Technologies Used

1. **React.js**: The front-end framework used to build the user interface (UI) of the extension. React provides a dynamic and responsive interface, allowing users to interact with the extension easily.
2. **Groq API**: A powerful language model API used to generate text summaries. It takes in user-provided text (e.g., selected text from a webpage) and returns a summarized version based on a system prompt.
3. **ElevenLabs API**: An API used to generate high-quality text-to-speech (TTS) audio. This API converts the summarized text into speech, which can be played back to the user.
4. **Chrome Extensions API**: Used to interact with the browser, allowing the extension to read the selected text from web pages, display UI elements, and store user settings locally using **chrome.storage.sync**.
5. **TailwindCSS**: A utility-first CSS framework used to style the extension's UI components, making the design simple yet responsive.

How the Code Works

The whole code for the working of the extension is available in the App.tsx file inside the src folder.

App.tsx file

Functions and Key Components

State Initialization

Our extension does state management managed by React's "useState" hook. This object holds various values critical to the app's functionality:

- **apiKey**: Stores the user's Groq API key for summarization.
- **systemPrompt**: Custom user prompt to guide summarization style.
- **isActive**: Boolean toggle indicating whether the extension is enabled.
- **summary**: Stores the generated summary from the Groq API.
- **isLoading**: Indicates if summarization is in progress.
- **error**: Stores error messages for display.
- **elevenLabsApiKey**: Stores the ElevenLabs API key for text-to-speech functionality.
- **voiceId**: ID for the ElevenLabs voice used for audio synthesis.
- **isPlayingAudio**: Tracks if audio playback is active.
- **currentView**: Determines the current interface view (`main`, `settings`, or `help`).
- **audioPlayer**: Stores an audio element for playback.
- **isPaused**: Tracks the playback state (paused or playing).
- **success**: Stores success messages for user feedback.

---

speakText function

Purpose: Converts the generated summary into speech using the ElevenLabs API.

1. Validates that **summary** and **elevenLabsApiKey** are set.
2. Make a POST request to the ElevenLabs API with:
    - Text to synthesize.
    - Model and voice settings.
3. Language code (**hi,** in this case, representing Hindi).
4. Plays the generated audio using an **Audio** element.
5. Handles playback events (like ending) to reset the state.
6. Currently, the extension would be able to handle the Hindi language, however, this is future work.

Important Notes:

- The **stability** and **similarity_boost** parameters configure voice characteristics.
- Uses **URL.createObjectURL** to load and play the audio.

---

togglePlayPause function

Purpose: Toggles between play and pause for the audio playback.

1. Check if an **audioPlaye**r instance exists in the state.
2. If the player is paused, resume playback. Otherwise, pauses it.
3. Updates the **isPaused** flag in the state.

---

use effect (Settings Initialization)

Purpose: Loads saved settings from Chrome's synchronized storage when the app initializes.

1. Fetches stored values for:
    - **apiKey**
    - **systemPrompt**
    - **isActive**
    - **elevenLabsApiKey**
    - **voiceId**
2. Updates the state with the retrieved values.
3. Currently, we are not using the voiceId. But can increase the functionality of the extension by using the variable.

---

saveSettings function

Purpose: Saves the user's API keys and other settings to Chrome's storage.

1. Validates that both **apiKey** and **elevenLabsApiKey** are present.
2. Saves settings using **chrome.storage.sync.set**.
3. Updates the **success** state and redirects the user to the **main** view if keys are set.

Important Notes:

- Displays success or error messages based on the operation's outcome.
- Uses **setTimeout** to clear success messages after 3 seconds.

---

summarizeText function

Purpose: Summarizes selected text using Groq API.

1. Retrieves selected text from the active webpage using `getSelectedText`.
2. Validates:
    - Text is selected.
    - The **apiKey** is available.
3. Creates a Groq client with the user's API key.
4. Sends a chat completion request to the Groq API:
    - Combines the **systemPrompt** and selected text.
    - Uses the **llama3-8b-8192** model for summarization.
5. Updates the **summary** state with the response or shows an error.

Important Notes:

- The model used (llama3-8b-8192) has 8 billion parameters and supports long-context tasks (up to 8192 tokens).
- We can increase the functionality of the app in the future by making the user use different models from the extension.
- Handles empty or null responses gracefully.

---

getSelectedText Function

Purpose: Retrieves the currently selected text from the active browser tab.

1. Uses **chrome.tabs.query** to identify the active tab.
2. Executes a script in the active tab to extract the selected text using **window.getSelection()**.

Important Notes:

- Returns an empty string if no text is selected.

---

areApiKeysSet function

Purpose: Verifies if both **apiKey** and **elevenLabsApiKey** are set in the state.

1. Returns **true** if both keys are available; otherwise, **false**.

Different Views/screens available in the extension

Main View

- It Displays:
  - A field to input the **systemPrompt**.
  - A button to trigger text summarization.
  - Generated summaries with playback options **(Listen, Pause, or Resume)**.
  - Success and error messages.

Settings View

- Allows users to:
  - Input their Groq API Key.
  - Input their ElevenLabs API Key.
  - Save these settings for future use.

Help View

- Provides a user guide:
  - How to obtain API keys.
  - Steps to use the extension effectively.
  - Disclaimer

Other Key Features

- **Error Handling**: Comprehensive error messages for both text summarization and audio synthesis.
- **Accessibility**: Using **aria-label, aria-live, and role** attributes to improve screen reader compatibility.
- **Dynamic State Management**: Reactively updates the UI based on user inputs and API responses.

Models Used

Groq Model: llama3-8b-8192

- **Capabilities**:
  - Processes up to 8192 tokens, ideal for long-text summarization.
  - A high-parameter model (8 billion) ensures a nuanced understanding of the text.
- **Application**: Used to summarize selected webpage text based on user-provided prompts.

ElevenLabs Text-to-Speech Model

- **Capabilities**:
  - Synthesizes natural-sounding speech.
  - Supports fine-tuning of voice stability and similarity.
- **Application**: Converts summaries into speech with configurable voice settings.

Improvements to be done:

1. If the user selects text that is more than the context length of the model, it will make the model return error.

Public Folder:

Manifest.json

This manifest.json file configures the **Adhd Ally** Chrome extension using **Manifest V3**. It specifies the extension's name, version (1.0), and permissions like activeTab (access the active browser tab), storage (save/retrieve data), and scripting (inject scripts into web pages). It also defines a popup (index.html) and an icon (icon_adhd_ally.png).

Other contents in the public folder

1. It contains the icon for the extension “icon_adhd_ally.png”
2. Manifest.json which will be helpful for the chrome extension

Working of the Extension:

User Experience with the ADHD Ally Extension:

Below is a walkthrough of how the ADHD Ally functions when a user, interacts with its various features.

---

1. Initial Setup:

- **Action**: Install the extension and open it by clicking its icon in the browser.
- **Output**:

    - We see a **Settings Icon (⚙️)**, **Help Icon (❓)**, and a text area prompting us with "How would you like me to help?".
    - If you have not set the API keys, you are redirected to the **Settings** page.

---

2. Navigating to Settings:

- **Action**: Click the ⚙️ **Settings** button at the top.
- **Output**:
    - A page titled **Settings** opens where we can:
        - Enter the **Groq API Key**.
        - Enter the **ElevenLabs API Key**.
        - Save the keys by clicking the "Save Settings" button.
- **Details**:
    - If we do not enter both keys and try to save, we see an error message: **"Both API keys are required"**.
    - If successful, we receive a success message: **"Settings saved successfully!"**.

---

3. Navigating to Help:

- **Action**: Click the ❓ **Help** button at the top.
- **Output**:
    - A **Help** section opens with detailed instructions on:
        - How to obtain the required API keys.
        - Steps for using the extension, include summarizing text and listening to summaries.
    - A "Go to Settings" button lets us navigate back to set API keys if needed.

---

4. Main View (Using the Extension):

- **Action**: Once both API keys are set, return to the main view.
- **Output**:
    - You see:
        - A text area titled **"How would you like me to help?"**, where you can type custom instructions (e.g., "Simplify the text" or "Break it into smaller parts").
        - A **Summarize** button to process selected text.

---

5. Summarizing Text:

- **Action**:
    - Highlight some text on a webpage.
    - Click the **Summarize** button after optionally providing a custom prompt.
- **Output**:

    - If text is selected, the app uses the **Groq API** to summarize it based on your prompt.
    - Once the summary is generated:
        - It appears under a **Summary** section in the app.
        - The summary respects your instructions, such as breaking the text into smaller parts or simplifying it.
    - If no text is selected, we see an error message: **"Please select some text first."**
    - If the API key is missing, we see: **"API key is required."**

---

6. Listening to the Summary:

- **Action**: Click the 🔊 **Listen** button after a summary is displayed.
- **Output**:
    - The app uses the **ElevenLabs API** to convert the summary to speech in the selected voice (default is "21m00Tcm4TlvDq8ikWAM").
    - We can hear the summary read aloud.
- **Additional Features**:
    - If we want to pause audio, click ⏸️ **Pause**.
    - To resume, click ▶️ **Resume**.
    - If the audio ends, it resets the play state.

---

7. Error Handling:

- **Scenario**: The app detects errors in various situations.
- **Examples**:
    - **When summarizing fails**: Displays a message like **"Failed to summarize text"** or **"Unknown error occurred during summarization."**
    - **When generating speech fails**: Displays a message like **"Failed to generate speech."**
    - If all is successful, a success message appears, e.g., **"Settings saved successfully!"**

For more details about how to run the extension in the browser, see the **readme.md** file

Note: I took the help of generative AI for the project to generate a few parts of code and the documentation.