WhatsApp-Integrated Retrieval-Augmented Generation (RAG) System

Objective

This system enables two-way conversational AI through WhatsApp, powered by:

- LangGraph
- OpenAI (GPT-3.5)
- Qdrant (Vector DB)
- Twilio (WhatsApp API)
- Flask + Ngrok for webhook tunneling

Overview Flow

1. User sends a WhatsApp message
2. Twilio Webhook receives it
3. Ngrok tunnels it to Flask
4. LangGraph queries context from Qdrant
5. GPT-3.5 generates a response
6. Response is sent back to user

Libraries & Their Purpose

Library	Purpose
`os`	Access environment variables
`dotenv`	Load .env config
`flask`	Web server for Twilio webhook

Library	Purpose
`pyngrok`	Create public URL for Flask
`twilio.twiml.messaging_response`	Format WhatsApp reply
`langchain_core`	Document structures
`RecursiveCharacterTextSplitter`	Chunk long texts
`OpenAIEmbeddings`	Convert text to embeddings
`ChatOpenAI`	Call GPT-3.5 API
`qdrant_client`	Qdrant DB connection
`VectorParams`, `Distance`	Set vector DB config
`StateGraph`, `RunnableLambda`	LangGraph DAG setup

Step-by-Step Code Breakdown

Step 1: Load Environment

```
from dotenv import load_dotenv
load_dotenv()
```

Loads `.env` file containing credentials like `OPENAI_API_KEY`, `TWILIO_AUTH_TOKEN`, `JIRA_API_TOKEN`, etc.

Step 2: Load & Chunk Text

```
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
texts = text_splitter.split_text(raw_text)
documents = [Document(page_content=chunk) for chunk in texts]
```

Breaks text into manageable chunks with overlap to preserve context.

Step 3: Generate Embeddings

```
embedding_function = OpenAIEmbeddings()
```

Embeds each text chunk into 1536D vector using OpenAI.

Step 4: Initialize Qdrant

```
qdrant_client = QdrantClient(host="localhost", port=6333)
qdrant_client.recreate_collection(
    collection_name="rag_collection",
    vectors_config=VectorParams(size=1536, distance=Distance.COSINE)
)
db = Qdrant(client=qdrant_client, collection_name="rag_collection", embeddin
gs=embedding_function)
db.add_documents(documents)
```

Connects to Qdrant and stores the embedded documents.

Step 5: Define Shared LangGraph State

```
class GraphState(TypedDict):
    question: str
    context: str
    answer: str
```

Tracks question, retrieved context, and answer.

Step 6: Retrieval Node

```
def retrieve(state: GraphState):
    query = state["question"]
    retriever = db.as_retriever()
    docs = retriever.invoke(query)
    context = "\n\n".join([doc.page_content for doc in docs])
    return {"question": query, "context": context}
```

Fetches relevant documents from Qdrant.

Step 7: Generate Node

```
llm = ChatOpenAI(model="gpt-3.5-turbo")
def generate(state: GraphState):
    prompt = f"""Answer the question using this context:\n\n{state['context']}\n
\nQuestion: {state['question']}"""
    response = llm.invoke(prompt)
    return {**state, "answer": response.content}
```

Uses GPT-3.5 to generate response.

Step 8: LangGraph Pipeline

```
graph = StateGraph(GraphState)
graph.add_node("retrieve", RunnableLambda(retrieve))
graph.add_node("generate", RunnableLambda(generate))
graph.set_entry_point("retrieve")
graph.add_edge("retrieve", "generate")
graph.add_edge("generate", END)
app = graph.compile()
```

Creates a 2-node graph: `retrieve → generate → END`

Step 9: Flask + Ngrok Webhook

```
@app.route("/whatsapp", methods=["POST"])
def whatsapp_webhook():
    incoming_msg = request.values.get('Body', '').strip()
    inputs = {"question": incoming_msg}
    result = app.invoke(inputs)
    answer = result.get("answer", "Sorry, I couldn't process your question.")

    response = MessagingResponse()
    msg = response.message()
    msg.body(answer)
    return str(response)
```

Receives WhatsApp message via Twilio, invokes LangGraph, replies.

Step 10: Launch with Ngrok

```
public_url = ngrok.connect(5000)
print(f"Ngrok Tunnel: {public_url}/whatsapp")
app.run()
```

Launches server at a public Ngrok URL to receive messages via Twilio.

Summary

Step	Action
1.	Load environment variables
2.	Load & chunk source text
3.	Generate embeddings
4.	Store embeddings in Qdrant
5.	Retrieve context from Qdrant
6.	Generate answer with GPT-3.5
7.	Reply via Twilio (WhatsApp)