NAME: Kartikey Sharma
ROLL NO. CSC/21/18
21059570020

IT ASSIGNMENT 2

Q1)Define the Navbar and Pagination components of Bootstrap. Give two advantages of using Bootstrap ?

ANS)

Navbar Component:

- The Bootstrap Navbar is a responsive and customizable navigation bar that helps create a consistent and user-friendly navigation structure for web applications.

Pagination Component:

- The Bootstrap Pagination component is used to divide content or data into multiple pages, providing an organized way to navigate through large datasets.

Advantages of Using Bootstrap:

- **Responsive Design:** Bootstrap ensures that web applications are mobile-friendly and adapt seamlessly to various screen sizes and devices.
- **Consistent Styling:** Bootstrap provides a consistent and well-designed set of styles and components, ensuring a cohesive and professional appearance across different parts of a website or web application.

Q2)Why do we need AJAX in web applications? How the browser handles AJAX Requests and Responses?

ANS)

- AJAX (Asynchronous JavaScript and XML) is used in web applications to enable asynchronous data retrieval from a server without requiring a full page reload. This allows for dynamic and interactive user experiences, as data can be fetched and updated in the background, enhancing responsiveness.

How Browser Handles AJAX Requests and Responses:

- When an AJAX request is made, the browser sends the request asynchronously to the server using JavaScript. The server processes the request and sends back a response, typically in JSON or XML format. The browser handles this response asynchronously, and the JavaScript callback function associated with the AJAX request processes and updates the webpage without requiring a full page reload.

Q3. Create an HTML file with a list of items, such as <ul> with multiple <li> elements. Use jQuery syntax to perform the following tasks:

(i) Select the first <li> element and change its text color to blue and background colour to yellow.

(ii) Select the last <li> element and change its background color to pink.

(iii) Select all <li> elements whose class attribute has a value of hot to add a new class name called cool.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>jQuery</title>
  <script src="https://code.jquery.com/jquery-
3.6.4.min.js"></script>
```

```
<style>
  body {
    font-family: 'Arial', sans-serif;
    margin: 20px;
    text-align: center;
  }

  ul {
    list-style-type: none;
    padding: 0;
  }

  li {
    padding: 15px;
    margin: 10px;
    border: 1px solid #4CAF50;
    background-color: #f8f8f8;
    color: #333;
  }

  .hot {
    background-color: #FF6347;
    color: #fff;
  }

  .cool {
```

```
  background-color: #3498db;
  color: #fff;
}

#applyChangesBtn {
  padding: 10px;
  background-color: #4CAF50;
  color: #fff;
  border: none;
  cursor: pointer;
  font-size: 16px;
}

p {
  font-size: 18px;
  margin-bottom: 20px;
}
ul {
  list-style-type: none;
  padding: 0;
}

li {
  padding: 10px;
  margin: 5px;
  border: 1px solid #ccc;
}
```

```
  </style>
</head>
<body>
    <p>
        >> What "Apply Changes" will Do :-<br><br>

      >  Select the first element(Office):<br>
         Change its text color to blue.<br>
         Change its background color to yellow.<br><br>

      >  Select the last  element(Home):<br>
         Change its background color to pink.<br><br>

      >  Select all elements with the class "hot":<br>
         Add a new class name "cool" to each of these
elements.</p>

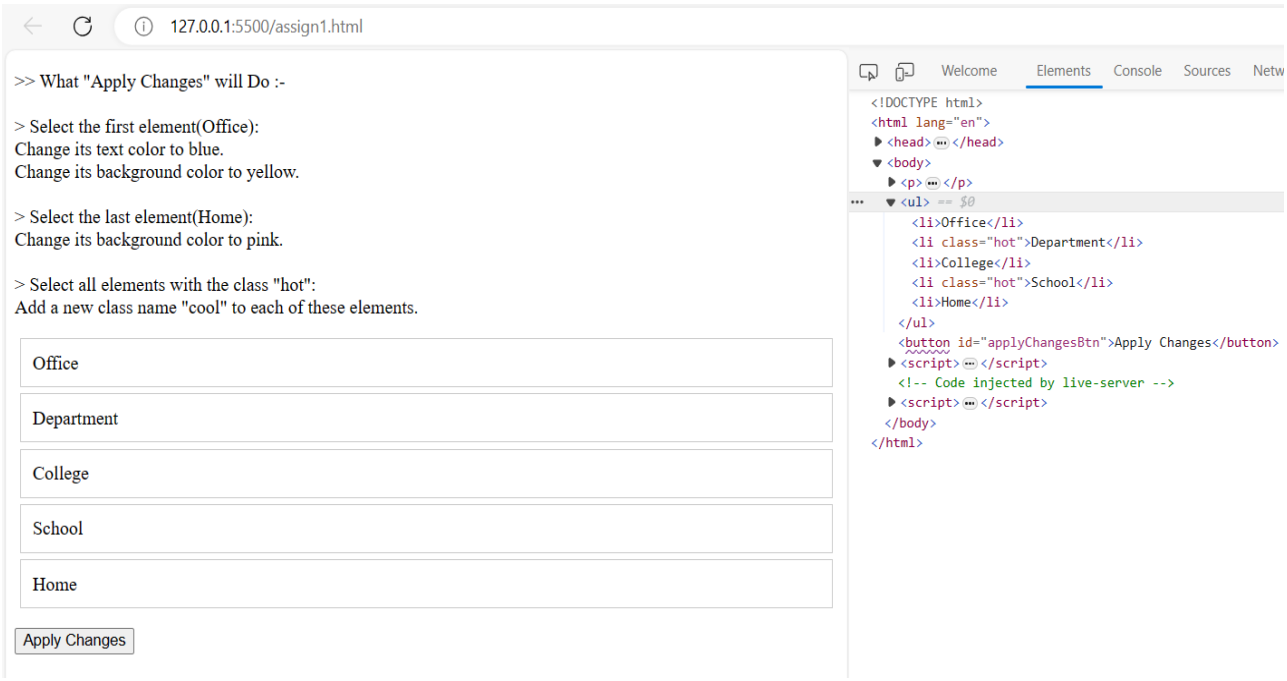
<ul>
    <li>Office</li>
    <li class="hot">Department</li>
    <li>College</li>
    <li class="hot">School</li>
    <li>Home</li>
</ul>

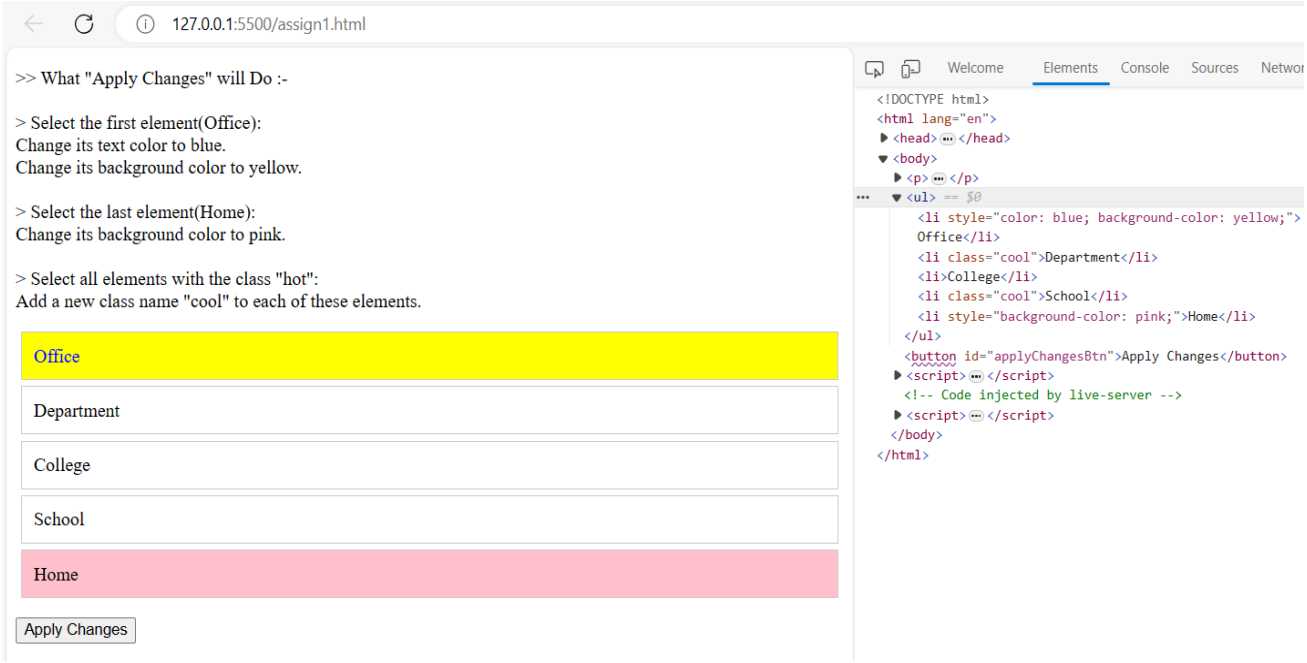
<button id="applyChangesBtn">Apply Changes</button>

<script>
```

```
  $(document).ready(function() {
    function applyChanges() {
      $('li:first').css({
        'color': 'blue',
        'background-color': 'yellow'
      });
      $('li:last').css('background-color', 'pink');
      $('li.hot').removeClass('hot').addClass('cool');
    }
    $('#applyChangesBtn').click(function() {
      applyChanges();
    });
  });
</script>


</body>
</html>
```



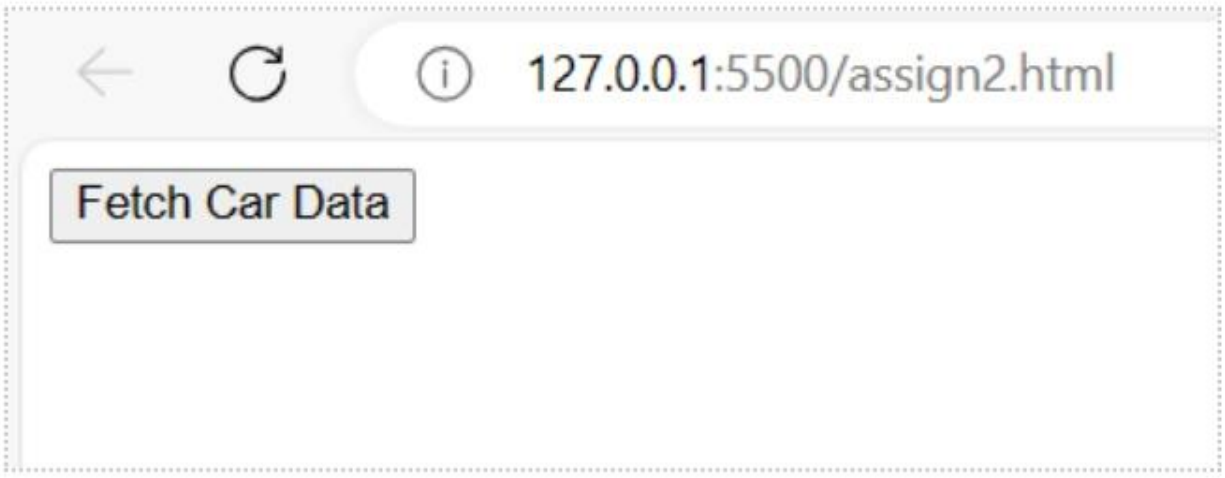


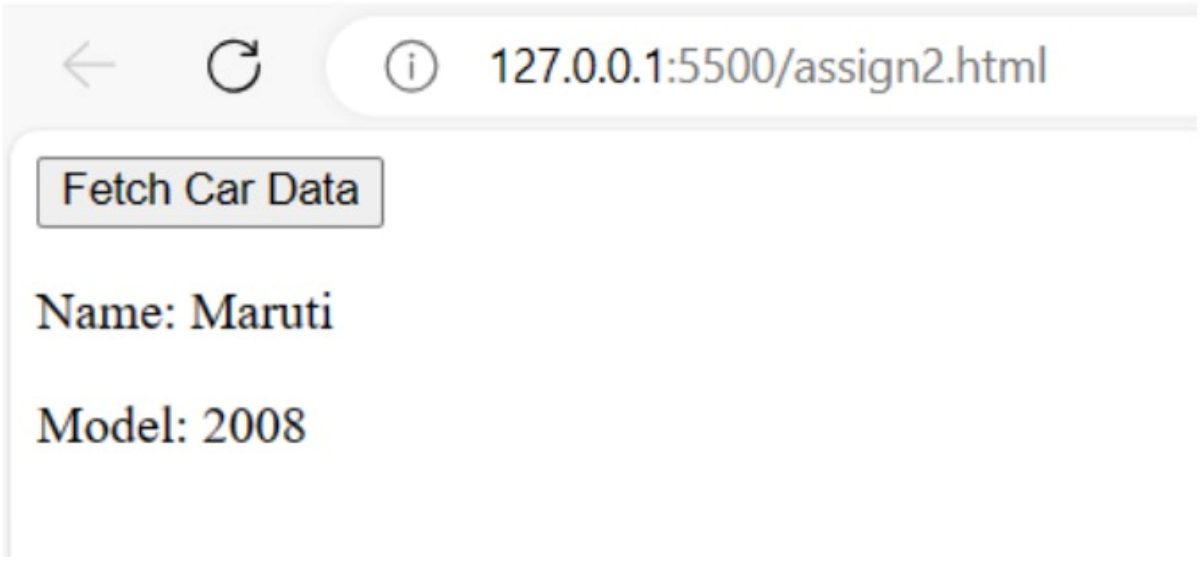
Q4. Create a JSON object that represents information about a Car with the following fields: "name", "model" . Create an HTML button that triggers an AJAX request to fetch this JSON data from a server or a local file when clicked. Provide the necessary JavaScript code to handle the onclick event and display the retrieved JSON data on the web page.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title> Car Data</title>
  <script src="https://code.jquery.com/jquery-
3.6.4.min.js"></script>
</head>
<body>
<button id="fetchDataBtn">Fetch Car Data</button>
<div id="carInfo"></div>
<script>
  $(document).ready(function() {
    $('#fetchDataBtn').click(function() {
      $.ajax({
        url: 'car.json',
        method: 'GET',
        dataType: 'json',
        success: function(data) {

          displayCarInfo(data);
        },
        error: function(error) {
          console.error('Error fetching data:', error);
        }
      });
    });
```

```
    function displayCarInfo(carData) {
      $('#carInfo').html(`<p>Name: ${carData.name}</p><p>Model: ${carData.model}</p>`);
    }
  });
</script>
</body>
</html>
```





Q5. Write the difference between innerHTML, textContent and innerText property of the JavaScript.

ANS)

- **innerHTML:** Represents the HTML content within an element and allows both HTML and text to be set or retrieved.
- **textContent:** Represents only the text content of an element and does not interpret HTML tags as elements.
- **innerText:** Similar to textContent, but it may produce different results in terms of spacing and line breaks due to its awareness of CSS styles affecting the visual presentation. It is not supported in Firefox.

Q6) What are event listeners? Why it is used. Explain it with an example.

ANS)

Event Listeners:

- Event listeners are functions in JavaScript that wait for a specific event to occur on an HTML element and then execute a predefined set of instructions or code.

Why They Are Used:

- Event listeners are used to add interactivity to web pages. They enable developers to respond to user actions (such as clicks, mouse movements, or keyboard inputs) and trigger appropriate actions or functions.

Example:

- In the context of a button click event listener:

```
const myButton = document.getElementById('myButton');

myButton.addEventListener('click', function() {

  alert('Button clicked!');
});
```

- In this example, the event listener waits for the "click" event on the button with the id "myButton" and, when triggered, displays an alert saying "Button clicked!".

Q7) . Differentiate between anonymous function, function expression and immediately invoked function expression in JavaScript with the help of an example.

ANS)

Anonymous Function:

- An anonymous function is a function without a name. It can be defined using the **`function`** keyword without providing a function name. It is often used in scenarios where the function is used only once.

```
// Example of an anonymous function
const sum = function(a, b) {
  return a + b;
};
```

Function Expression:

- A function expression is a way to define a function as part of an expression. It can be named or anonymous and can be assigned to a variable. Function expressions are often used in situations where functions are treated as values.
-

```
// Example of a named function expression
const multiply = function multiply(a, b) {
  return a * b;
};

// Example of an anonymous function expression
const divide = function(a, b) {
  return a / b;
};
```

Immediately Invoked Function Expression (IIFE):

- An IIFE is a function expression that is defined and immediately invoked. It is wrapped in parentheses to turn it into an expression, followed by an additional set of parentheses to invoke it immediately.

```
// Example of an IIFE
(function() {
  console.log('I am immediately invoked!');
})();
```

- Anonymous functions lack a name.
- Function expressions can be named or anonymous and are often used when functions are treated as values.
- IIFE is a function expression that is immediately invoked, providing a way to create a private scope and execute code immediately.

Q8) . Write the steps used in DNS operations.

ANS) **DNS (Domain Name System) Operations Steps:**

- **User Input or Application Request:**

A user or an application initiates a DNS request by entering a domain name (like www.example.com) in a web browser or making a network request.

- **Local DNS Cache Check:**

The local DNS resolver checks its cache to see if it already has the corresponding IP address for the requested domain. If found, it skips further steps and uses the cached information.

- **Recursive DNS Server Query:**

If the information is not in the local cache, the resolver queries a recursive DNS server. This server may have its own cache or continue the resolution process by contacting authoritative DNS servers.

- **Root DNS Server Query:**

If the recursive DNS server does not have the information, it contacts a root DNS server. The root server provides information about top-level domain (TLD) servers.

- **TLD DNS Server Query:**

The recursive server contacts the TLD server based on the top-level domain of the requested domain (e.g., ".com" for www.example.com). The TLD server provides information about the authoritative DNS server for the next level.

- **Authoritative DNS Server Query:**

The recursive server queries the authoritative DNS server, which holds the specific DNS records for the requested domain.

- **Response to Recursive Server:**

The authoritative server responds to the recursive DNS server with the requested DNS information, including the IP address associated with the domain.

- **Response to Local DNS Resolver:**

The recursive DNS server sends the IP address information back to the local DNS resolver that initiated the request.

- **Response to User or Application:**

Finally, the local DNS resolver responds to the user or application with the IP address. If the resolver supports caching, it may also cache the information for future use.

- **Caching:**

At various points in the process, DNS servers may cache information to improve performance and reduce the need for repeated queries. Cached information is used when available, but it has a limited time-to-live (TTL) before it needs to be refreshed.

Q9) . Create an HTML form in a file called car.html to get the following details of the car:

- Name of the manufacturer
- Name of the model
- Manufacturing Year
- Fuel type (petrol/diesel)
- Color
- Seating capacity
- Cubic capacity

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Car Details Form</title>
  <style>
body {
  font-family: 'Roboto', Arial, sans-serif;
  background-color: #f0f0f0;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}


form {
  background-color: #fff;
  border-radius: 10px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
  padding: 30px;
  width: 300px;
  text-align: center;
}


h2 {
  color: #333;
}
```

```
label {
  display: block;
  margin-bottom: 8px;
  color: #555;
}


input,
select,
button {
  width: 100%;
  padding: 12px;
  margin-bottom: 20px;
  border: 1px solid #ddd;
  border-radius: 6px;
  box-sizing: border-box;
  font-size: 14px;
  transition: border-color 0.3s, box-shadow 0.3s;
}


input:focus,
select:focus {
  border-color: #4caf50;
  box-shadow: 0 0 8px rgba(76, 175, 80, 0.3);
}
```

```
select {
  appearance: none;
}


button {
  background-color: #4caf50;
  color: #fff;
  cursor: pointer;
  border: none;
  border-radius: 6px;
  padding: 12px;
  font-size: 16px;
  transition: background-color 0.3s;
}


button:hover {
  background-color: #45a049;
}



  </style>
</head>
<body>


<h2>Car Details Form</h2>
```

```
<form id="carDetailsForm">
  <label for="manufacturer">Manufacturer:</label>
  <input type="text" id="manufacturer" name="manufacturer"
required>


  <label for="model">Model:</label>
  <input type="text" id="model" name="model" required>


  <label for="manufacturingYear">Manufacturing Year:</label>
  <input type="number" id="manufacturingYear"
name="manufacturingYear" min="1900" max="2100" required>


  <label for="fuelType">Fuel Type:</label>
  <select id="fuelType" name="fuelType" required>
    <option value="petrol">Petrol</option>
    <option value="diesel">Diesel</option>
  </select>


  <label for="color">Color:</label>
  <input type="text" id="color" name="color" required>


  <label for="seatingCapacity">Seating Capacity:</label>
  <input type="number" id="seatingCapacity"
name="seatingCapacity" min="1" required>


  <label for="cubicCapacity">Cubic Capacity:</label>
```

```
  <input type="number" id="cubicCapacity"
name="cubicCapacity" min="1" required>

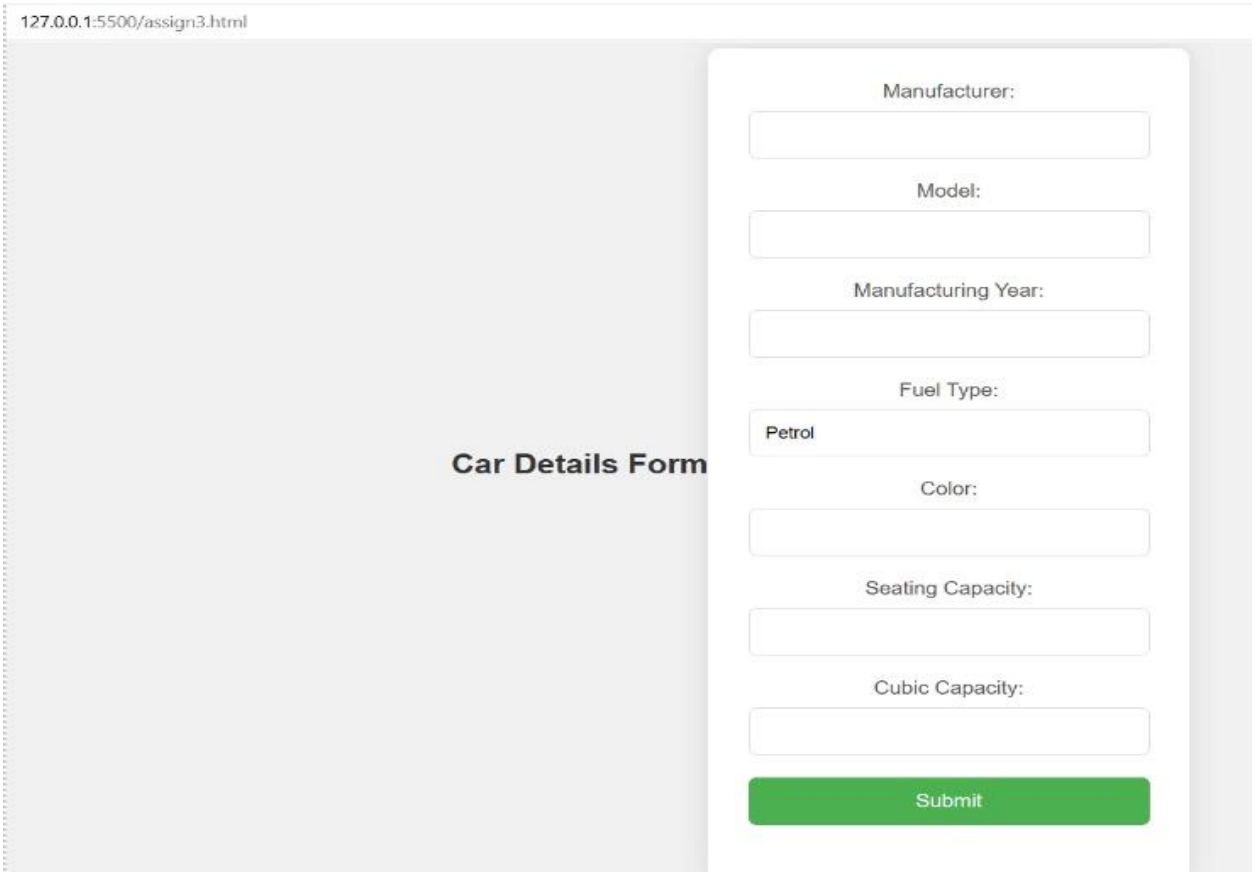


  <button type="button"
onclick="submitForm()">Submit</button>
</form>



<script>
  function submitForm() {
    const manufacturer =
document.getElementById('manufacturer').value;
    const model = document.getElementById('model').value;
    const manufacturingYear =
document.getElementById('manufacturingYear').value;
    const fuelType =
document.getElementById('fuelType').value;
    const color = document.getElementById('color').value;
    const seatingCapacity =
document.getElementById('seatingCapacity').value;
    const cubicCapacity =
document.getElementById('cubicCapacity').value;
    const carDetails = {
      "manufacturer": manufacturer,
      "model": model,
      "manufacturingYear": manufacturingYear,
      "fuelType": fuelType,
      "color": color,
      "seatingCapacity": seatingCapacity,
      "cubicCapacity": cubicCapacity
    };
```

```
    console.log(carDetails);
  }
</script>
</body>
</html>
```



Q10) Write JQuery code in a JavaScript file car.js to get the details of the car from the car.html on pressing submit button (in Question 9 above). Also write the JavaScript code in car.js to make a JavaScript object and JSON object from the above details and print both the JavaScript and JSON object on the console.

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <script src="https://code.jquery.com/jquery-
3.6.4.min.js"></script>
  <script src="car.js"></script>
  <title>Car Details Form</title>
  <style>
body {
  font-family: 'Roboto', Arial, sans-serif;
  background-color: #f0f0f0;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}


form {
  background-color: #fff;
  border-radius: 10px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
  padding: 30px;
  width: 300px;
  text-align: center;
}
```

```
h2 {
  color: #333;
}


label {
  display: block;
  margin-bottom: 8px;
  color: #555;
}


input,
select,
button {
  width: 100%;
  padding: 12px;
  margin-bottom: 20px;
  border: 1px solid #ddd;
  border-radius: 6px;
  box-sizing: border-box;
  font-size: 14px;
  transition: border-color 0.3s, box-shadow 0.3s;
}


input:focus,
select:focus {
```

```
  border-color: #4caf50;
  box-shadow: 0 0 8px rgba(76, 175, 80, 0.3);
}

select {
  appearance: none;
}

button {
  background-color: #4caf50;
  color: #fff;
  cursor: pointer;
  border: none;
  border-radius: 6px;
  padding: 12px;
  font-size: 16px;
  transition: background-color 0.3s;
}

button:hover {
  background-color: #45a049;
}


  </style>
</head>
<body>
```

```
<h2>Car Details Form</h2>


<form id="carDetailsForm">
  <label for="manufacturer">Manufacturer:</label>
  <input type="text" id="manufacturer" name="manufacturer"
required>


  <label for="model">Model:</label>
  <input type="text" id="model" name="model" required>


  <label for="manufacturingYear">Manufacturing Year:</label>
  <input type="number" id="manufacturingYear"
name="manufacturingYear" min="1900" max="2100" required>


  <label for="fuelType">Fuel Type:</label>
  <select id="fuelType" name="fuelType" required>
    <option value="petrol">Petrol</option>
    <option value="diesel">Diesel</option>
  </select>


  <label for="color">Color:</label>
  <input type="text" id="color" name="color" required>


  <label for="seatingCapacity">Seating Capacity:</label>
```

```
  <input type="number" id="seatingCapacity"
name="seatingCapacity" min="1" required>



  <label for="cubicCapacity">Cubic Capacity:</label>
  <input type="number" id="cubicCapacity"
name="cubicCapacity" min="1" required>



  <button type="button"
onclick="submitForm()">Submit</button>
</form>



<script>
  function submitForm() {
    const manufacturer =
document.getElementById('manufacturer').value;
    const model = document.getElementById('model').value;
    const manufacturingYear =
document.getElementById('manufacturingYear').value;
    const fuelType =
document.getElementById('fuelType').value;
    const color = document.getElementById('color').value;
    const seatingCapacity =
document.getElementById('seatingCapacity').value;
    const cubicCapacity =
document.getElementById('cubicCapacity').value;
    const carDetails = {
      "manufacturer": manufacturer,
      "model": model,
      "manufacturingYear": manufacturingYear,
      "fuelType": fuelType,
```

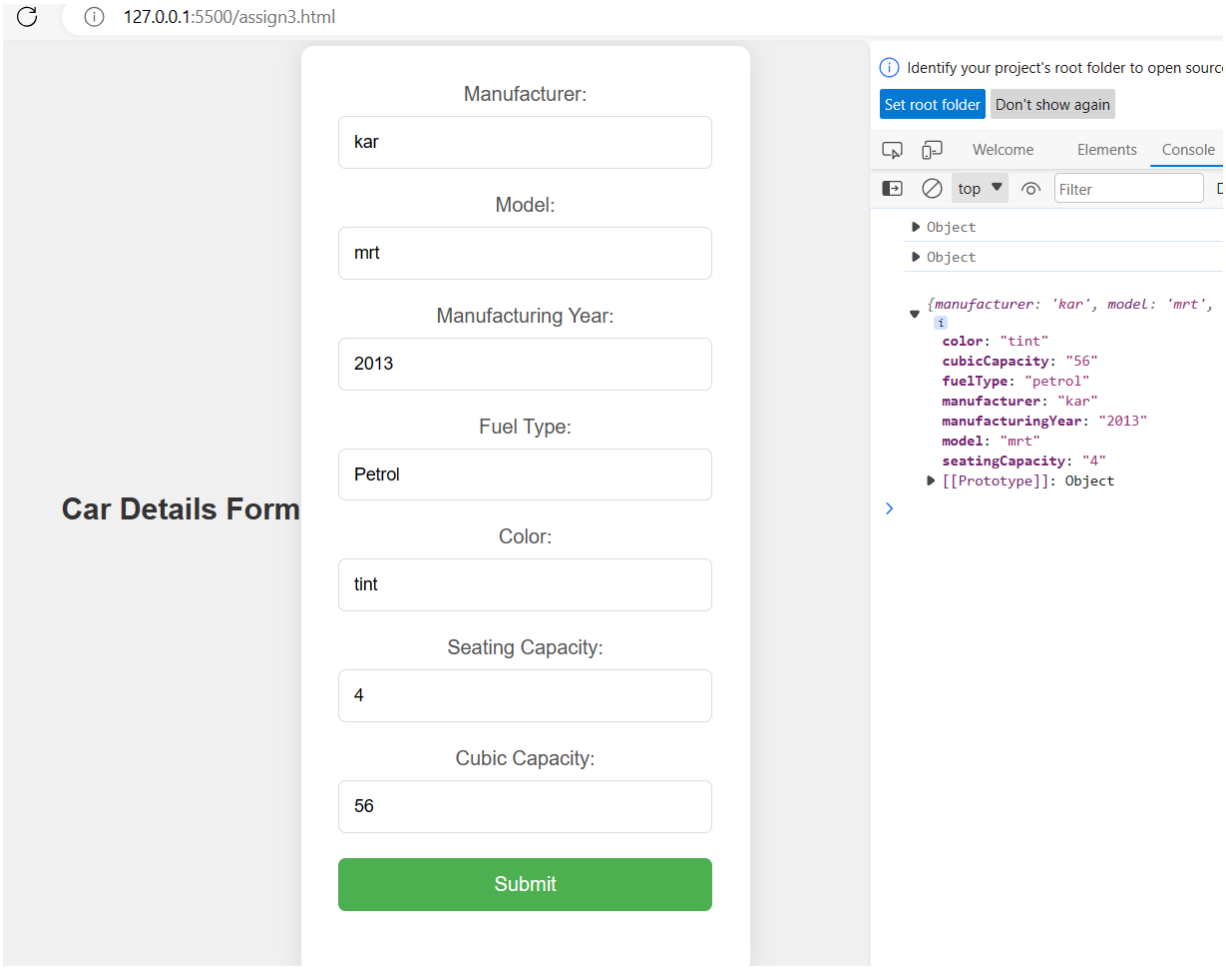
```
      "color": color,
      "seatingCapacity": seatingCapacity,
      "cubicCapacity": cubicCapacity
    };
    console.log(carDetails);
  }
</script>


</body>
</html>
```

JS :-

```
$(document).ready(function() {
    $('#submitBtn').click(function() {
      const manufacturer = $('#manufacturer').val();
      const model = $('#model').val();
      const manufacturingYear =
$('#manufacturingYear').val();
      const fuelType = $('#fuelType').val();
      const color = $('#color').val();
      const seatingCapacity = $('#seatingCapacity').val();
      const cubicCapacity = $('#cubicCapacity').val();
      const carDetailsJS = {
        manufacturer: manufacturer,
        model: model,
        manufacturingYear: manufacturingYear,
        fuelType: fuelType,
        color: color,
        seatingCapacity: seatingCapacity,
```

```
        cubicCapacity: cubicCapacity
      };
      const carDetailsJSON = JSON.stringify(carDetailsJS);
      console.log('JavaScript Object:', carDetailsJS);
      console.log('JSON Object:', carDetailsJSON);
    });
  });
```



Q11) . Describe the following JavaScript methods with examples:

(i) getElementByld()

(ii) querySelector()

(iii) querySelectorAll()

(iv) Math.random()

ANS)

(i) getElementById():

- **Description: `getElementById()`** is a method that returns the element with the specified ID attribute.
- **Example:**
- ```
  <html>
  <body>

  This is a paragraph.

  <script>
    // Get the element with the ID "demo"
    var element = document.getElementById("demo");

    // Change the content of the element
    element.innerHTML = "Hello, JavaScript!";
  </script>

  </body>
  </html>
  ```

(ii) querySelector():

- **Description: `querySelector()`** is a method that returns the first element that matches a specified CSS selector in the document.
- **Example:**
- ```
  <html>
  <body>

  <p class="example">This is a paragraph.</p>

  <script>
    // Get the first element with the class "example"
    var element = document.querySelector(".example");
  ```

```
  // Change the content of the element
  element.innerHTML = "Hello, JavaScript!";
</script>

</body>
</html>
```

(iii) querySelectorAll():

- **Description:** **`querySelectorAll()`** is a method that returns a NodeList representing a list of the document's elements that match the specified group of selectors.
- **Example:**
- ```
  <html>
  <body>

  This is the first paragraph.
  This is the second paragraph.

  <script>
    // Get all elements with the class "example"
    var elements = document.querySelectorAll(".example");

    // Change the content of each element
    elements.forEach(function(element) {
      element.innerHTML = "Hello, JavaScript!";
    });
  </script>

  </body>
  </html>
  ```

(iv) Math.random():

- **Description:** **`Math.random()`** is a method that returns a pseudo-random floating-point number between 0 (inclusive) and 1 (exclusive).
- **Example:**
  ```

```
// Generate a random number between 0 (inclusive) and 1 (exclusive)
var randomNumber = Math.random();

console.log(randomNumber);
```

Q12) Explain two methods used to send http request parameters to server and how does the structure of request objects vary in both cases.

ANS)
Two common methods used to send HTTP request parameters to a server are:

GET Method:

- **Method:** Parameters are appended to the URL.
- **Structure:** Parameters are part of the URL query string.
- **Example:** **`https://example.com/api?param1=value1&param2=value2`**
- **Request Object Structure:** There is no separate request object. Parameters are part of the URL and can be accessed by parsing the URL on the server.

POST Method:

- **Method:** Parameters are sent in the body of the HTTP request.
- **Structure:** Parameters are sent as part of the request body, typically in key-value pairs.
- **Example:** Request body in the format **`param1=value1&param2=value2`**
- **Request Object Structure:** In server-side languages like Node.js or Python, the server can parse the request body to retrieve the parameters.

```
// Example using Node.js with Express for handling POST requests

const express = require('express');
const bodyParser = require('body-parser');
const app = express();

// Middleware to parse request body
```

```
app.use(bodyParser.urlencoded({ extended: false }));

// POST endpoint
app.post('/api', (req, res) => {
  const param1 = req.body.param1;
  const param2 = req.body.param2;

  // Process parameters
  console.log(param1, param2);

  // Send response
  res.send('Received POST request');
});

// Start the server
const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

With the GET method, parameters are part of the URL, while with the POST method, parameters are sent in the request body. The server-side code needs to handle the request appropriately to extract and process these parameters.