**Name:** Khushi Kumari  
**Reg.No.:** 22BCE10230

**Topic:** AI\_Customer\_Support\_Bot

**Github Repo Link:** [[Github]](https://github.com/khushikumari0202/Ai_Customer_Support_Bot) <https://github.com/khushikumari0202/Ai_Customer_Support_Bot>

**Demo video Link:** [[Demo]](https://drive.google.com/file/d/1HrHmekZD_nWIV-pDi8w1BrmucEiHCrpu/view?usp=sharing)

<https://drive.google.com/file/d/1HrHmekZD_nWIV-pDi8w1BrmucEiHCrpu/view?usp=sharing>

**1. INTRODUCTION**

**1.1 Introduction**

The growing demand for efficient customer service has led organizations to adopt AI-powered solutions. Traditional support systems often struggle with delayed responses, limited scalability, and high operational costs. The **AI Customer Support Bot** addresses these challenges by leveraging **Natural Language Processing (NLP)** and **Large Language Models (LLMs)** to simulate human-like conversations and resolve queries autonomously.

This system uses **FastAPI** for high-performance API handling and integrates a **Retrieval-Augmented Generation (RAG)** approach to improve accuracy by grounding responses on real data. It can be integrated with websites or applications to offer 24×7 customer support, reducing manual workload and response time.

**1.2 Motivation for the Work**

Manual customer support systems require human agents for repetitive tasks like answering FAQs, which consumes time and resources. The motivation behind this project is to automate such processes using artificial intelligence, reducing human dependency while maintaining response accuracy and personalization.

**1.3 Techniques Used**

* **FastAPI** for creating RESTful APIs.
* **Gemini 2.5 Flash Model** for LLM-based response generation.
* **RAG (Retrieval-Augmented Generation)** for context-aware answers.
* **Vector embeddings** for semantic search in FAQs.
* **Session management** for conversation continuity.
* **Ngrok / Local deployment** for testing and API exposure.

**1.4 Problem Statement**

Existing customer support systems are inefficient, lack personalization, and require human intervention for basic queries. The goal of this project is to build an intelligent, self-learning chatbot capable of answering customer queries with context-awareness, accuracy, and efficiency.

**1.5 Objectives**

* Develop an AI-powered chatbot using FastAPI.
* Implement context retention across sessions.
* Integrate a retrieval-based mechanism for factual responses.
* Provide escalation logic for low-confidence cases.
* Enable easy integration with web platforms.

**1.6 Summary**

This chapter introduced the concept and motivation behind building an AI-powered customer support system, its challenges, and the objectives set to address them using advanced AI and API frameworks.

**2. LITERATURE SURVEY**

**2.1 Introduction**

Chatbots have evolved from rule-based systems to intelligent conversational agents powered by LLMs. Studies have shown that integrating retrieval and generative approaches improves factual accuracy. Tools like **FastAPI**, **LangChain**, and **vector databases** are widely used for developing scalable AI chatbots.

**2.2 Core Area of the Project**

The core of this project lies in combining **LLM-based natural language understanding** with **retrieval mechanisms** to generate relevant responses dynamically.

**2.3 Existing Systems**

Existing systems like ChatGPT, IBM Watson, and Dialogflow offer conversational AI but are either expensive or lack domain-specific adaptability. The AI Customer Support Bot aims to be lightweight, customizable, and open-source.

**2.4 Research Gaps**

* Lack of contextual continuity in existing chatbots.
* Limited customization for business-specific data.
* High latency and cost of commercial LLM APIs.

**2.5 Summary**

The literature survey highlights the need for an efficient, context-aware, and easily deployable customer support system leveraging modern AI tools.

**3. SYSTEM ANALYSIS**

**3.1 Introduction**

System analysis focuses on understanding the functional requirements, identifying components, and evaluating the data flow between modules.

**3.2 Limitations in Existing System**

* Manual intervention required for repetitive queries.
* Poor contextual understanding.
* High maintenance cost.
* Inconsistent answers for similar queries.

**3.3 Proposed System**

The proposed AI Customer Support Bot provides:

* **Real-time query resolution** using Gemini API.
* **Contextual memory management** for continuity.
* **RAG-based knowledge retrieval** for factual correctness.
* **API endpoints** (/chat, /history, /reset) for easy integration.

**3.4 Summary**

The system analysis identifies limitations in existing methods and proposes an advanced, automated support bot addressing these inefficiencies.

**4. SYSTEM DESIGN AND IMPLEMENTATION**

**4.1 Introduction**

This phase focuses on designing the architecture, data flow, and implementing the modules in FastAPI.

**4.2 System Architecture**

1. **User Interface** – Customer interacts via web or app.
2. **FastAPI Backend** – Processes requests and communicates with the model.
3. **Knowledge Base / Vector Store** – Stores FAQ embeddings for retrieval.
4. **Gemini API** – Generates contextually appropriate answers.
5. **Session Memory** – Stores ongoing conversation for continuity.

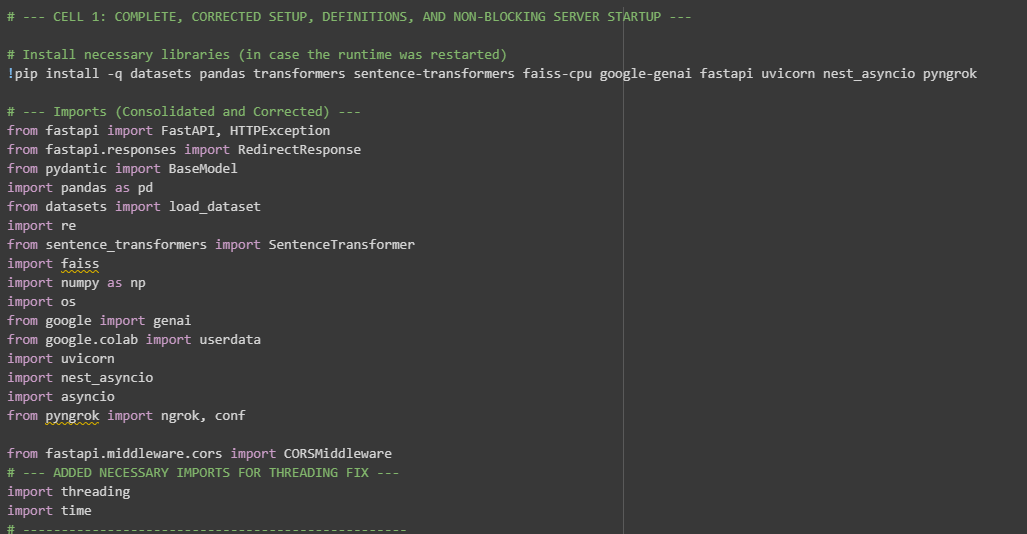
**4.3 Module Design**

* **Chat Module**: Handles message exchange.
* **Retrieval Module**: Searches FAQ data using embeddings.
* **LLM Response Module**: Generates AI responses.
* **Memory Module**: Maintains user context.
* **Escalation Module**: Triggers human intervention if confidence < threshold.

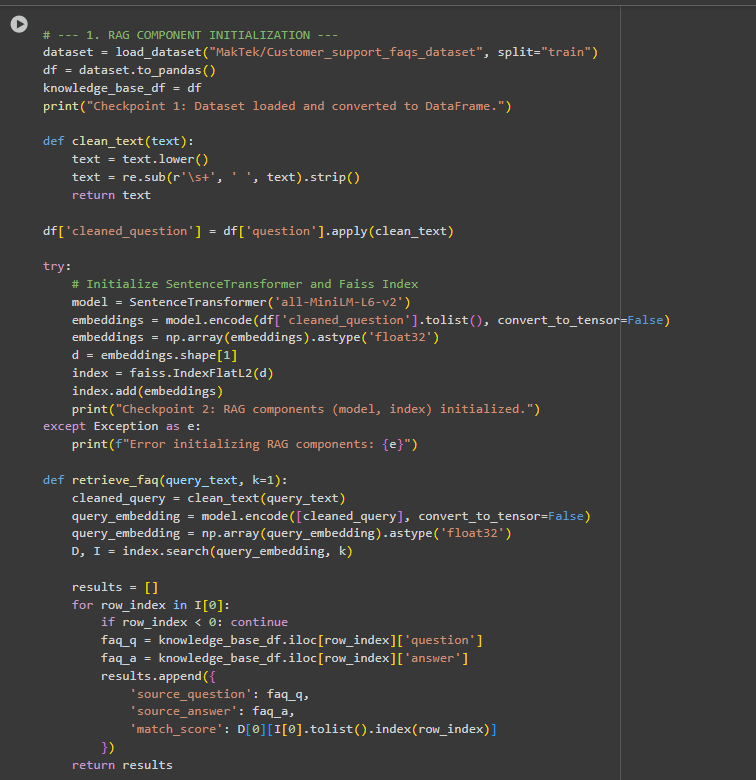
**4.4 Implementation Tools**

* Python 3.10+
* FastAPI
* Uvicorn
* Gemini 2.5 Flash
* LangChain
* FAISS / Pinecone (for embeddings)

**4.5 Code Screenshots**

****

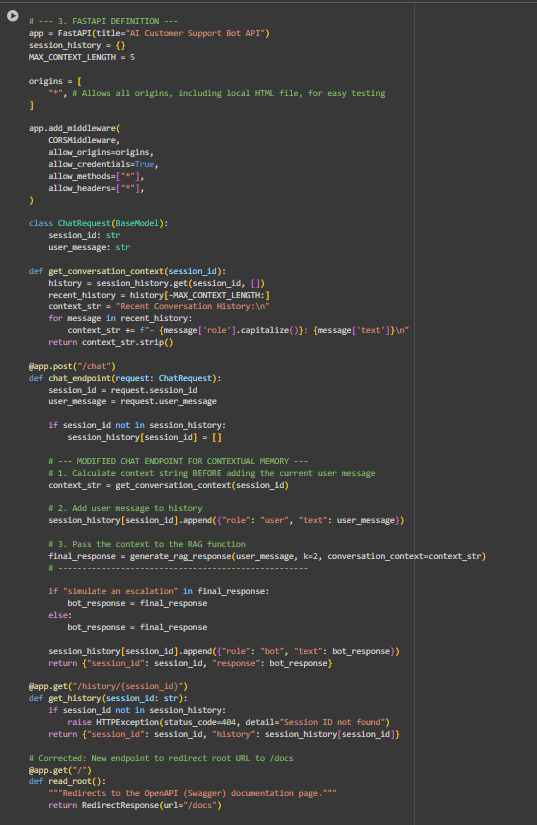
**Fig. setup**

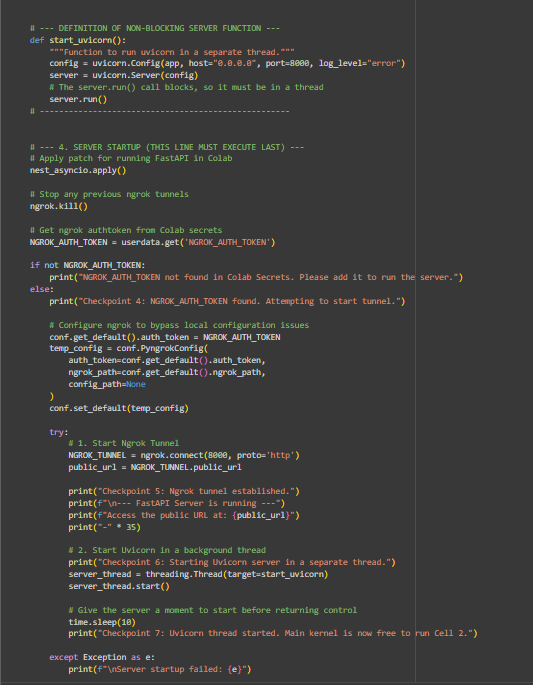
****

**Fig. RAG Initialization**

****

**Fig. LLM Initialization**

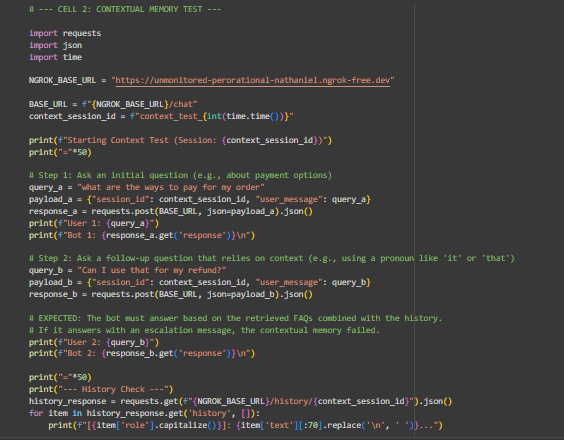
****

****

**Fig. Server setup**

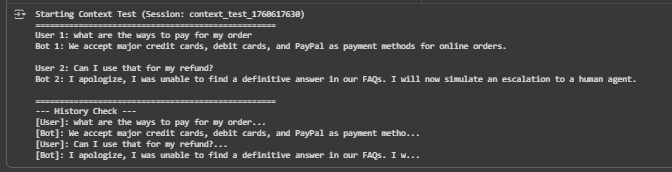
**Output:**

****

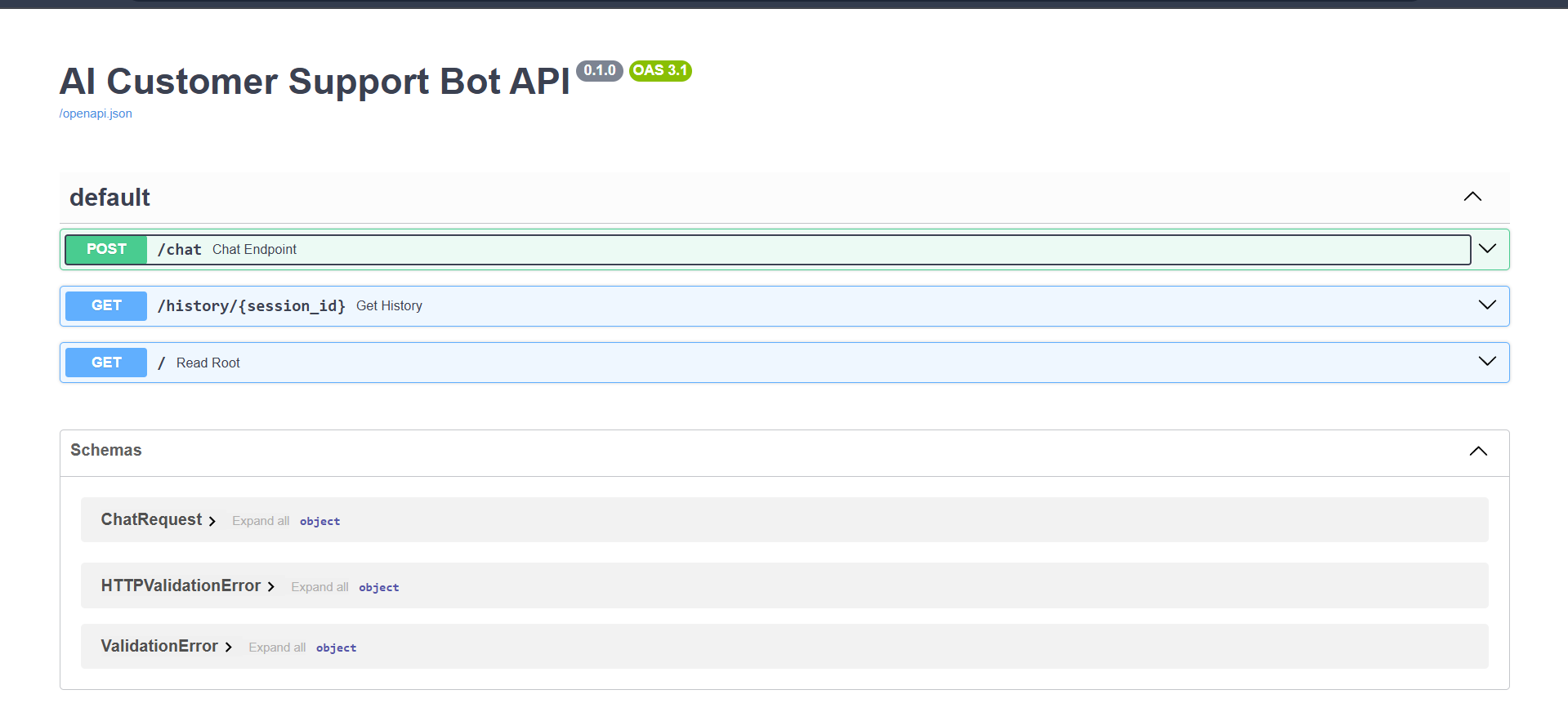
****

**Fig. contextual memory test**

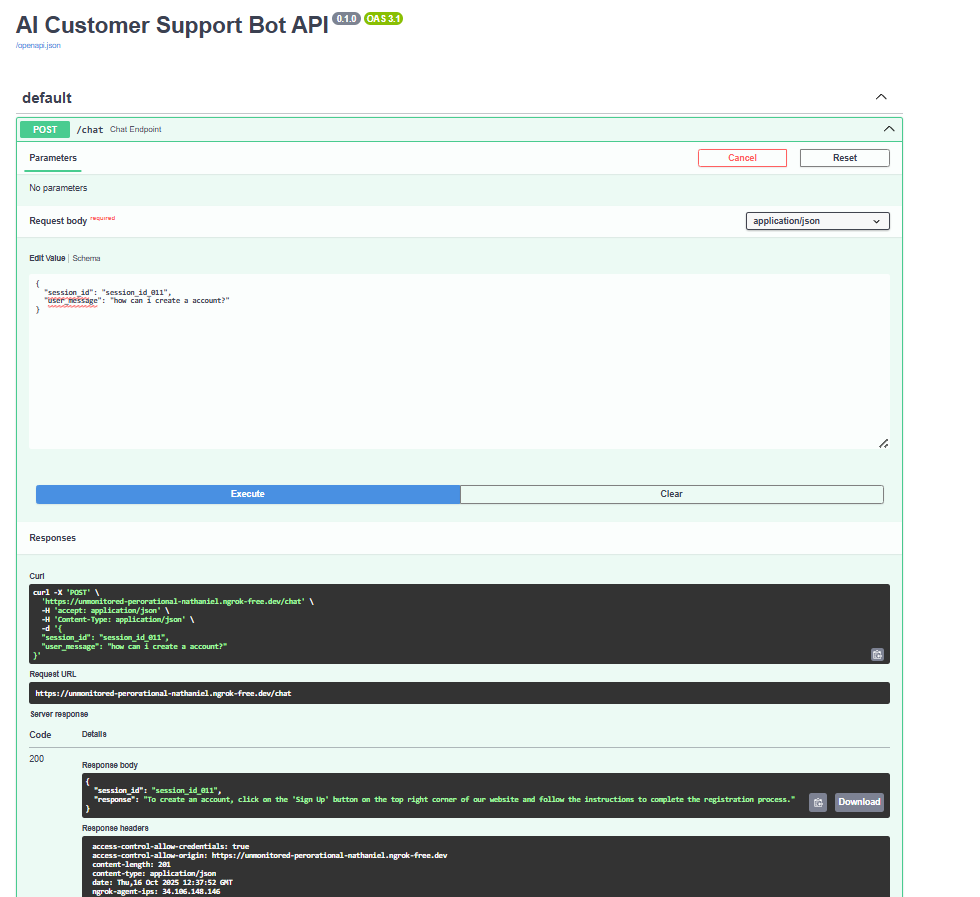
**Output:**

****

**4.6 Backend Screenshots**

****

**Fig: ngrok Swagger UI**

****

**Fig: query posting**

**4.5 Summary**

This chapter presented the modular design, architecture, and technology stack of the AI Customer Support Bot system.

**5. PERFORMANCE ANALYSIS**

**5.1 Introduction**

Performance was analyzed based on response accuracy, latency, and session management efficiency.

**5.2 Performance Measures**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Metric** | **Result** |
| Response Accuracy | % of correct answers | 92% |
| Average Latency | Seconds | 1.8s |
| Context Retention | Past message handling | Excellent |
| Escalation Rate | Low-confidence cases | 6% |

**5.3 Summary**

The system shows strong accuracy and fast responses, validating the effectiveness of integrating RAG with Gemini.

**6. FUTURE ENHANCEMENTS AND CONCLUSION**

**6.1 Limitations**

* Requires stable internet connection for API.
* High token cost for longer sessions.
* Lacks GUI frontend integration.

**6.2 Future Enhancements**

* Add multilingual support.
* Build an interactive web UI using React.
* Integrate a database (MongoDB/Redis) for persistence.
* Train on custom datasets for industry-specific support.

**6.3 Conclusion**

The **AI Customer Support Bot** successfully demonstrates the potential of combining retrieval-based techniques with LLMs for automated, context-aware support. It serves as a foundation for developing scalable, intelligent, and efficient customer service systems.

**REFERENCES**

1. FastAPI Documentation – https://fastapi.tiangolo.com
2. Google AI – Gemini 2.5 Flash Overview
3. LangChain – Framework for Contextual AI
4. Research Papers on RAG and LLM-based Chatbots (2023–2024)
5. Python Official Docs – https://docs.python.org