Hackathon Mission 1: The Empathetic Code Reviewer

Tagline: *Transforming Critical Feedback into Constructive Growth.*

The Premise

In software development, code reviews are the lifeblood of a healthy team. They catch bugs, improve quality, and spread knowledge. However, they are often a source of friction and a truly mundane task. Written feedback can feel blunt, impersonal, and at worst, discouraging. A junior developer might feel attacked by a comment like "This is inefficient," while the senior developer who wrote it was simply being direct. This communication gap slows down learning and can harm team morale. Your mission is to build an AI that acts as a bridge, translating raw critique into supportive, educational guidance, thereby freeing developers from the negative cycle of harsh feedback.

Your Mission

Create a program that takes a snippet of code and a list of direct, critical review comments, and then uses Generative AI to rewrite them. The AI should act as an ideal senior developer or a patient mentor, rephrasing the feedback to be empathetic, constructive, and educational, ensuring the recipient understands the "why" behind the suggestion, not just the "what."

Technical Specifications

- **Input:** Your program will process a JSON object with two keys: code_snippet and review_comments.

```
{
  "code_snippet": "def get_active_users(users):\n  results = []\n  for u in users:\n
if u.is_active == True and u.profile_complete == True:\n      results.append(u)\n
return results",
  "review_comments": [
    "This is inefficient. Don't loop twice conceptually.",
    "Variable 'u' is a bad name.",
    "Boolean comparison '== True' is redundant."
  ]
}
```

- **Required Output:** Your program must produce a single, well-formatted Markdown report. For **each** original comment, you must generate a section that includes:

1. **Positive Rephrasing:** A gentle and encouraging version of the feedback.
2. **The 'Why':** A clear, concise explanation of the underlying software principle (e.g., performance, readability, convention).
3. **Suggested Improvement:** A concrete code example demonstrating the recommended fix.

~~~~
Example structure for one part of the output:---
### Analysis of Comment: "This is inefficient. Don't loop twice conceptually."

* **Positive Rephrasing:** "Great start on the logic here! For better
performance, especially with large user lists, we can make this more efficient
by combining the checks."
* **The 'Why':** Iterating through a list and performing checks can become slow
as the list grows. By using more direct methods like list comprehensions, we
can often achieve the same result with cleaner and faster code.
* **Suggested Improvement:**
    ```python
    def get_active_users(users):
      return [user for user in users if user.is_active and
user.profile_complete]
    ```
---
~~~~

Tips for Standing Out

- **Contextual Awareness:** Can your AI's tone change based on the severity of the original feedback?
- **Link to Resources:** Can your AI provide a link to external documentation (e.g., a relevant part of a style guide like PEP 8 for Python, or an article on algorithmic complexity) to support its suggestion?
- **Holistic Summary:** Can you add a concluding paragraph that summarizes the overall feedback in an encouraging way?

Key Skills Tested

- Nuanced Prompt Engineering
- Empathetic AI Design
- Code Comprehension and Explanation

Hackathon Mission 2: The "Digital Skeptic" AI

Tagline: *Empowering Critical Thinking in an Age of Information Overload.*

The Premise

We are constantly exposed to a firehose of information online. The mundane, yet essential, task of vetting every article for bias, spin, or logical fallacies is mentally exhausting. As a result, it's easy to fall for misinformation or accept a one-sided narrative. The goal isn't to have an AI tell us what is "true" or "false," but to build an AI assistant that acts as a critical thinking partner. It can automate the initial analysis, freeing up our mental energy to make our own informed judgments.

Your Mission

Create a program that accepts the URL of an online news article. Your tool must fetch the article's content and use Generative AI to perform a "skeptical analysis." It should not make a final judgment but should instead highlight claims, analyze the language, and arm the reader with the right questions to ask.

Technical Specifications

- **Input:** Your program will take a single URL to a news article as a string input.
  - *Note on Implementation:* Your first step is to programmatically fetch the text content from the URL (e.g., using libraries like requests and BeautifulSoup in Python). If you encounter blocking or other web scraping issues, it is acceptable to save the article's text to a local file and read from that. Please make a note of your approach in your README.md file.
- **Required Output:** Your program must produce a single "Critical Analysis Report" in Markdown format. This report must contain the following distinct sections:
  1. **Core Claims:** A bulleted list summarizing the 3-5 main factual claims the article makes.
  2. **Language & Tone Analysis:** A brief analysis and classification of the article's language (e.g., "Appears neutral and factual," "Uses emotionally charged and persuasive language," "Reads as a strong opinion piece").
  3. **Potential Red Flags:** A bulleted list identifying any detected signs of bias or poor reporting, such as the use of loaded terminology, an over-reliance on anonymous sources, a lack of cited data, or failing to present opposing viewpoints.
  4. **Verification Questions:** A list of 3-4 insightful, specific questions a reader should ask to independently verify the article's content.

```
Example Output Structure:# Critical Analysis Report for: [Article Title]

### Core Claims
* Claim 1...
* Claim 2...
* Claim 3...

### Language & Tone Analysis
The language in this article is highly persuasive and uses emotionally charged
words like 'disastrous' and 'unprecedented' to frame the narrative.

### Potential Red Flags
* The article heavily relies on a single anonymous 'insider' for its most
significant claims.
* Statistical data is mentioned but no link to the source study is provided.
* Alternative explanations for the event are mentioned but immediately
dismissed without exploration.

### Verification Questions
1.  Can I find other independent reports from reputable sources that
corroborate the claims made by the 'insider'?
2.  Who funded the organization that published this article, and do they have a
known political or commercial agenda?
3.  What do experts with opposing views say about this topic?
```

Tips for Standing Out

- **Entity Recognition:** Can your AI identify the key people, organizations, and locations mentioned and suggest what a reader should investigate about them (e.g., "Investigate the author's previous work," "Look into the funding of 'The XYZ Institute'")?
- **Counter-Argument Simulation:** Can you prompt the AI to briefly summarize the article from a hypothetical "opposing viewpoint" to starkly highlight its potential biases?

Key Skills Tested

- Analytical Prompt Engineering
- Information Extraction & Synthesis
- Web Content Handling

Hackathon Evaluation Criteria & Scoring Rubric

Theme: "Freedom from Mundane: AI for a Smarter Life"

Guiding Principle

The primary goal of this evaluation is to identify candidates who demonstrate a strong aptitude for applying Generative AI to solve complex, real-world problems. We are looking for more than just coders; we are looking for creative problem-solvers who can effectively command AI models. The scoring is weighted to reflect this, prioritizing the **quality and nuance of the AI-generated output** over pure coding proficiency.

Scoring Breakdown

Each submission will be scored out of a total of **100 points**, broken down into four key categories:

Category	**Weighting**	**Description**
1. Functionality & Correctness	**25%**	Does the program run as expected and meet the core technical requirements of the problem statement?
2. Quality of AI Output & Prompt Engineering	**45%**	How insightful, detailed, and context-aware is the AI's output? This is the primary measure of skill.
3. Code Quality & Documentation	**20%**	Is the underlying code well-structured, readable, and clearly explained?
4. Innovation & "Stand Out" Features	**10%**	Did the participant go beyond the base requirements to add creative or exceptionally valuable features?

Detailed Scoring Rubric

1. Functionality & Correctness (25 Points)

This category assesses the fundamental execution of the project.

- **Excellent (21-25 pts):** The program runs flawlessly on the first try with the provided examples. It correctly handles all specified inputs and produces output in the exact format required (Markdown, JSON, etc.). No manual intervention is needed.
- **Good (16-20 pts):** The program runs but may have minor, easily fixable issues (e.g., a wrong file path). The output format is mostly correct but may have small deviations. The core logic is sound.
- **Satisfactory (10-15 pts):** The program runs after some debugging. It processes the input but the output format is incorrect or incomplete. Key parts of the required output are missing.
- **Needs Improvement (0-9 pts):** The program fails to run, crashes, or does not produce any meaningful output. The core requirements are not met.

2. Quality of AI Output & Prompt Engineering (45 Points)

This is the most critical category. It evaluates the participant's ability to "instruct" the AI.

- **Excellent (38-45 pts):** The output is exceptionally insightful, nuanced, and feels human-curated. The AI's response demonstrates a deep understanding of the context and goes beyond generic answers. The prompt engineering is clearly sophisticated.
  - **Mission 1 (Code Reviewer):** The feedback is genuinely empathetic, the technical explanations are clear and correct, and the suggested code is a significant improvement.
  - **Mission 2 (Digital Skeptic):** The analysis pinpoints subtle biases, the "red flags" are insightful, and the verification questions are sharp and actionable.
  - **Mission 3 (Fashion Critic):** The critique is specific to the visual details in the image. The analysis of strengths and weaknesses is balanced, expert-level, and highly constructive.
- **Good (30-37 pts):** The output meets all requirements of the problem statement and is accurate. The analysis is good but may lack the deep insight or "wow factor" of the top tier. The prompt was effective but straightforward.

    - **Mission 1 (Code Reviewer):** The feedback is positive but might feel slightly generic. The suggestions are correct but may not be the most elegant solution.
    - **Mission 2 (Digital Skeptic):** The analysis correctly identifies obvious points but may miss more subtle cues in the text.
    - **Mission 3 (Fashion Critic):** The critique correctly identifies the clothing items but the advice might be generic fashion advice not perfectly tailored to the image.
- **Satisfactory (20-29 pts):** The output is present but superficial. It may feel like a simple "pass-through" of the prompt without much refinement. Key sections might be weak or miss the point of the analysis.
    - **Mission 1 (Code Reviewer):** The rephrasing is just a slightly nicer version of the original; the "why" is missing or incorrect.
    - **Mission 2 (Digital Skeptic):** The claims are just extracted sentences; the red flags are generic (e.g., "the article could be biased").
    - **Mission 3 (Fashion Critic):** The output is a generic description of the clothes rather than a critique (e.g., "The person is wearing a blazer and pants.").
- **Needs Improvement (0-19 pts):** The AI output is irrelevant, nonsensical, factually incorrect, or completely fails to address the prompt's requirements.

3. Code Quality & Documentation (20 Points)

This category assesses standard software engineering best practices.

- **Excellent (17-20 pts):** The code is clean, well-commented, and easy to understand. Variables and functions have clear, logical names. The `README.md` is professional, providing a clear explanation of the approach and flawless instructions to run the code.
- **Good (13-16 pts):** The code is functional and readable but could be better organized. Comments may be sparse. The `README.md` has instructions, but they might be slightly unclear or miss a step.
- **Satisfactory (8-12 pts):** The code is messy, hard to follow, and lacks comments. The `README.md` is missing or provides minimal, unhelpful information.
- **Needs Improvement (0-7 pts):** The code is exceptionally difficult to read. No documentation is provided.

4. Innovation & "Stand Out" Features (10 Points)

This category rewards participants who went above and beyond the core requirements.

- **Excellent (8-10 pts):** The participant successfully implemented one or more of the suggested "Stand Out" features in a meaningful way. The solution shows exceptional creativity or a clever technical approach that adds significant value.
- **Good (5-7 pts):** An attempt was made to implement a bonus feature, but it may be partially functional or not fully integrated into the final output.
- **Satisfactory (1-4 pts):** The participant met the base requirements only. No attempt was made at bonus features.
- **Needs Improvement (0 pts):** Not applicable if the core requirements were not met.