Design Specification

Group 1

December 9, 2024

Contents

CAL STATE LA

Google Chrome Built-in AI Challenge

1 Introduction

1.1 Purpose

This document outlines the system architecture, UI components, and data structures that were used for our Chrome AI-Enhanced Web Application. It should serve as a blueprint for developers, designers, and stakeholders to learn and understand the design decisions and underlying technologies for this product.

1.2 Scope

The specifications will cover the frontend (in this case the Chrome extension), the AI API integrations, and the backend components that go into this.

2 System Architecture

The application runs entirely in the browser, using Chrome's built-in AI APIs. The way this works is as follows:

1. User interacts with the UI elements in the Chrome Extension.
2. UI sends a request to the AI APIs (Prompt, Summarization, Translation, Rewrite).
3. The AI models process the input and then return the results.
4. UI updates dynamically to display results without the need of any server calls.

3 Page-by-Page Breakdown

3.1 Main Dashboard

- **Purpose:** Central hub where all the AI features will be accessed from.
- **UI Elements:**
  - Navigation bar with links to "Prompts", "Summarize", "Translate", "Rewrite".
  - A responsive layout
- **Data Flow:** User input in forms is sent to the relevant AI API and then processed.

3.2 Prompt Page

- **Purpose:** Generate dynamic prompts based on the user's queries.
- **UI Elements:** Text field for input, "Generate Prompt" button, and results area.
- **Interactions:** User clicks "Generate Prompt" → JS function sends input to Prompt API → Render the suggestion below.

3.3 Summarize Page

- **Purpose:** Provide concise summaries of long texts.
- **UI Elements:** Large text area for input, upload option, "Summarize" button, and results panel.
- **Interactions:** Input text or selected file → Summarization API → Output a summarized version of the writing.

3.4 Translate Page

- **Purpose:** Translate text between languages.
- **UI Elements:** Source text area, language selectors (dropdowns), and "Translate" button.
- **Interactions:** Input text + selected languages → Translation API → Output in target language area.

3.5 Rewrite Page

- **Purpose:** Suggest alternative phrasings for user-provided text.
- **UI Elements:** Text input area, "Rewrite" button, suggestions panel.
- **Interactions:** User pastes text → Clicks "Rewrite" → API returns multiple suggestions → User selects a preferred suggestion.

4 Components and Tools

4.1 Frontend Components

- **React.js**: A library particularly designed for web applications and UI.
- **Material UI**: A library that provides ready-made UI components and layout grids.

4.2 AI APIs

- **Prompt API**: Accepts user queries, returns suggestions.
- **Summarization API**: Converts long text into a short summary.
- **Translation API**: Translates text between supported languages.
- **Rewrite API**: Suggests alternative wordings.

5 Performance Considerations

The AI APIs run locally, this was done in order to allow for quick responses and minimal latency.

6 Accessibility

- Option to customize font size.
- Light and Dark mode.
- Multiple languages.
- Proper ARIA labels for screen reader support.

7 Future Enhancements

- There are plants to add voice commands in order to allow for hands-free operations.
- The translation is limited to the languages provided by the API so there are plans of adding even more languages.