Project Write-Up

Ruslan Mukhamedvaleev

Accessible Articles

The writing in this PDF is an expansion of my earlier write-up on this project, which can be found at https://github.com/digitalRM/CougHacks2024. The original write-up on GitHub, and my writing here, are solely my work.

Goal

We competed in CougHacks 2024 to build an app in 24 hours, overnight. This is the project we ultimately built to address the hackathon's theme of building productivity tools.

"Efficiency and Productivity". Projects should be related to this theme: something that would make any kind of task faster and better, or in some way promote or relate to efficiency and productivity. Bonus points for using AI in some way.

Team

Our team consisted of four Kamiak High School students: Myself, Rachel Chu, Wanhao Zhang, and Murugan Sakthivel. Lindsey Ehrlich, another student from Kamiak, contributed to researching the potential use of local AI, but was neither a code contributor nor a team member.

- **Individual Contributions:**
  - Led the team as the only member with prior React/JavaScript experience.
  - Served as the sole designer, creating the UI layout, visual style, and user experience.
  - Implemented web scraping functionality to collect and process data.
- **Team Collaboration:**
  - Supported teammates with JavaScript training, code reviews, and project architecture guidance.
  - Collaborated on the Python backend, assisting with API integration and debugging.

Ideation

- We briefly considered building a consolidated search tool for different types of entertainment (e.g., movies, video games, books), a shopping platform for promoting small businesses, and a smart to-do list app before landing on our fourth and final idea: “Accessible Articles.”

- We thought about the issues we shared in common and realized that half of us had trouble reading online due to certain preconditions that made it more difficult to read text. Thus, the idea to create a tool to help others like us was born!

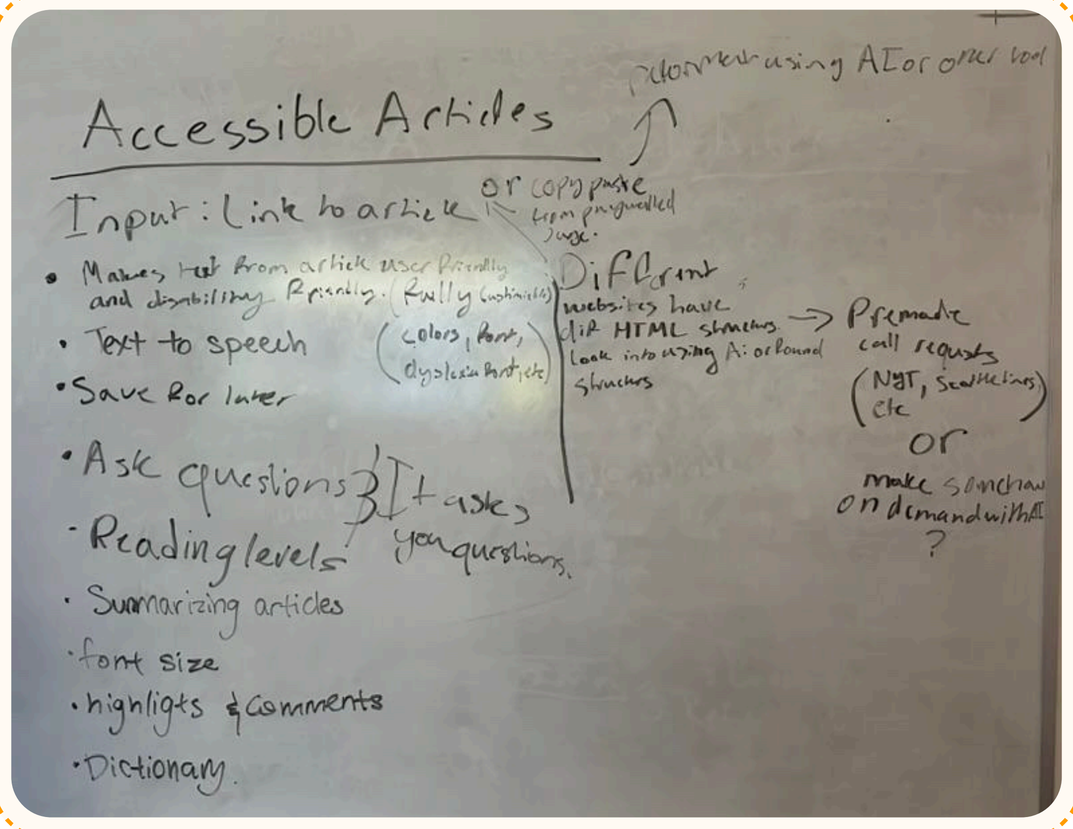
The Problem

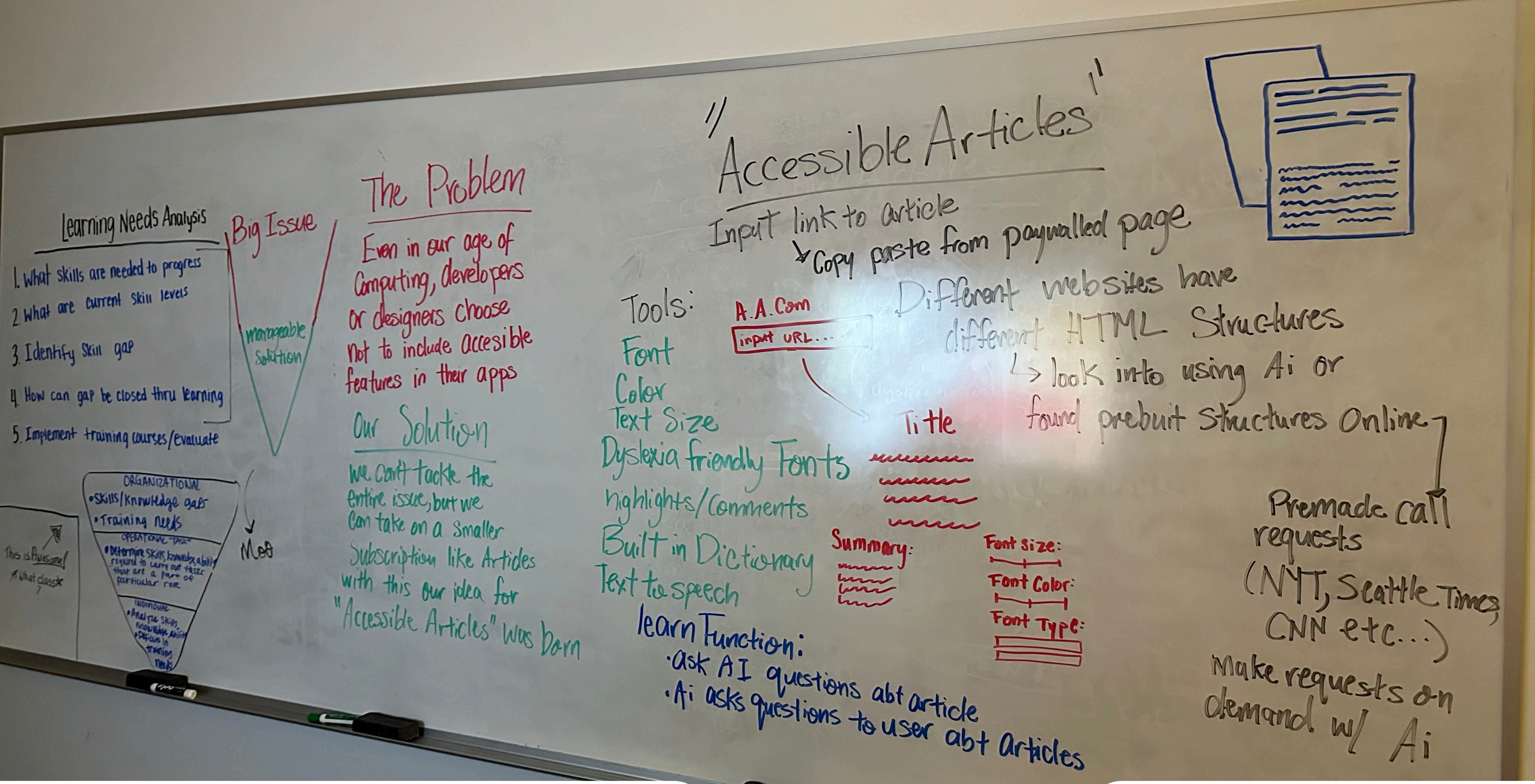
- Even with huge leaps in accessibility tools, designers often choose not to or fail to include accessible features in their apps. This makes performing simple tasks like reading incredibly difficult for people with disabilities like dyslexia or eye conditions like astigmatism.

- This was a deeply personal issue for me and one of my fellow teammates. Astigmatisms in both my eyes make it hard to read ultra-contrasting colors. For my teammate with Dyslexia, regular fonts often found on the web are extremely hard to read.

Planning

- We knew that we would not have the time nor the resources to take on the entire issue, so we decided to start with a small subsection: web-based articles.
- We wanted a tool that allowed users to fully customize articles they find across the internet to make them accessible.




Learning Needs Analysis
1. What skills are needed to progress
2. What are current skill levels
3. Identify skill gap
4. How can gap be closed thru learning
5. Implement training courses/evaluate
Big Issue
manageable solution
ORGANIZATIONAL
•Skills/Knowledge gaps
•Training needs
OPERATIONAL - TASK
•Determine skills, knowledge, ability required to carry out tasks that are a part of particular role
INDIVIDUAL
•Analyze skills, knowledge, ability
•Discuss in training needs
Moo
The Problem
Even in our age of Computing, developers or designers choose not to include accesible features in their apps
Our Solution
We can't tackle the entire issue, but we can take on a smaller subscription like Articles with this our idea for "Accessible Articles" was born
"Accessible Articles"
Input link to article
↳ Copy paste from paywalled page
Tools:
Font
Color
Text Size
Dyslexia friendly Fonts
highlights/comments
Built in Dictionary
Text to speech
learn Function:
·ask AI questions abt article
·Ai asks questions to user abt articles
A.A.Com
input URL...
Title
Summary:
Font size:
Font Color:
Font Type:
Different websites have different HTML Structures
↳ look into using Ai or found prebuit structures Online
Premade call requests (NYT, Seattle Times, CNN etc...)
Make requests on demand w/ Ai

Building

- For our stack, I chose Next.js (a React framework) for the front-end because of my experience with JavaScript, React, and JSX. Next.js was the best framework at the time for server-side rendering, which made things load faster.
- On the back end, we chose Python, as all of us had experience with the language, and I could jump in to help if there were any problems.
- We initially debated using Java, but decided against it because we were under a strict time constraint and had no prior experience with it. Using two scripting languages made the most sense at the moment.

Frontend

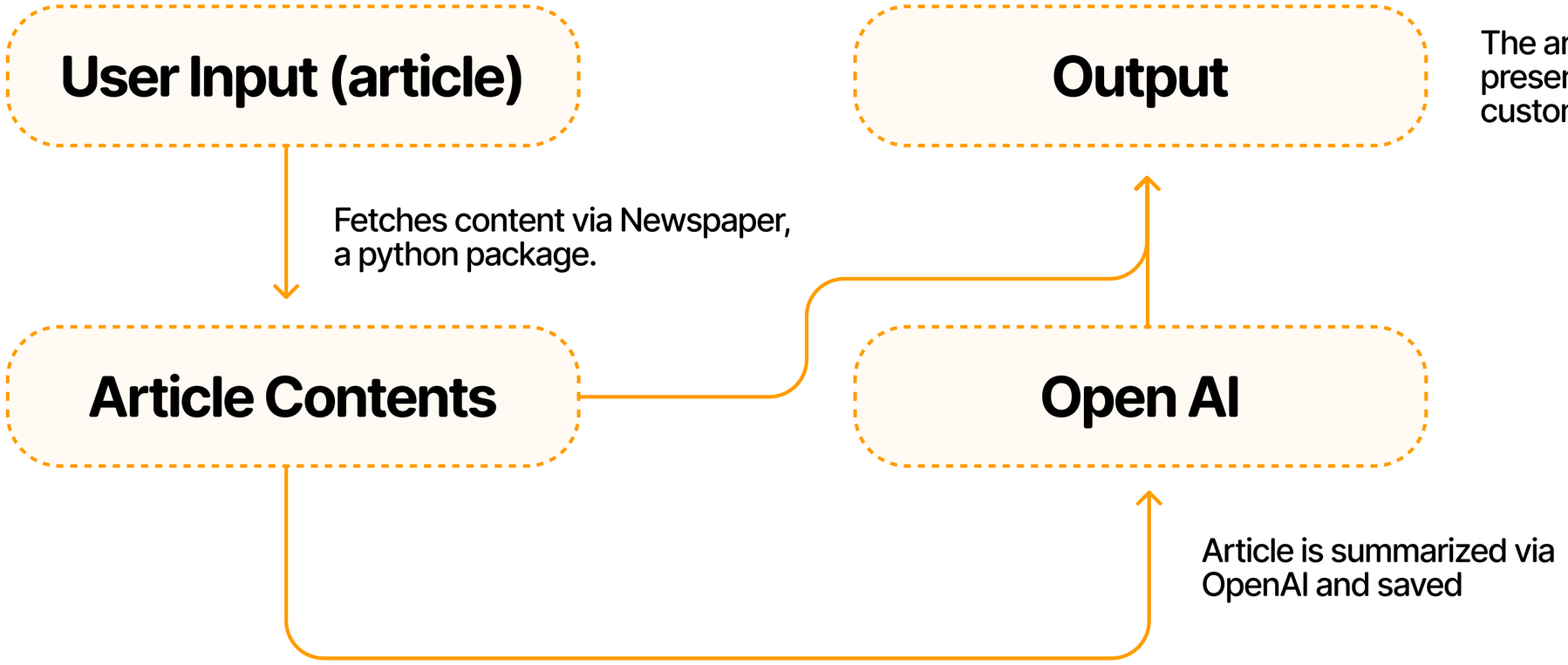
- I had never worked with handling global state, at the time, especially for variables that directly interact with the styling of the app.
- I decided that the best way to handle this was through a centralized page with child components that would receive the state, which, in hindsight, proved to be a good idea.
- Accessibility variables (font, colors, size) would directly affect styling from inside the parent component.
- They could be modified by child components (like the color changer component) through the passing down of state and state setters from parent.

```
const fontOptions = [
  { value: 1, label: 'Inter Font' },
  { value: 2, label: 'Inclusive Sans' },
  { value: 3, label: 'Comic Sans' },
  { value: 4, label: 'Roboto Mono' },
  { value: 5, label: 'Merriweather' },
];
```

```
const [backgroundColor, setbackgroundColor] = useState('#f8f8f8');
const [textColor, setTextColor] = useState('#111111');
const [fontFamily, setFontFamily] = useState('Inter');
```

```
return (
  <main style={{backgroundColor: backgroundColor}} className={`font-${fontFamily.toLowerCase()} bg-repeat w-full overflow-hidden p-3 sm:p-12 relative h-full min-h-screen`}>
    <Hero textColor={textColor} />
    <InputArticle url={url} setUrl={setUrl} fetchAPI={fetchAPI} isLoading={isLoading} error={error} backgroundColor={backgroundColor} setbackgroundColor={setbackgroundColor} textColor={te
    <Article data={data} isData={isData} enchancedData={enchancedData} isLoading={isLoading} textColor={textColor} fontSize={fontSize} />
  </main>
);
```

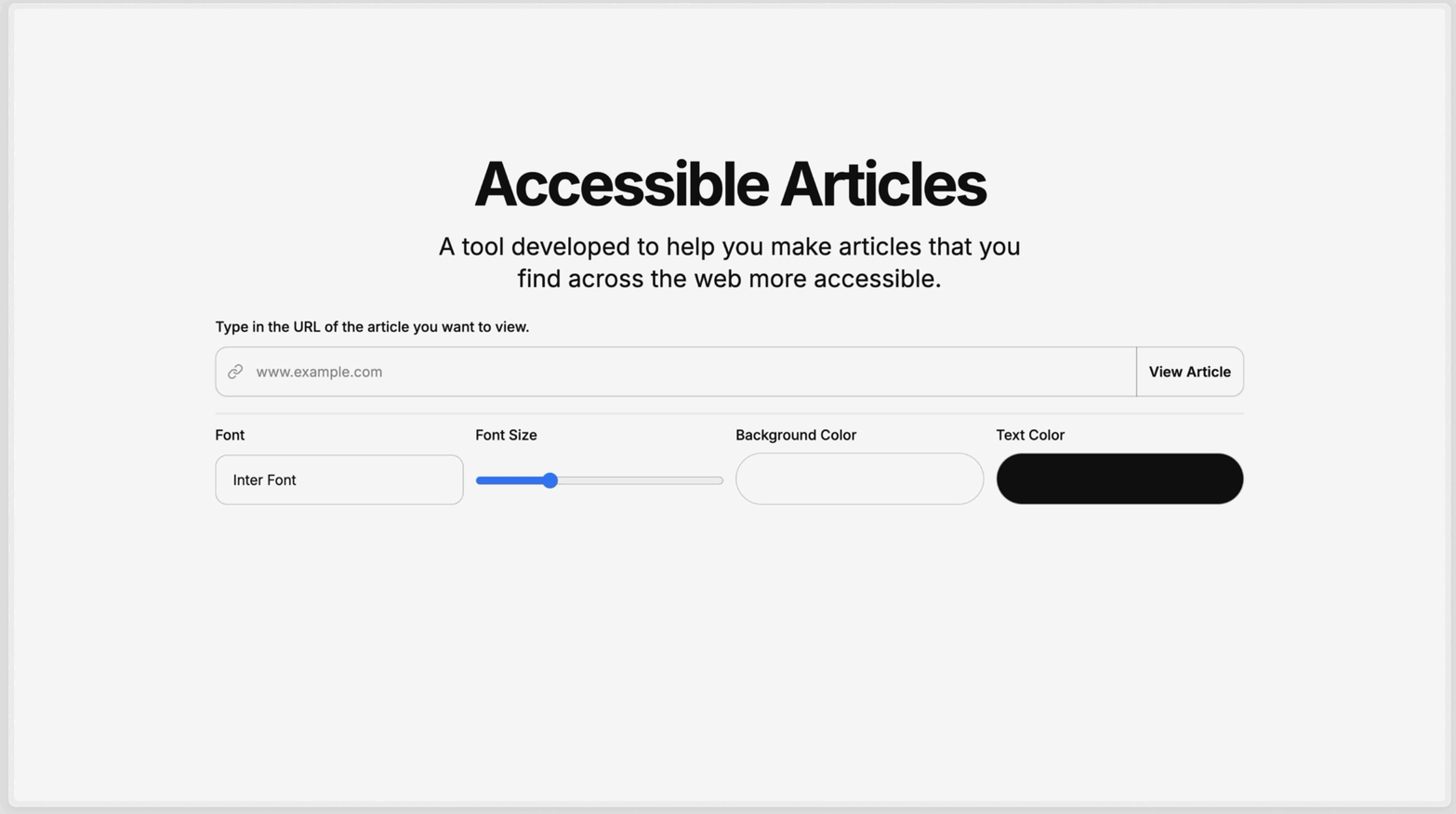
Data Flow

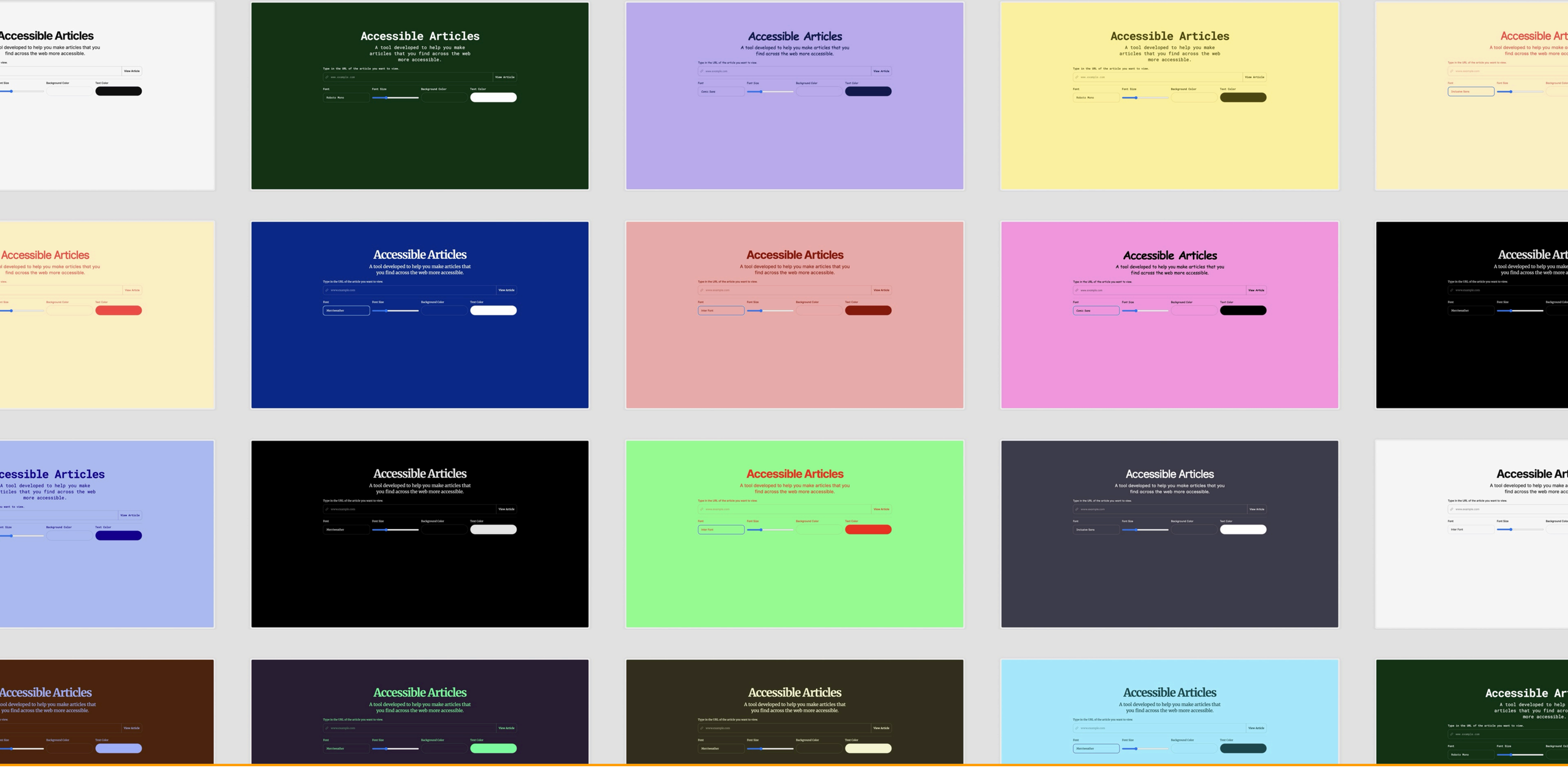

User Input (article)
Fetches content via Newspaper, a python package.
Article Contents
Open AI
Article is summarized via OpenAI and saved
Output
The article and the summary are presented back to the user with all customization options.

Tool

Here is how the final tool looks! You are able to customize font, background color, font size, and text color!

Code on GitHub: https://github.com/digitalRM/CougHacks2024





Accessible Articles Project Write-Up

Mukhamedvaleev 12