TASK-04

NAME : MUHAMMAD RIZWAN.
SLOT : FRIDAY (6:00 TO 9:00).
INSTRUCTER : SIR HAMZA SYED.

AI-DRIVEN DEVELOPMENT-30-DAYSCHALLENGE.

1. What Are MCP Servers?

MCP (Model Context Protocol) servers are special servers that work as a **bridge** between your AI model and the external tools it can use.
They allow your AI model or CLI to safely access things like:

- Files
- APIs
- Local functions
- Databases
- External systems (GitHub, Firebase, Supabase, etc.)

In simple terms:
An MCP server gives tools to your AI model so it can DO things instead of only talking.

---

2. Why MCP Servers Are Useful?

- You can instantly add new abilities to an AI model
- It works with a standard format
- No need to manually connect every tool
- Makes the system modular and easy to maintain
- Students don't need deep backend coding
- Just connect the server → AI model becomes more powerful

---

3. The Problem

Gemini CLI cannot build full agents itself.
It lacks strong agent building support.
So students face errors, confusion, incorrect SDK usage, and outdated workflows.

---

4. The Solution Context7

- Context7 is a **complete MCP server** that provides:
- Python documentation
- OpenAgents SDK documentation
- Supabase docs
- FastAPI docs
- All modern frameworks
- Auto updating docs

So Gemini CLI always works with the latest documentation
No outdated methods or errors
Perfect for building agents with OpenAIgents SDK

5.Task 4 Connect Context7 MCP Server to Gemini CL

---

6. Practical Task Study Notes Summarizer & Quiz Generator Agen

---

✅ Project Folder Structure (for GitHub)

```
/study-notes-agent
 ├── app.py
 ├── agent.py
 ├── pdf_utils.py
```

```
├── requirements.txt
├── README.md
├── screenshots/
│      └── gemini_prompt.png
```

Requirements.txt

```
streamlit
pypdf
openagents
google-generativeai
```

Pdf utils.py

```
from pypdf import PdfReader
def extract_pdf_text(pdf_file):
    reader = PdfReader(pdf_file)
    text = ""
    for page in reader.pages:
        text += page.extract_text() + "\n"
    return text
```

---

agent.py

Using OpenAgents SDK Context7 MCP

```
from openagents import Agentimport google.generativeai as genai
# Gemini API key (use environment variable)
genai.configure(api_key="YOUR_API_KEY")

agent = Agent(
    name="pdf_study_agent",
    instructions="""
You are a Study Notes Summarizer & Quiz Generator Agent.

You must:
1. Extract content from PDF text.
2. Generate a clean, structured summary.
3. Create MCQs or mixed quizzes based on the original PDF.
""",
```

```
    model="gemini-1.5-flash"
)
def generate_summary(text):
    prompt = f"""
Summarize the following PDF text into clean, structured notes:
{text[:15000]}
"""
    return agent.run(prompt)
def generate_quiz(text):
    prompt = f"""
Create a quiz using the following PDF content.
Include:
- 5 MCQs
- 5 True/False
- 5 Short Questions

PDF Content:{text[:15000]}
"""
    return agent.run(prompt)
```

---

app.py (Streamlit UI)

✔ PDF Upload
✔ Summary Display
✔ Quiz Button

```
import streamlit as stfrom pdf_utils import extract_pdf_textfrom
agent import generate_summary, generate_quiz

st.title("   Study Notes Summarizer & Quiz Generator Agent")

uploaded_file = st.file_uploader("Upload a PDF", type=["pdf"])
if uploaded_file:
    st.success("PDF uploaded successfully!")
    pdf_text = extract_pdf_text(uploaded_file)

    if st.button("Generate Summary"):
        with st.spinner("Summarizing..."):
            summary = generate_summary(pdf_text)
        st.subheader("   Summary")
        st.write(summary)

    if st.button("Generate Quiz"):
        with st.spinner("Generating Quiz..."):
            quiz = generate_quiz(pdf_text)
        st.subheader("? Quiz")
```

```
st.write(quiz)
```