🌐AI-Driven Development — 30-DayChallenge

(Task-2 “Theory + Practical + MCQs”)

🗀 Part A — Theory (Short Questions)

A. Nine Pillars Understanding

A1) Why is using AI Development Agents (like Gemini CLI) for repetitive setup tasks better for your growth as a system architect?

Ans) Using AI Development Agents reduces time spent on repetitive setup work and allows the developer to focus on system level thinking. This builds strong architecture skills, improves decision making and strengthens the ability to give clear instruction, essential qualities of a system architect.

B2) Explain how the Nine Pillars of AIDD help a developer grow into an M-Shaped Developer?

Ans) The Nine Pillars combine architecture, automation, testing, specifications and AI agents into one unified workflow. Working with all these pillars develops deep skills across multiple areas helping the developer become an M-Shaped professional with broad and strong capabilities.

B. Vibe Coding vs Specification-Driven Development

A1) why does Vibe Coding usually create problems after one week?

Ans) Vibe Coding lacks planning, structure and documentation. After a week, the project becomes confusing, difficult to debug and hard to extend because decisions were made randomly without clear direction.

B2) how would Specification-Driven Development prevent those problems?

Ans) Specification-Driven Development starts with clear written requirements before coding. This keeps the project organized, predictable, easier to maintain and simple to test. The system grows in a clean and scalable way.

C. Architecture Thinking

A1) how does architecture-first thinking change the role of a developer in AIDD?

Ans) Architecture-first thinking shifts the developer from “just writing code” to “designing complete systems.” The developer plans flows, organizes components and guides AI agents instead of focusing only on implementation details.

B2) Explain why developers must think in layers and systems instead of raw code.

And) Layered thinking keeps systems modular, clean, and scalable. It prevents tangled code and makes it easier for both humans and AI agents to understand, maintain and extend the project.

Part B — Practical Task

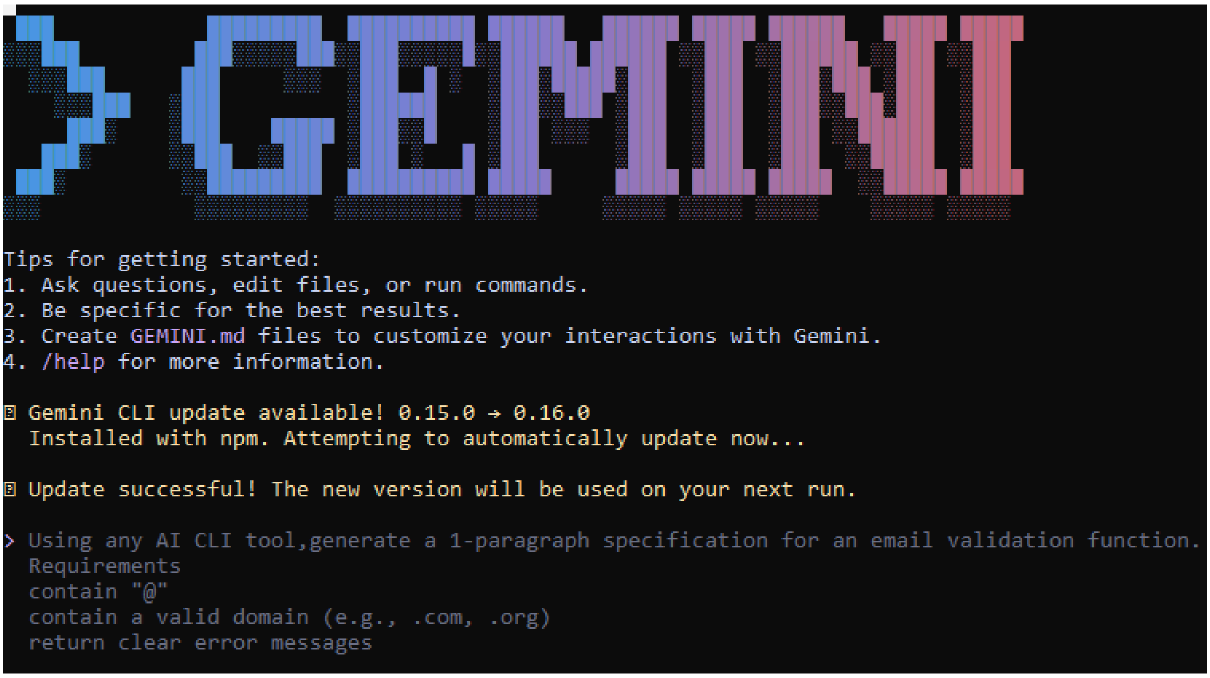
Task:

“Using any AI CLI tool, generate a 1-paragraph specification for an email validation function”.

Requirements:

- ✓ **Must contain “@”**
- ✓ **Must contain a valid domain (e.g., .com, .org)**
- ✓ **Should return clear error messages**

Screenshot

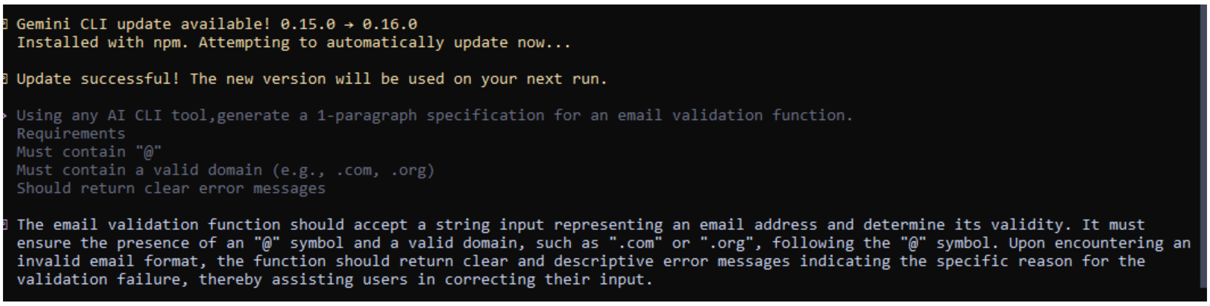


```
Tips for getting started:
1. Ask questions, edit files, or run commands.
2. Be specific for the best results.
3. Create GEMINI.md files to customize your interactions with Gemini.
4. /help for more information.

▯ Gemini CLI update available! 0.15.0 → 0.16.0
  Installed with npm. Attempting to automatically update now...

▯ Update successful! The new version will be used on your next run.

> Using any AI CLI tool,generate a 1-paragraph specification for an email validation function.
  Requirements
  contain "@"
  contain a valid domain (e.g., .com, .org)
  return clear error messages
```



```
▯ Gemini CLI update available! 0.15.0 → 0.16.0
  Installed with npm. Attempting to automatically update now...

▯ Update successful! The new version will be used on your next run.

> Using any AI CLI tool,generate a 1-paragraph specification for an email validation function.
  Requirements
  Must contain "@"
  Must contain a valid domain (e.g., .com, .org)
  Should return clear error messages

▯ The email validation function should accept a string input representing an email address and determine its validity. It must
  ensure the presence of an "@" symbol and a valid domain, such as ".com" or ".org", following the "@" symbol. Upon encountering an
  invalid email format, the function should return clear and descriptive error messages indicating the specific reason for the
  validation failure, thereby assisting users in correcting their input.
```

Part C — Multiple Choice Questions

- **1-B**
- **2-B**
- **3-B**
- **4-B**
- **5-C**