Part A — Theory (Short Questions)

1. Nine Pillars Understanding

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

Answer: AI Development Agents for repetitive setup tasks are better for your growth as a system architect because system architect's main responsibility is to plan systems, design architects, improve workflow and solve complex problems. Setting up the project is a repetitive task, and it always takes time to install respective packages and configure environments for the project. So, I think using AI Development Agents is a better option for doing repetitive tasks because the time system architect uses for setup that can be utilized to make the project better by improving the design and workflow. And it will help you to grow faster and work more efficiently.

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

Answer: There are three types of Developers.

L-Shaped Developers: L-Shaped Developers are those who have deep expertise in one area.

T-Shaped Developers: T-Shaped Developers are those who have little knowledge about everything.

M-Shaped Developers: M-Shaped Developers are those who have wide range of skills in multiple domains, along with deep expertise in two to three of them.

These Nine pillars of AIDD help a developer to grow into a M-Shaped developer in a way that every pillar of AIDD strengthens the different parts of developer's abilities.

1) **AI CLI with Smart Coding Agent:** This pillar helps the developers to work faster with the help of coding Agent.
2) **Markdown as a Programming Language:** This pillar helps the developers to write documentation, and specification because this AIDD is all about writing good structure and plan of the project.
3) **MCP Standard:** This pillar helps the developers to do the tasks of backend like API fetching, storage and data flow.
4) **AI-First IDEs:** this pillar is all about the Integrated Development Environment in which these AI tolls are built-in like Cursor or Zed.
5) **Linux Universal Development Environment:** This pillar provides a consistent and reliable development environment across machines and platforms.
6) **Test-Driven Development:** TDD emphasizes writing tests before code. It ensures that each part of the system works correctly, reduces bugs, and makes the codebase more reliable and maintainable over time.
7) **Specification-Driven Development with SpecKit Plus:** This pillar focuses on writing detailed specifications before coding.

8) **Composable Vertical Skills:** This pillar allows developers to assemble specialized tools and AI skills for specific industries or domains.
9) **Universal Cloud Deployment:** This pillar enables developers to deploy applications anywhere in the cloud efficiently.

2. Vibe Coding vs Specification-Driven Development

Question: Why does Vibe Coding usually create problems after one week?
Answer: Vibe coding is all about writing code without a proper plan and structure. At the start, this approach seems efficient and fast, but it will create many problems after some time. Because it's built on guessing instead of planning. This vibe coding is all about the instant vibe that comes at that time and when the time changes it feels like changing this or that thing after changing, this will create bugs in the programming which will be a headache for a developer.

Question: How would Specification-Driven Development prevent those problems?
Answer: Spec-Driven Development fixes the problems of vibe coding by giving the project clear structure and direction from the start. Instead of guessing, you write detailed specifications that explain exactly how a feature should work. This helps AI generate accurate code, keeps the project organized, makes debugging easier, and allows the system to grow without breaking. Overall, it turns messy, short-term coding into a clean, predictable, and scalable development process.

3. Architecture Thinking

Question: How does architecture-first thinking change the role of a developer in AIDD?
Answer: Architecture-first thinking changes a developer's role in AIDD by shifting their focus from writing code to designing the system's structure. Instead of acting as a coder, the developer becomes a system architect who plans the workflow, defines specifications, and guides AI agents to build the actual implementation. This makes the developer responsible for clarity, scalability, and overall system quality—not just code.

Question: Explain why developers must think in layers and systems instead of raw code.
Answer: In AI-Driven Development, thinking in layers means organizing a software system into distinct levels, each responsible for a specific job, rather than mixing everything into raw code. When using AI coding agents, these layers become even more important because the AI executes tasks according to the structure and instructions you provide.
Layer 1: Frontier Models (The Intelligence):
This layer is large Language Model, and it is all about thinking. When you ask for code suggestions, explanations, or debugging help, the frontier model processes your request and generates responses. It is accessed through layer 2 because we need an IDE for using this layer.
Layer 2: AI-First IDEs (The Workspace):

This layer provides an environment for a developer to write code and interact with AI.

Layer 3: Development Agents (The Orchestration):
This layer handles complex, multi-step workflows: refactoring entire modules, running tests, creating pull requests, debugging across multiple files

Part B — Practical Task (Screenshot Required)

Part C — Multiple Choice Questions

1) What is the main purpose of Spec-Driven Development?
Ans: Clear requirements before coding begins

2) What is the biggest mindset shift in AI-Driven Development?
Ans: Thinking in systems and clear instructions

3) Biggest failure of Vibe Coding?
Ans: Architecture becomes hard to extend

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

Ans: Handle repetitive tasks so dev focuses on design & problem-solving

5) What defines an M-Shaped Developer?

Ans: Deep skills in multiple related domains