Project-1d1

1. What are the pain points in using LLMs?

1. Lack of Contextual Grounding and Scope Hallucination

A major pain point is the LLM's inability to stay grounded in the specific, often unstated, constraints of a project. While amplifying the initial 10 use cases, the LLMs introduced features appropriate for a large-scale, enterprise system, not a small cafe's MVP.

Examples of Scope Hallucination: The LLMs suggested complex, regulation-heavy features like **WIC eligibility rules**, **IRS fringe benefit tracking**, and **public health data exports**. **Impact**: This created a significant amount of "noise." The generated requirements were not wrong in a general sense, but they were contextually inappropriate for the project's scale. This forces a human analyst to spend considerable time filtering out ideas that, while plausible, are entirely out of scope, a process that formed the basis of the "condensation" phase. The LLM hallucinates a much larger and more complex project than what is actually intended.

2. Inconsistent Quality and Formatting

The Problem: One model might produce detailed, well-structured flows, while another might generate abbreviated, hard-to-read tables. This creates a fragmented and unprofessional-looking requirements document. **Impact**: A human engineer must perform a tedious "unification" step, manually rewriting and reformatting all outputs to adhere to a single, consistent standard. This manual cleanup negates much of the time saved by the initial generation.

2. Any surprises? Eg different conclusions from LLMs?

The most surprising aspect was not a difference in conclusions, but the **significant variance in the style and scope of the LLM outputs** for the same amplification task. While most models generated detailed, prose-based use cases as requested, one provided its output in a highly condensed tabular format, which was a completely different interpretation of the task. Another surprise was the sheer **"scope explosion"** during amplification. The LLMs independently introduced complex, real-world business concepts that were far beyond the initial project's scope, such as:

- **WIC (Women, Infants, and Children) program eligibility**
- **IRS fringe benefit reporting** for employee meals
- **HIPAA compliance** considerations for health data

3. What worked best?

The LLMs performed best at the task they are most suited for: **rapid brainstorming and requirements amplification**. They excelled at taking a small, defined set of 10 use cases and quickly expanding it into a comprehensive list of ~30 possibilities by exploring various related domains like tax law, health regulations, and food safety. This process effectively front-loaded the creative work. It provided the project team with a rich set of raw material that served as the foundation for the subsequent strategic analysis in The reflection on prompt crafting also noted that using **careful, structured prompts** was highly effective at producing more precise and compliant outputs, highlighting the LLM's strength when properly guided.

4. What worked worst?

The LLMs completely failed at **strategic condensation and prioritization**. They were unable to understand the core concept of a Minimum Viable Product (MVP) or make any context-aware decisions about which features were essential versus optional. The LLMs could not:

- **Distinguish project priorities**: They gave equal weight to a critical function like "Place Order" and a "Version 2.0" feature like "Personalized Recommendations".
- **Understand implicit constraints**: They had no concept of the project's limited scope (a student team building a simple cafe app) and thus generated wildly complex features.

5. What pre-post processing was useful for structuring the import prompts, then summarizing the output?

Pre-processing (structuring prompts) was crucial for guiding the LLM toward a useful output. The most effective techniques were:

- **Providing Concrete Examples (Few-Shot Prompting)**: The best prompts included a specific, well-formatted example of a use case for the LLM to emulate. This was more effective than just describing the format in words.
- **Establishing a Persona and Context**: Starting the prompt with "You are a product manager designing an MVP..." or "You are an analyst working with all the documents in a project directory..." grounded the LLM's response.

Post-processing (summarizing and refining the output) was essential for turning the raw LLM generation into a coherent deliverable. Key steps included:

- **Thematic Categorization**: After the LLM generated ~30 use cases, the team manually grouped them into logical themes like "Core Functionality," "Optimizations & Enhancements," and "Advanced Compliance & Operations". This made the large volume of information easier to analyze.
- **Strategic Filtering**: The most important step was applying a human-centric filter—the **MVP philosophy**—to decide what to keep and what to discard. This involved weighing business value against development effort, a task the LLM could not do.

6. Did you find any best/worst prompting strategies?

Best Strategy: Iterative, Multi-Turn Dialogue

The most effective strategy was treating the process as a conversation rather than a single command.

1. **Prompt 1: Brainstorming MVP Scope**: First, ask the LLM to analyze the expanded list and decide what to *exclude* based on the MVP principle.
2. **Prompt 2: Generating the MVP Use Cases**: Then, based on the scope defined in the previous step, ask the LLM to write the 10 detailed MVP use cases.

3. **Prompt 3: Drafting the Reflection Document**: Finally, ask the LLM to reflect on the process itself.

This **iterative, chain-of-thought approach** breaks down a complex task into manageable parts, allowing the user to guide the LLM at each step. This yielded a far more focused and relevant final product than a single, massive prompt would have.

Worst Strategy: Broad, Zero-Shot Prompting

The least effective strategy, as highlighted in the project reflection, was the **"Zero-Shot Prompt"**. This involves giving a broad, generic instruction with little to no context or examples, such as, "Write a list of use cases for a food delivery system". This approach worked worst because it:

- **Invites "Scope Hallucination"**: Without grounding, the LLM pulls from its vast, generic training data, leading it to suggest overly complex and irrelevant features like IRS reporting.
- **Produces Inconsistent Results**: It gives the LLM too much freedom, resulting in outputs that may not follow the desired format or level of detail.