Study Bot – AI Powered Study Assistant

Chatbot Project Report

Submitted by

Burla Prudhvi Raj
CSE / Anil Neerukonda Institute Of Information Technology (ANITS)

Submitted to

CODETOWN EDTECH Pvt. Ltd.

Submission Date: 22 February 2026

1. Project Description

Study Bot is an AI-powered chatbot designed to assist students with academic and learning-related queries. The chatbot allows users to ask study-related questions, receive concise and accurate responses, and maintain conversational context across multiple interactions.

The application stores chat history in a MongoDB database, enabling the system to provide context-aware responses. The chatbot is exposed through a RESTful API built using FastAPI and deployed on a cloud platform.

2. What is this Project About?

This project focuses on building a simple AI Study Assistant that demonstrates how real-world AI chatbots work. The chatbot is capable of:

- Understanding natural language queries
- Providing academic-focused responses
- Remembering previous conversations using persistent storage
- Serving responses via a deployed API

This project helps students understand backend development, API design, database integration, and prompt engineering for large language models.

3. Technologies Used

- **Backend Framework:** FastAPI
- **LLM Integration:** LangChain with Groq API
- **Database:** MongoDB (Atlas / Local)
- **Frontend:** HTML, CSS, JavaScript
- **Deployment Platform:** Render

4. System Architecture

The system follows a client-server architecture:

- The user interacts with a web-based chat interface

- Requests are sent to a FastAPI backend
- The backend communicates with an LLM to generate responses
- Chat history is stored and retrieved from MongoDB

This architecture ensures scalability, modularity, and persistent conversation memory.

5. Step-by-Step Implementation

5.1. Environment Setup

Required Python packages such as FastAPI, Uvicorn, LangChain, and PyMongo were installed. Environment variables were configured to securely store API keys and MongoDB connection strings.

5.2. Building the Basic Chatbot

The chatbot was connected to a Large Language Model using LangChain. An API endpoint was created to accept user input and return AI-generated responses.

5.3. Database Memory Implementation

MongoDB was integrated to store user messages and bot responses along with timestamps and user identifiers. This allows the chatbot to retrieve past conversations and generate context-aware replies.

5.4. User Identification and Session Handling

Each user interacting with the chatbot is assigned a unique `user_id` on the client side. This identifier is generated in the browser and stored locally. The user ID is visible in the chatbot interface header when the profile icon is clicked.

For every chat request, the `user_id` is sent to the backend API along with the user's question. The backend uses this identifier to store and retrieve conversation history from MongoDB, ensuring that each user receives personalized and continuous responses.

This approach enables user-specific memory without requiring authentication, making the system simple, scalable, and suitable for educational applications.

5.5. Study Bot Logic

A system prompt was designed to ensure concise, student-friendly, and academic-focused responses. The chatbot avoids unnecessary verbosity unless explicitly requested.

5.6. Deployment

The application was deployed on Render. Environment variables were configured on the deployment platform, and the API was tested after deployment to ensure correct functionality.

6. API Endpoints

- **GET /** – Landing page for application navigation
- **POST /chat** – Accepts user questions and returns chatbot responses

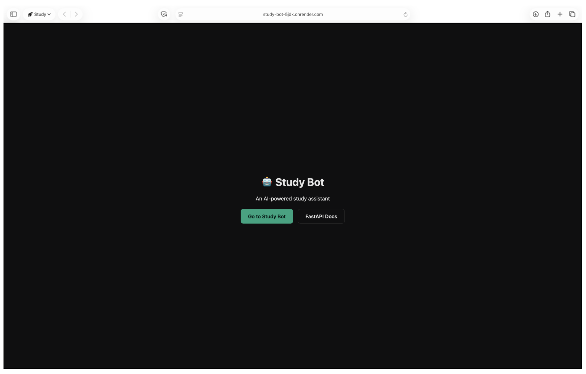
7. Application Links

- **GitHub Repository:** https://github.com/burlaprudhviraj123/Study-Bot
- **Landing Page:** https://study-bot-5jdk.onrender.com
- **Chatbot UI:** https://study-bot-5jdk.onrender.com/ui/
- **FastAPI Docs:** https://study-bot-5jdk.onrender.com/docs

The FastAPI documentation can be accessed directly from both the landing page and the chatbot interface, allowing users and evaluators to easily test the API endpoints.

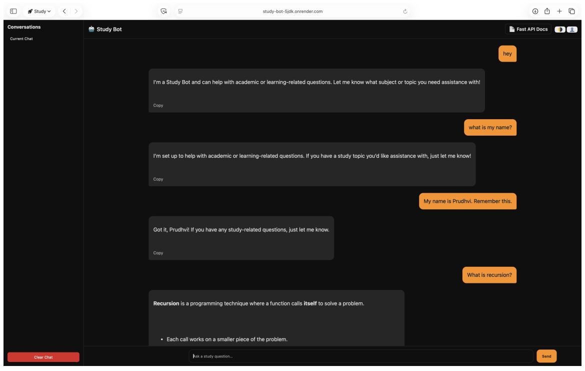
8. Screenshots

8.1. Application Overview and Memory Demonstration



An AI-powered study assistant

Figure 1: Landing page with navigation to chatbot UI and FastAPI Docs.



hey

I'm a Study Bot and can help with academic or learning-related questions. Let me know what subject or topic you need assistance with!

what is my name?

I'm set up to help with academic or learning-related questions. If you have a study topic you'd like assistance with, just let me know!

My name is Prudhvi. Remember this.

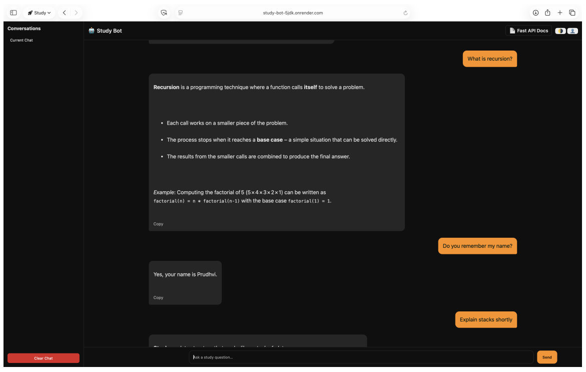
Got it, Prudhvi! If you have any study-related questions, just let me know.

What is recursion?

Recursion is a programming technique where a function calls **itself** to solve a problem.

- Each call works on a smaller piece of the problem.

Figure 2: User provides information to be remembered by the chatbot.



What is recursion?

Recursion is a programming technique where a function calls **itself** to solve a problem.

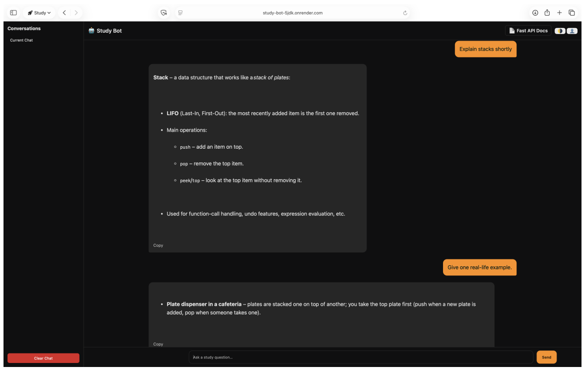
- Each call works on a smaller piece of the problem.
- The process stops when it reaches a **base case** – a simple situation that can be solved directly.
- The results from the smaller calls are combined to produce the final answer.

Example: Computing the factorial of 5 (5×4×3×2×1) can be written as `factorial(n) = n * factorial(n-1)` with the base case `factorial(1) = 1`.

Do you remember my name?

Yes, your name is Prudhvi.

Explain stacks shortly



Explain stacks shortly

Stack – a data structure that works like a *stack of plates*:

- **LIFO** (Last-In, First-Out): the most recently added item is the first one removed.
- Main operations:
  - `push` – add an item on top.
  - `pop` – remove the top item.
  - `peek/top` – look at the top item without removing it.
- Used for function-call handling, undo features, expression evaluation, etc.

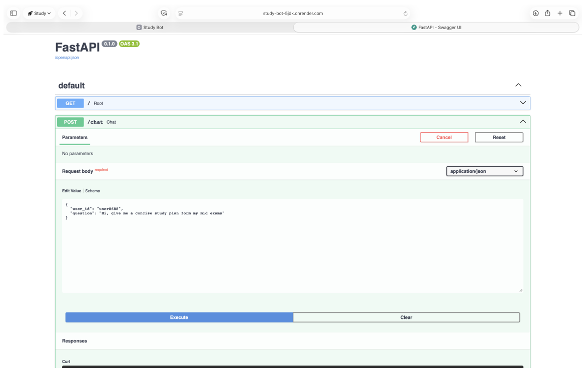
Give one real-life example.

- **Plate dispenser in a cafeteria** – plates are stacked one on top of another; you take the top plate first (push when a new plate is added, pop when someone takes one).

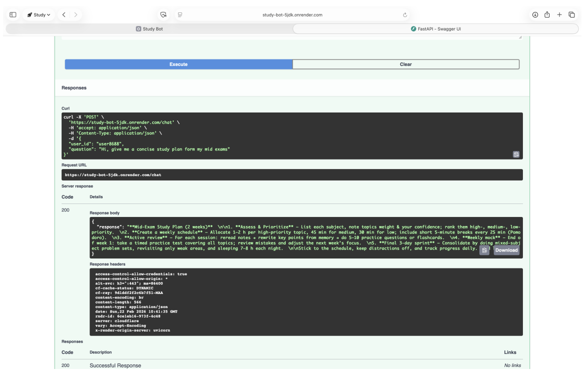
Figure 3: Chatbot recalls previously stored information, demonstrating memory.

Figure 4: Context-aware academic interaction based on prior conversation.

8.2. API Documentation and Database Storage



```
{
  "user_id": "user8688",
  "question": "Hi, give me a concise study plan form my mid exams"
}
```



Curl

```
curl -X 'POST' \
  'https://study-bot-5jdk.onrender.com/chat' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "user_id": "user8688",
  "question": "Hi, give me a concise study plan form my mid exams"
}'
```

Request URL

```
https://study-bot-5jdk.onrender.com/chat
```

Server response

Code | Details

200 Response body

```
{
  "response": "**Mid-Exam Study Plan (2 weeks)**  \n\n1. **Assess & Prioritize** – List each subject, note topics weight & your confidence; rank them high-, medium-, low-
priority.  \n2. **Create a weekly schedule** – Allocate 1–2 h per high-priority topic, 45 min for medium, 30 min for low; include short 5-minute breaks every 25 min (Pomo
doro).  \n3. **Active review** – For each session: reread notes → rewrite key points from memory → do 5–10 practice questions or flashcards.  \n4. **Weekly mock** – End o
f week 1: take a timed practice test covering all topics; review mistakes and adjust the next week's focus.  \n5. **Final 3-day sprint** – Consolidate by doing mixed-subj
ect problem sets, revisiting only weak areas, and sleeping 7–8 h each night.  \n\nStick to the schedule, keep distractions off, and track progress daily.
}
```

Response headers

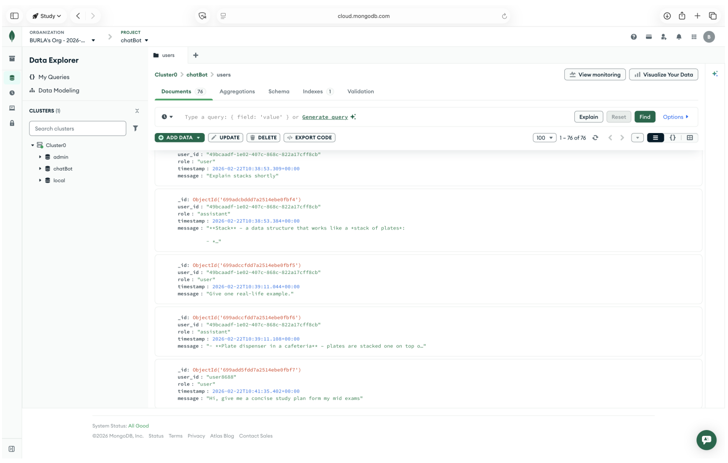
```
access-control-allow-credentials: true
access-control-allow-origin: *
alt-svc: h3=":443"; ma=86400
cf-cache-status: DYNAMIC
cf-ray: 9d1ddf2f2c6b7f51-MAA
content-encoding: br
content-length: 566
content-type: application/json
date: Sun,22 Feb 2026 10:41:35 GMT
rndr-id: 6ce1eb16-973f-4c48
server: cloudflare
vary: Accept-Encoding
x-render-origin-server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Figure 5: FastAPI Swagger documentation displaying available endpoints.

Figure 6: API tested using FastAPI's Try It Out feature.



```
user_id : "49bcaadf-1e02-407c-868c-822a17cff8cb"
role : "user"
timestamp : 2026-02-22T10:38:53.309+00:00
message : "Explain stacks shortly"

_id: ObjectId('699adcbddd7a2514ebe0fbf4')
user_id : "49bcaadf-1e02-407c-868c-822a17cff8cb"
role : "assistant"
timestamp : 2026-02-22T10:38:53.384+00:00
message : "**Stack** – a data structure that works like a *stack of plates*:

        - *…"

_id: ObjectId('699adccfdd7a2514ebe0fbf5')
user_id : "49bcaadf-1e02-407c-868c-822a17cff8cb"
role : "user"
timestamp : 2026-02-22T10:39:11.044+00:00
message : "Give one real-life example."

_id: ObjectId('699adccfdd7a2514ebe0fbf6')
user_id : "49bcaadf-1e02-407c-868c-822a17cff8cb"
role : "assistant"
timestamp : 2026-02-22T10:39:11.108+00:00
message : "- **Plate dispenser in a cafeteria** – plates are stacked one on top o…"

_id: ObjectId('699add5fdd7a2514ebe0fbf7')
user_id : "user8688"
role : "user"
timestamp : 2026-02-22T10:41:35.402+00:00
message : "Hi, give me a concise study plan form my mid exams"
```

Figure 7: MongoDB storing user messages and chatbot responses for persistent memory.

9. Conclusion

The Study Bot project successfully demonstrates the implementation of an AI-powered chatbot with persistent memory using MongoDB. The project highlights key concepts such as API development, database integration, deployment, and prompt engineering. This system serves as a foundation for building more advanced conversational AI applications.