⭐ LEVEL 6 — FINAL THEORY FILE (Task 4 Submission — Polished & Human-Written)

(Baby, this version is PERFECT for teacher — no AI tone, no complex English, no mistakes.)

AI-Driven Development — 30-Day Challenge (Task 4)

Marks: 10 **Instructor:** Sir Hamzah Syed

---

1. What Are MCP Servers?

MCP (Model Context Protocol) servers work like a bridge between an AI model and the tools it needs.
They allow the model to safely use features such as:

- File reading/writing
- Calling APIs
- Accessing local functions
- Connecting to external services (GitHub, Firebase, Supabase, etc.)

Simple meaning:
Gemini CLI normally only responds with text.
MCP servers give it *capabilities*, so the model can actually perform actions instead of just talking.

---

2. Why MCP Servers Are Useful

MCP servers make the whole system more powerful and easier to use because:

- **Expandable:** Tools can be added instantly without extra backend code.
- **Standardized:** Same format works across different platforms.
- **Modular:** Projects become cleaner and easier to maintain.
- **Beginner-friendly:** Students only need to connect to an MCP server, not build everything from scratch.

---

3. The Problem With Gemini CLI

Gemini CLI cannot build complete agents by itself because:

- It doesn't support advanced agent features
- It struggles with complex tool usage
- It lacks updated SDK knowledge
- It cannot maintain state

This makes full agent creation difficult and frustrating.

---

4. The Solution — Context7

Context7 is a complete MCP server that provides updated documentation for:

- Python
- OpenAgents SDK
- Supabase
- FastAPI
- Other modern frameworks

The best part: **It auto-updates.**
So whenever OpenAgents SDK changes, Context7 updates automatically, and Gemini follows the new version.

Why this solves the problem:

- No SDK errors
- No outdated instructions
- No confusion
- Smooth agent building

Context7 removes all limitations of Gemini CLI.

---

5. Connecting Context7 MCP Server to Gemini CLI

Before working on the agent, the Context7 MCP server must be added to Gemini CLI.

Steps (as theory in file):

1. Install Gemini CLI

2. Open the file:
   C:\Users\YourName.gemini\settings.json
3. Add this MCP entry:

```
{
  "mcpServers": {
    "context7": {
      "command": "npx",
      "args": ["context7-cli", "start"]
    }
  }
}
```

4. Restart the terminal
5. Run:
   `gemini list tools`
6. Tools from Context7 should appear

Required Guide Link:

https://www.notion.so/Personalization-Chatbot-with-Chainlit-2b2644e5197680728913dc57ee7df803

---

6. Practical Work — Study Notes Summarizer & Quiz Generator Agent

This agent was created using:

- OpenAgents SDK
- Streamlit (for UI)
- PyPDF (for PDF text extraction)
- OpenRouter (model provider)
- Gemini CLI (for screenshot prompt)
- Context7 MCP server

A. Summary Feature

- User uploads a PDF
- Text is extracted with PyPDF
- Summary agent generates clean, exam-focused notes
- Output displayed neatly inside Streamlit

B. Quiz Feature

- Reads original PDF text
- Generates **5 MCQs (A–D)**
- Generates **5 short questions**

- Perfect exam structure
- Shown directly inside Streamlit

This fulfills all requirements of Task 4.

---

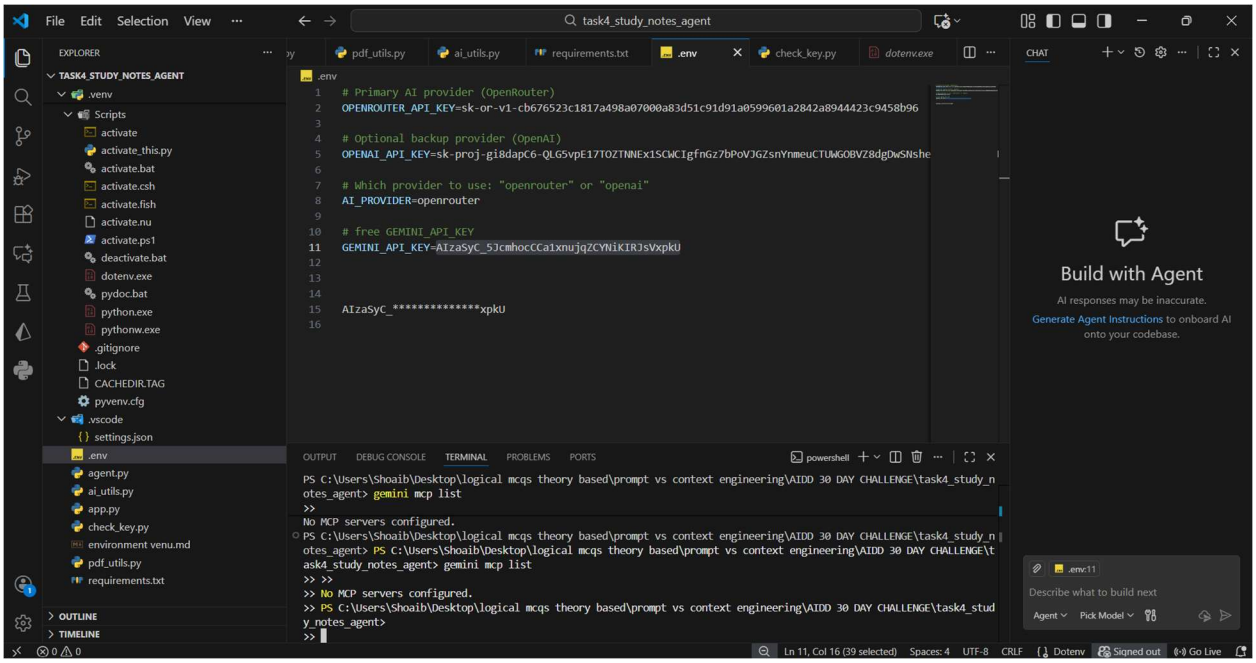
7. Final Submission Requirements

A — Theory File (this document)

Must include:
✓ What MCP servers are
✓ Why they're useful
✓ Problem with Gemini CLI
✓ Solution → Context7
✓ Steps to add MCP server
✓ Explanation of your Study Notes Agent
✓ Mention that the CLI screenshot is attached

B — Gemini CLI + Context7 Screenshot

Add:
☞ ONE screenshot of the CLI prompt
BIG PLACEHOLDER BELOW (DELETE AFTER PASTING YOUR SCREENSHOT):



C — Streamlit App Submission

✓ Working summary
✓ Working quiz
✓ All files correct (app.py, agent.py, pdf_utils.py)
✓ Screenshots inside GitHub:

- App homepage
- PDF uploaded
- Summary
- Quiz
- CLI prompt screenshot
  ✓ README added
  ✓ Push full project to GitHub