AI-CHATBOT SIMPLIFYING STUDY ABROAD PROCESSES IN NEPAL

INTERNSHITP PROJECT AT Verisk

Objective

- Provides personalized guidance to students about universities, courses, and scholarships, with aid in document prepration

- Reduces dependency on expensive consultancy services by offering free or affordable AI-driven solutions.

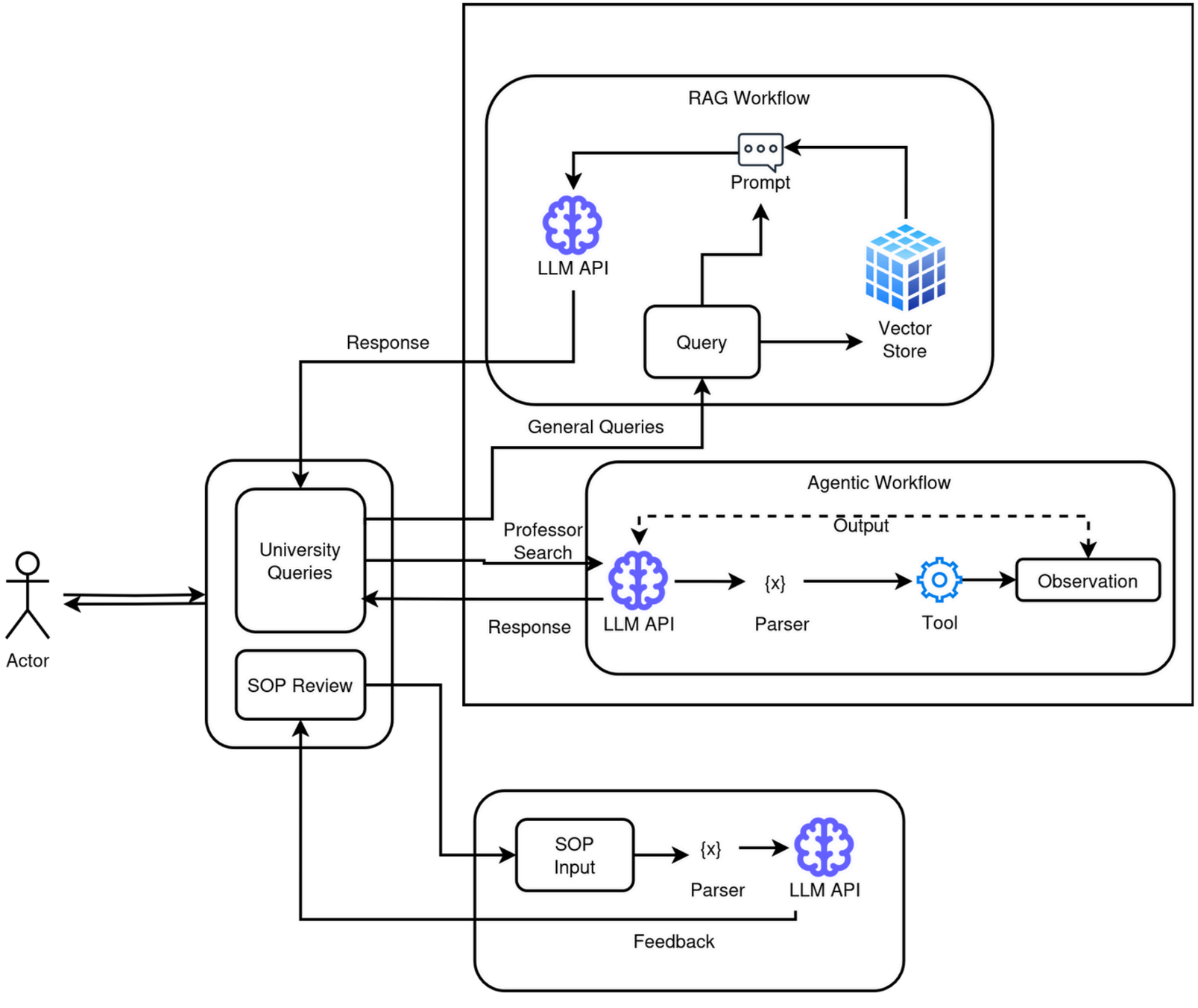
Features

- Chat based interface for searching and querying university their courses and scholarship schemes.
- Agentic Professor Search feature.
- Document preparation tool (SOP review).

Feasibility Study

- Focusing on a single country primarily for Graduate Studies
- Data collection through ranking websites, Kaggle datasets, university websites
- Open source LLMs such as Deepseek, frameworks such as Langchain, agentic frameworks such as CrewAI, SmolaAgents
- Retrieval Augmented Generation approach for updated and relevant information retrieval

Proposed Methodology



Functional Requirements - [1]

1. User Authentication & Profile Management

- The system should allow users to sign up and log in.
- Users should be able to edit and update their profile (name, email, academic interests).
- The system should store user preferences (e.g., favorite universities, saved professor profiles, past SOP reviews).
- Users should be able to track chat history for personalized recommendations.

Functional Requirements - [2]

2. Chatbot Interface

- The chatbot should have a user-friendly chat interface where users can interact with three key modules:
    - General Queries (University Search) – RAG-Based
    - Professor Search – Agentic Workflow
    - SOP Review – LLM-Based
- The chatbot should support both text input for interactions.
- The chatbot should allow follow-up questions within the same session.

Functional Requirements - [3]

3 University Search (RAG-Based)

- The user should be able to enter queries related to universities, such as: "Best universities in California for AI research", "Universities with low tuition fees for international students"
- The system should use similarity search in a vector database to retrieve relevant university information.
- The retrieved results should be passed to the LLM, which will generate a well-structured response.
- Users should be able to compare multiple universities side by side.
- Users should be able to bookmark/save universities for later reference.

Functional Requirements - [3]

4 Professor Search (Agentic Workflow)

- Users should be able to search for professors based on criteria like: Research Interests, University Affiliation, Number of Publications, etc
- The system should retrieve professor profiles from a curated set of tools available to the agent.
- The agentic workflow should:
- Retrieve professor profiles based on relevance.
- Summarize each professor’s research focus and publications.
- The chatbot should provide links to the professor’s Google Scholar or ORCID profile.
- Users should be able to bookmark/save professors details

Functional Requirements - [4]

5. SOP Review (LLM-Based)

- Users should be able to upload or paste their Statement of Purpose (SOP) for review.
- The LLM should analyze the SOP for: Grammar & Style,Content Structure, Clarity & Coherence, Strength of Research Interests
- The system should provide feedback and suggest improvements.
- The chatbot should provide examples of well-written SOPs for reference.

Functional Requirements - [5]

6. Chat History & User Data Management

- The system should store chat history for each user for personalized recommendations.
- Users should be able to delete or download their past chats.
- The chatbot should maintain session continuity, allowing users to resume previous interactions.
- The chatbot should provide a REST API for external integrations.

Non - Functional Requirements - [1]

1. Performance

- The chatbot should respond within 2 to 5 seconds for most queries.
- The system should support at least 100 concurrent users without performance degradation.

2. Scalability

- The system should be designed modularly to support future expansions.
- The vector database should handle large embeddings efficiently.

3. Availability & Reliability

4. Security

- User credentials should be stored using hashed passwords (bcrypt or Argon2).

5. User Experience (UX)

- The UI should be responsive and mobile-friendly.

Implementation Details

- Data Collection : Scraping using Scrapy/BeautifulSoup, Existing Datasets
- Data Preprocessing : Python, Numpy, Pandas

- RAG : Open source LLM (Local or using Groq) and Embeddings, Vector DB(Pinecone, Chroma) , Langchian
- Agentic Workflow : CrewAI / SmolaAgents, LLM (Local or using Groq)
- SOP Review : Using LLMs

- Frontend : HTML, CSS, Javascript
- Backend : FastAPI or Flask
- Database : Pinecone or Milvus (for vectordb), Cassandra (for users sessions)
- Local LLM using Ollama

Enhancements

- Voice based input for chat features
- Interactive map to show university locations.
- Plagiarism check for SOPs
- Multi Language support

THANK YOU