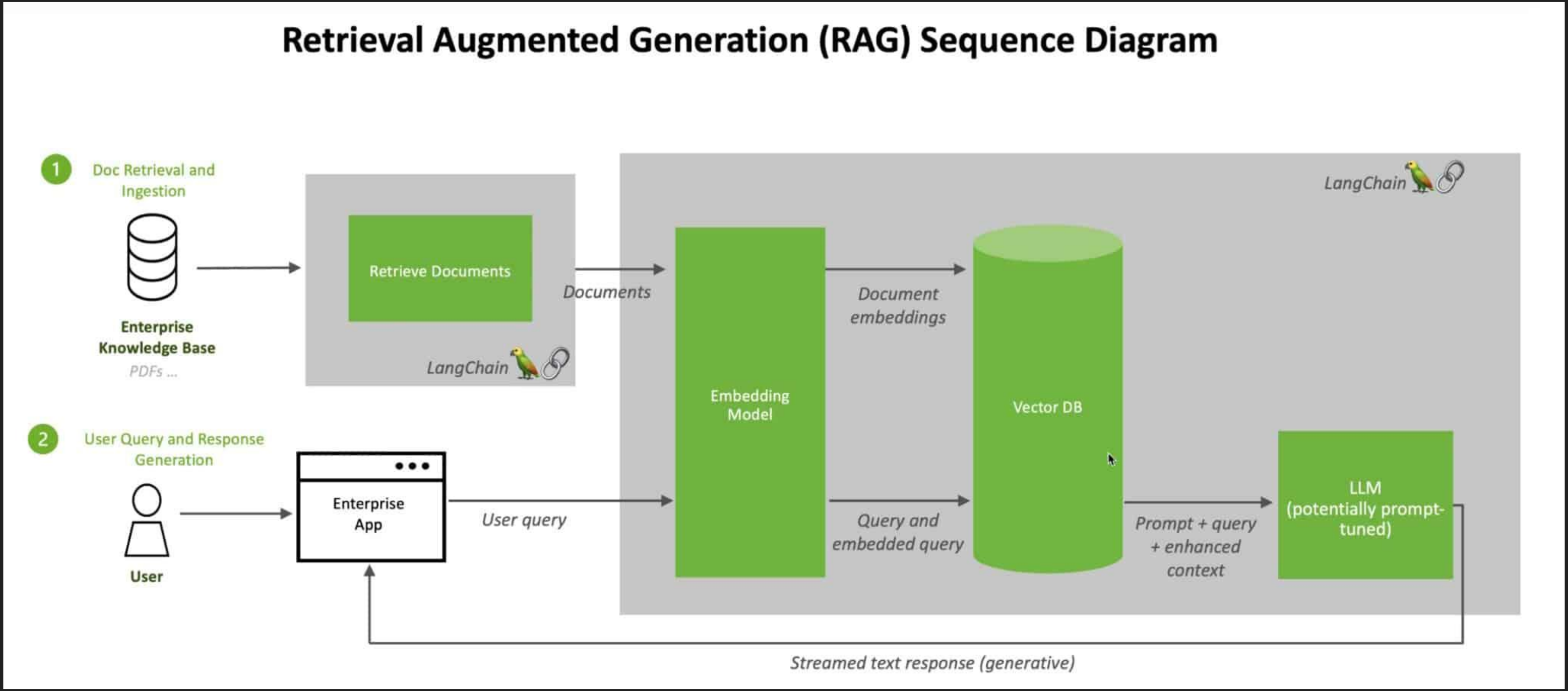
LlamaIndex with Next.js

https://github.com/danielbank/llamaindex-nextjs-demo

Daniel Bank

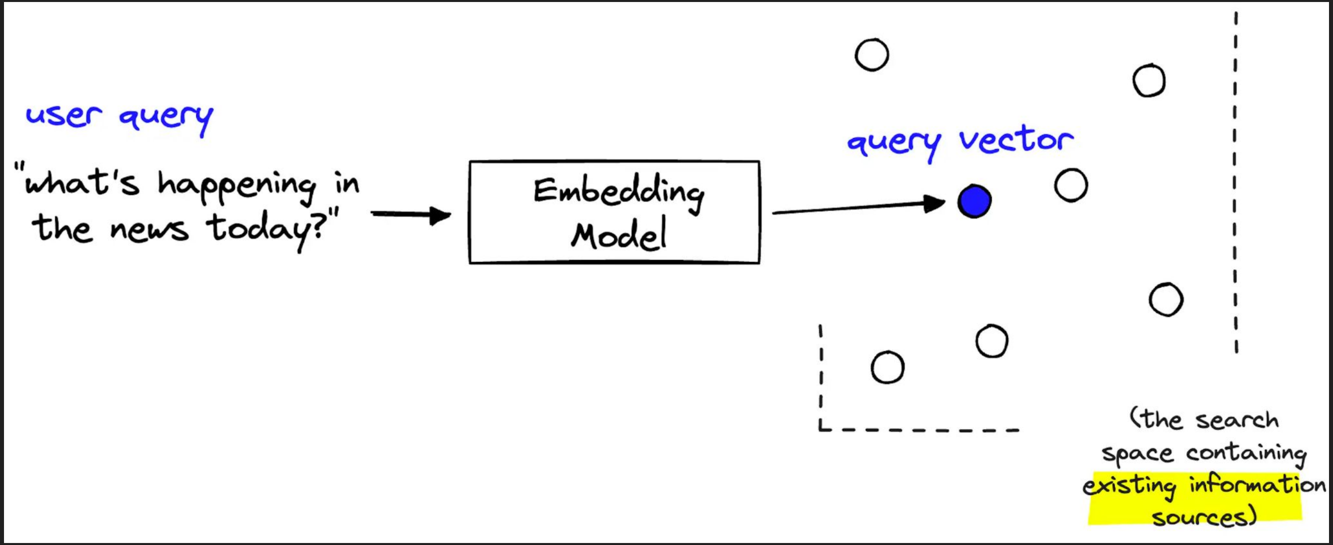
A Brief Review of Retrieval Augmented Generation (RAG)

Traditional RAG Pipeline (from NVIDIA's blog)

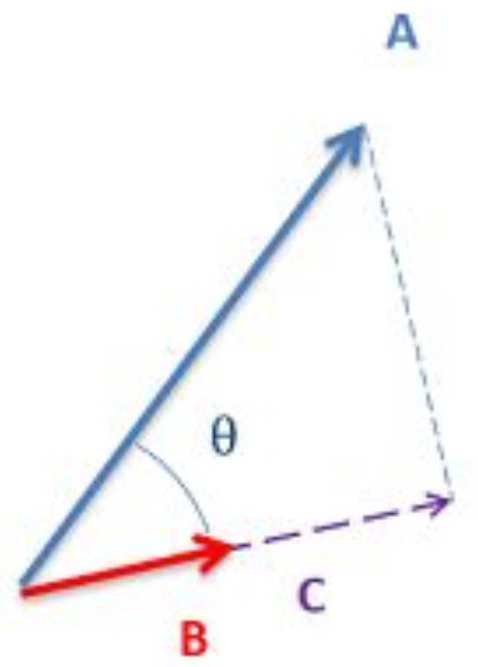

Retrieval Augmented Generation (RAG) Sequence Diagram
1
Doc Retrieval and Ingestion
Enterprise Knowledge Base
PDFs ...
Retrieve Documents
LangChain
Documents
Embedding Model
Document embeddings
Vector DB
LangChain
2
User Query and Response Generation
User
Enterprise App
User query
Query and embedded query
Prompt + query + enhanced context
LLM (potentially prompt-tuned)
Streamed text response (generative)

Embedding Models

- An embedding model is a neural network that converts text into dense vector representations (embeddings).
- Different types of Embedding Models based on modality (text, image, audio)
- [Massive Text Embedding Benchmark (MTEB)]


user query
"what's happening in the news today?"
Embedding Model
query vector
(the search space containing existing information sources)

Vector Similarity (Dot Product)



$$\vec{A} \cdot \hat{B} = |A||B| \cos(\theta)$$

if the magnitude of B is 1, then...

$$C = \vec{A} \cdot \hat{B} = |A| \cos(\theta)$$

Databases (A gentle introduction to Vector DBs)

- Relational Databases
  - When we think of databases, relational databases are usually what we picture
  - Great for Structured Data
  - Data is search by columns

ISBN	Year	Name	Author
0767908171	2003	A Short History of Nearly Everything	Bill Bryson
039516611X	1962	Silent Spring	Rachel Carson
0374332657	1998	Holes	Louis Sachar

Vector Databases (A gentle introduction to Vector DBs)

- Vast majority of data on the internet is unstructured (images, text, audio, video)
- Data is searched via content rather than keywords (similarity score, e.g. dot product)
  - Side note: PageRank (1998), the algorithm that powers Google Search, uses eigenvectors to determine page rank (similarity of pages to the query keywords)

Data UID[1]	Vector representation
00000000	[-0.31, 0.53, -0.18, …, -0.16, -0.38]
00000001	[0.58, 0.25, 0.61, …, -0.03, -0.31]
00000002	[-0.07, -0.53, -0.02, …, -0.61, 0.59]

Tour of the LlamaCloud Products

LlamaParse for Parsing Unstructured Documents

- LlamaParse is a proprietary parsing service, exceptionally adept at transforming PDFs containing complex tables into a neatly organized markdown format:
    - What language to use for OCR?
    - Is there any text at different orientations?
    - Are there special instructions for the Embedding Model?
- Get the data we intend to insert into the Vector DB in a well-defined shape
    - Example: We have text data in different file types (.md, .pdf, .csv, etc.)

LlamaCloud for Managed RAG Pipelines

- LlamaCloud is a managed RAG pipeline running on AWS which also can serve as a managed Vector DB (Qdrant)
- You can organize by:
  - Organizations
    - Projects
      - Indexes (Pipelines)
- Data Sources: Google Drive, SharePoint, S3 Bucket, Box, OneDrive
- Data Sinks: Pinecone, Postgres, Milvus, Azure AI Search
- Embedding Models: Open AI, Cohere, Gemini, HuggingFace, etc.

LlamaReport for Report Generation

Let’s build a RAG-Powered Web App with LlamaIndex and Next.js

Demo Time

- GitHub Repo: https://github.com/danielbank/llamaindex-nextjs-demo

LlamaIndex - https://github.com/run-llama/LlamaIndexTS

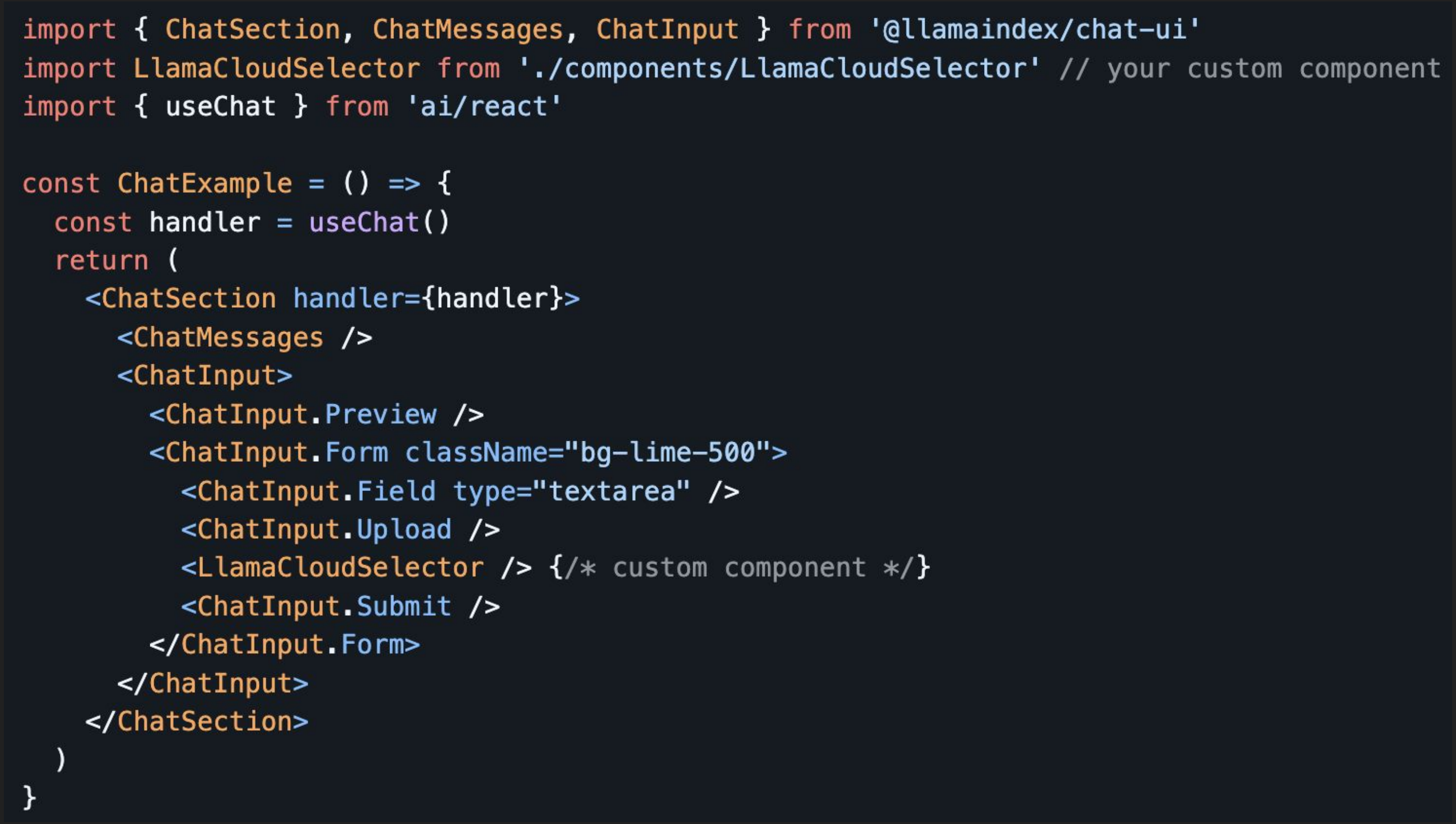
- **`npx create-llama@latest`** - Run the CLI tool to bootstrap a web app
- **`npm run generate`** - Create the Document Embeddings for the knowledge-base data in **`./data`** (assuming SimpleDirectoryReader example)
- **`npm run dev`** - Run the web app

Chat Components with LlamaIndex Chat UI

Batteries-Included <ChatSection />

```
import { ChatSection } from '@llamaindex/chat-ui'
import { useChat } from 'ai/react'
const ChatExample = () => {
  const handler = useChat()
  return <ChatSection handler={handler} />
}
```

Component Composition



```
import { ChatSection, ChatMessages, ChatInput } from '@llamaindex/chat-ui'
import LlamaCloudSelector from './components/LlamaCloudSelector' // your custom component
import { useChat } from 'ai/react'

const ChatExample = () => {
  const handler = useChat()
  return (
    <ChatSection handler={handler}>
      <ChatMessages />
      <ChatInput>
        <ChatInput.Preview />
        <ChatInput.Form className="bg-lime-500">
          <ChatInput.Field type="textarea" />
          <ChatInput.Upload />
          <LlamaCloudSelector /> {/* custom component */}
          <ChatInput.Submit />
        </ChatInput.Form>
      </ChatInput>
    </ChatSection>
  )
}
```