# Assignment 1 Journal - LangChain Implementation

## Why Journal?

**Remember: This journal is not for submission or instructor evaluation. Then why journal?**

Building an AI assistant is more than writing code - it’s a chance to sharpen how you **design, debug, and reason** about LLM-powered systems. Use this journal to capture your prompt choices, roadblocks, and “aha!” moments **while they’re fresh**. When you review the reference solution later, these notes will let you see *why* your approach differed/aligned and what patterns you can carry into future projects. It’ll also help jog your memory when you look back a few months from now.

## Part 1: Query Understanding Engine

1. **Classification System Design**: What key instructions did you include in your prompts to distinguish between query types? What iterations of prompts did you have to go through?

2. **Iterative Development**: As you test your system against the provided test cases as well as your own test cases, did you notice any gaps in your initial prompt? How did you improve the prompt iteratively? **Remember:** Try out your own test cases, do not limit yourself to the ones provided.

3. **Observations and surprises**: What are some instances where the model mislabeled a query? Note down a couple of such examples and how you tackled it.

## Part 2: Tool Integration (Calculator + [Bonus Datetime])

1. **Tool Usage Logic**: Explain your approach to detecting when calculations are needed and how your system decides whether to use the calculator or provide a direct response for simple calculations.

2. **Continued improvement**: Did any of your earlier sample queries fail after adding the tool? Note what changed and how did you fix it?

3. [**Bonus Implementation -** **ONLY if you implemented the DateTime tool**] What were the main challenges encountered? How did the prompt change to handle this new tool alongside the calculator?

4. **Extensibility:** What are some challenges you see for including more tools in the future? Think about one specific new tool (e.g. weather report, SQL query etc.) and how the workflow should change for the new tool?

## Part 3: Conversation Memory

1. **Memory strategy**: What prompt changes did you make to take memory into account?

2. **Limitations and Improvements**: What are some limitations of the current memory implementation where you are storing the entire conversation? How would you improve this in a real-world scenario? (**Hint:** Think about context length and long conversations, vector stores, persona based memory etc.)

## [Bonus] Part 4 - Thinking about Evaluation Metrics

**1. Metric-to-feature mapping**You just implemented a 2 step workflow agent (or more steps based on your approach). What are the high-level metrics that you need to use for each feature of your system?  
 • Choose one metric discussed in the lectures (Exact-Match, Fuzzy Keyword, Semantic Similarity, or LLM-Judge etc.) and explain *why* that metric tells you the most about quality.

**2. When exact/fuzzy fails**What are some cases where exact/fuzzy match metrics fail? What type of metrics would you consider in those cases?

**3. Designing an LLM-judge prompt**If you use an LLM-judge prompt for measuring any capability of the system, how would you craft the prompt? Doesn’t need to elaborate - high-level understanding is sufficient.

**4. Complexity of LLM Judges**

If you go with the LLM-judge route for any of the metrics, what additional work would you need to do? (hint: refer lecture material on evaluation metrics).

**5. Context-retention check**Suppose you want to measure whether the assistant correctly uses conversation history in follow-up questions.  
 • Outline a small LLM-judge rubric (or signal) you’d use to flag replies that *ignore* prior context.

**Mega Bonus:** Once you are comfortable answering the above questions related to metrics, think about how you’d implement a system to test different prompts, create test datasets, track different metrics etc. You don’t need to **actually implement it** but think about how/what needs to be built to do this. This will help you appreciate the complexity of monitoring and observability platforms when we introduce them.

## Overall Reflection

1. **Most impactful design choices**: What were the most important design decisions you made in your implementation, and why did you make these choices? Which LangChain components were most crucial to your solution?

2. **Biggest ‘aha’ moment**: You might’ve used ChatGPT-like services before, but implementing these workflow agent systems will deepen your understanding of how they work. What are some aha learning moments during this assignment?

3. **Hardest roadblock and solution**: Describe one significant challenge you faced and explain your approach to solving it.

4. **For the future**: What is one principle that you learnt from this assignment? Think about how you can take those learnings and implement them in your area of work.