Name: Rizwan

Slot: Friday 6 PM – 9 PM

AI-Driven Development — 30-Day Challenge

Task 2

Part A — Theory (Short Questions)

1. Nine Pillars Understanding

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

Using AI development agents is better for growth as a system architect because they take care of small coding tasks and let you focus on the bigger design of the system. Instead of wasting time fixing small errors, you spend more time understanding how different parts of a system connect and work together. This helps you think more clearly about structure, layers, data flow, and responsibilities. You learn to design better systems because the AI handles the low-level work, and you focus on planning, organizing, and improving the overall architecture.

How do the Nine Pillars help a developer grow into an M-Shaped Developer?

The Nine Pillars help a developer grow because they mix different skills like testing, specs, automation, AI tools, and system thinking. When a developer uses these pillars daily, they learn more than just coding. They learn architecture, planning, automation, and debugging. Over time, they become strong in multiple areas, not just one. This is what makes them an M-Shaped developer with deep skill in many connected fields.

2. Vibe Coding vs Specification-Driven Development

Why does Vibe Coding break after one week?

Vibe coding creates problems because there is no clear plan. The developer writes whatever feels right at the moment, so after a week the code becomes messy, hard to understand, and difficult to change. Even the developer forgets what they wrote and why they wrote it. This leads to bugs and confusion.

How would Specification-Driven Development prevent those problems?

Specification-Driven Development prevents these issues because the plan is written first. The developer knows exactly what the system should do before touching code. The specs guide the whole project, so the code becomes cleaner, easier to follow, and simple to update later. It reduces confusion and keeps the project stable.

3. Architecture Thinking

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

Architecture-first thinking changes the role of a developer because they stop acting like a normal coder and start thinking like a system planner. Instead of focusing on small functions, they focus on how the whole system works. They make decisions about layers, data flow, components, and agents. Their role becomes more strategic and less about typing code.

Why must developers think in layers and systems instead of raw code?

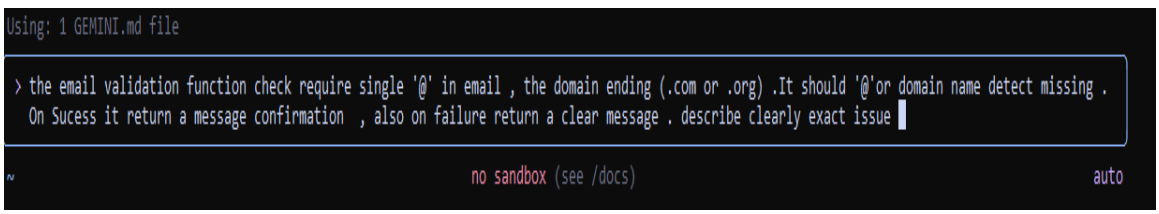
Developers must think in layers because modern software is too complex to build by guessing. Thinking in layers helps keep the system organized. Each part has a clear job, so future updates are easy. When a developer thinks in systems, the project becomes stable, scalable, and easier to maintain. Raw coding without structure leads to confusion and weak architecture.

Part B — Practical Task

CLI Prompt:

"Generate a one-paragraph specification for an email validation function with these requirements: must contain '@', must contain a valid domain (.com or .org), and must return clear error messages."

Specification Output:



Part C — Multiple Choice Answers

1. B

2. B

3. B

4. B

5. C

Reflection

This task shows how the role of a developer is shifting from writing code to shaping systems.

AI tools and structured methods like SDD help developers think bigger and operate across multiple skill areas, making the M-Shaped path more achievable.