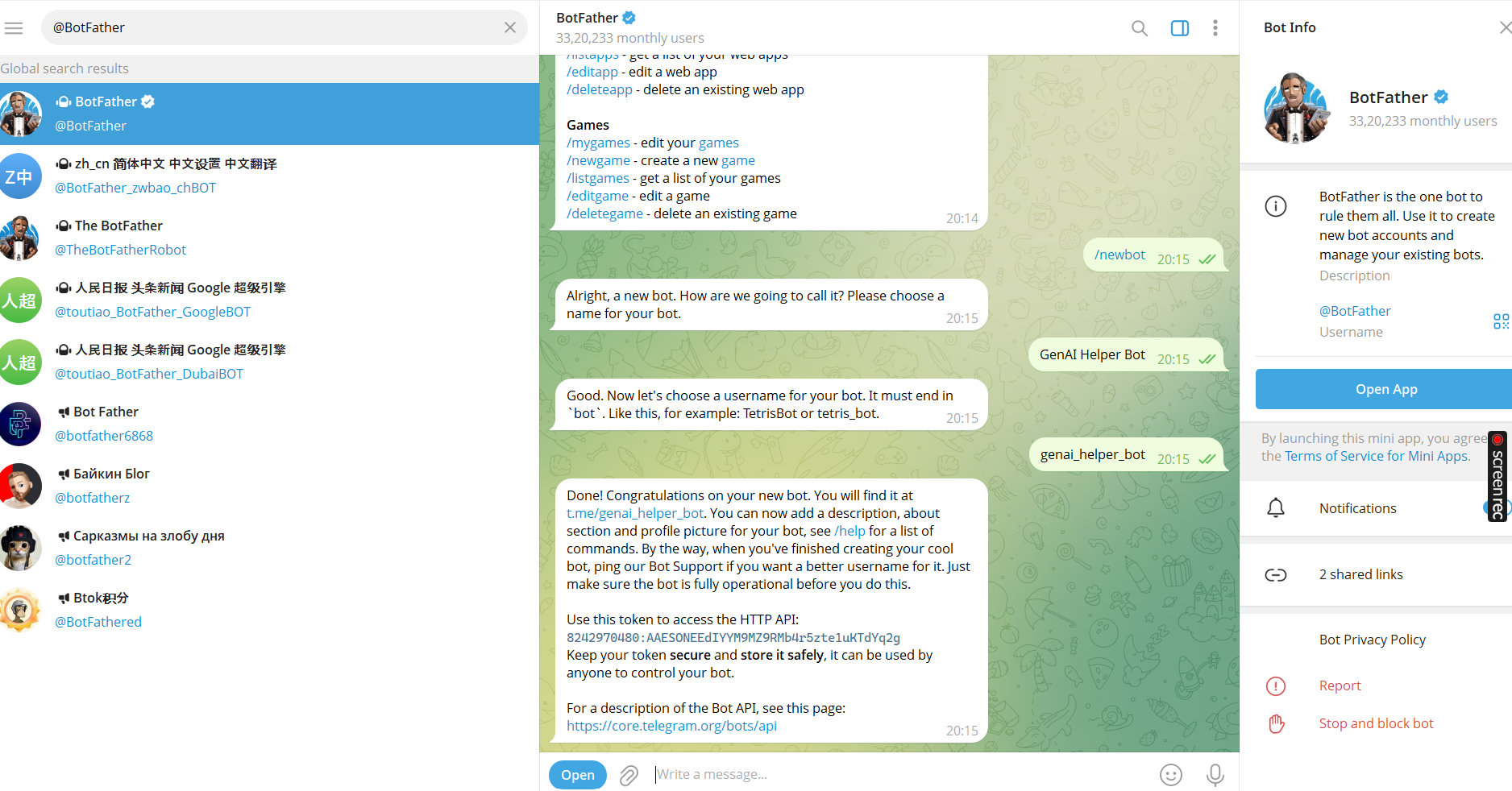
**GENAI BOT DOCUMENTATION**

**STEPS**

**1)Create a Token**

**How to Get It (Step-by-Step)**



**1️⃣ Open Telegram App**

You can do this on:

* Your phone (Telegram mobile app), or
* Telegram Web/Desktop (https://web.telegram.org)

**2️⃣ Search for BotFather**

In Telegram search bar, type **@BotFather** and open the verified one (with a blue checkmark ✅).

**3️⃣ Create a New Bot**

Send this command to BotFather:

/newbot

Then follow the prompts:

* Choose a **name** (e.g., GenAI Helper Bot)
* Choose a **username** (must end with “bot”, e.g. genai\_helper\_bot)

**4️⃣ Get Your Token**

After setup, BotFather replies like this:

Done! Congratulations on your new bot.

You will find it at t.me/genai\_helper\_bot.

Use this token to access the HTTP API:

8242970480:AAESONEEdIYYM9MZ9RMb4r5zte1uKTdYq21

✅ **Copy that token** — that’s your TELEGRAM\_TOKEN.

**⚙️ 5️⃣ Set the Token as Environment Variable**

Depending on your OS or environment:

**🪟 On Windows (PowerShell)**

setx TELEGRAM\_TOKEN " 8242970480:AAESONEEdIYYM9MZ9RMb4r5zte1uKTdYq21"

Then restart your terminal.

Now you can verify it:

echo $TELEGRAM\_TOKEN

**or**

**Optional**

If you don’t want to use environment variables,  
you can **hardcode the token**

TOKEN = "8242970480:AAESONEEdIYYM9MZ9RMb4r5zte1uKTdYq21"

**2)HOW TO TEST TOKEN**

**Step 1 — Open Command Prompt**

Open **Command Prompt**

cd F:\GENAI\genai\_bot\_project

**🧱 Step 2 — Create a Virtual Environment**

python -m venv venv

F:\GENAI\genai\_bot\_project\venv

**⚙️ Step 3 — Activate the Virtual Environment**

venv\Scripts\activate

After this, your prompt changes to look like:

(venv) F:\GENAI\genai\_bot\_project>

✅ That means the virtual environment is active.

**📦 Step 4 — Install Dependencies**

(venv) F:\GENAI\genai\_bot\_project>

pip install -r requirements.txt

It will download models like transformers, sentence-transformers, and python-telegram-bot.

**🔑 Step 5 — Set Your Telegram Token**

**Option A — Temporary (current session only)**

set TELEGRAM\_TOKEN=1234567890:AAE\_xYzExampleOfYourTokenValue

**Option B — Permanent (recommended)**

If you want it available every time you open CMD:

setx TELEGRAM\_TOKEN "1234567890:AAE\_xYzExampleOfYourTokenValue"

*(then close and reopen CMD)*

**🧪 Step 6 — Test Your Token**

Before running the bot, confirm the token works.

python extras\test\_token.py

If successful, you’ll see:

(venv) F:\GENAI\genai\_bot\_project>python extras\test\_token.py

✅ Token is valid!

Bot name: GenAI Helper Bot

Bot username: @genai\_helper\_bot

Bot ID: 8242970480If it says Unauthorized, recheck your token from **BotFather**.

**🤖 Step 7 — Run the Telegram Bot**

python telegram\_bot.py

You should see in your console:

INFO telegram\_bot - Starting bot...

INFO rag - Index rebuilt: 3 docs

**💬 Step 8 — Talk to Your Bot**

1. Open Telegram.
2. Search for your bot username (e.g., @genai\_helper\_bot).
3. Send:
4. /start

→ You’ll get a greeting message.

Then try:

/ask how can I reset my password?

and if you added docs (via extras/add\_docs.py), it will answer intelligently.  
Or send /image then upload a picture — it’ll generate a caption.

**🧰 Step 9 — (First-Run Setup) Add Documents for RAG**

If you haven’t added your documents yet:

python extras\add\_docs.py

It will read all .txt / .md files in data\docs and build the local knowledge base.

**🪵 Step 10 — Check Logs**

Logs are automatically stored in:

logs\bot.log

**🎉 You’re Done!**

✅ Summary of Commands for Windows CMD:

cd F:\GENAI\genai\_bot\_project

python -m venv venv

venv\Scripts\activate

pip install -r requirements.txt

set TELEGRAM\_TOKEN=your\_token\_here

python extras\test\_db\_connection.py

python extras\test\_token.py

python extras\add\_docs.py

python telegram\_bot.py

**Below are screenshot attached**

1)python extras\test\_token.py

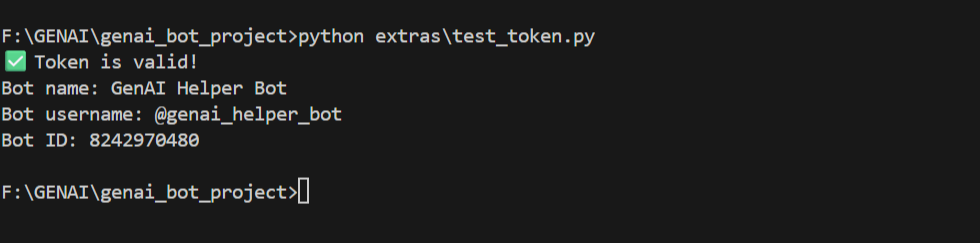
(venv) F:\GENAI\genai\_bot\_project>python extras\test\_token.py

✅ Token is valid!

Bot name: GenAI Helper Bot

Bot username: @genai\_helper\_bot

Bot ID: 8242970480



2) python extras\test\_db\_connection.py

✅ Connected successfully to SQLite database!

Database path: data\embeddings.db

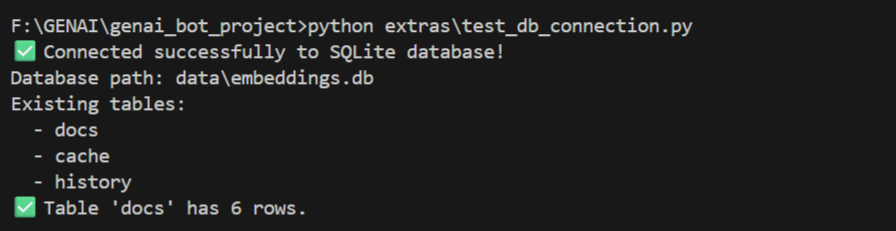
Existing tables:

- docs

- cache

- history

✅ Table 'docs' has 6 rows.



3) python -m extras.add\_docs

2025-10-29 23:32:36,111 INFO rag - Index rebuilt: 6 docs

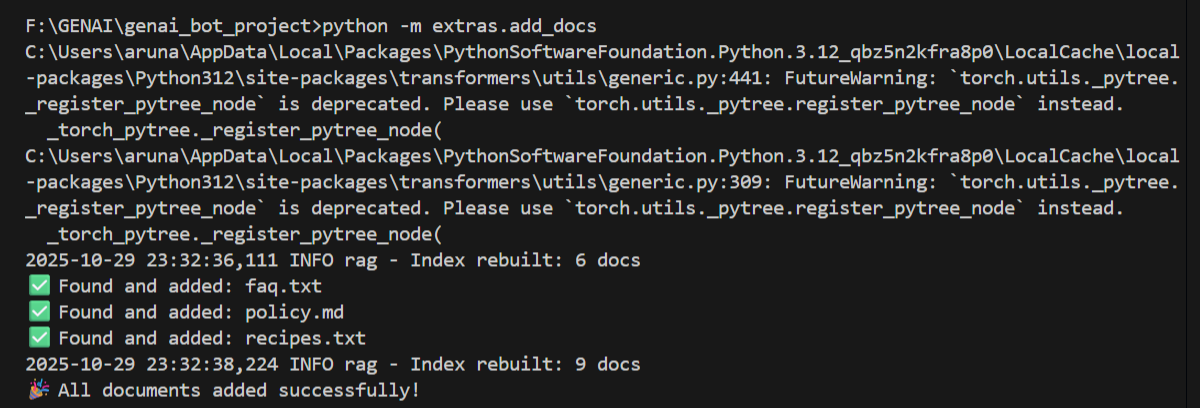
✅ Found and added: faq.txt

✅ Found and added: policy.md

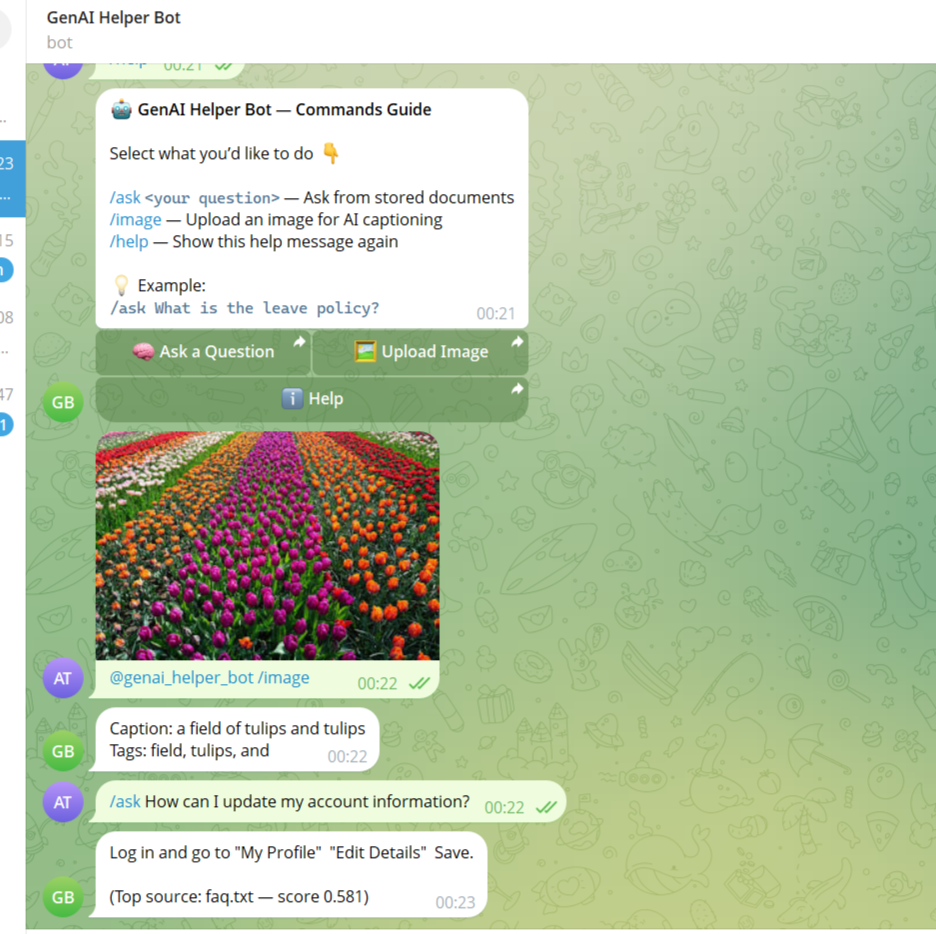
✅ Found and added: recipes.txt

2025-10-29 23:32:38,224 INFO rag - Index rebuilt: 9 docs

🎉 All documents added successfully!



4)python telegram\_bot.py



**GenAI Telegram Bot Project Report**

**📌 Project Title**

**Telegram Bot for Intelligent Document Querying and Image Captioning using GenAI**

**🎯 Objective**

To build a **lightweight GenAI-powered Telegram bot** that:

* Responds to user queries using a **Retrieval-Augmented Generation (RAG)** system.
* Generates **captions and tags for uploaded images** using a vision-language model.
* Provides clear and interactive responses to enhance user experience.

**🧱 System Architecture**

**Components:**

1. **Telegram Bot Interface**
   * Handles /ask, /image, /help, and /summarize commands.
   * Uses the python-telegram-bot library.
2. **RAG Engine (MiniRAG)**
   * Retrieves relevant chunks from local text documents.
   * Uses sentence embeddings + lightweight local LLM for generating answers.
3. **Vision Engine (VisionCaptioner)**
   * Generates descriptive captions and tags from uploaded images.
   * Uses a pretrained BLIP model from Hugging Face.
4. **SQLite Database**
   * Stores embeddings and user interaction history.
5. **Logging System**
   * Captures key bot events and errors for monitoring.

**⚙️ Tech Stack**

| **Layer** | **Technology** |
| --- | --- |
| Programming Language | Python 3.12 |
| Bot Framework | python-telegram-bot (v20.8) |
| Embedding Model | all-MiniLM-L6-v2 (Sentence Transformers) |
| LLM (Text Generator) | Ollama / HuggingFace Local Model |
| Vision Model | Salesforce/blip-image-captioning-base |
| Database | SQLite3 |
| Other Tools | Torch, Transformers, Pillow, Numpy |
| Logging | Python logging (rotating file logger) |

**🧩 Folder Structure**

genai\_bot\_project/

│

├── telegram\_bot.py # Main bot script

├── rag.py # Mini-RAG system

├── vision.py # Image captioning logic

├── db.py # SQLite database for history

├── utils.py # Logger setup

│

├── data/

│ └── docs/ # Knowledge base (faq.txt, policy.md, recipes.txt)

│

├── samples/

│ └── embeddings.db # Vector database

│

├── extras/

│ ├── add\_docs.py # Script to embed documents

│ └── test\_token.py # Telegram token test script

│

├── logs/

│ └── bot.log # Log file

│

└── README.md # Project documentation

**🤖 Models Used**

**1️⃣ Text Retrieval (Mini-RAG)**

* **Embedding Model**: all-MiniLM-L6-v2 (from SentenceTransformers)
* **Task**: Converts text chunks into vector embeddings.
* **Database**: SQLite + cosine similarity for top-k retrieval.
* **Response Generation**: Summarized using small LLM (Ollama or Hugging Face API).

**2️⃣ Vision Model**

* **Model**: Salesforce/blip-image-captioning-base
* **Framework**: Hugging Face Transformers
* **Purpose**: Generates captions and tags from images.

**🧪 Testing Steps**

**✅ Step 1: Environment Setup**

python -m venv venv

venv\Scripts\activate

pip install -r requirements.txt

**✅ Step 2: Test Telegram Token**

python extras/test\_token.py

Expected Output:

✅ Token is valid!

Bot name: GenAI Helper Bot

Bot username: @genai\_helper\_bot

**✅ Step 3: Add Documents to RAG**

python -m extras.add\_docs

Expected Output:

🎉 All documents added successfully!

**✅ Step 4: Run the Bot**

python telegram\_bot.py

Expected Log:

INFO telegram\_bot - Starting bot...

INFO rag - Index rebuilt: 9 docs

**✅ Step 5: Test Commands in Telegram**

| **Command** | **Description** | **Example** |
| --- | --- | --- |
| /start | Start the bot | Hello message |
| /help | Show available commands | Interactive menu |
| /ask <query> | Ask a question from docs | /ask What is the leave policy? |
| /image | Upload an image after this command | Gets AI caption |
| /summarize | Summarize last few interactions | /summarize |

**🧾 Sample Outputs**

**🧠 RAG Query Example**

**User:** /ask What is the leave policy?  
**Bot:**

Employees are entitled to 12 days of annual leave per year.  
*(Source: policy.md)*

**🖼️ Vision Caption Example**

**User:** /image (uploaded image of cat)  
**Bot:**

Caption: A brown cat sitting on a chair  
Tags: brown, cat, sitting

**🧰 Logging and Debugging**

All runtime logs are stored in:

logs/bot.log

Includes:

* Startup messages
* User command traces
* Error stack traces
* Model load information

**🧠 Performance Optimization**

| **Optimization** | **Description** |
| --- | --- |
| ✅ Local embeddings | No API dependency |
| ✅ SQLite vector storage | Lightweight and fast |
| ✅ Caching | Avoids re-embedding same text |
| ✅ GPU/CPU Auto-detection | Uses CUDA if available |
| ✅ Compact BLIP model | Reduces load time |