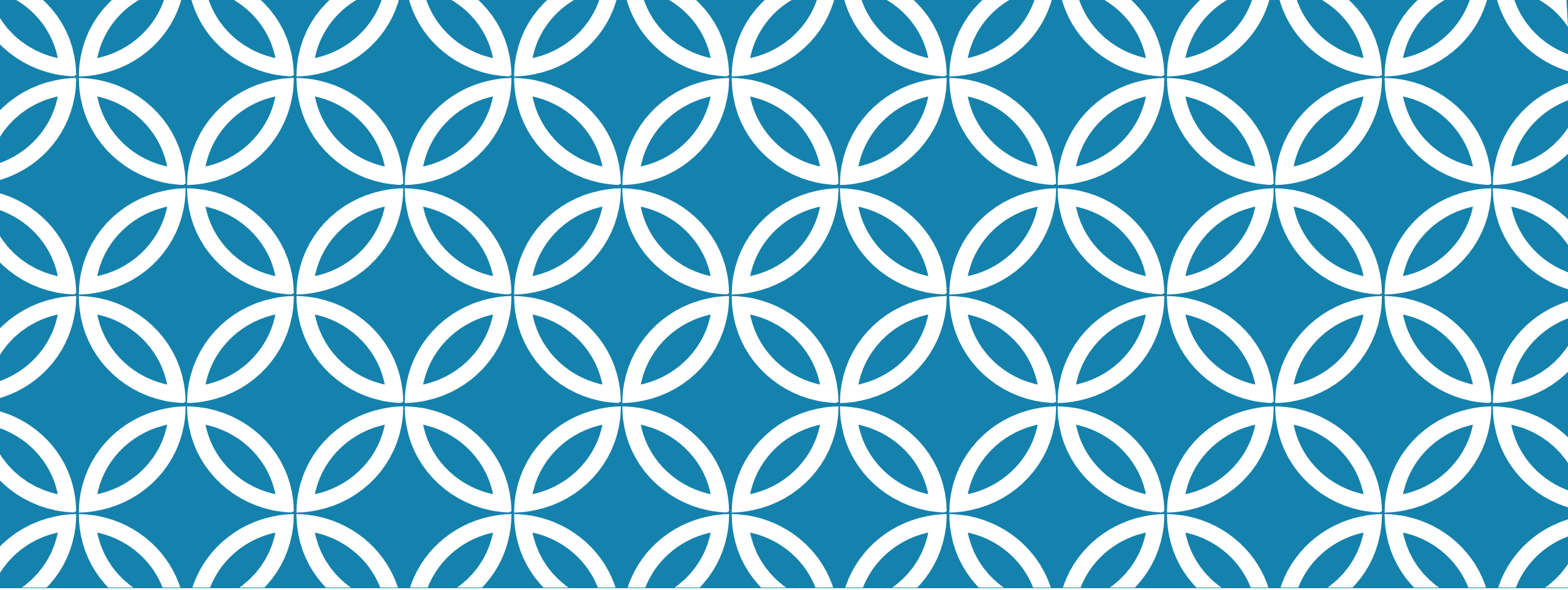


RESEARCH BUDDY

Hoa Nguyen

28-08-2023

OUTLINE

Application design

- Application functionalities
- Language models

Architecture design

- Architecture overview
- Architecture and deployment
- More on deployment
- How does conversational QA retrieval work?
- How does SQL agent work?
- How does summarizer chain work?
- How does TOPIC MODELLING work?

APPLICATION DESIGN

APPLICATION FUNCTIONALITIES

Main theme: chat with your own data

Functionalities

- Summarize and upload:
  - Upload your own internal PDF files that you want to ask the bot about. (e.g. papers, books, guideline…)
  - Summarize your PDF documents and Youtube videos without uploading.
  - Extract topics for each summary based on a predefined list of topics
  - You can decide whether the PDF document is useful based on its summary and upload to the knowledge base for question answering later.
- Research buddy
  - General question answering based on chatbot's own knowledge.
  - Question answering based on your uploaded documents with chain-of-thoughts.
  - The bot has chain-of-thoughts so it can decide whether to use your uploaded documents or its own knowledge.
  - The chain-of-thoughts can be glimpsed with option "Show intermediate steps".
  - The bot is also capable of doing simple calculations.
- Research bot
  - Question answering mostly based on your uploaded documents.
  - It can answer general questions not only based on your uploaded documents, but less flexible than research buddy.
  - The source documents for its answers can be checked with "Show source documents", so you can be more certain.
- Social media search
  - Display and visualize social media data for analysis.
  - Then you can decide to upload to the database, visualize and delete all data.
  - Question answering to derive insightful analytics from data in the sql database.
  - The agent's chain-of-thoughts can be glimpsed with option "Show intermediate steps".

APPLICATION DESIGN

LANGUAGE MODELS

Chat model: chatgpt - gpt-3.5-turbo

Embeddings model: openai - text-embedding-ada-002

These models are chosen simply because they are among the best models so far in the market and they are not the most expensive.

However, other models can also be used with the interface of langchain.

Lately there are more open source chat models that can perform quite comparably to chatgpt 3.5-turbo such as LLAMA2 70b parameters..

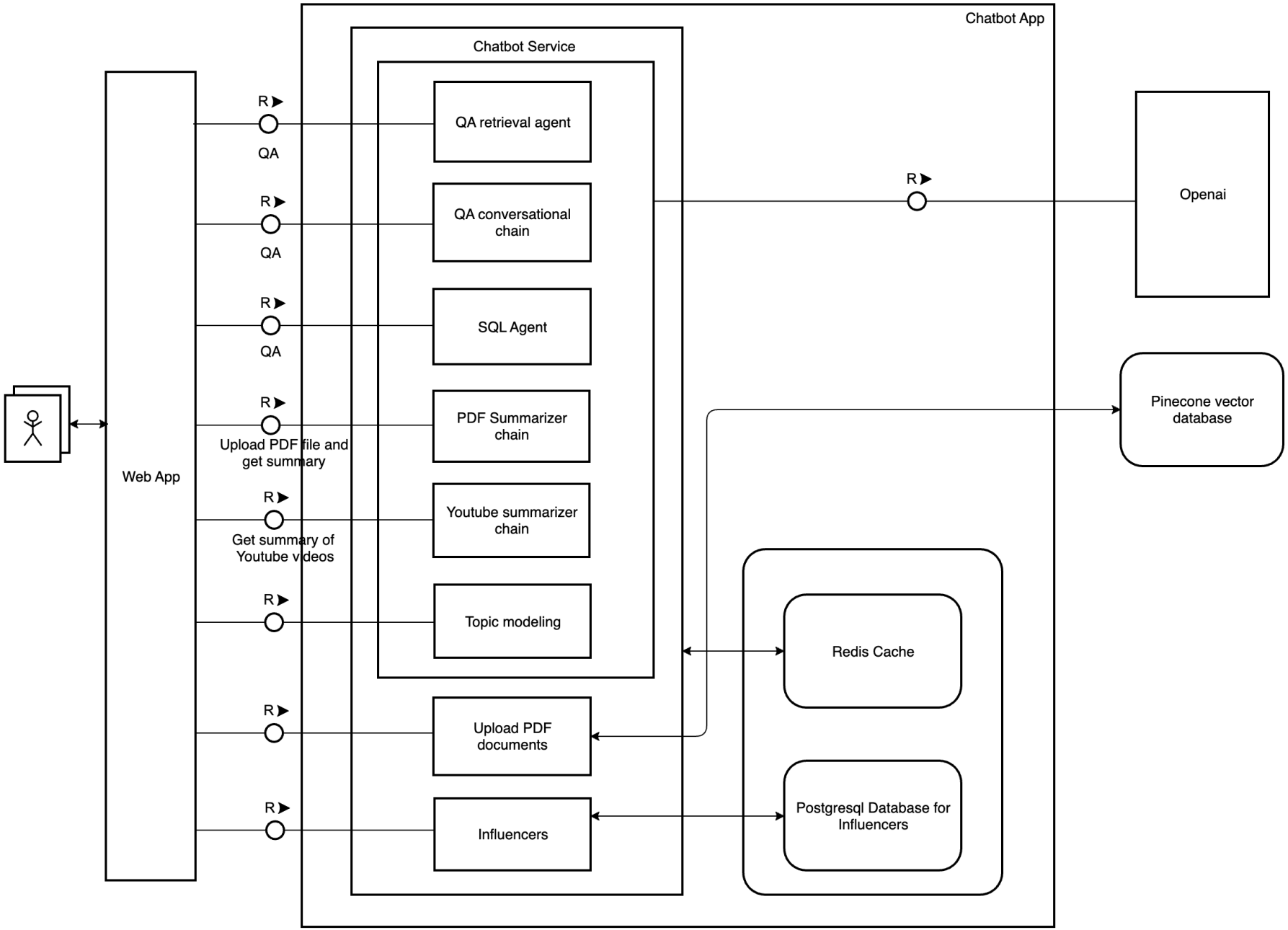
However, deploying them can be expensive without economy of scale, so for the purpose of demo app, I won't consider them.

For the demo purpose, finetuning may not be necessary, but we can finetune much smaller models with (human or AI – labelled) data to lower the cost or even better use smaller models with finetuned version for specific task (e.g. translating texts to SQL statement or topic modelling).

(E.g. code Llama-34b-instruct is only 6 percent accuracy points behind compared to few-shot GPT4 for SQL-generating tasks – snowflake llama2-code)

ARCHITECTURE DESIGN

ARCHITECTURE OVERVIEW



ARCHITECTURE DESIGN

ARCHITECTURE AND DEPLOYMENT

Web app consumes API from chatbot service.

Redis cache and postgresql database are deployed within the same private network as chatbot service for more secure and faster access.

The chatbot service consumes API from OpenAI and is connected to externally managed Pinecone vector database store.

(Because currently Pinecone is one of the leading vector stores, but we can totally deploy in-memory open-source vector store. Weaviate is also a great choice for open-source vector database).

Redis cache is used to cache chat history for longer conversations and get request for influencers.

As there is only 1 client app and both client app and chatbot service are deployed in the same platform, I deploy the client app and chatbot service in the same network and connect them within the private network.

And as there is no authentication and authorization required, there is no need to use API Gateway to for API calls. However, API is a great way to perform authentication, authorization, routing and load balancing for API calls.

Client app is exposed to the internet via application load balancer.

All passwords are managed via AWS secrets manager, and all services are deployed in AWS Elastic Container Service.

Note: Although this application is quite simple and I may not need to separate web app from the chatbot service, this serves as an example how I will build the same microservices for multiple client apps or separate payloads for different services to scale easily.

ARCHITECTURE DESIGN

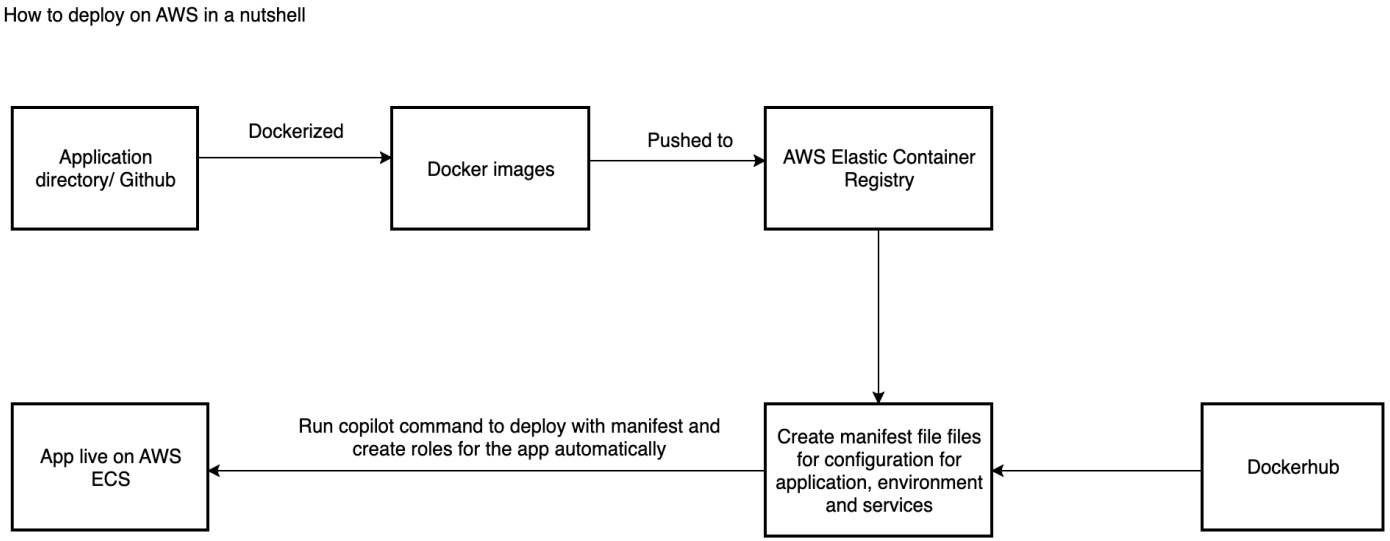
MORE ON DEPLOYMENT

Docker image can be built from application directory or github, and then are pushed to AWS container registry.

I can pull the docker image from AWS docker registry or pull a standard docker (e.g. for postgresql) from dockerhub to deploy it.

I use AWS copilot to deploy faster on AWS Fargate, Elastic Container Service which is basically serverless architecture.

Docker containers for database is mounted to AWS Elastic File System for persistent storage.



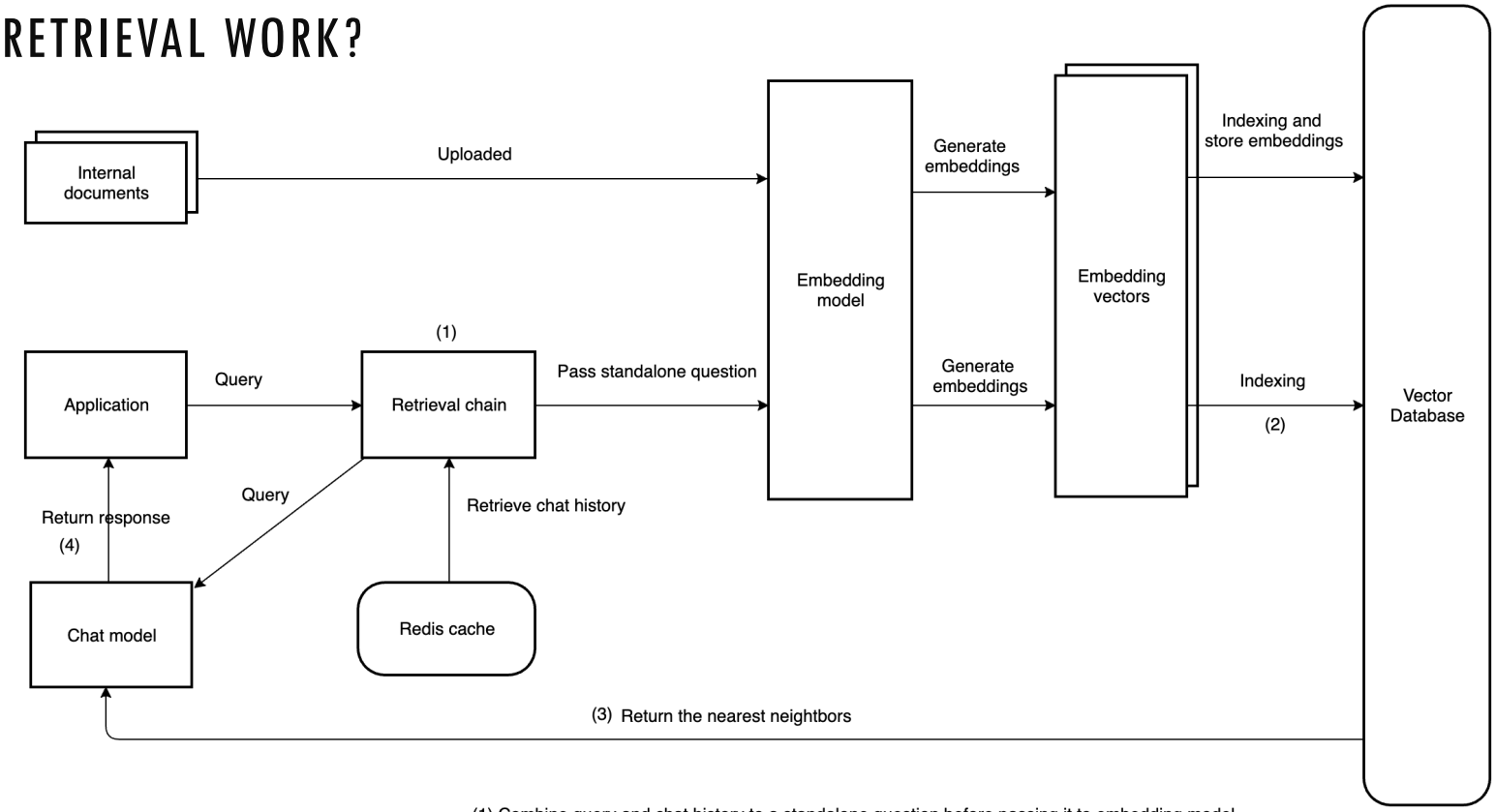
ARCHITECTURE DESIGN

HOW DOES CONVERSATIONAL QA RETRIEVAL WORK?

Conversational QA retrieval is used for research bot.

QA Agent is used for research buddy.

QA Agent is similar to conversational QA retrieval except that in the last step, it has chain-of-thoughts to return more conversation-like responses.



(1) Combine query and chat history to a standalone question before passing it to embedding model.

(2) Indexing: mapping vectors using an algo such as product quantization.. to a data structure that will enable faster searching.

(3) Compare the indexed query vector and find the nearest neighbors based on a similarity metric (e.g. cosine similarity).

(4) Combine the question and the relevant documents found to return a response.

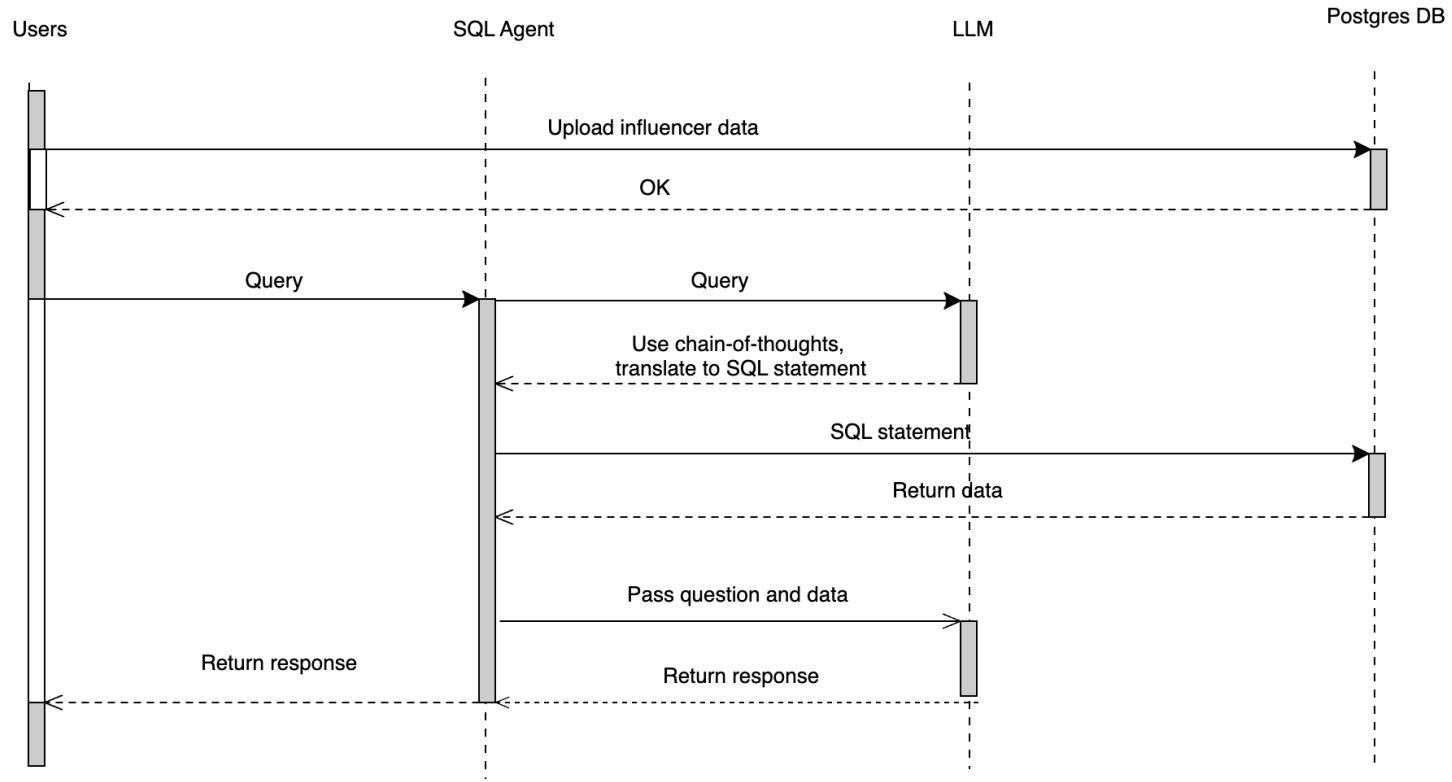
ARCHITECTURE DESIGN

HOW DOES SQL AGENT WORK?

SQL Agent has a trial and error mechanism. If for some reasons, sql statement cannot return data, it would try again with the modified sql statement.

SQL Agent uses zero_shot_react_description, so it means it does not have memory.

However, it's still in active development by langchain and could be improved later.

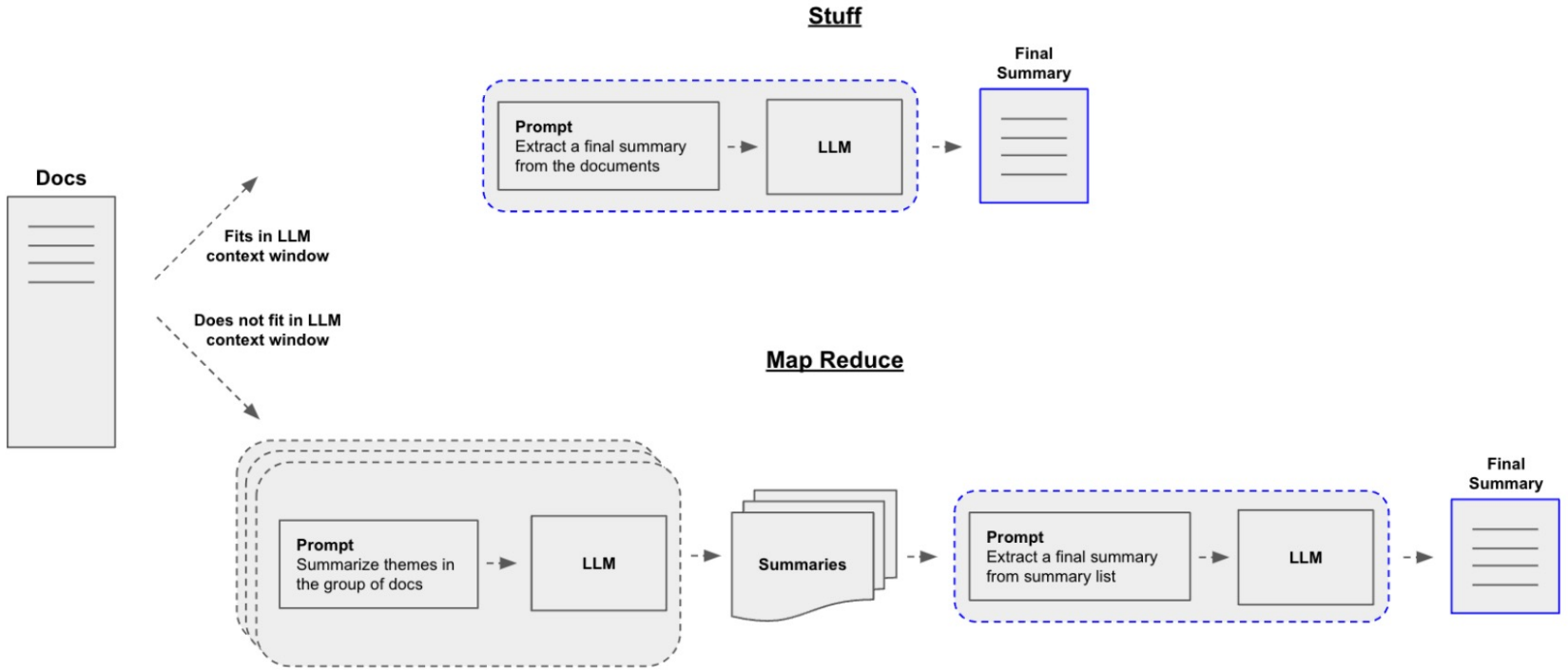


ARCHITECTURE DESIGN

HOW DOES SUMMARIZER CHAIN WORK?

For this app, user is able to upload a big PDF file or an Youtube video, so I use map reduce approach to extract summary.

For Youtube videos, there is one more step which extracts transcripts from youtube videos before sending them to summary chain.



From Langchain

ARCHITECTURE DESIGN

HOW DOES TOPIC MODELLING WORK?

I also use chat model to derive the topics from a predefined list of topics.

However, we can totally combine chat model with vector stores to derive the relevant topics based on topic descriptions.

Input

Our legacy financial infrastructure has both limited growth opportunities and contributed to the inequality of opportunities. Small businesses, even those with a banking relationship, often must rely on high-cost financing, such as credit cards, because traditional banking excludes them from loan financing. Decentralized finance, or DeFi, poses a challenge to the current system and offers a number of potential solutions to the problems inherent in the traditional financial infrastructure.

Schema

```
{
"properties": {
        "topic": enum from topic list
        }
}
```

Function

"name": "information extraction"
"description": "Extract info from the passage"
"parameter": {schema}

→ Topics: Economics, Blockchain

ARCHITECTURE DESIGN

SOME NOTES

As this is a demo app, the implementation of unit tests and integration tests has not been completed.

Some of the unit tests have been implemented in the chatbot service. For other modules, unit tests and integration test can mostly be implemented similarly.

For production app, it's best to implement all of the unit tests and integration tests and integrate them into CI/CD pipeline for safer deployment.

Thank you.