AIDD 30 DAYS-CHALLENGE (TASK-02)

By **SUNAINA ISMAIL**

Part A – Theory (Short Questions)

1. Nine Pillars Understanding

- **Why is using AI Development Agents(like Gemini CLI) for repetitive setup tasks better for your growth as a system architect?**
  AI agents are better for setup tasks because they can handle repetitive work quickly and accurately. For example, they can create project folders, install software libraries, and generate configuration files without mistakes. Humans might forget steps or make small errors, but AI agents do everything correctly every time. This saves a lot of time and helps developers focus on coding the main features instead of spending hours on setup. AI agents can also check for missing files or errors automatically, making the development process smoother.

- **B. Explain how the Nine Pillars of AIDD help a developer grow into an M-shaped Developer?**
  The Nine Pillars help developers become “M-shaped,” meaning they are skilled in many areas but also deep in some. For example, a developer can understand coding, testing, system design, and teamwork. Being M-shaped allows a developer to solve complex

problems because they can combine knowledge from different areas. It also helps them work better with AI tools and other team members, as they can think about the project as a whole rather than only focusing on one small part.

2. Vibe Coding vs. Specification-Driven Development (SDD)

- **Why does Vibe Coding usually create Problems After One Week?**
  Vibe coding is when a developer writes code based on intuition or ideas, without a clear plan. At first, it may feel fast, but after a few days, problems appear. The code can become messy, inconsistent, and full of errors. For example, one function might not work with another, or some features might be missing. It is also harder for other developers to understand the code, which makes teamwork difficult. Overall, it slows down the project in the long run.

- **How does SDD Prevents These Problems?**
  Specification-Driven Development (SDD) solves these problems by planning everything before coding. The spec explains exactly what each function should do, what input it takes, what output it should give, and how it should behave in different situations. For example, if we are creating a login system, the spec would say how to check the password, handle errors, and what messages to show. This makes the code clean, organized, and easy to test. It also helps teams work together because everyone knows what to expect from the code.

3. Architecture Thinking

- **How does Architecture-First Thinking Changes the Developer Role in AIDD?**
  Architecture-first thinking makes developers think like system designers, not just coders. Before writing code, they plan how each part of the system connects and works together. For example, in a shopping website, developers need to design how the product page, database, checkout system, and payment system communicate. This ensures the system is stable, scalable, and easier to update. Developers become more responsible for the system as a whole, rather than only small parts.

- **Explain Why Developers Must Think in Layers and systems instead of raw code?**
  Developers must think in layers and systems because it helps create software that is organized, reliable, and easy to improve over time. When a system is divided into clear layers such as the user interface, business logic, and database each layer has its own specific job. This separation prevents one change from accidentally breaking something in another part of the system. Layered thinking also stops developers from writing “raw code” where everything is mixed together, which often leads to confusion and errors. By thinking in systems, developers can plan how different parts will connect, grow, and work together smoothly. This approach makes debugging easier, improves

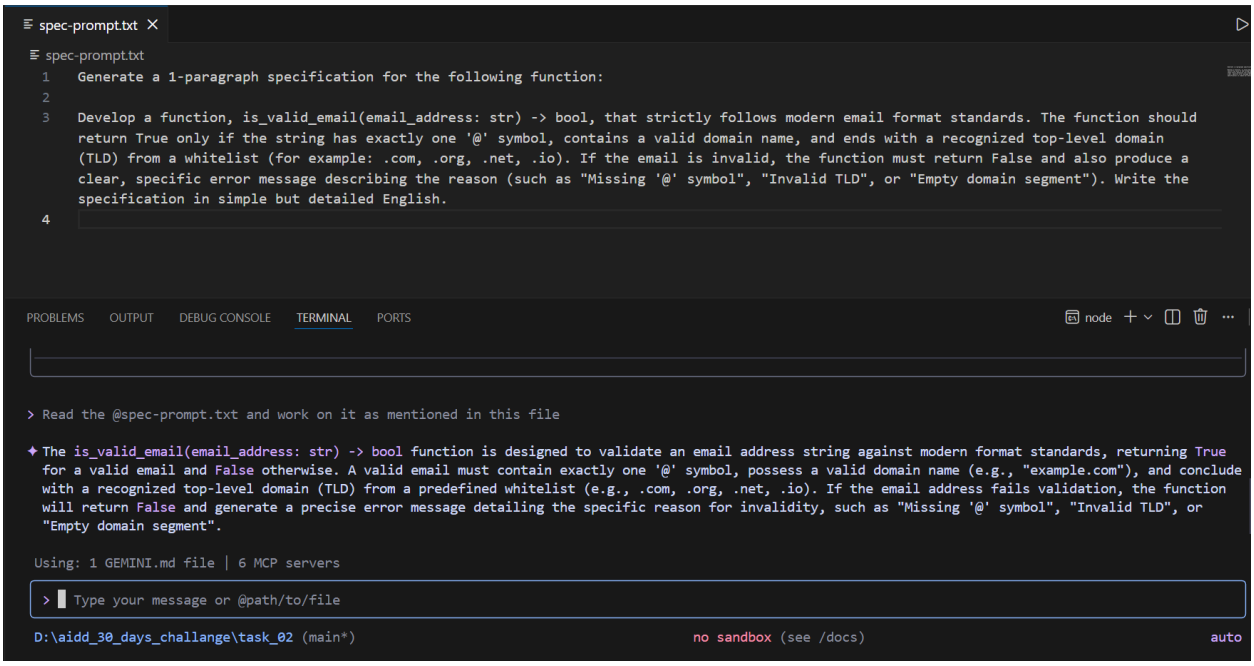
Part B – Practical Task

- My Prompt:
  Generate a 1-paragraph specification for the following function:
  Develop a function, is_valid_email(email_address: str) -> bool, that strictly follows modern email format standards. The function should return True only if the string has exactly one '@' symbol, contains a valid domain name, and ends with a recognized top-level domain (TLD) from a whitelist (for example: .com, .org, .net, .io). If the email is invalid, the function must return False and also produce a clear, specific error message describing the reason (such as "Missing '@' symbol", "Invalid TLD", or "Empty domain segment"). Write the specification in simple but detailed English.

- AI-Tool Output:(Gemini-CLI)
  The is_valid_email(email_address: str) -> bool function is designed to validate an email address string against modern format standards, returning

True for a valid email and False otherwise. A valid email must contain exactly one '@' symbol, possess a valid domain name (e.g., "example.com"), and conclude with a recognized top-level domain (TLD) from a predefined whitelist (e.g., .com, .org, .net, .io). If the email address fails validation, the function will return False and generate a precise error message detailing the specific reason for invalidity, such as "Missing '@' symbol", "Invalid TLD",or"Empty domain segment".

SCREENSHOT:



Part C – Multiple Choice Questions

1. What is the main purpose of Spec-Driven Development
   Correct answer B
2. What is the biggest mindset shift in AI-Driven Development
   Correct answer B
3. Biggest failure of Vibe Coding
   Correct answer B
4. Main advantage of using AI CLI agents like Gemini CLI
   Correct answer B
5. What defines an M Shaped Developer
   Correct answer C