Reflection

Throughout the process of generating and refining use cases for the food delivery service application, our group gained a clearer understanding of both the promise and the limitations of large language models in requirements engineering. We began by creating ten initial use cases, then added twenty more as new requirements emerged, and finally condensed everything into a minimal viable product. This iterative process showed how quickly LLMs can generate ideas, but also how easily the outputs can become vague or overwhelming without careful guidance and refinement.

The most significant challenges we encountered centered on accuracy, specificity, and trust. LLMs often produced vague or incorrect requirements that lacked the detail needed for implementation. There were also issues of fairness and trustworthiness, as well as a lack of traceability in how outputs were generated. It became clear that human oversight was always necessary. What surprised us most was how differently the models responded to the same prompts. Even when given identical instructions, some models produced structured, helpful outputs while others generated responses that were inconsistent or unusable. This highlighted how much the choice of model matters in the quality of results.

We also learned that the way we framed our prompts had a significant impact on success. Few-shot prompting, where we provided examples of the desired output, consistently produced the most accurate and complete requirements. When our prompts were detailed and specific about format and content, the results were far more useful. In contrast, zero-shot prompting was the least effective approach. Without examples or structure, the responses were vague, inconsistent, and often unusable. From this we concluded that good prompt engineering is critical, and that relying on the LLM alone is not a reliable strategy.

Pre- and post-processing strategies played an important role in structuring the interaction with the LLM. Before prompting, we found it useful to break down long requirement documents into smaller sections and to apply templates such as "As a [user], I want [requirement], so that [goal]." These methods helped focus the model and guide it toward producing actionable results. After receiving the outputs, we often had to filter them to remove duplicates, irrelevant items, or hallucinations, and then validate whether the remaining requirements met clarity and completeness standards. These steps were necessary to turn raw outputs into something usable in our project.

Based on these experiences, we realized that LLMs cannot serve as stand-alone tools for requirements amplification and condensation. They require an ecosystem of surrounding tools to make their outputs practical and reliable. For amplification, this might include edge case generation to capture alternative flows, requirement checks to flag non-functional needs like performance or security, and conflict detection to identify risks or dependencies. For condensation, useful tools would include duplicate removal to eliminate redundant requirements, priority ranking to order requirements by importance, and summary reports to condense lengthy sets into concise outputs for decision-makers.

If we were designing a startup to support requirements engineering with LLMs, we would focus on selling tools that provide this scaffolding. These would include elicitation chatbots that interact with stakeholders and translate conversations into structured requirements, amplification and condensation assistants that guide the process of expanding and summarizing requirements, compliance checkers to ensure that outputs align with industry standards, integrations with GitHub or similar platforms to connect requirements directly to workflows, and analytics dashboards that track requirement quality and evolution over time. These kinds of tools would allow LLMs to move beyond simple text generation and become reliable partners in the engineering process.

The project experience thus far has demonstrated both the strengths and weaknesses of using LLMs for requirements engineering. While they can accelerate brainstorming and automate parts of the process, they also require significant structure, oversight, and supporting tools to produce results that are accurate and actionable. With the right ecosystem in place, however, LLMs could become powerful aids for amplifying, condensing, and managing requirements in real-world engineering projects.