Corey Sabia, Matt Ruch, Dhaval Patel
Comsc 230
Group Project

Network Load Detector

What problem does it solve?

NLD (Network Load Detector) solves the problem of businesses that have a need for a stable network because it is critical to them that the network maintains 100% uptime. This program was created to allow network administrators to be able to detect problems on the network. This program allows for multiple nodes on the network to push data such as: back end load time, front end load time, and total load time. With this data the network admin would be able to discern if the detected problem was server side or client side. In other words if the problem is on the local network or on the internet.

Why is this needed now?

This solution is need now due to the vast use of the internet. Nowadays the internet as a role in almost every part of our day to day lives. With this product when a hiccup occurs it will allow for easy detection of the problem and offer a lot of troubleshooting data to generate a fix as soon as possible. Networks have become essential for many companies and home users over the years, any down time could potentially hurt bottom lines.

Day-to-Day Use

A network administrator of a small to large company or community. This program can be run on one node or hundreds so it could even be used on a home

network if the need is there. It will most likely be used for any business where constant internet access or very stable network access is absolutely necessary.

Competitors and What Makes NLD Better?

Our app is better than others that are on the market due to that fact that it is hosted in the cloud and only a small very low power physical machine has to be on the network. The program itself is very lightweight therefore it can be run on existing hardware without much overhead. The program is sent to clients as a docker image so all of the dependencies are already built in and require no setup. The only set up is to have a kibana/Elasticsearch setup and create profiles to send the data to.

Marketing Plan

Package Plans

The app will probably end up with a basic free version with premium versions available. This can be done due to the very low system requirements of the program and all of the database and view layer can be done in the cloud. With the premium version there will be more support for the database being set up and more consistent updates due to that fact a new docker image will have to be made for every update as well as purpose built hardware for the local node.

Pricing

Tiered versions that start at a free basic app with some reduced functionality all the way up to a 24/7 support version with an optional technician. There will be several tiers of paid for versions of the product. The highest end package, $4000,

will include 24/7 technical support along with purpose built hardware for the nodes on the network. On the other end the lowest end paid for package, coming in around $1800, will have less technical support hours along with no hardware included.

Distribution

The app will be sent out in a docker image and the database can be setup with AWS or locally with a different service therefore all the client will need is a local machine to run the program on.

Venture Capital Needed

For this project to get off the ground venture capital is needed to fund the project. We will need in the ballpark of $300,000 to get this project off the ground

Employment

Several employees will have to be hired to keep this project up to date as well as sell packages.

3 Programmers: 75,000 per year

We need to have 3 programmers to make the necessary changes to the program as well as properly test it and ensure it is ready for public release.

1 Marketing/Management : 75,000 per year

This person's responsibility is to keep the project on track as well as sell as many copies as possible.

Projected Market-Share and Revenue, Year 1

Not yet deployed in the market.

Competitive Advantage

Our competitive advantage is that anyone who needs a network monitor now has access to one, with a free tier we can build up trust with users who are reluctant to pay for a package and then in the future may opt to upgrade. In addition to this with several levels of packages there is a package that falls in their budget. In addition, the free tier will give the users an outlook on the ease-of-use and automation.

Technical Summary

Model - Database Design

The design of the database is for ease of use for the consumer, it is very easy to navigate and see the data that you want to chart. Elasticsearch is the database being used. The unique thing about Elasticsearch is the indexibility of the data. Any piece of data is easily searchable and can be used for various purposes without much effort.

View - Kibana

Kibana is a plugin for elasticsearch for visualization of the data. With Kibana, the performance data on the database are plotted in an interactive and clear way. Kibana also lets the user customize the plots. Some of the most useful options are customizing live time plots for certain data and data sources. Because of these

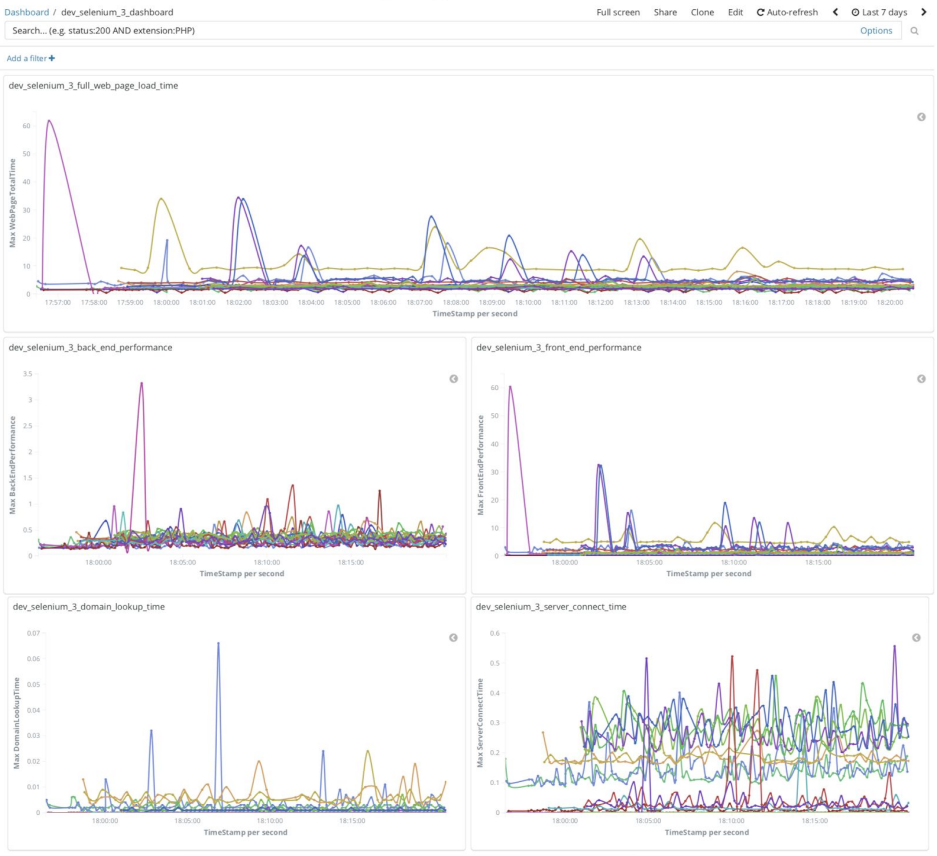
customization options, Kibana allows a robust management and monitoring option for users. (See figures below)

Controller - Python, JavaScript

The controlling program that actually retrieves all of the data to be stored in the database was made in python and used javascript, selenium, cron and chromedriver. With all of these combined into the program we were able to open up chrome and point it to a website. This was done by importing selenium and telling it where the chrome driver was located in the directory. With this library now available python was able to launch chrome and run javascript functions on it. The javascript allowed us to pull performance data of the website such as dns server times and frontend/backend load times. This was then put into a json format and pushed to AWS where our instance of Elasticsearch was running. All of this was done within a pip environment to allow for easy update of dependencies. When major development was finished, a Docker image was created for easy deployment and management of dependencies.

Software Design and Management Goals:

For this program Github was used to allow the team to have the most recent code without any inconsistencies. The code could use some dependency optimization due to the way docker requests them many more are installed than necessary.



dev_selenium_3

Count	9866
Size in bytes	4.39 MB
Query total	16000

Mappings ▾ json

- BackEndPerformance *float*
- DomComplete *date*
- DomContentLoadedEventEnd *date*
- DomContentLoadedEventStart *date*
- DomContentLoadedEventTime *float*
- DomainLookupEnd *date*
- DomainLookupStart *date*
- DomainLookupTime *float*
- FrontEndPerformance *float*
- LoadEventEnd *date*
- LoadEventStart *date*
- LoadEventTime *float*
- NavigationStart *date*
- ProgramTotalTime *float*
- ResponseStart *date*
- ServerConnectEnd *date*
- ServerConnectStart *date*
- ServerConnectTime *float*
- Source *text*
- TimeStamp *date*
- WebPage *text*
- WebPageTotalTime *float*