Statement of Work for Project

CS-CrimsonChat

Harvard AC215

Artem Dinh, Sukanya Krishna, Riley Li, Matthew Retchin

Contact Information
tien_long.dinh@tufts.edu (Artem) and mretchin@g.harvard.edu (Matthew)

Problem Statement

To develop an application that can answer common CS student questions about academic and extracurricular activities and requirements within Harvard University through a friendly chatbot interface.

Minimum Components for a Good Project

- Large Data: Many Web pages under Harvard CS domain, courses, event lists.
- Scalability: Ability to handle multiple users chatting with the chatbot simultaneously.
- Complex Models: Use of RAG+LLM architecture with vector database.
- Computationally Expensive Inference: Selection of efficient architecture and embedding parameters for cost-effective model execution and storage usage.

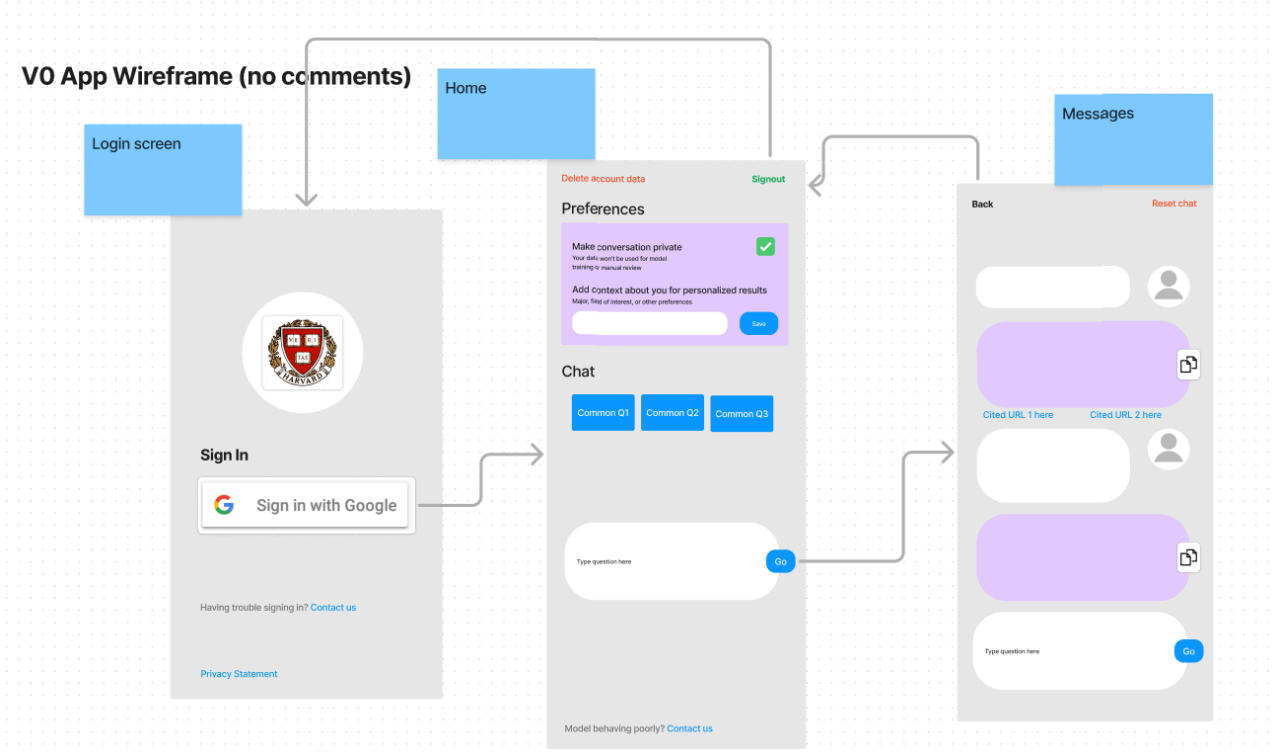
Objectives

1. A pipeline for continuous public data collection and processing from Harvard University Computer Science department website.
2. Data storage in a cloud environment with an efficient vectorization mechanism.
3. Implementation of RAG and LLM backend hosted on the cloud and calls third-party APIs.
4. Scalable backend to service many users at the same time with resource usage monitoring and limit measures.
5. User-friendly frontend design with authentication.
6. Chatbot interface for users to questions with rate limiting.

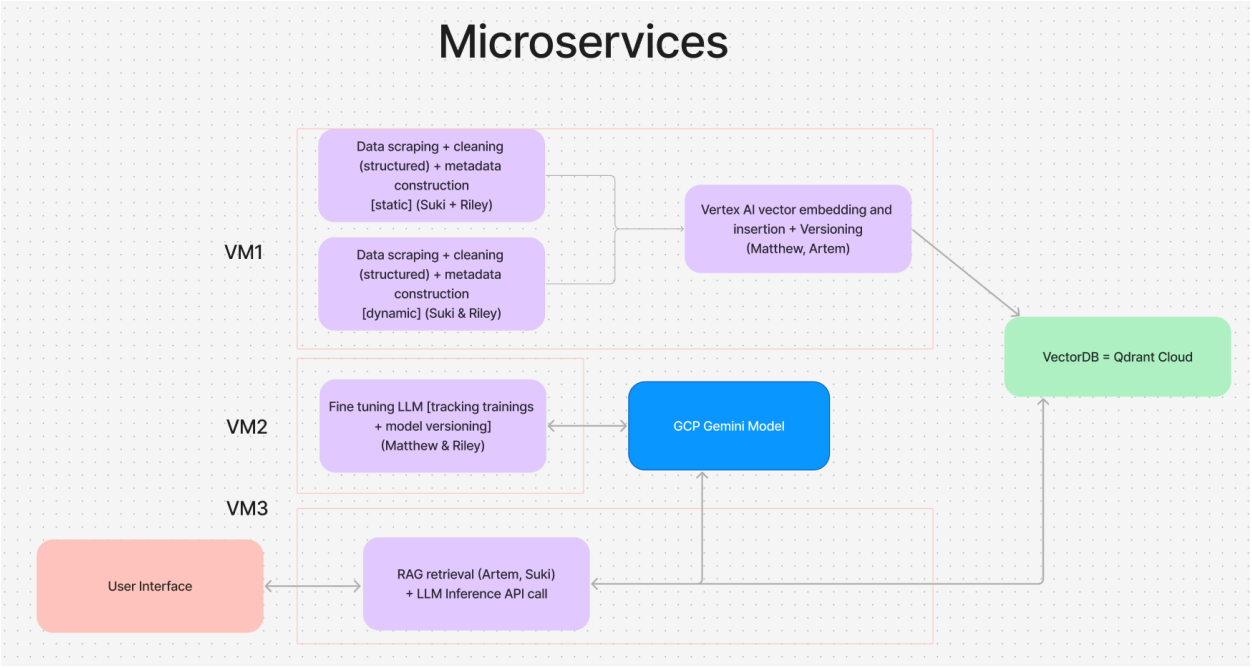
Learning Emphasis

The project will focus on constructing a scalable RAG and LLM engine with an efficient data pipeline, utilizing the course's concepts of LLM, cloud storage, containers, APIs, frontend design, and scaling.

Application UI Mockup Design



UI <-> Microservice Backend Dataflow



Resources

- Collect and preprocess a dataset of Harvard University-wide and CS department information
    - **Data Source**: The dataset will be sourced from pages on Harvard's public website. These include pages and calendars on events at Harvard, the course catalog, CS degree requirements pages, logistics pages (deadlines), and CS faculty description pages.
    - **Data Description**: Most of the data is unstructured text scraped from public Harvard web pages (cleaned so that it is just text and not containing HTML tags), with some structured data associated with each page or item.
    - **Data Attributes**: Each page will be associated with a URL, date of web scrape/API access, and its unstructured textual content, possibly with additional structured information such as where an event is happening (for event pages) or what semester a course takes place (for a course catalog page).
    - **Data Relevance**: This is a dataset comprehensive enough to answer most questions a CS student would have about their degree program.
    - **Data Quality**: Since Harvard doesn't want to mislead its students, we expect this data to be mostly of very high quality and without logical contradictions. We expect to need to merge all event data into a calendar that serves as a central source of truth, but this event data will likely exist in calendars for which we can use APIs. Luckily, all course data exists in a central source of truth already on Harvard's website, so we just need to download it and adopt a database schema matching the existing structure of the website. Please see the Limitations and Risks section for additional thoughts on how data quality could limit the application.
- Model
    - We will finetune an open model like Llama 3 (or whatever is within our budget, guided by the TAs) on our data to enable us to have customized embeddings of unstructured text.
    - The vector embeddings stored in a vector database will be retrieved during Retrieval Augmented Generation (RAG), after which we will use the OpenAI 4o API for ChatGPT to answer user queries given the retrieved context.
- Vector Database
    - After the extraction process, the data is processed and structured for storage in a vector database. We will use either Weaviate or Qdrant, both of which are optimized for handling vectorized data. This database structure is particularly effective for semantic retrieval, which is essential for the chatbot to provide accurate and contextually relevant responses.

Research and Development

We will refer to recent academic publications on RAG optimization to select cost-effective and scalable embedding vector dimensions, RAG's search algorithms, and prompt fine-tuning to prevent abuse and toxic speech. And we refer to practical knowledge from blog posts made by industry leaders (Google, LangChain, OpenAI, etc.).

Fun Factor

As we learn how to use a real-world dataset for semantic interactive search involving both structured and unstructured data, a principal goal of our project is to focus on the events that take place at Harvard. These events will be both curricular and extracurricular. Thus, the interactive search engine will expand awareness across the student population to possible fun events on campus where they can meet potential friends with similar academic and avocational interests.

Limitations and Risks

One major limitation comes from data quality. Event data for instance can come from many different sources, and contain inconsistent, incomplete, and outdated information so it would be difficult to centralize this information onto a calendar. Unstructured data from different sources can also be difficult to process for query responses, so preprocessing strategies need to be utilized in order to extract relevant information and incorporate it into the AI models. Likely, there would be an API that would sync and retrieve regular updates cross-referencing all of these sources, but it would be important to monitor these pipelines to ensure that they are working properly as website structures can change on a regular basis and we would need to add to these pipelines following institutional changes.

Another problem can come from the use of RAG, LLMs, and vector embedding for performing real-time queries as this can be very computationally expensive and lead to both high storage costs and high processing costs. It would be important to make sure to optimize the embeddings and data processing pipelines with performance tradeoff.

There is a risk that students will be tempted to take the generated text from the chat as gospel just because the chat is custom designed to provide a URL to support its answer. We will provide a disclaimer to reduce our liability and ensure students do not make mistakes.

Since we are creating an app or user interface, it would be important to ensure that the design is optimal for accommodating different user requirements and preferences. We could try to perform a user study in order to get iterative feedback and make design decisions accordingly.

Milestones

Data exploration, collection, and preprocessing: 10/18
RAG+LLM architecture design and implementation: 10/18
RAG +LLM prompt fine-tuning and CLI & Presentation: 10/31
Frontend Development & Full-Stack : 11/15
Deployment, Scaling, and Testing: 12/11

References

Li et al, A Survey on Retrieval-Augmented Text Generation, 2022.
https://arxiv.org/abs/2202.01110

Qdrant Docs. qdrant.tech, 2024.
https://qdrant.tech/documentation

Weaviate Docs. weaviate.io, 2024.
https://weaviate.io/developers/weaviate