

## Part A — Theory (Short Questions)

### Nine Pillars Understanding

- **Why is using AI Development Agents (like Gemini CLI) for repetitive setup tasks better for your growth as a system architect?**  
AI Agents (jaise Gemini CLI) woh kaam fast aur error-free kar dete hain jo repeat hote hain. Jaise setup, config, scaffolding Is se hum apni energy **Complex Architecture or System Design** pr laga dete hen. Yeh humari skills (**Higher-Level Thinking**) ko grow karti hain, jo ek system architect ke liye zaroori hoti hain.
- **Explain how the Nine Pillars of AIDD help a developer grow into an M-Shape Developer.**

Nine Pillars humari multiple sides ko develop karte hain. architecture, tools, automation, testing, documentation, collaboration, AI usage.

In sab se hum sirf “coder” nahi rehte, balke **multi-skilled (M-Shaped)** developer ban jaate hen jo system design, coding, aur AI tools sab manage kar sakta hai.

### Vibe Coding vs Specification-Driven Development

- **Why does Vibe Coding usually create problems after one week?**

Vibe coding bina planning hota hai , jo idea aaye, code likh diya.

Initially acha lagta hai, lekin 1 week baad code messy ho jata hai, structure toot jata hai, bugs aate hain aur naye features add karna mushkil ho jata hai.

Basically project control se bahar ho jata hai.

- **How would Specification-Driven Development prevent those problems?**

Specification-driven development me coding se pehle **clear plan** ban jata hai:

kya banana hai, kyu banana hai, kis tarah banana hai.

Is se code structured hota hai, future changes aasaan hoti hain, aur project stable rehta hai.

Aise projects long term me bhi toot-te nahi.

## Architecture Thinking

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

Is approach me developer sirf code likhne wala nahi hota. Woh pehle **complete system ka design**, flows, layers, aur tools decide karta hai.

Is se developer ki thinking improve hoti hai aur woh **system architect** banne lagta hai.

- **Explain why developers must think in layers and systems instead of raw code.**

Layers me sochne se code clean, organized, scalable aur easy-to-maintain hota hai.

Aapko pata hota hai kaunsa part business logic ka hai, kaunsa UI ka, kaunsa database ka.

Raw code me sochne se system ka big picture clear nahi hota aur project jaldi messy ho jata hai.

## **Part B – Practical Task:**

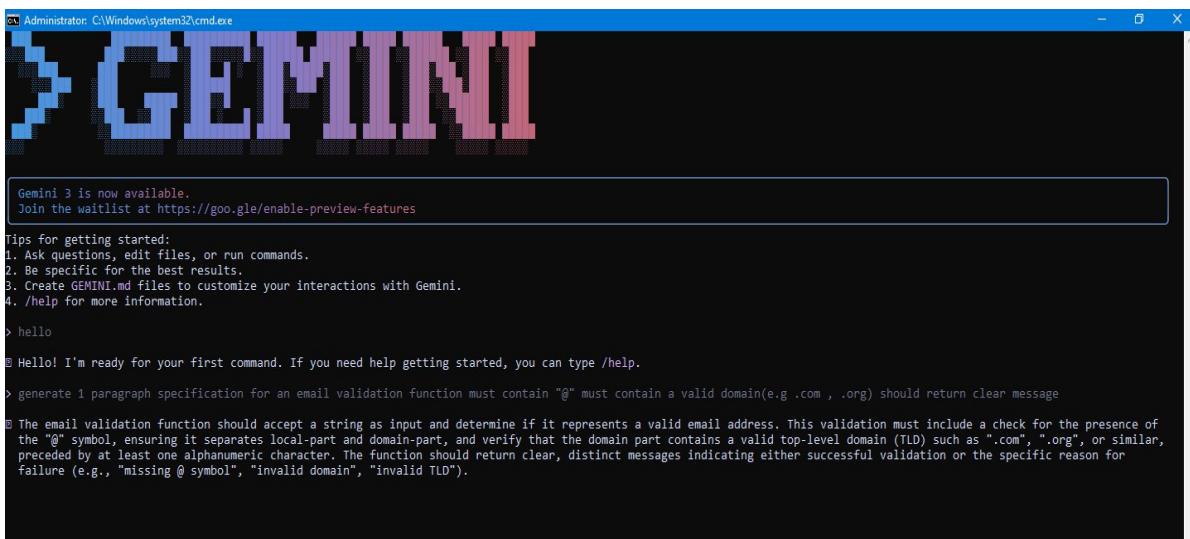
### **Prompt:**

generate 1 paragraph specification for an email validation function must contain "@" must contain a valid domain(e.g .com , .org) should return clear message.

### **Output:**

The email validation function should accept a string as input and determine if it represents a valid email address. This validation must include a check for the presence of the "@" symbol, ensuring it separates local-part and domain-part, and verify that the domain part contains a valid top-level domain (TLD) such as ".com", ".org", or similar,

preceded by at least one alphanumeric character. The function should return clear, distinct messages indicating either successful validation or the specific reason for failure (e.g., "missing @ symbol", "invalid domain", "invalid TLD").



The screenshot shows a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The window features a large, stylized "GEMINI" logo composed of blue and purple pixels. Below the logo, a message box displays the text: "Gemini 3 is now available. Join the waitlist at <https://goo.gle/enable-preview-features>". A section titled "Tips for getting started:" lists four items: 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. The command line shows the user entering "hello" and receiving a response: "Hello! I'm ready for your first command. If you need help getting started, you can type /help." The user then types "generate 1 paragraph specification for an email validation function must contain '@' must contain a valid domain(e.g .com , .org) should return clear message" and receives a detailed response about email validation rules.

## Part C – Multiple Choice

### Questions:

1. **B.** Clear requirements before coding begins
2. **B.** Thinking in systems and clear instructions
3. **B.** Architecture becomes hard to extend
4. **B.** Handle repetitive tasks so dev focuses on design & problem-solving
5. **C.** Deep skills in multiple related domains