Chrome Extension project - Documentation

one page of explanation of how your web scraping tool works, the technologies you used and why you chose to build it in that way.

➙ How does our web scraping tool work?

1) Introduction:

Puppeteer is a Node library that provides a high-level API to control headless browsers (browsers without a graphical user interface). It's often used for web scraping, automating tasks in a browser. It can scrape the data from the web by using some function of it like evaluate. It can click on the button, scroll the window, focus, wait for the content loading etc. headlessly.

2) Installation and import:

```
npm install puppeteer
const puppeteer = require(‘puppeteer’)
```

➙ Technologies we used :

1) Technologies we used for Backend:➙ Node.js (web framework)

Installed packages:➙ puppeteer, body-parser, cors, dotenv, express, moment

2) Technologies we used for Frontend:➙ HTML, CSS, JS

3) manifest.json file :➙ Need for creating a chrome extension.

➙ Why we chose to build it in this way :

1) First of all, In Backend we use **Node.js** framework because it has web scraping tools like **‘puppeteer’** and **‘cheerio’** and also I am very familiar with NodeJS.

2) We use **‘puppeteer’** in both web scraping tools because, **‘cheerio’** can work well and easy with static web-pages, and on second side **‘puppeteer’** has very extraordinary features of control the web browser and pages, which is need for this project, so we go for that.

3) Also, for Frontend we need to build a simple UI which has form, so that we use technologies, which is basic for making a website UI.

→ **How to run project :**

1) Clone the repo

2) Go to your Terminal for backend repo and hit **“npm install”** which is install all dependencies project had. And then hit **“npm start”** for start the server.

3) Go to **“chrome://extensions/”** and on **“developer mode”** and upload **“manifest.json”** file available in the frontend repo.

4) **Ta-da!!,** your project ready to use.