I have worked in the transparency company.

Please check the company with this link :

https://www.lsaranktracker.com/

I have built the LSA rank tracker with the Laravel framework.

The main mission of this project is to track your keywords’ positions in Google LSA and help you increase your qualitative traffic from local search ads.

I used the Laravel Livewire framework and Vite for the front-end and used the MySQL database.

Please check the web application with this link:

<https://app.lsaranktracker.com/>

Challenges:

1: Reducing the responsiveness time using background processing of Laravel Queue and Job.

In this project, we bring the LSA ranking data from API.

We have to bring the data for each point in radius 30 km from selected location. In this case, we usually have the 60 ~ 300 ranking points.

This means we need to call API 60 ~ 300 times.

If we use the original foreground method, we can't reach out the responsiveness time because each API calling needs about a second.

So, I processed that in background using Laravel queue and used the parallel method with several queues.

Also, found the best number of process according to the state of server.

2: Data fetch using scheduler.

In this project, we have to get the ranking data every five minutes for premium account and every an hour for free account.

Also, we have to generate the weekly report automatically so that user can see their business ranking, reviews, competitors and so on.

In Laravel, to implement this function, we usually use Cron.

But recently, we don't use the Cron for safe processes in server side.

So, I used the Laravel Scheduler.

I built the customized php artisan commands and connected to scheduler.

And then registered the scheduler in the server.

Every schedule process was perfect.

3: API calling with Random proxy.

To get the LSA ranking, we have to use the proxy.

But sometimes, the proxy fails due to proxy server issues.

It was a very serious error.

To fix this problem, the special method has been implemented.

First, I created the database table with 500 proxies.

And then created the failed proxy table in database.

When we call the API to get the data, we select the random proxy from 500 proxies.

If the proxy failed, it is sent to the failed proxy table. And then we choose the another random proxy.

We save the failed proxy table and so, we don't use the failed proxies again.

This method was very efficient and solved the proxy failed error.

4: Screenshot google map with ranking points for report.

To generate the report, we have to screenshot the Google map with ranking points so that the user can see the rankings in the map.

I used the Puppeteer Node.js library.

The main problem is that cPanel sever doesn't support the Chrome GUI mode in general mode. In this case, we have to install the additional Graphic environment such as Xorg.

But, doing so adversely affects the server performance.

So, I used the Puppeteer headless mode and, in this case, we don't need to install additional environment in server because it uses headless mode of browser.

5: PHP version management.

When I built the project in Local, I used the latest PHP version.

However, cPanel server supported only PHP 8.1.9.

So, I managed the PHP version.