Decentralized RAG: Combining Retrieval Augmented Generation with Blockchains

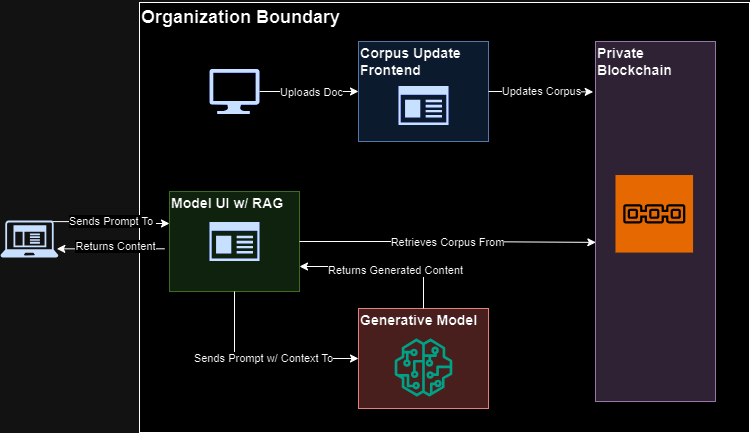
Reed White (12367576)

# Project Architecture

## On-Chain Storage

### High-Level Application Flow

The following architecture has been implemented in this project for the on-chain corpus storage version:

FIGURE I. High-Level On-Chain Architecture

With the following components:

* Corpus Administrator Frontend: the corpus GUI webapp should be loaded onto an internal server and only exposed to administrative users. This reads and writes to the Corpus smart contract.
* Private Blockchain: the blockchain should be deployed and maintained internally to the organization’s/company’s logical boundaries. It should only be exposed to systems that connect to it, such as the model or the administrative frontend. It contains the Corpus smart contract that stores the corpus.
* Model UI w/RAG: the RAG system should be loaded onto an end-user-facing server and exposed to end users. This reads from the Corpus smart contract and submits context and end-user queries to the generative model.
* Generative Model: this should be installed onto a machine capable of running a large-language model (LLM). It receives context and end user queries from the RAG, generates text, and sends the generated text as a response.
* Administrative Users: these are internal company users that maintain the corpus. They should connect to the corpus administrator frontend via browsers within the organization’s boundaries. They will require MetaMask to perform more operations.
* End Users: these are external company users/customers that interact with the model frontend. They connect to the model via browsers over the Internet or desired intranet. They have no additional requirements.

### Application Flows

Users for this system are split into two distinct categories: administrators and End users. Administrators represent internal users within the organizational boundaries of the hosting company that submit documents to, remove documents from, and maintain the corpus of their company. End users represent external users outside of the organizational boundaries that interact with the chatbot/RAG. The high-level application diagram can be broken down to represent each flow:

#### Administrators

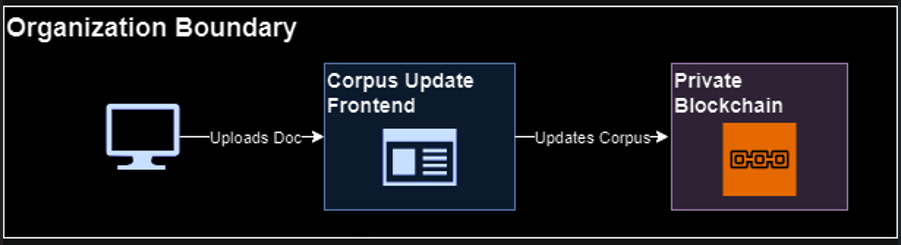


FIGURE II. On-Chain Administrator Flow

Administrator users modify the corpus stored in the blockchain via the Node.js Corpus Administrator Frontend application. MetaMask is required in order to submit or modify (write) documents via the frontend, but not to retrieve (read) documents.

#### End Users

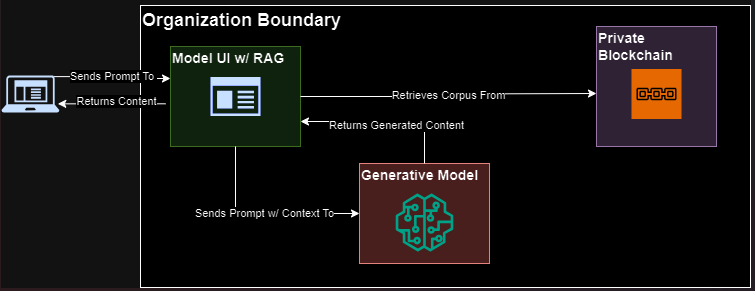


FIGURE III. On-Chain End User Flow

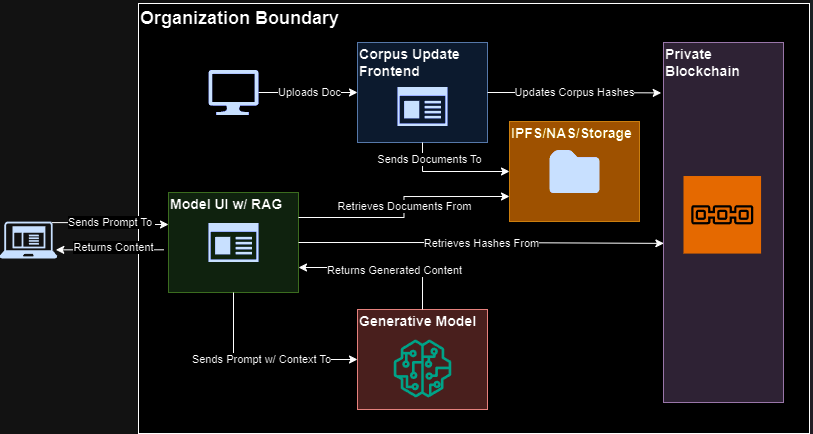
End users interact solely with the chatbot/Model Frontend. The current design does not require users to authenticate nor have MetaMask, as it is intended to be a public-facing chatbot. Because the RAG system only reads from the blockchain, it also does not have any MetaMask, wallet, or private keys.

## 

## Off-Chain Storage

### High-Level Application Flow

The following architecture has been implemented in this project for the off-chain corpus storage version:

FIGURE IV. High-Level Off-Chain Architecture

With the following components:

* Corpus Administrator Frontend: the corpus GUI webapp should be loaded onto an internal server and only exposed to administrative users. This reads and writes to the HashCorpus smart contract and the off-chain storage (tested with local storage, but intended to be a file share or NAS).
* Private Blockchain: the blockchain should be deployed and maintained internally to the organization’s/company’s logical boundaries. It should only be exposed to systems that connect to it, such as the model or the administrative frontend. It contains the HashCorpus smart contract that stores the hashes for the corpus.
* IPFS/NAS/Storage: corpus documents should be stored on highly-available storage external to the blockchain, such as NAS, an NFS share, or IPFS. This can be and was tested locally on the Corpus Administrator Frontend, but this is not recommended.
* Model UI w/RAG: the RAG system should be loaded onto an externally-facing server and exposed to end users. This retrieves hashes from the HashCorpus smart contract, documents from the shared storage, generates hashes for the latter and compares with the former, and if they match it submits the verified documents as context along with end-user queries to the generative model.
* Generative Model: this should be installed onto a machine capable of running a large-language model (LLM). It receives context and end user queries from the RAG, generates text, and sends the generated text as a response.
* Administrative Users: these are internal company users that maintain the corpus. They should connect to the corpus administrator frontend via browsers within the organization’s boundaries. They will require MetaMask to perform more operations.
* End Users: these are external company users/customers that interact with the model frontend. They connect to the model via browsers over the Internet or desired intranet. They have no additional requirements.

### 

### Application Flows

Users for this system are split into two distinct categories: administrators and End users. Administrators represent internal users within the organizational boundaries of the hosting company that submit documents to, remove documents from, and maintain the corpus of their company. End users represent external users outside of the organizational boundaries that interact with the chatbot/RAG. The high-level application diagram can be broken down to represent each flow:

#### Administrators

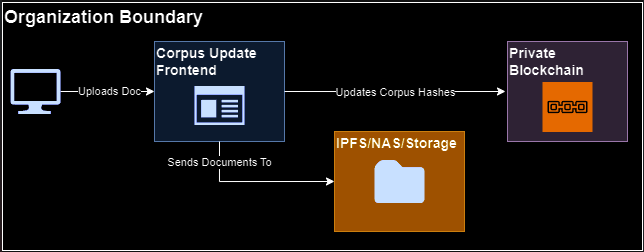


FIGURE V. Off-Chain Administrator Flow

Administrator users modify the corpus via the Node.js Corpus Administrator Frontend application. The application stores hashes in the blockchain and stores the full documents in some form of shared storage, such as a NAS or IPFS. MetaMask is required in order to submit or modify (write) documents via the frontend, but not to retrieve (read) documents.

#### End Users

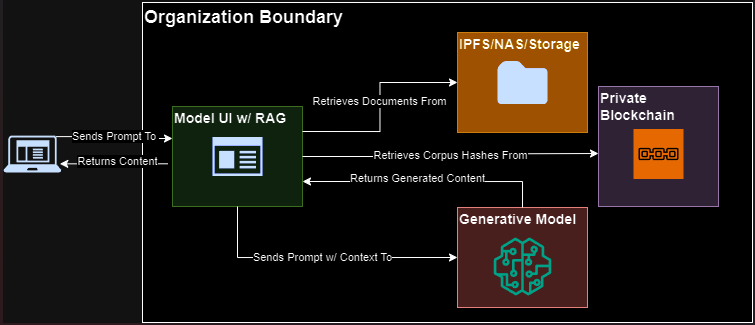


FIGURE VI. Off-Chain End User Flow

End users interact solely with the chatbot/Model Frontend. The current design does not require users to authenticate nor have MetaMask, as it is intended to be a public-facing chatbot. Because the RAG system only reads from the blockchain, it also does not have any MetaMask, wallet, or private keys. The RAG retrieves hashes from the blockchain and documents from the shared storage. Documents are not loaded into the RAG unless the locally-calculated hashes of the full documents match the hashes for the documents stored in the blockchain.