AIDD 30-Day Challenge — Task 2

Student Name: **Muhammad Nabeel Ali**

Lead Teacher: **Sir Hamzah Syed**

Slot: **Friday 6 to 9pm**

📁Part A — Theory (Short Answers)

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?

   Because AI agents remove low-value setup work so you can focus on high-level system design, specifications, and architecture skills that grow you into a system architect.

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

   Because each pillar builds a different deep and broad skill—architecture, agents, multimodal thinking, DevOps, evaluation, and collaboration—shaping you into a multi-specialized M-Shaped Developer.

2. **Vibe Coding vs Specification-Driven Development**

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

   Because Vibe Coding skips proper planning and specifications, so after one week the code becomes messy, inconsistent, and hard to maintain.

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

Because Specification-Driven Development provides clear rules, structure, and intentions upfront, so the system stays consistent and maintainable as it grows.

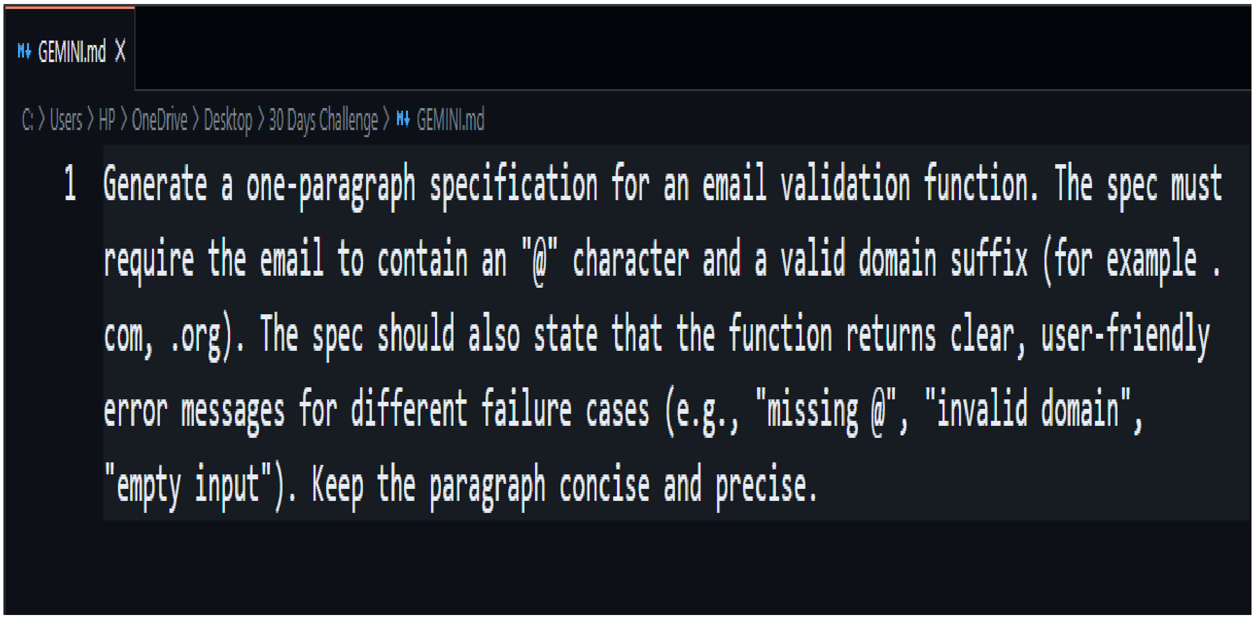
3. **Architecture Thinking**
   Q1: How does architecture-first thinking change the role of a developer in AIDD?

   It shifts the developer’s role from writing code to designing systems, defining specifications, and directing AI to build the implementation.

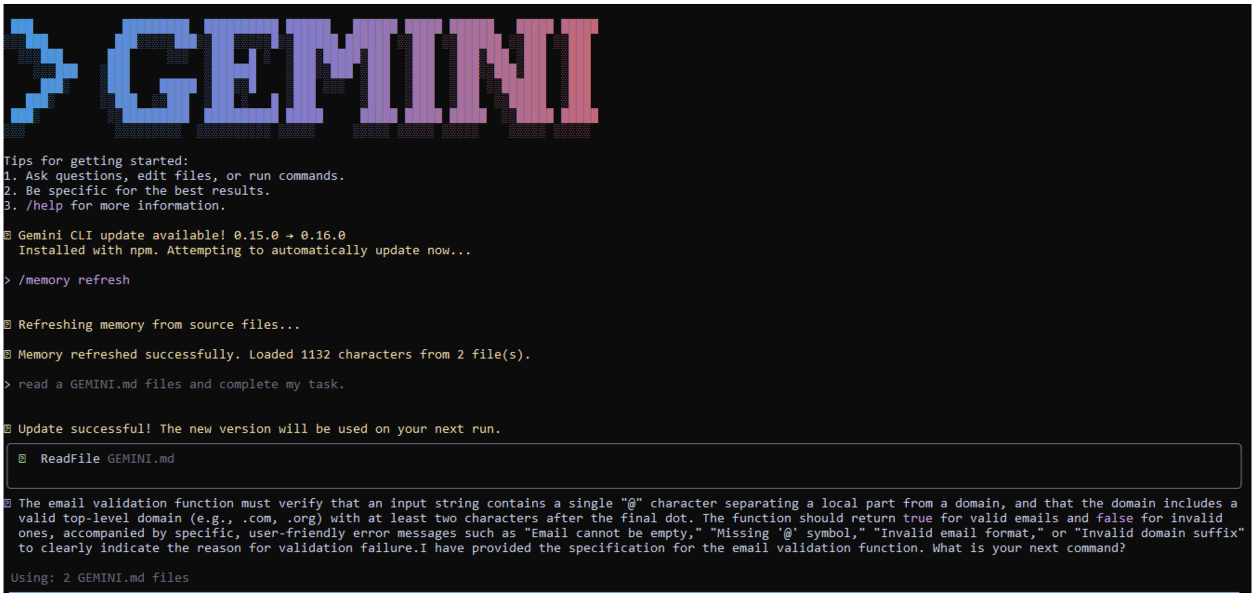
   Q2: Explain why developers must think in layers and systems instead of raw code.

   Because thinking in layers and systems creates scalable, maintainable architectures, while thinking in raw code leads to fragile, unstructured solutions.

📁 Part B — Practical Task


GEMINI.md X
C: > Users > HP > OneDrive > Desktop > 30 Days Challenge > GEMINI.md

Generate a one-paragraph specification for an email validation function. The spec must require the email to contain an "@" character and a valid domain suffix (for example .com, .org). The spec should also state that the function returns clear, user-friendly error messages for different failure cases (e.g., "missing @", "invalid domain", "empty input"). Keep the paragraph concise and precise.



```
Tips for getting started:
1. Ask questions, edit files, or run commands.
2. Be specific for the best results.
3. /help for more information.

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

> /memory refresh


▯ Refreshing memory from source files...

▯ Memory refreshed successfully. Loaded 1132 characters from 2 file(s).

> read a GEMINI.md files and complete my task.


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

 ▯  ReadFile GEMINI.md

▯ The email validation function must verify that an input string contains a single "@" character separating a local part from a domain, and that the domain includes a
  valid top-level domain (e.g., .com, .org) with at least two characters after the final dot. The function should return true for valid emails and false for invalid
  ones, accompanied by specific, user-friendly error messages such as "Email cannot be empty," "Missing '@' symbol," "Invalid email format," or "Invalid domain suffix"
  to clearly indicate the reason for validation failure.I have provided the specification for the email validation function. What is your next command?

 Using: 2 GEMINI.md files
```

📁 Part C — Multiple Choice Questions

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

Reflection

AI-Driven Development (AIDD) shifts a developer's mindset from writing raw code to thinking in **systems, layers, and clear specifications**. Using **Spec-Driven Development** ensures requirements are defined before coding, preventing messy, hard-to-extend architectures common in Vibe Coding. Tools like **AI CLI agents (e.g., Gemini CLI)** handle repetitive setup tasks, freeing developers to focus on design, problem-solving, and system thinking. By mastering the **Nine Pillars of AIDD**, developers grow into **M-Shaped Developers** with deep skills across multiple domains while collaborating effectively with AI. This approach emphasizes clarity, scalability, and long-term maintainability, shaping modern software architects.