

AIDD 30 Day Challenge

Task 2:

Part A: Theory (Short Questions)

1. Nine Pillars Understanding:

Q1: Why is using AI Development Agents (like Gemini CLI) for repetitive setup tasks better for your growth as a system architect?

A: Using AI Development Agents for repeat setup tasks helps you save time, work more consistently, and focus on high level architecture. This allows you to grow faster as a system architect because the AI handles the small, routine work for you.

. Time saving . Focus on high-level design . Consistency & Reliability . Skill Leverage & Growth

Q2: Explain how the Nine Pillars of AIDD help a developer grow into an M-Shaped Developer.

A: Nine Pillars developer ko multiple skills ek saath sikhate hain. Har pillar ek new capability deta hai jaise:

- Specs likhna
- Testing karna
- AI agents use karna
- System architecture samajhna
- Automation aur tools chalana
- MCP se integrations
- AI ke through multi step workflows banana

Jab developer ye sub pillars use karta hai, to wo sirf coding nahi, design + testing + automation + AI orchestration sab seekhta hai. Is tarah wo 2 4 domains me deep expert ban jata hai yahi M-Shaped Developer hota hai.

In short:

- Specs → Thinking and planning skills
- TDD → Quality and testing mindset
- Agents → Automation and AI collaboration
- MCP → Integration and system-level understanding
- AI CLI tools → Speed + technical workflow mastery
- Architecture thinking → Big-picture vision
- SDD workflow → Structured engineering approach
- Vibe → Spec transition → Creativity + discipline
- Tooling ecosystem → Practical execution skills

2: Vibe Coding vs Specification-Driven Development:

Q1: Why does Vibe Coding usually create problems after one week?

A: Vibe Coding creates problems after one week because there is no clear plan, no structure, and no record of decisions. As the project grows, the code becomes messy, inconsistent, and hard to maintain.

You forget why you wrote something, AI generates random variations, and fixing bugs becomes difficult. The whole project becomes confusing.

Q2: How would Specification-Driven Development prevent those problems?

A: SDD prevents these problems by giving the project a clear structure before coding begins. Specifications act like a map: they define requirements, logic, and decisions in writing. AI follows the same specs every time, so the code stays

consistent, organized, and easy to update. Because everything is documented, the project stays clean and maintainable for weeks or months.

- Code structure clear hota hai.
- AI har baar same direction follow karta hai.
- Random prompts random code ka issue nahi hota.
- Sab decisions document ho jate hain.
- Project maintainable rehta hai.
- Bugs aur changes karna easy ho jata hai.

3. Architecture Thinking:

Q1: How does architecture-first thinking change the role of a developer in AIDD?

A: Architecture first thinking makes the developer act less like a “coder” and more like a system designer.

- You focus on structure, not lines of code .
- You decide how every part connects.
- AI follow your plan.
- You take decision high level like a team lead.

Q2: Explain why developers must think in layers and systems instead of raw code.

A: Developers should think in layers because it keeps the project clean, simple, and easy to understand. If all the code is mixed together, it becomes confusing. But when the UI, logic, and database are in separate layers, everything is clear, bugs are easier to find, and adding new features becomes simple. AI also understands which part does what so it can generate better code.

Part B: Practical Task

```
> Write a 1-paragraph specification for an email validation function.  
Requirements: Must contain "@", must have a valid domain (like .com or .org),  
and should return clear error messages if invalid.
```

```
~ no sandbox (see /docs)
```

```
Microsoft Windows [Version 10.0.10240.6391]  
(c) Microsoft Corporation. All rights reserved.  
C:\Users\xpsx360@gemini  
> GEMINI  
Tips for getting started:  
1. Ask questions, edit files, or run commands.  
2. Be specific for the best results.  
3. Create GEMINI.md files to customize your interactions with Gemini.  
4. /help for more information.  
  
> Write a 1-paragraph specification for an email validation function.  
Requirements: Must contain "@", must have a valid domain (like .com or .org),  
and should return clear error messages if invalid.  
  
◆ The email validation function will accept a single string as input and verify its format against standard email conventions. It will confirm that the string contains exactly one "@" symbol, which must separate a non-empty username from a domain. The domain itself must contain at least one period, and the top-level domain (e.g., ".com", ".org") must be at least two characters long. If the input string fails any of these criteria, the function will return a clear, specific error message detailing the nature of the invalidity, such as "Error: Missing '@' symbol" or "Error: Invalid domain extension." A valid email address will result in a success confirmation.
```

Part C: Multiple Choice Questions

1. What is the main purpose of Spec-Driven Development?

Answer: B. Clear requirements before coding begins

2. What is the biggest mindset shift in AI-Driven Development?

Answer: B. Thinking in systems and clear instructions

3. Biggest failure of Vibe Coding?

Answer: B. Architecture becomes hard to extend

4. Main advantage of using AI CLI agents (like Gemini CLI)?

Answer: B. Handle repetitive tasks so dev focuses on design & problem-solving

5. What defines an M-Shaped Developer?

Answer: C. Deep skills in multiple related domains

Reflection

In AI driven development, coding is no longer just about writing lines of code, planning and design system thinking are equally important. AI tools handle our repetitive and low level tasks, allowing us to focus on high level decision and design . By using the nine pillars, we can develop multiple skills at the same time and become an M shaped developer. Vibe Coding may feel easy in the short term, but Specification Driven Development (SDD) and architecture thinking ensure long term clean, maintainable, and scalable code.