Humanli.ai – AI/ML Engineer Assignment

Role

AI/ML Engineer

Assignment

Website-Based Chatbot Using Embeddings

Duration

2–3 Days

Company

Humanli.ai

Overview

At Humanli.ai, we build intelligent systems that transform how humans interact with information. This assignment is designed to evaluate your ability to design and implement a **real-world AI-powered retrieval and question-answering system** using modern LLMs, embeddings, and vector databases.

Assignment Objective

Build an AI-powered chatbot that:

- Accepts a **website URL** as input
- Crawls and extracts content from the website
- Creates embeddings from the extracted content
- Allows users to ask questions **strictly related to the website**

- Returns **accurate, context-aware answers** based **only on the website content**

Core Requirements

1. URL Input

- User enters a **valid website URL**
- The system should handle:
    - Invalid or unreachable URLs
    - Empty or unsupported content gracefully

---

2. Website Crawling & Content Extraction

- Extract meaningful textual content from the provided URL

Must:

- Remove irrelevant sections such as:
    - Headers
    - Footers
    - Navigation menus
    - Advertisements
- Avoid duplicate content

Mandatory:

- HTML pages

---

3. Text Processing & Chunking

- Clean and normalize extracted text
- Split content into **semantic chunks**
- Chunk size and overlap should be **configurable**
- Maintain metadata for each chunk:
    - Source URL
    - Page title (if available)

---

4. Embeddings & Vector Storage

- Generate embeddings from text chunks using **any embedding model**
    - Examples: OpenAI, HuggingFace, SentenceTransformers
- Store embeddings in a **vector database**
- Vector database choice should be **based on project requirements**

Examples include (not limited to):

- Qdrant
- FAISS
- ChromaDB
- Pinecone
- Weaviate
- Embeddings must be **persisted and reusable**, not recreated on every query

---

5. AI Frameworks & LLM Usage

- You may use AI orchestration frameworks, including but not limited to:

    - LangChain
    - LangGraph
- You may use **any LLM model**:
    - Open-source or proprietary
- The chosen LLM model must be:
    - Clearly mentioned
    - Justified in the README

Prompt design must ensure:

- Answers are generated **only from retrieved website content**
- No hallucinated or external knowledge responses

---

6. Question Answering Logic

- Users can ask natural language questions
- If the answer is **not available on the website**, the chatbot must respond exactly with:

“The answer is not available on the provided website.”

---

7. Short-Term Memory (Conversation Context)

- Implement **short-term conversational memory**
- Memory should:
    - Maintain context across multiple user queries
    - Be limited to the **current session only**

---

8. User Interface (Mandatory)

- Build a **Streamlit-based application**

The UI must allow:

- Entering a website URL
- Indexing the website
- Asking questions via a chat interface
- Viewing chatbot responses clearly
- Provide a **public Streamlit application link**
    - If public deployment is not possible, provide clear local run instructions

---

Deliverables (Mandatory)

1. GitHub Repository

- Complete source code
- Clean and modular project structure
- No hardcoded secrets

2. README.md

Must include:

- Project overview
- Architecture explanation
- Frameworks used (LangChain / LangGraph, if any)
- **Which LLM model you used and why**
- **Which vector database you used and why**

- Embedding strategy
- Setup and run instructions
- Assumptions, limitations, and future improvements

---

3. Streamlit Application Link

- Publicly accessible Streamlit app
  OR
- Clearly documented local execution steps

Important Notes

- Answers must be **strictly grounded in website content**
- Hardcoded answers are not allowed
- Plagiarized code without explanation is not permitted
- Code quality, clarity, and reasoning are as important as functionality

---