AI Codebase Comprehension Reflection – Task Management System

1. Initial vs. Final Understanding of the Task Manager Codebase

At first, the Task Manager project looked confusing because I was unfamiliar with its structure and command-line interface. I only understood that it was a Python application used to manage tasks. Looking at the files and commands was overwhelming, and I wasn't sure how everything worked together.

After exploring the documentation and using AI to explain the project, my understanding improved significantly. I learned that the system allows users to create, list, update, and manage tasks through different CLI commands. Tasks can be filtered by priority, status, due date, or overdue status. The application also supports adding and removing tags, viewing task details, displaying statistics, and running unit tests to ensure the code functions correctly. I now understand how the different commands work together to provide a complete task management system.

2. The Most Valuable Insights Gained from Each Prompt

The AI prompts helped me understand the project step by step instead of trying to learn everything at once.

The most valuable insights were:

- Understanding the overall purpose of the application before looking at the code.
- Learning what each CLI command does and when it should be used.
- Understanding how tasks can be created, updated, filtered, and organized.
- Seeing how tags, priorities, and due dates improve task management.
- Learning the importance of unit testing to verify that changes do not break the application.

- Realizing that reading documentation first makes understanding a codebase much easier.

These explanations helped me understand both the functionality of the application and the logic behind its design.

3. My Approach to Implementing the New Business Rule

If I needed to implement a new business rule, I would first study the existing code to understand where similar functionality is already implemented. I would identify the files responsible for handling tasks and determine where the new rule should be added.

Next, I would carefully modify only the relevant section of the code while keeping the existing functionality unchanged. After making the changes, I would run the unit tests to check that everything still works correctly. Finally, I would test the new feature using different scenarios to make sure it behaves as expected and update the documentation if necessary.

This approach reduces errors and makes the code easier to maintain.

4. Strategies I Have Developed for Approaching Unfamiliar Code in the Future

This exercise taught me several useful strategies that I will continue using when working with unfamiliar codebases:

- Start by reading the README and documentation before opening the source code.
- Understand the overall purpose of the project before focusing on individual files.
- Break the project into smaller sections instead of trying to understand everything at once.
- Use AI to explain unfamiliar code, commands, and programming concepts in simple language.
- Test the application regularly after making changes.

- Read and understand existing code before adding new features.
- Keep notes while exploring the project to make future work easier.

Following these steps will help me understand new projects more quickly and confidently.

Conclusion

This exercise showed me that understanding an unfamiliar codebase becomes much easier when using documentation together with AI guidance. Instead of feeling overwhelmed, I learned how to analyse a project step by step, understand its functionality, and confidently plan modifications. The experience has improved both my technical understanding and my confidence in approaching new software projects in the future.