CSC 510 Project 1d1

Due: Monday, Sept 22

Objective: Start thinking about proj2

1. Pain Points in Using LLMs

- **Inconsistent Output Formatting**: LLMs often skipped sections like subflows or alternative flows unless explicitly instructed, requiring manual cleanup.
- **Hallucination of Irrelevant Features**: Some LLMs generated features outside campus context (e.g., multi-state tax compliance, AI-powered inventory optimization), which were unnecessary for MVP validation.
- **Context Window Limitations**: Particularly for Claude, long documents caused truncation or omission of parts of use cases.
- **Version Control and Reproducibility**: Without proper prompt logging, outputs could not easily be reconstructed, leading to potential loss of prior work.

2. Surprises

- **Differences Between LLMs**: Claude and LLaMA occasionally produced divergent conclusions regarding which features were essential or unnecessary. For example, Claude prioritized inventory tracking subflows more comprehensively, while LLaMA focused on payment integration flows.
- **Zero-Shot vs Careful Prompting**: Zero-shot prompting often omitted critical alternative flows or preconditions. Careful prompting with explicit examples produced much more complete, structured outputs.
- **LLM Understanding of MVP Concept**: LLMs sometimes suggested advanced features (loyalty programs, voice ordering) as core, indicating that careful framing of “minimal viable product” is critical to align outputs with project goals.

3. What Worked Best

- **Careful Prompting with Examples**: Providing a full example of a use case structure (preconditions, main flow, subflows, alternative flows) consistently produced the most complete and actionable outputs.
- **Iterative Feedback Loops**: Showing partial LLM outputs and asking for missing flows or expanded details helped achieve completeness.
- **Structured Logging**: Keeping a record of all prompts and responses (for both LLaMA and Claude) enabled comparison and ensured reproducibility.

4. What Worked Worst

- **Zero-Shot Prompting**: Outputs were often incomplete or inconsistent, with missing preconditions or alternative flows.

- **Overly Broad Prompts**: Prompts without clear scope boundaries led LLMs to suggest unnecessary complex features, increasing manual filtering work.

- **Long Unsegmented Inputs**: Feeding the LLM entire 1b1 use case documents without chunking caused truncation or output loss.

5. Useful Pre- and Post-Processing

- **Pre-Processing**:
    - Chunked long use case documents into sections to respect context window limits.
    - Added explicit instructions for output structure (preconditions, main flow, subflows, alternative flows).
    - Included MVP guidance to filter out non-essential features.
- **Post-Processing**:
    - Validated outputs against original requirements to ensure no missing flows.
    - Merged multiple LLM outputs to create consolidated use cases.
    - Highlighted differences between LLaMA and Claude outputs to identify critical gaps or conflicts.

6. Best and Worst Prompting Strategies

- **Best**:
    - Careful, example-based prompting with explicit instructions about structure, scope (MVP), and exclusion rationale.
    - Iterative prompting: asking LLMs to expand, verify, or clean up outputs in successive rounds.
- **Worst**:
    - Zero-shot prompts without examples or structure.
    - Overly vague instructions like “generate all use cases” without specifying format or MVP focus.
    - Prompts expecting LLMs to know campus-specific context implicitly.

Conclusion :

Overall, careful, structured prompting combined with pre-processing (chunking, scope limitation) and post-processing (validation, merging, gap analysis) proved essential to effectively use LLMs for requirements amplification and condensation. Iterative human oversight was necessary to ensure completeness, relevance, and alignment with the MVP. Both Claude and LLaMA provided valuable outputs, but differences in emphasis highlighted the importance of cross-verifying results across models.