Contents

[Overview 1](#_Toc211855401)

[Components 1](#_Toc211855402)

[Data Flow Diagram 2](#_Toc211855403)

[Testing 3](#_Toc211855404)

[Deployment 3](#_Toc211855405)

[Repository Structure 3](#_Toc211855406)

[Tech Stack 4](#_Toc211855407)

## Overview

This system enables intelligent querying of environmental data using a LangChain-powered agent. It combines document-based retrieval (RAG), real-time API access (USGS), and local LLM inference (Ollama) to answer sustainability-related questions. The architecture is modular, extensible, and both Local and Docker driven for reproducibility.

## System Context Diagram

A group of rectangular colored rectangles

AI-generated content may be incorrect.

Fig 1.1 System Diagram showing main components

## Components

1. **Ingestion Pipeline (ingest.py)**

Responsible for embedding and indexing unstructured environmental documents.

* **Document Loader**: Loads PDFs, text files.
* **Chunking Strategy**: Splits documents into manageable semantic units.
* **Embedding Model**: Uses SentenceTransformer to convert chunks into vector embeddings.
* **Vector Store**: Stores embeddings in FAISS for fast similarity search.
* **Output**: Serialized FAISS index + chunk metadata for RAG retrieval.

2. **Agent Logic (AIBot.py)**

Implements the LangChain agent that orchestrates tool usage and LLM reasoning.

* **LLM**: Uses OllamaLLM for local inference.
* **Tools**:
  + RAG Retriever: Retrieves relevant chunks from FAISS index.
  + USGS Water API: Fetches water quality data via HTTP.
* **Agent Type**: ZERO\_SHOT\_REACT\_DESCRIPTION with embedded prompt-style tool descriptions.
* **Execution Flow**:
* Thought → Action → Action Input → Observation → Final Answer
* Max 3 iterations with fallback summary on timeout or parse failure.

3. **UI Interface (AIBot\_UI.py)**

Provides a simple frontend for user interaction using streamlit.

* **Framework**: Streamlit or FastAPI-based UI
  + **Features**:
* Text input for queries
* Final answer rendering

4. **Backend Orchestration (backend.py)**

Serves the agent via API or CLI.

* **FastAPI Server**: Exposes /ask endpoint for agent queries
* **Docker Entry point**: Runs uvicorn for containerized deployment
* **Logging**: Captures agent steps, tool usage, and errors

## Data Flow Diagram

**User Query** → UI or API

1. **Agent** → Selects Tool
2. **Tool** → RAG or USGS API
3. **Observation** → Returned to Agent
4. **LLM Reasoning** → Final Answer
5. **Response** → Rendered in UI or returned via API

A group of rectangular colored labels

AI-generated content may be incorrect.

Fig 1.2 Data Flow showing User Query to LLM Response

## Testing

* **Unit Tests**: test\_agent.py validates agent responses and tool triggers
* **Edge Cases**: Handles malformed input, tool name errors, and iteration limits

## Deployment

* **Docker**: Includes Dockerfile and requirements.txt.
* **Reproducible**: All dependencies pinned.
* **Portable**: API implementation with Frontend calling API to Backend to handle User Queries for easy Cloud deployment.

## Repository Structure

├── AIBot.py # Agent logic

├── AIBot\_UI.py # Frontend interface

├── ingest.py # Document embedding pipeline

├── backend.py # FastAPI backend

├── test\_agent.py # Unit tests

├── Dockerfile # Container setup

├── requirements.txt # Python dependencies

├── docs/design.md # This document

└── README.md # Project overview

## Tech Stack

|  |  |  |
| --- | --- | --- |
| Layer | Technology | Purpose |
| Frontend (UI) | Streamlit | Simple chat interface for user queries |
| Backend API | FastAPI | Handles RAG + API + LLM orchestration |
| Vector Database | FAISS | Store & retrieve document embeddings (unstructured data) |
| Embeddings | HuggingFace Sentence Transformers (all-MiniLM-L6-v2) | Convert text into semantic vectors |
| LLM | TinyLLM via Ollama | Local open-source reasoning model for synthesis and decision support |
| Structured Data API | USGS Water Quality API (or EPA Facility Registry / CDC Tracking) | Provides real-time, factual environmental data |
| Data Loader | LangChain WebBaseLoader / PyMuPDF | Fetch public reports or documents for ingestion |
| Environment Management | Python 3.10+, venv | Runtime environment |