Agricultural and Food Engineering Department Indian Institute of Technology Kharagpur,West Bengal ,India

---

Name of Organization	**:**	**AppViewX Pvt Ltd**
Role	**:**	**Summer Intern**
Training Period	**:**	**May 14, 2022 – July 29, 2022 (68 days)**
Project	**:**	**Dashboard for real-time API monitoring**

Submitted by:

Prashant kumar

18AG36019

Mentor:

Ashwin Balaji

Technical Head

Acknowledgement

I am extremely grateful to AppViewX Pvt Ltd for providing me this opportunity to complete my internship with them. It was delightful to work with a great team under an excellent learning environment.

I would like to express my gratitude towards my mentor for the whole internship period, Mr. Ashwin Balaji. It was such a pleasure to work under his guidance on this project.

I am always thankful to my family, teachers and friends for their encouragement and support.

appviewX
www.appviewx.com

Aug 03, 2022

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Prashant Kumar** (Reg.No.: 18AG36019) from **Indian Institute of Technology, Kharagpur** completed his Internship with **AppViewX** for the period May 24, 2022, to July 29, 2022.

His performance and conduct during the course of the Internship has been good.

For AppViewX Private Limited

Nivrutha Sampath Vice President – Human Resources

Table of Content

Topic	Page
About the Organization	5
Introduction to the Project	6
Objectives	6
Overview	6
Details of the project tools	8
Application Programming Interface	8
Chrome Developer tools	9
Grafana web application	10
Workflow of the Project	11
Conclusion	14

About the Organization

AppViewX is a product based corporate firm, namely CERT+ and ADC+. It is revolutionizing the way NetOps and DevSecOps teams deliver Machine Identity Management and Application Delivery Automation solutions to enterprise IT. The AppViewX Platform is a modular software application that enables the automation and orchestration of network infrastructure and public key infrastructure (PKI) using an intuitive, context-aware, visual workflow. It quickly and easily translates business requirements into automation workflows that improve agility, enforce compliance, eliminate errors, and reduce cost.

A key player in the machine identity management space, AppViewX CERT+ helps secure some of the most demanding Fortune 1000 organizations in financial services and banking, healthcare, oil and gas, manufacturing, and high tech.

CERT+ is a next-gen machine identity and PKI management suite that allows for end-to-end automation of certificate and key life cycles across environments and vendors. It makes certificate management streamlined and efficient, allowing for endless upward scalability and cryptographic agility.

Throughout my internship period, I worked with the CERT+ product-based team of its Bangalore based office.

Introduction to the Project

Objectives

The project aimed to achieve the following:

- To create a dashboard for real-time monitoring of APIs.
- To test the APIs and set corresponding ranges for its response time.
- To classify each API call into GOOD / WARNING / BAD category based on response time and mode of call.
- To create a visualization chart for each API using Grafana Web Application.

Overview

CERT+ (one of the products of AppViewX) deals with digital certificate life-cycle management and aims to automate many processes related to it such as issue, renewal, revoke, expiry and so on. It serves many different certificate authorities across the globe.

Internal working of the whole process involves a lot of complex and sequential API calls at almost all the moments. This leads to the need of a real-time monitoring system for the APIs- to monitor their health and response time.

Now there are certain factors which determine the response time of an API. A function call to API may be synchronous or asynchronous based on its request and response methods. If it makes a request to the server and halts any further process until it receives the response, we term it as a synchronous call otherwise asynchronous. Hence, response time is dependent on the type of function call. Also, some APIs are bound to make a sequential call, in this case the absolute response time of an API is affected by its preceding APIs.

We aim to classify an API call based on its response time as GOOD / WARNING / BAD. Now, even after we receive the response time of an API, we must determine some threshold values to determine the range for each of the classes.

This can be achieved by storing the response time of an API to multiple manual calls and setting their mean value as the threshold for the first class (GOOD).

We also need to have a visualization dashboard which allows us to monitor the APIs in an efficient manner. This was achieved using the Grafana Web application.

Details of the project tools

This section covers the computer science concepts in the core of the project and different tools used in the project.

Application Programming Interface

An Application Programming Interface, or more commonly referred as API, is a set of programming codes that enables data transmission between one software product and another. It also contains the terms of this data exchange.

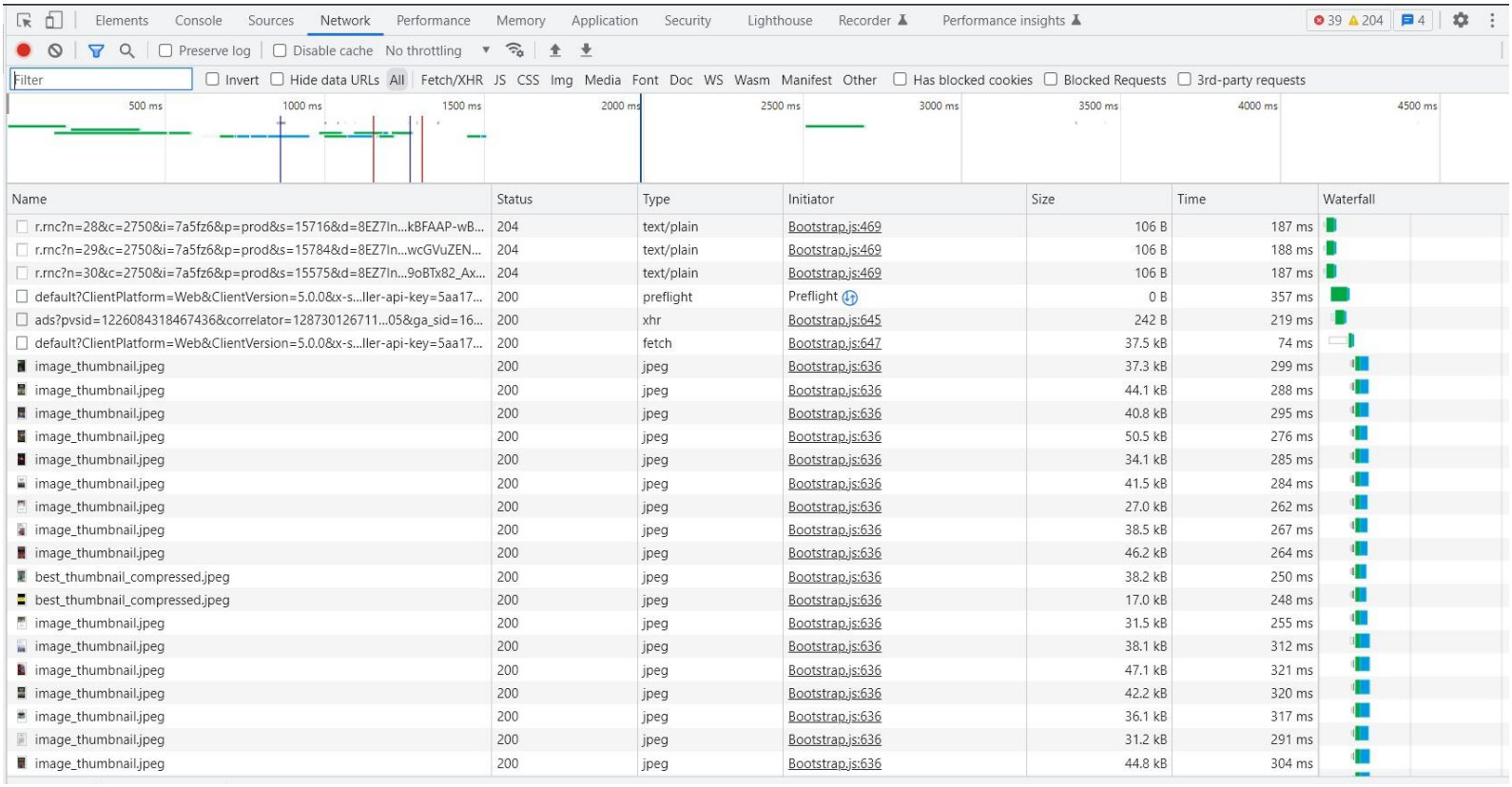
Application programming interfaces consist of two components:

- Technical specification describing the data exchange options between solutions with the specification done in the form of a request for processing and data delivery protocols
- The software interface is written to the specification that represents it.

Each API contains function calls – language statements that request software to perform actions and services. Response time is defined as the time taken by the API to fetch the response from the server after the request is made.
These days, the most used APIs are REST APIs.

Chrome Developer tools

Chrome DevTools is a set of web developer tools built directly into the Google Chrome browser. DevTools help edit pages on the fly and diagnose problems quickly, which ultimately helps to make better websites faster.



Name	Status	Type	Initiator	Size	Time	Waterfall
r.rnc?n=28&c=2750&i=7a5fz6&p=prod&s=15716&d=8EZ7In...kBFAAP-wB...	204	text/plain	Bootstrap.js:469	106 B	187 ms	
r.rnc?n=29&c=2750&i=7a5fz6&p=prod&s=15784&d=8EZ7In...wcGVuZEN...	204	text/plain	Bootstrap.js:469	106 B	188 ms	
r.rnc?n=30&c=2750&i=7a5fz6&p=prod&s=15575&d=8EZ7In...9oBTx82_Ax...	204	text/plain	Bootstrap.js:469	106 B	187 ms	
default?ClientPlatform=Web&ClientVersion=5.0.0&x-s...ller-api-key=5aa17...	200	preflight	Preflight	0 B	357 ms	
ads?pvsid=1226084318467436&correlator=128730126711...05&ga_sid=16...	200	xhr	Bootstrap.js:645	242 B	219 ms	
default?ClientPlatform=Web&ClientVersion=5.0.0&x-s...ller-api-key=5aa17...	200	fetch	Bootstrap.js:647	37.5 kB	74 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	37.3 kB	299 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	44.1 kB	288 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	40.8 kB	295 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	50.5 kB	276 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	34.1 kB	285 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	41.5 kB	284 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	27.0 kB	262 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	38.5 kB	267 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	46.2 kB	264 ms	
best_thumbnail_compressed.jpeg	200	jpeg	Bootstrap.js:636	38.2 kB	250 ms	
best_thumbnail_compressed.jpeg	200	jpeg	Bootstrap.js:636	17.0 kB	248 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	31.5 kB	255 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	38.1 kB	312 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	47.1 kB	321 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	42.2 kB	320 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	36.1 kB	317 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	31.2 kB	291 ms	
image_thumbnail.jpeg	200	jpeg	Bootstrap.js:636	44.8 kB	304 ms	

DevTools consists of many panels which collect different pieces of information on a webpage. Some of the panels are:

- Elements panel
- Console panel
- Sources panel
- Network panel
- Performance panel

- Memory panel
- Application panel
- Security panel

The most utilized panel throughout the project was Network Panel.

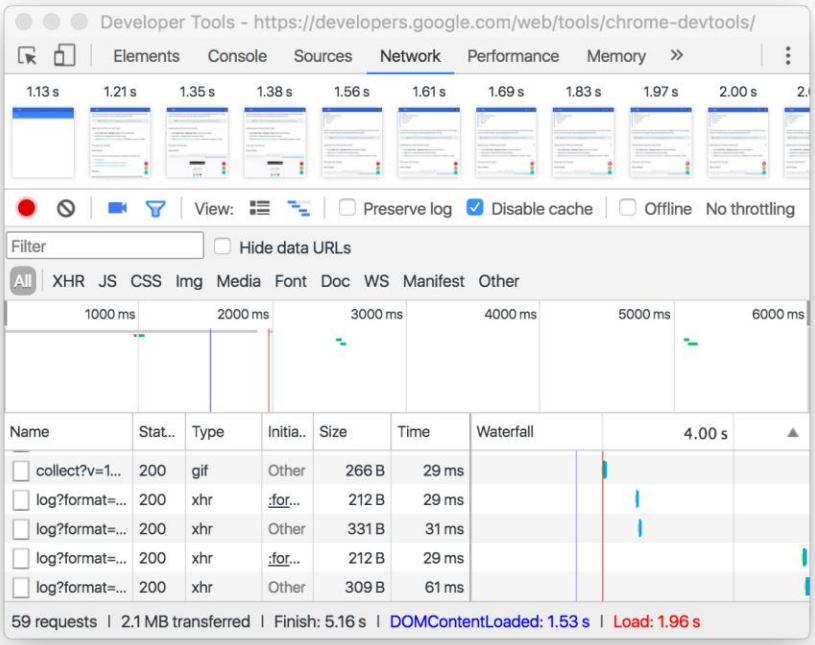


Fig. A network panel of DevTools

Grafana web application

Grafana is a multi-platform open-source analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected to supported data sources. It also supports additional capabilities such as a self-hosted installation or an account on the Grafana Labs cloud service. It is expandable through a plug-in system. End users can create complex monitoring dashboards

using interactive query builders. Grafana is divided into a front end and back end, written in TypeScript and Go, respectively.

Fig. Grafana Web Application (Analytics and Charts)

Workflow of the Project

The whole workflow of the project can be divided into following steps:

1. Collecting data

This step involved making manual calls to trigger an API and collect its response time to estimate time range for different classes of the API.

In this phase, almost 300 APIs were tested, totaling to almost 900 API calls. For each API, the mean of the response times was assumed to be the threshold value for the first class (GOOD). Anything beyond this value would be classified as WARNING / BAD.

For example, if we set the range for each class of response time for as API as:

0-5 second as GOOD

5-10 second as WARNING

More than 10 second as BAD

In this case, we aim to automatically classify a particular API call into one of the three categories based on its response time.

The range also depended upon other factors such as mode of call (synchronous/asynchronous) and whether the call is to be sequential or not.

1. Implementing queries

This step tries to automate the process of classification of an API call by using Prometheus queries on the response time of an API call.

For example, we take a query as:

GOOD range query: 0-5 second

(((sum(requests_latency_seconds_by_apiid{apiid="cert-ca-entrust-submit", quantile="0.99"} > 0)) <= bool 5) == 1) * 0

WARNING range query: 5-10 second

(((sum(requests_latency_seconds_by_apiid{apiid="cert-ca-entrust-submit", quantile="0.99"} > 0)) > bool 5)*((sum(requests_latency_seconds_by_apiid{apiid="cert-ca-entrust-submit",quantile="0.99"} > 0)) < bool 10)) == 1) * 1

BAD range query: more than 10 second

(((sum(requests_latency_seconds_by_apiid{apiid="cert-ca-entrust-submit", quantile="0.99"} > 0)) >= bool 10) == 1) * 2

Similarly, we could write a query for each API to classify its responses.

1. Creating visualization plot

Now that we have already classified an API call, we must have a convenient way to visualize this whole time series data so that we can identify any malfunction in the internal calling efficiently. This was achieved by mapping the query results to Grafana web application and then creating a time series plot of the same. The different categories were color coded for convenience. GOOD was coded to GREEN, WARNING was coded to YELLOW and BAD was coded to RED.

Fig. Showing the time series plot of Certificate report API in Grafana web application

Conclusion

By the end of the project, we successfully setup a dashboard for real-time monitoring of the APIs based on their response times, mode of call etc. We automated the whole process by writing queries on top of the response time data of the APIs. Furthermore, we were able to provide a convenient and easy visualization tool for the time series data of API responses using Grafana web application.