Section 3: Technical Leadership & Critical Thinking

1. **Code Review & Best Practices**

A junior developer submits messy, AI-generated code with unnecessary

complexity.

How would you:

■ Guide them towards better coding practices?

■ Ensure the team follows consistent coding standards?

Answer :

1. Guide the Junior Developer Toward Better Practices

- Build trust and start with positive

- Encourage them to write and solve the problem first, then use AI suggest to find more efficent way

- Code readability is important (clear names and identical case type)

- Not only copy and paste but understanding the logic, try to breakdown AI Generated code and search pros and cons

2. Ensure Consistent Team Standards

- Adopt a linter/formatter (important)

- Document coding standards (naming conventions and anti-patterns)

- Give an example (submit clean code and well commented)

- Always discus about current issue to find the best solutions

2. **Team Collaboration & Conflict Resolution**

○ Your team disagrees on the best tech stack for a new feature.

○ How do you facilitate a decision-making process while keeping the team motivated?

Answer :

- Align on Goals: Clarify requirements (performance, scalability, deadlines).
- Throw pros and cons for each idea on the table.
- Grab your top picks. Build a scrappy little prototype, do a heavy test and see what falls apart first, and learn from it. Real world trial beats endless theory.
- Pick & Stick: Once you've picked your horse, back it 100%. No more grumbling or "I told you so" later on.
- Bottom line? Chase the best results, not your ego.

3. **Handling Technical Debt**

○ Your inherited codebase has legacy code with poor documentation.

○ What strategies would you use to refactor and modernize it without breaking existing functionality?

Answer :
First of all this is a very often case, I think every programmer ever feel the same case

- First thing is, understand the task and start running the codebase, see how it works.
- Start small, try with the easiest task or try to add some logic or change something.
- As you go through, you're the one that have responsible for documenting what you did, so start self documenting.
- Refactor the code step by step until the code become clean and easy to understand.
- Make sure your self documentary not missing