

Project Report: AI-Powered Study Assistant

1. Project Overview

This project involves the development of an **AI-powered Study Assistant Chatbot** designed to help students with academic queries. The core objective was to build a system that not only answers questions but also maintains a **Persistent Contextual Memory**. Unlike basic chatbots, this assistant can recall previous interactions, providing a more personalized and human-like tutoring experience.

2. Tech Stack Used

- **Backend Framework:** FastAPI (Python)
- **LLM Model:** Llama-3.3-70b (via Groq Cloud API)
- **AI Orchestration:** LangChain
- **Database:** MongoDB Atlas (Cloud)
- **API Testing:** Swagger UI (FastAPI Docs)
- **Deployment:** Render

3. Memory Efficient (How's it works)

- The most critical feature of this project is the **Persistent Memory** handled via MongoDB Atlas.
- **Data Persistence:** Every interaction (User Query and Bot Response) is stored as a document in the MongoDB 'chats' collection, mapped to a unique user_id.
- **Contextual Retrieval:** Before processing a new request, the system queries the database to fetch the **last 5 interactions** for that specific user_id.
- **Contextual Injection:** These past messages are formatted into LangChain HumanMessage and AIMessage objects and passed to the LLM as a prefix to the current question. This ensures the AI model "remembers" the user's name and previous academic context.

4. Key Features

- **Session Persistence:** The bot retains memory across different sessions and server restarts.
- **High Performance:** Utilizing FastAPI and Groq's LPU (Language Processing Unit) ensures near-instantaneous response times.
- **Educational Guardrails:** Configured with a SystemMessage to act specifically as a "Helpful AI Study Assistant."

5. Project Links

- **GitHub Repository:** <https://github.com/knehaprajapati/chatbot/tree/main>
- **Hosted API Link:** <https://chatbot-2t0g.onrender.com/>

6. API Test Screenshots

1. Screenshot 1 (Introduction):

The screenshot shows the API test interface for the AI Study Bot. At the top, there is a header with the title "AI Study Bot" and version "0.1.0 OAS 3.1". Below the header, there is a link to "/openapi.json". The main interface is titled "default". It shows two tabs: "GET" and "POST". The "POST" tab is selected, showing the endpoint "/chat Chat Endpoint". Under the "Parameters" section, it says "No parameters". In the "Request body" section, the word "required" is highlighted in red. The "application/json" content type is selected. Below the request body, there is a code editor with the following JSON input:

```
{  
  "user_id": "neha",  
  "message": "what is my name?"  
}
```

2. Screenshot 2 (Memory Proof):

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/chat' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "user_id": "neha",
    "message": "what is my name?"
}'
```

Request URL

```
http://127.0.0.1:8000/chat
```

Server response

Code	Details
200	Response body

```
{
  "user_id": "neha",
  "bot_response": "Your name is Neha Prajapati. You told me earlier when you introduced yourself."
}
```

Responses	
Code	Description
200	Successful Response

3. Screenshot 3 (MongoDB):

Cluster1 > study_bot_db > chats

View monitoring Visualize Your Data

Documents 4 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

EXPLAIN Reset FIND Options

+ ADD DATA UPDATE DELETE EXPORT CODE 25 1 - 4 of 4

user_id : "neha" user_query : "hii this is neha prajapati" bot_response : "Hello Neha Prajapati, it's nice to meet you. How can I assist you with..."
_id: ObjectId('699ac5cf6f9f7eb172d3bea5') user_id : "hello " user_query : "what is my name?" bot_response : "I don't have any information about your name. <small>Find a large language mod...</small> "
_id: ObjectId('699ac6076f9f7eb172d3bea8') user_id : "neha" user_query : "what is my name?" bot_response : "Your name is Neha Prajapati."

Terms Privacy Atlas Blog Contact Sales