Name : Javeria Roll number : 00065511

AI DRIVEN-DEVELOPMENT(AIDD) 30-DAYS CHALLENGE

TASK DAY 02

📁Part A — Theory (Short Questions)

NINE PILLARS UNDERSTANDING

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

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

Using AI Development Agents like Gemini CLI for repetitive setup tasks boosts your growth as a system architect by freeing you from manual work and letting you focus on high-level design. For example, agents can set up project folders, boilerplate code, or microservice templates automatically, so you can concentrate on system architecture, workflows, data flow, and service integration. Over time, this strengthens your architectural thinking as you guide AI agents and make strategic design decisions.

The Nine Pillars of AIDD help a developer grow into an M-Shaped developer by building both depth and breadth in skills. AI CLI and coding agents handle repetitive code, freeing you to focus on design and architecture. Markdown as Programming turns specifications into executable instructions, teaching structured thinking, while the MCP Standard ensures seamless integration across tools and platforms. AI-First IDEs improve collaboration with AI, and a Linux Universal Development Environment keeps work consistent across machines.

Test-Driven Development enforces quality and iterative improvement, and Spec-Driven Development structures projects for maintainability. Composable Vertical Skills let you reuse domain expertise, expanding your knowledge across different areas, and Universal Cloud Deployment teaches standardized deployment and infrastructure management.

Together, these pillars give you deep technical expertise while broadening your understanding of tools, processes, and systems, enabling you to design, build, and deploy complex systems independently.

VIBE CODING VS SPECIFICATION-DRIVEN DEVELOPMENT

Why does Vibe Coding usually create problems after one week?

How would Specification-Driven Development prevent those problems?

Vibe Coding usually creates problems after one week because it lacks clear structure and planning. Developers or AI just follow their intuition or mood, writing code without defined specifications. Initially, it may seem fast, but as the project grows:

- Code becomes messy and inconsistent.
- Bugs and errors increase, making debugging difficult.
- Features may conflict with each other.
- Maintaining or updating the system becomes time-consuming.

In short, without a clear plan, short-term speed turns into long-term chaos.

Specification Driven Development prevents those problems because you define the structure of the system before any coding begins. When every feature, flow, and rule is written clearly in the specification, the AI or developer follows a consistent plan instead of guessing. For example, if you are building a coffee shop app, the spec defines how the menu works, how orders flow, and how payments are handled. This keeps the code organized and reduces conflicts. If you remember a new requirement later, you simply update the spec and generate clean updates. This approach stops the chaos that comes from unplanned coding and keeps the project stable.

ARCHITECTURE THINKING

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

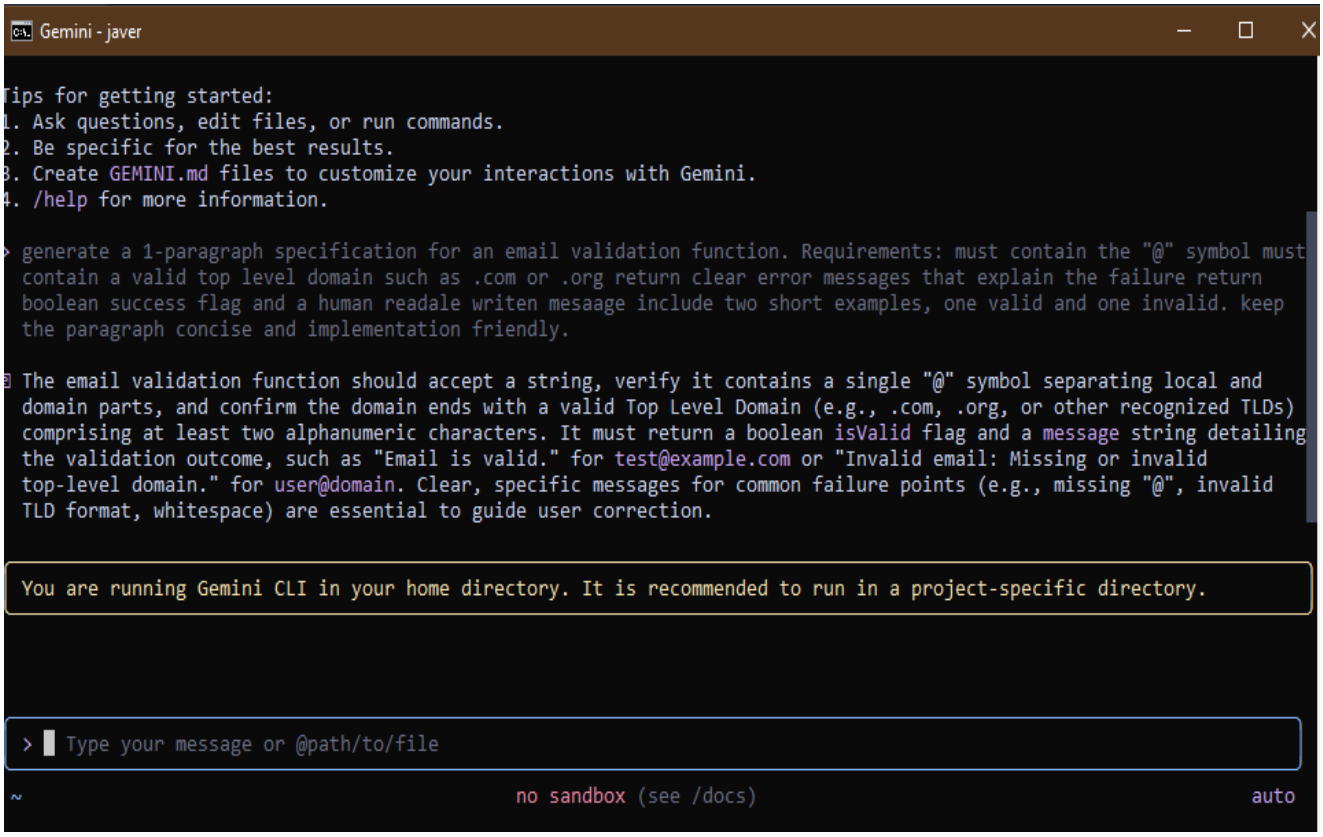
Explain why developers must think in layers and systems instead of raw code.

Architecture first thinking changes the role of a developer in AIDD because the developer stops acting like a manual coder and starts acting like a system designer. The focus moves to planning flows, defining components, and guiding AI agents instead of writing every line by hand. The developer makes decisions about structure, data movement, reliability, and integration. AI handles routine code generation, so the developer becomes responsible for the shape of the system, not just the syntax.

Developers must think in layers and systems instead of raw code because modern applications work through connected parts. Each layer handles a different job, like frontend, backend, data, and infrastructure. If you only think in raw code, you miss how these layers interact. Thinking in systems helps you plan how components communicate, how data flows, and how the app scales. This leads to cleaner design, fewer errors, and better use of AI generated code.

📁 Part B — Practical Task

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



📁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