Node.js, a popular runtime environment for running JavaScript on servers and for building web applications. NPM serves as a central repository for thousands of open-source packages and libraries that can be easily installed and managed in your Node.js projects. Here are some key points about NPM:

Package Management: NPM is primarily used for managing and installing Node.js packages. Developers can use it to install packages, manage dependencies, and update libraries in their Node.js applications.

Command-Line Tool: NPM provides a command-line interface for interacting with packages. You can use commands like `npm install`, `npm init`, and `npm update` to handle various aspects of package management.

Package.json: Each Node.js project typically includes a `package.json` file, which contains metadata about the project, as well as a list of its dependencies. NPM uses this file to manage project dependencies and scripts.

Version Control: NPM allows you to specify version ranges for your dependencies to ensure compatibility and stability within your project. It also provides semantic versioning (SemVer) for package version management.

Global and Local Installation: NPM can install packages globally (available across the system) or locally (within a specific project directory) depending on your needs.

Scripts: NPM lets you define custom scripts in your `package.json` file. These scripts can be executed using the `npm run` command and are often used for tasks like building, testing, and deploying applications.

b) REPL (Read-Eval-Print Loop):

A REPL is an interactive programming environment commonly used for interpreted programming languages like JavaScript. It provides a simple way to interactively execute code, see immediate results, and experiment with language features. Here's a brief overview of REPL:

Interactive Environment: The REPL is a command-line interface or tool where you can enter code, which is immediately executed, and the results are displayed. It's a great way to quickly test and prototype code.

Debugging and Testing: Developers often use a REPL for debugging code snippets, testing small code fragments, and exploring language features. It's particularly useful for learning and experimenting with a programming language.

Immediate Feedback: The primary advantage of a REPL is the immediate feedback it provides. As you enter code, it's evaluated and the results are displayed, allowing you to make adjustments and see the impact in real-time.

Available for Many Languages: While commonly associated with languages like Python and JavaScript, REPLs exist for a wide range of programming languages, each tailored to the specific language's features and syntax.

3. Explain Routing in ExpressJS along with an example

Routing in Express.js is a fundamental concept that allows you to define how your application responds to different HTTP requests. It helps you map specific URLs (or routes) to particular functions or handlers, enabling you to create a structured and organized web application with various endpoints. Express.js provides a flexible and easy-to-use routing system.

Example of Routing in Express.js:

```
const express = require('express');
const app = express();
const port = 3000;
// Route for the homepage
app.get('/', (req, res) => {
res.send('Hello, World!');
});
// Route for echoing a user-provided message
app.get('/echo/:message', (req, res) => {
const message = req.params.message;
res.send(`You said: ${message}`);
});
app.listen(port, () => {
console.log(`Server is running on port ${port}`);
});
```

Output :

