LLM Apps

LlamaIndex: End to End Production Professional 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

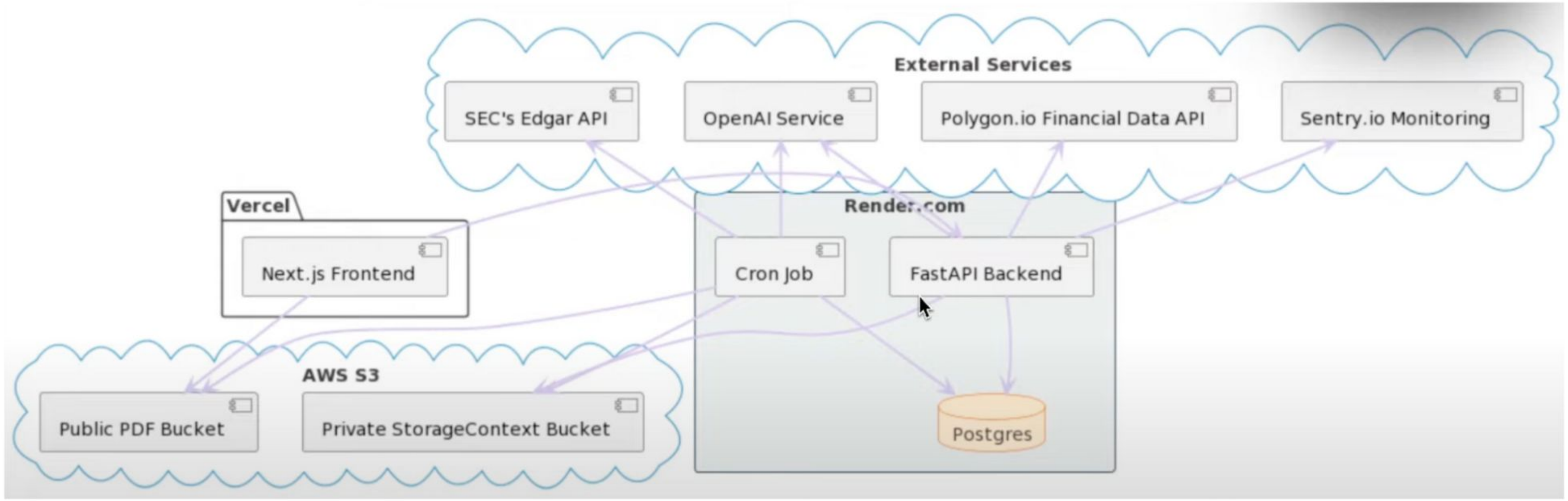
See the working app live

- 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)

Architecture



Frontend

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

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

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)

Setup and start the frontend server

- Create a Github Codespace.
- cd frontend
- ls
- **This is a basic vercel app**
- npm install
- source the .env.example folder, load the environment variables that are here
- the url there is the local one of the backend (localhost/8000), will have to be changed when we use the backend in the cloud.
- set -a
- source .env.example
- npm run dev
- that starts running the app in localhost/3000
- It comes with live reload, so if you edit any UI file it will show immediately. For example, you can edit the title in components/landing-page/TitleAndDropdown.tsx

Setup the backend

- With the app open in one terminal, open a second terminal and cd backend
- **This is a fastAPI python backend app.** Most of this app is based in the templates fastAPI offers
- In the github folder, go to /backend and read the readMe file.
- cd backend
- **No need to install pyenv nor docker if you are running from the devcontainer image in Github Codespaces**
- cat .python-version: confirms you have 3.11.3
- poetry shell
- Now in the github codespaces terminal you see (llama-app-backend-py3.11) as virtual environment
- poetry install
- Now you have to create the backend/.env file
  - cp .env.development .env
  - set -a
  - source .env

Start the backend server

- make migrate (runs the database migrations)

- make run (starts the server locally)
    - This spins up the Postgres 15 DB & Localstack in their own docker containers.
    - The server will not run in a container but will instead run directly on your OS.
    - This is to allow for use of debugging tools like pdb

Enter your private keys

- Open your .env file and replace the placeholder value for the OPENAI_API_KEY with your own OpenAI API key

- At some point you will want to do the same for the other secret keys in here like POLYGON_IO_API_KEY, AWS_KEY, & AWS_SECRET

- To follow the SEC's Internet Security Policy, make sure to also replace the SEC_EDGAR_COMPANY_NAME & SEC_EDGAR_EMAIL values in the .env file with your own values.

- Source the file again with set -a then source .env

- **Remember to add .env in the .gitignore file**

Populate your local DB with sample SEC filings

- make seed_db_local
  - If this step fails, you may find it helpful to run make refresh_db to wipe your local database and re-start with emptied tables.

- You can run “make run” again and you should see some documents loaded at http://localhost:8000/api/document

If you have any issues in the previous process

- For any issues in setting up the above or during the rest of your development, you can check for solutions in the following places:
  - backend/troubleshooting.md
  - Open & already closed Github Issues
  - The #sec-insights discord channel

SEC Document downloader

- **We have a script to easily download SEC 10-K & 10-Q files.**
    - This is a single step of the larger seed script described in the next section.
    - Unless you have some use for just running this step on it's own, you probably want to stick to the Seed script described in the section below.
    - However, the setup instructions for this script are a pre-requisite for running the seed script.

- **No API keys are needed to use this**, it calls the SEC's free to use Edgar API.

- **The instructions below explain a process to use the script to download the SEC filings, convert the to PDFs, and store them in an AWS S3 bucket.**

Setup / Usage instructions

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

- These steps assume you've already followed the steps above for setting up your dev workspace.

- Next, we will see:
  - Setup AWS CLI
  - Setup s3fs
  - Install wkhtmltopdf
  - Get into your poetry shell with poetry shell from the project's root directory.
  - Run the script
  - Go to AWS Console and verify you're seeing the SEC files in the S3 bucket.

Setup AWS CLI

- Install AWS CLI
  - **This step can be skipped if you're running from the devcontainer image in Github Codespaces**
  - Steps:
    - curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
    - unzip awscliv2.zip
    - sudo ./aws/install

- Configure AWS CLI
  - This is mainly to set the AWS credentials that will later be used by s3fs
  - Run aws configure and enter the access key & secret key for a AWS IAM user that has access to the PDFs where you want to store the SEC files.
    - set the default AWS region to us-east-1 (what we're primarily using).

Setup s3fs

- Install s3fs
  - **This step can be skipped if you're running from the devcontainer image in Github Codespaces**
  - sudo apt install s3fs

- Setup a s3fs mounted folder
  - Create the mounted folder locally mkdir ~/mounted_folder
  - s3fs llama-app-web-assets-preview ~/mounted_folder
    - You can replace llama-app-web-assets-preview with the name of the S3 bucket you want to upload the files to.

Install wkhtmltopdf

- **This step can be skipped if you're running from the devcontainer image in Github Codespaces**

- Steps:
    - sudo apt-get update
    - sudo apt-get install wkhtmltopdf

Setup / Usage instructions: Last steps

- Get into your poetry shell with poetry shell from the project's root directory.

- Run the script
  - python scripts/download_sec_pdf.py -o ~/mounted_folder --file-types="['10-Q','10-K']"
  - Take a break while it's running, it'll take a while.

- Go to AWS Console and verify you're seeing the SEC files in the S3 bucket.

Seed DB Script

There are a collection of scripts we have for seeding the database with a set of documents. The script in scripts/seed_db.py is an attempt at consolidating those disparate scripts into one unified command.

This script will:

- Download a set of SEC 10-K & 10-Q documents to a local temp directory
- Upload those SEC documents to the S3 folder specified by $S3_ASSET_BUCKET_NAME
- Crawl through all the PDF files in the S3 folder and upsert a database row into the Document table based on the path of the file within the bucket

Seed DB Script: Use Cases

This is useful for times when:

- You want to setup a local environment with your local Postgres DB to have a set of documents in the documents table
  - When running locally, this will use localstack to store the documents into a local S3 bucket instead of a real one.

- You want to update the documents present in either Prod or Preview DBs
  - In fact, this is the very script that is run by the llama-app-cron cron job service that gets setup by the render.yaml blueprint when deploying this service to Render.com.

Seed DB Script: Usage (1)

To run the script, make sure you've:

- Activated your Python virtual environment using poetry shell
- Installed all the pre-requisite dependencies for the SEC Document Downloader script.
- Defined all the environment variables from .env.development within your shell environment according to the environment you want to execute the seed script (e.g. local, preview, prod environments)

After that you can run python scripts/seed_db.py to start the seed process.

Seed DB Script: Usage (and 2)

To make things easier, the Makefile has some shorthand commands.

- **make seed_db**
  - Just runs the seed_db.py script with no CLI args, so just based on what env vars you've set
- **make seed_db_preview**
  - Same as make seed_db but only loads SEC documents from Amazon & Meta
  - We don't need to load that many company documents for Preview environments.
- **make seed_db_local**
  - To be used for local database seeding
  - Runs seed_db.py just for `$AMZN & $META` documents
  - Sets up the localstack bucket to actually serve the documents locally as well, so you can load them in your local browser.
- **make seed_db_based_on_env**
  - Automatically calls one of the above shorthands based on the RENDER & IS_PREVIEW_ENV environment variables