SUSTAINABLE DEVELOPMENT GOALS

> Yes, the virus only kills a small percentage of those afflicted. Yes, the flu kills 10s of thousands of people annually. Yes, 80% of people will experience lightweight symptoms with COVID19. Yes the mortality rate of COVID19 is relatively low (1–2%). All of this true, but is immaterial. They are the wrong numbers to focus on...
>
> -- Jason S Warner

#BuildForSDG Cohort-1 JavaScript Assessment

This assessment empowers you to **attempt** helping society and leaders prepare for the **real rig problem** of COVID-19, which is **its impact on lives, our health systems, and the economy**:

> 1. Too many patients, not enough hospitals and beds. A serious shortage of ventilators, masks and other PPE if *we don't practice extreme social distancing*.
> 2. Job losses or freezes, low cash flow and low production of essentials like food, from too many being sick or people caring for their sick loved ones.

Project Setup & Submission Process

> Go here for a guide and step-by-step video on how to setup your project and submit your work.

Challenges

You will be building a **novelty** COVID-19 infections ~~predictor~~ estimator following guidelines outlined in the challenges.

Your estimator will receive input data structured as:

```
{
  region: {
      name: "Africa",
      avgAge: 19.7,
      avgDailyIncomeInUSD: 5,
      avgDailyIncomePopulation: 0.71
    },
  periodType: "days",
  timeToElapse: 58,
  reportedCases: 674,
  population: 66622705,
  totalHospitalBeds: 1380614
}
```

and the output will be required to be an impact estimatation having the data structure specified as :

```
{
  data: {},          // the input data you got
  impact: {},        // your best case estimation
  severeImpact: {}   // your severe case estimation
}
```

This assessment is broken down into several challenges, incrementally helping you build out your COVID-19 impact estimator.

Locate the `src/estimator.js` file. This is where you will be writing all of your code. Feel free to create your own additional functions within this file (or in other files, which you then import into this file) to get the work done, but make sure to not alter the definition for the `covid19ImpactEstimator` function or its export declaration. `covid19ImpactEstimator` is the entry point to your work, and the one thing we'll be evaluating.

> Your computations should be saved into the `impact` and `severeImpact` objects of the return output data structure.

Challenge 1

Given the `reportedCases` property in the input data to your `covid19ImpactEstimator` function, estimate the number of **currently infected** people as `reportedCases * 10`. This figure will be part of your output estimate, saved as the `currentlyInfected` property (i.e `impact.currentlyInfected` is `reportedCases * 10`).

According to Harvard Medical School / Massachusetts General Hospital (see the references at the end of this guide), there are likely up to **50x more people infected** than known reported cases. Based on this, compute `currentlyInfected` for the `severeImpact` output as `reportedCases * 50` (i.e `severeImpact.currentlyInfected` is `reportedCases * 50`)

To estimate the number of infected people **30 days** from now, note that `currentlyInfected` doubles every **3 days**, so you'd have to multiply it by a factor of 2.

E.g: `currentlyInfected x (2 to the power of *factor*)` where factor is `10` for a `30 day` duration (there are **10** sets of **3 days** in a perioid of **30 days**)

Effectively, the projected number of infections after a 30 day period is

```
currentlyInfected x (2 to the power of 10)
```

OR, simply

```
currentlyInfected x 1024
```

This computation should be saved as `infectionsByRequestedTime`, and should done both for `impact` and `severeImpact`, using their respective `currentlyInfected` values.

> For the rest of this assessment, just like you have done for `currentlyInfected` and `infectionsByRequestedTime`, you will be doing all required computations in two fold. One for `impact`, and the other for `severImpact`

> PS: Your estimator will be required to make estimations over periods in days, weeks and months. We recommend you make an effort to normalise the duration input to days, and then do your computation based on that

Challenge 2

Determine `15%` of `infectionsByRequestedTime`. This is the estimated number of **severe** positive cases that will require hospitalization to recover. Represent this as `severeCasesByRequestedTime` and make it a part of your estimation output

Given `severeCasesByRequestedTime`, determine the number of available beds. On average, 65% of hospital beds are already occupied by patients and many hospitals usually are at 90% or 95% capacity. We want to expect only a `35%` bed availability in hospitals for severe COVID-19 positive patients.

Based on the above, the `totalHospitalBeds` input data, and your `severeCasesByRequestedTime`, estimate the number of available hospital beds for severe COVID-19 positive patients. E.g `23` (meaning only 23 hospital beds will be available), or `-350` (meaning there'll be a shortage of 350 beds for severe patients after hospitals are full to capacity)

Represent your beds computation as `hospitalBedsByRequestedTime` and make it a part of your estimation output.

Challenge 3

Determine `5%` of `infectionsByRequestedTime`. This is the estimated number of severe positive cases that will require **ICU care**. Represent this as `casesForICUByRequestedTime` and make it a part of your estimation output.

Also, determine `2%` of `infectionsByRequestedTime`. This is the estimated number of severe positive cases that will require **ventilators**. Represent this as `casesForVentilatorsByRequestedTime` and make it a part of your estimation output.

Finally, given the estimated number of infected people by the requested time and the AVG daily income of the region, estimate how much money the economy is likely to lose over the said period. Save this as `dollarsInFlight` in your output data structure. E.g if **65%** of the region (the majority) earn **$1.5** a day, you can compute `dollarsInFlight` over a **30 day period** as:

```
(infectionsByRequestedTime x 0.65) x 1.5 x 30;
```

> PS: At this point, you've now completed the first part of building your estimator. Did you notice that it is not accounting for deaths or recoveries over time? This is a novelty estimator, but we hope it gives you ideas of how you can use technology, open source, and research to attempt providing advocacy and advisory to any cause, including the COVID-19 pandemic.

Challenge 4

Here, you are **only required** to do **one of the tasks** below. Ideally, you should choose the frontend or backend task basd on your skill and preference, which should also be in alignment with the data you submitted while applying into the #BuildforSDG program. Finally, once you make it into the program, the task you completed here will strongly influence the role you will play in your team.

> If you feel up to it, try completing both the frontend and backend tasks, especially if you want to function as fullsatck developer in the program.

Tasks A - For Frontend Developers

Build a simple single page web app that can allow the data passed to your estimator function to be inputed from a UI. Your goal is to build a simple UI that

1. Scores a **minimum of 75** for performance, best practices, SEO, and accessibility, as measured by Chrome's Lighthouse tool. Host your web app with GitHub pages (or Netlify), and make sure the root of your project has a **.env** file with a `FRONTEND` entry pointing to the link of your hosted and running UI app. E.g:

```
FRONTEND=http://url-of-hosted-and-running-web-app
```

2. Can allow a user enter data with some input fields, and then click on a button which passes the data to the estimation function you built from challeng 1 to 3. In the UI, designate each input field with a HTML attribute as follows:

  - input for `population` data has attribute of `data-population`
  - input for `timeToElapse` data has attribute of `data-time-to-elapse`
  - input for `reportedCases` data has attribute of `data-reported-cases`
  - input for `totalHospitalBeds` data has attribute of `data-total-hospital-beds`
  - input for `periodType` data has attribute of `data-period-type` and should be a SELECT or RADIO input field with values limited to `days`, `weeks`, or `months`
  - the button for submitting the data from the UI should have an attribute of `data-go-estimate`.

> Feel free to handle submission with the `data-go-estimate` button and pass the data from the input fields to your estimation function. Though this is not required (we won't even check), doing it helps you have a functional UI you can show off!

Task B - For Backend Develoeprs

Build a simple REST API around your estimator function and host it on Heroku or any other free service. Your REST API should allow a user make a HTTP POST request to your endpoint, providing the data normally passed to your estimator function, and get back the estimation data produced by your estimator function. Specifically, your API should :

1. Have a `/api/v1/on-covid-19` endpoint that can take the input data and return the estimation for it. Sure you can handle the response using the function you built from challenge 1 to 3. Also, have the default response format be **JSON**, but allow the user specify a response format by terminating the request URL with `/json` for JSON and `/xml` for XML. E.g: If your API is hosted on `http://example.com`, the endpoint will be `http://example.com/api/v1/on-covid-19/` and `http://example.com/api/v1/on-covid-19/xml` will produce the response in XML format and with the right **HTTP response headers** for XML content. Make sure the root of your project has a **.env** file with a `BACKEND` entry pointing to the link of your hosted and running REST API. E.g:

```
BACKEND=http://url-of-hosted-and-running-REST-api
```

3. Maintain a request / response time difference log that is accessibale via `/api/v1/on-covid-19/logs`. The logs endpoint should produce a **newline-separated** list of entries for each request / response cycle. For every request you get and for which you served a response, log an entry into the logs. Your logs can be saved to a database, a JSON file or whatever you decide is best for persistence, but after calling your API 3 times, there should be 3 additional entries in the logs. A valid response from the `/logs` endpoint should be **text data** with entries containing a timestamp, the request path, and how long it took to handle the request. Separate timestamp, the request path, and the time difference with 2 tab spaces (i.e `\t\t`), but recall that each record needs to be on a new line. See below for an example

   ```
   1585703407605    on-covid-19         done in 1.20 seconds
   1585703407625    on-covid-19/json    done in 1.35 seconds
   1585703407659    on-covid-19/xml     done in 2.50 seconds
   ```

References

- 19.7 years is the AVG age in Africa - worldometers.info
- AVG daily income is ~$5 for 85% of Africans - blogs.worldbank.org
- COVID-19 infections doubles every 3 days. Expect 50x (at least 10x) recported cases to be actual number of infected people - Jason S. Warner
- Several stats from Harvard Medical School / Massachusetts General Hospital - move to **00:09:42** of this video

---