Member A – Prompt Engineering + AI Logic

✅ **Tasks:**

1. Design Effective Prompts

- Create a prompt that asks the AI model to:
    - Generate 3 MCQs with 4 options each and specify the correct answer.
    - Generate 3 flashcards with question-answer format.
- Save the prompt function as: build_prompt(summary_text)

2. Implement AI Integration

- Create the function call_ai_model(prompt) to send the prompt to Gemini or OpenAI.
- Place these functions in:
  utils/quiz_generator.py

3. Unit Test Prompt & Response

- Test call_ai_model() using a sample summary.
- Confirm the returned structure includes both "mcqs" and "flashcards" keys with expected data format.

👤 **Member B – Flask API + Postman Testing**

✅ **Tasks:**

1. Create Flask Endpoint

- Add to app.py:
    - Create POST route /generate_quiz
    - Accept JSON with key "summary"
    - Call Member A's build_prompt() and call_ai_model()
    - Return response as JSON

2. Postman API Testing

- Use Postman to test:

    - URL: http://localhost:5000/generate_quiz
    - Method: POST
    - Header: Content-Type: application/json
    - Body:

json

CopyEdit

```
{
  "summary": "sample summarized lecture text"
}
```

3. Edge Case Handling

- Test what happens if:
    - Summary field is missing
    - Input is very short or empty
    - Input is irrelevant or random

📦 **Final Joint Deliverable**

- ✅ Fully working /generate_quiz API endpoint
- ✅ Prompt logic written and tested with an AI model
- ✅ Output format: 3 MCQs + 3 flashcards
- ✅ Proper error handling (missing/empty input)
- ✅ Postman tests confirming output format