Team Information

Team name : StonyBallerz
Project number : 4 **(github)**

Team Distribution

Burak Torman: Backend, set up server and queue system. Connected front-end with back-end.
Alvin Joseph: Frontend and some backend for DB connection and queries.
Kathan Shah: Made tables Schemas and some frontend code.
Preet Patel: Made the backend code for generating typosquat Variants and used puppeteer to crawl domains.

Description

In this project, we have designed a distributed typosquatting detector. Our architecture consists of a master node and N worker nodes. Master receives a post request from the client with the domain name to be processed, generates typo variants and creates a job for each one. Workers register and consume the jobs to process it, then return the result to master. Each worker takes care of one job at a time by visiting this typosquat domain using a Puppeteer crawler, takes a screenshot of the page and fills a buffer with the source HTML code. Once we have all the components, this data is returned to the master, and is stored in the database as well as displayed on frontend. The master then assembles these in an HTML report which the user can inspect using the web dashboard.

Architecture:

Frontend HTML, CSS and JavaScript
Backend: NodeJS, ExpressJS, MySQL, Redis server, puppeteer

- We chose NodeJS because the documentation for Headless Chrome Crawler was using Node Js. There were simple node packages to implement master/worker system and to also learn a new technology along with the project.
- We chose Redis because all other similar libraries seemed more complicated to set up.

Instructions:

Install Redis (see link below) and NodeJs then all packages must be installed from the node package manager.

- bee-queue, Express, mysql, async, puppeteer.

1-) Start Redis server(must)

1. Open shell.
2. Cd to redis-5.0.7
3. Type “src/redis-server”, now the redis server will be running.

2-) Start master node

1. Open another shell.
2. Cd to our src/ directory.
3. Type “node master.js”, now the server will be running.

3-) Start worker node

1. Open another shell.
2. Cd to our src/ directory.
3. Type “node worker.js”, now one worker node will be registered.
4. To scale and register more workers repeat steps 1-3.

4-) Open the app

1. Open a web browser.
2. Type “localhost:3005” for the url.
3. A homepage will appear, click the start button to proceed.
4. Now you will be in the main page, enter a domain name you want to test to text field(www.google.com).
5. After 45 seconds results will pop up.
6. Click on any result.

Other Info

- We connected over localhost but redis allows us to connect over remote connections, so instead of opening workers on the same machine on different shells, you can create workers on a remote machine and register to master. To do so when you start redis server the specific host and port number needs to be specified along with our worker.js and master.js file where we specified localhost. Use the section “Configure Redis Remote Access“ for the following link: https://linuxize.com/post/how-to-install-and-configure-redis-on-ubuntu-18-04/