LLM Apps

Deep Dive: SEC Insights Full Stack App

End to End Production Professional App

- The LlamaIndex team has open sourced the project SEC Insights. Right now this is one of the most advanced and sophisticated production-ready LLM Apps available.

- It is worthy to study it in detail.

Main sources

- Github repo: https://github.com/run-llama/sec-insights/tree/main/backend
    - This is up-to-date
- Youtube video: https://www.youtube.com/watch?v=2O52Tfj79T4
    - This is out-to-date in some sections. When different, follow the instructions in the github repo since they are the most up-to-date.

Goals of the app

- Chat application
- RAG technique
- Answers questions about SEC 10k and 10Q documents
- Production-ready
- Full-stack repo
- Ready for you to fork
- All the setup is open source and is easy to deploy on Vercel and Render.com

See the working app live

- secinsights.ai

Features

- QA chat grounded in source-of-truth SEC documents
- PDF viewer
- Token-level streaming of chat responses
- Streaming of reasoning steps (sub-questions)
- Citation of source data
- Use of API-based tools (in addition to semantic search)

Goals of the Deep Dive

- Deploy to production.
- Understand the role of LLM.
- Understand the placement of the LlamaIndex code.
- Check if there is a testing framework in place.
- Additional parts to pay attention to:
  - Github codespace.
  - AWS S3.
  - Vercel and Render.com.
  - LocalStack.
  - Cron Job.

Tech Stack

- Frontend
  - React / Next.js
  - Tailwind CSS
- Backend
  - FastAPI
  - Docker
  - SQLAlchemy
  - OpenAI
    - GPT-3.5-turbo > GPT 4
    - Text-embedding-ada-002
  - PGVector
  - LlamaIndex

- Infrastructure
  - Render.com
    - Backend hosting
    - Postgres 15
  - Vercel
    - Frontend hosting
  - AWS
    - Cloudfront
    - S3

LLM Logic: role and placement

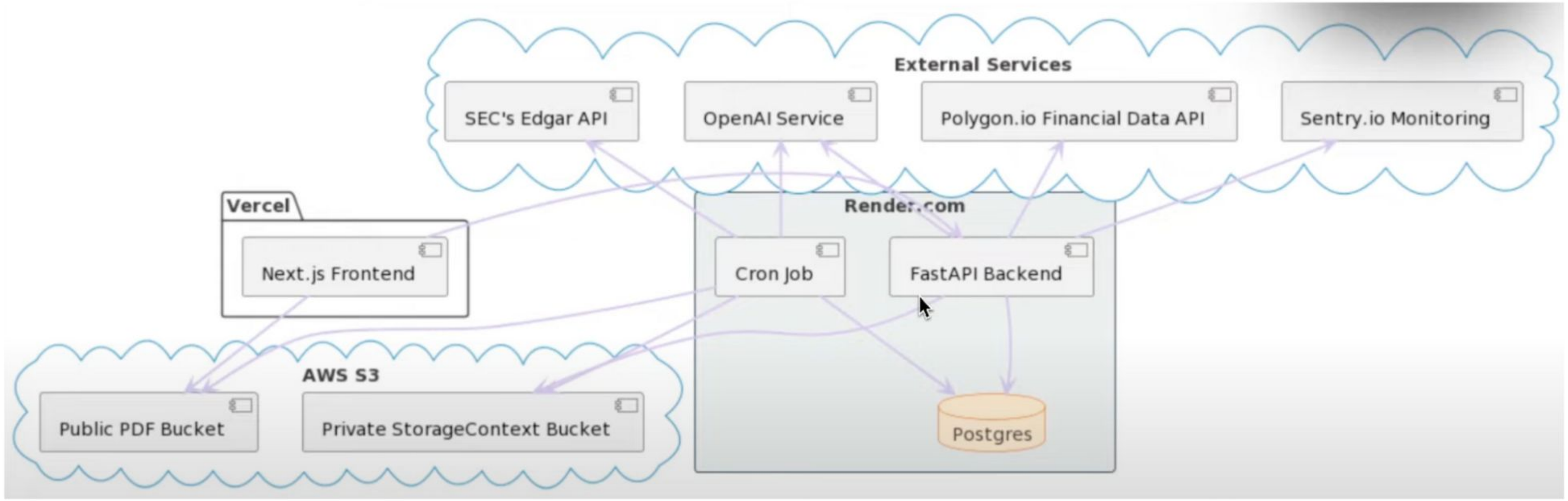
- Role: advanced RAG application
- Main placement: /backend/app/chat/engine.py
- Secondary placements in same folder:
    - messaging.py
    - pg_vector.py
    - A_response_synth.py

- Python version of LlamaIndex

Testing functionality

- Limited testing functionality in /backend/tests/app/chat/test_engine.py
    - TestGetChatHistory

Architecture



Backend

- Render.com: hosting most of the backend.
  - Similar to AWS but easier to use.

- FastAPI Backend Service.
  - Traditional load-balanced API Service.
  - Load balancing requests to service instances.
  - Auto-scaling

- Postgres 15 database
- Cron job service
- All of the above prepared for us in the file render.yaml

AWS S3

- You will have go to AWS and setup this yourself

- Private StorageContext Bucket
  - metadata from the llamaindex library

- Public PDF Bucket

Frontend

- NextJS
- Hosted in Vercel
- Interacts with the FastAPI backend for some of the chat endpoints

External services

- Polygon.io (Financial Data API, Numeric Data). Good example on how to integrate tools in your chat agent.
- SEC's Edgar API
- OpenAI Service (LLM)
    - The cron service is the main worker that calls the OpenAI Embedding API to get the embeddings for the given SEC documents.
    - The cron service calls the Edgar API from the SEC, get the PDFs, store them in the AWS Public PDF Bucket, run the embeddings on them and store the embeddings in the Postgres database (we use the PG Vectorstore integration).
    - The cron job can run at whatever schedule you set in the render.yaml file.

- Sentry.io (Production-level Monitoring Service. It will ping you whenever there is an error in the backend service, or threshold errors, etc. You can also do Performance Monitoring: what sections of the code are taking more time in your service)

Contents of the render.yaml file

- Configuration for deploying and managing the app in Render.com
- General configurations.
- DB configuration.
- Services:
  - Web service.
  - Cron service.
- Environment variable groups.
  - General settings: variables for all environments.
  - Production environment.
  - Preview environment.

AWS S3

- How to create an account.
- How to create a private bucket.
- How to create a public bucket.

Cron Job

- What is it.
- How to set it up in Render.com

Dev Environment

- Github Codespace.
- Terminal commands in the github codespace.
    - Frontend

Backend

- Terminal commands for /backend in the github codespace.
  - Poetry shell
  - Poetry install
  - .env file
  - Source .env
  - Temporary environment variables
  - Start the backend server
    - Make migrate
    - Make run

  - Final environment variables

DB population with SEC filings

- Make seed_db_local

How to use this app with your own private documents

- Example working with the app in your local environment.
- How to load a new document.

For any issues

- Troubleshooting guide.
- Github issues.
- Discord channel.

SEC Document Downloader

- Script to download SEC files.

Setup / Usage instructions

- Pre-requisite setup steps to use the downloader script to loan the SEC PDFs directly into an S3 bucket.

Seed DB Script

- Collection of scripts for seeding the database with a set of SEC documents.

Deep dive into the key LlamaIndex logic

- Main file
- Secondary files.
- Other interesting files.