Functional Requirement (FR) Classification

These prompts were used to classify user reviews into 'feature request', 'bug report', or 'other'.

Zero-Shot Prompt

This was the foundational prompt, designed to test the models' out-of-the-box understanding without any examples.

> "You are an expert software requirements engineer. Your task is to classify user feedback into one of three categories: 'feature request', 'bug report', or 'other'. Only return the category name."

Few-Shot Prompt

This prompt provided the model with examples to guide its learning and align the output format.

> "You are an expert software requirements engineer. Your task is to classify user feedback into one of three categories: 'feature request', 'bug report', or 'other'.
> Here are a few examples:
> Input: 'The app crashes every time I open the camera.'
> Output: 'bug report'
> Input: 'It would be great to have a dark mode option.'
> Output: 'feature request'
> Input: 'I love the new update, great work!'
> Output: 'other'
> Input: [user_review]
> Output:"

Chain-of-Thought (CoT) Prompt

This prompt instructed the model to articulate its reasoning process before providing a final answer, helping to improve accuracy and transparency.

> "You are an expert software requirements engineer. Your task is to classify user feedback into one of three categories: 'feature request', 'bug report', or 'other'.
> First, think step-by-step about the user's request and your reasoning, then provide the final category.
> Input: 'The app crashes every time I open the camera.'
> Thought: 'The user is reporting an issue where the app stops working unexpectedly. This is a clear defect or error in the software. Therefore, it is a bug

> report.'
> Output: 'bug report'
> Input: [user_review]
> Output:"

Non-Functional Requirement (NFR) Classification

These prompts were used to classify user reviews into one of six NFR categories.

Zero-Shot Prompt

This prompt tested the models' ability to map unstructured feedback to a more complex, predefined taxonomy.

> "You are an expert software requirements engineer. Your task is to classify user feedback into one of six non-functional requirement categories: 'usability', 'reliability', 'performance', 'portability', 'security', or 'other'. Only return the category name."

Few-Shot Prompt

This prompt provided examples to guide the model on how to classify NFRs, especially with nuanced user language.

> "You are an expert software requirements engineer. Your task is to classify user feedback into one of six non-functional requirement categories: 'usability', 'reliability', 'performance', 'portability', 'security', or 'other'. Here are a few examples:
> Input: 'The app is so laggy and slow to respond.'
> Output: 'performance'
> Input: 'I can't find the settings menu, it is very confusing.'
> Output: 'usability'
> Input: 'The app frequently crashes without warning.'
> Output: 'reliability'
> Input: [user_review]
> Output:"

Chain-of-Thought (CoT) Prompt

This prompt was used to see if a step-by-step reasoning process would improve the model's ability to handle the ambiguity inherent in NFRs.

> "You are an expert software requirements engineer. Your task is to classify user

> feedback into one of six non-functional requirement categories: 'usability', 'reliability', 'performance', 'portability', 'security', or 'other'. First, think step-by-step about the user's request and your reasoning, then provide the final category.
> Input: 'The app takes forever to load on my old phone.'
> Thought: 'The user is complaining about the speed of the application, which is a key aspect of performance. The reference to an older phone also suggests a potential issue with portability or efficiency. The primary concern, however, is the loading time, which directly relates to performance.'
> Output: 'performance'
> Input: [user_review]
> Output:"

Software Requirements Specification (SRS) Generation

These prompts were used to generate a structured SRS document from user reviews.

Zero-Shot SRS Prompt

This was the baseline prompt for generating a structured SRS document without any guiding examples.

> "You are a requirements engineer. Your task is to generate a Software Requirements Specification (SRS) based on the following user reviews. The SRS must include the following sections: Introduction, Purpose, Functional Requirements, Non-Functional Requirements, User Stories, and a Glossary."

Few-Shot SRS Prompt

This prompt was built on the Zero-Shot prompt and included examples of a raw user review and its corresponding SRS section to guide the output format and style.

> (This prompt was similar to the Zero-Shot prompt but was augmented with 2-3 examples of user reviews and their correctly formatted SRS requirements to provide in-context learning.)

Chain-of-Thought (CoT) SRS Prompt

This prompt instructed the model to first analyze the user reviews before writing the SRS to improve the quality of the generated document.

> (This prompt was a combination of the Zero-Shot SRS prompt and CoT

instructions, directing the model to think through the requirements and then generate the SRS based on that reasoning.)

Constraint-Based SRS Prompt

This prompt was designed to impose strict rules on the output to combat common flaws like redundancy and hallucination.

> "You are a professional requirements engineer. Your task is to generate a Software Requirements Specification (SRS) from the following user reviews. STRICTLY adhere to the following rules:
>
> 1. The document must have sections for Functional Requirements and Non-Functional Requirements.
> 2. Each requirement must be unique and non-redundant.
> 3. AVOID including any implementation details or technical jargon."

Role-Based SRS Prompt

This prompt gave the model a persona to influence the tone, style, and level of detail, aiming for a professionally articulated and highly actionable output.

> "You are a Senior Software Requirements Engineer. Your task is to generate a comprehensive and professionally articulated Software Requirements Specification (SRS) from the following user reviews. The output should be highly actionable for a development team and organized into Functional Requirements and Non-Functional Requirements sections. Use clear, unambiguous language suitable for a senior-level audience."