```
# Prompt 1: Base Zero-shot Prompting


BASE_PROMPT = """
Based on the following classified user feedback, generate a comprehensive Software
Requirements Specification (SRS) document.


Classified Feedback:
{classified_feedback}


Generate a well-structured SRS document that includes the following sections, using
clear and concise language:
1. Introduction and Purpose
2. System Overview
3. Functional Requirements
4. Non-Functional Requirements
5. System Features
6. External Interfaces
7. Performance Requirements
8. Design Constraints


Ensure the generated SRS is coherent, logically organized, and directly addresses the
classified feedback provided.
"""


# Prompt 2: Role-Based Prompting
ROLE_BASED_PROMPT = """
As a **Senior Software Requirements Engineer** with over 10 years of experience in
drafting comprehensive and high-quality Software Requirements Specification (SRS)
documents, analyze the following classified user feedback. Your expertise should
ensure the SRS is professional, precise, and highly actionable for a development team.
```

```
Classified Feedback:
{classified_feedback}


Generate a well-structured SRS document that includes the following sections, using clear, concise, and unambiguous language:
1. Introduction and Purpose
2. System Overview
3. Functional Requirements
4. Non-Functional Requirements
5. System Features
6. External Interfaces
7. Performance Requirements
8. Design Constraints


Ensure the generated SRS is coherent, logically organized, and directly addresses the classified feedback provided, focusing on **what** the system shall do, rather than **how** it is implemented.
"""


# Prompt 3: Constraint-Based Prompting
CONSTRAINT_BASED_PROMPT = """
Based on the following classified user feedback, generate a comprehensive Software Requirements Specification (SRS) document.


Classified Feedback:
{classified_feedback}
```

```
Generate a well-structured SRS document that includes the following sections, using clear and concise language:

1. Introduction and Purpose

2. System Overview

3. Functional Requirements

4. Non-Functional Requirements

5. System Features

6. External Interfaces

7. Performance Requirements

8. Design Constraints


Crucially, ensure the generated SRS is coherent, logically organized, and directly addresses the classified feedback provided. **STRICTLY avoid including any implementation details, specific design solutions, or technical jargon that is not directly inferable from the user feedback. Each requirement must be unique and non-redundant. Ensure all requirements are stated in a clear, testable, and unambiguous manner.**
"""


# Prompt 4: Few-Shot Prompting

FEW_SHOT_PROMPT = """

Based on the following classified user feedback, generate a comprehensive Software Requirements Specification (SRS) document.


Here are examples of how classified user feedback should be effectively translated into a precise and actionable SRS requirement:

- **Example Feedback:** "The app crashes frequently."

- **Example SRS Requirement:** "The system shall exhibit a crash-free operation rate of 99.9% during normal usage, ensuring application stability."


- **Example Feedback:** "I want to send high-res photos."
- **Example SRS Requirement:** "The system shall support the transmission of images up to 10MB in resolution while maintaining optimal performance."

Classified Feedback:
{classified_feedback}

Generate a well-structured SRS document that includes the following sections, using clear and concise language:
1. Introduction and Purpose
2. System Overview
3. Functional Requirements
4. Non-Functional Requirements
5. System Features
6. External Interfaces
7. Performance Requirements
8. Design Constraints

Ensure the generated SRS is coherent, logically organized, and directly addresses the classified feedback provided.
"""

# Prompt 5: Chain-of-Thought (CoT) Inspired Prompting
COT_PROMPT = """
Carefully and methodically analyze each piece of the following classified user feedback. First, identify the core user need or issue for each feedback item. Then, systematically synthesize these into a comprehensive Software Requirements Specification (SRS) document. Think step-by-step about how each piece of feedback translates into a formal requirement.

Classified Feedback:
```

{classified_feedback}

Generate a well-structured SRS document that includes the following sections, using clear and concise language:

1. Introduction and Purpose

2. System Overview

3. Functional Requirements

4. Non-Functional Requirements

5. System Features

6. External Interfaces

7. Performance Requirements

8. Design Constraints

Ensure the generated SRS is coherent, logically organized, and directly addresses the classified feedback provided.

"""