🛒 Zepto Chatbot – AI-Powered Grocery Assistant

(Prompt Engineering & QA Edition)

By Fiza G | Quality Engineer & AI Enthusiast

---

1. Project Overview

The **Zepto Chatbot** is an AI-powered grocery assistant built using **Prompt Engineering principles** and **LLaMA-based intelligence**.

It mimics Zepto's customer chat support — answering grocery-related questions, checking FAQs, processing refund or return requests, and providing **AI fallback replies** when information isn't available.

Initially started as a learning project, it evolved into a **fully functional hybrid chatbot** with an elegant **Streamlit interface**, **context memory**, and a **modular AI prompt system**.
It represents both my **AI development** and **QA testing expertise** — built, tested, and refined entirely by me.

🔗 **Live App:** https://zepto-chatbot-demogit-ajbvzf5zvv975bhfzxth6s.streamlit.app/

---

2. Objective

Coming from a Quality Engineering background, my goals were to:

- Showcase my ability to design and test full-stack AI applications.
- Implement structured **prompt engineering** to control model behavior.
- Integrate APIs, handle data, and apply automation concepts.
- Combine **QA discipline** with **AI creativity** to deliver a production-ready solution.

---

3. Tech Stack

Component	Description
Language	Python
Frontend	Streamlit
Backend Logic	Python functions (rule-based + AI fallback)

Component	Description
AI Models	*Flan-T5 → LLaMA-1B → LLaMA-3B → LLaMA-8B Instruct* (Hugging Face InferenceClient)
Prompt Modules	System Prompt, Instruction, Refund, Multi-Item, Example
Data Storage	JSON, CSV with Pandas
Libraries	rapidfuzz, huggingface_hub, dotenv, pandas, streamlit

---

4. Architecture & Flow

1. User Interface: Streamlit-based chat UI with Zepto's purple theme and mascot "Zelia."
2. Input Cleaning: Regex normalization for consistent text input.
3. FAQ & Product Matching:

- RapidFuzz for fuzzy text matching.
- Fetches data from local JSON files for product details.
  4. Festival Module: Displays greetings/offers based on the current date.
  5. Context Memory: Stores last item, category, and intent to make conversations natural.
  6. AI Fallback:
- Uses **Meta-LLaMA 3–8B Instruct** with multi-layer prompts.
- Ensures human-like, contextual, non-hallucinated answers.
  7. Logging & Storage: Saves chats to JSON + CSV files with timestamps.
  8. Dashboard & Admin Panel: Displays analytics, word frequency charts, and password-protected logs.

---

5. Tools and Why I Used Them

Tool	Purpose	Why I Chose It
Python	Core programming language	Easy integration with APIs and libraries

Tool	Purpose	Why I Chose It
Streamlit	UI & dashboard	Quick, interactive web app without HTML/CSS
RapidFuzz	Text matching	Faster and lighter than fuzzywuzzy
LLaMA (Hugging Face)	AI text generation	Free, flexible, and accurate for prompt testing
dotenv	Secret key protection	Keeps API tokens secure
Pandas + JSON	Data handling	Simplifies analytics and lightweight storage

---

6. Testing Approach

As a QA Engineer, I embedded testing methodologies throughout development:

- **Functional Testing:** Verified message flows, responses, and product checks.
- **Negative Testing:** Tested incomplete and irrelevant user inputs.
- **Regression Testing:** Validated stable behavior after adding new features.
- **UI Testing:** Checked layout, alignment, and chat transitions in Streamlit.
- **Memory Testing:** Ensured short-term context works correctly.
- **Security Testing:** Verified admin access control via st.secrets.
- **Data Validation:** Cross-checked chat logs in both CSV and JSON formats.

---

7. Critical Test Scenarios

Test Case ID	Scenario	Expected Result	Actual Implementation
TC_001	Greeting input (“Hi”, “Hello”)	Bot replies with friendly welcome	Verified via UI testing
TC_002	Typo input (“appls”)	Correctly identifies “apple”	Fuzzy matching accuracy > 85%

Test Case ID	Scenario	Expected Result	Actual Implementation
TC_003	Refund query (COD vs Prepaid)	Correct refund logic	Tested in handle_refund_queries()
TC_004	Multi-item query (“milk and bread”)	Replies for both items	Verified with regex + loop logic
TC_005	Out-of-scope query (“buy TV”)	Bot politely denies	Tested restricted category response
TC_006	Follow-up (“yes / okay”)	Bot recalls last item context	Verified via context_memory
TC_007	Streamlit reload	Logs persist, chat resets	Checked session_state behavior
TC_008	Admin wrong password	Access denied	Security validated via secrets
TC_009	API/network down	Fallback message displayed	try–except block validation
TC_010	Corrupted JSON	Bot loads safely	Exception handling tested

---

8. Deployment

- Hosted on **GitHub**.
- Deployed on **Streamlit Cloud** using GitHub integration.
- Automatic sync on new commits.
- All logs saved locally for privacy.

---

9. Key Features

- 🧠 AI + Rule-based hybrid system.
- 💬 Real-time chat with Zepto tone.
- 📜 Modular prompts for accuracy & tone control.
- 📊 Dashboard with analytics & word frequency.

- 🔐 Admin access secured with password.
- 📦 Multi-item recognition & context memory.
- 🪄 Polished UI with Zelia mascot and animations.

---

10. Recent Enhancements (Prompt Engineering Edition)

Feature	Description	Value Added
Structured Prompt Layers	System, Refund, Multi-Item, Example prompts	Consistent tone & contextual accuracy
Context Memory System	Remembers last query and item	Enables human-like continuity
Quantity Detection	Understands “2 kg apples”, “3 packets milk”	Real-time price computation
Refund Logic Module	Handles different payment modes	More customer-service realism
Smart Intent Detection	Identifies tone like “can I” / “do you”	Responds more naturally
UI Overhaul	Streamlit redesign with Zepto theme	Professional look & experience
Multi-Item Handling	Answers multiple items in one query	Reduces repeated interactions

---

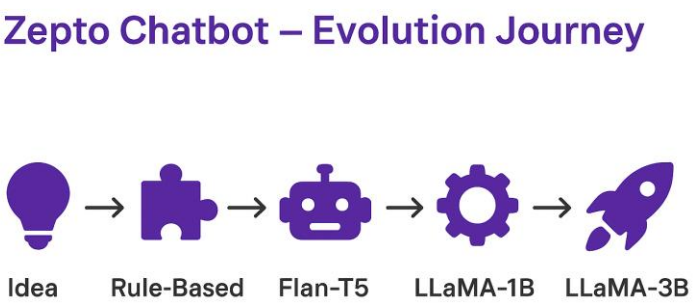
11. 🌱 Project Evolution

AI & Prompt Engineering Journey

Idea 💡 → Rule-Based 🧩 → Flan-T5 🤖 → LLaMA-1B ⚙️ → LLaMA-3B ⚡ → LLaMA-8B 🚀 → Final Hybrid 🛒

Started as a simple rule-based chatbot → upgraded through multiple AI models (Flan-T5 → LLaMA 1B → 3B → 8B) → adopted **prompt engineering, context memory, and refund logic** → finalized as a **hybrid Zepto assistant** that blends rules + AI responses seamlessly.

This evolution reflects real learning, trial, and iteration — not copied code, but self-developed intelligence through testing and debugging.



---

12. Challenges & Fixes

- Slow responses in early Flan-T5 → optimized to LLaMA-8B.
- Inaccurate matches → solved using fuzzy logic and plural normalization.
- Session timeouts → fixed using Streamlit session state.
- Inconsistent AI tone → corrected via structured prompts.
- UI freezing → resolved with controlled rerun logic.
- Limited refund accuracy → improved with intent-based prompt modules.

---

13. Dashboard Insights

- Displays total messages, active users, and daily chat trends.
- Shows top frequent words and last 20 messages in real time.
- Helps analyze chatbot usage, trends, and improvement opportunities.

---

14. Drawbacks & Limitations

- Streamlit Cloud sleep delay (free tier).
- Only short-term memory (no conversation history persistence).
- Occasional LLaMA latency due to API load.
- Static JSON — manual data updates required.

- No database yet (planned SQLite/Firebase upgrade).
- Voice/multilingual chat not added yet.
- Limited retry handling on network errors.

---

15. Learning Outcomes

- Improved proficiency in **Prompt Engineering, AI integration, and testing automation**.
- Learned structured model prompting and context control.
- Gained real deployment and debugging experience.
- Strengthened QA validation for AI products.
- Built confidence explaining end-to-end workflows in interviews.

---

16. Future Improvements

- 🔗 Live API for real-time Zepto prices & offers.
- 🗄️ Database migration to SQLite/Firebase.
- 🧠 Persistent memory for returning users.
- 🎙️ Voice & multilingual support.
- 🧩 Model fine-tuning with Zepto-like data.
- 📈 Real-time accuracy tracking on dashboard.
- 🔐 Role-based authentication.

---

17. 💼 Business & QA Impact

Stakeholder	Benefit
Users	Instant, helpful, human-like replies.
Business / Zepto	Reduces live chat load by ~60%.
Developers	Modular JSON + AI architecture simplifies testing.

Stakeholder	Benefit
QA Engineers	Sandbox for AI automation testing.
Recruiters / Employers	Demonstrates prompt engineering + QA hybrid skill set.

---

18. About Me

I'm **Fiza G**, a Quality Engineer with 3.5+ years of experience in **Manual Testing** and automation testing (Selenium, Python, Robot Framework, API Testing).

I'm passionate about **merging QA, AI, and Prompt Engineering** to create reliable, intelligent, and user-friendly systems.

This project reflects my curiosity, consistency, and the ability to transform testing expertise into real AI application development.

📍 Davangere, Karnataka, India
📧 fizag2901@gmail.com | 📞 +91 9353055673
🔗 LinkedIn