- ➔ GitHub: https://github.com/danielnashed/yann_fan
- ➔ Website: https://yann-fan.vercel.app/

YANN FAN

A RAG App for Yann LeCun Fans

TABLE OF CONTENTS

PROBLEM STATEMENT

Build a RAG Chatbot App

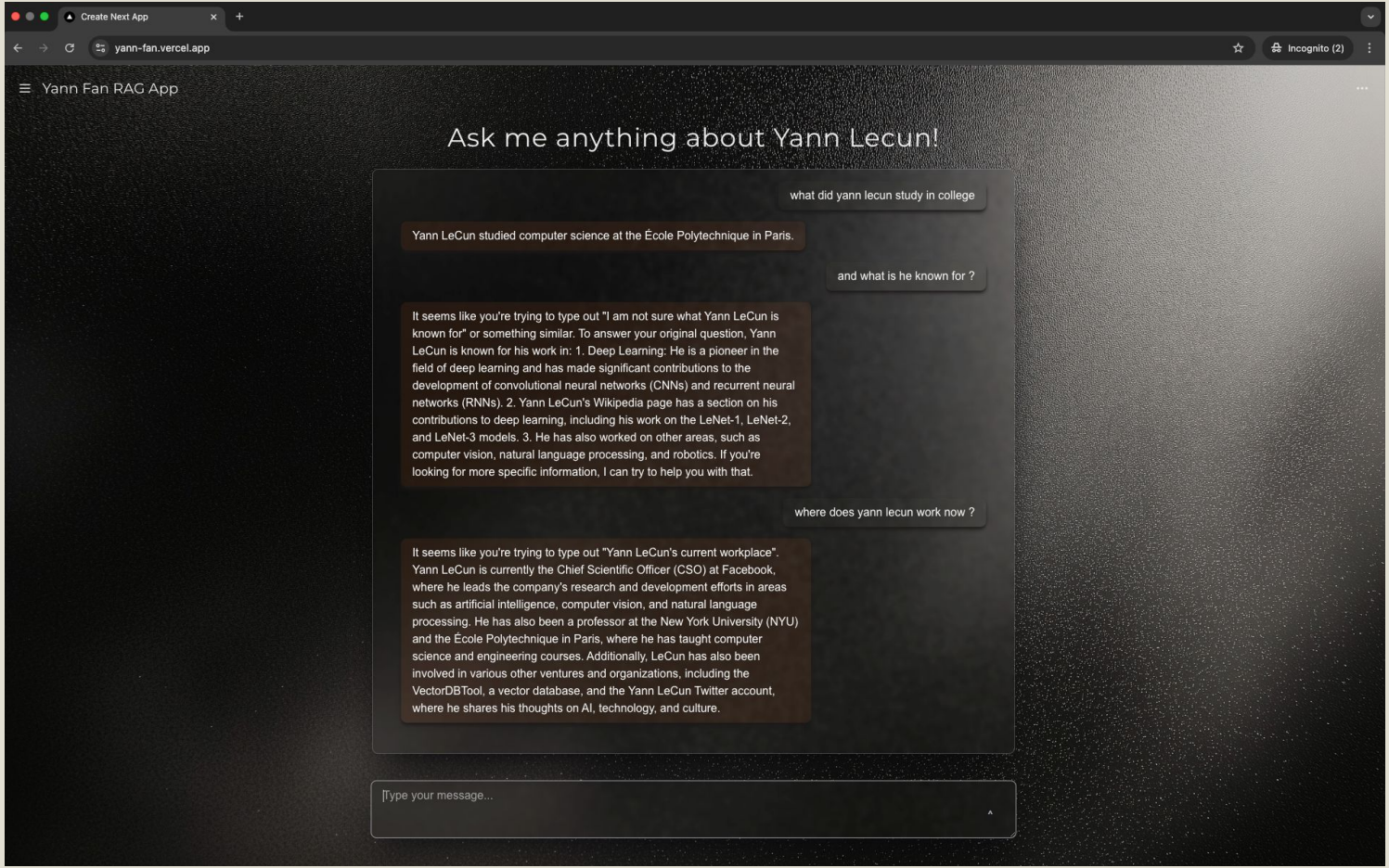
- ➔ Users upload documents about Yann LeCun (PDFs or DOCs or images)
- ➔ Users can query uploaded documents through an LLM chatbot interface

GOALS

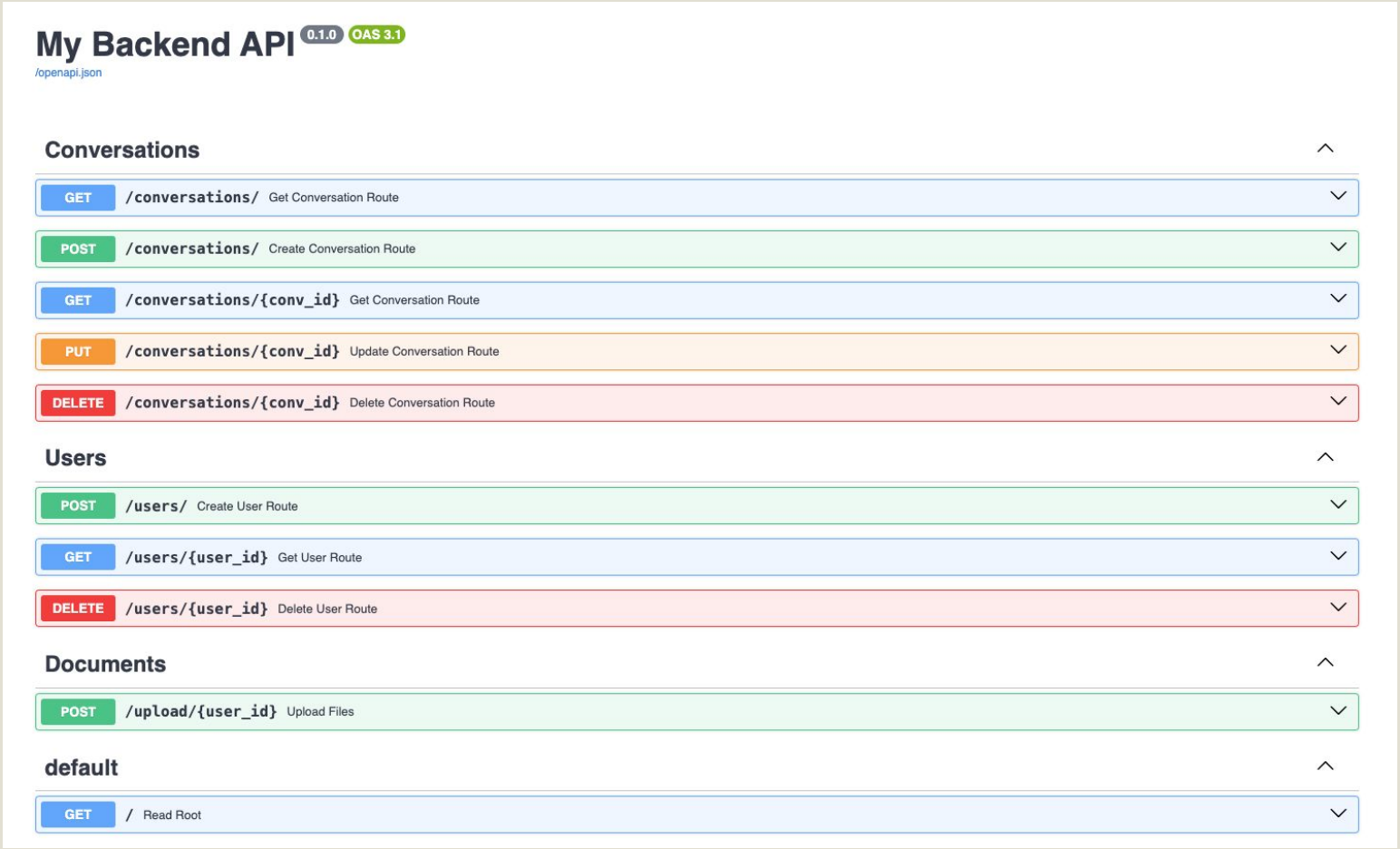
- Build a web app with chatbot interface that lets users ask questions about Yann LeCun
- Stretch goals:
  - Slick UI/UX
  - Build agentic RAG
  - Deploy app on cloud
  - Multi-modal embeddings
  - Try to minimize cost (don't use OpenAI)

DEMO

Website: https://yann-fan.vercel.app/

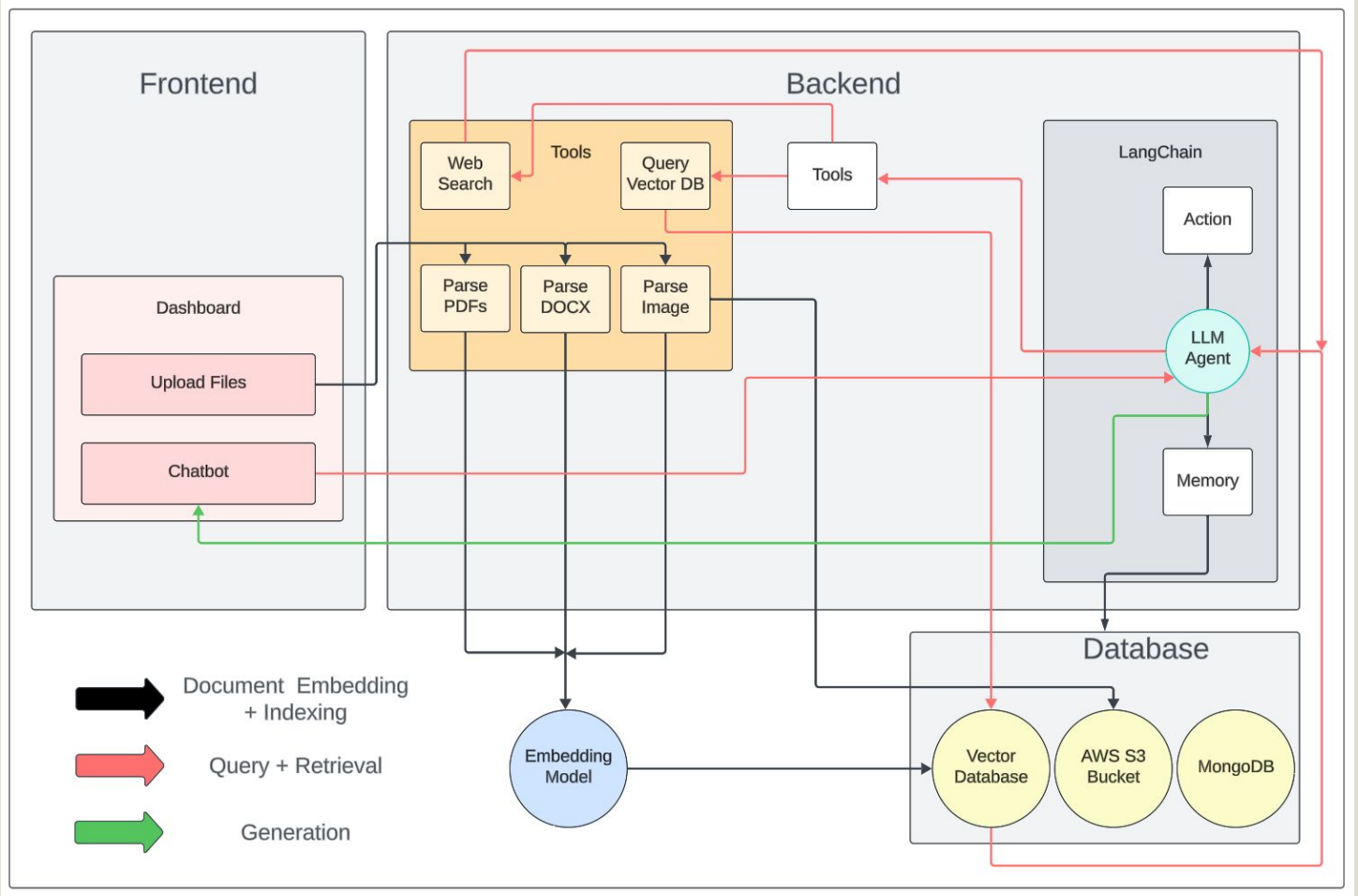


DEMO


My Backend API
0.1.0
OAS 3.1
/openapi.json
Conversations
GET /conversations/ Get Conversation Route
POST /conversations/ Create Conversation Route
GET /conversations/{conv_id} Get Conversation Route
PUT /conversations/{conv_id} Update Conversation Route
DELETE /conversations/{conv_id} Delete Conversation Route
Users
POST /users/ Create User Route
GET /users/{user_id} Get User Route
DELETE /users/{user_id} Delete User Route
Documents
POST /upload/{user_id} Upload Files
default
GET / Read Root

SYSTEM OVERVIEW

SYSTEM DIAGRAM



TECH STACK

- Backend
  - FastAPI for web framework
    - Very **performant** as it is built on top of PyDantic so suitable for **high API throughput** (streaming multiple large file uploads)
    - **Async** operations ideal for RAG apps to access multiple databases + APIs concurrently
    - Automatically generates **OpenAPI + Swagger docs** for API
  - MongoDB for NoSQL database
    - JSON structure ideal to store vector embeddings along with metadata
  - Pinecone for vector database
    - Efficient Approximate Nearest Neighbor search for **fast similarity search**
    - **Scales horizontally** across billions of vectors
  - AWS S3 for cloud object storage (like PDFs, Images uploaded by user)
  - AWS Lambda for serverless function
    - Deploy backend in Lambda to minimize cost (only pay for CPU + memory when function executes)
  - AWS ECR for storing docker images
  - AWS API Gateway for passing API requests through to Lambda function
  - AWS CloudFormation for **automating provisioning** of AWS resources

TECH STACK

- Frontend
  - Next.js for frontend framework
    - More **production-ready** than using React alone
    - Built-in server-side rendering (**SSR**) so more responsive
    - **SEO** optimization so better for search engine rankings
  - TailwindCSS to design directly in HTML (no need for CSS files)
  - daisyUI to recycle existing modern UI components
- LLM
  - Groq-llama-3.2-1b for LLM API calls
    - **Large context window** - 128k tokens
    - Very **fast inference** (3100 tokens/s) - runs on LPU not GPU
    - **Cheaper than OpenAI** ($0.08/1M tokens vs $0.75/1M tokens for chatgpt-4o-mini)
    - Model card on HuggingFace: https://huggingface.co/meta-llama/Llama-3.2-1B
  - LangChain for **agentic** framework - many tool integrations + allows easy setup of agentic workflows
  - Jina.ai for **multi-modal** vector embedding model (jina-clip-v2)
    - Offers both text + image embedding models with free 1M tokens

FUTURE IMPROVEMENTS

FUTURE IMPROVEMENT

- ➔ **Integrate More Tools**: integrate more tools from LangChain like Arxiv for papers, Wikipedia for articles, X.com API to access Yann Lecun tweets
- ➔ **Retrieval Re-ranking**: After LLM retrieves top K results from vector database, use another LLM to rerank results based on scoring relevancy
- ➔ **Hybrid Search**: implement keyword-based search along with similarity search to improve accuracy of retrieval
- ➔ **Refine Prompt**: ask another LLM to tweak user prompt
- ➔ **Batch or Real-time Streaming**: automate streaming of relevant documents like newly published papers, new LinkedIn posts, new tweets instead of relying on user to upload data
- ➔ **Context Management**: implement memory model to save most important tokens from a chat history for future interactions with user
- ➔ **VM instead of Serverless:** deploying on EC2 means no cold start time as seen with the Lambda deployment