Tel Hai Study Buddy

In this project, you are going to research, design and implement a chatbot that bases its answers on multiple types of media sources. Your client is Tel Hai college, and this product is intended to be used by all of the students, regardless of the degree they are studying. During the given time period, your goal is not to build an entire product (not enough time), but to prove that building this kind of product is possible by implementing a quick and dirty MVP of it, and deploy it for the students to interact with.

Product description

Core features

Multimodal Search:

- Allows users to search for lecture materials across multiple formats: documents, video recordings, and audio files.
- Provides results that are timestamped for video and audio, guiding users directly to the relevant parts of the **original input.**

Natural Language Query Support:

- Enables students to ask questions or search for content using conversational language, such as requesting specific topics, dates, or lecture details.
- Recognizes and responds to both specific and broad queries about lecture material.

For the core product, GUI is not required.

Optional features

After you're done with the core features, choose any subset of features you'd like from this list and implement them. Start with the features that you think are the most cost effective (low cost, high value for the user), **the features are not numbered by priority**. In any one of these features, before implementing the actual product, create a basic POC (proof of concept) and call the staff for review.

1. **GUI**
    - Create a basic user interface (web\app) for your product. Pay attention - in this feature in particular, make sure to search for existing tools and use them. When done with the research, call the existing staff.
2. **Personalized Recommendations**:
    - Suggests lecture materials, readings, or supplementary resources based on previous searches and user preferences.

3. **Real-Time Assistance**:
    - Interacts with users in real-time to answer follow-up questions, clarify concepts, and provide additional related material (Requires some kind of GUI/CLI).
4. **Advances filtering features**:
    - Allow the users the option to choose, during chat setup, which types of content (Video, Audio, Documents) they want the answers to be based on, allowing them to choose only a specific content type.
5. **Support multiple users:**
    - Support multiple users in your product.
    - Research authentication ways, show the staff the results of your research before starting to implement.
6. **Manual Content Bookmarking**:
    - Enables users to bookmark or save specific Audio\Video\Documents, with a specific timestamp for future reference and quick access.
7. **Automatic Content Categorization**:
    - Automatically tag your materials based on its content, when first extracting the data from them.
    - Allow users to search for specific tags.
8. **Voice Interaction Support**:
    - Allows users to perform searches and interact with the chatbot via voice commands for a hands-free experience.
9. **Recognize the lector:**
    - Think of a way to distinguish between students and lectors, and include that in the extracted text.
10. **Auto-Completion & Query Suggestions**:
    - Suggests potential queries and auto-completes as users type, helping them refine their searches and find content more efficiently. Think about a way to integrate your Google project with this one.
11. **Multi-Language Support**:
    - Supports multiple languages to cater to diverse academic needs and to ensure accessibility for all students. Which languages are important to support? Document your reasoning
12. **Multiple input formats:**
    - Support additional input formats (file types\extensions) of video, audio or text. Which formats are important to support? Find out, and document your reasoning

Research

In this exercise, you should use external tools and libraries as much as you can, in order to reduce the amount of code you have to develop and maintain yourself.
Hugging face is a good place to start your search. A research phase is a crucial part of a development cycle, when your product contains new concepts to the team. Its length varies according to the project, and the amount of concepts you have to develop from scratch.

Input parsing

Your goal for the input parsing phase is to convert all the different media types to text.
There are a lot of technologies available that could do that, most of them are AI based.

Step 1: Audio Transcription

Goal: Research technologies that convert audio into text and create a PoC in a Jupyter notebook.

- **Research Focus**: Identify multiple tools or algorithms for converting spoken language into text, considering factors like noise, language support, and performance.
- **Testing Requirement**: Each team must test at least two different transcription technologies on a variety of audio samples (e.g., different accents, background noise, varying audio quality). Analyze accuracy, speed, and handling of edge cases.
- **PoC Task**: Demonstrate the transcription of multiple audio files using the technologies researched. Compare the results (text accuracy, processing time) in your notebook and discuss strengths and weaknesses.In addition, you can choose to implement any subset of features you'd like from this list. Choose to start with the features that you think are the most cost effective (low cost, high value for the user)

Step 2: Image Recognition in Video & Audio Transcription

Goal: Research technologies for recognizing objects or scenes in video, and create a PoC that transcribes video audio.

- **Research Focus**: Explore at least two technologies for object detection in video frames and audio transcription from video.
- **Testing Requirement**: Run multiple tests on different video samples, including videos with varying resolutions, lighting conditions, and object complexities. Transcribe audio from these videos using the selected tools.
- **PoC Task**: In the Jupyter notebook, showcase the processing of several video clips, highlighting object detection and audio transcription accuracy. Include performance metrics such as processing speed, object detection accuracy, and quality of audio transcription. Compare how different tools perform on each task.

Step 3: Document Parsing

Goal: Research document parsing technologies and implement a PoC in Jupyter to extract structured data from documents (e.g., PDFs or scanned images).

- **Research Focus**: Investigate multiple methods for parsing documents, focusing on the ability to handle structured and unstructured formats, as well as noisy or low-quality scans.
- **Testing Requirement**: Test at least two parsing technologies across various document types, including clean PDFs, scanned documents with noise, and complex forms with tables. Evaluate extraction accuracy and speech

When done with any of the steps, call the course staff for review

Retrieval-Augmented Generation (RAG)

Your goal in the RAG phase is to implement a system to retrieve relevant data for a given query.

Step 1: Text Embeddings

Goal: Explore technologies that transform text into embeddings, creating a PoC to evaluate their performance on a diverse range of textual inputs.

- **Research Focus**: Identify several models that convert text into vector embeddings, considering factors such as language coverage, embedding dimensionality, and how well the embeddings capture contextual information.
- **Testing Requirement**: Experiment with at least two different embedding models on various text types (short vs long text, technical vs conversational language). Assess each model's ability to generate meaningful embeddings in terms of semantic similarity and computational efficiency.

Step 2: Vector Database

Goal: Research and evaluate tools for storing and querying embeddings, producing a PoC that enables scalable and efficient embedding searches.

- **Research Focus**: Analyze different vector databases for their performance, scalability, and ease of integration. Consider key factors such as real-time search capability, handling large datasets, and indexing speed.
- **Testing Requirement**: Test at least two vector databases, benchmarking them for query performance on both small and large datasets. Evaluate search precision, time to index new embeddings, and scalability under high query volumes.

Step 3: Generation

Goal: Explore how large language models can generate content based on embeddings retrieved from a vector database, producing a PoC to evaluate their generation quality.

- **Research Focus**: Investigate different models that leverage content from vector databases to produce context-rich, relevant responses. Assess the models based on their fluency, coherence, and ability to integrate retrieved content meaningfully into the generated output.
- **Testing Requirement**: Test two or more generation models by using them to answer questions based on retrieved text or documents. Measure their output for relevance, completeness, and contextual accuracy, particularly in challenging or ambiguous scenarios.

Deployment

Your goal in the Deployment Phase is to implement a containerized system and deploy it on a scalable cloud infrastructure, enabling robust and flexible handling of user queries with the RAG architecture.

Step 1: Containerization

Goal: Package your RAG system and file parsing system into Docker containers for consistent and efficient deployment across different environments.

Research Focus: Investigate containerization technologies, focusing on Docker for packaging the system, including all dependencies and services (embedding models, vector database, and generation models). Consider factors such as container size, build times, and resource efficiency during runtime.

Testing Requirement:

1. Create a Dockerfile that installs all necessary components: the text embedding models, vector database tools, and generation models.
2. Experiment with multi-stage builds to optimize container size.
3. Test container functionality locally to ensure that all services (embedding generation, vector database querying, and content generation) work correctly in the containerized environment

Step 2: Cloud Deployment on Google Cloud Platform (GCP)

Goal: Deploy the containerized RAG system to the Google Cloud Platform, making it available as a scalable service.

Research Focus: Explore GCP's infrastructure services for deploying containerized applications, focusing on Kubernetes Engine (GKE) or Cloud Run for scalable deployment.

Consider aspects such as auto-scaling, load balancing, resource allocation, and continuous deployment capabilities.

Testing Requirement:

1. Deploy your Docker container on GCP using either Cloud Run or Kubernetes (GKE) to handle incoming queries at scale.
2. Test the deployment with different workloads, ensuring the system remains responsive under high query volumes.
3. Benchmark the system for latency, throughput, and fault tolerance in a cloud environment, ensuring that queries are processed efficiently even under heavy loads.

Research product - a detailed, ipynb file for each of the mentioned categories, containing POC for each technology you select. Each notebook should compare different tools, performing the same tasks. You should pick tasks that your future product should perform, so you will see the advantages and disadvantages of each technology performing the actual tasks that you need from it.

The factors you should measure are -

- Results quality
    - Read about methodologies for comparison between different models, and pick an cost effective method
- Efficiency (performance profiling)
- Memory usage (memory profiling)

When done with any of the steps, call the course staff for review