

AI-Driven Development – 30-Day Challenge

Part A – Theory (Short Questions)

1. Nine Pillars Understanding:

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

A- Using AI for boring setup work lets me focus on bigger system design. It saves time so I can learn architecture planning instead of just typing code.

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

A- The Nine Pillars give you AI tools for different domains like testing, planning, coding, and deployment. Instead of being expert in just one area, you can use these tools to work across multiple domains.

2. Vibe Coding vs Specification-Driven Development:

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

A- Vibe coding fails after one week because you're coding without planning or documentation. Initially, you remember everything, but after days pass, you forget why you made certain code decisions. The code becomes messy, hard to understand, and difficult to modify. When you need to add new features or fix bugs, you struggle because there's no clear structure or documentation to guide you.

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

A- SDD prevents these problems because you write clear requirements FIRST before coding. It's like having a blueprint before building a house. You always know what to build, how to build it, and what results to expect. The specifications act as your guide and documentation throughout the project.

3. Architecture Thinking:

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

A- Architecture-first thinking changes the developer from a **coder** to a **system designer**. Instead of starting with writing code, you start with designing the complete system structure - how components connect, data flows, and scalability. AI tools handle the implementation, so you focus on high-level planning and strategic decisions.

ii- Explain why developers must think in layers and systems instead of raw code?

A- Thinking in layers and systems is like building with LEGO blocks instead of carving from a single stone. Layers (like frontend, backend, database) make code modular, so you can change one part without breaking everything. Systems thinking ensures all parts work together smoothly. Raw code thinking creates a tangled mess that's hard to fix or improve.

Part B – Practical Task (Screenshot Required)

Task:

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

```
Windows PowerShell x + 
> GEMINI
> Generate a 1-paragraph specification for an email validation function that must contain "@".
Gemini CLI update available! 0.15.0 => 0.17.1
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 a 1-paragraph specification for an email validation function that must contain "@".
The validateEmail function will assess a string to confirm it represents a valid email format. It must verify that the input contains an '@' symbol and a valid domain extension like .com or .org. The function should also check that the domain includes a valid top-level domain (TLD) such as ".com" or ".org", which must be preceded by at least one character. For each specific validation failure, the function must return a clear and distinct error message indicating the exact issue, such as "Error: Email must contain an '@' symbol!" or "Error: Invalid or missing domain extension like .com". If all conditions are met, the function will return a success status, confirming the email's validity.

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

> Type your message or @path/to/file
no sandbox (see /docs) auto
```

1- Your exact CLI prompt

A- Generate a one-paragraph specification for an email validation function.

The function must check that the email contains an '@' symbol,

has a valid domain extension such as .com or .org,

and should return clear and specific error messages for invalid formats.

2- Generated Specification:

A- “The email validation function checks whether the provided email contains one '@' symbol, includes text before and after it, and ends with a valid domain extension such as .com or .org. If any requirement is missing, the function returns a clear and specific error message indicating the exact issue, such as missing

'@', invalid domain, or incorrect format. If all checks pass, the function returns a success response confirming the email is valid."

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 tool

Task 02 Completed!