PDF QUERY TOOL

Cabdul Ciise

AGENDA

Introduction

Project Timeline

High Level Architecture

BE/FE Implementation

Persistence Layer

Demo

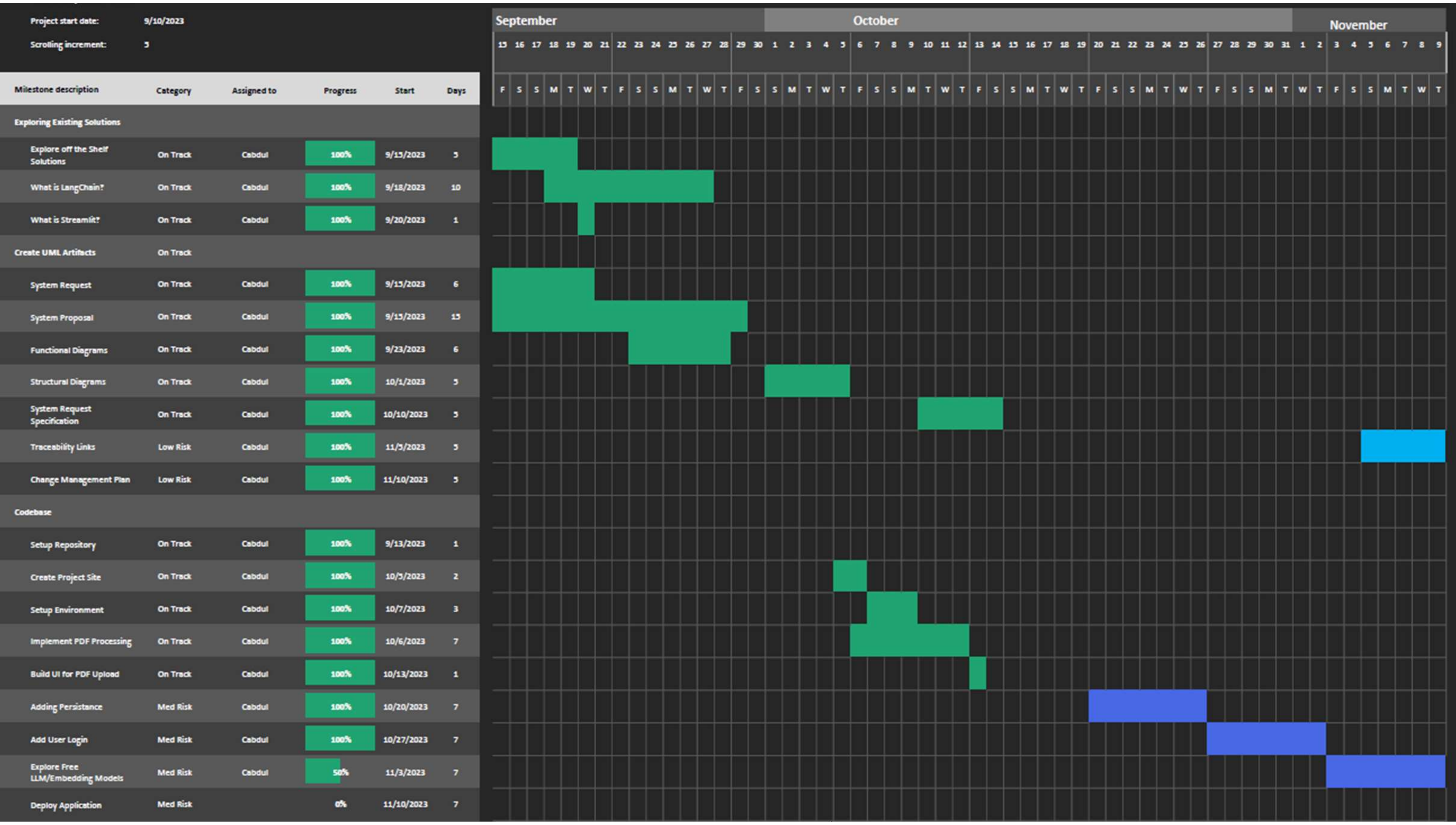
INTRODUCTION

With the growth and adoption of AI and ML around us, I was interested to take on a project in this space.

This project is focused on creating an MVP: an app that takes in a collection of PDFs and coupled with a LLM, provide a chat-like interaction for querying documents.

Essentially, ChatGPT for your PDFs.

The bigger goal after this project is to take what I've learned and wrap this question answering bot in something like an Azure function so we can provide our customers with an easy to integrate solution....unless there's something out there that does it already cheaply...

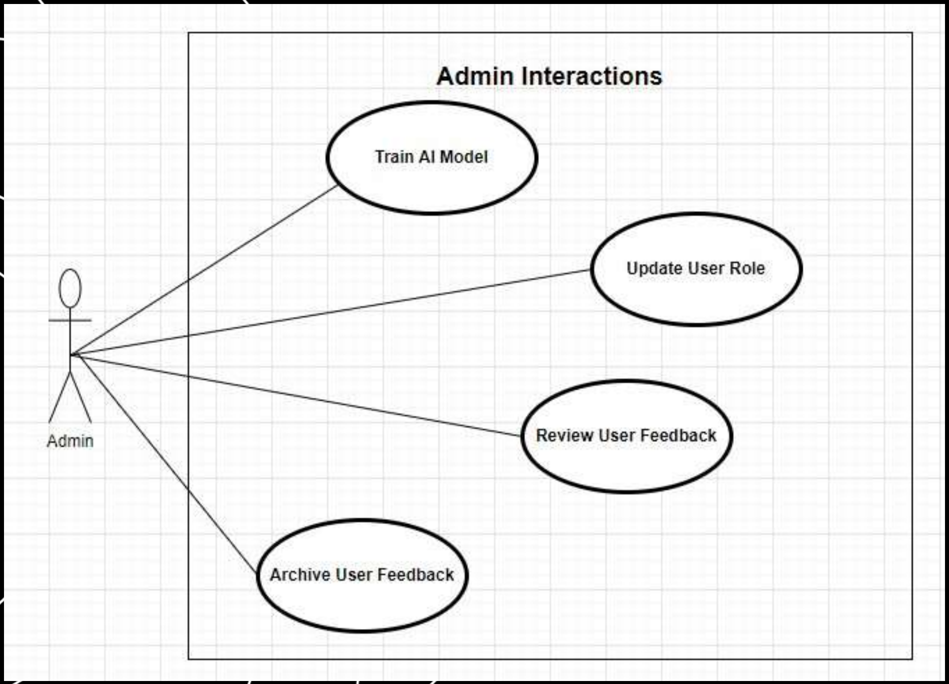


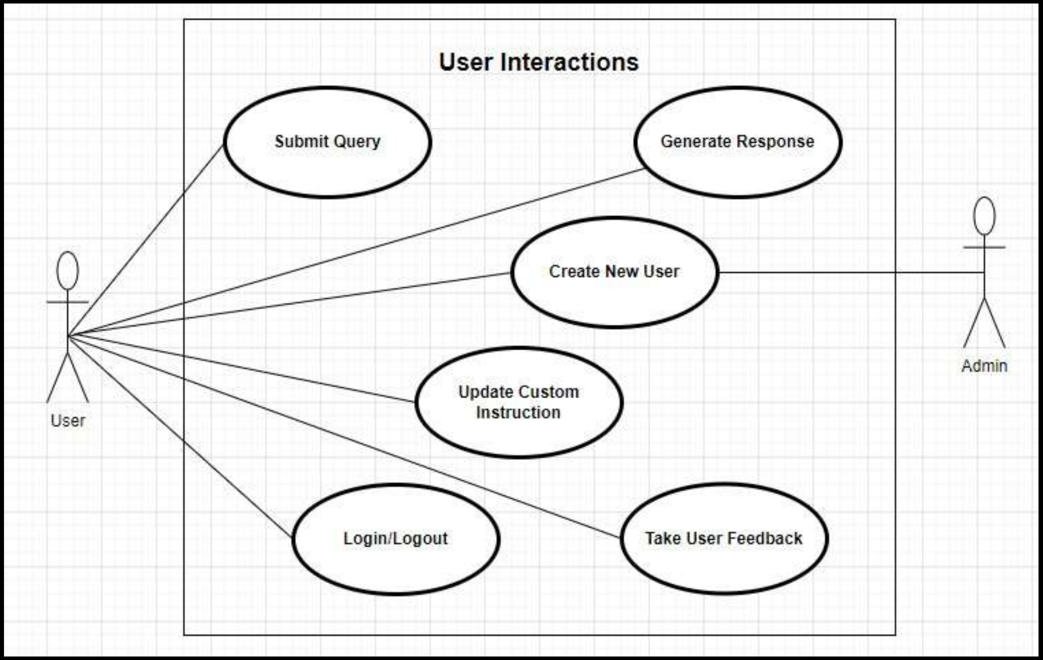
Project start date: 9/10/2023

Scrolling increment: 5

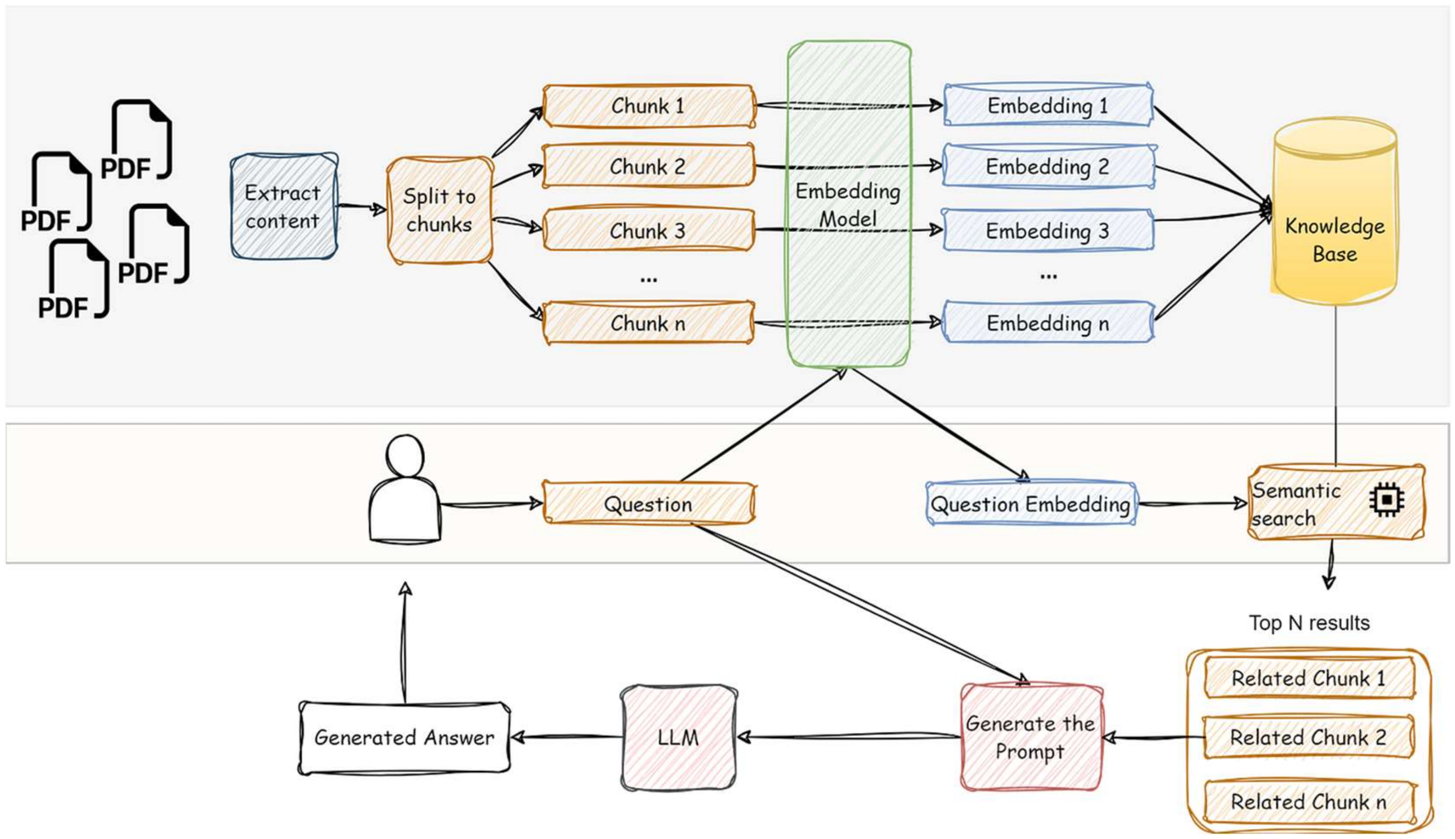
Milestone description	Category	Assigned to	Progress	Start	Days
Exploring Existing Solutions					
Explore off the Shelf Solutions	On Track	Cabdul	100%	9/15/2023	5
What is LangChain?	On Track	Cabdul	100%	9/18/2023	10
What is Streamlit?	On Track	Cabdul	100%	9/20/2023	1
Create UML Artifacts	On Track				
System Request	On Track	Cabdul	100%	9/15/2023	6
System Proposal	On Track	Cabdul	100%	9/15/2023	15
Functional Diagrams	On Track	Cabdul	100%	9/23/2023	6
Structural Diagrams	On Track	Cabdul	100%	10/1/2023	5
System Request Specification	On Track	Cabdul	100%	10/10/2023	5
Traceability Links	Low Risk	Cabdul	100%	11/5/2023	5
Change Management Plan	Low Risk	Cabdul	100%	11/10/2023	5
Codebase					
Setup Repository	On Track	Cabdul	100%	9/13/2023	1
Create Project Site	On Track	Cabdul	100%	10/5/2023	2
Setup Environment	On Track	Cabdul	100%	10/7/2023	3
Implement PDF Processing	On Track	Cabdul	100%	10/6/2023	7
Build UI for PDF Upload	On Track	Cabdul	100%	10/13/2023	1
Adding Persistance	Med Risk	Cabdul	100%	10/20/2023	7
Add User Login	Med Risk	Cabdul	100%	10/27/2023	7
Explore Free LLM/Embedding Models	Med Risk	Cabdul	50%	11/3/2023	7
Deploy Application	Med Risk		0%	11/10/2023	7

USE CASE DIAGRAMS





HIGH LEVEL ARCHITECTURE



Source

BE PYTHON/FLASK IMPLEMENTATION

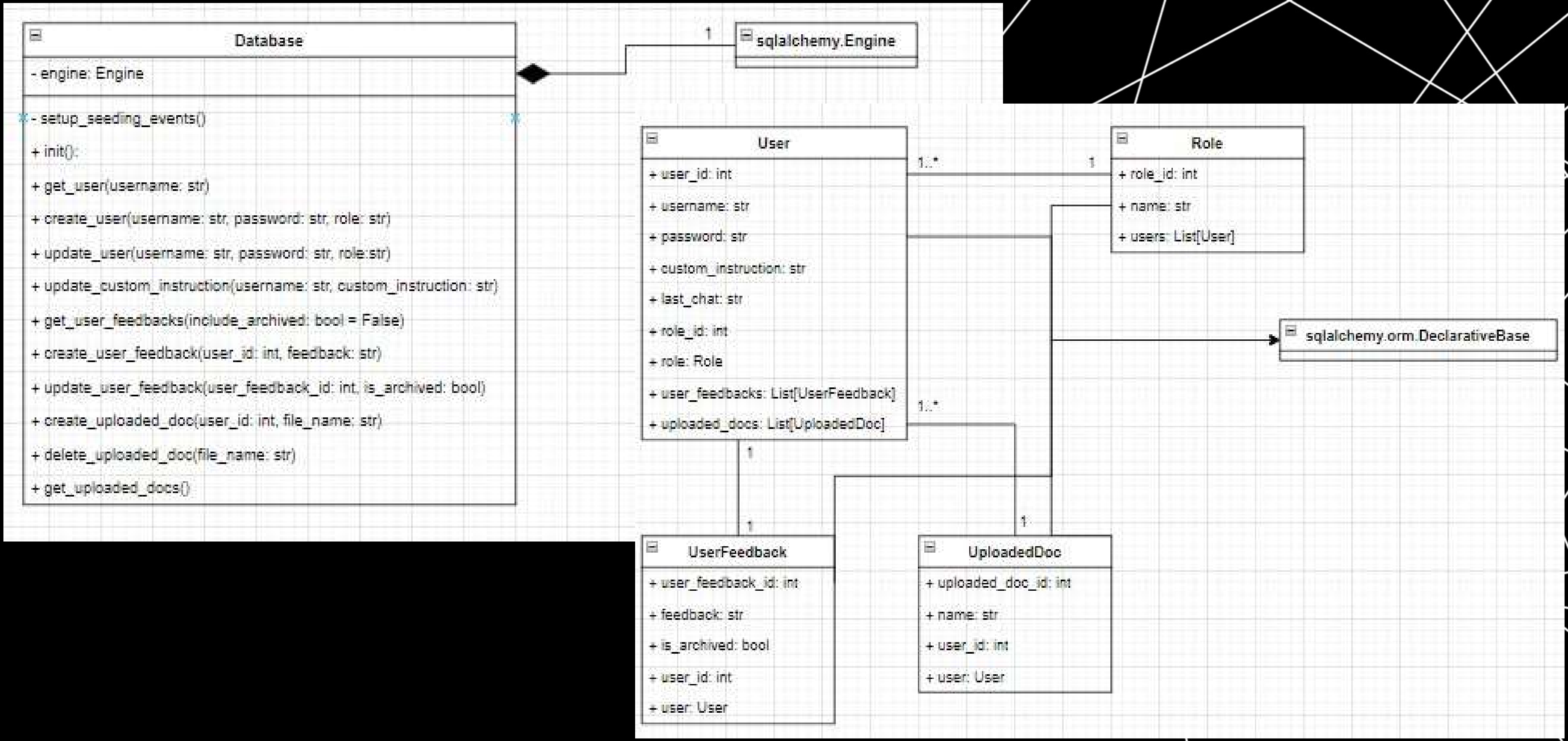
Using the following libraries	
	LangChain – framework for apps using language models
	Python-dotenv – virtual environment
	Pypdf2 – pdf reader
	Chromdb – vector store
	Bcrypt – password encryption
	Sqlalchemy – database ORM
	Flask – web server for API support

FE VUE IMPLEMENTATION

Using following dependencies

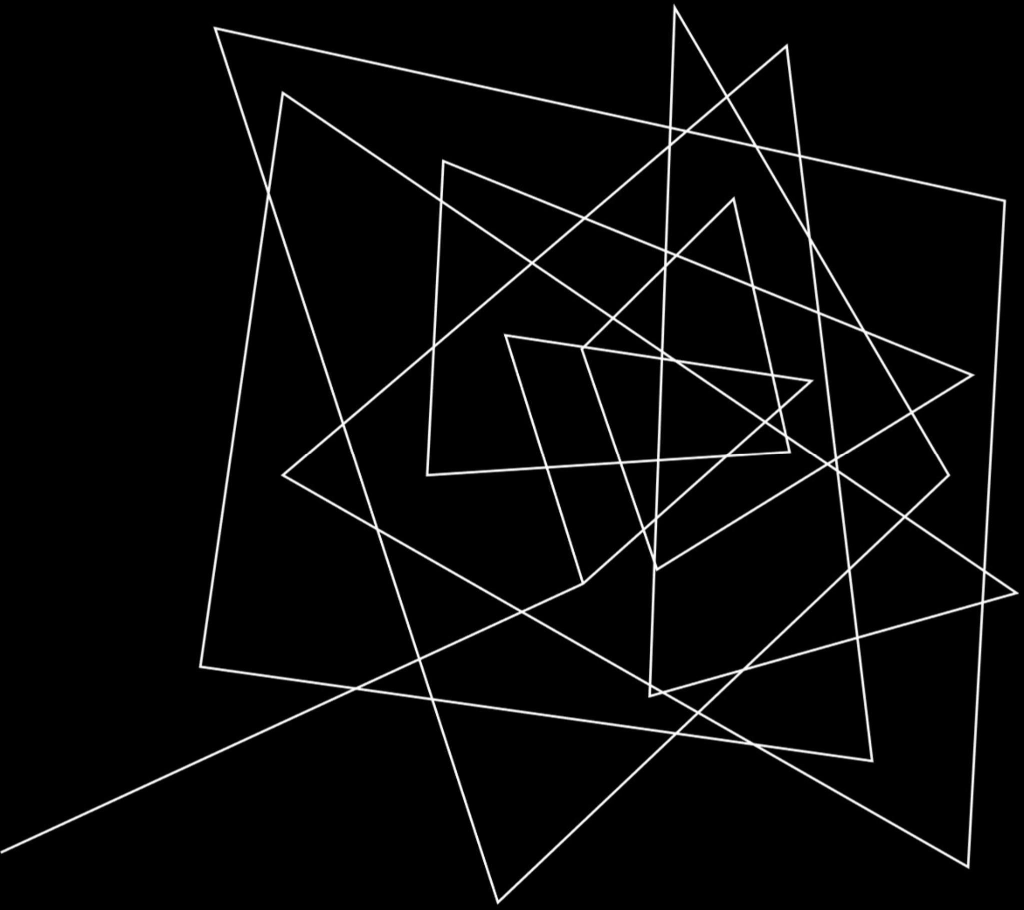
1. **Vite –** FE local development server
2. **Vue –** JS framework for building UI, like react and angular
3. **Vue-Router** – FE page routing
4. **Axios –** for making API calls
5. **Fetch –** for making streaming API call responses
   (axios does not support streaming from client side)
6. **Moment** – formatting datetime
7. **PrimeVue –** UI components (Theme, Datatable, Dialogs, Buttons, InputTexts, TextArea)
8. **PrimeIcons** – UI icons

PERSISTENCE (SQLITE) CLASS DIAGRAM


Database
- engine: Engine
- setup_seeding_events()
+ init():
+ get_user(username: str)
+ create_user(username: str, password: str, role: str)
+ update_user(username: str, password: str, role:str)
+ update_custom_instruction(username: str, custom_instruction: str)
+ get_user_feedbacks(include_archived: bool = False)
+ create_user_feedback(user_id: int, feedback: str)
+ update_user_feedback(user_feedback_id: int, is_archived: bool)
+ create_uploaded_doc(user_id: int, file_name: str)
+ delete_uploaded_doc(file_name: str)
+ get_uploaded_docs()
1
sqlalchemy.Engine
User
+ user_id: int
+ username: str
+ password: str
+ custom_instruction: str
+ last_chat: str
+ role_id: int
+ role: Role
+ user_feedbacks: List[UserFeedback]
+ uploaded_docs: List[UploadedDoc]
1..*
1
Role
+ role_id: int
+ name: str
+ users: List[User]
sqlalchemy.orm.DeclarativeBase
1..*
1
1
1
UserFeedback
+ user_feedback_id: int
+ feedback: str
+ is_archived: bool
+ user_id: int
+ user: User
UploadedDoc
+ uploaded_doc_id: int
+ name: str
+ user_id: int
+ user: User

PERSISTENCE CHROMADB

- ChromaDB python library creates a SQLite database on instantiation if one does not exist.
- This is where the knowledge base lives – the vector embeddings.
- Semantic search is run against this knowledge base to get top N matches for user prompts.
- My fav motto is, “If it ain’t broke, don’t fix it”, but with more time, I would have chosen to combine both ChromaDB’s database with my application’s SQLite database.



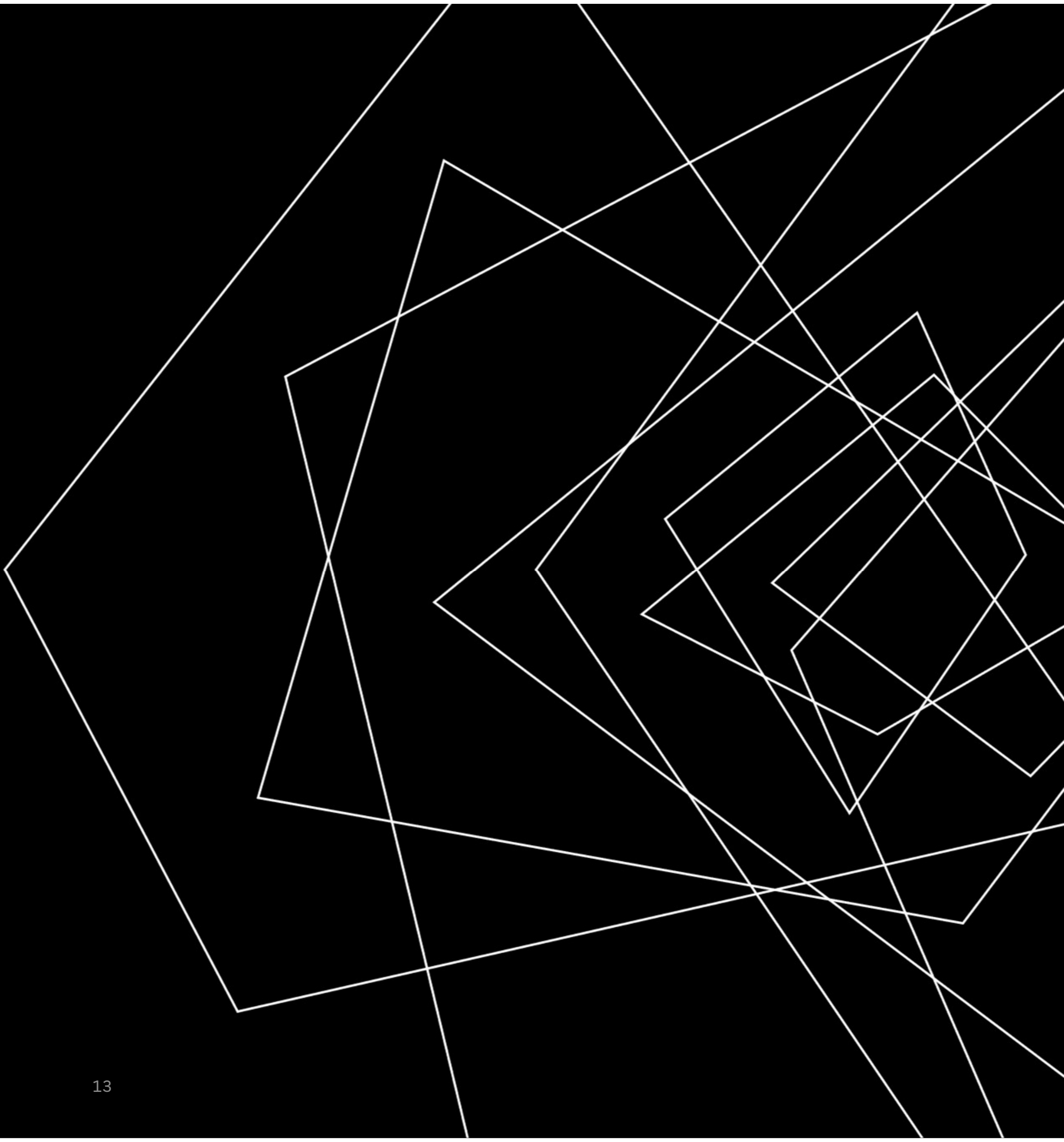
DEMO TIME

FUTURE FEATURES

- Setup CI/CD pipeline for the app and deploy it.
- Explore free LLMs and embedding models which will allow me to put this site up for “free”.
- Combine both ChromaDb’s and my applications SQLite databases, there’s no need to have two separate ones.
- Update app to use third part authentication like Google Auth, etc. Current implementation is not the most secure approach.

YOU CAN RUN THIS APP

- The app can only be run locally (for now).
- I made the GitHub repository public, so anyone can clone the repo and run the app locally.
- I only ask if you do use the app, let me know what you think about it (over email).
- Just follow the instructions on the source README.



REFLECTION

01

I am definitely happy with the “final” product.

02

There are more features I would like to add as discussed.

03

Certain implementation aspects took longer than expected since I was not familiar with Python/Flask prior.

ANY QUESTIONS