CS 410 Project Final Status Report

1. **What is the final update on the project tasks?**

For the course project, our team has been able to complete all the pending tasks mentioned in our progress report. (See updated table at the end of report). This includes the integration between the frontend and backend as well as several bug fixes required. Another major task that was completed was setting up the automated deployment pipeline of the backend to Google Cloud Run via Github Workflow Actions. We were also able to package the frontend browser extension and get it published in the Chrome Web Store. We have written all the documentation for installing and using the extension and included it on our Github repo as well. For the project demo, we recorded a powerpoint presentation that showcases the different features of our application and have posted a link to it in our projects README file.

We originally had one additional task to generate a dataset of questions from the lecture transcripts and evaluate the systems' retrieval augmentation generation (RAG) performance using a set of metrics. However, as was mentioned in the progress report, the team member (Ehsan Sarfaraz) who was supposed to work on this task, decided to leave the group in the middle of the project. Thus we did not complete this task as we did not have enough resources to work on it. This was communicated to the TAs and the Professor informed us that we would not lose any points because of this.

Overall, all the planned tasks from the project proposal have been done besides this and we feel that the project has been successfully completed given that we have lost one member.

2. **What are lessons learned ?**

One of the lessons learned from this project was the importance of evaluation and benchmarking for information retrieval systems. We realized that we can build the best RAG pipeline in the world for our chatbot, but without the proper evaluation system and reliable metrics in place, fine-tuning the performance is very difficult and becomes more time-consuming trial and error.

Another lesson learned was the value of keeping a simplified architecture and system design. Although we chose to keep a monolithic architecture, we kept scalability in mind and opted to structure the backend code into several different API services. While this made development easier, it made the integration more difficult. We also designed the backend to be fully cloud native and ended up using multiple cloud services and providers, which ended up making the deployment process a bit more complicated. These both led to another lesson learned, which is the importance of giving attention to the integration & deployment time when doing the project planning. In the end things worked out fine for us, but had we done it again, we would likely keep a simpler design for the backend.

3. What are the future plans for the project?

We strongly believe that we have built a useful and usable piece of software using the concepts learned from this course. We believe that the future direction of this project can be expanded to include a completed RAG benchmarking and evaluation pipeline. This would be automated so that the performance of the system could be evaluated in real-time based on user queries and the uploaded course content. We also would like to have full support for other text formats including Powerpoint and Word documents. Another useful feature that could be added is a voice-chat feature for the chatbot interface. This would use a Text-to-speech API to respond to the user's question. We could also use the browsers builtin Speech-to-text API to allow users to dictate their questions, thus allowing for more natural conversations with our AI powered chatbot.

For improving the retrieval of relevant documents related to the user's question, we would also like to explore query-rewriting and expansion using ChatGPT. For example, we could take the users' question and generate an initial response from ChatGPT without providing any context in the prompt. We could then take this response along with the original question and use that to find the relevant documents. This could help in a situation where the user's question is not similar to embedded documents, but the answer to their question is.

For further improving the system performance we would also like to try adding semantic caching to the question & answering backend service. Currently, the caching mechanism works only when the exact same question is asked by another user. By using semantic caching, we would not need an exact match, allowing for a more effective use of the cache when similar queries are seen.

LINK TO PROJECT PRESENTATION:
https://github.com/abaldeo/CS410_CourseProject/blob/main/docs/PRESENTATION.md

LIST OF PROJECT TASKS:

#	Description	Staus	Estimated Effort (Hrs)	Time Spent (Hrs)	Comments	Assignee
1	Define project scope and objectives.	COMPLETED	3	3		All

2	Research and select appropriate development tools and frameworks.	COMPLETED	3	6	Frontend: React & Plasmo Backend: FastAPI & Langchain	All
3	Setting up the project infrastructure, including the development environment and project repository	COMPLETED	6	9	Using Github codespaces for development environment	Avinash Baldeo
4	Design the user interface of the Chrome extension, including file upload, summarization, and chatbot triggers.	COMPLETED	15	12		Kacper Dural
5	Implement the file upload functionality to allow users to upload lecture transcripts and slides.	COMPLETED	15	12		Colton Bailey
6	Implement text preprocessing to clean the uploaded text data (removing HTML tags, special characters, URLs).	COMPLETED	3	3		Avinash Baldeo
7	Implement text summarization using LLM	COMPLETED	15	12		Zach Pohl
8	Implementing the text embedding to generate dense vectors from the uploaded documents	COMPLETED	5	5	Embedding service is ready however for go-live we are not enabling embedding new uploaded documents.	Avinash Baldeo

9	Set up a vector database and implement indexing for documents	COMPLETED	3	3	Using Zillvus (Serverless Milvus)	Avinash Baldeo
10	Implement a Q&A chatbot interface using the Retrieval Augmentation Generation (RAG) technique.	COMPLETED	12	12		Avinash Baldeo
11	Integrate a large language model (e.g. ChatGPT or Llama2) into the chatbot for answering questions.	COMPLETED	6	6	Using OpenAI GPT3.5 model	Avinash Baldeo
12	Create a cache/database system to store responses for similar questions to improve system performance.	COMPLETED	3	3	We have implemented in-memory cache for the chatbot	Avinash Baldeo

13	Investigate and implement hybrid search combining semantic similarity and BM25 search for improved context retrieval.	COMPLETED	6	8	This feature is implemented but will be turned off for go-live as it requires more effort for fine-tuning as we don't have an evaluation benchmark setup. Similarity search is performing well for our use case	Avinash Baldeo
14	Generate a dataset of questions and relevant context to evaluate the system's performance using metrics like MRR & NDCG.	CANCELED	6	6		Ehsan Sarfaraz
15	Develop a mechanism to provide citations to the sources of course content used in chatbot responses.	COMPLETED	3	3	In some cases LLM doesn't return the sources, so citation is not possible.	Avinash Baldeo
16	Conduct testing and debugging to ensure the system works as expected.	COMPLETED	6	6		All
17	Document the project, including user guides and technical documentation.	COMPLETED	3	5		All

18	Prepare a project presentation demo.	COMPLETED	3	6		Avinash Baldeo
19	Conduct final testing, performance optimization, and quality assurance.	COMPLETED	6	3		All
20	Submit the project and prepare for presentation to the class.	COMPLETED	3	2		All
		Total (Hrs)	125	125		