AIDD-30-Day Challenge Task-2

Part-A : Theory (Short Questions)

1. Nine Pillars Understanding:

A. Why is using AI Development Agents (like Gemini CLI) for repetitive setup tasks better for your growth as a system architect?
Ans: AI Development Agents like Gemini CLI or Claude Code automate repetitive and boring setup tasks such as creating environment, installing dependencies and setting up file structure.

As AI handles all repetitive tasks then the developer focuses on high-level system thinking such as architecture design, logic planning, and integration structure.

Due to this growth **as a system architect** becomes fast because developer does not waste his time on repetitive tasks but spent his **energy on problem-solving and design**.

B. Explain how the Nine Pillars of AIDD help a developer grow into an M-Shaped Developer?
Ans: Traditional development required **separate specialists for multiple domains** such as frontend, backend, AI integration, testing, and deployment, but **Nine Pillars of AIDD** makes developers not only fast but make him **multi-skilled (M-shaped**) developer.

Because each pillar covers separate specific skill or infrastructure are, and when these works together then developer becomes a **system architect** who **handle all domains** such as frontend, backend, AI integration, testing, and deployment.

Example:
- AI CLI and Coding Agents handle repetitive tasks.
- Markdown make specifications executable as Programming Language.
- MCP plays role of a bridge between AI-First IDEs and AI tools.
- Linux and Cloud Deployment provides universal environment to the developer.

The **purpose of these pillars** is to make a such developer who build complete systems including different domains with the help of AI that will be fast, scalable and intelligent.

2. Vibe Coding vs Specification-Driven Development:

A. Why does Vibe Coding usually create problems after one week?
Ans:
Vibe coding means writing code that we fell right while writing the code without clear plan and specifications.
In the starting it may works perfect and looks fast way to build a software but after one or two weeks problems starts in the software working. Then code becomes very messy and duplicate logic etc. So it is very difficult for developer to understand the code, to debug the code or to add a new feature.

In simple words for a short term vibe coding feels fast and best method for software development but for a long term code starts showing problems and it becomes very difficult and nightmare for developer to maintain that code.
We can say vibe coding **“speed without structure”.**

B. How would Specification-Driven Development prevent those problems?
Ans:

Unlike Vibe coding, **Specification-Driven Development** (SDD) focuses on **clear planning** and **clear structure** for software development before starting to code.
That means first we define specifications or rules such as **"what will do this system"**, **"How it will do"**, **what will be sequence or flow?** Then with the help of AI Tools we start writing code on the basis of those specifications and rules.
It's benefit is that software features follow planned direction and code becomes organized and scalable. Developers can understand, debug and add new features easily event after a long time.

3. Architecture Thinking:

A. How does architecture-first thinking change the role of a developer in AIDD?
Ans:
Architecture-first thinking **means** that now the role of developer is not just only **writing a code** but his role becomes a **system designer**.

In AI-Driven Development, AI can write mostly raw code if developer will not tell AI, the structure and clear rules. So here the role of developer shifts and developer thinks, **how the parts of system will interact?**, **what will be the flow of data?**, and **how structure will be maintained?**.

Example: For creating a **Login Page**:
First I write code , HTML, CSS and JS.
Now I think how login system backend, database, and AI model will be connected.
That's why my role shifted from "coder" to "architect".

B. Explain why developers must think in layers and systems instead of raw code:
Ans:
Developers should must think in layers and systems instead of raw code because modern software is not just a simple code file but it is interconnected system of multiple layers such as frontend, backend, database, API, AI layer etc.

If developers only think of a raw code then overall flow and scalability will not be maintained.
If developers use layered thinking then each layer of software becomes independent , reusable and easier to manage and change later.

Example:

If I am building an e-comerce website, I will think of:

- **UI layer:** product display aur design
- **Logic layer**: add-to-cart aur checkout functions
- **Data layer:** Sanity CMS aur Stripe integration

In this approach system will run smoothly.

Part B — Practical Task (Screenshot Required)

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

By Muhammad Raheem Shar