Maintaining and Expanding Software for Component Validation

This exercise focuses on strategies for working with existing code bases and ensuring the software remains maintainable as new features and requirements are introduced.

1. Working with Existing Code

How would you approach understanding and contributing to an existing code base with minimal disruption?

What practices would you follow to ensure your changes integrate well with the current structure?

2. Ensuring Maintainability

What techniques would you use to keep the code base clean, modular, and easy to maintain as new features are added?

How would you handle code documentation and testing to support long-term maintainability?

3. Balancing Flexibility and Stability

How would you design or refactor the software to make it flexible for future changes while ensuring the existing functionality remains stable?

Which design patterns or principles would you apply to achieve this balance

1.

First, I try to understand what the software does and how the components interact. Next, I identify the modules, their responsibilities, and how they are related. Following the setup instructions, I run the project locally to see how everything works in practice.

Then I find where the feature or bug I'm working on is located and understand the design decisions and style of the accompanying code.

Next, while maintaining the code style, naming conventions, and structure already in use, I make minimal, targeted changes, changing only what is needed.

If my changes affect the logic, I update the unit tests, and make sure to comment the code so that my logic is clear.

After making the changes, if possible, I run the full project and tests to make sure the program still works properly.

After that, I would ask for a code review.

2.

I make minimal changes, modifying only what is necessary to avoid introducing new, unrelated problems. I try to preserve the existing code style, naming, and structure to keep the code readable and consistent. Before making any changes, I make sure that the new features integrate well without breaking existing modules.

Then again, if my changes affect the logic, I update the unit tests, and make sure to comment the code so that my logic is clear.

After making the changes, if possible, I run the full project and tests to make sure the program still works properly.

3.
I would design classes so that new behavior can be added (e.g., through inheritance or composition) without changing existing code. This protects working code from being accidentally broken.