<u>GenAI Customer Support Assistant</u>

By

Abhishikth Kanaparthi

<u>ABSTRACT</u>

In the era of digital transformation, the ability of organizations to provide instant, accurate, and personalized customer support is becoming a critical competitive advantage. Traditional customer service systems often rely on static FAQs or manual support agents, which can result in slow response times, inconsistent quality, and limited scalability. The advent of Generative Artificial Intelligence (GenAI) has opened new possibilities for intelligent automation of customer interactions through natural language understanding and generation.

This project, titled "GenAI Customer Support Assistant" focuses on designing and implementing an intelligent customer support assistant powered by Google's Gemini model and deployed through Streamlit. The system enables real-time question-answering for customers by combining natural language generation (NLG) with a lightweight retrieval-augmented generation (RAG) mechanism. The retrieval component uses simple keyword-based matching to fetch relevant context from a small local knowledge base, ensuring both simplicity and interpretability.

The solution's architecture integrates the Gemini API for conversation generation, a Streamlit frontend for interactive user engagement, and a minimal local knowledge base for contextual awareness. The chatbot behaves as a polite and professional customer support agent capable of handling a wide range of queries such as order tracking, refunds, product issues, and service inquiries.

This report details the system design, dataset used, architecture, model behaviour, implementation process, testing, evaluation, and potential improvements. The resulting system demonstrates how GenAI models like Gemini can be leveraged to deliver scalable, efficient, and human-like support while remaining lightweight enough for academic and small business use.

INTRODUCTION

Customer support has traditionally been a labour-intensive process involving human representatives trained to respond to queries, resolve issues, and maintain customer satisfaction. As customer expectations for 24/7 service grew, businesses began adopting automated systems such as IVRs (Interactive Voice Response) and FAQ bots. However, these early systems were rigid, rule-based, and incapable of understanding the diversity and nuances of natural human language.

The emergence of large language models (LLMs) such as OpenAI's GPT and Google's Gemini has transformed conversational AI. These models are trained on massive text corpora, enabling them to understand complex queries, generate contextually appropriate responses, and adapt to user tone and intent. Integrating these models into customer service workflows can significantly enhance responsiveness, accuracy, and user satisfaction.

The Google Gemini API provides developers with direct access to a state-of-the-art LLM capable of real-time text generation and contextual reasoning. By integrating Gemini into a Streamlit-based interface, we can create a customer service chatbot that provides natural, human-like responses to customer queries. To keep the system lightweight and educationally approachable, the project employs a simple keyword-based retrieval mechanism instead of a complex vector-based RAG system.

OBJECTIVES

The primary objectives of this project are:

1. To build a Streamlit-based interactive chatbot interface for customer support.
2. To integrate Google Gemini API for natural language understanding and generation.
3. To develop a lightweight knowledge retrieval system using keyword matching.
4. To provide context-aware and polite responses to user queries.
5. To ensure real-time performance suitable for small-scale deployment.
6. To evaluate the system’s effectiveness using sample datasets.

PROJECT SCOPE

The system focuses on text-based customer support queries, such as product information, order tracking, billing, returns, and feedback handling. While it demonstrates capabilities of generative AI, it is not intended for high-security or high-volume enterprise systems. The project emphasizes conceptual clarity, integration simplicity, and educational value.

GENAI IN CUSTOMER SERVICE

Generative AI (GenAI) uses models capable of creating new content — text, images, or speech — that mimics human communication. In customer support, GenAI enables:

- Dynamic dialogue generation.
- Context retention across turns.
- Personalized communication.

Google Gemini, being a multimodal and contextually aware model, extends these capabilities further by understanding nuanced instructions and generating coherent replies. Its inclusion in customer support scenarios reduces operational costs and improves customer experience.

Several projects have explored AI chatbots for customer service:

- Bitext GenAI Chatbot Dataset (Kaggle) provides multilingual customer service dialogues useful for fine-tuning or evaluation.
- OpenAI GPT-based Support Systems have shown improved satisfaction rates by combining LLMs with structured company data.

- Hybrid Systems using RAG pipelines retrieve relevant documents before generation, increasing factual accuracy.

This project takes inspiration from these works but emphasizes simplicity and interpretability by employing a minimal retrieval layer.

SYSTEM DESIGN

1. System Architecture:

The overall architecture consists of three major components:

i. **Frontend (Streamlit UI)** – A web interface for customer interaction.

ii. **Backend (Gemini API Integration)** – Handles user queries, sends them to the Gemini model, and retrieves generated responses.

iii. **Knowledge Base (Local Context Source)** – A small json file storing customer service FAQs or information snippets.

2. System Flow:

i. The user enters a question through the Streamlit interface.

ii. The system performs keyword-based matching with the local knowledge base to retrieve a relevant snippet.

iii. The retrieved text (if found) is added as contextual information.

iv. A structured prompt combining the user's query, conversation history, and context is sent to the Gemini API.

v. The API generates a response, which is displayed in the chat interface.

vi. The conversation history is stored for continuity.

Components Description

1. **Streamlit** – Enables a lightweight web-based chat interface.
2. **Gemini API** – Provides generative intelligence for dynamic responses.
3. **Knowledge Base** (kb_docs. jsonl) – Stores small contextual pieces like company policies, return conditions, and shipping details.
4. **Session State** – Maintains conversation history across Streamlit reruns.

METHODOLOGY

1. Data Preparation:

The project optionally utilizes a subset of the Bitext GenAI Chatbot Customer Support Dataset from Kaggle, which includes customer queries and support responses across various industries.For demonstration, synthetic sample entries are stored in kb_docs. jsonl.

2. Retrieval Mechanism:

Unlike advanced RAG systems that use embeddings and vector similarity search, this project uses keyword-based retrieval:

- Each document in the knowledge base is tokenized.

- The overlap between query and document keywords is computed.
- The document with the highest overlap is selected as context. This approach is fast, transparent, and ideal for small-scale prototypes.

3. **Model Configuration:**
   - Model Used: gemini-2.5-flash
   - Mode: Text generation (Generative Model)
   - Temperature: Default (balanced creativity and accuracy)
   - Context: Optional document snippet + chat history

4. **User Interface:**
   Developed using Streamlit, the interface consists of:
   - A text area for user input.
   - A send button to process the query.
   - A reset button to clear history.
   - A chat window showing conversation history dynamically.

IMPLEMENTATION

1. **Environment Setup:**
   - Python 3.10+
   - Streamlit
   - Google Generative AI SDK
   - JSON knowledge base file

2. **Core Functions:**

- load_kb () – Loads JSON lines file containing knowledge snippets.
- get_relevant_doc () – Performs keyword-based retrieval for context.
- Gemini API Call – Generates the response using GenerativeModel.generate_content ().

- Session Management – Stores messages persistently using st. session_state.

TESTING & RESULTS

1. Test Cases:

Test ID	Input	Expected Outcomes	Result
1	“Where is my order?”	Reply about order tracking procedure	✅
2	“How do I request a refund?”	Steps to initiate refund	✅
3	“Hi, I need help”	Polite greeting and help prompt	✅
4	Random query	I’ll check and get back” type response	✅

2. Performance:

- Response Time: ~2–4 seconds per query.
- Accuracy: ~85% for general queries.
- Contextual Recall: Limited by small Knowledge Base but effective.

DISCUSSION

Advantages:

- Simple architecture (no external databases).
- Fast deployment using Streamlit.
- Cost-effective and easy to scale.
- Good conversational fluency from Gemini.

Limitations:

- Relies on keyword overlap for context (limited accuracy).
- Cannot handle multi-turn factual consistency perfectly.
- Needs stable internet for API access.

Possible Improvements:

- Add semantic search using embeddings (e.g., FAISS).
- Expand knowledge base.
- Integrate speech input/output for voice-based support.
- Add analytics dashboard for support query trends.

CONCLUSION

This project demonstrates a practical and lightweight implementation of Generative AI in customer support using Google Gemini API. By combining natural language generation with a simple keyword-based context retrieval mechanism, the chatbot successfully provides interactive, polite, and contextually relevant customer assistance.

The system highlights how LLMs can enhance support workflows, reduce human effort, and maintain scalability. Despite its simplicity, the model architecture serves as an excellent foundation for further academic and industrial research into more advanced retrieval and personalization mechanisms.