TIRUMALA ENGINEERING COLLEGE

Affiliated to Jawaharlal Nehru Technological University Kakinada

Approved by AICTE and Accredited by NAAC & NBA

Jonnalagadda, Narasaraopet, PIN: 522601

LABORATORY RECORD

MEAN STACK TECHNOLOGIES - MODULE I

(Skill Oriented Course)

Submitted By:

Name : Sneha Doppalapudi
Roll Number : 22NE1A0539
Branch : III/IV COMPUTER SCIENCE AND ENGINEERING
Section : A

CERTIFICATE

This is to certify that this is the bonafied record work of work done by Mr/Ms **SNEHA DOPPALAPUDI** Regd.Of **22NE1A0539** Of **THIRD** year B.Tech **SECOND** semester in the **MEAN STACK TECHNOLOGIES MODULE 1** Laboratory during the academic year **2024-25** Performed **12** number of experiments out of **12**

Internal Examiner ***External Examiner***

Head of the Department

INDEX

	D) Consider that developer needs to declare a manufacturer's array holding 4 objects with id and price as a parameter and needs to implement an arrow function - myfunction to populate the id parameter of manufacturers array whose price is greater than	49		
	E) Declare a function - getMobileByManufacturer with two parameters namely manufacturer and id, where manufacturer value should passed as Samsung and id parameter should be optional while invoking the function, if id is passed as 101 then this function	50-51		
10	A) Implement business logic for adding multiple Product values into a cart variable which is type of string array.	52		
	B) Declare an interface named - Product with two properties like productId and productName with a number and string datatype and need to implement logic to populate the Product details.	53		
	C) Declare an interface named - Product with two properties like productId and productName with the number and string datatype and need to implement logic to populate the Product details	54		
	D) Declare an interface with function type and access its value.	55		
11	A) Declare a productList interface which extends properties from two other declared interfaces like Category,Product as well as implementation to create a variable of this interface type.	56		
	B) Consider the Mobile Cart application, Create objects of the Product class and place them into the productlist array.	57		
	C) Declare a class named - Product with the below-mentioned declarations: (i) productId as number property (ii) Constructor to initialize this value (iii) getProductId method to return the message "Product id is <<id value>>".	58		
	D) Create a Product class with 4 properties namely productId, productName, productPrice, productCategory with private, public, static, and protected access modifiers and accessing them through Gadget class and its methods.	59-60		
12	A) Create a Product class with 4 properties namely productId and methods to setProductId() and getProductId().	61		
	B) Create a namespace called ProductUtility and place the Product class definition in it. Import the Product class inside productlist file and use it.	62		
	C) Consider the Mobile Cart application which is designed as part of the functions in a module to calculate the total price of the product using the quantity and price values and assign it to a totalPrice variable	63		
	D) Create a generic array and function to sort numbers as well as string values.	64		

Experiment-01

A) Include the Metadata element in Homepage.html for providing description as "IEKart's is an online shopping website that sells goods in retail. This company deals with various categories like Electronics, Clothing, Accessories etc.

```
<html>

<head>

<title>IEKart's Online Shopping</title>

<meta name="description" content="IEKart's is an online shopping website that sells goods in retail. This company deals with various categories like Electronics, Clothing, Accessories etc.">

</head>

</html>
```

B) Enhance the Homepage.html of IEKart's Shopping Application by adding appropriate sectioning elements.

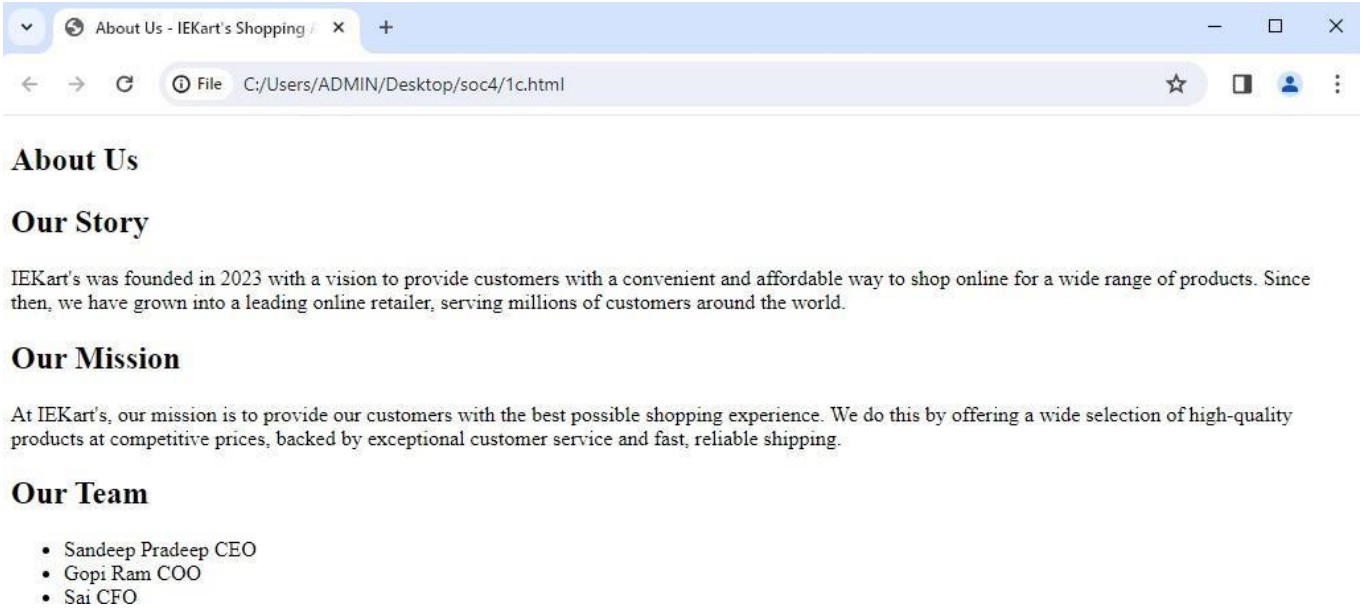
```
<!DOCTYPE html>

<html>

<head>

<title>IEKart's Shopping Application</title>

<meta name="description" content="IEKart's is an online shopping website that sells goods in retail. This company deals with various categories like Electronics, Clothing, Accessories etc.">

</head>

<body>

<header>

<!-- Add a header section for the site logo and navigation links -->

<nav>

<!-- Navigation links here -->

</nav>

</header>

<main>

<!-- Add a main section for the main content of the page -->

<section id="featured-products">

<h2>Featured Products</h2>
```

```
<!-- Featured products content here -->
</section>
<section id="product-categories">
<h2>Product Categories</h2>
<!-- Product categories content here -->
</section>
<section id="new-arrivals">
<h2>New Arrivals</h2>
<!-- New arrivals content here -->
</section>
<section id="deals">
<h2>Deals</h2>
<!-- Deals content here -->
</section>
</main>
<footer>
<!-- Add a footer section for copyright and contact information -->
</footer>
</body>
</html>
```

C) Make use of appropriate grouping elements such as list items to "About Us" page of IEKart's Shopping Application

```
<!DOCTYPE html>

<html>

<head>

<title>About Us - IEKart's Shopping Application</title>

<meta name="description" content="Learn more about IEKart's, an online shopping

website that sells goods in retail.">

</head>

<body>

<header>

<!-- Add a header section for the site logo and navigation links -->

<nav>

<!-- Navigation links here -->

</nav>

</header>

<main>

<!-- Add a main section for the main content of the page -->

<section>

<h1>About Us</h1>

<h2>Our Story</h2>

<p>IEKart's was founded in 2023 with a vision to provide customers with a convenient and affordable way to shop online for a wide range of products. Since then, we have grown into a leading online retailer, serving millions of customers around the world.</p>

<h2>Our Mission</h2>

<p>At IEKart's, our mission is to provide our customers with the best possible shopping experience. We do this by offering a wide selection of high-quality products at competitive prices, backed by exceptional customer service and fast, reliable shipping.</p>

<h2>Our Team</h2>

<ul>

<li>

<span class="team-member-name">Sandeep Pradeep</span>

<span class="team-member-title">CEO</span>
```

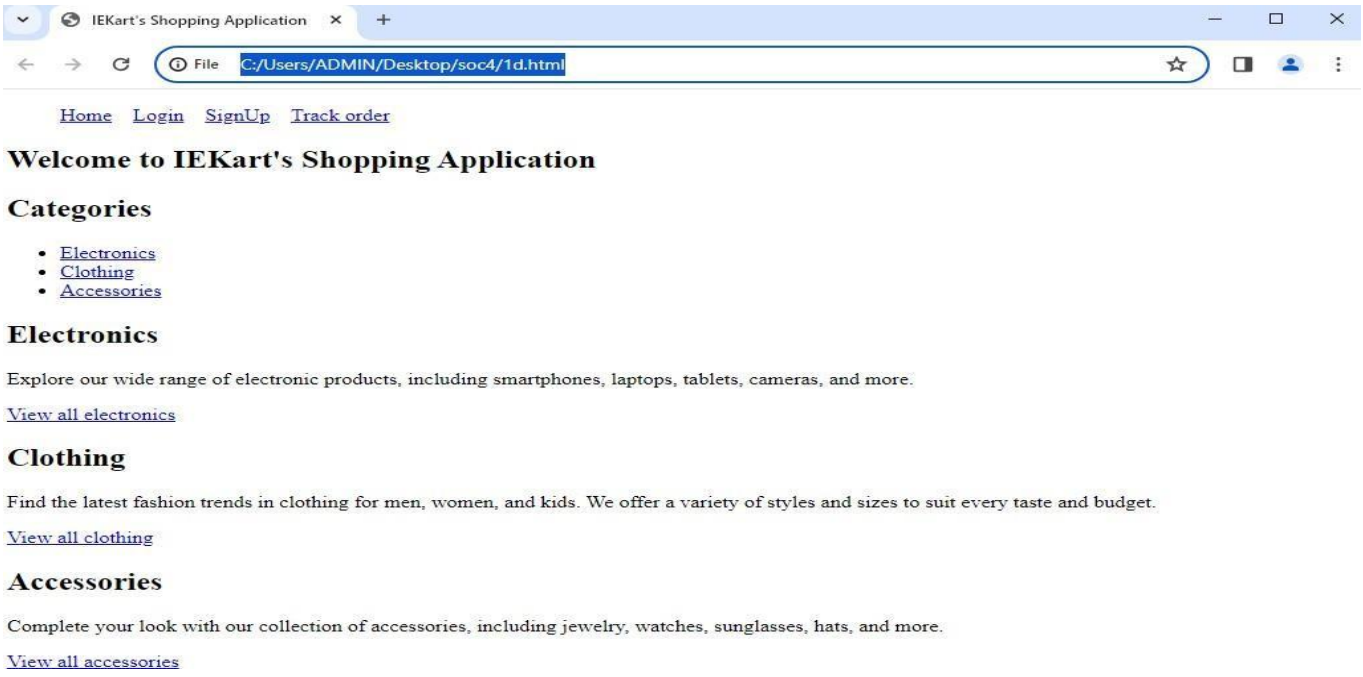
```
</li>
<li>
<span class="team-member-name">Gopi Ram</span>
<span class="team-member-title">COO</span>
</li>
<li>
<span class="team-member-name">Sai</span>
<span class="team-member-title">CFO</span>
</li>
</ul>
</section>
</main>
<footer>
<!-- Add a footer section for copyright and contact information -->
</footer>
</body>
</html>
```



D) Link "Login", "SignUp" and "Track order" to "Login.html", "SignUp.html" and "Track.html" page respectively. Bookmark each category to its details of IEKart's Shopping application.

```
<!DOCTYPE html>
<html>
<head>
<title>IEKart's Shopping Application</title>
<meta name="description" content="An online shopping website that sells goods in retail.">
</head>
<body>
<header>
<!-- Add a header section for the site logo and navigation links -->
<nav>
<ul>
<a href="index.html">Home</a>   
<a href="login.html">Login</a>   
<a href="signup.html">SignUp</a>   
<a href="track.html">Track order</a>
</ul>
</nav>
</header>
 <main>
<!-- Add a main section for the main content of the page -->
<section>
<h1>Welcome to IEKart's Shopping Application</h1>
<h2>Categories</h2>
<ul>
<li><a href="#electronics">Electronics</a></li>
<li><a href="#clothing">Clothing</a></li>
<li><a href="#accessories">Accessories</a></li>
</ul>
```

```
<h2 id="electronics">Electronics</h2>

<p>Explore our wide range of electronic products, including smartphones, laptops, tablets, cameras, and more.</p>

<p><a href="product/electronics.html">View all electronics</a></p>

<h2 id="clothing">Clothing</h2>

<p>Find the latest fashion trends in clothing for men, women, and kids. We offer a variety of styles and sizes to suit every taste and budget.</p>

<p><a href="product/clothing.html">View all clothing</a></p>

<h2 id="accessories">Accessories</h2>

<p>Complete your look with our collection of accessories, including jewelry, watches, sunglasses, hats, and more.</p>

<p><a href="product/accessories.html">View all accessories</a></p>

</section>

</main>

<footer>

<!-- Add a footer section for copyright and contact information -->

</footer>

</body>

</html>
```



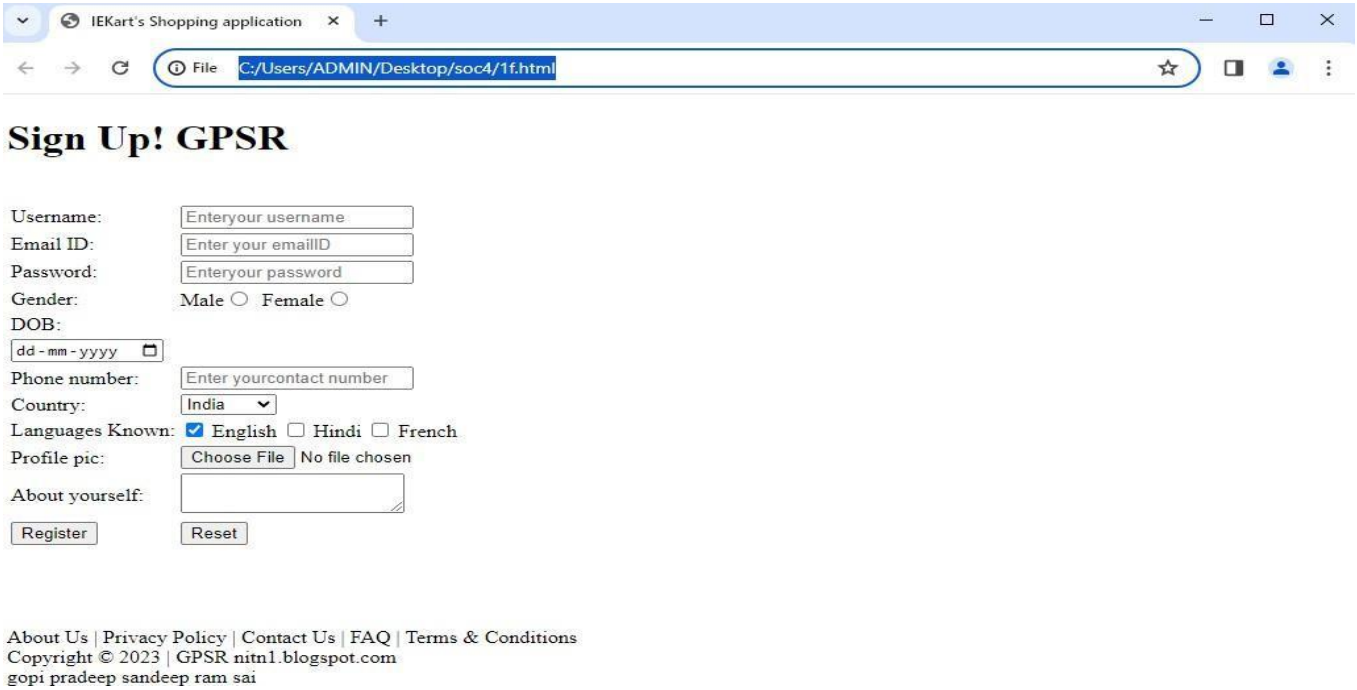
E) Add the © symbol in the Home page footer of IEKart's Shopping application.

```
<!DOCTYPE html>
<html>
<head>
<title>IEKart's Shopping Application</title>
<meta name="description" content="An online shopping website that sells goods in retail.">
</head>
<body>
<header>
<!-- Add a header section for the site logo and navigation links -->
</header>
<main>
<!-- Add a main section for the main content of the page -->
 </main>
<footer>
 <!-- Add a footer section for copyright and contact information -->
<p>&copy; 2023 IEKart's Shopping Application. All rights reserved.</p>
</footer>
</body>
</html>
```

F) Add the global attributes such as content editable, spellcheck, id etc. to enhance the Signup Page functionality of IEKart's Shopping application.

```
<!DOCTYPE html>

<html>

<head>

<meta name="keywords" content="Online, Shopping" /> <title> IEKart's Shopping

application </title>

</head>

<body>

<header>

<h1>Sign Up! GPSR </h1>

</header>

<article><form action="success.html" method="get">

<table><tr><td><br><label for="username">Username:</label></td>

<td><br><input type="text" id="username" placeholder="Enter your username" required /></td>

</tr><tr><td><label for="email_id">Email ID:</label></td>

<td><input type="email" id="email_id" placeholder="Enter your email ID" required /></td>

</tr><tr><td><label for="password">Password:</label></td>

<td><input type="password" id="password" placeholder="Enter your password" required /></td>

</tr><tr><td><label for="gender">Gender:</label></td>

<td><label for="male">Male<input type="radio" id="male" name="gender" value="M"/>  <label
for="female">Female<input type="radio" id="female" name="gender" value="F" /></td></tr><tr>

<td><label for="dob">DOB:</label></td></tr><td><input type="date" id="dob" required /></td></tr><tr>

<td><label for="phone_no">Phone number:</label></td>

<td><input type="text" id="phone_no" placeholder="Enter your contact number" pattern="+ [0-9] {12}"
required /></td></tr><tr>

<td><label for="country">Country:</label></td>

<td><select id="country" placeholder="Select your country"> <option value="India"/>India <option
value="India" />USA
```

```
<option value="India" />UK<option value="India" />Canada<option value="India" />Belgium<option value="India" />France </select></td></tr><tr>

<td><label id="language">Languages Known:</label></td><td><input type="checkbox" name="language" id="english" value="English" checked="checked" />

<label for="english">English </label> <input type="checkbox" name="language" id="hindi" value="Hindi"/> <label for="hindi"> Hindi</label> <input type="checkbox" name="language" id="french" value="French"/> <label for="french">French</label></td> </tr><tr>

<td><label for="pic">Profile pic:</label></td> <td><input type="file" id="pic" required/></td>

</tr><tr>

<td><label for="yourself" dir="ltr">About yourself:</label></td>

<td><textarea></textarea></td> <!-- Add contenteditable and spellcheck attribute -->

</tr><tr>

<td><input type="submit" value="Register" /> <td><input type="reset" value="Reset"/> </tr>

</table></form> <br /> <br /> <br /> <br />

</article><footer>

<nav> About Us | Privacy Policy | Contact Us | FAQ | Terms & Conditions </nav>

Copyright &copy; 2023  </footer>

<aside> gopi pradeep sandeep ram sai </aside>

</body>

</html>
```



Experiment-02

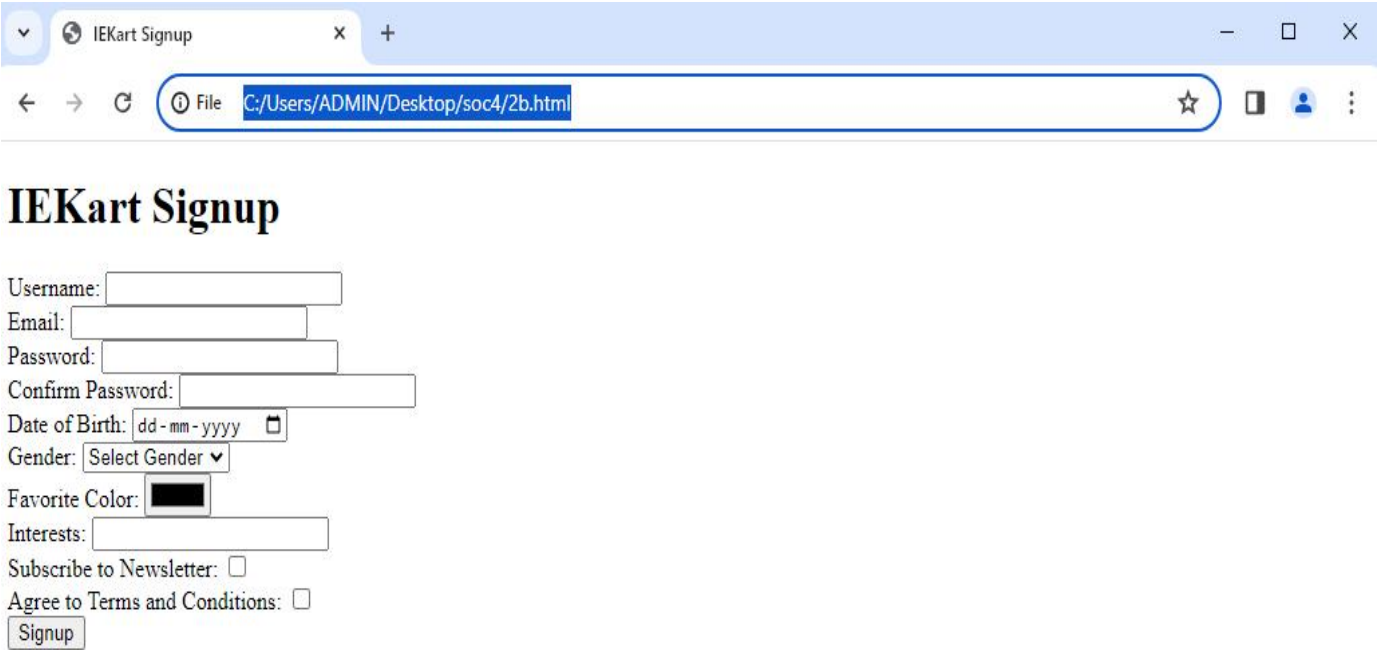
A) Enhance the details page of IEKart's Shopping application by adding a table element to display the available mobile/any inventories.

```
<h1>IEKart's Shopping </h1>

<table border="1" cellspacing="0" cellpadding="10">

<tr><th colspan="2">Mobile/Any Inventory</th>

<th colspan="2">Price</th><th>Availability</th></tr>

<tr><td colspan="2">Samsung Galaxy S21 Ultra</td>

<td colspan="2">$1,199.99</td><td>In stock</td></tr>

<tr><td rowspan="2">Apple iPhone 12 Pro Max</td>

<td>Description</td>

<td colspan="2">A14 Bionic chip, Pro camera system, Ceramic Shield front cover</td>

<td rowspan="2">Out of stock</td></tr>

<tr><td>Price</td><td colspan="2">$1,099.99</td></tr>

<tr><td rowspan="3">Google Pixel 5</td><td>Description</td>

<td colspan="2">5G capable, 90Hz OLED display, Night Sight camera mode</td>

<td>In stock</td></tr><tr><td>Price</td>

<td colspan="2">$699.99</td><td rowspan="2">One left in stock</td></tr>

<tr><td colspan="2">Color options: Just Black, Sorta Sage</td></tr>

</table>
```

B) Using the form elements create Signup page for IEKart's Shopping application.

```
<!DOCTYPE html>

<html>

<head>

<title>IEKart Signup</title>

</head>

<body>

<h1>IEKart Signup</h1>

<form action="process-signup.php" method="post">

<label for="username">Username:</label>

<input type="text" id="username" name="username" required>

<br><label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<br><label for="password">Password:</label>

<input type="password" id="password" name="password" required> <br>

<label for="confirm-password">Confirm Password:</label>

<input type="password" id="confirm-password" name="confirm-password"required>

<br>

<label for="dob">Date of Birth:</label>

<input type="date" id="dob" name="dob" required>

<br>

<label for="gender">Gender:</label>

<select id="gender" name="gender" required>

<option value="">Select Gender</option>

<option value="male">Male</option>

<option value="female">Female</option>

<option value="other">Other</option>
```

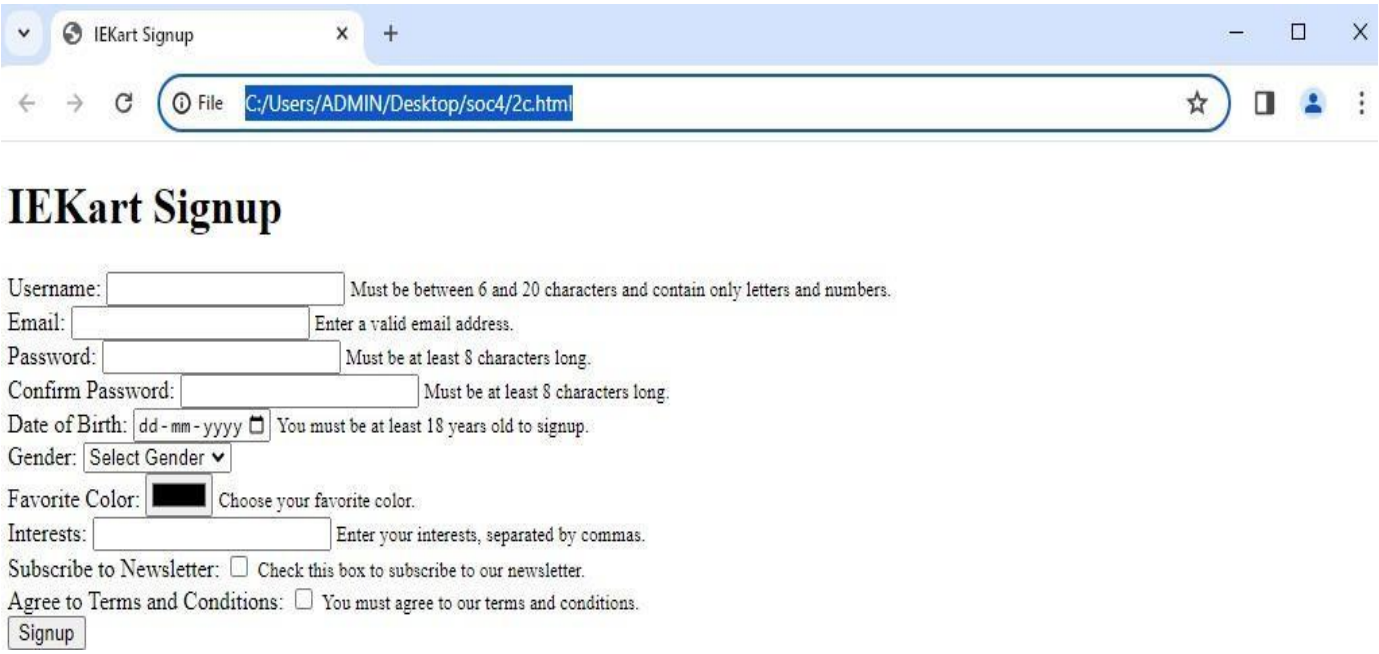
```
</select>

<br>

<label for="favorite-color">Favorite Color:</label>

<input type="color" id="favorite-color" name="favorite-color">

<br>

<label for="interests">Interests:</label>

<input type="text" id="interests" name="interests">

<br>

<label for="newsletter">Subscribe to Newsletter:</label>

<input type="checkbox" id="newsletter" name="newsletter"

value="yes"> <br>

<label for="agreement">Agree to Terms and Conditions:</label>

<input type="checkbox" id="agreement" name="agreement"

required> <br>

<input type="submit" value="Signup">

</form>

</body>

</html>
```



C) Enhance Signup page functionality of IEKart's Shopping application by adding attributes to input elements.

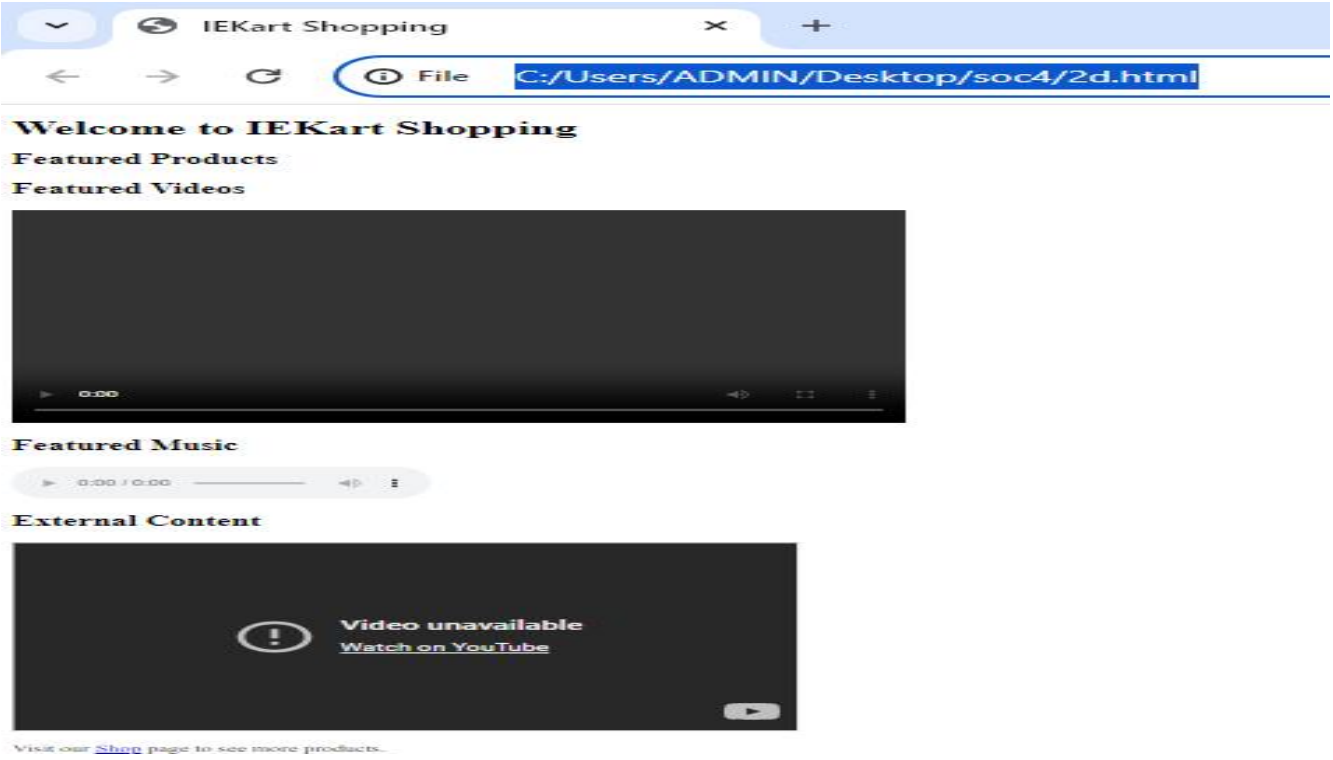
```
<!DOCTYPE html>

<html>

<head>

<title>IEKart Signup</title>

</head>

<body>

<h1>IEKart Signup</h1>

<form action="process-signup.php" method="post">

<label for="username">Username:</label>

<input type="text" id="username" name="username" required minlength="6"maxlength="20" pattern="[a-zA-Z0-9]+">

<small>Must be between 6 and 20 characters and contain only letters and numbers.</small><br>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<small>Enter a valid email address.</small><br>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required minlength="8">

<small>Must be at least 8 characters long.</small>

<br><label for="confirm-password">Confirm Password:</label>

<input type="password" id="confirm-password" name="confirm-password" required minlength="8">

<small>Must be at least 8 characters long.</small><br>

<label for="dob">Date of Birth:</label>

<input type="date" id="dob" name="dob" required max="2005-01-01"> <small>You must be at least 18 years old to signup.</small><br>

<label for="gender">Gender:</label>

<select id="gender" name="gender" required>

<option value="">Select Gender</option>
```

```
<option value="male">Male</option>

<option value="female">Female</option>

<option value="other">Other</option></select>

<br><label for="favorite-color">Favorite Color:</label>

<input type="color" id="favorite-color" name="favorite-color">

<small>Choose your favorite color.</small><br>

<label for="interests">Interests:</label>

<input type="text" id="interests" name="interests">

<small>Enter your interests, separated by commas.</small><br>

<label for="newsletter">Subscribe to Newsletter:</label>

<input type="checkbox" id="newsletter" name="newsletter" value="yes"> <small>Check this box to subscribe to our newsletter.</small><br>

<label for="agreement">Agree to Terms and Conditions:</label>

<input type="checkbox" id="agreement" name="agreement" required> <small>You must agree to our terms andconditions.</small><br>

<input type="submit" value="Signup">

</form>

</body>

</html>
```



D) Add media content in a frame using audio, video, iframe elements to the Home page of IEKart's Shopping application.

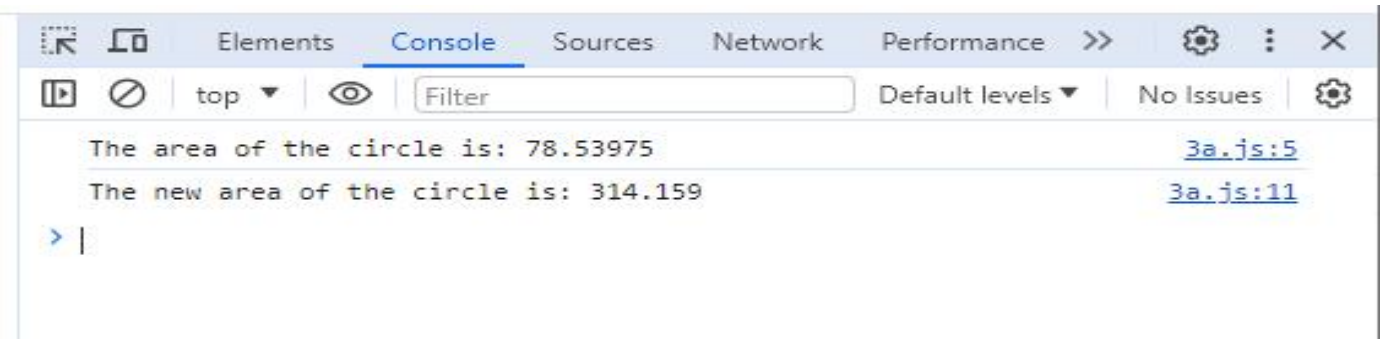
```
<!DOCTYPE html>

<html>

<head>

<title>IEKart Shopping</title>

</head><body><h1>Welcome to IEKart Shopping</h1>

<h2>Featured Products</h2>

<!-- Add product images here -->

<h2>Featured Videos</h2>

<video src="featured-video.mp4" controls width="640" height="360"></video>

<h2>Featured Music</h2><audio src="featured-music.mp3" controls></audio><h2>External Content</h2>

<iframe src="https://www.youtube.com/embed/dQw4w9WgXcQ" width="560" height="315"></iframe>

<p>Visit our <a href="shop.html">Shop</a> page to see more products.</p>

</body>

</html>
```



Experiment-03

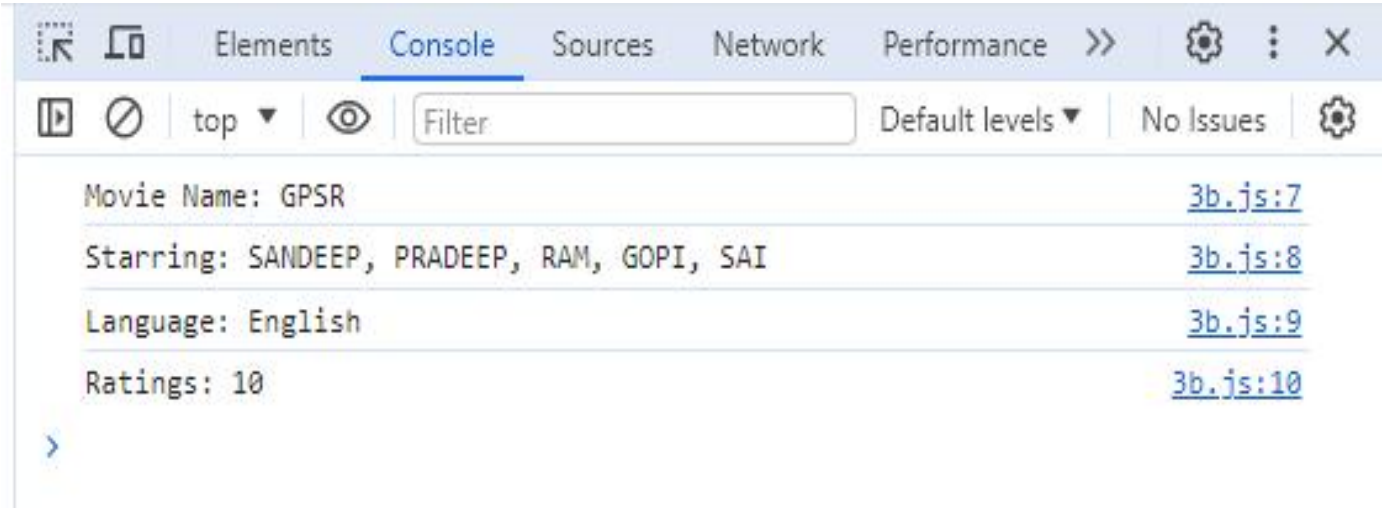
A) Write a JavaScript program to find the area of a circle using radius (var and let - re assign and observe the difference with var and let) and PI (const)

```
// using var

var radius = 5;

var pi = 3.14159;

var area = pi * radius * radius;

console.log("The area of the circle is: " + area);

// using let

let newRadius = 10;

let newArea;

const newPi = 3.14159;

newArea = newPi * newRadius * newRadius;

console.log("The new area of the circle is: " + newArea);
```



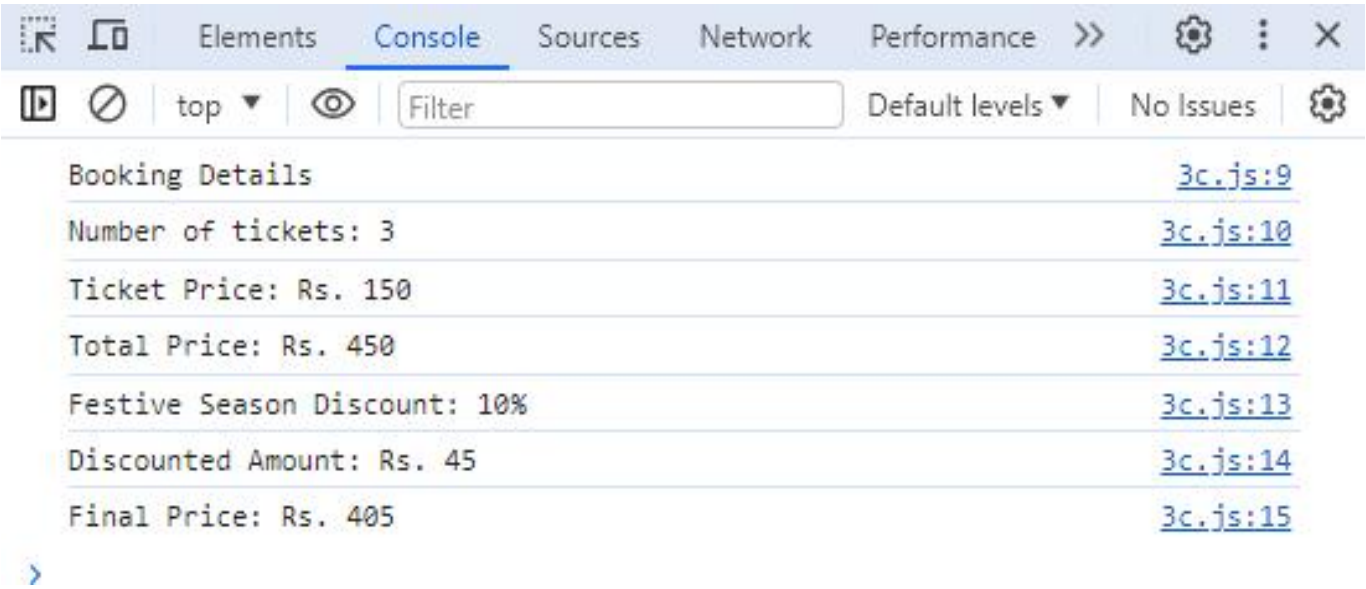
B) Write JavaScript code to display the movie details such as movie name, starring, language, and ratings. Initialize the variables with values of appropriate types. Use template literals wherever necessary.

```
// initialize variables

const movieName = "GPSR";

const starring = ["SANDEEP", "PRADEEP","RAM","GOPI","SAI"];

const language = "English";

const ratings = 10.0;

// display movie details using template literals

console.log(`Movie Name: ${movieName}`);

console.log(`Starring: ${starring.join(", ")}`);

console.log(`Language: ${language}`);

console.log(`Ratings: ${ratings}`);
```



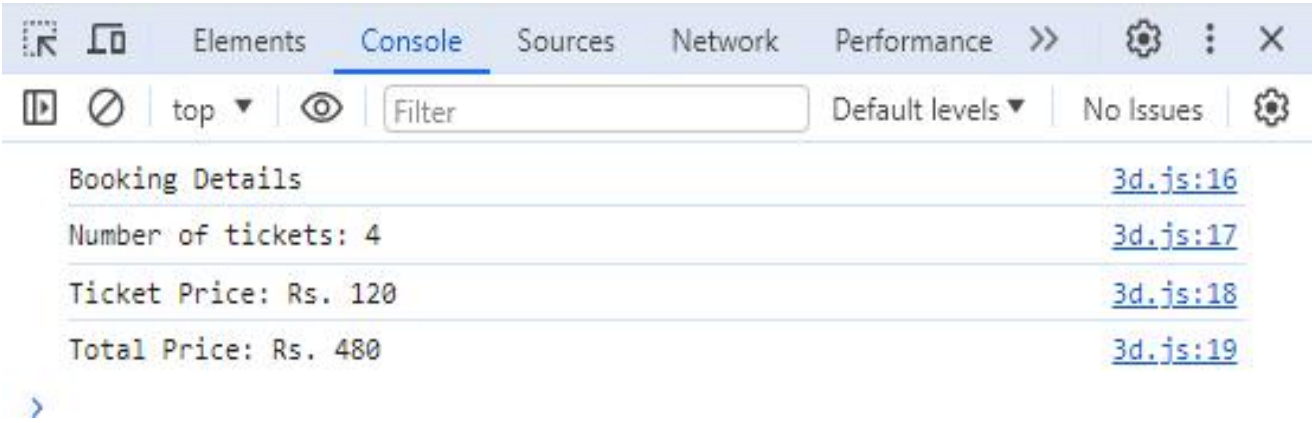
C) Write JavaScript code to book movie tickets online and calculate the total price, considering the number of tickets and price per ticket as Rs. 150. Also, apply a festive season discount of 10% and calculate the discounted amount.

```
// initialize variables

const ticketPrice = 150;

let numOfTickets = 3;

let totalPrice = numOfTickets * ticketPrice;

const festiveSeasonDiscount = 0.1;

let discountedAmount = totalPrice * festiveSeasonDiscount;

let finalPrice = totalPrice - discountedAmount;

// display booking details

console.log(`Booking Details`);

console.log(`Number of tickets: ${numOfTickets}`);

console.log(`Ticket Price: Rs. ${ticketPrice}`);

console.log(`Total Price: Rs. ${totalPrice}`);

console.log(`Festive Season Discount: ${festiveSeasonDiscount * 100}%`);

console.log(`Discounted Amount: Rs. ${discountedAmount}`);

console.log(`Final Price: Rs. ${finalPrice}`);
```



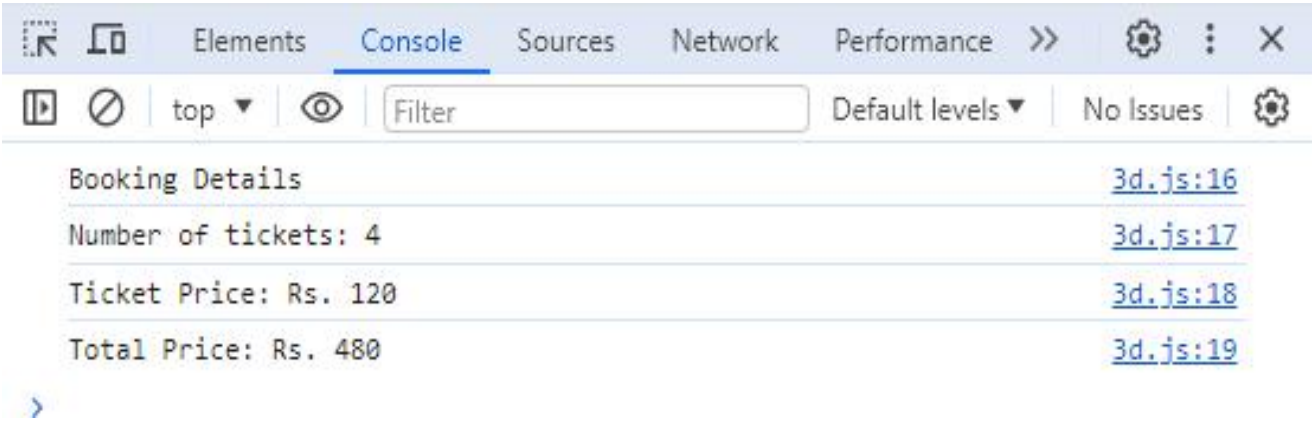
D) Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed.

```
// initialize variables

let numOfTickets = 4;

let ticketPrice = 150;

let totalPrice;

// apply conditions to calculate total price

if (numOfTickets <= 2) {

totalPrice = numOfTickets * ticketPrice;

} else if (numOfTickets >= 6) {

console.log(`Sorry, booking is not allowed for 6 or more seats.`);

} else {

   ticketPrice = 120;

   totalPrice = numOfTickets * ticketPrice;}

   // display booking details

   if (totalPrice) {

   console.log(`Booking Details`);

   console.log(`Number of tickets: ${numOfTickets}`);

   console.log(`Ticket Price: Rs. ${ticketPrice}`);

   console.log(`Total Price: Rs. ${totalPrice}`);

}
```



E) Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed.

```
// initialize variables

let numOfTickets = 4;

let ticketPrice = 150;

let totalPrice;

// apply conditions to calculate total price

if (numOfTickets <= 2) {

totalPrice = numOfTickets * ticketPrice;

} else if (numOfTickets >= 6) {

console.log(`Sorry, booking is not allowed for 6 or more seats.`);

} else {

   ticketPrice = 120;

   totalPrice = numOfTickets * ticketPrice;}

   // display booking details

   if (totalPrice) {

   console.log(`Booking Details`);

   console.log(`Number of tickets: ${numOfTickets}`);

   console.log(`Ticket Price: Rs. ${ticketPrice}`);

   console.log(`Total Price: Rs. ${totalPrice}`);

}
```



Experiment-04

A) Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed.

```
function bookMovieTickets(numOfTickets)

  { let ticketPrice = 150;

  let totalPrice;

  if (numOfTickets <= 2) {

  totalPrice = numOfTickets * ticketPrice;

  } else if (numOfTickets >= 6) {

  console.log("Booking is not allowed for 6 or more seats.");

  return; } else {

  ticketPrice = 120;

  totalPrice = numOfTickets * ticketPrice; }

  console.log(`Booking Details\nNumber of tickets: ${numOfTickets}\nTicket Price:

  Rs. ${ticketPrice}\nTotal Price: Rs. ${totalPrice}`);

  }

bookMovieTickets(2);

bookMovieTickets(4);

bookMovieTickets(8);
```

B) Create an Employee class extending from a base class Person. Hints: (i) Create a class Person with name and age as attributes. (ii) Add a constructor to initialize the values (iii) Create a class Employee extending Person with additional attributes role

```
class Person
  { constructor(name, age)
  { this.name = name;
  this.age = age;
  }
 }
class Employee extends Person
  { constructor(name, age, role) {
    super(name, age);  // Call the parent class constructor
    this.role = role;
  }
 }
const john = new Employee('John Doe', 30, 'Manager');
console.log(john);
```

C) Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed

```
const seatsInput = document.querySelector("#seats-input");

const bookNowButton = document.querySelector("#book-now-button");

bookNowButton.addEventListener("click", function() {

const numSeats = Number(seatsInput.value);

if (numSeats <= 0) {

alert("Please enter a valid number of seats.");

return;

} else if (numSeats > 5) {

alert("Sorry, we cannot book more than 5 seats at a time.");

return;

}

let totalPrice = numSeats * 150;

if (numSeats === 2) {

totalPrice = totalPrice - (totalPrice * 0.1);

}

alert(`Total price for ${numSeats} seats is Rs. ${totalPrice}.`);

});
```

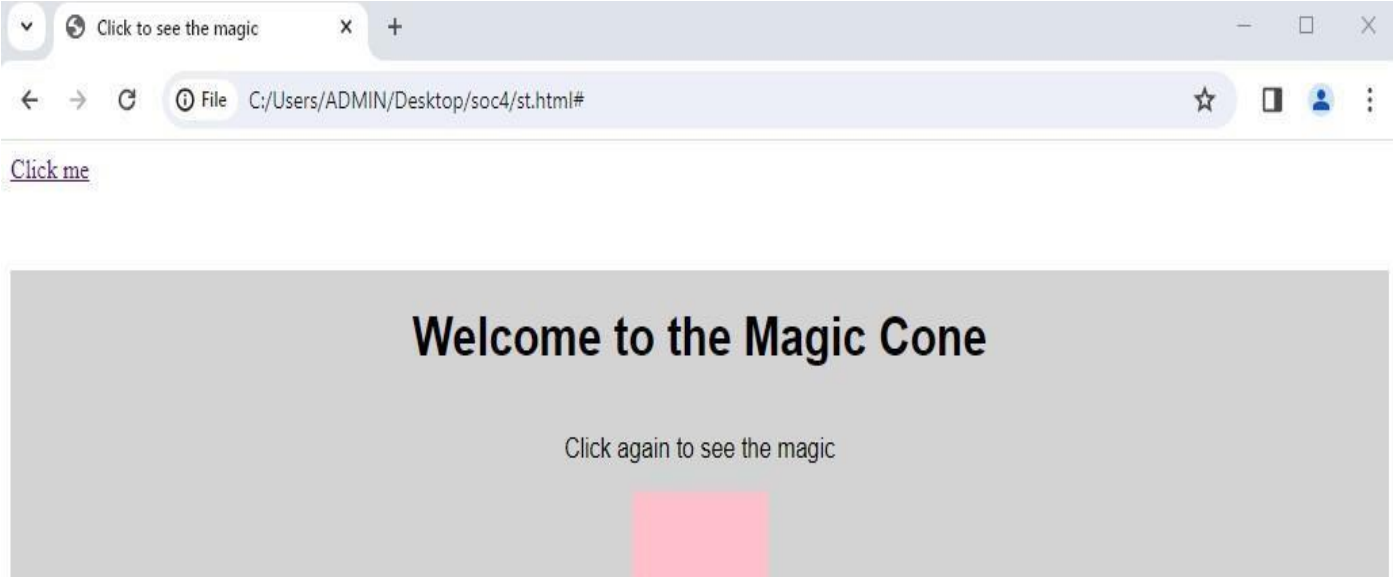
D) If a user clicks on the given link, they should see an empty cone, a different heading, and a different message and a different background color. If user clicks again, they should see a re-filled cone, a different heading, a different message

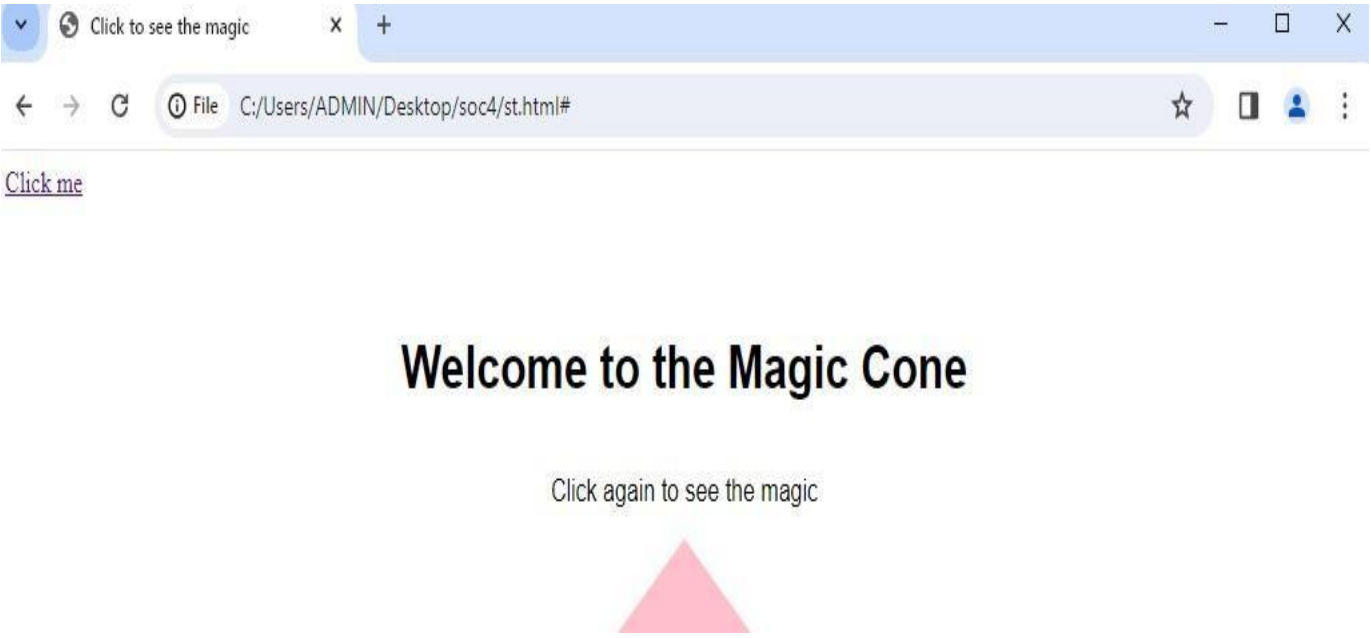
```
<!DOCTYPE html>

<html>

<head>

  <title>Click to see the magic</title>

  <style>

    /* CSS */

    .cone

      { width:

      0;

      height: 0;

      border-bottom: 50px solid pink;

      border-left: 50px solid transparent;

      border-right: 50px solid transparent;

    }

#container

      { display: flex;

      flex-direction: column;

      align-items: center;

      margin-top: 50px;

      font-family: Arial, sans-serif;

    }

#container.filled {

      background-color: lightgray;

    }

  </style>

</head>
```

```
<body>
  <a href="#" id="link">Click me</a>
  <div id="container"></div>

  <script>
    // JavaScript
    document.addEventListener('DOMContentLoaded', function()
    { const link = document.getElementById('link');
      const container = document.getElementById('container');

      link.addEventListener('click', () => {
        const heading = document.createElement('h1');
        heading.textContent = 'Welcome to the Magic Cone';

        const message = document.createElement('p');
        message.textContent = 'Click again to see the magic';

        const cone = document.createElement('div');
        cone.classList.add('cone');

        if    (container.classList.contains('filled'))
          {       cone.style.backgroundColor       =
          'white';
          container.classList.remove('filled');
        } else {
          cone.style.backgroundColor = 'pink';
          container.classList.add('filled');
        }
```

```
container.innerHTML = '';
```

```
      container.appendChild(heading);
      container.appendChild(message);
      container.appendChild(cone);
    });
  });
 </script>
</body>
</html>
```



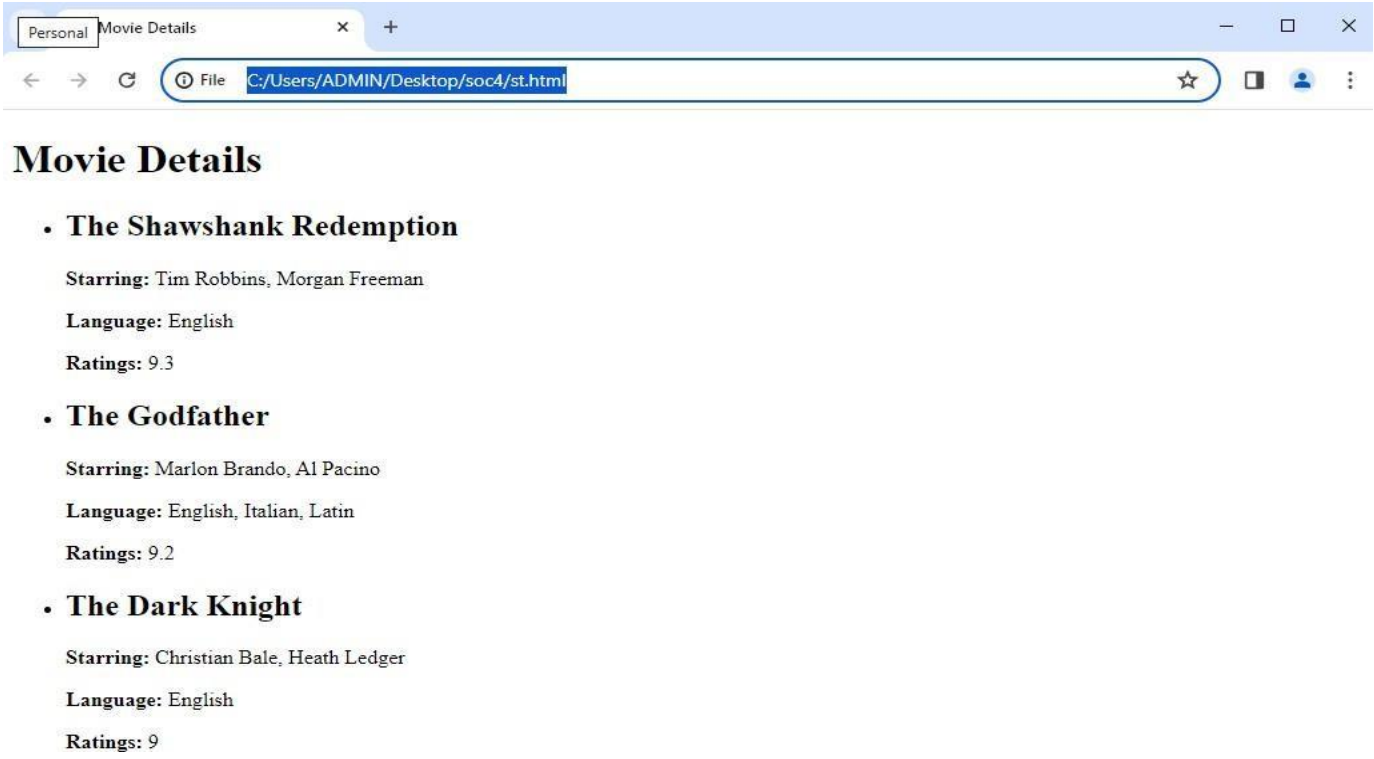


Experiment-05

A) Create an array of objects having movie details. The object should include the movie name, starring, language, and ratings. Render the details of movies on the page using the array.

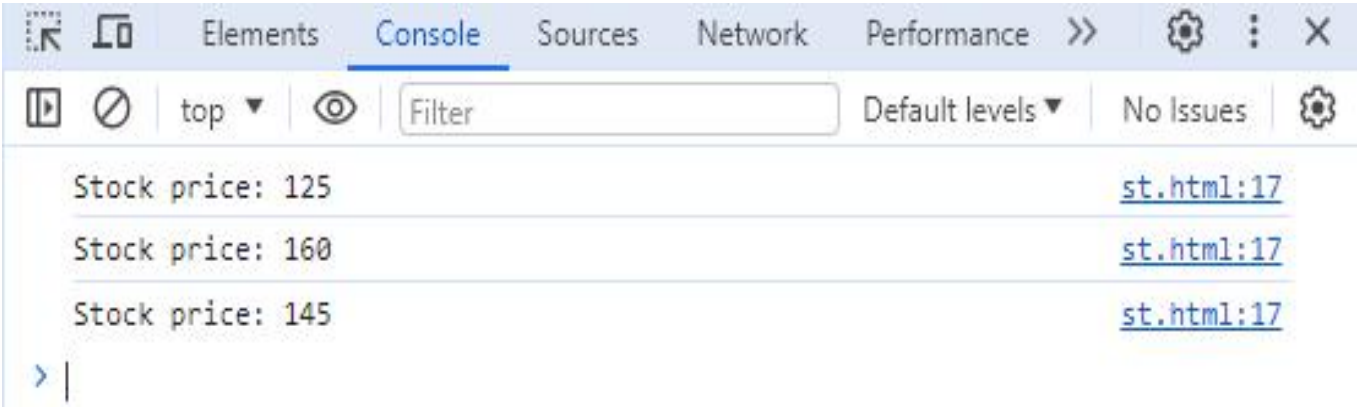
```
<!DOCTYPE html>

<html>

<head>

  <title>Movie Details</title>

</head>

<body>

  <h1>Movie Details</h1>

  <ul id="movies"></ul>

<script>

    const movies = [

      {

        name: "The Shawshank Redemption",

        starring: "Tim Robbins, Morgan Freeman",

        language: "English",

        ratings: 9.3

      }, {

        name: "The Godfather",

        starring: "Marlon Brando, Al Pacino",

        language: "English, Italian, Latin",

        ratings: 9.2

      },

      {

        name: "The Dark Knight",

        starring: "Christian Bale, Heath Ledger",

        language: "English",
```

```
      ratings: 9.0
    } ];
  const moviesList = document.getElementById("movies");
 movies.forEach(movie => {
    const movieDetails = `
      <li>
        <h2>${movie.name}</h2>
        <p><strong>Starring:</strong> ${movie.starring}</p>
        <p><strong>Language:</strong> ${movie.language}</p>
        <p><strong>Ratings:</strong> ${movie.ratings}</p>
      </li>
    `;
    moviesList.insertAdjacentHTML("beforeend", movieDetails);});
 </script>
</body>
</html>
```



B) Simulate a periodic stock price change and display on the console. Hints: (i) Create a method which returns a random number - use Math.random, floor and other methods to return a rounded value. (ii) Invoke the method for every three seconds and stop when

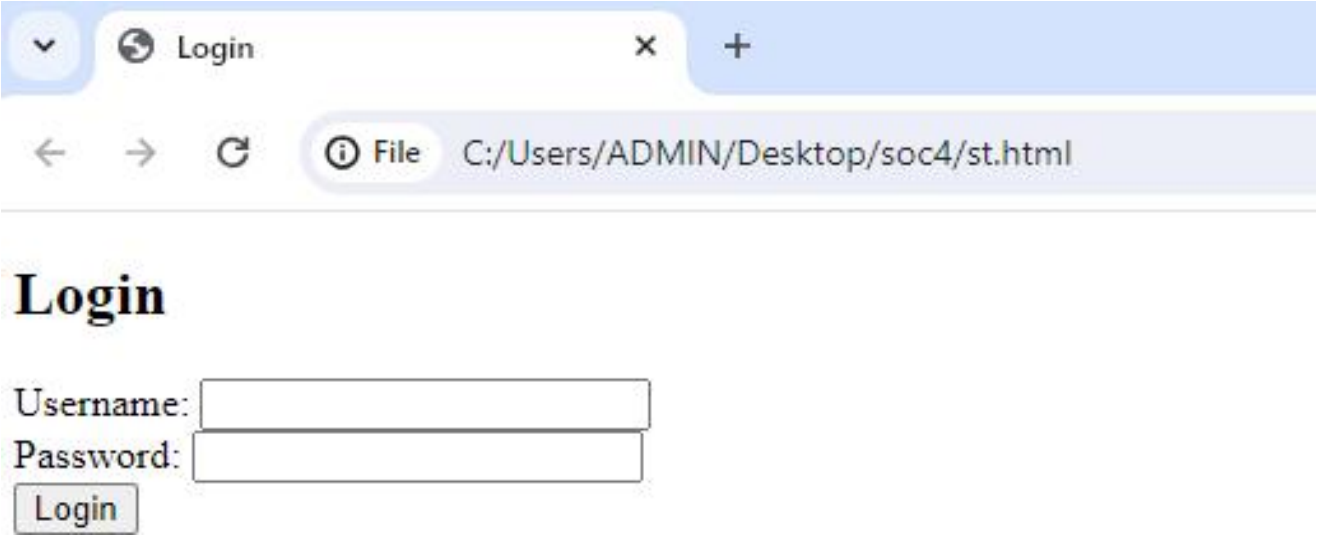
```
function getStockPrice() {
    // generate a random price between 100 and 200
    return Math.floor(Math.random() * 100) + 100;
  }
function logStockPrice() {
    const price = getStockPrice();
    console.log(`Stock price: ${price}`);
  }
 // log the stock price every three seconds
  const intervalId = setInterval(logStockPrice, 3000);
 // stop logging after 10 seconds
  setTimeout(() => clearInterval(intervalId), 10000);
```



C) Validate the user by creating a login module. Hints: (i) Create a file login.js with a User class. (ii) Create a validate method with username and password as arguments. (iii) If the username and password are equal it will return "Login Successful"

```
 <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login</title>
</head>
<body>
<h2>Login</h2>
<form id="login-form">
  <div>
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required>
  </div>
  <div>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required>
  </div>
  <button type="submit">Login</button>
</form>
<script src="login.js"></script>
<script>
  const loginForm = document.getElementById('login-form');
 loginForm.addEventListener('submit', function(event) {
    event.preventDefault(); // Prevent form submission
```

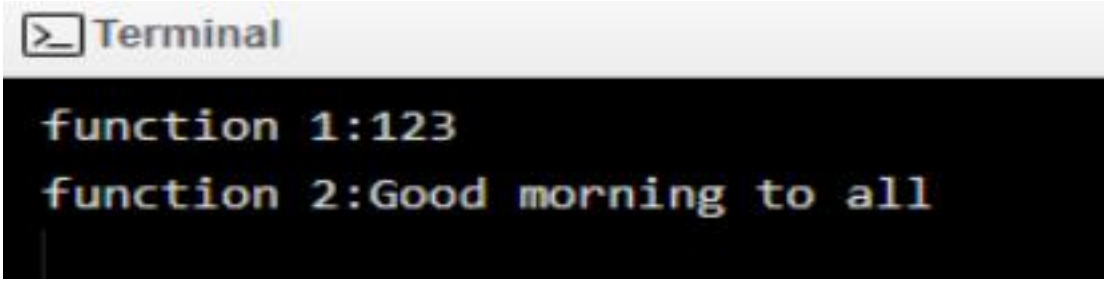
```
    const username = document.getElementById('username').value;

    const password = document.getElementById('password').value;

// Validate user

    const user = new User();

    const result = user.validate(username, password);

 if (result === "Login Successful") {

      alert("Login Successful");

      // Redirect or perform other actions upon successful login

    } else {

      alert("Invalid username or password");

    }

  });

</script>

</body>

</html>
```



Experiment-06

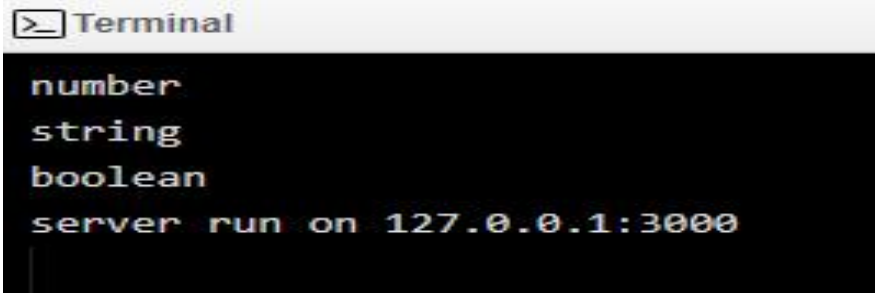
A) Verify how to execute different functions successfully in the Node.js platform

```
function myData()

{ return 123;

}

console.log("function 1:"+myData());

function myValue() {

return "Good morning to all";

}

console.log("function 2:"+myValue());
```



B) Write a program to show the workflow of JavaScript code executable by creating web server in Node.js

```
var a=8;

var b="sandy";

var d=true;

console.log(typeof(a));

console.log(typeof(b));

console.log(typeof(d));

var http=require("http");

const host="127.0.0.1";

const port=3000;

var server =http.createServer(function(){})

server.listen(port, host,

function(){ console.log("server run on

"+host+":"+port);

})
```



C) Write a Node.js module to show the workflow of Modularization of Node application

MAIN.js

```
// Importing the module
const calculator = require('./calculator');
console.log("Addition:", calculator.add(5, 3));
console.log("Subtraction:", calculator.subtract(10, 4));
console.log("Multiplication:", calculator.multiply(2, 6));
console.log("Division:", calculator.divide(20, 5));
```

CALCULATOR.js

```
function add(a, b)
  { return a + b;
}
function subtract(a, b)
  { return a - b;
}
function multiply(a, b)
  { return a * b;
}
function divide(a, b)
  { if (b === 0) {
     return "Cannot divide by zero!";
  }
  return a / b;
}
module.exports =
  { add,
  subtract,
  multiply,
  divide
};
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\ADMIN\Desktop\soc4> node main.js
Addition: 8
Subtraction: 6
Multiplication: 12
Division: 4
PS C:\Users\ADMIN\Desktop\soc4>
```

D) Write a program to show the workflow of restarting a Node application.

SERVER.js

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

const app = express();

const bodyParser = require("body-parser");

// Middleware to parse urlencoded form data

const urlencodedParser =

bodyParser.urlencoded({ extended: false });

// Serve static files from the 'public' directory

app.use(express.static('public'));

// Route to serve the HTML file

app.get('/html/sample1.html', function(req, res) {

   res.sendFile(__dirname + '/html/sample1.html');

});

// Route to handle form submission

app.post('/login', urlencodedParser, (req, res) => {

   console.log(req.body.fname);

   console.log(req.body.lname);

   res.send(req.body.lname);

});

// Start the server

app.listen(4000, () => {

   console.log("Server is running");

});
```

SAMPLE1.html

```
<!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>hello</title>

</head>

<body>

<form method="POST"
action="http://127.0.0.1:4000/login">

<input type="text" name="fname"/>

<input type="text" name="lname"/>

<input type="submit">

</form>

</body>

</html>
```

E) Create a text file src.txt and add the following data to it. Mongo, Express, Angular, Node

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

// Reading file

fs.readFile('src.txt', (err, data) =>

   { if (err) {

      throw err;

   } else {

      console.log("File content:");

      console.log(data.toString());

   }

});

// Writing to file

fs.writeFile('test.txt', 'Good morning', (err) =>

   { if (err) {

      console.log(err);

   } else {

      console.log("Data written to the file");

   }

});

// Appending to file

fs.appendFile('test.txt', ' everyone', (err) =>

   { if (err) {

      console.log(err);

   } else {

      console.log("Data appended to the file");

   }

});

// Opening file
```

```
fs.open('test.txt', 'r+', (err, f) =>
  { if (err) {
     console.log(err);
  } else {
     console.log("File opened successfully");
     console.log(f);
  }
});
// Deleting file
fs.unlink('test.txt', (err) => {
  if (err)
     { console.log(err);
  } else {
     console.log("File deleted successfully");
  }
});
```

```
PS C:\Users\ADMIN\Desktop\soc4> node kk.js
File opened successfully
5
File deleted successfully
Data appended to the file
Data written to the file
File content:
Mongo, Express, Angular, Node
```

Experiment-07

A) Implement routing for the AdventureTrails application by embedding the necessary code in the routes/route.js file.

ROUTE.js

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

const router = express.Router();

router.get('/', (req, res) => {

   res.send('Things - get data');

});

router.post('/', (req, res) =>

   { res.send('hello world');

});

module.exports = router;
```

APP.js

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

const app = express();

const things = require('./route');

app.use('/things', things);

app.listen(8000, '127.0.0.1', () => {

   console.log('Server is running on http://127.0.0.1:8000');

});
```

```
PS C:\Users\ADMIN\Desktop\soc4> node app.js
Server is running on http://127.0.0.1:8000
```

B) In myNotes application: (i) we want to handle POST submissions. (ii) display customized error messages. (iii) perform logging.

Node.js file

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

const bodyParser = require("body-parser");

const app = express();

const urlencodedparser = bodyParser.urlencoded({extended:false});

// Middleware function to serve static files

app.use(express.static('public'));

// Serving HTML file

app.get('/html/ac.html', function(req, res) {

  res.sendFile("C:/Users/user/OneDrive/Desktop/soc4" + "/html/ac.html");

});

// POST method handling

app.post('/login', urlencodedparser, (req, res) => {

  // Logging

  console.log("Received POST request.");

  console.log("First Name:", req.body.fname);

  console.log("Last Name:", req.body.lname); // Corrected to "lname"

  // Send response

  res.send(req.body.fname + " " + req.body.lname);

});

// Listening to port 4000

app.listen(4000, () => {

  console.log("Server is running on port 4000");

});
```

Html file

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>hello</title>

</head>

<body>

<form method="POST" action="http://127.0.0.1:4000/login">

<input type="text" name="fname"/>

<input type="text" name="lname"/> <!-- Corrected "lname" to "lname" assuming it's intended for "last name" -->

<input type="submit">

</form>

</body>

</html>
```

C) Write a Mongoose schema to connect with MongoDB.

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

var app =express();

mongoose.set('strictQuery', false);

mongoose.connect('mongodb://127.0.0.1:27017/CT',

{ useNewUrlParser:true,

useUnifiedTopology:true

}).then(()=>{console.log('mongodb connected...')})

app.listen(8000,()=>{

console.log("server run on http://127.0.0.1:8000");

});
```

MongoDB is connected

D) Write a program to wrap the Schema into a Model object.

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

const Note = require('./models/note'); // Importing the Note model

// Connect to MongoDB

mongoose.connect('mongodb://127.0.0.1:27017/CT',

{ useNewUrlParser: true,

  useUnifiedTopology: true

})

.then(() => {

  console.log('MongoDB connected...');

  // Your application logic goes here

})

.catch((err) => {

  console.error('MongoDB connection error:', err);

});
```

```
PS C:\Users\ADMIN\Desktop\soc4> MongoDB connected...
>> Note saved successfully: { _id: 6019bd72c2eaf52308f92a0e, title: 'First Note', content: 'This is the content of my first note.', createdAt: 2021-02-03T10:42:26.917Z }
>> All notes: [ { _id: 6019bd72c2eaf52308f92a0e, title: 'First Note', content: 'This is the content of my first note.', createdAt: 2021-02-03T10:42:26.917Z } ]
>>
```

Experiment-08

A) Write a program to perform various CRUD (Create-Read-Update-Delete) operations using Mongoose library functions.

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

// Connect to MongoDB

mongoose.connect('mongodb://localhost:27017/myDatabase',

{ useNewUrlParser: true,

  useUnifiedTopology: true,

})

.then(() => {

  console.log('Connected to MongoDB...');

  // Perform CRUD operations

  createNote();

})

.catch(error => console.error('Error connecting to MongoDB:', error));

// Define schema

const noteSchema = new

  mongoose.Schema({ title: String,

  content: String,

});

// Define model

const Note = mongoose.model('Note', noteSchema);

// Create operation

async function createNote()

  { try {

    const newNote = new

      Note({ title: 'First Note',

      content: 'This is the content of my first note.',
```

```
    });
    const savedNote = await newNote.save();
    console.log('Note created:', savedNote);
    readNotes();
  } catch (error) {
    console.error('Error creating note:', error);
  }
}
// Read operation
async  function  readNotes()
  { try {
    const notes = await Note.find();
    console.log('All  notes:',  notes);
    updateNote(notes[0]._id);
  } catch (error) {
    console.error('Error reading notes:', error);
  }
}
// Update operation
async function updateNote(noteId)
  { try {
    const updatedNote = await Note.findOneAndUpdate(
      { _id: noteId },
      { $set: { content: 'Updated content' } },
      { new: true }
    );
    console.log('Note updated:', updatedNote);
    deleteNote(updatedNote._id);
  } catch (error) {
```

```
  console.error('Error updating note:', error);
 }
}
// Delete operation
async function deleteNote(noteId)
 { try {
  const deletedNote = await Note.findByIdAndDelete(noteId);
  console.log('Note deleted:', deletedNote);
 } catch (error) {
  console.error('Error deleting note:', error);
 }
}
```

```
Connected to MongoDB...
Note created: {
  title: 'First Note',
  content: 'This is the content of my first note.',
  _id: new ObjectId('66121f641d5da21906404546'),
  __v: 0
}
All notes: [
  {
    _id: new ObjectId('66121f641d5da21906404546'),
    title: 'First Note',
    content: 'This is the content of my first note.',
    __v: 0
  }
]
Note updated: {
  _id: new ObjectId('66121f641d5da21906404546'),
  title: 'First Note',
  content: 'Updated content',
  __v: 0
}
Note deleted: {
  _id: new ObjectId('66121f641d5da21906404546'),
  title: 'First Note',
  content: 'Updated content',
  __v: 0
}
```

Experiment-09

A) On the page, display the price of the mobile-based in three different colors. Instead of using the number in our code, represent them by string values like GoldPlatinum, PinkGold, SilverTitanium.

```
enum MobileColor{

   GoldPlatinum = "GoldPlatinum", PinkGold = "PinkGold",

   SilverTitanium = "SilverTitanium",

   }

   const mobilePrice = 999.99;

   console.log('The price of the mobile in ${MobileColor.GoldPlatinum} color is $${mobilePrice}.');

   console.log('The price of the mobile in ${MobileColor.PinkGold} color is $${mobilePrice}.');

   console.log('The price of the mobile in ${MobileColor. SilverTitanium} color is $${mobilePrice}.');
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
The price of the mobile in GoldPlatinum color is $999.99.
The price of the mobile in PinkGold color is $999.99.
The price of the mobile in SilverTitanium color is $999.99.
```

B) Define an arrow function inside the event handler to filter the product array with the selected product object using the productId received by the function. Pass the selected product object to the next screen.

```
interface
   Product{ id:
   number; name:
   string; price:
   number;
   };
const products: Product[] = [
   { id: 1, name: "Product 1", price: 9.99 },
   { id: 2, name: "Product 2", price: 19.99 },
   { id: 3, name: "Product 3", price: 29.99 },
   ];
   function handleClick(productId: number) {
   const selectedProduct = products.find((product) => product.id === productId);
   if (selectedProduct) {
   // Navigate to the next screen with the selected product object
   console.log("Selected product:", selectedProduct);
   }
   }
   // Example usage
   handleClick(2);
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
Selected product: { id: 2, name: 'Product 2', price: 19.99 }
```

C) Consider that developer needs to declare a function - getMobileByVendor which accepts string as input parameter and returns the list of mobiles.

```
type  Mobile  =
  { brand: string;
  model:  string;
  price: number;
};
const mobiles: Mobile[] = [
  { brand: "Apple", model: "iPhone 12", price: 999 },
  { brand: "Samsung", model: "Galaxy S21", price: 799 },
  { brand: "Google", model: "Pixel 5", price: 699 },
];
function getMobileByVendor(vendor: string): Mobile[] {
  const filteredMobiles = mobiles.filter((mobile) => mobile.brand === vendor);
  return filteredMobiles;
}
// Example usage
const appleMobiles = getMobileByVendor("Apple");
console.log(appleMobiles);
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
[ { brand: 'Apple', model: 'iPhone 12', price: 999 } ]
```

D) Consider that developer needs to declare a manufacturer's array holding 4 objects with id and price as a parameter and needs to implement an arrow function - myfunction to populate the id parameter of manufacturers array whose price is greater than or equal

```
type Manufacturer =
   { id: number;
   price: number;
};
const manufacturers: Manufacturer[] = [
   { id: 1, price: 9.99 },
   { id: 2, price: 19.99 },
   { id: 3, price: 29.99 },
   { id: 4, price: 39.99 }
];
const myFunction = (price: number): void =>
   { manufacturers.forEach((manufacturer) => {
      if (manufacturer.price > price)
         { manufacturer.id += 10;
      }
   });
};
// Example usage
console.log(manufacturers);
myFunction(30);
console.log(manufacturers);
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
[
  { id: 1, price: 9.99 },
  { id: 2, price: 19.99 },
  { id: 3, price: 29.99 },
  { id: 4, price: 39.99 }
]
[
  { id: 1, price: 9.99 },
  { id: 2, price: 19.99 },
  { id: 3, price: 29.99 },
  { id: 14, price: 39.99 }
]
```

E) Declare a function - getMobileByManufacturer with two parameters namely manufacturer and id, where manufacturer value should passed as Samsung and id parameter should be optional while invoking the function, if id is passed as 101.

```
type Mobile =
  { id: number;
  brand: string;
  model: string;
  price: number;
};
const mobiles: Mobile[] = [
  { id: 100, brand: "Apple", model: "iPhone 12", price: 999 },
  { id: 101, brand: "Samsung", model: "Galaxy S21", price: 799 },
  { id: 102, brand: "Google", model: "Pixel 5", price: 699 }
];
function getMobileByManufacturer(manufacturer: string, id: number | null = null): Mobile | null
  { const filteredMobiles = mobiles.filter((mobile) => mobile.brand === manufacturer);
  if (id === null) {
    return filteredMobiles.length > 0 ? filteredMobiles[0] : null;
  } else {
    const mobile = filteredMobiles.find((mobile) => mobile.id === id);
    return mobile || null;
  }
}
// Example usage
const samsungMobiles = getMobileByManufacturer("Samsung");
console.log(samsungMobiles);
const samsungMobile101 = getMobileByManufacturer("Samsung", 101);
console.log(samsungMobile101);
```

```
const appleMobiles = getMobileByManufacturer("Apple");

console.log(appleMobiles);

const appleMobile101 = getMobileByManufacturer("Apple", 101);

console.log(appleMobile101);
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
{ id: 101, brand: 'Samsung', model: 'Galaxy S21', price: 799 }
{ id: 101, brand: 'Samsung', model: 'Galaxy S21', price: 799 }
{ id: 100, brand: 'Apple', model: 'iPhone 12', price: 999 }
null
```

Experiment-10

A) Implement business logic for adding multiple Product values into a cart variable which is type of string array.

```
let cart: string[] = [];

function addToCart(...products: string[]): string[]

   { cart = [...cart, ...products];

   return cart;

}

// Example usage

console.log(addToCart("Product A"));

console.log(addToCart("Product B", "Product C"));

console.log(addToCart("Product D", "Product E", "Product F"));
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
[ 'Product A' ]
[ 'Product A', 'Product B', 'Product C' ]
[
  'Product A',
  'Product B',
  'Product C',
  'Product D',
  'Product E',
  'Product F'
]
```

B) Declare an interface named - Product with two properties like productId and productName with a number and string datatype and need to implement logic to populate the Product details.

```
// Define the Product interface

interface Product {

   productId: number;

   productName: string;

}

// Function to create a Product object

function createProduct(id: number, name: string): Product {

   // Create and return the Product object

   return {

      productId: id,

      productName: name,

   };

}

// Example usage

const productA = createProduct(1, "Product A");

console.log(productA);
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
{ productId: 1, productName: 'Product A' }
```

C) Declare an interface named - Product with two properties like productId and productName with the number and string datatype and need to implement logic to populate the Product details.

```
// Define the Product interface

interface Product {

   productId: number;

   productName: string;

}

// Function to print product details

function printProductDetails(product: Product)

   { console.log(`Product ID: ${product.productId}`);

   console.log(`Product Name: ${product.productName}`);

}

// Example usage

const productA: Product = { productId: 1, productName: "Product A" };

const productB: Product = { productId: 2, productName: "Product B" };

printProductDetails(productA);

printProductDetails(productB);
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
Product ID: 1
Product Name: Product A
Product ID: 2
Product Name: Product B
```

D) Declare an interface with function type and access its value

```
type MyFunc = (a: number, b: number) => number;

const myFunc: MyFunc = (a, b) => a + b;

console.log(myFunc(2, 3));
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
5
```

Experiment-11

A) Declare a productList interface which extends properties from two other declared interfaces like Category,Product as well as implementation to create a variable of this interface type

```
interface Category

   { name: string;

   id: number;

}

interface Product

   { name: string;

   price: number;

}

interface ProductList extends Category, Product

   { description: string;

}

const myProductList: ProductList =

   { name: 'My Product List',

   id: 123,

   price: 19.99,

   description: `This is a list of my products`,

};

console.log(myProductList);
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
{
  name: 'My Product List',
  id: 123,
  price: 19.99,
  description: 'This is a list of my products'
}
```

B) Consider the Mobile Cart application, Create objects of the Product class and place them into the productlist array.

```
class Product {
   constructor(public name: string, public price: number)
      { console.log("Name of the product is " + name);
      console.log("Price of the product is " + price);
   }
}
const productList: Product[] = [
   new Product('iPhone 13 Pro', 999),
   new Product('Samsung Galaxy S21', 799),
   new Product('Google Pixel 6', 599),
];
console.log(productList);
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
Name of the product is iPhone 13 Pro
Price of the product is 999
Name of the product is Samsung Galaxy S21
Price of the product is 799
Name of the product is Google Pixel 6
Price of the product is 599
[
  Product { name: 'iPhone 13 Pro', price: 999 },
  Product { name: 'Samsung Galaxy S21', price: 799 },
  Product { name: 'Google Pixel 6', price: 599 }
]
```

C) Declare a class named - Product with the below-mentioned declarations: (i) productId as number property (ii) Constructor to initialize this value (iii) getProductId method to return the message "Product id is <<id value>>".

```
class Product {

 constructor(public productId: number) {}

 getProductId() {

 return `Product id is ${this.productId}`;

}

}

const myProduct = new Product(123);

console.log(myProduct.getProductId());
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
Product id is 123
```

D) Create a Product class with 4 properties namely productId, productName, productPrice, productCategory with private, public, static, and protected access modifiers and accessing them through Gadget class and its methods

```
class Product
   { constructo
   r(
      private productId: number,
      public productName: string,
      protected productPrice: number,
      readonly productCategory: string
   ) {}
 public getProductDetails(): string {
return `Product ID: ${this.productId}, Name: ${this.productName}, Price: ${this.productPrice}, Category: ${this.productCategory}`;
   }
static getProductName(product: Product): string
      { return product.productName;
   }
}
class Gadget {
   private products: Product[] = [];
public addProduct(product: Product) {
      this.products.push(product);
   }
public getProductDetails() {
      this.products.forEach((product) => console.log(product.getProductDetails()));
   }
 public getProductNames() {
      this.products.forEach((product) => console.log(Product.getProductName(product)));
   }
}
const myGadget = new Gadget();
```

```
const myProduct = new Product(1, 'iPhone 13 Pro', 999, 'Mobiles');
```

```
myGadget.addProduct(myProduct);

console.log('Product Details:');

myGadget.getProductDetails();

console.log('Product Names:');

myGadget.getProductNames();
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
Product Details:
Product ID: 1, Name: iPhone 13 Pro, Price: 999, Category: Mobiles
Product Names:
iPhone 13 Pro
```

Experiment-12

A) Create a Product class with 4 properties namely productId and methods to setProductId() and getProductId().

```
class Product {
   private productId: number;
public setProductId(productId: number)
      { this.productId = productId;
   }
public getProductId()
      { return this.productId;
   }
}
const myproduct=new Product();
myproduct.setProductId(123);
console.log(myproduct.getProductId());
```

```
C:\Users\ADMIN\Desktop\soc4>node mp.js
123
```

B) Create a namespace called ProductUtility and place the Product class definition in it. Import the Product class inside productlist file and use it.

```
// productUtility.ts

export class Product {

   constructor(

      private productId: number,

      public productName: string,

      private productPrice: number,

      public productCategory: string

   ) {}

 public getProductDetails(): string {

      return `Product ID: ${this.productId}, Name: ${this.productName}, Price: ${this.productPrice},
Category: ${this.productCategory}`;

   }

}

// productlist.ts

import { Product } from './productutility';

const myProduct = new Product(1, 'iPhone 13 Pro', 999, 'Mobiles');

console.log(myProduct.getProductDetails());
```

```
C:\Users\ADMIN\Desktop\soc4>node productlist.js
Product ID: 1, Name: iPhone 13 Pro, Price: 999, Category: Mobiles
```

C) Consider the Mobile Cart application which is designed as part of the functions in a module to calculate the total price of the product using the quantity and price values and assign it to a totalPrice variable.

```
// app.ts

import { calculateTotalPrice } from './pu';

const quantity = 5;

const price = 10;

const totalPrice = calculateTotalPrice(quantity, price);

console.log(`Total price: ${totalPrice}`);

// pu.ts

export function calculateTotalPrice(quantity: number, price: number): number

   { return quantity * price;

}
```

```
C:\Users\ADMIN\Desktop\soc4>node app.js
Total price: 50
```

D) Create a generic array and function to sort numbers as well as string values.

```
function sortArray<T>(inputArray: T[]): T[] {
   const sortedArray = inputArray.slice().sort(); // Make a copy of the input array to avoid mutating it
   return sortedArray;
}
const stringArray: string[] = ["banana", "apple", "orange", "grape"];
const sortedStringArray = sortArray(stringArray);
console.log(sortedStringArray); // Output: ["apple", "banana", "grape", "orange"]
const numberArray: number[] = [3, 6, 2, 8, 1, 5];
const sortedNumberArray = sortArray(numberArray);
console.log(sortedNumberArray);
```

```
C:\Users\ADMIN\Desktop\soc4>node pu.js
[ 'apple', 'banana', 'grape', 'orange' ]
[ 1, 2, 3, 5, 6, 8 ]
```