Before launching the program, we have to setup a few things

1. Have python3 installed

2. Install pip3

3.Create the virtual env, in the console go to the folder where the code and the other files are and type: `python –m venv venv`

4. Activate the virtual env in the same console type: `source venv/bin/activate`

5. Install validators, in the same console type: `pip3 install validators`

6. Install requests, in the same console type: `pip3 install requests`

7. You can now launch the app with: `python3 monitor.py`

**Warning: before launching, the console has to be big enough (about 200 characters width and 60 line height) I recommend putting it fullscreen.**

You can pass different arguments to the program:

> --file "file.txt": this imports a file located in the same folder in which websites and intervals are listed and will be monitored,an example file "testfile.txt" is already present in the app folder
>
> --website "https://google.com" --interval 2: these two arguments have to be given together, they launch the program with a site already being monitored with the given interval
>
> --test 1: Instead of launching the program this launches a test of the alert systemThe tests consist of launching the thread that monitors sites which may be down with a fake site It then simulates an metrics for that site and check wether the thread behaves like it should
>
> Three tests exist (--test 1, --test 2, and --test 3) :
>
> Test 1 simulates a site that goes under 80% availability for more than 2 minutes and then comes back at more than 80% availability for more than 2 minutes.
>
> Test 2 simulates a site that goes under 80% availability for more than 2 minutes and then comes back at more than 80% availability for less than 2 minutes, goes back under 80% for some time and finally comes back over 80% for more than 2 minutes.
>
> Test 3 simulates a site that goes under 80% availability for less than 2 minutes and then comes back at more than80% availability for more than 2 minutes.

A final option exist for testing the program: I have created a fake site hosted on heroku. Every 6 minutes it changes from respondingwith status code 200 100% of the time to 60% of the time (40% 404 and 500).

This test is long but you can launch the program do something else and check from time to time, you should have errors popping in the current alert and finally being resolvedand printed in the past alerts (like in the image below). This site is listed in the « testfile.txt » and the address is https://fake-site.herokuapp.com

When you launch the program you will need to wait 10 seconds for the display to refresh before you can see the sites metrics

Here's how the program works:


In the top left corner you can add new websites with their complete url (https://domain.com) and hit enter to continueYou will now be asked for the interval in seconds you can put any number int or float, press enter again confirm your input

If your input has no problems (incorrect url, interval NaN...) the window will go back to its original state. Wait 10 seconds for the display to refresh and you will start to see the site metricsIf you made a mistake in typing the site or interval you can't erase and rewrite, hit enter until you get an error message like « incorrect url »

hit enter once more to get the window back to its original state and start again.

**To close the program enter « exit » in the url input. You have to wait no more than 10 seconds for the threads to stop andthen the console should be good to use again.**

In the top right corner is the alert window. There are two parts to it, current alerts for sites that are

currently down,and passed alerts for sites that are were down but have come back up.In the current alerts the alerts are red when the site is under 80% availability and green when it comes over 80% availability, if it stays above 80% for more

than 2 minutes it will be removed and a new log will be added to the passed alerts.

In the middle are the two displays one refreshed every 10 seconds which shows the metrics for the past 10 minutes, the other refreshed every minute whichshows the metrics for the past hour

If you start the program with sites passed as arguments, you will have errors printed in the bottom center of the window if the arguments don't match the correct format

Improvements:

If there a re more than 20 monitored sites or more than five current or passed alerts the display will overflow and that case is not handled, I should add warning and stop the addition of more sites and erase old answers so that it doesn't overflow

The main problem with the program is its spatial complexity. It saves in memory metrics of each request for each site, this object quickly becomes very big. Metrics older than an hour are useless and should be erased, I would have done a thread which every (5-10) minutes erases old metrics from the list, if I had more time. A better solution could be to save in memory just the average that we display, that way we would only have to object for each site in memory, the 10 minute average and the hourly average). But the problem with this solution is that the operations made between each request, on the monitor thread, have to be as quick as possible so that the interval between requests given by the user is respected. Calculating the average each time could be to long to respect the interval.

In terms of just improving upon this program I would probably create an interface (probably web) and have the program run as a server on which the interface connects, it would allow for more flexibility and a nicer design. It would also allow the user to have a better input and change more things while the program runs. Separating the interface and the actual monitoring would help have a clearer program too, the server would just have to handle monitoring and alert handling the rest would be handled by the web interface.