PROJECT REPORT

ACADEMIC QUERY CHATBOX

[Study Bot – AI Powered Study Assistant]

PROJECT OVERVIEW

The Study Bot is an AI-powered chatbot designed to assist students with academic and learning-related questions. The chatbot uses a Large Language Model (LLM) to generate intelligent responses and integrates MongoDB to store and retrieve previous conversations.

Unlike a basic chatbot, this Study Bot maintains conversational memory by storing chat history in a database. This enables the bot to provide contextual and more relevant responses over time.

MAIN GOAL

The project demonstrates real-world AI system design involving:

- LLM API integration
- Backend API development using FastAPI
- Database integration using MongoDB
- Deployment of a live API

WORKFLOW OF THE CHATBOT

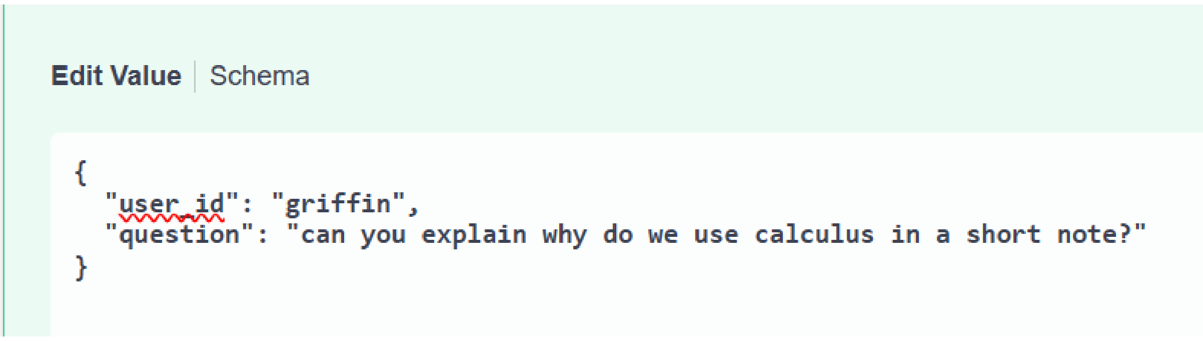
PREDEFINED PROMPT :

“Act as a knowledgeable Study Bot that explains academic concepts clearly with examples and structured steps. Provide honest, practical career guidance, skill-building advice, and learning roadmaps tailored to the student’s interests.Keep the ideas and instruction in a clean manner.”

~ This is the prompt injected to the chatbot

Step 1: User Sends a Question:

The user sends a POST request to the FastAPI endpoint with a study-related query.



```
{
  "user_id": "griffin",
  "question": "can you explain why do we use calculus in a short note?"
}
```

Step 2: System Prompt is Applied:

The chatbot uses the predefined system prompt which was mentioned above.

Step 3: LLM Generates Response:

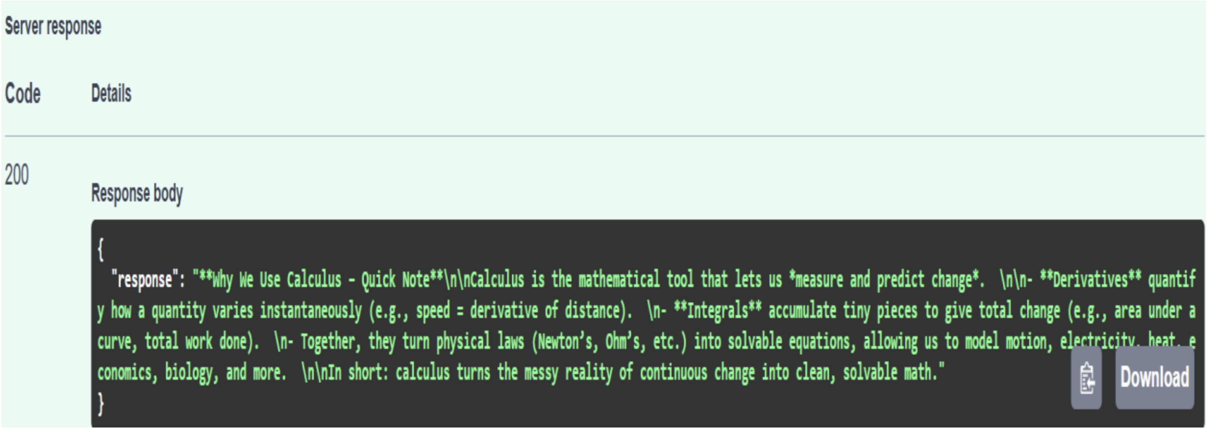
The chatbot is connected to an LLM (via LangChain and Groq API).
The LLM processes:

- System prompt
- Previous conversation (if any)
- Current user question

Then generates a contextual response.

Step 4: API Returns Response:

The response is returned as JSON:



```
{
  "response": "**Why We Use Calculus - Quick Note**\n\nCalculus is the mathematical tool that lets us *measure and predict change*.  \n\n- **Derivatives** quantif
y how a quantity varies instantaneously (e.g., speed = derivative of distance).  \n- **Integrals** accumulate tiny pieces to give total change (e.g., area under a
curve, total work done).  \n- Together, they turn physical laws (Newton's, Ohm's, etc.) into solvable equations, allowing us to model motion, electricity, heat, e
conomics, biology, and more.  \n\nIn short: calculus turns the messy reality of continuous change into clean, solvable math."
}
```

{

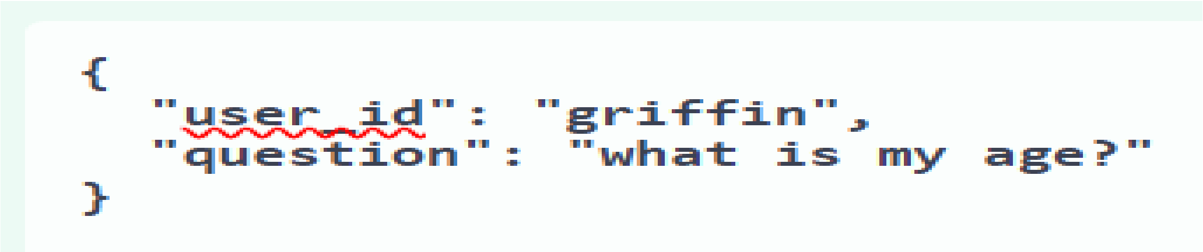
"response": "**Why We Use Calculus – Quick Note**\n\nCalculus is the mathematical tool that lets us *measure and predict change*. \n\n- **Derivatives** quantify how a quantity varies instantaneously (e.g., speed = derivative of distance). \n- **Integrals** accumulate tiny pieces to give total change (e.g., area under a curve, total work done). \n- Together, they turn physical laws (Newton’s, Ohm’s, etc.) into solvable equations, allowing us to model motion, electricity, heat, economics, biology, and more. \n\nIn short: calculus turns the messy reality of continuous change into clean, solvable math."

}

How Memory Is Implemented?

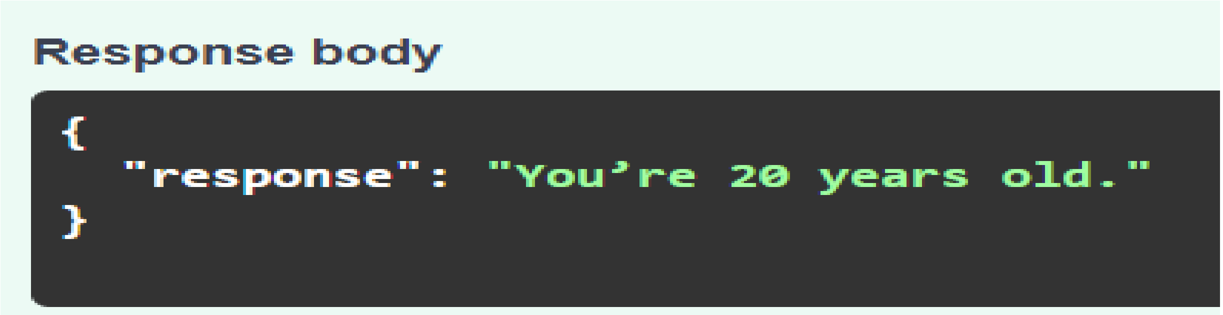
Memory is implemented using MongoDB to store and retrieve chat history.

QUESTION:



```
{
  "user_id": "griffin",
  "question": "what is my age?"
}
```

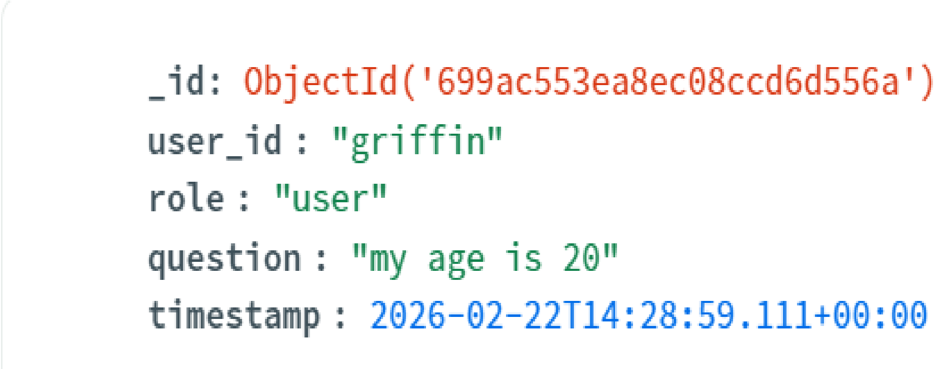
RESPONSE:



Response body

```
{
  "response": "You’re 20 years old."
}
```

~ PREVIOUS MEMORY OF THE USER GETS STORED USING THE MONGODB DATABASE



```
_id: ObjectId('699ac553ea8ec08ccd6d556a')
user_id : "griffin"
role : "user"
question : "my age is 20"
timestamp : 2026-02-22T14:28:59.111+00:00
```

This image depicts that the user with user_id has already mentioned his age in one of his previous chat. This gets internally stored in the chatbot's database (mongoDB).

HOW MEMORY WORKS:

Step 1: Store Conversation

After generating the bot response:

- The user message
- The bot response
- The timestamp

are stored in MongoDB.

Step 2: Retrieve Previous Messages

When a new question is asked:

- The application fetches previous messages from MongoDB.
- These messages are appended to the prompt.
- The LLM receives conversation history.

BENEFITS OF MEMORY STORAGE:

~ Context-aware replies

~ Follow-up question understanding

~ Continuous conversation

DEPLOYMENT PROCESS

Step 1: Push Code to GitHub

The complete project was uploaded to a GitHub repository including

- app_1.py (main file)
- requirements.txt
- README.md

Step 2: Connect GitHub to Render

- Render account created
- Repository connected
- New Web Service created

Step 3: Configure Environment Variables

- GROQ_API_KEY
- MONGODB_URI

were added in the Render dashboard

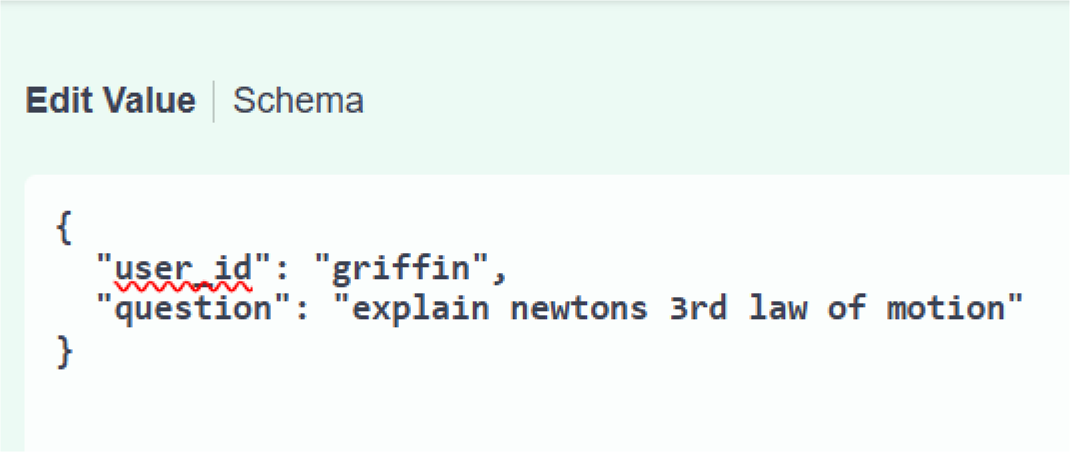
Step 4: Deploy Application

Render builds the project using:

`uvicorn app_1:app --host 0.0.0.0 --port 10000` ~ command

After successful deployment, a public API link is generated.

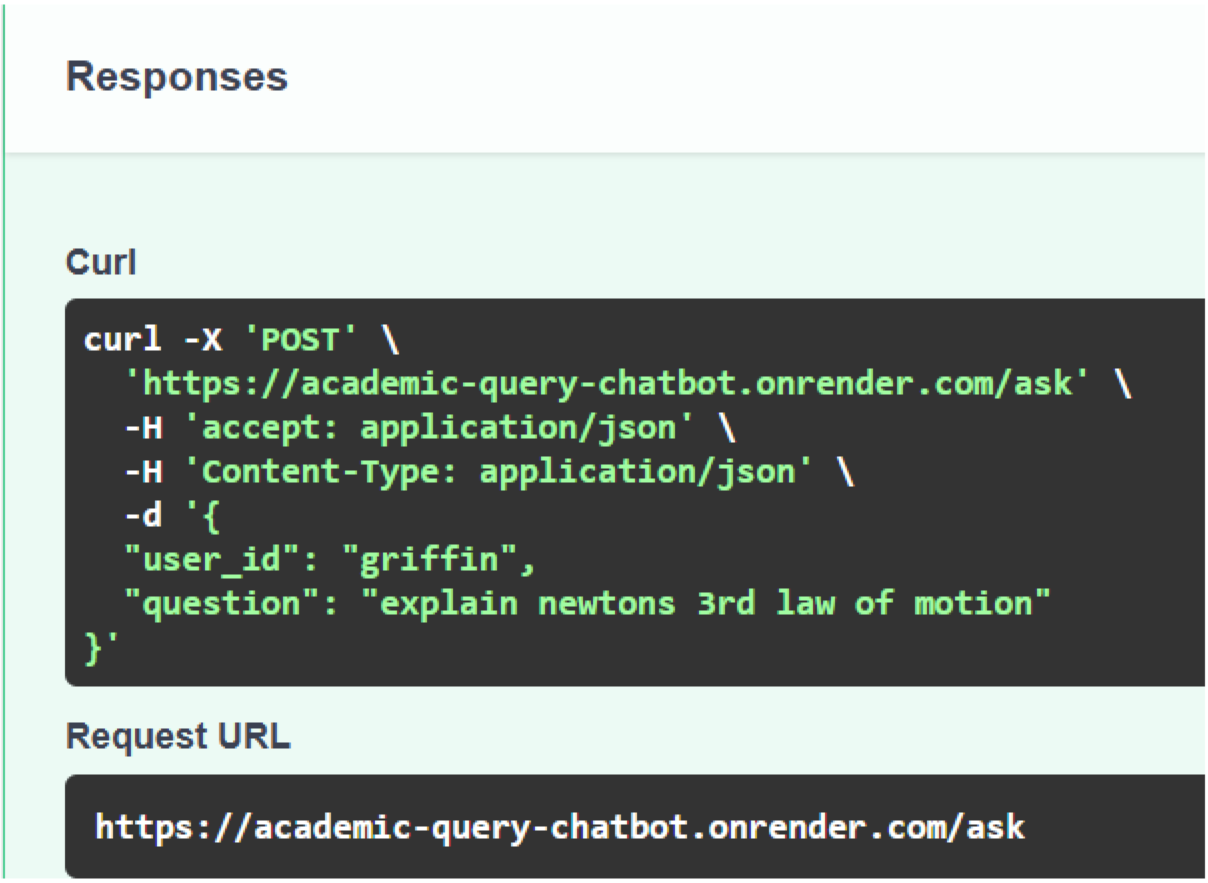
QUESTION:



```
{
  "user_id": "griffin",
  "question": "explain newtons 3rd law of motion"
}
```

RESPONSE:

~ curl


Responses

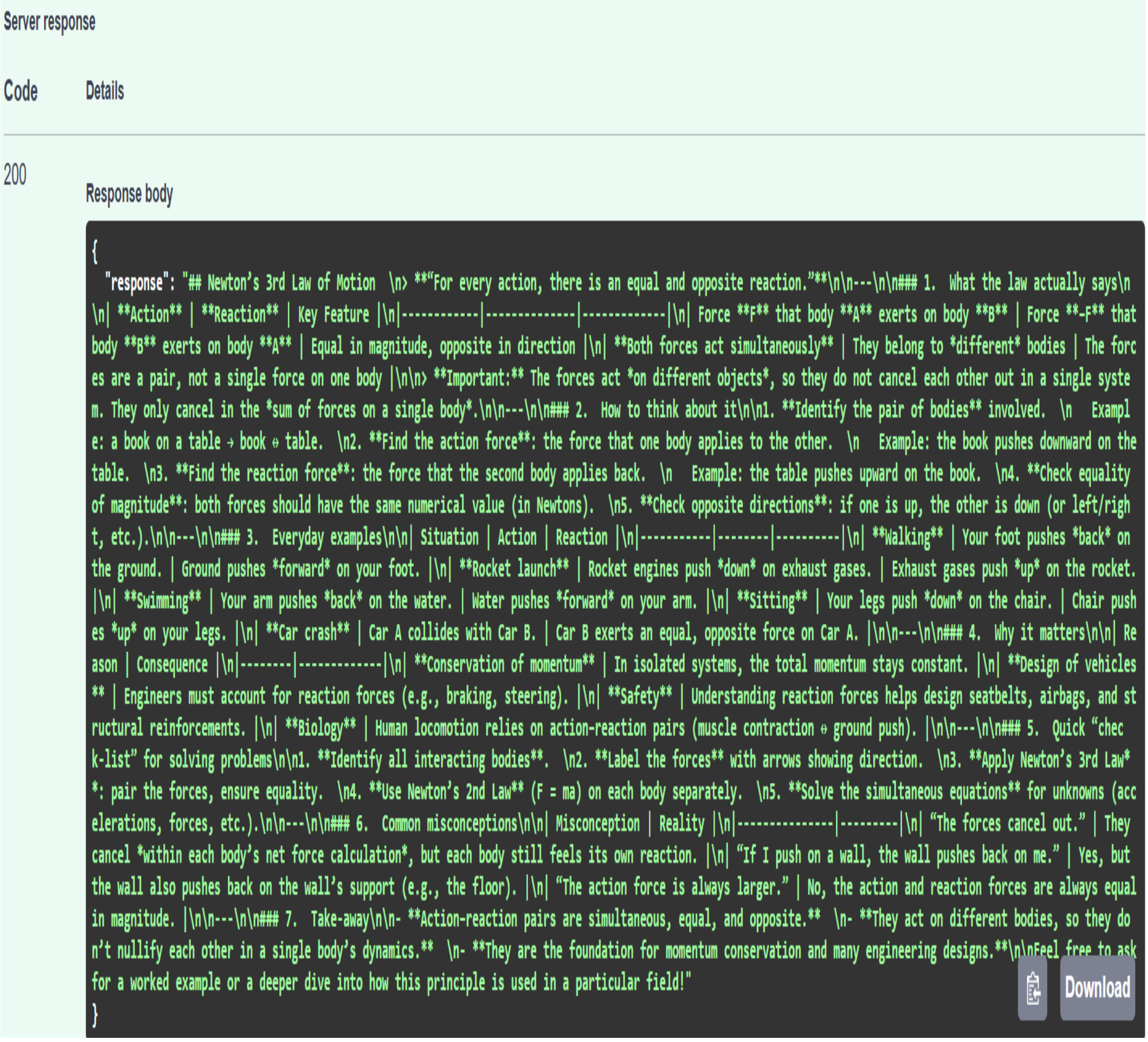
Curl

```
curl -X 'POST' \
  'https://academic-query-chatbot.onrender.com/ask' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "user_id": "griffin",
  "question": "explain newtons 3rd law of motion"
}'
```

Request URL

```
https://academic-query-chatbot.onrender.com/ask
```

~ response of the bot



```
{
  "response": "## Newton’s 3rd Law of Motion  \n> **“For every action, there is an equal and opposite reaction.”**\n\n---\n\n### 1.  What the law actually says\n
\n| **Action** | **Reaction** | Key Feature |\n|-----------|-------------|------------|\n| Force **F** that body **A** exerts on body **B** | Force **-F** that
body **B** exerts on body **A** | Equal in magnitude, opposite in direction |\n| **Both forces act simultaneously** | They belong to *different* bodies | The forc
es are a pair, not a single force on one body |\n\n> **Important:** The forces act *on different objects*, so they do not cancel each other out in a single syste
m. They only cancel in the *sum of forces on a single body*.\n\n---\n\n### 2.  How to think about it\n\n1. **Identify the pair of bodies** involved.  \n   Exampl
e: a book on a table → book ↔ table.  \n2. **Find the action force**: the force that one body applies to the other.  \n   Example: the book pushes downward on the
table.  \n3. **Find the reaction force**: the force that the second body applies back.  \n   Example: the table pushes upward on the book.  \n4. **Check equality
of magnitude**: both forces should have the same numerical value (in Newtons).  \n5. **Check opposite directions**: if one is up, the other is down (or left/righ
t, etc.).\n\n---\n\n### 3.  Everyday examples\n\n| Situation | Action | Reaction |\n|----------|--------|---------|\n| **Walking** | Your foot pushes *back* on
the ground. | Ground pushes *forward* on your foot. |\n| **Rocket launch** | Rocket engines push *down* on exhaust gases. | Exhaust gases push *up* on the rocket.
|\n| **Swimming** | Your arm pushes *back* on the water. | Water pushes *forward* on your arm. |\n| **Sitting** | Your legs push *down* on the chair. | Chair push
es *up* on your legs. |\n| **Car crash** | Car A collides with Car B. | Car B exerts an equal, opposite force on Car A. |\n\n---\n\n### 4.  Why it matters\n\n| Re
ason | Consequence |\n|--------|------------|\n| **Conservation of momentum** | In isolated systems, the total momentum stays constant. |\n| **Design of vehicles
** | Engineers must account for reaction forces (e.g., braking, steering). |\n| **Safety** | Understanding reaction forces helps design seatbelts, airbags, and st
ructural reinforcements. |\n| **Biology** | Human locomotion relies on action-reaction pairs (muscle contraction ↔ ground push). |\n\n---\n\n### 5.  Quick “chec
k-list” for solving problems\n\n1. **Identify all interacting bodies**.  \n2. **Label the forces** with arrows showing direction.  \n3. **Apply Newton’s 3rd Law*
*: pair the forces, ensure equality.  \n4. **Use Newton’s 2nd Law** (F = ma) on each body separately.  \n5. **Solve the simultaneous equations** for unknowns (acc
elerations, forces, etc.).\n\n---\n\n### 6.  Common misconceptions\n\n| Misconception | Reality |\n|--------------|--------|\n| “The forces cancel out.” | They
cancel *within each body’s net force calculation*, but each body still feels its own reaction. |\n| “If I push on a wall, the wall pushes back on me.” | Yes, but
the wall also pushes back on the wall’s support (e.g., the floor). |\n| “The action force is always larger.” | No, the action and reaction forces are always equal
in magnitude. |\n\n---\n\n### 7.  Take-away\n\n- **Action-reaction pairs are simultaneous, equal, and opposite.**  \n- **They act on different bodies, so they do
n’t nullify each other in a single body’s dynamics.**  \n- **They are the foundation for momentum conservation and many engineering designs.**\n\nFeel free to ask
for a worked example or a deeper dive into how this principle is used in a particular field!"
}
```

RENDER URL :

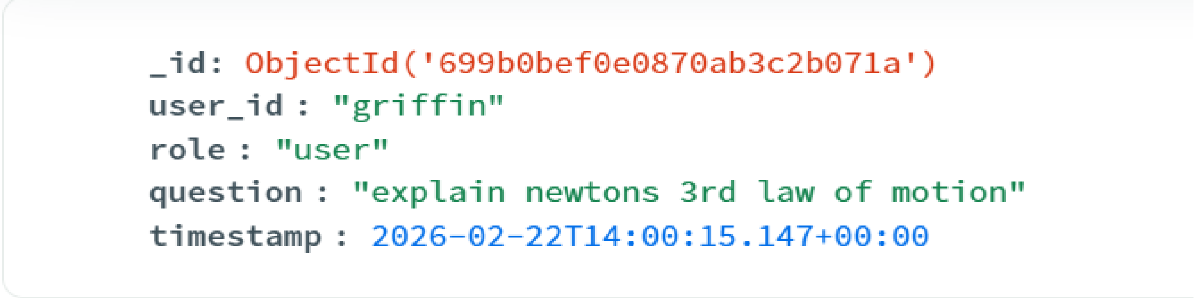
https://academic-query-chatbot.onrender.com/docs

GITHUB REPO:

https://github.com/Dhesshwin-B-G/academic_query_chatbot

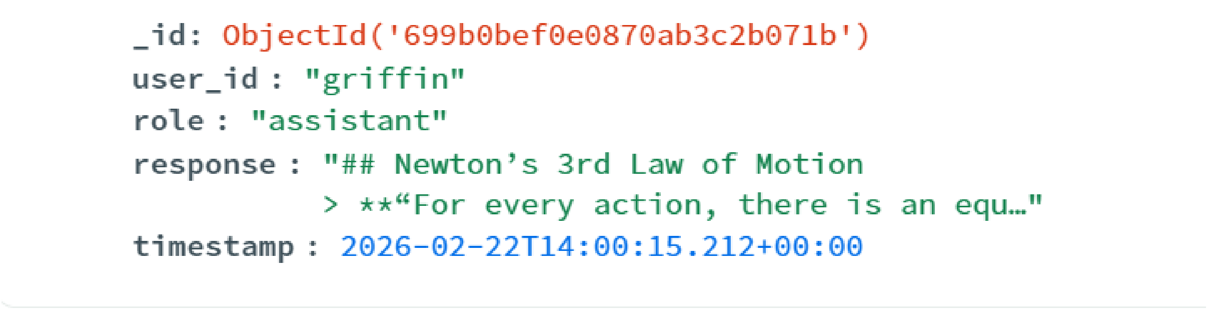
UPDATION OF MEMORY IN THE DATABASE (MONGODB):

USER’S QUESTION:



```
_id: ObjectId('699b0bef0e0870ab3c2b071a')
user_id : "griffin"
role : "user"
question : "explain newtons 3rd law of motion"
timestamp : 2026-02-22T14:00:15.147+00:00
```

CHATBOT’S RESPONSE:



```
_id: ObjectId('699b0bef0e0870ab3c2b071b')
user_id : "griffin"
role : "assistant"
response : "## Newton’s 3rd Law of Motion
           > **“For every action, there is an equ…"
timestamp : 2026-02-22T14:00:15.212+00:00
```

CONCLUSION

This project successfully demonstrates:

- Integration of LLM with a backend application
- Persistent memory using MongoDB
- API-based chatbot architecture
- Real-world cloud deployment

The Study Bot simulates how modern AI assistants maintain context and deliver intelligent academic responses.