Comprehensive Prompt Engineering Guide

Table of Contents

General Tips for Effective Prompting

1. Be Clear and Specific

- Clearly state your task or question at the beginning of your message.
- Provide context and details to help the AI understand your needs.
- Break complex tasks into smaller, manageable steps.

Example:

- Bad: "Help me with a presentation."
- Good: "I need help creating a 10-slide presentation for our quarterly sales meeting. The presentation should cover our Q2 sales performance, top-selling products, and sales targets for Q3. Please provide an outline with key points for each slide."

2. Use Examples

- Provide examples of the kind of output you're looking for.
- If you want a specific format or style, show the AI an example.

3. Encourage Thinking

- For complex tasks, ask the AI to "think step-by-step" or "explain your reasoning."
- This can lead to more accurate and detailed responses.

4. Iterative Refinement

- If the AI's first response isn't quite right, ask for clarifications or modifications.
- You can always say "That's close, but can you adjust X to be more like Y?"

5. Leverage the AI's Knowledge

- AI models often have broad knowledge across many fields. Don't hesitate to ask for explanations or background information.
- Include relevant context and details so that the AI's response is maximally targeted to be helpful.

6. Use Role-Playing

- Ask the AI to adopt a specific role or perspective when responding.
- This can help in getting more nuanced or specialized responses.

Task-Specific Tips

Content Creation

1. Specify your audience.
2. Define the tone and style.
3. Define output structure.

Document Summary and Q&A

1. Be specific about what you want.
2. Use document names.
3. Ask for citations.

Data Analysis and Visualization

1. Specify the desired format.
2. Ask for specific metrics or trends.

Brainstorming

1. Use the AI to generate ideas by asking for a list of possibilities or alternatives.
2. Request responses in specific formats like bullet points, numbered lists, or tables for easier reading.

Troubleshooting and Maximizing Performance

1. Allow the AI to acknowledge uncertainty.
2. Break down complex tasks.
3. Include all contextual information for new requests.

Custom Prompts Analysis

GitLab (Unit Tests) Prompt

Here's the full prompt created for generating unit tests:

```
You are an expert at generating test cases based on the provided code or stored
data. Your goal is to create a minimum of 25 diverse test cases covering all the
code and its various functions. Ensure the test cases address multiple objectives,
including functionality, edge cases, error handling, performance, security, and
different user scenarios.

Instructions:
-------------
Use this specific format:
- Test Case: [Action to be performed and specific details, e.g., "Click on the
login button and enter a password of length 9 and a valid email"]
- Expected Result: [Outcome of the action, e.g., "Authentication succeeded"]

Guidelines:
1. Generate diverse test cases covering all functions and components.
2. Ensure no more than 3 test cases focus on the same function or component.
3. Address different test objectives such as functionality, edge cases, error
handling, performance, security, and user scenarios.
4. Ensure the test cases are specific, clear, and concise.
5. Ensure no repetitive test cases and cover different aspects of functionality.

Ensure the output is in plain text without any formatting or styles to be suitable
for CSV input. Do not include any additional notes or comments outside of the
specified format.

Examples:
---------
- Test Case: Click on the login button and enter a password of length 9 and a
valid email
- Expected Result: Authentication succeeded
- Test Case: Enter a PDF document to process
- Expected Result: PDF entered successfully
- Test Case: Modify personal data
- Expected Result: Personal data successfully modified
- Test Case: Submit an empty form
- Expected Result: Error message indicating required fields are missing
- Test Case: Upload a file larger than the maximum allowed size
```

```
- Expected Result: Error message indicating the file is too large
- Test Case: Access a restricted page without logging in
- Expected Result: Redirect to login page with an access denied message
- Test Case: Perform a database query for a non-existent record
- Expected Result: No records found message
- Test Case: Simulate multiple concurrent users logging in
- Expected Result: System handles concurrent logins without performance
degradation
- Test Case: Enter an invalid email format during registration
- Expected Result: Error message indicating invalid email format

Generate a variety of test cases for the provided code, ensuring balanced coverage
across all functions. Provide specific details in your test cases without adding
any unnecessary text or notes. Ensure each test case is unique and well-detailed.
```

Analysis of the GitLab Prompt:

1. Role and Objective Setting:
    - Defined the AI's role as an "expert at generating test cases" and set a clear objective to create "a minimum of 25 diverse test cases."
    - This immediately sets the context and expectations for the AI's response. It ensures that the AI understands its role and the scope of the task.
2. Comprehensive Coverage:
    - Specified that the test cases should cover "multiple objectives, including functionality, edge cases, error handling, performance, security, and different user scenarios."
    - This ensures that the generated test cases will be diverse and cover all important aspects of software testing, not just basic functionality.
3. Specific Format:
    - Provided a clear format for each test case, including "Test Case" and "Expected Result."
    - This standardizes the output, making it easy to read and process. It also ensures that each test case includes both the action and the expected outcome.
4. Detailed Guidelines:
    - Provided a list of specific guidelines for generating the test cases.
    - These guidelines further refine the AI's approach, ensuring diversity (no more than 3 cases per function), specificity, and non-repetitiveness in the generated test cases.
5. Output Specifications:
    - Specified that the output should be in plain text, suitable for CSV input, without additional formatting or comments.
6. Examples:
    - Provided several examples of test cases in the desired format.

Jira (Acceptance Criteria) Prompt

Here's the full prompt created for generating acceptance criteria:

```
You are a professional agile scrum master highly skilled in crafting precise
Acceptance Criteria based solely on the descriptions provided in the user stories.
If a user story lacks a description, do not include any related acceptance
criteria for that particular story.

Please adhere to the following format:
- Issue ID:
- Acceptance Criteria:

Please follow these guidelines:
1. Develop a comprehensive set of acceptance criteria that covers all user
stories.
2. Ensure that the acceptance criteria are clear, specific, and concise.
3. If a user story does not have a description, indicate "none" as the acceptance
criteria.
4. For each issue id, I expect you to provide me with 1 acceptance criteria.

Example: (you must follow like this format)
- Issue ID: LR-1
- Acceptance Criteria: The user can upload a profile picture.
- Issue ID: LR-1
- Acceptance Criteria: The system prompts the user to crop the image if it exceeds
the specified dimensions.
- Issue ID: LR-2
- Acceptance Criteria: The user can search for products by category.
- Issue ID: LR-2
- Acceptance Criteria: The search results display relevant products based on the
selected category.
- Issue ID: LR-3
- Acceptance Criteria: none
```

Analysis of the Jira Prompt:

1. Role Definition:
    - Defined the AI's role as a "professional agile scrum master highly skilled in crafting precise Acceptance Criteria."
    - This sets the context and level of expertise expected in the response. It primes the AI to think in terms of agile methodologies and best practices for acceptance criteria.

2. Clear Instructions:
    - Provided clear instructions on how to handle user stories with and without descriptions.
    - This prevents the AI from making assumptions about missing information and ensures consistent handling of all user stories.
3. Specific Format:
    - Defined a specific format for the output, including "Issue ID" and "Acceptance Criteria."
    - This standardizes the output, making it easy to read and integrate into Jira or other project management tools.
4. Concise Guidelines:
    - Provided a list of specific guidelines for developing the acceptance criteria.
    - These guidelines ensure that the generated criteria are comprehensive, clear, specific, and concise. They also provide clear instructions on how to handle edge cases (like stories without descriptions).
5. Quantity Specification:
    - Specified that you expect one acceptance criterion per issue ID.
    - This sets clear expectations for the amount of output and prevents the AI from generating an excessive number of criteria per story.
6. Examples:
    - Provided several examples of acceptance criteria in the desired format.
    - Examples clarify expectations and help the AI understand the level of detail and specificity required in the acceptance criteria.
7. Flexibility:
    - Included an example of how to handle a user story without a description.
    - This demonstrates how to handle exceptions or edge cases, ensuring consistency even when input data is incomplete.

Overall, these prompts are highly effective because they provide clear instructions, set specific expectations, and give the AI enough guidance to generate useful acceptance criteria while maintaining flexibility for different scenarios. The combination of role-setting, specific format, clear guidelines, and examples makes this prompt particularly strong for generating consistent and useful acceptance criteria.

Conclusion

These custom prompts demonstrate several key principles of effective prompt engineering:

1. **Clear Role Definition:** Both prompts start by defining a specific role for the AI, which helps set the context and level of expertise expected in the response.
2. **Specific Objectives:** Each prompt clearly states what is expected from the AI, including the number and type of outputs.
3. **Structured Format:** Both prompts provide a specific format for the output, which ensures consistency and makes the results easier to use.
4. **Comprehensive Guidelines:** The prompts include detailed guidelines that cover various aspects of the task, including how to handle edge cases.
5. **Examples:** Both prompts include examples that clarify expectations and demonstrate the desired output format.
6. **Specificity:** The prompts are highly specific about what they want, leaving little room for misinterpretation.
7. **Flexibility:** While being specific, the prompts also allow for flexibility where needed, such as handling user stories without descriptions.

By incorporating these elements, you can create prompts that are likely to produce consistent, high-quality outputs that meet your specific needs. These prompts serve as excellent examples of how to craft effective instructions for AI models in various contexts.

When creating your own prompts, consider incorporating these elements:

- Define the AI's role clearly.
- State your objectives explicitly.
- Provide a structured format for the output.
- Include comprehensive guidelines.
- Offer examples of the desired output.
- Be specific about your requirements.
- Allow for flexibility where necessary.

By following these principles, you can create prompts that effectively harness the capabilities of AI models to produce the results you need.