**AI DOCUMENTATION FOR TASK 2A AND 4A**

I used AI primarily for guiding me in steps where I got stuck and with learning new techniques where my previous knowledge was not enough.   
One of the main thing for which I used the help of AI was the README file. I already knew few things about markdown structing (as we used it in the last project) but I wanted to know what I could do more. Although we needed the basic things for the readme file, out of curiosity I asked AI what components a Markdown formatted readme file can have. It gave me good ideas on the insertion of images and use of dashes (---)  
Below is the full conversation with it.

**You said**:  
Please guide me on creating a README.md file that is done in markdown formating. I already know a few things like the concept of #, ##, ### (for folders and subfolders), the use of \*,\*\*. and the use of - for listing. What other things are there in a such file?

2/2

**ChatGPT said:**

Updated saved memory

In addition to the basics you already know, here are some other useful elements and syntax for your README.md file in Markdown formatting:

**1. Code Blocks**

* For inline code, use backticks (`):

markdown

CopyEdit

`print("Hello World")`

* For multi-line code blocks, use triple backticks (```) or specify the language for syntax highlighting:

markdown

CopyEdit

```python

def greet():

print("Hello World")

CopyEdit

**2. Links**

* To create a hyperlink, use:

markdown

CopyEdit

[Link Text](http://example.com)

**3. Images**

* To embed an image, use:

markdown

CopyEdit

![Alt Text](http://example.com/image.jpg)

**4. Tables**

* You can create tables using pipes (|):

markdown

CopyEdit

| Header 1 | Header 2 | Header 3 |

|----------|----------|----------|

| Row 1, Column 1 | Row 1, Column 2 | Row 1, Column 3 |

| Row 2, Column 1 | Row 2, Column 2 | Row 2, Column 3 |

**5. Blockquotes**

* To add blockquotes, use the > symbol:

markdown

CopyEdit

> This is a blockquote

**6. Task Lists**

* To create a task list (checklist), use - [ ] for unmarked and - [x] for marked tasks:

markdown

CopyEdit

- [x] Task 1

- [ ] Task 2

- [x] Task 3

**7. Horizontal Line**

* To create a horizontal line (separator), use three hyphens (---), asterisks (\*\*\*), or underscores (\_\_\_):

markdown

CopyEdit

---

**8. Strikethrough**

* To apply strikethrough text, use ~~:

markdown

CopyEdit

~~This is a strikethrough~~

**9. Escaping Characters**

* If you need to display special characters like #, \*, or backticks, you can escape them with a backslash (\):

markdown

CopyEdit

\# This is a hash symbol

**10. Header Links**

* You can create links to specific sections in the document by using headers:

markdown

CopyEdit

[Link to section](#section-name)

Headers are auto-generated in a specific format for links, replacing spaces with hyphens and making everything lowercase.

These elements can help make your README file more informative and easier to navigate. Let me know if you'd like to explore any of these in more detail!

4o mini

**You said:**

### 3\. \*\*Images\*\* \* To embed an image, use: markdown CopyEdit `![Alt Text](http://example.com/image.jpg)`

Can this work on files present in my github repo?

**ChatGPT said:**

Yes, you can definitely use images from your GitHub repository in your README.md file! To do this, you need to reference the image using a relative path (if the image is in your repository) or the direct URL if it's hosted elsewhere.

**1. Using a Relative Path**

* If the image is located in your repository (for example, in an images/ folder), you can reference it like this:

markdown

CopyEdit

![Alt Text](images/my-image.jpg)

* This will display the image from your repository.

**2. Using the Raw URL from GitHub**

* If you want to use the URL from GitHub directly (in case the image is stored in the repository), go to the image file on GitHub, click on it, and then click "Download" or "Open in new tab" to get the raw URL.
* Once you have the raw URL, use it like this:

markdown

CopyEdit

![Alt Text](https://raw.githubusercontent.com/username/repository/branch/images/my-image.jpg)

Both approaches will allow you to embed images hosted within your GitHub repository or from external sources! Let me know if you need help with a specific example.

4o mini

**You said:**

why the [alt text] ? is it just there for when the image is loading or has failed loading, or does it serve any other purpose?

**ChatGPT said:**

Yes, you're right! The [Alt Text] serves multiple purposes:

**1. Fallback Text (for when the image doesn't load)**

* If the image fails to load for any reason (e.g., broken link, slow connection, or file not found), the alt text will be displayed in place of the image. This is important for ensuring your content remains accessible even if the image can't be shown.

**2. Accessibility**

* Alt text is crucial for accessibility, especially for users who rely on screen readers (e.g., visually impaired users). Screen readers will read out the alt text, allowing those users to understand the content of the image.

**3. Search Engine Optimization (SEO)**

* Alt text can improve the searchability of your images on the web. Search engines like Google use alt text to better understand the content of an image, which can help the image show up in relevant searches.

**4. Description of the Image**

* Alt text provides a brief description of the image's content or purpose, which is useful if the image is not immediately clear or if someone is scrolling without looking at the image itself.

Here’s an example:

markdown

CopyEdit

![A scenic view of the mountains in Gilgit-Baltistan](images/mountains.jpg)

In this case, if the image doesn't load or if it's being read by a screen reader, the alt text "A scenic view of the mountains in Gilgit-Baltistan" provides meaningful context to the viewer.

Alt text is a good practice for making your content more inclusive and SEO-friendly!

**2A. EXTRACTION OF PLACES USING REGEX**

For this part, AI was not very helpful. The AI was consulted to correct a bug in the final part of the code where a DataFrame was being incorrectly constructed using rows instead of the actual frequency data (data\_for\_df)

I want to save the output to a file named regex\_counts.tsv that has columns: placename, month-year, and count. I already created data\_for\_df and the TSV header, but I’m not sure if this part is working correctly. Here's my code.

df = pd.DataFrame(rows, columns=["placename", "month", "count"])

df.to\_csv("regex\_counts.tsv", sep="\t", index=False)

But it doesn’t look right.

**🤖 ChatGPT:**  
You're close, but there's a small issue. The line:

python

CopyEdit

df = pd.DataFrame(rows, columns=["placename", "month", "count"])

is incorrect because rows is just your gazetteer data — it doesn’t contain the mention counts. You should be using data\_for\_df, which holds the actual results.

Update that section to:

python

CopyEdit

df = pd.DataFrame(data\_for\_df, columns=["placename", "month", "count"])

df.to\_csv("regex\_counts.tsv", sep="\t", index=False)

This will correctly write the placename, month, and count values to the TSV file.

**4A: CREATING A MAP WITH PLOTLY**

For this part, I used ai assistant Gemini (which is directly available from google search bar). There was no direct question from it, there were series of small queries I did about map ideas, things I should include in my map, and how best to present my data. I did not ask for any particular code apart from dataframe structure, which was also done through google search (answered by Gemini).