AI-Driven Development — 30-Days Challenge

Task 2 :

Name : Yusra Anum
Class Slot: Friday — 6:00 PM to 9:00 PM
Instructor: Sir Hamzah Syed

Part A — Theory (Short Questions)

Nine Pillars Understanding

1.Why is using AI Development Agents for repetitive setup tasks better for your growth as a system architect?

Using an AI agent for boring, repetitive tasks (like setting up a new project with the same config files) is better for my growth because it forces me to *think about the task* instead of just *doing the task*. I have to design a clear specification for the AI to follow. This is what a system architect does—they design the blueprint and the requirements, they don't lay every single brick themselves. It lifts my focus from the low-level "how" to the high-level "what" and "why."

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

The Nine Pillars create a structured process that pushes me to be broad *and* deep—the definition of an M-Shaped developer.

- The Broad Skills (the top of the 'M') come from pillars like Product & UX Thinking, System Architecture, and Security. I can't just think about code; I have to think about the entire system, the user, and safety from the start.
- The Deep Skills (the legs of the 'M') are reinforced by pillars like Spec-Driven Development and Precision Debugging. I go deep into the details to create perfect specifications and then use tools like the AI Agent to build it, forcing me to understand the core logic and fix complex bugs precisely.

Instead of just being a coder who knows one thing deeply (I-shaped), the pillars force me to be a architect-designer-debugger who can span the whole process.

Vibe Coding vs Specification-Driven Development

3.Why does Vibe Coding usually create problems after one week?

Vibe coding is when I just start typing code based on a general feeling or a loose idea, asking the AI for small bits without an overall plan. It creates problems after about a week because that's when the initial "vibe" runs out. I end up with a pile of code that doesn't fit together well. There's no clear structure, so when I need to add a new feature or fix a bug, I don't know where things are or how they connect. The code becomes a tangled mess that is hard to understand and even harder to change.

4.How would Specification-Driven Development prevent those problems?

Specification-Driven Development prevents this by making me write a detailed "blueprint" *before* any code is generated. I have to define the components, their responsibilities, the data flow, and the APIs. This acts as a source of truth. When I, or the AI, build from this spec, everything has a designated place and a clear purpose. A week later, when I need to make a change, I can look at the spec to understand the system's design instantly, and any new code has to fit into that pre-defined structure, preventing the tangled mess.

Architecture Thinking

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

In traditional coding, my role was to write the code to make a feature work. With architecture-first thinking in AIDD, my primary role shifts to being a designer and a systems engineer. My main job is to design a robust, scalable system architecture and then write the precise specifications that an AI Agent will use to write the code. I'm moving from a "builder" to an "architect and foreman."

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

Thinking in raw code is like a bricklayer only thinking about the next brick. Thinking in layers and systems is like being the building architect who thinks about the foundation, the plumbing, the electrical, and the structure all working together.

If I only think in code, I create a system where everything is tangled together (tightly coupled). Changing one thing breaks three others. When I think in layers (e.g., UI layer, business logic layer, data layer) and systems (e.g., authentication system, payment system), I build with loose coupling. This means each part has a specific job and interacts with others through clean interfaces. It makes the entire application easier to understand, debug, and most importantly, change and scale over time without falling apart.

Part B — Practical Task (Screenshot Required)

Task:

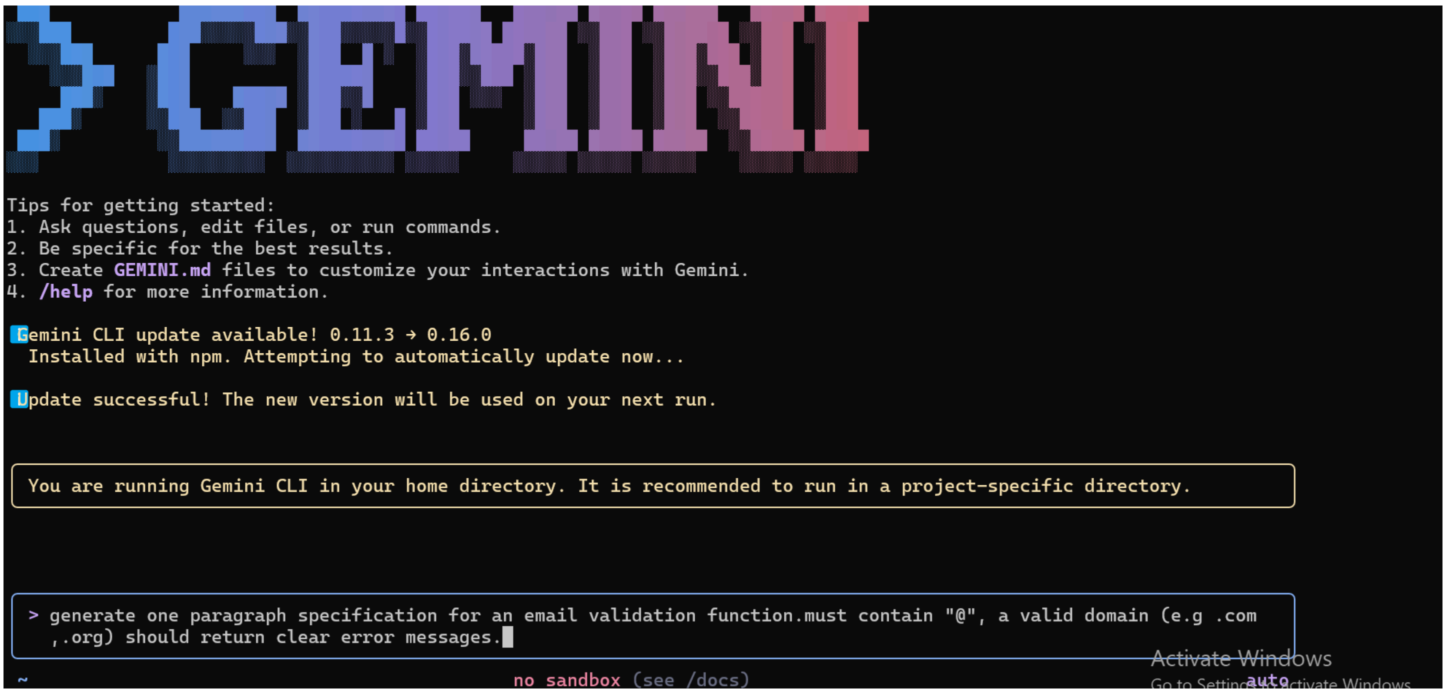
Using any AI CLI tool, generate a 1-paragraph specification for an email validation function.

Requirements:

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

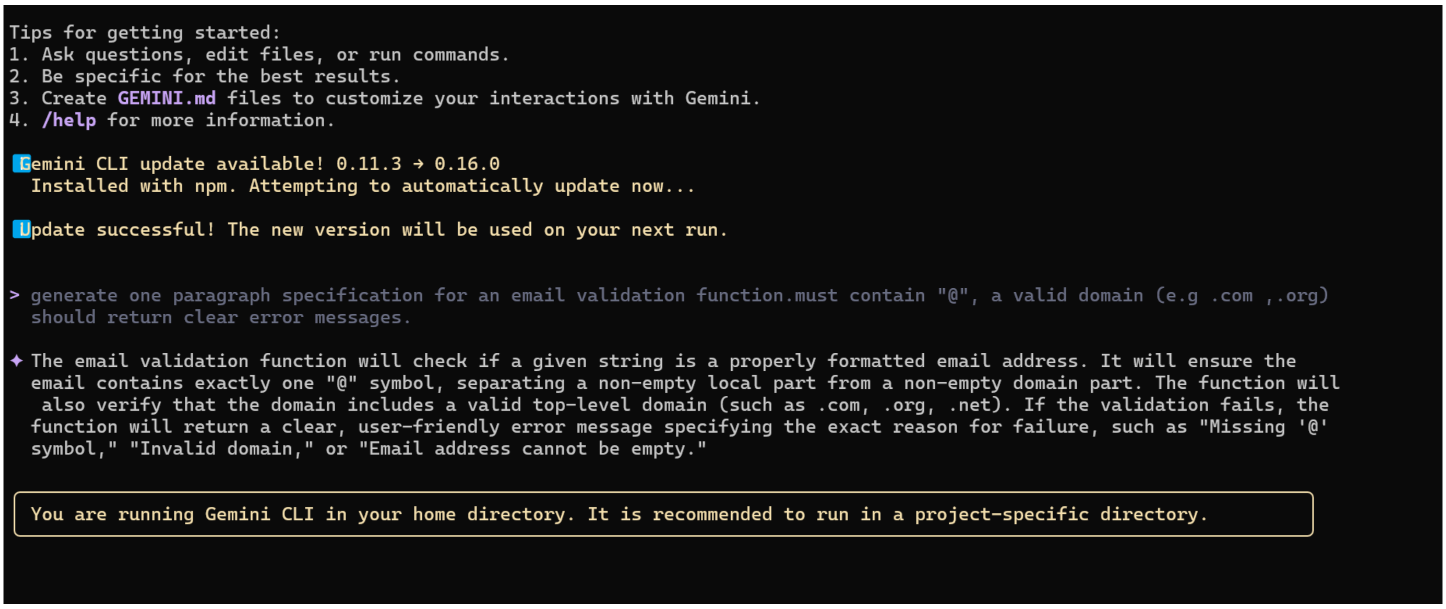
Submission:

1️⃣Your exact CLI prompt (text or 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! 0.11.3 → 0.16.0
 Installed with npm. Attempting to automatically update now...

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


 You are running Gemini CLI in your home directory. It is recommended to run in a project-specific directory.



 > generate one paragraph specification for an email validation function.must contain "@", a valid domain (e.g .com
   ,.org) should return clear error messages.
 ~                              no sandbox (see /docs)                              auto
```

2️⃣The 1-paragraph specification generated by the CLI



```
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! 0.11.3 → 0.16.0
 Installed with npm. Attempting to automatically update now...

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


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

✦ The email validation function will check if a given string is a properly formatted email address. It will ensure the
  email contains exactly one "@" symbol, separating a non-empty local part from a non-empty domain part. The function will
   also verify that the domain includes a valid top-level domain (such as .com, .org, .net). If the validation fails, the
  function will return a clear, user-friendly error message specifying the exact reason for failure, such as "Missing '@'
  symbol," "Invalid domain," or "Email address cannot be empty."

 You are running Gemini CLI in your home directory. It is recommended to run in a project-specific directory.
```

Part C — Multiple Choice Questions

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

A. Make coding faster
✅ B. Clear requirements before coding begins
C. Remove developers
D. Avoid documentation

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

A. Writing more code manually
✅ B. Thinking in systems and clear instructions
C. Memorizing more syntax
D. Working without any tools

3.Biggest failure of Vibe Coding?

A. AI stops responding
✅ B. Architecture becomes hard to extend
C. Code runs slow
D. Fewer comments written

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

A. They replace the developer completely
✅ B. Handle repetitive tasks so dev focuses on design & problem-solving
C. Make coding faster but less reliable
D. Make coding optional

5.What defines an M-Shaped Developer?

A. Knows little about everything
B. Deep in only one field
✅ C. Deep skills in multiple related domains
D. Works without AI tools