

AI Website Summarizer - Solution Documentation

1. Introduction

The **AI Website Summarizer** is a modern React application designed to provide concise summaries of web pages. By leveraging the power of Large Language Models (LLMs) via the Groq API, the application extracts content from a given URL and generates a brief, easy-to-read summary. This tool is particularly useful for quickly understanding the core message of long articles, blog posts, or documentation without reading the entire text.

2. Architecture Overview

The application follows a client-side architecture with integration to external APIs for content fetching and summarization.

Data Flow

1. **User Input:** The user enters a public URL into the application.
2. **Validation:** The application validates the URL format.
3. **Content Fetching (Proxy):** To bypass Cross-Origin Resource Sharing (CORS) restrictions, the application uses `api.allorigins.win` as a proxy to fetch the raw HTML content of the target URL.
4. **Content Cleaning:** The raw HTML is processed to remove scripts, styles, and HTML tags, extracting only the meaningful text content.
5. **AI Summarization:** The cleaned text is sent to the **Groq API** (using the `llama-3.1-8b-instant` model).
6. **Formatting & Display:** The AI-generated summary is formatted (converting markdown-like syntax to HTML) and displayed to the user.

3. Key Features

3.1 URL Validation

Before processing, the application ensures the input is a valid URL with `http` or `https` protocol. This prevents unnecessary API calls and provides immediate feedback to the user.

3.2 CORS Handling

Directly fetching external websites from a browser often fails due to CORS policies. The solution employs `allorigins.win` as a proxy service to successfully retrieve content from any public URL.

3.3 Intelligent Content Extraction

The application doesn't just send raw HTML to the AI. It implements a cleaning mechanism that:

- Removes `<script>` and `<style>` tags to eliminate non-visible code.
- Strips all HTML tags to leave only the text.
- Normalizes whitespace.
- Truncates the text to 4000 characters to ensure it fits within the model's context window and optimizes performance.

3.4 AI-Powered Summarization

The core intelligence is provided by the **Groq API**, utilizing the `llama-3.1-8b-instant` model. This model is chosen for its speed and efficiency in processing natural language tasks.

3.5 Smart Formatting

The raw text returned by the AI is processed to enhance readability:

- **Bold Text:** `**text**` is converted to `text`.
- **Lists:** Bullet points and numbered lists are detected and structured with `` and `` tags.
- **Headings:** Lines ending with a colon (e.g., "Key Points:") are formatted as headings.

4. Technical Implementation

4.1 Core Components

`src/App.tsx`

The main component handling the application lifecycle:

- **State Management:** Uses `useState` for `url`, `summary`, and `loading` states.
- **handleSummarize :** Orchestrates the fetching, cleaning, and summarization process.
- **formatSummary :** A utility function that parses the AI response and converts it into structured HTML for display.

`src/services/ai.ts`

Encapsulates all external API interactions:

- **fetchWithRetry :** A robust fetch wrapper that attempts to retrieve content multiple times (default 2 retries) in case of network instability.
- **summarizeText :** Sends the cleaned text to the Groq API with a system prompt instructing it to "Summarize the following webpage content briefly."

`src/components/Loader.tsx`

Provides visual feedback to the user while the asynchronous operations (fetching and summarizing) are in progress.

4.2 Technologies Used

- **Frontend Framework:** React 19
- **Build Tool:** Vite
- **Language:** TypeScript
- **Styling:** CSS (with responsive design considerations)
- **AI Provider:** Groq (Llama 3.1)

5. Setup and Installation

Prerequisites

- Node.js (v18 or higher)
- npm or yarn
- A Groq API Key

Steps

1. **Clone the repository:**

```
git clone <repository-url>
cd ai-summarize-url
```

2. Install dependencies:

```
npm install
```

3. Configure Environment Variables:

Create a `.env` file in the root directory and add your Groq API key:

```
VITE_GROQ_API_KEY=your_groq_api_key_here
```

4. Run the development server:

```
npm run dev
```

5. Build for production:

```
npm run build
```

6. Future Improvements

- **History:** Save previously summarized URLs and their summaries.
- **Custom Prompts:** Allow users to specify the focus of the summary (e.g., "Technical details", "Market overview").
- **Multi-Language Support:** Summarize content in different languages.
- **Export:** Allow users to download the summary as a PDF or text file.