**1a) 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.**

**AIM:** 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.

**DESCRIPTION:**

**Case-insensitivity:**

HTML is not a Case Sensitive Language because, during parsing, all HTML elements are converted to lowercase first. HTML5, the latest version of HTML, which is a new web standard, is also case insensitive. XHTML, an older version of HTML, was case-sensitive for lowercase letters.

**Platform independency:**

The HyperText Markup Language (HTML) is the publishing language of the World Wide Web. The first version of HTML was described by Tim Berners-Lee in late 1991. The current W3C Recommendation for HTML is HTML5.

**Attributes:**

* All HTML elements can have **attributes**
* Attributes provide **additional information** about elements
* Attributes are always specified in **the start tag**
* Attributes usually come in name/value pairs like: **name="value"**

**Metadata Element:**

Metadata is data (information) about data. <meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings. Metadata will not be displayed on the page, but is machine parsable.

**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">

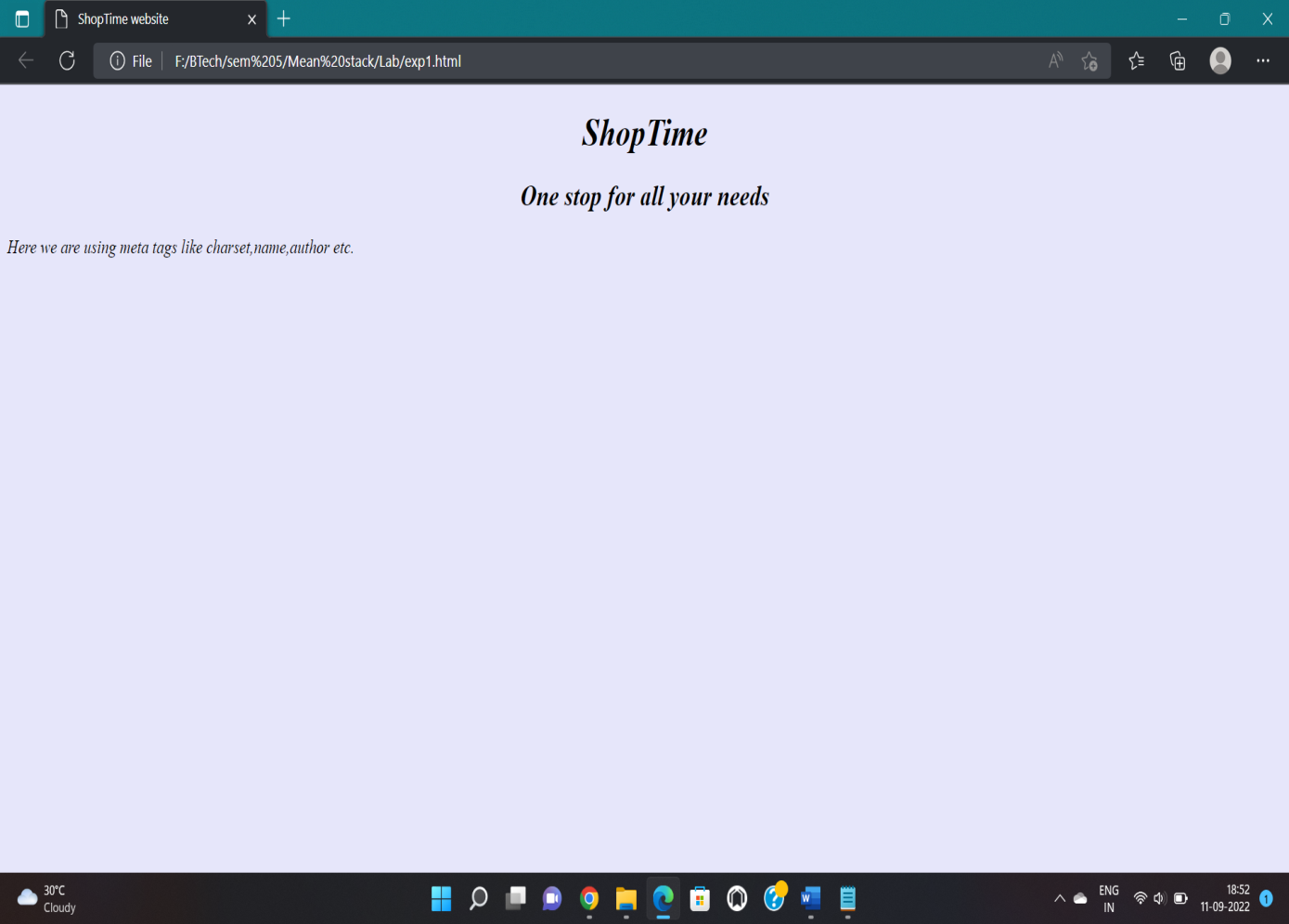
<h1 align="center"><i>ShopTime<i></h1>

<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>

**OUTPUT:**



**Viva Questions:**

**1.What is Doctype and Types of elements HTML?**

A.All **HTML** documents must start with a <!**DOCTYPE**> declaration. The declaration is not an **HTML** tag. It is an "information" to the browser about what document type to expect. In **HTML** 5, the declaration is simple: <!**DOCTYPE html**> Browser Support. Element …

An HTML element is defined by a start tag, some content, and an end tag.

## **HTML Elements**

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

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

Examples of some HTML elements:

<h1>My First Heading</h1>

<p>My first paragraph.</p>

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

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

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

**DESCRIPTION:**

**Sectioning Elements:**

In HTML, a section is a semantic element for creating standalone sections in a web page. These sections should be made up of related content, like contact information.

The section element should only be used if there isn't a more specific element to represent the related content.

The sectioning elements in HTML5 are **<nav> , <aside> , <article> , and <section> . <body>** is also kind of a sectioning element since all content lying inside of it is part of the default document section.

**How to Use Section in HTML**

To use sections in HTML, wrap the related elements you want to group together in section tags. Most sections should contain a heading.

**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>

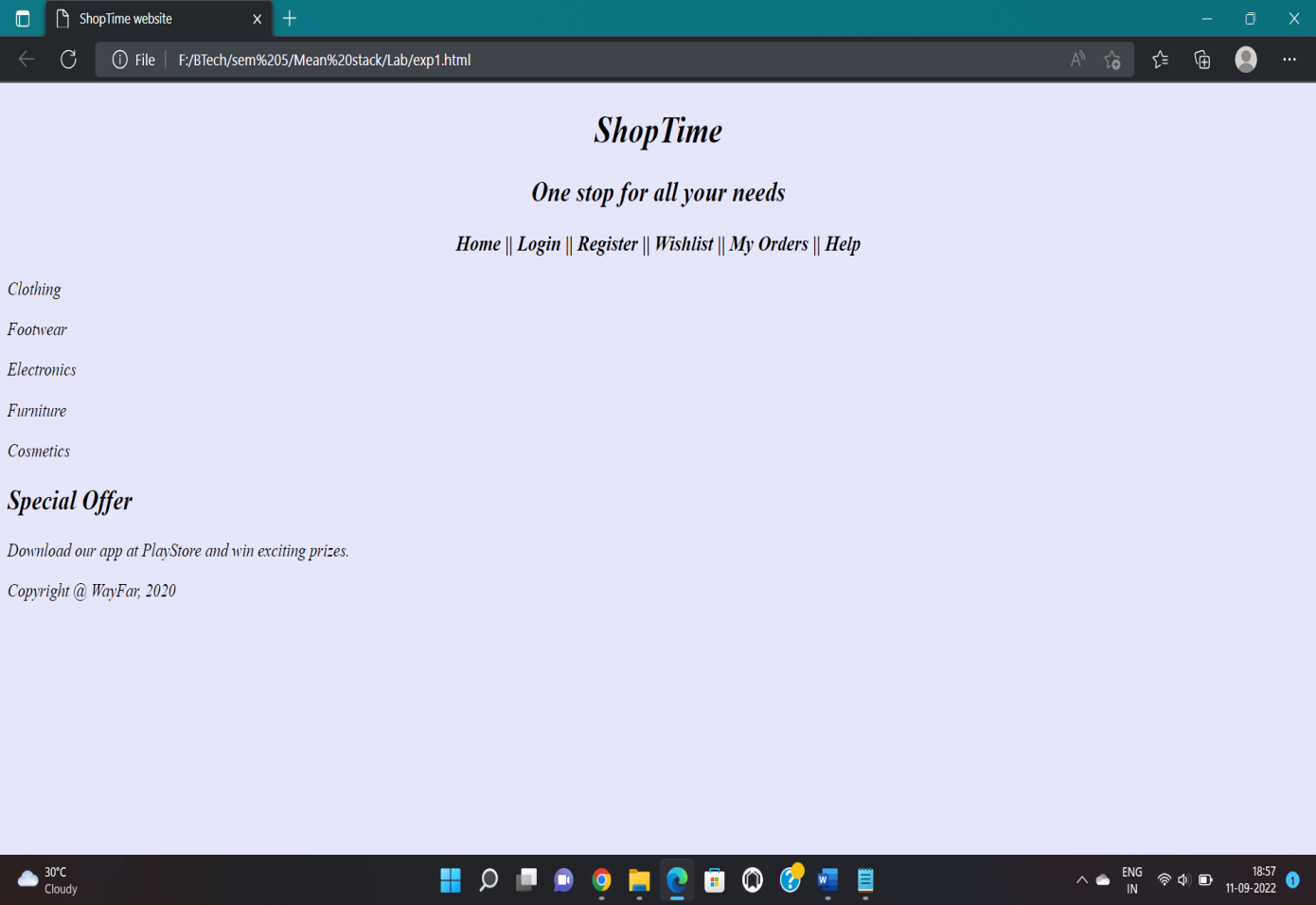
<footer>

Copyright @ WayFar, 2020

</footer>

</html>

**OUTPUT:**



**Viva Questions:**

1.**What are different types of Sectioning elements?**

a.The sectioning elements in HTML5 are **<nav> , <aside> , <article> , and <section> . <body>** is also kind of a sectioning element since all content lying inside of it is part of the default document section. This experiment is mapped with PO1,PO2,PO3,PO5,PO10,PO12.

**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.**

**AIM :**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.

**DESCRIPTION:**

**<div>**

The div (division) element is a generic block-level element, most often used to divide page content into blocks. A block element is a page element that starts a new line and has a width equal to the entire page or the parent container.

You’ll very often see divs used to group related paragraphs, images, headings, and links. For example, a three-paragraph article may be enclosed in a div, and a navigation menu containing links might be enclosed in another div. Using divs this way makes it easier to identify different sections of a page and apply styling to them with CSS.

**<span>**

The span element is a generic inline element, typically used to apply styling to a portion of inline content. An inline element does not start a new line and only takes up as much space on the page as its content. Span tags are used on small segments of text, links, images, and other HTML elements that appear inline with the surrounding content.

**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>

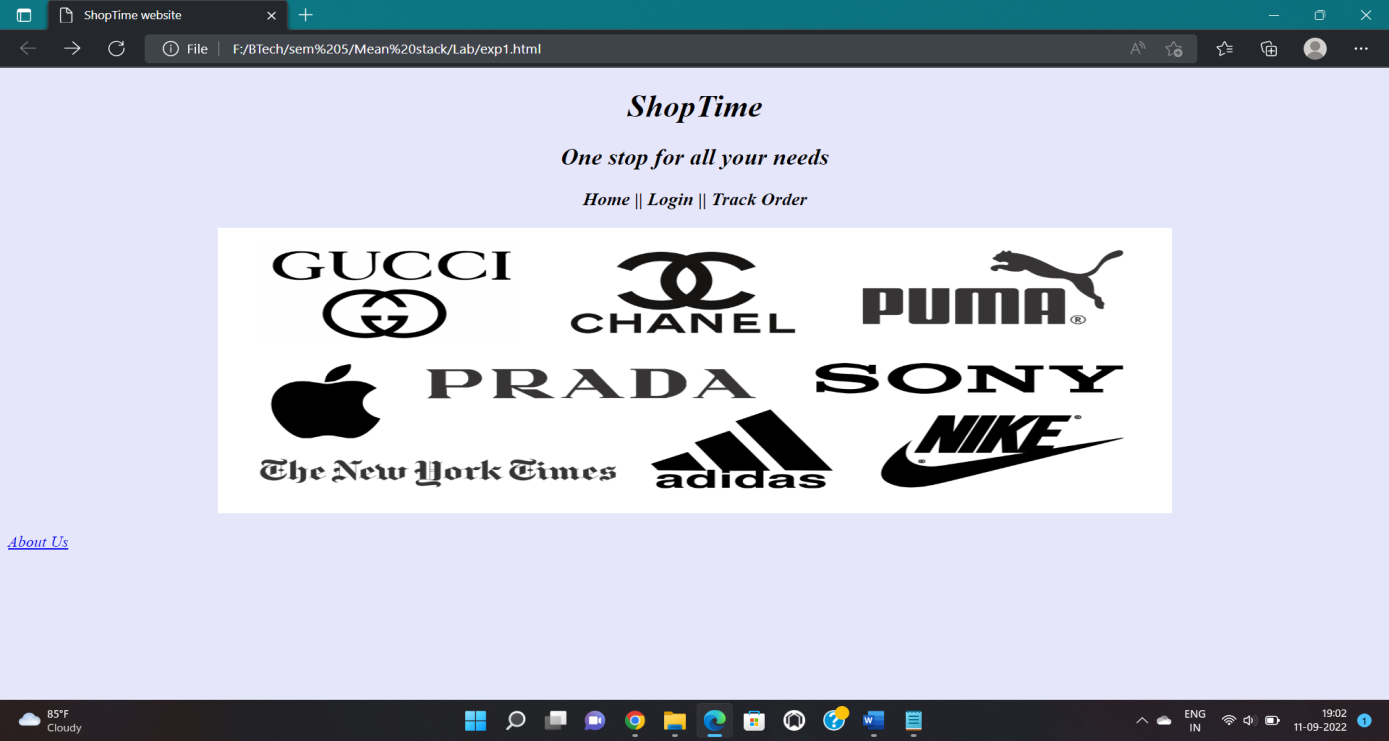
<footer>

<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>

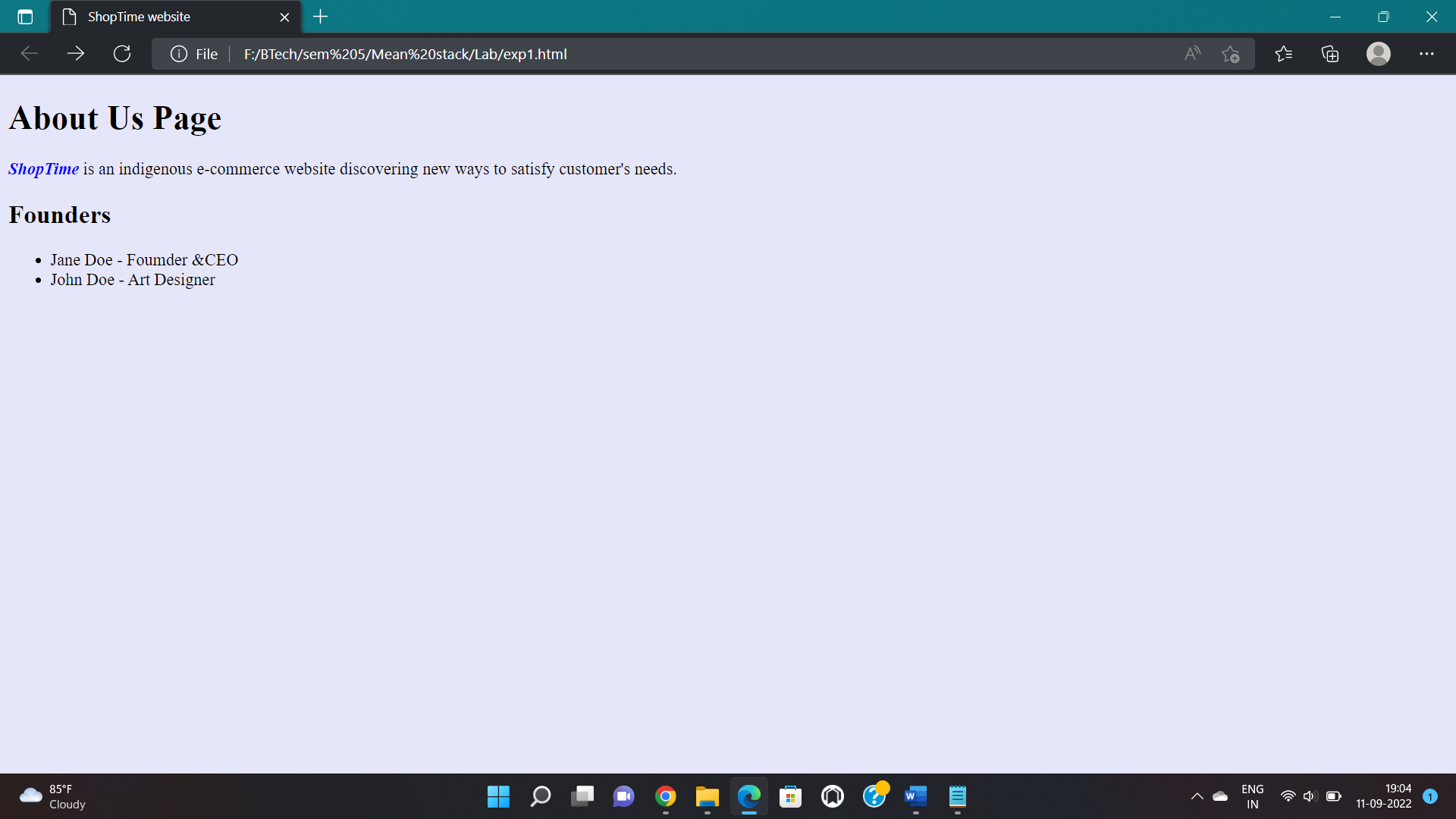
<li>John Doe - Art Designer</li>

</ul>

</body>

</html>

**OUTPUT:**



**Viva Question:**

**1.What is Paragraph Element?**

**a.**The <p> HTML element represents a paragraph. Paragraphs are usually represented in visual media as blocks of text separated from adjacent blocks by blank lines and/or first-line indentation, but HTML paragraphs can be any structural grouping of related content, such as images or form fields.

**2.What is Division Element?**

**a.**The div tag is known as Division tag. The div tag is used in HTML to make divisions of content in the web page like (text, images, header, footer, navigation bar, etc). Div tag has both open(<div>) and closing (</div>) tag and it is mandatory to clos..

**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**.

**AIM :** 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.

**DESCRIPTION**:

**Link Element**:

The link element allows authors to link their document to other resources.

The destination of the link(s) is given by the href attribute, which must be present and must contain a valid non-empty URL potentially surrounded by spaces.

A link element must have rel attribute.

The types of link indicated (the relationships) are given by the value of the rel attribute, which, if present, must have a value that is a set of space-separated tokens. The allowed keywords and their meanings are defined in a later section.

Two categories of links can be created using the link element: Links to external resources and hyperlinks. The link types section defines whether a particular link type is an external resource or a hyperlink. One link element can create multiple links (of which some might be external resource links and some might be hyperlinks); exactly which and how many links are created depends on the keywords given in the rel attribute. User agents must process the links on a per-link basis, not a per-element basis.

**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>

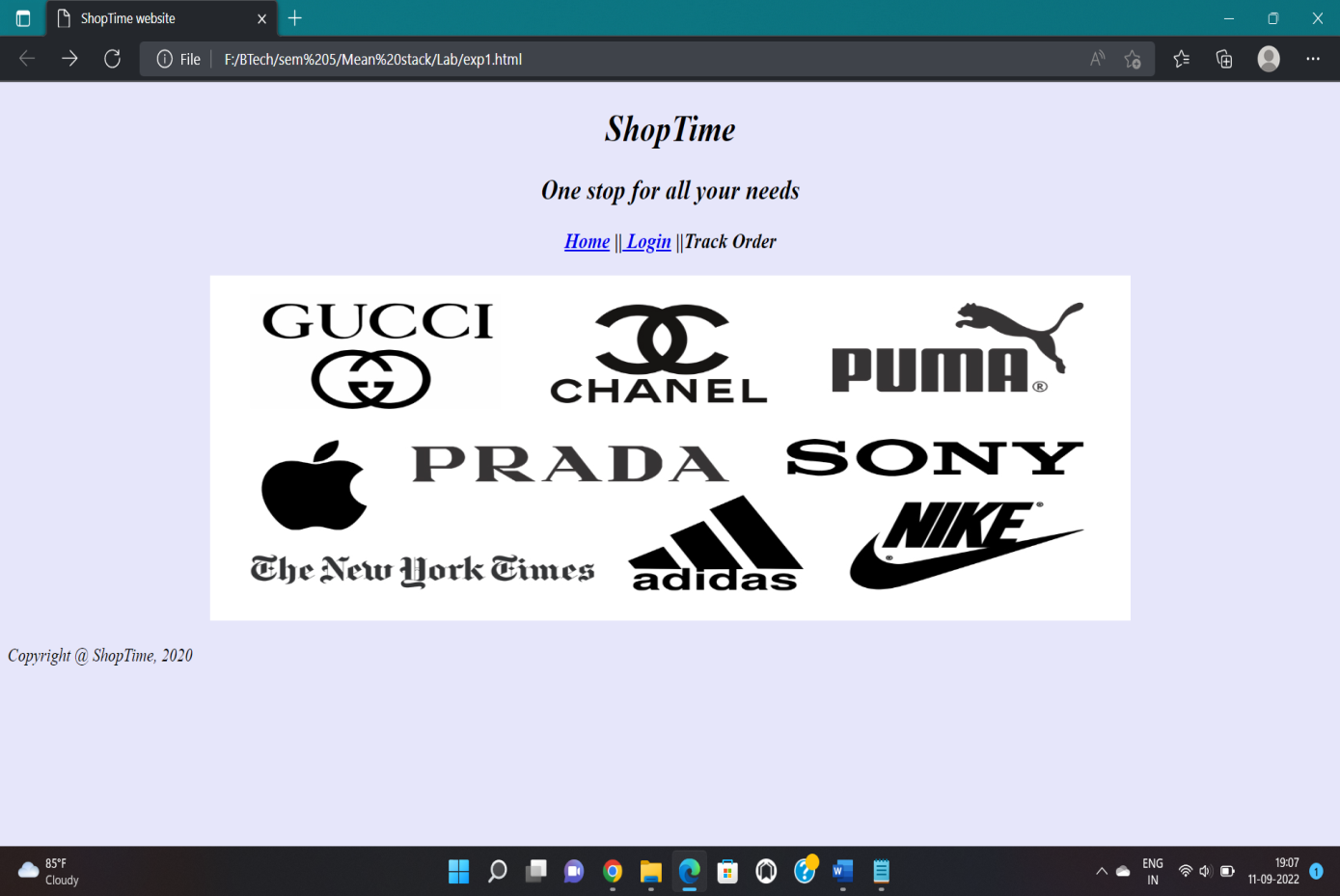
</body>

<footer>

Copyright @ ShopTime, 2020

</footer></html>

**OUTPUT:**



**Viva Question:**

**1)What is Use of Link Element in HTML?**

**A.**The link element allows authors to link their document to other resources.

The destination of the link(s) is given by the href attribute, which must be present and must contain a valid non-empty URL potentially surrounded by spaces.

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

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

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

**DECRIPTION:**

Some characters are reserved in HTML.

If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags.

Character entities are used to display reserved characters in HTML.

A character entity looks like this:

&entity\_name;

OR

&#entity\_number;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Result** | **Description** | **Entity Name** | **Entity Number** |  |
|  | non-breaking space | &nbsp; | &#160; |
| < | less than | &lt; | &#60; |  |
| > | greater than | &gt; | &#62; |
| & | ampersand | &amp; | &#38; |
| " | double quotation mark | &quot; | &#34; |
| ' | single quotation mark (apostrophe) | &apos; | &#39; |
| ¢ | cent | &cent; | &#162; |  |
| £ | pound | &pound; | &#163; |
| ¥ | yen | &yen; | &#165; |
| € | euro | &euro; | &#8364; | [Try it »](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_ent_euro" \t "_blank) |
| © | copyright | &copy; | &#169; |  |
| ® | registered trademark | &reg; | &#174; |

**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>

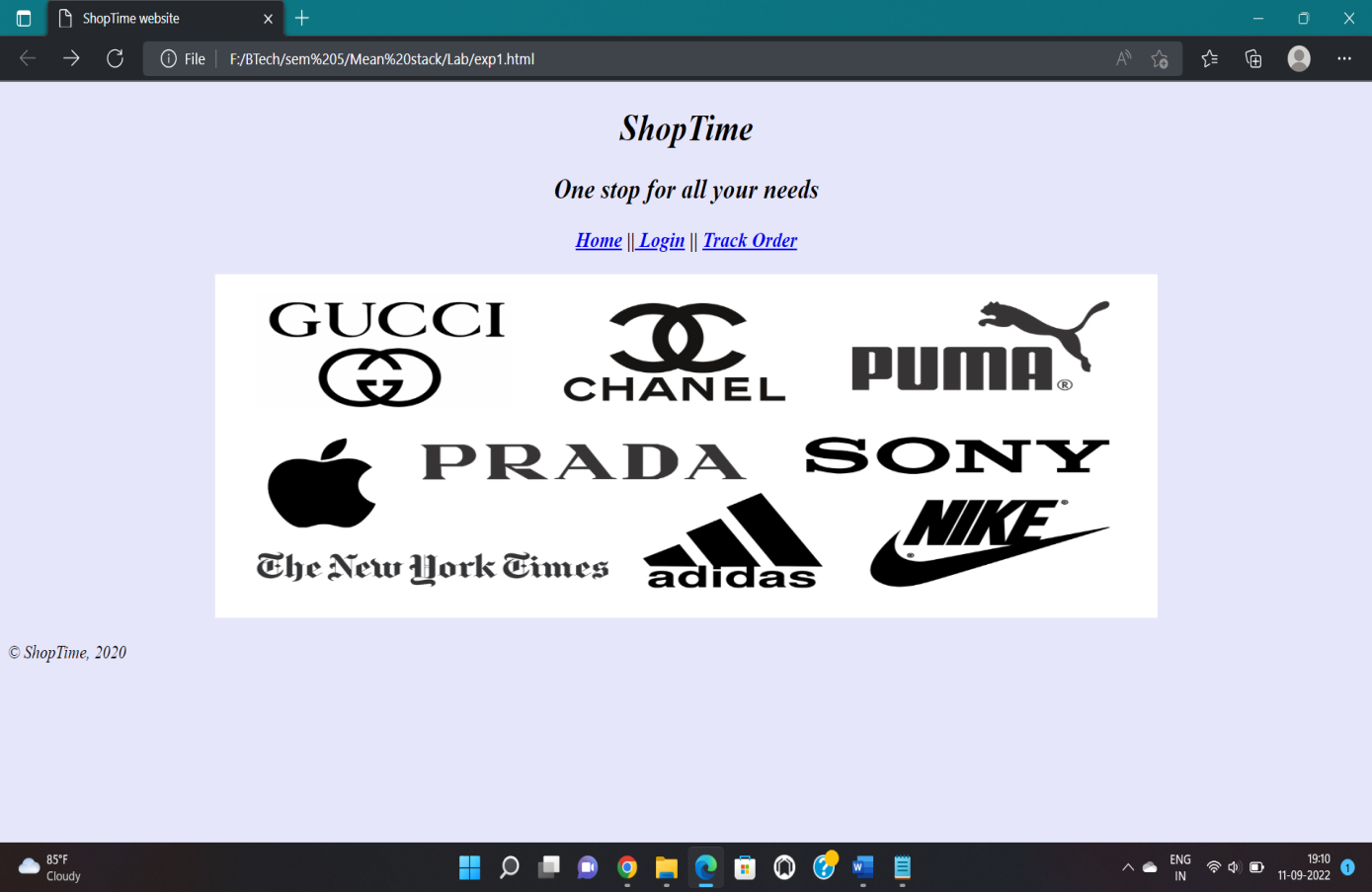
</body>

<footer>

&copy; ShopTime, 2020

</footer></html>

**OUTPUT:**



**Viva Question:**

**1.What are Different types of Character Entities?**

**a.**Reserved characters in HTML must be replaced with character entities.

## **HTML Entities**

Some characters are reserved in HTML.

If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags.

Character entities are used to display reserved characters in HTML.

**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.**

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

**DESCRIPTION:**

**Global Attributes:**

The global attributes are attributes that can be used with all HTML elements.

The contenteditable attribute specifies whether the content of an element is editable or not. Note: When the contenteditable attribute is not set on an element, the element will inherit it from its parent.

The spellcheck attribute specifies whether the element is to have its spelling and grammar checked or not.

The following can be spellchecked:

Text values in input elements (not password)

Text in <textarea> elements

Text in editable elements

**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>

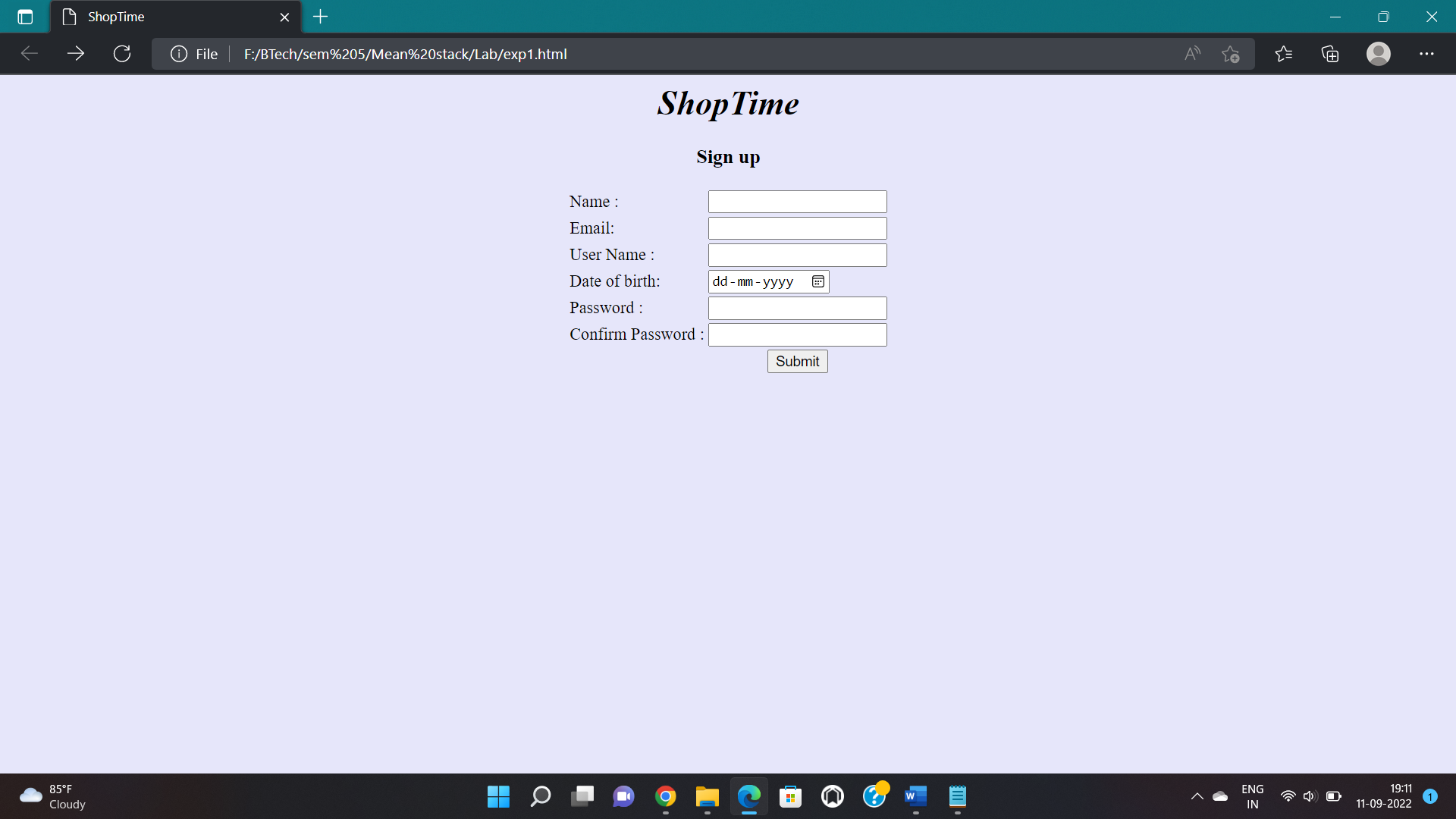
<tr><td></td><td align="center"><input type="submit"></td></tr>

</center>

</form>

</body></html>

**OUTPUT:**



**Viva Question:**

**1.What is Importance of Global Attributes?**

**a.**The global attributes are attributes that can be used with all HTML elements.The contenteditable attribute specifies whether the content of an element is editable or not.

**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.**

**AIM:**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.

**DESCRIPTION:**

The <table> tag defines an HTML table.

An HTML table consists of one <table> element and one or more <tr>, <th>, and <td> elements.

The <tr> element defines a table row, the <th> element defines a table header, and the <td> element defines a table cell.

An HTML table may also include <caption>, <colgroup>, <thead>, <tfoot>, and <tbody> elements.

Each table cell is defined by a <td> and a </td> tag.

td stands for table data.

Everything between <td> and </td> are the content of the table cell.

**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>

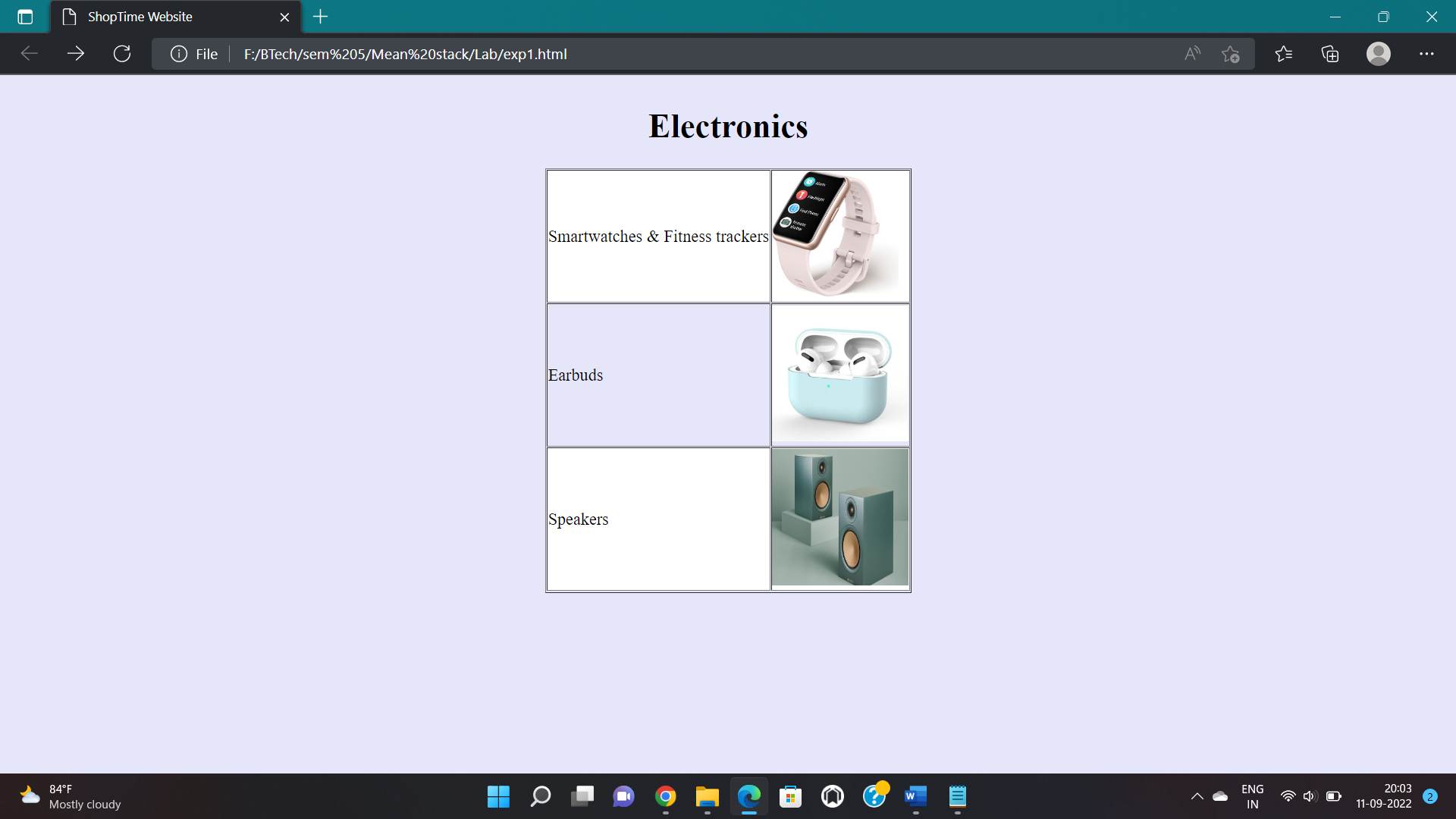
<tr bgcolor="white"><td>Speakers</td><td><img src="https://www.monitoraudio.com/site/assets/files/33185/silver100le-actionblock-2.jpg" width = "130px" height = "130px" alt="Speakers"></td></tr></b>

</table>

</body>

</html>

**Output:**



**Viva Question:**

**1.What are the table elements in html?**

**A.** An HTML table consists of one <table> element and one or more <tr>, <th>, and <td> elements. The <tr> element defines a table row, the <th> element defines a table header, and the <td> element defines a table cell.

**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.**

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

**DESCRIPTION:**

An HTML form is used to collect user input. The user input is most often sent to a server for processing.

The HTML <form> element is used to create an HTML form for user input:

<form>

.

form elements

.

</form>

The <form> element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

All the different form elements are covered in this chapter: HTML Form Elements.

**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>

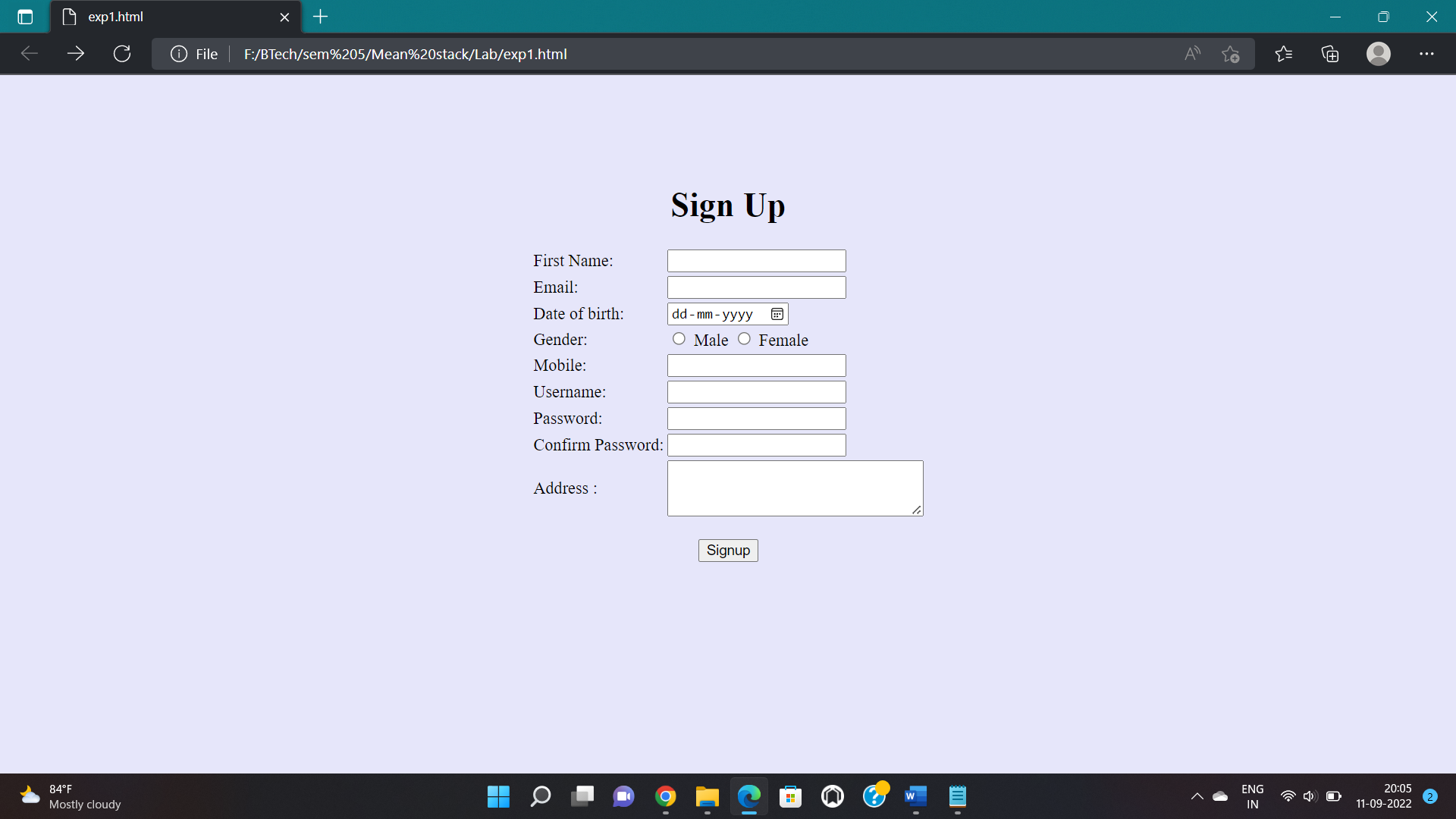
<br><button type="submit">Signup</button>

</form>

</body>

</html>

**OUTPUT:**



**Viva Questions:**

**1.What is the importance of form element?**

A. An HTML form is used to collect user input. The user input is most often sent to a server for processing.

**2.What is Datalist Element?**

**A.** The <datalist> tag specifies a list of pre-defined options for an <input> element.

The <datalist> tag is used to provide an "autocomplete" feature for <input> elements. Users will see a drop-down list of pre-defined options as they input data.

**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.**

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

**DESCRIPTION:**

Here are the different input types you can use in HTML:

<input type="button">

<input type="checkbox">

<input type="color">

<input type="date">

<input type="datetime-local">

<input type="email">

<input type="file">

<input type="hidden">

<input type="image">

<input type="month">

<input type="number">

<input type="password">

<input type="text"> defines a single-line text input field

<input type="password"> defines a password field

**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>

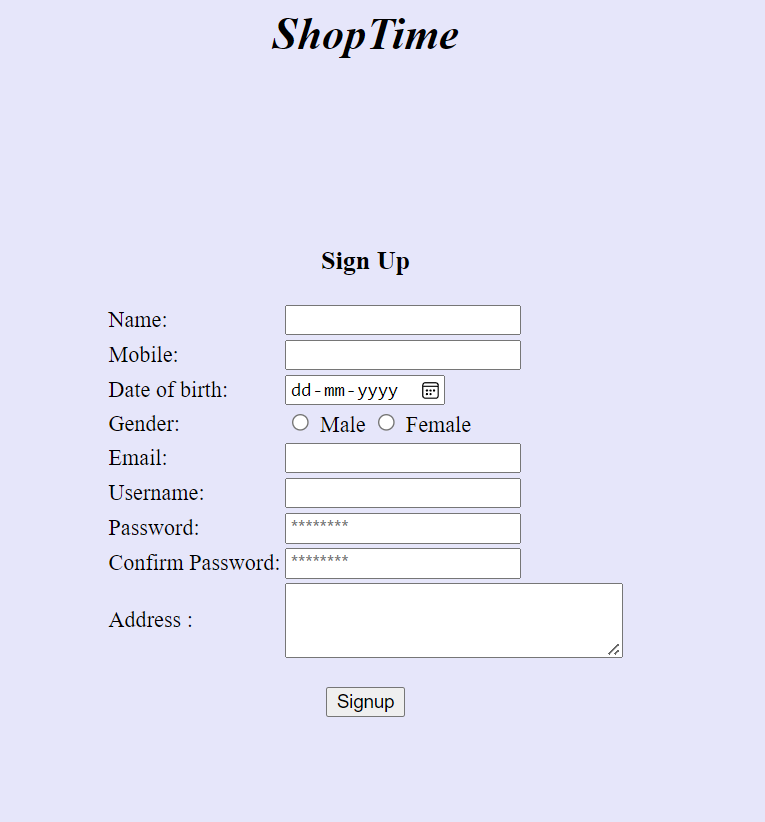
<br><button type="submit">Signup</button>

</form>

</body>

</html>

**Output:**



**Viva Question**

**1. What are the new input attributes introduce by HTML5?**

A. HTML5 has introduced new attributes like date, color, month, time, week, datetime-local, email, number, range, tel, url, search etc.

**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**.

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

**DESCRIPTION:**

The **media** attribute specifies what media/device the linked document is optimized for.

This attribute is used to specify that the target URL is designed for special devices (like iPhone) , speech or print media.

This attribute can accept several values.

**Iframe:**

An HTML iframe is used to display a web page within a web page.

The HTML <iframe> tag specifies an inline frame.

An inline frame is used to embed another document within the current HTML document.

The HTML **<video>** element is used to show a video on a web page.

The controls attribute adds audio controls, like play, pause, and volume.

The <source> element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.

The text between the **<audio> and </audio>** tags will only be displayed in browsers that do not support the <audio> element.

**Syntax:**

<iframe src="url" title="description"></iframe>

**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>

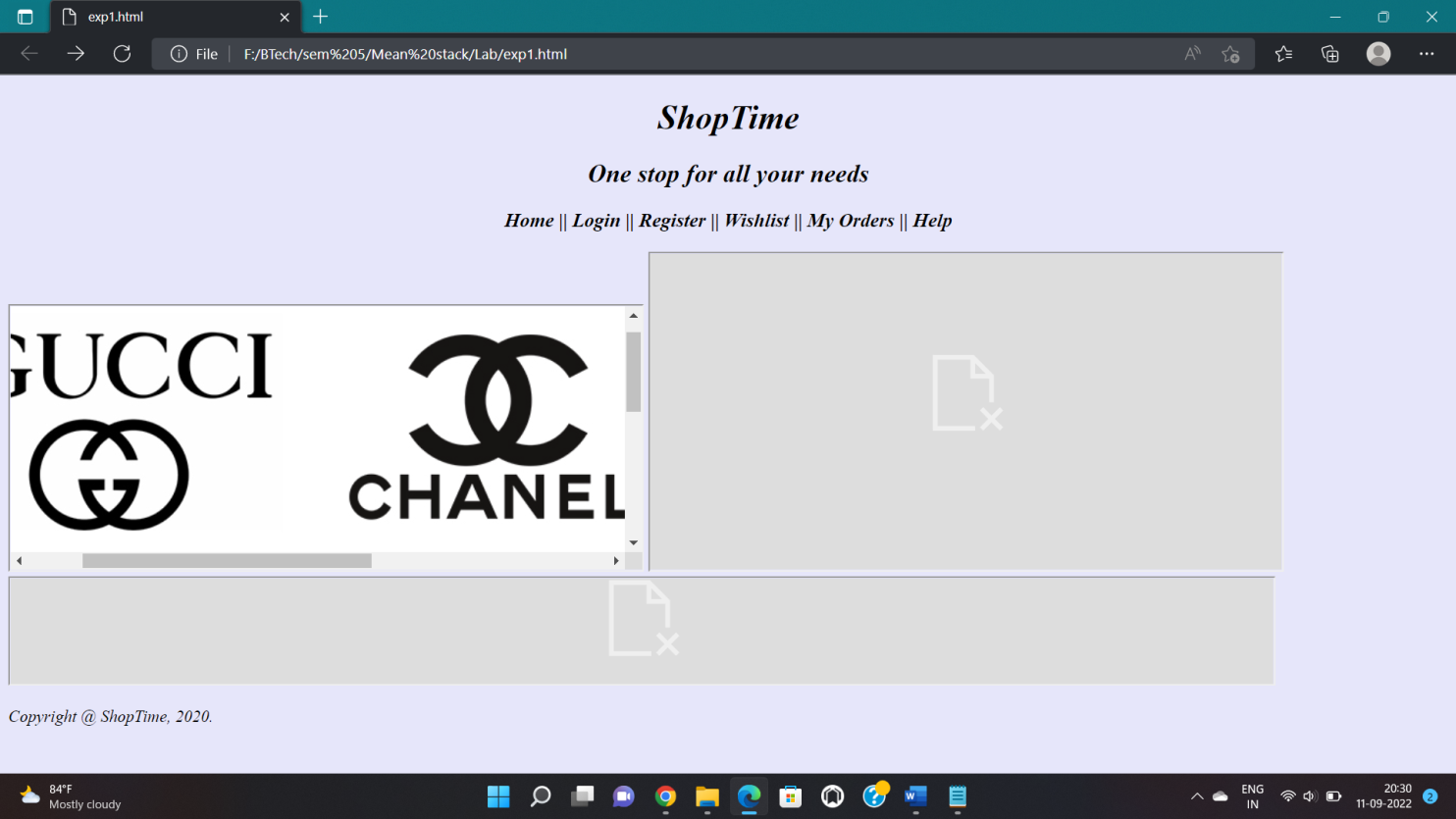
<footer>

Copyright @ ShopTime, 2020.

</footer>

</html>

**Output:**



**Viva Questions**

**1.What is the syntax of iframe element?**

A.<iframe src="url" title="description"></iframe>

**2.What is the use of Media element in HTML?**

The media attribute specifies what media or device the linked document is optimized for. This attribute is used to specify that the target URL is designed for special devices (like iPhone), speech or print media.

**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).**

**AIM:**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).

**DESCRIPTION:**

JavaScript Identifiers are names given to variables, functions, etc. It is the same as identifiers in other programming languages like C, C++, Java, etc. Let’s see identifiers for variable names.

The following are legal variable names −

val

val1

result

While naming your variables in JavaScript, keep the following rules in mind.

You should not use any of the JavaScript reserved keywords as a variable name. These keywords are mentioned in the next section. For example, break or boolean variable names are not valid.

JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, 5demo is an invalid variable name but \_5demo is a valid one.

JavaScript variable names are case-sensitive. For example, Name and name are two different variables.

**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;

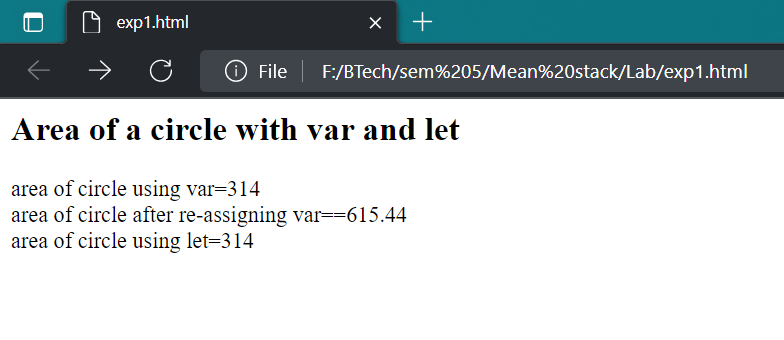
document.write("Area of circle using let="+area+"<br>");

</script>

</body>

</html>

**OUTPUT:**



**Viva Questions:**

**1.What is an identifier?**

A.An identifier is a sequence of characters in the code that identifies a variable, function, or property. In JavaScript, identifiers are case-sensitive and can contain Unicode letters, $ , \_ , and digits (0-9), but may not start with a digit.

**2.** **What are the different types of variables in JavaScript?**

A.There are two types of variables in JavaScript : local variable and global variable.

**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.**

**AIM:** 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.

**DESCRIPTION :**

Every Variable has a data type that tells what kind of data is being stored in a variable. There are two types of data types in JavaScript.

Primitive data types

Non-primitive data types

**Primitive data types:** The predefined data types provided by JavaScript language are known as primitive data types. Primitive data types are also known as in-built data types.

Below is a list of Primitive Data Types:

1.Number

2.String

3.Undefined

4.Boolean

5.Null.....etc

**Non-primitive data types:** The data types that are derived from primitive data types of the JavaScript language are known as non-primitive data types. It is also known as derived data types or reference data types.

Below is a list of Non-primitive data types.

Non-primitive Data Types

Object

Array

**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:**



**Viva Qusetions**

**1. How to identify data type in JavaScript?**

You can use the typeof operator to find the data type of a JavaScript variable**.**

**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**.

**AIM:**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 .

**DESCRIPTION:**

**Operator :**

JavaScript operators are symbols that are used to perform operations on operands. For example:

var sum=10+20;

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

* Arithmetic Operators
* Comparison (Relational) Operators
* Bitwise Operators
* Logical Operators
* Assignment Operators
* Special Operators

**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 ;

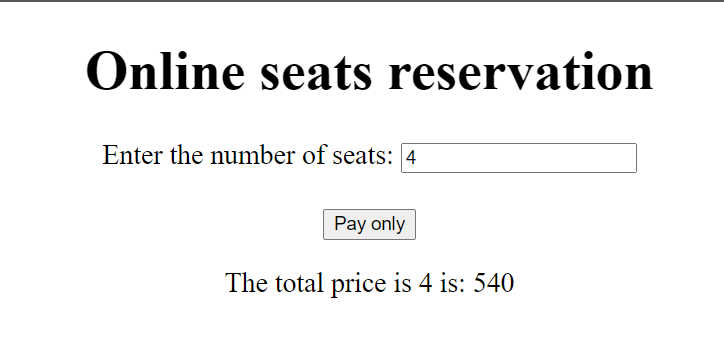
}

</script>

</body>

</html>

**OUTPUT:**



**Viva Questions**

**1.Which operator is overloaded in JavaScript?**

A. Operators can be overloaded on certain JavaScript objects that declare themselves as having overloaded operators. The following operators may have overloaded behavior: Mathematical operators: unary + , - , ++ , -- ; binary + , - , \* , / , % , \*\*

**2.Why do we use operators in JS?**

A.Operators are used to assign values, compare values, perform arithmetic operations, and more. There are different types of JavaScript operators: Arithmetic Operators. Assignment Operators.

**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.**

**AIM:**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.

**DESCRIPTION :**

**Conditional Statements**

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

Use if to specify a block of code to be executed, if a specified condition is true

Use else to specify a block of code to be executed, if the same condition is false

Use else if to specify a new condition to test, if the first condition is false

Use switch to specify many alternative blocks of code to be executed

**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");

}

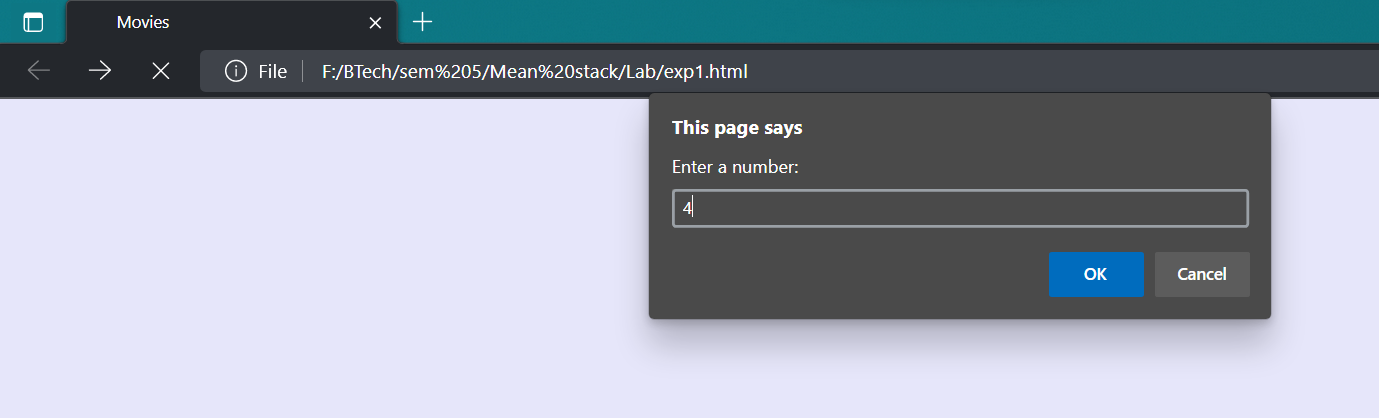
}

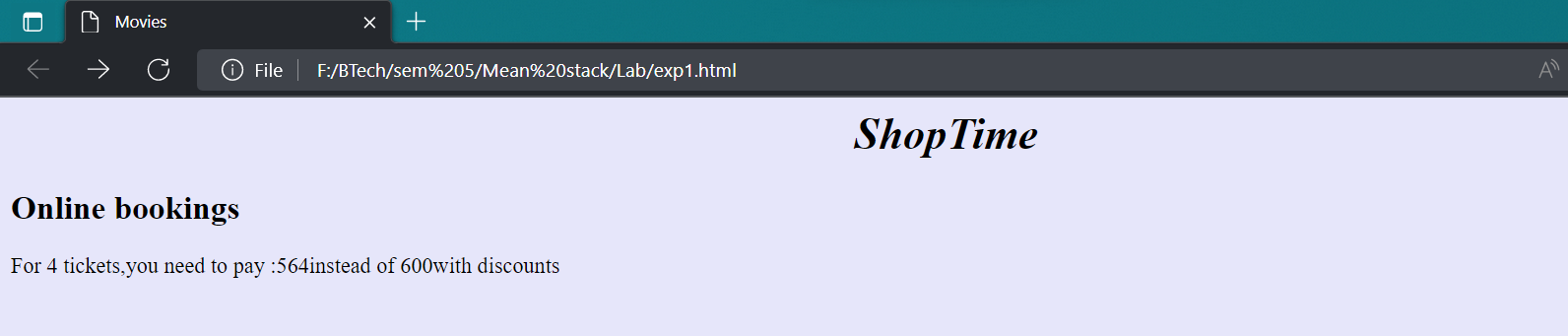
</script>

</body>

</html>

**OUTPUT:**





**Viva Questions**

**1. What is an example of a conditional statement in JavaScript?**

A. If Statement is an Example of conditional statement.

**2.How many conditional statements do we have in JavaScript?**

A.We have four types of conditional statements in JavaScript: An if statement executes a specified code segment if the given condition is ''true. '

**3.Write the Syntax for switch Statement?**

A.switch(expression) {

case x:

// code block

break;

case y:

// code block

break;

default:

// code block

}

**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.**

**AIM:**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.

**DESCRIPTION:**

**JavaScript Loops**

Loops can execute a block of code a number of times.

Loops are handy, if you want to run the same code over and over again, each time with a different value.

**Different Kinds of Loops**

JavaScript supports different kinds of loops:

for - loops through a block of code a number of times

for/in - loops through the properties of an object

for/of - loops through the values of an iterable object

while - loops through a block of code while a specified condition is true

do/while - also loops through a block of code while a specified condition is true

**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");

}

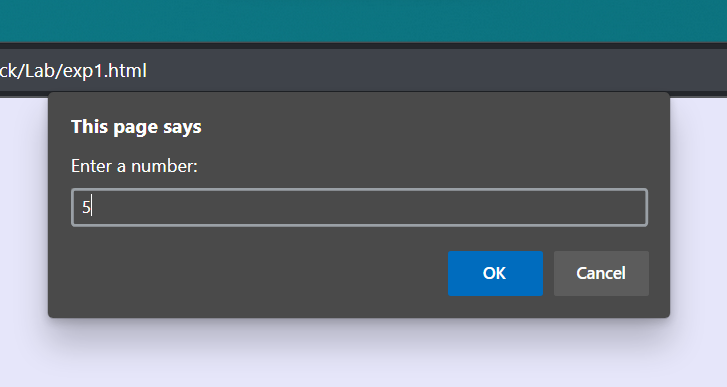
}

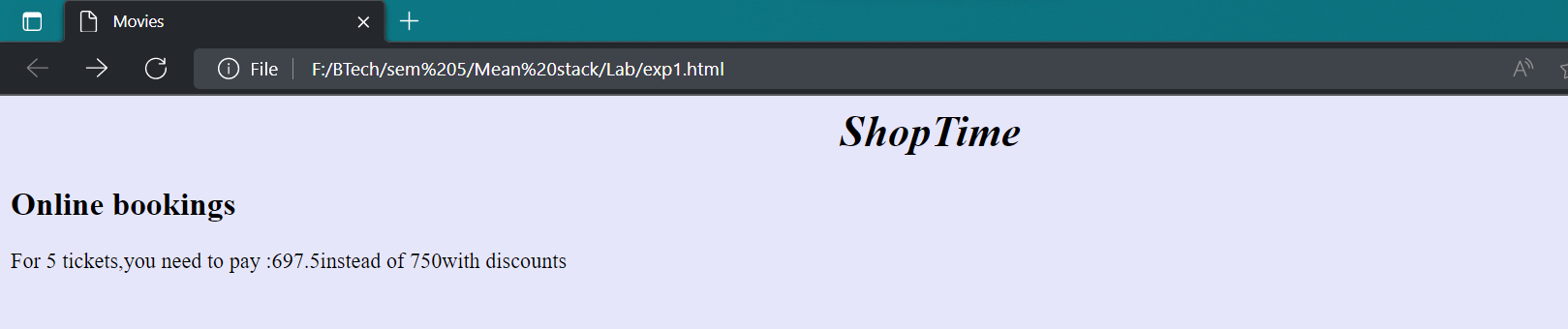
</script>

</body>

</html>

**OUTPUT:**





**Viva Questions**

**1.** **What is the purpose of loops in JavaScript?**

A.Loops are used in JavaScript to perform repeated tasks based on a condition. Conditions typically return true or false . A loop will continue running until the defined condition returns false .

**2.What are the advantages of loops?**

A. 1) It provides code reusability.

2) Using loops, we do not need to write the same code again and again. 3) Using loops, we can traverse over the elements of data structures (array or linked lists).

**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.**

**AIM:** To write a Javascript code to book movie tickets online and calculate the total price based on the given 3 conditions using functions.

**DESCRIPTION**: A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses (). Function names can contain letters, digits, underscores, and dollar signs (same rules as variables). The parentheses may include parameter names separated by commas: (parameter1, parameter2, ...) **Syntax:**

function name(parameter1, parameter2, parameter3)

{

// code to be executed

}

**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>

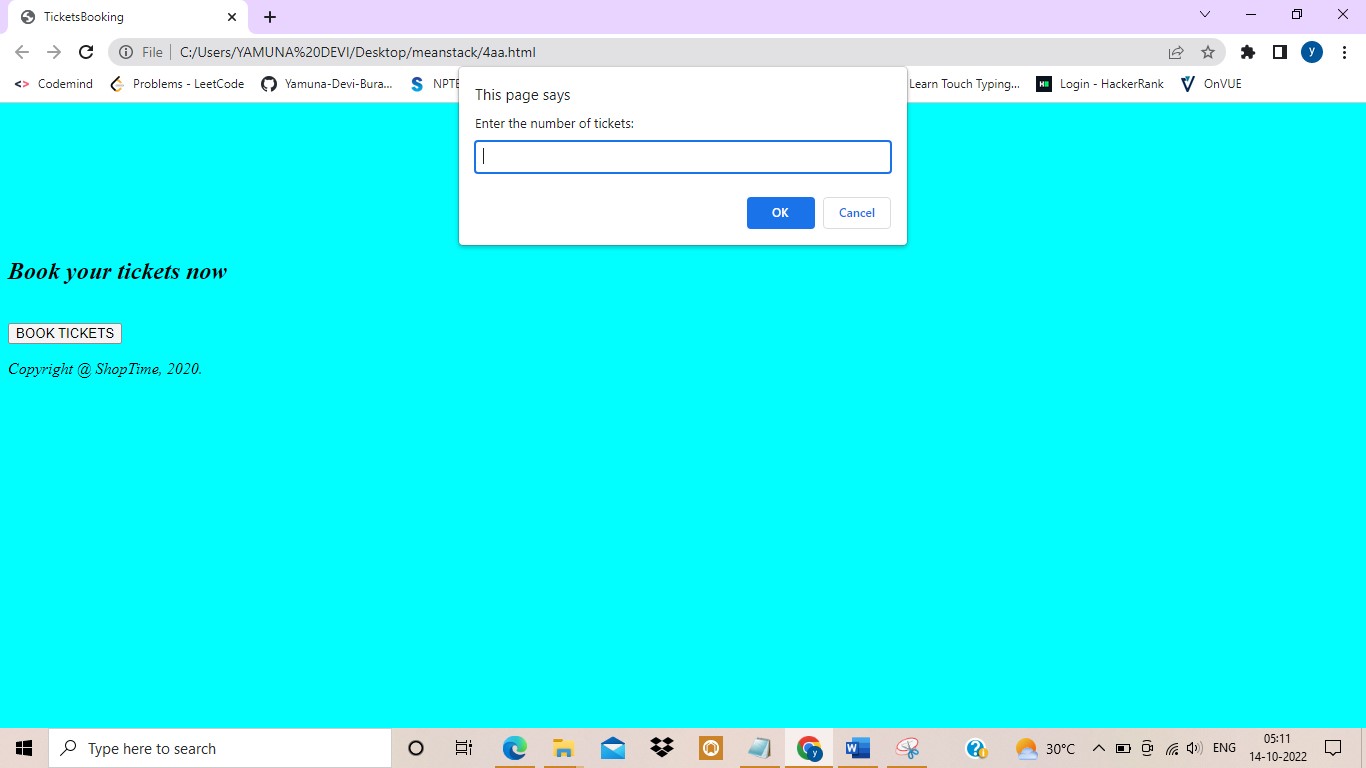
<footer>

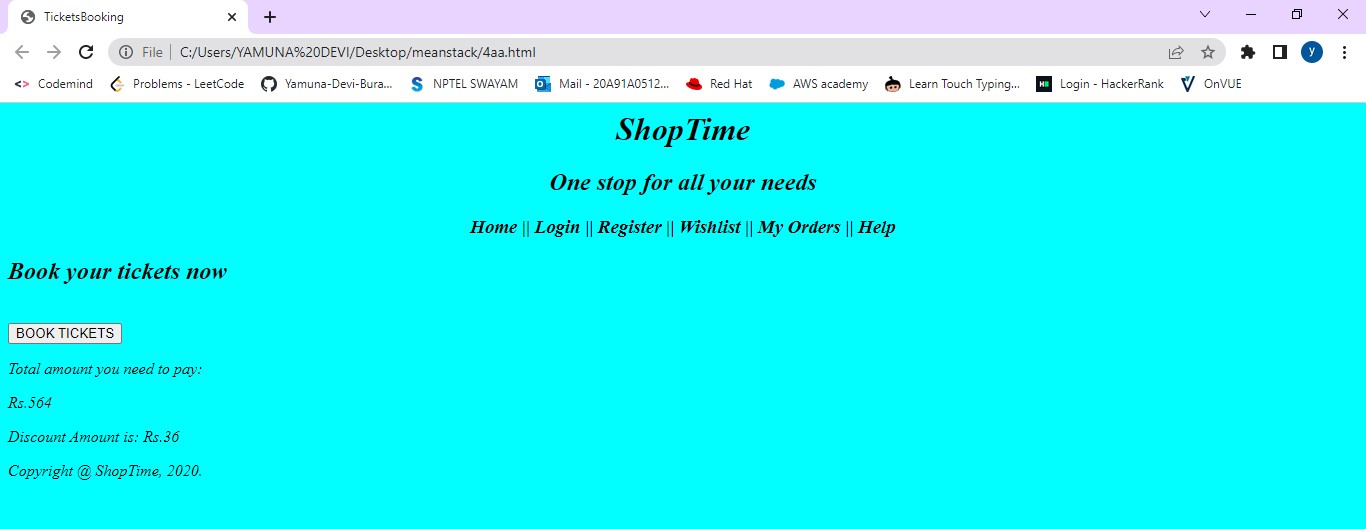
Copyright @ ShopTime, 2020.

</footer>

</html>

**OUTPUT:**





**Viva Questions**

**1.What is an arrow function?**

A.Arrow function is one of the features introduced in the ES6 version of JavaScript. It allows you to create functions in a cleaner way compared to regular functions. For example, This function // function expression let x = function(x, y) { return x \* y; }

**2.What is difference between function and arrow function in JavaScript?**

A.Since regular functions are constructible, they can be called using the new keyword. However, the arrow functions are only callable and not constructible, i.e arrow functions can never be used as constructor functions. Hence, they can never be invoked with the new keyword.

**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**

**AIM**:To write a Javascript with classes,creating and inheriting classes.

**DESCRIPTION:**

To create a class inheritance, use the extends keyword. The super() method refers to the parent class. By calling the super() method in the constructor method, we call the parent's constructor method and gets access to the parent's properties and methods. Classes also allows you to use getters and setters. It can be smart to use getters and setters for your properties, especially if you want to do something special with the value before returning them, or before you set them. To add getters and setters in the class, use the get and set keywords.

**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>

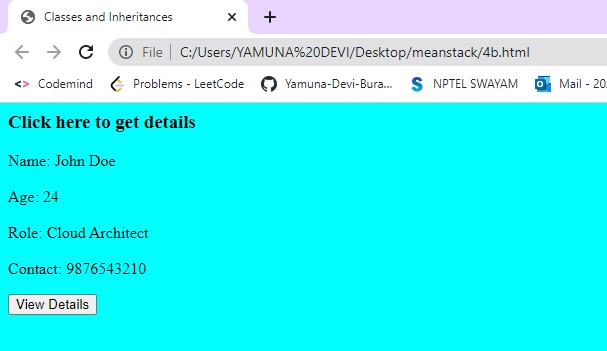
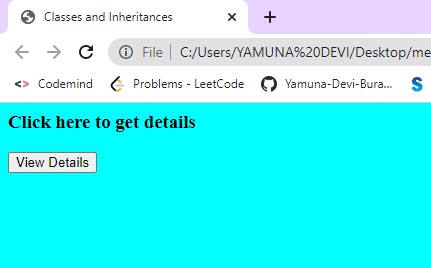
<p id="id1">

</p></center>

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

</body>

</html>



**Output**

**Viva Questions**

**1.** **What is the difference between a class and a function in JavaScript?**

A.Functions, expressions and declarations can be overwritten, whereas classes can be extended, but not overwritten.

**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.**

**AIM:** To write a Javascript code to book movie tickets online and calculate the total price.

**DESCRIPTION:**

HTML events are "things" that happen to HTML elements. An HTML event can be something the browser does, or something a user does.

JavaScript lets you execute code when events are detected. HTML allows event handler attributes, with JavaScript code, to be added to HTML elements.

**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>

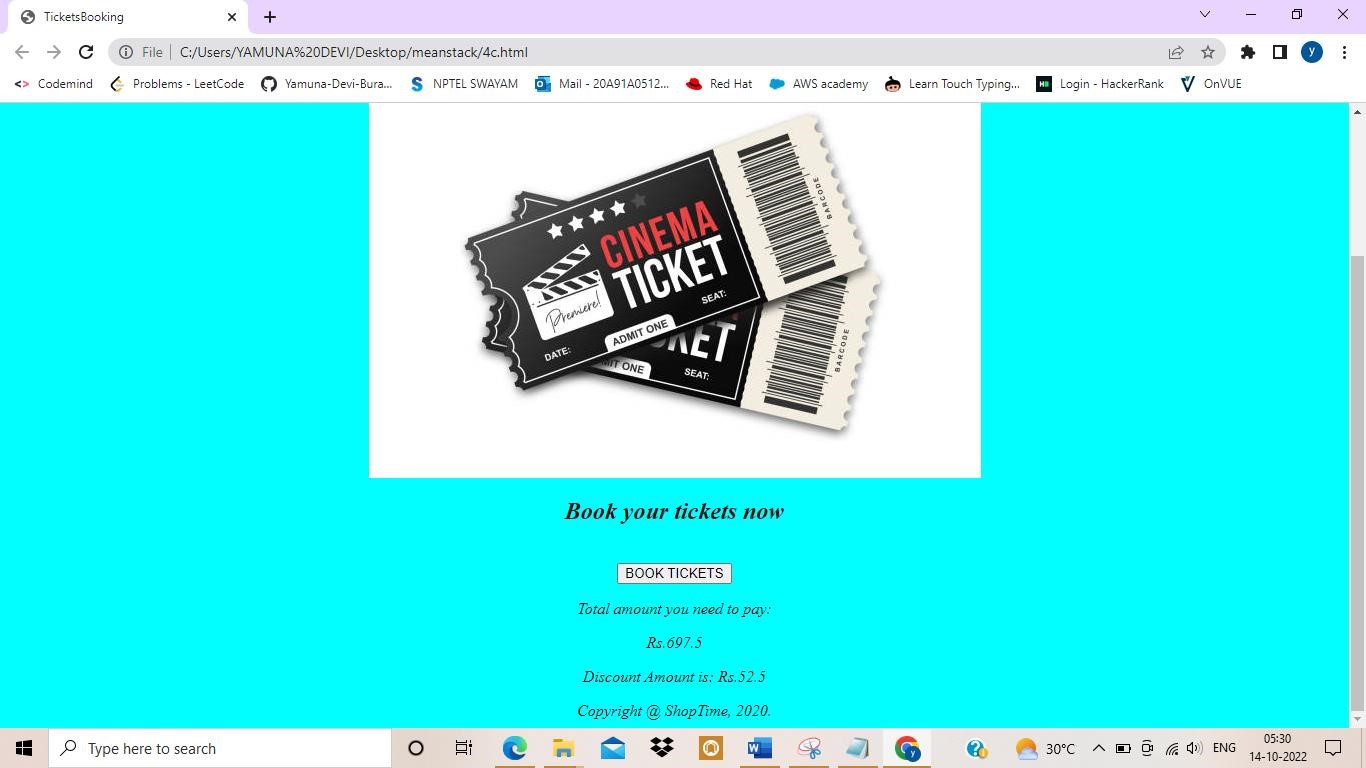
Copyright @ ShopTime, 2020.

</footer></center>

</html>

**OUTPUT:**





**Viva Questions**

**1.What are different types of events in JavaScript?**

A.Image result for in-built events and handlers in js

Types of Events in JavaScript

Onclick.

Onkeyup.

Onmouseover.

Onload.

Onfocus.

**2.** **What are events and event handlers in JavaScript?**

A.In html, there are various events which represents that some activity is performed by the user or by the browser. When javascript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called Event Handling.

**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.**

**AIM**:To write a Javascript with Objects, Creating Objects, Combining and cloning Objects using Spread operator, Destructuring Objects, Browser Object Model, Document Object Model.

**DESCRIPTION**:

A JavaScript object has properties associated with it.

A property of an object can be explained as a variable that is attached to the object. Object properties are basically the same as ordinary JavaScript variables, except for the attachment to objects. The properties of an object define the characteristics of the object. You access the properties of an object with a simple dotnotation. objectName.propertyName

**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>

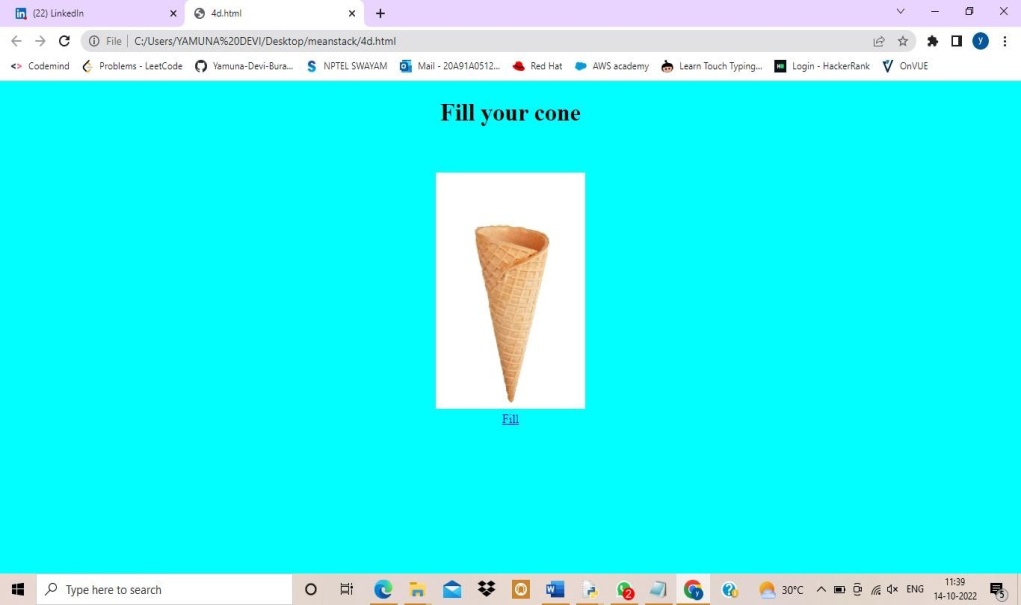
<a href="javascript:fun()" id="link">Eat</a>

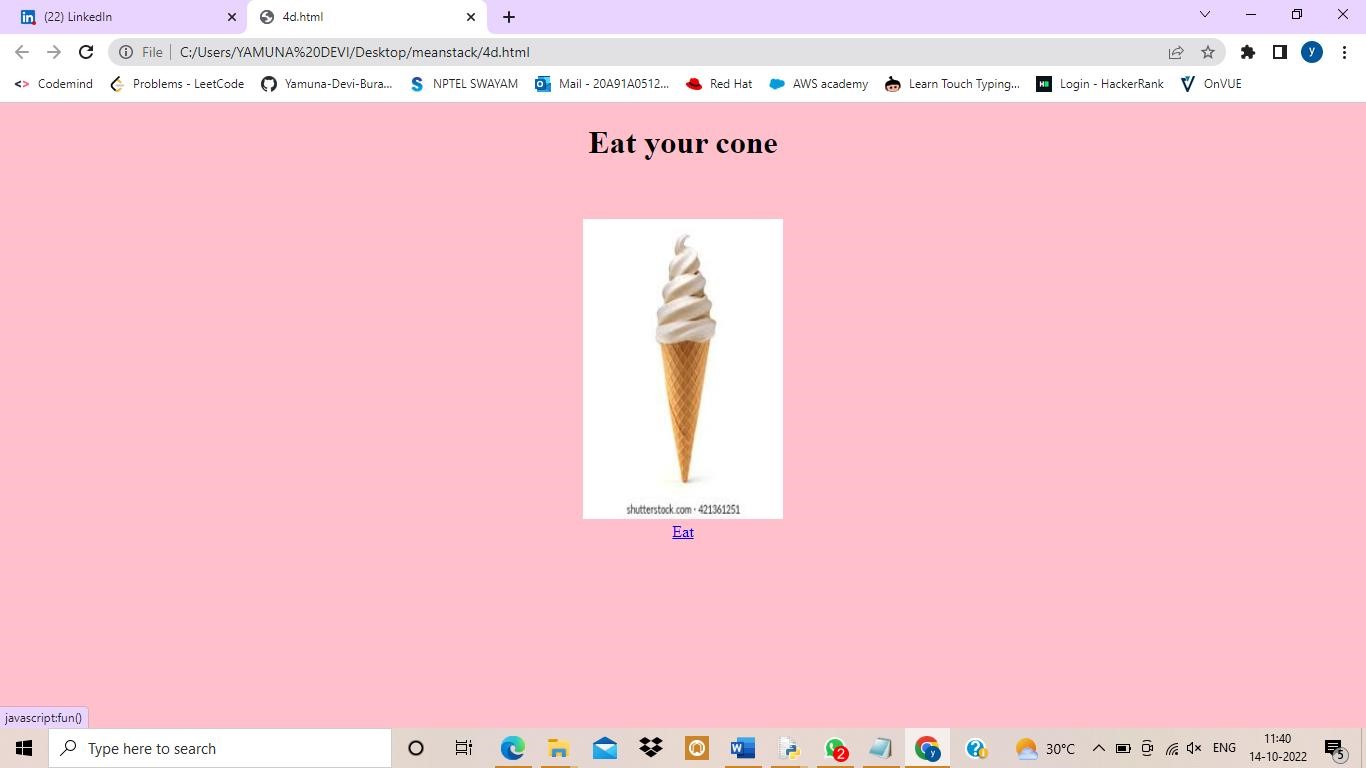
</center>

</body>

</html>

**OUTPUT:**





**Viva Questions:**

**1.How do you create an object in JavaScript?**

A.To create an object, use the **new** keyword with Object() constructor, like this: **const person = new Object();** Now, to add properties to this object, we have to do something like this: person.

**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.**

**AIM:** To create an array of object having movie details.

**DESCRIPTION:**

**Creating an array:-** Using an array literal is the easiest way to create a JavaScript Array.

**Syntax:**

const*array\_name* = [*item1*, *item2*, ...];

**Destructuring an Array:-**When destructuring in javascript, a syntax you would want to keep in mind would be placing the element you want to destructure on the right side of the assignment operator and placing the variable you want to access on the left side of the assignment operator. The variables should be in { } when destructuring objects and [ ] when destructuring arrays.

**const [var1, var2, ...] = arrayName;**

**Accessing Arrays:-**The items of an array are called elements. To access an array element, you have to write the array name, followed by square brackets and pass the index value of the element you want to access to the square brackets.

**Array Methods**

In JavaScript, there are various array methods available that makes it easier to perform useful calculations.

Some of the commonly used JavaScript array methods are: concat():-joins two or more arrays and returns a result indexOf():-searches an element of an array and returns its position find():-returns the first value of an array element that passes a test

findIndex():-returns the first index of an array element that passes a test

forEach():-calls a function for each element

includes():-checks if an array contains a specified element push():-aads a new element to the end of an array and returns the new length of an array

unshift():-adds a new element to the beginning of an array and returns the new length of an array

pop():-removes the last element of an array and returns the removed element

shift():-removes the first element of an array and returns the removed element

sort():-sorts the elements alphabetically in strings and in ascending order

slice():-selects the part of an array and returns the new array splice():-removes or replaces existing elements and/or adds new elements.

**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="Martian.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 = [ "The Martian",

"English",

"10",

"Matt Damon",

];

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:**



**Viva Questions**

**1. What is destructuring array in JavaScript?**

A.Destructuring the array in JavaScript simply means extracting multiple values from data stored in objects and arrays. The destructing assignment syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.

**2. What are the 10 JavaScript array methods?**

A.Important JavaScript Array Methods: map(), forEach(), reduce(), filter(), sort(), includes(), find(), indexOf(), some(), concat() Array methods are indispensable in JavaScript and there are numerous array methods.

**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.**

**AIM:**To stimulate a periodic stock price change and display on the console.

**DESCRIPTION:**

To use the random function

**Syntax: Math.random()**

To use the setInterval function

**Syntax:**

myInterval = setInterval(function, milliseconds);

To stop the execution of setInterval function

**Syntax:**clearInterval(myInterval);

To create a Promise we have to use to following Syntax

**Syntax:**

let myPromise = new Promise(function(Resolve, Reject) {

Resolve(); // when successful

Reject(); // when error

});

**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);

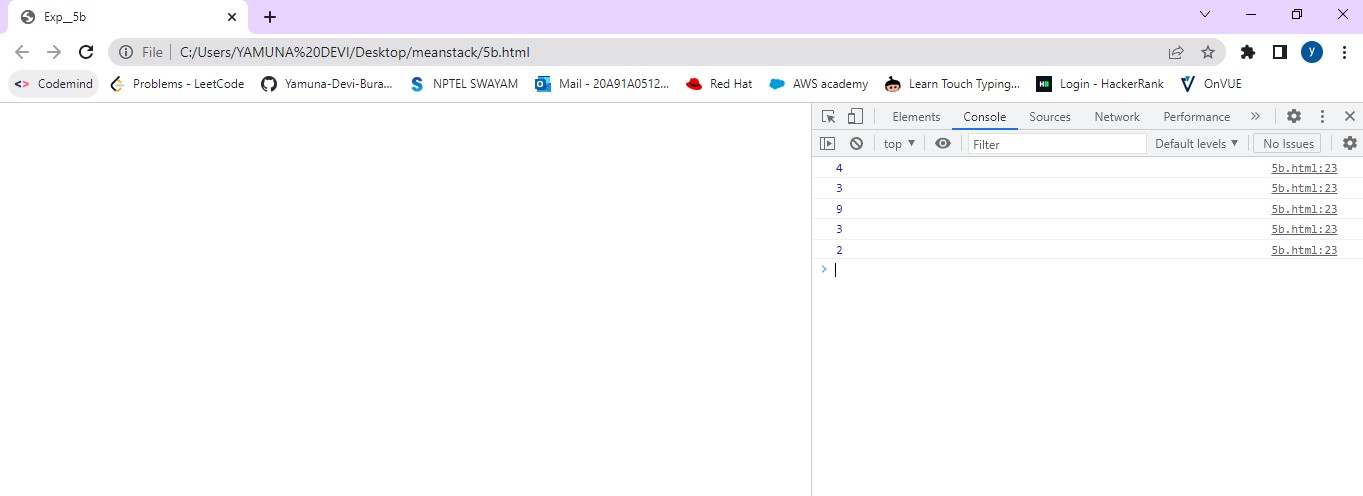
}

</script>

</body>

</html>

**Output:**



**Viva Questions**

**1.What is callback function in Javascript?**

A.A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.

**2.What is asynchronous programming?**

A. Asynchronous programming is a technique that enables your program to start a potentially long-running task and still be able to be responsive to other events while that task runs, rather than having to wait until that task has finished.

**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.**

**AIM:** Creating Modules, Consuming Modules Validate the user by creating a login module.

**DESCRIPTION:**

**Modules**

JavaScript modules allow you to break up your code into separate files.

This makes it easier to maintain a code-base.

Modules are imported from external files with the import statement.

Modules also rely on type="module" in the <script> tag.

A module in JavaScript is just a file containing related code.

In JavaScript, we use the import and export keywords to share and receive functionalities respectively across different modules.

The export keyword is used to make a variable, function, class or object accessible to other modules. In other words, it becomes a public code.

The import keyword is used to bring in public code from another module.

**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)

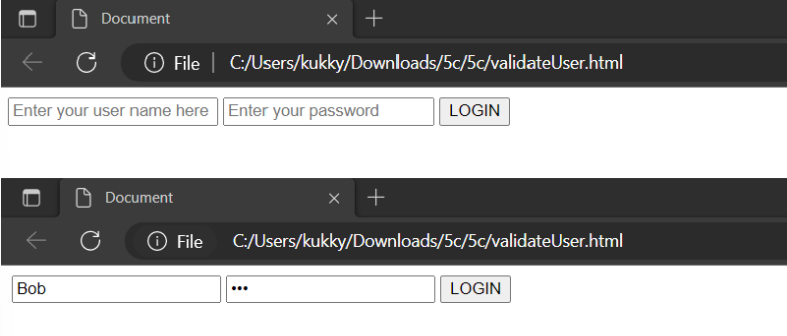
{

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

}

}

**OUTPUT:**



**Viva Questions**

**1.** **What is the purpose of modules in JavaScript?**

A.JavaScript modules allow you to break up your code into separate files. This makes it easier to maintain a code-base. Modules are imported from external files with the import statement.

**2.What are the two types of module exports in JavaScript?**

A.Every module can have two different types of export, named export and default export.

**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.**

**AIM**: Learning about use of Node.js and verifying how to execute different functions successfully in the Node.js platform.

**DESCRIPTION:**

Node.js is an open source server environment.Node.js allows you to run JavaScript on the server. Create a JavaScript file. Execute the JavaScript file using node.js. Let us create our first Node.js program.

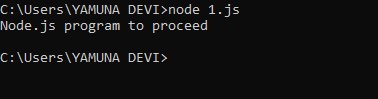
**Step 1**: Create a folder NodeJS in C drive and create a new JavaScript file, 1.js inside the folder. Type the below code inside the JavaScript file.

**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:**



**Viva Questions**

**1.** **What is NPM?**

A.NPM stands for Node Package Manager, responsible for managing all the packages and modules for Node.js.

**2) Is Node.js free to use?**

A.Yes. It is released under MIT license and is free to use.

**3) What is the purpose of Node.js?**

A.These are the following purposes of Node.js:

Real-time web applications

Network applications

Distributed systems

General purpose applications

**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.**

**AIM:** Creating a web server in Node.js and showing the workflow of JavaScript code executable by creating web server in Node.js.

**DESCRIPTION:**

Using require() and createServer() method Running a web server in Node.js

**Step 1**: Create a new JavaScript file httpserver.js and include the

HTTP module.

**Step 2**: Use the createServer() method of the HTTP module to create a web server.

**Step 3**: Save the file and start the server using the node command. When the file executes successfully, we can observe the following output in the console.

**Step 4:** We will observe the output in the browser.

**PROGRAM**

var http = require('http');

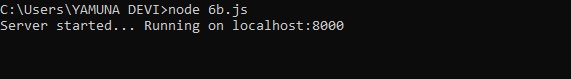
http.createServer(function (req, res) { res.writeHead(200, {'Content-Type': 'text/html'}); res.end(‘Server is connected**.**');

}).listen(8000);

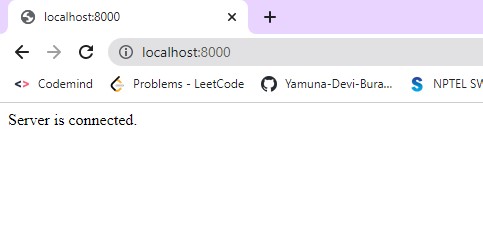
console.log("Server started... Running on localhost:8000");

**OUTPUT:**

Output on the command prompt –



Output on the browser:



**Viva Questions**

**1.** **which module is used to create a web server in node.js?**

A. The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.

**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**.

**AIM:** Write a Node.js module to show the workflow of Modularization of Node application .

**DESCRIPTION**: Modularization is a software design technique in which the functionality of a program is separated into independent modules, such that each module contains the desired functionality.

**Readability:** Modular code highly organizes the program based on its functionality. This allows the developers to understand what each piece of code does in the application.

**Easier to debug**: When debugging large programs, it is difficult to detect bugs. If a program is modular, then each module is discrete, so each module can be debugged easily by the programmer.

**Reusable Code**: Modular code allows programmers to easily reuse code to implement the same functionality in a different program. If the code is not organized modularly into discrete parts, then code reusability is not possible.

**Reliability**: Modular code will be easier to read. Hence it will be easier to debug and maintain the code which ensures smoother execution with minimum errors.

**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>");

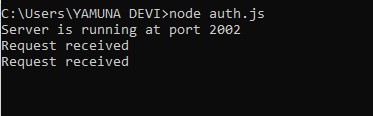
console.log("Request received");

});

server.listen(2002);

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

# Output on the command prompt



# Output in the browser



**Viva Questions**

**1.What is Modular programming?**

**A.** In Node. js, modular systems are implemented natively, based on the common JavaScript specifications. This way, you implement a local namespace within every module.

**2.What is the command used to import external libraries?**

**A.** The “require” command is used for importing external libraries.

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

**AIM:**Program to show the workflow of restarting a node application.

**DESCRIPTION:**

Whenever we are working on a Node.js application and we do any change in code after the application is started, we will be required to restart the Node process for changes to reflect. In order to restart the server and to watch for any code changes automatically, we can use the Nodemon tool.

**Nodemon**

Nodemon is a command-line utility that can be executed from the terminal. It provides a different way to start a Node.js application. It watches the application and whenever any change is detected, it restarts the application. It is very easy to get started with this tool.

To install it in the application, run the below command.

**npm install nodemon –g**

Once the 'nodemon' is installed in the machine, the Node.js server code can be executed by replacing the command "node" with "nodemon".

**PROGRAM**

const http = require("http");

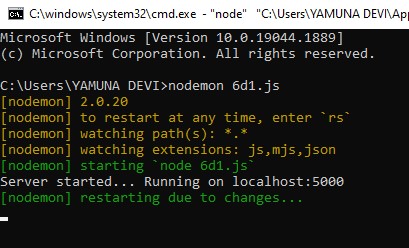
var server = http.createServer((req, res) => { res.write("Hello ! I have created my second server!"); res.end();

});

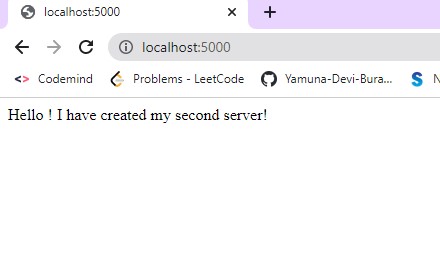
server.listen(5000);

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

**Output in command prompt:**



**Output in the browser:**



**Modified code in the file nodemon1.js**

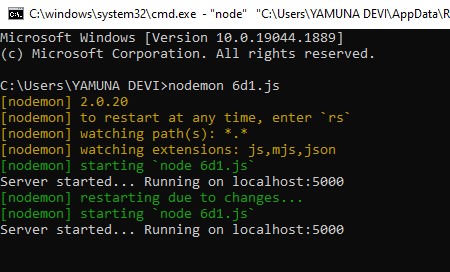
const http = require("http");

var server = http.createServer((req, res) => { res.writeHead(200,{'Content-Type': 'text/html'}); res.write("Hello ! I have created my second server!"); res.end(); });

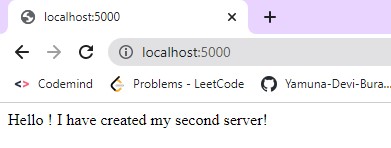
server.listen(5000);

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

**Output in command prompt:**



**Output in the browser**



**Viva Questions**

**1.** **what is nodemon used for in a node.js application?**

**A.** Nodemon is a command-line tool that helps with the speedy development of Node. js applications. It monitors your project directory and automatically restarts your node application when it detects any changes.

**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.**

**AIM:**

To create a text file src.txt and add the following data to it.

**DESCRIPTION:**

**Some of the file operations that we will be discussing are:**

1. Writing data to a file
2. Reading data from a file
3. Updating content in a file

The File System module has the following methods for creating a new file and writing data to that file:

- writeFile()

- appendFile()

**WriteFile:**

The fs.writeFile() method will overwrite the content if the content already exists.

If the file does not exist, then a new file will be created with the specified name and content.

**Syntax:**

fs.writeFile(file, data, callback);

1. file: Placeholder to give the file name in which you are going to write the data.
2. data: The data/content must be written to the file.
3. callback: The callback method, that will be executed, when 'writeFile()' function is executed. This callback will be executed in both success as well as failure scenarios.

**AppendFile:**

The appendFile() method first checks if the file exists or not. If the file does not exist, then it

creates a new file with the content, else it appends the given content to the existing file.

**Syntax:**

fs.appendFile(path, data, callback)

1. path: Placeholder to give the file name in which you are going to append the data.
2. data: The data/content which must be appended to the file.
3. callback: The callback method, that will be executed, when 'appendFile()' function is executed. This callback will be executed in both success as well as failure scenarios.

**PROGRAM**

const fs = require('fs');

const src = "source.txt";

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

// incase of any error if (error)

{

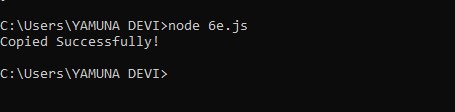
console.error(error); return;

}

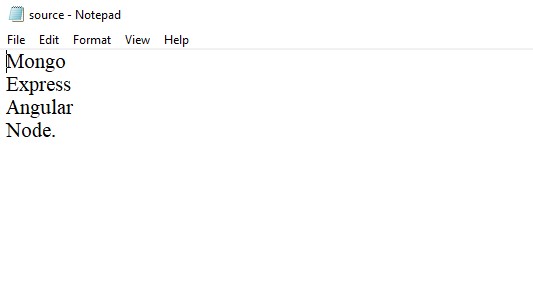
console.log("Copied Successfully!");

});

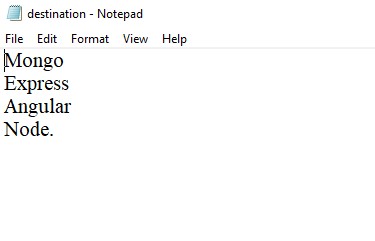
**Output in command prompt:**



**Source file:source.txt**



**Destination.txt**



**Viva Questions**

1. **what are the basic file operations in nodejs?**

A. Common use for the File System module:

Read files

Create files

Update files

Delete files

Rename files

**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.**

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

**DESCRIPTION:**

**Routing:** The application object has different methods corresponding to each of the HTTP verbs (GET, POST, PUT, DELETE). These methods are used to receive HTTP requests.**Syntax: router.method(path,handler) router:** express instance or router instance **method:** one of the HTTP verbs**path:** is the route where request runs

**handler:** is the callback function that gets triggered whenever a request comes to a particular path for a matching request type .

**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');

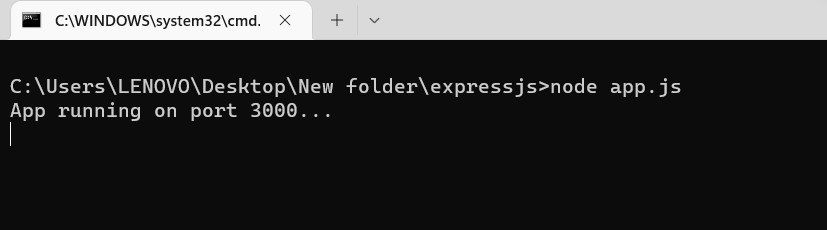
const app = express(); app.use('/', route);

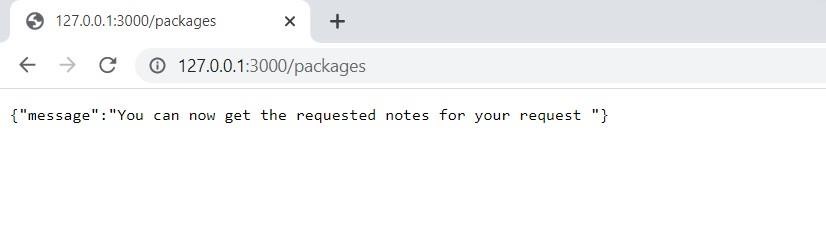
const port = process.env.PORT || 3000; app.listen(port, () => {

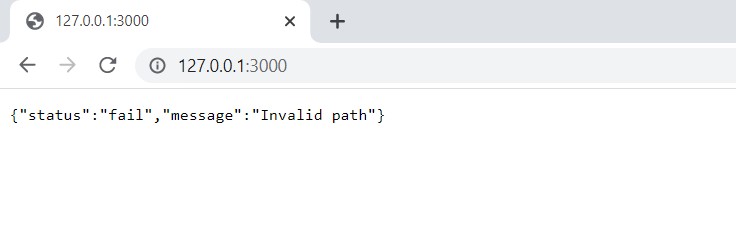
console.log(`App running on port ${port}...`);

});

**OUTPUT:**







**Viva Questions**

**1. Is Express.js front-end or backend framework?**

A.Express.js or Express is a JavaScript backend framework.

**2.what is routing in express.js?**

A.Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).

**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.**

**AIM:** 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.

**DESCRIPTION:**

A middleware can be defined as a function for implementing different cros-cutting concerns such as authentication, logging, etc.

Themain arguments of a middleware function are the **request** object, **response** object, and also the **next** middleware function defined in the application.

A function defined as a middleware can execute any task mentioned below:

* Any code execution.
* Modification of objects - request and response.
* Call the next middleware function.
* End the cycle of request and response.

**Example to Define a MiddleWare.**

**const mylogger = async (req, res, next) => {**

**console.log(new Date(), req.method, req.url); next();**

**};**

**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) {

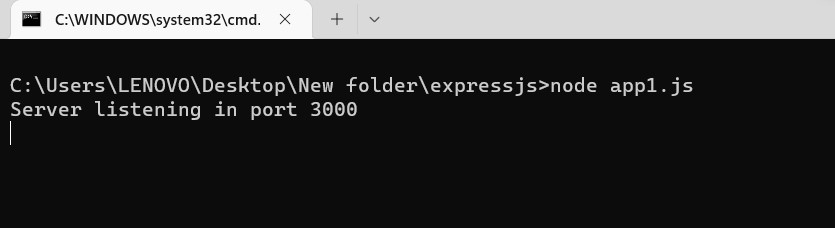
console.log(`Req method is ${req.method}`); console.log(`Req url is ${req.url}`); next();

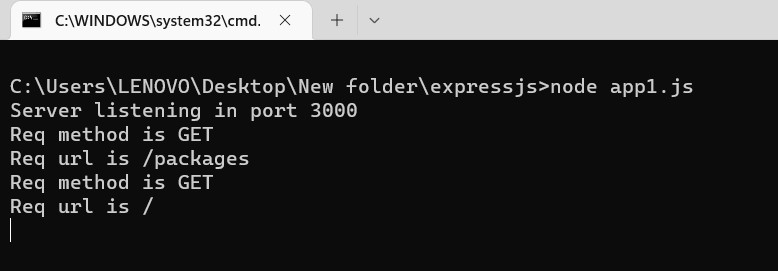
};

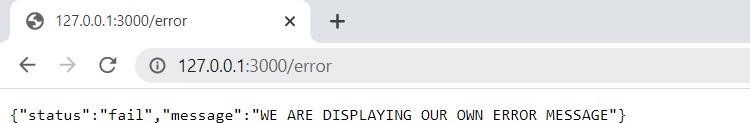
app.use(mylogger); app.use('/', router); app.listen(3000);

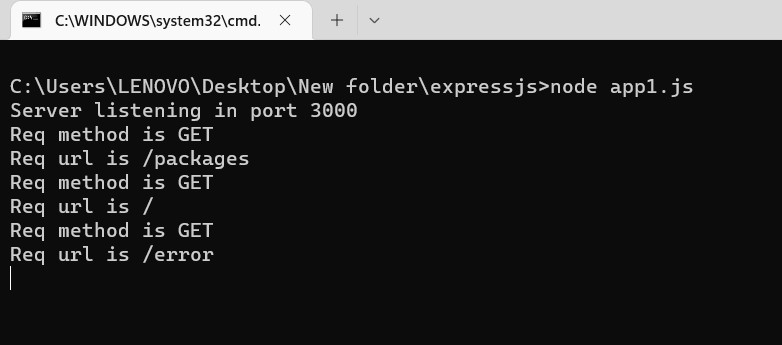
console.log('Server listening in port 3000');

**Output:**









**Viva Questions**

**1.** **How many types of middleware are there in Express js?**

A.An Express application can use the following types of middleware: Application-level middleware. Router-level middleware. Error-handling middleware.

**2.** **Is app use () a middleware in Express?**

A.The app. use() method mounts or puts the specified middleware functions at the specified path.

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

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

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

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

**DESCRIPTION:**

Before we get into the specifics of validation syntax, please keep the following rules in mind:

•Validation is defined in the SchemaType

•Validation is middleware. Mongoose registers validation as a pre('save') hook on every schema by default.

•You can disable automatic validation before save by setting the validateBeforeSave option

•You can manually run validation using doc.validate(callback) or doc.validateSync()

•You can manually mark a field as invalid (causing validation to fail) by using doc.invalidate(...)

•Validators are not run on undefined values. The only exception is the required validator.

**PROGRAM:**

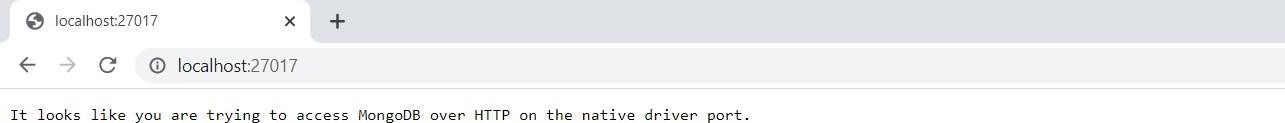
const express=require('express') const mongoose=require('mongoose') const app=express()

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

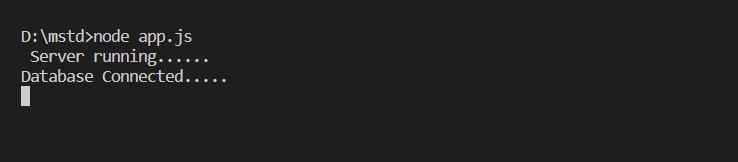
=majority";

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:**



**Viva Questions**

**1. What is a schema in mongoose?**

A. Mongoose Schema defines the structure and property of the document in the MongoDB collection. This Schema is a way to define expected properties and values along with the constraints and indexes.

**2.** **What is the use of Mongoose in MongoDB?**

A.Mongoose is a Node. js-based Object Data Modeling (ODM) library for MongoDB. It is akin to an Object Relational Mapper (ORM) such as SQLAlchemy for traditional SQL databases. The problem that Mongoose aims to solve is allowing developers to enforce a specific schema at the application layer.

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

**Module Name: Models**

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

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

**DESCRIPTION**: Schema wrapping (@graphql-tools/wrap) creates a modified version of a schema that proxies, or "wraps", the original unmodified schema. This technique is particularly useful when the original schema cannot be changed, such as with remote schemas.

Schema wrapping works by creating a new "gateway" schema that simply delegates all operations to the original subschema. A series of transforms are applied that may modify the shape of the gateway schema and all proxied operations; these operational transforms may modify an operation prior to delegation, or modify the subschema result prior to its return.

**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:**



**Viva Questions**

**1. What is model in Mongoose?**

A Mongoose model is a wrapper of the Mongoose schema.

**2.** **What is the difference between model and schema?**

A.The database schema is one that contains list of attributes and instructions to tell the database engine how data is organised whereas Data model is a collection of conceptional tools for describing data, data-relationship and consistency constraints.

**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.**

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

**DESCRIPTION:**

**CRUD OPERATIONS**

**Create:** We’ll be setting up a post request to ‘/save’ and we’ll create a new student object with our model and pass with it the request data from Postman.Once this is done, we will use .save() to save it to the database.

**Retrieve:** To retrieve records from a database collection we make use of the .find() function.

**Update:** Just like with the delete request, we’ll be using the \_id to target the correct item.

.findByIdAndUpdate() takes the target’s id, and the request data you want to replace it with.

**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

});

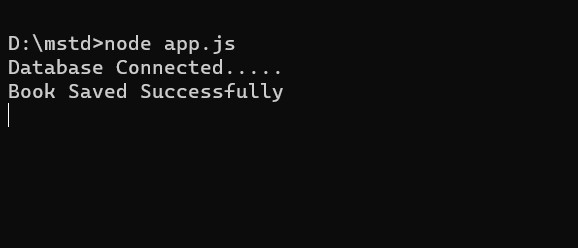
book1.save(function(err){

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

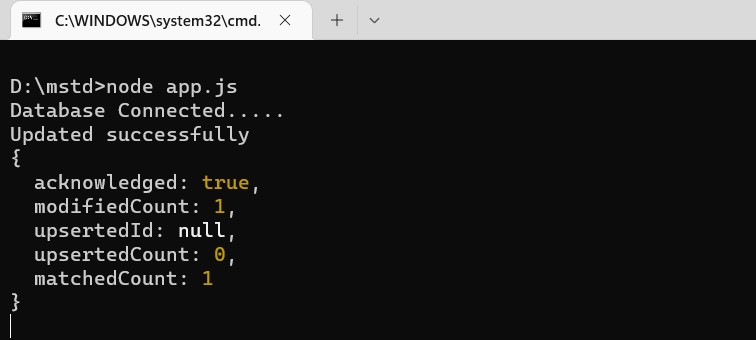
});

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);

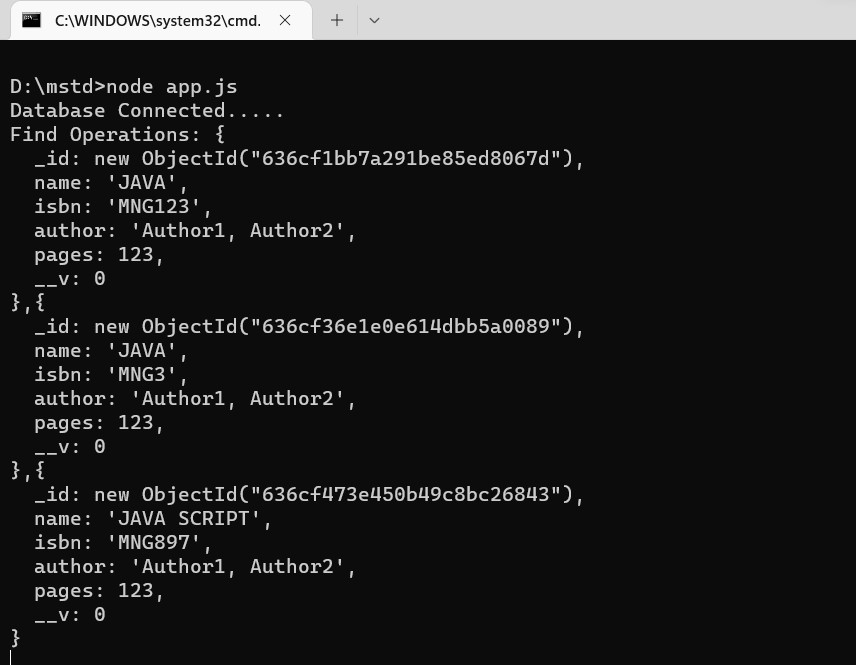
var db = mongoose.connection; var deleteBook = function(){

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

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

deleteBook();

**Afterdeleting records Database is:**



**Viva Questions**

**1.** **Can you explain how to apply CRUD operations in MongoDB?**

A.CRUD stands for Create, Read, Update, and Delete. In MongoDB, these operations can be performed on documents in a collection. To create a document, you can use the insert() or save() method. To read a document, you can use the find() method. To update a document, you can use the update() method. To delete a document, you can use the remove() method.

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

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

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

**AIM:** 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) {

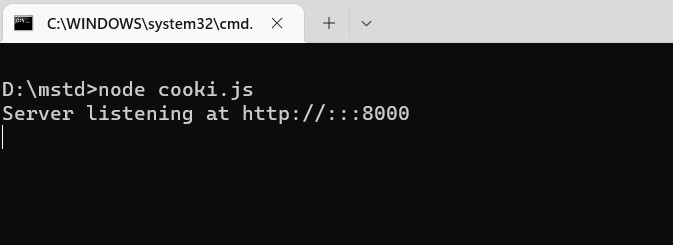
res.status(200).send('Welcome to Aditya');

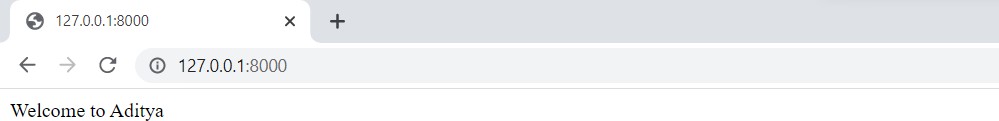
}); var server = app.listen(8000, function () {

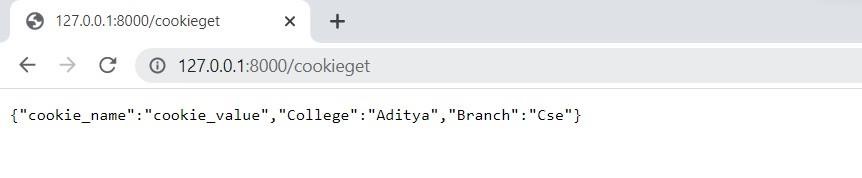
var host = server.address().address; var port = server.address().port; console.log('Server listening at http://%s:%s', host, port);

});

**Output:**







**Viva Questions**

1. **What is the importance of Cookies in express js?**
2. Cookies are simple, small files/data that are sent to client with a server request and stored on the client side. Every time the user loads the website back, this cookie is sent with the request. This helps us keep track of the user's actions.

**2.Which middleware used for creating cookies?**

A.Now to use cookies with Express, we will require the cookie-parser. cookie-parser is a middleware which parses cookies attached to the client request object. To use it, we will require it in our index. js file; this can be used the same way as we use other middleware.

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

**Module Name: Sessions**

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

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

**DESCRIPTION:**

A website is based on the HTTP protocol. HTTP is a stateless protocol which means at the end of every request and response cycle, the client and the server forget about each other.

This is where the session comes in. A session will contain some unique data about that client to allow the server to keep track of the user’s state. In session-based authentication, the user’s state is stored in the server’s memory or a database.

The following libraries will help us setup a Node.js session.

Express - a web framework for Node.js used to create HTTP web servers. Express provides an easy-to-use API to interact with the webserver.

Express-session - an HTTP server-side framework used to create and manage a session middleware. This tutorial is all about sessions. Thus Express-session library will be the main focus.

Cookie-parser - used to parse cookie header to store data on the browser whenever a session is established on the server-side.

**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)

})

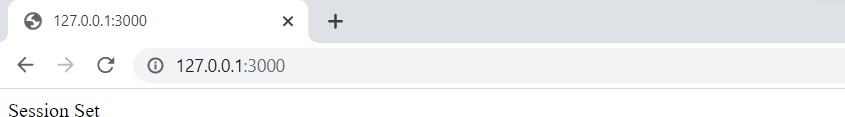
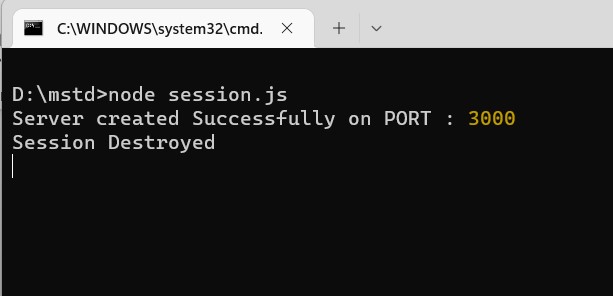
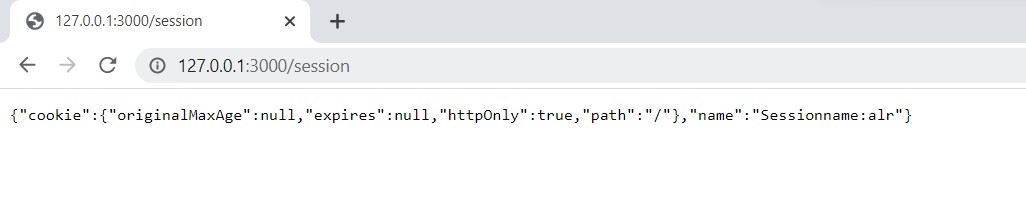
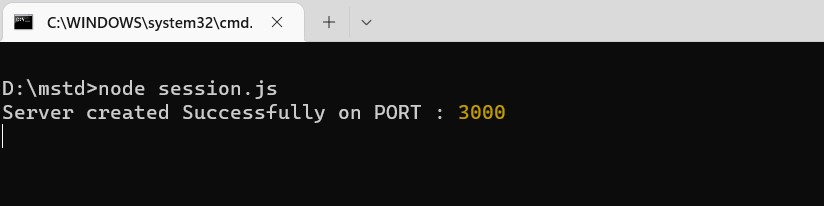
app.listen(PORT, function(error)

{ if(error) throw error

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

})

**OUTPUT:**



**Viva Questions**

**1. What is session management with cookies?**

A.The cookie allows the server to identify the user and retrieve the user session from the session database, so that the user session is maintained.

**2.What is an example of a session cookie?**

A.The most common example of a session cookie is the shopping cart that nearly all e-commerce websites use.

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

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

**AIM :**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>

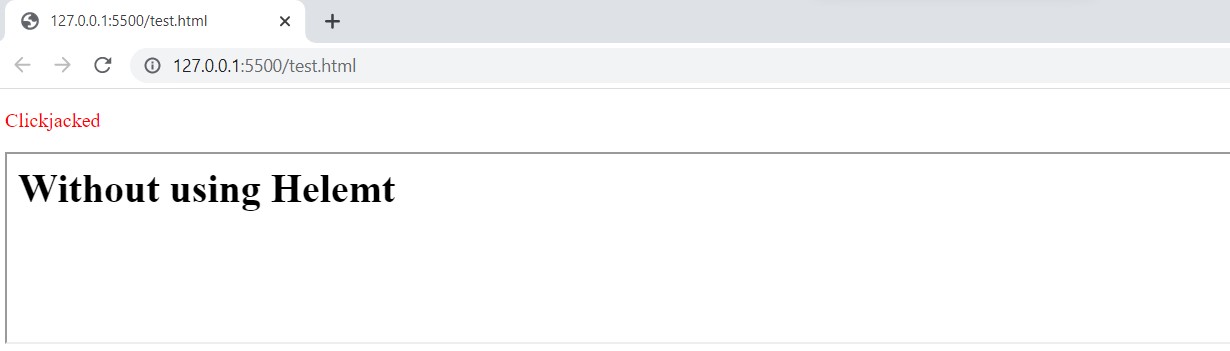
<body>

<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:**



**Viva Questions**

**1.** **What is helmet middleware?**

A.Helmet. js is an open source JavaScript library that helps you secure your Node. js application by setting several HTTP headers.

**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**

**AIM :** 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.

**DESCRIPTION:**

TypeScript is a syntactic superset of JavaScript which adds static typing.

This basically means that TypeScript adds syntax on top of JavaScript, allowing developers to add types.

TypeScript lets you write JavaScript the way you really want to. TypeScript is a typed superset of JavaScript that compiles to plain JavaScript. TypeScript is pure object oriented with classes, interfaces and statically typed like C# or Java. The popular JavaScript framework Angular 2.0 is written in TypeScript. Mastering TypeScript can help programmers to write object-oriented programs and have them compiled to JavaScript, both on server side and client side.

**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:**



**Viva Questions**

**1. How can we get TypeScript and install it?**

**A.** TypeScript can be installed and managed with the help of node via npm (the Node.js package manager). To install TypeScript, first ensure that the npm is installed correctly, then run the following command which installs TypeScript globally on the system.

$ npm install -g typescript

**2.** **How to compile a Typescript file?**

A.Here is the command which is followed while compiling a Typescript file into JavaScript.

$ tsc <TypeScript File Name>

**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.**

**AIM:**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:**

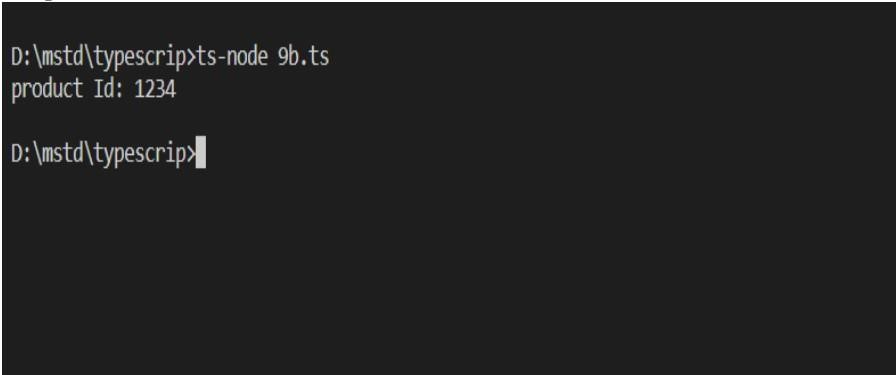
var getproductdetails=(productId : number):string=>{

return "product Id:"+productid

};

console.log(getproductdetails(1234));

**Output:**



**Viva Questions**

**1.What is the syntax for creating an arrow function?**

A.Users can follow the syntax below to create an arrow function in TypeScript.

var variable = (param1: type, ...other params): return\_type => {

// code for the function

};

**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.**

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

**DESCRIPTION:**

Functions are the basic building block of any application which holds some business logic. The process of creating a function in TypeScript is similar to the process in JavaScript.

In functions, parameters are the values or arguments that passed to a function. The TypeScript, compiler accepts the same number and type of arguments as defined in the function signature. If the compiler does not match the same parameter as in the function signature, then it will give the compilation error.

**We can categorize the function parameter into the three types:**

* Optional Parameter
* Default Parameter
* Rest Parameter

Function return type in TypeScript is nothing but the value which we want to return from the function. Function return type used when we return value from the function. We can return any type of value from the function or nothing at all from the function in TypeScript. Some of the return types is a string, number, object or any, etc. If we do not return the expected value from the function, then we will have an error and exception. In the coming section, we will discuss the internal working and how to implement different return types or different functions in detail.

**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'];

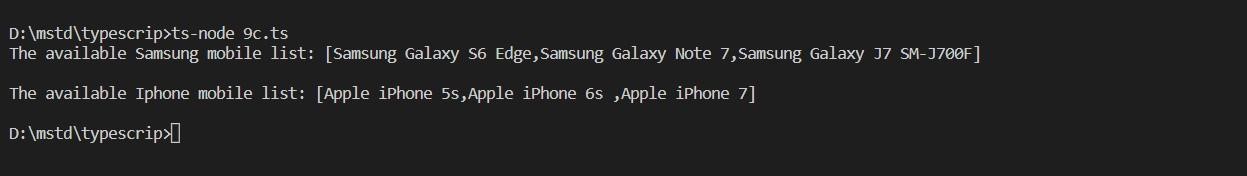
return mobileList;

} }

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

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

**OUTPUT:**



**Viva Questions**

**1.Is return type required in TypeScript?**

**A.**Functions in TypeScript often don't need to be given an explicit return type annotation.

**2.What is the parameter type?**

**A.** a parameter is a type, and an argument is an instance. A parameter is an intrinsic property of the procedure, included in its definition.

**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.**

**AIM:**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.

**DESCRIPTION:**

**Arrow Function:**

Fat arrow notations are used for anonymous functions i.e for function expressions. They are also called lambda functions in other languages.

**Syntax:**

(param1, param2, ..., paramN) => expression

Using fat arrow =>, we dropped the need to use the function keyword. Parameters are passed in the parenthesis (), and the function expression is enclosed within the curly brackets { }.

**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) =>

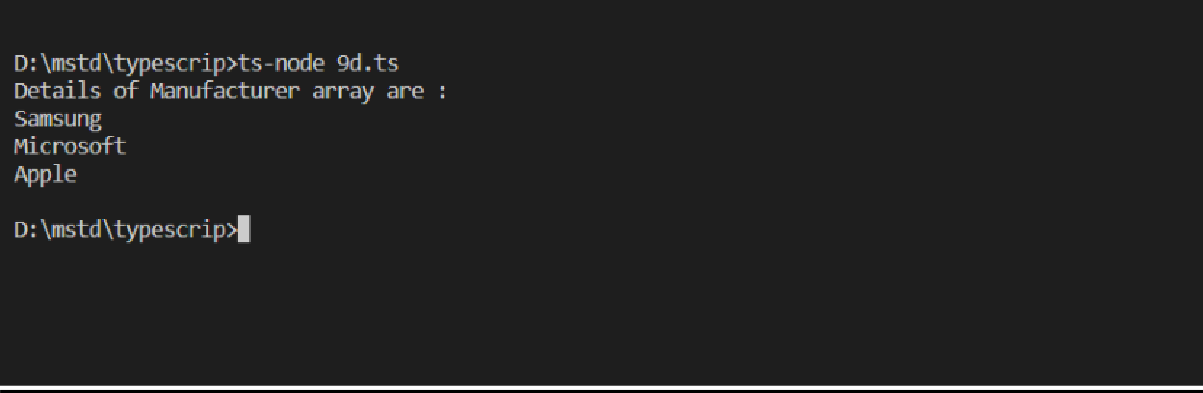
m.price >= 150);

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

}

} myFunction();

**OUTPUT:**



**Viva Questions**

**1.** **What is the difference between arrow function and normal function in TypeScript?**

A.Since regular functions are constructible, they can be called using the new keyword. However, the arrow functions are only callable and not constructible, i.e arrow functions can never be used as constructor functions.

**2.** **Is arrow function faster than function?**

A.It's not related to the difference in speed of execution - as you've said, that's more or less identical.

**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**

**AIM:**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.

**DESCRIPTION:**

**Optional Parameters**

In TypeScript, every parameter is assumed to be required by the function. You can add a ? at the end of a parameter name to set it as optional.

**For example**, the lastName parameter of this function is optional:

function buildName(firstName: string, lastName?: string) {

// ...

}

Optional parameters must come after all non-optional parameters:

function buildName(firstName?: string, lastName: string) // Invalid

**Default Parameters**

If the user passes undefined or doesn't specify an argument, the default value will be assigned. These are called default-initialized parameters.

**For example,** "Smith" is the default value for the lastName parameter.

function buildName(firstName: string, lastName = "Smith") {

// ...

}

buildName('foo', 'bar'); // firstName == 'foo', lastName == 'bar'

buildName('foo'); // firstName == 'foo', lastName == 'Smith'

buildName('foo', undefined); // firstName == 'foo', lastName == 'Smith'

**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 {

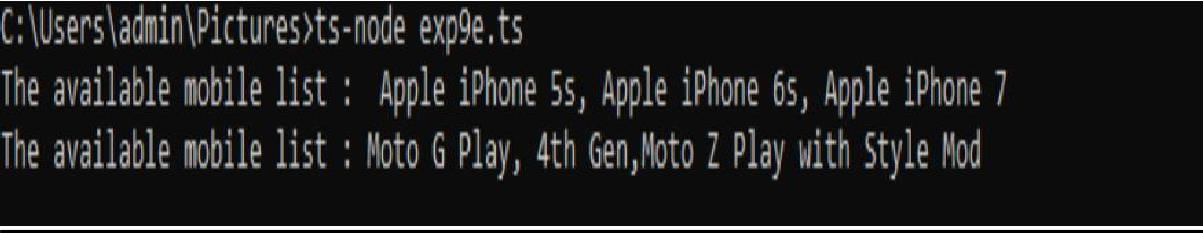
mobileList = [' Nokia 105', ' Nokia 230 Dual Sim'];

return mobileList;}}

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

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

**OUTPUT:**



**Viva Questions**

**1. What is the difference between default parameter and optional parameter TypeScript?**

A.The difference between calling a function and omitting an optional parameter vs a default parameter is: If an optional parameter has no value, the parameter's value is undefined. If a default parameter has no value, the parameter's value is the default value

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

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

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

**DESCRIPTION:**

**TypeScript - Rest Parameters**

In the function chapter, you learned about functions and their parameters. TypeScript introduced rest parameters to accommodate n number of parameters easily.

When the number of parameters that a function will receive is not known or can vary, we can use rest parameters. In JavaScript, this is achieved with the "arguments" variable. However, with TypeScript, we can use the rest parameter denoted by ellipsis ....

We can pass zero or more arguments to the rest parameter. The compiler will create an array of arguments with the rest parameter name provided by us.

**PROGRAM:**

const cart: string[] = [];

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

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

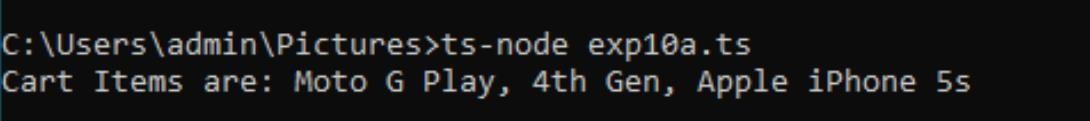
}

Returncart;

}

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

**OUTPUT:**



**Viva Questions**

**1. What is rest parameter in TypeScript?**

The rest parameter syntax allows a function to accept an indefinite number of arguments as an array, providing a way to represent variadic functions in JavaScript.

**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.**

**AIM:**Declare an interface named - Product with two properties like productId and productName with a number and string data type and need to implement logic to populate the Product details.

**DESCRIPTION:**

An interface is a syntactical contract that an entity should conform to. In other words, an interface defines the syntax that any entity must adhere to.

Interfaces define properties, methods, and events, which are the members of the interface. Interfaces contain only the declaration of the members. It is the responsibility of the deriving class to define the members. It often helps in providing a standard structure that the deriving classes would follow.

Let’s consider an object −

var person = {

FirstName:"Tom",

LastName:"Hanks",

sayHi: ()=>{ return "Hi"}

};

If we consider the signature of the object, it could be −

{

FirstName:string,

LastName:string,

sayHi()=>string

}

To reuse the signature across objects we can define it as an interface.

**Declaring Interfaces**

The interface keyword is used to declare an interface. Here is the syntax to declare an interface −

**Syntax**

interface interface\_name {

}

**PROGRAM:**

interface Product { productId: number ; productName: string ;

}

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

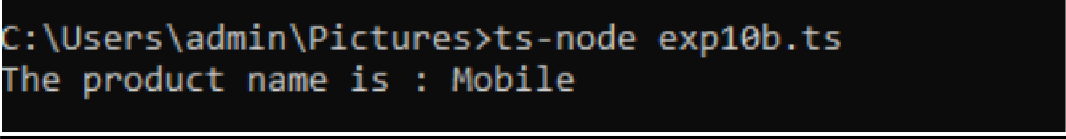
}

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

const productDetails: string = getProductDetails(prodObject);

console.log(productDetails);

**OUTPUT:**



**Viva Questions**

**1.What is a interface in TypeScript?**

A.An interface is a syntactical contract that an entity should conform to.

**2.What are the two types in interface TypeScript?**

A.TypeScript has two special types, null and undefined , that have the values null and undefined respectively.

**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.**

**AIM:**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.

**DESCRIPTION:**

**Duck Typing:**

Duck-Typing is a method/rule used by TypeScript to check type compatibility for more complex variable types. This method is used to compare two objects by determining whether they have the same type of matching names or not. It means we can't change a variable's signature.

The duck-typing technique in TypeScript is used to compare two objects by determining if they have the same type matching properties and objects members or not. For example, if we assign an object with two properties and a method and the second object is only assigned with two properties. The typescript compiler raises a compile-time error in such situations when we create a variable of object1 and assign it a variable of the second object type.

**PROGRAM:**

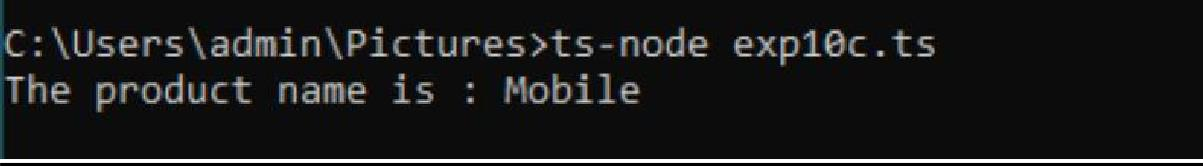
interface Product { productId: number; productName:string;

}

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:**

**Viva Questions**

**1. What is duck typing in TypeScript?**

A.Duck-Typing is a method/rule used by TypeScript to check type compatibility for more complex variable types. This method is used to compare two objects by determining whether they have the same type of matching names or not. It means we can't change a variable's signature.

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

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

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

**DESCRIPTION:**

A function type has two parts: parameters and return type. When declaring a function type, you need to specify both parts with the following syntax:

(parameter: type, parameter:type,...) => type

Code language: PHP (php)

The following example shows how to declare a variable which has a function type that accepts two numbers and returns a number:

let add: (x: number, y: number) => number;

Code language: JavaScript (javascript)

In this example:

The function type accepts two arguments: x and y with the type number.

The type of the return value is number that follows the fat arrow (=>) appeared between parameters and return type.

**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:**



**Viva Questions**

**1.what is function?**

A. Functions are the fundamental building block of any application in JavaScript. They're how you build up layers of abstraction, mimicking classes, information hiding, and modules. In TypeScript, while there are classes, namespaces, and modules, functions still play the key role in describing how to do things.

**2.How many types of functions are there in TypeScript?**

A.In TypeScript, functions can be of two types: named and anonymous.

**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.**

**AIM:**To declare a productList interface which extends properties from two other declared interfaces like Category.

**DESCRIPTION:** An interface can be extended from an already existing one using the extends keyword. In the code below, extend the productList interface from both the Category interface and Product interface.

**Example:**

interface Category

{

categoryName:string;

}

interface Product

{

productName:string; productid:number;

}

interface productList extends Category,Product

{

list:[‘Samsung’,’Motorola’,’LG’ ]

}

**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']

};

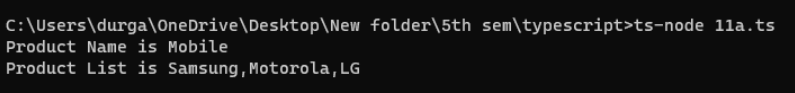
const listProduct = productDetails.list;

const pname: string = productDetails.productName;

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

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

**OUTPUT:**

****

**Viva Questions**

**1.How do you extend an interface?**

A.An interface can extend another interface in the same way that a class can extend another class. The **extends** keyword is used to extend an interface, and the child interface inherits the methods of the parent interface.

**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.**

**AIM:**To consider the Mobile Cart application, Create objects of the Product class and place them into the productlist array.

**DESCRIPTION:** TypeScript is object oriented JavaScript. TypeScript supports object-oriented programming features like classes, interfaces, etc. A class in terms of OOP is a blueprint for creating objects. A class encapsulates data for the object. Typescript gives built in support for this concept called class. JavaScript ES5 or earlier didn’t support classes. Typescript gets this feature from ES6. Use the class keyword to declare a class in TypeScript.

The syntax for the same is given below –

class class\_name

{

//class scope

}

**PROGRAM:**

class Product

{

static productPrice: string;

productId: number;

constructor()

{

this.productId =1234;

}

getProductId(): string

{

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

}

}

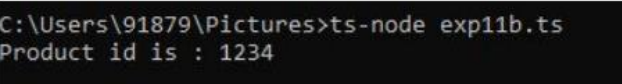
const product: Product = new Product();

const p={

producti :product.getProductId(), };

console.log(p.producti);

**OUTPUT:**

****

**Viva Questions**

**1.Is a class a type in TypeScript?**

A.TypeScript treats a class as both value and type. This implicit type declared by TypeScript describes the shape of the instance a class produces.

**2.What are the functions of class in TypeScript?**

A.A class in TypeScript can hold any number of properties, functions, “getters” and “setters”.

**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 <>"**

**AIM:**To 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 <>" .

**DESCRIPTION:** A constructor is a special function of the class that is automatically invoked when we create an instance of the class in Typescript. We use it to initialize the properties of the current instance of the class. Using the constructor parameter properties or Parameter shorthand syntax, we can add new properties to the class. We can also create multiple constructors using the technique of constructor method overload. The constructor method in a class must have the name constructor. A class can have only one implementation of the constructor method. The constructor method is invoked every time we create an instance from the class using the new operator. It always returns the newly created object.

**PROGRAM:**

class Product

{

static productPrice: string;

productId: number;

constructor(productId: number)

{

this.productId = productId;

}

getProductId(): string

{

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

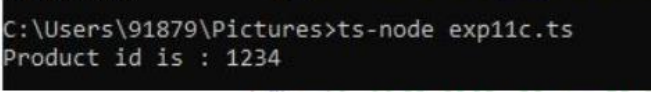
}

}

const product: Product = new Product(1234);

console.log(product.getProductId());

**OUTPUT:**

****

**Viva Questions**

**1. How to create class constructor TypeScript?**

A.To create a class constructor in TypeScript, you can use the keyword constructor inside the class body followed by the () symbol (parameters closing and opening brackets) and then write the {} symbol.

**2.** **Why do we write a constructor?**

A.There are the following reasons to use constructors: We use constructors to initialize the object with the default or initial state.

**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.**

**AIM:**To 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.

**DESCRIPTION:**Like other programming languages, Typescript allows us to use access modifiers at the class level. It gives direct access control to the class member. These class members are functions and properties. We can use class members inside its own class, anywhere outside the class, or within its child or derived class.The access modifier increases the security of the class members and prevents them from invalid use. We can also use it to control the visibility of data members of a class. If the class does not have to be set any access modifier, TypeScript automatically sets public access modifier to all class members.

**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');

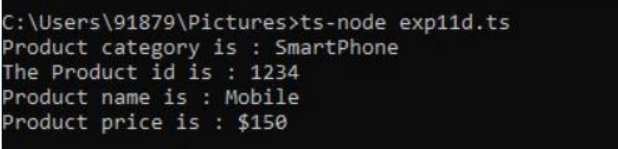
g.getProduct();

g.getProductId();

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

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

**OUTPUT:**

****

**Viva Questions**

**1. What is access modifiers in angular?**

A.An access modifier is a keyword that we can apply to a member of a class to control its access from the outside.

**2.** **What are the modifiers for TypeScript classes?**

A.TypeScript provides three access modifiers to class properties and methods: private , protected , and public .

**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().**

**AIM:**To create a Product class with 4 properties namely productId and methods to setProductId() and getProductId().

**DESCRIPTION:** In typescript, the method is a piece of code that has been declared within the class and it can be carried out when it is called. Method property in it can split a huge task into little sections and then execute the particular operation of that program so that code can be reusable which can improve the module from the program.

**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:**

****

**Viva Questoins**

**1. What is the difference between property and method in TypeScript?**

A.Properties define it, while methods allow it to do things. However, a method is really just a property that can be called through references to functions. A property can be either a value or a function, and the function is known as the method.

**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.**

**AIM:** To create a namespace called ProductUtility and place the Product class definition in it. Import the Product class inside productlist file and use it.

**DESCRIPTION:** In typescript, the method is a piece of code that has been declared within the class and it can be carried out when it is called. Method property in it can split a huge task into little sections and then execute the particular operation of that program so that code can be reusable which can improve the module from the program. The classes or interfaces which should be accessed outside the namespace should be marked with keyword export. To access the class or interface in another namespace,

the syntax will be

namespaceName.className

**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

{

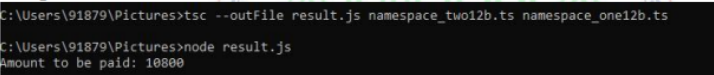
return price \* quantity;

}

}

}

**OUTPUT:**

****

**Viva Questions**

**1. What is a namespace in TypeScript?**

A.Namespaces are a TypeScript-specific way to organize code. Namespaces are simply named JavaScript objects in the global namespace.

**2.What is the purpose of a namespace?**

A.Namespaces are used to organize code into logical groups and to prevent name collisions that can occur especially when your code base includes multiple libraries.

**3.what is the syntax for declaring a namespace?**

**A.** namespace <namespace\_name> {

export interface I1 { }

export class c1{ }

}

**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.**

**AIM:** To creating and using Modules Consider the Mobile Cart application.

**DESCRIPTION:** A module refers to a set of standardized parts or independent units that can be used to construct a more complex structure.

TypeScript modules provides a way to organize the code for better reuse.

export interface InterfaceName

{

//Block of statements

}

**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);

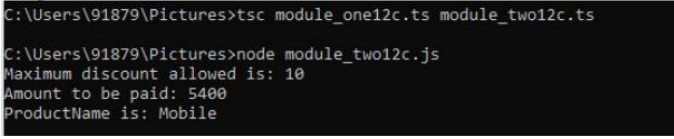
const discount = MaxDiscountAllowed(2);

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

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

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

**OUTPUT:**

****

**Viva Questions**

**1. What is a module in TypeScript?**

A.In TypeScript, just as in ECMAScript 2015, any file containing a top-level import or export is considered a module.

**2.What is the syntax for declaring a module?**

A. import { class/interface name } from 'module\_name';

**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.**

**AIM:**To create a generic array and function to sort numbers as well as string values.

**DESCRIPTION:** Whenever any program or code is written or executed, one major thing one always takes care of which is nothing but making reusable components which further ensures the scalability and flexibility of the program or the code for a long time. Generics, thus here comes into the picture as it provides a user to flexibly write the code of any particular data type (or return type) and that the time of calling that user could pass on the data type or the return type specifically. Generics provides a way to make the components work with any of the data types (or return types) at the time of calling it for a certain number of parameters (or arguments). In generics, we pass a parameter called type parameter which is put in between the lesser sign (<),and the greater sign (>) for example, it should be like <type\_parameter\_name> .

**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'];

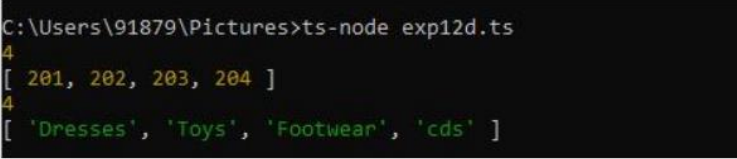
const idList = orderDetails(orderid);

console.log(idList);

const nameList = orderDetails(ordername);

console.log(nameList);

**OUTPUT:**

****

**Viva Questions**

**1. What is a generic function in TypeScript?**

A.Generics in TypeScript are a method for creating reusable components or functions that can handle multiple types. Generics are a powerful tool that can assist us in creating reusable functions

**2. What is generic constraints in TypeScript?**

A.Using type parameters in generic constraints

TypeScript allows you to declare a type parameter constrained by another type parameter. The following prop() function accepts an object and a property name. It returns the value of the property.

**13. Design any front end web application using HTML, CSS and validate client side using Javascript. Also use typescript, Node.js, Express.js wherever applicable.**

**AIM:** Design any front end web application using HTML, CSS and validate client side using Javascript. Also use typescript, Node.js, Express.js wherever applicable.

**PROGRAM:**