BROWSER BUDDY DESIGN SPECIFICATION

SP-17 Browser Plugin



Elliott Brown Nathan Lynes Jacob Germana-McCray

Professor Perry | 02/04/24

Kennesaw State University | 4850-02/01 Senior Project

Table of Contents

1. Introduction

1.1 Outline

This document will go into details of the internal workings of the project, as well as present demonstrations and user stories which will show the intended use case and flow of the project.

1.2 System Overview

Browser buddy will function as an interface for the OpenAI GPT-4 LLM. This will be accomplished through an array of services with individual responsibilities to ensure a separation of concerns. The following diagram briefly illustrates the flow of interaction between a user and the service.

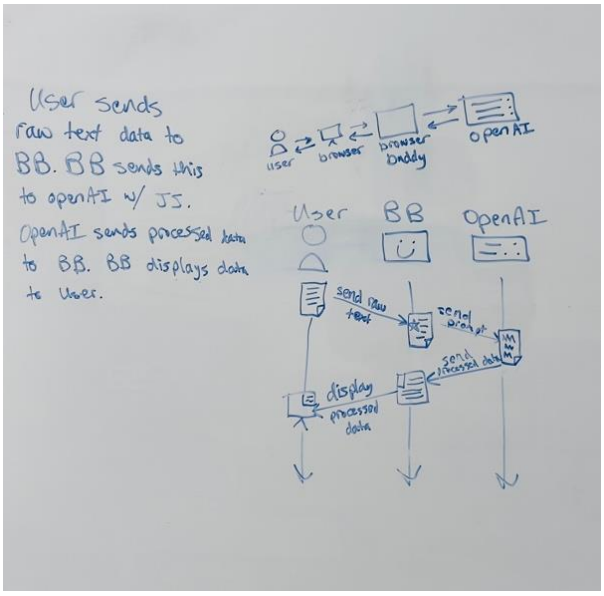

Figure 1. Simplified System Overview

2. Design Considerations

This section highlights determinants for the system's design.

2.1 Assumptions and Dependencies

- Summarized content will be sufficiently comprehendible.
- Users have Firefox and a stable internet connection.
- Third-party API will stay consistently available.

2.2 General Constraints

- Avoid heavy resource usage and slow performance.
- Support cross-platform between MacOS & Windows.

2.3 Goals and Guidelines

- Provide a one-click summarization experience.
- User-friendly interface and UX.
- Modular components for maintainability.
- Ensure responsiveness.

2.4 Development Methods

We will abide by a general programming format and style guide; such is defined below.

2.4.1 HTML/CSS

HTML/CSS will conform to the HTML5 & CSS3 standards. Kebab-case styles will be observed. Four-space indentation will be used for nested statements. Maximum column space will be set to 80 characters.

2.4.2 JavaScript

JavaScript will follow ES6 standards. There will be no third-party libraries used. Camel-case style will be used. Four-space indentation will be used for nested statements. Maximum column space will be set to 80 characters.

3. Architecture

Contained here is a list of directories that our software will be housed in. A basic description of each section is provided.

3.1 Root

Root of the browser buddy extension.

- / (root)
  - Manifest.json
  - README.md

3.2 Assets

Iconography that will be used throughout the program.

- /assets – static assets
  - Icons/

3.3 Source

The logic and styling for the extension. Each file will be explained further in section 5.2.

- /src – Core extension source code
  - Background.js
  - contentScript.js
  - summarize.js
  - api.js
  - cache.js
- /stylesheet.css – style for the popup.

3.4 Documentation

Documentation will provide a succinct description of how the program functions, as well as a developer quick-start guide. The user documentation will describe the usual flow of the program understandable to an average user.

- /docs – Documentation
    - Developer Documentation.md
    - System Design.md
    - User Documentation.md

4. Policies

- Meet browser extension security policies.
- Request only required permissions.
- Transmit no user data externally.
- Cache summaries using browser storage best practice.

5. Detailed Design of System

5.1 Classification and Definition

Browser Buddy is a *web extension*, which is an application that is executed within a web browser. Specifically, this web extension uses external APIs to process user-provided data.

5.2 Detailed Overview

The browser buddy extension will consist of several files operating as individual services.

- Background – Adds a right-click option for summarization, listens for a selection and sends the any highlighted text on the browser to summarize.js.
- Summarize – processes highlighted text into a prompt. This consists of appending a hardcoded string designed to guide the LLM into providing a better response, as well as an option string which will enable the ability to change “formats” of the response (i.e. bulleted list, paragraph, simple terms). The fully appended prompt “promptStr” will then be sent to the api_handler.
- API_Handler – prepares a POST request using JavaScript’s fetch() method using the prompStr built in the summarize service. After this, the api_handler will wait for a response from OpenAI, process each byte of the stream into readable strings, and push them to the content_script.
- Content_script – Updates the DOM using the response gathered by the API_Handler. Builds the widget that displays the response.

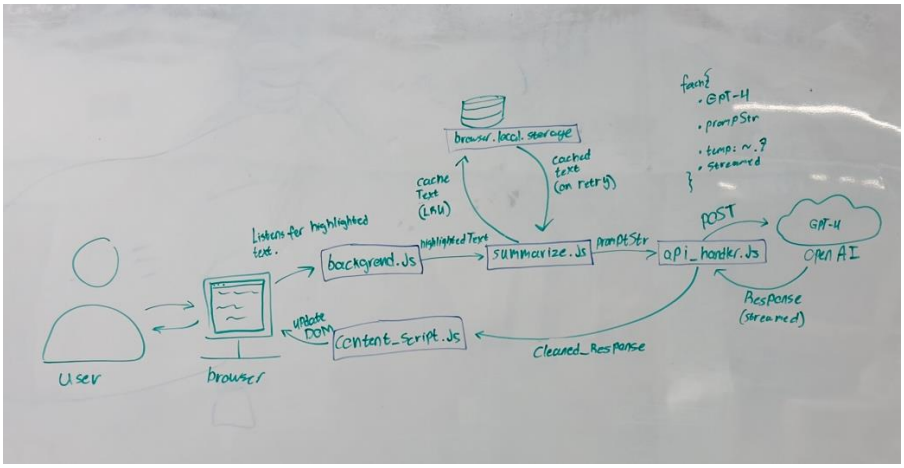

Figure 2. Software Design Diagram of the Browser Buddy extension.

5.3 Constraints

If the extension was to operate on a large scale, we would not be able to keep the hardcoded API key that we are currently using.

Future use would require that we publicly share our API key (this is a security problem) or require each user to pay for a private key. This change does open the door to monetizing Browser Buddy, as advertisements may be used to fund the API key payments. This limitation does not imply a significant change of functionality to support a large user base, so we only mildly consider it a limiting factor in our development.

5.4 User Stories

"I'm David, and I want to understand as much information on webpages as fast as I can, so that I can quickly participate in conversation about the topic therein."

- David is often late preparing for meetings at his office job, and he often is familiarizing himself with the points of content in the meeting as it happens. David wants to gain a quick and foundational understanding of the topics described on Wikipedia.

"I'm Emmy, and I want to read summaries of my friend's blogs in bullet points, because they talk too much, and I want to read it in specifically bullet point form."

- Emmy's friends post too much on their blogs and she often is expected to know about the things they have posted! Emmy wants to quickly go through their posts for the week and be done with it so that she does not miss valuable information about her friends' lives!

"I'm Eathan, and I want to black box new skills as quickly as possible so that I can complete my assignments in time."

- Eathan's boss often gives him assignments to create something for him that requires skills he has never learned before. Eathan is motivated to meet the expectations placed on him and needs to quickly understand many sources of information which introduce these new skills to him to succeed.

5.5 Resources and Processing

Due to the size of this project, the resource requirement is being considered as negligible. Web extensions—especially ones of this magnitude—should not require much memory or processing power. All processing will be done locally, so performance is dependent from machine to machine.

5.6 Interface

Our interface will be very briefly covered here. Figures 3 and 4 depict our GUI.

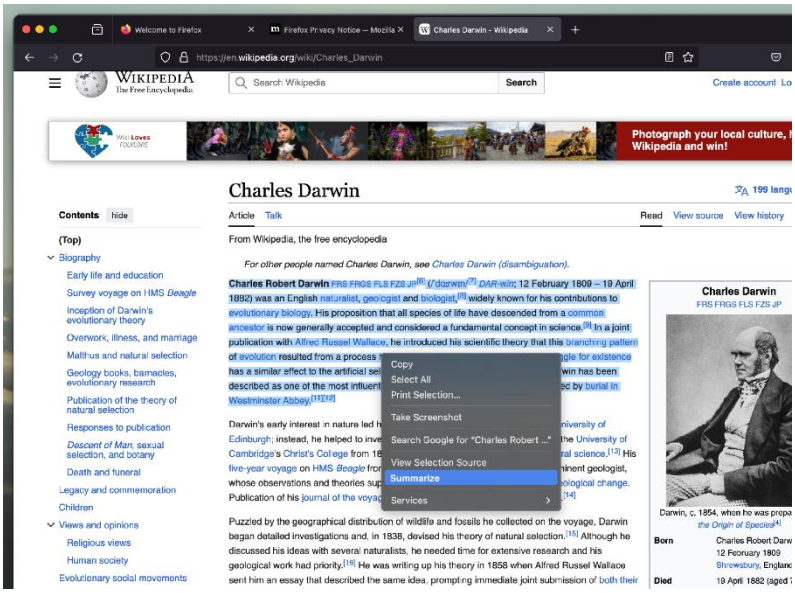

Figure 3. Demonstration of the "Summarize" button in the context menu.

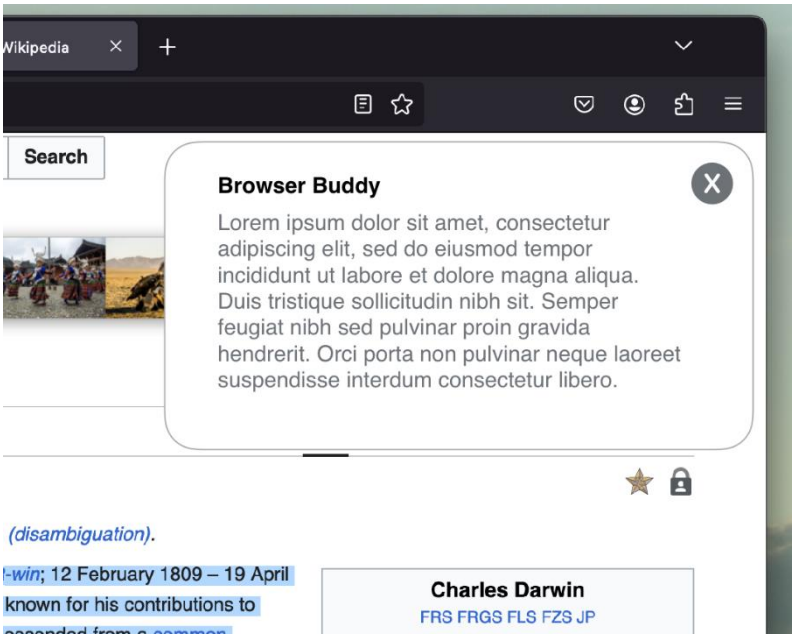

Figure 4. Demonstration of the styling of a summarization after the "Summarize" button is pressed.