

AIDD Task 2:

📁 Part A — Theory (Short Questions)

1. Nine Pillars Understanding:

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

A: Using AI development agents for repetitive tasks is a very strategic decision because it reduces our time to do the same tasks again and again and allows us to invest more time in decision making and debugging.

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

A: Nine pillars is necessary because every pillar gives developers a separate strong skill such as : using AI tools,creating AI agents for repetitive tasks, writing Specs,getting domain knowledge,VIBE coding. When these skills combine in one mind, the developer becomes a multi-skill and deep understanding geek.

Just like that developer not only does coding but can smartly integrate his knowledge in other domains and this thing is necessary for a M-shaped developer.

Vibe Coding vs Specification-Driven Development

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

A:Vibe Coding creates coding problems because initially there is no plan, the code becomes messy, and as the project grows, managing everything becomes difficult.

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

A:In an SDD, it is first clearly written what the system is supposed to do. This fixes the requirements before coding even begins. Result: less confusion, less rework, fewer bugs, and the team knows exactly what to build

Architecture Thinking

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

A: Architecture-first thinking transforms a developer's role from a coder to a system designer. In this approach, the developer first creates a clear system design and then writes the code. In AIDD, the developer essentially guides the AI with a strong design.

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

A: When you think only in raw code, everything becomes messy, slow, and confusing. Layers and systems help you divide the project into small, clear parts—like the UI layer, logic layer, and data layer. This makes understanding, testing, reusing, and fixing code 100x easier. System thinking teaches a developer to see the big picture, not just lines of code.

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

```
Windows PowerShell
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.

> write a single, developer focused paragraph outlining the requirements for an email validation function. the description should specify that the input must include an "@" symbol , contain a valid domain extension like .com or .org and provide clear, human readable error messages when validation does not pass. The paragraph should remain concise and, technical and appropriate for engineering use.

✦ Implement an isValidEmail(email: string): boolean function that validates email string formats. The function must ensure the input string contains a single @ symbol separating the local-part and domain, and that the domain includes a top-level domain (TLD) such as .com, .org, .net, etc., following a period. On failure, the function should return false and provide a distinct, human-readable error message (e.g., "Missing '@' symbol", "Invalid domain format", "Unsupported TLD") to facilitate user feedback and debugging.

> i was asking about to write a specification

✦ My apologies. Here is a more formal specification for the email validation function.

Specification: `validateEmail` Function

1. Function Signature

validateEmail(email: string): { isValid: boolean; error?: string; }

2. Description

This function validates an email address string based on a set of defined structural rules. It returns a result object indicating whether the email is valid and, if not, provides a clear, human-readable error message detailing the reason for failure.

3. Validation Rules

The input email string must adhere to the following criteria to be considered valid:

* Presence of "@" Symbol: The string must contain exactly one @ symbol.
* Local-Part and Domain: The @ symbol must separate a non-empty local-part and a non-empty domain part.
* Domain Structure: The domain part must contain at least one period (.).
* Top-Level Domain (TLD): The substring following the final period in the domain must correspond to a valid TLD (e.g., .com, .org, .net, .io). The check should be case-insensitive.

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
```

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

Name: Mohammad Danish

Roll No: 00037323

