Decentralized RAG: Combining Retrieval Augmented Generation with Blockchains

Reed White (12367576)

# Project Proposal

This project aims to integrate two modern computing designs: Retrieval Augmented Generation (RAG) generative modeling and blockchains. RAG generative modeling is a system in which generative models, such as ChatGPT, use an additional post-processing “retrieval” layer that retrieves context based on similarity to the user prompt. This context, retrieved from a corpus, is then fed into the generative model along with the user prompt. This allows entities to “fine-tune” their generative models without fully fine-tuning or quantizing their backing generative models, which is an extremely time-consuming and expensive process.

The corpus used by these RAG designs are typically stored on-site in the form of a vector database. This project proposes that this vector database could be replaced with a blockchain (private or public, depending on the requirements), which would allow documents to be added to the corpus in a secure, auditable manner. A blockchain would also make the corpus decentralized and distributed, granting a higher-level of availability and portability than is provided by a centralized vector database.

# Project Plan

For the sake of this proof-of-concept project, the blockchain will be assumed to be private and permissionless (although this concept should be applicable to public or permissioned blockchains). This project will store the corpus either directly in the blockchain or by using an off-chain, decentralized storage service, depending on testing. Documents will be fed to the corpus by a JavaScript-based browser frontend, and they will be retrieved by a generative model frontend. The corpus may be stored in a single block/decentralized storage unit, or may be distributed across the blockchain, depending on testing. The generative model frontend is not currently intended to read from the blockchain on every retrieval, and will instead retrieve the corpus from the blockchain and store it locally in memory at startup, to reduce calls to the blockchain and increase performance. This project will be implemented according to the specifications below.

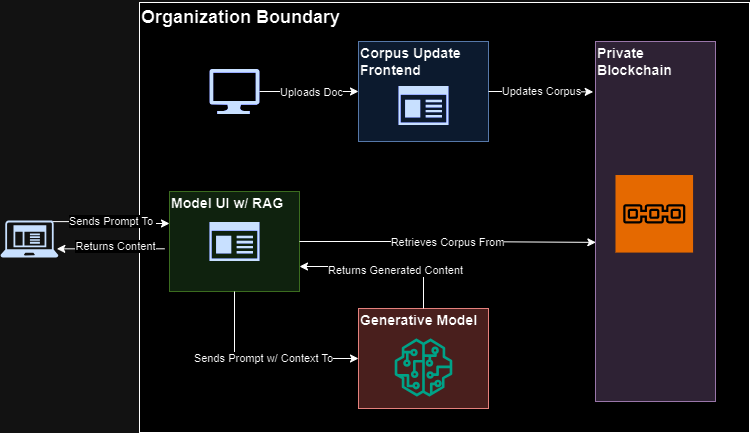
## Features

The following features will be implemented in this project:

* RAG system: this system will use Retrieval Augmented Generation to generate text content, in the form of a chatbot.
* Datastore in a Blockchain: this system will leverage a blockchain to host the datastore required to store the corpus for the RAG. The corpus will either be stored directly in the blockchain or will be mapped to off-chain storage.
* RAG UI: this system will include a webapp GUI for interacting with the RAG
* Datastore UI: this system will include a webapp for adding and modifying documents to the corpus in the blockchain

## Architecture

The following architecture is expected to be implemented in this project:

FIGURE I. Expected High-Level Architecture

With the following components:

* Model UI w/ RAG: the model interaction GUI webapp and the RAG logic will run in a single process. The generative model is also expected to be loaded and called by this process.
* Corpus Update Frontend: the corpus GUI webapp will run in a separate process, on a separate port.
* Private Blockchain: the blockchain will be loaded and tested locally.
* Users: both end users and administrative corpus users are expected to access the apps via web browsers. The latter will require MetaMask to interact with the corpus UI.

## Technology Stack and Dependencies

The following technologies, at a minimum, will be leveraged to implement this project:

### Blockchain

* Hardhat
* Solidity

### Frontend

* HTML
* Web3.js
* MetaMask

### Model

* Ollama (llama 3.1)
* LlamaIndex
* Streamlit

Additional technologies, such as IPFS, may be added during development and testing.

## Deliverables

The following will be delivered as a part of this project:

* Architectural Design
* User Manual
* System Manual
* Video Demonstration
* Testing Results (transaction times, retrieval times, etc.)
* Source Code
* Final Presentation

## Timeline

The following timeline has been developed to track the implementation of this project:

| **Task Name** | **Week 6 (Sep 23 - Sep 29)** | **Week 7 (Sep 30 - Oct 6)** | **Week 8 (Oct 7 - Oct 13)** | **Week 9 (Oct 14 - Oct 20)** | **Week 10 (Oct 21 - Oct 27)** | **Week 11 (Oct 28 - Nov 3)** | **Week 12 (Nov 4 - Nov 10)** | **Deliverables** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Research/  Planning |  |  |  |  |  |  |  | Project Proposal, Timeline |
| Contract Development |  |  |  |  |  |  |  | Solidity Contract |
| Contract Deployment |  |  |  |  |  |  |  | Contract Deployed in Local Blockchain |
| Frontend Development |  |  |  |  |  |  |  | HTML, JavaScript Files |
| Frontend Deployment |  |  |  |  |  |  |  | Application Server Deployed in localhost |
| RAG Development |  |  |  |  |  |  |  | Python Notebook/Files |
| RAG Deployment |  |  |  |  |  |  |  | RAG Deployed with GUI |
| Contract Testing |  |  |  |  |  |  |  | Contract Testing Results |
| Frontend Testing |  |  |  |  |  |  |  | Frontend Testing Results |
| RAG Testing |  |  |  |  |  |  |  | RAG Testing Results |
| Documentation Development |  |  |  |  |  |  |  | Design, User Manual, System Manual |
| Video Development |  |  |  |  |  |  |  | Video Demonstration |
| Final Report |  |  |  |  |  |  |  | Final Pitch |