EX No: 1 Design a Webpage using all HTML elements

CODE:

home.html

```
<html> <head>
<Title>HOME PAGE</title>
</head>
<frameset cols="20%,80%" Scrolling =no border = 0>
<frame src="left.html" name="left"/>
<frame src="right.html"name="right"/>
</frameset>
</html>
```

Left.html

```
<html>  <head>
<title> Left Page </title>
</head>    <center>
<img src="college.jpeg" height=300 width=300>
</center>
<br/><br/><br/>   <center>
<h3><b><a href="right.html" target=right>HOME</a></b></h3>
<h3><b><a href="image.html" target=right>IMAGES</a></b></h3>
<h3><b><a href="symposium.html" target=right>SYMPOSIUM</a></b></h3>
<h3><b><a href="contact.html" target=right>CONTACT</a></b></h3>
</center>    </body>    </html>
```

Right.html

```
<html>    <head>
<title> Right Page </title>
</head>   <center>
<h1>St. JOSEPH'S INSTITUTE OF TECHNOLOGY</h3>
<h3>JEPPIAR NAGAR, OMR </h3>
<h2>DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE </h2>
<img src="admin.jpeg" width="500" Height="300">
</center>
<div id="right"><br> St. Joseph’s Institute of Technology is a higher education institution in OMR, Chennai, India,
```

established in the year 2011, with 6 UG courses (B.E. – CSE, ECE, EEE & Mech. Engg.,

B.Tech. IT & B.Tech Artificial Intelligence and Data Science is offered from the year 2021. MBA is offered from the year 2022.

Our college is affiliated to the Anna University and approved by AICTE.

The courses offered from the inception (B.E. – CSE, ECE & Mech. Engg. & B.Tech. IT) have been accredited by NBA in 2017 in the shortest span of 6 years since inception,

and granted Autonomous status by UGC in 2022 which ascertains our quality.

 Copy Rights.All Rights Reserved.

</body>

</html>

Image.html

<html> <head>

<title> Image Link </title>

</head>

<h1> DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE </h1>

 <img src="it.gif"><img src=it1.jpeg" width=”600” height= “400”>

About us

 WHO WE ARE

Department of Artificial Intelligence and Data Science with state-of-art research

facilities and well qualified faculty members produces young aspiring

Engineers

<h2>VISION </h2>

To achieve and sustain international prominence in education,

research and professional skill development

<h2>MISSION</h2>

To set a standard in education by developing the intellectual strength of students and guiding them to have a sound and

updated knowledge in the emerging areas of technology

```
</div></div><div id="bottom">

<center><br><br><font size="2"> Copy Rights.All Rights Reserved.</font></center>

</div>    </center>   </body>  </html>
```

Symposium.html

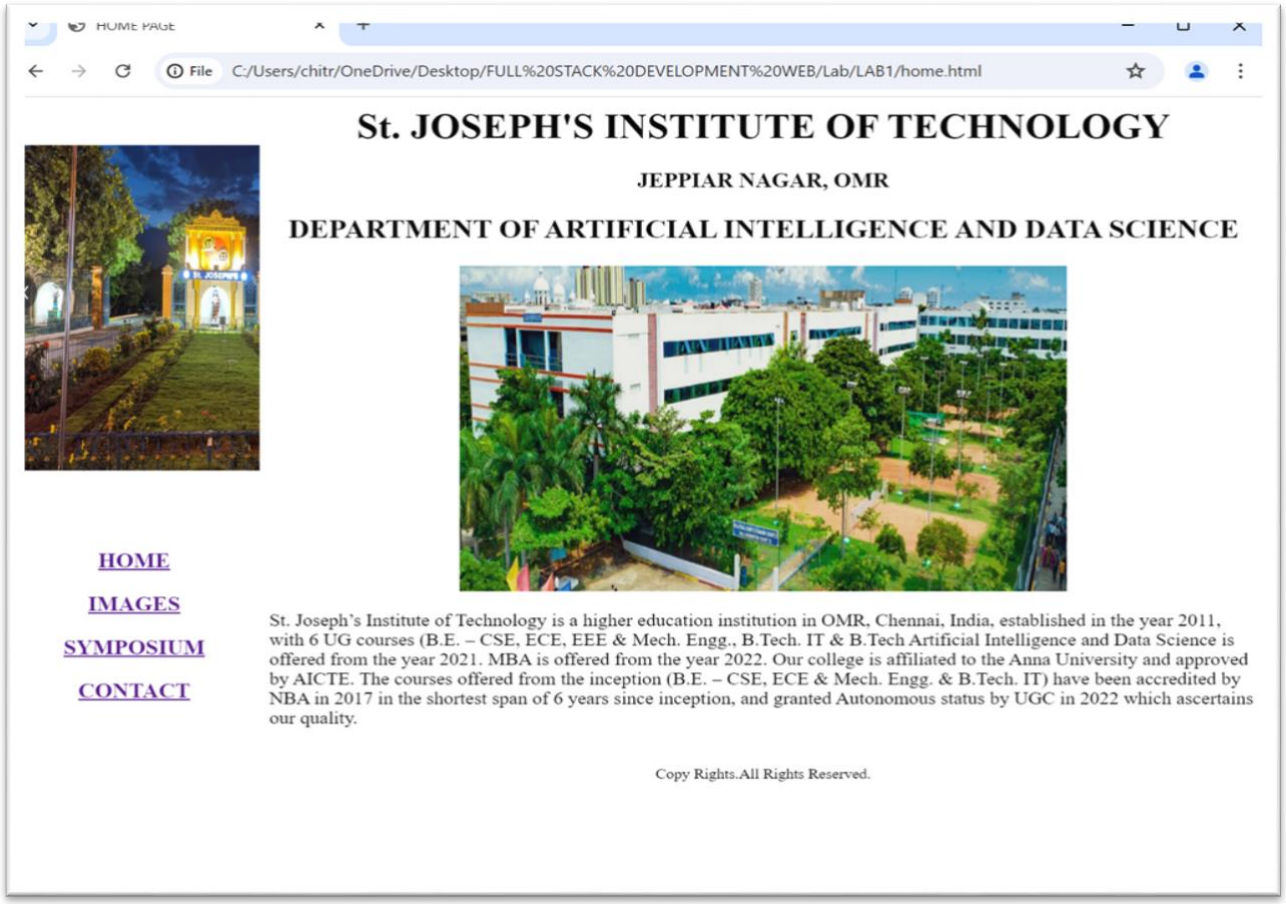
```
<html>   <head>

<title> Symposium </title>

</head>  <body>

<center> <h1> DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE
</h1>  <br/><br/>

<form action="localhost" method="post">

<table border=”2”>

<tr> <td> Name </td>

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

</tr>  <tr> <td> College Name </td>

<td> <input type="text" name="cname"></td>

</tr> <tr> <td> Department </td>

<td> <input type="text" name="dept"></td>

</tr>

<tr> <td> Mail ID </td>

<td> <input type="text" name="mail"></td>

</tr>

<tr> <td> Phone No </td>

<td> <input type="text" name="pno"></td>

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

<td><input type="submit" value="Submit"></td>

</tr>

</form>  </center>  </body>   </html>
```

Contact.html

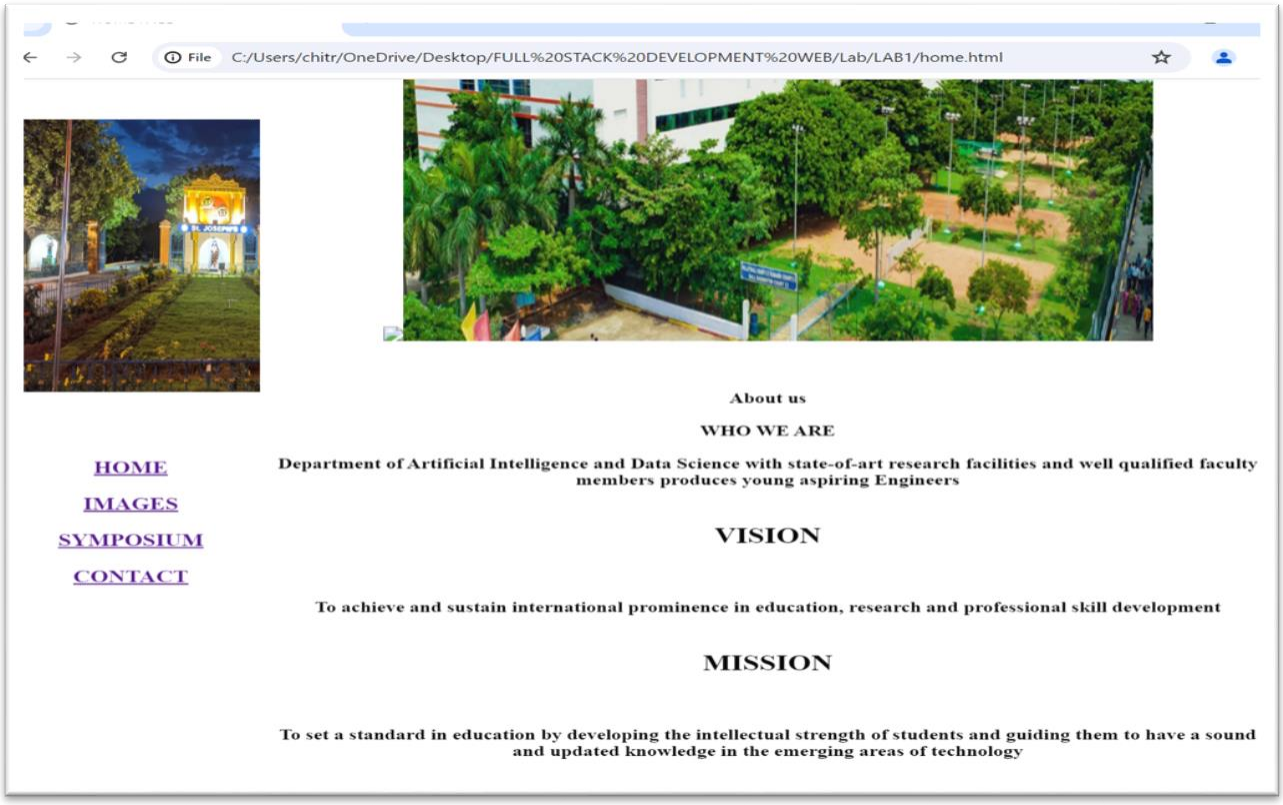
```
<html>  <head>

<title> Contact Page </title>

</head>  <body>

<center>
```

```
<h1>St. JOSEPH'S INSTITUTE OF TECHNOLOGY</h3>
<h2> DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE </h2>
<br/><br/><br/><br/>
<img src="it11.jpg" height="500" width="500">
</center><br/><br/><br/><br/>
<br/><br/><br/><br/>
<table>
<tr> <th> Phone no: </th>
<td> 123456789 </td>
</tr>
<tr> <th> EMAIL-ID: </th>
<td> xyz@gmail.com </td>
</tr>
</table> </body> </html>
```

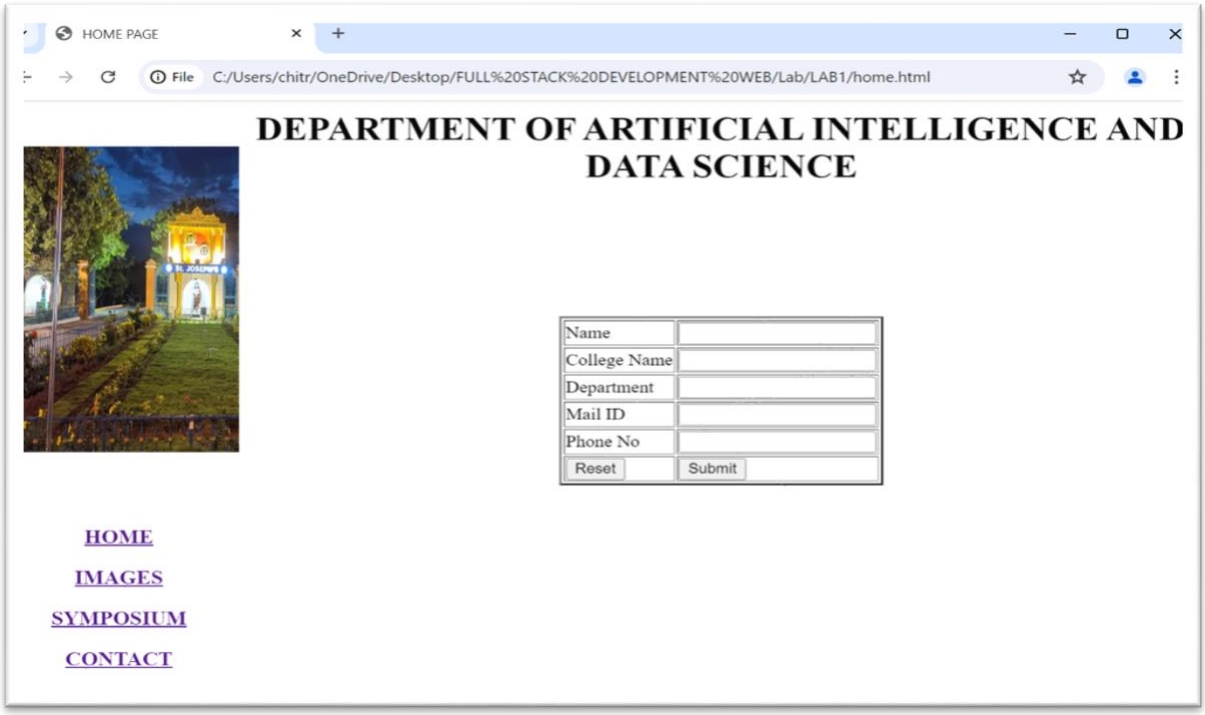
Output : Home .html



Images.html



Symposium.html



Result:

Ex.No : 2 Create a Web Page with all types of Cascading Style Sheets and CSS Selectors

Program

style.html

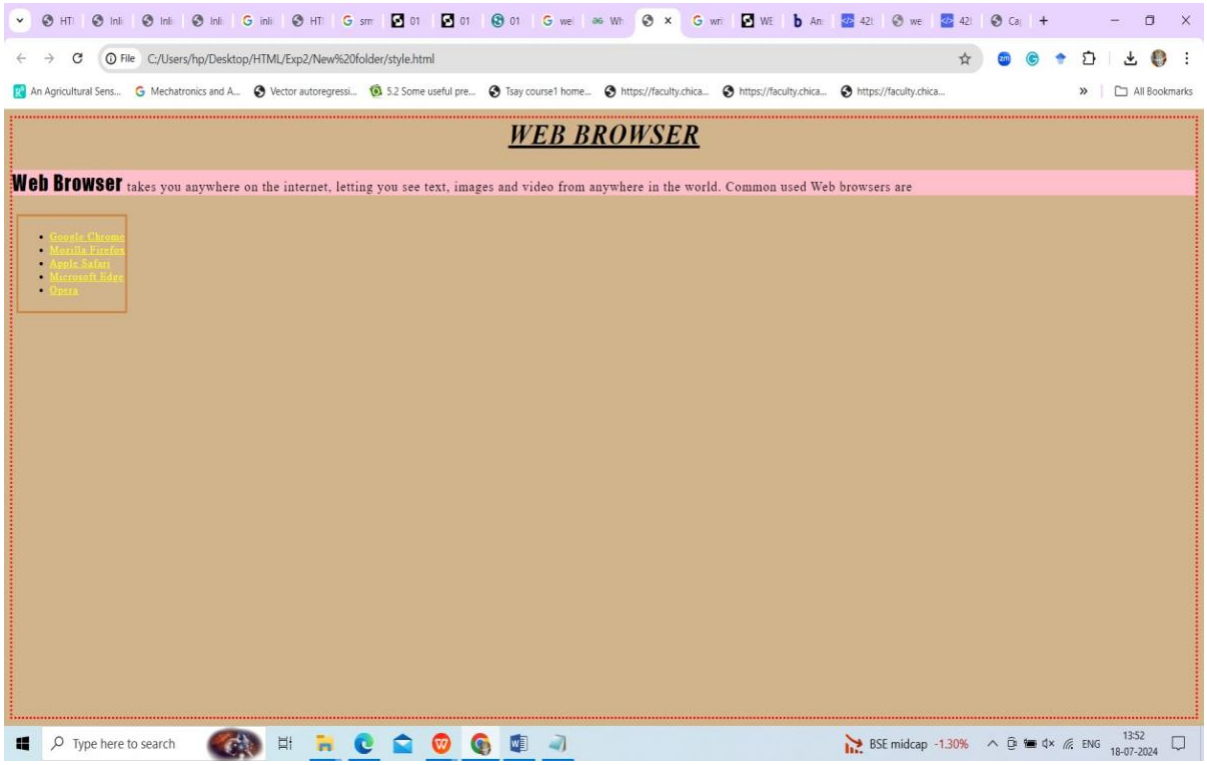
```
<html> <head>
<title> Popular Web Browser </title>
<!--Extended Style Sheet--->
<link rel="stylesheet" type="text/css" href="style.css">
<!--Embedded Style Sheet--->
<style type="text/css">
p{
background-color:pink;
text-align:justify;
margin:2 em 7em;
}
</style> </head>
<body id="body">
<h1> WEB BROWSER </h1>
<p> <span style="font:200 x-large fantasy"> Web Browser </span>
 takes you anywhere on the internet, letting you see text, images and video from anywhere in the world. Common used Web browsers are
</p> <!-- Inline Style Sheet -->
<table style ="background-position:center;text-align:Center;padding:3px;">
<tr> <td align ="left">
<div class="div">
<ul>
<li><a href="">Google Chrome </a> </li>
<li><a href="">Mozilla Firefox </a> </li>
<li><a href="">Apple Safari </a> </li>
<li><a href="">Microsoft Edge </a> </li>
<li><a href="">Opera </a> </li>
</ul></div> </td> </tr> </table> </body> </html>
```

style.css

```
h1,h2 {
```

```
text-decoration: Underline;
font-style: italic;
text-aligh:center;}
#body{
background-color:tan;
border: red dotted;
text-align:center;
}
.div
{ border:peru solid; }
*{ letter-spacing: 1px; }
a:link
{ color:black; }
a:visited
{ color:yellow; }
a:hover
{color: green; }
a:active
{ colot:blue; }
ul li
{ font-size:small;}
```

Output:



Result:

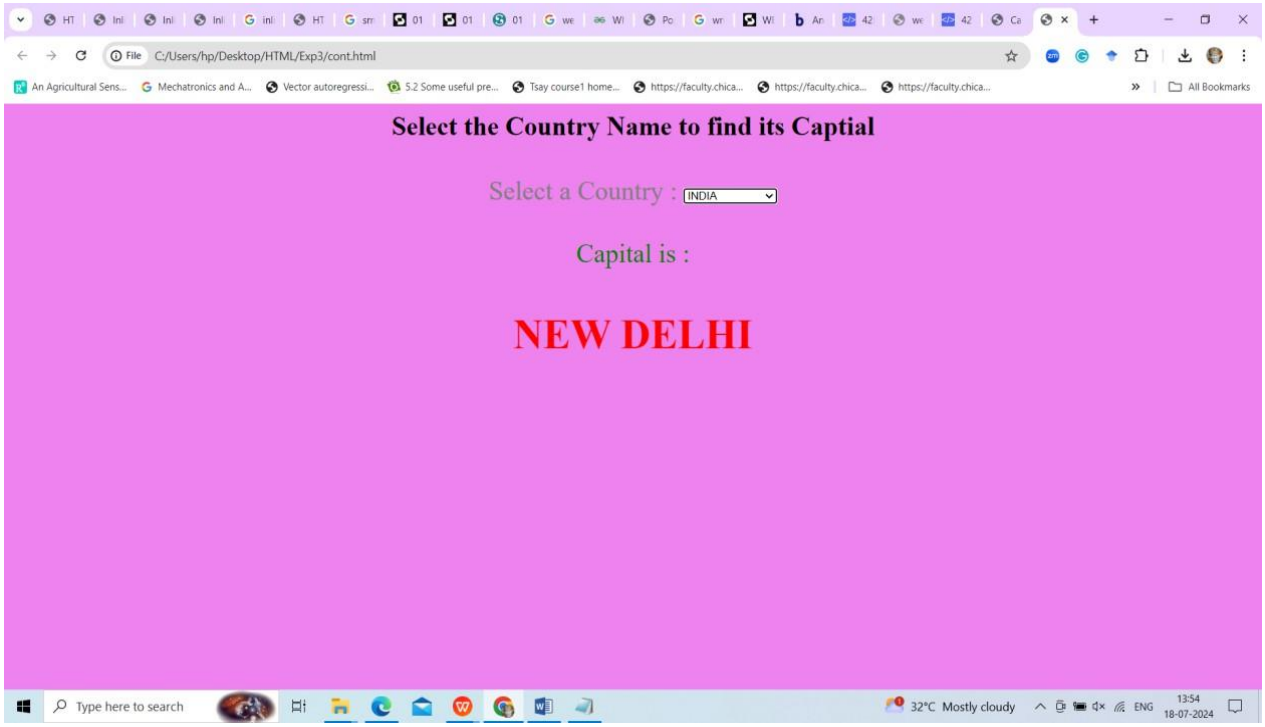
Ex.No: 3 Write an HTML page that contains a selection box with a list of 5 countries. When the user selects a country, its capital should be printed next in the list. Add CSS to customize the properties of the font of the capital (color, bold and font size).

Program

```
<html>
<head>
<title>Capitals of countries</title>
<center>
<h1> Select the Country Name to find its Captial </h1> </center>
<body bgcolor="violet">
<style>
p{
color:red;
font-weight:bold;
font-size:50;
}
</style>
<script language="javascript">
function capital()
{
var cunt=document.forms["frm1"].country.value;
var capital=" Select a country ";
if( cunt=="india")
{ capital="NEW DELHI"; }
if( cunt=="china")
{
capital="BEIJING";
}
if( cunt=="pakistan")
{ capital="ISLAMABAD"; }
if( cunt=="bangladesh")
{ capital="DHAKA"; }
if( cunt=="japan")
{ capital="TOKYO"; }
```

```
if( cunt=="select")
{
  capital="Select a country";
}
document.getElementById("capt").innerHTML=capital;
}
</script>    </head>
<body>
<form name="frm1">
<br/> <center>
<font color="gray" size="6">
Select a Country : <select name="country" onchange="capital()">
<option value="select">--SELECT--</option>
<option value="india">INDIA</option>
<option value="china">CHINA</option>
<option value="pakistan">PAKISTAN</option>
<option value="bangladesh">BANGLADESH</option>
<option value="japan">JAPAN</option>
</select>
<br/>
<br/>
<font color="green" size="6">Capital is :</font> <p id="capt"></p>
</center>
</form>
</body>
</html>
```

Output



Result:

Exp.no :4 Write client-side scripts for validating web form control using DHTML

Code:

email.js

```
function ValidateEmail(input) {

var validRegex = /^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/;

  if (input.value.match(validRegex)) {

  alert("Valid email address!");

  document.form1.text1.focus();

  return true;

 } else {

   alert("Invalid email address!");

   document.form1.text1.focus();

   return false;  }  }
```

email.html

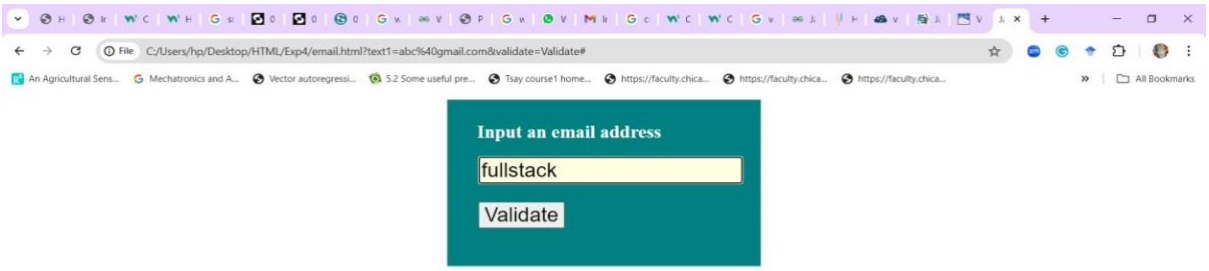
```
<!DOCTYPE html>

<html lang="en">

  <head>  <meta charset="utf-8" />

    <title>JavaScript Email Validation</title>

    <link rel="stylesheet" href="email.css" type="text/css" />

  </head>

  <body onload="document.form1.text1.focus()">

    <div class="mail">

      <h2>Input an email address</h2>

      <form name="form1" action="#">

        <ul>

          <li><input type="text" name="text1" /></li>

          <li> </li>

          <li class="validate">

            <input

              type="submit"

              name="validate"

              value="Validate"
```

```
            onclick="ValidateEmail(document.form1.text1)"
          />           </li>
        <li> </li>         </ul>
     </form>     </div>
   <script src="email.js"></script>
 </body> </html>
```

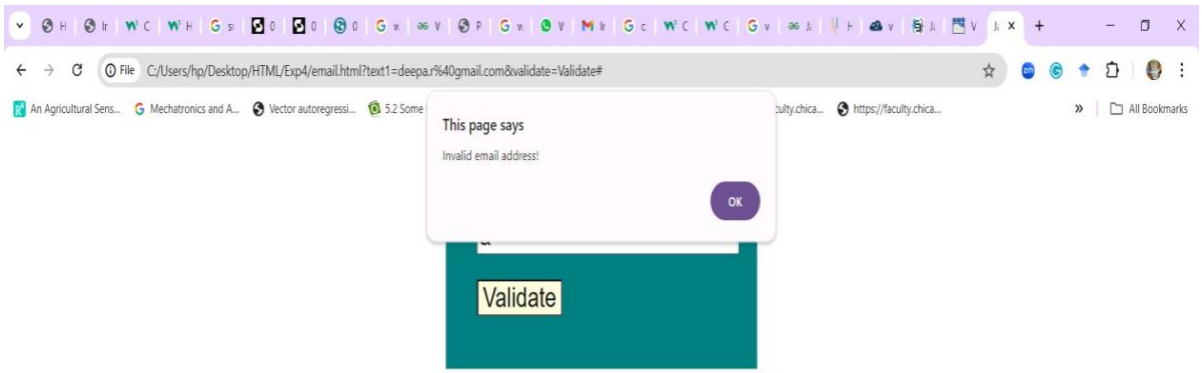
email.js

```
li {
  list-style-type: none;
  font-size: 16pt;
}
.mail             {
  margin: auto;
  padding-top:       10px;
  padding-bottom: 10px;
  width:400px;
  background:teal;
  border: 1px soild silver;
}
.mail h2 {
  margin-left:     38px;
  color: white;
}
input {  font-size: 20pt; }
input:focus, textarea:focus { background-color: lightyellow; }
input validate {font-size: 12pt;}
.rq {
  color: #ff0000;
  font-size: 10pt;
}
```

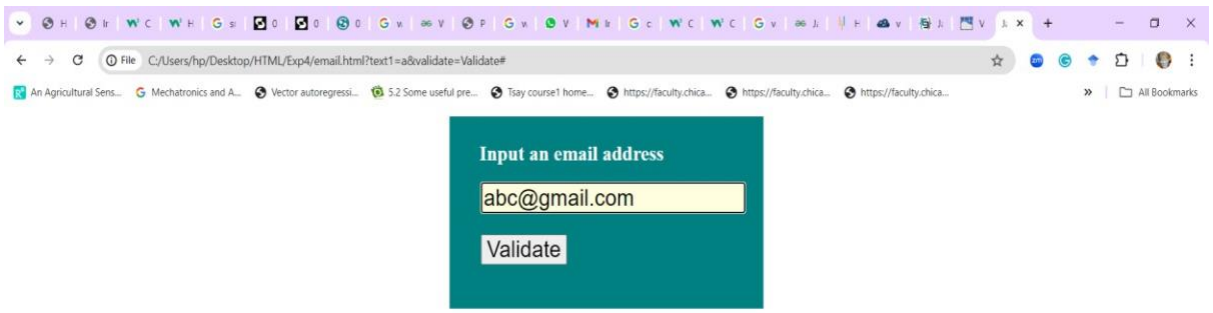
OUTPUT:


C:/Users/hp/Desktop/HTML/Exp4/email.html?text1=abc%40gmail.com&validate=Validate#
Input an email address
fullstack
Validate

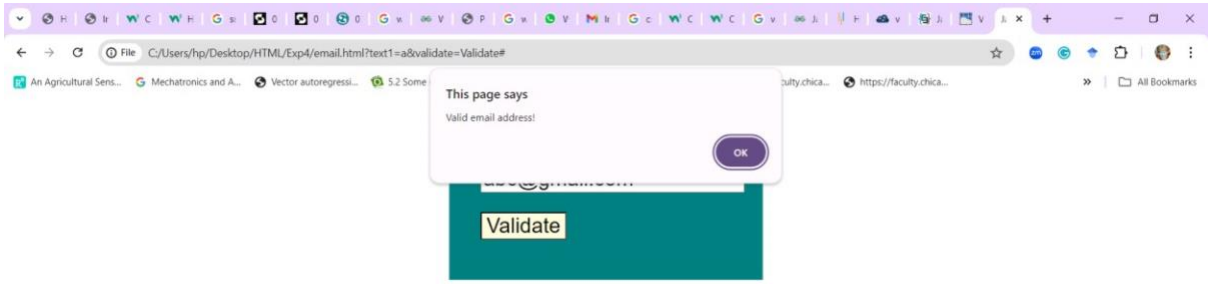

Type here to search
32°C Mostly cloudy
ENG
13:08
20-07-2024


C:/Users/hp/Desktop/HTML/Exp4/email.html?text1=deepa.r%40gmail.com&validate=Validate#
This page says
Invalid email address!
OK
Validate


English (United States)
English (India) keyboard
To switch input methods, press Windows key+Space.
Type here to search
Watchlist ideas
ENG
13:07
20-07-2024


File C:/Users/hp/Desktop/HTML/Exp4/email.html?text1=a&validate=Validate#
Input an email address
abc@gmail.com
Validate


Type here to search
Watchlist ideas
ENG
13:07
20-07-2024


File C:/Users/hp/Desktop/HTML/Exp4/email.html?text1=a&validate=Validate#
This page says
Valid email address!
OK
Validate


Type here to search
Watchlist ideas
ENG
13:08
20-07-2024

Result:

Exp 5: To Design the following

a. **Design a Image Slider Show**

Program

Image.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.3.0/css/all.min.css" integrity="sha512-SzlrxWUlpfuzQ+pcUCosxcglQRNAq/DZjVsC0lE40xsADsfeQoEypE+enwcOiGjk/bSuGGKHEyjSoQ1zVisanQ==" crossorigin="anonymous" referrerpolicy="no-referrer" />
    <script src="script.js" defer></script>
    <title>Image Slider</title>
</head>
<body>
    <div class="container">
        <a class="btn btn-left"><i class="fa-solid fa-caret-left"></i></a>
        <a class="btn btn-right"><i class="fa-solid fa-caret-right"></i></a>
    </div>
</body>
</html>
```

Script.js

```
function initializeImageSlider(containerSelector, buttonSelector, images) {
    const container = document.querySelector(containerSelector);

    const btns = document.querySelectorAll(buttonSelector);
    const imgList = images;

    let index = 0;
```

```
    console.log(index);



    btns.forEach((button)           =>          {
        button.addEventListener('click', () => {
            if (button.classList.contains('btn-left')) {
                index--;
            if (index < 0) { \
            index = imgList.length - 1;
                }
            } else {
                index++;
                if (index === imgList.length) {
                    index = 0;
                }
}
            container.style.background = `url("img/${imgList[index]}.jpg") center/cover fixed
no-repeat`;

});
    });
}
// Usage example:

initializeImageSlider('.container', '.btn', ["1", "2", "3", "4"]);
```

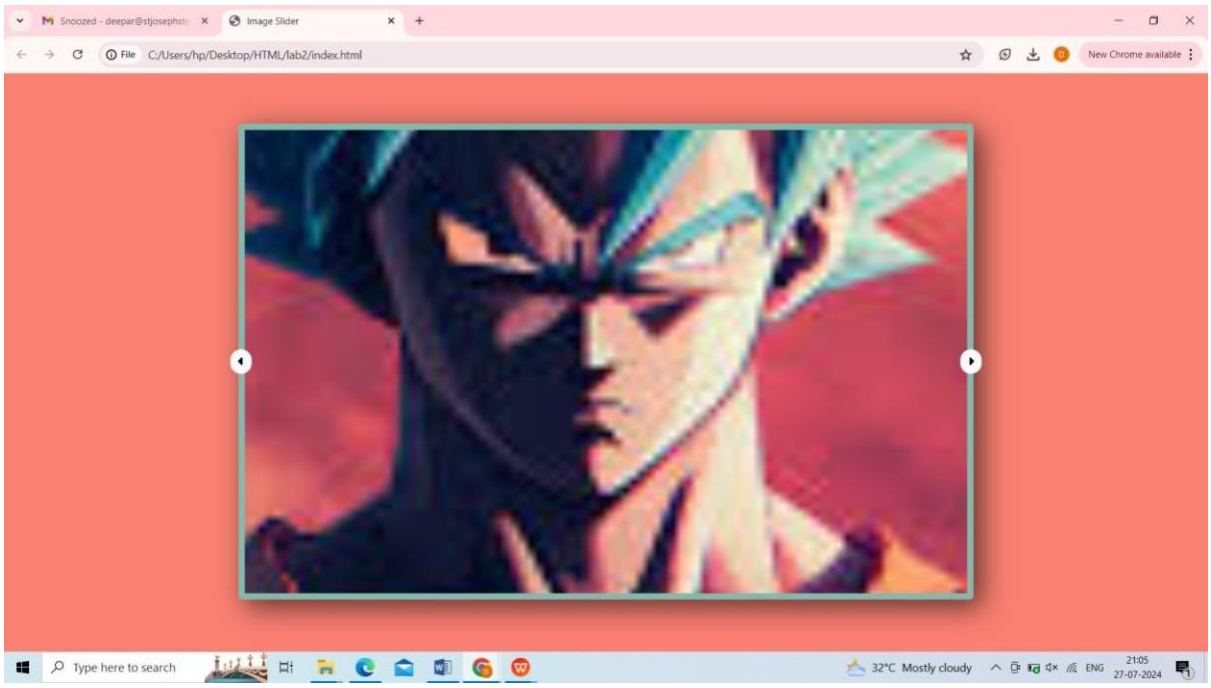
style.css

```
body{
    margin:0;
    padding:0;
```

```
    box-sizing:      border-box;
    background-color: salmon;
}
.container{
    min-height: 80vh;
    width:60%;
    border:8px solid rgb(129, 187, 174);
    margin:4rem auto;
    border-radius: 5px;
    box-shadow: 10px 10px 35px rgba(0,0,0,0.6);
    position:relative;
    background: url("img/1.jpg") center/cover fixed no-repeat;
    transition:background 1s ease-in-out;
}
.btn-left{
    position:absolute;
    top:50%;
    left:-2%;
    background-color: white;
    padding:.4em          .6em;
    border-radius: 2em;
    transform:translateY(calc(-50%));
    cursor: pointer;
}
.btn-right{
    position:absolute;
```

```
top:50%;

right:-2%;

background-color: white;

padding:.4em .6em;

border-radius: 2em;

transform:translateY(calc(-50%));

cursor: pointer; }
```

Output



b. Design a Digital Clock

- **Clock.html**

```
<!DOCTYPE html>

<html lang="en">

<head>

        <meta charset="UTF-8">

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

        <title>Digital Clock</title>

        <link rel="stylesheet" href="clo.css">

</head>

<body>
        <div id="clock">8:10:45</div>

        <script src="digital.js"></script>

</body> </html>
```

clo.css

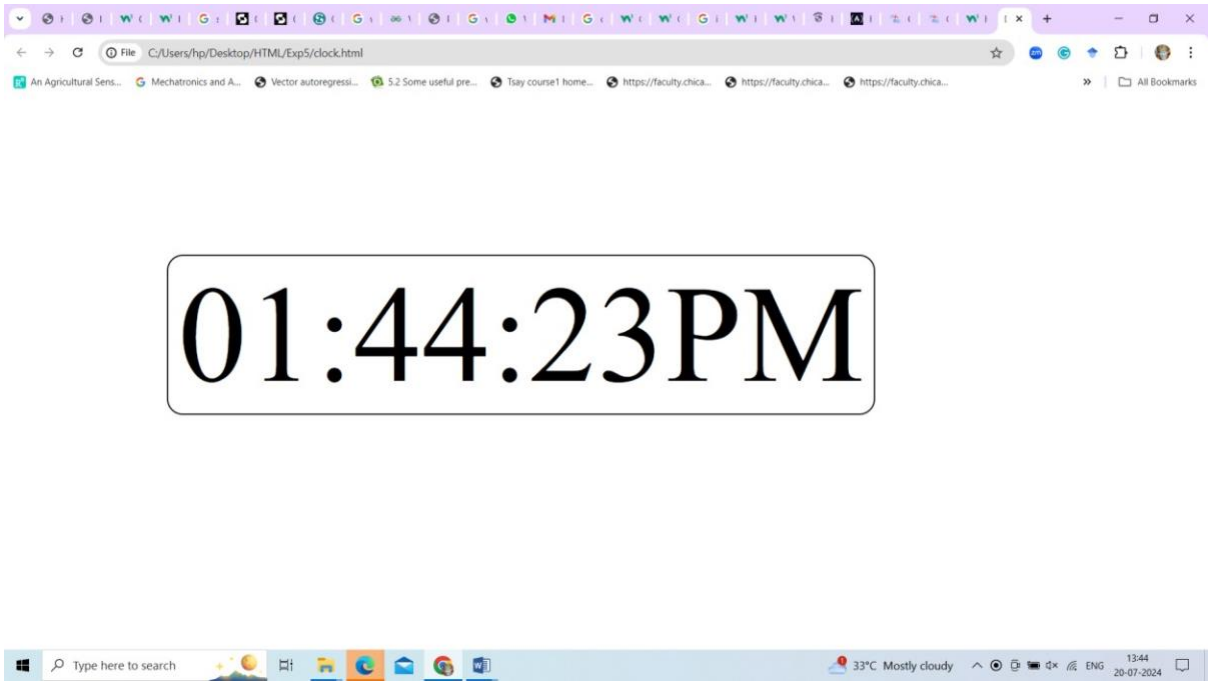
```
#clock {

        font-size:  175px;
        width:      900px;
        margin:     200px;
        text-align: center;
        border: 2px solid black;
        border-radius: 20px;

}
```

Digital.js

```
// Calling showTime function at every second
setInterval(showTime, 1000);

 // Defining showTime funcion
```

```
function showTime() {
	// Getting current time and date
	let time = new Date();

	let hour = time.getHours();
	let min = time.getMinutes();
	let sec = time.getSeconds();
	am_pm = "AM";


	// Setting time for 12 Hrs format
	if (hour >= 12) {

		if (hour > 12) hour -= 12;
		am_pm = "PM";

	} else if (hour == 0) {hr
		= 12;
		am_pm = "AM";

	}


	hour =hour < 10 ? "0" + hour : hour;
	min = min < 10 ? "0" + min : min;sec = sec < 10 ? "0" + sec : sec;

	let currentTime =

		hour +
		":" +

		min +
		":" +

		sec	+
		am_pm;


	//	Displaying	the	time
	document.getElementById(

		"clock"

	).innerHTML = currentTime;

}

showTime()
```

Output :



Result:

Ex: 6 Design a shopping cart application using React. You Shopping webpage a should have the provisions for selecting the list of items from different category. Once the items are selected on clicking the submit button the items in the cart with its price should be displayed.

Program.

```
import React, { useState } from 'react';
 const items = [
   { id: 1, name: 'Apple', price: 10 },
   { id: 2, name: 'Banana', price: 5 },
   { id: 3, name: 'Orange' , price:20 },
   { id: 4, name: 'Papaya' , price:15}
 ];
 function App() {
   const [cart, setCart] = useState([]);
   const [totalPrice, setTotalPrice] = useState(0);
   const handleAddToCart = (item) => setCart([...cart, item]);
   const handleSubmit = () => {
     const total = cart.reduce((sum, item) => sum + item.price, 0);
     setTotalPrice(total);
   };
   return (
     <div>
       <h2>Shopping Cart</h2>
       {items.map(item => (
         <div key={item.id}>          {item.name} - ${item.price}
           <button onClick={() => handleAddToCart(item)}>Add to    Cart</button>
         </div>
       ))}
       <h3>Cart:</h3>
       {cart.map((item, index) => (
         <div key={index}>{item.name} - ${item.price}</div>
       ))}
       <button onClick={handleSubmit}>Submit</button>
       {totalPrice > 0 && <h3>Total Price: ${totalPrice}</h3>}
     </div>
   );
 }
export default App;
```

Output:

Shopping Cart

Apple - $10 Add to Cart
Banana - $5 Add to Cart
Orange - $12 Add to Cart

Cart:

Submit

Shopping Cart

Apple - $10 Add to Cart
Banana - $5 Add to Cart
Orange - $12 Add to Cart

Cart:

Apple - $10
Banana - $5
Orange - $12
Submit

Total Price: $27

Result:

Ex: 7 Design an Online super market using ExpressJs and MongoDB database a) Perform a search based on product id or name b) On retrieving the results , display the product details of different brands in table format with the price field in stored order using React.

Express code

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors'); // Import cors

const app = express();
const port = 3000;

// Use CORS middleware
app.use(cors()); // This will allow all origins by default

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/online_supermarket', {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
  .then(() => console.log('MongoDB connected'))
  .catch(err => console.error(err));

// Define product schema
const productSchema = new mongoose.Schema({
  id: Number,
  name: String,
  brand: String,
  price: Number
});

const Product = mongoose.model('Product', productSchema);

// API endpoints
app.get('/products', async (req, res) => {
  try {
    const products = await Product.find();
    res.json(products);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

app.get('/products/:id', async (req, res) => {
  try {
    const product = await Product.findById(req.params.id);
    if (product == null) {
      return res.status(404).json({ message: 'Product not found' });
    }
    res.json(product);
```

```
  } catch (err) {
     res.status(500).json({ message: err.message });
  }
});

app.listen(port, () => {
  console.log(`Server listening on port ${port}`);
});
```

React Code:

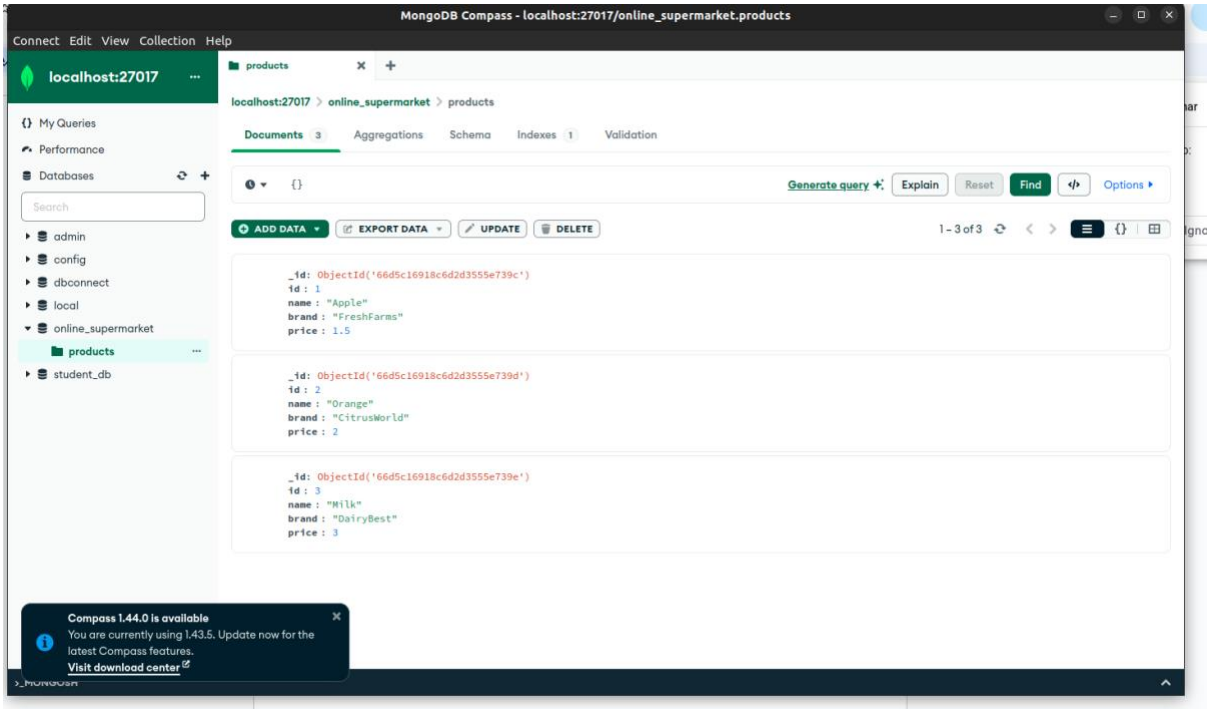
```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import './App.css'; // Import the CSS file

function App() {
 const [products, setProducts] = useState([]);
 const [filteredProducts, setFilteredProducts] = useState([]);
 const [searchTerm, setSearchTerm] = useState('');
 const [error, setError] = useState(null);

 useEffect(() => {
  const fetchProducts = async () => {
   try {
    const response = await axios.get('http://localhost:3000/products');
    console.log('Products data:', response.data);
    setProducts(response.data);
    setFilteredProducts(response.data);     } catch (error) {
    console.error("There was an error fetching the products!", error);
    setError("Failed to fetch products.");
   }
  };
  fetchProducts();
 }, []);

 useEffect(() => {
  // Filter and sort products whenever searchTerm changes
  const filtered = products.filter(product =>
   product.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
   product.id.toString().includes(searchTerm)
  );

  // Sort products by price in ascending order
  const sorted = filtered.sort((a, b) => a.price - b.price);

  setFilteredProducts(sorted);
 }, [searchTerm, products]);

// Group products by brand
 const groupByBrand = (products) => {
```

```
  return products.reduce((acc, product) => {
    if (!acc[product.brand]) {
      acc[product.brand] = [];
    }
    acc[product.brand].push(product);
    return acc;
  }, {});
};

const productsByBrand = groupByBrand(filteredProducts);

return (
  <div className="app-container">
    <h1>Online Supermarket</h1>
    {error && <p className="error">{error}</p>}

    <input
      type="text"
      placeholder="Search by product ID or name"
      value={searchTerm}
      onChange={(e) => setSearchTerm(e.target.value)}
      className="search-bar"
    />

    {Object.keys(productsByBrand).length > 0 ? (
      Object.keys(productsByBrand).map(brand => (
        <div key={brand} className="brand-section">
          <h2 className="brand-name">{brand}</h2>
          <table className="product-table">
            <thead>
              <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Price</th>
              </tr>
            </thead>
            <tbody>
              {productsByBrand[brand].map(product => (
                <tr key={product._id}>
                  <td>{product.id}</td>
                  <td>{product.name}</td>
                  <td>${product.price.toFixed(2)}</td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      ))
    ) : (
      <p>No products found</p>
```

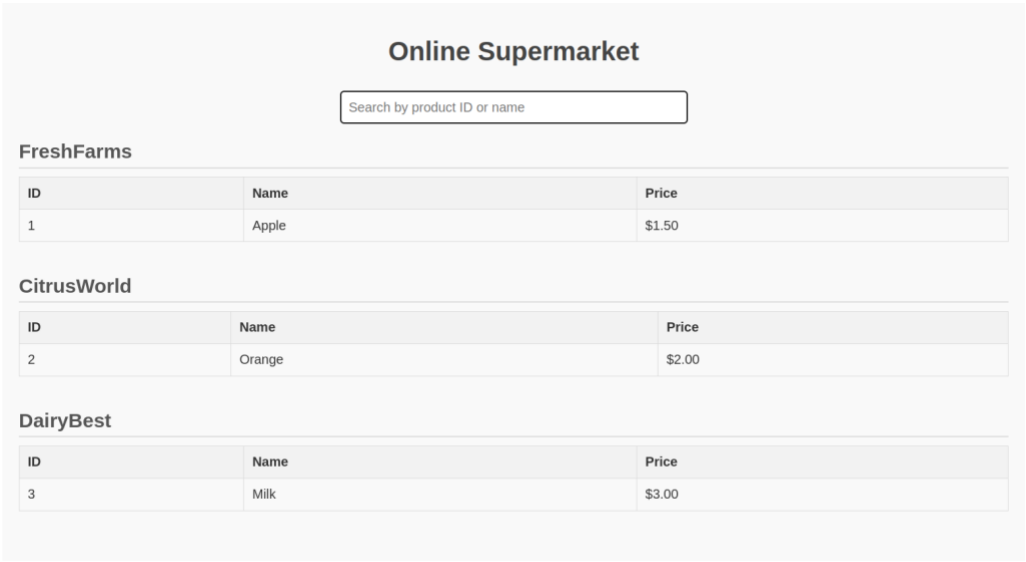
```
    )}
  </div>
 );
}
export default App;
```

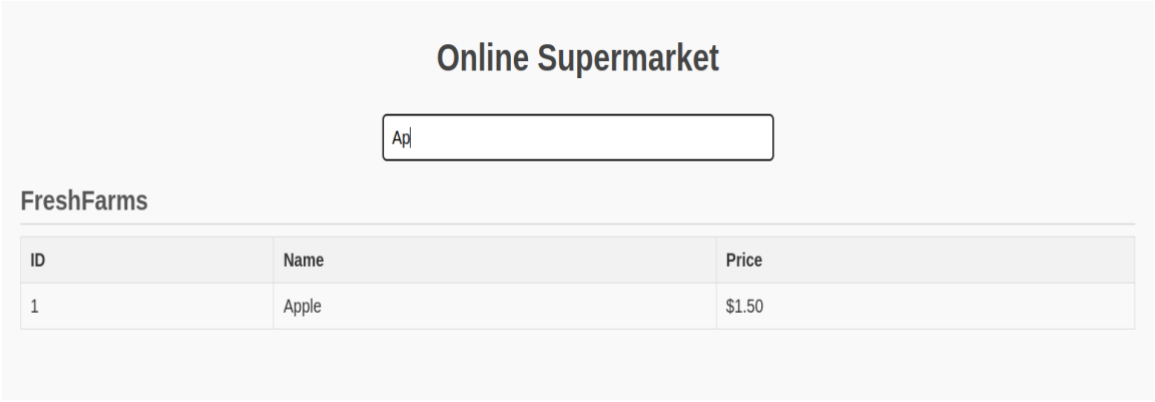
MongoDB Schema:

```
{
  "_id": "ObjectId",
  "id": "Number",
  "name": "String",
  "brand": "String",
  "price": "Number"
}
```



OUTPUT:





Result:

Ex: 8. Create a <TodoItem> component in React and reuse it inside a <TodoList> component.

Program:

React Code:

TodoItem.jsx:

```
import React from 'react';
import '../components/TodoList';

function TodoItem({ todo, toggle }) {
 return (
   <li
     onClick={() => toggle(todo.id)}
     style={{ textDecoration: todo.completed ? 'line-through' : 'none' }}
   >
     {todo.text}
   </li>
);
}
export default TodoItem;
```

TodoList.jsx

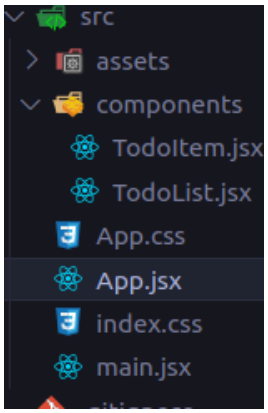
```
// src/components/TodoList.jsx
import React, { useState } from 'react';
import TodoItem from '../components/TodoItem.jsx';

function TodoList() {
 const [todos, setTodos] = useState([
   { id: 1, text: 'Learn React', completed: false },
   { id: 2, text: 'Build a to-do app', completed: false },
   { id: 3, text: 'Master JavaScript', completed: false },
 ]);

 const toggleTodo = (id) => {
   setTodos(todos.map(todo =>
     todo.id === id ? { ...todo, completed: !todo.completed } : todo
   ));
 };
 return (
   <ul>
     {todos.map(todo => (
       <TodoItem
         key={todo.id}
         todo={todo}
         toggle={() => toggleTodo(todo.id)}
       />
     ))}
```

```
  </ul>
 );   }
export default TodoList;
```

App.jsx:

```
import React from 'react';
import '../src/App.css'
import TodoList from './components/TodoList';

function App() {
 return (
   <div className="app-container">
     <h1 className="app-title">My To-Do List</h1>
     <TodoList />
   </div>
 );
}
export default App;
```

Project Structure:



Output:


My To-Do List
Learn React
Build a to-do app
Master JavaScript


My To-Do List
Learn React
Build a to-do app
Master JavaScript

Result:

Ex: 10 Create a basic CRUD operation API by following API following REST syntax for a given model student with the following fields [field names].

Code:

Set Up Your API in Postman

1. Open Postman and create a new request collection for your Student API to keep everything organized.
2. Ensure your server is running (i.e., node server.js or npx nodemon server.js).
3. Use the base URL for your requests: http://localhost:5000/api/students.

2. CRUD Operations Using Postman

1. Create a Student (POST Request)

- Endpoint: POST http://localhost:5000/api/students
- Headers:
  - Content-Type: application/json
- Body: Select "raw" and set the type to JSON. Use the following JSON structure

```
{
 "name": "John Doe",
 "age": 21,
 "email": "johndoe@example.com",
 "grade": "A"
}
```

- Send the request: You should receive a response with the newly created student object and a 201 Created status.

2. Get All Students (GET Request)

- Endpoint: GET http://localhost:5000/api/students
- Headers: None required.
- Send the request: This will return an array of all students currently in the database.

3. Get a Single Student by ID (GET Request)

- Endpoint: GET http://localhost:5000/api/students/:id
- Example: Replace :id with the actual ID of a student (you can get this from the list of students you retrieved in the previous step).
  - E.g., GET http://localhost:5000/api/students/64f9082399995d0f1a7c8abc
- Send the request: This will return the details of the specified student.

4. Update a Student (PUT Request)

- Endpoint: PUT http://localhost:5000/api/students/:id
- Headers:
  - Content-Type: application/json
- Body: Use JSON format to specify the fields you want to update

```
{
 "name": "Jane Doe",
 "age": 22,
 "email": "janedoe@example.com",
 "grade": "B"
}
```

- Send the request: You should receive a response with the updated student data.

5. Delete a Student (DELETE Request)

- Endpoint: DELETE http://localhost:5000/api/students/:id
- Example: Replace :id with the ID of the student you want to delete.
  - E.g., DELETE http://localhost:5000/api/students/64f9082399995d0f1a7c8abc

- Send the request: You should receive a confirmation message that the student has been deleted.

OUTPUT:
POST (Create):

json
Copy code

```
{
  "_id": "64f9082399995d0f1a7c8abc",
  "name": "John Doe",
  "age": 21,
  "email": "johndoe@example.com",
  "grade": "A",
  "__v": 0
}
```


GET (All Students):

json
Copy code

```
[
  {
    "_id": "64f9082399995d0f1a7c8abc",
    "name": "John Doe",
    "age": 21,
    "email": "johndoe@example.com",
    "grade": "A",
    "__v": 0
  }
]
```


GET (Single Student):

json
Copy code

```
{
  "_id": "64f9082399995d0f1a7c8abc",
  "name": "John Doe",
  "age": 21,
  "email": "johndoe@example.com",
  "grade": "A",
  "__v": 0
}
```


PUT (Update):

json
Copy code

```
{
  "_id": "64f9082399995d0f1a7c8abc",
```

```
  "name": "Jane Doe",
  "age": 22,
  "email": "janedoe@example.com",
  "grade": "B",
  "__v": 0
}
```


DELETE (Remove):

json
Copy code


```
{
  "message": "Student deleted successfully"
}
```


Result: