**Technical task for Frontend Engineer**

**General requirements:**

The following task contains 4 separate tasks that you are required to complete. Please carefully read the instructions and make sure you understand the task before its implementation.

For all 4 tasks you are required to record the whole process of task implementation with explanatory comments from your side. Feel free to explain the reasons behind choosing a specific tool, architecture or implementation philosophy. Please make sure your microphone and camera is on, and you’re sharing your screen while working on your tasks.

Submission requirements:

1. The video file of your task implementation.
2. All of required files with your code (GitHub link).
3. Deadline for submission: March 12, Tuesday 18:00.
4. Please send your submission to [anar@polygraf.ai](mailto:anar@polygraf.ai) email.

Feel free to contact me if you have any questions,

Good Luck!

**Task #1**

**Title**

Create a Chrome Extension for Amazon Product Data Scraping

**Task Description**

Your task is to develop a Chrome extension that allows users to scrape product data from Amazon. The extension should have a single button labeled "Scrape," and when clicked, it should scan the currently open Amazon product page and display the data in the specified JSON format in the terminal.

**Requirements**

1. **Chrome Extension Setup**

- Create a Chrome extension that includes a manifest file.

- Add a popup with a single button labeled "Scrape."

2. **Scraping Functionality**

- When the "Scrape" button is clicked, your extension should identify the currently open Amazon product page and scrape the following information:

```

{

"product": {

"in\_platform\_id": "Product ASIN/ID",

"title": "Product Title",

"platform\_rating": "Product Rating (if available)",

"img": "Product Image URL",

"reviews\_count": "Number of Reviews",

"price": "Product Price",

"seller": {

"in\_platform\_id": "Seller ID",

"name": "Seller Name",

"profile\_url": "Seller Profile URL",

"platform": {

"name": "amazon",

"url": "https://www.amazon.com"

}

},

"url": "Product URL"

},

"reviews": [

{

"in\_platform\_id": "Review ID",

"review": "Review Text",

"is\_recommended": "Is Recommended (true/false)", true if number is >3

"platform\_rating": "Review Rating",

"author": {

"in\_platform\_id": "Author ID",

"full\_name": "Author's Full Name",

"profile\_url": "Author's Profile URL"

}

},

// Additional reviews follow

}

}

```

3. **Display Data**

- Display the scraped data in a user-friendly manner in the terminal.

4. **Error Handling**

- Implement error handling to notify users if the extension is unable to scrape data from the current page, or if there are any other issues.

5. **Testing**

- Write tests to ensure the functionality of your extension. Consider testing against various Amazon product pages to ensure compatibility.

**Deliverables**

1. A working Chrome extension that scrapes and displays Amazon product data.

2. Source code and any necessary build files.

3. Testing documentation and any necessary setup instructions.

**Bonus Points (Optional)**

1. **Options Page**: Create an options page where users can configure the extension's behavior or customize the output format.

2. **Rate Limiting**: Implement a rate limiting mechanism to prevent excessive scraping.

**Task #2**

**Title**

Next.js task

Make a page with next.js app router that retrieves a given post from `[https://jsonplaceholder.typicode.com/posts/{N}](https://jsonplaceholder.typicode.com/posts/%7BN%7D)` based on path parameter `N`. The page should have SSR but also work during client-side navigation, preferably without an unnecessary round trip to the next.js server. Add Next/Previous post buttons at the bottom. Using react-query (aka tanstack query) and plugins for it is welcomed, as for data fetching library, anything goes but we use <https://github.com/sindresorhus/ky>. As usual, annotate everything with types.

**Task #3**

**Title**

Typescripttask

Please refer to the “typescript.ts” file for the task description.

**Task #4**

**Title**

React task

Please refer to the “App.tsx” file for the task description.

Path: react/src/App.tsx