# Routes

Make a simple document containing the following for each route your app executes:  
● Description of the functionality of the route  
● Request path  
● Request parameters, types, and descriptions  
● Query parameters (parameters used in the SQL / NoSQL query)  
● Response parameters, types, and descriptions.

**Route 1: /api/climate-summary (for query 1)**

**Method:** GET  
**Description:** Returns average daily temperature, CO2 emissions, and sea level by country for the given year.

**Request Parameters:**

* None

**Query Parameters (SQL):**

* year (for temperature, CO2, and sea level)

**Response Parameters:**

* country\_name (string): Name of the country
* avg\_daily\_temp (float): Average surface temperature in 2020
* avg\_co2\_emissions (float): Average CO2 emissions in 2020
* avg\_sea\_level (float): Average sea level in 2020

**Route 2: /api/crop-wildfires**

**Method:** GET  
**Description:** Returns wildfire area, CO2 production, crop type, and average crop yield for countries in provided continent on the provided year.

**Request Parameters:**

* None

**Query Parameters (SQL):**

* continent
* year

**Response Parameters:**

* country\_name (string): Country name
* total\_wildfire\_area (float): Total burned area
* total\_co2\_production (float): Total CO2 production
* crop\_type (string): Crop name
* avg\_crop\_yield (float): Average yield of the crop

**Route 3: /api/continent-crop-yield**

**Method:** GET  
**Description:** Returns average crop yield and irrigation access by crop and continent.

**Request Parameters:**

* None

**Query Parameters (SQL):**

* None

**Response Parameters:**

* continent (string): Continent name
* crop (string): Crop name
* avg\_yield (float): Average yield
* avg\_irrigation\_access (float): Average access to irrigation

**Route 4: /api/urban-co2**

**Method:** GET  
**Description:** Compares urban-to-rural population ratio with CO2 per capita over time.

**Request Parameters:**

* None

**Query Parameters (SQL):**

* None

**Response Parameters:**

* country\_name (string): Country name
* year (integer): Year of record
* urban\_rural\_ratio (float): Ratio of urban to rural population
* co2\_per\_capita (float): Per capita CO2 emissions

**Route 5: /api/urban-wildfires-co2**

**Method:** GET  
**Description:** Returns urban and rural population, average CO2 emissions, and wildfire area per country for 2020.

**Request Parameters:**

* None

**Query Parameters (SQL):**

* year = 2020

**Response Parameters:**

* country\_name (string): Country name
* urban\_pop (integer): Urban population
* rural\_pop (integer): Rural population
* avg\_co2 (float): Average CO2 emissions
* total\_wildfire\_area (float): Total wildfire area

**Route 6: /api/top-co2-countries**

**Method:** GET  
**Description:** Retrieves top countries with highest CO2 per capita in 2019.

**Request Parameters:**

* None

**Query Parameters (SQL):**

* Limit (optional, default = 10) => SQL: LIMIT 10

**Response Parameters:**

* country\_name (string): Country name
* co2\_per\_capita (float): CO2 per capita emissions

**Route 7: /api/urban-majority**

**Method:** GET  
**Description:** Retrieves countries where urban population exceeds rural in latest year.

**Request Parameters:**

* None

**Query Parameters (SQL):**

* urban\_pop > rural\_pop
* year = MAX(year)

**Response Parameters:**

* country\_name (string): Country name
* urban\_pop (integer): Urban population
* rural\_pop (integer): Rural population

**Route 8: /api/wildfire-hotspots**

**Method:** GET  
**Description:** Retrieves countries with more than 10 wildfire events.

**Request Parameters:**

* None

**Query Parameters (SQL):**

* None

**Response Parameters:**

* country\_name (string): Country name
* wildfire\_events (integer): Number of wildfires

**Route 9: /api/wildfire-vs-crop-yield**

**Method:** GET  
**Description:** Analyzes the relationship between wildfire area and average crop yield.

**Request Parameters:**

* None

**Query Parameters (SQL):**

* None

**Response Parameters:**

* country\_name (string): Country name
* total\_wildfire\_area (float): Total burned area
* avg\_crop\_yield (float): Average crop yield

**Route 10: /api/sea-level-vs-co2**

**Method:** GET  
**Description:** Returns sea level trends and CO2 emissions by country and year.

**Request Parameters:**

* None

**Query Parameters (SQL):**

* None

**Response Parameters:**

* country\_name (string): Country name
* year (integer): Year of record
* co2\_emissions (float): CO2 emissions
* avg\_sea\_level (float): Average sea level for the year

Using (possibly updated) queries from Milestone 3, consolidate the API routes for your  
server application. While you are not required to implement much on the client side, you should think about the client-side functionality supported by each route you design.  
Roughly, every route should use (possibly different versions of) one query with a few  
parameters. Specifically, you must have routes corresponding to each of the (at least 10) queries, plus any other ‘auxiliary’ routes that may or may not use SQL queries. Please note that a route may exist without a query associated with it (e.g., adding a user). Also be mindful of the 4 complex queries that you must include in the final project. You get to decide what request parameters are important based on what your query needs to execute properly.

req.params (if the param is a path param) or in req.query (if the param is a query param).

API Spec

Create an API specification for these routes. We encourage you to use the specification from the WebDB Assignment Part 1 instructions as a starting point. Feel free to include other useful information (like HTTP status codes), etc. Alternatively, you may also choose to create an OpenAPI-style specification. It is acceptable to make a few changes to the API as you progress. However, you will be required to include the final version of the specification in the final project report.  
An effective way to define an API spec will include, for each route, the following:  
1. The request path (ex. /getUser) along with the method (i.e., GET, POST, DELETE, PUT)  
and a description of what the route is doing (e.g., retrieves a user by username).  
2. The request params, including the name (e.g., username), param type (path or  
query), data type (e.g., integer, string, etc.), required / optional indicator, and  
description (e.g., username corresponding to a user).  
3. The response params, including the name, data type, and description.