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

Teacher In- Charge 20/10/23

Head of the Department

Date 20/10/23

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

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 Assignment 1		3/8/23	
3]	Develop a web page using bootstrap framework Grid, forms, Buttons, Navbar		3/8/23	Saurabh 20/10/23
4]	a. write a program on conditional statement, loops		10/8/23	
	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		12/8/23	Saurabh 20/10/23
	b. write a Js program to implement promise (call back), fetch, Asynchronous Javascript		12/8/23	
6]	a. write a program to implement props & State		24/8/23	
	b. write a program to implement forms and events		24/8/23	Saurabh 20/10/23
7]	a. write a program to implement Reacts Js router and animation.		31/8/23	
	b. write a program to implement concept of React Hooks		4/9/23	

THADOMAL SHAHANI TSEC ENGINEERING SOLLEGE

CONTENTS

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

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>

My first paragraph.

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. : Represents paragraphs of text.

6. <a>: Creates hyperlinks to other web pages or resources.

7. <img>: Embeds images into the page.

8. and <ol>: Establish unordered and ordered lists, respectively.

9. : Represents individual items within lists.

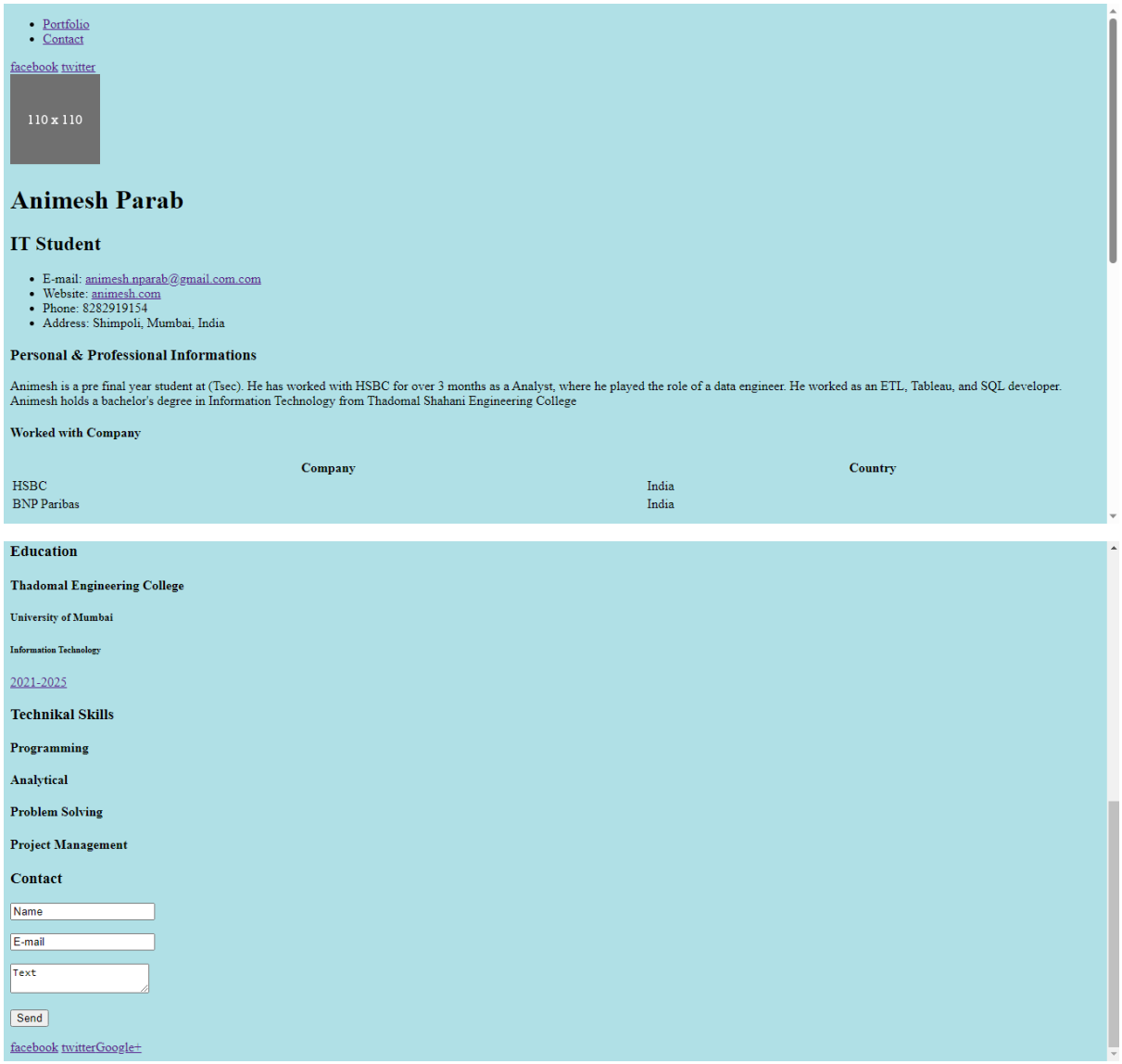
10. and : Generic containers for grouping and styling elements.

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:



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 poperties
- 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:

Profile

Home Skills Contact

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 ofa data engineer.

He worked on a software like data visualization.

He hold a degree in Information Technology.

Portfolio

Home Skills Contact

Skills

Languages

- C++
- Java
- Python
- Javascript

Frameworks

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

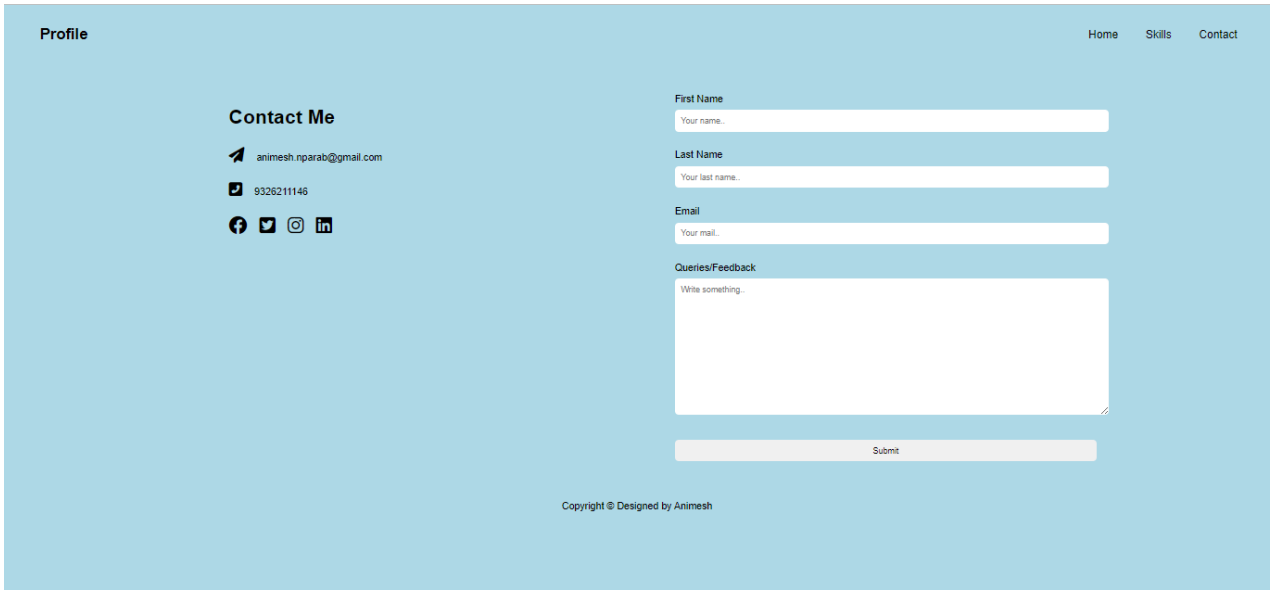
Projects

Student Managment System

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

Hospital Managment System

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



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-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
<title>Porfile</title>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
<div class="container-fluid" >
<a class="navbar-brand" href="#">Porfile</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.html">Contact Us</a>
</li>
</ul>
</div>
```

index - Notepad

File Edit Format View Help

```
</div>
</nav>
<div class="container" style="margin-left:22% ; margin-top:8%; marginbottom:6em;" >
<div class="coloum">
<div class="row-1">
<h2>Hello,</h2>
<h1>Hii<strong> <font color="Red">Animesh Parab</font></strong></h1>
<h2>Software Developer</h2>
</div>
<div class="row-2">
<img src="Profile.jpg" alt="Profile-image" width="200px" height="200px"
border-radius="50">
</div>
</div>
</div>
<div class="container" style="margin-top: 2em; margin-bottom: 4em;
width:70%;">
<h2 style="padding-bottom: 1.5rem; text-align:center;">About me</h2>
<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 ofa data engineer.
He worked on a software like data visualization.
He hold a degree in Information Technology.</p>
</div>
<!-- -->
<!-- -->
<div class="container">
<h2 style="margin-bottom: 1em;">Skills</h2>
<div class="accordion accordion-flush" id="accordionFlushExample">
<div class="accordion-item">
<h2 class="accordion-header" id="flush-headingOne">
<button class="accordion-button collapsed" type="button" data-bstoggle="collapse" data-bs-target="#flush-collapseOne" aria-expanded="false"
aria-controls="flush-collapseOne">
Languages
</button>
</h2>
<div id="flush-collapseOne" class="accordion-collapse collapse" arialabelledby="flush-headingOne" data-bs-parent="#accordionFlushExample">
```

index - Notepad

File Edit Format View Help

```
<div id="flush-collapseOne" class="accordion-collapse collapse" arialabelledby="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>
<div class="accordion-item">
<h2 class="accordion-header" id="flush-headingTwo">
<button class="accordion-button collapsed" type="button" data-bstoggle="collapse" data-bs-target="#flush-collapseTwo" aria-expanded="false"
aria-controls="flush-collapseTwo">
Frameworks
</button>
</h2>
<div id="flush-collapseTwo" class="accordion-collapse collapse" arialabelledby="flush-headingTwo" data-bs-parent="#accordionFlushExample">
<div class="accordion-body">
<ol>
<li>HTML</li>
<li>CSS</li>
<li>Django</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-bstoggle="collapse" data-bs-target="#flush-collapseThree" aria-expanded="false"
aria-controls="flush-collapseThree">
Tools
```

index - Notepad

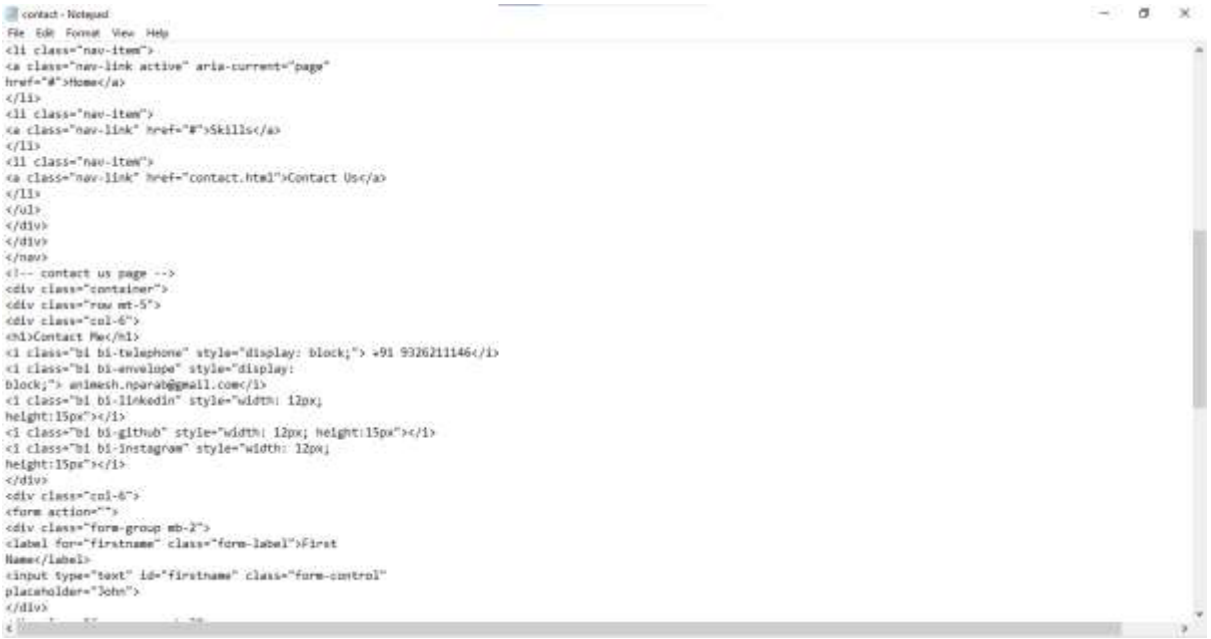
File Edit Format View Help

```
Tools
</button>
</h2>
<div id="flush-collapseThree" class="accordion-collapse collapse" arialabelledby="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>
<!-- -->
<div class="container" style=" margin-top: 5em; ">
<h2>Projects</h2>
<br>
<div class="row">
<div class="col-sm-4">
<div class="card" style="width: 20rem; ">
<img src="image2.jpg" class="card-img-top" alt="..." width="250px"
height="250px">
<div class="card-body">
<h5 class="card-title">Hospital Mangement System</h5>
<p class="card-text">It is a syste used by hospital for managing their hospitl inefficient way.</p>
<a href="#" class="btn btn-primary">visit website</a>
</div>
</div>
</div>
<div class="col-sm-4">
<div class="card" style="width: 24rem;">
<img src="image1.jpg" class="card-img-top" alt="..." width="250px"
height="250px">
<div class="card-body">
<h5 class="card-title">Student manageent system</h5>
```

Ln 107, Col 1 100% Windows (CRLF) UTF-8

```
<div class="card-body">
<h5 class="card-title">Student manageent system</h5>
<p class="card-text">It is system used by education institude for managing their institude</p>
<a href="#" class="btn btn-primary">visit website</a>
</div>
</div>
</div>
</div>
</div>
<footer style="margin-top: 4em;">
<center>
<p>Copyright &copy; Designed by Animesh Parab</p>
</center>
</footer>
<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>
```

```
contact - 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-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
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>
<link rel="stylesheet" href="dist/css/bootstrap.min.css">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/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="#">Portfolio</a>
<button class="navbar-toggler" type="button" data-bstoggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" ariaexpanded="false" aria-label="Toggle navi
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav">
<ul class="navbar-nav">
<li class="nav-item">
```

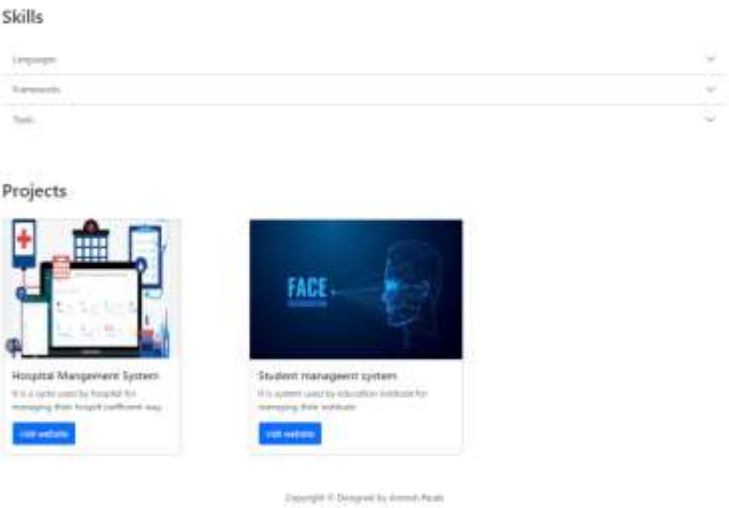


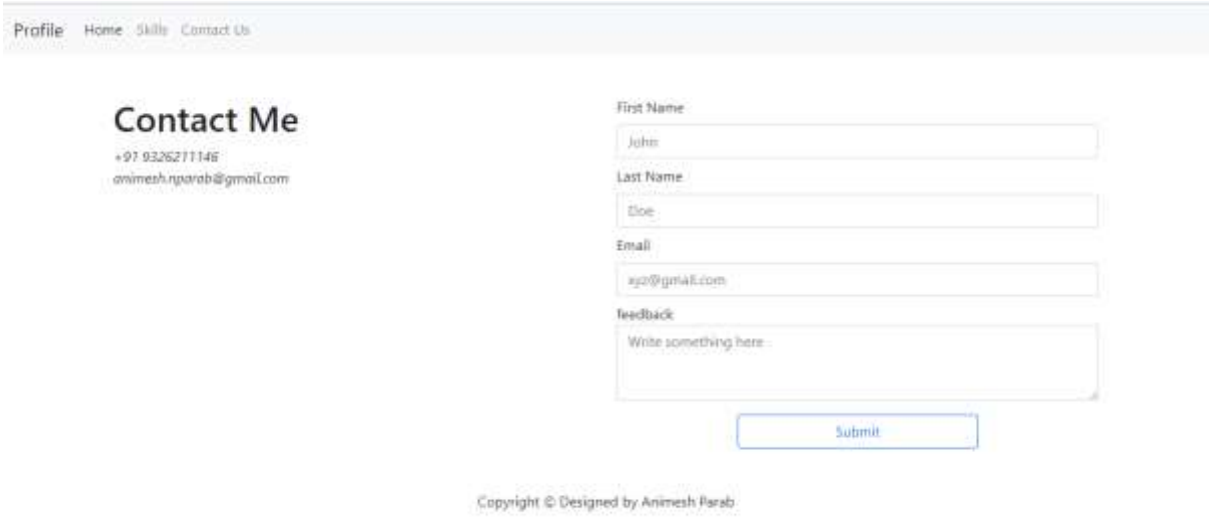
```
contact - Notepad
File  Edit  Format  View  Help
<li class="nav-item">
<a class="nav-link active" aria-current="page"
href="#">Home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Skills</a>
</li>
<li class="nav-item">
<a class="nav-link" href="contact.html">Contact Us</a>
</li>
</ul>
</div>
</div>
</nav>
<!-- contact us page -->
<div class="container">
<div class="row mt-5">
<div class="col-6">
<h1>Contact Me</h1>
<i class="bi bi-telephone" style="display: block;"> +91 9326211146</i>
<i class="bi bi-envelope" style="display:
block;"> animesh.nparab@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>
```

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="formlabel">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>
</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>
```





Conclusion:

Bootstrap is a powerful tool that simplifies web development by providing a ready-touse set of components and styles. It empowers developers to create professionallooking 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

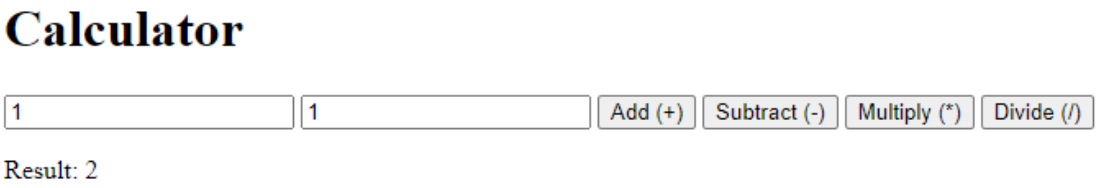
Aim: WAP in JS to study conditional Statements, Loops and Functions

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>
```



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.

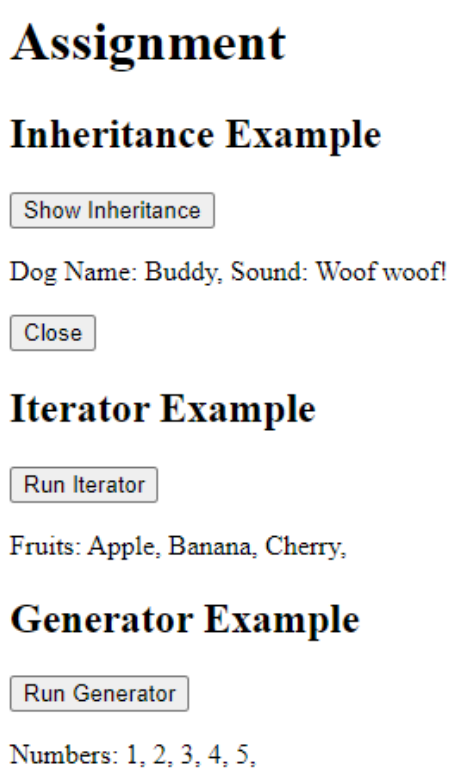
3. Generators-

   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 :

```html
<!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('Buddy');

    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';
    }



    // 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



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>
```

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.

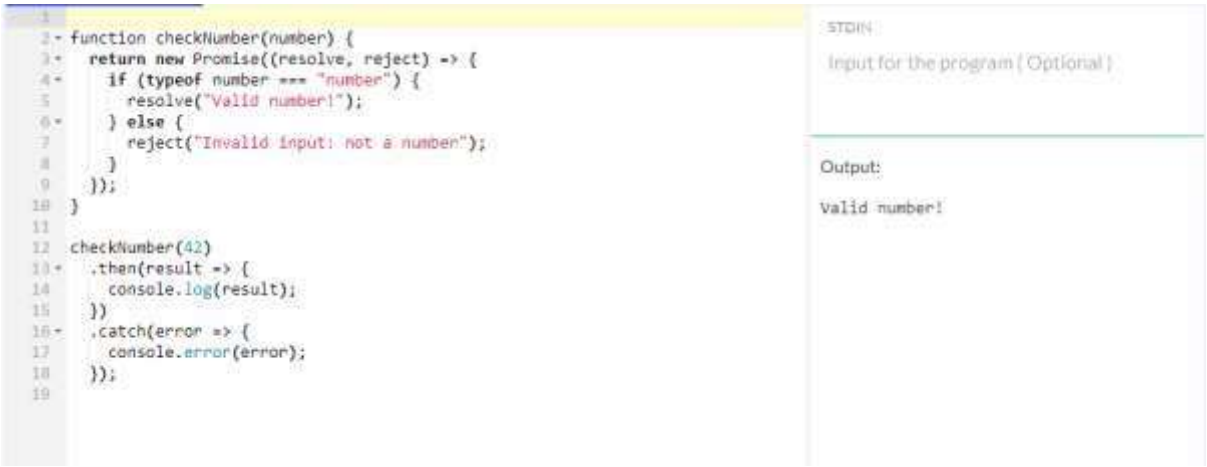
Promises provide a chainable syntax for handling asynchronous operations. The **.then()** method is used to handle fulfilment, and the **.catch()** method is used to handle rejection. Additionally, you can use the **.finally()** method to attach a callback that runs regardless of whether the promise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



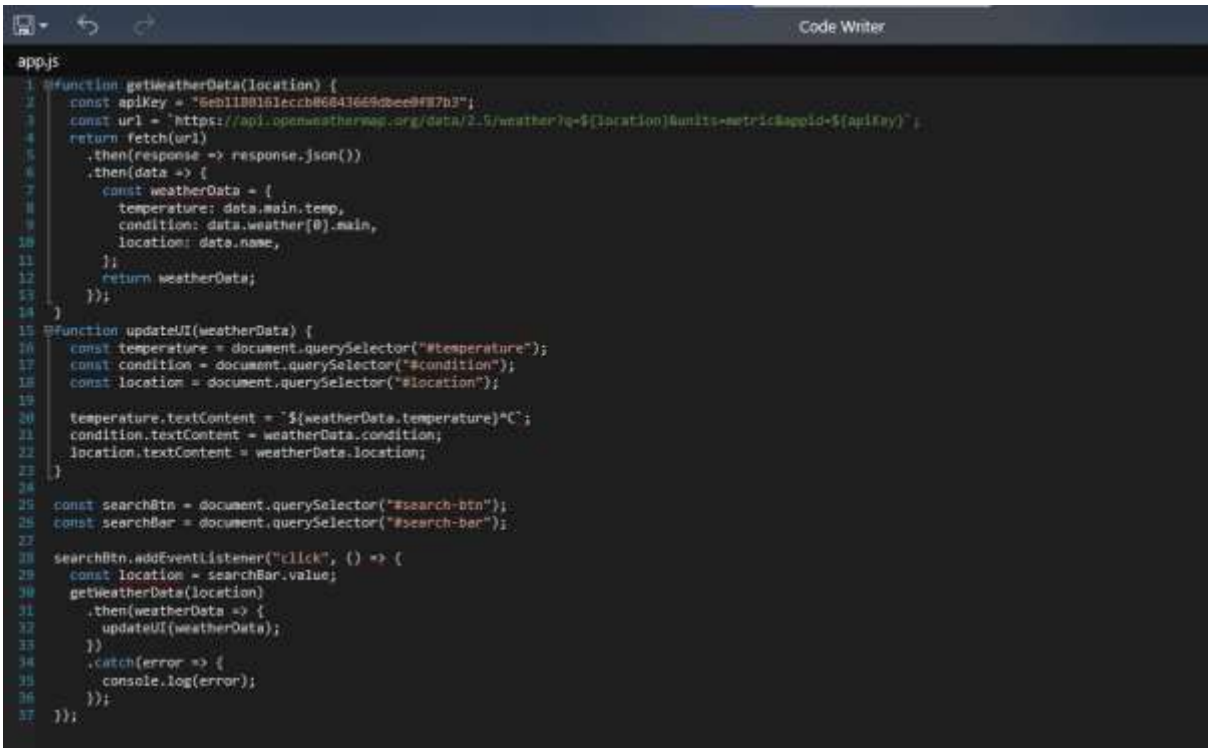
```
function checkNumber(number) {
  return new Promise((resolve, reject) => {
    if (typeof number === "number") {
      resolve("Valid number!");
    } else {
      reject("Invalid input: not a number");
    }
  });
}

checkNumber(42)
  .then(result => {
    console.log(result);
  })
  .catch(error => {
    console.error(error);
  });
```

STDIN

Input for the program (Optional)

Output:

Valid number!

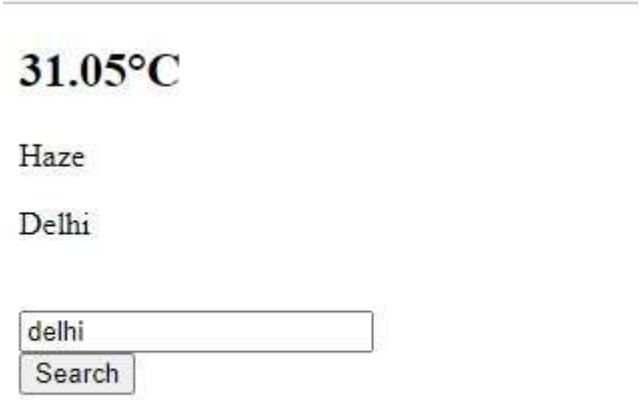


Code Writer

app.js

```
function getWeatherData(location) {
  const apiKey = "6eb11801d1eccb0684366d0bee0f87b3";
  const url = `https://api.openweathermap.org/data/2.5/weather?q=${location}&units=metric&appid=${apiKey}`;
  return fetch(url)
    .then(response => response.json())
    .then(data => {
      const weatherData = {
        temperature: data.main.temp,
        condition: data.weather[0].main,
        location: data.name,
      };
      return weatherData;
    });
}
function updateUI(weatherData) {
  const temperature = document.querySelector("#temperature");
  const condition = document.querySelector("#condition");
  const location = document.querySelector("#location");

  temperature.textContent = `${weatherData.temperature}°C`;
  condition.textContent = weatherData.condition;
  location.textContent = weatherData.location;
}

const searchBtn = document.querySelector("#search-btn");
const searchBar = document.querySelector("#search-bar");

searchBtn.addEventListener("click", () => {
  const location = searchBar.value;
  getWeatherData(location)
    .then(weatherData => {
      updateUI(weatherData);
    })
    .catch(error => {
      console.log(error);
    });
});
```

Out put :



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

  export default DateTimeDisplay;
```

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 :-

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.

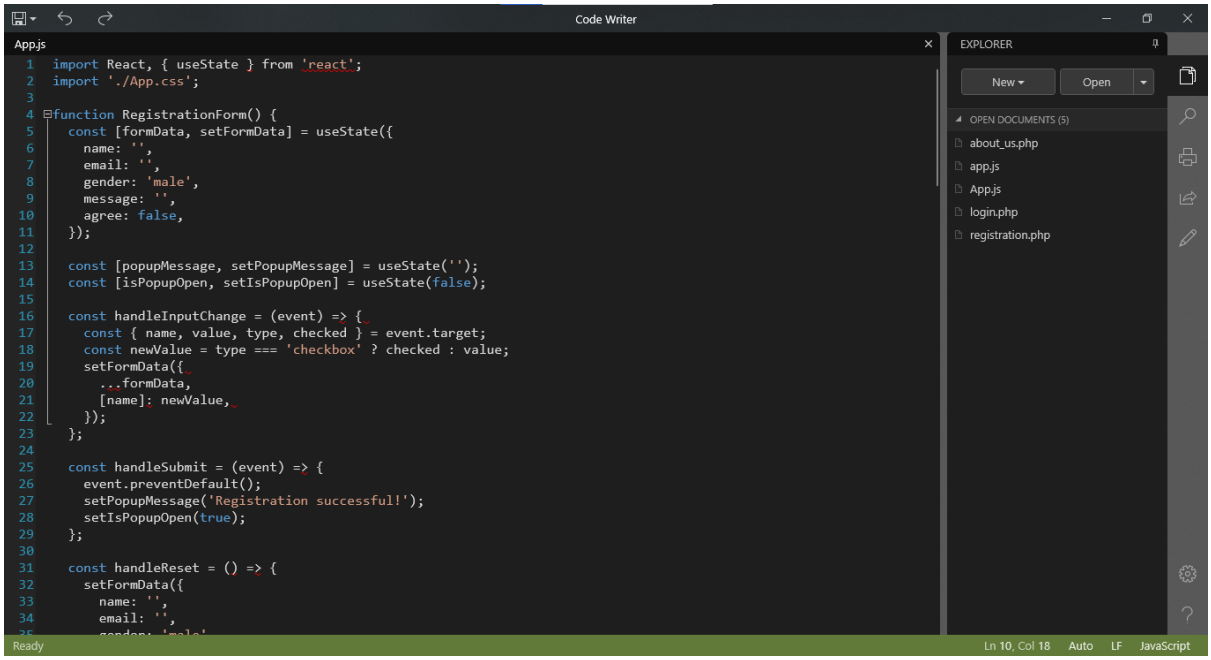
6. CSS Styling : CSS is used to style the form elements and the popup/modal. Proper styling enhances the user experience by making the form visually appealing.

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.

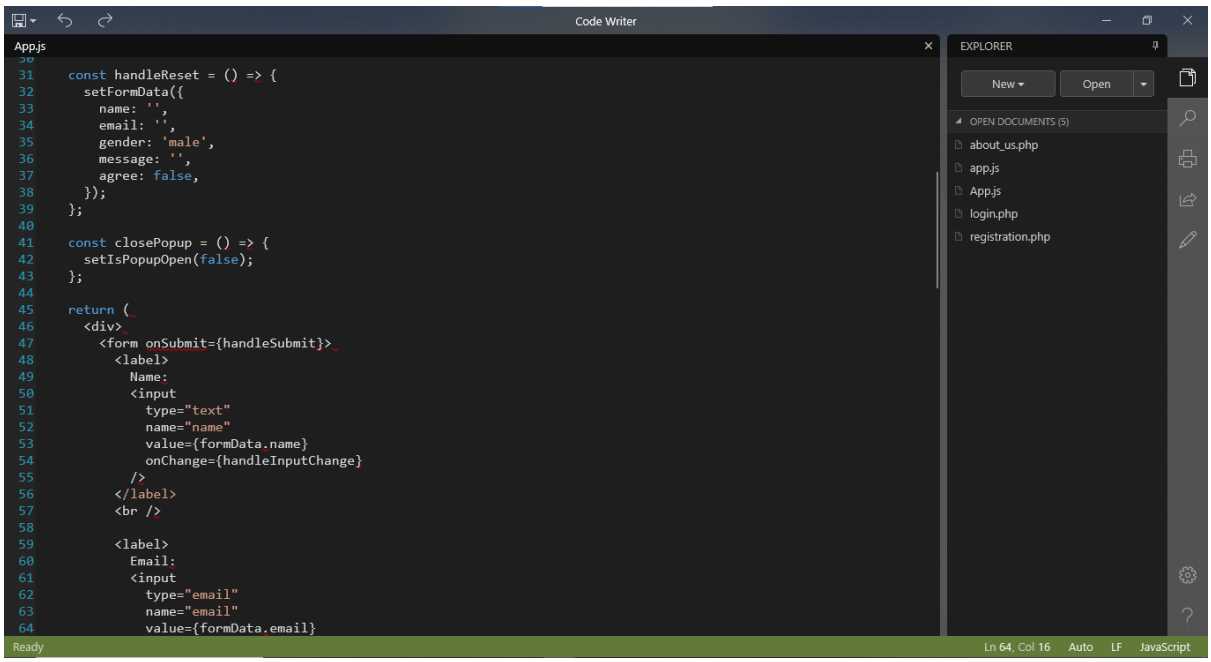
8. Popup/Modal : The popup/modal is a UI element that appears on top of the main content, usually to provide additional information or feedback to the user. It's controlled by state and can be shown or hidden based on user actions or conditions.

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.

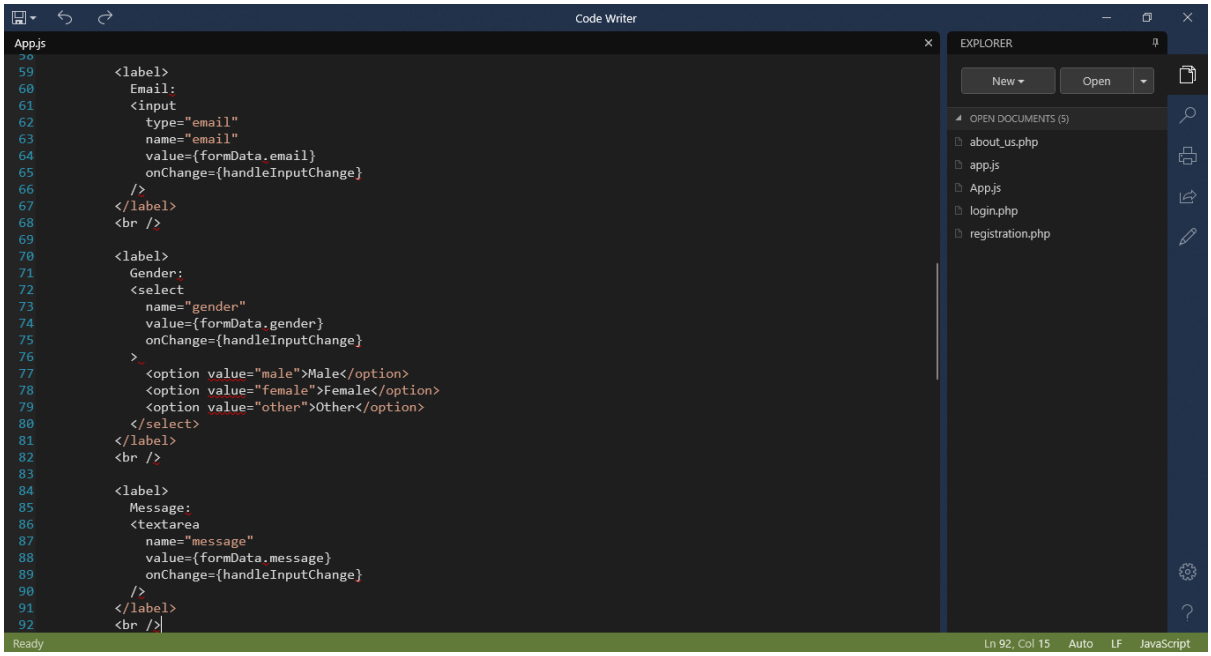
Code :-




```
import React, { useState } from 'react';
import './App.css';

function RegistrationForm() {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    gender: 'male',
    message: '',
    agree: false,
  });

  const [popupMessage, setPopupMessage] = useState('');
  const [isPopupOpen, setIsPopupOpen] = useState(false);

  const handleInputChange = (event) => {
    const { name, value, type, checked } = event.target;
    const newValue = type === 'checkbox' ? checked : value;
    setFormData({
      ...formData,
      [name]: newValue,
    });
  };

  const handleSubmit = (event) => {
    event.preventDefault();
    setPopupMessage('Registration successful!');
    setIsPopupOpen(true);
  };

  const handleReset = () => {
    setFormData({
      name: '',
      email: '',
```



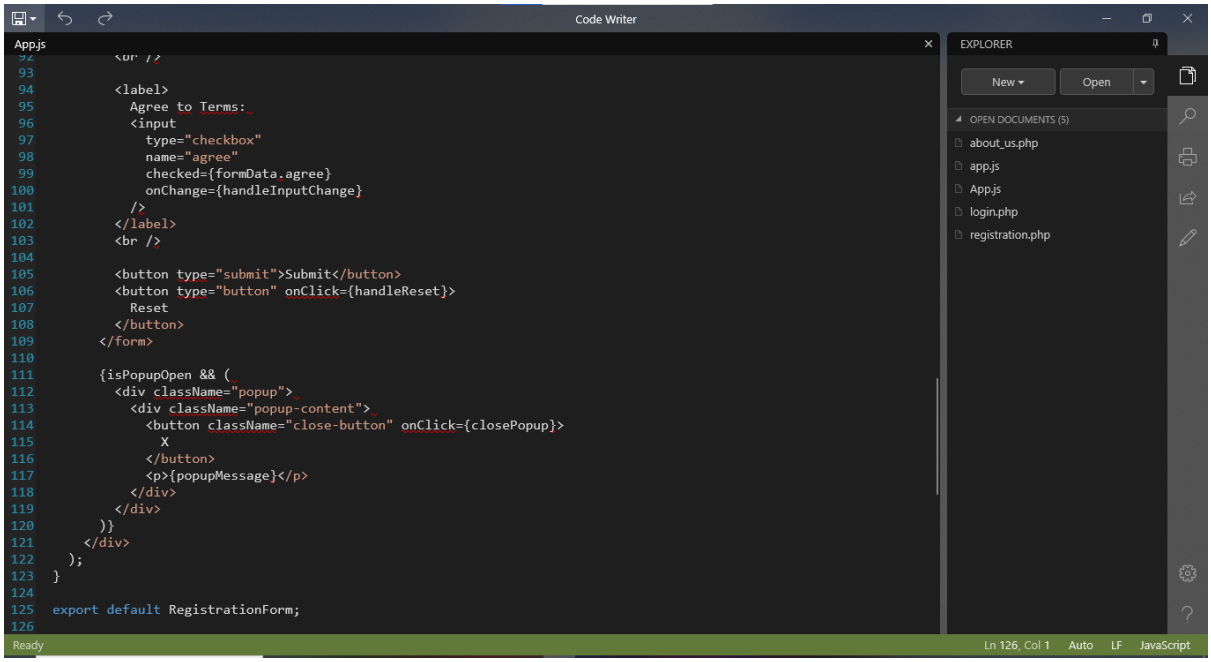

```
  const handleReset = () => {
    setFormData({
      name: '',
      email: '',
      gender: 'male',
      message: '',
      agree: false,
    });
  };

  const closePopup = () => {
    setIsPopupOpen(false);
  };

  return (
    <div>
      <form onSubmit={handleSubmit}>
        <label>
          Name:
          <input
            type="text"
            name="name"
            value={formData.name}
            onChange={handleInputChange}
          />
        </label>
        <br />

        <label>
          Email:
          <input
            type="email"
            name="email"
            value={formData.email}
```


Code Writer
App.js
EXPLORER
New
Open
OPEN DOCUMENTS (5)
about_us.php
app.js
App.js
login.php
registration.php
<label>
Email:
<input
type="email"
name="email"
value={formData.email}
onChange={handleInputChange}
/>
</label>

<label>
Gender:
<select
name="gender"
value={formData.gender}
onChange={handleInputChange}
>
<option value="male">Male</option>
<option value="female">Female</option>
<option value="other">Other</option>
</select>
</label>

<label>
Message:
<textarea
name="message"
value={formData.message}
onChange={handleInputChange}
/>
</label>

Ready
Ln 92, Col 15
Auto
LF
JavaScript


Code Writer
App.js
EXPLORER
New
Open
OPEN DOCUMENTS (5)
about_us.php
app.js
App.js
login.php
registration.php
<label>
Agree to Terms:
<input
type="checkbox"
name="agree"
checked={formData.agree}
onChange={handleInputChange}
/>
</label>

<button type="submit">Submit</button>
<button type="button" onClick={handleReset}>
Reset
</button>
</form>
{isPopupOpen && (

<button className="close-button" onClick={closePopup}>
X
</button>
{popupMessage}

)}

);
}
export default RegistrationForm;
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;
```

App - Notepad

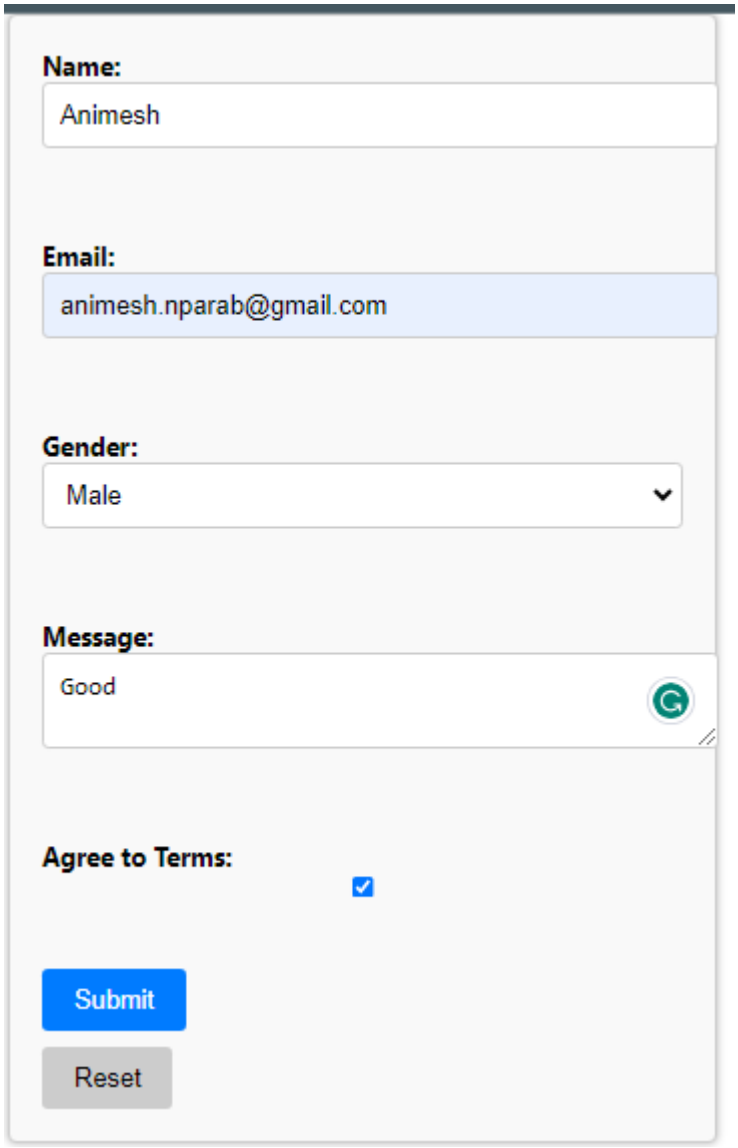
File Edit Format View Help

```
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;
```

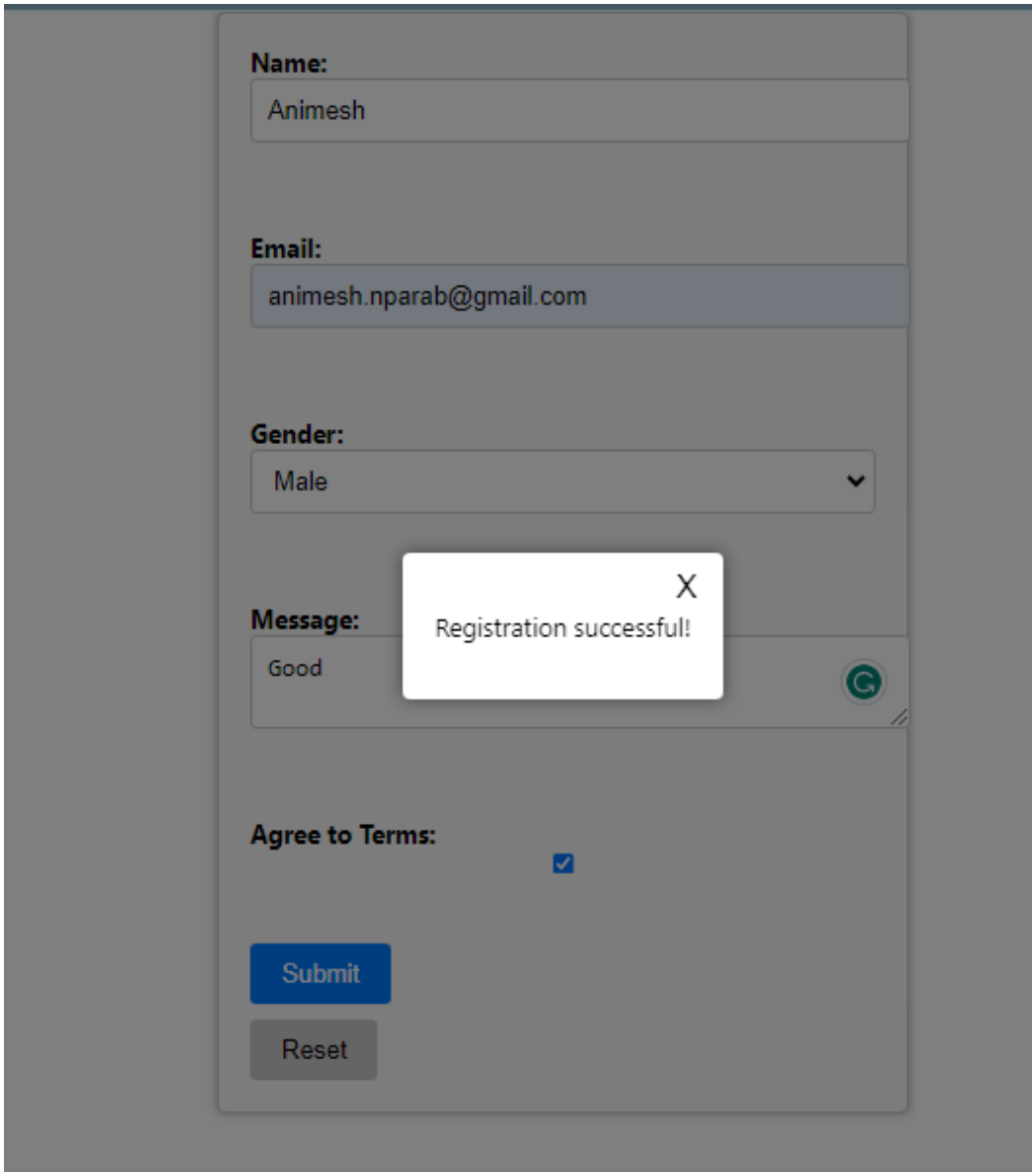
Ln 54, Col 11 | 100% | Unix (LF) | UTF-8


App - Notepad
File Edit Format View Help
}
.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;
}
Ln 100, Col 1
100%
Unix (LF)
UTF-8

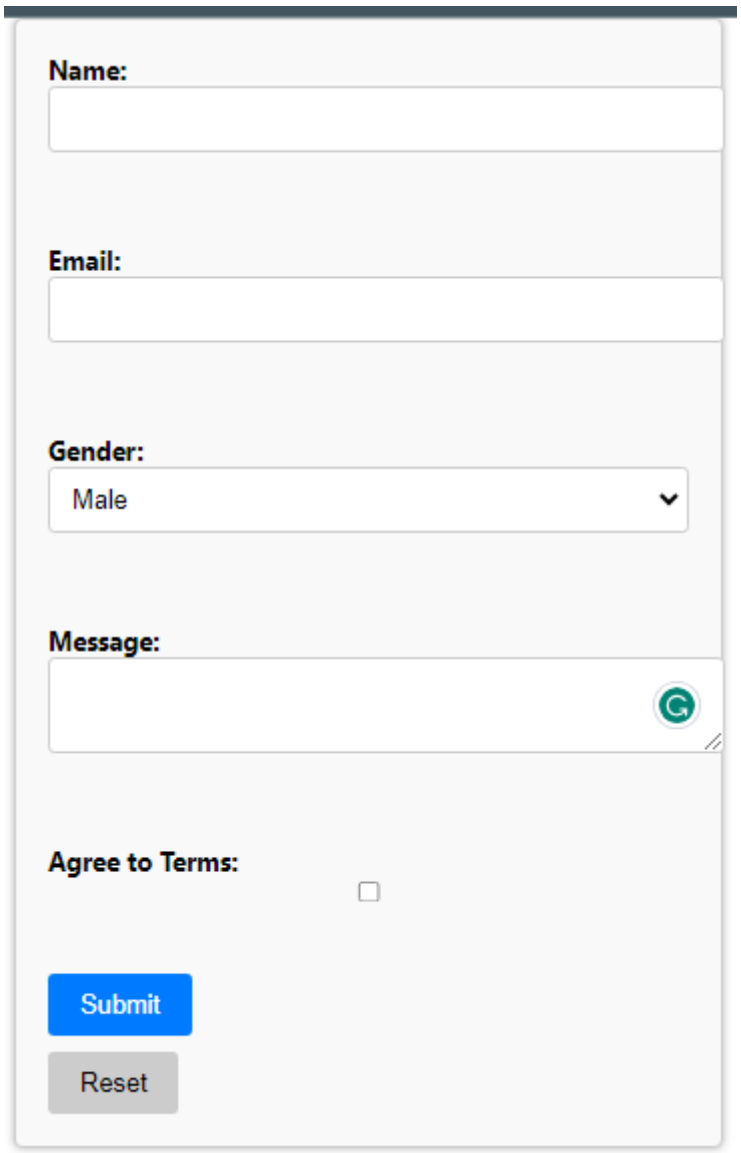
Code output :-


Name:
Animesh
Email:
animesh.nparab@gmail.com
Gender:
Male
Message:
Good
Agree to Terms:
Submit
Reset

After clicking Submit button



After clicking 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:-

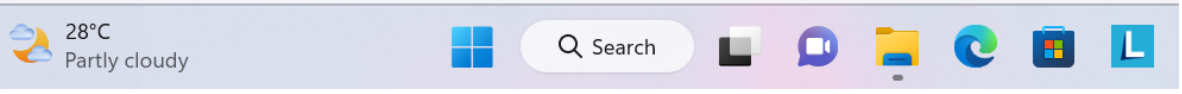
App.js

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



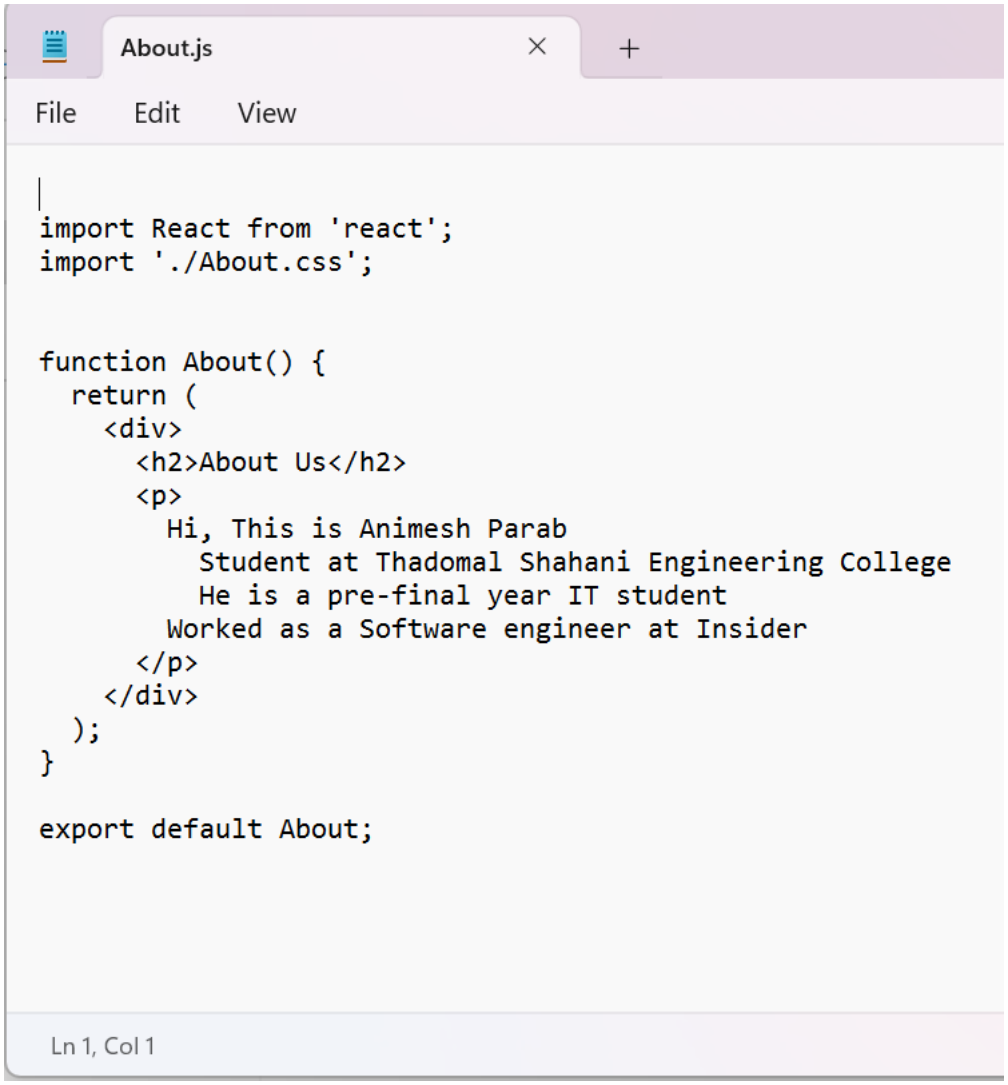
Home.js

App.js Home.js

File Edit View

```
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

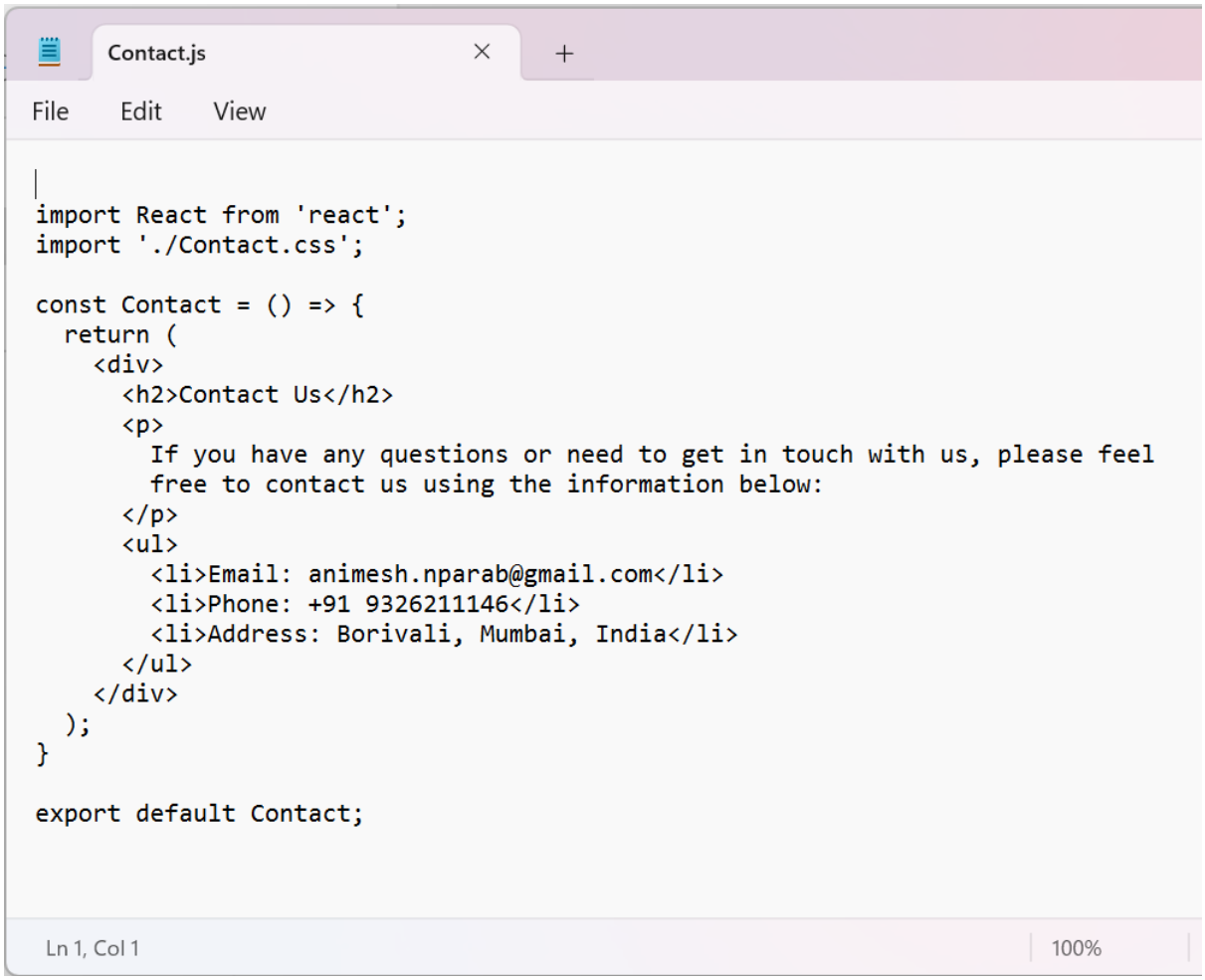

About.js
File
Edit
View
import React from 'react';
import './About.css';
function About() {
return (

<h2>About Us</h2>

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

);
}
export default About;
Ln 1, Col 1

Contact.js


Contact.js
File
Edit
View
import React from 'react';
import './Contact.css';
const Contact = () => {
return (

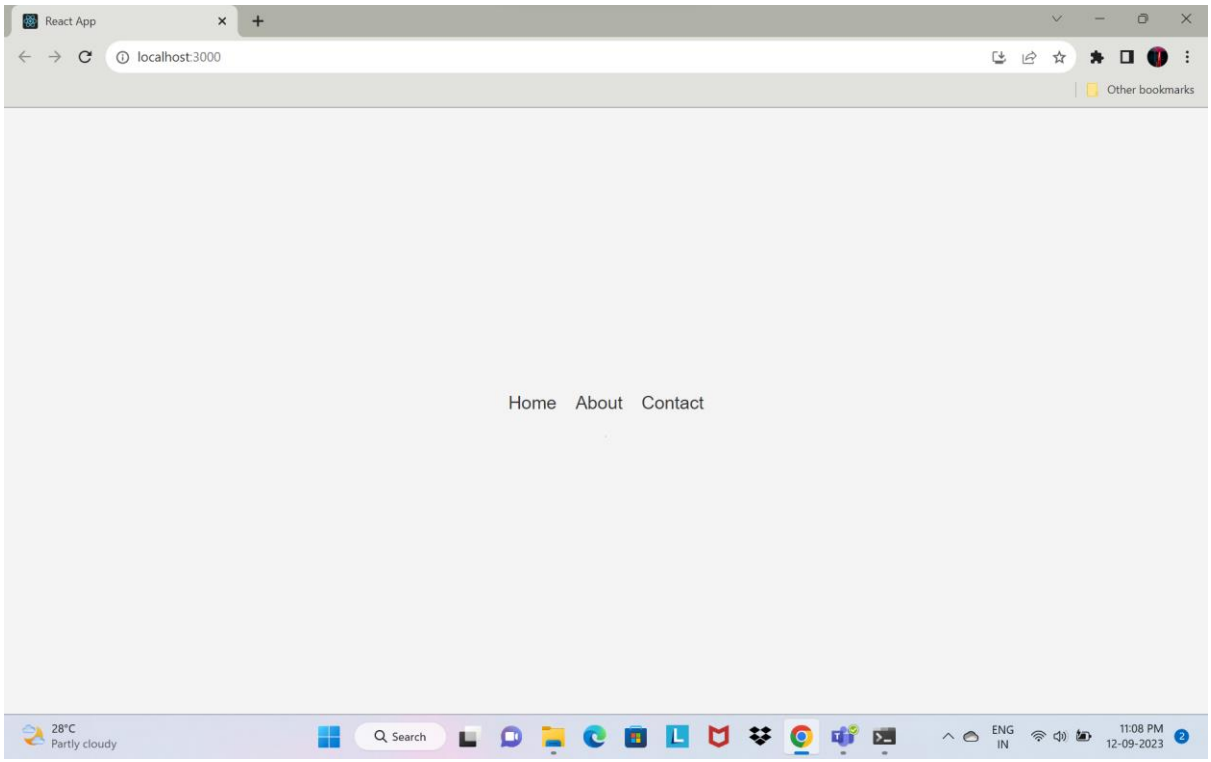
<h2>Contact Us</h2>

If you have any questions or need to get in touch with us, please feel
free to contact us using the information below:

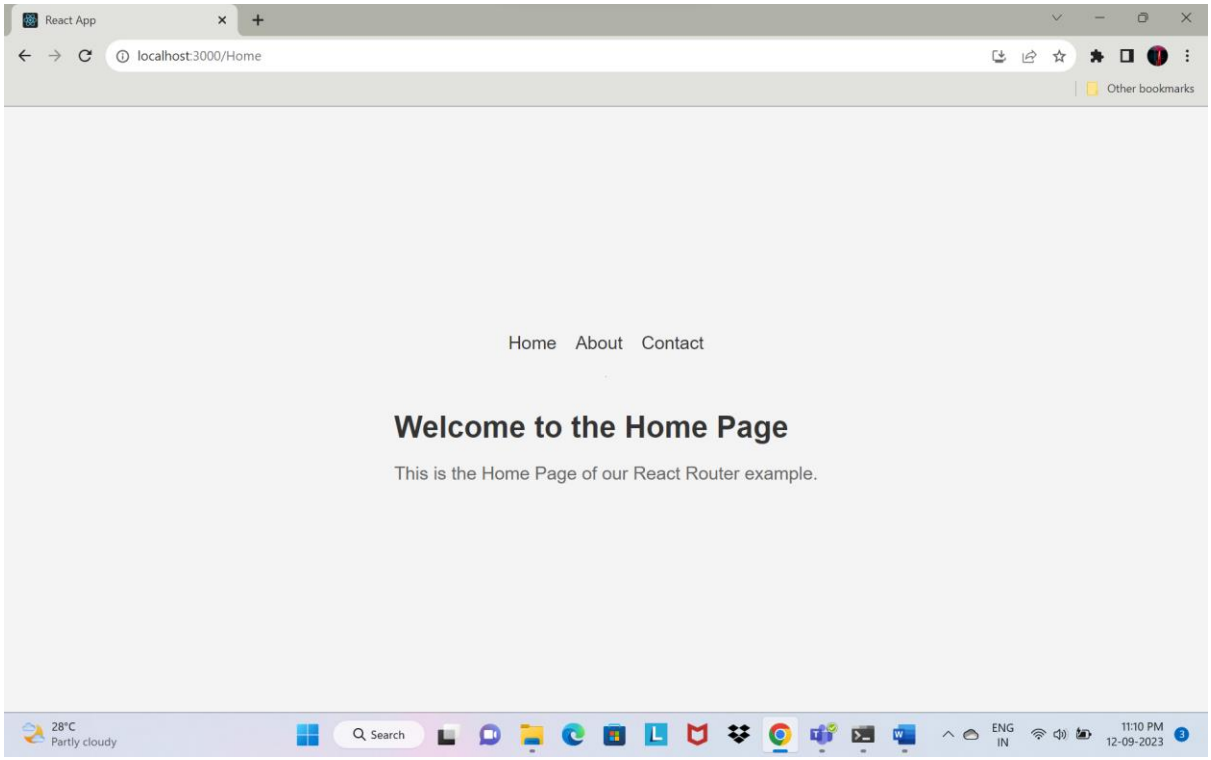
Email: animesh.nparab@gmail.com
Phone: +91 9326211146
Address: Borivali, Mumbai, India

);
}
export default Contact;
Ln 1, Col 1
100%

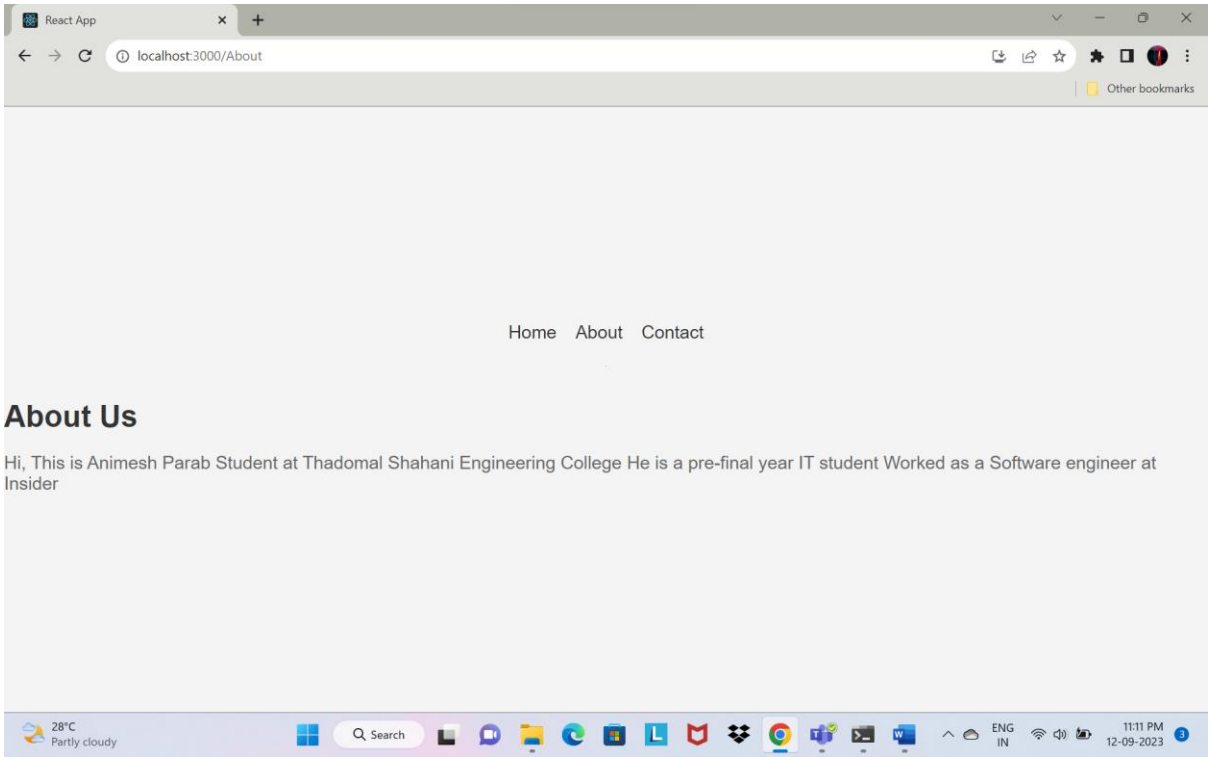
Out Put:-



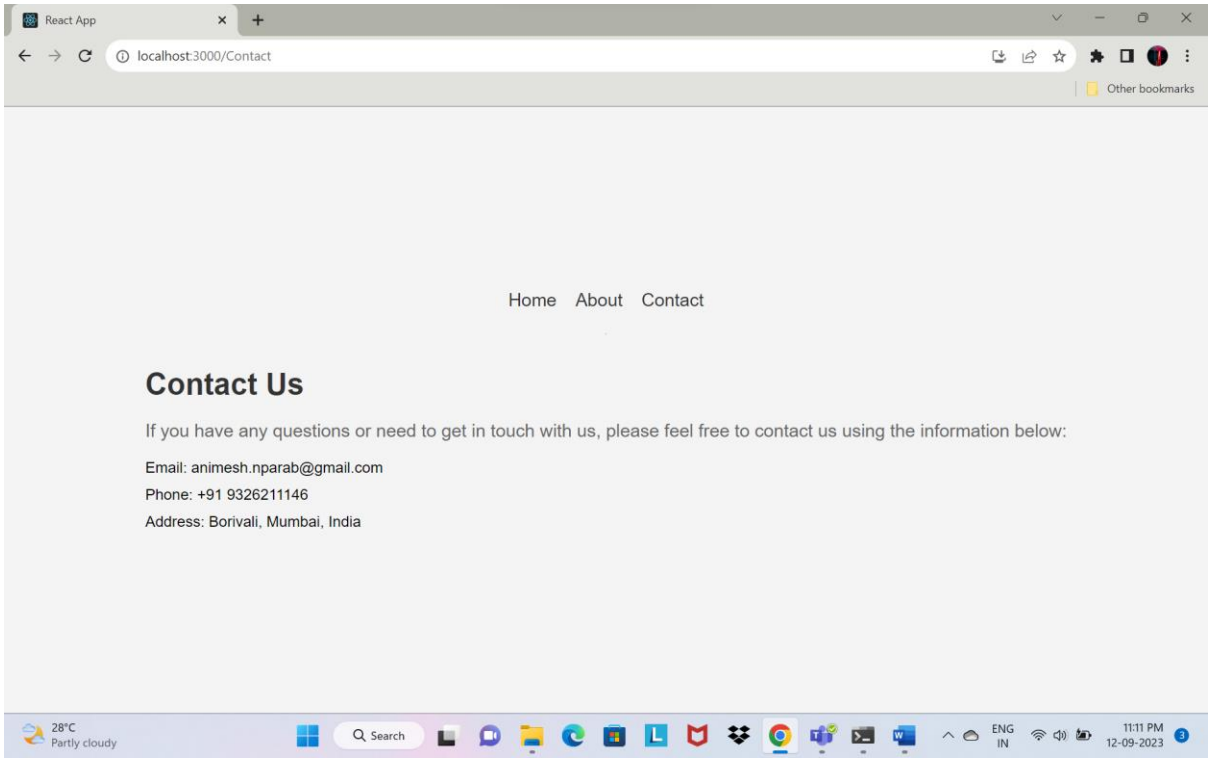
After clicking Home Button



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 :-

App.js

File Edit View

```
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;
```

Ln 20, Col 9 100% Unix (LF) UTF-

Output :-

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.

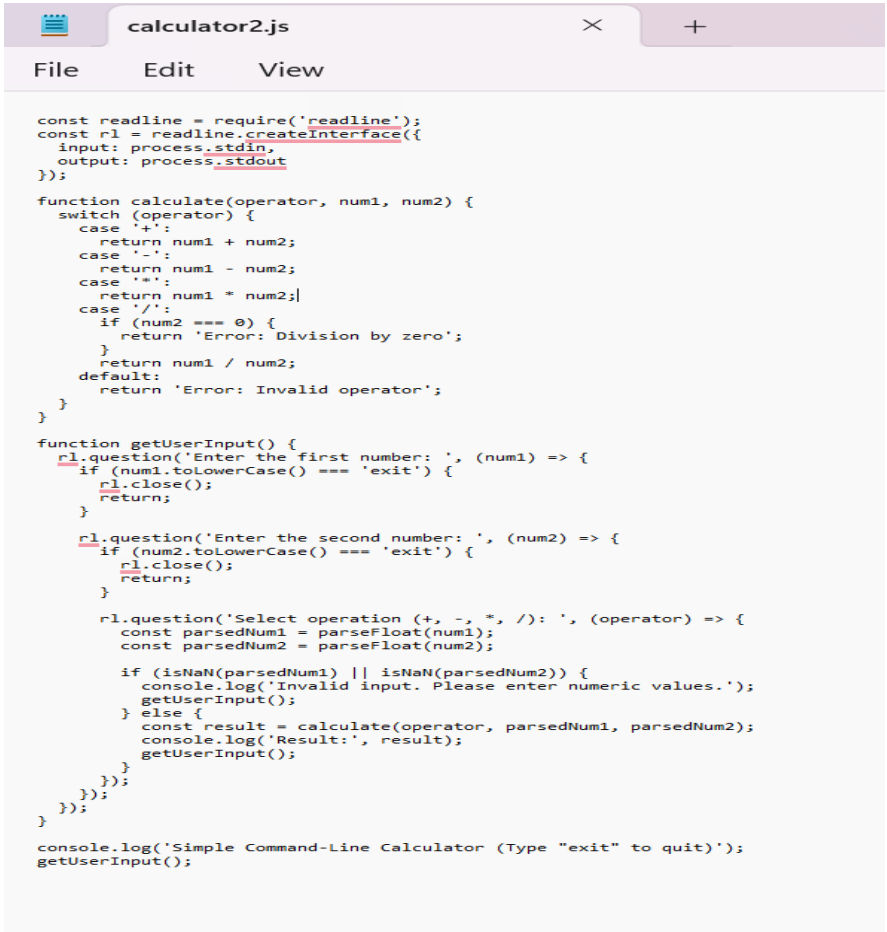
- Math and Calculations: REPL can be used for performing mathematical calculations and equations.

- Prototyping and Experimentation: It is valuable for experimenting with new ideas and algorithms.

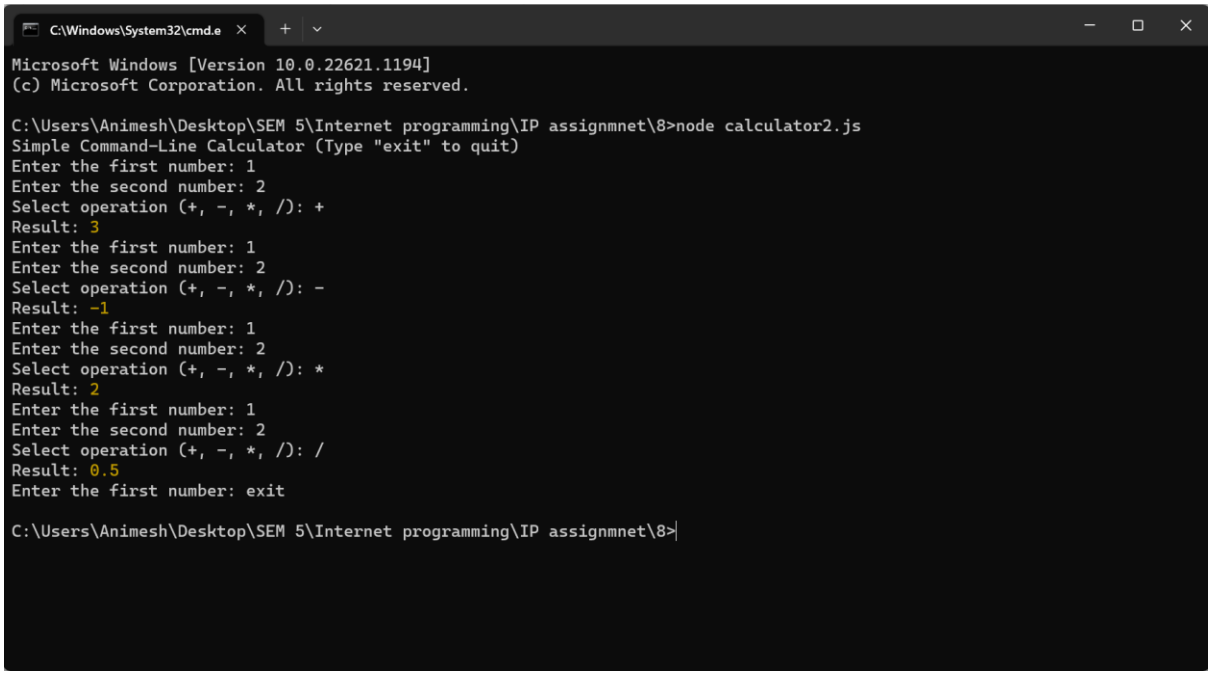
- Data Analysis: Data scientists use REPL to explore datasets and test data manipulation commands.

- Debugging: Debugging small portions of code to identify and fix errors.

Code :-



Output :-



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

Animesh Parab T2-T21 88

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 role can help in situations such as access .
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.

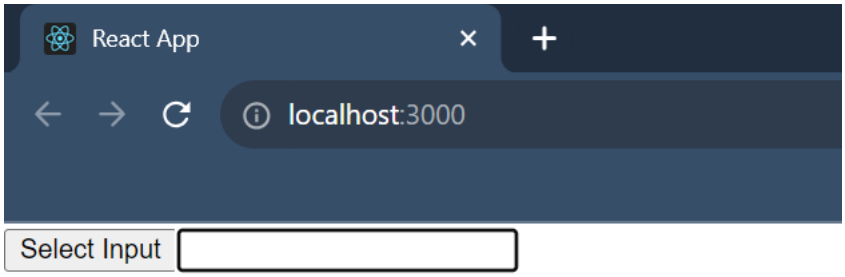
Code:

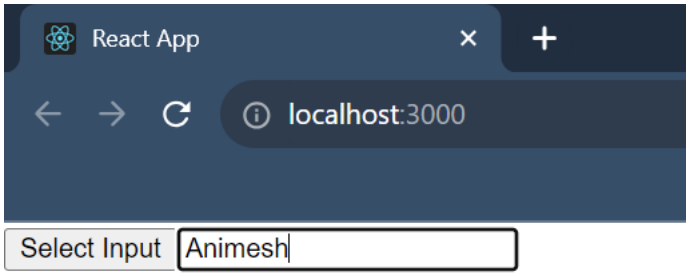
App.js

File Edit View

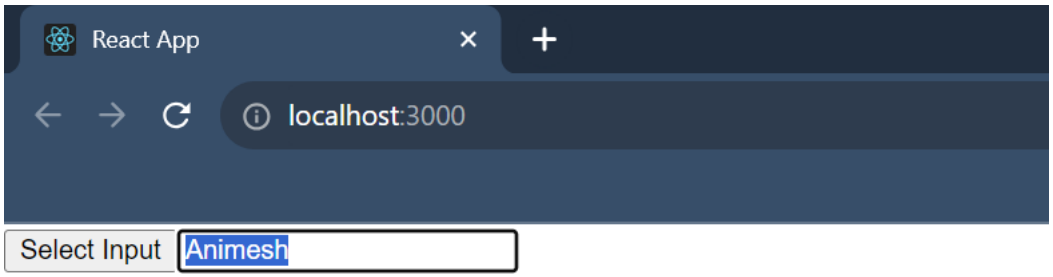
```
import React, { useRef } from 'react';

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.

Animesh Parab 88 T2-T21

Assignment 8 Nodejs File

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.

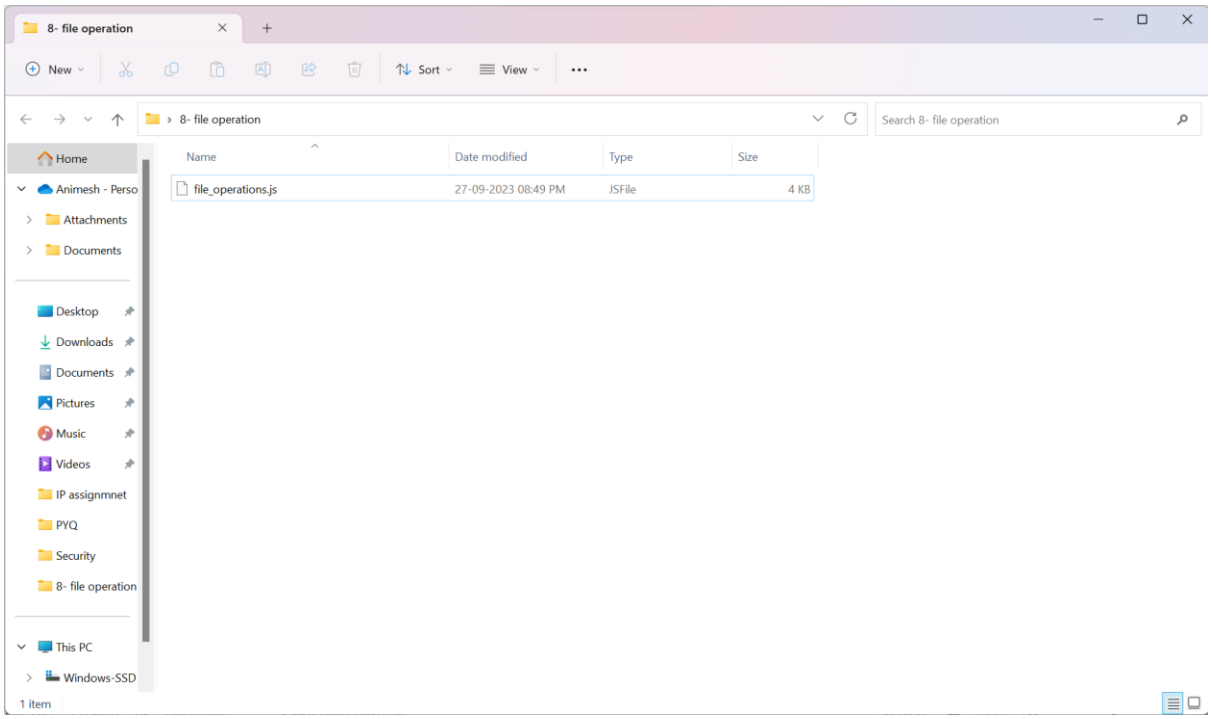
3. Writing Data to a File (Option 'c'): Users are asked to enter both a filename and content. The program utilizes the `fs.writeFile` method to write the provided content to the specified file.

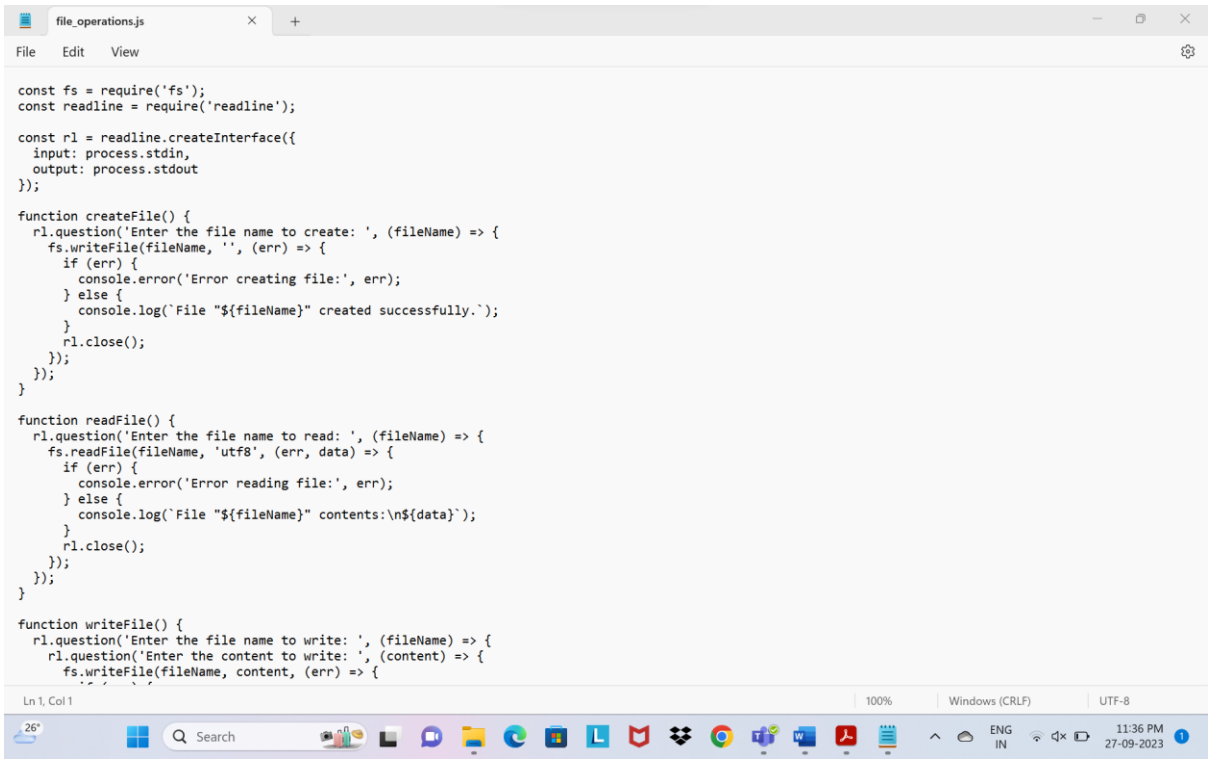
4. Renaming a File (Option 'd'): Users must input the current filename and the new filename. The program utilizes the `fs.rename` method to rename the file.

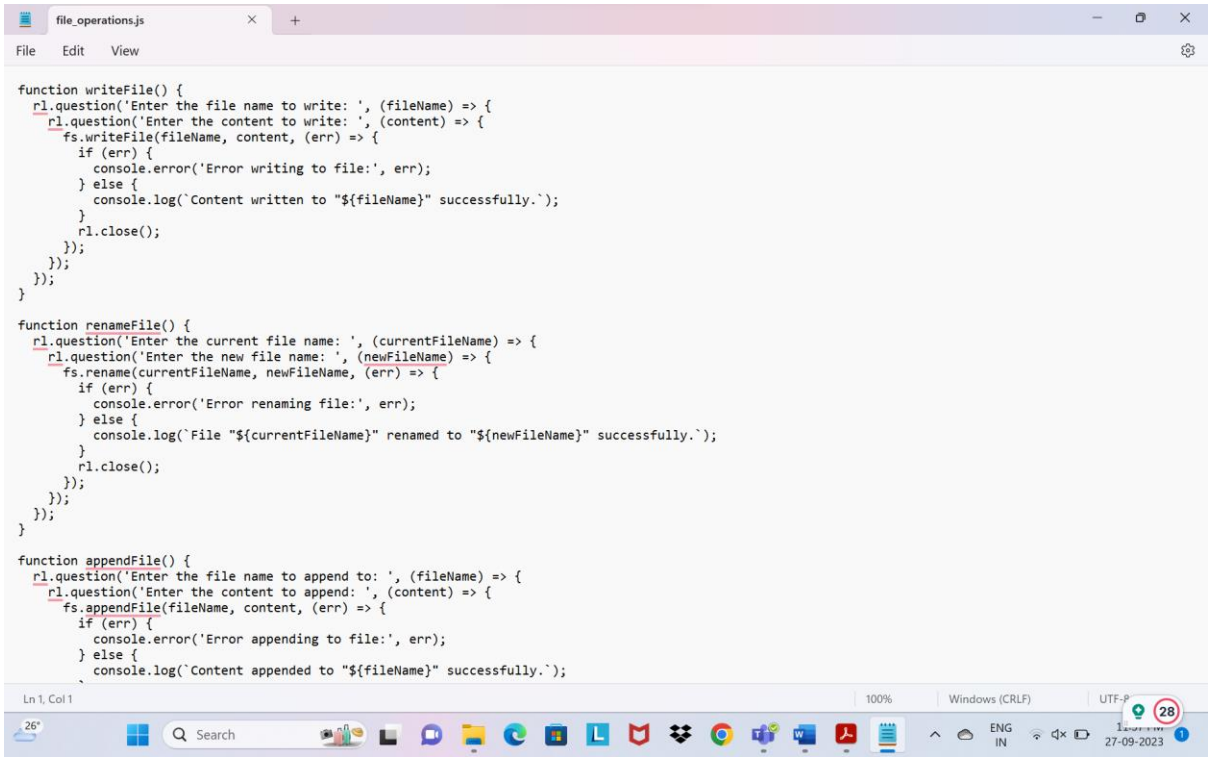
5. Appending Data to a File (Option 'e'): Users are asked to enter a filename and content to append. The program uses the `fs.appendFile` method to append the provided content to the specified file.

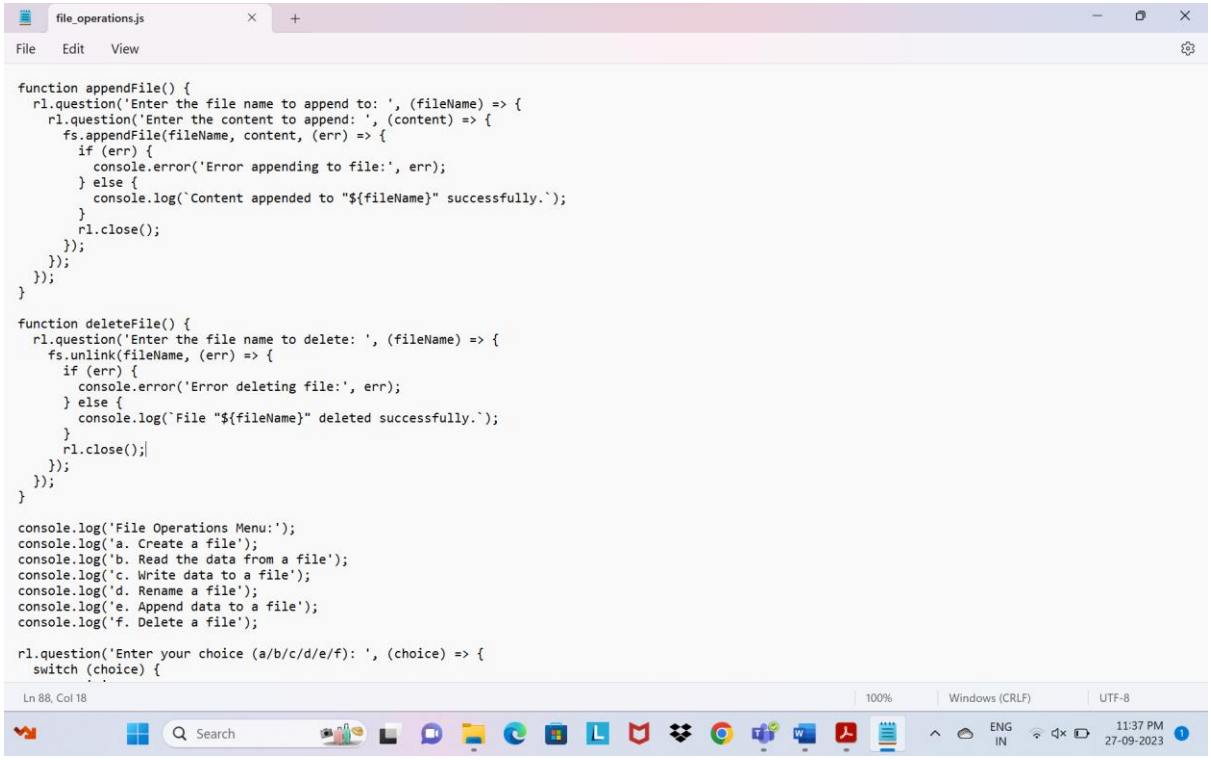
6. Deleting a File (Option 'f'): The program prompts the user to enter the filename they want to delete. It then uses the `fs.unlink` method to delete the specified file.

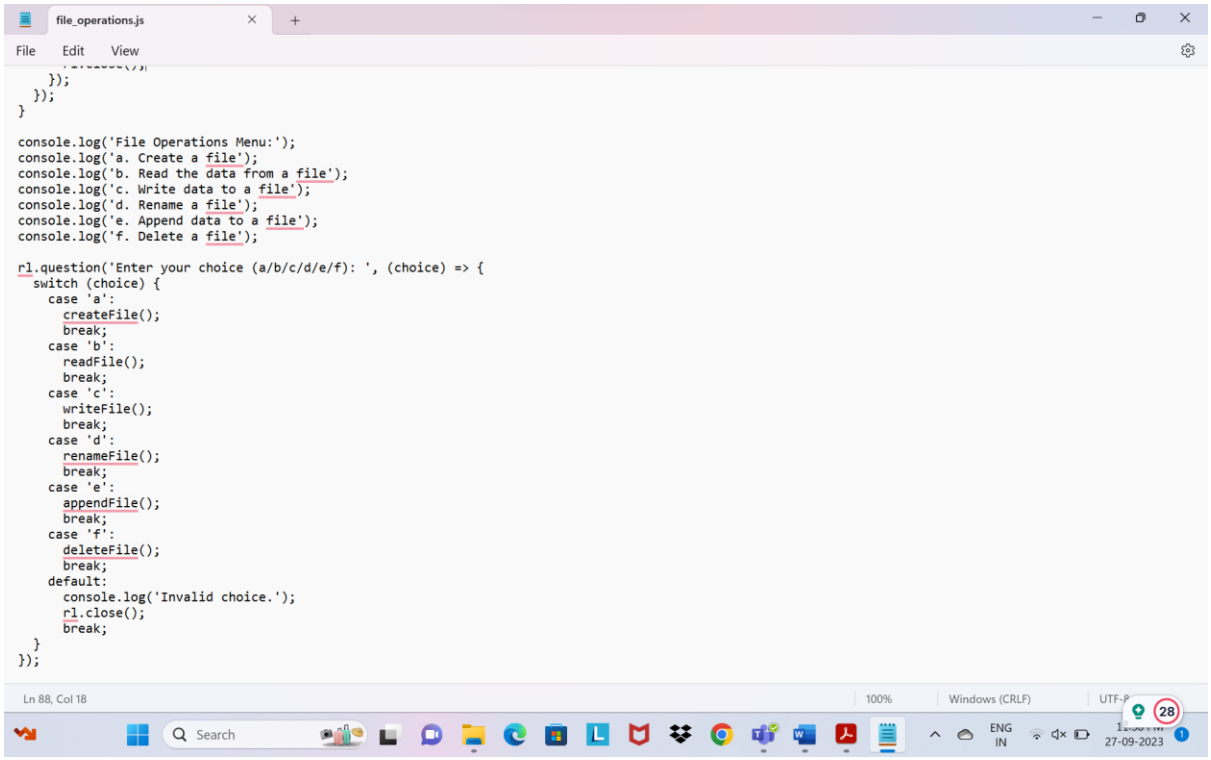
Code :-


8- file operation
New
Sort
View
8- file operation
Search 8- file operation
Home
Animesh - Perso
Attachments
Documents
Desktop
Downloads
Documents
Pictures
Music
Videos
IP assignmnet
PYQ
Security
8- file operation
This PC
Windows-SSD
Name
Date modified
Type
Size
file_operations.js
27-09-2023 08:49 PM
JSFile
4 KB
1 item


file_operations.js
File
Edit
View
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) => {
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8
Search
ENG
IN
11:36 PM
27-09-2023

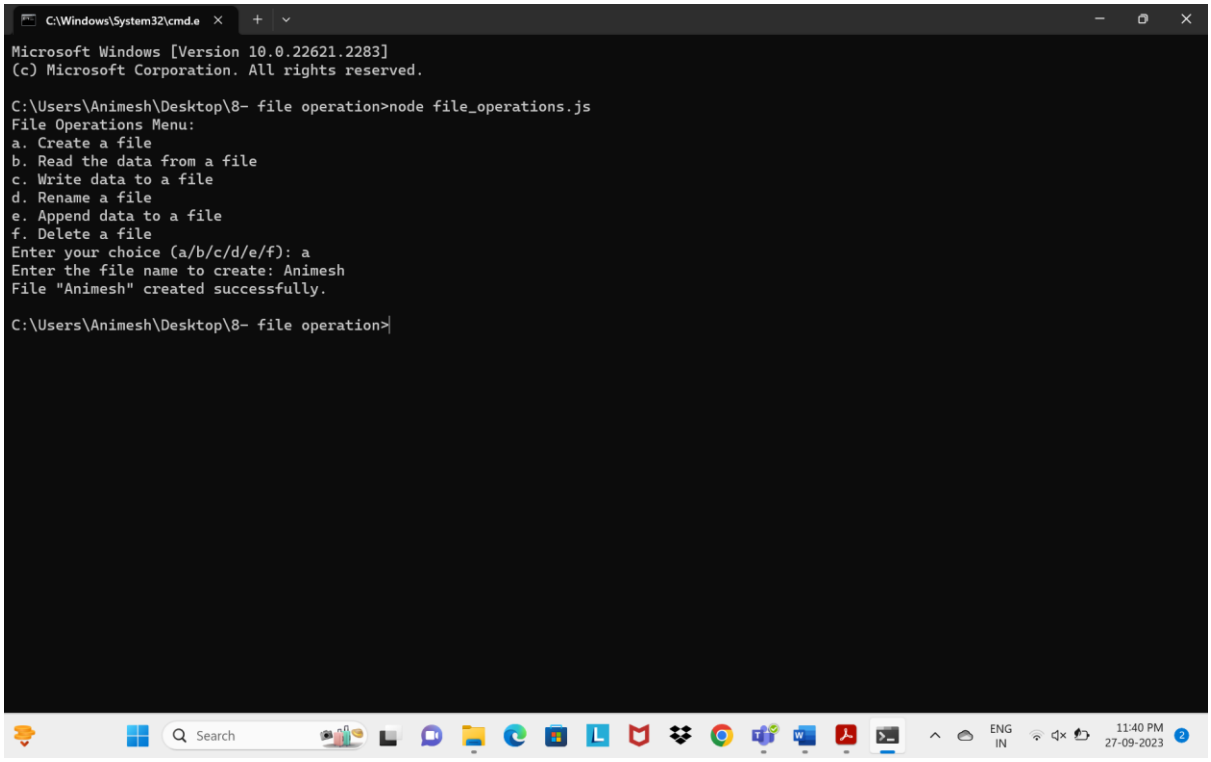

file_operations.js
File Edit View
function writeFile() {
rl.question('Enter the file name to write: ', (fileName) => {
rl.question('Enter the content to write: ', (content) => {
fs.writeFile(fileName, content, (err) => {
if (err) {
console.error('Error writing to file:', err);
} else {
console.log(`Content 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)
Search
ENG
IN
27-09-2023


file_operations.js
File Edit View
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) {
Ln 88, Col 18
100%
Windows (CRLF)
UTF-8
Search
ENG
IN
11:37 PM
27-09-2023

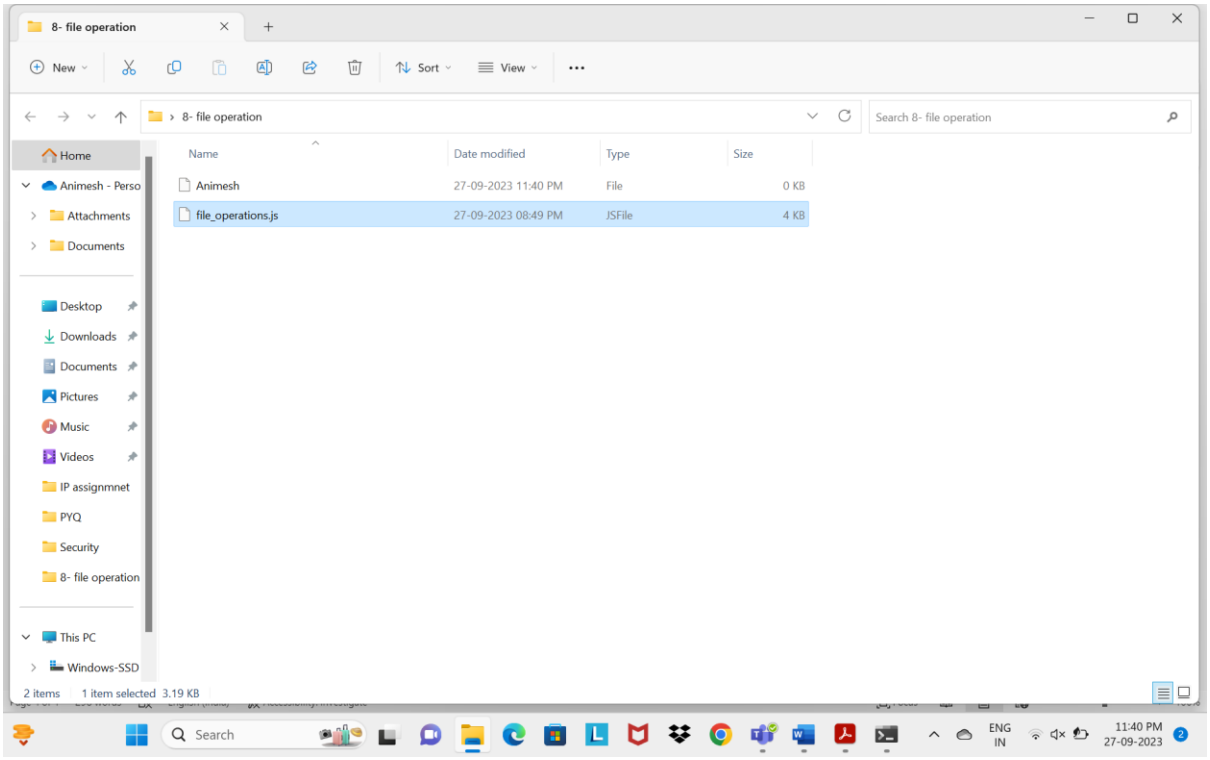


```
    });
  });
}

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;
  }
});
```

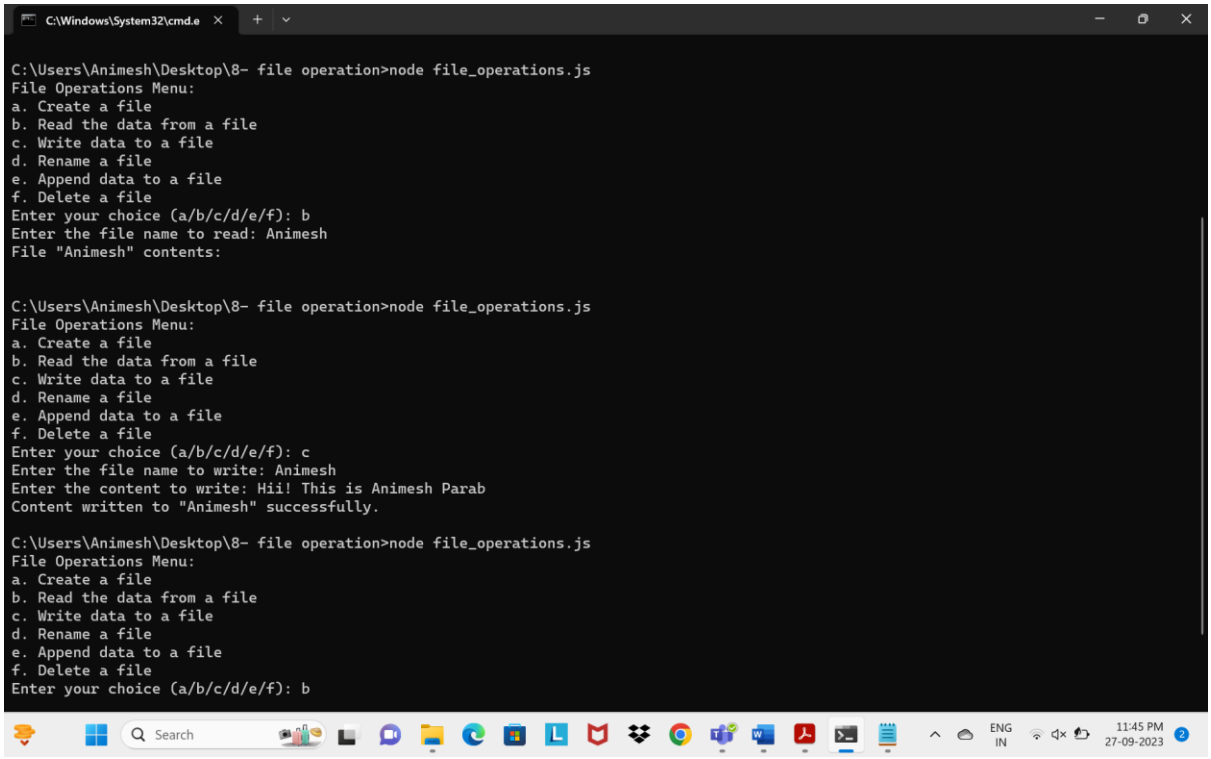
Code output :-

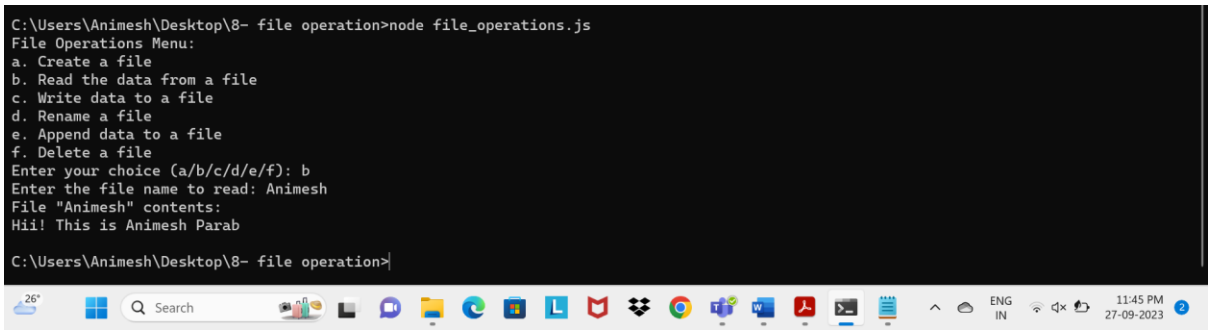


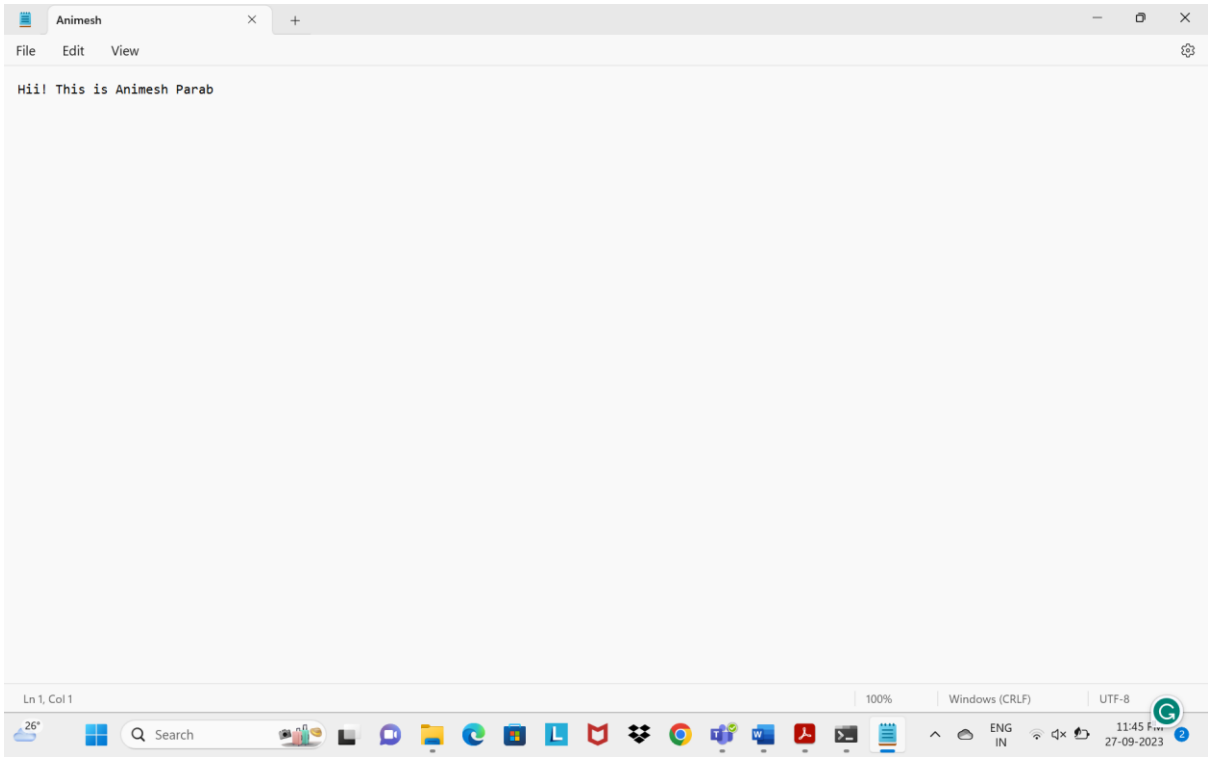
```
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

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>
```


8- file operation
New
Sort
View
8- file operation
Search 8- file operation
Home
Animesh - Perso
Attachments
Documents
Desktop
Downloads
Documents
Pictures
Music
Videos
IP assignmnet
PYQ
Security
8- file operation
This PC
Windows-SSD
Name
Date modified
Type
Size
Animesh
27-09-2023 11:40 PM
File
0 KB
file_operations.js
27-09-2023 08:49 PM
JSFile
4 KB
2 items
1 item selected 3.19 KB
Search
ENG
IN
11:40 PM
27-09-2023

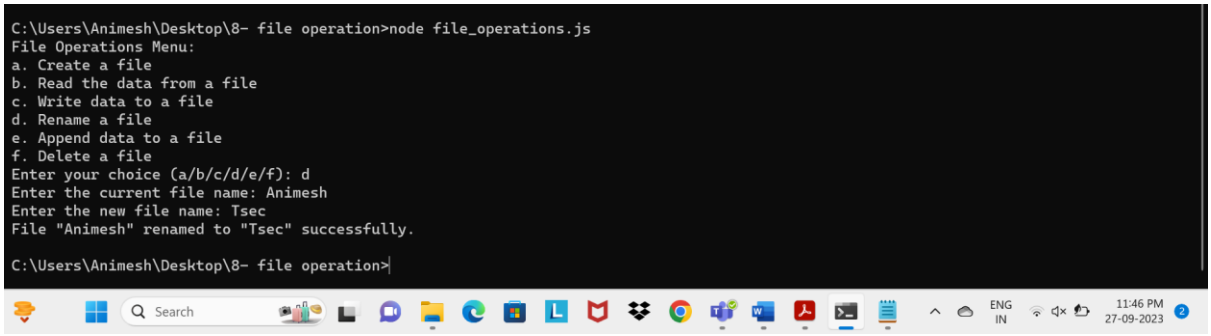
For read a data from file and write a data from file

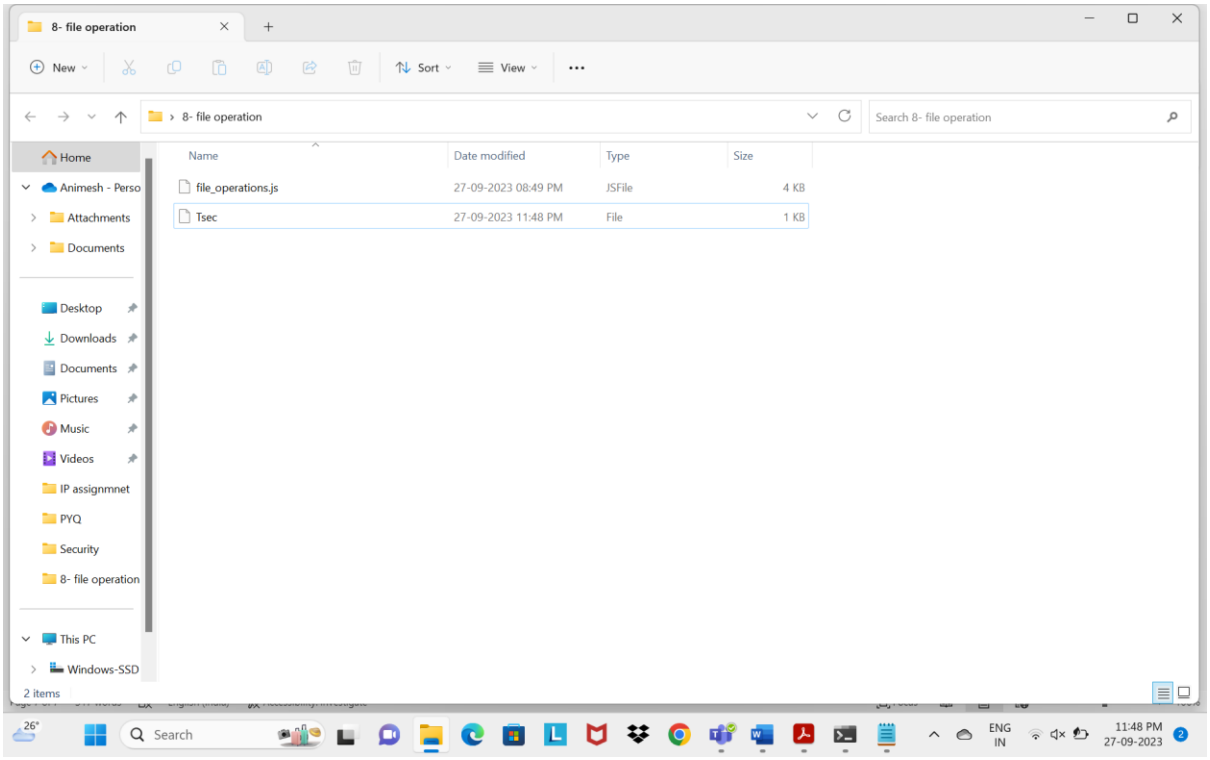




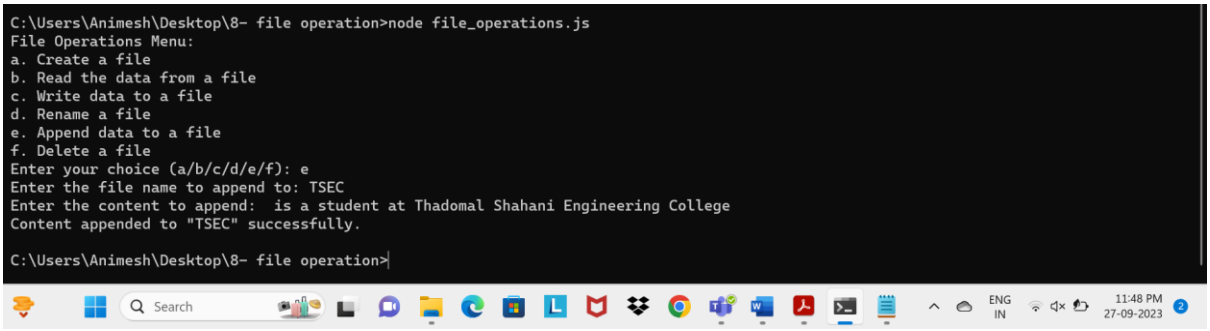


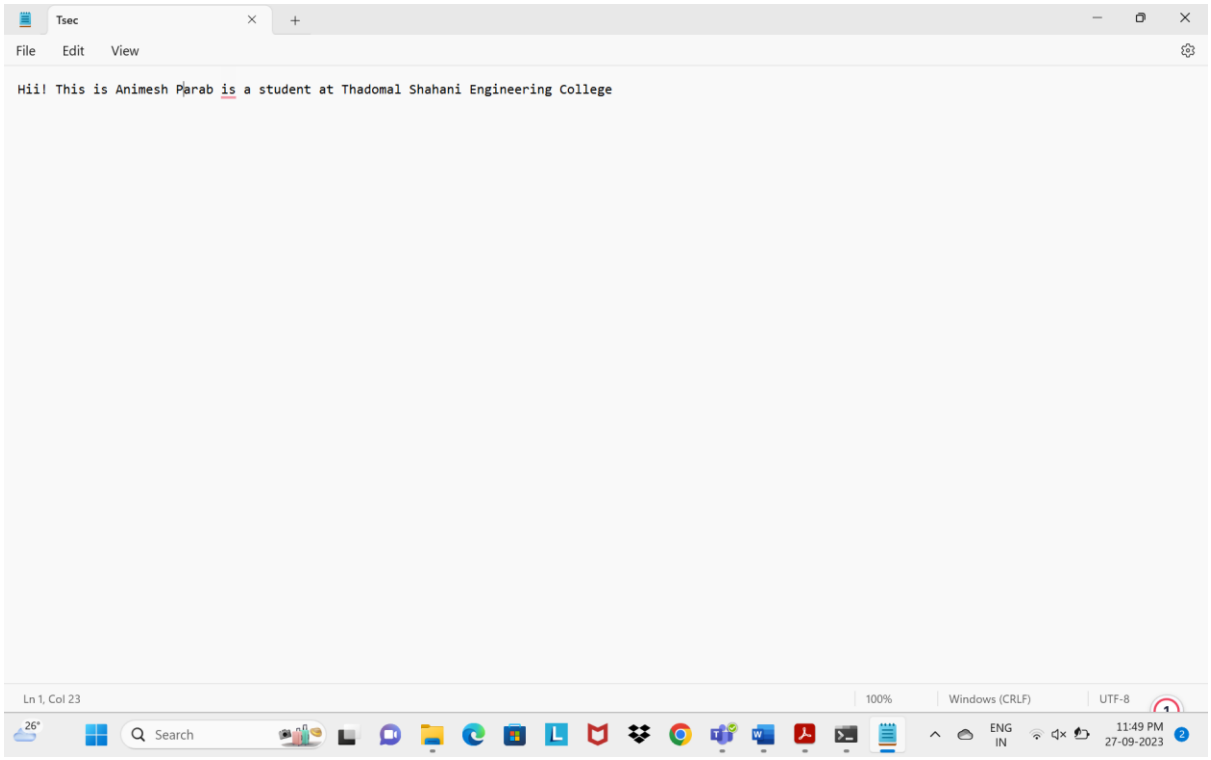
Rename file name



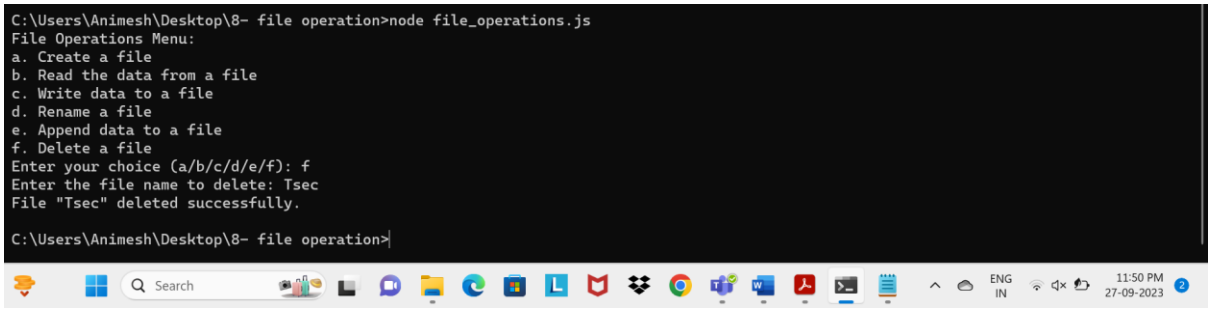


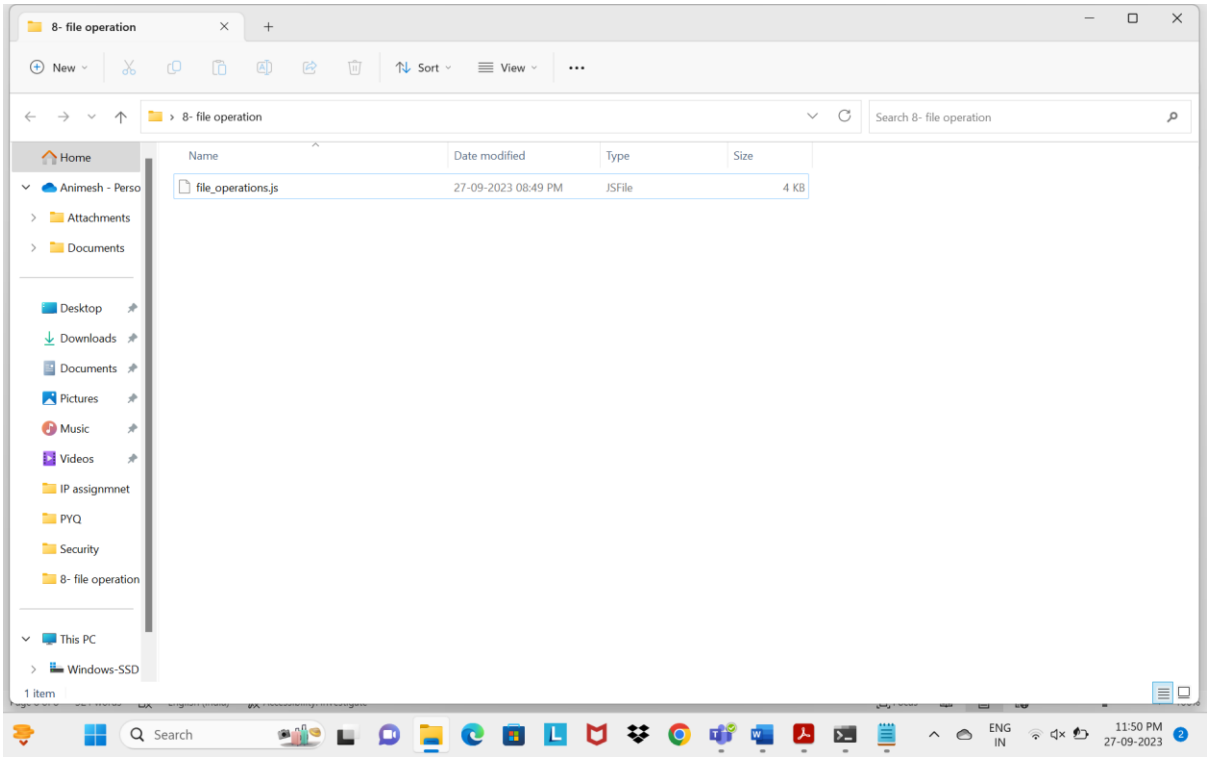
Append Data to file





For Delete a file





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

Animesh Parab T2-T21 88

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.

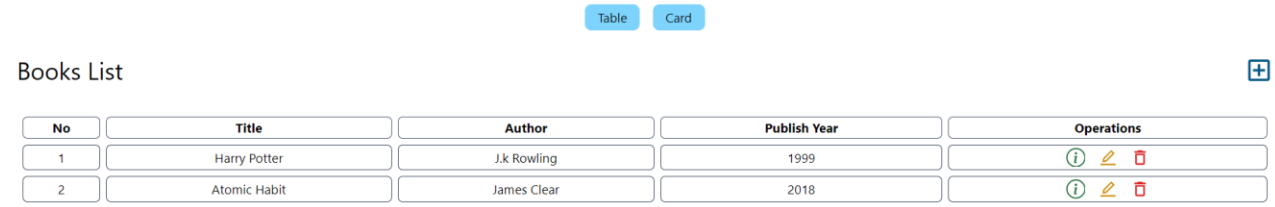
The CRUD acronym is a great, memorable framework for building applications and constructing usable models. We can examine why CRUD is essential from two fronts; developers and end-users.

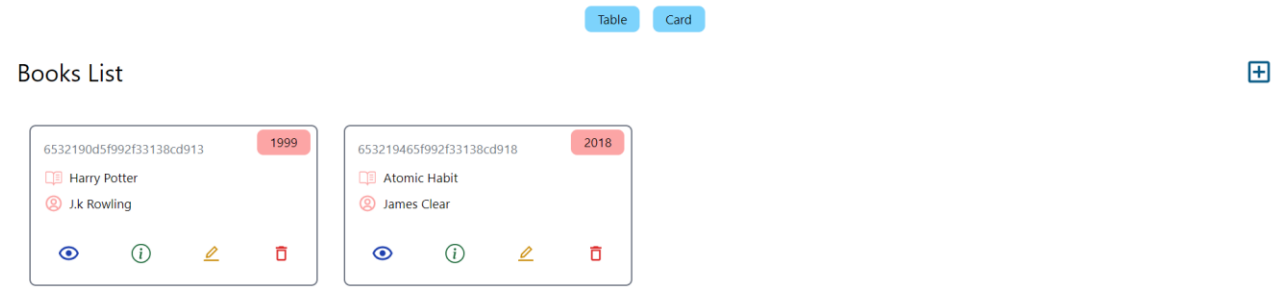
1. For developers, CRUD operations are essential to app development. In a similar vein, REST, a superset of CRUD, is necessary for website development.
2. For end-users, CRUD operations allow them to fill out forms, edit data, manage their admin panels and user settings.

CRUD offers many other benefits including:

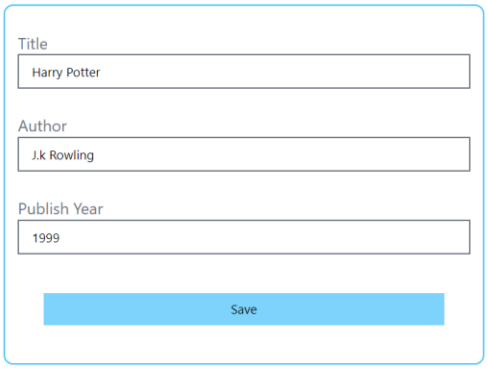
- Security control
- It is more performant vs. SQL statements
- It’s tried and tested
- It’s a recognized pattern and recognizable by most devs
- It simplifies the process and provides an easy framework for new developers to learn

Code and Output :

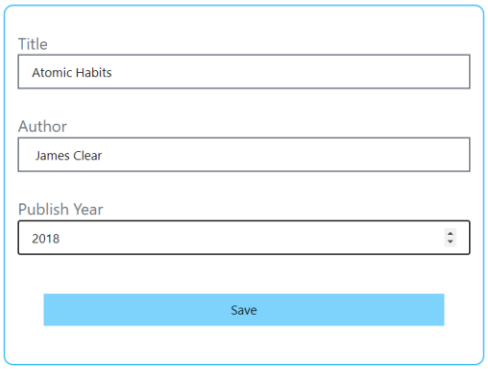


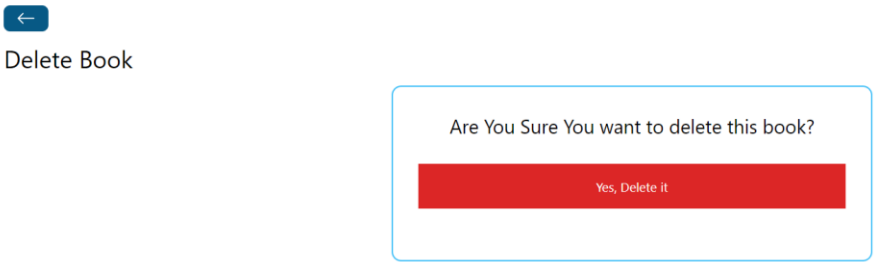


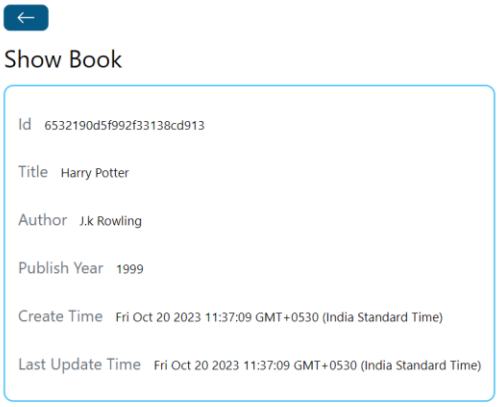
Edit Book



Create Book







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

Animesh Parab 88 T2 – T21

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

ii. 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.

iii. 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.

iv. 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.

Animesh Parab T2-T21 88

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 setState) 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 :