**RAG LLM Chat App with LangChain, Streamlit, OpenAI and Anthropic APIs**

**✅ Step-by-Step: Create a New GitHub Repository**

**🧱 Step 1: Go to GitHub**

1. Open your browser and go to <https://github.com>
2. Log in to your GitHub account

**📁 Step 2: Create a New Repository**

1. Click the **"+" icon** at the top right
2. Select **"New repository"**

**📝 Step 3: Fill in Repository Details**

* **Repository name**: rag-llm-chat-app
* **Description**:  
  RAG Chat App using LangChain, Streamlit, OpenAI, and Anthropic APIs
* **Visibility**: Choose either:
  + ✅ **Public** – anyone can see it
  + 🔒 **Private** – only you can see it
* (Optional) Check: **"Add a README file"**
* (Optional) Check: **".gitignore → Python"**
* Click **Create repository**

**💻 Step 4: Push Your Local Code (if code is on your computer)**

If you already built the app locally, open **Terminal / Command Prompt**, then:

bash

CopyEdit

cd path/to/your/project-folder

# Initialize Git

git init

# Add remote (replace USERNAME with your GitHub username)

git remote add origin https://github.com/USERNAME/rag-llm-chat-app.git

# Add and commit files

git add .

git commit -m "Initial commit"

# Push to GitHub

git branch -M main

git push -u origin main

**🖥️ Step 5: Verify**

* Go to <https://github.com/USERNAME/rag-llm-chat-app>
* You should see your project files uploaded!

For the requirement

You will have txt file name requirments

pip install -r requirements.txt

How to find the API for the chat GPT

Go to

<https://platform.openai.com/docs/overview>

pip install openai

from openai import OpenAI

client = OpenAI(

api\_key="sk-proj-Y7WkhWaENBUP8y8se\_wzpUzSbLQwHhEpCUbiiBkS86QxinK6tyR0d6zKBPmGb4aAIuKxPTwRuFT3BlbkFJw5ubTzrCyB0mEo1aUvkGHjztcTzubIlvzsdq9vFqpi0pgW1H6pdt1cqoCGY2ZT9rddHsxN40AA"

)

completion = client.chat.completions.create(

model="gpt-4o-mini",

store=True,

messages=[

{"role": "user", "content": "write a haiku about ai"}

]

)

print(completion.choices[0].message);

IF you want to save the user chat history and you use is in the future past of the retiverl

how can I save this for the future so if the same user came after one month

**ChatGPT said:**

Great question — if you want your app to **remember document embeddings for future sessions (even after a month)**, here’s exactly how to do it with **ChromaDB + LangChain**, step-by-step:

**✅ Goal:**

* You want to **embed documents once**
* Save them **locally**
* Then be able to **reload them later**, e.g., when a user comes back in a month

**✅ Step-by-step setup**

**🔹 1. Save embeddings locally with persist\_directory**

python

CopyEdit

from langchain\_community.vectorstores import Chroma

from langchain\_openai import OpenAIEmbeddings

# First-time setup: embed and persist

vector\_db = Chroma.from\_documents(

documents=documents\_chunks,

embedding=OpenAIEmbeddings(),

persist\_directory="chroma\_storage", # Folder where vectors will be saved

)

vector\_db.persist() # Saves the database to disk

* documents\_chunks is your list of document chunks
* This creates the folder chroma\_storage/ and saves everything there

**🔹 2. Later (1 day or 1 month later), reload the database:**

python

CopyEdit

from langchain\_community.vectorstores import Chroma

from langchain\_openai import OpenAIEmbeddings

# Reload existing vector store

vector\_db = Chroma(

persist\_directory="chroma\_storage",

embedding\_function=OpenAIEmbeddings()

)

You can now:

python

CopyEdit

vector\_db.similarity\_search("What is the user asking?")

➡️ You don’t need to re-embed the documents again.

**🧠 Optional: Support multiple users**

If you want to store **user-specific data** for many users:

python

CopyEdit

user\_id = "user\_123"

persist\_path = f"user\_data/{user\_id}\_vectors"

# Save or load user-specific Chroma DB

vector\_db = Chroma.from\_documents(

documents=user\_chunks,

embedding=OpenAIEmbeddings(),

persist\_directory=persist\_path,

)

vector\_db.persist()

Later:

python

CopyEdit

vector\_db = Chroma(

persist\_directory=persist\_path,

embedding\_function=OpenAIEmbeddings()

)

**✅ Summary**

| **Action** | **Code/Command** |
| --- | --- |
| Save locally | persist\_directory="chroma\_storage" |
| Save to disk | vector\_db.persist() |
| Load later | Chroma(persist\_directory=..., embedding\_function=...) |
| Save per user (optional) | persist\_directory=f"user\_data/{user\_id}\_vectors" |

Let me know if you want to:

* Save user queries along with the vector store
* Handle large datasets across months or years
* Use a real database like Qdrant or MongoDB later

I can guide you through those too.

**You said:**

ok what if I want to save the history in for example a Azuru cloud

**ChatGPT said:**

Great! If you want to **store your conversation history (or embeddings, or both) on Azure Cloud**, you have a few solid options depending on **what exactly you want to store**.

Let’s break it down:

**🧠 1. What do you want to save to Azure?**

| **Data Type** | **Example Content** | **Best Azure Storage Option** |
| --- | --- | --- |
| **Conversation history** | Chat messages: user ↔️ bot | Azure Cosmos DB, Azure Table Storage, Azure SQL |
| **Vector embeddings** | LangChain / Chroma vector DB | Azure Blob Storage (for persist folder) or Cosmos DB with qdrant |
| **Files / PDFs** | Original user documents | Azure Blob Storage |

**✅ Option A: Save Chat History to Azure Cosmos DB**

Cosmos DB is a scalable NoSQL database, great for saving:

* user\_id
* timestamp
* messages: user & AI turns
* metadata: source, tags, etc.

**💡 How it fits:**

LangChain allows you to **customize the memory** system to use your own DB backend (instead of in-memory).

**🔧 Example: Save history to Cosmos DB**

1. Create a **Cosmos DB** account in Azure
   * Choose **API = MongoDB** or **NoSQL** (easier)
2. Install dependencies:

bash

CopyEdit

pip install azure-cosmos

1. Example Python logic:

python

CopyEdit

from azure.cosmos import CosmosClient

import os

import uuid

COSMOS\_URL = os.getenv("COSMOS\_URL")

COSMOS\_KEY = os.getenv("COSMOS\_KEY")

client = CosmosClient(COSMOS\_URL, COSMOS\_KEY)

db = client.get\_database\_client("chat-db")

container = db.get\_container\_client("chat-history")

def save\_chat(user\_id, message, role):

container.create\_item({

"id": str(uuid.uuid4()),

"user\_id": user\_id,

"message": message,

"role": role, # 'human' or 'ai'

})

You can then retrieve messages later and restore them into LangChain memory.

**✅ Option B: Save Embeddings to Azure Blob Storage (using Chroma)**

If you want to persist your **Chroma vector store** in Azure Blob Storage:

1. After calling:

python

CopyEdit

Chroma(..., persist\_directory="chroma\_storage")

vector\_db.persist()

1. Upload the chroma\_storage/ folder to Azure:

bash

CopyEdit

az storage blob upload-batch \

--account-name <your\_account> \

--destination my-container/chroma\_storage \

--source chroma\_storage

1. Later, download it back before reloading the vector DB.

You can automate upload/download using Python and azure-storage-blob.

**✅ Option C: Use Qdrant or Weaviate hosted on Azure (for vector storage)**

If you prefer a **production-grade** vector DB instead of Chroma:

* You can run **Qdrant** or **Weaviate** as a container on Azure
* LangChain supports both
* These let you persist vectors directly to Azure cloud with no local .persist() step

**🧠 Summary: Best Options**

| **Want to store...** | **Use...** |
| --- | --- |
| Chat history for many users | Azure Cosmos DB (NoSQL or Mongo) |
| Chroma vector DB | Save persist\_directory to Azure Blob |
| Embeddings at scale (cloud-native) | Qdrant or Weaviate on Azure |
| Original PDFs/files | Azure Blob Storage |

Would you like me to generate working sample code for:

* Cosmos DB chat storage?
* Uploading chroma\_storage/ to Azure Blob?
* Running Qdrant on Azure?

Let me know your use case!