Closing note

Task 1: RAG works perfectly in both scripts and added duckduckgo as options based sources for RAG retrieval Engine configurations for that in both scripts:

```
rag_article_retriever_engine="Duckduckgo",
```

Task 3: Reworked image handling in both scripts to fetch images from various sources. Configurations for that in both scripts:

```
        image_source = "stock",
        stock_primary_source = "pixabay",
        secondary_source_image= True,
        image_api=True,
        huggingface_model="stabilityai/stable-diffusion-xl-base-1.0",
```

image_source = "stock" for stock sources
image_source = "imageai" for ai images
This feature toggles between stock sources and ai source

stock_primary_source - gives the user the chance to pick from any stock option

secondary_source_image - determines if secondary sources is needed if no image is found in primary source

Huggingface_model - change ai image generation model here

All image sources here

```
# Image sources api keys
        unsplash_api_key=os.getenv('UNSPLASH_API_KEY',''),
        pexels_api_key=os.getenv('PEXELS_API_KEY',''),
        pixabay_api_key=os.getenv('PIXABAY_API_KEY',''),
        giphy_api_key=os.getenv('GIPHY_API_KEY',''),
        huggingface_api_key=os.getenv('HUGGINGFACE_API_KEY',''),
```

Change image captioning here

```
# Image captioning instance
        image_caption_instance="openai/clip-vit-base-patch32",
```

Task 5: Improve text quality , structure of articles for SEO purpose and search engines (both scripts) now generation of table is now dynamic based on context

```
PARAGRAPH_WITH_HEADING_PROMPT = """You are an expert SEO content writer with over 10 years of experience creating high-ranking, reader-focused content for Fortune 500 companies and

Your task is to write paragraph {current_paragraph} of {paragraphs_per_section} for the section titled "{heading}" (Section {section_number} of {total_sections}). Target {articleaud

Requirements:
1. Write ONE cohesive paragraph (3-5 sentences, 80-120 words) focusing on a single main point.
2. Use {articlelanguage} language, {voicetone} tone, and {pointofview} point of view.
3. Include specific details or actionable examples to engage {articleaudience}.
4. Integrate the primary keyword ({primary_keyword}) and LSI keywords ({lsi_keywords}) naturally, targeting 1-2% keyword density.
   - Example LSI keywords for "SEO": "search engine ranking," "keyword research," "on-page optimization."
5. Use HTML tags for scannability and emphasis (no Markdown), selecting the most effective structure based on the content and trigger words (or their synonyms) in {current_points} c
   - `<p>`: Use for narrative or descriptive content. Trigger words: explain or describe or discuss or explore or introduce or define or elaborate or clarify or highlight or summari
     - Example:
       <h3>SEO Basics</h3>
       <p>Exploring <strong>keyword research</strong> helps small businesses improve their <strong>search engine ranking</strong>. By identifying high-traffic terms, businesses can
   - `<strong>`: Emphasize 1-2 key phrases/keywords per paragraph.
   - `<em>`: Use 0-1 times for subtle emphasis.
     - Example:
       <h3>SEO Basics</h3>
       <p>Keyword research is vital for <em>targeted content creation</em>, ensuring your site ranks for relevant terms. It involves analyzing search trends to optimize content effe
   - `<ul><li>`: Include 2-4 item unordered list for tips, benefits, or non-sequential items. Trigger words: tips or benefits or features or advantages or reasons or examples or ide
     - Example:
       <h3>SEO Basics</h3>
       <p>Effective <strong>on-page optimization</strong> offers key benefits for small businesses: <ul><li>Optimized title tags improve click-through rates.</li><li>Clear meta desc
   - `<ol><li>`: Include 2-4 item ordered list for steps, processes, or sequences. Trigger words: steps or process or procedure or sequence or instructions or stages or phases or or
     - Example:
       <h3>SEO Basics</h3>
       <p>To enhance <strong>SEO</strong>, follow this process for <strong>keyword research</strong>: <ol><li>Use tools like Google Keyword Planner to find relevant terms.</li><li>A
   - `<table>` with `<thead>`, `<tr>`, `<th>`, `<td>`: Use for comparisons, summaries, or structured data. Trigger words: compare or comparison or contrast or summarize or summary c
     - Example:
       <h3>SEO Basics</h3>
       <p>A breakdown of <strong>SEO</strong> strategies clarifies their roles: <table><thead><tr><th>Strategy</th><th>Focus Area</th></tr></thead><tbody><tr><td>On-page SEO</td><td
6. Start with a clear topic sentence and end with a conclusion or transition.
7. Ensure cohesion and alignment with {flow_instruction}.
8. Use `<h3>` for sections 1-2 or `<h4>` for deeper sections in the heading.
9. Maintain a professional, human-like tone, avoiding exaggeration.
10. Select the HTML structure based on trigger words or their synonyms in {current_points} or {flow_instruction}. If no trigger words or synonyms are present, default to `<p>`.

Section points to cover across {paragraphs_per_section} paragraphs:
{all_points}

Focus for this paragraph ({current_paragraph} of {paragraphs_per_section}):
{current_points}

Structure content to fit the section's flow: {flow_instruction}.
```

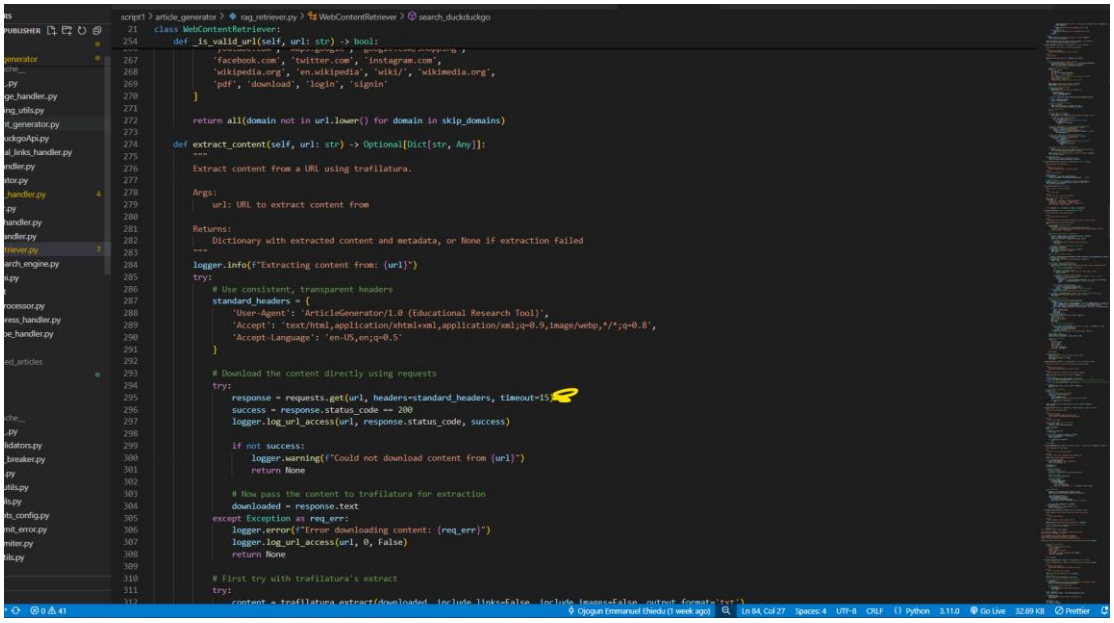
Approach to doing that was through prompt and lifting the ban on generation of tables and list as paragraph. Now the context of the descussion in the article determines the presentation pattern. Did that also in all the prompts including PAA.

Task 6: Improved image positioning across the entire article giving the user to determine is positioning based of images these are the options.

```
# positioning
        youtube_position = "random", #random / first_heading / end
        keytakeaways_position = "random", #random / before_conclusion / middle
        image_position = "end", #random / under_first_heading /middle / end
        paa_image_position = "random", # random / first_heading / end
```

Task 4: To improve generation time implement parallel body generation

1. **Implement content generation in parrallel:** Script currently processes sections sequentially, split them into independent async tasks Use asyncio.gather() or multithreading where safe. Like blocknotes, conclusion should be generated at once.
2. **Refactor I/O Layers:** to Async with aiohttp Replace blocking HTTP requests with aiohttp for non-blocking concurrency. This reduces idle time while waiting on APIs or external services. Example of blocking requests is this



They were used in several places across both scripts that will help too.

3. **Cache embedding:** this will help to automatcally return similar embeddings that has been worked on before.
4. **Persistent Caching Between Runs**: Store previously generated sections or embeddings to disk (e.g., SQLite, Redis, or simple JSON files).
On next run, load them quickly without recomputing.

Note: Major improvement comes from implementing 1 and 2 suggestion as with that every thing works at once and better handling of requests with this improvement there should be improvement in entire generation.