Modelling

Model Training

Build Dataset

Build Question Answer Dataset

Model Selection with Boundries

Evaluation

Fine-tuning

Convertion to Ollama

Serving

Backend

Host Models Vanilla, Tuned, Embedding

Host Vector Store

APIs for Chat, Embedding, Retrieval

Document Embedding

Chat

Embed-One

Retrieval k, vector

Vector Store

Interfacing

App

User Input

Embedding

Retrieval or Agent

Chat History

Chat - UI

Feedback

Quality A. - View

Feedback Data

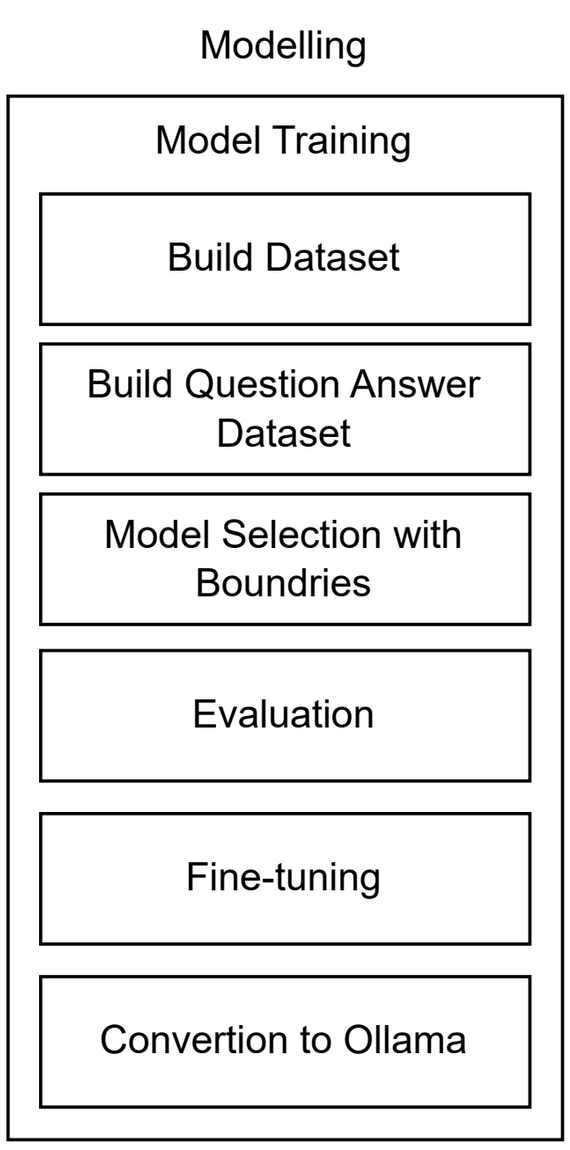
Modelling

Description:

You are part of a development team that focuses on fine-tuning open-source large language models to better serve their company's use cases. You are not working alone on the project but with two other teams that rely on your model.

An important step for the success of the project is to align with the backend team and clarify how they intend to use the model once it is ready and how you will make it accessible to them. Therefore, you need to agree on a common interface and model format.

The user interface team is tasked with collecting feedback from users based on their perception of the answer quality. To utilize this feedback to identify issues with the model, you need to coordinate with your colleagues on a common feedback format that allows you to efficiently process the feedback. You also need to determine how the feedback will be made available. This process does not need to result in a fully automated re-tuning system, but it should clearly explain how the feedback will be used.



Fine-tuning Acceptance Criteria:

- Identify a use case that benefits from a fine-tuned model.
- Collect data and build a dataset to fine-tune your model.
- Build or generate a dataset to evaluate the performance of your fine-tuned model.
- Select a model to fine-tune. Boundaries for the selection are provided by the lecturers.
- Perform the fine-tuning using dedicated cloud resources (Hyperstack).
- Evaluate the quality of your fine-tuned model based on the evaluation dataset you have built.
- Convert the model to an Ollama-compatible format.
- Make the model available to your student colleagues. Preferably HuggingFace
- Provide a theoretical explanation of how feedback could be used to identify performance declines in the model's answers.

Backend

Description:

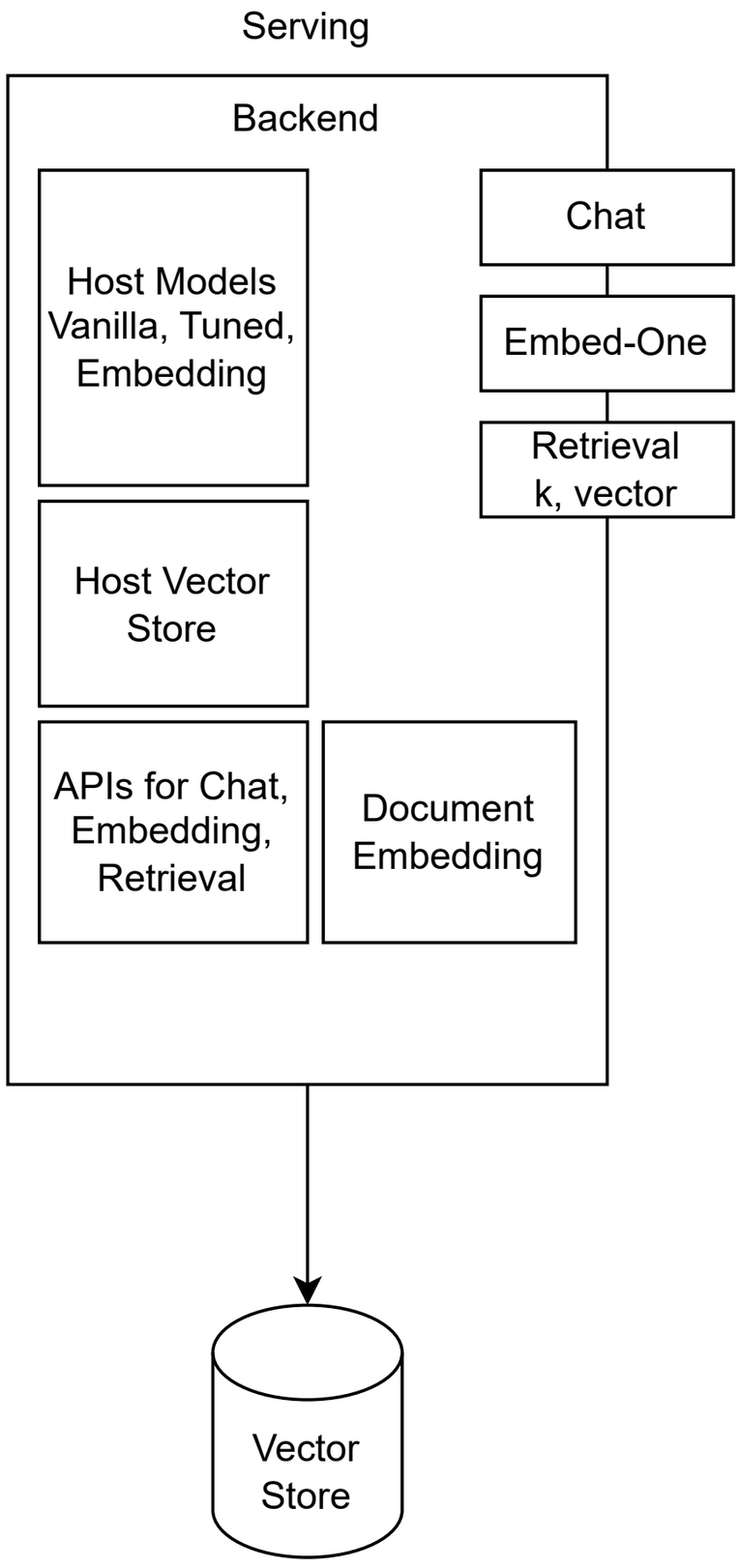
You are part of a team that builds the backbone of your large language model application. You serve as the connection between the user-facing interface and the model fine-tuning process.

You are working closely with both the fine-tuning team and the user interface team.

You need to align with the fine-tuning team on where and how you will receive the model after the tuning process is complete. Until a fine-tuned model is ready for use, you will mock a service that behaves like the interface agreed upon with the fine-tuning team. Once the model is available, you will replace the mock interface with the actual connection to the model. You should consider using the untuned model selected by the fine-tuning team in the interim.

You will also need to coordinate with the user interface team on the endpoints they should use to connect with the backend you are building. You should take into account different settings, such as using RAG (retrieval-augmented generation) or choosing between a tuned or vanilla (untuned) model. These features should be configurable by the user through the interface.

Additionally, you will handle the embedding of documents for a RAG approach. This can be done via a manual onboarding process or by providing an endpoint for this purpose.



Backend Acceptance Criteria:

- You will build an API backend that allows the UI to communicate with your backend.
- You will handle requests of different types, such as RAG or various model configurations.
- Your backend will fetch and host the models.
- You will provide endpoints to perform embedding and retrieval.
- You will host a database capable of storing the embeddings of the onboarded documents.
- You will write tests to ensure the quality of your backend.
- Come up with a theoretical concept of making the backend horizontal scalable

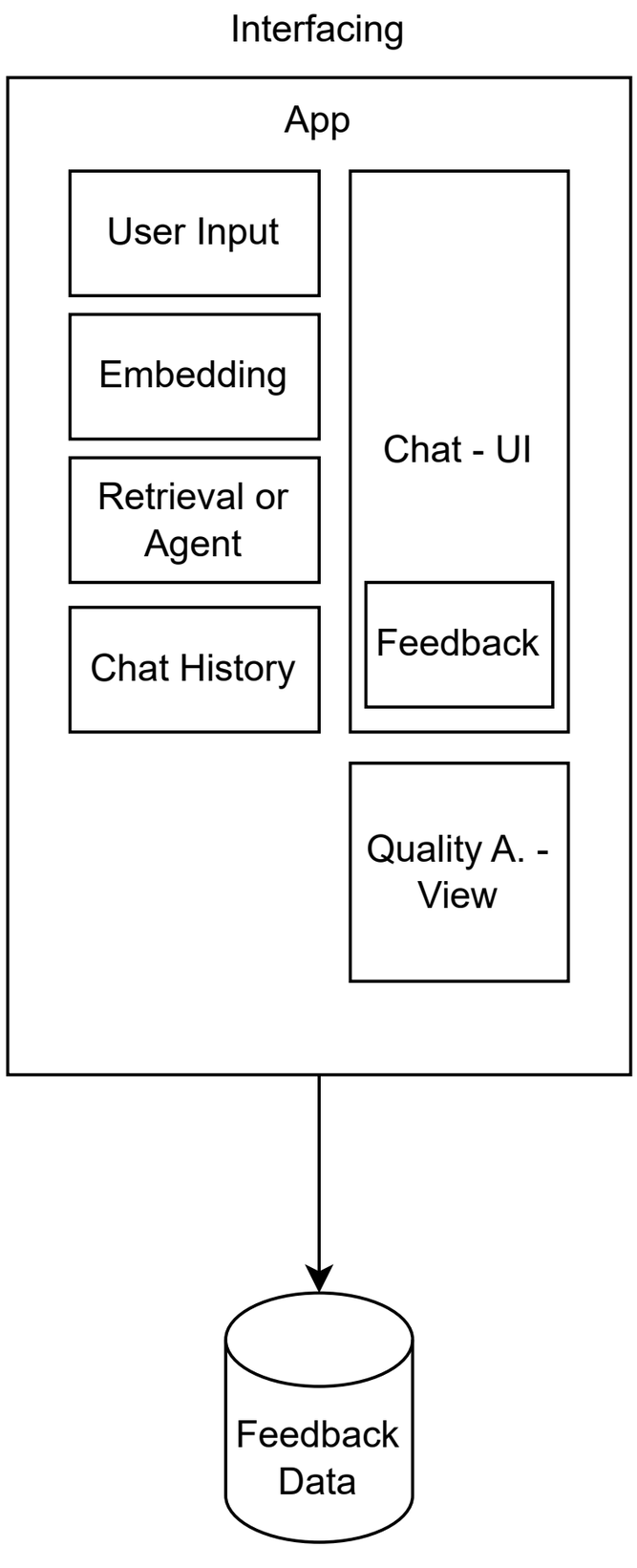
User Interface

Description:

You are part of the team developing the user interface for your chat application. You are free to choose a framework that suits your ideas.

You will closely collaborate with the backend team to align on how to consume their service. You will take the lead in generating ideas that are useful to your users and will coordinate with the backend team on how the service should be built and consumed. While no fully functional backend is available, you will use a mock interface that serves a simple chat endpoint. To remain flexible, you have decided to handle the RAG approach in two steps: first, embedding your question through a dedicated "embedding" endpoint on the backend, and then issuing a second request to a dedicated "retrieval" endpoint.

Your team will create a chat interface that provides all the functionalities discussed with the backend team. You will prepare requests to the backend, display results, and preserve a chat history. For quality assurance, you will include a feature that allows users to upvote or downvote an answer based on its perceived quality. This feedback will be valuable for the fine-tuning team, so you will align with them on the feedback format and how to provide it. Additionally, some of your managers are interested in answer quality statistics. To address this, you will provide a view that displays the collected feedback.



Interface Acceptance Criteria:

- You will build a chat interface.
- You will offer an option for model selection.
- You will provide feedback capabilities.
- You will host or create a database to store the feedback.
- You will provide a statistics view of the feedback results within your user interface.
- You will handle chat histories.

Grading

The final grading (pass or fail) will be based on a presentation. Every group member must actively participate in the presentation, contributing equally. Each presenter must be well-informed enough to answer questions about the topic they present.

Criteria:

1. **Project Workflow:**
    - Explain how your team approached the project step by step.
    - Include details about collaborations with other teams.
    - Ensure there is a clear and coherent narrative ("red thread").
    - Provide reasoning for your decisions.
2. **Code Walkthrough:**
    - Present selected workflows that form the core of your project.
    - Be prepared to answer questions about the implementation.
3. **Presentation of Final Solution:**
    - Showcase your solution by explaining its purpose and demonstrating how you addressed the problem.
4. **Extra Points:**
    - Creativity in management, coding, ideation, or other aspects of the project.