Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./~~Miss~~ SAIKARTHIK IYER

of IT Department, Semester V with

Roll No. 41 has completed a course of the necessary

experiments in the subject IP under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2024 - 2025

22/10/24

Teacher In- Charge

Head of the Department

Date 22/10/24

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1	Develop a Web page by using HTML/ HTML 5 Tags.		15/7/24	
2	Using CSS and CSS 3 enhance the Web page created in ass1		22/7/24	
3	Design a web page using concept of typography, color, transition		29/7/24	
4	Develop a web page using bootstrap framework.		5/8/24	
5	Write a program in JS to study conditional statement, loops, function, inheritance, iterator and generators.		12/8/24	
6	Write a JS program to study arrow function, Dom manipulation and CSS manipulation		2/9/24	22/10/24
7	Design a HTML 5 form and validate using Javascript		16/9/24	
8	Write a react program to implement the ~~concept~~ React JS		23/9/24	
9	Write a react program to implement the concept of Asynchronous programming using promises.		30/9/24	
10	Write a prog in node JS to a) Create a file b) Read the data c) Write the data to a file d) Rename a file e) delete a file		7/10/24	
11	Class assignment 1		15/9/24	
12	class assignment 2.		5/10/24	

Saikarthik Iyer

T13 41

AIM : Develop a Web Page by using html tags Theory : HTML (HyperText Markup Language) structures web content. The basic structure of an HTML document includes several key elements and follows a specific hierarchy. Here's a brief overview of the typical structure:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Shoe Store</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <nav>
            <div class="left-nav cursor">
                <img class="invert" src="./images/logo.svg" alt="">
                <p>Saiyer8</p>
            </div>
            <div class="right-nav">
                <ul>
```

```
            <li><a href="#">Home</a></li>
            <li><a href="#about">About Us</a></li>
            <li><a href="#products">Products</a></li>
            <li><a href="#featured">Featured</a></li>
            <li><a href="#pricelist">Pricing</a></li>
            <li><a href="#reviews">Reviews</a></li>
            <li><a href="#subscribe">Contact</a></li>
          </ul>
        </div>
      </nav>
    </header>
    <div class="container">
      <section id="about">
        <h2>About Us</h2>
        <p>
```

Welcome to TechNext, your premier destination for cutting-edge laptops and exceptional service. At TechNext, we're passionate about providing top-tier laptops that cater to every need—from high-performance machines for gamers and professionals to sleek, portable options for students and everyday users.

Founded by tech enthusiasts who understand the value of a reliable, powerful laptop, we pride ourselves on curating a selection of the latest models from leading brands. Our knowledgeable team is dedicated to offering personalized recommendations and unparalleled customer support to ensure you find the perfect device.

At TechNext, we believe technology should enhance your life, not complicate it. Explore our collection today and experience the difference of shopping with a team that genuinely cares about your tech needs.</p>

```
    </section>
    <section id="products">
        <h2>Our Products</h2>
        <div class="products">
            <div class="product">
                <img src="images/laptop1.jpg" alt="Shoe 1">
                <h3>Acer A632 GenX</h3>
                <p>₹50,000.00</p>
                <button>Add to Cart</button>
            </div>
            <div class="product">
                <img src="images/laptop2.jpg" alt="Shoe 2">
                <h3>ASUS Rogue X3</h3>
                <p>₹4,50,000</p>
                <button>Add to Cart</button>
            </div>
            <div class="product">
                <img src="images/laptop3.jpg" alt="Shoe 3">
                <h3>Acer K9</h3>
                <p>₹60,000.00</p>
                <button>Add to Cart</button>
```

```
        </div>
    </div>
</section>
<section id="featured">
    <h2>Featured Products</h2>
    <div class="featured-products">
        <div class="product">
            <img src="images/laptop1.jpg" alt="Shoe 4">
            <h3>HP Intel Core</h3>
            <p>₹16,800.00</p>
            <button>Add to Cart</button>
        </div>
        <div class="product">
            <img src="images/laptop2.jpg" alt="Shoe 5">
            <h3>Macbook Pro</h3>
            <p>₹1,20,000.00</p>
            <button>Add to Cart</button>
        </div>
        <div class="product">
            <img src="images/laptop3.jpg" alt="Shoe 5">
            <h3>ASUS Blazed G54</h3>
            <p>₹10,200.00</p>
            <button>Add to Cart</button>
        </div>
    </div>
```

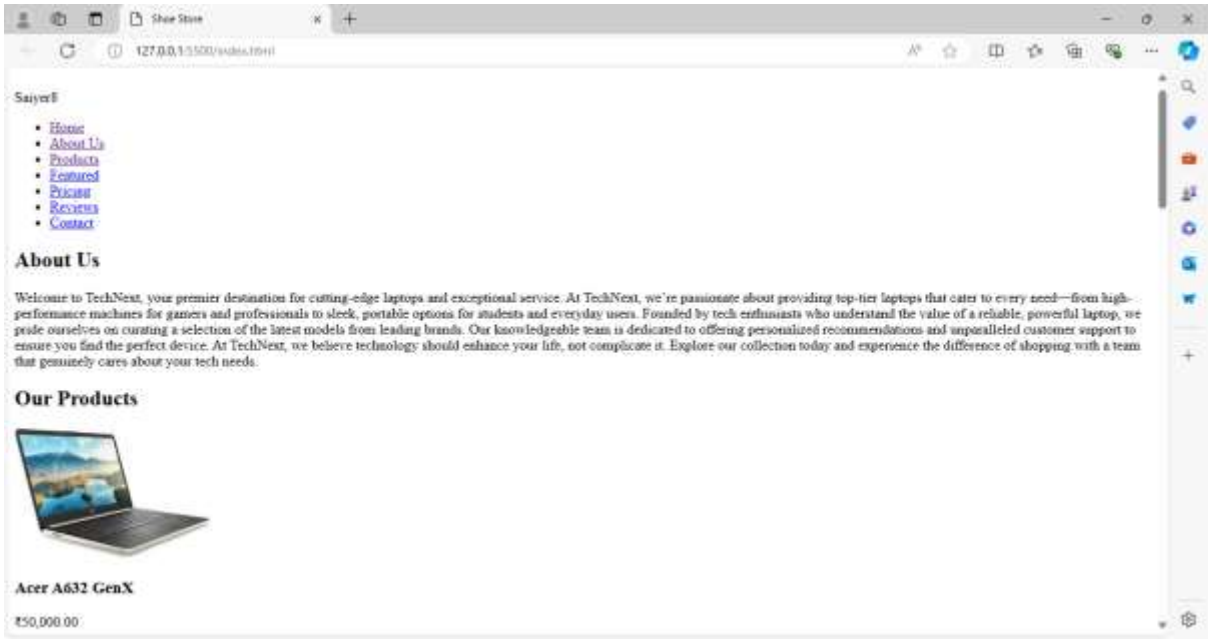
```
    </section>

    <section id="pricelist">
        <h2>Price List</h2>
        <table id="price-table">
            <thead>
                <tr>
                    <th>Product</th>
                    <th>Description</th>
                    <th>Price</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>Acer A632 GenX</td>
                    <td>Elegant black leather Laptop for formal
occasions.</td>
                    <td>₹50000.00</td>
                </tr>
                <tr>
                    <td>ASUS Rogue X3</td>
                    <td>Comfortable brown Laptop for everyday
Use.</td>
                    <td>₹450000.00</td>
                </tr>
```

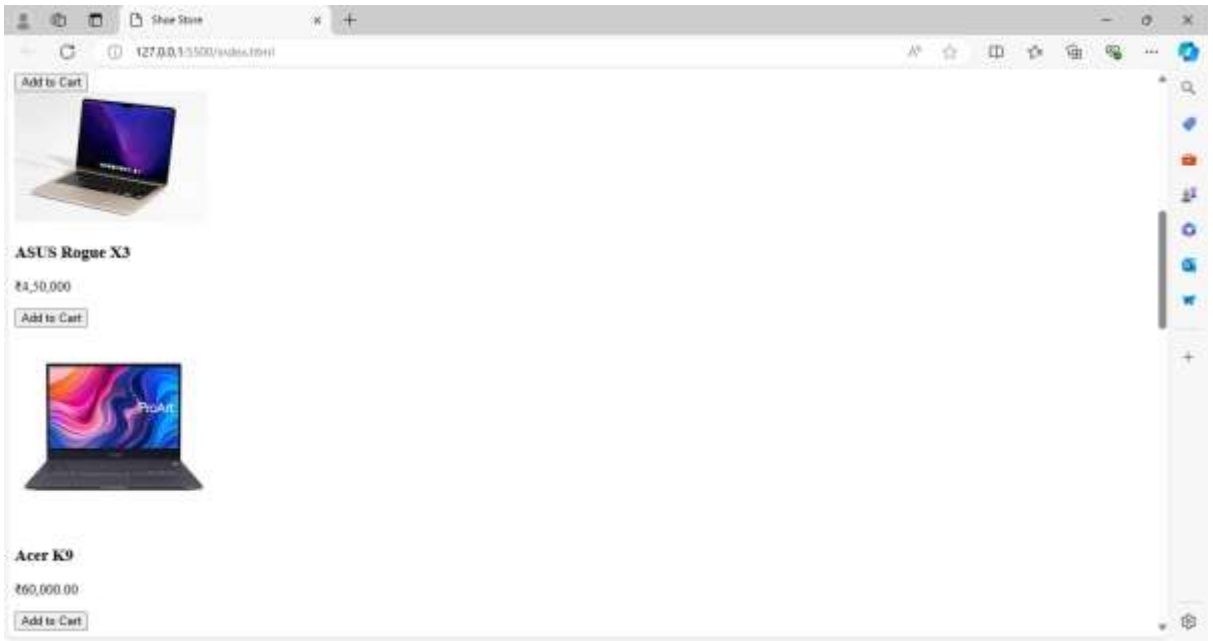
```
                <tr>
                    <td>Acer K9</td>
                    <td>Durable sports Laptop for athletic activities.</td>
                    <td>₹60000.00</td>
                </tr>
                <tr>
                    <td>HP Intel Core</td>
                    <td>Premium leather Laptop for special events.</td>
                    <td>₹168000.00</td>
                </tr>
                <tr>
                    <td>Macbook Pro</td>
                    <td>High-performance running Laptop for
runners.</td>
                    <td>₹120000.00</td>
                </tr>
                <tr>
                    <td>ASUS Blazed G54</td>
                    <td>Soft and comfortable Laptop for long walks.</td>
                    <td>₹10200.00</td>
                </tr>
            </tbody>
        </table>
    </section>
```

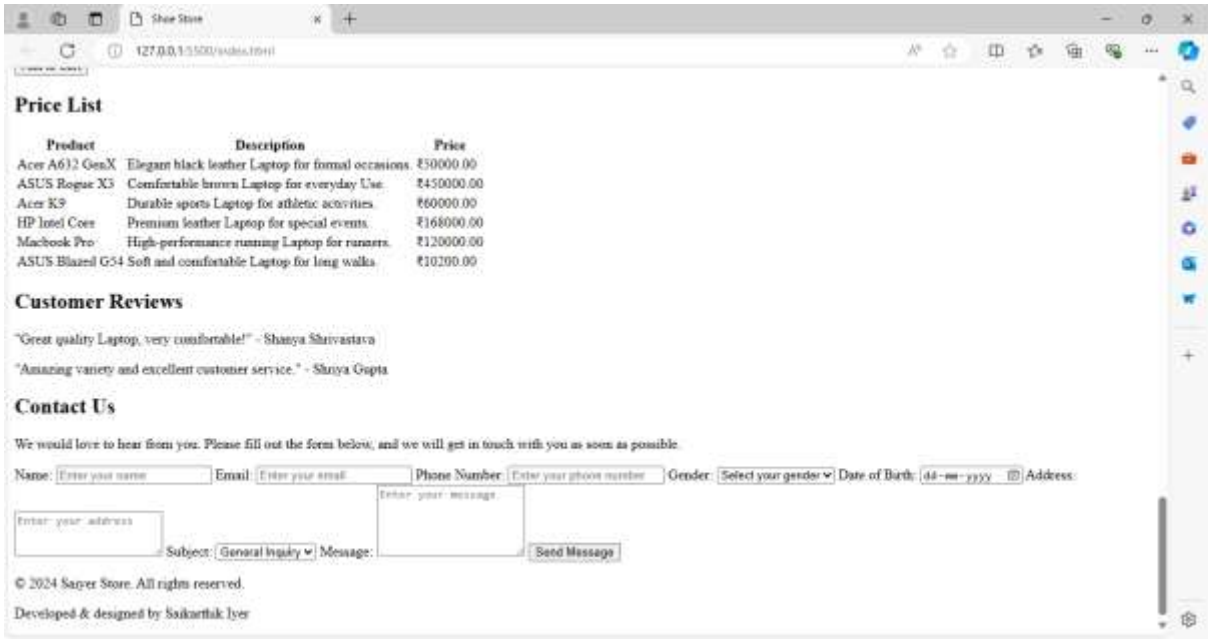
```
    <section id="reviews" class="reviews">
        <h2>Customer Reviews</h2>
        <div class="review">
            <p>"Great quality Laptop, very comfortable!" - Shanya Shrivastava</p>
        </div>
        <div class="review">
            <p>"Amazing variety and excellent customer service." - Shriya Gupta</p>
        </div>
    </section>
    <section id="subscribe" class="subscribe">
        <h2>Contact Us</h2>
        <p>We would love to hear from you. Please fill out the form below, and we will get in touch with you as soon as possible.</p>
        <form id="contact-form">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name" placeholder="Enter your name" required>

            <label for="email">Email:</label>
            <input type="email" id="email" name="email" placeholder="Enter your email" required>

            <label for="phone">Phone Number:</label>
```

```
        <input type="tel" id="phone" name="phone"
placeholder="Enter your phone number">

        <label for="gender">Gender:</label>
        <select id="gender" name="gender">
            <option value="" disabled selected>Select your
gender</option>
            <option value="male">Male</option>
            <option value="female">Female</option>
            <option value="non-binary">Non-Binary</option>
            <option value="other">Other</option>
        </select>

        <label for="dob">Date of Birth:</label>
        <input type="date" id="dob" name="dob">

        <label for="address">Address:</label>
        <textarea id="address" name="address" placeholder="Enter
your address" rows="3"></textarea>

        <label for="subject">Subject:</label>
        <select id="subject" name="subject">
            <option value="general">General Inquiry</option>
            <option value="support">Support</option>
            <option value="feedback">Feedback</option>
            <option value="other">Other</option>
```

```
            </select>

            <label for="message">Message:</label>
            <textarea id="message" name="message"
placeholder="Enter your message" rows="5" required></textarea>

            <button type="submit">Send Message</button>
        </form>
    </section>

</div>
<footer>
    <p>&copy; 2024 Saiyer Store. All rights reserved.</p>
    <p>Developed & designed by Saikarthik Iyer</p>
</footer>
</body>
</html>
```


Shoe Store
127.0.0.1:5500/index.html
Saiyer8
Home
About Us
Products
Featured
Pricing
Reviews
Contact
About Us
Welcome to TechNext, your premier destination for cutting-edge laptops and exceptional service. At TechNext, we're passionate about providing top-tier laptops that cater to every need—from high-performance machines for gamers and professionals to sleek, portable options for students and everyday users. Founded by tech enthusiasts who understand the value of a reliable, powerful laptop, we pride ourselves on curating a selection of the latest models from leading brands. Our knowledgeable team is dedicated to offering personalized recommendations and unparalleled customer support to ensure you find the perfect device. At TechNext, we believe technology should enhance your life, not complicate it. Explore our collection today and experience the difference of shopping with a team that genuinely cares about your tech needs.
Our Products
Acer A632 GenX
₹50,000.00


Shoe Store
127.0.0.1:5500/index.html
Add to Cart
ASUS Rogue X3
₹4,50,000
Add to Cart
Acer K9
₹60,000.00
Add to Cart



Conclusion: LO1 achieved - Identified and applied the appropriate HTML tags to develop a webpage

Saikarthik Iyer

T13 41

AIM : Develop a Web Page by using html tags Theory : HTML (HyperText Markup Language) structures web content. The basic structure of an HTML document includes several key elements and follows a specific hierarchy. Here's a brief overview of the typical structure:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Shoe Store</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header>
        <nav>
            <div class="left-nav cursor">
                <img class="invert" src="./images/logo.svg" alt="">
                <p>Saiyer8</p>
            </div>
            <div class="right-nav">
                <ul>
```

```
            <li><a href="#">Home</a></li>
            <li><a href="#about">About Us</a></li>
            <li><a href="#products">Products</a></li>
            <li><a href="#featured">Featured</a></li>
            <li><a href="#pricelist">Pricing</a></li>
            <li><a href="#reviews">Reviews</a></li>
            <li><a href="#subscribe">Contact</a></li>
          </ul>
        </div>
      </nav>
    </header>
    <div class="container">
      <section id="about">
        <h2>About Us</h2>
        <p>
```

Welcome to TechNext, your premier destination for cutting-edge laptops and exceptional service. At TechNext, we're passionate about providing top-tier laptops that cater to every need—from high-performance machines for gamers and professionals to sleek, portable options for students and everyday users.

Founded by tech enthusiasts who understand the value of a reliable, powerful laptop, we pride ourselves on curating a selection of the latest models from leading brands. Our knowledgeable team is dedicated to offering personalized recommendations and unparalleled customer support to ensure you find the perfect device.

At TechNext, we believe technology should enhance your life, not complicate it. Explore our collection today and experience the difference of shopping with a team that genuinely cares about your tech needs.</p>

```
    </section>
    <section id="products">
        <h2>Our Products</h2>
        <div class="products">
            <div class="product">
                <img src="images/laptop1.jpg" alt="Shoe 1">
                <h3>Acer A632 GenX</h3>
                <p>₹50,000.00</p>
                <button>Add to Cart</button>
            </div>
            <div class="product">
                <img src="images/laptop2.jpg" alt="Shoe 2">
                <h3>ASUS Rogue X3</h3>
                <p>₹4,50,000</p>
                <button>Add to Cart</button>
            </div>
            <div class="product">
                <img src="images/laptop3.jpg" alt="Shoe 3">
                <h3>Acer K9</h3>
                <p>₹60,000.00</p>
                <button>Add to Cart</button>
```

```
        </div>
    </div>
</section>
<section id="featured">
    <h2>Featured Products</h2>
    <div class="featured-products">
        <div class="product">
            <img src="images/laptop1.jpg" alt="Shoe 4">
            <h3>HP Intel Core</h3>
            <p>₹16,800.00</p>
            <button>Add to Cart</button>
        </div>
        <div class="product">
            <img src="images/laptop2.jpg" alt="Shoe 5">
            <h3>Macbook Pro</h3>
            <p>₹1,20,000.00</p>
            <button>Add to Cart</button>
        </div>
        <div class="product">
            <img src="images/laptop3.jpg" alt="Shoe 5">
            <h3>ASUS Blazed G54</h3>
            <p>₹10,200.00</p>
            <button>Add to Cart</button>
        </div>
    </div>
```

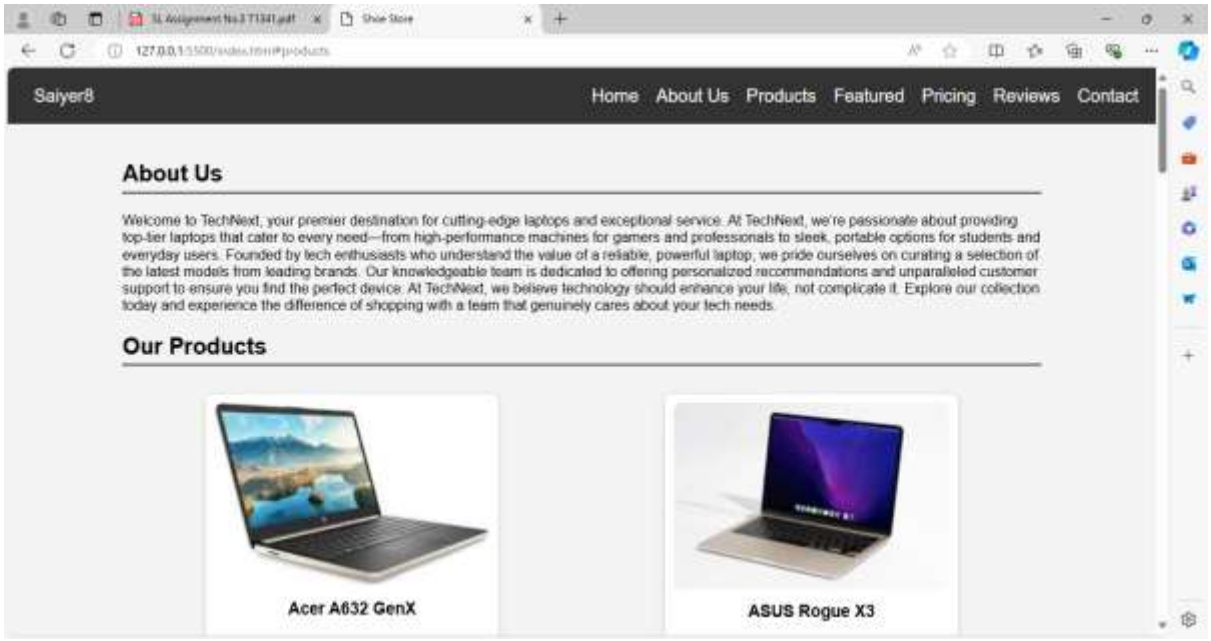
```
    </section>

    <section id="pricelist">
        <h2>Price List</h2>
        <table id="price-table">
            <thead>
                <tr>
                    <th>Product</th>
                    <th>Description</th>
                    <th>Price</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>Acer A632 GenX</td>
                    <td>Elegant black leather Laptop for formal
occasions.</td>
                    <td>₹50000.00</td>
                </tr>
                <tr>
                    <td>ASUS Rogue X3</td>
                    <td>Comfortable brown Laptop for everyday
Use.</td>
                    <td>₹450000.00</td>
                </tr>
```

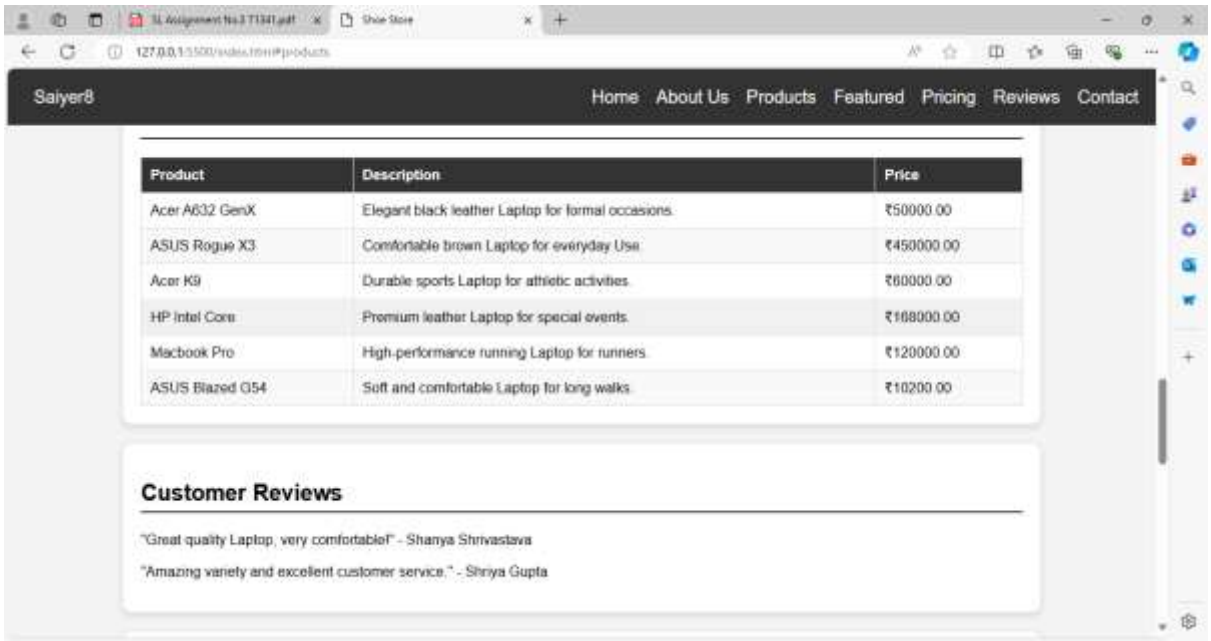
```
                <tr>
                    <td>Acer K9</td>
                    <td>Durable sports Laptop for athletic activities.</td>
                    <td>₹60000.00</td>
                </tr>
                <tr>
                    <td>HP Intel Core</td>
                    <td>Premium leather Laptop for special events.</td>
                    <td>₹168000.00</td>
                </tr>
                <tr>
                    <td>Macbook Pro</td>
                    <td>High-performance running Laptop for
runners.</td>
                    <td>₹120000.00</td>
                </tr>
                <tr>
                    <td>ASUS Blazed G54</td>
                    <td>Soft and comfortable Laptop for long walks.</td>
                    <td>₹10200.00</td>
                </tr>
            </tbody>
        </table>
    </section>
```

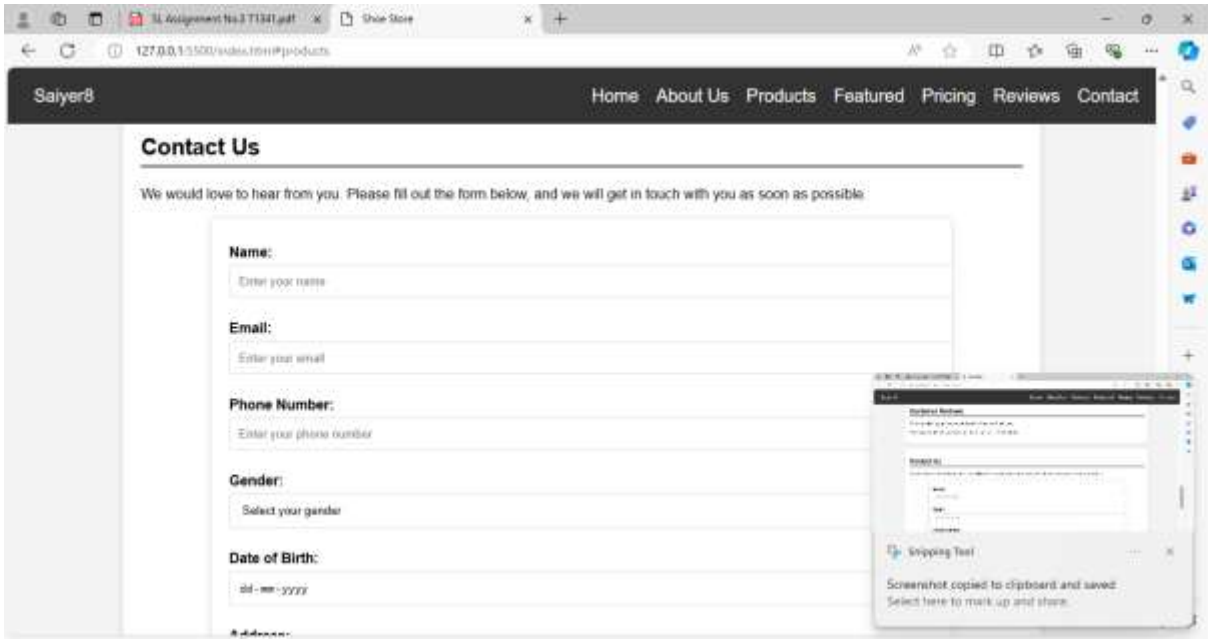
```
    <section id="reviews" class="reviews">
        <h2>Customer Reviews</h2>
        <div class="review">
            <p>"Great quality Laptop, very comfortable!" - Shanya Shrivastava</p>
        </div>
        <div class="review">
            <p>"Amazing variety and excellent customer service." - Shriya Gupta</p>
        </div>
    </section>
    <section id="subscribe" class="subscribe">
        <h2>Contact Us</h2>
        <p>We would love to hear from you. Please fill out the form below, and we will get in touch with you as soon as possible.</p>
        <form id="contact-form">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name" placeholder="Enter your name" required>

            <label for="email">Email:</label>
            <input type="email" id="email" name="email" placeholder="Enter your email" required>

            <label for="phone">Phone Number:</label>
```

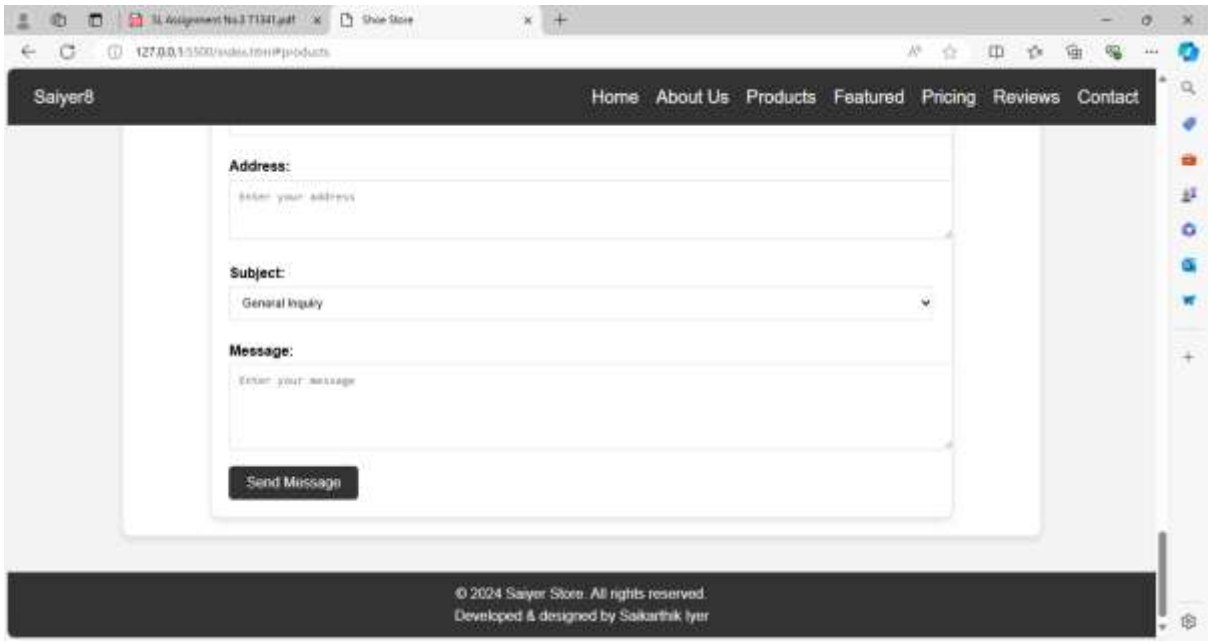
```
        <input type="tel" id="phone" name="phone"
placeholder="Enter your phone number">

        <label for="gender">Gender:</label>
        <select id="gender" name="gender">
            <option value="" disabled selected>Select your
gender</option>
            <option value="male">Male</option>
            <option value="female">Female</option>
            <option value="non-binary">Non-Binary</option>
            <option value="other">Other</option>
        </select>

        <label for="dob">Date of Birth:</label>
        <input type="date" id="dob" name="dob">

        <label for="address">Address:</label>
        <textarea id="address" name="address" placeholder="Enter
your address" rows="3"></textarea>

        <label for="subject">Subject:</label>
        <select id="subject" name="subject">
            <option value="general">General Inquiry</option>
            <option value="support">Support</option>
            <option value="feedback">Feedback</option>
            <option value="other">Other</option>
```

```
            </select>

            <label for="message">Message:</label>
            <textarea id="message" name="message"
placeholder="Enter your message" rows="5" required></textarea>

            <button type="submit">Send Message</button>
        </form>
    </section>

</div>
<footer>
    <p>&copy; 2024 Saiyer Store. All rights reserved.</p>
    <p>Developed & designed by Saikarthik Iyer</p>
</footer>
</body>
</html>
```


Saiyer8
Home About Us Products Featured Pricing Reviews Contact
About Us
Welcome to TechNext, your premier destination for cutting-edge laptops and exceptional service. At TechNext, we're passionate about providing top-tier laptops that cater to every need—from high-performance machines for gamers and professionals to sleek, portable options for students and everyday users. Founded by tech enthusiasts who understand the value of a reliable, powerful laptop, we pride ourselves on curating a selection of the latest models from leading brands. Our knowledgeable team is dedicated to offering personalized recommendations and unparalleled customer support to ensure you find the perfect device. At TechNext, we believe technology should enhance your life, not complicate it. Explore our collection today and experience the difference of shopping with a team that genuinely cares about your tech needs.
Our Products
Acer A632 GenX
ASUS Rogue X3


Saiyer8
Home About Us Products Featured Pricing Reviews Contact
Product
Description
Price
Acer A632 GenX
Elegant black leather Laptop for formal occasions.
₹50000.00
ASUS Rogue X3
Comfortable brown Laptop for everyday Use.
₹450000.00
Acer K9
Durable sports Laptop for athletic activities.
₹60000.00
HP Intel Core
Premium leather Laptop for special events.
₹168000.00
Macbook Pro
High-performance running Laptop for runners.
₹120000.00
ASUS Blazed G54
Soft and comfortable Laptop for long walks.
₹10200.00
Customer Reviews
"Great quality Laptop, very comfortable!" - Shanya Shrivastava
"Amazing variety and excellent customer service." - Shriya Gupta





Conclusion: LO1 achieved - Identified and applied the appropriate HTML tags to develop a webpage

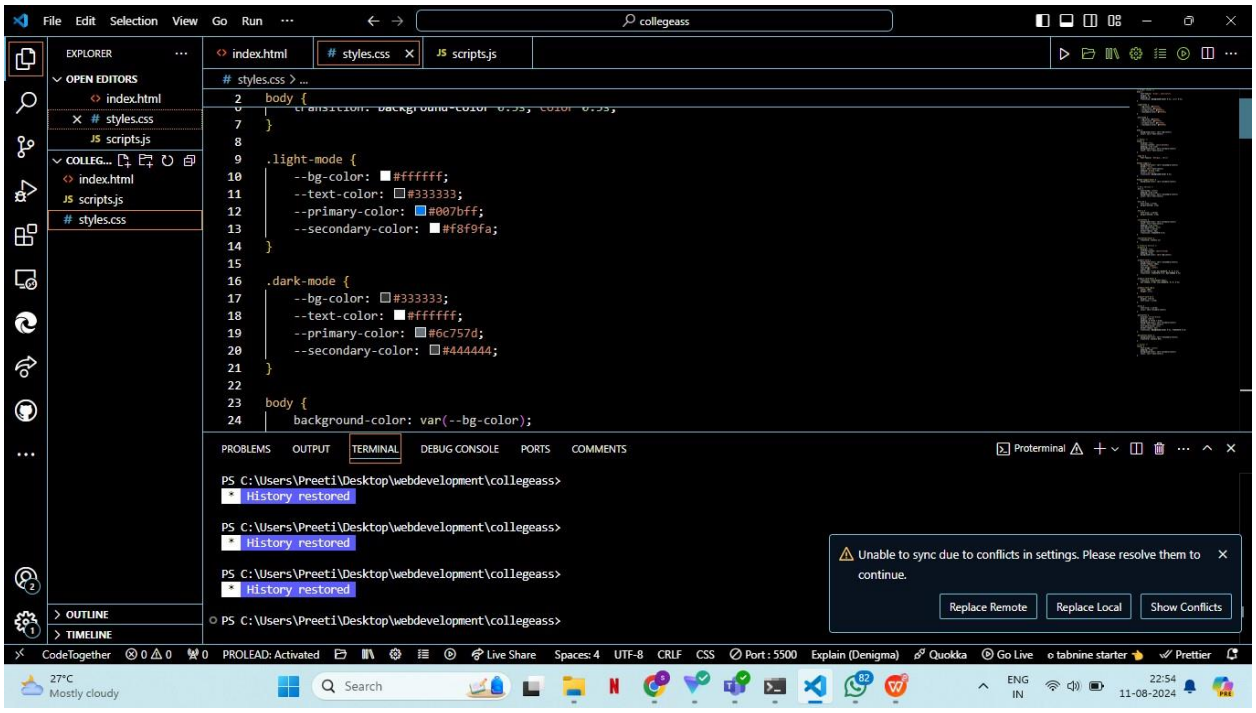
Saikarthik Iyer
T1341

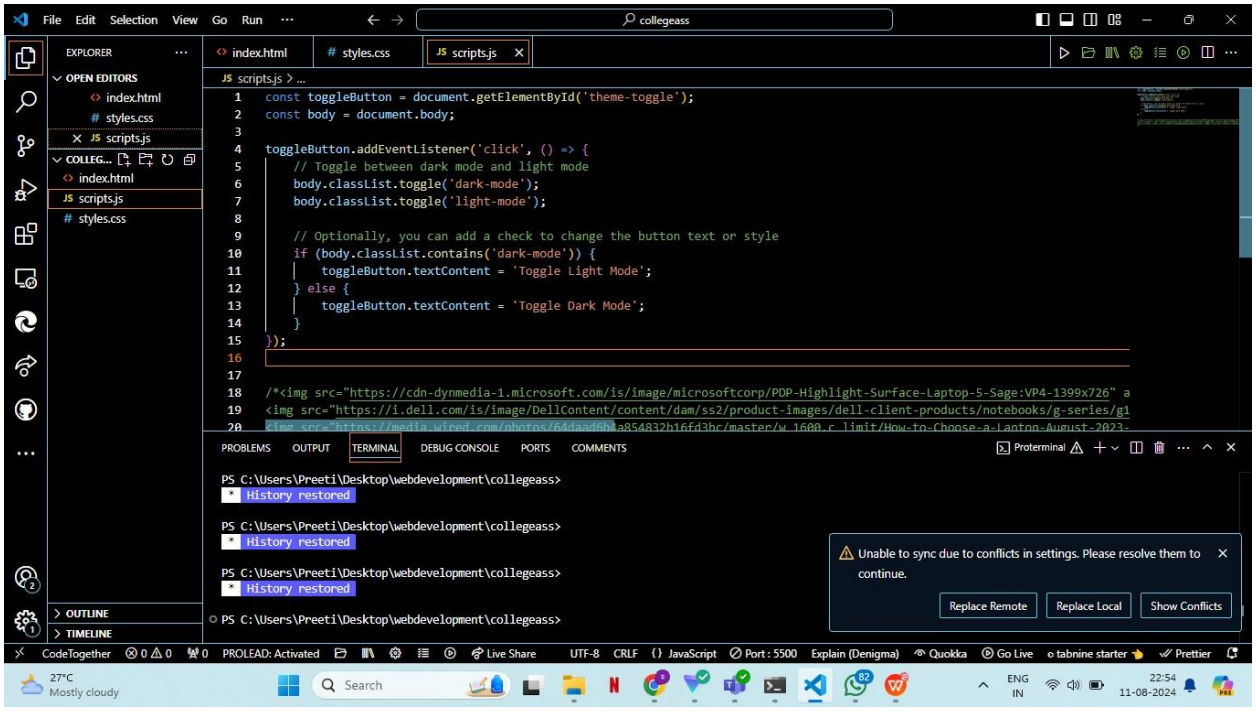
Assignment 3:

Aim:**Design a web page having the concept of the following**

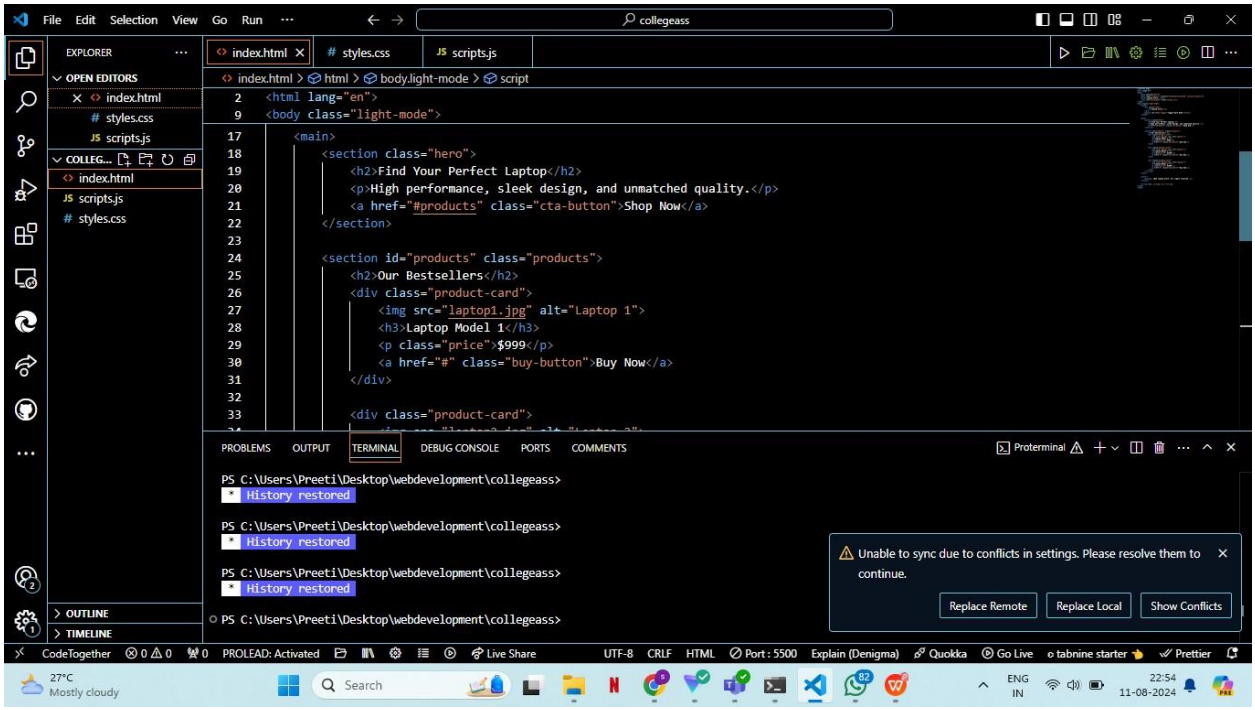
(Typography, color mode, Transitions, Transformations and Animations.)

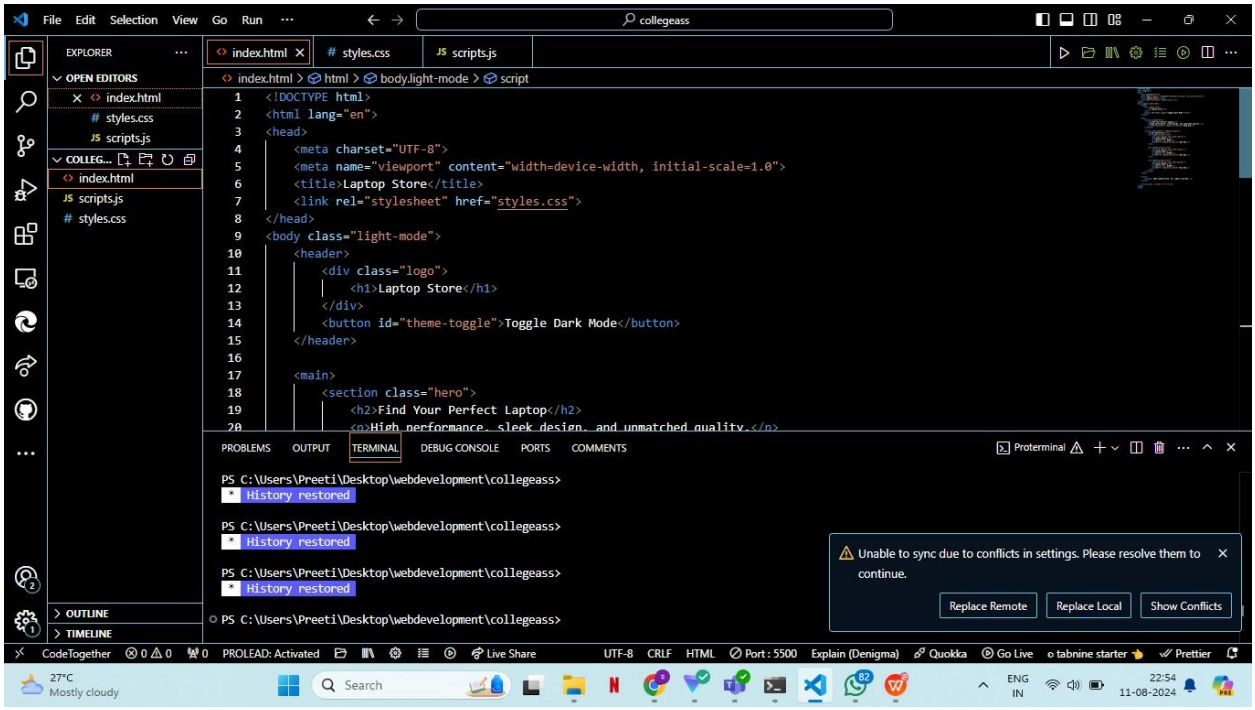
code:





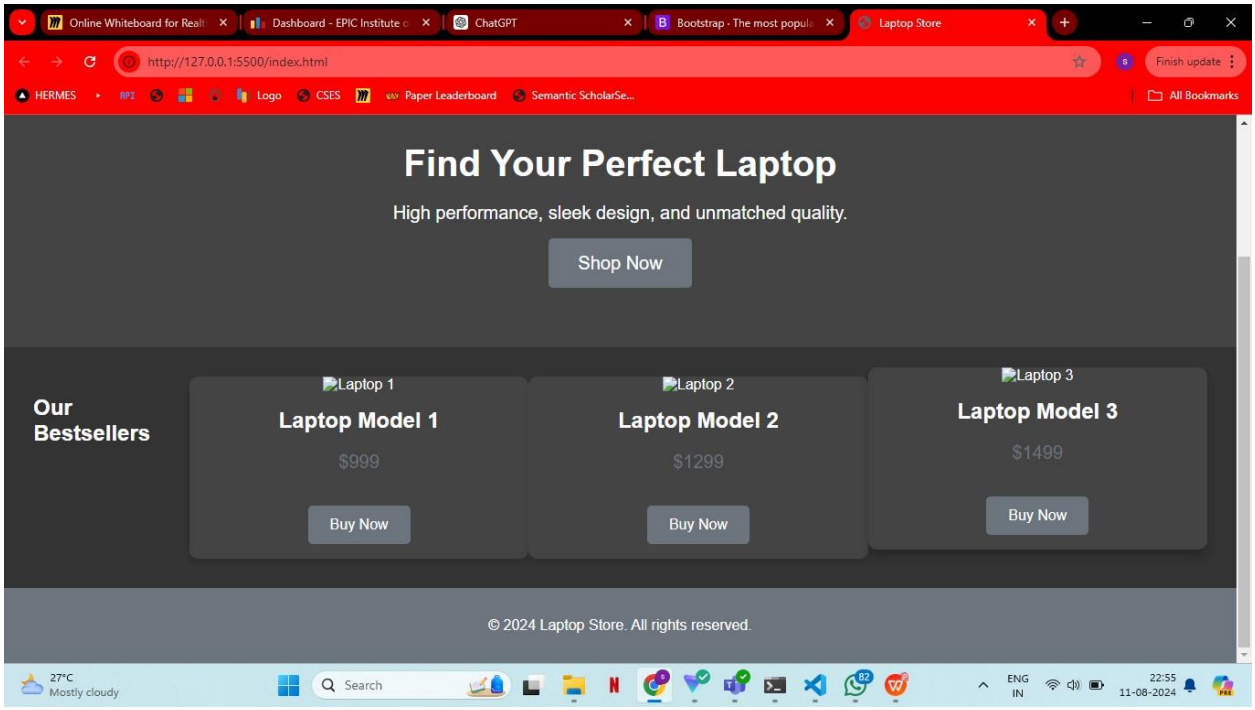
Saikarthik Iyer
T1341

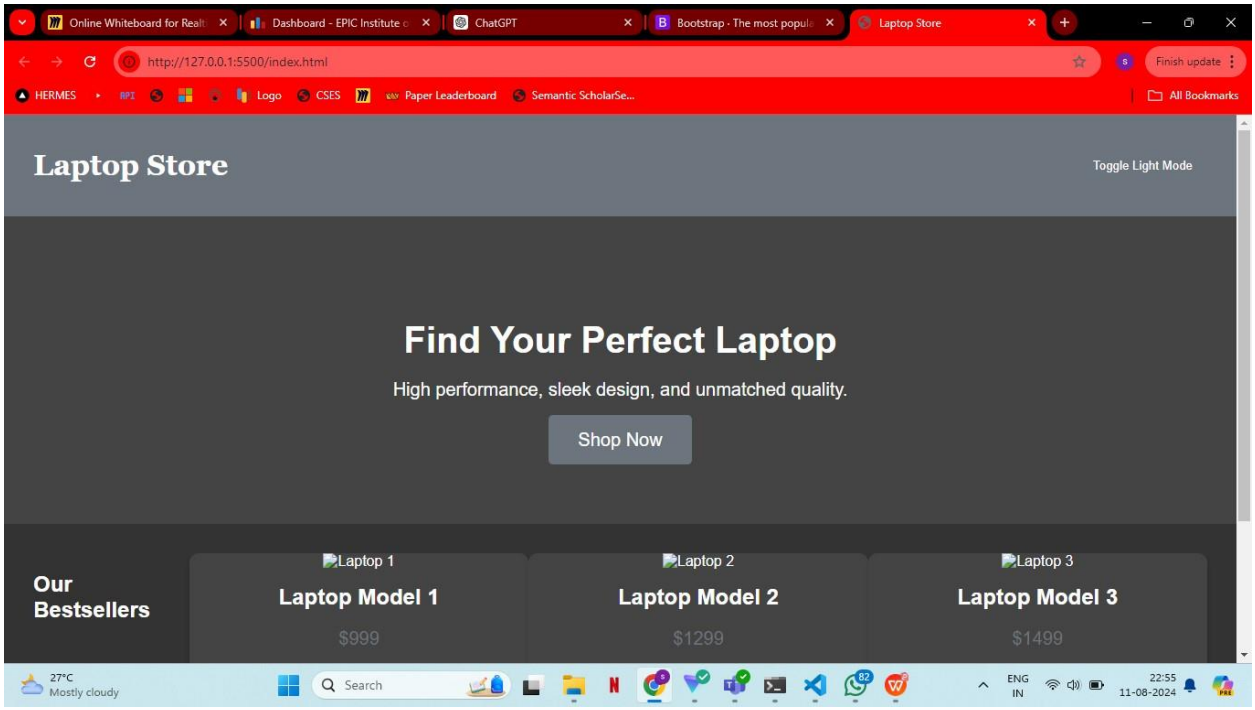




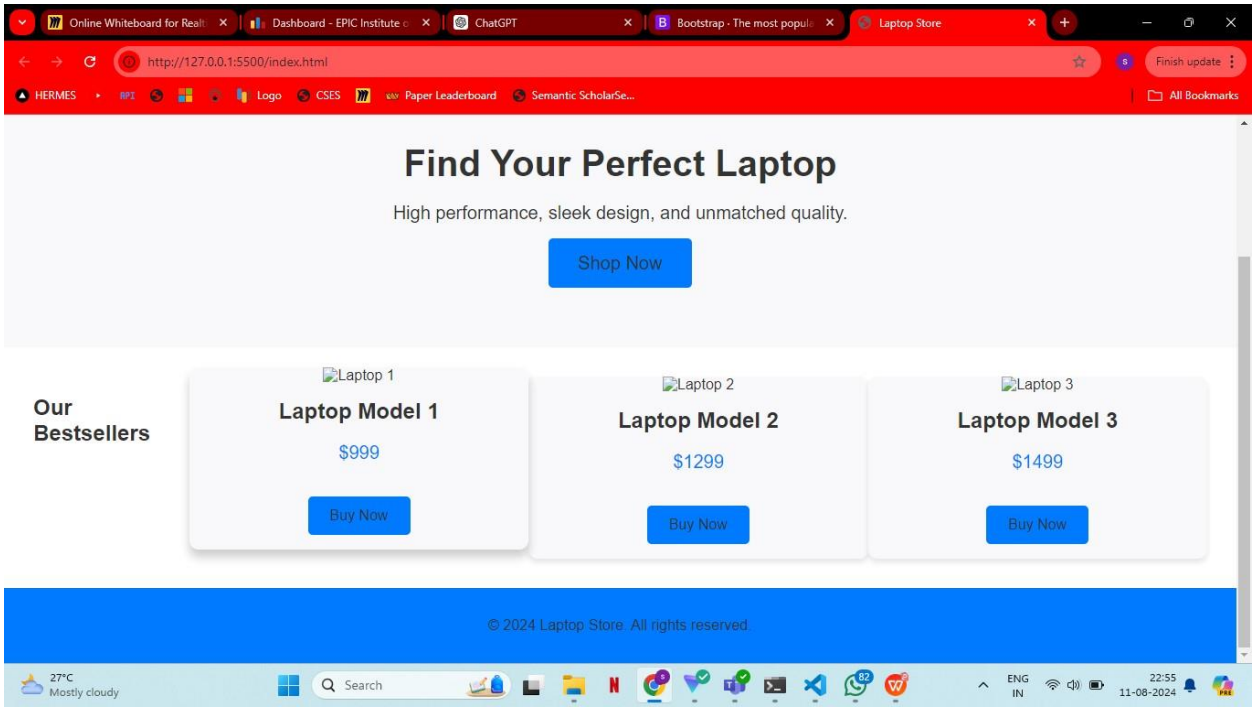
Output:

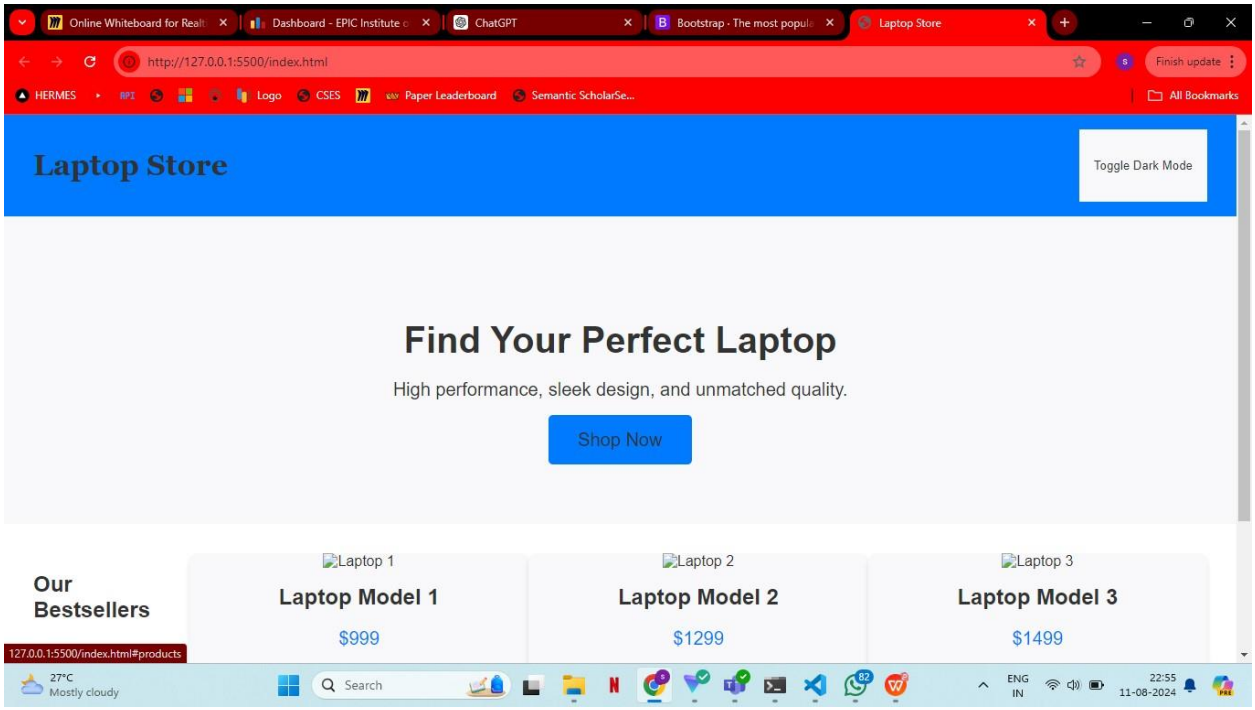
Saikarthik Iyer
T1341





Saikarthik Iyer
T1341





Conclusion:

Saikarthik Iyer
T1341

In conclusion, JavaScript was key in adding interactivity, enabling dark mode toggling, and creating engaging animations and transitions, making the web page dynamic and user-friendly.

Saikarthik Iyer
T13 54

Saikarthik Iyer
T1341

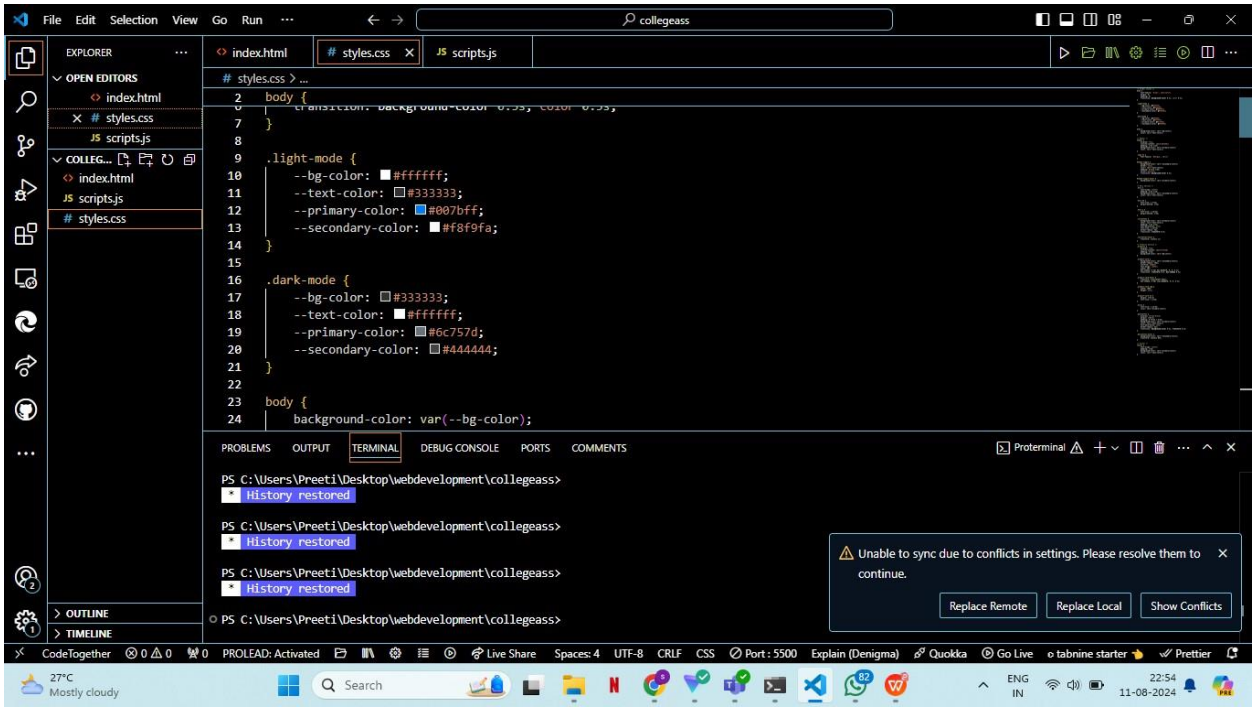
Assignment 4:

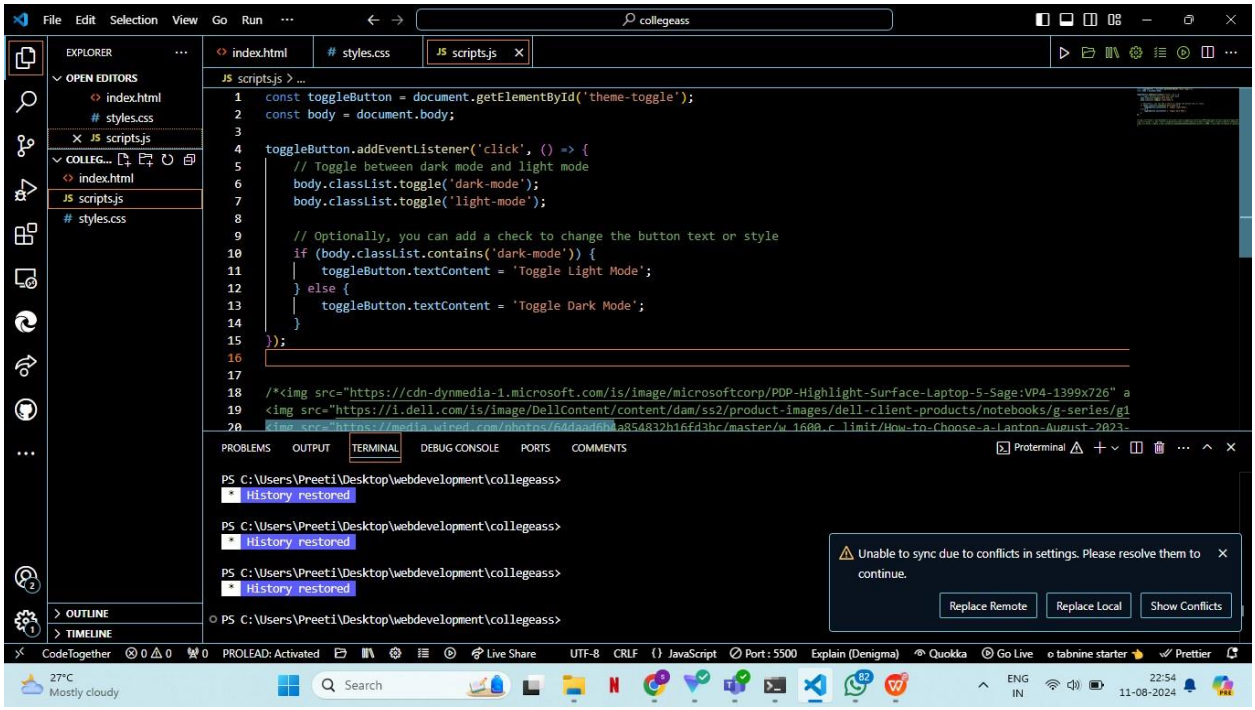
Aim:**Develop a web page using the Bootstrap framework.**

(Grid system, Forms, Button, Navbar, Breadcrumb, Jumbotron should be used.)

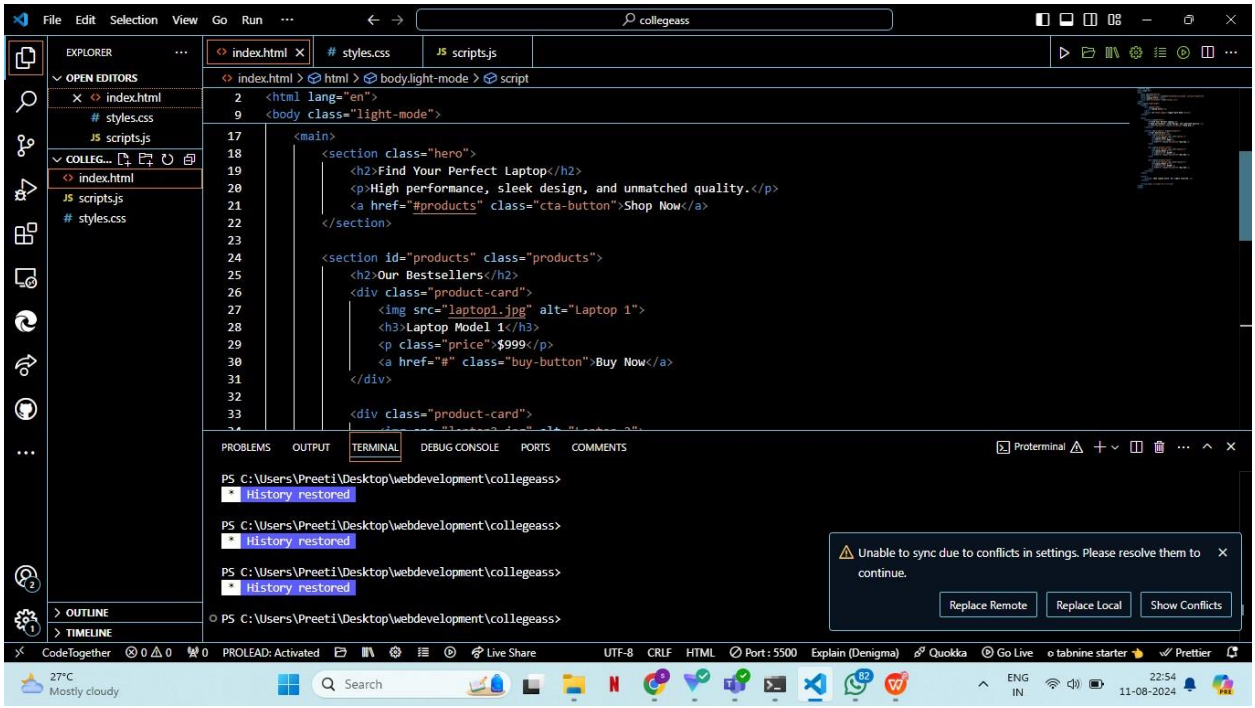
code:

Saikarthik Iyer
T1341



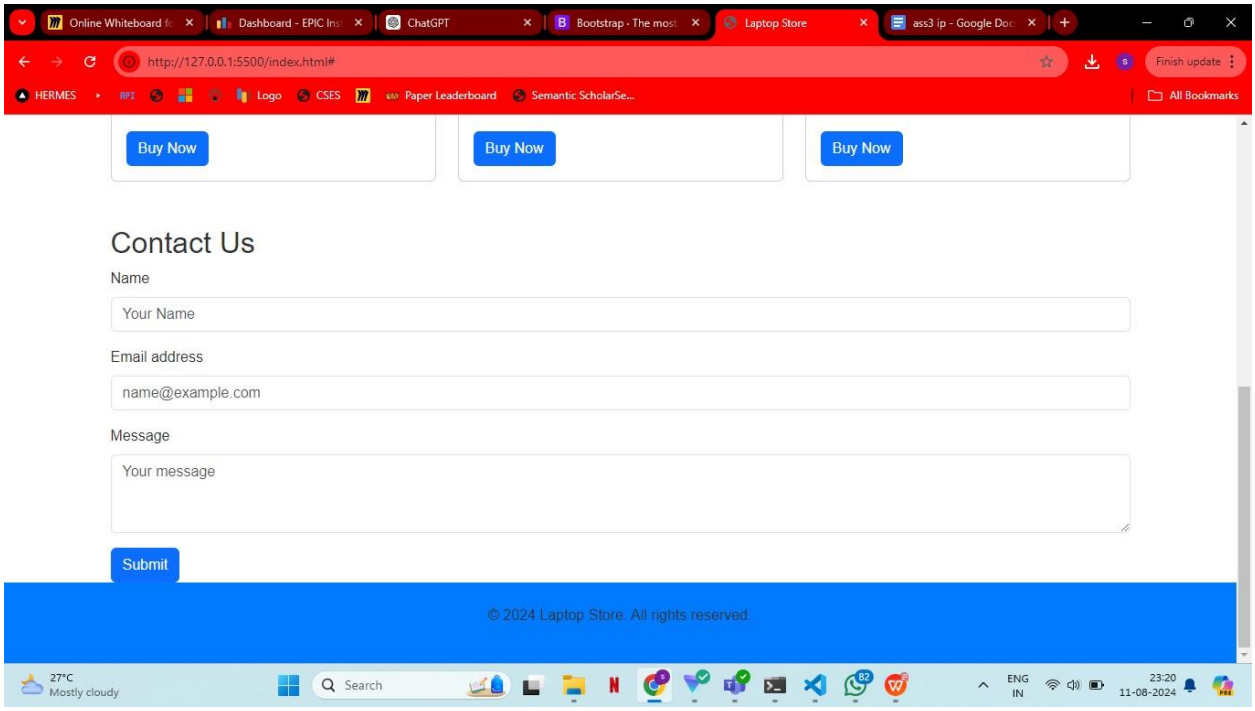


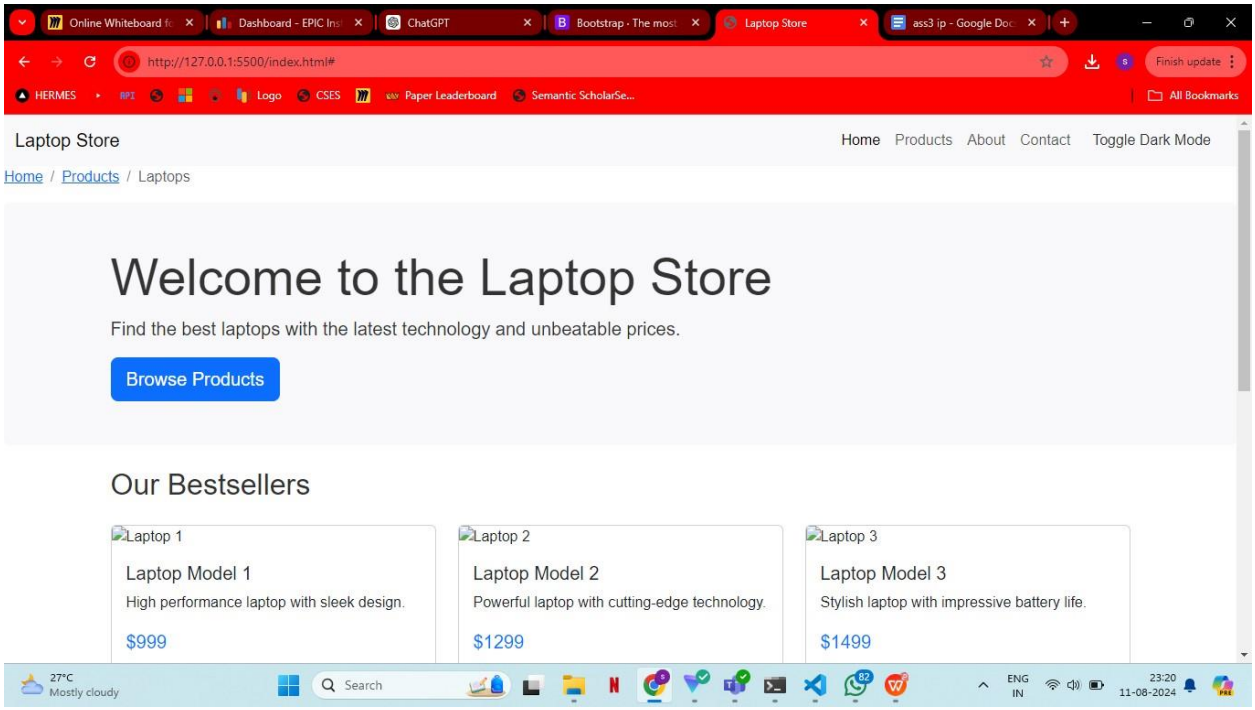
Saikarthik Iyer
T1341



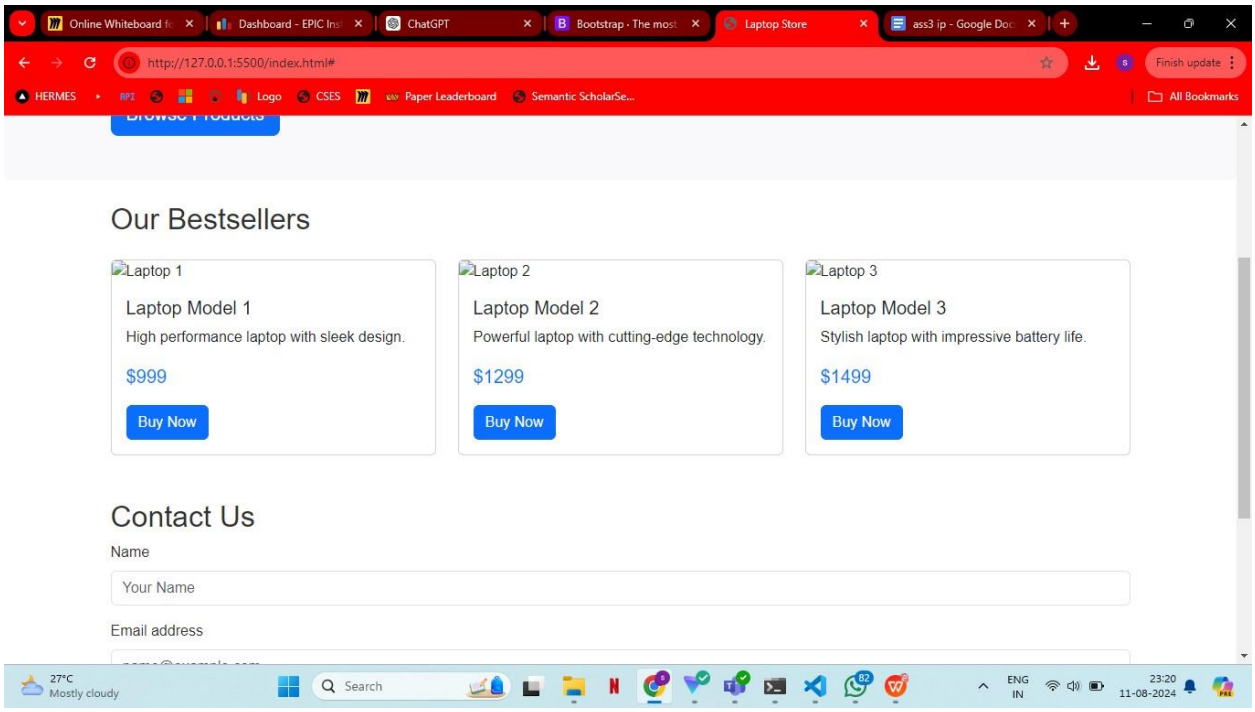
output:

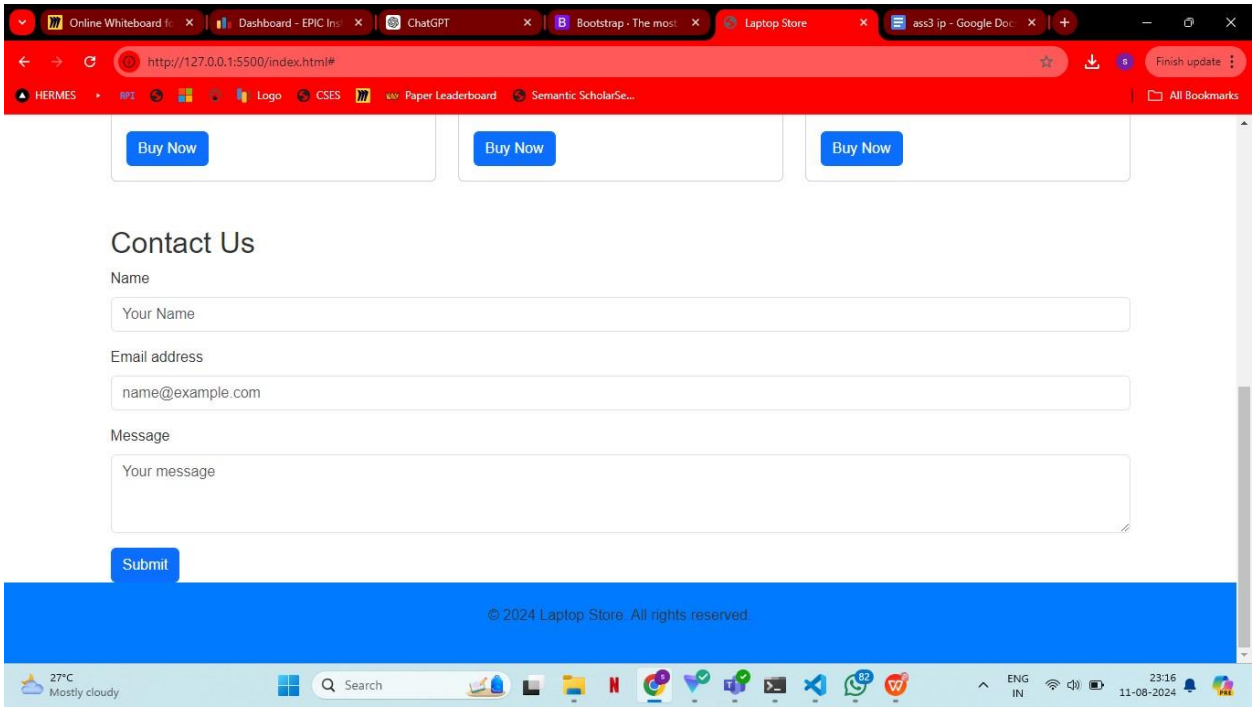
Saikarthik Iyer
T1341





Saikarthik Iyer
T1341





Conclusion:

Saikarthik Iyer
T1341

In conclusion, JavaScript was key in adding interactivity, enabling dark mode toggling, and creating engaging animations and transitions, making the web page dynamic and user-friendly.

Saikarthik Iyer
T1341

Saikarthik Iyer
T1341

Aim: Write a program in JavaScript (JS) to study conditional Statements, Loops, Functions, Inheritance, Iterators and Generators

Theory:

Here's some theoretical information on the key concepts to help write about "Write a program in JavaScript (JS) to study Conditional Statements, Loops, Functions, Inheritance, Iterators, and Generators."

1. **Conditional Statements**

Conditional statements allow you to control the flow of a program by executing different blocks of code depending on whether a condition is `true` or `false`. In JavaScript, common conditional statements include:

- `if` statement: Executes a block of code if a specified condition is `true`.
- `else` statement: Specifies a block of code to be executed if the condition in the `if` statement is `false`.
- `else if` statement: Used to specify a new condition if the first condition is false.
- `switch` statement: Evaluates an expression, matching the expression's value to a case and executing corresponding code.

Example:

```js
let age = 18; if
(age >= 18) {
  console.log("You are eligible to vote.");
} else { console.log("You are not eligible to
  vote.");
}
```

2. **Loops**

Loops are used to repeatedly execute a block of code as long as a specified condition holds true. JavaScript supports several types of loops:

- `for` loop: Repeats a block of code a certain number of times.
- `while` loop: Repeats a block of code as long as a condition is `true`.
- `do...while` loop: Executes a block of code once and then continues executing as long as the condition remains `true`.
- `for...of` and `for...in` loops: These iterate over arrays, objects, or other iterable data structures.

Example:

Saikarthik Iyer
T1341

```js
for (let i = 0; i < 5; i++) {
  console.log(i); // Prints 0 to 4
}
```

3. **Functions**
Functions in JavaScript are reusable blocks of code designed to perform a specific task. They can accept input through parameters and return values. JavaScript functions can be:
- **Function Declarations**: Declared using the `function` keyword.
- **Function Expressions**: Stored in a variable.
- **Arrow Functions**: A shorter syntax for function expressions introduced in ES6.

Example:
```js
function add(a, b) {
  return a + b;
}
let sum = add(3, 4);
console.log(sum); // Outputs 7
```

4. **Inheritance**
Inheritance in JavaScript is a mechanism where one object can inherit properties and methods from another. ES6 introduced the `class` syntax, which makes implementing inheritance easier.
- The `extends` keyword is used to create a child class that inherits from a parent class.
- The `super` keyword is used to call the constructor of the parent class.

Example:
```js
class Animal {
  constructor(name) {
    this.name = name;
  }
  speak() {
    console.log(`${this.name} makes a sound.`);
  }
}

class Dog extends Animal {
  speak() {
```

```
    console.log(`${this.name} barks.`);
  }
}

let dog = new Dog('Rex');
dog.speak(); // Outputs: Rex barks.
```

5. **Iterators**
Iterators are objects that provide a mechanism to traverse through a collection (such as arrays) one element at a time. An iterator has a `next()` method, which returns the next value in the sequence. Iterators are often used behind the scenes in `for...of` loops.

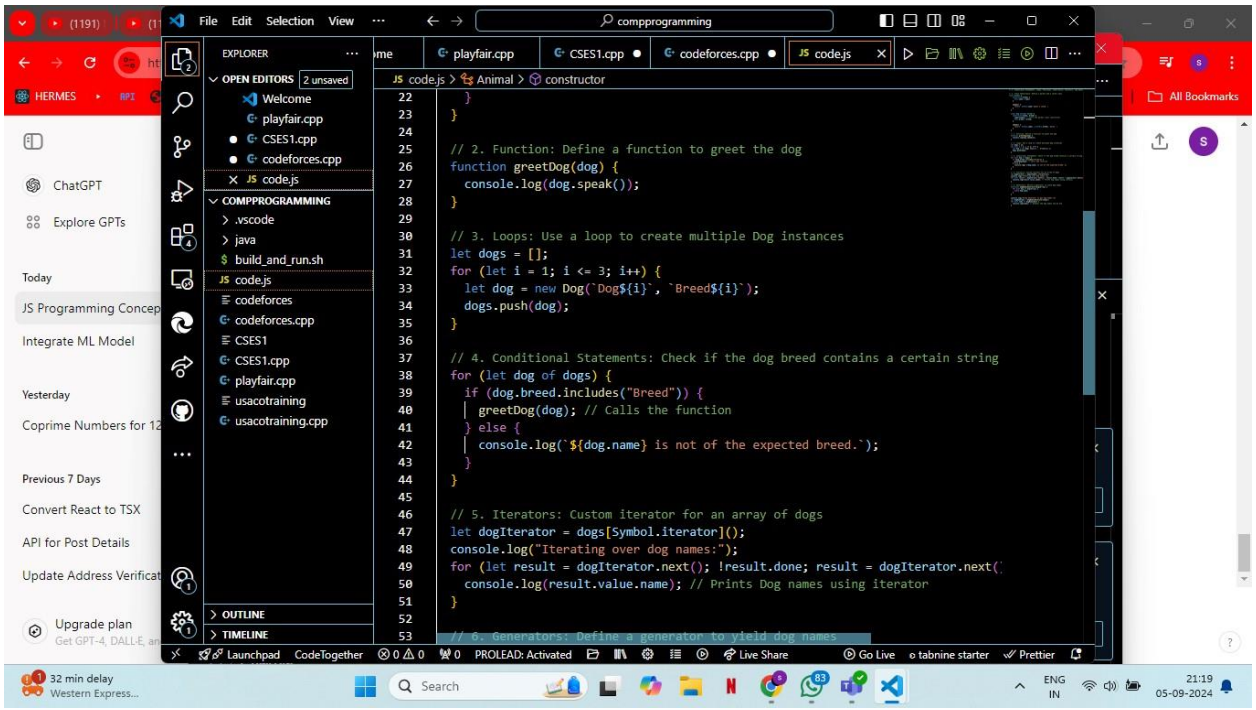
Example:
```js
let arr = [1, 2, 3]; let iterator =
arr[Symbol.iterator]();
console.log(iterator.next().value); // Outputs 1
console.log(iterator.next().value); // Outputs 2
```

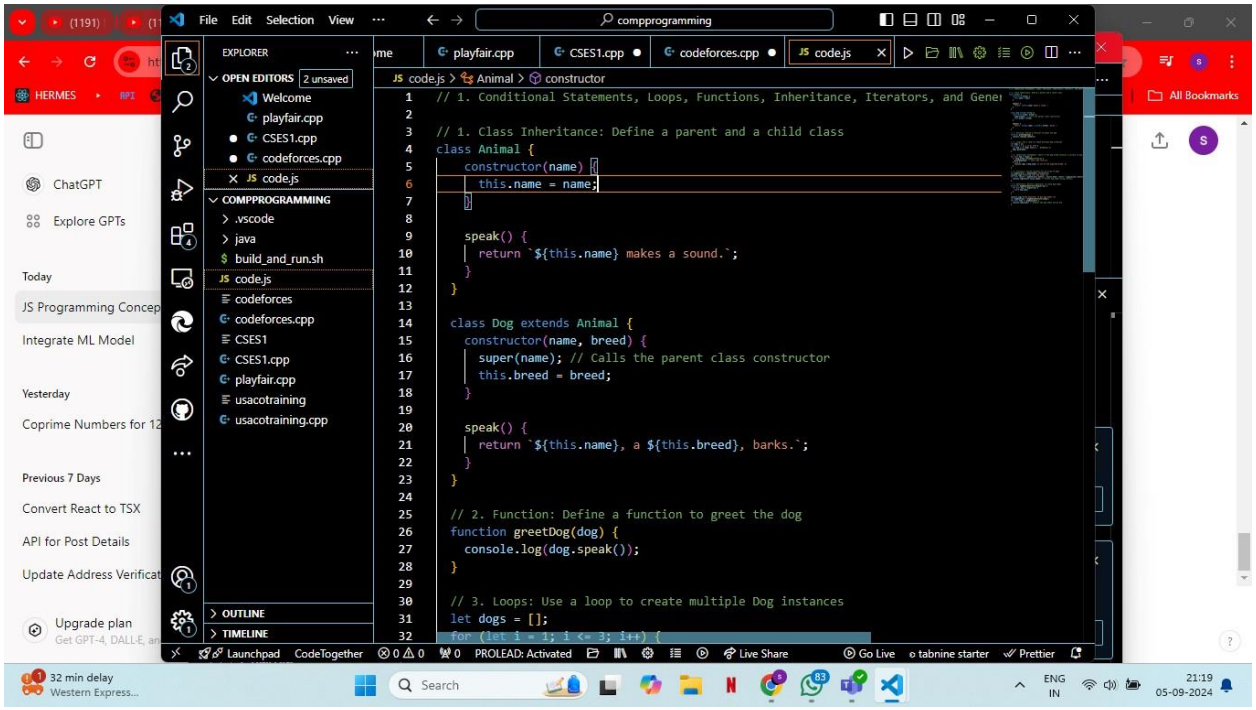
6. **Generators**
Generators are special functions in JavaScript that can be paused and resumed. They are defined using the `function*` syntax and can yield multiple values one at a time. Generators provide an elegant way to implement custom iterators.

Example:
```js
function* generatorExample() {
  yield 1;
  yield 2;
  yield 3;
}

let gen = generatorExample();
console.log(gen.next().value); // Outputs 1
console.log(gen.next().value); // Outputs 2
```

Saikarthik Iyer
T1341

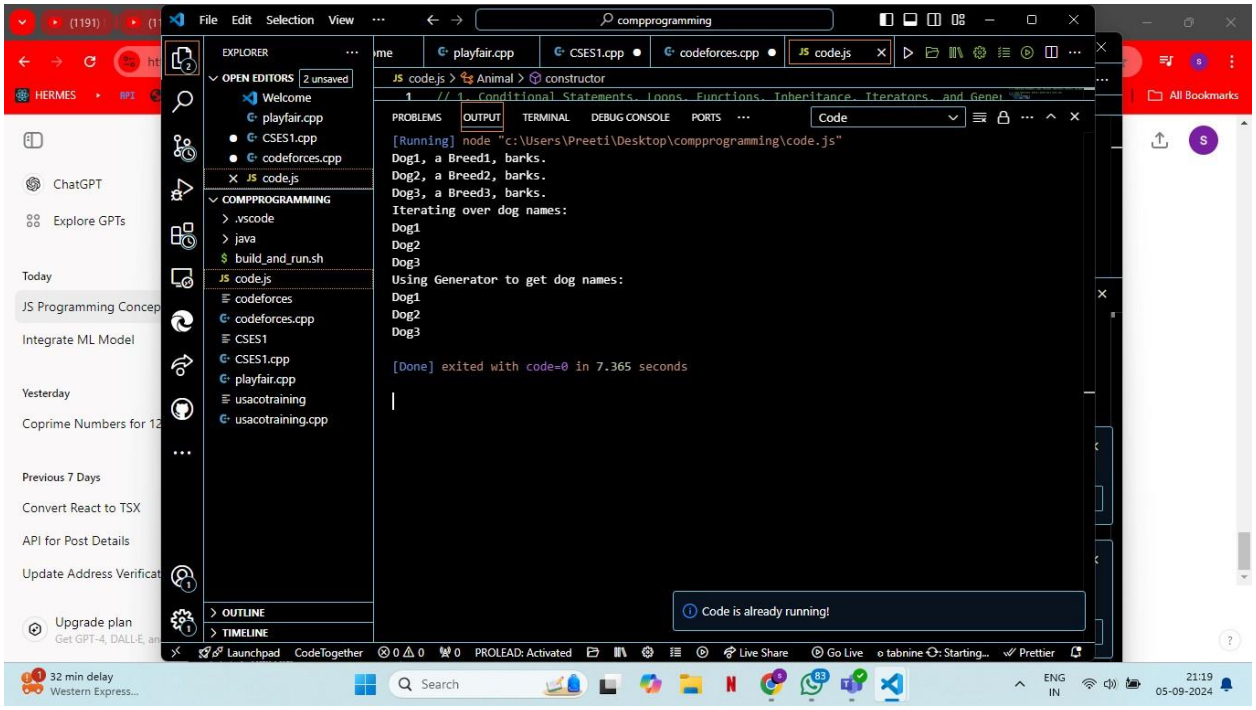
CODE:





OUTPUT:

Saikarthik Iyer
T1341




```

## Conclusion

In JavaScript, understanding these fundamental concepts—conditional statements, loops, functions, inheritance, iterators, and generators—is crucial for writing efficient and maintainable code. They form the building blocks for more advanced programming patterns and structures, allowing developers to handle complex logic and manage flow control in their applications effectively.

This theoretical overview gives a foundation to explore these concepts further and apply them in real-world JavaScript projects.
```

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 6

AIM: Write a program to study DOM and CSS Manipulations.

THEORY:

DOM (Document Object Model) manipulation and **CSS manipulation** are essential techniques used in web development to dynamically modify the structure and style of web pages.

DOM Manipulation:

DOM manipulation refers to the process of programmatically interacting with the HTML structure of a web page. It allows us to add, modify, or remove elements and their attributes, as well as respond to user interactions.

Methods:

document.getElementById(): Retrieves an element by its unique id attribute. document.querySelector(): Returns the first element that matches a specified CSS selector. document.createElement(): Creates a new HTML element. element.appendChild(): Appends a child element to the specified parent element. element.removeChild(): Removes a child element from the specified parent element. element.innerHTML: Gets or sets the HTML content of an element.

CSS Manipulation:

CSS manipulation involves changing the appearance or style of HTML elements using JavaScript. It enables us to modify properties like color, size, position, and visibility dynamically.

Methods:

querySelector: Selects and returns the first element in the document that matches a specified CSS selector. querySelectorAll: Selects and returns all elements in the document that match a specified CSSselector.

Both DOM and CSS manipulation are crucial for creating interactive and dynamic web applications. While DOM manipulation focuses on changing the structure and content of the page, CSS manipulation focuses on altering the visual appearance and layout. These techniques are commonly used in conjunction to create responsive and user-friendly web experiences.

OUTPUT:

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 6

I am Shreya Kamath and I'm 19 years old.

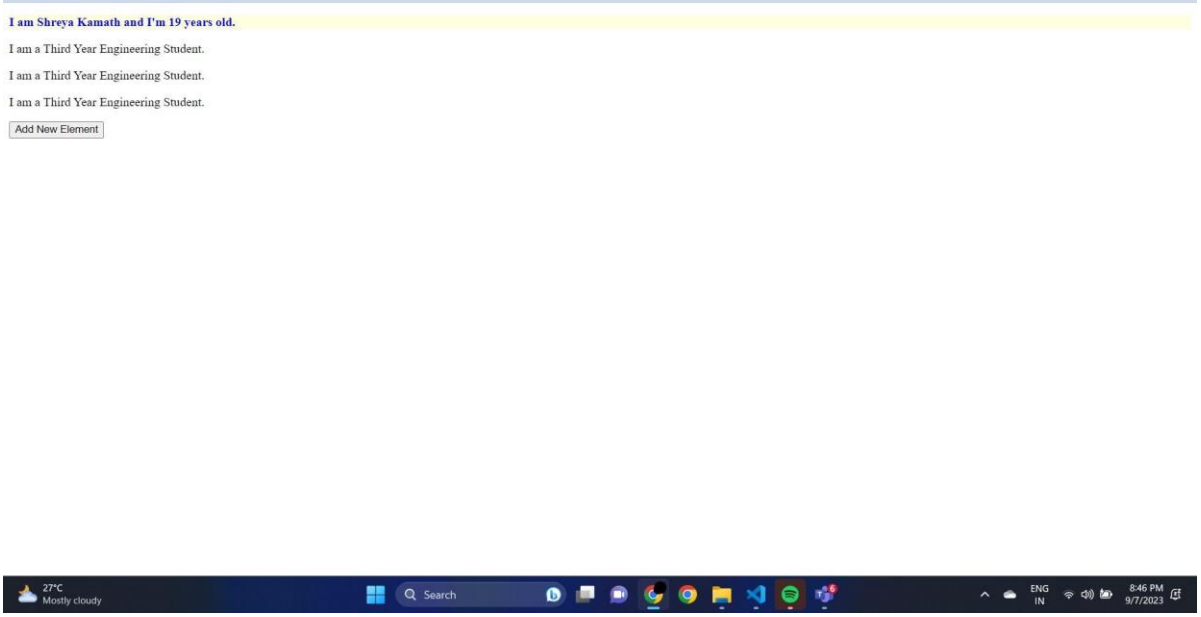
Add New Element

I am Shreya Kamath and I'm 19 years old.

I am a Third Year Engineering Student.

Add New Element

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 6



CODE:

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Manipulation</title>
  <style>
    .highlight { color:
      blue; font-
      weight: bold;
    }
  </style>
</head>
<body>
  <div id="container">
    <p class="intro">Hi, I am Shreya Kamath.</p>
  </div>

  <button id="addButton">Add New Element</button>

  <script> const container =
    document.getElementById("container");

    // Creating a new element const age =
    document.createElement("p");
    age.textContent = "I am 19 years old.";
```

```
    container.appendChild(age);


    // Removing the initial paragraph const initialPara =
    container.querySelector(".intro");
    container.removeChild(initialPara);

    // Replacing the paragraph const replacedPara =
    document.createElement("p");
        replacedPara.textContent = "I am Shreya Kamath and I'm 19 years
old."; container.replaceChild(replacedPara,
    age);

    // CSS manipulation using querySelector const
    replacedWithCSS = container.querySelector("p");
    replacedWithCSS.classList.add("highlight");

    //     Adding     background     color     to     the     paragraph
    replacedWithCSS.style.backgroundColor = "lightyellow";

    // Function to add a new element function addNewElement() {
    const newPara = document.createElement("p"); newPara.textContent
    = "I am a Third Year Engineering Student.";
    container.appendChild(newPara);
    }

    // Adding event listener to the button const addButton
    = document.getElementById("addButton");
    addButton.addEventListener("click", addNewElement);
  </script>
</body>
</html>
```

CONCLUSION:

Hence, by using various methods like document.createElement(), document.getElementById(), querySelector() and querySelectorAll(), I have learned the fundamentals and executed a program to perform DOM and CSS Manipulation.

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 7

AIM: Write a program to implement the concept of forms and events in ReactJS.

LAB OUTCOME:
LO5: Construct Front-end applications using React.

THEORY:

In React, **forms** are a crucial part of building interactive web applications that collect and manage user input. Forms allow users to input data, such as text, selections, and more, and then submit that data to a server for processing or use within the application itself. React provides a powerful and flexible way to create and manage forms, making it easier to handle user interactions and data management efficiently.

Implementing Forms in React:
Form Element: The form typically starts with an HTML <form> element to wrap the form components. This element manages form submission and provides a way to handle user input.

Input Fields: React provides various input field components, such as <input>, <textarea>, and <select>, which can be controlled or uncontrolled. Controlled inputs are managed by maintaining a piece of state for each input field, while uncontrolled inputs rely on the DOM directly.

State Management: To create controlled inputs, a state is maintained for each input field. This allows tracking user input and controlling the field's value.

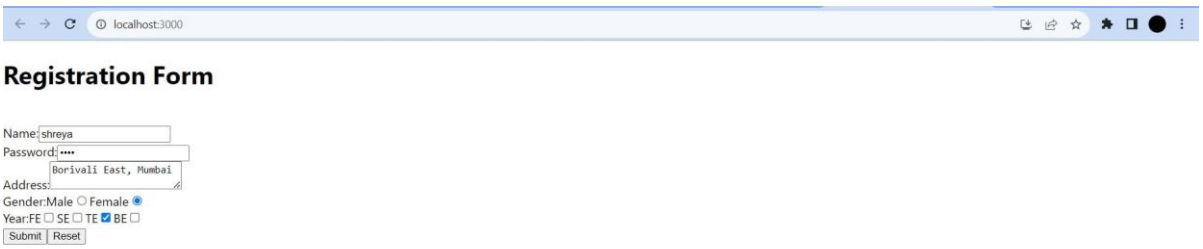
Event Handling: Event handlers, such as onSubmit, are defined to capture user input and update the corresponding state when it changes. This ensures that the user interface and the state are always in sync.

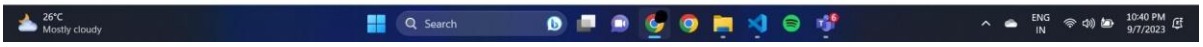
Submit Handling: When the user submits the form, the submission is handled by preventing the default browser behaviour using preventDefault() and then sends the form data to a server or performs actions within the application.

React offers various form components and tools to enhance form functionality, including options like radio buttons, checkboxes, validation, and form libraries like Formik and Redux-Form. These tools make form development more efficient and help manage complex forms with ease. Additionally, controlled and uncontrolled components based on specific needs and preferences can be used, providing flexibility in how form data and user interactions are handled.

OUTPUT:

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 7





CODE:

```
//Form.js import React, { Component }

from 'react';

class Form extends Component
  { constructor(props)
  {super(props); this.state =
  {
```

```
      username: '',
      password: '',
      address: '',
      isLoggedIn: false,
      showPopup: false,
      gender: '',
    };
  }

  handleUsernameChange = (event) => { const input =
    event.target.value.replace(/[^A-Za-z]/g, '');
    this.setState({ username: input });

  };

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

  handleAddressChange = (event) =>
    {   this.setState({   address:
    event.target.value });

  handleUsernameKeyPress = (event) =>
    {const key = event.key; if (!/[a-zA-
    Z]/.test(key)) {
    event.preventDefault(); this.setState({
    showPopup: true }); setTimeout(() => {
    this.setState({ showPopup: false });
      }, 1500);
    }
  };

  handleGenderChange = (event) =>
    {   this.setState({   gender:
    event.target.value });

  handleSubmit = (event) => {
    event.preventDefault();

      if (this.state.username === 'shreya' && this.state.password ===
'sk0912') {
```

```
      this.setState({ isLoggedIn: true });
    }
  };

  reset = (event) => { this.setState({ username: '', password: '',
address: '', gender: '', isLoggedIn: false });
  };

  render() { if (this.state.isLoggedIn) { return
    <div>Welcome, {this.state.username}!</div>;



    } else
      { return
      (
        <form onSubmit={this.handleSubmit}>
          <div>
            <h1>Registration Form</h1>
            <br></br>
            <label>Name:</label>
            <input type="text"
              value={this.state.username}
              onChange={this.handleUsernameChange}
              onKeyPress={this.handleUsernameKeyPress}
            />
              {this.state.showPopup && <div className="popup">Enter only
Alphabets.</div>}
          </div>
          <div>
            <label>Password:</label>
            <input type="password"
              value={this.state.password}
              onChange={this.handlePasswordChange}
            />
          </div>
          <div>
            <label>Address:</label>
            <textarea value={this.state.address}
              onChange={this.handleAddressChange}
```

```
            />
          </div>
          <div>
            <label>Gender:</label>
            <label>Male</label>
            <input type="radio" value="male"
              checked={this.state.gender ===
              'male'}
              onChange={this.handleGenderChange}
            />
            <label>Female</label>
            <input

              type="radio" value="female"
              checked={this.state.gender ===
              'female'}
              onChange={this.handleGenderChange}
            />
          </div>
          <div>
            <label>Year:</label>
            <label>FE</label>
            <input type="checkbox" />
            <label>SE</label>
            <input type="checkbox" />
            <label>TE</label>
            <input type="checkbox" />
            <label>BE</label>
            <input type="checkbox" />
          </div>
          <button type="submit">Submit</button>
          <button onClick={this.reset}>Reset</button>
        </form>
      );
    }
  } } export
default Form;
```

```
//app.js import React

from 'react';
```

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 7

```
import RegiForm from './Form.js';

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

CONCLUSION:

I have understood the fundamentals and created a registration form in ReactJS which uses various form elements like textbox, textarea, check box, radio button, submit button and reset button handling onSubmit, onClick and keyDown events.

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 8

AIM: Write a program to implement the concept of React Hooks.

LAB OUTCOME:
LO5: Construct Front-end applications using React.

THEORY:
Hooks are a feature introduced in React to allow developers to use state and other React features in functional components. Prior to hooks, state management and other React features were primarily available in class components. Hooks enable functional components to manage state, side effects, and other React features without the need for class components.

1. **useState:**

- useState is a hook that allows us to manage state in a functional component.
- It returns an array with two elements: the current state value and a function to update that value.
- We provide an initial state value as its argument.

Example:

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

2. **useEffect:**

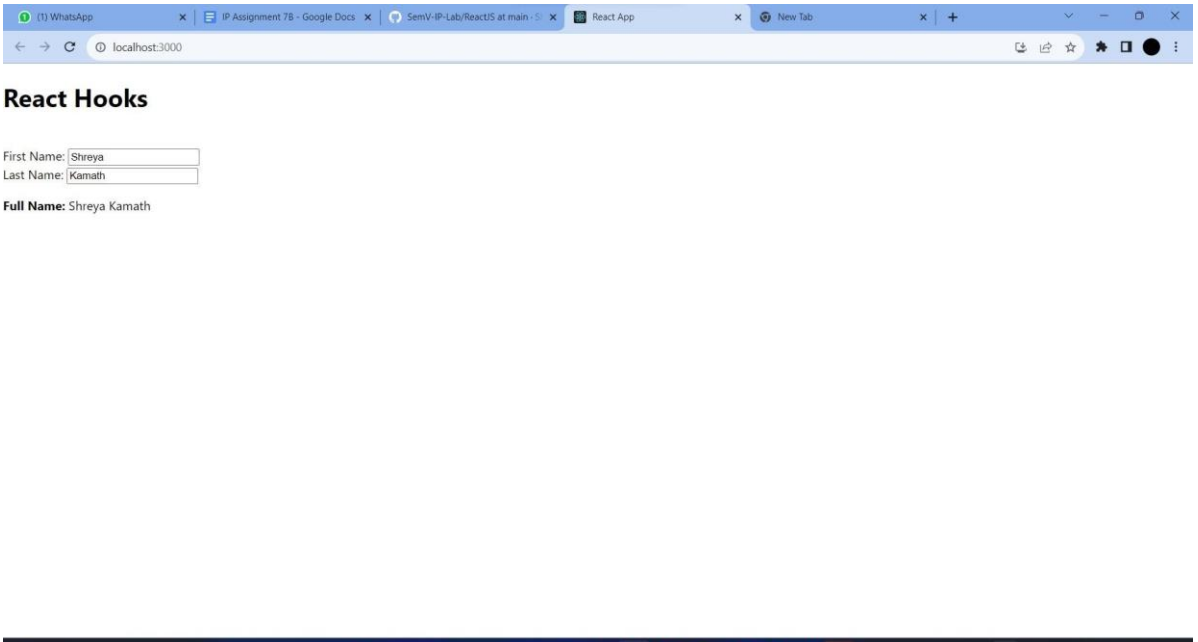
- useEffect is a hook for handling side effects in functional components.
- It takes two arguments: a function containing the code for the side effect and an optional array of dependencies.
- The function is called after every render, and it can perform tasks like data fetching, DOM manipulation, or setting up subscriptions.
- The optional array of dependencies specifies which values from the component's scope should be tracked. When any of these dependencies change, the effect is re-run.
- To replicate the behaviour of componentDidMount, we pass an empty dependency array ([]). For componentDidUpdate, we specify the dependencies that should trigger the effect.

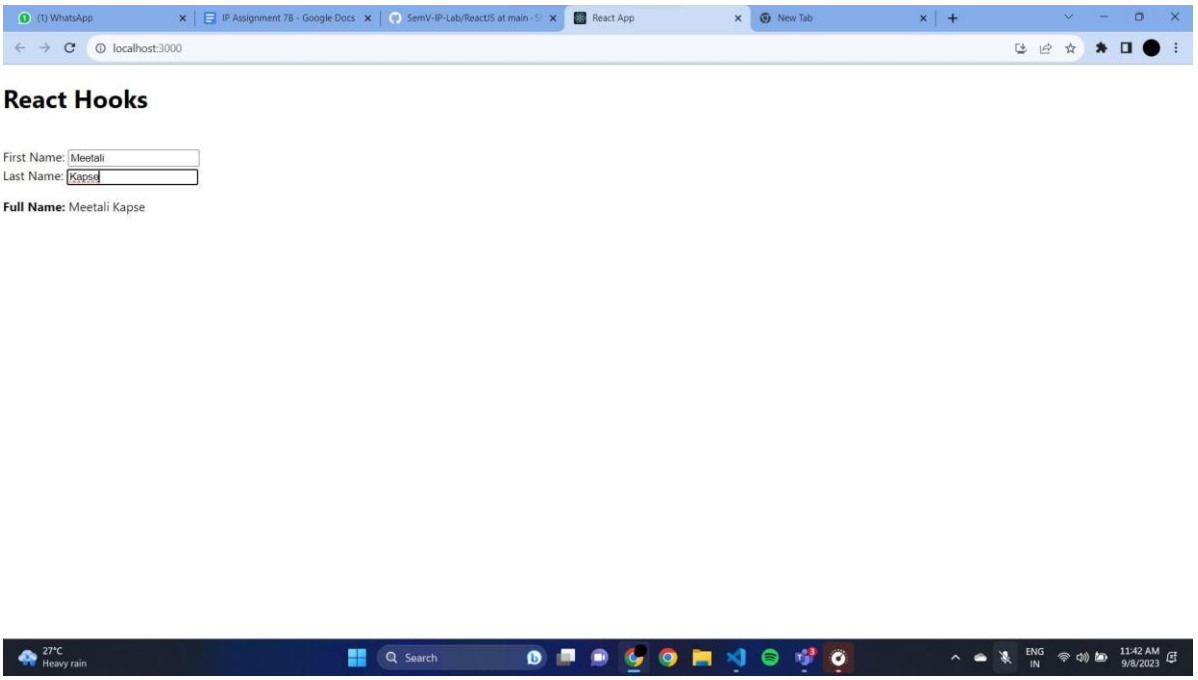
Example:

```
useEffect(() => {
  // Code for the side effect },
[dependency1, dependency2]);
```

OUTPUT:

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 8





CODE:

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 8

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


function App() {


  const [firstName, setFirstName] = useState('Shreya');
  const [lastName, setLastName] = useState('Kamath');
  const [fullName, setFullName] = useState('');
```

```
  useEffect(() =>
    { setFullName(`${firstName}
    ${lastName}`);
  }, [firstName, lastName]);

  return (
    <div>
        <h1>React Hooks</h1>
        <br></br>
      <div>
        <label>First Name: </label>
        <input type="text" value={firstName}
        onChange={(e) => setFirstName(e.target.value)}
        />
      </div>
      <div>
        <label>Last Name: </label>
        <input type="text" value={lastName}
        onChange={(e) => setLastName(e.target.value)}
        />
      </div>

      <div>
        <p><b>Full Name:</b> {fullName}</p>
      </div>
    </div>
  ); } export
default App;
```

CONCLUSION:

Hence, I have understood the basic fundamentals of Hooks in ReactJS and implemented a program to demonstrate the same.

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 9

AIM: Write a Javascript program to implement the concept of promises(callback) and fetch(client server communication).

LAB OUTCOME:
LO4: Use JavaScript to develop interactive web pages.

THEORY:

Promises in JavaScript offer a structured approach for handling asynchronous operations, aiding in maintaining clean and readable code. Promises represent a future event or value that may not be immediately available. They transition through three states: pending, fulfilled, and rejected.

Within a Promise, an asynchronous task is encapsulated, which could be tasks such as fetching data from an external API, reading files, or any process that takes time. Promises initiate in the pending state and are resolved with a result or rejected with an error upon completion.

Promises provide two primary methods:
1. ‘then()’: This method manages the result when a Promise is fulfilled, enabling the chaining of multiple `then` methods to execute a series of tasks sequentially.

2. ‘catch()’: For error handling when a Promise faces rejection, the `catch` method is employed. It's usually placed at the end of a chain of `then` methods to capture and manage potential errors.

3. ‘fetch()’: used to make network requests, typically to retrieve data from a specified URL

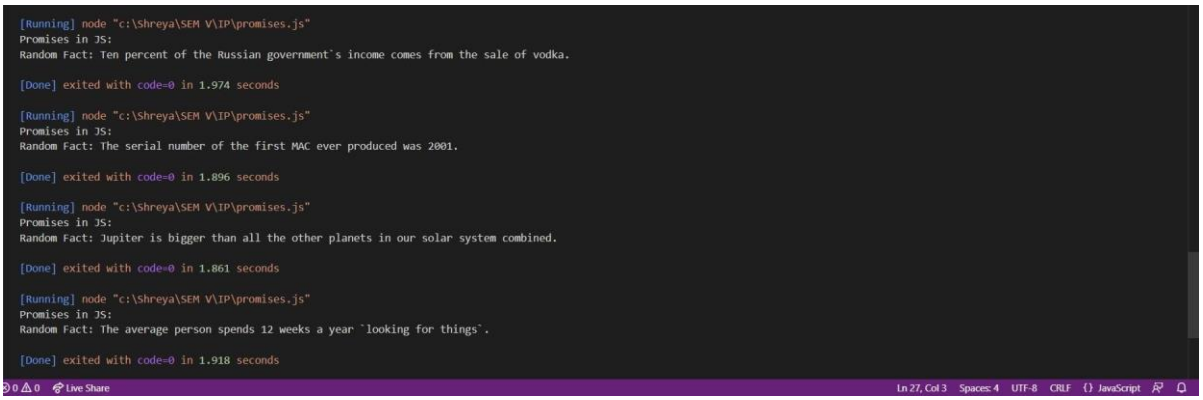
Promises enhance the readability and maintainability of asynchronous code, offering an organised structure. They are especially valuable when combined with the ‘fetch()’ method to retrieve data from external APIs, commonly used in web development for tasks like making network requests or interfacing with libraries like Axios.

CODE:
```
console.log("Promises in JS:") function
fetchRandomFact() { return new
Promise((resolve, reject) =>
      { fetch('https://uselessfacts.jsph.pl/random.json?language=en
      ')
        .then((response) =>
          {if (!response.ok)
          { throw new Error('Error!');
          }
          return response.json();
        })
        .then((data) =>
          { resolve(data.text)
          ;
        })
        .catch((error) =>
          {reject(error);
```

```
});
```

```
  });
}

fetchRandomFact()
  .then((fact) =>
    {  console.log('Random  Fact:',
    fact);
  })
  .catch((error)   =>   {   console.error('Error
  fetching text:', error); });
```

OUTPUT:



CONCLUSION:

In summary, I have grasped the fundamental concept of promises in JavaScript, and executed a program to generate facts that have been fetched from an external API. This knowledge aligns with the aim of using JavaScript to develop interactive web applications, enhancing practical web development skills.

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 10

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

LAB OUTCOME:
LO6: Construct back end applications using Node.js/Express.

THEORY:
File handling in Node.js involves working with the file system to perform various operations on files, such as creating, reading, updating, renaming, appending, and deleting files. Node.js provides a built-in module called fs (file system) that allows you to interact with the file system in a non-blocking, asynchronous way.

1. **Creating a File:**

Use the fs.writeFile() method to create a new file. You can specify the filename, content, and an optional callback function to handle errors.

2. **Reading from a File:**

The fs.readFile() method reads data from an existing file. You can specify the filename, encoding (e.g., 'utf-8' for text files), and a callback function to process the data.

3. **Writing to a File:**

Use the fs.writeFile() method again to overwrite the content of an existing file. This method will create the file if it doesn't exist.

4. **Renaming a File:**

The fs.rename() method allows you to change the name of a file. This operation is useful for moving or renaming files within the file system.

5. **Appending Data to a File:**

To add data to an existing file without overwriting its content, you can use the fs.appendFile() method. This is commonly used for log files and data collection.

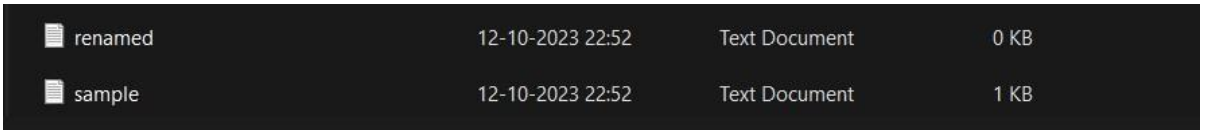
6. **Deleting a File:**

The fs.unlink() method is used to remove a file from the file system. It's important to check whether the file exists before attempting to delete it to avoid errors.

File handling in Node.js is asynchronous, meaning that these operations don't block the execution of your program. They take callbacks as arguments, which are executed when the file operation is completed. Proper error handling is essential to catch and respond to any issues during file operations.

OUTPUT:

Saikarthik Iyer / IP Lab / T13 / Roll No. 41
Assignment 10




```
PS C:\Users\HP> node filehandling.js
File created successfully.
File content: Hello, World!
File updated successfully.
File renamed successfully.
Data appended to the file.
File deleted successfully.
PS C:\Users\HP>
```

CODE:

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

// Create a file fs.writeFile('sample.txt', 'Hello,
World!', (err) =>
  { if (err) {
    console.error('Error creating the file:', err);
  } else {
    console.log('File created successfully.');

    // Read data from the file fs.readFile('sample.txt',
    'utf-8', (readErr, data) =>
      { if (readErr) { console.error('Error reading the
        file:', readErr);
      } else {
        console.log('File content:', data);

        // Write data to the file fs.writeFile('sample.txt', 'Updated
        content', (writeErr) =>
          { if (writeErr) { console.error('Error updating the
            file:', writeErr);
          } else {
            console.log('File updated successfully.');

            //    Rename    the    file    fs.rename('sample.txt',
            'renamed.txt', (renameErr) =>
              { if (renameErr) { console.error('Error renaming the
              file:', renameErr); } else {
```

```
                console.log('File renamed successfully.');

                // Append data to the file fs.appendFile('renamed.txt',
                     '\nAppended content',
(appendErr) => { if (appendErr) { console.error('Error appending
                  data:', appendErr);
                  } else { console.log('Data appended to the
                    file.');

                    //   Delete   the   file   fs.unlink('renamed.txt',
                    (deleteErr)     =>     {     if     (deleteErr)     {
                    console.error('Error deleting the file:',
deleteErr);
                      } else {
                        console.log('File deleted successfully.');
                      }
                    });
                  }
                });
              }
            });
          }
        });
      }
    });
  }
});
```

CONCLUSION:

Hence, by using various file handling functions such as fs.unlink(), fs.appendFile(), fs.readFile() and more, I have implemented a program to demonstrate file handling in NodeJS.

Saiparthiv Iyer
T13 41

Written Assignment - 1

1 Explain Promises with an example

⇒ A promise in Javascript is an object that represents the eventual completion (or failure) of an asynchronous operation and its resulting value. Promises are used in to handle asynchronous operations, providing a cleaner and more manageable way to deal with callbacks and avoid callback hells.

A promise can be in one of these 3 states :-

1. Pending - The initial state, neither fulfilled nor rejected
2. Fulfilled - The operation completed successfully and promise a resulting value.
3. Rejected - The operation failed and the promise has a reason for the failure.

Example :
```
function fetchData() {
    return new Promise (resolve, reject) => {
        Set Timeout ( () => {
            const data = { id:1, name : 'Thor" };
            if (data) { resolve (data); }
            else { reject ('data not found'); }
        }, 2000);
    });
}
```

Here, fetchData() function returns a promise that is resolved if data is available, & reject with a message 'data not found' if data is not available after a timeout of 2000 ms, or 2 seconds.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q2 Explain arrow functions with example

→ Arrow functions are a shorter syntax of writing functions in Javascript. It was introduced in ES6 and offer a concise way of writing functions. Arrow functions do not have their own 'this' arguments or 'super' binding. They often used for non method functions and are especially useful in cases like callbacks or functional programming.

Syntax → (parameter1, para2, ...) => {statements}

Example →
```
// Traditional function expression
const add = function (a, b) {
    return (a+b);
};
// equivalent arrow function
const addArrow = (a, b) => a+b;
```

If the function body has only one expression, we can omit the curly braces & the 'return' keyword. Here, both function returns the sum of 'a' & 'b' & produce some output but arrow function provides a more concise way to write functions in JS, making code shorter & easier to read and understand.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

3 Explain REST API? What are the principles of REST API?

→ A REST API (Rest Representational State transfer application programming interface) is a set of rules & conventions for building and interfacing with web services. RESTful API's use HTTP requests to perform standard operations like GET (retrieve data), POST (send data), PUT (update, data) & DELETE (remove data). REST is stateless, meaning each request from a client to the server must contain all the information the server needs to fulfil the request.

Principles of REST API:

1 Statelessness - Each request from a client to server must contain all the info needed to understand & process the request. The server does not store the client's state between requests.

2 Client Server architecture - The client & server are independent, they can evolve seprately client ~~sent~~ sends request and server processes them & sends responses.

3 Uniform interface :- Rest API follows a uniform interface, making it easy for developers to understand and use.

4 Cacheability - Responses from the server can be

marked as cacheable or non-cacheable. If cacheable, clients can reuse the response for similar requests, improving performance.

5 Layered System: REST APIs should be layered, allowing the architecture to be composed of multiple intermediaries that handles different aspects of communication between the client & the server.

Q4 Compare XML & JSON

⇒ XML (extensible markup language) & JSON (Java. Script Object Notation). & both formats used to structure data, commonly used in data interchange between systems.

Syntax:

1). XML - Uses more complex, tags based structure similar to HTML

```
<user>
<id> 123 </id>
<name> John </name>
</user>
```

2). JSON - Uses simpler key values pair structure enclosed by { }

```
{
"id" : 123
"name" : "John".
}
```

Readability:

1. XML - More verbose & can be harder to read due to need of opening & closing tags.
2. JSON - More concise & easier to read, especially for those familiar with JS.

Data types:

1. XML - all data treated as text, additional data types require explicit handling (eg - 'CDATA', attributes)
2. JSON - supports various data types natively, including strings, numbers, arrays, booleans & objects.

Usage:

1. XML - often used in configuration files, document, storage & in systems where extensibility is important
2. JSON - commonly used in web APIs & data interchange due to its simplicity and efficiency

Parsing:

1. XML - requires XML parser, which can be more complex & slower compared to JSON.
2. JSON - can be easily parsed using standard functions in many programming languages.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

5 Explain different types of components in react JS with an example.

⇒ 1. Class components:

> Class components are ES6 classes that extend 'React Component' & have access to lifecycle methods & states.

> They must have a 'render()' method that returns JSX

Example:

```
class Welcome extends React.Component {
    render() {
        return <h1> Hello, {this.props.name}
               </h1>;
    }
}
```

2. Function components:

> These are simple JavaScript functions that returns JSX. They do not have their own state or lifecycle methods.

> Function components are easier to write & understand.

Example:

```
function Welcome (props) {
    return <h1> Hello, {props.name} </h1>
}
```

Class components are more rich in feature, with access to lifecycle methods & 'this' keyword.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

6 Explain the features of React JS

→ 1) Component based architecture
React encourages breaking down the UI into reusable components, making code more manageable & easier to maintain.

2 Virtual DOM:
React uses a virtual DOM to efficiently update and render components when the state of an object changes, react first updates the virtual DOM, then compares it with real DOM, & at finally applies only the changes, improving performance.

3 Declarative UI:
React enables developers to describe what the UI should look like based on the current application state. React automatically updates & renders the right components when data changes.

4 Unidirectional data flow
Data in react flows in one direction from parent to child components which simplifies the debugging process and makes the application more predictable

5 JSX (JavaScript XML):
- JSX is syntax extension for JavaScript that allows developers to write HTML like code directly within JS. It makes code much more

readable & easier to write by combining HTML & JS.

6) Hooks.

⇒ Hooks allows function component to use state and other react features without writing a class. Hooks like 'useState' & 'useEffect' have simplified component logic and made functional components more powerful.

7) React developer tools.

⇒ React offers a set of developer tools that helps in inspecting component hierarchies, analyzing performance & debugging more effectively.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Written Assignment - 01

Name : Saikarthik Iyer
Batch : T13
Roll No. : 41
Subject : IP

Q1. ~~Explain Promises with an example~~
Q1. What are React ref? When to use Refs and When
=> ~~A from~~ not to use refs?

=> In react, refs are special objects that provide a reference to DOM node or react element. Its essentially a way to access the underlying DOM element or react element directly, allowing users to perform actions that are difficult to achieve through react's declarative approach.
A ref is created using React.createRef() in class component or useRef() in functional component.

```
function MyComp() {
  const myRef = React.useRef(null);
  React.useEffect() => {
    my.ref.current.focus();
  }, []);
  return <input ref = {myRef} />;
}
```

When to use Refs:
1) Imperative DOM manipulation: When you need to directly manipulate the DOM tasks like focusing elements, scrolling.

3) Integrating with third party libraries: when working with third-party libraries that require direct access to DOM elements.
4) Performance optimization: In rare cases, Refs can be used for performance optimization.

When not to use Refs:

1) Managing component state: avoid using Refs to manage component state. React's state management mechanism is more effective
2) Modifying props: Never use Refs to directly modify props. Props are intended to be immutable.
3) Creating mutable references: Excessive use of mutable references can make your code harder to reason about & maintain.

Q.2 Explain Hooks in React JS:

→ Hooks are new functions introduced in React 16.8 that allows you to use State & other react features without writing a class component. They provide a more concise & declarative way to write react components.

1) State Hook - useState
   > used to manage state within a functional component
   > Takes an initial state value & returns an array with two elements, current state value & a function to update the state.

example:

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

27 Effect Hooks – useEffect

> used to perform side effects in functional components.
> Takes a callback function that runs after a component renders.

example:

```
import{useEffect, usestate} from'react';
function MyComp(){
  const[count, setCount]= useState(0);
  useEffect(()=>{
    document.title = `You clicked ${count} times`;
  });
```

other Hooks:

- useContext: used to access value from a context provider
- useRef: used to create a mutable reference to a DOM element or react element.

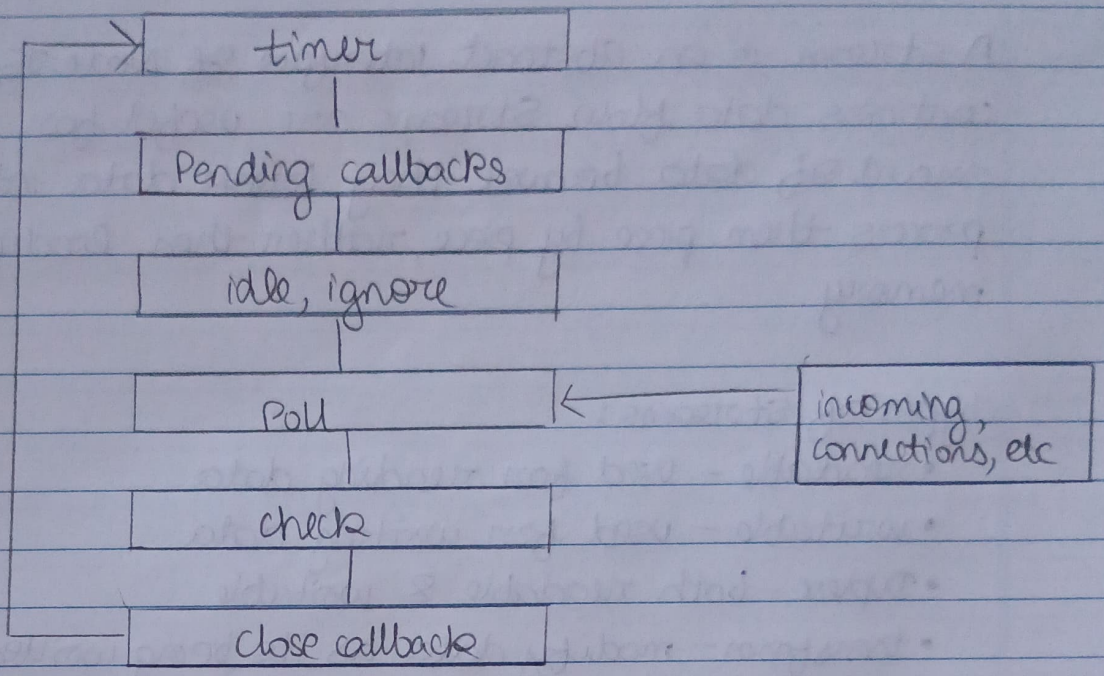
Q.3 Explain event loops in Node JS ?

→ The event loop allows Node JS to perform non blocking I/O operations despite the fact that JS is single-threaded. It is done by assigning operations to the OS whenever & whereever possible.

- An event loop is an endless loop, which waits for tasks, executes them, & then sleeps until it recieves more tasks.
- It executes tasks from the event queue only when call stack is empty.
- It allows us to use callbacks & promises
- It executes tasks starting from the oldest first.

The event loop can be broken down into different phases:

1) Timers phase: executes callbacks scheduled by setTimeout() & setInterval().
2) Pending callback phase: executes I/O callbacks that are deferred.
3) Idle, Prepare phase: ~~executes I/O callbacks that~~ Internal operations handled by libuv.
4) Poll phase: retrieves new I/O events (like file reading, network communication), where event loop will stay idle if there are no new events.
5) check phase: executes callbacks registered by ~~setTimeout()~~ setImmediate()
6) close callback phase: executes close event callbacks (eg. socket.on('close')).



Phases of event loops

Q.4 What are Buffers & Streams in Node JS ? Explain with example.

→ A Buffer in Node.JS is a temporary storage area for a chunk of memory, primirarily used to deal with binary data. It is designed to work efficiently with raw data.

example -

```
const buffer = Buffer.from('Hello, world');
console.log (buffer);
console.log (buffer.toString());
buffer[0] = 72;
console.log (buffer.toString());
```

A stream is an abstract interface of Node.JS used to handle continous data flow. Streams are useful for working with large amount of data because they break data into smaller chunks & process them piece by piece, rather than loading entire dataset into memory.

Types of Streams:

- readable - used for reading data
- writable - used for writing data
- Duplex - both readable & writable
- transform - modify data as its being written & read.

example -

```
const fs = require('fs');
const readable = fs.createReadStream('example.txt', {
    encoding: 'utf8' });
readable.on('data', (chunk) => {
    console.log(chunk);
});
readable.on('end', () => {
    console.log('no more data');
});
```

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Q5 Explain Routing in express JS along with an example

→ Routing is a process of mapping incoming HTTP requests to specific functions or handlers within your Node.js application. Express JS provides a powerful & flexible routing system that allows you to define routes based on various criterias, such as HTTP methods, URL paths, & query parameters.

Basic routing -
It involves defining a route using app.method() syntax, example:

```
const express = require('express');
const app = express();
app.get('/', (req, res) => {
    res.send('Hello');
});
app.listen(3000, () => {
    console.log('server on 3000 port');
});
```

Advanced routing -

- route parameter - defined using (:) in URL path.

```
app.get('/users/:id', (req, res) => {
    const uid = req.params.id;
});
```

- query parameter - can be accessed using req.query object

```
app.get('/search', (req, res) => {
    const qv = req.query.q ;
});
```

- middleware - used to perform actions before/after handling a req.

```
app.use('/api', (req, res, next) => {
    next();
});
```

Explain REST API? what are the principles of REST API?

→ a REST API (Representational state transfer application programing interface) is a set of rules & conventions for building & interacting with web services. RESTful API's use HTTP requests to perform standard operations like GET (retrieve data), POST (send data), PUT (update data), & DELETE (remove data). REST is stateless, meaning each request from a client to the server must contain all the information the server needs to fullfil the request.

Principles of REST API:

1) Statelessness: each req. from a client to server must contain all the info. needed to understand & process the request. The server does not store the client's state between requests.
2) client-server architecture: The client & server are independent, they can evolve seperately. Client sends req, & server processes them & sends responses.
3) Uniform interface: REST API follows a uniform interface, making it easy for developers to understand & use.
4) Cacheability: Responses from the server can be marked as cacheable or non-cacheable. If cacheable, clients can reuse the response for similar requests, improving performance
5) Layered system: REST APIs should be layered, allowing the architecture to be composed of multiple intermediaries that handles different aspects of communication between the client & the server.