Hritish **102218050** **BS2**

MODERN WEB APPLICATIONS
Assignment

1. Design an Accordion using CSS and JavaScript.

Solution:

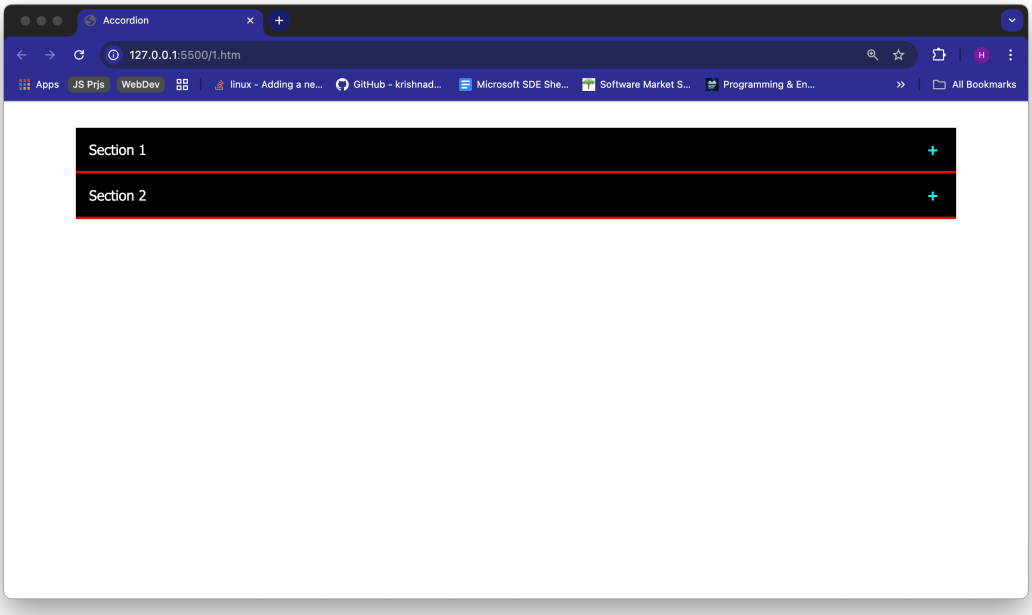
CODE:

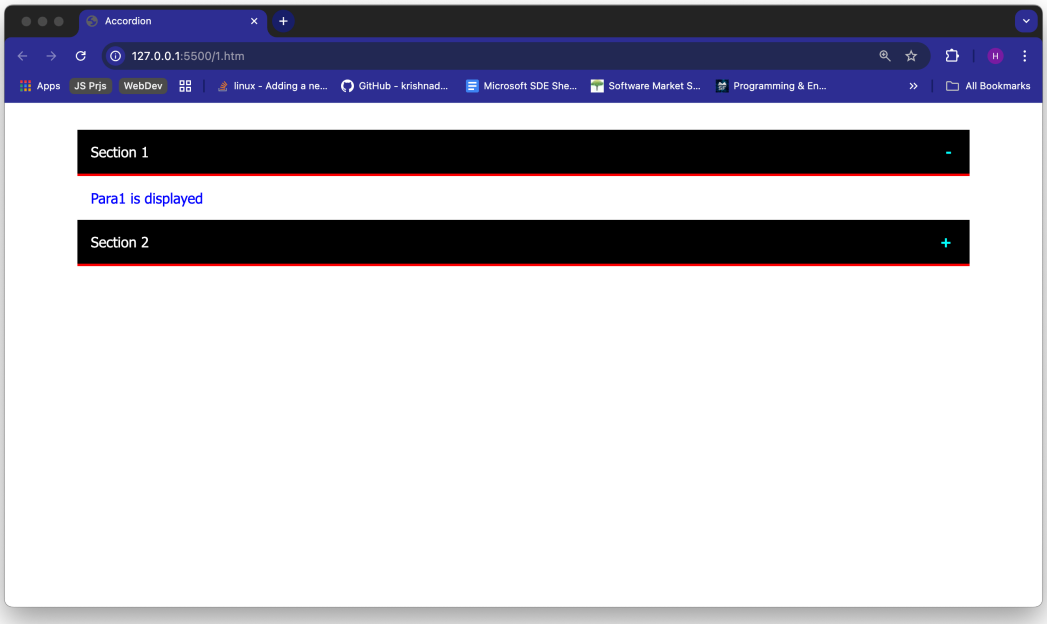
```
<!DOCTYPE html>
<html>
<head>
  <title>Accordion</title>
  <style>

    .accordion {
      width: 90%;
      max-width: 1000px;
      margin: 30px auto;
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    }

    .accordion-item {
      background-color: #eee;
    }

    .accordion-header {
      background-color: black;
      color: white;
      padding: 15px;
      cursor: pointer;
      position: relative;
      border-bottom: 3px solid red;
    }

    .accordion-header::after {
      content: '+';
      position: absolute;
      right: 20px;
      font-weight: bold;
```

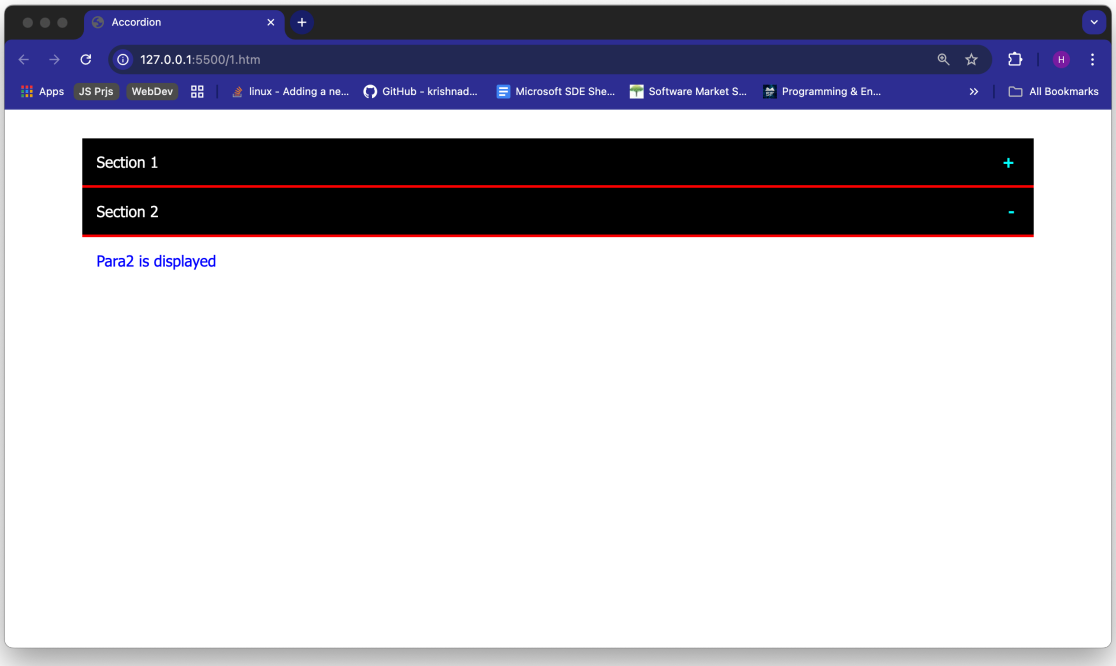
```
      color: cyan;
    }

    .accordion-header.active::after {
      content: '-';
    }

    .accordion-body {
      display: none;
      color: blue;
      padding: 15px;
      background-color: white;
    }
  </style>
</head>
<body>

<div class="accordion">
  <div class="accordion-item">
    <div class="accordion-header" onclick="toggleAccordion(this)">Section 1</div>
    <div class="accordion-body">Para1 is displayed</div>
  </div>
  <div class="accordion-item">
    <div class="accordion-header" onclick="toggleAccordion(this)">Section 2</div>
    <div class="accordion-body">Para2 is displayed</div>
  </div>
</div>

<script>
  function toggleAccordion(element) {
    const body = element.nextElementSibling;
    const isActive = element.classList.contains('active');

    document.querySelectorAll('.accordion-header').forEach(header => {
      header.classList.remove('active');
      const next = header.nextElementSibling;
      if (next && next.classList.contains('accordion-body')) {
        next.style.display = 'none';
      }
    });

    if (!isActive) {
      element.classList.add('active');
      body.style.display = 'block';
    }
  }
</script>
```

```
</body>
</html>
```

OUTPUT:







2. Design the Form using Flex Box with the following input elements Text Fields – Name, Where you born, Email address and with Submit Button.

Solution:

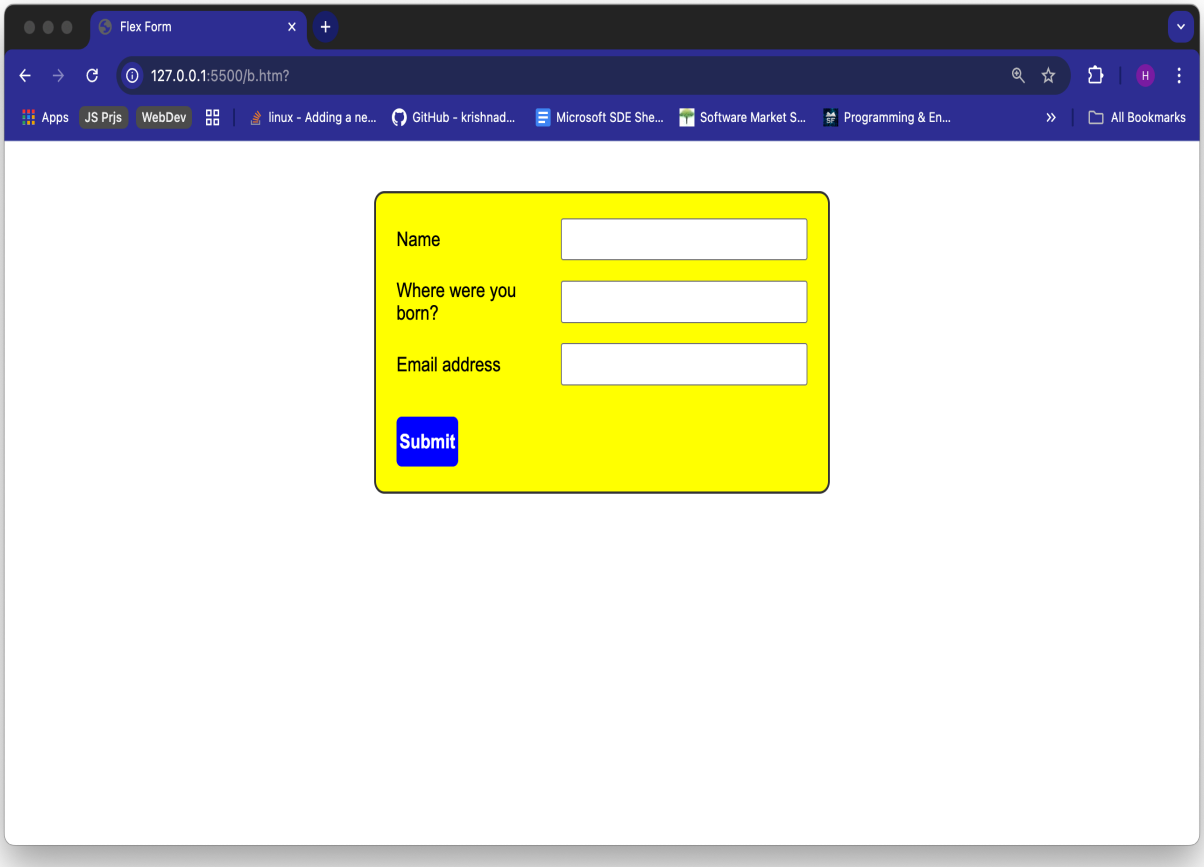
CODE:

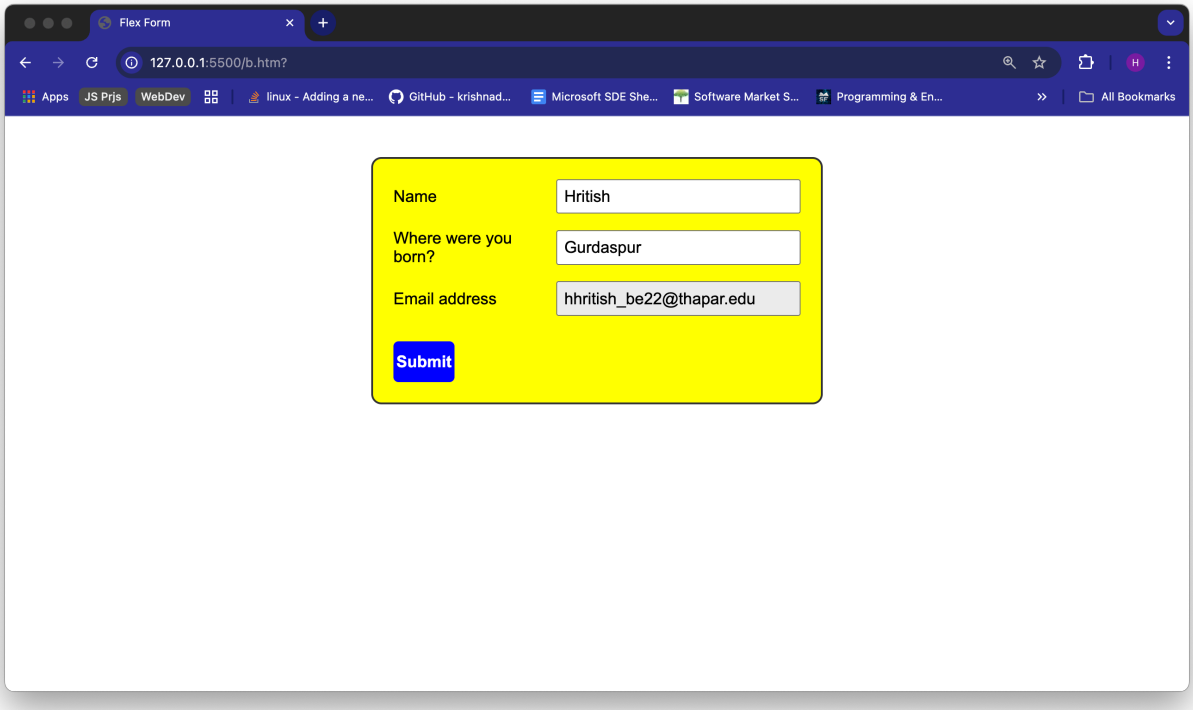
```
<!DOCTYPE html>
<html>

<head>
    <title>Flex Form</title>
    <style>
        body {
            font-family: Arial;
            background-color: white;
        }

        form {
            display: flex;
            flex-direction: column;
            width: 400px;
            margin: 40px auto;
```

```
            padding: 20px;
            background-color: yellow;
            border: 2px solid #333;
            border-radius: 10px;
        }

        .form-row {
            display: flex;
            align-items: center;
            margin-bottom: 15px;
        }

        .form-row label {
            width: 160px;
            margin-bottom: 0;
        }

        .form-row input {
            flex: 1;
            margin-bottom: 0;
            font-size: 16px;
            padding: 6px;
        }

        input[type="submit"] {
            background-color: blue;
            color: white;
            font-weight: bold;
            cursor: pointer;
            font-size: 16px;
            border: none;
            border-radius: 5px;
            width: 60px;
            height: 40px;
            padding: 0;
            margin-top: 10px;
            margin-left: 0;
        }
    </style>
</head>

<body>

    <form>
        <div class="form-row">
            <label for="name">Name</label>
            <input type="text" id="name">
        </div>
```

```
        <div class="form-row">
            <label for="birth">Where were you born?</label>
            <input type="text" id="birth">
        </div>
        <div class="form-row">
            <label for="email">Email address</label>
            <input type="email" id="email">
        </div>
        <input type="submit" value="Submit">
    </form>

</body>

</html>
```

Output:





3. Design the Sub Menu Navigation Bar using CSS by applying the appropriate style rules.
   When the link 2 is hovered, it displays the sub menu links.

   Solution:

   CODE:

```
<!DOCTYPE html>
<html>
<head>
  <title>Sub Menu Navigation Bar</title>
  <style>
    body {
      background-color: grey;
      font-family: Arial;
      margin: 0;
      padding: 0;
    }
```

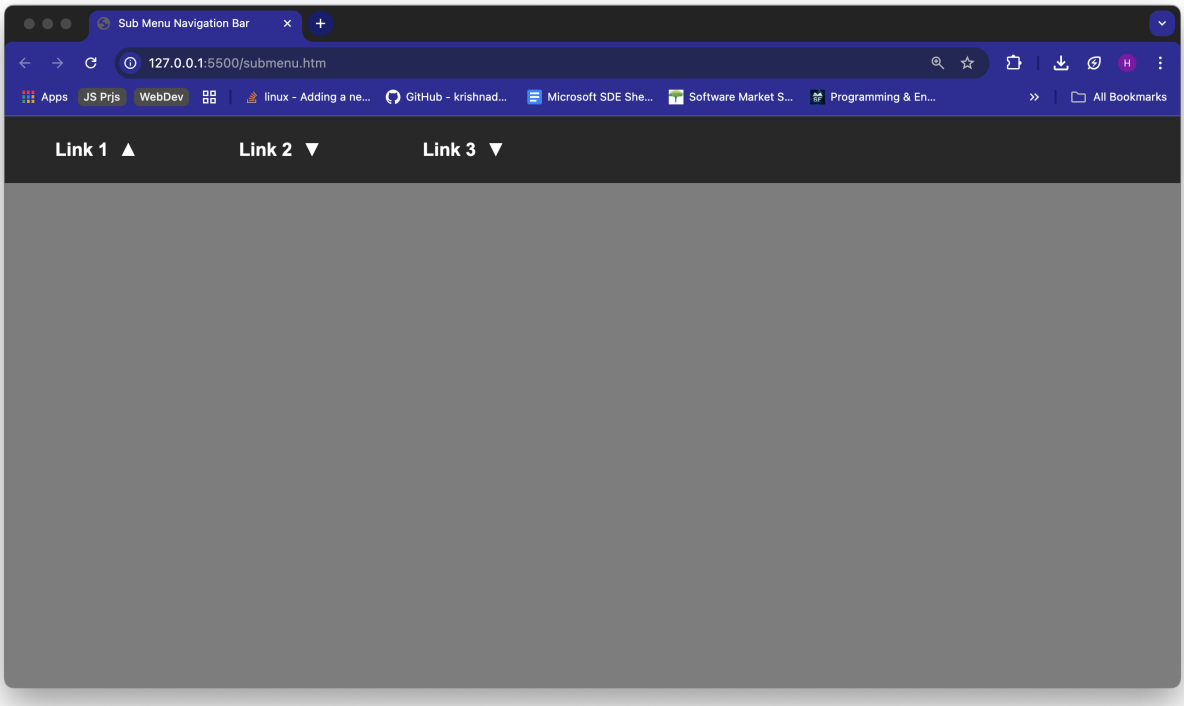
```css
#navbar > ul {
  background-color: #2c2c2c;
  list-style-type: none;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: flex-start;
}

#navbar li {
  position: relative;
}

#navbar a {
  text-decoration: none;
  color: white;
  display: block;
  width: 120px;
  padding: 20px;
  font-size: 16px;
  font-weight: bold;
  text-align: center;
}

#navbar li a:hover {
  background-color: black;
}

#navbar li a.active,
#navbar li a.active:hover {
  background-color: red;
}

.arrow {
  margin-left: 5px;
}

.submenu {
  display: none;
  position: absolute;
  top: 60px;
  background-color: red;
  color: white;
  font-weight: bold;
  padding: 20px;
  text-align: center;
  width: 120px;
```

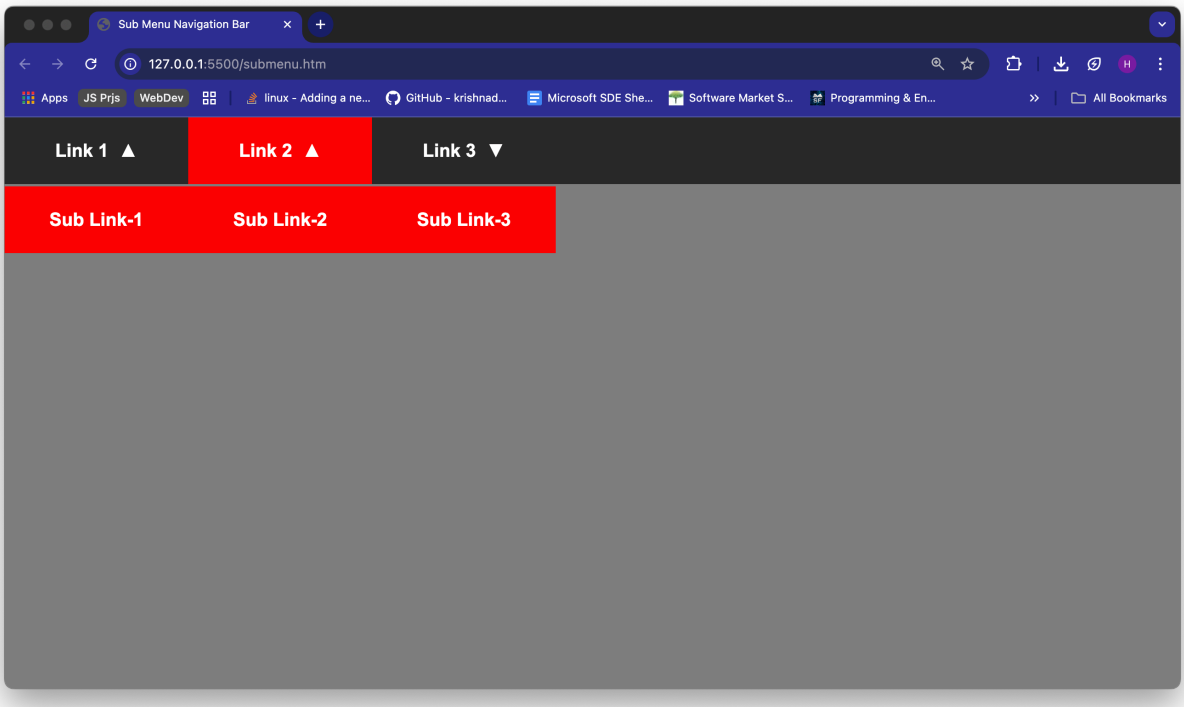
```
    }

    .submenu.show {
      display: block;
    }
  </style>
</head>
<body>

<div id="navbar">
  <ul>
    <li>
      <a href="#" id="l1">Link 1 <span class="arrow" id="a1">▼</span></a>
    </li>
    <li>
      <a href="#" id="l2">Link 2 <span class="arrow" id="a2">▼</span></a>
    </li>
    <li>
      <a href="#" id="l3">Link 3 <span class="arrow" id="a3">▼</span></a>
    </li>
  </ul>
</div>

<div class="submenu" id="s1" style="left: 0;">Sub Link-1</div>
<div class="submenu" id="s2" style="left: 140px;">Sub Link-2</div>
<div class="submenu" id="s3" style="left: 280px;">Sub Link-3</div>


<script>
  const l2 = document.getElementById('l2');
  const s1 = document.getElementById('s1');
  const s2 = document.getElementById('s2');
  const s3 = document.getElementById('s3');
  const a2 = document.getElementById('a2');

  let isVisible = false;

  l2.addEventListener('click', function (e) {
    e.preventDefault();
    isVisible = !isVisible;

    s1.classList.toggle('show', isVisible);
    s2.classList.toggle('show', isVisible);
    s3.classList.toggle('show', isVisible);

    a2.textContent = isVisible ? '▲' : '▼';
    l2.classList.toggle('active', isVisible);
  });
```

```
  document.addEventListener('click', function (e) {
    const isInside = l2.contains(e.target) || s1.contains(e.target) ||
s2.contains(e.target) || s3.contains(e.target);
    if (!isInside) {
      s1.classList.remove('show');
      s2.classList.remove('show');
      s3.classList.remove('show');
      a2.textContent = '▼';
      l2.classList.remove('active');
      isVisible = false;
    }
  });

  document.getElementById('l1').addEventListener('click', function(e) {
    e.preventDefault();
    const a1 = document.getElementById('a1');
    a1.textContent = a1.textContent === '▼' ? '▲' : '▼';
  });

  document.getElementById('l3').addEventListener('click', function(e) {
    e.preventDefault();
    const a3 = document.getElementById('a3');
    a3.textContent = a3.textContent === '▼' ? '▲' : '▼';
  });
</script>

</body>
</html>
```

Output:

Hritish 102218050 BS2



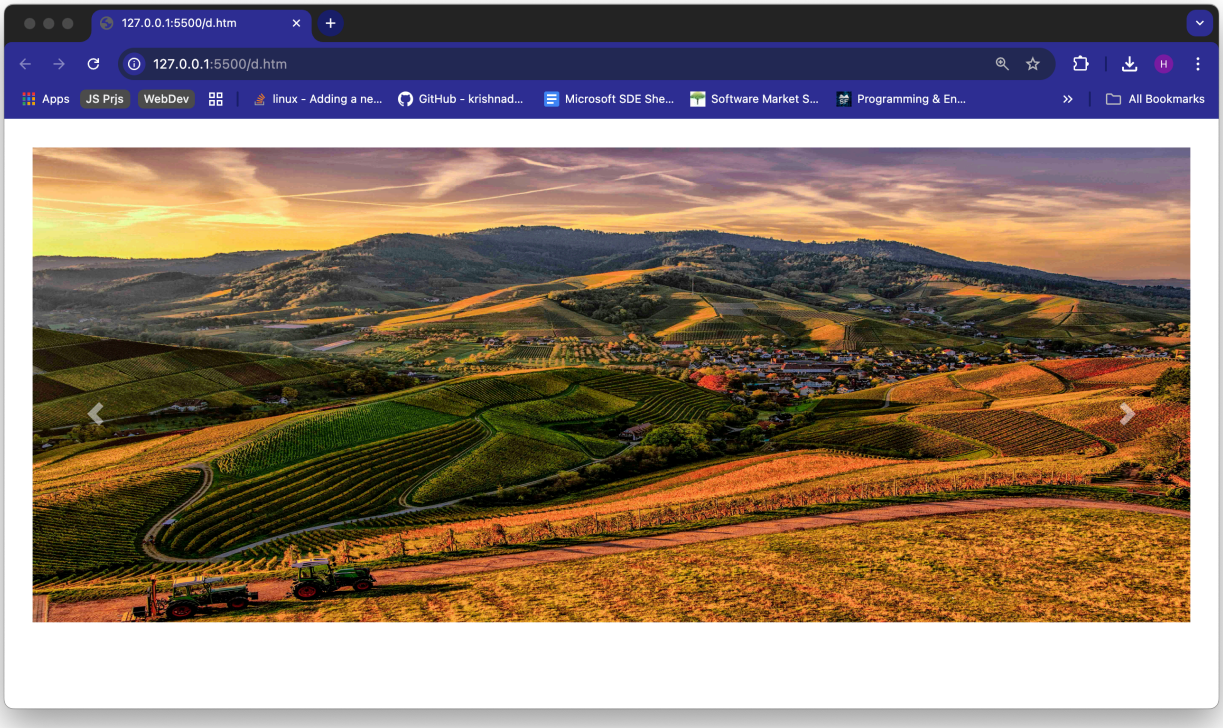


4. Develop the Carousel using BootStrap 4.6

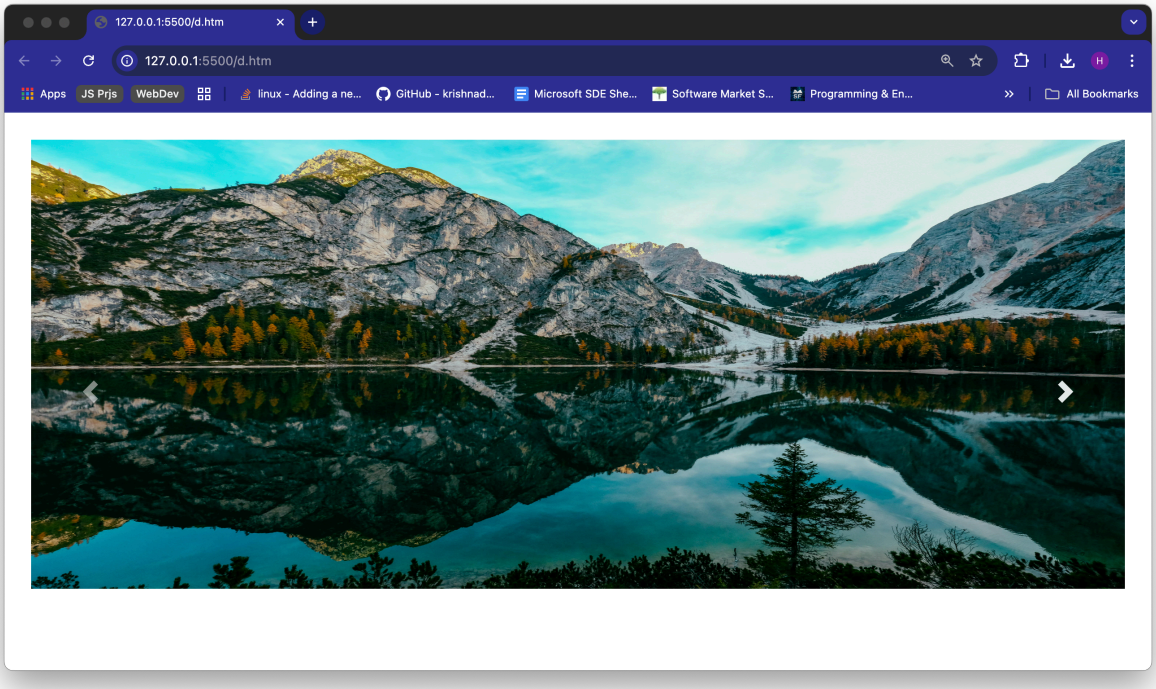
```
<html>

<head>
    <title></title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css"
        integrity="sha384-
xOolHFLEh07PJGoPkLv1IbcEPTNtaed2xpHsD9ESMhqIYd0nLMwNLD69Npy4HI+N"
crossorigin="anonymous">
</head>

<body>
    <div class="carousel mt-4 mx-4" data-ride="carousel" id="slide">
        <div class="carousel-inner">

            <div class="carousel-item active">
                <img src="./a.jpg" width="100%" height="400px">

            </div>
            <div class="carousel-item">
                <img src="./b.jpg" width="100%" height="400px">
            </div>
            <div class="carousel-item">
                <img src="./c.jpg" width="100%" height="400px">
            </div>

        </div>
    </div>

    <button class="carousel-control-prev" data-target="#slide" data-slide="prev">
        <span class="carousel-control-prev-icon"> </span>
    </button>

    <button class="carousel-control-next" data-target="#slide" data-slide="next">
        <span class="carousel-control-next-icon"></span>
    </button>
    <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js"
```

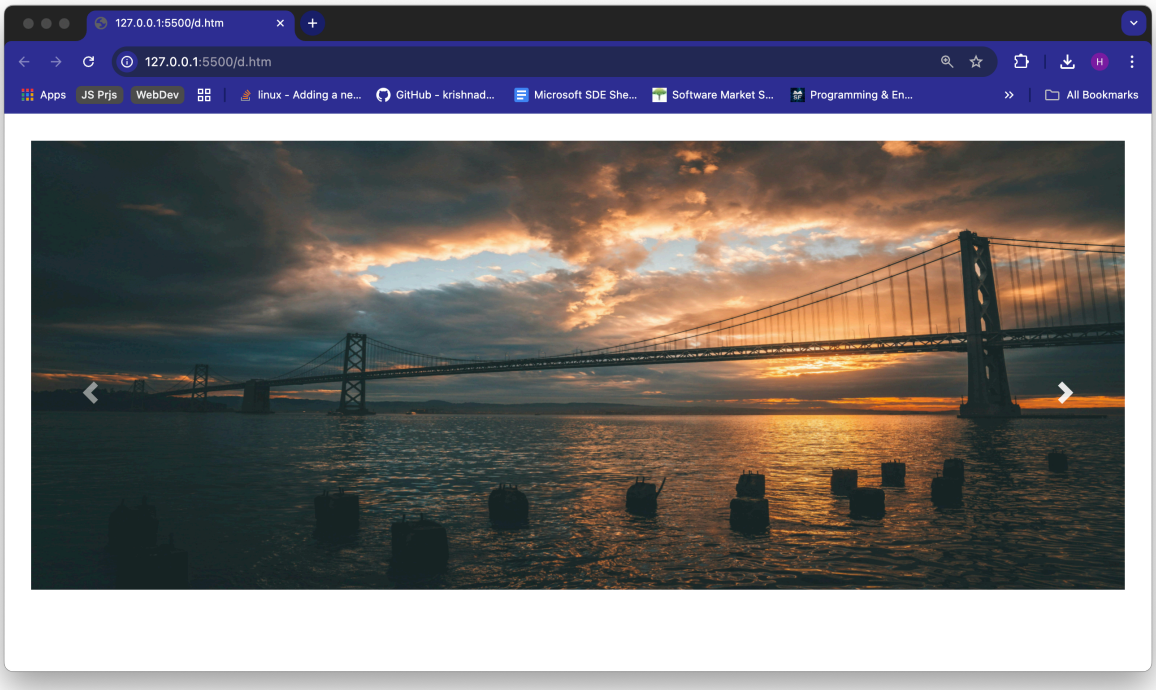
```
        integrity="sha384-
Fy6S3B9q64WdZWQUiU+q4/2Lc9npb8tCaSX9FK7E8HnRr0Jz8D6OP9dO5Vg3Q9ct"
        crossorigin="anonymous"></script>

</body>

</html>
```

Output:



Hritish **102218050** **BS2**





Here's a link attached to demonstrate the working of the carousel:

Link

5. How the Media Queries are implemented in CSS with their Significance.

Solution:

Media queries in CSS are a way to make your website adapt to different devices and screen sizes, which is essential for creating a responsive design. In simple terms, media queries let you apply specific CSS styles only when certain conditions are true, such as when someone is viewing your site on a phone, tablet, or desktop.

media queries use the `@media` rule. This allows you to target styles based on factors like screen width, height, orientation, and resolution. For example, you might want to change the layout, font size, or background color when the screen is less than 600 pixels wide, so your site remains readable and easy to use on smaller devices.

```
    @media only screen and (max-width: 600px) {
            body {
                background-color: lightblue;
            }
        }
```

The main significance of media queries is that they make responsive web design possible. Without them, your website might look great on a desktop but become difficult to navigate or read on a smartphone. With media queries, you can adjust everything from layout and font sizes to navigation menus and images, ensuring your site is flexible and user-friendly no matter how someone accesses it. For instance, you can use media queries to stack navigation links vertically on small screens, or to hide certain elements that aren't needed on mobile devices.

```
@media (max-width: 768px) {
            .navbar {
                flex-direction: column;
            }

            .menu-item {
                font-size: 18px;
            }
        }
```

In this case, when the screen is 768 pixels wide or less (like on a tablet or phone), the navigation bar switches to a vertical layout and the menu items become larger for easier tapping.