**POC Submission Document: InfoChatAgent**

**1. Project Overview**

InfoChatAgent is a Proof of Concept (POC) for a web scraping and retrieval augmented generation (RAG) agent. It scrapes public web pages or static HTML files, creates a local vector index of the content, and provides a question-answering interface via a Streamlit UI with citations and inline passages.

**2. Objectives**

* Demonstrate end-to-end data scraping and processing.
* Convert raw HTML to clean text, chunk it, embed, and index locally.
* Enable querying with grounded answers and source citations.
* Provide a clear roadmap from MVP to production readiness.

**3. Key Features**

* Static scraping using requests, readability-lxml, and BeautifulSoup.
* Optional mini-crawler to follow top N Stack Overflow question links.
* Text cleaning and chunking with configurable sizes and overlaps.
* Embedding generation via sentence-transformers (default: all-MiniLM-L6-v2).
* Persistent FAISS vector index with metadata.
* Retriever with Maximal Marginal Relevance (MMR) for diversity.
* Optional use of OpenAI for fluent, cited answer generation.
* Streamlit-based UI including summary bullets, insights, and source passages.

**4. Architecture**



The architecture consists of a user interfacing with a Streamlit UI to ask queries. The UI triggers scraping and extraction of clean text from web pages, followed by chunking and embedding creation. Embeddings are stored in a FAISS vector index. Queries retrieve a diverse set of relevant chunks, which are used for extractive or generative answers depending on configuration.

**5.** **Directory Structure**

For reference, the main project structure is:

.  
├── app.py # Streamlit UI  
├── cli.py # Command-line interface for operations  
├── requirements.txt # Dependencies  
├── README.md  
├── .env.sample # Environment variable sample  
├── data/ # JSONL docstore  
├── indexes/ # FAISS indices + metadata  
├── src/  
│ └── infochat\_agent/  
│ ├── \_\_init\_\_.py  
│ ├── config.py # Configuration management  
│ ├── scrape.py # Scraping logic + mini-crawler + caching  
│ ├── processing.py # Chunking utilities  
│ ├── embeddings.py # Embedding wrappers  
│ ├── index.py # FAISS index building and querying  
│ └── rag.py # RAG pipeline with MMR and optional OpenAI generation  
└── docs/  
 └── POC.md # This document

**6. Setup and Execution**

* Create and activate a Python virtual environment.
* Install dependencies via pip install -r requirements.txt.
* (Optional) Configure .env with OPENAI\_API\_KEY for generative answers.
* Run the Streamlit UI using streamlit run app.py.
* In the sidebar, users can enable crawling, set limits, scrape content, build the index, and ask questions.

**7. Demo Scenario**

* Use default Stack Overflow Python questions tag URL.
* Toggle following top question links with a limit (e.g., 10–20).
* Scrape and build the FAISS index.
* Query common Python issues and observe the summary, insights, and source passages displayed.

**8. Roadmap and Enhancements**

* Short term: enhance reranking precision with cross-encoders, support sitemaps and domain allowlists, add UI polish features like caching and export.
* Medium term: multimodal support with OCR and image embeddings.
* Production readiness: robust logging, Docker/CI-CD, authentication, rate limiting.

**9. Limitations**

* Dynamic content scraping is not included (consider later use of Playwright).
* No enterprise authentication or long-term governance in this POC.

**10. Risks and Mitigations**

* Potentially sparse content on dynamic pages mitigated by link crawling and future dynamic scraping.
* Hallucination risks mitigated by grounded prompts and citation display.
* Rate limiting and robots.txt respected with caching and throttling.

**11. Submission Bundle**

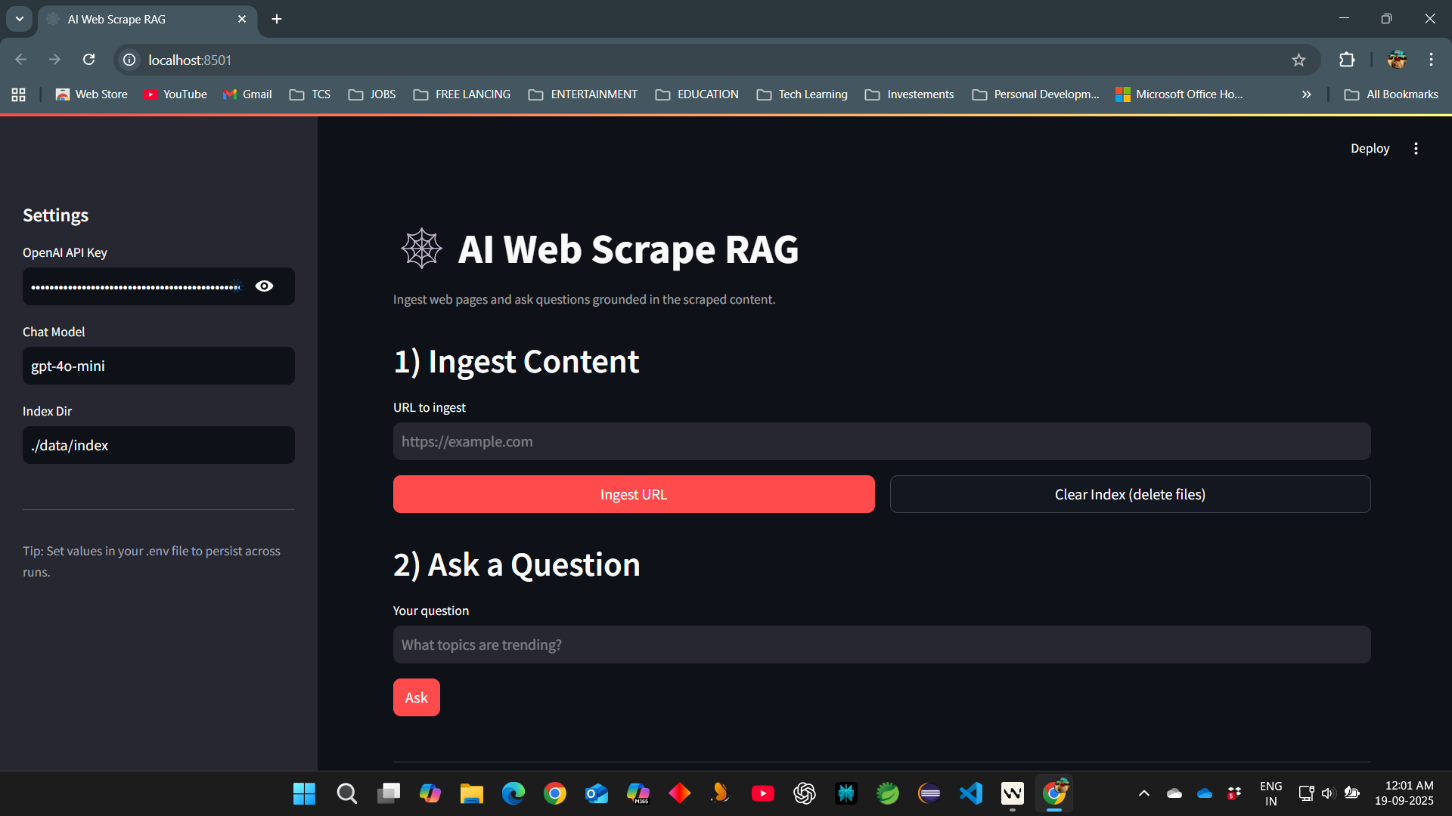
The submission includes this document and the source tree with the following key items:

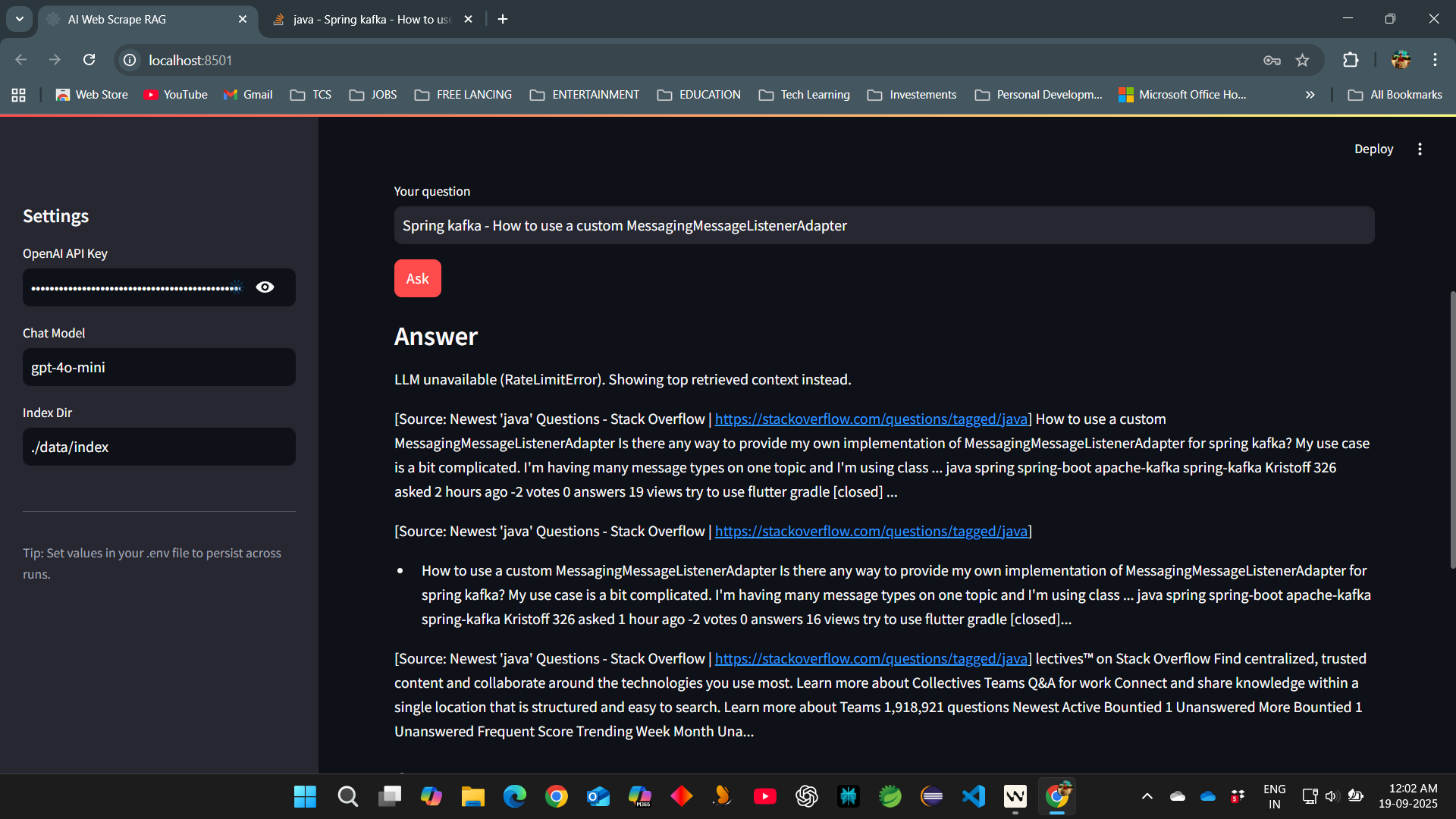
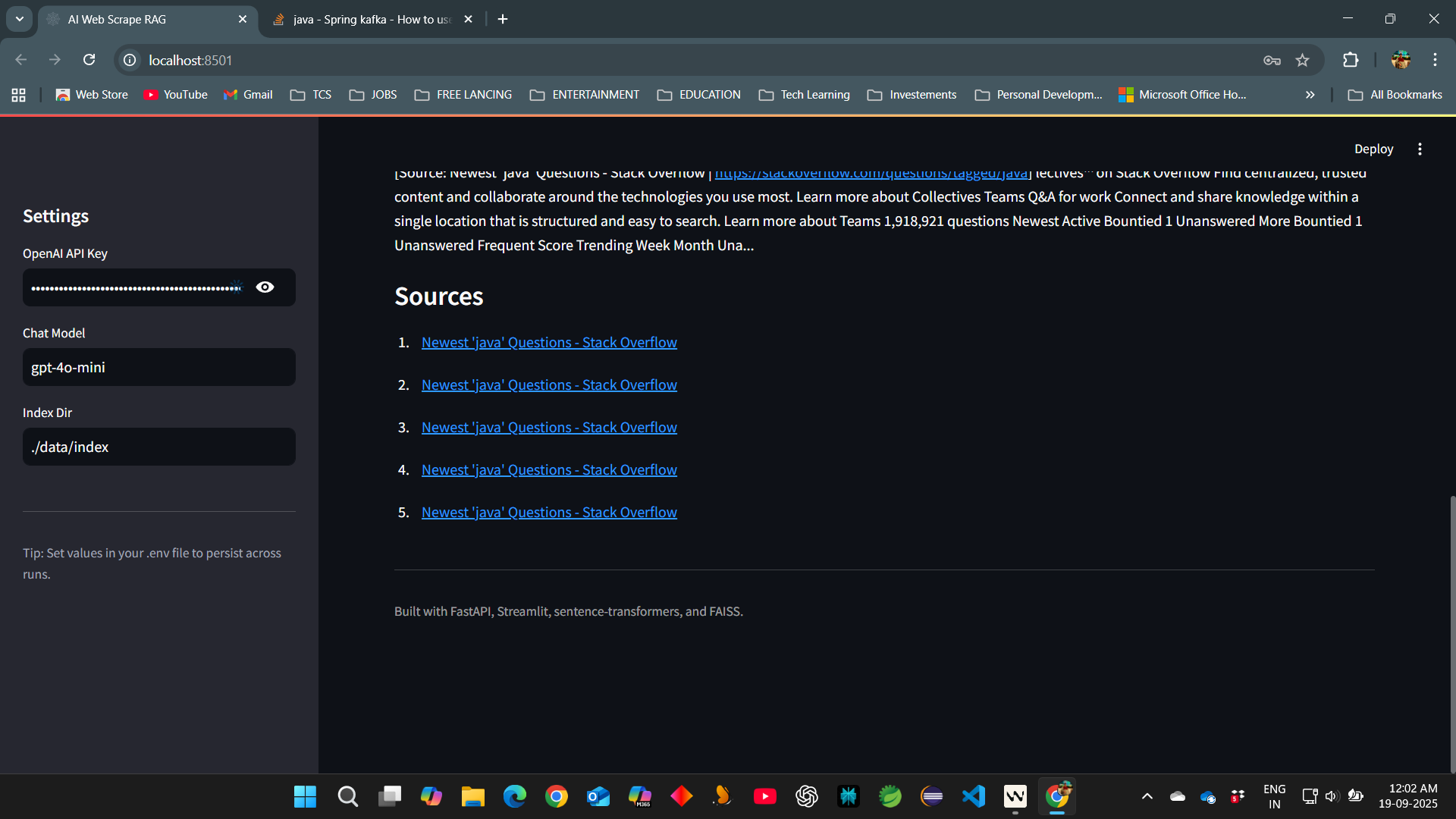
* src/ folder
* app.py
* cli.py
* requirements.txt
* .env.sample
* README.md
* docs/POC.md

A suggested packaging command (Linux/macOS):

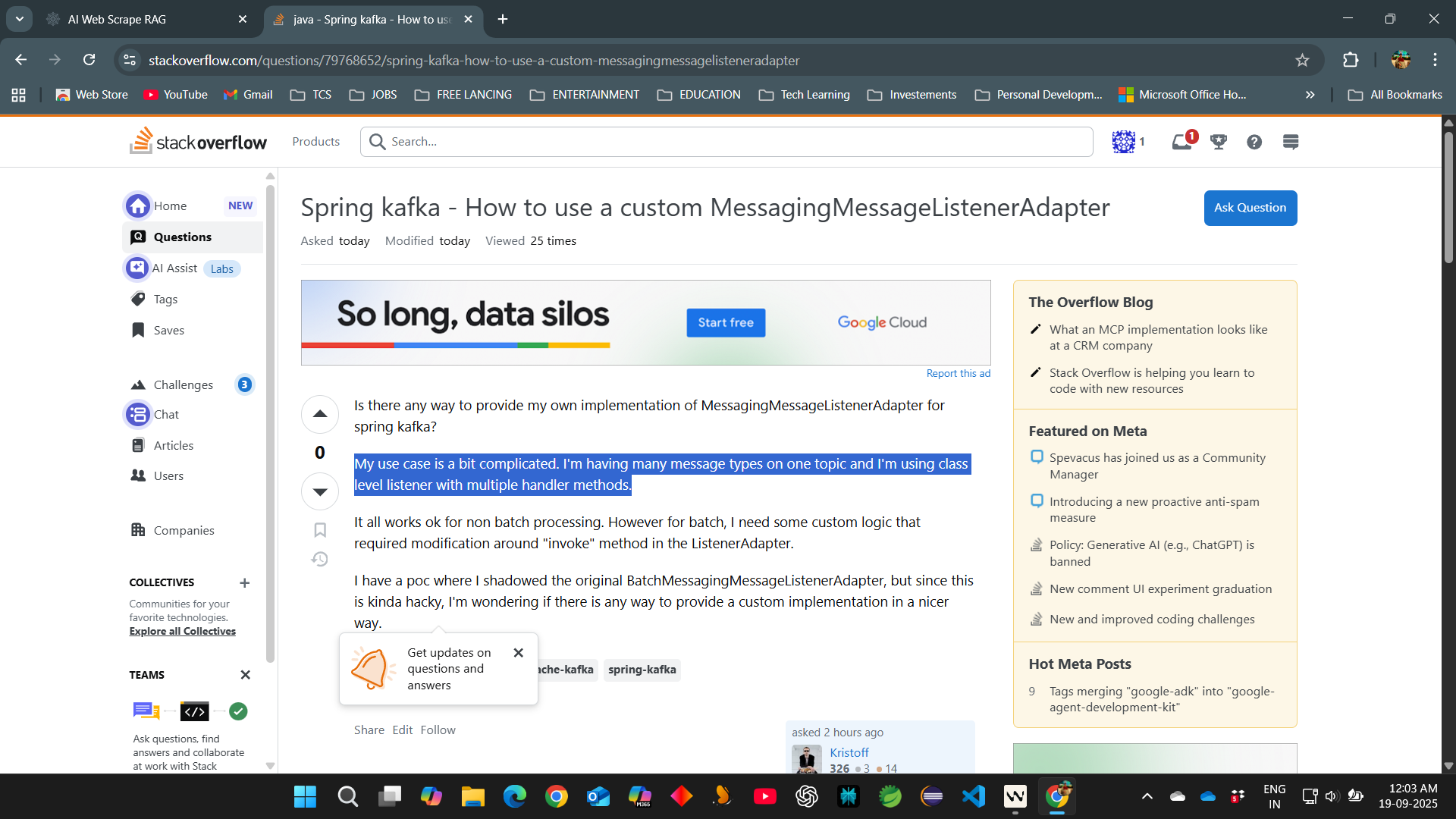
zip -r infochatagent\_poc.zip src app.py cli.py requirements.txt .env.sample README.md docs

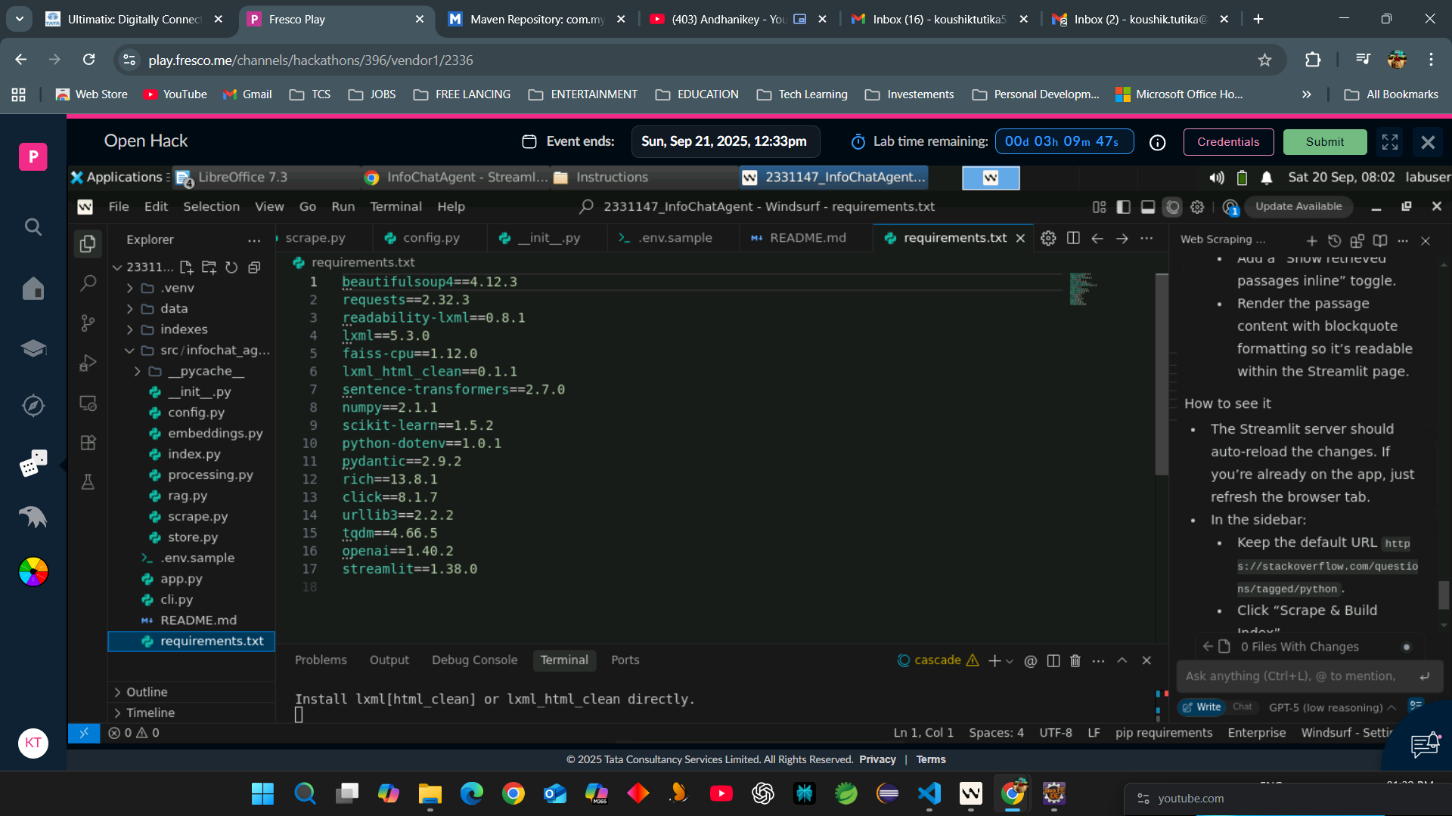
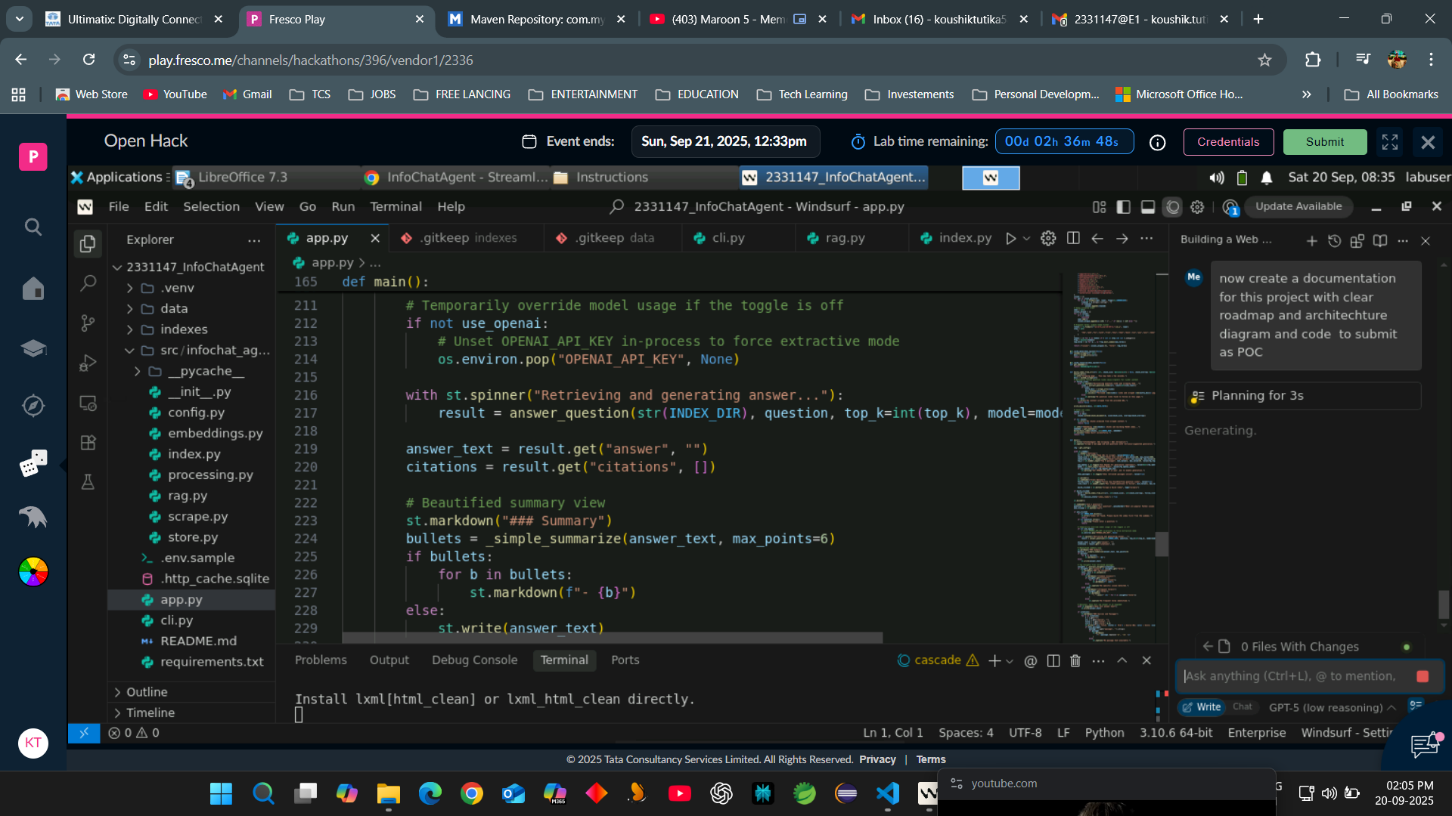
**Key Screen Shots**

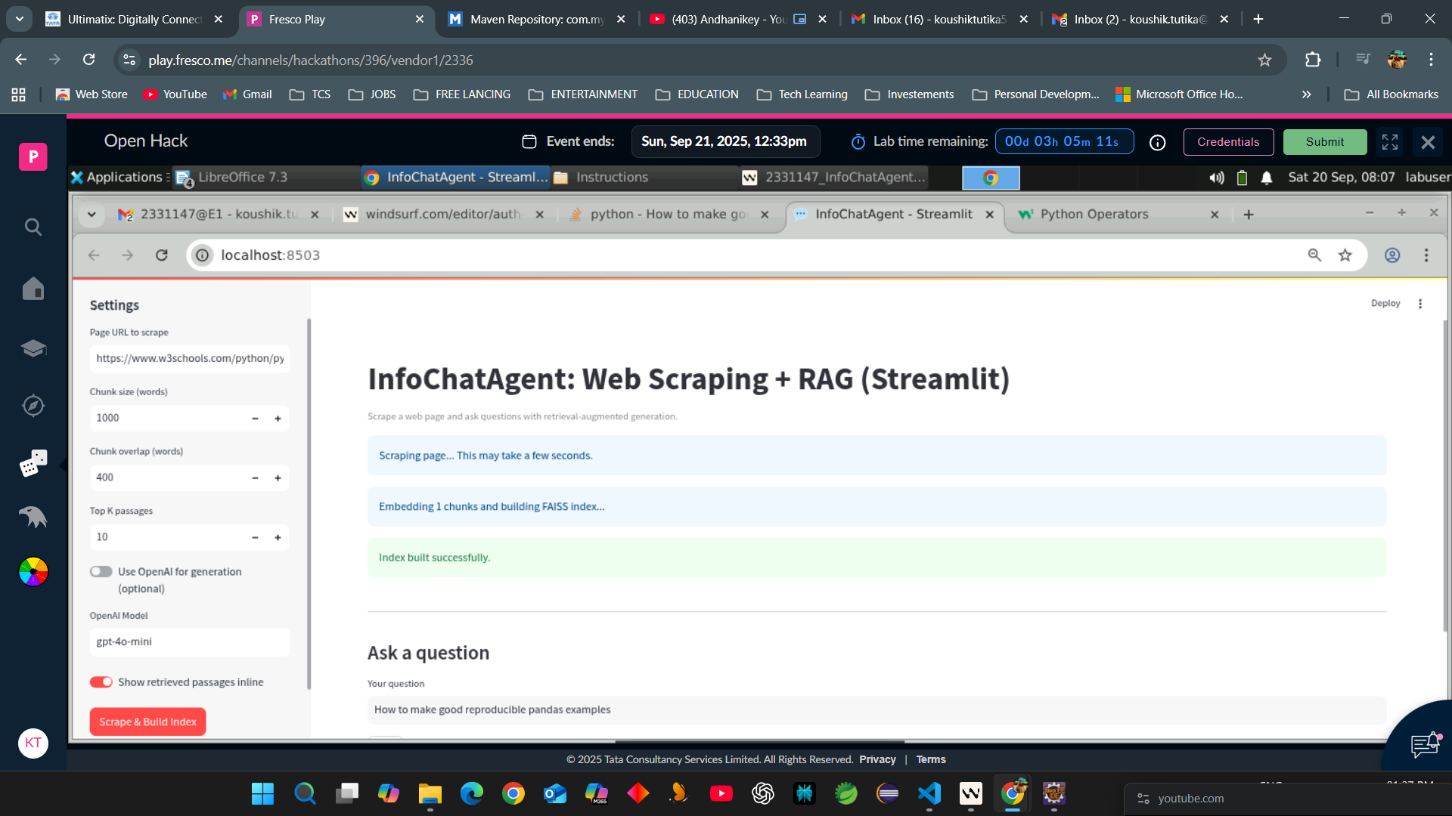
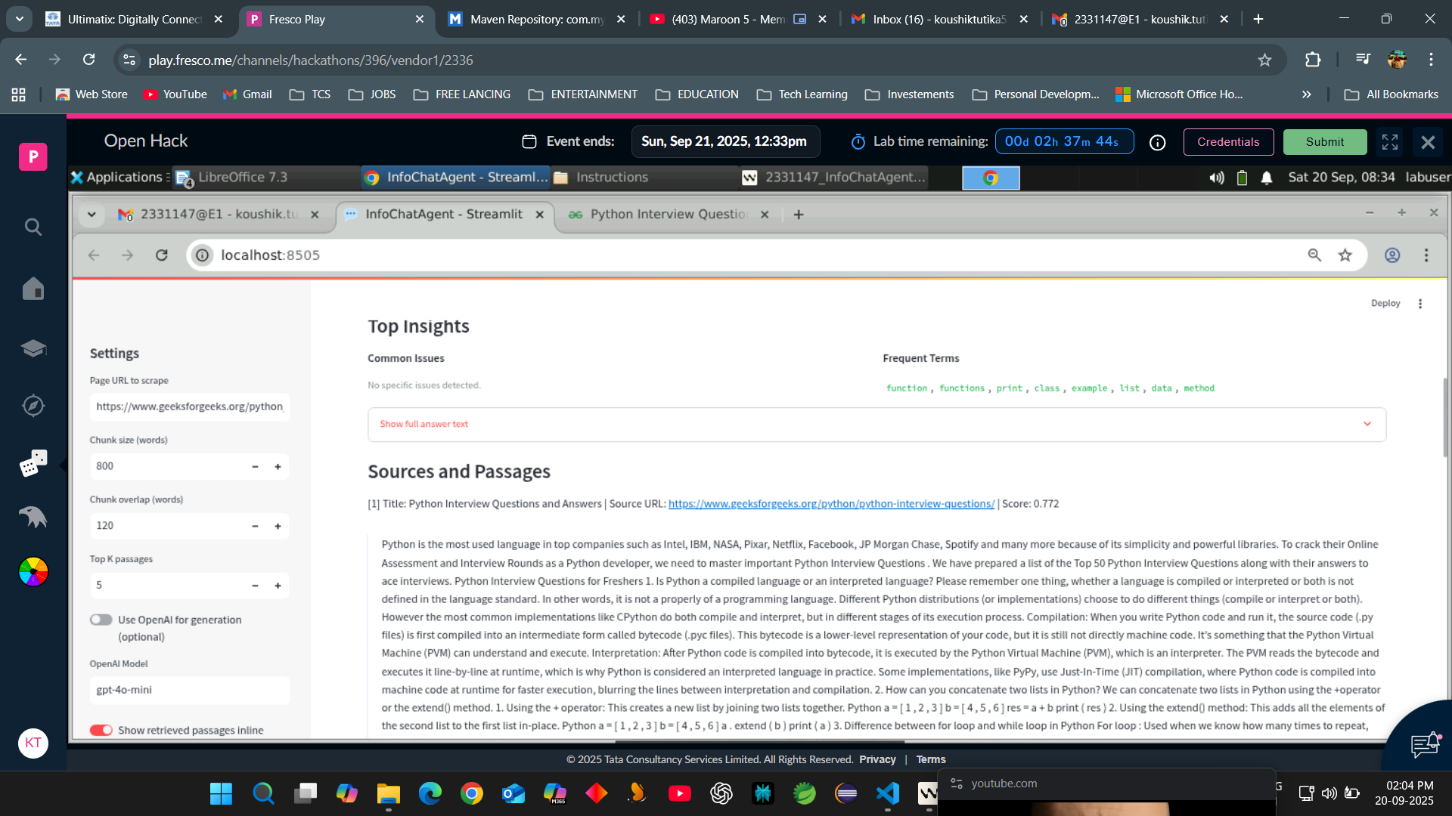
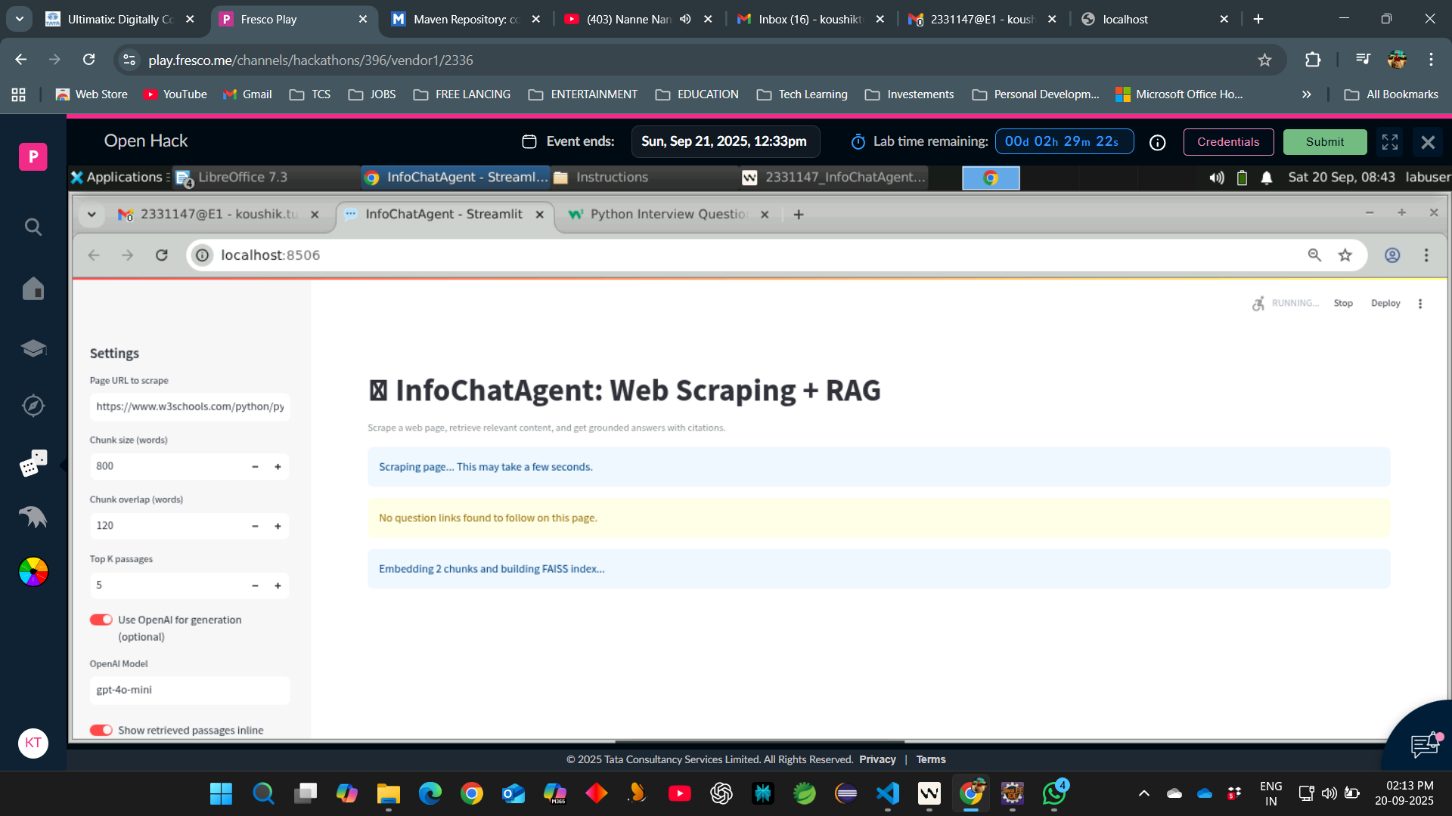
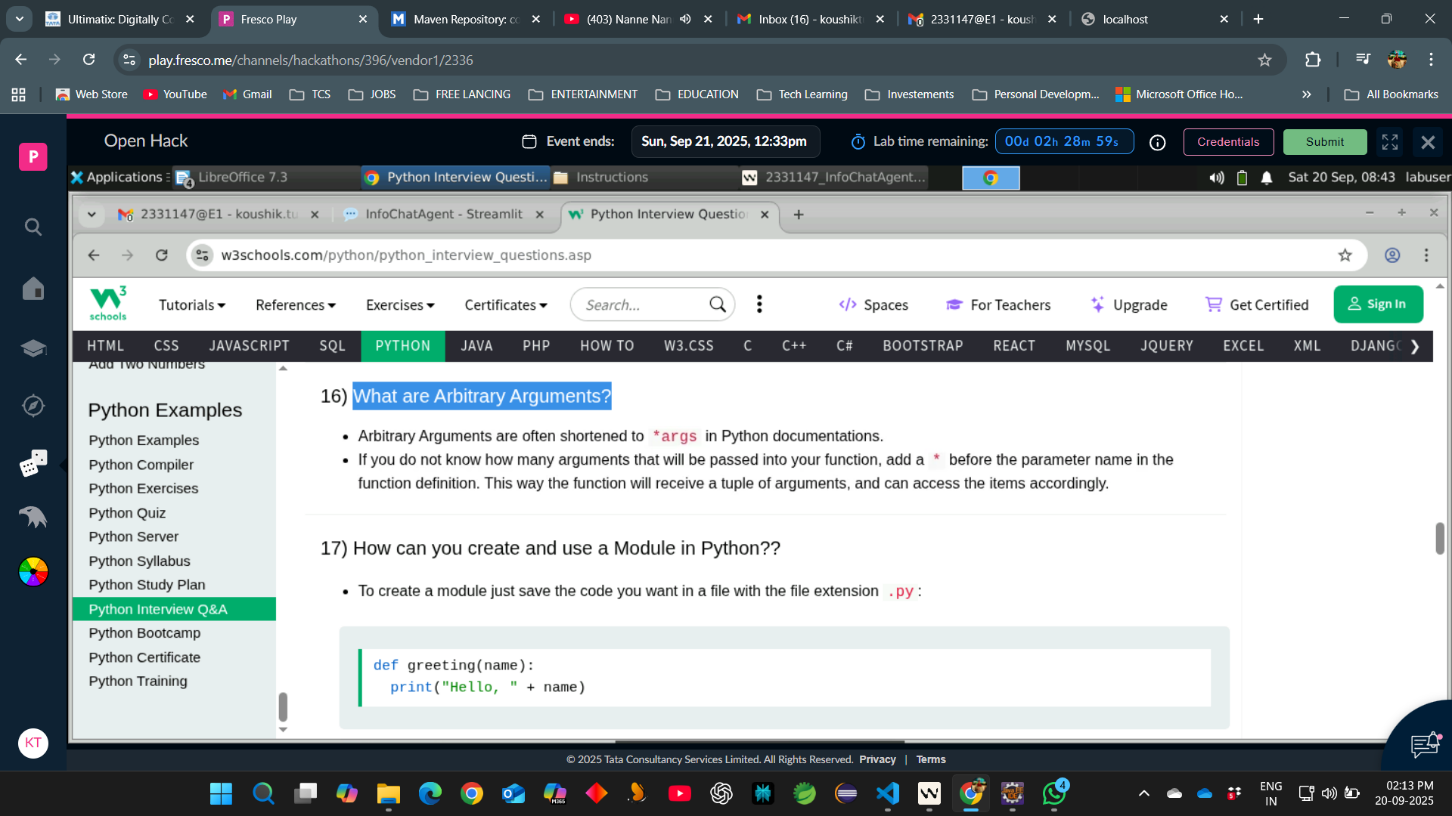
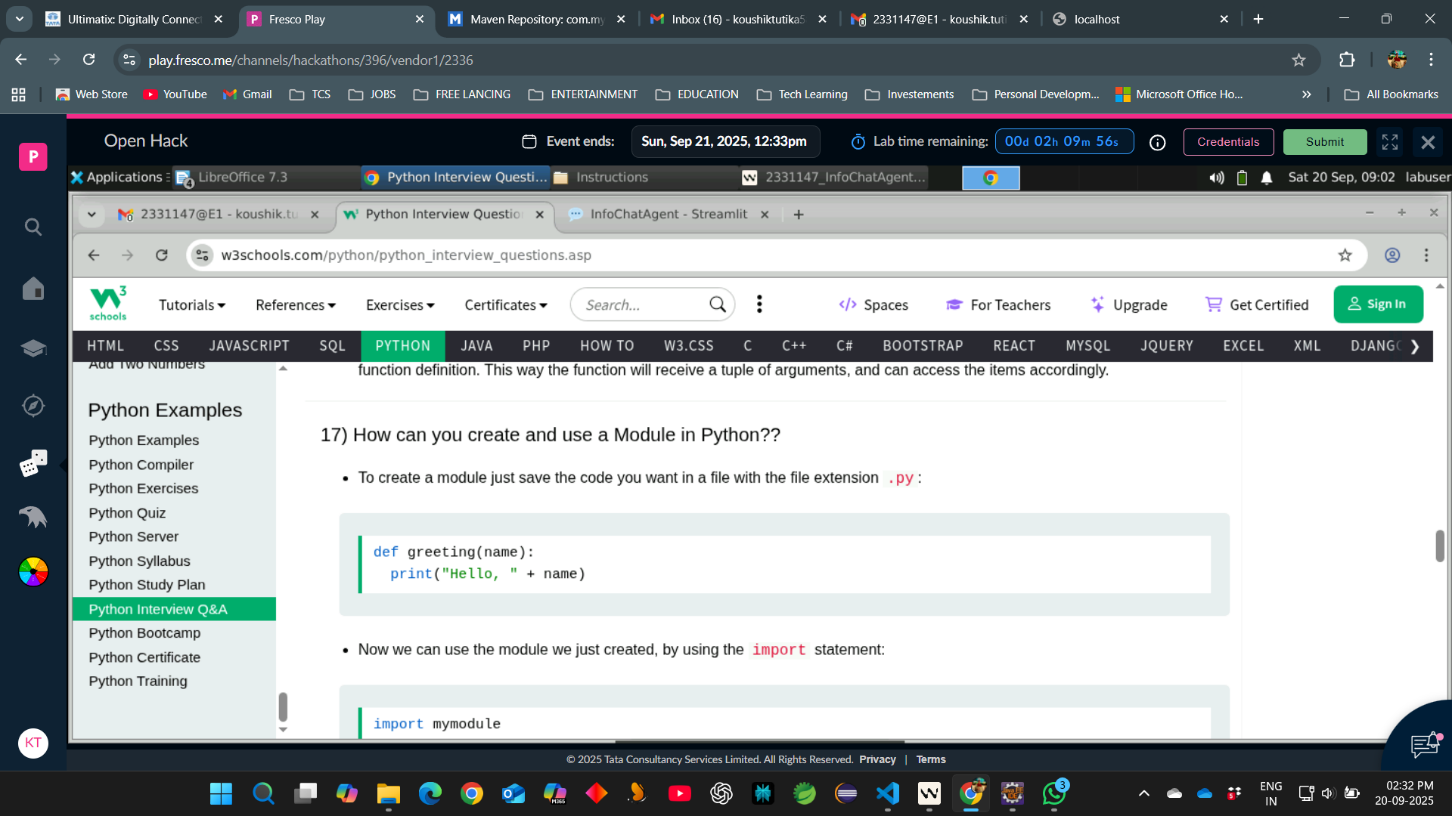
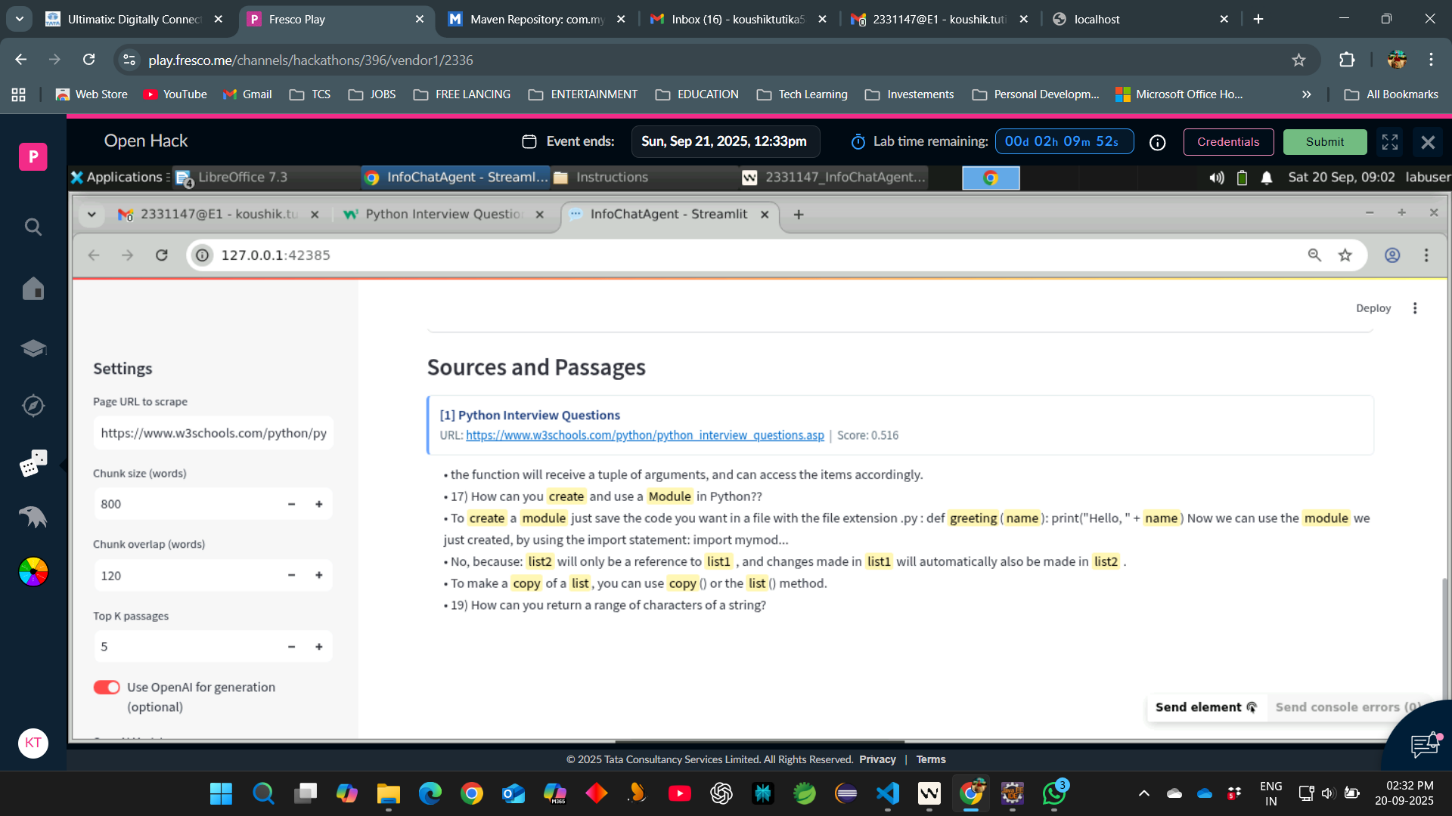
****

****

**Answer generated from reference url stackoverflow**

****

****

****

Responses generated by Agent reference taken from W3 schools