**AI chat bot architecture 1**

1. I created “indexContent.js” script in the folder “scripts” that automatically index pages from my websites. In this script I specified the path to the pages that I want to be indexed. **TODO:** change localhost paths when you host the website.
2. The script generates the file” content.json” that contains the indexed content of my web pages This file provides context to the AI chatbot.
3. This content contains too many data that have to be cleaned out, so I did it by creating a python script clean\_json.py located also in the folder scripts as indexContent.js. This script deletes all the emojis, javascript references, file paths, metadata, special characters or unnecessary symbols. The final content that is used by Open AI is saved in cleaned\_content\_by\_script.json file
4. This file is read in chat.js file, when the content is loaded and used by Open AI when generating the response.

scripts/clean\_json.py **2.**

scripts/indexContent.js **1.**

Save the content under: content.json

Index the content from the website

Clean the content

Ç√

Save the content under: cleaned\_content\_by\_script.json

Ç√

ß

Pages/api/chat.js **3.**

Combine all the website content into a single string for OpenAI

Load the indexed content from cleaned\_content\_by\_script.json

Use OpenAI to generate a response

**Struggles:**

The billing was a bit difficult to understand. The first call in which I asked “Does Sofija have a PhD?” costed 10 cents, the next call was the question “From which University” and it costed additional 5 cents. Then the next questions costed between 1 and 5 cents. I paid 10$ credit and at some point OpenAI reserved additional 5$, I am not sure why. I asked copilot to explain this to me and here is the answer:

The billing behavior you described aligns with OpenAI's usage-based pricing model, but it can sometimes be confusing due to how charges are calculated and reserved. Here's an explanation of the key points:

**Why Did the First Call Cost 10 Cents?**

* The cost of an OpenAI API call depends on:
  1. **Model Used**: You are using gpt-4o, which is more expensive than gpt-3.5-turbo.
  2. **Tokens Used**: The cost is based on the number of tokens in the input (prompt) and output (response).
     + **Input Tokens**: The combined size of the [contentString](vscode-file://vscode-app/private/var/folders/vg/z7fzcqj51msdnkbxs59l4r600000gn/T/AppTranslocation/DDA6877F-5B7A-449B-86EF-17DCA81F3CB8/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") (from [cleaned\_content\_by\_script.json](vscode-file://vscode-app/private/var/folders/vg/z7fzcqj51msdnkbxs59l4r600000gn/T/AppTranslocation/DDA6877F-5B7A-449B-86EF-17DCA81F3CB8/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ")) and the question.
     + **Output Tokens**: The size of the generated response.
  3. If the [contentString](vscode-file://vscode-app/private/var/folders/vg/z7fzcqj51msdnkbxs59l4r600000gn/T/AppTranslocation/DDA6877F-5B7A-449B-86EF-17DCA81F3CB8/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") is large, the input token count will be high, leading to higher costs.

**Why Did the Next Call Cost 5 Cents?**

* The second question likely had fewer tokens in the input or output compared to the first call, resulting in a lower cost.
* OpenAI charges proportionally based on the number of tokens processed.

**Why Did Subsequent Questions Cost Between 1 and 5 Cents?**

* The cost varies depending on:
  + The size of the [contentString](vscode-file://vscode-app/private/var/folders/vg/z7fzcqj51msdnkbxs59l4r600000gn/T/AppTranslocation/DDA6877F-5B7A-449B-86EF-17DCA81F3CB8/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") (if it remains the same, this is constant).
  + The length of the question and the response.
  + Shorter questions and responses use fewer tokens, leading to lower costs.

**Why Did OpenAI Reserve an Additional $5?**

* OpenAI uses a **pre-authorization system** for billing:
  + When you make API calls, OpenAI may reserve additional funds (e.g., $5) to ensure you have sufficient credit for future usage.
  + This is not an additional charge but a temporary hold. The unused portion of the reserved amount will be released back to your account after a certain period.

**How to Monitor and Understand Billing?**

1. **Check Token Usage**:
   * OpenAI provides token usage details in the API response. You can log this information to understand how many tokens are being used per call:

console.log("Token usage:", response.usage);

* + Example output:

{

  "prompt\_tokens": 1000,

  "completion\_tokens": 150,

  "total\_tokens": 1150

}

1. **Review OpenAI's Pricing**:
   * Check the pricing for gpt-4o on OpenAI's [pricing page](vscode-file://vscode-app/private/var/folders/vg/z7fzcqj51msdnkbxs59l4r600000gn/T/AppTranslocation/DDA6877F-5B7A-449B-86EF-17DCA81F3CB8/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).
   * Example: If gpt-4o costs $0.03 per 1,000 tokens for input and $0.06 per 1,000 tokens for output, a call with 1,150 tokens would cost:
     + Input: 1,000 tokens × $0.03 = $0.03
     + Output: 150 tokens × $0.06 = $0.009
     + Total: $0.039 (rounded to 4 cents).
2. **Check Your OpenAI Account**:
   * Log in to your OpenAI account and review the **Usage** and **Billing** sections for detailed breakdowns of charges and reserved amounts.

**How to Reduce Costs?**

1. **Optimize the**[contentString](vscode-file://vscode-app/private/var/folders/vg/z7fzcqj51msdnkbxs59l4r600000gn/T/AppTranslocation/DDA6877F-5B7A-449B-86EF-17DCA81F3CB8/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o "):
   * If the [contentString](vscode-file://vscode-app/private/var/folders/vg/z7fzcqj51msdnkbxs59l4r600000gn/T/AppTranslocation/DDA6877F-5B7A-449B-86EF-17DCA81F3CB8/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") is large, consider summarizing or reducing its size before sending it to OpenAI.
   * For example, you could limit the number of entries or truncate long sections.
2. **Use a Cheaper Model**:
   * If gpt-4o is not strictly necessary, consider using gpt-3.5-turbo, which is significantly cheaper.
3. **Set a Token Limit**:
   * Use the [max\_tokens](vscode-file://vscode-app/private/var/folders/vg/z7fzcqj51msdnkbxs59l4r600000gn/T/AppTranslocation/DDA6877F-5B7A-449B-86EF-17DCA81F3CB8/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") parameter to control the length of the response and reduce output token costs:

max\_tokens: 100, // Limit the response to 100 tokens

**Summary**

* The first call cost more because of the larger input and output token count.
* Subsequent calls cost less due to shorter inputs or outputs.
* The $5 reservation is a temporary hold by OpenAI to ensure sufficient credit.
* To better understand and control costs, monitor token usage and consider optimizing the input content or switching to a cheaper model.

**Limitations of the Current Implementation**

1. **Inefficiency:**
   * The entire website content is sent with every question, which increases token usage and costs.
2. **Scalability:**
   * If the website content grows, the [contentString](vscode-file://vscode-app/private/var/folders/vg/z7fzcqj51msdnkbxs59l4r600000gn/T/AppTranslocation/DDA6877F-5B7A-449B-86EF-17DCA81F3CB8/d/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") may exceed OpenAI's token limit.
3. **Performance:**
   * Sending large amounts of data with every request can slow down the response time.

**Suggestions for Improvement**

1. **Use Embeddings and a Vector Database:**
   * Preprocess the website content into embeddings and store them in a vector database (e.g., Pinecone, Weaviate).
   * Query the database for relevant content based on the user's question and send only the relevant content to OpenAI.
2. **Summarize Content:**
   * Summarize the website content into a shorter version to reduce token usage.
3. **Cache Responses:**
   * Cache frequently asked questions and their responses to avoid redundant API calls.

Scripts to run:

node indexContent.cjs – to index the content of the website and store it in scripts/content.json

python3 /Users/sofka/Documents/sofka-website/scripts/clean\_json.py – clean the content from scripts/content.json and store the cleaned content into scripts/cleaned\_content\_by\_script.json

generateEmbeddings.js – add new content to the quadrant collection (vector database). If the old content is visible then delete the quadrant collection by using the script deleteCollection.js and recreate it by using the script createQuadrantCollection.js