

PROJECT TITLE: AI Study Assistant Chatbot

SUBJECT: Generative ai chatbot

SUBMITTED BY:

Name: ABIN PANICKER

INSTITUTION:MES INSTITUTE OF TECHNOLOGY AND MANAGEMENT,CHATHANNOOR

SUBMITTED TO: Dev Town

MONTH & YEAR:

ABSTRACT

The AI Study Assistant Chatbot is an intelligent conversational system developed using FastAPI and integrated with a Large Language Model (LLM) through the Groq API. The chatbot is designed to answer academic and study-related questions while maintaining conversation history using MongoDB.

The system stores user interactions in a database and retrieves previous conversations to generate context-aware responses. This project demonstrates how modern AI assistants maintain conversational memory and persist user data in real-world applications.

INTRODUCTION

Artificial Intelligence-powered chatbots are widely used in educational platforms to assist students in learning. However, many basic chatbots do not maintain conversational memory.

This project aims to build a Study Bot that:

- Answers academic questions
- Maintains context across conversations
- Stores user data in a database
- Provides API-based access

The system uses MongoDB for memory persistence and an LLM API for generating intelligent responses.

OBJECTIVES

The main objectives of this project are:

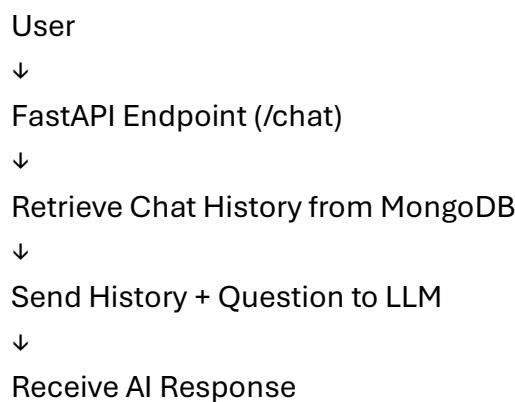
1. To build an AI-powered Study Assistant.
2. To integrate a Large Language Model (LLM) API.
3. To store chat history in MongoDB.
4. To retrieve previous conversations for contextual responses.
5. To deploy the application as a REST API.

TECHNOLOGIES USED

Technology	Purpose
Python	Backend development
FastAPI	API framework
Uvicorn	ASGI server
LangChain	LLM integration
Groq API	Language Model provider
MongoDB Atlas	Cloud database
python-dotenv	Environment variable management

SYSTEM ARCHITECTURE

System Workflow:



```
↓  
Store Conversation in MongoDB  
↓  
Return Response to User
```

This architecture ensures both intelligence and memory persistence.

MEMORY IMPLEMENTATION

Step 1: Store Conversations

Each user message and AI response is stored in MongoDB with:

- user_id
- role (human or ai)
- message
- timestamp

Example:

```
collection.insert_one({  
    "user_id": request.user_id,  
    "role": "human",  
    "message": request.question,  
    "timestamp": datetime.utcnow()  
})
```

Step 2: Retrieve Chat History

Before generating a new response, the chatbot retrieves previous messages:

```
def get_chat_history(user_id):  
    chats = collection.find({"user_id": user_id}).sort("timestamp", 1)
```

These messages are passed to the LLM as conversation history.

Step 3: Context Injection

The history is injected into the prompt template:

```
response = chain.invoke({  
    "history": history,  
    "question": request.question  
})
```

This allows the chatbot to understand previous context and respond intelligently.

This demonstrates real-world conversational memory implementation.

API ENDPOINTS

GET /

Used to check if the API is running.

Response:

AI Study Assistant API is running!

POST /chat

Used to send study-related questions.

Request Body:

```
{  
    "user_id": "student1",  
    "question": "What is Newton's First Law?"  
}
```

Response:

```
{  
    "response": "Explanation of Newton's First Law..."  
}
```

STUDY BOT LOGIC

The chatbot includes a system prompt:

"You are an AI Study Assistant. You answer academic and learning-related questions clearly and simply. Use previous conversation context when available. If the question is not study-related, politely guide the user back to academic topics."

This ensures:

- Academic focus
- Clear explanations
- Controlled responses

DEPLOYMENT

The application was deployed using Render.

Deployment Steps:

1. Upload code to GitHub.
2. Connect repository to Render.
3. Set environment variables:
 - GROQ_API_KEY
 - MONGODB_URI
4. Start command:
`unicorn app:app --host 0.0.0.0 --port 10000`

After deployment, the API is accessible via a public URL.

TESTING AND RESULTS

The chatbot was tested using Swagger UI.

Tests performed:

- ✓ Study-related question
- ✓ Follow-up question (memory test)
- ✓ MongoDB data verification
- ✓ API response validation

The chatbot successfully:

- Generated intelligent answers
- Retrieved previous context
- Stored chat history correctly

SCREENSHOTS

The screenshot shows the AI Study Assistant - Swagger UI interface. The URL in the browser is `127.0.0.1:8000/docs#/default/chat_chat_post`. The main area displays the `POST /chat` endpoint. Under the `Request body` section, the schema is defined as:

```
{ "user_id": "student1", "question": "What is Newton's First Law?" }
```

At the bottom, there are `Execute` and `Clear` buttons.

The screenshot shows the detailed response for the `POST /chat` endpoint. It includes:

- Curl:**

```
curl -X 'POST' \
'http://127.0.0.1:8000/chat' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{ "user_id": "student1", "question": "What is Newton's First Law?" }'
```
- Request URL:** `http://127.0.0.1:8000/chat`
- Server response:**
 - Code:** 200
 - Response body:**

```
{ "response": "**Newton's First Law (Law of Inertia)** \n\n>An object will stay at rest or keep moving in a straight line at constant speed unless an external force acts on it.\n\n>In other words, a body does not change its state of motion on its own; it needs a net external force to accelerate, decelerate, or change direction."
```
 - Response headers:**

```
access-control-allow-credentials: true
access-control-allow-origin: *
content-length: 334
content-type: application/json
date: Sun, 22 Feb 2026 17:45:59 GMT
server: uvicorn
```
- Responses:**
 - Code:** 200
Description: Successful Response
Media type: application/json

The screenshot shows a JSON API response with validation errors. The top part displays an example value as a string and a schema definition. Below this, a validation error object is shown, indicating a problem with the 'loc' field. The media type is set to application/json. There are no links present.

```
"string"
422 Validation Error
application/json
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    },
    {
      "input": "string",
      "ctx": {}
    }
  ]
}
```

The screenshot shows the 'Schemas' section of a documentation tool. It lists three schemas: ChatRequest, HTTPValidationError, and ValidationError, each with an expand all link.

Schemas
ChatRequest > Expand all <code>object</code>
HTTPValidationError > Expand all <code>object</code>
ValidationError > Expand all <code>object</code>

CONCLUSION

The AI Study Assistant Chatbot successfully demonstrates how AI systems can integrate LLMs with database-backed memory to provide contextual responses. The project showcases real-world AI assistant architecture, including backend API development, cloud database storage, and deployment.

I WASN'T ABLE TO ADD THE MongoDB Atlas collection showing stored messages DUES TIME LIMIT OF UPLOAD