First calculate the co2 emissions:

$$CO^2\ Emissions\ (g) = \left(\frac{Data\ Transfer\ (MB)}{1024}\right) \times Energy\ Intensity\ \left(\frac{KWh}{GB}\right) \times Carbon\ Intensity\ (g\ CO_2\ /kWh) \times Website\ Traffic$$

Data Transfer: Server sends request to url and finds the size of the website as it's loaded to RAM. From this we can see how much transfer happens per click

Energy Intensity: We then need to calculate how energy intensive a click is. This energy intensity is calculated in KWh/GB. It basically sums up how much energy is required to push data through a wire. There is differences depending on the transfer protocol. Eg. UDP is less energy intensive compared to TCP as it doesn't rely on a handshake for data validation.

Carbon Intensity:

We then need to calculate how carbon intensive a click is. This carbon intensity is calculated in grams. We have to options on calculating this:

1. Can use global average.

(Optional) 2. Can use *Electricity Maps* for specific calculation (https://static.electricitymaps.com/api/docs/index.html)

(Optional) **Website Traffic**: measured in users. There are many ways to measure traffic to a website. Google analytics offers and API: https://developers.google.com/analytics/devguides/reporting/data/v1

Second calculate if the url is within the green energy scheme (thegreenwebfoundation)

i. https://admin.thegreenwebfoundation.org/api-docs/
ii. Install 'CO2.js' and run on internal server. You can then implement the back end to talk to it on the server.

Thirdly plug the results into an internal grading system and output to user

Will just need to create an internal grading system that we can compare our results to, taking into account:

CO2 Intensity (See above for formula), hosting type.

PSEUDOCODE

```
// Step 1: Calculate CO2 Emissions

function calculate_CO2_emissions(data_transfer_MB, energy_intensity_KWh_per_GB,
carbon_intensity_CO2_per_KWh, website_traffic) {
    // Convert Data Transfer from MB to GB
    data_transfer_GB = data_transfer_MB / 1024

    // Calculate CO2 Emissions
    CO2_emissions = data_transfer_GB * energy_intensity_KWh_per_GB *
carbon_intensity_CO2_per_KWh * website_traffic

    return CO2_emissions
}

// Step 2: Fetch Data Transfer Size from URL

function get_data_transfer_size(url) {
    // Server sends request to the URL and finds the size of the website
    data_transfer_MB = request_website_size_in_MB(url)

    return data_transfer_MB
}

// Step 3: Calculate Energy Intensity based on transfer protocol

function get_energy_intensity(protocol) {
    // Check if the protocol is UDP or TCP
    if protocol == 'UDP':
        energy_intensity_KWh_per_GB = 0.7  // Example lower intensity for UDP
    else:
        energy_intensity_KWh_per_GB = 0.9  // Higher intensity for TCP

    return energy_intensity_KWh_per_GB
}

// Step 4: Calculate Carbon Intensity

function get_carbon_intensity(option) {
    if option == 'global_average':
        carbon_intensity_CO2_per_KWh = 475  // Global average CO2 in grams per KWh
    else if option == 'electricity_map':
        carbon_intensity_CO2_per_KWh = fetch_from_electricity_map_API()  // Call API for
specific carbon intensity

    return carbon_intensity_CO2_per_KWh
}

// Step 5: (Optional) Fetch Website Traffic from Google Analytics

function get_website_traffic(url) {
```

```
    website_traffic = fetch_traffic_from_google_analytics_API(url)

    return website_traffic
}

// Step 6: Check if the URL is in the Green Energy Scheme

function check_green_energy(url) {
    green_energy_status = fetch_green_energy_status_from_API(url)  // Use The Green Web
Foundation API

    return green_energy_status
}

// Step 7: Grading System for CO2 Output

function grade_website(CO2_emissions, hosting_type) {
    if CO2_emissions < threshold_low and hosting_type == 'green':
        grade = 'A'
    else if CO2_emissions < threshold_medium:
        grade = 'B'
    else:
        grade = 'C'

    return grade
}

// Main function to execute all steps

function main(url, protocol, traffic_option, carbon_option) {
    data_transfer_MB = get_data_transfer_size(url)
    energy_intensity_KWh_per_GB = get_energy_intensity(protocol)
    carbon_intensity_CO2_per_KWh = get_carbon_intensity(carbon_option)

    if traffic_option == 'use_analytics':
        website_traffic = get_website_traffic(url)
    else:
        website_traffic = 1  // Default value for one click

    CO2_emissions = calculate_CO2_emissions(data_transfer_MB, energy_intensity_KWh_per_GB,
carbon_intensity_CO2_per_KWh, website_traffic)

    green_energy_status = check_green_energy(url)

    hosting_type = 'green' if green_energy_status else 'non-green'

    grade = grade_website(CO2_emissions, hosting_type)

    output_to_user(CO2_emissions, grade)
}

// Example Usage
main('https://example.com', 'TCP', 'use_analytics', 'global_average')
```

1. calculate_CO2_emissions: Calculates CO_2 emissions based on data transfer, energy intensity, carbon intensity, and website traffic.

2. get_data_transfer_size: Fetches the size of the website in MB.

3. get_energy_intensity: Determines energy intensity based on the protocol used (UDP vs. TCP).

4. get_carbon_intensity: Fetches carbon intensity, either using a global average or the Electricity Map API.

5. get_website_traffic: Optionally fetches website traffic using the Google Analytics API.

6. check_green_energy: Checks if the website is hosted on a green energy server using the Green Web Foundation API.

7. grade_website: Assigns a grade based on CO_2 emissions and hosting type.

8. main: Orchestrates the whole process by calling other functions.