Development Report: Generative AI-Powered Summarization Extension

Group 12:
Marius Hauger, Hauk Hauge Mo and Lars Erik Risholm

Introduction:

After seeing newspaper companies using GPT API to summarize their articles on the top of the page, we got the idea to make a chrome-extension that summarizes a full webpage. This way you could summarize whichever blog or article you'd like. It's also a cool way to get introduced to both making an extension, as well as implementing the GPT-3.5 Turbo API into our application.

Objective:

Develop a chrome web extension that leverages generative AI to summarize the webpage a user is browsing on into concise text.

Division of Labor:

With prior experience in Flask, setting up a backend development and API routing was the right initial job for Marius.

With some prior experience with using and installing Chrome extension-apps, Hauk took the challenge to learn the necessary difference between creating a webapp and chrome extension.

Lars took the job to signing up for the OpenAI account and focusing on the integration and optimal utilization of the GPT-3.5 Turbo API.

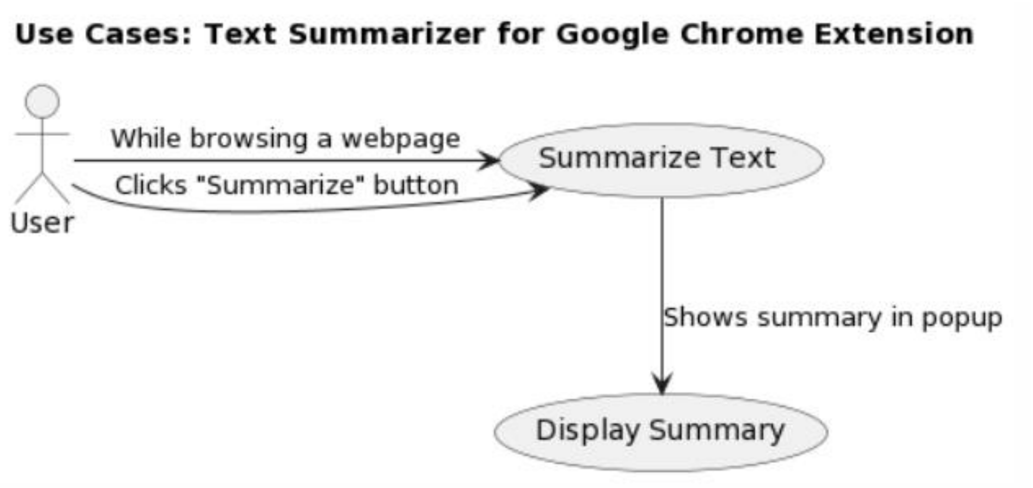
Although this was the initial labor-division, it was important that we worked together.

After some hours hard individual work, we met prepared and ready to work on the UML diagrams together.

UML - Diagrams

We believe that utilizing UML diagrams is an excellent method for planning and visualizing the structure of the application we are developing. After several days of individual work and thorough preparation, our group met well-prepared to create multiple diagrams. Our objective was to provide clear illustrations of the functionality and interactions within our text summary extension.

Use case diagram:


Use Cases: Text Summarizer for Google Chrome Extension
User
While browsing a webpage
Clicks "Summarize" button
Summarize Text
Shows summary in popup
Display Summary

Use case descriptions:

Use case: Summarize text

Actor: User

Goal: Get a summary of text from a web page

Description:

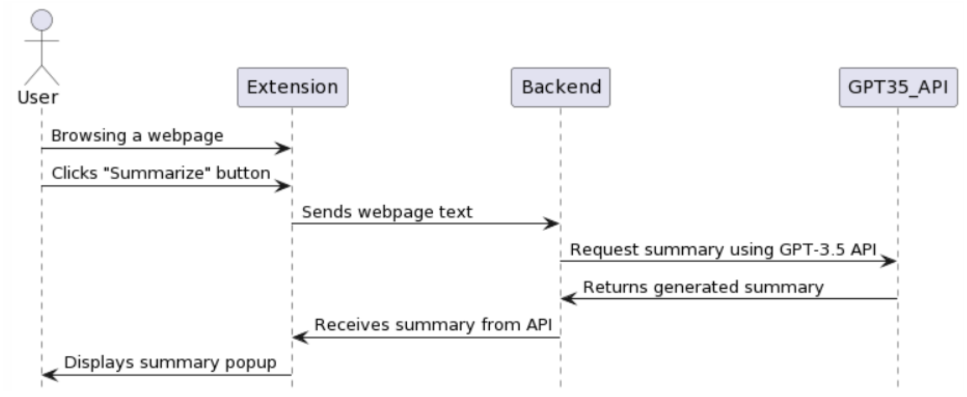
- The user surfes webpages when the extension is loaded
- The user clicks the "Summarize" button in the extension.
- The extension generates a summary of the content on the webpage.
- The summary is then displayed to the user in the extension's pop-up window.

End goal: The user receives a summarized version of the text content from the web page.

The use case diagram shows the user interactions with the app in a simple clear way. It describes how the user will use the extension to summarize text from a web page.

The use case descriptions provide a slightly more detailed description for the use case and the main goal.

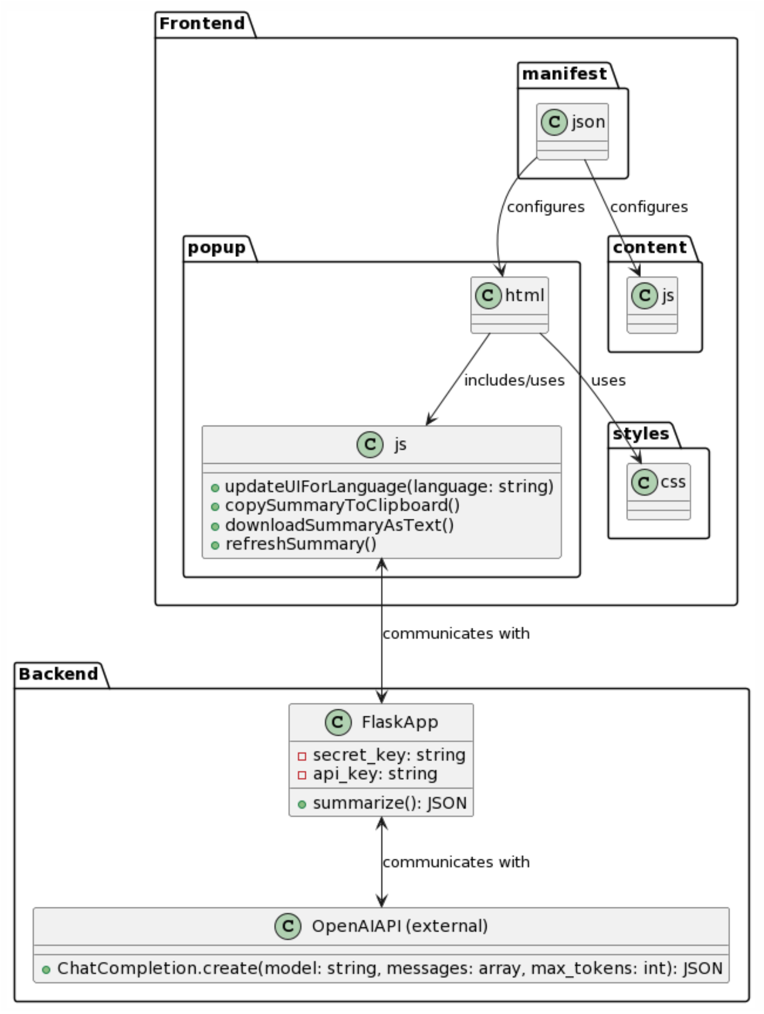
Sequence diagram: «Summarize text»



User
Extension
Backend
GPT35_API
Browsing a webpage
Clicks "Summarize" button
Sends webpage text
Request summary using GPT-3.5 API
Returns generated summary
Receives summary from API
Displays summary popup

The sequence diagram provides the most detailed explanation yet. It shows the dataflow through the system. Here you can easily see how the backend communicates with the API we integrated into our extension, and ultimately shows the user the summary inside the extension-popup.

Class diagram:



The class diagram gives an even better understanding of how all the front-end and back-end components are working together. It is quite self-explanatory, but to describe in words:
The file, app.py is where we do the connection to the OpenAI using a private OpenAi key. In the same

file, we also have the router with the function “summarize/”. This does the connection to the eventlisteners in the javascript (front-end) that is also connected to the html. (The extension popup)

When the “Summarize-button” is clicked, the JavaScript reads all the “innerhtml”- elements(the webpage text), sends it to the OpenAI cloud together with our prompt in app.py that tells it to summarize the text. The magic happens for us in OpenAi on their already trained AI-model GPT-3.5 Turbo, then sends back the summarized text and displays it in the html pop-up. We also have added some extra features that gives the user the opportunity to change language (for both translation and displayed text), copy the text to clipboard and download the text as a file.”

Challenges and Solutions:

API Token Limitation:

During the integration phase, we identified a token constraint in the GPT-3.5 Turbo model. If an article was too long, all the words on the page would not be handed, leading to a thrown exception. Changing to GPT4 helped, but this is a more expensive model, so we chose another solution for this problem.

We chose to truncate the input text to fit within the model's acceptable range, ensuring the essence of the content was retained. So basically, if the article was short enough to be summarized without being truncated, it just summarized as normal,

but if the article was too large, the input-text would be truncated, summarized and with a warning printed after: «This summary is based on a truncated text. Please upgrade to a premium subscription for complete text summaries. »

Another possible solution would be to scan and divide the entire page into fitting parts within the limit and summarize them bit by bit, but we have not implemented this, as we think the summarization time would be too long.

AI Assistance in Development:

In developing the core functionalities of our Chrome extension, we leaned on our existing expertise and supplemented it with AI tools for efficiency. Our backend integration with the OpenAI GPT-3.5 Turbo API was guided by the official tutorials, but we also watched a few youtube videos to fortify our knowledge. It turned out to be very easy, in fact, just a few lines of code for the core API call, though additional code was written to ensure robust error handling and to process the API responses effectively.

For front-end adjustments, particularly the aesthetics of the extension, we engaged with ChatGPT to expedite HTML and CSS modifications. We also needed some assistance with the implementation of the event listeners in popup.js. While the AI's input was instrumental in accelerating the development, the original ideas, functionality concepts, and the majority of implementation were independently crafted by our team. We wanted to build on our previous experience and combine it, so we all learned more together, and avoiding over-reliance on AI-generated code.

GitHub Copilot also played a role, primarily through its autocomplete feature. Moreover, when encountering bugs, we effectively used ChatGPT as a debugging assistant by querying it with error messages for quick resolutions.

In summary, we used AI strategically to enhance our productivity without compromising our learning objectives.

User Experience flaws:

It was a delight when we got the first summary to work. However, we noticed several things that needed improvement.

Through a lot of testing, we adjusted the design and added some new features to improve the user experience. For example, the summarization-process takes roughly 10-15 seconds (which is slightly slower than expected), so we added a blinking text to show the user that the process is being handled. Another idea was to simulate a real-time text writing similar to ChatGPT, but we had some performance issues regarding quality of the summary, which led us to discarding that idea.

We also added buttons that let you copy the text to the clipboard, download as a txt file, and refresh the summary. These functions were made in popup.js and popup.html.

At first, we tried to make a language detector that detects the language of the input-text and makes sure the output text is the same language. This worked. However, we figured that by giving the user the chance to select between English and Norwegian summaries himself would be an even better user experience.

Conclusion:

This was a very fun and interesting project that we are proud of. It's impressive that such a great asset as the GPT-3.5 Turbo API could relatively easily be integrated into our web extension app.

We have learned a lot with this project, and knowing how to integrate a Generative AI-Powered API into our applications really opens up for new opportunities in the future!

We are surfing the Ai-wave 🌊 🏄 🧑‍💻 🏖️ 😎

P.S for link to video-demonstration see here:
Video Demonstration

You may also see readme.md on the GitHub repository for how to clone and install this project on your own environment. Remember to use your own private API key from OpenAI.

GitHub repository for this project