Task Manager Codebase Comprehension Summary

Msesenyane Makhongela

Initial vs Final Understanding of the Codebase

- When I first opened the project, I relied mostly on file names and directory structure. I assumed the storage file was the final layer of the system and did not realise that task data was written to a JSON file at runtime. I also did not clearly understand how business logic and persistence were separated.

- After exploring the codebase further, I understood that the project follows a layered structure. The CLI handles user input and commands, the application layer contains the task logic, and the storage layer handles reading and writing to `tasks.json`. The models mainly define data structure and constants rather than behaviour. Responsibilities between files are now clear, and the flow of data through the system makes sense.

Most Valuable Insights from Each Prompt

- Project Structure Prompt helped me identify the main entry point of the application and clarified file responsibilities. It also corrected my wrong assumptions about where data is stored.

- Feature Location Prompt showed that new features usually affect multiple files. It highlighted the importance of following existing implementation patterns instead of adding functionality in just one place.

- Domain Understanding Prompt clarified what a task represents in the system and where business rules are enforced. It also pointed out that some rules are implicit and not formally documented.

Approach to Implementing the New Business Rule

To implement the overdue task rule, I would modify the logic in the application layer where tasks are processed. I would add a check for how long a task has been overdue and mark it as abandoned if it exceeds seven days, unless it has high priority. Before implementing, I would confirm with the team how often this rule should run and whether it should happen automatically or be triggered by a command.

Strategies for Approaching Unfamiliar Code in the Future

This exercise showed me that understanding the project structure before diving into code is more effective. Focusing on one feature at a time and using AI to validate assumptions helped reduce confusion. Writing down observations and documenting understanding as I explored the code made gaps easier to identify and clarified how different parts of the system interact.