**MEAN STACK TECHNOLOGIES LAB MODULE – I**

**Skill Oriented Course-III1a) Course Name: HTML5 - The Language**

**Module Name: Case-insensitivity, Platformindependency, DOCTYPE Declaration, Types of Elements, HTML Elements - Attributes, Metadata Element 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.**

**Program:**

<!DOCTYPE HTML>

<head>

<title>ShopTime website</title>

<meta charset="UTF-8">

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

<meta name="keywords" content="clothing,footwear,shopping">

<meta name="author" content="Myself">

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

</Head>

<body bgcolor="cyan">

<h1 align="center"><i>ShopTime<i></h1>

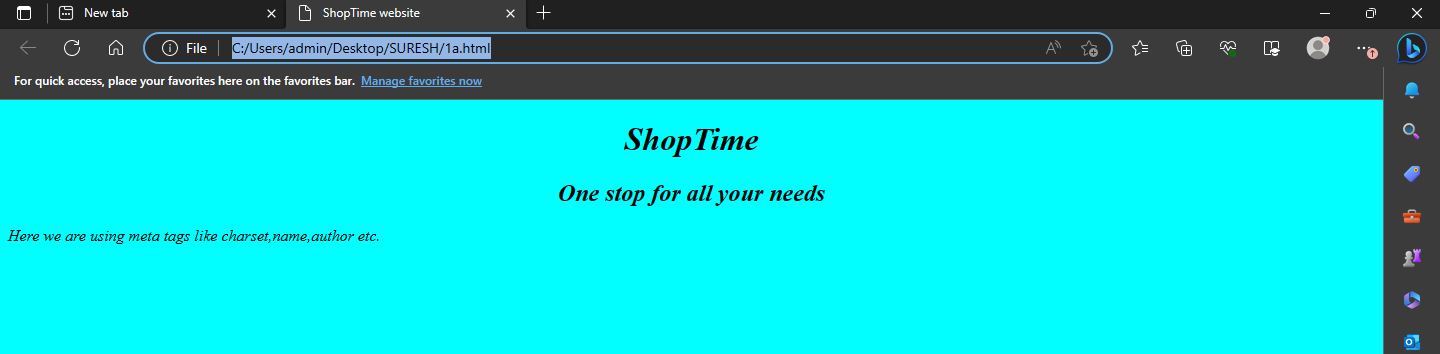
<h2 align="center"><i>One stop for all your needs<i></h2>

<p>Here we are using meta tags like charset,name,author etc.</p>

</body>

</html>

**Outpu**



**1b) Course Name: HTML5 - The Language**

**Module Name: Sectioning Elements Enhance the Homepage.html of IEKart's Shopping Application by adding appropriate sectioning elements.**

**Program:**

<!DOCTYPE HTML>

<head>

<title>ShopTime website</title>

</Head>

<body bgcolor="cyan">

<h1 align="center"><i>ShopTime<i></h1>

<h2 align="center"><i>One stop for all your needs<i></h2>

<nav align="center"><h3>

Home || Login || Register || Wishlist || My Orders || Help</h3></nav>

<main>

<section>

<p>Clothing</p>

</section>

<section>

<p>Footwear</p>

</section>

<section>

<p>Electronics</p>

</section>

<section>

<p>Furniture</p>

</section>

<section>

<p>Cosmetics</p>

</section>

</main>

<article>

<h1>Special Offer</h1>

<aside>

<p>Download our app at PlayStore and win exciting prizes.</p></aside>

</article><header>

</body>

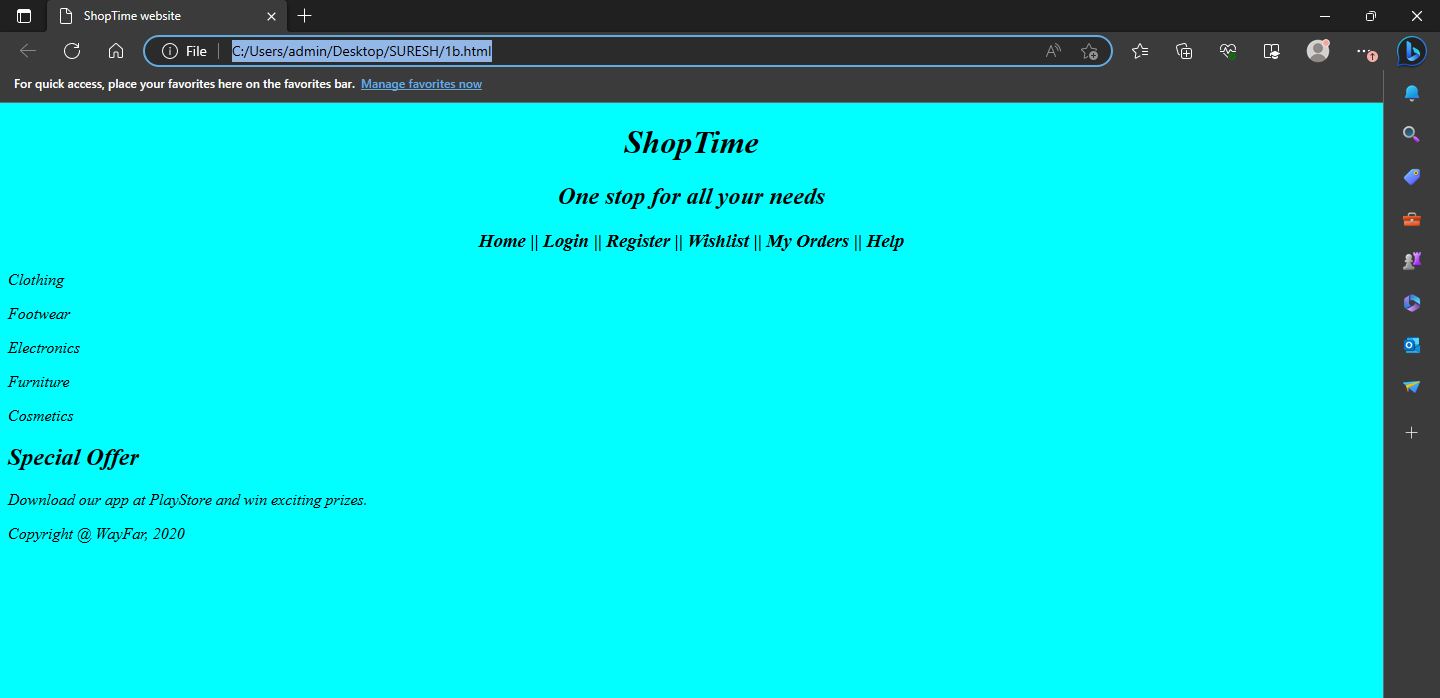
<footer>

Copyright @ WayFar, 2020

</footer>

</html>

**Output:**

****

**1c) Course Name: HTML5 - The Language**

**Module Name: Paragraph Element, Division and Span Elements, List Element Make use of appropriate grouping elements such as list items to "About Us" page of IEKart's Shopping Application.**

**Program:**

<!DOCTYPE HTML>

<head>

<title>ShopTime website</title>

</Head>

<body bgcolor="lavender">

<h1 align="center"><i>ShopTime<i></h1>

<h2 align="center"><i>One stop for all your needs<i></h2>

<nav align="center"><h3>

Home || Login || Track Order </h3>

</nav>

<center>

<p><img src="https://www.logomaker.com/wp-content/uploads/2018/01/FLS-Blog-Black-Logos\_Hero.jpg" alt="Top shopping brands" width="950" height="300"</p>

</center>

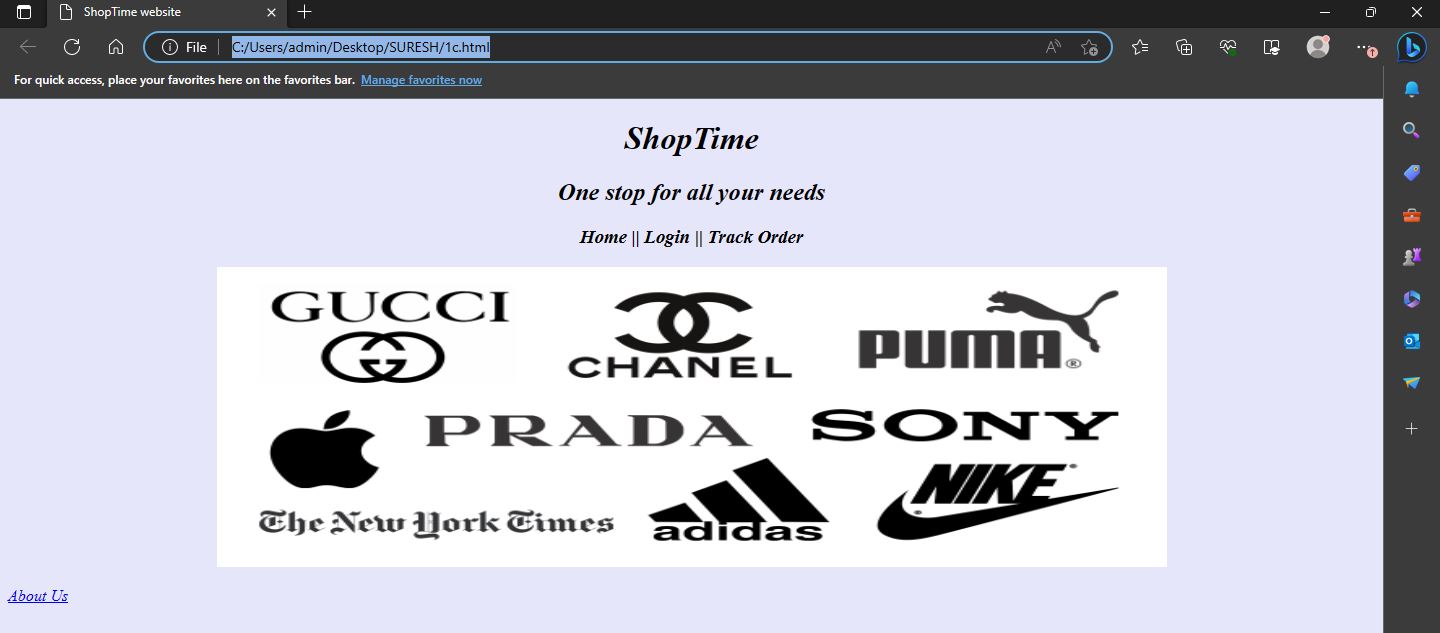
</body>

<footer>

<a href=aboutus.html">About Us</a>

</footer>

</html>

**Output:** ****

**Aboutus.html**

<!DOCTYPE HTML>

<html>

<head>

<title>ShopTime website</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

</Head>

<body bgcolor="cyan">

<div class="about-section">

<h1>About Us Page</h1>

<p><span style="color:blue;font-weight:bold"><i>ShopTime</i></span> is an indigenous e-commerce website discovering new ways to satisfy customer's needs.</p></div>

<h2>Founders</h2>

<ul>

<li>Jane Doe - Foumder &CEO</li>

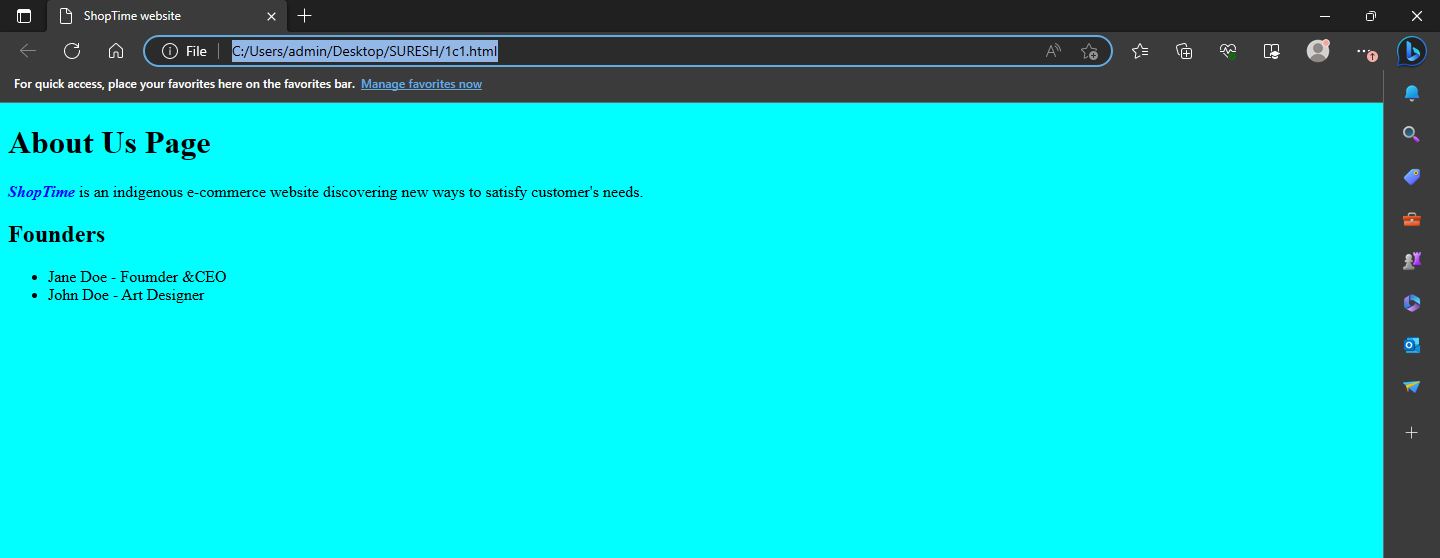
<li>John Doe - Art Designer</li>

</ul>

</body>

</html>

**Output:**

****

**1d) Course Name: HTML5 - The Language**

**Module Name: Link Element 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**.

**Program**:

<!DOCTYPE HTML>

<head>

<title>ShopTime website</title>

<meta charset="UTF-8">

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

<meta name="keywords" content="clothing,footwear,shopping">

<meta name="author" content="Myself">

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

</Head>

<body bgcolor="lavender">

<h1 align="center"><i>ShopTime<i></h1>

<h2 align="center"><i>One stop for all your needs<i></h2>

<nav align="center"><h3>

<a href=”home.html”>Home</a> ||<a href=”login.html”> Login</a>

||<ahref=”trackorder.html”>Track Order</a></h3>

</nav>

<center>

<p><img src="homeimg.png" alt="Top shopping brands" width="1350" height="300"</p>

</center>

</body>

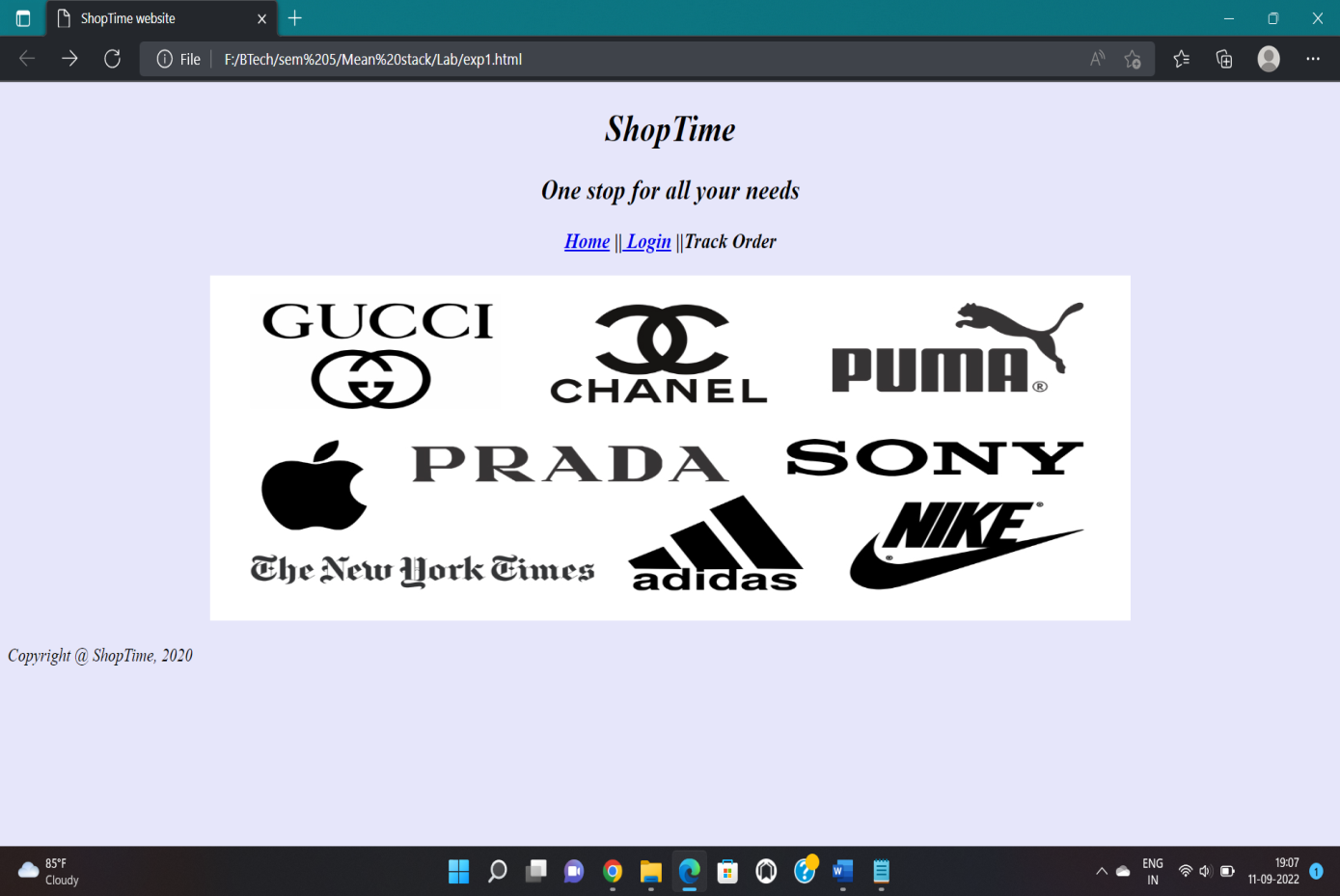
<footer>

Copyright @ ShopTime, 2020

</footer>

</html>

**Output:**

****

**1e) Course Name: HTML5 - The Language**

**Module Name: Character Entities Add the © symbol in the Home page footer of IEKart's Shopping application.**

**Program:**

<!DOCTYPE HTML>

<head>

<title>ShopTime website</title>

<meta charset="UTF-8">

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

<meta name="keywords" content="clothing,footwear,shopping">

<meta name="author" content="Myself">

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

</Head>

<body bgcolor="lavender">

<h1 align="center"><i>ShopTime<i></h1>

<h2 align="center"><i>One stop for all your needs<i></h2>

<nav align="center"><h3>

<a href=”home.html”>Home</a> ||<a href=”login.html”> Login</a> || <a href=”trackorder.html”>Track Order</a></h3>

</nav>

<center>

<p><img src="https://www.logomaker.com/wp-content/uploads/2018/01/FLS-Blog-Black-Logos\_Hero.jpg" alt="Top shopping brands" width="950" height="300"</p>

</center>

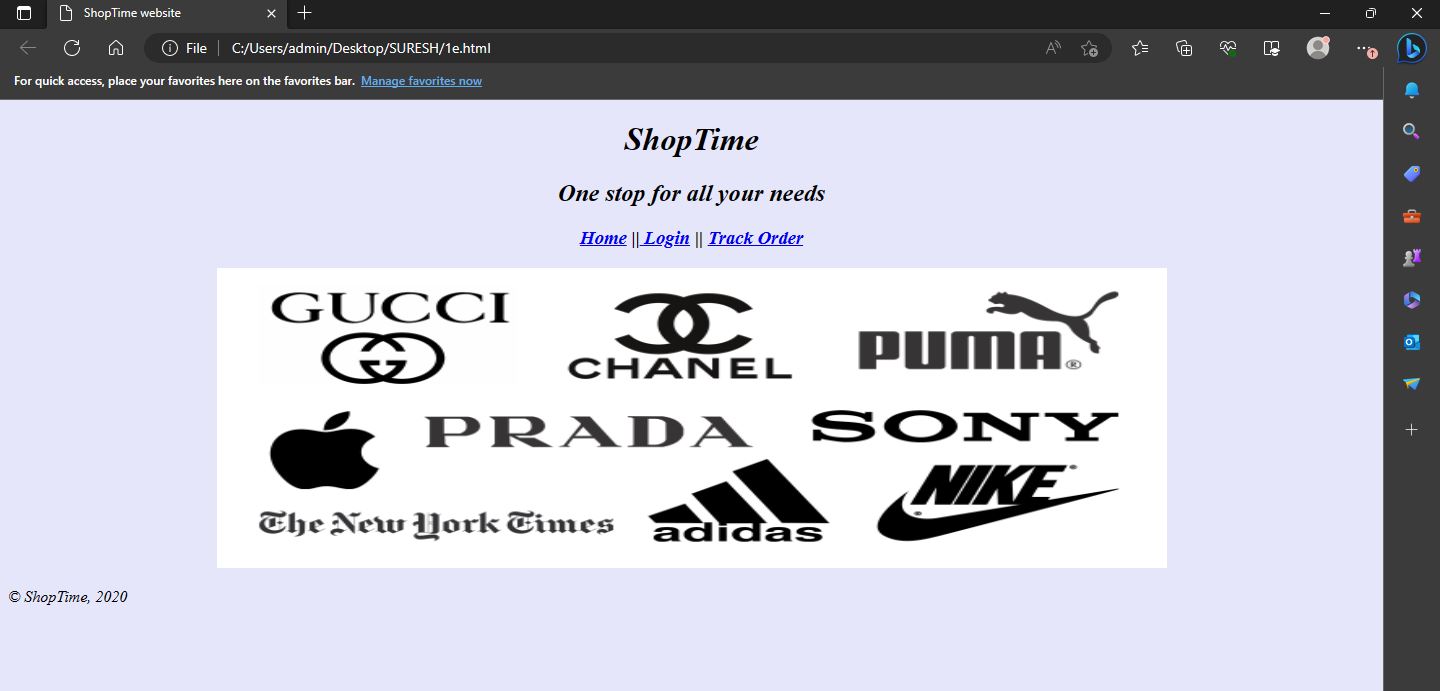
</body>

<footer>

&copy; ShopTime, 2020

</footer></html>

**Output:**

****

**1f) Course Name: HTML5 - The Language**

**Module Name: HTML5 Global Attributes Add the global attributes such as contenteditable, spellcheck, id etc. to enhance the Signup Page functionality of IEKart's Shopping application.**

**Program:**

<html>

<head><title>ShopTime</title></head>

<body bgcolor="cyan">

<h1 align="center"><i>ShopTime</i></h1>

<form>

<center>

<h3>Sign up</h3>

<table>

<tr><td>Name :</td><td><input type="text"></td></tr>

<tr><td>Email:</td><td><input type="email" contenteditable="true" spellcheck="true" ></td></tr>

<tr><td>User Name :</td><td><input type="text"></td></tr>

<tr><td>Date of birth:</td><td><input type="date"></td></tr>

<tr><td>Password :</td><td><input type="password"></td></tr>

<tr><td>Confirm Password :</td><td><input type="password"></td></tr>

<tr><td></td><td align="center"><input type="submit"></td></tr>

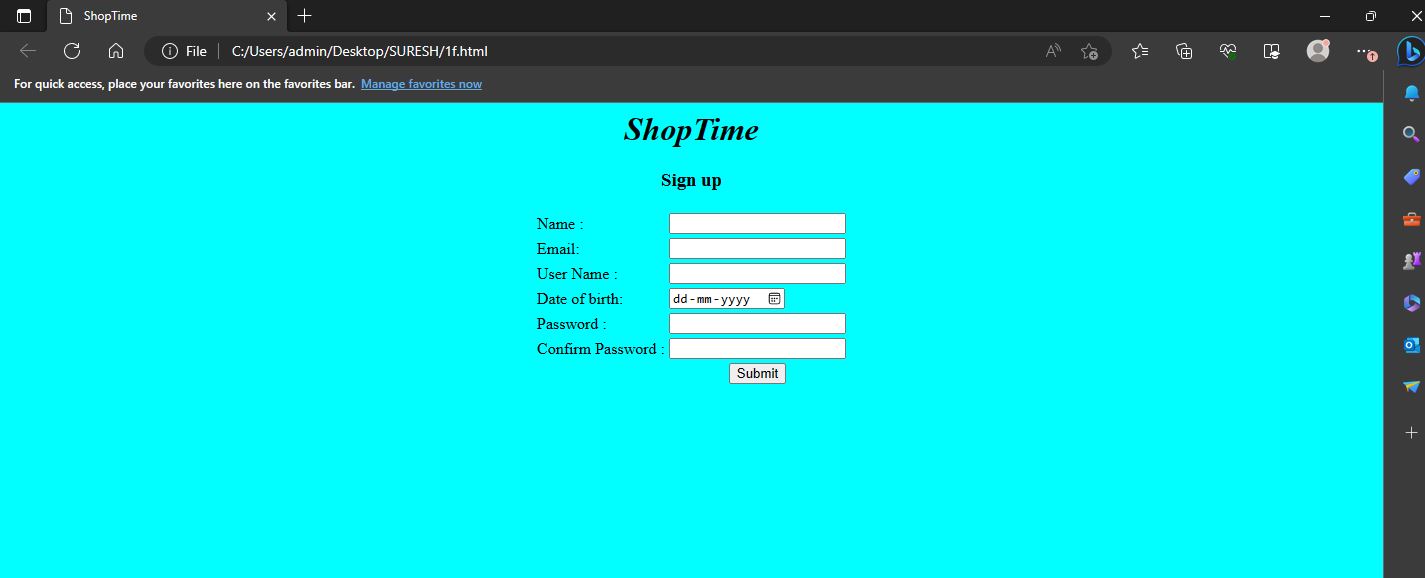
</center>

</form>

</body>

</html>

**Output:**



**2a) Course Name: HTML5 - The Language**

**Module Name: Creating Table Elements, Table Elements : Colspan/Rowspan Attributes, border, cellspacing, cellpadding attributes Enhance the details page of IEKart's Shopping application by adding a table element to display the available mobile/any inventories.**

**Program:**

<!DOCTYPE HTML>

<html>

<head>

<title>ShopTime Website</title>

</head>

<body bgcolor="lavender">

<table cellspacing="1" cellpadding="0" border="1" align="center">

<caption><h1>Electronics</h1></caption>

<b><tr bgcolor="white"><td>Smartwatches & Fitness trackers</td><td><img src="https://consumer.huawei.com/content/dam/huawei-cbg-site/common/mkt/pdp/wearables/watch-fit/dynamic/watch-fit/img/pc/huawei-watch-fit-personal-assistant-3.jpg" width = "120px" height = "120px" alt="Smartwatches"></td></tr>

<tr><td>Earbuds</td><td><img src="https://i.pinimg.com/originals/db/96/6c/db966cbb958a6c398e4e099f423ffb56.jpg" width = "130px" height = "130px" alt="Earbuds"></td></tr>

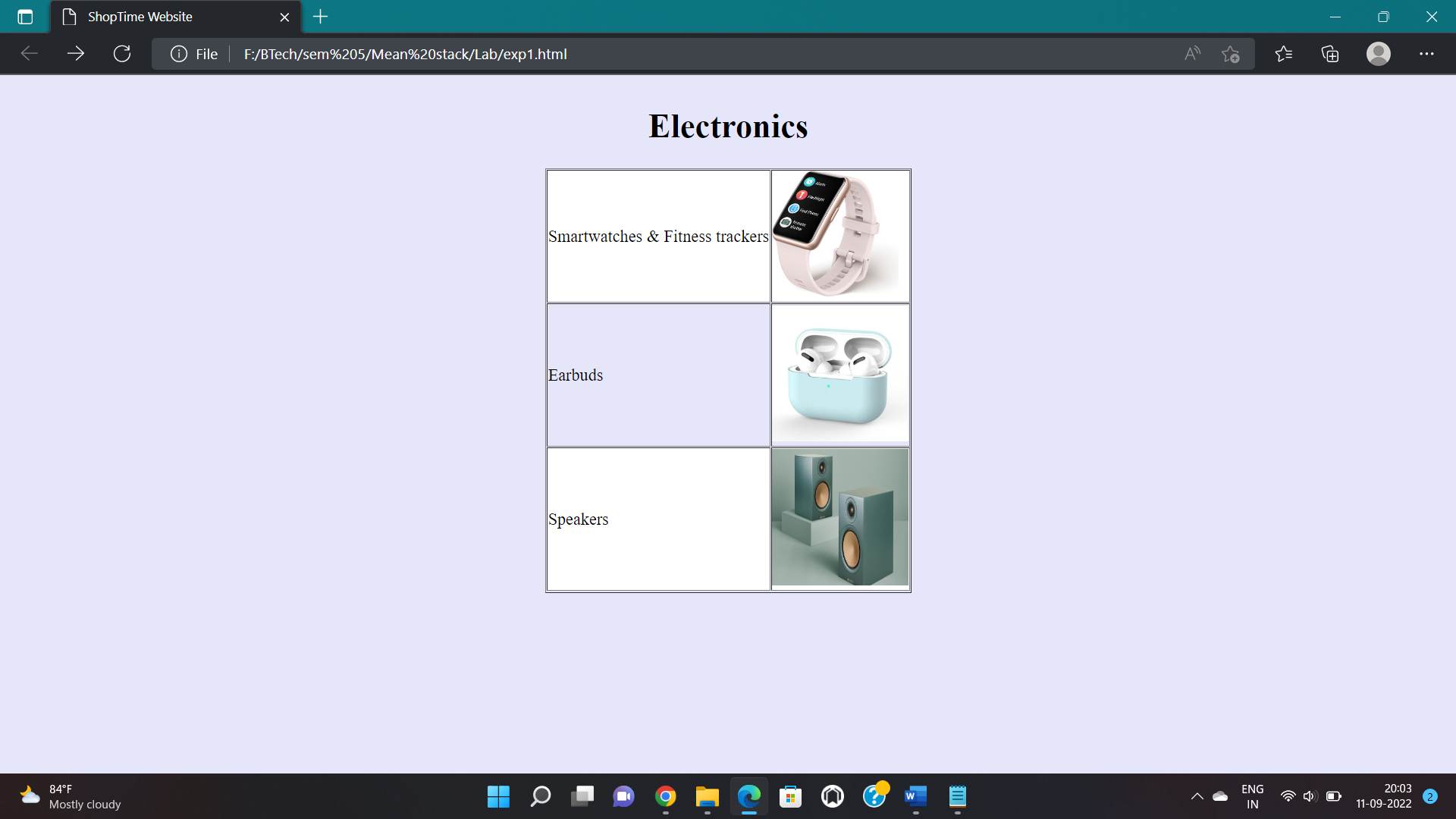
<tr bgcolor="white"><td>Speakers</td><td><img src="https://www.monitoraudio.com/site/assets/files/33185/silver100le-actionblock-2.jpg" width = "130px" height = "130px" alt="Speakers"></td></tr></b>

</table>

</body>

</html>

**Output:**

****

**2b) Course Name: HTML5 The Language**

**Module Name: Creating Form Elements, Color and Date Pickers, Select and Datalist Elements Using the form elements create Signup page for IEKart's Shopping application.**

**PROGRAM:**

<html>

<body bgcolor="lavender">

<form align="center">

<table align=center>

<caption><h1>Sign Up</h1></caption>

<tr><td><label>First Name:</label></td><td><input type="text"></td><br>

<tr><td><label>Email:</label></td><td><input type="email" ><br>

<tr><td><label>Date of birth:</label></td><td><input type="date"></td>

<tr><td><label>Gender: </label><td><input type="radio" name="gender" value="Male"> Male <input type="radio" name="gender" value="Female"> Female</td><br>

<tr><td><label>Mobile:</label></td><td><input type="number" ><br>

<tr><td><label>Username:</label></td><td><input type="text"><br>

<tr><td><label>Password:</label></td><td><input type="password" ><br>

<tr><td><label>Confirm Password:</label></td><td><input type="password" ><br>

<tr rowspan="3"><td><label>Address :<br></label></td><td><textarea rows="3" cols="30" ></textarea></td></tr><br><br>

</table>

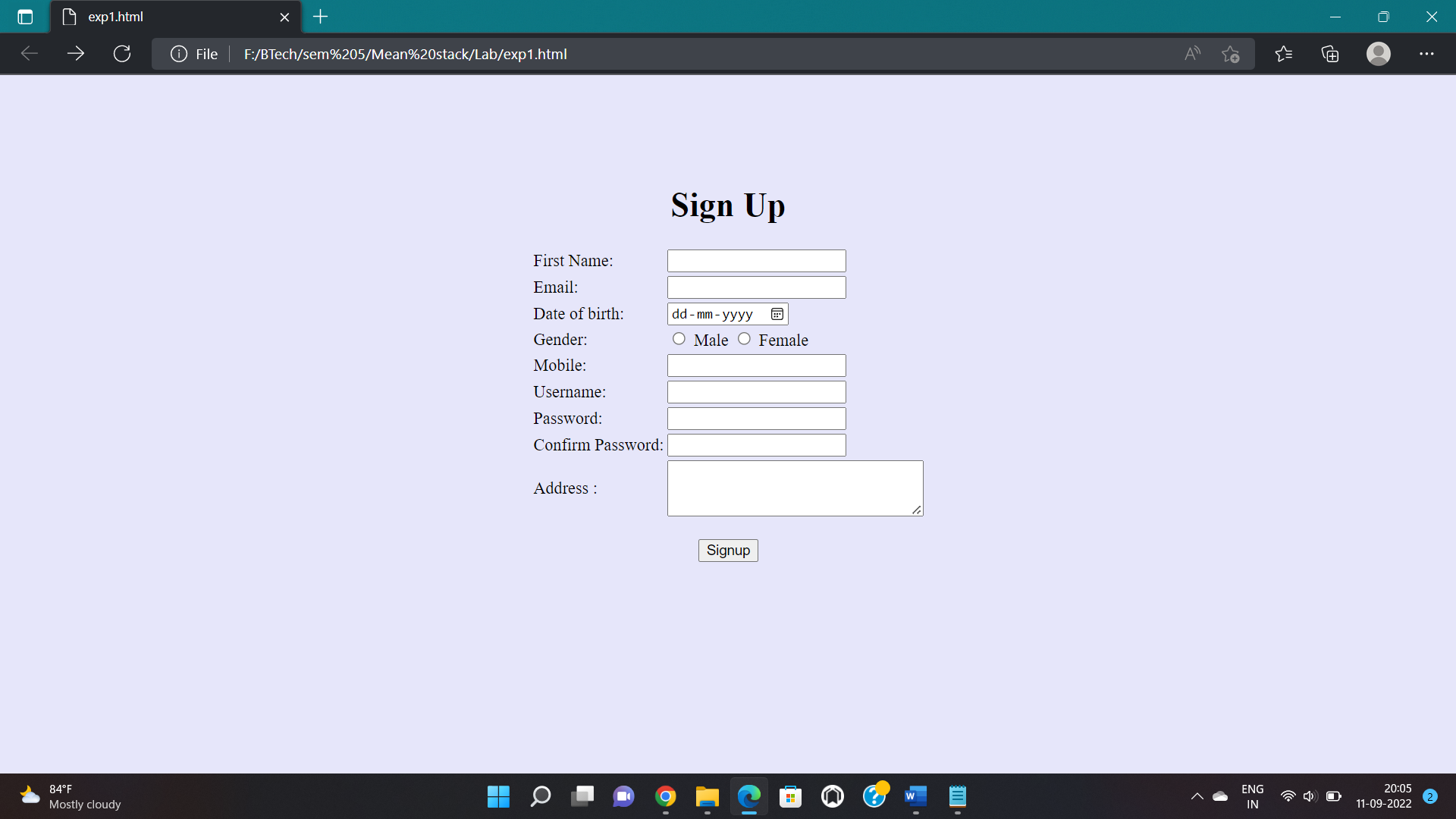
<br><button type="submit">Signup</button>

</form>

</body>

</html>

**Output:**

****

**2c) Course Name: HTML5 The Language**

**Module Name: Input Elements - Attributes Enhance Signup page functionality of IEKart's Shopping application by adding attributes to input elements.**

**Program:**

<html>

<body bgcolor="lavender">

<form align="center">

<table align=center>

<h1 align="center"><i>ShopTime</i></h1>

<caption><h3>Sign Up</h3></caption>

<tr><td><label>Name:</label></td><td><input type="text"></td><br>

<tr><td><label>Mobile:</label></td><td><input type="number" autocomplete="on"></td><br>

<tr><td><label>Date of birth:</label></td><td><input type="date"></td>

<tr><td><label>Gender: </label><td><input type="radio" name="gender" value="Male"> Male <input type="radio" name="gender" value="Female"> Female</td><br>

<tr><td><label>Email:</label></td><td><input type="email"><br>

<tr><td><label>Username:</label></td><td><input type="text" pattern="[A-Zaz]" maxlength="20" minlength="9"><br>

<tr><td><label>Password:</label></td><td><input type="password" placeholder="\*\*\*\*\*\*\*\*"><br>

<tr><td><label>Confirm Password:</label></td><td><input type="password" placeholder="\*\*\*\*\*\*\*\*"><br>

<tr rowspan="3"><td><label>Address :<br></label></td><td><textarea rows="3" cols="30" spellcheck="true" ></textarea></td></tr><br><br>

</table>

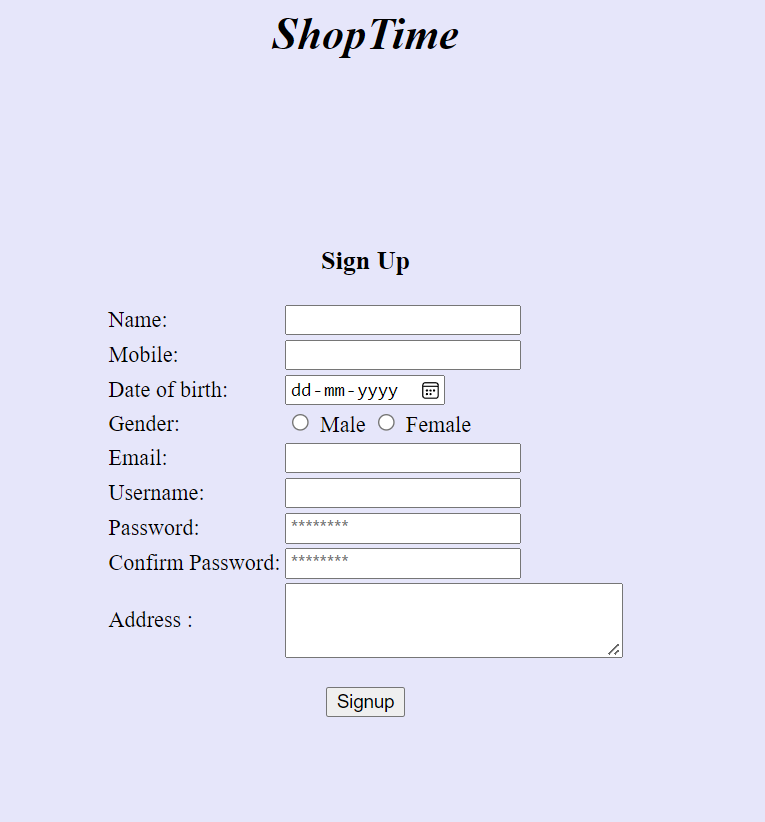
<br><button type="submit">Signup</button>

</form>

</body>

</html>

**Output:**

****

**2d) Course Name: HTML5 The Language**

**Module Name: Media, Iframe Add media content in a frame using audio, video, iframe elements to the Home page of IEKart's Shopping application**.

**Program:**

<!DOCTYPE html>

<html>

<body bgcolor="lavender">

<h1 align="center"><i>ShopTime<i></h1>

<h2 align="center"><i>One stop for all your needs<i></h2>

<header>

<nav align="center"><h3>

Home || Login || Register || Wishlist || My Orders || Help</h3>

</nav>

<center>

</header>

<p>

<iframe src="homeimg1.png" name="iframe\_1 height="300" width="600" title="Iframe Example"></iframe>

<iframe src="video1.Mp4" name="iframe\_2" height="300" width="600" title="Iframe Example"></iframe>

<iframe src="audio.Mp3" name="iframe\_3" height="100" width="1200" title="Iframe Example"></iframe></p>

</body>

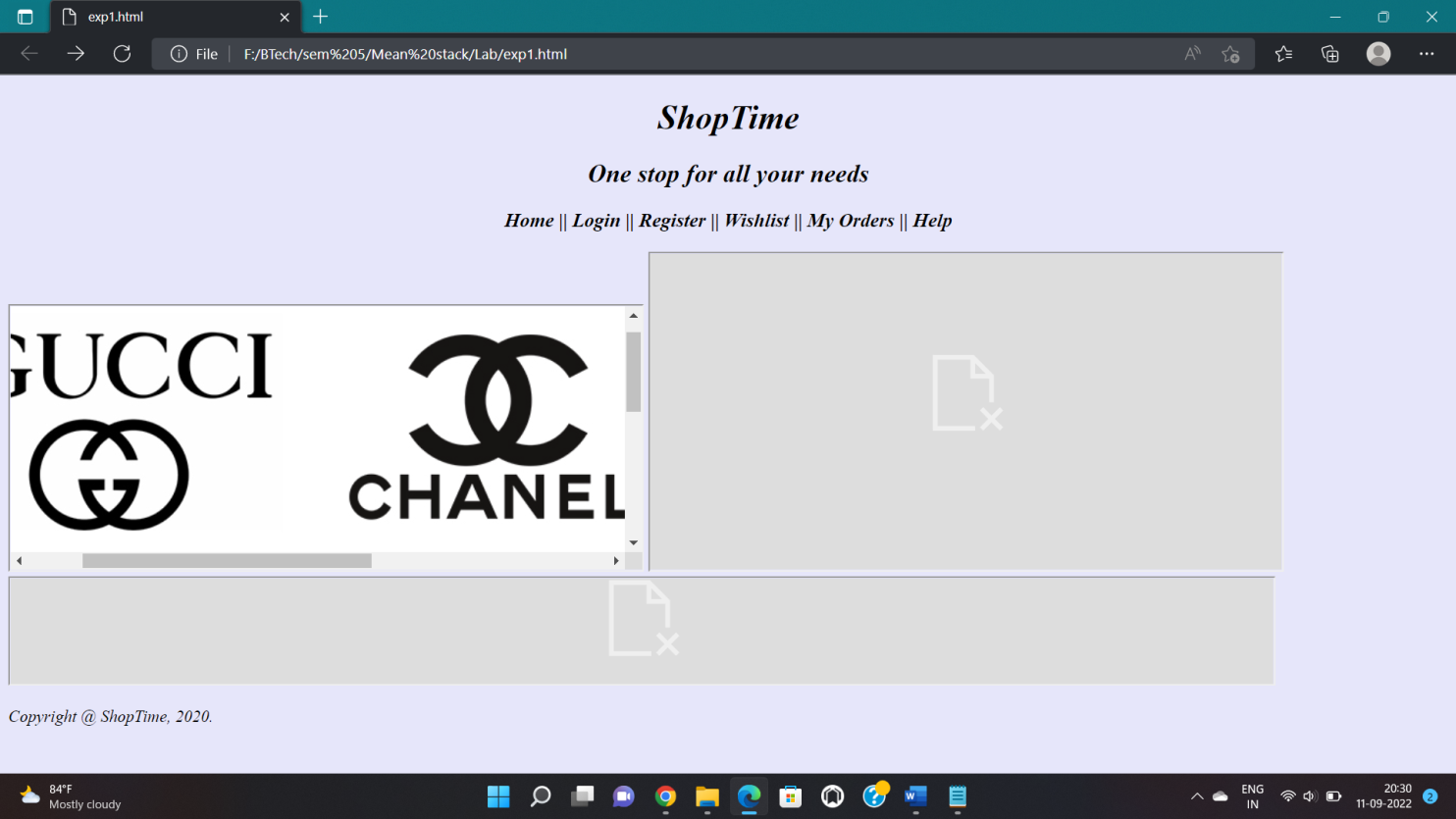
<footer>

Copyright @ ShopTime, 2020.

</footer>

</html>

**Output:**

****

**3.a Course Name: Javascript**

**Module Name: Type of Identifiers Write a JavaScript program to find the area of a circle using radius (var and let - reassign and observe the difference with var and let) and PI (const).**

**Program:**

<html>

<body>

<h2>Area of a circle with var and let</h2>

<script>

var rad=10;

const pi=3.14;

area=pi\*rad\*rad;

document.write("Area of circle using var="+area+"<br>");

var rad=14;

area=pi\*rad\*rad;

document.write("Area of circle after re-assigning var=="+area+"<br>");

let radius=10;

area=pi\*radius\*radius;

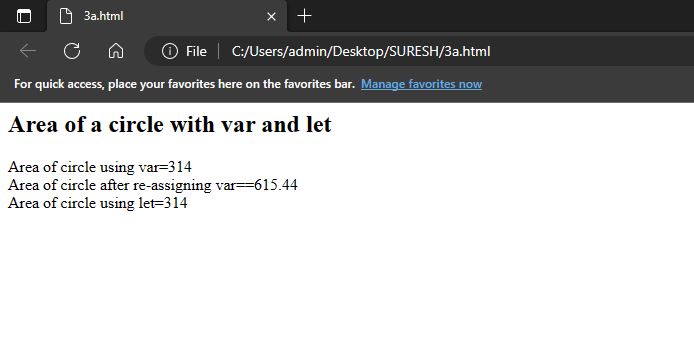
document.write("Area of circle using let="+area+"<br>");

</script>

</body>

</html>

**Output:**

****

**3b) Course Name: Javascript**

**Module Name: Primitive and Non Primitive Data Types 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.**

**Program:**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Template Literals</h2>

<p id="demo"></p>

<p id="demo1"></p>

<script>

let MovieName = "Transformers";

let Starring = "Meghan Fox";

let Language = "English";

var Rating = "7.9";

let Movie = "San Andreas";

let Cast = "Dwayne Johnson";

let Lang = "English";

var Ratings = "8.5";

let text = `The movie "${MovieName}" starring ${Starring} originally in ${Language} has rating of ${Rating}`;

document.getElementById("demo").innerHTML = text;

let text1= `The movie "${Movie}" starring ${Cast} originally in ${Lang} has rating of ${Ratings}`;

document.getElementById("demo1").innerHTML = text1;

</script>

</body>

</html>

**Output:** ****

**3c) Course Name: Javascript**

**Module Name: Operators and Types of Operators**

**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**.

**Program:**

<!DOCTYPE html>

<html>

<head>

</head>

<body style = "text-align: center; font-size: 20px;">

<h1> Online seats reservation </h1>

Enter the number of seats: <input id = "number">

<br><br>

<button onclick = "m()">Pay only</button>

<p id = "res"></p>

<script>

function ticket(num)

{

actual=num\*150;

discount=(actual/10); afterdisc=actual-discount; return afterdisc

}

function m()

{

var num = document.getElementById("number").value;

var f = ticket(num);

document.getElementById("res").innerHTML="The total price is " + num + " is: " + f ;

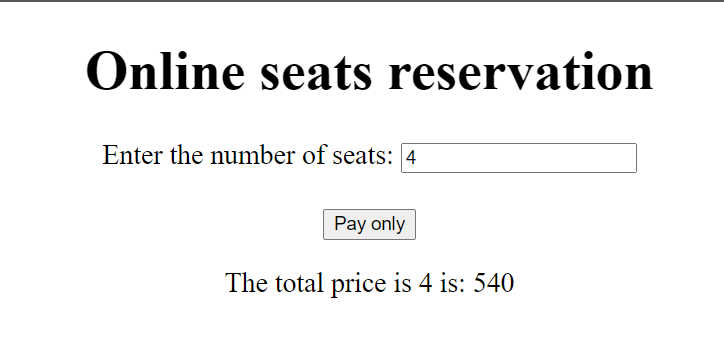
}

</script>

</body>

</html>

**Output:**

****

**3d) Course Name: Javascript**

**Module Name: Types of Statements, Non - Conditional Statements, Types of Conditional Statements, if Statements, switch Statements Write a JavaScript code to book movie tickets online and calculate the total price based on the 3conditions:**

**(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.**

**(c) If seats to be booked are more than 2 but less than 6, based on the number of seats booked, do the following - Calculate total cost by applying discounts of 3, 5, 7, 9, 11 percent, and so on for customer 1,2,3,4 and 5. Try the code with different values for the number of seats.**

**PROGRAM:**

<html>

<head>

<title>Movies</title>

</head>

<body bgcolor="lavender">

<h1 align="center"><i>ShopTime</i></h1>

<h2>Online bookings</h2>

<script bgcolor="lavender">

n=window.prompt("Enter a number:");

if(n<=2)

{

tcost=n\*150

document.write("For n tickets,you need to pay :",tcost);

}

else if(n>=6)

{

document.write("Bookings are not Allowed");

}

else

{

if (n==3)

{

t1=150-(150\*(3/100));

t2=150-(150\*(5/100));

t3=150-(150\*(7/100));

tcost=t1+t2+t3;

document.write("For 3 tickets,you need to pay :",tcost,"instead of ",(150\*3),"with discounts");

}

else if (n==4)

{

t1=150-(150\*(3/100));

t2=150-(150\*(5/100));

t3=150-(150\*(7/100));

t4=150-(150\*(9/100));

tcost=t1+t2+t3+t4;

document.write("For 4 tickets,you need to pay :",tcost,"instead of ",(150\*4),"with discounts");

}

else

{

t1=150-(150\*(3/100));

t2=150-(150\*(5/100));

t3=150-(150\*(7/100));

t4=150-(150\*(9/100));

t5=150-(150\*(11/100));

tcost=t1+t2+t3+t4+t5;

document.write("For 5 tickets,you need to pay :",tcost,"instead of ",(150\*5),"with discounts");

}

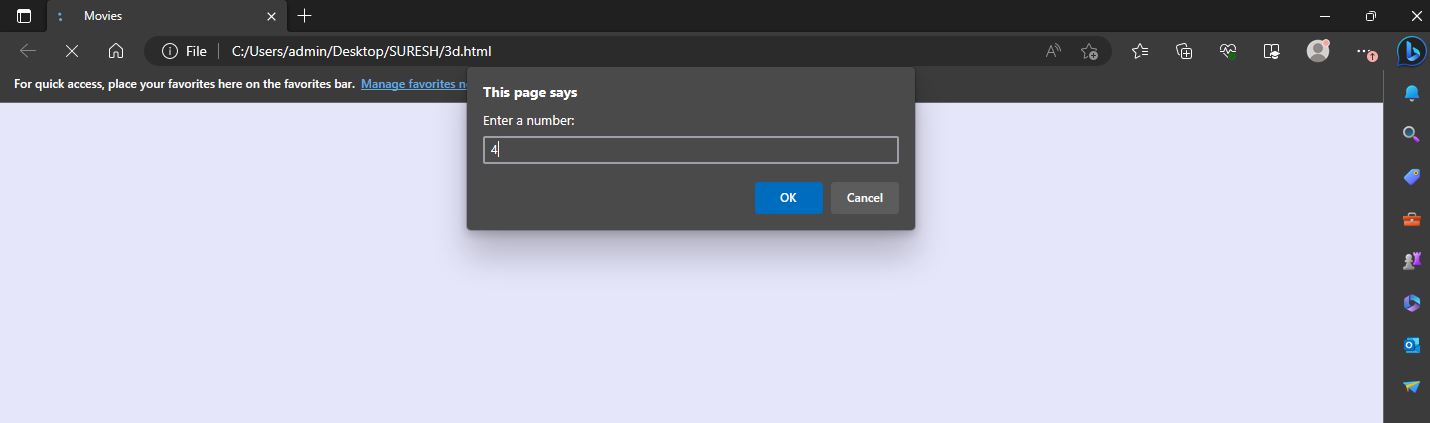
}

</script>

</body>

</html>

**Output:**

****

****

**3e. Course Name: Javascript**

**Module Name: Types of Loops**

**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.**

**(c) If seats to be booked are more than 2 but less than 6, based on the number of seats booked, do the following - Calculate total cost by applying a discount of 3, 5, 7, 9, 11 percent, and so on for customers till 5 respectively. Try the code with different values for the number of seats. Implement the problem statement using 'for' loop, 'while' loop and 'do-while' loop.**

**Program:**

<html>

<head>

<title>Movies</title>

</head>

<body bgcolor="cyan">

<h1 align="center"><i>ShopTime</i></h1>

<h2>Online bookings</h2>

<script bgcolor="cyan">

n=window.prompt("Enter a number:");

while(n<=2)

{

tcost=n\*150

document.write("For n tickets,you need to pay :",tcost);

}

if(n>=6)

{

document.write("Bookings are not Allowed");

}

else

{

if (n==3)

{

t1=150-(150\*(3/100));

t2=150-(150\*(5/100));

t3=150-(150\*(7/100));

tcost=t1+t2+t3;

document.write("For 3 tickets,you need to pay :",tcost,"instead of ",(150\*3),"with discounts");

}

else if (n==4)

{

t1=150-(150\*(3/100));

t2=150-(150\*(5/100));

t3=150-(150\*(7/100));

t4=150-(150\*(9/100));

tcost=t1+t2+t3+t4;

document.write("For 4 tickets,you need to pay :",tcost,"instead of ",(150\*4),"with discounts");

}

else

{

t1=150-(150\*(3/100));

t2=150-(150\*(5/100));

t3=150-(150\*(7/100));

t4=150-(150\*(9/100));

t5=150-(150\*(11/100));

tcost=t1+t2+t3+t4+t5;

document.write("For 5 tickets,you need to pay :",tcost,"instead of ",(150\*5),"with discounts");

}

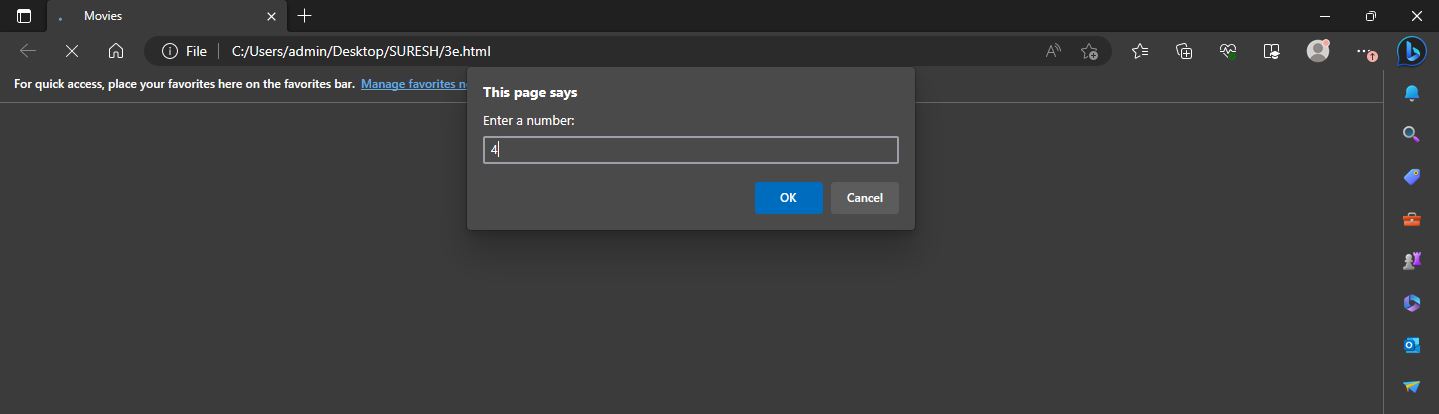
}

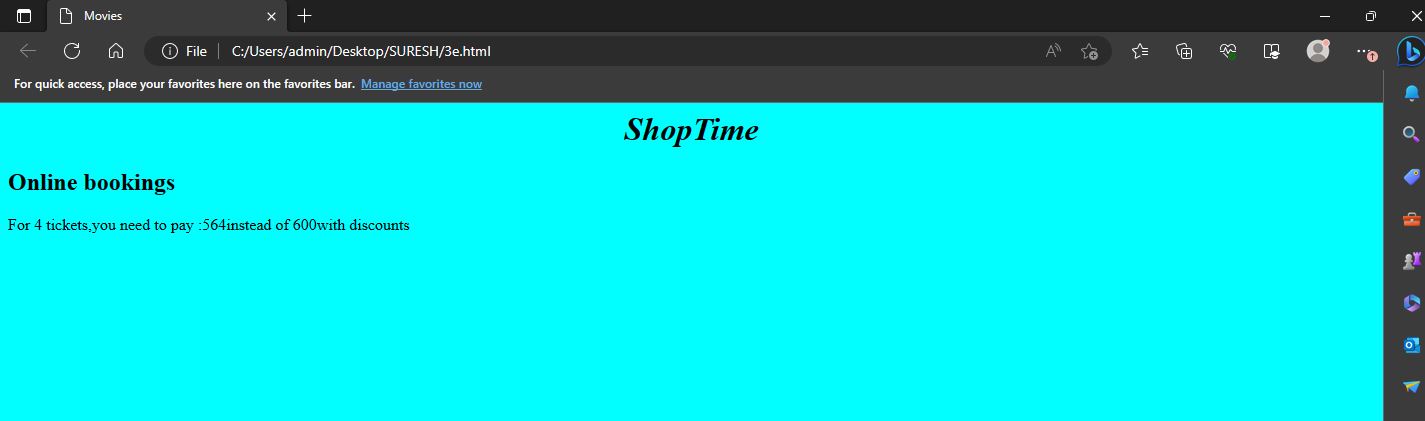
</script>

</body>

</html>

**Output:**

****



**4.a Course Name: Javascript**

**Module Name: Types of Functions, Declaring and Invoking Function, Arrow Function, Function Parameters, Nested Function, Built-in Functions, Variable Scope in Functions 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.**

**(c) If seats to be booked are more than 2 but less than 6, based on the number of seats booked, do the following - Calculate total cost by applying a discount of 3, 5, 7, 9, 11 percent, and so on for customers till 5 respectively. Try the code with different values for the number of seats.**

**Write the following custom functions to implement given requirements:**

1. **calculateCost(seats): Calculate and display the total cost to be paid by the customer for the tickets they have bought.**
2. **calculateDiscount(seats): Calculate discount on the tickets bought by the customer. Implement using arrow functions.**

**Program:**

<html>

<head>

<title>TicketsBooking</title>

<script> var x;

var y;

var z; fun=()=>

{

var a=prompt("Enter the number of tickets:"); if(a<6)

{

document.getElementById("id").innerHTML="Total amount you need to pay:";

document.getElementById("id1").innerHTML="Rs."+calculateCost(a);

document.getElementById("id2").innerHTML="Discount Amount is: Rs.”+calculateDiscount(a);

}

else

{

document.getElementById("id").innerHTML="Sorry! You can book upto 5 tickets only in online!!";

document.getElementById("id1").innerHTML=""; document.getElementById("id2").innerHTML="";

}

}

const p=150; calculateCost=(a)=>{

var i=1;

s=0;

j=0;

k=0.03

if(a>2 && a<6)

{

do

{

j=p-(p\*k); s+=j; j=0; k+=0.02; i+=1

}

while(i<=a);

}

else if(a<=2)

{

s=p\*a; }

else

s=0;

return s;

}

calculateDiscount=(a)=>

{

var g=calculateCost(a); var z=a\*p;

return z-g;

}

</script>

</head>

<body bgcolor="cyan">

<center><h1><i>ShopTime</i></h1></center>

<h2 align="center"><i>One stop for all your needs<i></h2>

<header>

<nav align="center"><h3>

Home || Login || Register || Wishlist || My

Orders || Help</h3>

</nav>

<center>

</header>

<h2>Book your tickets now</h2>

<br>

<input type="button" value="BOOK TICKETS" onclick="fun()">

<p id="id"></p>

<p id="id1"></p>

<p id="id2"></p>

</body>

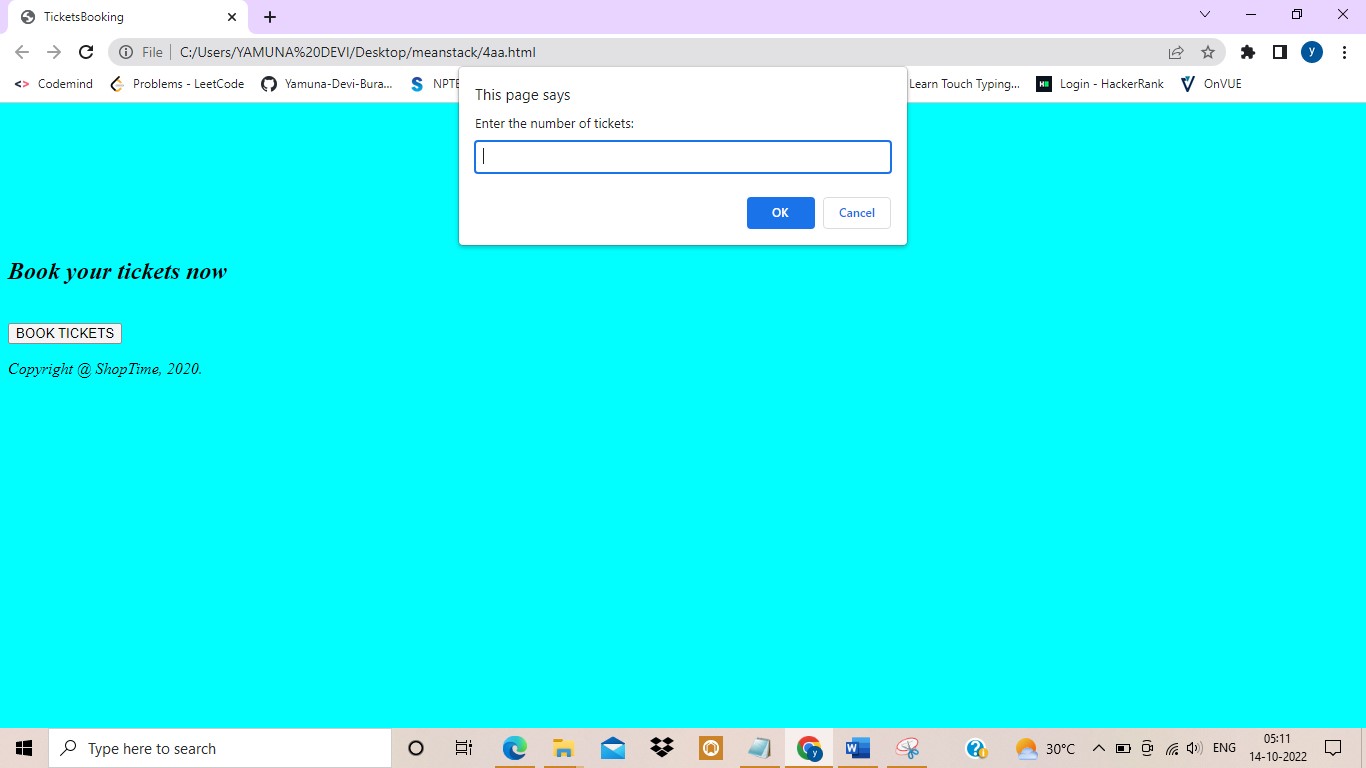
<footer>

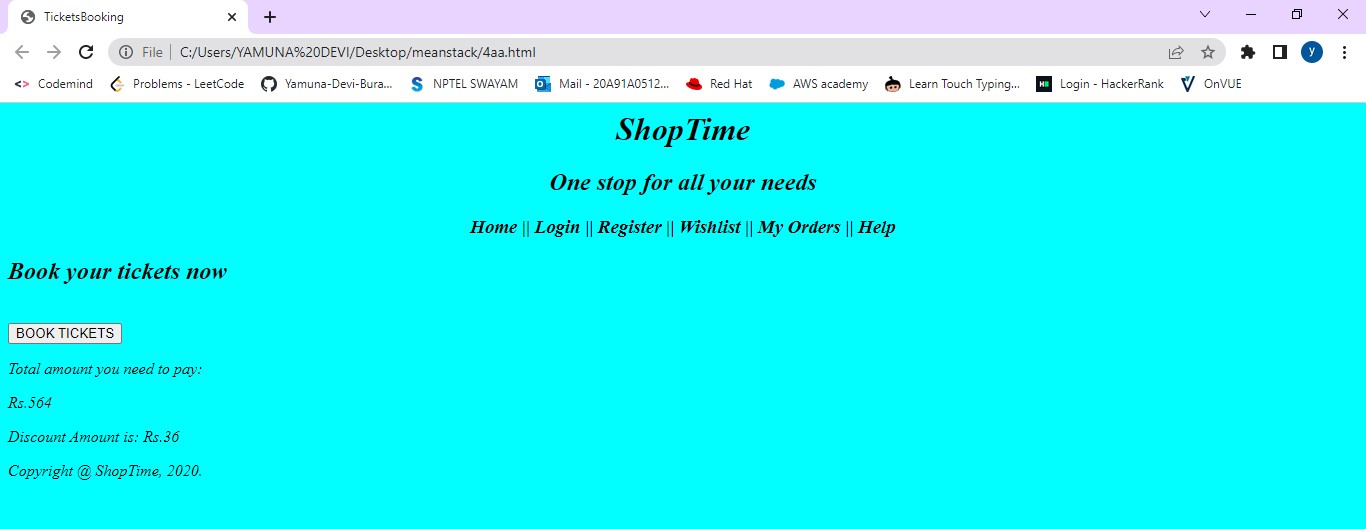
Copyright @ ShopTime, 2020.

</footer>

</html>

**Output:**

****



**4.b Course Name: Javascript Module Name: Working With Classes, Creating and Inheriting Classes 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 and contact**

**(iv) The constructor of the Employee to accept the name, age, role and contact where name and age are initialized through a call to super to invoke the base class constructor**

**(v)Add a method getDetails() to display all the details of Employee**

**Program:**

<html>

<head>

<title>Classes and Inheritances</title>

<script> class Person

{

constructor(name,age)

{

this.name=name; this.age=age;

}

det()

{

return "Name: "+this.name+"<br>"+"<br>"+"Age:

"+this.age;

} }

class Employee extends Person

{

constructor(name,age,role,contact)

{

super(name,age); this.roll=role; this.contact=contact;

}

getDetails()

{

return this.det()+"<br>"+"<br>"+"Role:

"+this.roll+"<br>"+ "<br>"+"Contact: "+this.contact;

}

}

function fun()

{

let v=new Employee("John Doe",24,"Cloud Architect","9876543210");

document.getElementById("id1").innerHTML=v.getDetails();

}

</script>

</head>

<body bgcolor="cyan">

<h1 style="background-color:white"><center></center></h1>

<h3>Click here to get details </h3>

<p id="id1">

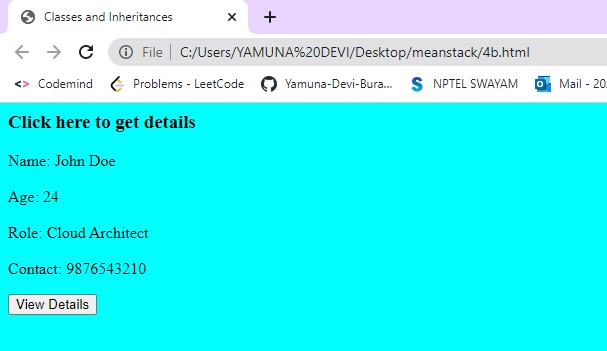
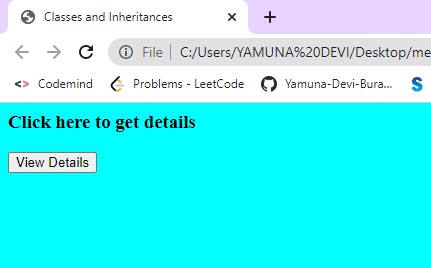
</p></center>

<input type="button"value="View Details" onclick="fun()">

</body>

</html>

**Output:**



**4.c Course Name: Javascript Module Name: In-built Events and Handlers 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.**

**(c) If seats to be booked are more than 2 but less than 6, based on the number of seats booked, do the following - Calculate total cost by applying discounts of 3, 5, 7, 9, 11 percent, and so on for customer 1,2,3,4 and 5. Try the code with different values for the number of seats. Write the following custom functions to implement given requirements:**

**(i) calculate Cost(seats): Calculate and display the total cost to be paid by the customer for the tickets he has bought.**

**(ii) calculate Discount(seats): Calculate discount on the tickets bought by the customer. Invoke this function only when the user clicks on a link/button.**

**Program:**

<html>

<head>

<title>TicketsBooking</title>

<script> var x;

var y;

var z; fun=()=>

{

var a=prompt("Enter the number of tickets:"); if(a<6)

{

document.getElementById("id").innerHTML="Total amount you need to pay:";

document.getElementById("id1").innerHTML="Rs."+calculateCost(a);

document.getElementById("id2").innerHTML="Discount Amount is: Rs."+calculateDiscount(a);

} else {

document.getElementById("id").innerHTML="Sorry! You can book upto 5 tickets only in online!!";

document.getElementById("id1").innerHTML=""; document.getElementById("id2").innerHTML="";

}

}

const p=150;

calculateCost=(a)=>{

var i=1;

s=0;

j=0;

k=0.03;

if(a>2 && a<6)

{

do

{

j=p-(p\*k); s+=j; j=0; k+=0.02; i+=1;

}

while(i<=a);

}

else if(a<=2)

{

s=p\*a;

}

else

s=0;

return s;

}

calculateDiscount=(a)=>

{

var g=calculateCost(a); var z=a\*p;

return z-g;

}

</script>

</head>

<body bgcolor="cyan">

<center><h1><i>ShopTime</i></h1></center>

<h2 align="center"><i>One stop for all your needs<i></h2>

<header>

<nav align="center"><h3>

Home || Login || Register || Wishlist || My

Orders || Movies || Help</h3>

</nav>

<center>

</header>

<center><img src="tickets.jpg" alt="Tickets"></img>

<h2>Book your tickets now</h2>

<br>

<input type="button" value="BOOK TICKETS" onclick="fun()"><p id="id"></p>

<p id="id1"></p>

<p id="id2"></p>

</body>

<footer>

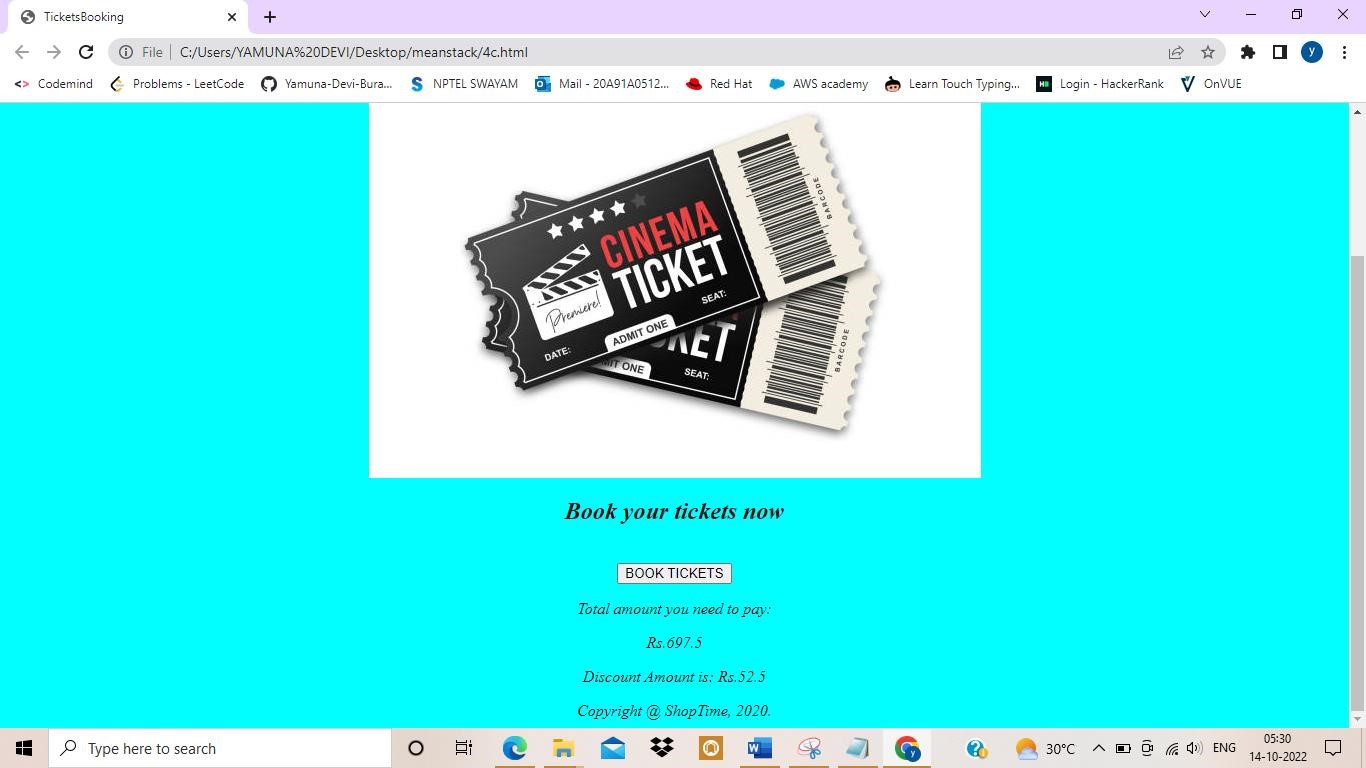
Copyright @ ShopTime, 2020.

</footer></center>

</html>

**Output:**

****

****

**4.d Course Name: Javascript**

**Module Name: Working with Objects, Types of Objects,Creating Objects, Combining and cloning Objects using Spread operator, Destructuring Objects, Browser Object Model, Document Object Model 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 refilled cone, a different heading, a different message, and a different color in the background.**

**Program:**

<!DOCTYPE html>

<html><script> var c=0; function fun()

{ if(c==0)

{

document.body.style.backgroundColor = "cyan"; document.getElementById("id1").innerHTML="Fill your cone"; document.getElementById("imag").src="cone.jpg"; document.getElementById("link").innerHTML="Fill";

c=1;

}

else

{

document.body.style.backgroundColor = "pink"; document.getElementById("id1").innerHTML="Eat your cone"; document.getElementById("imag").src="cone1.jpg"; document.getElementById("link").innerHTML="Eat";

c=0;

}

}

</script>

<center>

<h1 id="id1">Eat your cone</h1>

<br><br>

<img src="cone1.jpg" alt="Reload"height="300px" width="200px" id="imag">

<br>

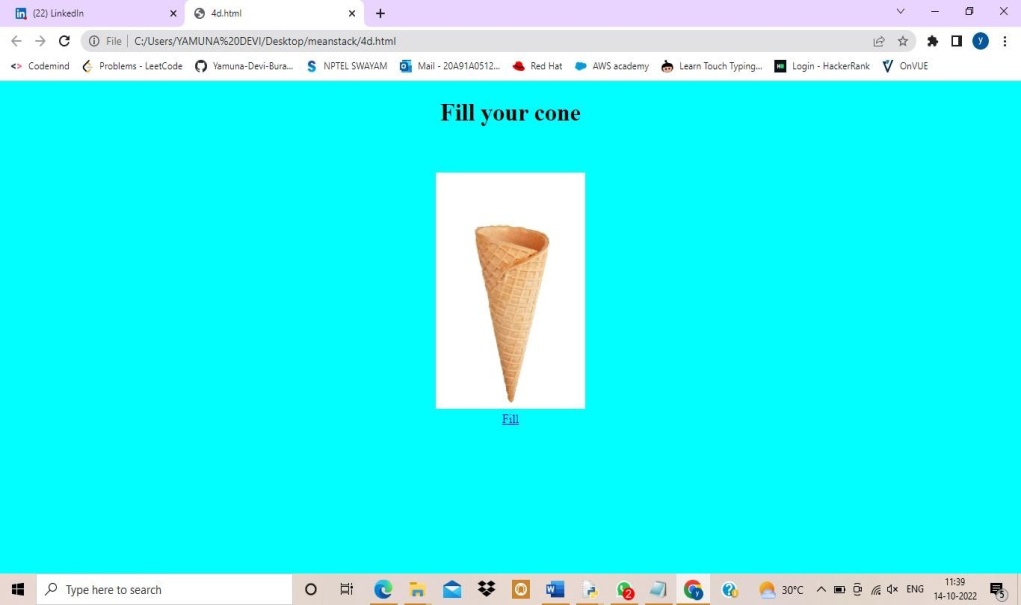
<a href="javascript:fun()" id="link">Eat</a>

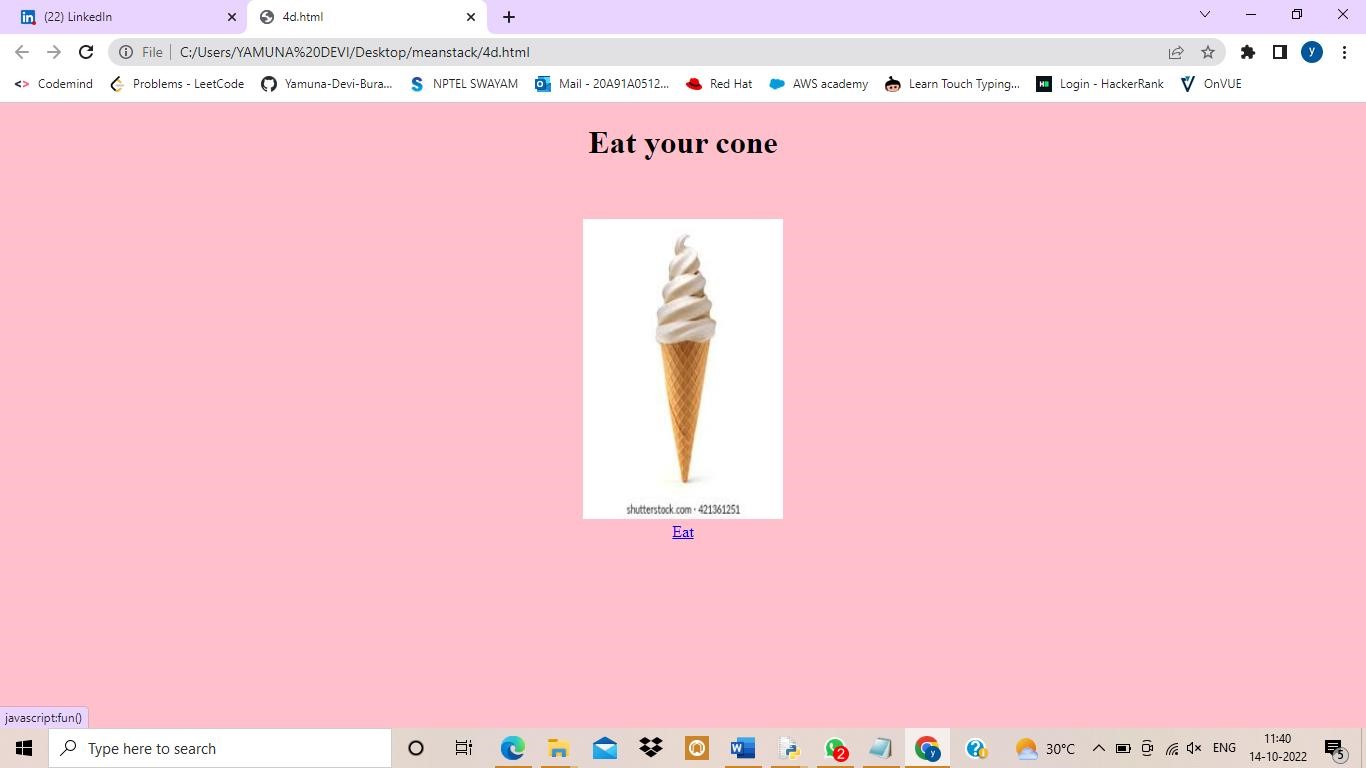
</center>

</body>

</html>

**Output:**

****

****

**5.a Course Name: Javascript**

**Module Name: Creating Arrays, Destructuring Arrays, Accessing Arrays, Array Methods**

**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.**

**Program:**

<!DOCTYPE html>

<html>

<body bgcolor="cyan">

<center><h1><i>ShopTime</i></h1>

<h2 align="center"><i>One stop for all your needs<i></h2>

<header>

<nav align="center"><h3>

Home || Login || Register || Wishlist || My

Orders || Movies || Help</h3>

</nav>

</header></center>

<I><h2>JavaScript Arrays</h2></I>

<img src="bahubali.jpg" width="300px" height="300px"></img>

<B><h1 id="demo1"></h1></B>

<p id="demo2"></p>

<p id="demo3"></p>

<p id="demo4"></p>

<script> const Movie = [ "BAHUBALI",

"Telugu",

"10",

"Prabhas",

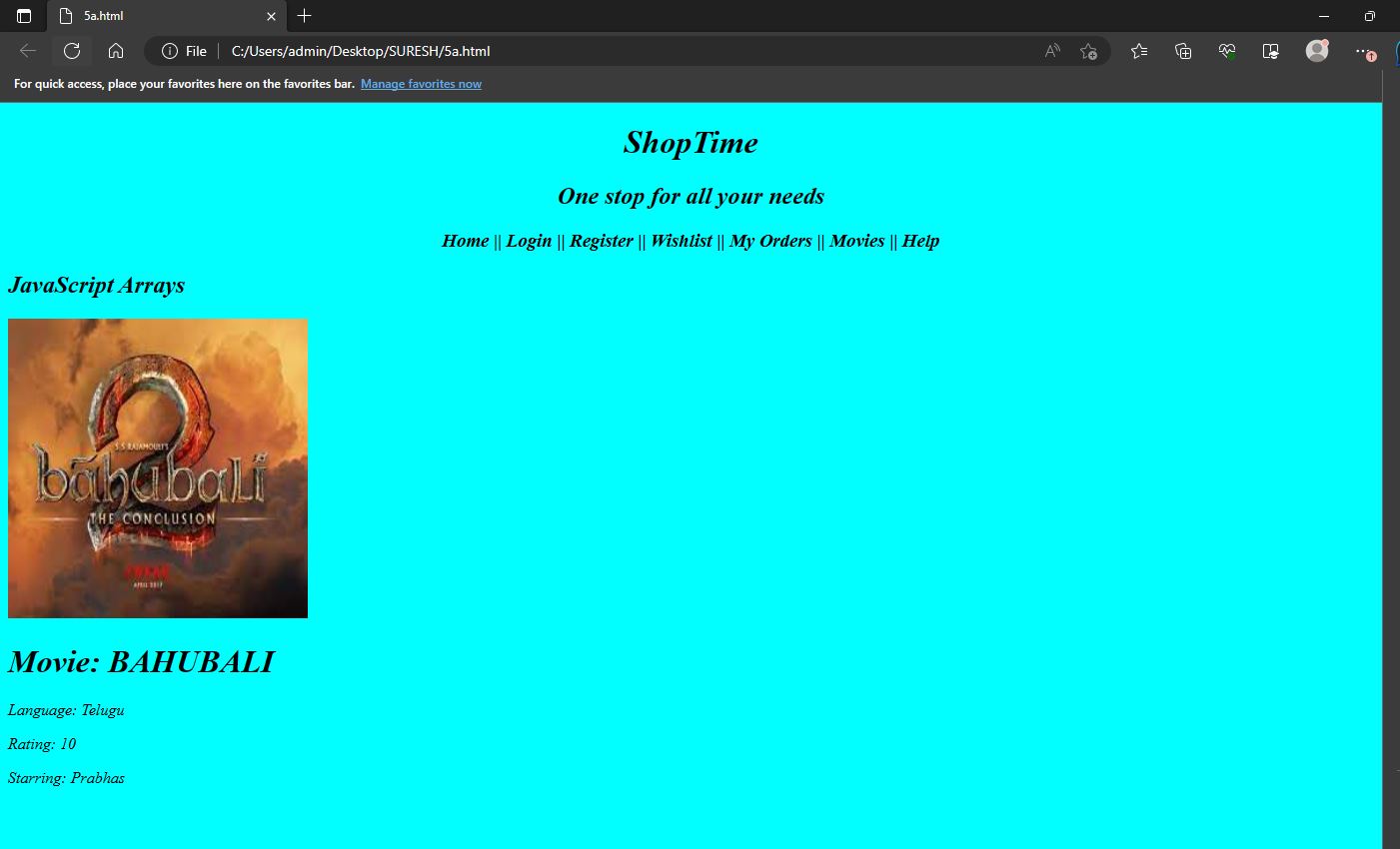
];

document.getElementById("demo1").innerHTML = "Movie: "+Movie[0]; document.getElementById("demo2").innerHTML = "Language: "+Movie[1]; document.getElementById("demo3").innerHTML = "Rating: "+Movie[2]; document.getElementById("demo4").innerHTML = "Starring: "+Movie[3];

</script>

</body>

</html>

**Output:** 

**5.b Course Name: Javascript**

**Module Name:Introduction to Asynchronous Programming, Callbacks, Promises, Async and Await, Executing Network Requests using Fetch API 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 the count is 5 – use the setInterval method.**

**(iii) Since setInterval is an async method, enclose the code in a Promise and handle the response generated in a success callback.**

**(iv) The random value returned from the method every time can be used as a stock price and displayed on the console.**

**Program:**

<!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, initialscale=1.0">

<title>Exp\_\_5b</title>

</head>

<body>

<script>

let c=0;

conststock=setInterval(stokc,3000);

function stokc(){

var myPromise = new Promise(function (resolve, reject)

{

setTimeout(function ()

{

var a=Math.floor(Math.random() \* 10);

resolve(a);

},

3000);

});

myPromise.then(

function (data)

{

console.log(data);

},

function (error) {

console.log(error);

}

);

c+=1; if(c==5)

{

Stop();

}

}

function Stop() {

clearInterval(stock);

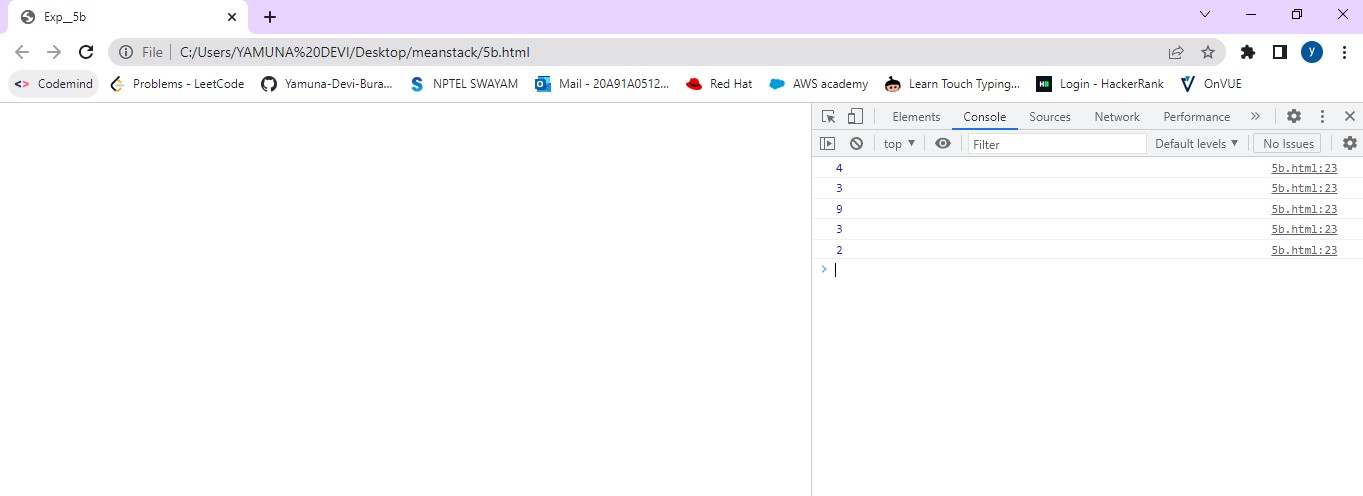
}

</script>

</body>

</html>

**Output:**

****

**5.c Course Name: Javascript**

**Module Name: Creating Modules, Consuming Modules 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" else will return "Unauthorized access".**

**(iv) Create an validateUser.html file with textboxes username and password and a submit button.**

**(v) Add a script tag in HTML to include validateUser.js file.**

**(vi) Create an validateUser.js file which imports login module and invokes validate method of User class.**

**(vii) On submit of the button in HTML the validate method of the User class should be invoked.**

**(viii) Implement the validate method to send the username and password details entered by the user and capture the return value to display in the alert.**

**Program:**

<!DOCTYPE html>

<html lang = “en”>

<head>

<meta charset=”UTF-8/>

<meta http-euiv=”X-UA-Compatible” content=”IE=edge”/>

<meta name = “viewname” content=”width = device-width, initial-scale=1.0”/>

<title>Document</title>

</head>

<body>

<input type = “text” name=”name” id=”name” placeholder=”Enter your user name here”/>

<input type = “password” name=”pass” id=”password” placeholder=”Enter your password”/>

<button type = “submit” id =”btn”> LOGIN</button>

<script src = “validateUser.js” type=”module”></script>

< script src = “login.js” type=”module”></script>

</body>

</html>

**ValidateUser.js**

import { User } from './login.js';

document.getElementById('btn').addEventListener('click',() =>{

let username = document.getElementById('name').value;

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

let user1 = new User(“abc”,’123’);

document.writeln(user1.validateUser(username,password));

});

**Login.js**

Class User{

Constructor(name,pass)

{

this.username = name;

this.password = pass;

}

validateUser(name,pass)

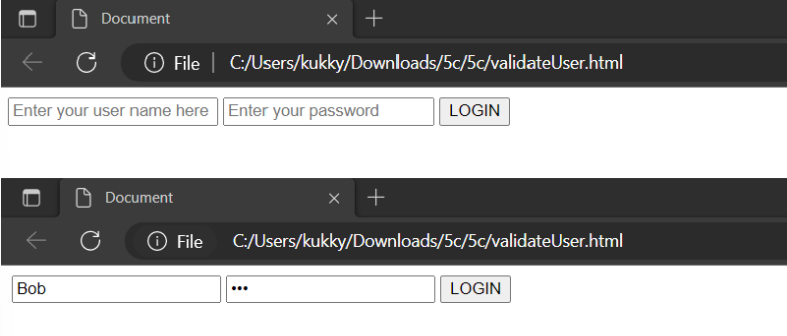
{

return name = this.name && pass==this.password) ? ”Login Successful” : “Unauthorized access”;

}

}

**Output:**

****

**6.a Course Name: Node.js**

**Module Name: How to use Node.js Verify how to execute different functions successfully in the Node.js platform.**

**Program:**

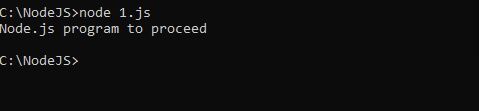
**Step 1**: Create a folder NodeJS in C drive and create a new JavaScript file, 1.js inside the folder. Type the below code inside the JavaScript file.

**On notepad:**

**console.log("Node.js program to proceed");**

**Step 2**: Navigate to the created NodeJS folder in the NodeJS command prompt and execute the JavaScript file, 1.js using the node command. node 1.js

**Step3**: After the successful interpretation of the code, we can see the output in the Node.js command prompt as shown below



**Program-2:**

function tester()

{

var m=10;

var message;

if (m%2==0)

{

message = "m is prime";

}

else

{

message = "m is not prime";

}

console.log(message);

}

tester(); **Output:**

****

**6.b Course Name: Node.js**

**Module Name: Create a web server in Node.js**

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

**Program:**

var http = require('http');

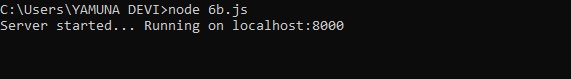
http.createServer(function (req, res) { res.writeHead(200, {'Content-Type': 'text/html'}); res.end(‘Server is connected**.**');

}).listen(8000);

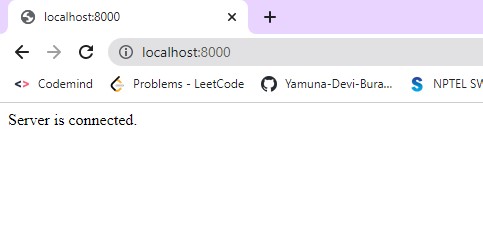
console.log("Server started... Running on localhost:8000");

**Output:**

Output on the command prompt –



Output on the browser:



**6.c Course Name: Node.js Module Name: Modular programming in Node.js Write a Node.js module to show the workflow of Modularization of Node application**

**Program:**

module.js

exports.authenticateUser = (a, b) => {

return a+b;

};

**Auth.js**

const http = require("http");

var dbmodule = require("./module");

var server = http.createServer((request, response) => {

result = dbmodule.authenticateUser(2000,2);

response.writeHead(200, { "Content-Type": "text/html" }); response.end("<html><body><h1>" + result + "- You have connected to the localhost 2002 </h1></body></html>");

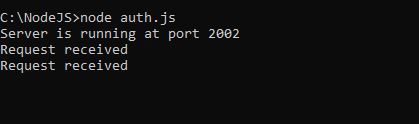
console.log("Request received");

});

server.listen(2002);

console.log("Server is running at port 2002")

# Output on the command prompt

****

**Output in the browser**

****

**6.d Course Name: Node.js Module Name: Restarting Node Application Write a program to show the workflow of restarting a Node application.**

**Program:**

const http = require("http");

var server = http.createServer((req, res) => { res.write("Hello ! I have created my second server!"); res.end();

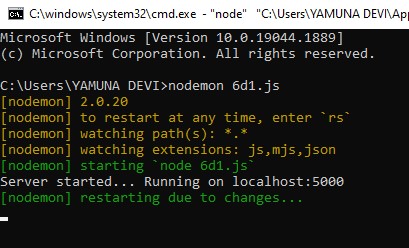
});

server.listen(5000);

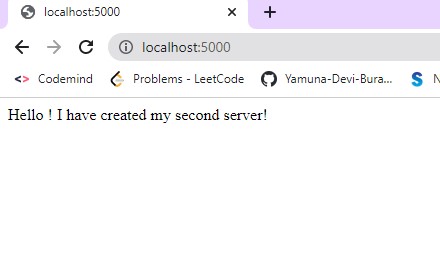
console.log("Server started... Running on localhost:5000");

**Output:**

**Output in command prompt:**

****

**Output in the browser:**

****

**6.e Course Name: Node.js**

**Module Name: File Operations Create a text file src.txt and add the following data to it. Mongo, Express, Angular, Node.**

**Program:**

const fs = require('fs');

const src = "source.txt";

const dest = "destination.txt"; fs.copyFile(src, dest, (error) => {

// incase of any error if (error)

{

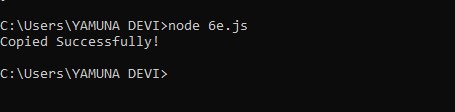
console.error(error); return;

}

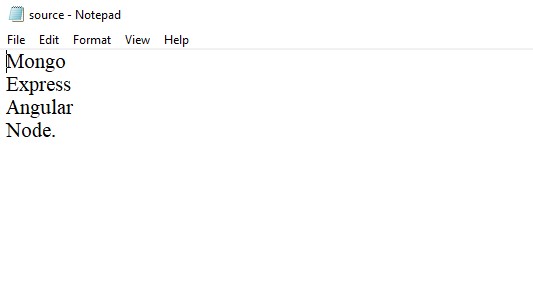
console.log("Copied Successfully!");

});

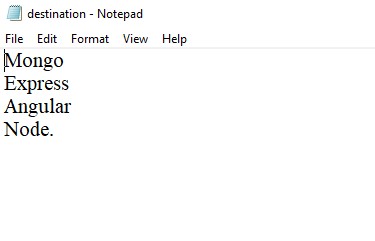
**Output in command prompt:**

****

**Source file:source.txt**

****

**Destination.txt**

****

**7.aCourse Name: Express.js**

**Module Name: Defining a route, Handling Routes, Route Parameters, QueryParameters Implement routing for the AdventureTrails application by embedding the necessary code in the routes/route.js file.**

**Program:**

**//myNotes.js File**

exports.packages = async (req, res) => {

try {

res.status(200).json({

message: 'You can now get the requested notes for your request ', });

} catch (err) { res.status(404).json({

status: 'fail', message: err,

});

}

};

exports.bookpackage = async (req, res) => {

try {

res.status(201).json({

data: 'New booking added for the POST request', });

} catch (err) { res.status(404).json({ status: 'fail',

message: err.errmsg,

});

}

};

exports.invalid = async (req, res) => {

res.status(404).json({ status: 'fail', message: 'Invalid path',

}); };

**Routing/route.js File**

const express = require('express');

const routing = express.Router();

const notesController = require('../Controller/myNotes'); routing.get('/packages', notesController.packages); routing.post('/bookpackage', notesController.bookpackage); routing.all('\*', notesController.invalid);

module.exports = routing;

**//App.js File** const express = require('express'); const route = require('./routes/route');

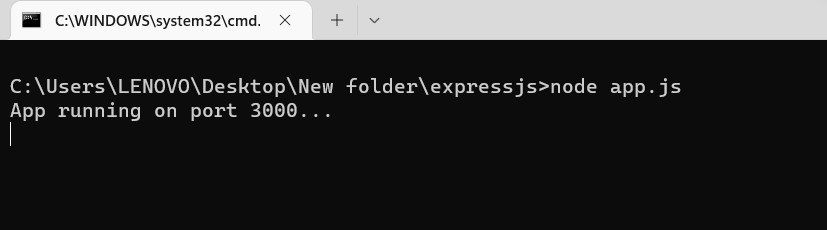
const app = express(); app.use('/', route);

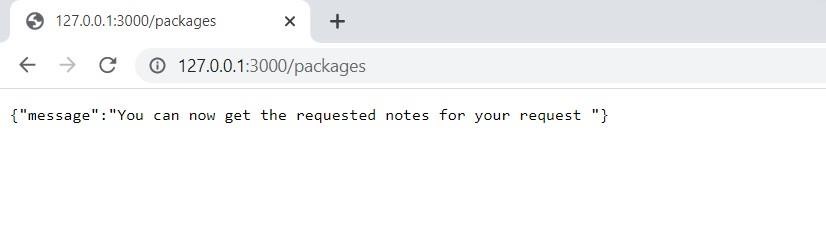
const port = process.env.PORT || 3000; app.listen(port, () => {

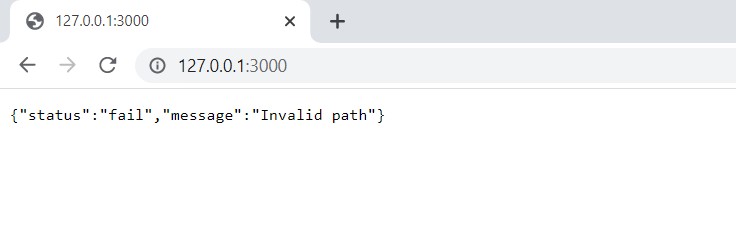
console.log(`App running on port ${port}...`);

});

**Output:**

****

****

****

**7.bCourse Name: Express.js**

**Module Name: How Middleware works, Chaining of Middlewares, Types of Middlewares In myNotes application: (i) we want to handle POST submissions. (ii) display customized error messages. (iii) perform logging.**

**Program:**

**//Route1.js file** const express = require('express'); const router = express.Router(); const myController = require('../Controller/myNotes1'); router.get('/', myController.myMethod); router.get('/about', myController.aboutMethod); module.exports = router; **//myNotes.js File** exports.myMethod = async (req, res, next) => { res.send('<h1>Welcome</h1>');

};

exports.aboutMethod = async (req, res, next) => {

res.send('<h1>About Us Page</h1>');

};

**//app1.js File** const express = require('express'); const router = require('./Routes/route1'); const app = express();

const mylogger = function (req, res, next) {

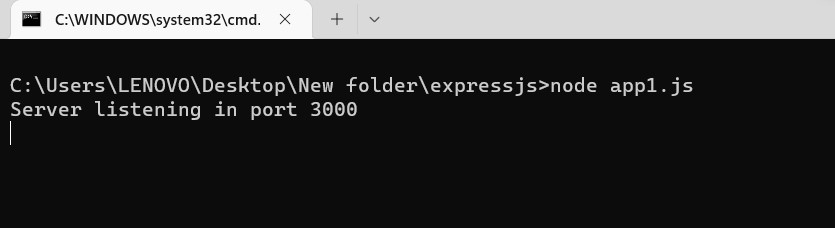
console.log(`Req method is ${req.method}`); console.log(`Req url is ${req.url}`); next();

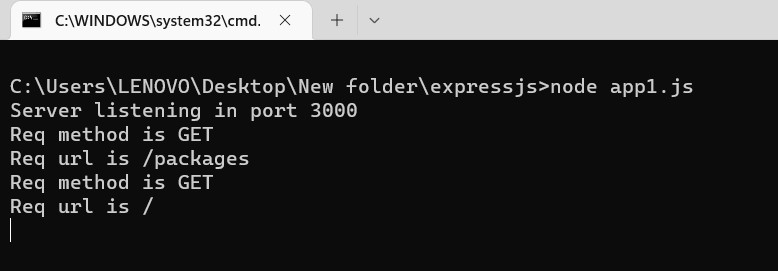
};

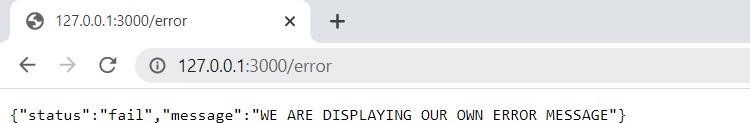
app.use(mylogger); app.use('/', router); app.listen(3000);

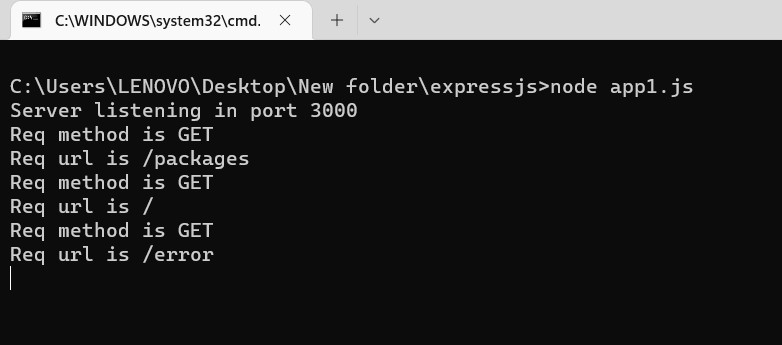
console.log('Server listening in port 3000');

**Output:**

****



****

****

**7.cCourse Name: Express.js**

**Module Name: Connecting to MongoDB with Mongoose, Validation Types and Defaults**

**Write a Mongoose schema to connect with MongoDB.**

**Program:**

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

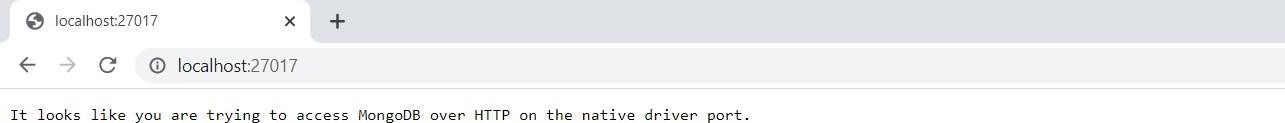
app.listen(3000,()=>console.log(" server running. .... ")) const

url="mongodb+srv://mstdatabase:mstdatabase@cluster0.xx7bb4u.mongodb.net/?retryWrites=true&w

=majority";

mongoose.connect(url).then(()=>console.log("Database Connected. ... ")).catch(err=>console.log(err));

**Output:**



**Creating schema**

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

app.listen(3000,()=>console.log(" Server running. .... ")) const

url="mongodb+srv://mstdatabase:mstdatabase@cluster0.xx7bb4u.mongodb.net/?retryWrites=true&w

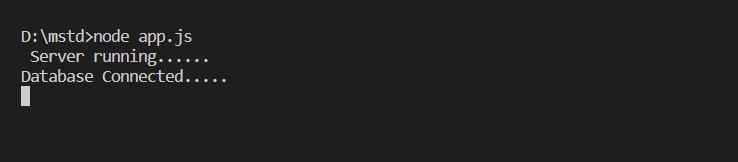
=majority";

mongoose.connect(url).then(()=>console.log("Database Connected. ... ")).catch(err=>console.log(err)); var bookSchema = mongoose.Schema({ name: String,

isbn: {type: String, index: true}, author: String, pages: Number

});

**Output:**

****

**7.dCourse Name:Express.js**

**Module Name: Models**

**Write a program to wrap the Schema into a Model object.**

**Program:**

const express=require('express')

constmongoose=require('mongoose')

const url="mongodb://0.0.0.0:27017/Hell";

mongoose.connect(url,{useNewUrlParser:true},{useUnifiedTopology:true}).then(()=>console.log("DatabaseConnected....")).catch(err=>

console.log(err));

var bookSchema = mongoose.Schema({

name: String,

isbn: {

type: String, index: true},

author: String,

pages: Number

});

var Book = mongoose.model("Book",bookSchema);

var db = mongoose.connection;

db.on("error", console.error.bind(console,"connection error:")); db.once("open", function(){ console.log("Connected to DB");

});

**Output:** ****

**8.a Course Name: Express.js**

**Module Name: CRUD Operations**

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

**Program:**

**Create:**

const express=require('express') const mongoose=require('mongoose') const url="mongodb://0.0.0.0:27017/Hell";

mongoose.connect(url,{useNewUrlParser:true},{useUnifiedTopology:true}).then(()=>console.log("D atabase Connected. ... ")).catch(err=>console.log(err)); var bookSchema = mongoose.Schema({ name: String,

isbn: {type: String, index: true}, author: String, pages: Number

});

var Book = mongoose.model("Book", bookSchema);

var db = mongoose.connection; var book1 = new Book({ name:"Mongoose Demo 1", isbn: "MNG123", author: "Author1, Author2", pages: 123

});

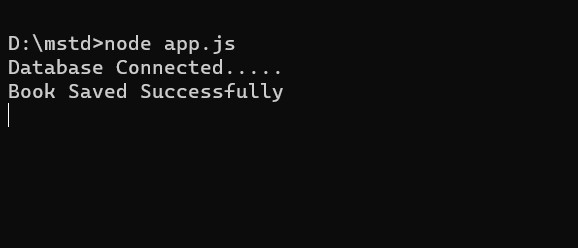
book1.save(function(err){

if ( err )

throw err;

console.log("Book Saved Successfully");

});

****

# Read

Const express=require('express')

Const mongoose=require('mongoos’)consturl="mongodb://0.0.0.0:27017/Hell";

mongoose.connect(url,{useNewUrlParser:true},{useUnifiedTopology:true}).then(()=>console.log("DatabaseConnected...")).catch(err=>

console.log(err));

var bookSchema = mongoose.Schema({

name: String,

isbn: {type: String,

index: true},

author: String,

pages: Number

});

var Book = mongoose.model("Book", bookSchema);

var db = mongoose.connection; var queryBooks = function(){

Book.find( function(err, result){

if ( err )

throw err;

console.log("Find Operations: " + result);

}); } queryBooks();

**Output:**

****

# Update

const express=require('express') const mongoose=require('mongoose') const url="mongodb://0.0.0.0:27017/Hell";

mongoose.connect(url,{useNewUrlParser:true},{useUnifiedTopology:true}).then(()=>console.log("D atabase Connected. ... ")).catch(err=>console.log(err)); var bookSchema = mongoose.Schema({ name: String,

isbn: {type: String, index: true}, author: String, pages: Number

});

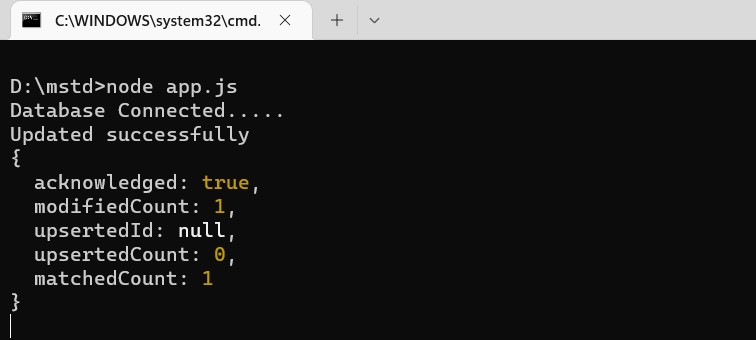
var Book = mongoose.model("Book", bookSchema); var db = mongoose.connection; var updateBook = function(){

Book.updateOne({$name: "JAva"}, {$set: {name: "JAVA"}}, function(err, result){ console.log("Updated successfully"); console.log(result);

});

}updateBook();

**Output:**

****

**Delete**

const express=require('express') const mongoose=require('mongoose') const url="mongodb://0.0.0.0:27017/Hell";

mongoose.connect(url,{useNewUrlParser:true},{useUnifiedTopology:true}).then(()=>console.log("D atabase Connected. ... ")).catch(err=>console.log(err)); var bookSchema = mongoose.Schema({ name: String,

isbn: {type: String, index: true}, author: String, pages: Number});

var Book = mongoose.model("Book", bookSchema);

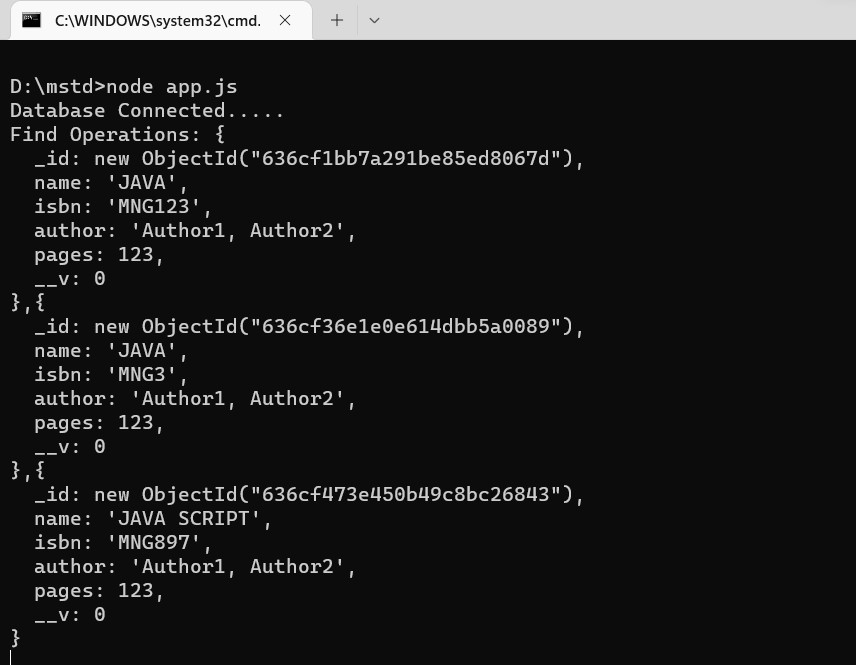
var db = mongoose.connection; var deleteBook = function(){

Book.deleteMany({name:"JAva"},function(err,result){if(err) console.log(err); else

console.log("deleted")}).exec(); }

deleteBook();

**After deleting records Database is:**

****

**8.c Course Name: Express.js**

**Module Name: Why Session management,Cookies**

**Write a program to explain session management using cookies.**

**Program:**

var express=require('express');

var cookieParser=require('cookie-parser');

var app = express(); app.use(cookieParser());

app.get('/cookieset',function(req, res)

{ res.cookie('cookie\_name', 'cookie\_value');

res.cookie('College', 'Aditya'); res.cookie('Branch', 'Cse');

res.status(200).send('Cookie is set');

});

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

res.status(200).send(req.cookies);

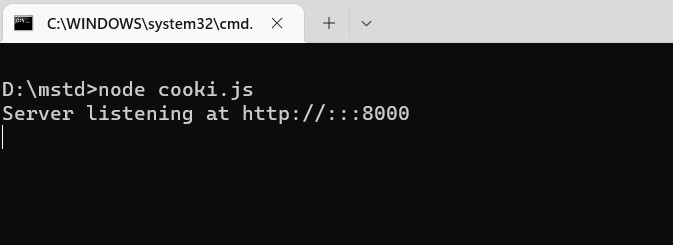
}); app.get('/', function (req, res) {

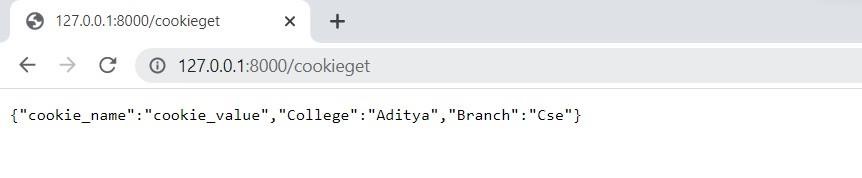
res.status(200).send('Welcome to Aditya');

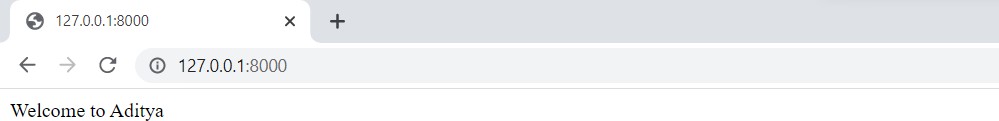
}); var server = app.listen(8000, function () {

var host = server.address().address; var port = server.address().port; console.log('Server listening at http://%s:%s', host, port);

});

**Output:** ****





**8.d Course Name: Express.js**

**Module Name: Sessions**

**Write a program to explain session management using sessions.**

**Program:**

const express = require("express")

const session = require('express-session')

const app = express()

var PORT = process.env.port || 3000

app.use(session({

secret: 'Your\_Secret\_Key',

resave: true,

saveUninitialized: true

}))

app.get("/", function(req, res)

{

req.session.name ='Sessionname:alr'

return

res.send("Session Set")

})

app.get("/session", function(req, res)

{ var name = req.session.name

return res.send(name)

})

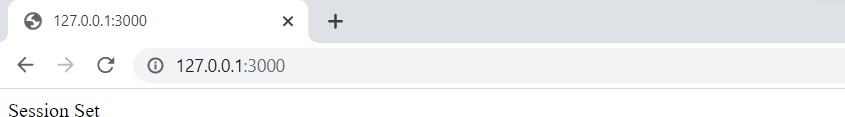
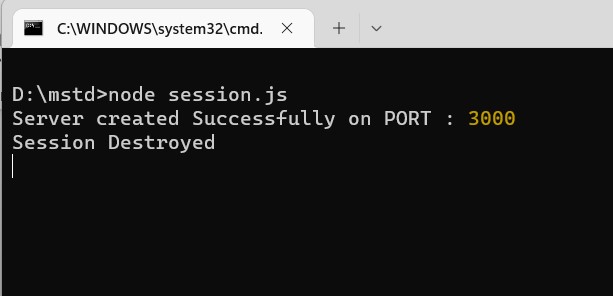
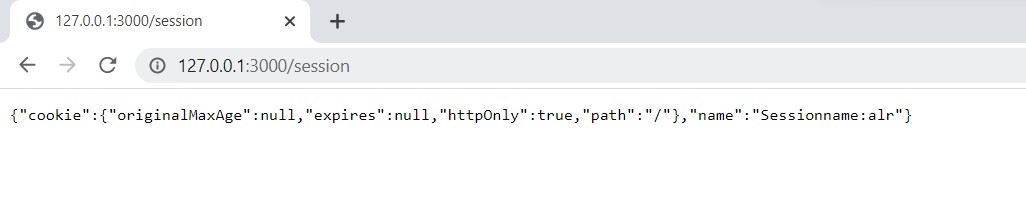
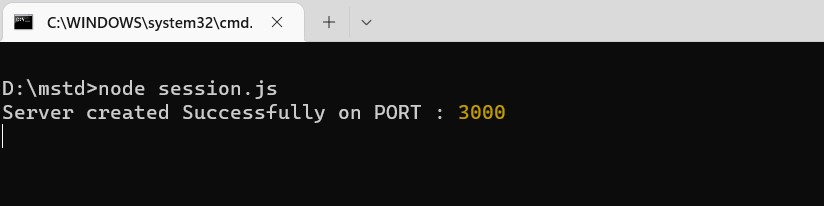
app.listen(PORT, function(error)

{ if(error) throw error

console.log("Server created Successfully on PORT :", PORT)

})

**Output:**



**8.e Course Name: Express.js**

**Module Name:Why and What Security, Helmet Middleware Implement security features in myNotes application.**

**Program:**

**App.js**

const express = require('express');

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

const app = express(); app.use('/', routing); app.listen(3000);

console.log('Server listening in port 3000');

**route.js**

const express = require('express');

const router = express.Router(); router.get('/', function (req, res) { res.send('<h1>Express</h1>');

});

router.get('/about', function (req, res) {

res.send('About Us Page');

});

module.exports = router;

**test.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<style> p { color: red;

} iframe {

width: 100%; height: 90%

}

</style>

</head>

<body>

<p>Clickjacked</p>

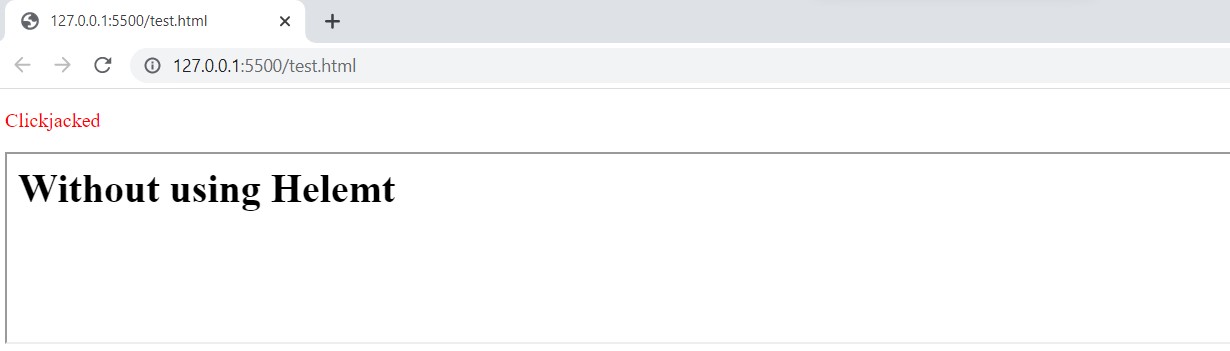
<iframe src="http://localhost:3000">

</iframe>

</body>

</html>

**Output:**

****

**Implementing HelmetApp.js**

const express = require('express'); const helmet = require('helmet'); const routing = require('./route'); const app = express(); app.use(helmet()); app.use('/', routing); app.listen(3000);

console.log('Server listening in port 3000');

**Output:**

****

**9.a Course Name: Typescript**

**Module Name: Basics of TypeScript**

**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**

**Program:**

const obj:{GoldPlatinum: string}={GoldPlatinum:"$1000"}

const ob1:{PinkGold: string,}={PinkGold:"$900"}

const ob2:{SilverTitanium: string}={SilverTitanium:"$1500"} console.log("\nMobilecolor Price\n")

console.log("\nGoldPlatinum:\t "+obj.GoldPlatinum+"\n") console.log("PinkGold:\t "+ob1.PinkGold+"\n")

console.log(" SilverTitanium:\t"+ob2.SilverTitanium+"\n")

**Output:**

****

**9.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.**

**Program:**

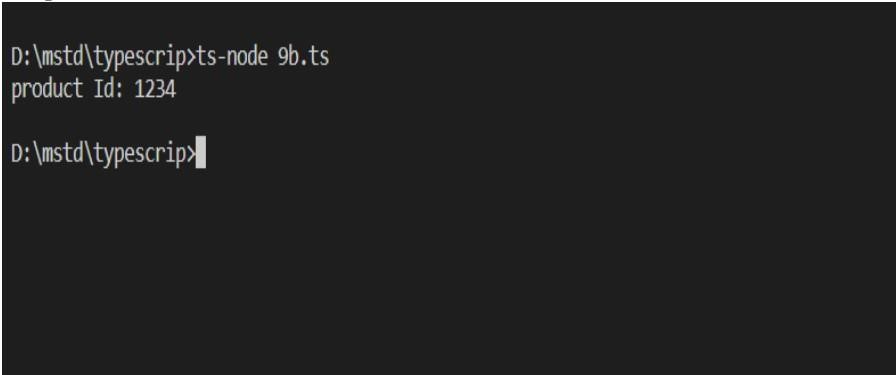
var getproductdetails=(productId : number):string=>{

return "product Id:"+productid

};

console.log(getproductdetails(1234));

**Output:**

****

**9.c Course Name: Typescript**

**Module Name: Parameter Types and Return Types**

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

**PROGRAM:**

function getMobileByManufacturer(manufacturer: string): string[]

{

let mobileList: string[];

if (manufacturer === 'Samsung')

{ mobileList = ['Samsung Galaxy S6 Edge', 'Samsung Galaxy Note 7',

'Samsung Galaxy J7 SM-J700F'];

return mobileList;

}

else if (manufacturer === 'Apple') {

mobileList = ['Apple iPhone 5s', 'Apple iPhone 6s ', 'Apple iPhone 7'];

return mobileList;

}

else {

mobileList = ['Nokia 105', 'Nokia 230 Dual Sim'];

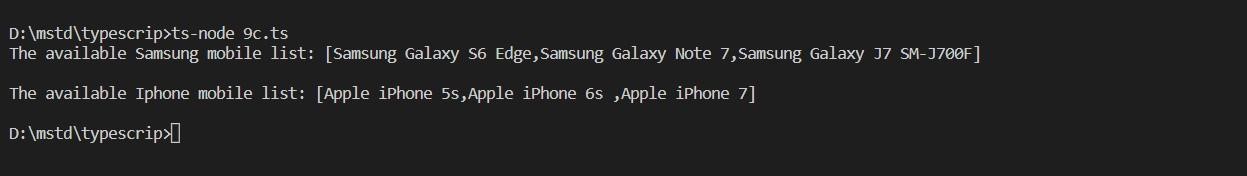
return mobileList;

} }

console.log('The available Samsung mobile list: [' + getMobileByManufacturer('Samsung')+']');

console.log('\nThe available Iphone mobile list: [' + getMobileByManufacturer('Apple')+"]");

**Output:**

****

**9d)Course Name: Typescript**

**Module Name: Arrow Function**

**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 to 150 dollars then below mentioned code snippet would fit into this requirement.**

**Program:**

var manufacturers = [{ id: 'Samsung', price: 150 },

{ id: 'Microsoft', price:200 },

{ id: 'Apple', price:00 },

{ id: 'Micromax', price: 100 } ];

var test;

console.log('Details of Manufacturer array are: ');

function myFunction() {

test = manufacturers.filter((m) =>

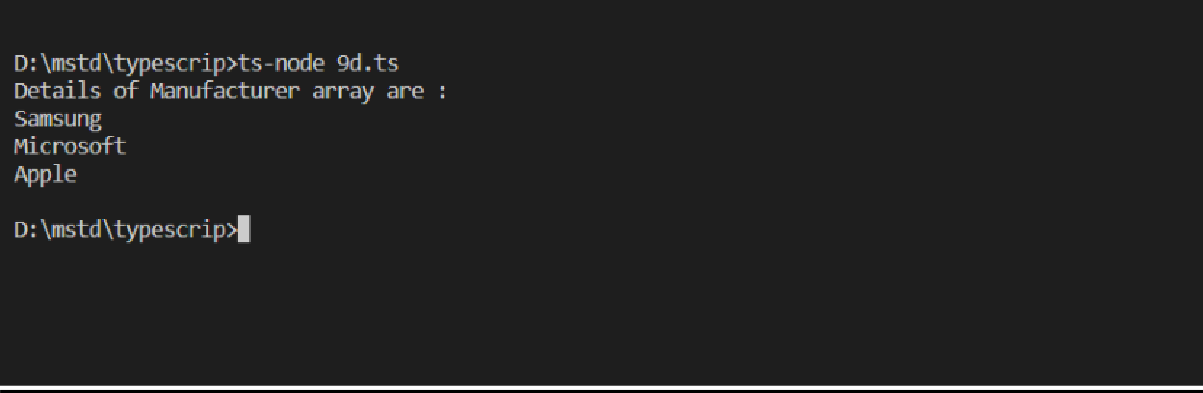
m.price >= 150);

for (var item of test) { console.log(item.id);

}

} myFunction();

**Output:**

****

**9e)Course Name: Typescript**

**Module Name: Optional and Default Parameters 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 should return Moto mobile list and if manufacturer parameter is either Samsung/Apple then this function should return respective mobile list and similar to make Samsung as default Manufacturer. Below mentioned code-snippet would fit into this requirement.**

**Program:**

function getMobileByManufacturer(manufacturer: string = 'Samsung', id?: number):

string[]{ let mobileList: string[];

if (id) { if (id === 101) {

mobileList = ['Moto G Play, 4th Gen', 'Moto Z Play with Style Mod'];

return mobileList;

}}

if (manufacturer === 'Samsung') {

mobileList = [' Samsung Galaxy S6 Edge', ' Samsung Galaxy Note 7',

' Samsung Galaxy J7 SM-J700F'];

return mobileList;

}

else if (manufacturer === 'Apple') {

mobileList = [' Apple iPhone 5s', ' Apple iPhone 6s', ' Apple iPhone 7'];

return mobileList;

}

else {

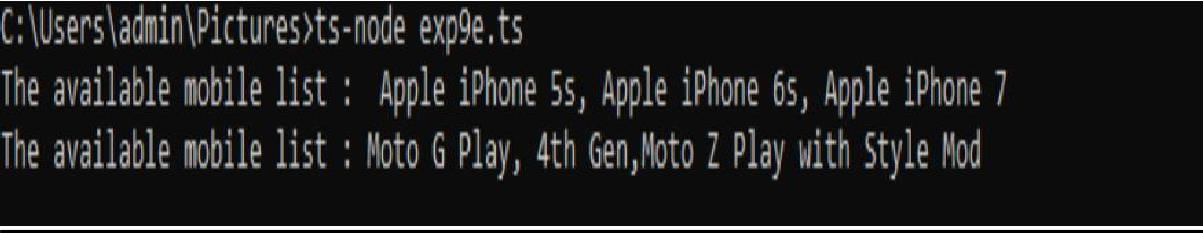
mobileList = [' Nokia 105', ' Nokia 230 Dual Sim'];

return mobileList;}}

console.log('The available mobile list : ' + getMobileByManufacturer('Apple'));

console.log('The available mobile list : ' + getMobileByManufacturer(undefined, 101))

**Output:**

****

**10.a)Module Name: Rest Parameter**

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

**Program:**

const cart: string[] = [];

const pushtoCart = (item: string) => { cart.push(item); }; function addtoCart(...productName: string[]): string[] {

for (const item of productName) { pushtoCart(item);

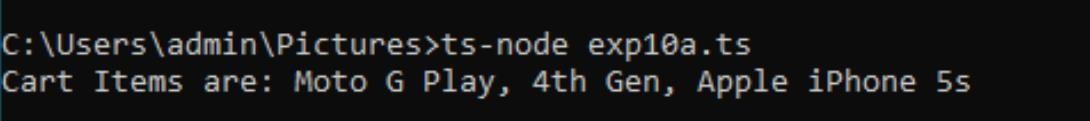
}

Returncart;

}

console.log('Cart Items are:' + addtoCart(' Moto G Play, 4th Gen', ' Apple iPhone 5s'));

**Output:**

****

**10.b)Module Name: Creating an Interface**

**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.**

**Program:**

interface Product { productId: number ; productName: string ;

}

function getProductDetails(productobj: Product): string { return 'The product name is : ' + productobj.productName;

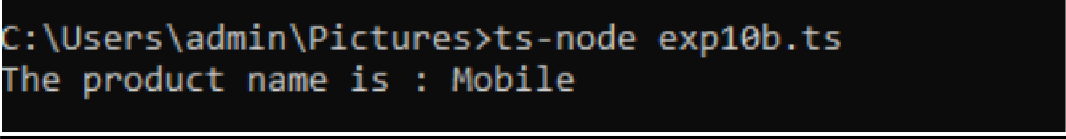
}

const prodObject = {productId: 1001, productName: 'Mobile'};

const productDetails: string = getProductDetails(prodObject);

console.log(productDetails);

**Output:**

****

**10.cModule Name: Duck Typing**

**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.**

**Program:**

interface Product { productId: number; productName:string;

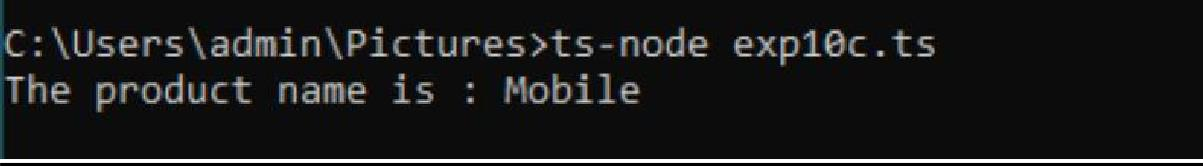
}

function getProductDetails(productobj: Product): string { return 'The product name is : ' + productobj.productName;

}

const prodObject = {productId: 1001, productName: 'Mobile', productCategory: 'Gadget'}; const productDetails: string = getProductDetails(prodObject); console.log(productDetails);

**Output:**

****

**10.d)Module Name: Function Types**

**Declare an interface with function type and access its value.**

**Program:**

function CreateCustomerID(name: string, id: number): string { return 'The customer id is ' + name + ' ' + id;

}

interface StringGenerator {

(chars: string, nums: number): string;

}

let idGenerator: StringGenerator; idGenerator = CreateCustomerID;

const customerId: string = idGenerator('Mr.Tom', 101); console.log(customerId);

**Output:**

****

**11.a Course Name: Typescript**

**Module Name: Extending Interfaces**

**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.**

**Program**:

interface Category

{

categoryName: string;

}

interface Product

{

productName: string;

productId: number;

}

interface ProductList extends Category, Product

{

list: Array;

}

const productDetails: ProductList = {

categoryName: 'Gadget', productName: 'Mobile',

productId: 1234, list: ['Samsung', 'Motorola', 'LG']

};

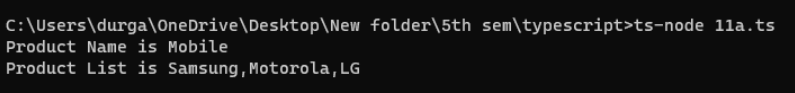
const listProduct = productDetails.list;

const pname: string = productDetails.productName;

console.log('Product Name is ' + pname);

console.log('Product List is ' + listProduct);

**Output:**

****

**11.b) Course Name: Typescript**

**Module Name: Classes**

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

**Program:**

class Product

{

static productPrice: string;

productId: number;

constructor()

{

this.productId =1234;

}

getProductId(): string

{

return 'Product id is : ' + this.productId;

}

}

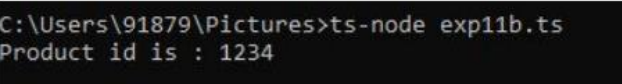
const product: Product = new Product();

const p={

producti :product.getProductId(), };

console.log(p.producti);

**Output:**

****

**11.c) Course Name: Typescript**

**Module Name: Constructor**

**Declare a class named - Product with the below-mentioned declarations:**

1. **productId as number property**
2. **Constructor to initialize this value**
3. **getProductId method to return the message "Product id is <>"**

**Program:**

**Program:**

class Product

{

static productPrice: string;

productId: number;

constructor(productId: number)

{

this.productId = productId;

}

getProductId(): string

{

return 'Product id is : ' + this.productId;

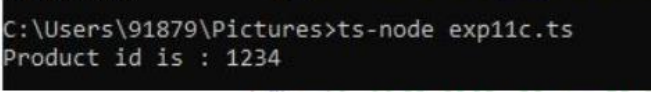
}

}

const product: Product = new Product(1234);

console.log(product.getProductId());

**Output:**

****

**11.d) Course Name: Typescript**

**Module Name: Access Modifiers**

**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.**

**Program:**

class Product

{

static productPrice = 150;

private productId: number;

public productName: string;

protected productCategory: string;

constructor(productId: number, productName:string , productCategory:string)

{

this.productId = productId;

this.productName = productName;

this.productCategory = productCategory;

}

getProductId()

{

console.log('The Product id is : ' + this.productId);

}

}

class Gadget extends Product

{

getProduct(): void

{

console.log('Product category is : ' + this.productCategory);

}

}

const g: Gadget = new Gadget(1234, 'Mobile', 'SmartPhone');

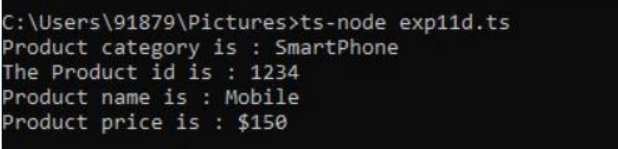
g.getProduct();

g.getProductId();

console.log('Product name is : ' + g.productName);

console.log('Product price is : $' + Product.productPrice);

**Output:**

****

**12.a) Course Name: Typescript**

**Module Name: Properties and Methods**

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

**Program:**

// declaring a Product class

class Product

{

static productPrice: string;

productId: number;

constructor(productId: number)

{

this.productId = productId;

}

getProductId(): string

{

return 'Product id is : ' + this.productId;

}

}

const product: Product = new Product(2345);

console.log(product.getProductId());

**Output:**

****

**12.b) Course Name: Typescript**

**Module Name: Creating and using Namespaces**

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

**Program:**

namespace\_one12b.ts:

import util = Utility.Payment;

let paymentAmount = util.CalculateAmount(1800, 6);

console.log(`Amount to be paid: ${paymentAmount}`);

namespace\_two12b.ts:

namespace Utility

{

export namespace Payment

{

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

{

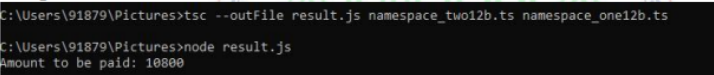
return price \* quantity;

}

}

}

**Output:**

****

**12.c Course Name: Typescript**

**Module Name: Creating and using Modules**

**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.**

**Program:**

module\_one12c.ts:

export function MaxDiscountAllowed(noOfProduct: number): number

{

if (noOfProduct > 5)

{

return 30;

}

else

{

return 10;

}

}

class Utility

{

CalculateAmount(price: number, quantity: number): number

{

return price \* quantity;

}

}

interface Category

{

getCategory(productId: number): string;

}

export const productName = 'Mobile';

export {Utility, Category};

module\_two12c.ts:

import { Utility as mainUtility, Category, productName, MaxDiscountAllowed } from

"./module\_one12c";

const util = new mainUtility();

const price = util.CalculateAmount(1350, 4);

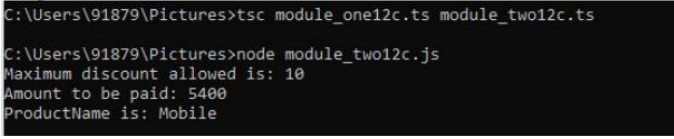
const discount = MaxDiscountAllowed(2);

console.log(`Maximum discount allowed is: ${discount}`);

console.log(`Amount to be paid: ${price}`);

console.log(`ProductName is: ${productName}`);

**Output:**

****

**12.d Course Name: Typescript**

**Module Name: What is Generics, What are Type Parameters, Generic Functions, Generic Constraints**

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

**Program:**

// declaring a Generic Array named orderDetails

function orderDetails<T>(arg: Array<T>): Array <T>

{

console.log(arg.length);

return arg;

}

const orderid: Array = [201, 202, 203, 204];

const ordername: Array = ['Dresses', 'Toys', 'Footwear', 'cds'];

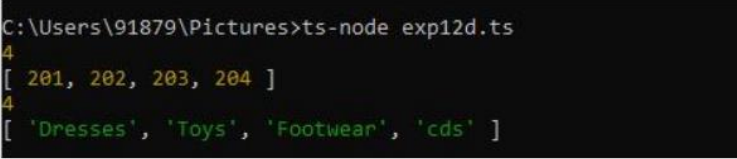
const idList = orderDetails(orderid);

console.log(idList);

const nameList = orderDetails(ordername);

console.log(nameList);

**Output:**

****