CS410 Team literallyAnything Final Project Documentation

Archna Sobti (archnas2), Anjali Kumar (anjalik4), Aryan Vakharia (aryannv2), Manu PIllai (mpillai6)

---

[1] /setup does not exist within our repository and is only a sublink here for document navigation. However, /frontend and /backend do reflect our repository's actual structure

Set Up:

First, clone the repository. Then, run the following commands, starting from the root of the repository.

Frontend:

```
cd frontend         // Navigates to the frontend folder
npm i               // Installs necessary dependencies
npm run build       // Builds project for usage in Chrome
```

Go to Google Chrome, click on the extension icon in the browser navbar, and select “Manage Extensions.” Then, click “Load unpacked” in the top left corner of the page. Navigate to /frontend/dist, and select the folder. You should be able to select the extension “News Article Summarizer” when you click on the extension icon in your Chrome navbar.

Backend:

```
cd backend          // Navigates to the backend folder
npm i               // Installs necessary dependencies
node server.js      // Runs the backend server
```

The backend uses Python 2.7. Please create an environment using this version of Python and pip install the following dependencies:

- metapy
- bs4
- gensim
- html.parser

Once you have started running the backend, you can start passing URLs for news articles to the chrome extension for summarization! Just paste the URL in the extension’s text box and click “Submit.”

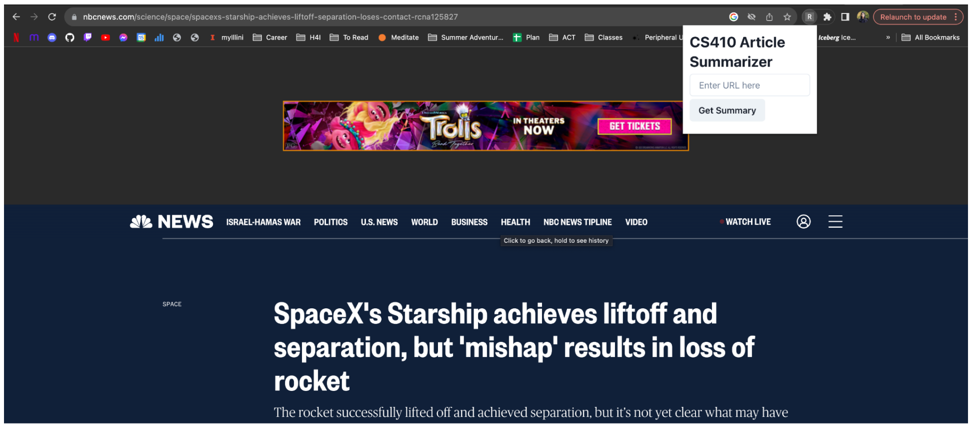
Frontend

> /frontend/dist/manifest.json

- This file is used by the Chrome Extension to know what it can and cannot do
  - For example, it can access the current active tab

> /frontend/src/App.tsx

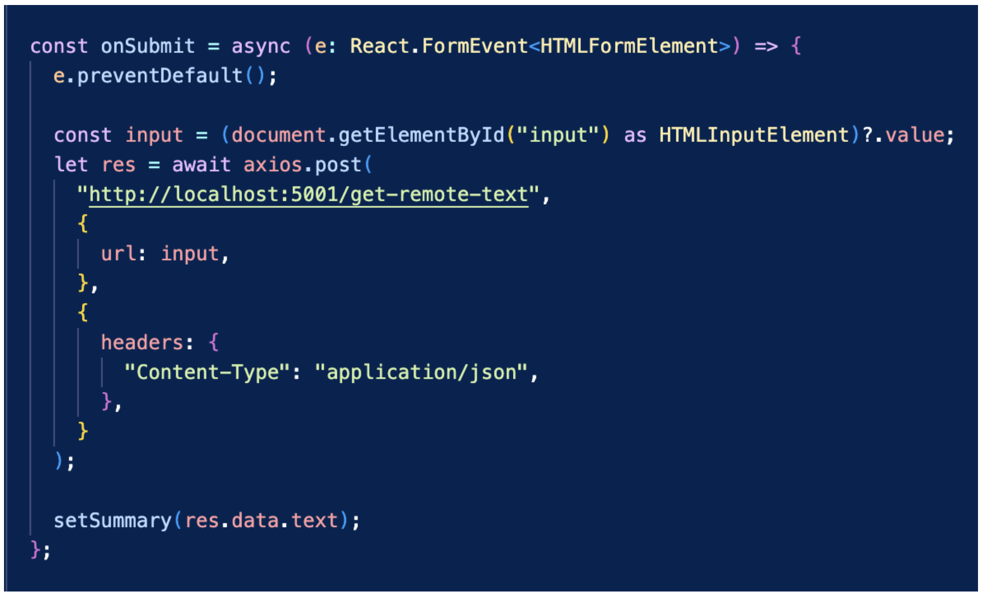
- This file displays the UI for the chrome extension, and it updates to display the summary once it has been received from the backend



SpaceX's Starship achieves liftoff and separation, but 'mishap' results in loss of rocket

The rocket successfully lifted off and achieved separation, but it's not yet clear what may have

- onSubmit: This function sends the URL to the backend and retrieves the summary when the user clicks on the submit button



```
const onSubmit = async (e: React.FormEvent<HTMLFormElement>) => {
  e.preventDefault();

  const input = (document.getElementById("input") as HTMLInputElement)?.value;
  let res = await axios.post(
    "http://localhost:5001/get-remote-text",
    {
      url: input,
    },
    {
      headers: {
        "Content-Type": "application/json",
      },
    }
  );

  setSummary(res.data.text);
};
```

> /frontend/src/index.tsx

- Renders the App.tsx component

Backend

> /backend/config.toml

- When we were trying to use BM25 for summarization, this would be used for identifying our training data

> /backend/summary.py

- clean_text(html_text)
    - Params: html text is the entirety of the webpage’s text content, including all the html tags
    - What it Does: It cleans the text by removing all of the html and javascript tags that came with scraping the text data from the news article’s url
    - Output: The content from the news article that is useful
- summarize_text(params)
    - Params: article_text
    - What it Does: summarize_text takes in the cleaned text and summarizes it using gensim’s summarize tool, which is a TextRank summarizer.
    - Output: The output summary will consist of the most representative sentences and will be returned as a string, divided by newlines.

> /backend/server.js

- This file contains the API for the frontend to call to retrieve the summary
- This file only contains one route
- Routes:
    - “/get-remote-text”
        - Request Body:
            - The route request body must contain the URL that is being summarized
        - Calls summary.py to summarize the article, and it returns the summary

> backend/views/index.ejs

- This is the html file we used to test if the summarize function was working properly before connecting it to the front end