Vamshi Krishna Mummadi

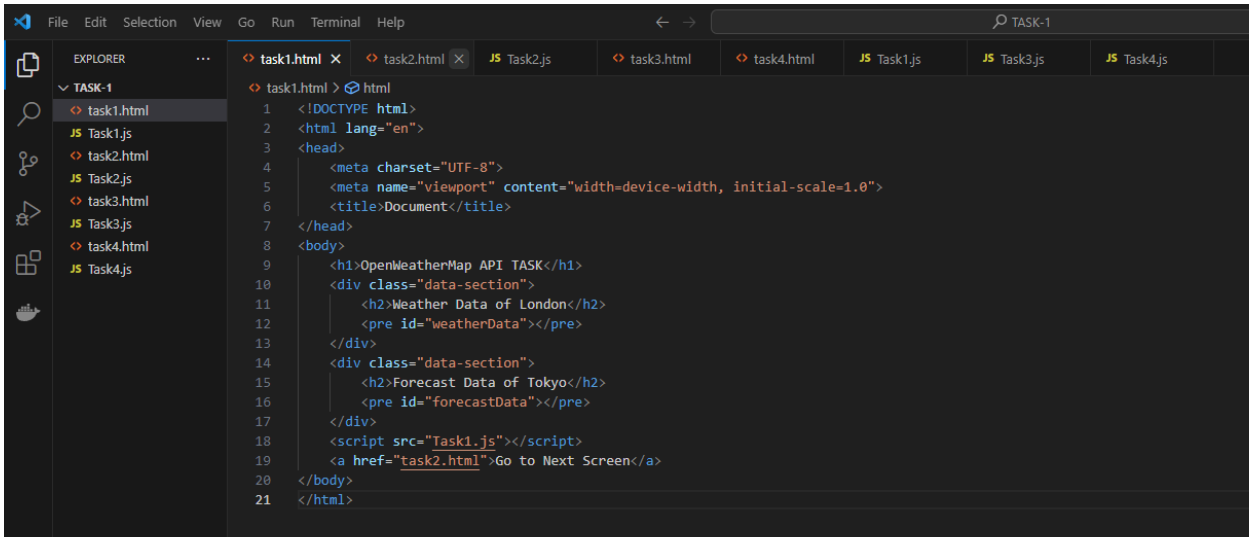
API scavenger hunt Assignment

Task-1:

My experience with the OpenWeatherMap API has been quite positive. The API is relatively easy to use, and it provides a wide range of weather-related data, making it a valuable resource for developers. The documentation is comprehensive and well-organized, which greatly aids in understanding how to make API requests and work with the responses.

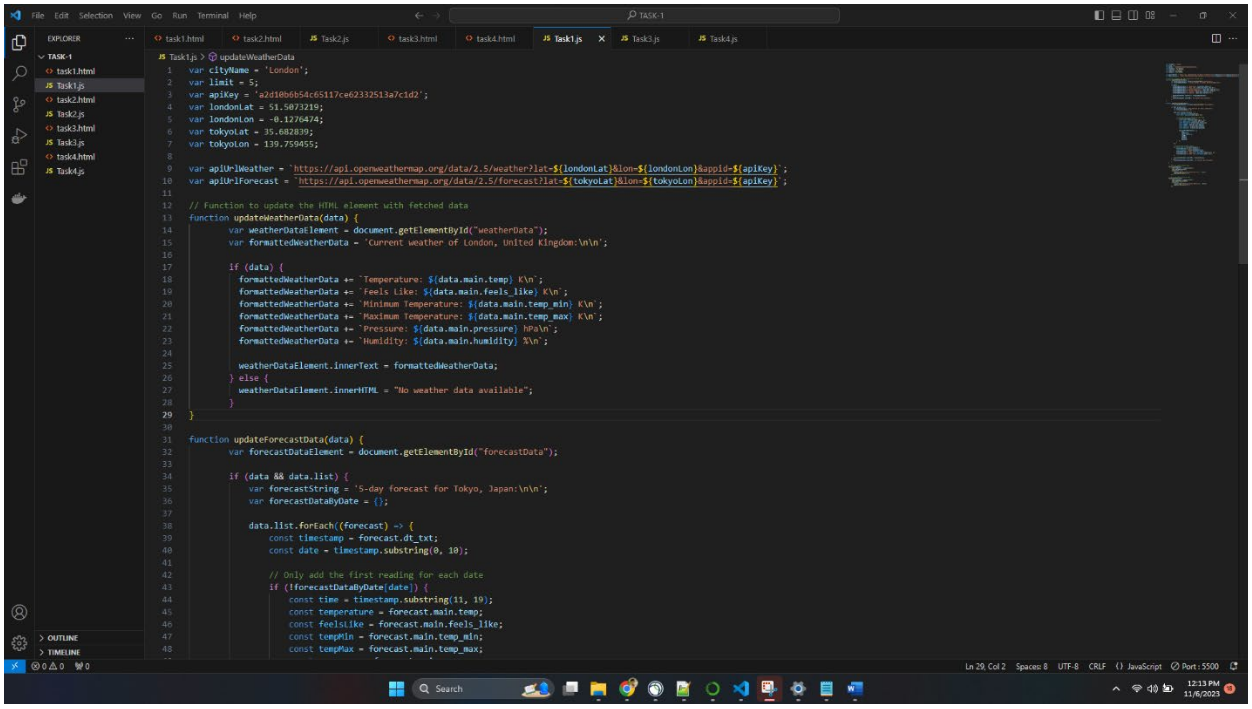
In terms of capabilities, OpenWeatherMap offers not only current weather data but also forecasts, historical weather information, and data about air quality, which is quite extensive. This versatility opens up numerous potential applications, from building weather apps and websites to integrating weather information into various projects, such as travel planning, outdoor event management, and more. Overall, the OpenWeatherMap API is a robust and user-friendly tool that can be leveraged for a wide array of weather-related applications.

HTML:

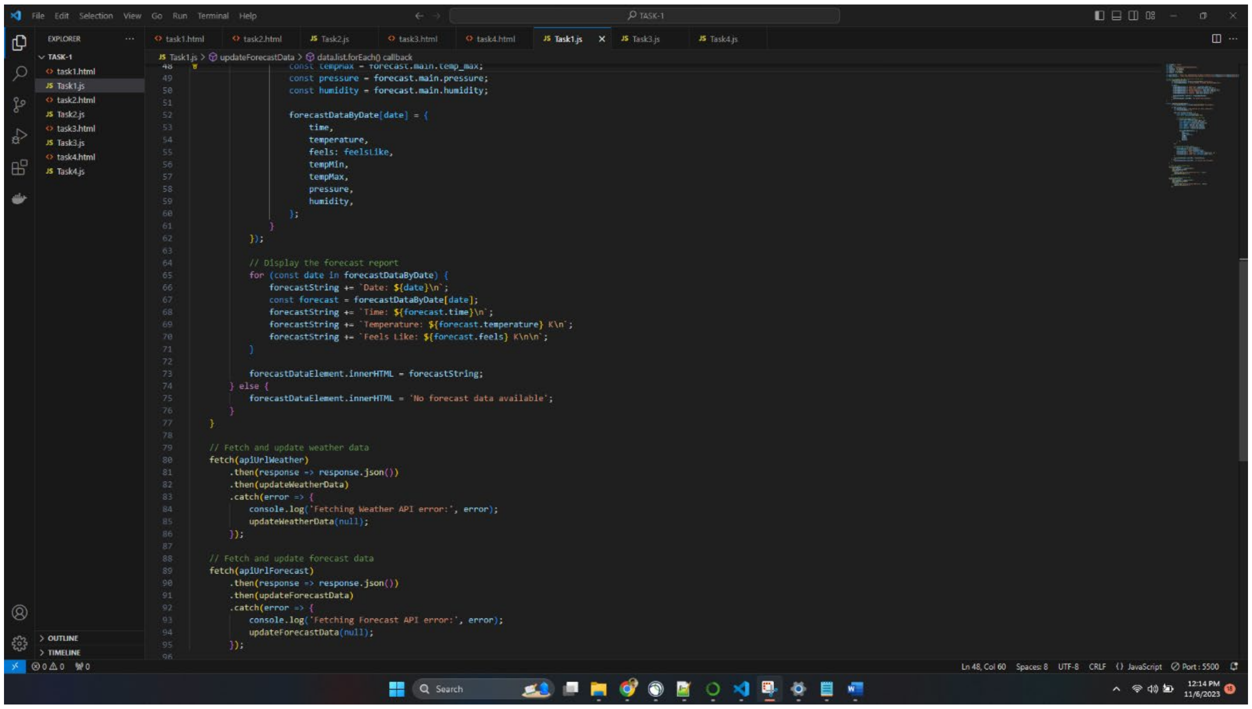


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>OpenWeatherMap API TASK</h1>
    <div class="data-section">
        <h2>Weather Data of London</h2>
        <pre id="weatherData"></pre>
    </div>
    <div class="data-section">
        <h2>Forecast Data of Tokyo</h2>
        <pre id="forecastData"></pre>
    </div>
    <script src="Task1.js"></script>
    <a href="task2.html">Go to Next Screen</a>
</body>
</html>
```

JavaScript:

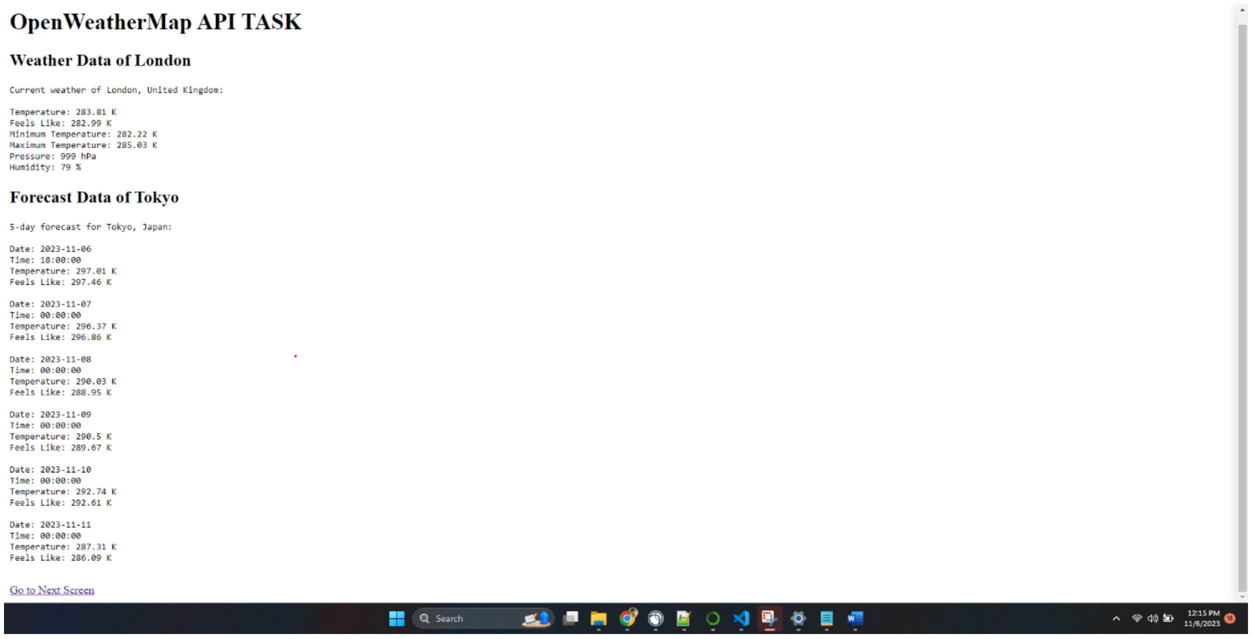


```javascript
var cityName = 'London';
var limit = 5;
var apiKey = 'a2d10b6b54c65117ce62332513a7c1d2';
var londonLat = 51.5073219;
var londonLon = -0.1276474;
var tokyoLat = 35.682839;
var tokyoLon = 139.759455;

var apiUrlWeather = `https://api.openweathermap.org/data/2.5/weather?lat=${londonLat}&lon=${londonLon}&appid=${apiKey}`;
var apiUrlForecast = `https://api.openweathermap.org/data/2.5/forecast?lat=${tokyoLat}&lon=${tokyoLon}&appid=${apiKey}`;

// Function to update the HTML element with fetched data
function updateWeatherData(data) {
        var weatherDataElement = document.getElementById("weatherData");
        var formattedWeatherData = 'Current weather of London, United Kingdom:\n\n';

        if (data) {
          formattedWeatherData += `Temperature: ${data.main.temp} K\n`;
          formattedWeatherData += `Feels Like: ${data.main.feels_like} K\n`;
          formattedWeatherData += `Minimum Temperature: ${data.main.temp_min} K\n`;
          formattedWeatherData += `Maximum Temperature: ${data.main.temp_max} K\n`;
          formattedWeatherData += `Pressure: ${data.main.pressure} hPa\n`;
          formattedWeatherData += `Humidity: ${data.main.humidity} %\n`;

          weatherDataElement.innerText = formattedWeatherData;
        } else {
          weatherDataElement.innerHTML = "No weather data available";
        }
}

function updateForecastData(data) {
        var forecastDataElement = document.getElementById("forecastData");

        if (data && data.list) {
            var forecastString = '5-day forecast for Tokyo, Japan:\n\n';
            var forecastDataByDate = {};

            data.list.forEach((forecast) => {
                const timestamp = forecast.dt_txt;
                const date = timestamp.substring(0, 10);

                // Only add the first reading for each date
                if (!forecastDataByDate[date]) {
                    const time = timestamp.substring(11, 19);
                    const temperature = forecast.main.temp;
                    const feelsLike = forecast.main.feels_like;
                    const tempMin = forecast.main.temp_min;
                    const tempMax = forecast.main.temp_max;
```



```javascript
                    const pressure = forecast.main.pressure;
                    const humidity = forecast.main.humidity;

                    forecastDataByDate[date] = {
                        time,
                        temperature,
                        feels: feelsLike,
                        tempMin,
                        tempMax,
                        pressure,
                        humidity,
                    };
                }
            });

            // Display the forecast report
            for (const date in forecastDataByDate) {
                forecastString += `Date: ${date}\n`;
                const forecast = forecastDataByDate[date];
                forecastString += `Time: ${forecast.time}\n`;
                forecastString += `Temperature: ${forecast.temperature} K\n`;
                forecastString += `Feels Like: ${forecast.feels} K\n\n`;
            }

            forecastDataElement.innerHTML = forecastString;
        } else {
            forecastDataElement.innerHTML = 'No forecast data available';
        }
    }

    // Fetch and update weather data
    fetch(apiUrlWeather)
        .then(response => response.json())
        .then(updateWeatherData)
        .catch(error => {
            console.log('Fetching Weather API error:', error);
            updateWeatherData(null);
        });

    // Fetch and update forecast data
    fetch(apiUrlForecast)
        .then(response => response.json())
        .then(updateForecastData)
        .catch(error => {
            console.log('Fetching Forecast API error:', error);
            updateForecastData(null);
        });
```

Result:



OpenWeatherMap API TASK

Weather Data of London

```
Current weather of London, United Kingdom:

Temperature: 283.81 K
Feels Like: 282.99 K
Minimum Temperature: 282.22 K
Maximum Temperature: 285.03 K
Pressure: 999 hPa
Humidity: 79 %
```

Forecast Data of Tokyo

```
5-day forecast for Tokyo, Japan:

Date: 2023-11-06
Time: 18:00:00
Temperature: 297.01 K
Feels Like: 297.46 K

Date: 2023-11-07
Time: 00:00:00
Temperature: 296.37 K
Feels Like: 296.86 K

Date: 2023-11-08
Time: 00:00:00
Temperature: 290.03 K
Feels Like: 288.95 K

Date: 2023-11-09
Time: 00:00:00
Temperature: 290.5 K
Feels Like: 289.67 K

Date: 2023-11-10
Time: 00:00:00
Temperature: 292.74 K
Feels Like: 292.61 K

Date: 2023-11-11
Time: 00:00:00
Temperature: 287.31 K
Feels Like: 286.09 K
```

Go to Next Screen

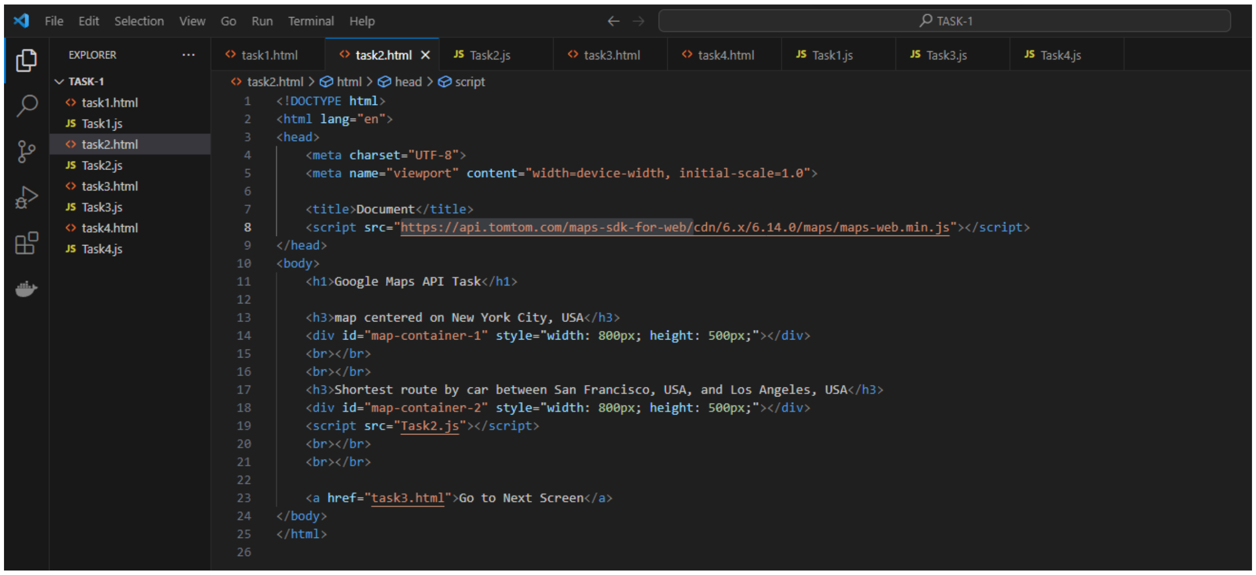
Task-2:

My experience with the TomTom Maps SDK for Web has been quite impressive. The SDK is user-friendly and well-documented, making it easy for developers to integrate dynamic maps and location-based services into their web applications. The API's capabilities are extensive, offering features like interactive maps, geocoding, and routing, allowing for the creation of powerful location-aware applications.

The potential applications for the TomTom Maps SDK are diverse and significant. From building navigation apps and ride-sharing services to incorporating location-based search and mapping into e-commerce platforms, the SDK opens up a world of possibilities. The interactive maps and advanced routing features provide valuable tools for enhancing user experiences and adding a spatial dimension to a wide range of web applications. Overall, the TomTom Maps SDK for Web is a versatile and user-friendly solution with great potential for enhancing location-based web applications.

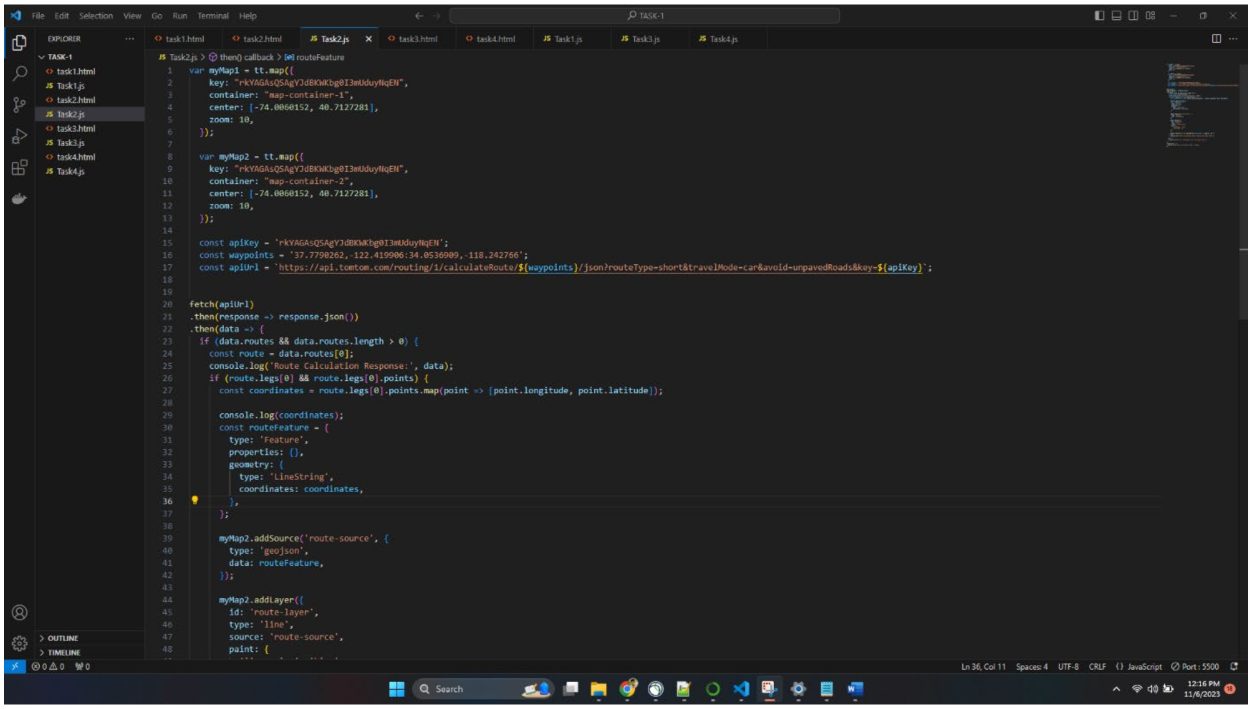
Vamshi Krishna Mummadi

HTML:

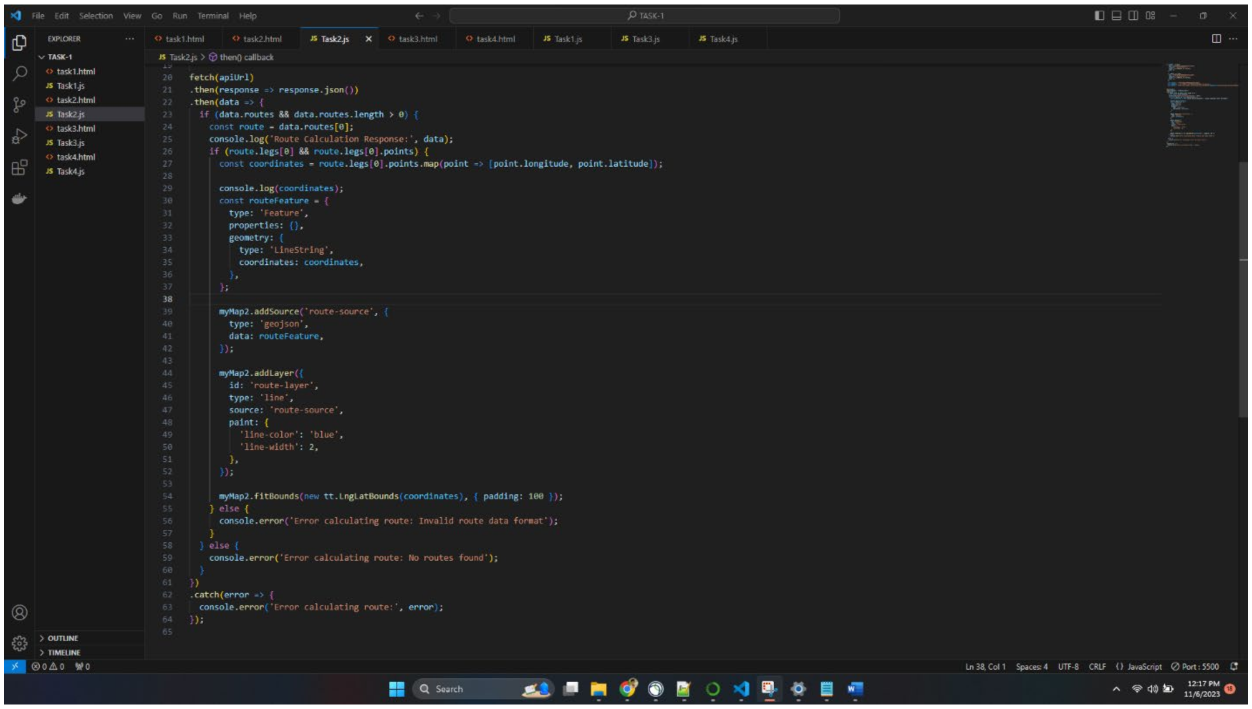


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>
    <script src="https://api.tomtom.com/maps-sdk-for-web/cdn/6.x/6.14.0/maps/maps-web.min.js"></script>
</head>
<body>
    <h1>Google Maps API Task</h1>

    <h3>map centered on New York City, USA</h3>
    <div id="map-container-1" style="width: 800px; height: 500px;"></div>
    <br></br>
    <br></br>
    <h3>Shortest route by car between San Francisco, USA, and Los Angeles, USA</h3>
    <div id="map-container-2" style="width: 800px; height: 500px;"></div>
    <script src="Task2.js"></script>
    <br></br>
    <br></br>

    <a href="task3.html">Go to Next Screen</a>
</body>
</html>

```

JavaScript:



```javascript
var myMap1 = tt.map({
    key: "rkYAGAsQSAgYJdBKWKbg0I3mUduyNqEN",
    container: "map-container-1",
    center: [-74.0060152, 40.7127281],
    zoom: 10,
  });

  var myMap2 = tt.map({
    key: "rkYAGAsQSAgYJdBKWKbg0I3mUduyNqEN",
    container: "map-container-2",
    center: [-74.0060152, 40.7127281],
    zoom: 10,
  });

  const apiKey = 'rkYAGAsQSAgYJdBKWKbg0I3mUduyNqEN';
  const waypoints = '37.7790262,-122.419906:34.0536909,-118.242766';
  const apiUrl = `https://api.tomtom.com/routing/1/calculateRoute/${waypoints}/json?routeType=short&travelMode=car&avoid=unpavedRoads&key=${apiKey}`;


fetch(apiUrl)
.then(response => response.json())
.then(data => {
  if (data.routes && data.routes.length > 0) {
    const route = data.routes[0];
    console.log('Route Calculation Response:', data);
    if (route.legs[0] && route.legs[0].points) {
      const coordinates = route.legs[0].points.map(point => [point.longitude, point.latitude]);

      console.log(coordinates);
      const routeFeature = {
        type: 'Feature',
        properties: {},
        geometry: {
          type: 'LineString',
          coordinates: coordinates,
        },
      };

      myMap2.addSource('route-source', {
        type: 'geojson',
        data: routeFeature,
      });

      myMap2.addLayer({
        id: 'route-layer',
        type: 'line',
        source: 'route-source',
        paint: {
```

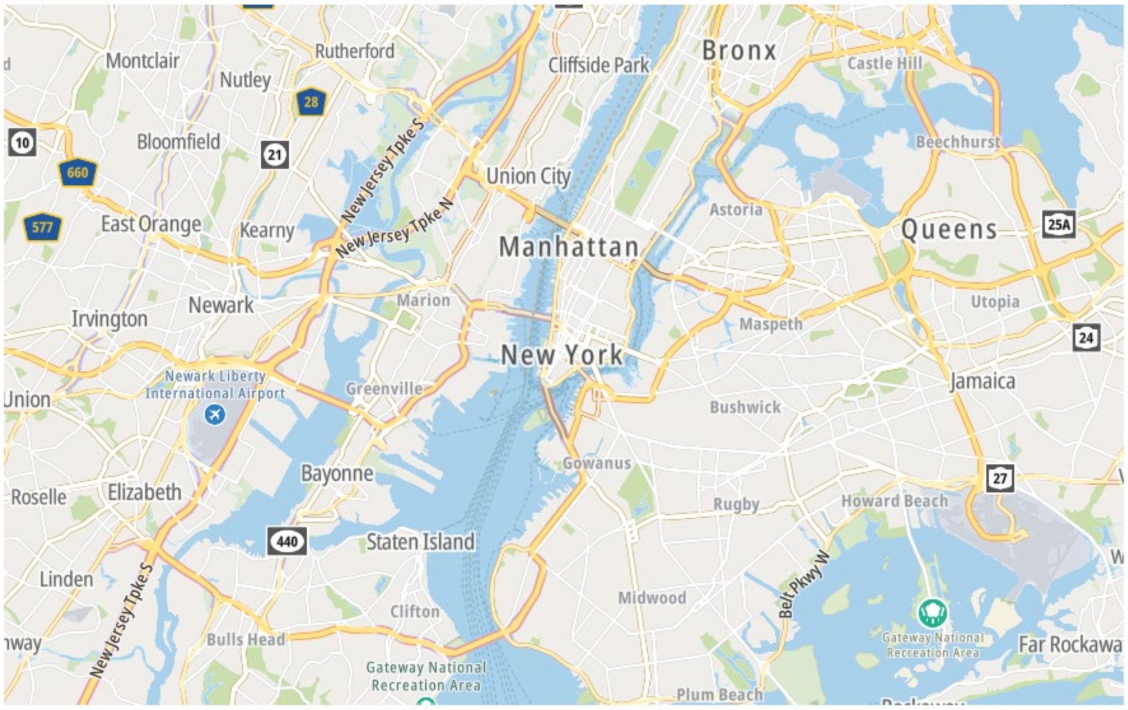


```
fetch(apiUrl)
.then(response => response.json())
.then(data => {
  if (data.routes && data.routes.length > 0) {
    const route = data.routes[0];
    console.log('Route Calculation Response:', data);
    if (route.legs[0] && route.legs[0].points) {
      const coordinates = route.legs[0].points.map(point => [point.longitude, point.latitude]);

      console.log(coordinates);
      const routeFeature = {
        type: 'Feature',
        properties: {},
        geometry: {
          type: 'LineString',
          coordinates: coordinates,
        },
      };

      myMap2.addSource('route-source', {
        type: 'geojson',
        data: routeFeature,
      });

      myMap2.addLayer({
        id: 'route-layer',
        type: 'line',
        source: 'route-source',
        paint: {
          'line-color': 'blue',
          'line-width': 2,
        },
      });

      myMap2.fitBounds(new tt.LngLatBounds(coordinates), { padding: 100 });
    } else {
      console.error('Error calculating route: Invalid route data format');
    }
  } else {
    console.error('Error calculating route: No routes found');
  }
})
.catch(error => {
  console.error('Error calculating route:', error);
});
```

Result:

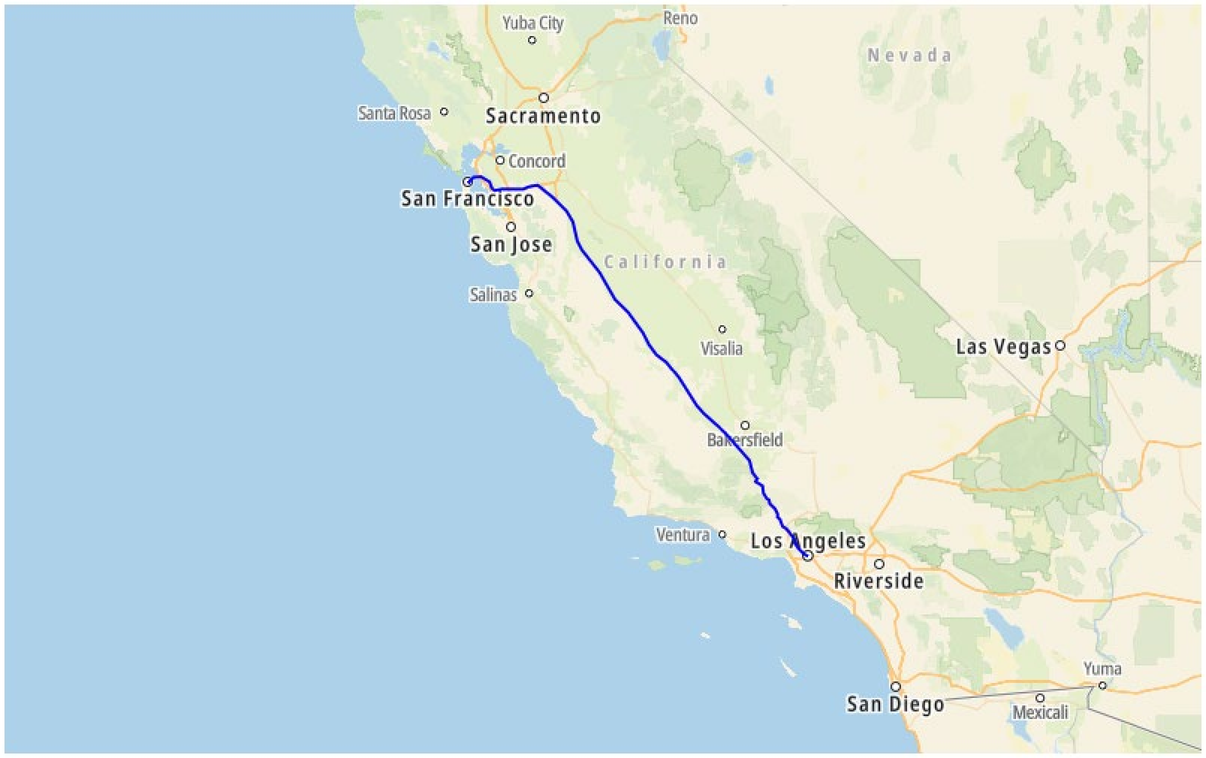
Google Maps API Task

map centered on New York City, USA



©TomTom

Shortest route by car between San Francisco, USA, and Los Angeles, USA



©TomTom

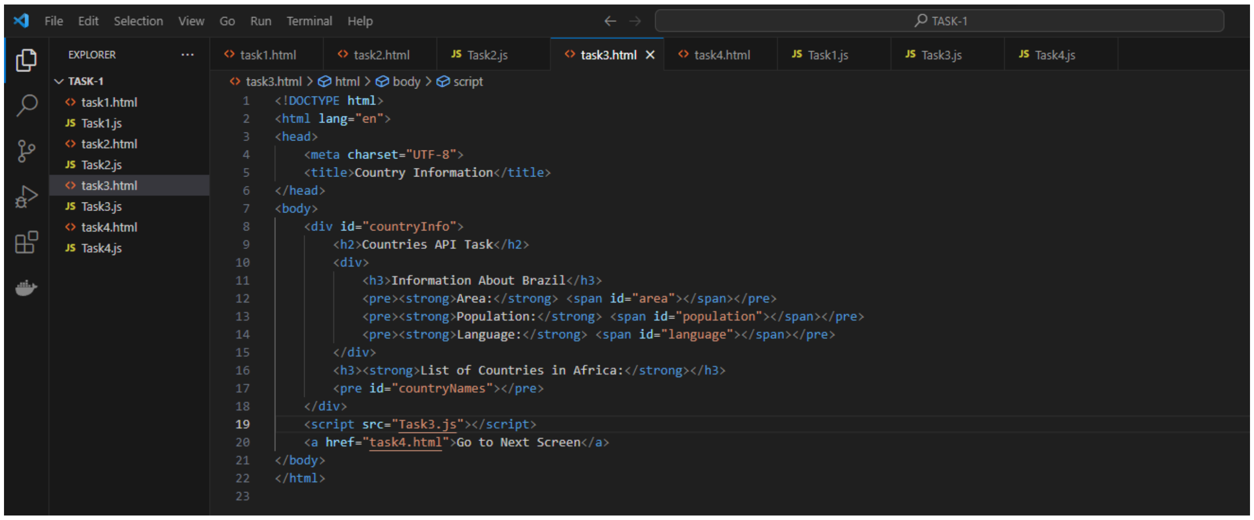
Task-3:

My experience with the RestCountries API has been quite positive. The API is straightforward to use and provides a wealth of data related to countries, making it a valuable resource for developers looking to access comprehensive information about nations worldwide. The API's documentation is well-structured and easy to navigate, which greatly aids in understanding how to make requests and work with the responses.

In terms of capabilities, RestCountries offers not only basic details about countries but also a wide array of additional data such as currency information, time zones, and even regional borders. This versatility opens up numerous potential applications, from building educational platforms and travel-related websites to integrating country-specific information into e-commerce and research projects. Overall, the RestCountries API is a robust and user-friendly tool that can be leveraged for a wide array of applications that require accurate and up-to-date country-related data.

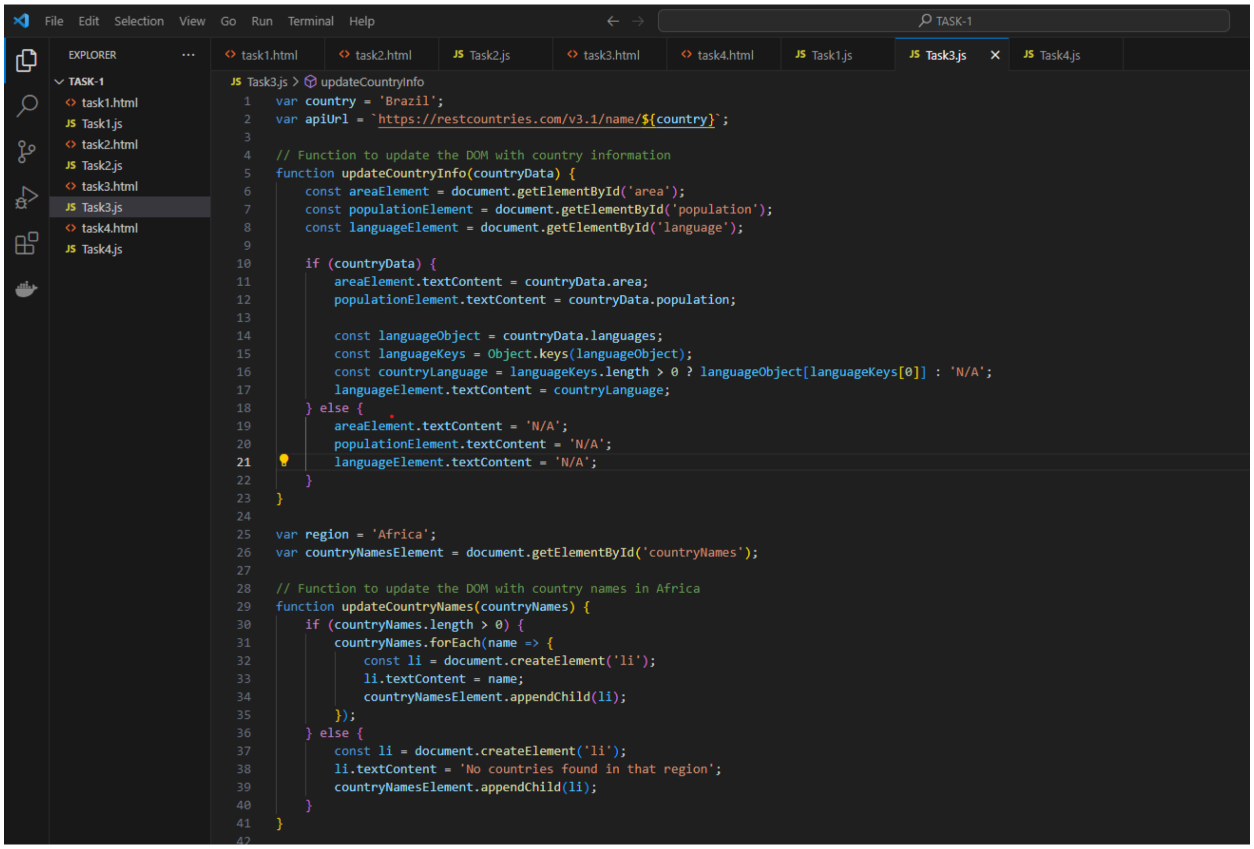
HTML:

Vamshi Krishna Mummadi

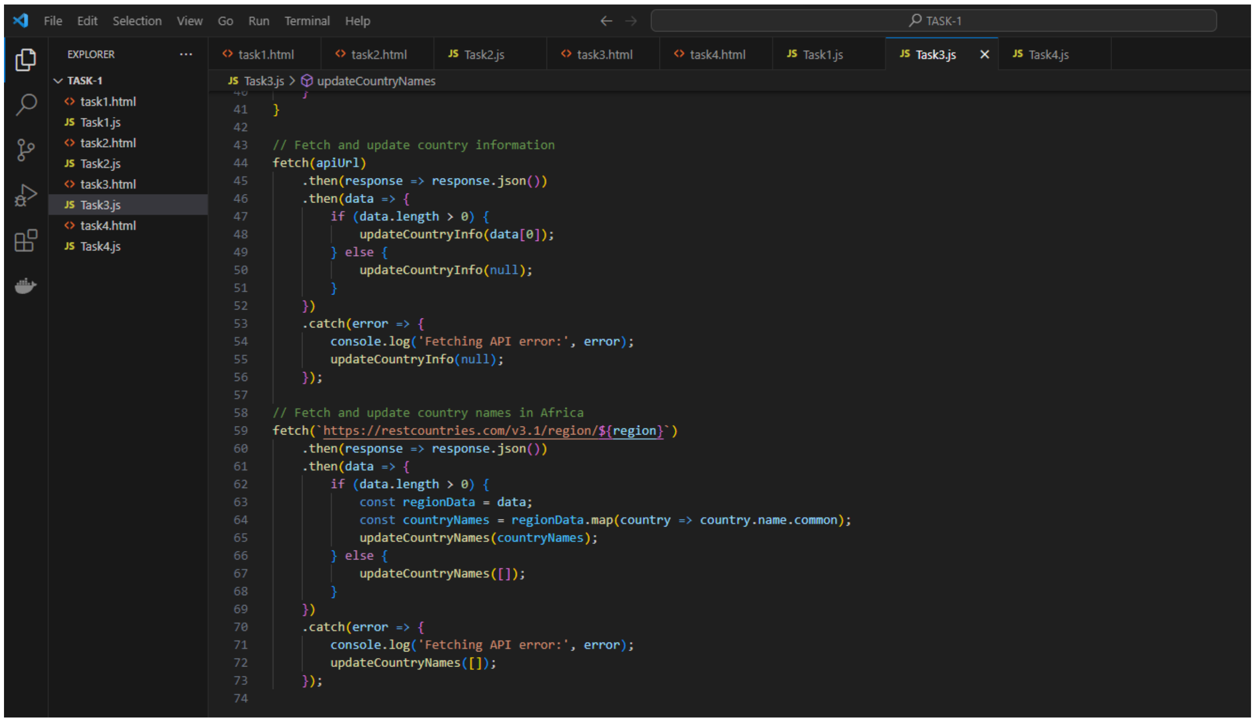


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Country Information</title>
</head>
<body>
    <div id="countryInfo">
        <h2>Countries API Task</h2>
        <div>
            <h3>Information About Brazil</h3>
            <pre><strong>Area:</strong> <span id="area"></span></pre>
            <pre><strong>Population:</strong> <span id="population"></span></pre>
            <pre><strong>Language:</strong> <span id="language"></span></pre>
        </div>
        <h3><strong>List of Countries in Africa:</strong></h3>
        <pre id="countryNames"></pre>
    </div>
    <script src="Task3.js"></script>
    <a href="task4.html">Go to Next Screen</a>
</body>
</html>
```

JavaScript:

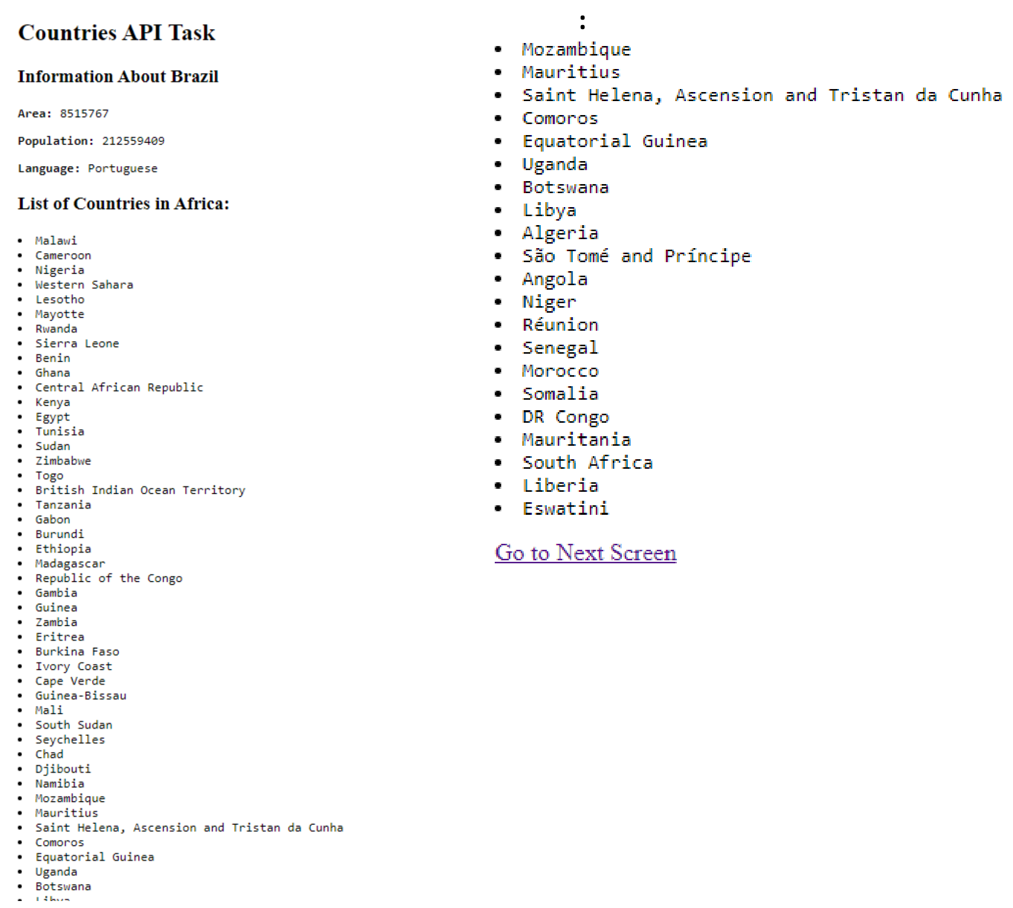


```javascript
var country = 'Brazil';
var apiUrl = `https://restcountries.com/v3.1/name/${country}`;

// Function to update the DOM with country information
function updateCountryInfo(countryData) {
    const areaElement = document.getElementById('area');
    const populationElement = document.getElementById('population');
    const languageElement = document.getElementById('language');

    if (countryData) {
        areaElement.textContent = countryData.area;
        populationElement.textContent = countryData.population;

        const languageObject = countryData.languages;
        const languageKeys = Object.keys(languageObject);
        const countryLanguage = languageKeys.length > 0 ? languageObject[languageKeys[0]] : 'N/A';
        languageElement.textContent = countryLanguage;
    } else {
        areaElement.textContent = 'N/A';
        populationElement.textContent = 'N/A';
        languageElement.textContent = 'N/A';
    }
}

var region = 'Africa';
var countryNamesElement = document.getElementById('countryNames');

// Function to update the DOM with country names in Africa
function updateCountryNames(countryNames) {
    if (countryNames.length > 0) {
        countryNames.forEach(name => {
            const li = document.createElement('li');
            li.textContent = name;
            countryNamesElement.appendChild(li);
        });
    } else {
        const li = document.createElement('li');
        li.textContent = 'No countries found in that region';
        countryNamesElement.appendChild(li);
    }
}
```



```
}

// Fetch and update country information
fetch(apiUrl)
    .then(response => response.json())
    .then(data => {
        if (data.length > 0) {
            updateCountryInfo(data[0]);
        } else {
            updateCountryInfo(null);
        }
    })
    .catch(error => {
        console.log('Fetching API error:', error);
        updateCountryInfo(null);
    });

// Fetch and update country names in Africa
fetch(`https://restcountries.com/v3.1/region/${region}`)
    .then(response => response.json())
    .then(data => {
        if (data.length > 0) {
            const regionData = data;
            const countryNames = regionData.map(country => country.name.common);
            updateCountryNames(countryNames);
        } else {
            updateCountryNames([]);
        }
    })
    .catch(error => {
        console.log('Fetching API error:', error);
        updateCountryNames([]);
    });
```

Result:



Countries API Task

Information About Brazil

Area: 8515767

Population: 212559409

Language: Portuguese

List of Countries in Africa:

- Malawi
- Cameroon
- Nigeria
- Western Sahara
- Lesotho
- Mayotte
- Rwanda
- Sierra Leone
- Benin
- Ghana
- Central African Republic
- Kenya
- Egypt
- Tunisia
- Sudan
- Zimbabwe
- Togo
- British Indian Ocean Territory
- Tanzania
- Gabon
- Burundi
- Ethiopia
- Madagascar
- Republic of the Congo
- Gambia
- Guinea
- Zambia
- Eritrea
- Burkina Faso
- Ivory Coast
- Cape Verde
- Guinea-Bissau
- Mali
- South Sudan
- Seychelles
- Chad
- Djibouti
- Namibia
- Mozambique
- Mauritius
- Saint Helena, Ascension and Tristan da Cunha
- Comoros
- Equatorial Guinea
- Uganda
- Botswana

:

- Mozambique
- Mauritius
- Saint Helena, Ascension and Tristan da Cunha
- Comoros
- Equatorial Guinea
- Uganda
- Botswana
- Libya
- Algeria
- São Tomé and Príncipe
- Angola
- Niger
- Réunion
- Senegal
- Morocco
- Somalia
- DR Congo
- Mauritania
- South Africa
- Liberia
- Eswatini

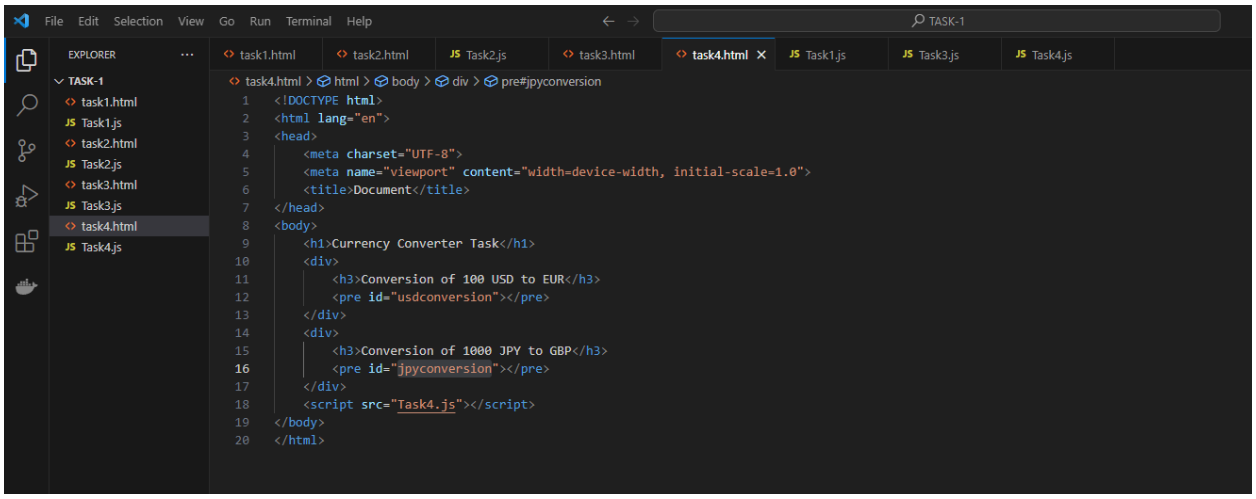
Go to Next Screen

Task-4:

My experience with the ExchangeRate API has been quite positive. The API is simple and easy to use, providing accurate and up-to-date exchange rate information for various currencies. Its simplicity makes it accessible for both novice and experienced developers. The API's well-organized documentation provides clear instructions for making requests and handling responses, which greatly facilitates its integration into various applications.

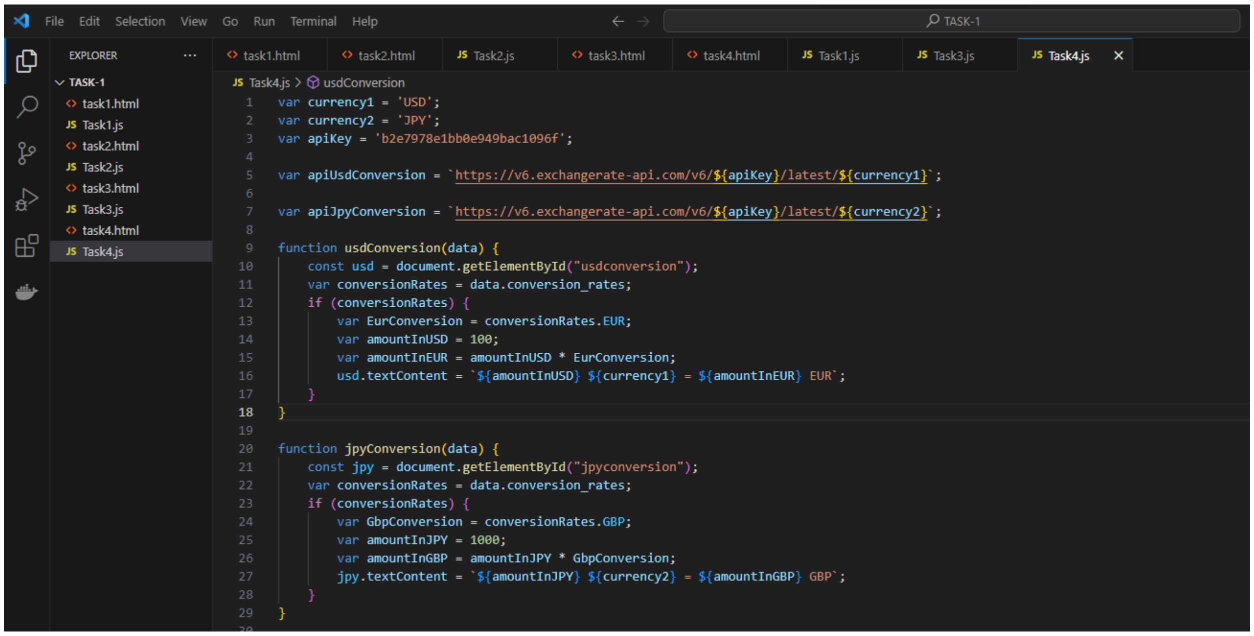
In terms of capabilities, the ExchangeRate API excels at providing real-time exchange rate data, currency conversion, and historical rate retrieval. This functionality opens up numerous potential applications, from building currency conversion tools and financial applications to integrating exchange rate information into e-commerce platforms for international transactions. The API's versatility, ease of use, and reliable data make it a valuable resource for a wide range of applications that require currency-related information. Overall, the ExchangeRate API is a user-friendly and capable tool that can enhance a variety of financial and international trade applications.

HTML:

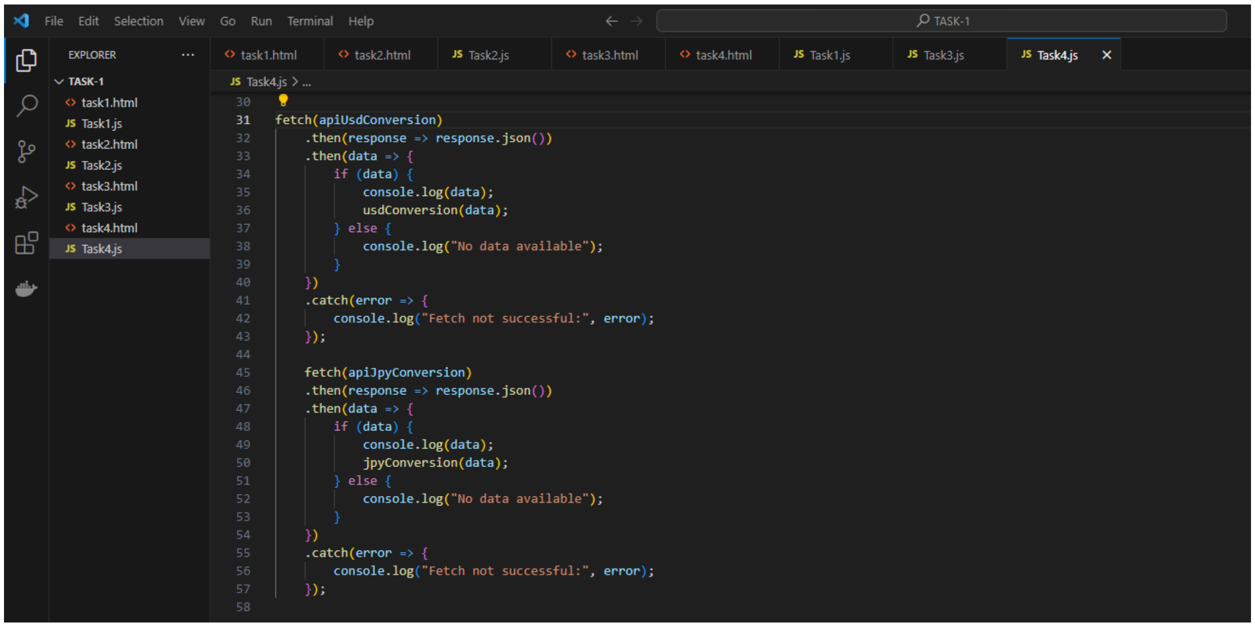

File Edit Selection View Go Run Terminal Help
TASK-1
EXPLORER
TASK-1
task1.html
Task1.js
task2.html
Task2.js
task3.html
Task3.js
task4.html
Task4.js
task1.html task2.html Task2.js task3.html task4.html Task1.js Task3.js Task4.js
task4.html > html > body > div > pre#jpyconversion

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Currency Converter Task</h1>
    <div>
        <h3>Conversion of 100 USD to EUR</h3>
        <pre id="usdconversion"></pre>
    </div>
    <div>
        <h3>Conversion of 1000 JPY to GBP</h3>
        <pre id="jpyconversion"></pre>
    </div>
    <script src="Task4.js"></script>
</body>
</html>
```

JavaScript:

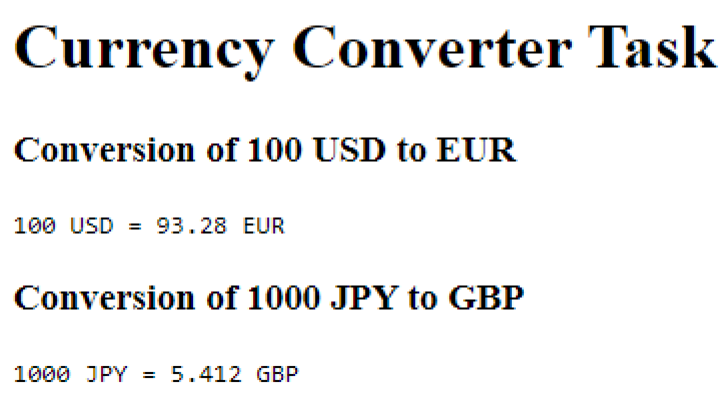


```
var currency1 = 'USD';
var currency2 = 'JPY';
var apiKey = 'b2e7978e1bb0e949bac1096f';

var apiUsdConversion = `https://v6.exchangerate-api.com/v6/${apiKey}/latest/${currency1}`;

var apiJpyConversion = `https://v6.exchangerate-api.com/v6/${apiKey}/latest/${currency2}`;

function usdConversion(data) {
    const usd = document.getElementById("usdconversion");
    var conversionRates = data.conversion_rates;
    if (conversionRates) {
        var EurConversion = conversionRates.EUR;
        var amountInUSD = 100;
        var amountInEUR = amountInUSD * EurConversion;
        usd.textContent = `${amountInUSD} ${currency1} = ${amountInEUR} EUR`;
    }
}

function jpyConversion(data) {
    const jpy = document.getElementById("jpyconversion");
    var conversionRates = data.conversion_rates;
    if (conversionRates) {
        var GbpConversion = conversionRates.GBP;
        var amountInJPY = 1000;
        var amountInGBP = amountInJPY * GbpConversion;
        jpy.textContent = `${amountInJPY} ${currency2} = ${amountInGBP} GBP`;
    }
}
```



```

fetch(apiUsdConversion)
    .then(response => response.json())
    .then(data => {
        if (data) {
            console.log(data);
            usdConversion(data);
        } else {
            console.log("No data available");
        }
    })
    .catch(error => {
        console.log("Fetch not successful:", error);
    });

    fetch(apiJpyConversion)
    .then(response => response.json())
    .then(data => {
        if (data) {
            console.log(data);
            jpyConversion(data);
        } else {
            console.log("No data available");
        }
    })
    .catch(error => {
        console.log("Fetch not successful:", error);
    });
```

Result:



Currency Converter Task

Conversion of 100 USD to EUR

```
100 USD = 93.28 EUR
```

Conversion of 1000 JPY to GBP

```
1000 JPY = 5.412 GBP
```