Weekly Project Report

Name: Anbhi Thakur
Date: 17 August 2025

---

1. Introduction

This week's work was centered on extending and refining the LLM-powered chatbot system into a more complete, production-ready solution. The chatbot was originally developed as a basic conversational interface with document upload and retrieval capabilities, but the scope of work has steadily expanded to integrate advanced features such as **vector embeddings, Retrieval-Augmented Generation (RAG), agentic AI behavior, function calling, and now an Answer Consistency Check mechanism**.

The ultimate aim is to create a **humanoid conversational assistant** that not only responds to user queries but also reasons about its own answers, checks for accuracy, and has the flexibility to extend its capabilities by calling external tools.

This report documents the development lifecycle, challenges, design decisions, and future enhancements.

---

2. Summary of Progress

The key milestones addressed this week were:

1. **Baseline Chatbot with Docker Deployment**
   - FastAPI backend built for chat and file analysis.
   - Streamlit frontend developed as a **ChatGPT-style clone** for user interaction.
   - Deployment tested using Docker with simultaneous FastAPI + Streamlit processes.
2. **Attachment Analysis Functionality**
   - Integrated support for **PDFs, images, and plain text files**.
   - Implemented OCR with **Tesseract** for scanned/image-based documents.
   - Implemented text chunking, OpenAI embeddings, and semantic search.
   - Automatic summarization of uploaded content using GPT-4o-mini.
3. **Vector Embedding & RAG**
   - Built custom in-memory vector database using `sklearn`'s Nearest Neighbors.
   - Enabled semantic retrieval of document chunks for context-aware answers.
   - Improved retrieval strategy with overlapping chunking.

4. **Agentic AI Extension**
    - Added an `/agent` endpoint that enables reasoning with **trace outputs**.
    - Implemented scaffolding for tool usage and multi-step reasoning.
5. **Function Calling Framework**
    - Designed system to allow the chatbot to **trigger external tools**.
    - Sample tools considered: web search, code runner, and data formatter.
    - Modularized so new tools can be plugged in without refactoring.
6. **Answer Consistency Check Mechanism** (New)
    - A new feature introduced this week.
    - Two strategies explored:
        1. **Dual-Response Generation:** For every query, generate two answers with slightly varied prompts, then compare for semantic overlap.
        2. **Self-Reflection Pass:** After generating an answer, prompt the model again with:
           "Review the answer above. Is it consistent, complete, and correct? If not, refine it."
    - This ensures answers are not only correct but **more reliable and self-validated**, mimicking a human double-checking their own work.

---

3. Technical Implementation

3.1 Backend (FastAPI)

- Endpoints:
    - `/chat` – Core conversation with optional RAG.
    - `/upload` – Handles file uploads, extraction, embedding, and summarization.
    - `/docs/list` – Returns stored chunks metadata.
    - `/agent` – Delegates to agentic reasoning module.
- Key Libraries:
    - **PyMuPDF (fitz):** PDF parsing.
    - **Pytesseract + Pillow:** Image OCR.
    - **Scikit-learn:** Nearest neighbor search for RAG.
    - **FastAPI + Uvicorn/Gunicorn:** Backend service.

3.2 Frontend (Streamlit)

- Sidebar navigation with five modules:
    1. Chatbot
    2. Upload & Analyze
    3. Stored Chunks
    4. Agentic AI
    5. Agent Tools
- UI styled to resemble a ChatGPT-like environment:

    - `st.chat_input` for messages.
    - `st.chat_message` for role-based conversation display.
    - JSON viewer for agent traces.

3.3 Answer Consistency Mechanism

Implemented in backend within `/chat` pipeline:

- **Step 1:** First generate Answer A (normal response).
- **Step 2:** Generate Answer B with a rephrased system prompt.
- **Step 3:** Compare semantic similarity (using embeddings).
- **Step 4:** If similarity is low, trigger a refinement step.
- **Step 5:** Deliver the consistent and reviewed final answer.

This brings a **"second pair of eyes"** effect, reducing hallucinations and making the chatbot feel more trustworthy.

---

4. Development Lifecycle

1. **Research Phase**
    - Studied RAG techniques, OCR tools, and lightweight vector storage.
    - Benchmarked different consistency-checking methods (self-reflection vs dual answers).
2. **Implementation Phase**
    - Refactored backend to modularize embedding storage.
    - Integrated OCR and hybrid PDF parsing (text + fallback to OCR).
    - Added agent routing and tool scaffolding.
    - Implemented consistency check logic as a middleware.
3. **Testing Phase**
    - Tested with PDFs (text-based vs scanned).
    - Uploaded images of handwritten notes → OCR worked reliably with ~90% accuracy.
    - Checked retrieval pipeline on a 20-page document; relevant chunks retrieved in <2s.
    - Consistency mechanism tested with factual Q&A (e.g., math queries, history facts).
4. **Challenges**
    - Docker build failed due to unavailable `pytesseract==0.3.11` → resolved by downgrading to `0.3.10`.
    - Errors in POST routes (`Cannot POST /chat`) fixed by ensuring frontend requests matched backend endpoints.
    - Ensuring low latency while generating dual responses required caching and batch API calls.

5. Evaluation

- **Accuracy:** Improved with RAG + consistency checks.
- **Reliability:** More stable answers due to reflection mechanism.
- **User Experience:** Streamlit interface is clean, interactive, and intuitive.
- **Scalability:** Vector DB currently in-memory but can migrate to FAISS or Milvus.
- **Agentic Abilities:** Early but extendable with more tools.

6. Future Enhancements

1. **Tool Expansion**
   - Implement real web search + sandboxed code execution.
   - Data formatting/conversion (CSV ↔ JSON, unit conversions, etc.).
2. **Advanced Answer Checking**
   - Move from pairwise comparison to a **confidence score** mechanism.
   - Integrate external fact-check APIs for knowledge grounding.
3. **Knowledge Persistence**
   - Switch from pickle + numpy to a production-grade vector store.
   - Add multi-user session persistence.
4. **Humanoid Behaviors**
   - More natural "thinking aloud" trace outputs.
   - Emotional alignment (empathetic tone when responding).

7. Conclusion

This week's work significantly advanced the chatbot from a simple interface into a **humanoid AI assistant** capable of:

- Conversational interaction
- File analysis (PDF, image, text)
- Retrieval-augmented answers
- Multi-step agentic reasoning
- Tool calling framework
- Self-checking answers through consistency mechanisms

The system now not only answers but **thinks about its answers**, creating a more reliable and human-like experience.