

This proposal was one of the accepted submissions for Google Summer of Code 2022. The author/contributor uploaded it publicly through commit efce929: <https://github.com/InternetHealthReport/gsoc/commit/efce9293ab8eabb887af828db96fd1c94c7ee91d>



Google Summer of Code

Proposal - Google Summer of Code 2022

Exploration of open datasets on the IHR website

By
Aniket Pal



Table of Contents

About me	3
Why I decided to work for IHR?	3
Project Title	4
A brief summary of the proposal: Abstract	4
How do I plan to achieve: Milestones	4
Visualization widgets for M-Lab data	5
Integration of transparency report	15
IHR exploratory dashboard	20
Documenting it	26
Plan of Action: Timeline	27
Duration wise timeline	27
Apart from the vital contributions: Extras	29
Open Source contributions and Projects	29
Expectations from Mentor	31
What are my other commitments?	31
What do I plan after the GSoC period is over?	32
Who am I: About me	32
The experience I possess: Places worked	33

About me

Name: Aniket Pal
Timezone: Asia/Kolkata(UTC+05:30)
Email: aniketindian8@gmail.com
Course: B.Tech in Electrical Engineering
University: National Institute of Technology, Rourkela
Country: India
GSoC Full Time: Yes
Links: [Github](#) [LinkedIn](#) [Dev.to](#) [Twitter](#)
Resume: [Resume](#)

Why did I decide to work for IHR?

I was seeking to do a challenging project this summer, the Google Summer of Code 2022 provides the perfect opportunity. I have contributed to other open-source organizations however after reviewing project ideas, I decided that I should focus on the IHR project. Once this decision was made and after reaching out to [Romain](#) sir, it proved to be a good fit for reaching my project and learning goals. IHR's motive is to help network operators, policymakers, and other stakeholders, with a better understanding of the Internet's infrastructure and its evolution, which I want to join in. I always wanted to test my networking knowledge in practice, IHR covers a very big sector I was able to understand a bit of it but, I got fascinated. Till now, I have made some open-source contributions to organizations but never actually thought of getting into GSoC as a participant. Being a part of the IHR community not only for GSoC but in the future is as well, is planned. While being very well versed with the application stack IHR uses while making some code contributions to the existing codebase will give me a head start in the learning curve, it also allows for a more immediate impact on the overall IHR Project mission. Furthermore, recently I became IHR's GitHub member which created a sense of belonging to the organization. Last and more importantly, the recent Pandemic moves me to contribute to the information sector where my contribution will have a high impact.

Project Title:

Exploration of open datasets on the IHR website

A brief summary of the proposal: Abstract

The audience of IHR largely consists of researchers, network operators, policymakers, and educators. The goal of the proposal is to make the IHR's internet application an end-to-end solution for stakeholders to work on. Currently, the IHR website provides plots of internet latency and routing for AS and countries. The goal of this software subsystem is to add new plots showing speed test data collected by Measurement-Lab so that users can monitor how disconnections and routing events impact throughput. While providing the users with a quick lookup for the NDT test it becomes immensely important to co-relate IHR traffic analysis with Google and Cloudflare reports. Collecting data from similar services would also pump IHR's anomaly detector to complement IHR's alerting system. With IHR the hub for multiple network reports and metrics, it is vital to have a dashboard to customize visual data. I propose to develop a customized IHR dashboard that is self-exploratory. The dashboard would let end users explore the IHR dataset and monitor any event with respect to time and legends according to their purpose. The dashboard would let the user select a set source and destination networks, time periods, and a metric and correspondingly plot the data. Ultimately, adding an assistant to select a set of important networks from a selected country or global statistics about the plotted data.

How do I plan to achieve: Milestones

I have decided to combine three software subsystems that can scale up IHR's application for end-users. The project includes three following subsections:

1. Visualization widgets for Measurement Lab dataset
2. Integration of transparency report
3. IHR exploratory dashboard

The visualization widgets and transparency reports will all be integrated into the final dashboard to provide the end-users with easy accessibility to all features. The process for the widgets and the transparency reports are quite similar in nature. Both fetch the data from their sources i.e. MLab, Google, and Cloudflare, analyze the data, and create visual graphs of them. They will both also be having filters/buttons for search and comparisons. It basically aims to develop an algorithm to extract data from multiple sources in a time-dependent dataset and integrate it with IHR's AS or country reports. I look at these integrations from the perspective of enabling a whole new dimension of use cases. So our aim should be to provide it to end-users in an extremely flexible form so that people can build upon it for their personal use cases that they can think of. In addition to that, a major advantage of these techniques is that they can work in near real-time. Hence it unlocks completely new avenues for internet health matrices. Since all the three features are closely linked, another reason to build these projects as a whole is to maintain a uniform codebase which would ease the process for future researchers and contributors.

Visualization widgets for M-Lab data

With the aim to build dynamic components for the IHR website from the NDT data of Measurement Labs I aim to integrate their Rest API. The Rest API:

<https://statistics.measurementlab.net/>

This is the endpoint in which we can pass in parameters and fetch data according to our needs. The Endpoint can be modulated according to the parameter we pass in. The parameters it accepts as the query is:

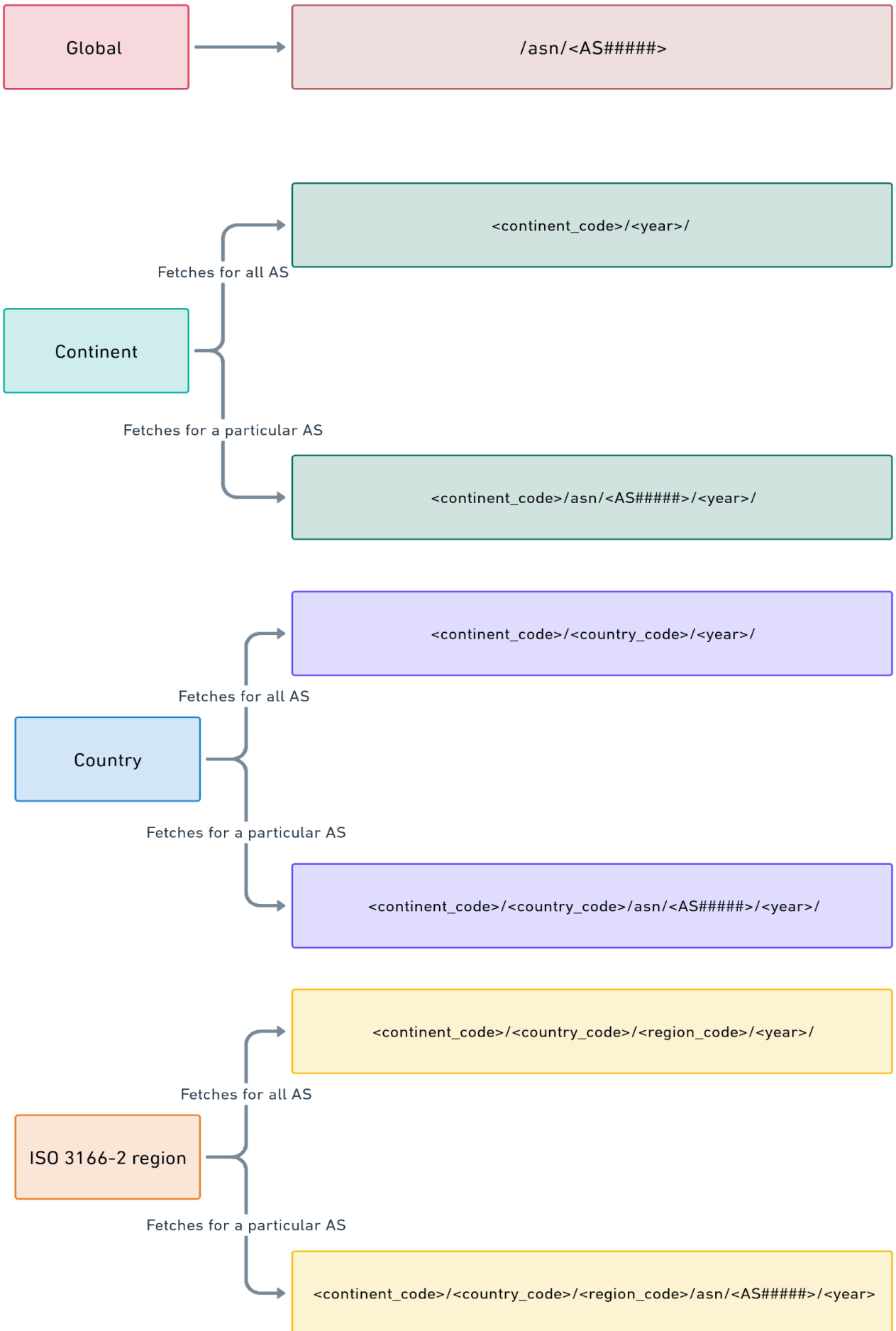
- Continent
- Country
- Year

- ASN

At the top level, a version number is assigned which provides the current version of the API. As per my proposal, I would be using v0 stats-api. Therefore, currently, my endpoint would look something like this:

<https://statistics.measurementlab.net/v0>

The API at each level, aggregated by year, and ASN are provided. There are particular endpoints according to geographic locations:



NDT data may be aggregated by any geography. After doing the `get` calls we get key and value pairs from the API. The keys and values we get are:

Field	Description
"date":"2020-01-01",	The date in YYYY-MM-DD format.
"bucket_min":0,	The lower bound of the bucket which the statistics in this object represent.
"bucket_max":1.7782794100389228	The upper bound of the bucket which the statistics in this object represent.
"dl_LOG_AVG_rnd1":25.591	The LOG Average of download measurements in this aggregation, using the first randomly selected test per IP address in the set. Value is presented in megabits per second.
"dl_LOG_AVG_rnd2":25.577	The LOG Average of download measurements in this aggregation, using the second randomly selected test per IP address in the set. Value is presented in megabits per second.
"dl_minRTT_LOG_AVG_rnd1":26.256	The LOG Average of Minimum Round Trip Time of download measurements in this aggregation, using the first randomly selected test per IP

	address in the set. Value is presented in milliseconds.
"dl_minRTT_LOG_AVG_rnd2":26.268	The LOG Average of Minimum Round Trip Time of download measurements in this aggregation, using the second randomly selected test per IP address in the set. Value is presented in milliseconds.
"dl_frac_bucket":0.057	The fraction of download measurements within this histogram bucket.
"dl_samples_bucket":10695	The number of download measurement samples in this bucket.
"dl_samples_day":188725	The number of download measurement samples on this day.
"download_MIN":0	The minimum download speed in megabits per second on this day.
"download_Q25":8.141	The first quartile (25th percentile) download speed in megabits per second on this day.
"download_MED":31.95	The median download speed in megabits per second on this day.
"download_AVG":79.745	The average or mean download speed in megabits per second on this day.

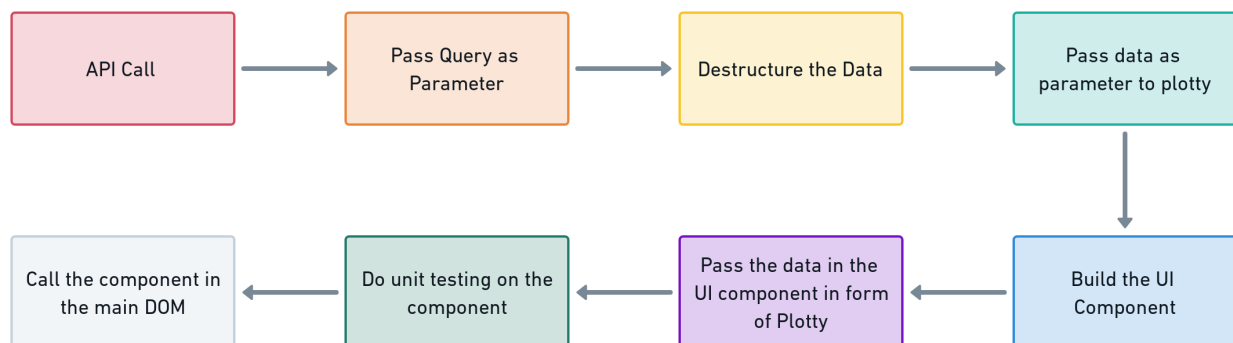
"download_Q75":97.572	The upper quartile (75th percentile) download speed in megabits per second on this day.
"download_MAX":3655.15	The maximum download speed in megabits per second on this day.
"download_minRTT_MED":25	The median Minimum Round Trip Time in milliseconds
"ul_LOG_AVG_rnd1":6.589	The LOG Average of upload measurements in this aggregation, using the first randomly selected test per IP address in the set. Value is presented in megabits per second.
"ul_LOG_AVG_rnd2":6.589	The LOG Average of upload measurements in this aggregation, using the second randomly selected test per IP address in the set. Value is presented in megabits per second.
"ul_minRTT_LOG_AVG_rnd1":24.988	The LOG Average of Minimum Round Trip Time of upload measurements in this aggregation, using the first randomly selected test per IP address in the set. Value is presented in milliseconds.

"ul_minRTT_LOG_AVG_rnd2":25.003	The LOG Average of Minimum Round Trip Time of upload measurements in this aggregation, using the second randomly selected test per IP address in the set. Value is presented in milliseconds.
"ul_frac_bucket":0.113	The fraction of upload measurements within this histogram bucket.
"ul_samples_bucket":20769	The number of upload measurement samples in this bucket.
"ul_samples_day":183326	The number of upload measurement samples on this day.
"upload_MIN":0	The minimum upload speed in megabits per second on this day.
"upload_Q25":2.356	The first quartile (25th percentile) upload speed in megabits per second on this day.
"upload_MED":7.857	The median upload speed in megabits per second on this day.
"upload_AVG":28.034	The average or mean upload speed in megabits per second on this day.
"upload_Q75":17.306	The upper quartile (75th percentile) upload speed in megabits per second on this

	day.
"upload_MAX":3199.958	The maximum upload speed in megabits per second on this day.
"upload_minRTT_MED":23.83	The median Minimum Round Trip Time in milliseconds for upload measurements on this day.

Getting the data is important but for me converting the JSON pair into graphs would make visualization and analysis more vital. Plotting the JSON using Plotly.JS visualization can be made easy.

The main requirement goes around with building with IHR's endpoints so the data passed as parameters will be according to IHR's guidelines and geo-points. Building components that are comparable with the existing matrix of **threshold** and **packet loss**, will give end users a more detailed analysis of the accuracy of the predictions or reports generated. The **general workflow** is to get the data from the API, destructure the data and get the corresponding data as per requirement and pass the destructured data into Plotly.



The ultimate aim of the project is to keep track of the IHR's concerned ASes. When the parameter is passed from the IHR dashboard or any

similar input field only those particular ASes should be passed for example AS2914 NTT-COMMUNICATIONS-2914

Let's take an example to get the data for Maryland in 2021. Getting started with the API Call in Vue using [Axios](#) from the endpoint over [here](#)



```
new Vue({
  el: '#app',
  data () {
    return {
      info: null,
      loading: true,
      errored: false
    }
  },
  mounted () {
    axios
      .get('https://statistics.measurementlab.net/v0/NA/US/US-MD/2021/histogram_daily_stats.json')
      .then(response => {
        this.info = response.data.bpi
      })
      .catch(error => {
        console.log(error)
        this.errored = true
      })
      .finally(() => this.loading = false)
  }
})
```

Once we have the data in total we can use the filter function to sort the data according to our need. Like over here:

```

new Vue({
  el: '#app',
  data () {
    return {
      info: null,
      loading: true,
      errored: false
    }
  },
  filters: {
    // will add filter according to need
  },
  mounted () {
    axios
      .get('https://statistics.measurementlab.net/v0/NA/US/US-MD/2021/histogram_daily_stats.json')
      .then(response => {
        this.info = response.data.bpi
      })
      .catch(error => {
        console.log(error)
        this.errored = true
      })
      .finally(() => this.loading = false)
  }
})

```

Once, we have filtered the data it's time for us to test in the console. Then we will pass the data to Plotly and plot it.

```

var myJson = info;
var figure = JSON.parse(myJson);

Plotly.newPlot('graph-div', figure.data, figure.layout);

```

Integration of transparency report

Transparency report showcases traffic levels per country and many services like Google and Cloudflare have taken the step to keep a track of all these traffic and disruptions. I aim to research and survey similar services to get a lot of reliable data. Talking about Cloudflare, it has a feature called the Cloudflare Radar which keeps us up to date with internet trends and insights.

This is the endpoint for Netflow:

```
https://radar.cloudflare.com/api/netrep/net/netflowchangerange
```

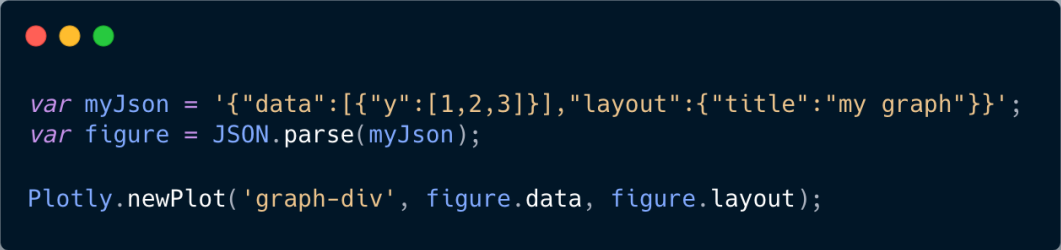
To show user-specific statistics, the parameters it accepts as the query are:

- Country
- Date
- Metric

The goal of this project is to fetch the data from similar services like Google and Cloudflare and use them to create graphs for better visualization. I will also add these graphs to the IHR country reports. One particular example of fetching data for Internet traffic change from Ukraine over the last 30 days needs to pass date_token and location_token to get the data

```
<Netflow API Endpoint>?date_token=last_30_days&location_token=UA
```

Setting the parameters to show results of the last 30 days in Ukraine, we get a JSON output as shown below. I will then parse this JSON file and use it to plot the graph with plotly.js.



```
var myJson = '{"data":[{"y":1,2,3}], "layout":{"title":"my graph"}}';  
var figure = JSON.parse(myJson);  
  
Plotly.newPlot('graph-div', figure.data, figure.layout);
```

In the above code example, I have stored dummy data in myJson object which is being parsed into JSON and correspondingly stored in the figure. We call the newPlot function from Plotly and pass the following parameters. The following approach can be easily used while fetching data from third-party resources and plotting them using Plotly.

The parameters we get from the Rest API endpoint:


```

{
  "startTimeStamp": "2022-03-04T10:20:00Z",
  "endTimeStamp": "2022-04-03T10:20:00Z",
  "timeRange": [717 items],
  "min": "0.099255",
  "prevOffsetHours": "672"
}

```

Pin Pointing to one particular data:

The screenshot shows the Cloudflare Radar API response viewer. The URL is `https://radar.cloudflare.com/api/netrep/netflowchangerange?date_token=last_30_days&location_token=UA#`. The response is a JSON object with a `timeRange` array. A tooltip is shown over the `tin` property of one item in the array, displaying the following nested structure:

```

{
  "items": [
    {
      "name": "netflowChange",
      "value": "0.527914",
      "timeStamp": "2022-03-03T15:00:00Z"
    },
    {
      "name": "httpChange",
      "value": "0.440541"
    },
    {
      "name": "netflowChangePrev",
      "value": "0.920377",
      "timeStamp": "2022-02-03T15:00:00Z"
    },
    {
      "name": "httpChangePrev",
      "value": "0.871486",
      "timeStamp": "2022-02-03T15:00:00Z"
    }
  ]
}

```

When it comes to traffic analysis Google's traffic becomes one of the prominent zones to be explored. The ratio of HTTPS:HTTP requests handled by Google shows transparency in the condition of the region.

transparencyreport.google.com/traffic/overview

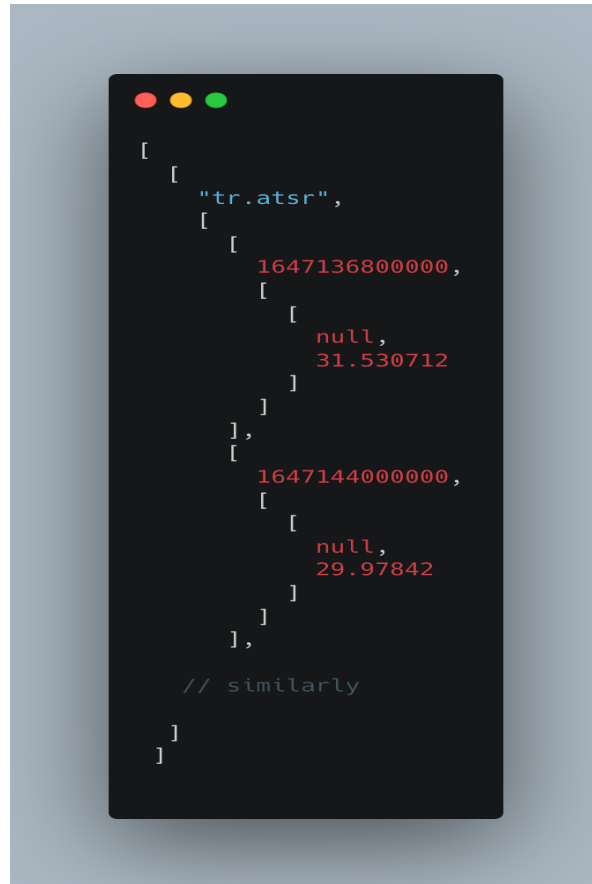
Logs the number of visits to every Google product, along with an approximation of the geographic region where the visit originated. A visible decrease in traffic in a specific region may mean that users there cannot access a product or service. This report documents real-time disruptions that we can observe, along with known disruptions reported by external parties. The API endpoint which shows the data is:

<https://transparencyreport.google.com/transparencyreport/api/v3/traffic/>

Upon passing parameters, we can get the traffic for each product on the basis of location and duration. For this API endpoint, we need to pass, the start date, end date, country, and product number. In the start query, we need to pass the start date, in the end query we need to pass the end date, for the region country code, and in the product parameter the product identification number.

end-point?start=" "&end=" "®ion=" "&product=" "

Upon sending a `get` request to the endpoint we get key-value pairs which when plotted in the graph give a visual representation. The value pair that we get for one particular region:



```
[
  [
    "tr.atsr",
    [
      1647136800000,
      [
        null,
        31.530712
      ]
    ]
  ],
  [
    1647144000000,
    [
      null,
      29.97842
    ]
  ],
  // similarly
]
```

Similarly, another awesome set of charts can be fetched and plotted from OONI Measurement Aggregation Toolkit (MAT).

<https://explorer.ooni.org/chart/mat>

In OONI we get to select from a lot of parameters and hence reduce our search space while maximizing the desired result. For the particular endpoint, the parameters we can pass are

- Country
- ASN
- From
- Until
- Test Types
- Matrices
 - Measurement Day
 - Website Categories
 - Countries
- Domain
- Website Categories

The basic parameters can be modified according to the need. OONI provides a multi-parameter passing feature. One sample endpoint:

```
https://explorer.ooni.org/chart/mat?test_name=web_connectivity&domain=twitter.com
&since=2022-03-15&until=2022-04-15&axis_x=measurement_start_day&axis_y=probe_asn
```

IHR exploratory dashboard

Integrating the transparency reports, visualizing packet loss, and getting NDT test results ultimately need filtering according to end-users needs. Building a dashboard from where users can use input filaments to post parameters and get a graph accordingly. Simply, getting a Graph for AS##### from 3rd March 2021 to 4th March 2021 with the precision of 30 minutes for each data over Maryland, USA. Sending out requesting with so many parameters in the API call and fetching the data becomes more of a developer's job and decreases the UX, and making the task of Researchers and network operators more difficult to examine the reports and shoot up their research work.

Building a controller which can act as a GUI to pass in parameters. Parameters like Location can be selected from Map, duration from the

calendar, and time from the clock, ultimately making the GUI examine the data across multiple Graph easier.



To provide an opportunity for users to compare statistics and explore the dashboard more, I plan to create a section where the user can select a data source and a new graph will be plotted on top of the original IHR's graph. We could use an add/compare button and set the parameters following which we'll be able to see the IHR data and the data from the third party with proper dynamic legends for clarity as well.

We need to develop input components that can pass data as queries to third-party REST APIs. Using **v-model** to bind a data property to the input and then simply pass it into the Axios call as param.




Defining a dummy addDuration() method,



```
methods: {  
  addMessage( ) {  
    this.messages.push(this.message);  
  }  
}
```

Once, we know how to pass data via input let us pass query via,



```
const res = await axios.get('API', { params });
```

Coming to a conclusion just from one graph is not possible, or even believing one data source as a point of truth is not reliable. Over here, comparison between graphs plays a very vital role. I was planning to plot one graph for selected params say time duration, location, and the initial graph would have a dataset corresponding to IHR's data. Then

the user gets an option to compare more graphs from other data so they can use an add/compare button and set the parameters after that on the master graph with IHR data the secondary data from third-party sources would join in and dynamic legends would be drawn following the `useEffect()` principle.



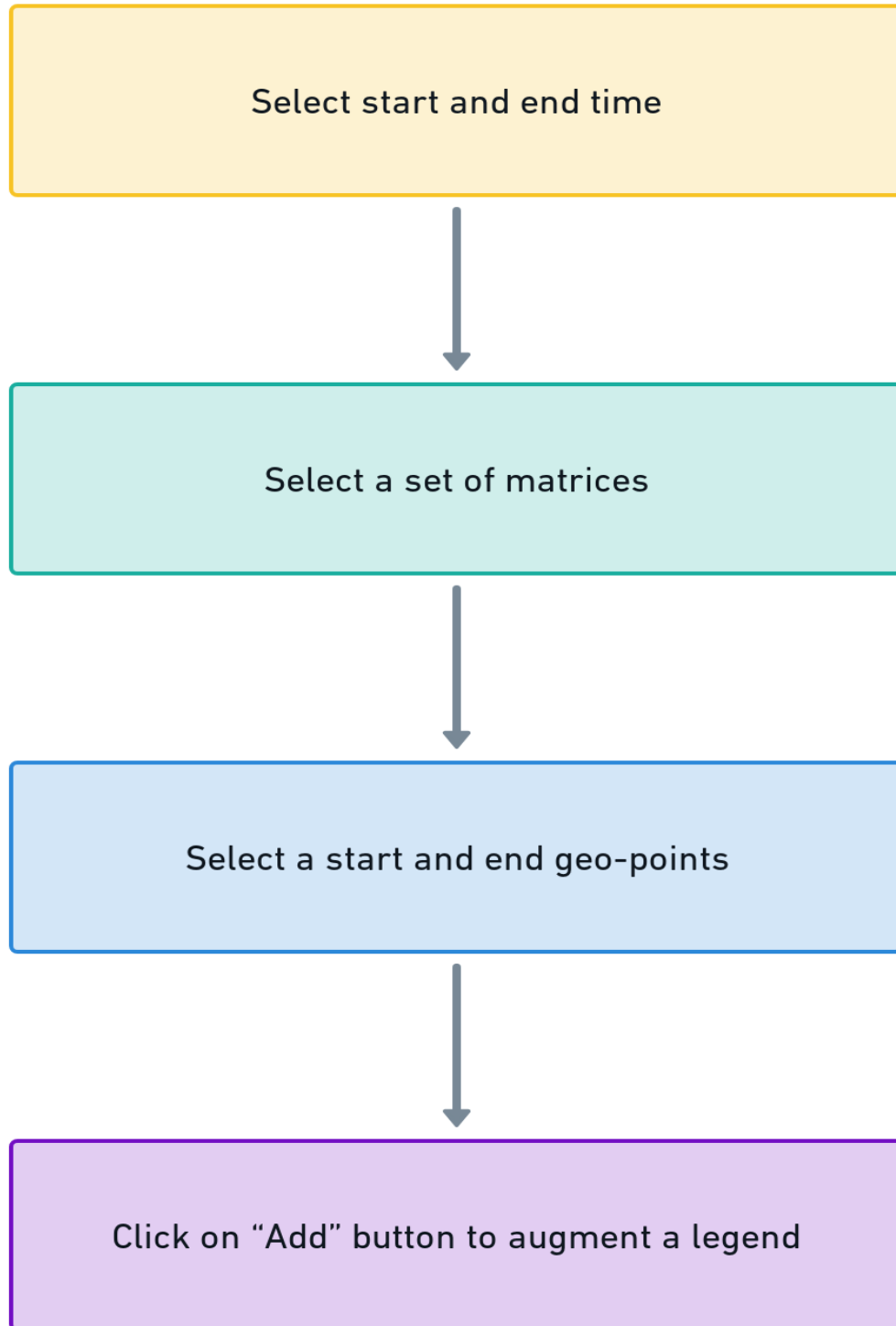
```
const [query, setQuery] = useState('');
useEffect(() => {
  const fetchData = async () => {
    const result = await axios(
      `<API end point>?query=${query}`,
    );

    setData(result.data);
  };

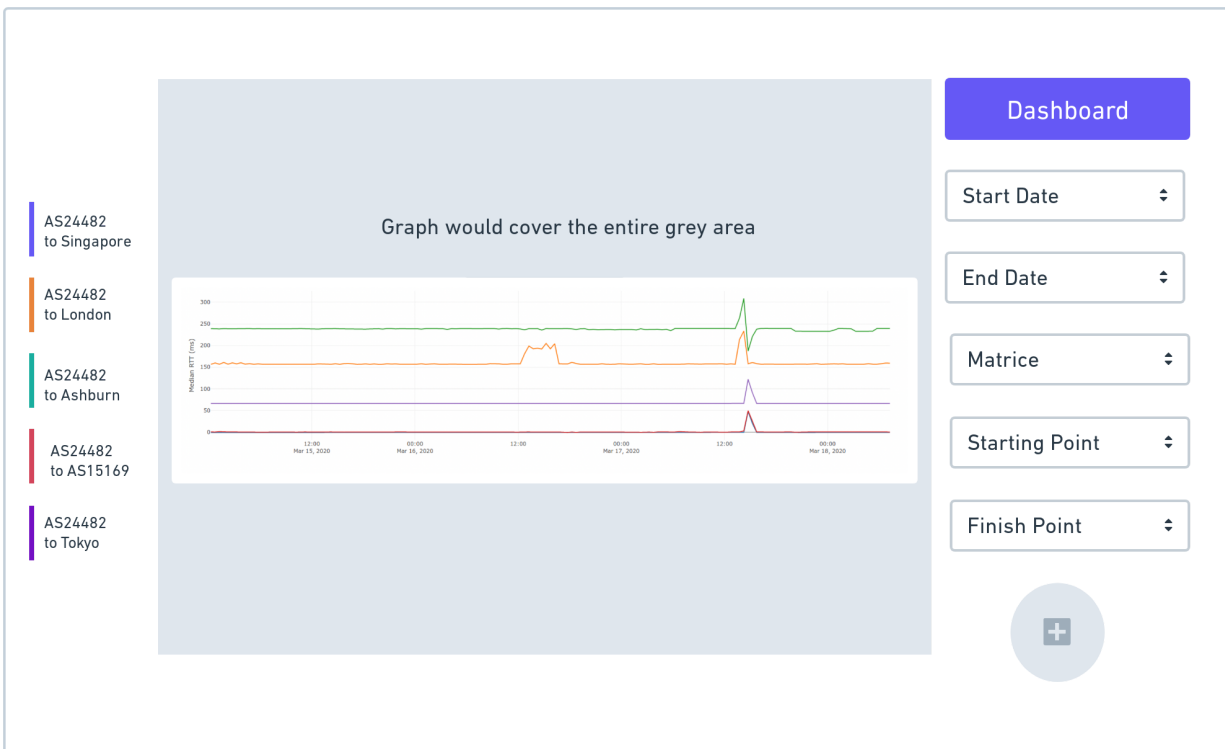
  fetchData();
}, [query]);
```

The exploratory dashboard would follow a simplistic approach for the convenience of the end-user. At the start, the user would select a start and end time which would have a precision of 15 minutes. Once the duration is determined, the user selects from a set of matrices which includes delay, dependency, throughput from MLab data, and traffic from transparency reports at its basic. After the duration and set of matrices are determined the user gets to select the geo-points with the

precision of ISO codes. For every legend chosen the user can add more legends for detailing and deleting as per the need with the delete button provided in the dashboard. The workflow for the user is as shown below:



Personally, I believe the UI should be simple because most of our concerned end-users are not developers. The way I have planned to implement the dashboard now is:



When users clicks on the “Dashboard” they can hide or remove legends from the following graph. For example the user would have the option to remove or hide the legend “AS24482 to London”. If the user wants to add more legends the “+” button will be there to guide them through.

Documenting it

Lastly, I will ensure that all methods have appropriate docstrings and comments wherever required so that it is easy for a new person to understand our code and contribute to our application. I will add proper documentation, so that anyone who wants to use it, can go through it and get started.

I think this part is really important though trivial, as with this extension the functionality will become much more convenient for those who want to use it as well as extend it and will save time for a lot of great

engineers and scientists. It will also help me showcase my project properly.

Since the users of the Repository and Library are going to be really great scientists and engineers, I don't want to idle their time. My documentation will be concise and to the point, while at the same time explaining everything in detail.

Though the initial documentation will be written as a GitHub ReadMe and will be done in the markup language. I will also add formal documentation, created with the help of Hugo's Docsy or similar after discussion with mentors. Again, like the UI this doesn't require a whole lot of intuition and henceforth I will be shifting it for the buffer period or Post Soc period depending upon the circumstances.

Plan of Action: Timeline

1st April - 20th May

- a) Getting familiar with the code
- b) Fixing bugs
- c) Resolve [#55](#)
- d) Resolve [#33](#)
- e) Remove Legends from [#53](#)
- f) Work on the responsive [#47](#)
- g) Resolving bugs that arises while resolving the above issues
- h) Build a more concrete understanding of IHR's work and research papers
- i) Discuss other project approaches with mentor
- j) Build test components with Plotly.js to create a concrete understanding of the module
- k) Request to create [.github repository](#)
- l) Create a GitHub portfolio while attaching COC, CG, and Security guidelines

20th May - 12th June

- a) Community Bonding
- b) Understand how to fetch data for the MLab integration
- c) Build test component after fetching data from [MLab](#)
- d) Research and devise a way to fetch data from [Cloudflare](#) and [Google](#)
- e) Create a basic UI for the explanatory dashboard in Figma
- f) Understand the alerting system and the threshold point
- g) Upgrade the modules while making sure nothing breaks in prod
- h) Create a CI/CD pipeline that would make the development process fast
- i) Create a knowledge graph for IHR
- j) Work intensively on ranking up the website

13th June - 29th July

- a) Add the functionality to auto plot speed test data
- b) Work on the legends of the MLab integration
- c) Integrate Cloudflare's traffic charts
- d) Resolve the bugs in the graphs, similarly faced in hegemony graph [#53](#)
- e) Start building the UI components for explanatory dashboard

30th July - 19th September

- a) Integrate Google's traffic charts
- b) Develop a comparison graph between Google, Cloudflare, and IHR traffic report
- c) Add functionality to the exploratory dashboard and make it scalable and functional
- d) Resolve all the bugs created during the implementation of three software subsystems
- e) Buffer time to resolve issues or feature decided before
- f) Write the report for final evaluation

Apart from the vital contributions: Extras

Now this will be a fun time and should not be considered as a part of the Goal.

- Contribute to other ongoing projects
- Fix Bugs and build some amazing features
- Try to improve the installation guide and the project readme in general
- Write an analogy of IHR's AS to Electricity generation, transmission, and distribution for end users to have a firm understanding of the internet.
- Write blogs to showcase the project we are building and what it aims for
- Make a video to showcase how can everyone get benefited from it
- Use tailwind or similar CSS pre-processor to revamp the application
- Add documentation for terminologies used in the IHR website so that people who are starting out with Internet Health Report analysis can get a kickstart along with developers whose only focus is to do code contributions.
- Run lighthouse and fix the corresponding issues to fire up the application and improve the application in terms of Performance, Accessibility, Best Practices, and SEO

The timeline might change according to planning or difficulty of the feature implementation or bug resolution. The prime focus would be to keep everything under the assigned timeline and add the extras if time permits.

Open Source contributions and Projects

I started contributing to opensource a few months back to large organizations. Before that, for more than 1.5 years I maintained projects for my campus club, [Webwiz](#), and another organization [Betaoverflow](#) to motivate beginners to contribute to open-source and personal projects on [Github](#). Since I started quite late contributing to IHR, I have a few merged PRs. I worked on other organizations and personal organization and have mentored more than 1000 people in software development being a mentor in several programs which promotes open-source, writing blogs, delivering talks, and founding and running a community!

Organization	Contributions
Internet Health Report	https://github.com/InternetHealthReport/ihr-website/pull/53
Internet Health Report	https://github.com/InternetHealthReport/ihr-website/pull/59
Internet Health Report	https://github.com/InternetHealthReport/ihr-website/pull/58
Internet Health Report	https://github.com/InternetHealthReport/ihr-website/pull/51
Internet Health Report	https://github.com/InternetHealthReport/ihr-website/pull/49
Internet Health Report	https://github.com/InternetHealthReport/ihr-website/pull/35
Internet Health Report	https://github.com/InternetHealthReport/ihr-website/pull/32
Internet Health Report	https://github.com/InternetHealthReport/ihr-website/pull/31
Internet Health Report	https://github.com/InternetHealthReport/ihr-website/pull/29
Internet Health Report	https://github.com/InternetHealthReport/ihr-website/pull/27
Moja Global	https://github.com/moja-global/community-website/pull/194
Margarita humanitarian	https://github.com/margaritahumanitarian/helpafamily/pull/220#issuecomment-939078006
Anitab Org	https://github.com/anitab-org/anitab-forms-web/pull/168

Fission	https://github.com/fission/fission.io/pull/70#event-5520449698
Fission	https://github.com/fission/fission.io/pull/73
AutoDL	https://github.com/Auto-DL/Auto-DL/issues/370#event-5418089297

Expectations from Mentor

Guide me to understand the existing codebase of the [IHR website](#), terminologies, and research papers published wherever I am incapable of doing so on my own. Suggest resources to have a clearer view of how things are done ideally. Help me come to a decision when I have more than one way of doing things and tell me why that is the best decision. Ultimately, take time to review my work and provide their timely insight while helping me in optimizing the code with respect to Time, Space, and CPU utilization complexities.

What are my other commitments?

My university end semester exams are scheduled from 22nd April 2022 to 2nd May 2022. **So I have no commitments in the summer other than GSoC.** I will be available for at least 40 hours a week through online platforms and am ready to extend whenever needed. I would be working full-time for GSoC. I have no exams so I shouldn't have any problem. Since I am a freelance content writer, I would accept offers for writing only after consulting the mentors.

What do I plan after the GSoC period is over?

As discussed before, I would like to keep contributing to IHR even after GSoC and will be available to resolve issues and manage to pull requests. Even if I am not selected this year, I would like to help this project by resolving issues, suggesting new ideas, and participating in discussions along with making code contributions. I usually help out people with code and will love to mentor some young coders for IHR projects. And I will like to keep contributing to the IHR organization and do whatever I can to help.

Who am I: About me

I am Aniket Pal, a pre-final year undergrad from the National Institute of Technology, Rourkela, India. I love creating web applications that stay on the internet and can be accessed by anyone. Currently, I focus on writing production-level code and have a knack for Design and DevOps. I am a freelance content writer, currently, I am writing for MUO(Make Use Of), Alan AI, and Mergify with offers pending. I consider myself to be a Hackathon addict. In the last 6 months, I have won 10+ international hackathons and even got funding from IBM for one of my prototypes to develop further. I have experience working with several organizations starting from leading a campus club to tech giants. I believe in sustainable development and currently, I am also a GitHub Extern with Numocity, where my focus is building EV-based software to automate vehicle charging. This spring I was one of the few MLH Fellows who graduated. During my fellowship, I collaborated with 15 fellows from 5 different countries while building production-level projects to make the world a better place to live with. Being the GitHub Campus Expert, I mentor students in Opensource.

The experience I possess: Places worked

- **Numocity**
As a Github Extern, Researched and developed a tool to optimize the inventory of EV batteries for Charge Point Operators which ultimately increased the efficiency in the supply chain of the business. In the process developed an algorithm to determine the SOH of the battery which is being used by the BMS manufacturer.
- **MLH Spring Fellow**
An intensive program collaborated with 15+ students from 4 different countries. Built out a portfolio of projects, and experimented with new technologies (React, Jekyll, Rust, Python) during a short, impactful hackathon sprint. The fellowship was backed by Meta, GitHub, Adobe, AWS, Intuit, and more. Lead the frontend team with delegating tasks, conducting daily stand-ups in order to build a highly performant & responsive application
- **Tech Mahindra**
Worked for a period of 2 months as a Backend intern and mostly around integrating APIs. Optimized the neural network in terms of latency and efficiency. Cannot disclose work because of Contract.
- **BeFinSavvy**
Developed a dynamic and interactive platform for financial report generation, with features like the dashboard, administration panel, mailing system, firebase authentication, google Pub/Sub, etc., that ensures automation in report generation and user satisfaction.
- **AiBorne Tech**
Dent detection with a color picker model for car service. Deployed the DLCV model, integrated with a web application for Maruti Suzuki. Managed the cloud servers and developed the web application solo.
- **Operational Research Society of India**
Designed and revamped user-friendly React web application. Fixed

bugs from an existing website and implemented enhancements that significantly improved its functionality as an organization

- **Needs Official**

Developed a cross-platform React Native mobile application for a college startup for buying and selling products on 30 different college campuses in India. Developed with modern UI elements, multi-level auth, push notifications, messaging, and multi-screens.

- **Hackodisha**

Founded and organized the largest hackathon in Eastern India. Hackodisha powered by Postman had over 1600 hackers, 40 + sponsors, and more than 1,00,000 social media reach making it one of the largest hackathons of the subcontinent. Being the Lead organizer communicated and kept all the working teams in sync. Meanwhile, took the web application from prototype to production.