MINOR PROJECT REPORT

“TEXT SUMMARIZER USING AI”

University School of Information, Communication and Technology

Guru Gobind Singh Indraprastha University, Delhi

Submitted by: YASH ARYAN (06816403220)

Batch: B.Tech (CSE) 7th Semester

Mentor: Dr. Priyanka Bhutani

TABLE OF CONTENT

Candidate's Declaration

I, Yash Aryan (Enrollment No.06816403220), a student of B.Tech (CSE 7th Semester), USICT, Guru Gobind Singh Indraprastha University, hereby declare that the work which is presented in this Minor Project Report entitled "Text Summarizer using AI" is an original and authentic work of mine under the technical guidance of Dr. Priyanka Bhutani, Assistant Professor, USIC&T. I declare that the work in this project has not been submitted in full or in any part for any diploma or degree course of this or any other University to the best of my knowledge and belief. I will be solely responsible myself for any copyright infringement or plagiarism, if any, in the said work, and declare that all necessary due acknowledgement has been made in the content of said work. My supervisor/ guide shall not be held responsible for full or partial violation of copyright or intellectual property rights or any type of plagiarism involved above in the said work.

Name: Yash Aryan

Enrolment Number: 06816403220

Course: B. Tech CSE 7th Semester, University School of Information, Communication & Technology, USICT_GGSIPU, New Delhi-110078

Date: 20-11-2023

Acknowledgement

It gives me immense pleasure to take this opportunity to acknowledge my obligation to my mentor, Dr. Priyanka Bhutani, University School of Information and Communication Technology, GGSIPU, who has not only guided me throughout the project but also made a great effort in making the project a success. I am highly thankful to my guide for her keen interest, valuable guidance, technical acumen, round the clock encouragement, moral support & suggestions in the completion of the project.

Name: Yash Aryan

Enrolment Number: 06816403220

Course: B. Tech CSE 7th Semester, University School of Information, Communication & Technology, USICT_GGSIPU, New Delhi-110078

Date: 20-11-2023

Abstract

This summarizer uses article extractor and summarizer API for extracting the article from the link and summarizing it. This project was made with React, Vite.js and Rapid API. When a user pastes a link to an article in the search bar, it extracts the article from that link, summarizes the article and presents the summary to the user within five seconds. The search history of user is also stored, which can be further used by copying the link.

A link to the source code is also available in the upper right corner of the web page. The user can clone the repository and run the project on his local machine by using “npm run dev” command in the terminal. To build the project locally, user must use “npm run build” command in the terminal.

This project is deployed on netlify and can be accessed by following the link: https://frabjous-tarsier-14495c.netlify.app/

Problem Statement

In the ever-evolving digital landscape, where information is constantly being generated and disseminated at an unprecedented rate, the need for effective tools to manage and process this vast data has become increasingly crucial. Among the many technological advancements that address this challenge, text summarizers stand out as indispensable tools that empower users to navigate the information overload and extract the essence of lengthy texts.

The need for text summarizers stems from the sheer volume of information that we encounter daily. From news articles and research papers to emails and social media feeds, we are constantly bombarded with text-based content. Devoting time to read and comprehend each piece of information in its entirety is often impractical, if not impossible. Text summarizers alleviate this burden by providing a quick overview of the key points and main ideas, allowing us to make informed decisions about whether to delve deeper into the original text.

Moreover, text summarizers play a vital role in enhancing comprehension and retention of information. By condensing complex concepts into a more succinct format, these tools help us grasp the core message more effectively and retain the information for longer periods. This is particularly beneficial for students, professionals, or anyone who needs to learn and retain information quickly.

As we move deeper into the digital age, the need for text summarizers will continue to grow. These tools have the potential to revolutionize the way we interact with information, enabling us to navigate the information overload more effectively, enhance our understanding of complex topics, and make informed decisions based on a comprehensive grasp of the available data.

Functional Requirements

1. URL Input:

- The application must allow users to input URL for summarization.

2. User Interaction:

- Users must be able to interact with the web application.

3. Summary Display:

- The application must display the generated summary to the user.

4. Integration with Rapid API:

- The application must effectively integrate with the Rapid API for utilizing external services or data sources related to text summarization.
- API calls should be handled securely and efficiently.

5. Responsive Design:

- The user interface must be responsive and accessible on various devices, including desktops, tablets, and mobile phones.
- It must provide an optimal user experience across different screen sizes.

Non-functional Requirements

1. Performance:

- The summarization process must be fast and responsive, providing quick results to users.
- The application must handle simultaneous requests efficiently, ensuring good performance under varying loads.

2. Security:

- User input and data must be handled securely to prevent vulnerabilities such as cross-site scripting or injection attacks.
- API calls to Rapid API must be secured using appropriate authentication mechanisms.

3. Reliability:

- The application must be reliable and available for users at all times.
- It must include error handling and recovery mechanisms to ensure a smooth user experience.

4. User Experience (UX):

- The user interface must be intuitive and user-friendly.
- It must provide clear instructions, feedback, and visual cues to guide users through the summarization process.

5. Compatibility:

- The application must be compatible with modern web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.
- It must adhere to web standards and best practices for cross-browser compatibility.

Design and Implementation

- **Tech Stack Used**
  - **Front-end:** ReactJs, Vite
  - **API:** Article Extractor and Summarizer API
  - **Deployment Platform:** Netlify
  - **Additionals**
    - Rapid API
    - VS Code Extension: ES7+ React/Redux/React-Native snippets

- **Hardware and Software Interfaces**
  - **Hardware**
    - Fast internet enabled mobile or computer device
  - **Software**

Software Used	**Description**
ReactJs	ReactJs is a free and open-source front-end JavaScript library for building user interfaces based on components.
Vite	Vite is a frontend build tool that significantly improves the frontend development experience. It consists of two major parts: • A dev server that serves your source files over native ES modules, with rich built-in features and astonishingly fast Hot Module Replacement (HMR). • A build command that bundles your code with Rollup, pre-configured to output highly optimized static assets for production.

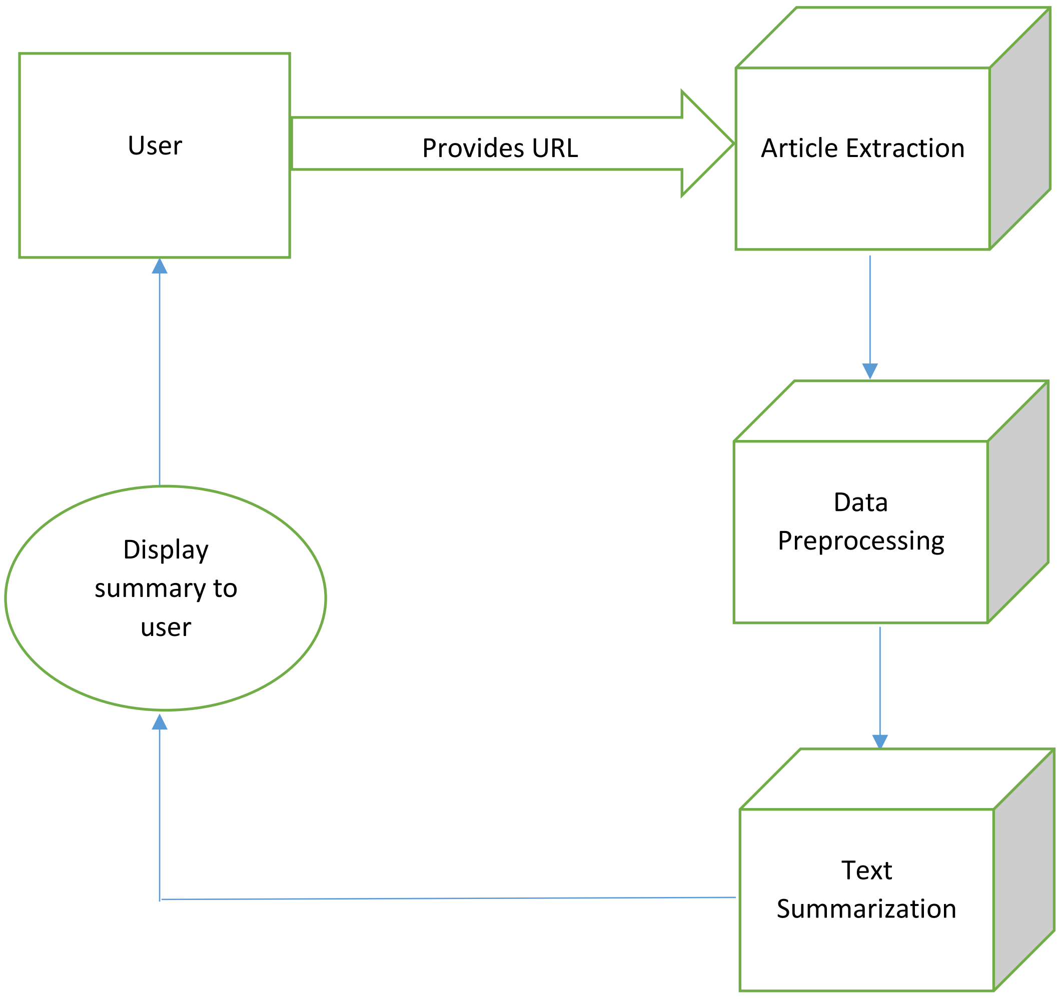
Components

- **Architecture for Front-End:** Achieved using ReactJs and Vite. ReactJs is a free and open-source front-end JavaScript library for building user interfaces based on components. Vite is a frontend build tool that significantly improves the frontend development experience. It consists of two major parts:
    - A dev server that serves your source files over native ES modules, with rich built-in features and astonishingly fast Hot Module Replacement (HMR).
    - A build command that bundles your code with Rollup, pre-configured to output highly optimized static assets for production.

- **Article extracting and summarizing Services:** Achieved by Article Extractor and Summarizer API. It extracts news/article body from a URL and uses GPT to summarize (and optionally translate) the article content. Useful for text mining purposes. Leverages powerful and flexible web scraping engine (ScrapeNinja.net) with high-quality rotating proxies under the hood.

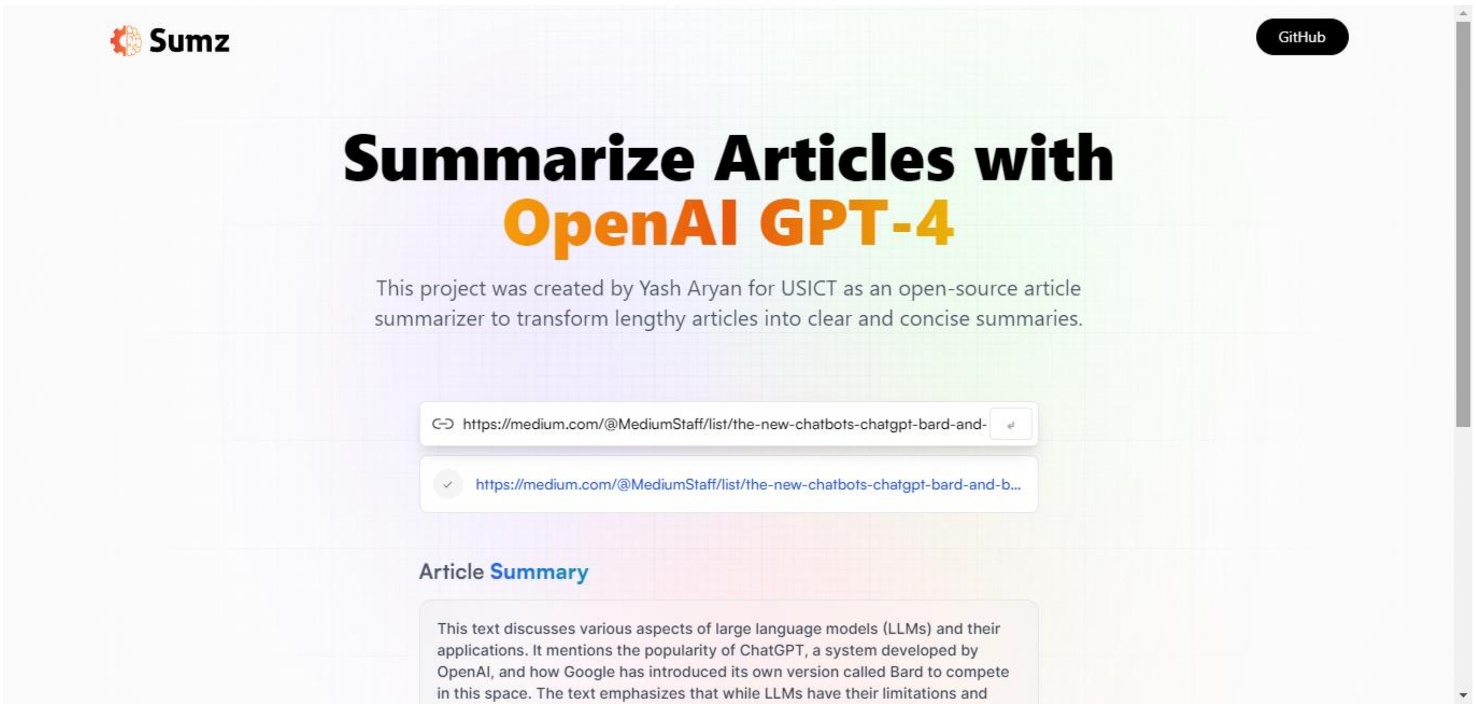
Workflow

Web Application Workflow Overview



- User provides URL of an article to be summarized
- Extracts the article to be summarized
- Uses GPT to summarize the article
- Displays summary of article to the user

UI Snapshots



Summarize Articles with OpenAI GPT-4

This project was created by Yash Aryan for USICT as an open-source article summarizer to transform lengthy articles into clear and concise summaries.

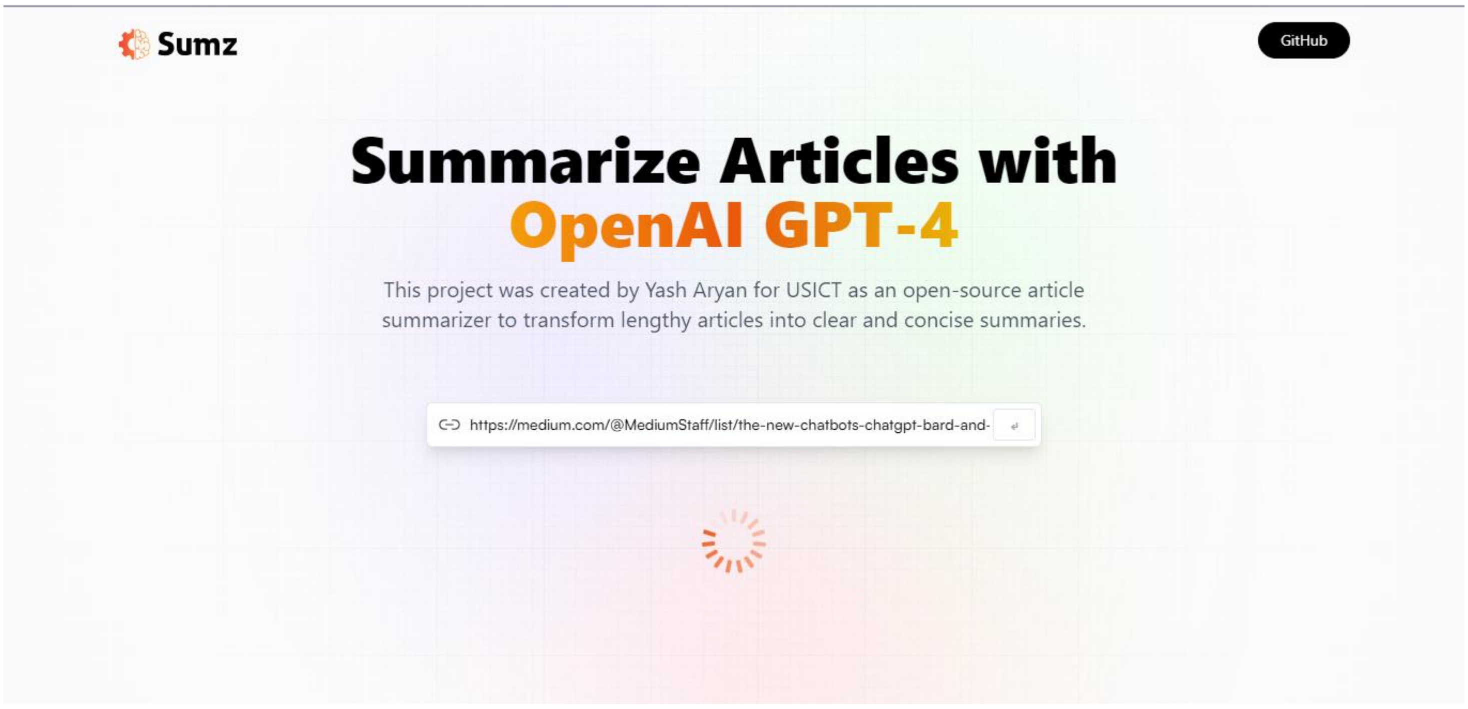
https://medium.com/@MediumStaff/list/the-new-chatbots-chatgpt-bard-and-

https://medium.com/@MediumStaff/list/the-new-chatbots-chatgpt-bard-and-b...

Article Summary

This text discusses various aspects of large language models (LLMs) and their applications. It mentions the popularity of ChatGPT, a system developed by OpenAI, and how Google has introduced its own version called Bard to compete in this space. The text emphasizes that while LLMs have their limitations and

Snapshot 1

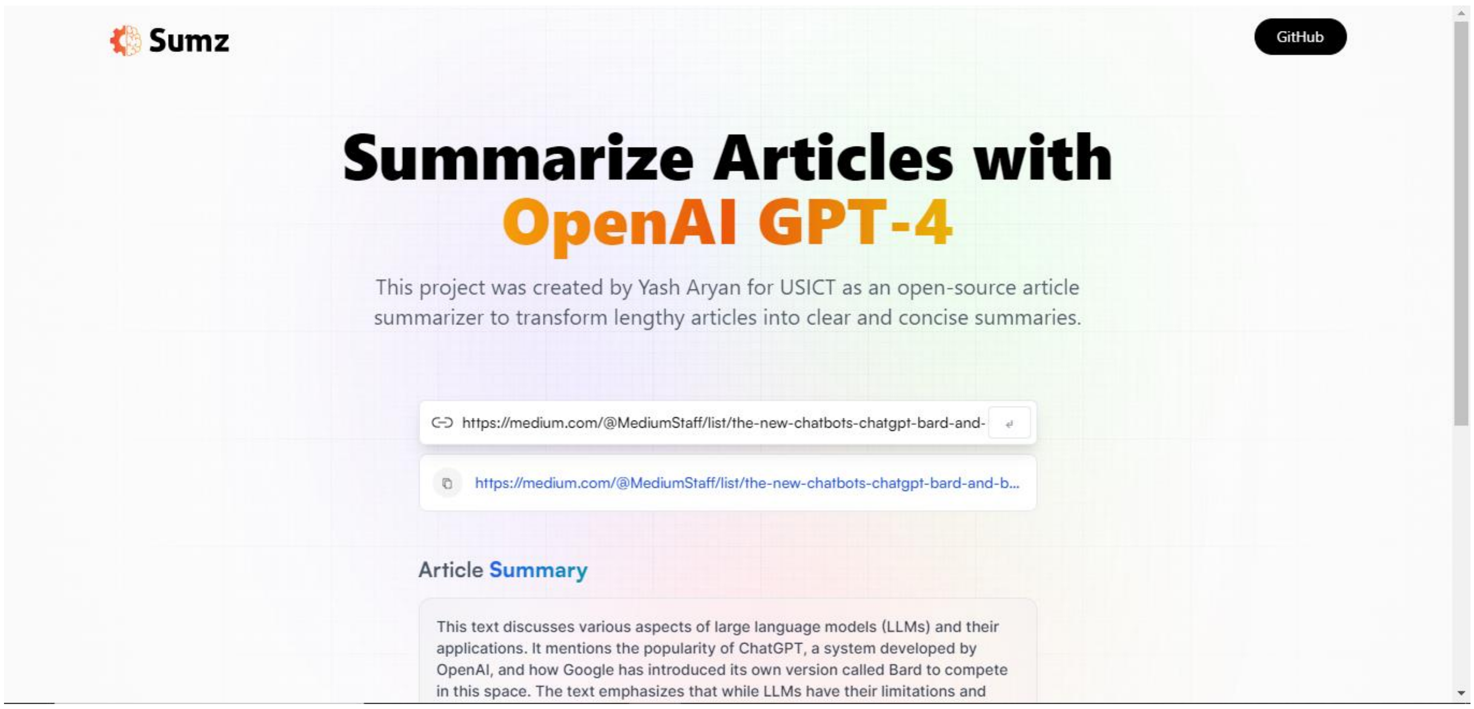


Summarize Articles with OpenAI GPT-4

This project was created by Yash Aryan for USICT as an open-source article summarizer to transform lengthy articles into clear and concise summaries.

https://medium.com/@MediumStaff/list/the-new-chatbots-chatgpt-bard-and-

Snapshot 2



Summarize Articles with OpenAI GPT-4

This project was created by Yash Aryan for USICT as an open-source article summarizer to transform lengthy articles into clear and concise summaries.

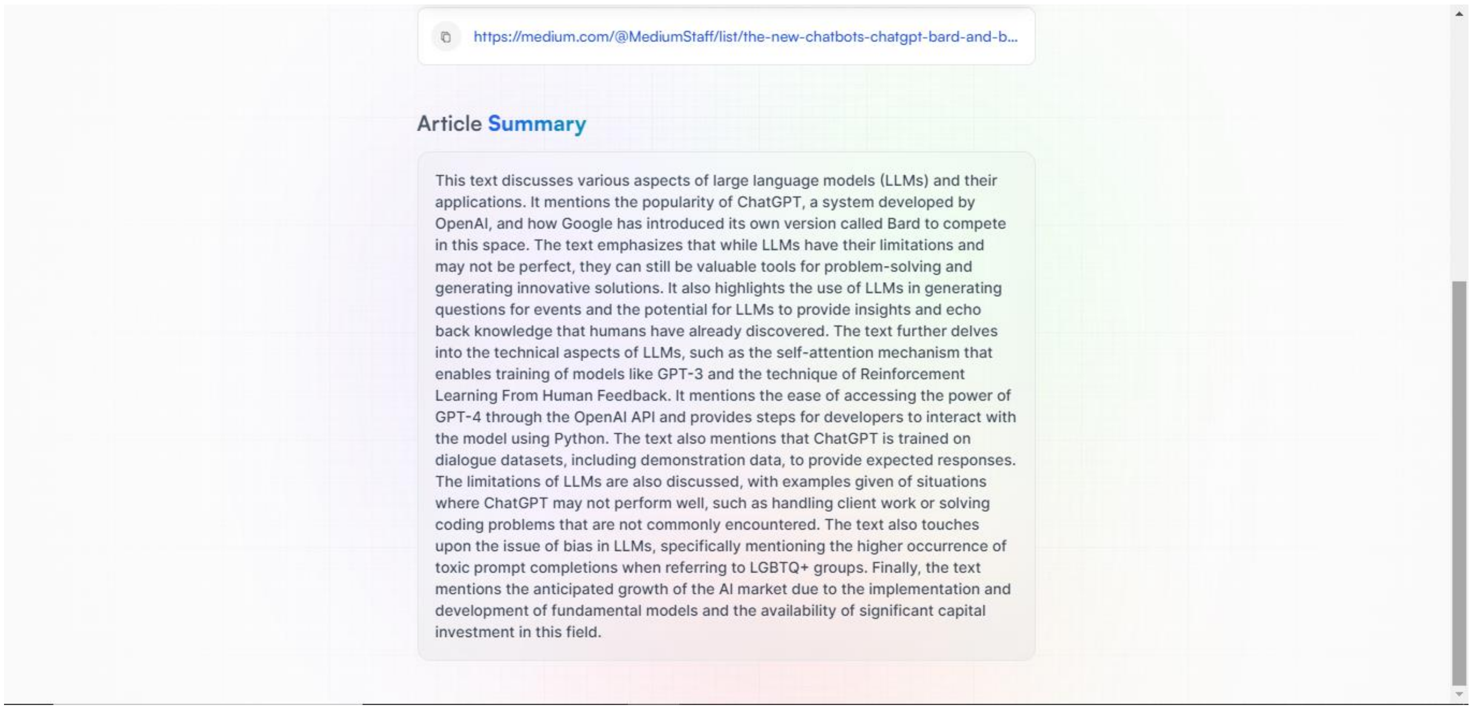
https://medium.com/@MediumStaff/list/the-new-chatbots-chatgpt-bard-and-

https://medium.com/@MediumStaff/list/the-new-chatbots-chatgpt-bard-and-b...

Article Summary

This text discusses various aspects of large language models (LLMs) and their applications. It mentions the popularity of ChatGPT, a system developed by OpenAI, and how Google has introduced its own version called Bard to compete in this space. The text emphasizes that while LLMs have their limitations and

Snapshot 3



https://medium.com/@MediumStaff/list/the-new-chatbots-chatgpt-bard-and-b...

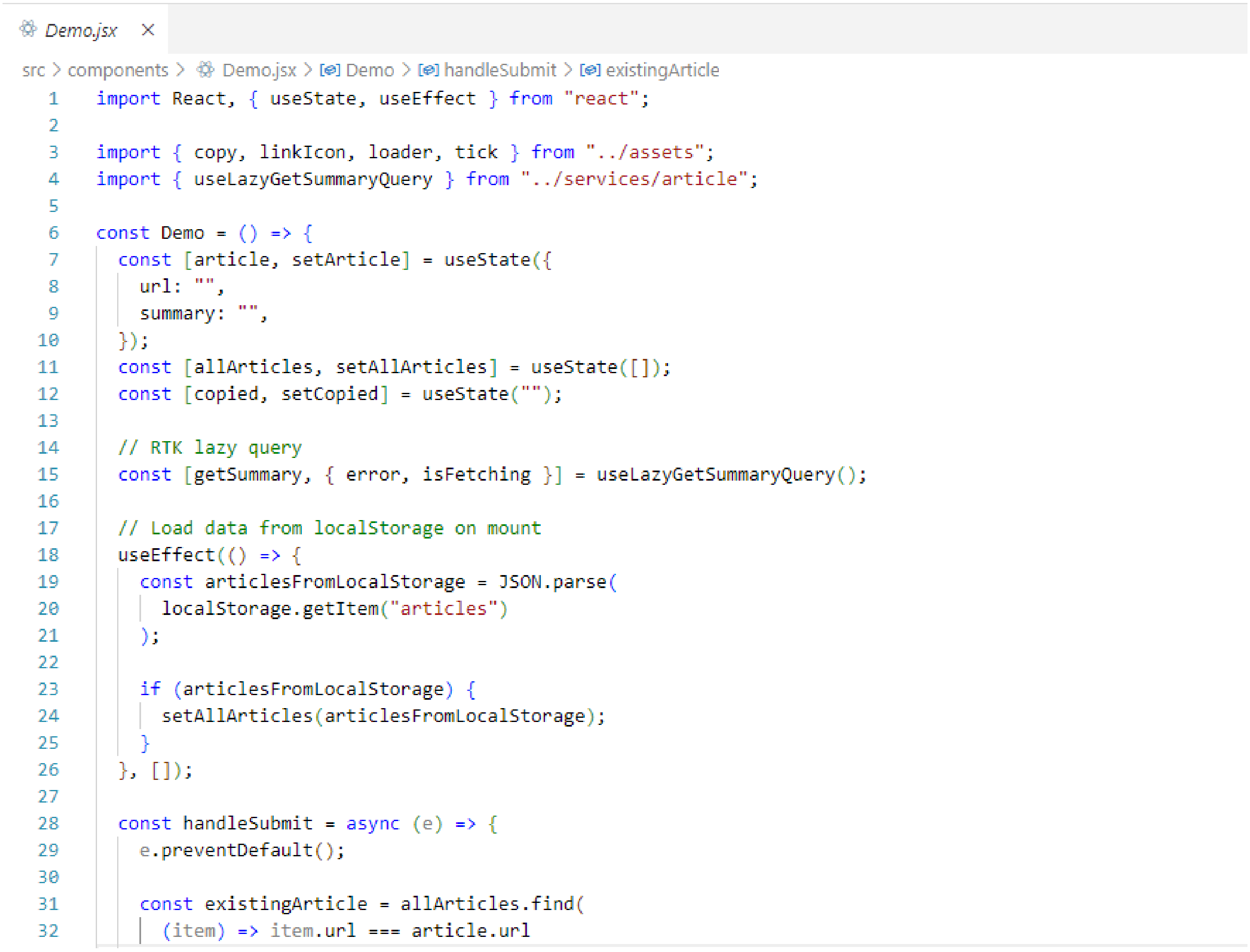
Article Summary

This text discusses various aspects of large language models (LLMs) and their applications. It mentions the popularity of ChatGPT, a system developed by OpenAI, and how Google has introduced its own version called Bard to compete in this space. The text emphasizes that while LLMs have their limitations and may not be perfect, they can still be valuable tools for problem-solving and generating innovative solutions. It also highlights the use of LLMs in generating questions for events and the potential for LLMs to provide insights and echo back knowledge that humans have already discovered. The text further delves into the technical aspects of LLMs, such as the self-attention mechanism that enables training of models like GPT-3 and the technique of Reinforcement Learning From Human Feedback. It mentions the ease of accessing the power of GPT-4 through the OpenAI API and provides steps for developers to interact with the model using Python. The text also mentions that ChatGPT is trained on dialogue datasets, including demonstration data, to provide expected responses. The limitations of LLMs are also discussed, with examples given of situations where ChatGPT may not perform well, such as handling client work or solving coding problems that are not commonly encountered. The text also touches upon the issue of bias in LLMs, specifically mentioning the higher occurrence of toxic prompt completions when referring to LGBTQ+ groups. Finally, the text mentions the anticipated growth of the AI market due to the implementation and development of fundamental models and the availability of significant capital investment in this field.

Snapshot 4

Code Snippets

Demo.jsx

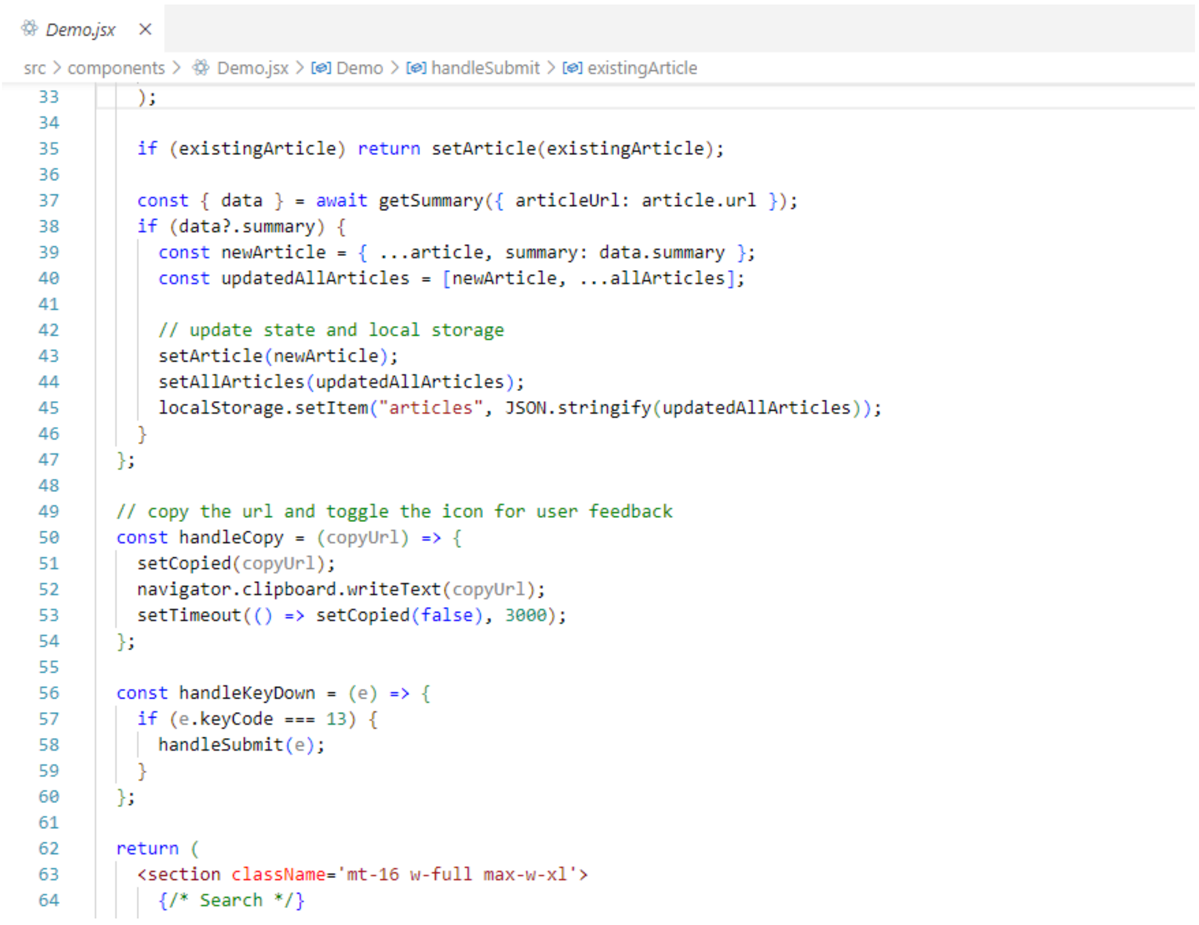


Demo.jsx ×

src > components > Demo.jsx > Demo > handleSubmit > existingArticle

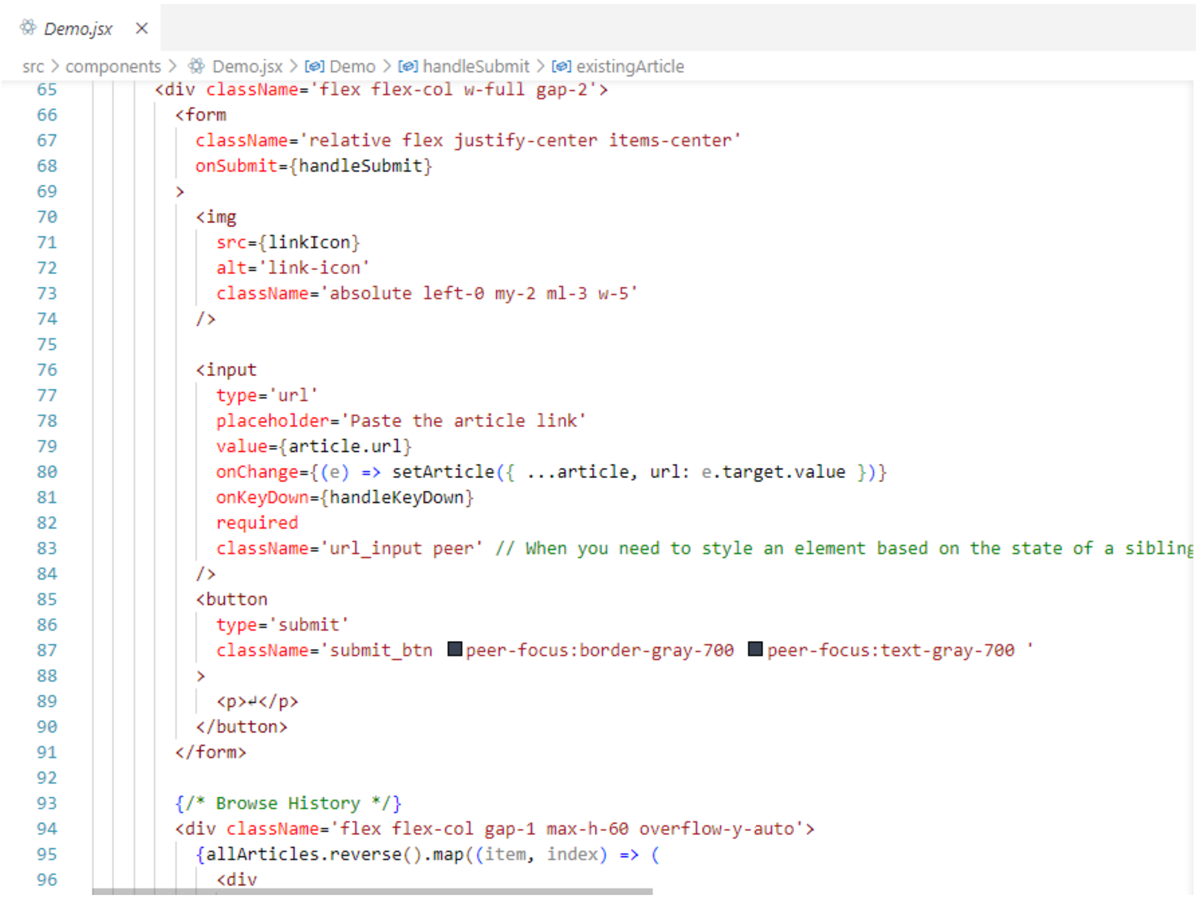
```jsx
import React, { useState, useEffect } from "react";

import { copy, linkIcon, loader, tick } from "../assets";
import { useLazyGetSummaryQuery } from "../services/article";

const Demo = () => {
  const [article, setArticle] = useState({
    url: "",
    summary: "",
  });
  const [allArticles, setAllArticles] = useState([]);
  const [copied, setCopied] = useState("");

  // RTK lazy query
  const [getSummary, { error, isFetching }] = useLazyGetSummaryQuery();

  // Load data from localStorage on mount
  useEffect(() => {
    const articlesFromLocalStorage = JSON.parse(
      localStorage.getItem("articles")
    );

    if (articlesFromLocalStorage) {
      setAllArticles(articlesFromLocalStorage);
    }
  }, []);

  const handleSubmit = async (e) => {
    e.preventDefault();

    const existingArticle = allArticles.find(
      (item) => item.url === article.url
```

Snippet 1


Demo.jsx
src > components > Demo.jsx > Demo > handleSubmit > existingArticle

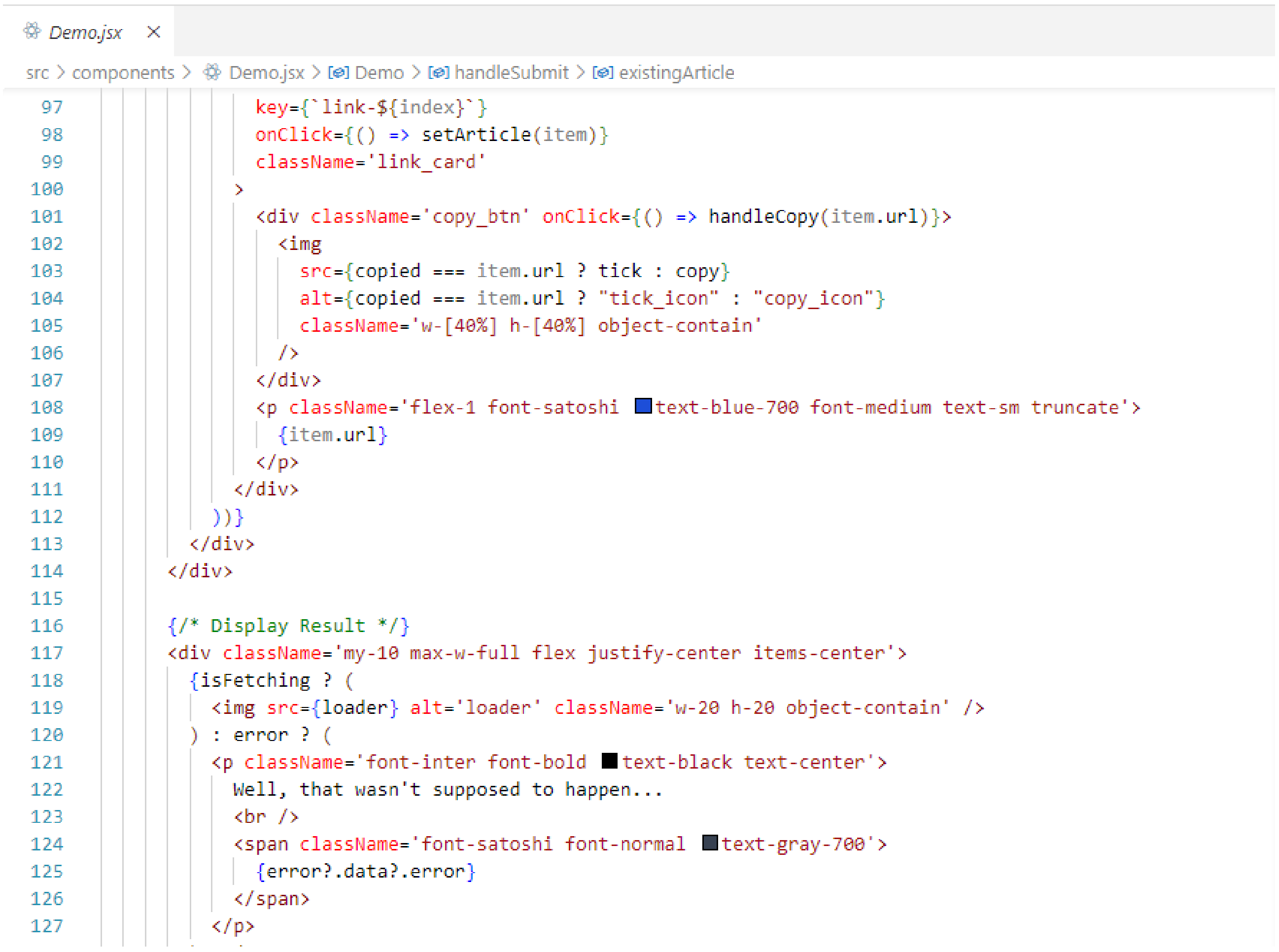
```jsx
    );

    if (existingArticle) return setArticle(existingArticle);

    const { data } = await getSummary({ articleUrl: article.url });
    if (data?.summary) {
      const newArticle = { ...article, summary: data.summary };
      const updatedAllArticles = [newArticle, ...allArticles];

      // update state and local storage
      setArticle(newArticle);
      setAllArticles(updatedAllArticles);
      localStorage.setItem("articles", JSON.stringify(updatedAllArticles));
    }
  };

  // copy the url and toggle the icon for user feedback
  const handleCopy = (copyUrl) => {
    setCopied(copyUrl);
    navigator.clipboard.writeText(copyUrl);
    setTimeout(() => setCopied(false), 3000);
  };

  const handleKeyDown = (e) => {
    if (e.keyCode === 13) {
      handleSubmit(e);
    }
  };

  return (
    <section className='mt-16 w-full max-w-xl'>
      {/* Search */}
```

Snippet 2


Demo.jsx
src > components > Demo.jsx > Demo > handleSubmit > existingArticle

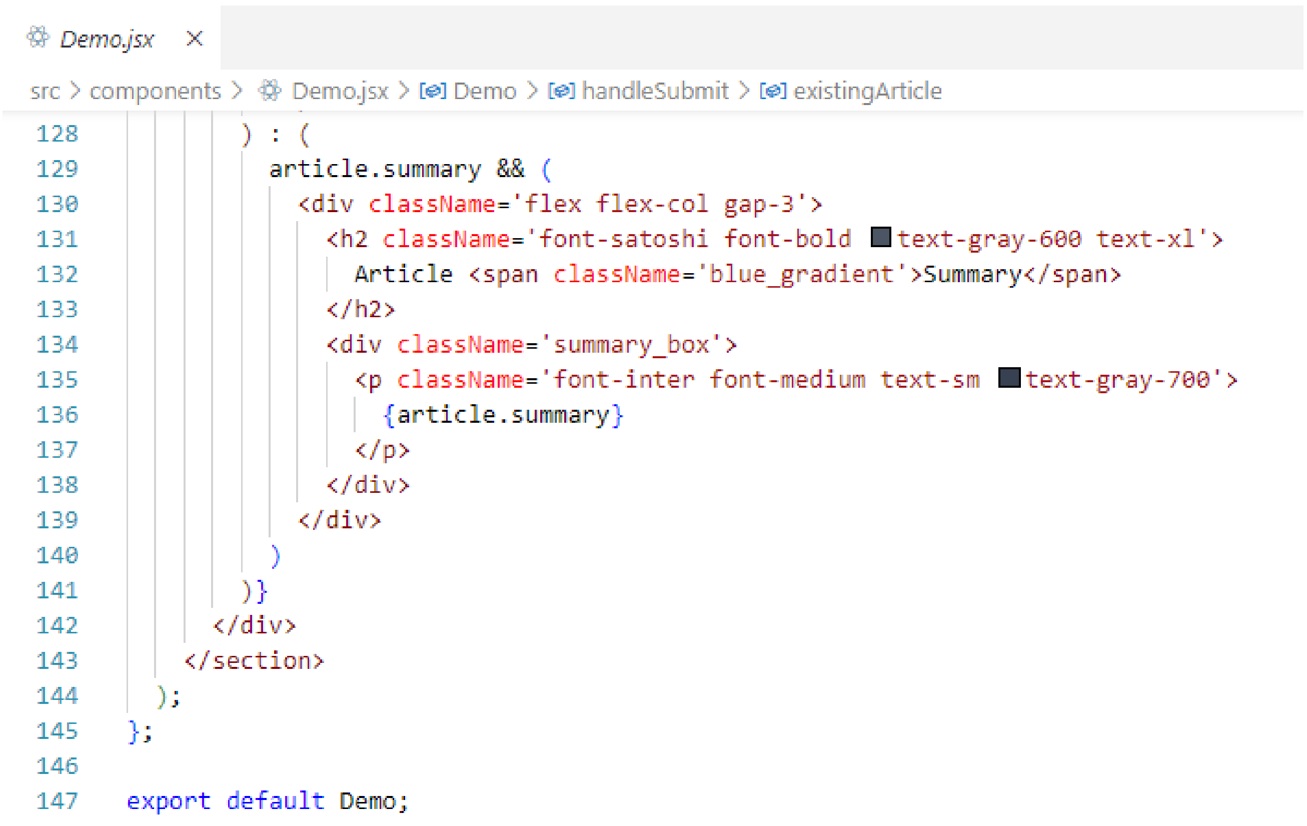
```jsx
      <div className='flex flex-col w-full gap-2'>
        <form
          className='relative flex justify-center items-center'
          onSubmit={handleSubmit}
        >
          <img
            src={linkIcon}
            alt='link-icon'
            className='absolute left-0 my-2 ml-3 w-5'
          />

          <input
            type='url'
            placeholder='Paste the article link'
            value={article.url}
            onChange={(e) => setArticle({ ...article, url: e.target.value })}
            onKeyDown={handleKeyDown}
            required
            className='url_input peer' // When you need to style an element based on the state of a sibling
          />
          <button
            type='submit'
            className='submit_btn peer-focus:border-gray-700 peer-focus:text-gray-700 '
          >
            <p>↵</p>
          </button>
        </form>

        {/* Browse History */}
        <div className='flex flex-col gap-1 max-h-60 overflow-y-auto'>
          {allArticles.reverse().map((item, index) => (
            <div
```

Snippet 3


Demo.jsx ×
src > components > Demo.jsx > Demo > handleSubmit > existingArticle

```jsx
              key={`link-${index}`}
              onClick={() => setArticle(item)}
              className='link_card'
            >
              <div className='copy_btn' onClick={() => handleCopy(item.url)}>
                <img
                  src={copied === item.url ? tick : copy}
                  alt={copied === item.url ? "tick_icon" : "copy_icon"}
                  className='w-[40%] h-[40%] object-contain'
                />
              </div>
              <p className='flex-1 font-satoshi text-blue-700 font-medium text-sm truncate'>
                {item.url}
              </p>
            </div>
          ))}
        </div>
      </div>

      {/* Display Result */}
      <div className='my-10 max-w-full flex justify-center items-center'>
        {isFetching ? (
          <img src={loader} alt='loader' className='w-20 h-20 object-contain' />
        ) : error ? (
          <p className='font-inter font-bold text-black text-center'>
            Well, that wasn't supposed to happen...
            <br />
            <span className='font-satoshi font-normal text-gray-700'>
              {error?.data?.error}
            </span>
          </p>
```

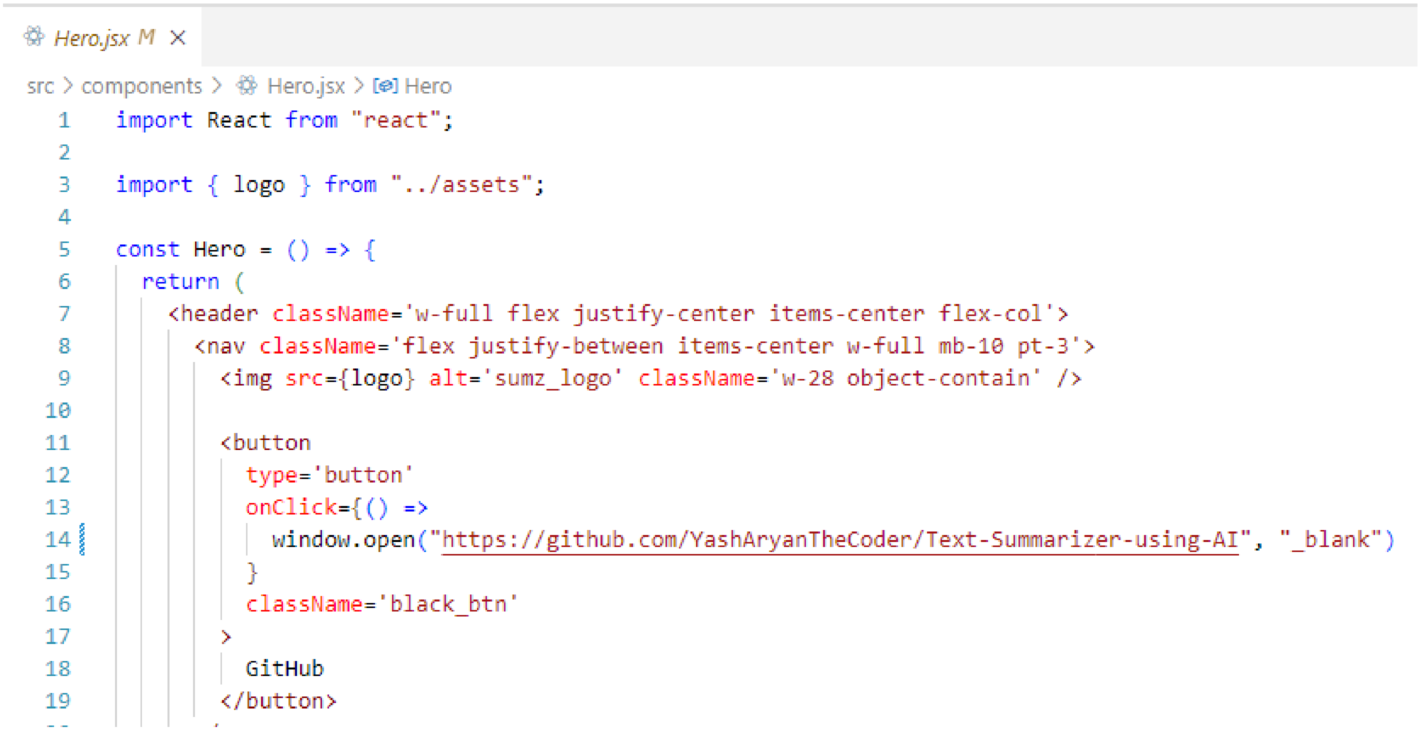
Snippet 4


Demo.jsx ×
src > components > Demo.jsx > Demo > handleSubmit > existingArticle

```jsx
        ) : (
          article.summary && (
            <div className='flex flex-col gap-3'>
              <h2 className='font-satoshi font-bold text-gray-600 text-xl'>
                Article <span className='blue_gradient'>Summary</span>
              </h2>
              <div className='summary_box'>
                <p className='font-inter font-medium text-sm text-gray-700'>
                  {article.summary}
                </p>
              </div>
            </div>
          )
        )}
      </div>
    </section>
  );
};

export default Demo;
```

Snippet 5

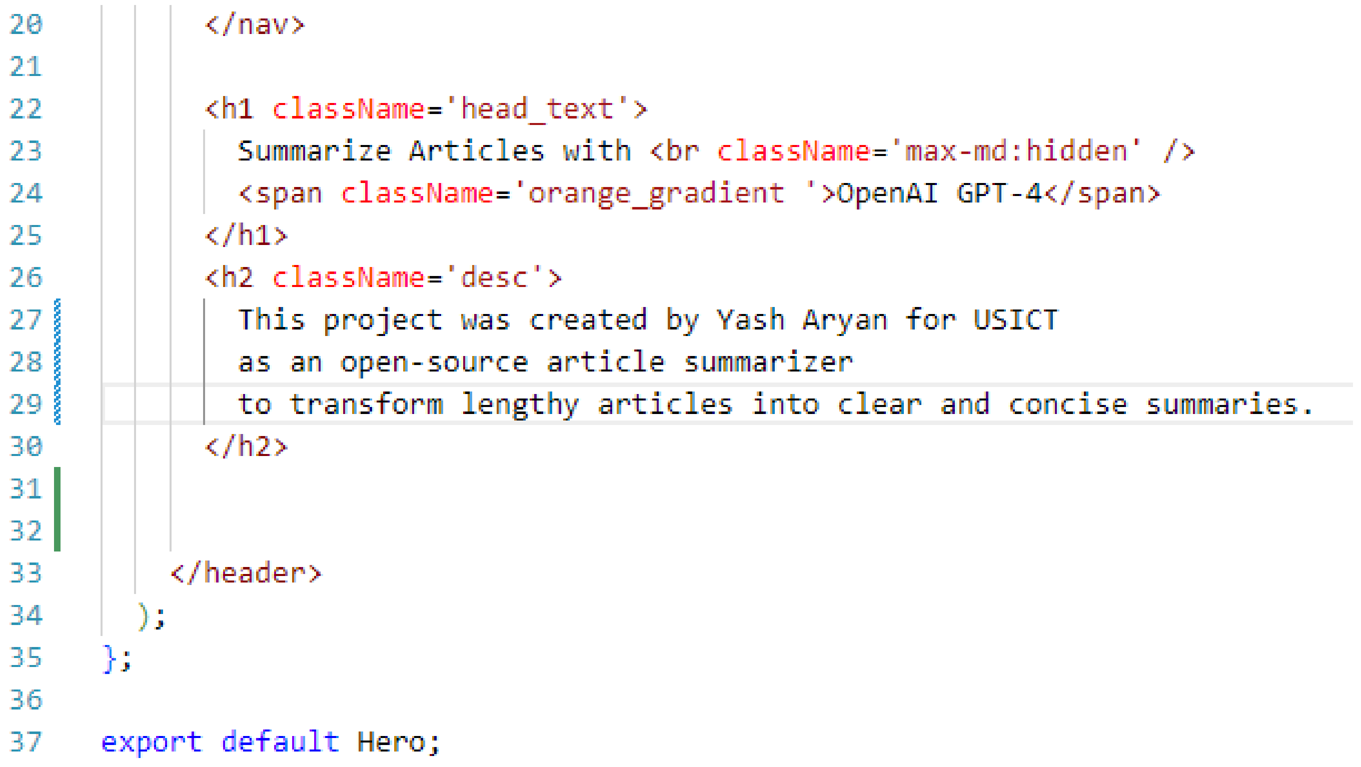
Hero.jsx


Hero.jsx M ×

src > components > Hero.jsx > Hero

```
import React from "react";

import { logo } from "../assets";

const Hero = () => {
  return (
    <header className='w-full flex justify-center items-center flex-col'>
      <nav className='flex justify-between items-center w-full mb-10 pt-3'>
        <img src={logo} alt='sumz_logo' className='w-28 object-contain' />

        <button
          type='button'
          onClick={() =>
            window.open("https://github.com/YashAryanTheCoder/Text-Summarizer-using-AI", "_blank")
          }
          className='black_btn'
        >
          GitHub
        </button>
```

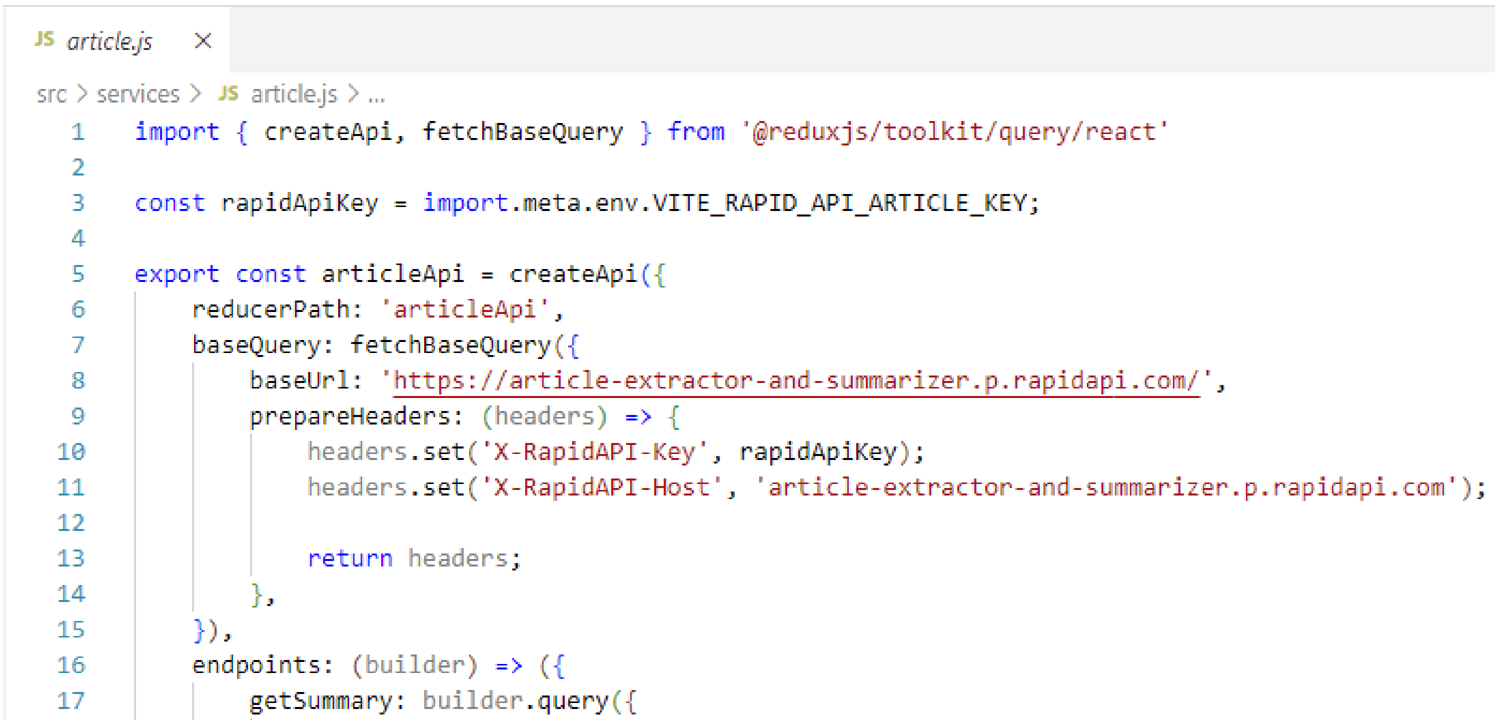
Snippet 6



```
      </nav>

      <h1 className='head_text'>
        Summarize Articles with <br className='max-md:hidden' />
        <span className='orange_gradient '>OpenAI GPT-4</span>
      </h1>
      <h2 className='desc'>
        This project was created by Yash Aryan for USICT
        as an open-source article summarizer
        to transform lengthy articles into clear and concise summaries.
      </h2>


    </header>
  );
};

export default Hero;
```

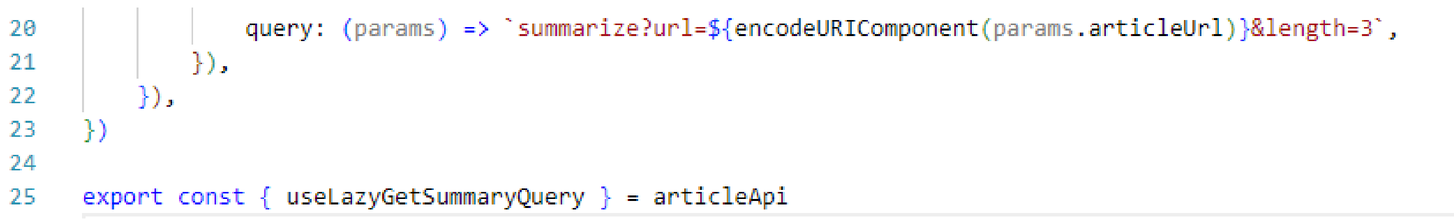
Snippet 7

article.js



```
import { createApi, fetchBaseQuery } from '@reduxjs/toolkit/query/react'

const rapidApiKey = import.meta.env.VITE_RAPID_API_ARTICLE_KEY;

export const articleApi = createApi({
    reducerPath: 'articleApi',
    baseQuery: fetchBaseQuery({
        baseUrl: 'https://article-extractor-and-summarizer.p.rapidapi.com/',
        prepareHeaders: (headers) => {
            headers.set('X-RapidAPI-Key', rapidApiKey);
            headers.set('X-RapidAPI-Host', 'article-extractor-and-summarizer.p.rapidapi.com');

            return headers;
        },
    }),
    endpoints: (builder) => ({
        getSummary: builder.query({
```

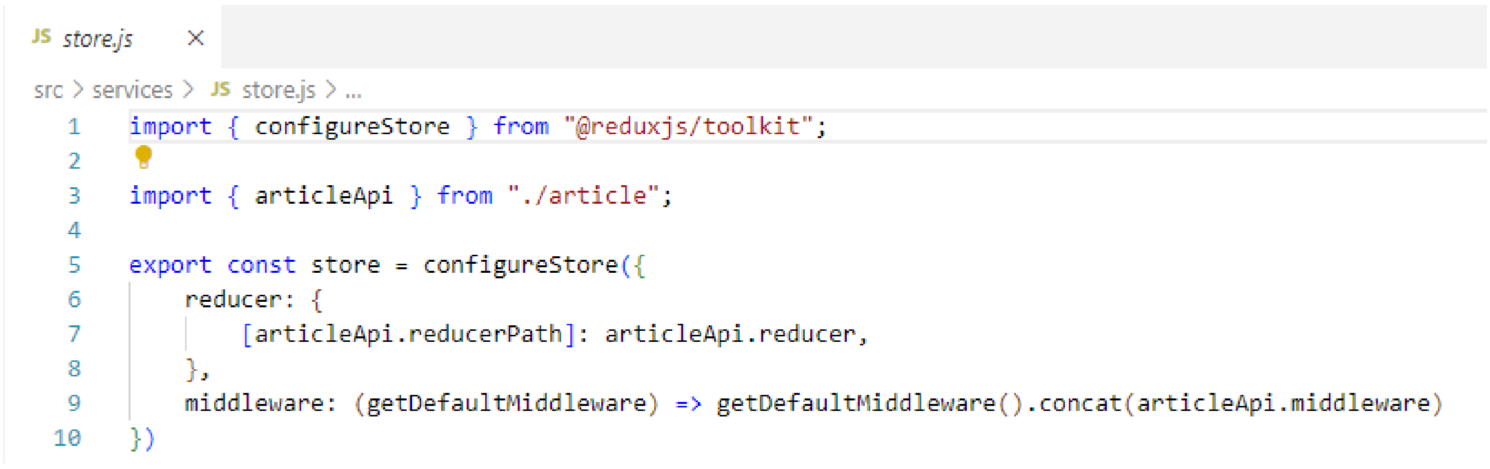
Figure 5



```
            query: (params) => `summarize?url=${encodeURIComponent(params.articleUrl)}&length=3`,
        }),
    }),
})

export const { useLazyGetSummaryQuery } = articleApi
```

Snippet 8

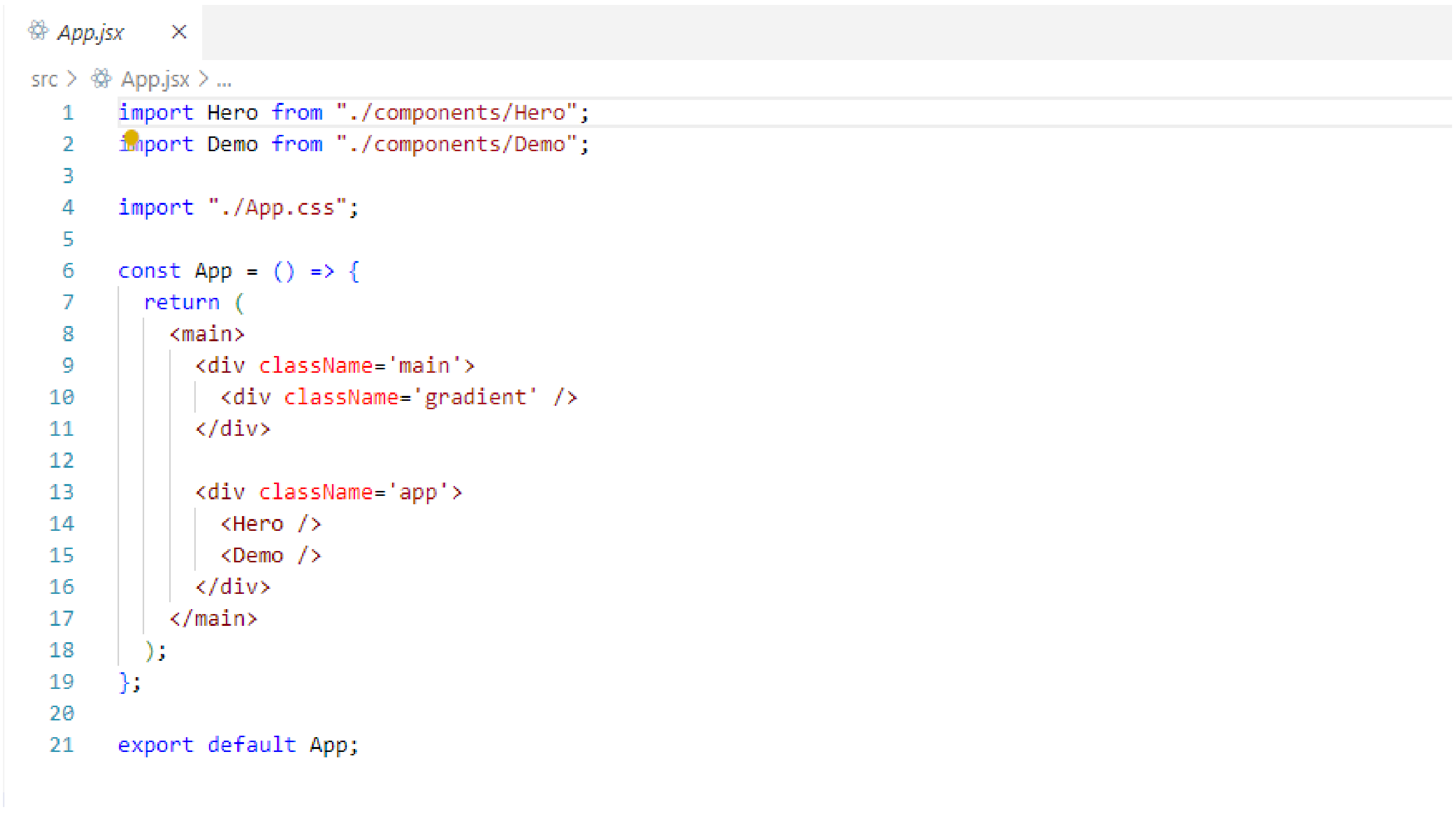
store.js



```
import { configureStore } from "@reduxjs/toolkit";

import { articleApi } from "./article";

export const store = configureStore({
    reducer: {
        [articleApi.reducerPath]: articleApi.reducer,
    },
    middleware: (getDefaultMiddleware) => getDefaultMiddleware().concat(articleApi.middleware)
})
```

Snippet 9

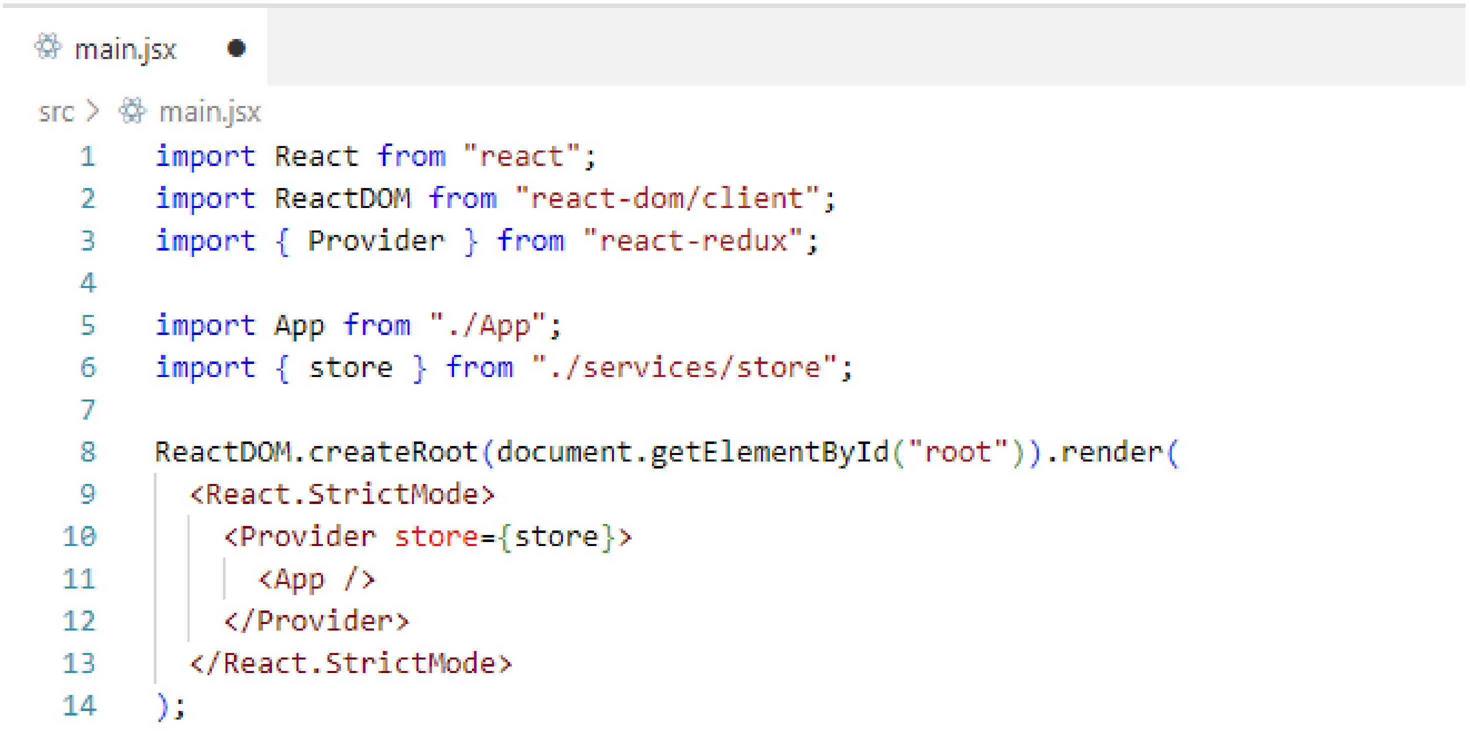
App.jsx



```jsx
import Hero from "./components/Hero";
import Demo from "./components/Demo";

import "./App.css";

const App = () => {
  return (
    <main>
      <div className='main'>
        <div className='gradient' />
      </div>

      <div className='app'>
        <Hero />
        <Demo />
      </div>
    </main>
  );
};

export default App;
```

Snippet 10

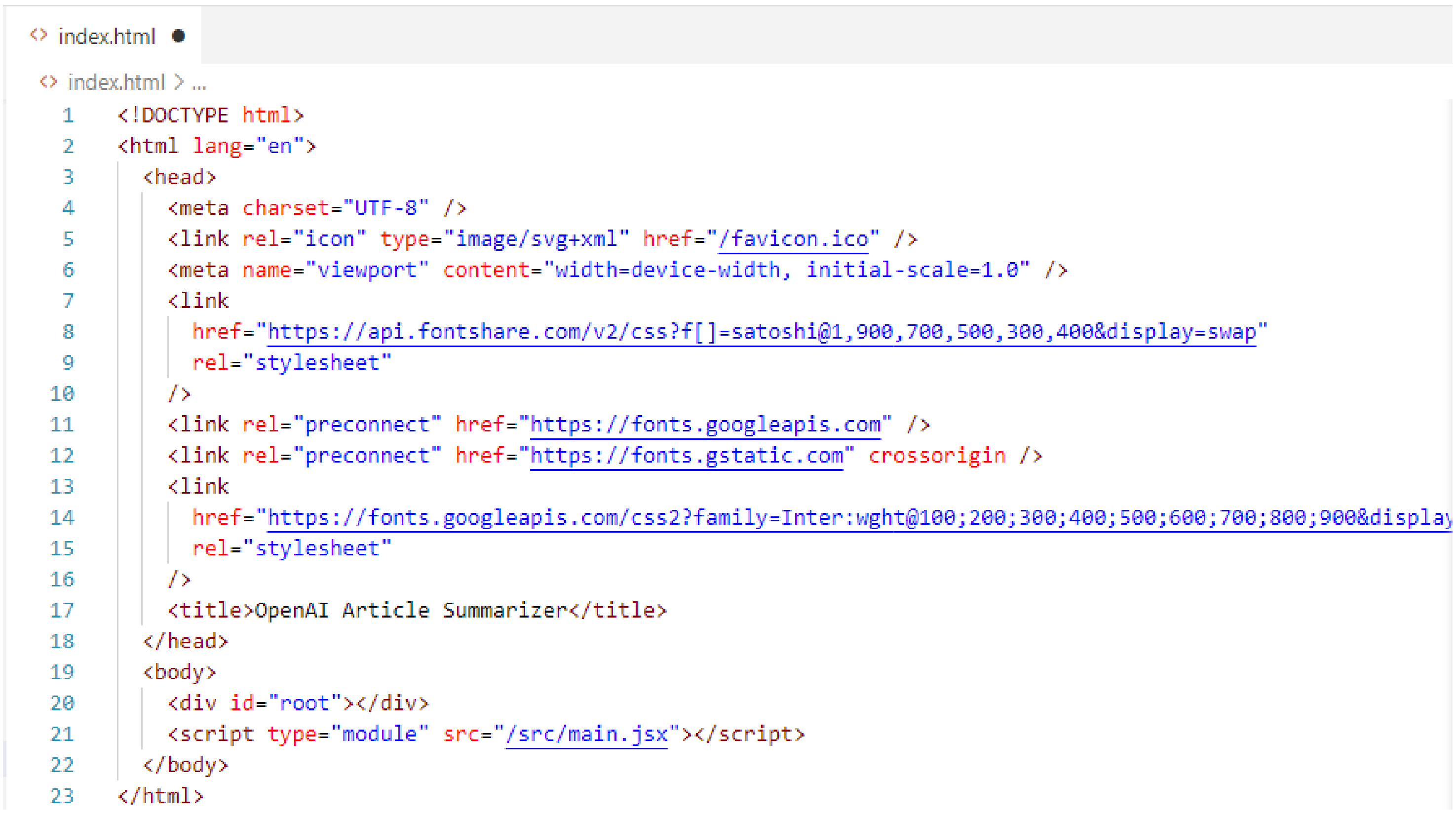
main.jsx



```jsx
import React from "react";
import ReactDOM from "react-dom/client";
import { Provider } from "react-redux";

import App from "./App";
import { store } from "./services/store";

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
);
```

Snippet 11

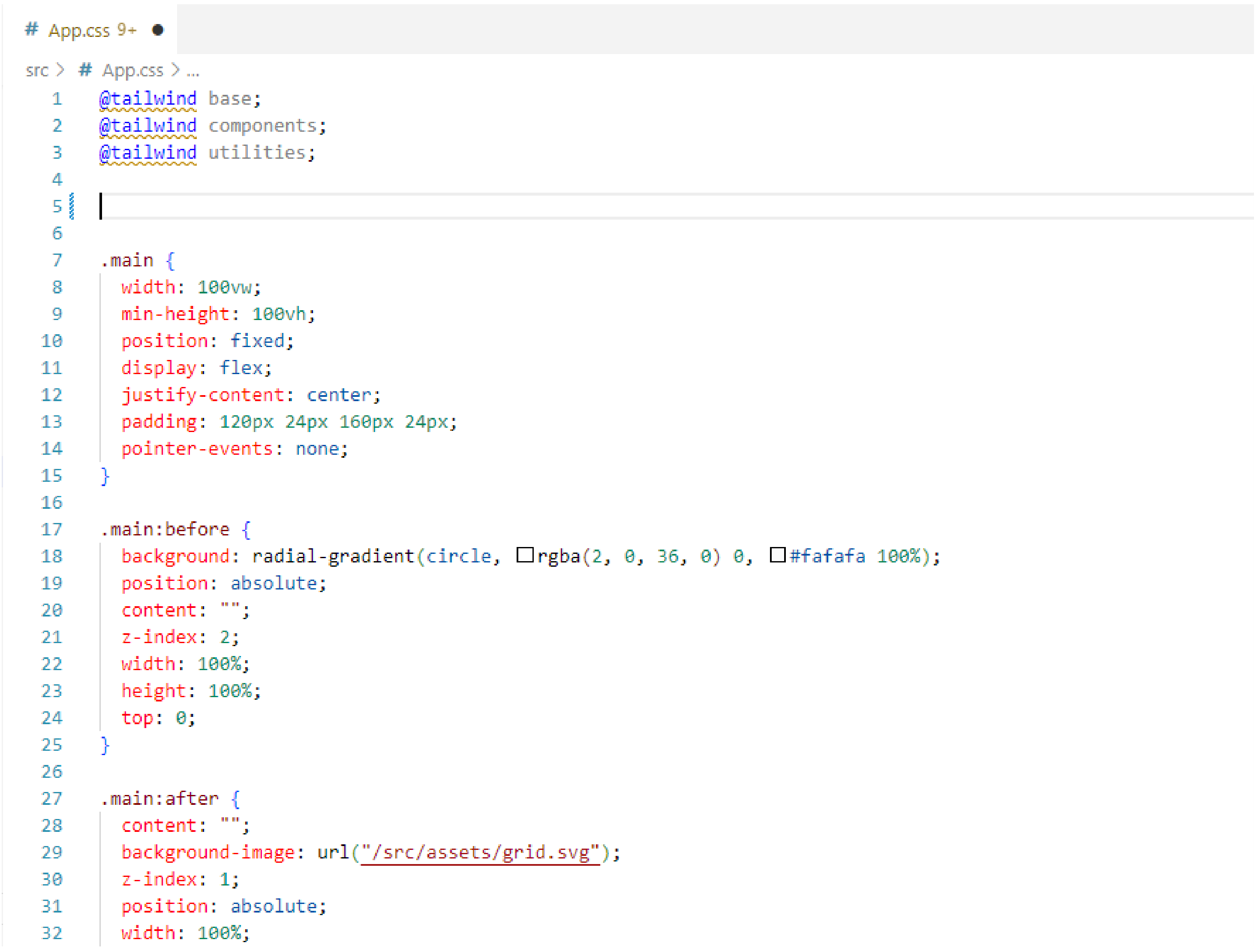
index.html



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://api.fontshare.com/v2/css?f[]=satoshi@1,900,700,500,300,400&display=swap"
      rel="stylesheet"
    />
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link
      href="https://fonts.googleapis.com/css2?family=Inter:wght@100;200;300;400;500;600;700;800;900&display
      rel="stylesheet"
    />
    <title>OpenAI Article Summarizer</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

Snippet 12

App.css

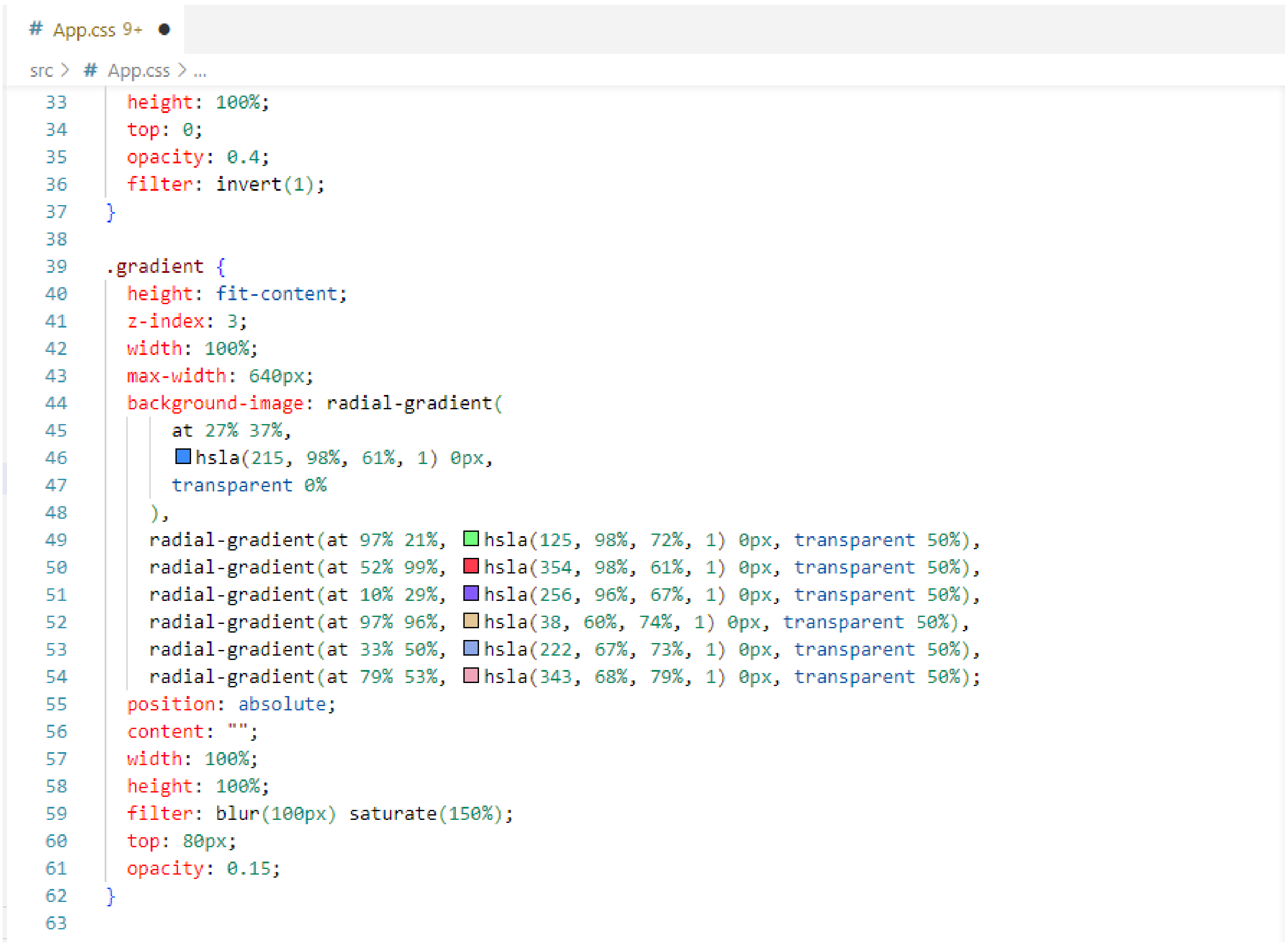


App.css 9+ ●

src > # App.css > ...

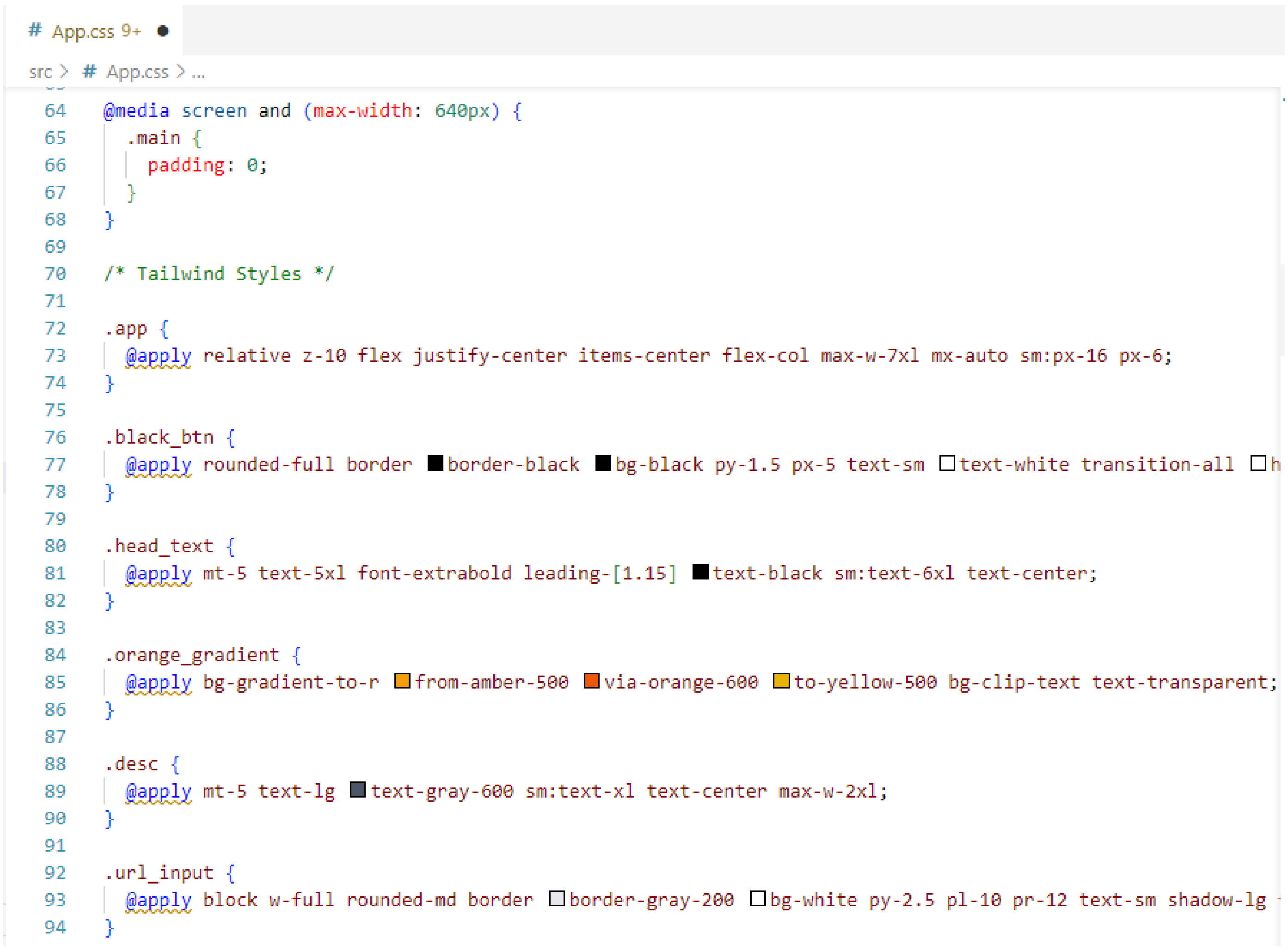
```css
@tailwind base;
@tailwind components;
@tailwind utilities;



.main {
  width: 100vw;
  min-height: 100vh;
  position: fixed;
  display: flex;
  justify-content: center;
  padding: 120px 24px 160px 24px;
  pointer-events: none;
}

.main:before {
  background: radial-gradient(circle, □rgba(2, 0, 36, 0) 0, □#fafafa 100%);
  position: absolute;
  content: "";
  z-index: 2;
  width: 100%;
  height: 100%;
  top: 0;
}

.main:after {
  content: "";
  background-image: url("/src/assets/grid.svg");
  z-index: 1;
  position: absolute;
  width: 100%;
```

Snippet 13



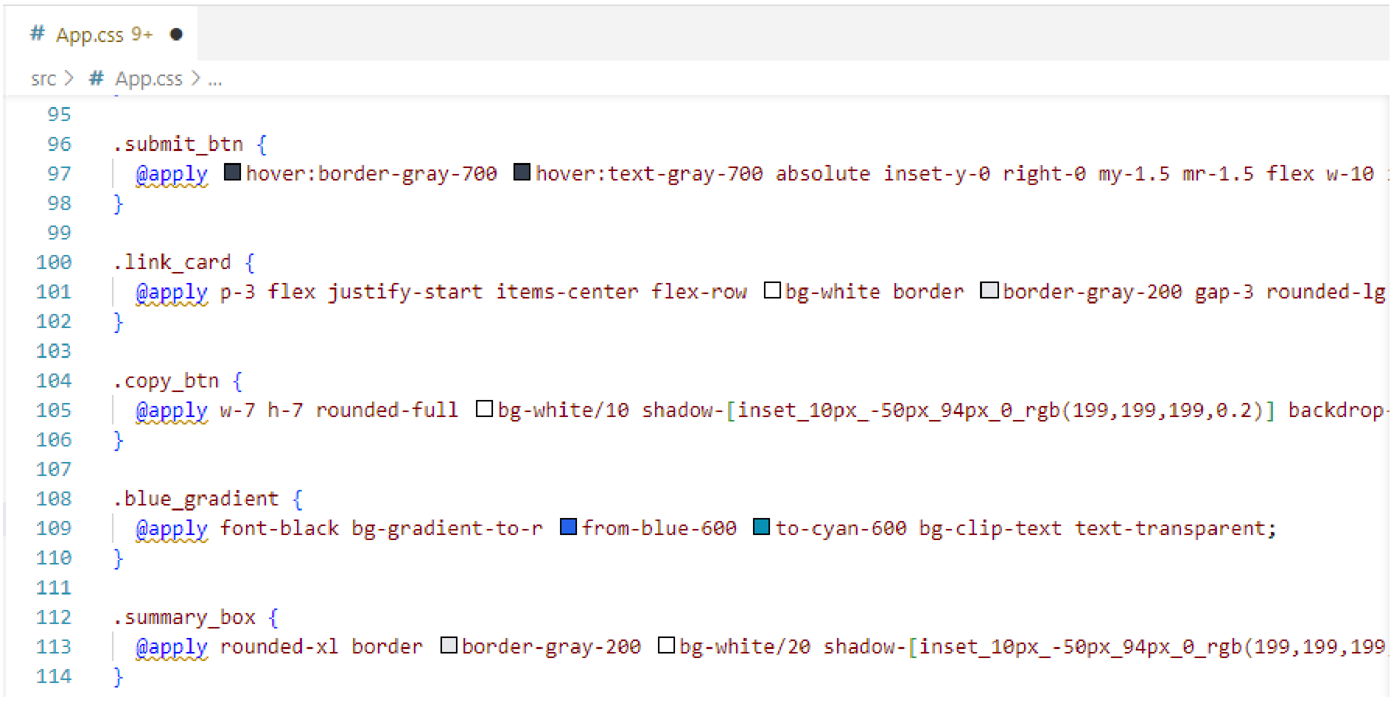
```css
  height: 100%;
  top: 0;
  opacity: 0.4;
  filter: invert(1);
}

.gradient {
  height: fit-content;
  z-index: 3;
  width: 100%;
  max-width: 640px;
  background-image: radial-gradient(
      at 27% 37%,
      hsla(215, 98%, 61%, 1) 0px,
      transparent 0%
    ),
    radial-gradient(at 97% 21%, hsla(125, 98%, 72%, 1) 0px, transparent 50%),
    radial-gradient(at 52% 99%, hsla(354, 98%, 61%, 1) 0px, transparent 50%),
    radial-gradient(at 10% 29%, hsla(256, 96%, 67%, 1) 0px, transparent 50%),
    radial-gradient(at 97% 96%, hsla(38, 60%, 74%, 1) 0px, transparent 50%),
    radial-gradient(at 33% 50%, hsla(222, 67%, 73%, 1) 0px, transparent 50%),
    radial-gradient(at 79% 53%, hsla(343, 68%, 79%, 1) 0px, transparent 50%);
  position: absolute;
  content: "";
  width: 100%;
  height: 100%;
  filter: blur(100px) saturate(150%);
  top: 80px;
  opacity: 0.15;
}
```

Snippet 14



```css
@media screen and (max-width: 640px) {
  .main {
    padding: 0;
  }
}

/* Tailwind Styles */

.app {
  @apply relative z-10 flex justify-center items-center flex-col max-w-7xl mx-auto sm:px-16 px-6;
}

.black_btn {
  @apply rounded-full border border-black bg-black py-1.5 px-5 text-sm text-white transition-all h
}

.head_text {
  @apply mt-5 text-5xl font-extrabold leading-[1.15] text-black sm:text-6xl text-center;
}

.orange_gradient {
  @apply bg-gradient-to-r from-amber-500 via-orange-600 to-yellow-500 bg-clip-text text-transparent;
}

.desc {
  @apply mt-5 text-lg text-gray-600 sm:text-xl text-center max-w-2xl;
}

.url_input {
  @apply block w-full rounded-md border border-gray-200 bg-white py-2.5 pl-10 pr-12 text-sm shadow-lg
}
```

Snippet 15



```css

.submit_btn {
  @apply hover:border-gray-700 hover:text-gray-700 absolute inset-y-0 right-0 my-1.5 mr-1.5 flex w-10
}

.link_card {
  @apply p-3 flex justify-start items-center flex-row bg-white border border-gray-200 gap-3 rounded-lg
}

.copy_btn {
  @apply w-7 h-7 rounded-full bg-white/10 shadow-[inset_10px_-50px_94px_0_rgb(199,199,199,0.2)] backdrop-
}

.blue_gradient {
  @apply font-black bg-gradient-to-r from-blue-600 to-cyan-600 bg-clip-text text-transparent;
}

.summary_box {
  @apply rounded-xl border border-gray-200 bg-white/20 shadow-[inset_10px_-50px_94px_0_rgb(199,199,199
}
```

Snippet 16

Future Work

- **Advanced summarization techniques**
  - Implement more sophisticated text summarization algorithms, such as deep learning-based models, to generate more comprehensive and informative summaries.
- **Summarization length options**
  - Allow users to select the desired length of the summary, catering to different needs and preferences.
- **Topic selection**
  - Provide the ability to specify the main topic of the text to be summarized, enabling more focused and relevant summarization.
- **Sentiment analysis**
  - Integrate sentiment analysis to identify the overall emotional tone of the text and incorporate it into the summary.
- **Keywords extraction**
  - Identify and display the most important keywords from the text to provide a quick overview of the key points.
- **Supporting multiple languages**
  - Enable the summarization of text in different languages to broaden the app's usability.
- **Summarizing from various sources**
  - Allow users to paste text directly, input URLs to web pages, or upload documents for summarization.

CONCLUSION

This summarizer uses article extractor and summarizer API for extracting the article from the link and summarizing it. This project was made with React, Vite.js and Rapid API. When a user pastes a link to an article in the search bar, it extracts the article from that link, summarizes the article and presents the summary to the user within five seconds. The search history of user is also stored, which can be further used by copying the link.

This project is deployed on netlify and can be accessed by following the link: https://frabjous-tarsier-14495c.netlify.app/

REFERENCES

[1] Jon Duckett, “HTML and CSS: Design and Build Web Sites”, 2011

[2] Chris Aquino, “Front-End Web Development: The Big Nerd Ranch Guide”, 2016

[3] Chong Lip Phang, “Mastering Front-End Web Development”, 2020

[4] Deitel & Goldberg, “Internet and world wide web- How to program”, Pearson Education

[5] Hans Bergsten, “Java Server Pages”, SPD O’Reilly

[6] Chris Bates, “Web programming, building internet applications”, 2nd edition, Wiley