



DevTown

## FINAL PROJECT REPORT

### AI-Powered Study Bot with Persistent Memory

#### Student Information :

- ✓ **Full name :** Khadija Ait Hajjou
- ✓ **Bootcamp :** Generative AI Chatbot – Build Your First AI Chatbot
- ✓ **Duration :** 3 Days Intensive Training
- ✓ **Project name :** Study\_Bot
- ✓ **Submission date :** February 22, 2026

#### Project Overview :

- ✓ **Core Technology :** FastAPI, LangChain, and Groq LLM
- ✓ **Database :** MongoDB Atlas (for Chat History Persistence)

## **1. Project Introduction :**

The goal of this project was to develop a functional AI Study Bot capable of assisting students with academic queries. Built using FastAPI and LangChain, the bot leverages the Groq Llama-3 model to provide fast and accurate responses. The application is designed to be scalable and secure, using environment variables to protect sensitive API keys.

## **2. Technical Architecture :**

- ✓ **Backend :** Developed with Python and FastAPI for high-performance API management
- ✓ **AI Engine :** Powered by Groq's Large Language Model (LLM) for natural language processing
- ✓ **Database :** MongoDB Atlas is used for storing chat history and maintaining session persistence

## **3. Memory & Persistence :**

- ✓ Every user interaction is stored in a database collection.
- ✓ The bot retrieves the previous conversation history based on a unique `session_id`.
- ✓ This allows the AI to maintain context, remember the user's name, and answer follow-up questions accurately.

## **4. Skills Acquired :**

During this intensive 3-day Bootcamp, I gained hands-on experience in the following areas:

- ✓ **Backend Development** : Building RESTful APIs using the FastAPI framework.
- ✓ **AI Integration** : Connecting and prompting Large Language Models (LLM) via the Groq API.
- ✓ **Database Management** : Implementing cloud storage and session persistence with MongoDB Atlas.
- ✓ **DevOps and Deployment** : Managing version control with github and deploying live applications on Render.

## **5. Conclusion :**

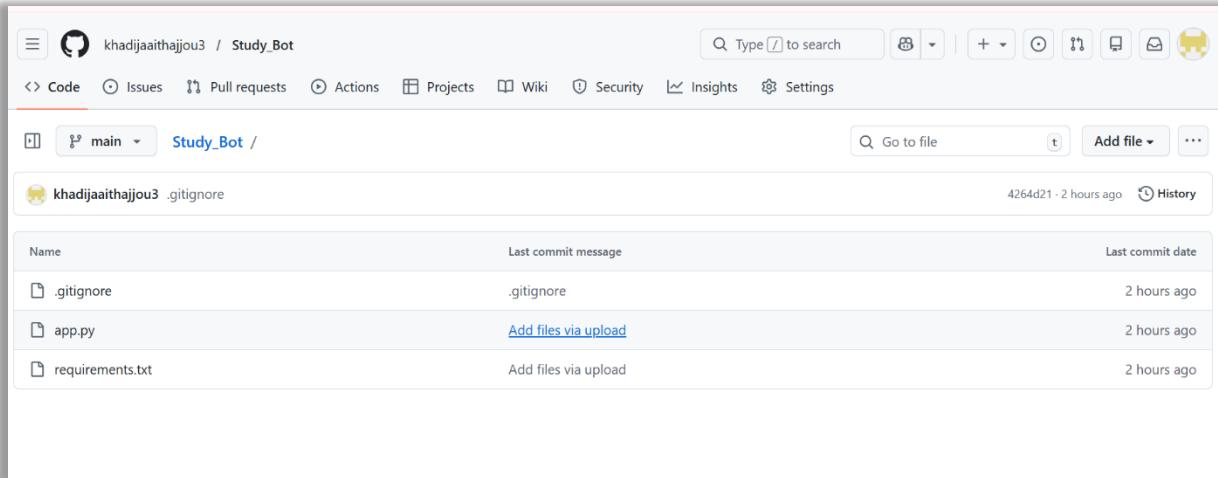
The Study\_Bot project shows how to connect an AI with a database like MongoDB Atlas. Using FastAPI and Groq, I built a chatbot that can remember past messages to help students better. This project proves that I can create, protect, and launch a complete AI application from start to finish.

**Note on Deployment:** To prioritize data security and follow best practices regarding personal information, I have chosen to host the application locally for this demonstration. All features, including the FastAPI server, Groq LLM integration, and MongoDB Atlas persistence, are fully functional and tested as shown in the project screenshots.

## **6. Project screenshots :**

### **1. GitHub Repository :**

This screenshot shows my GitHub repository named Study\_Bot. It contains all the necessary project files, including the source code, requirements, and configuration files.



## 2. Backend Implementation (VS Code) :

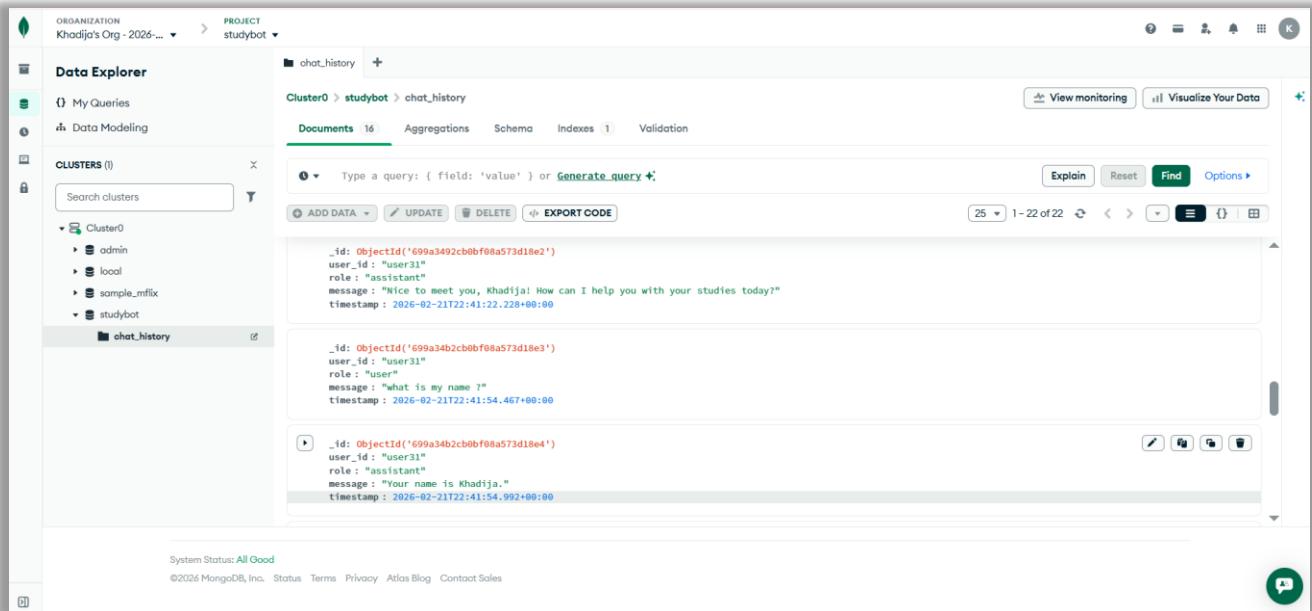
This image displays the Python code in VS Code. It highlights the integration of FastAPI with the Groq LLM and the connection setup for the MongoDB database.

A screenshot of VS Code showing the 'app.py' file. The code integrates FastAPI with the Groq LLM and connects to a MongoDB database. The code includes imports for os, dotenv, langchain\_groq, langchain\_core.prompts, pymongo, datetime, fastapi, fastapi.middleware.cors, and pydantic. It defines a ChatRequest BaseModel and sets up a FastAPI application with middleware for CORS and MongoDB connection.

### 3. Database (MongoDB Atlas) :

This view from MongoDB Atlas proves that the chatbot saves messages.

The data shows that the bot correctly remembers the user's name ("Khadija"), confirming that the persistent memory is working.



The screenshot shows the MongoDB Atlas Data Explorer interface. The left sidebar displays the organization 'Khadija's Org - 2026-' and project 'studybot'. Under 'CLUSTERS', 'Cluster0' is selected, showing collections like 'admin', 'local', 'sample\_mflix', and 'studybot'. The 'chat\_history' collection is currently selected. The main pane shows three documents in the 'Documents' tab. Each document contains fields: '\_id', 'user\_id' (set to 'user31'), 'role' (set to 'assistant'), 'message' (containing the user's message), and 'timestamp' (showing the timestamp of the message). The interface includes a search bar, a query builder, and various navigation and export options.

### 4. Local Server Execution (Uvicorn) :

This screenshot shows the FastAPI server running locally using Uvicorn. The 'Application startup complete' message indicates that the backend is ready to handle requests.

```
(venv) PS C:\Users\HP\Downloads\CHATBOT> uvicorn main:  
app --reload  
INFO:     Will watch for changes in these directories:  
['C:\\\\Users\\\\HP\\\\Downloads\\\\CHATBOT']  
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press  
CTRL+C to quit)  
INFO:     Started reloader process [22428] using StatR  
eload
```

### 5. Chatbot Interaction :

This is a live test of the Study Bot. I first told the bot my name, and in the next question, it correctly answered 'your name is Khadija'. This

proves the AI successfully retrieves my session history from MongoDB to maintain context.

```
\Downloads\CHATBOT\venv\Scripts\python.exe c:/Users/HP  
/Downloads/CHATBOT/app.py  
oV1  
Ask a question :my name is khadija  
c:\Users\HP\Downloads\CHATBOT\app.py:51: DeprecationWa  
rning: datetime.datetime.utcnow() is deprecated and sc  
heduled for removal in a future version. Use timezone-  
aware objects to represent datetimes in UTC: datetime.  
datetime.now(datetime.UTC).  
    "timestamp": datetime.utcnow()  
c:\Users\HP\Downloads\CHATBOT\app.py:58: DeprecationWa  
rning: datetime.datetime.utcnow() is deprecated and sc  
heduled for removal in a future version. Use timezone-  
aware objects to represent datetimes in UTC: datetime.  
    "timestamp": datetime.utcnow()  
Nice to meet you, Khadija! How can I help you with you  
r studies today?  
Ask a question :what is my name ?  
Your name is Khadija.
```