🌐 AI-Driven Development — 30-Day Challenge
📝 Task 2

📁 Part A — Theory (Short Questions)

Q1. 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.

Answer Q 1.

- When you use same configuration in every project, it consumes your time and mental energy. AI does all this in seconds, and you can focus on system architecture and decision making. There is a higher change of mistakes in manual repetitive work. AI gives you a steady and reliable setup, so you can focus on keeping your architecture clean and your project easy to maintain.
- The Nine Pillar of AIDD give developers a balanced mix of AI Skills, Data Skills, Development Skills and Soft Skills. Because of this mix, a Developer becomes the capable of many different areas, not just one.

Q2. Vibe Coding vs Specification-Driven Development

- Why does Vibe Coding usually create problems after one week?
- How would Specification-Driven Development prevent those problems?

Answer Q 2.

- Vibe Coding usually create after one week because it focuses on coding without planning. The code works but hard to change or add new feature later. Other developer may not know what the code does. Skipping proper testing means problems show up after few days.
- Specification-Driven Development prevent those problems by planning everything before writing code. You write detailed specifications like what the code should do, how it should work, before starting. Everyone knows what to build, so code is understandable. Clear requirement help developers to avoid mistake and do proper test.

Q3. 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.

Answer Q 3.

- In AIDD (AI-Integrated Driven Development) developer don’t just write code. They plan the system first and then use AI Tools to help build it. This way the coding is clean, easy to change and work well for the long time.
- Breaking the system into layers like UI, Logic, Database keep the code neat and easier to manage. Different developer can work on different layers at the same time without breaking the whole system.

📁 Part B — Practical Task (Screenshot Required)

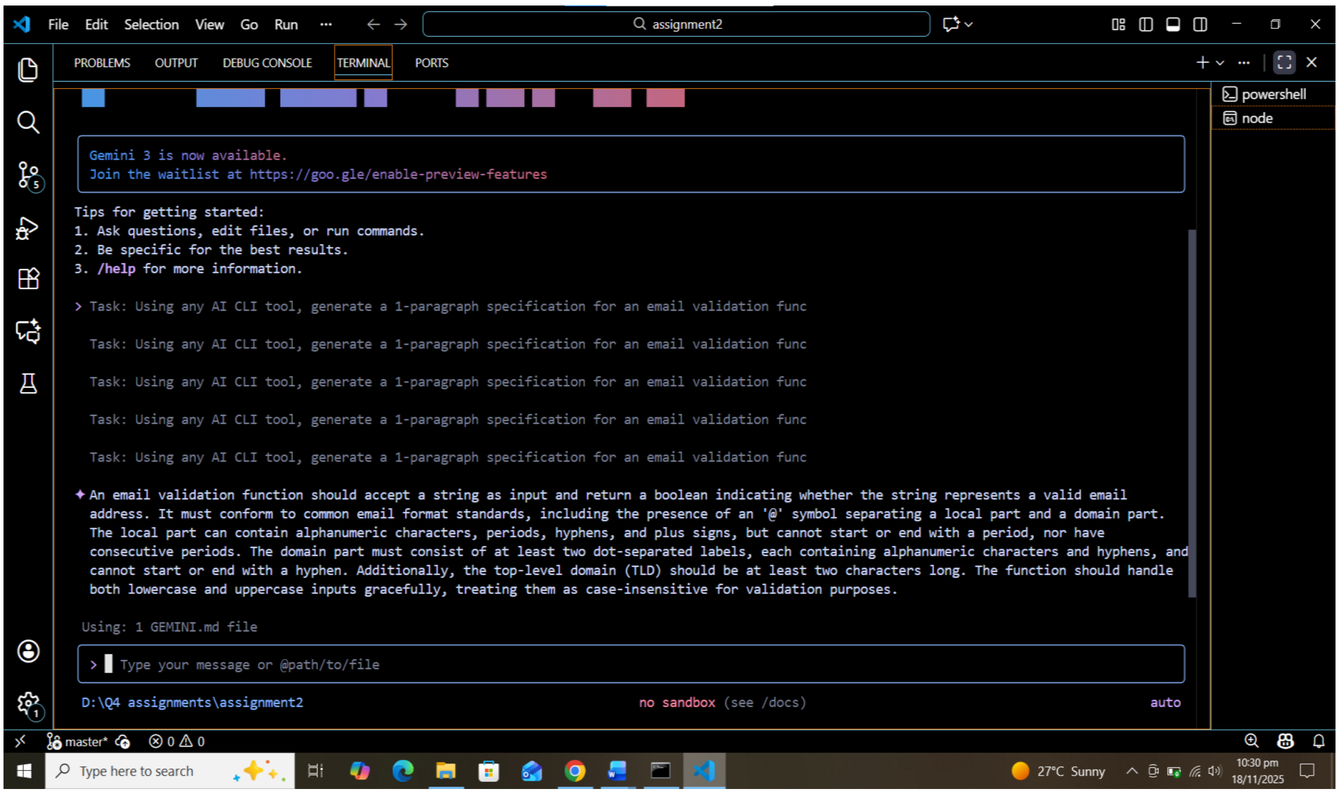
Task: Using any AI CLI tool, generate a 1-paragraph specification for an email validation function.

Requirements:

- **Must contain “@”**
- **Must contain a valid domain (e.g., .com, .org)**
- **Should return clear error messages**

Submission:

1. **Your exact CLI prompt (text or screenshot)**



```
Gemini 3 is now available.
Join the waitlist at https://goo.gle/enable-preview-features

Tips for getting started:
1. Ask questions, edit files, or run commands.
2. Be specific for the best results.
3. /help for more information.

> Task: Using any AI CLI tool, generate a 1-paragraph specification for an email validation func

  Task: Using any AI CLI tool, generate a 1-paragraph specification for an email validation func

  Task: Using any AI CLI tool, generate a 1-paragraph specification for an email validation func

  Task: Using any AI CLI tool, generate a 1-paragraph specification for an email validation func

  Task: Using any AI CLI tool, generate a 1-paragraph specification for an email validation func

✦ An email validation function should accept a string as input and return a boolean indicating whether the string represents a valid email
  address. It must conform to common email format standards, including the presence of an '@' symbol separating a local part and a domain part.
  The local part can contain alphanumeric characters, periods, hyphens, and plus signs, but cannot start or end with a period, nor have
  consecutive periods. The domain part must consist of at least two dot-separated labels, each containing alphanumeric characters and hyphens, and
  cannot start or end with a hyphen. Additionally, the top-level domain (TLD) should be at least two characters long. The function should handle
  both lowercase and uppercase inputs gracefully, treating them as case-insensitive for validation purposes.

 Using: 1 GEMINI.md file

 > Type your message or @path/to/file

 D:\Q4 assignments\assignment2                    no sandbox (see /docs)                    auto
```

2. **The 1-paragraph specification generated by the CLI**

Paragraph

✦ An email validation function should accept a string as input and return a boolean indicating whether the string represents a valid email address. It must conform to common email format standards, including the presence of an '@' symbol separating a local part and a domain part. The local part can contain alphanumeric characters, periods, hyphens, and plus signs, but cannot start or end with a period, nor have consecutive periods. The domain part must consist of at least two dot-separated labels, each containing alphanumeric characters and hyphens, and cannot start or end with a hyphen. Additionally, the top-level domain (TLD) should be at least two characters long. The function should handle both lowercase and uppercase inputs gracefully, treating them as case-insensitive for validation purposes.

📁 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