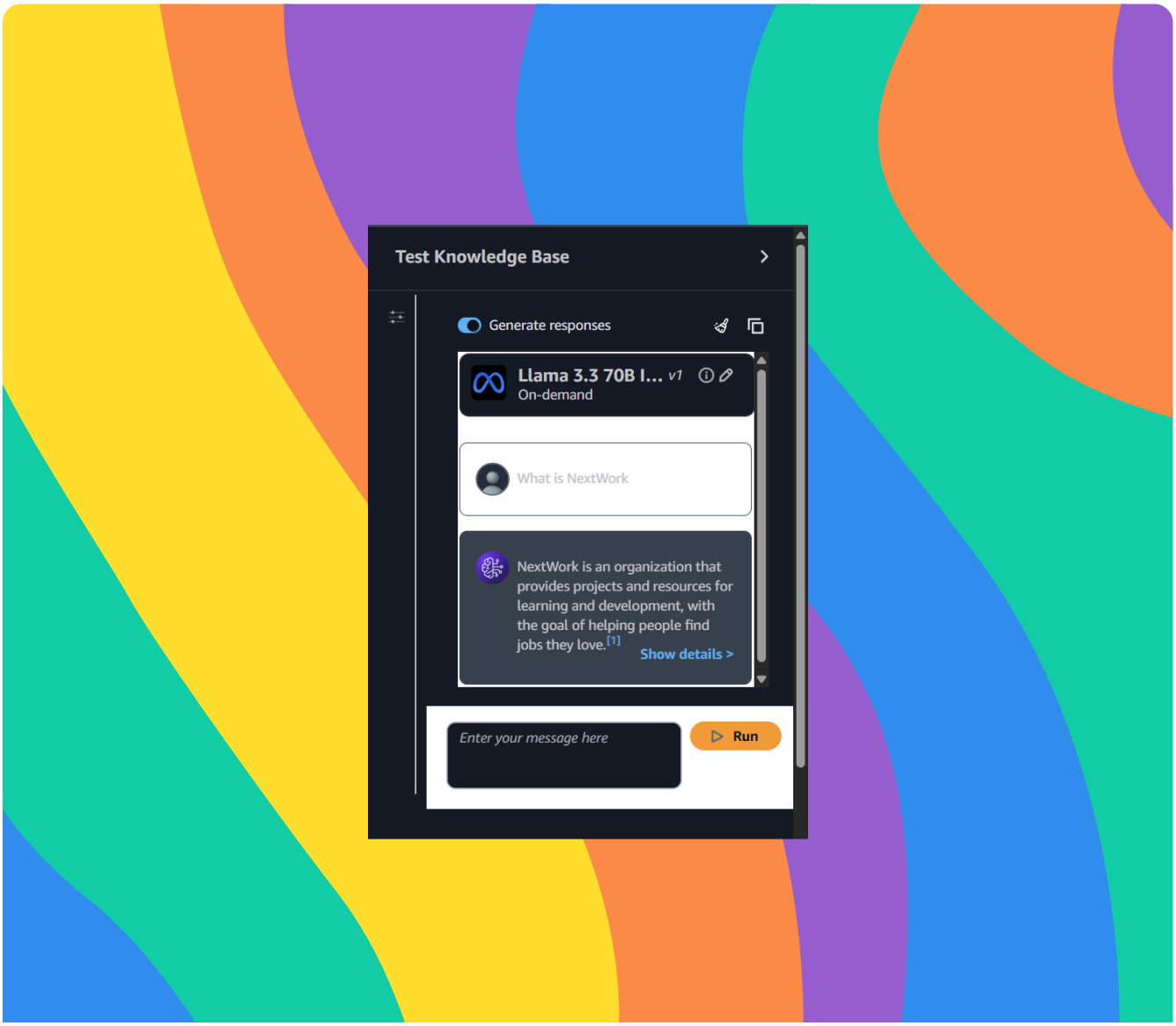
nextwork.org

Set Up a RAG Chatbot in Bedrock

jyothesh karnam



jyothesh karnam
NextWork Student

Introducing Today's Project!

RAG (Retrieval Augmented Generation) is an AI techniquie that lets you train an AI model on your own personal documents. In this project, I will demonstrate RAG by setting up a RAG chatbot in Amazon Bedrock.

Tools and concepts

Services I used in this project were Amazon Bedrock, S3, and OpenSearch Serverless. Key concepts I learnt include knowledge bases, requesting access to AI models, how chatbots generate responses (i.e. AI models + knowledge base), vector stores.

Project reflection

This project took me approximately two hours including project demo time. The most challenging part was the error with my AI models (and understanding on-demand vs pre-provisioned inference). It was most rewarding to level up my chatbot's responses!

I did this project today to learn more about Bedrock and RAG. This project definitely met my goals – awesome to learn both over a hands-on project!

jyothesh karnam
NextWork Student

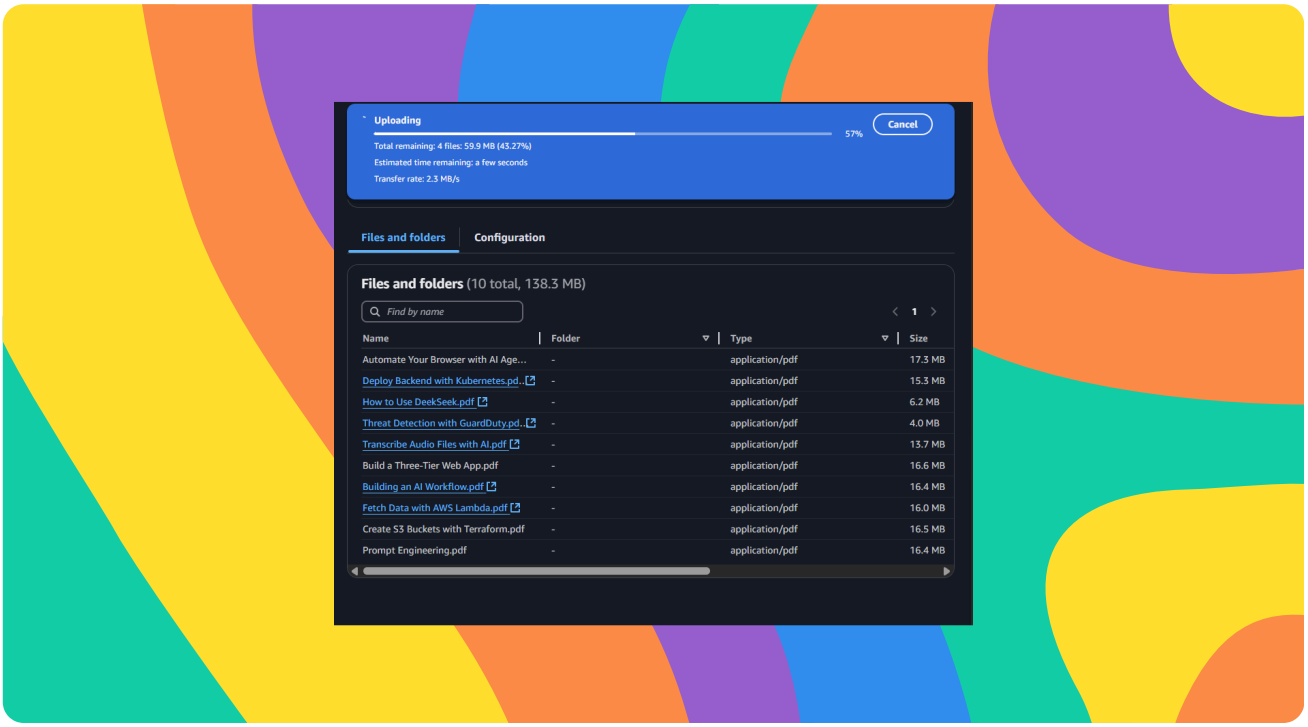
Understanding Amazon Bedrock

Amazon Bedrock is an AWS service that makes it easy to build generative AI applications. This is because Bedrock is like an AI model marketplace that lets us find, use an test models from different providers in a central place. I am using Bedrock in this project to create a knowledge base.

My Knowledge Base is connected to S3 because S3 is going to be the storage/source for my Knowledge Base's raw documents. S3 is AWS's storage service, where you can store all kinds of objects (e.g. videos, documents, audio) in the same bucket.

In an S3 bucket, I uploaded the documents that will make up my chatbot's knowledge. My S3 bucket is in the same region as my Knowledge Base because Bedrock is a regional service – data must live in the same region as the Bedrock resource (Kbase).

jyothesh karnam
NextWork Student



jyothesh karnam
NextWork Student

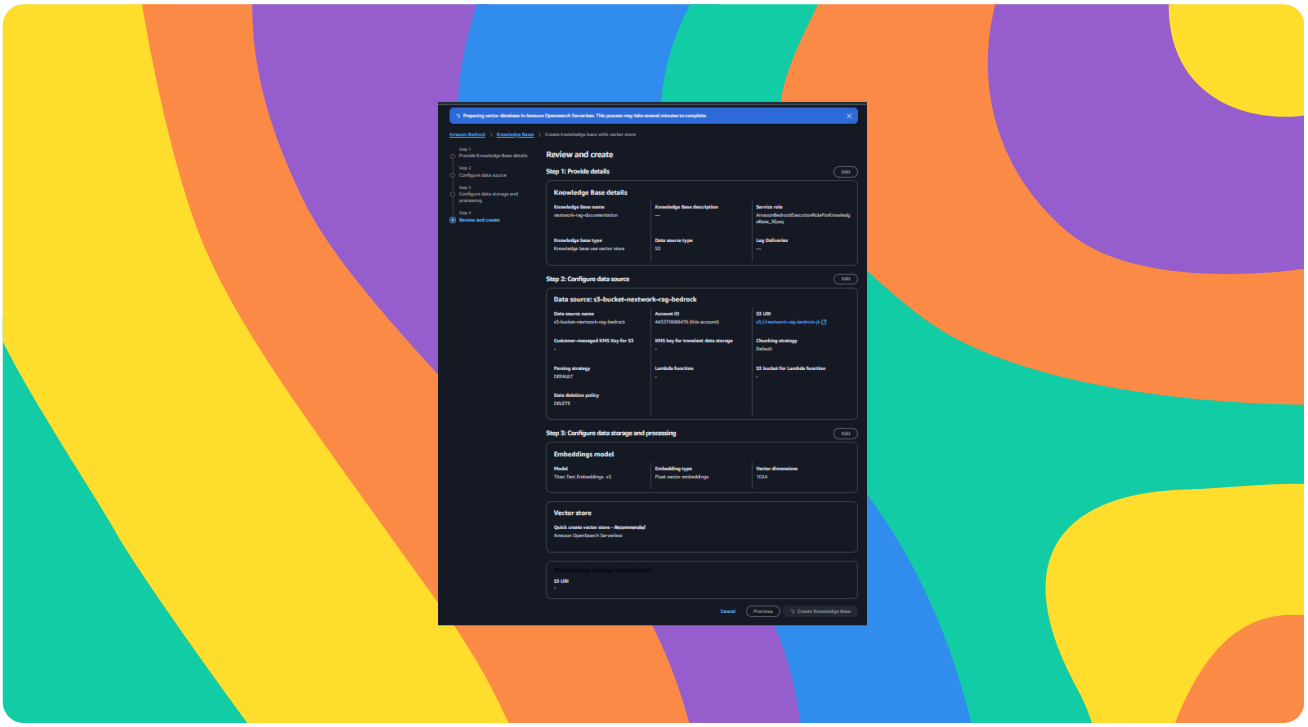
My Knowledge Base Setup

My Knowledge Base uses a vector store, which means a search engine/database that stores data based on their semantic meaning! When I query my Knowledge Base, OpenSearch will find the relevant chunks of data to the query, and pass it to Bedrock.

Embeddings are vector representations of the semantic meaning of a chunk of text. The embedding model I'm using is Titan Text Embeddings v2 because it's fast, accurate and a lot more affordable!

Chunking is the process of splitting large documents into smaller, manageable pieces called chunks, which makes it easier for the model to retrieve and understand relevant information during a query. In my Knowledge Base, chunks are set to a specific size and overlap value to ensure context is preserved across splits, improving the accuracy of responses.

jyothesh karnam
NextWork Student

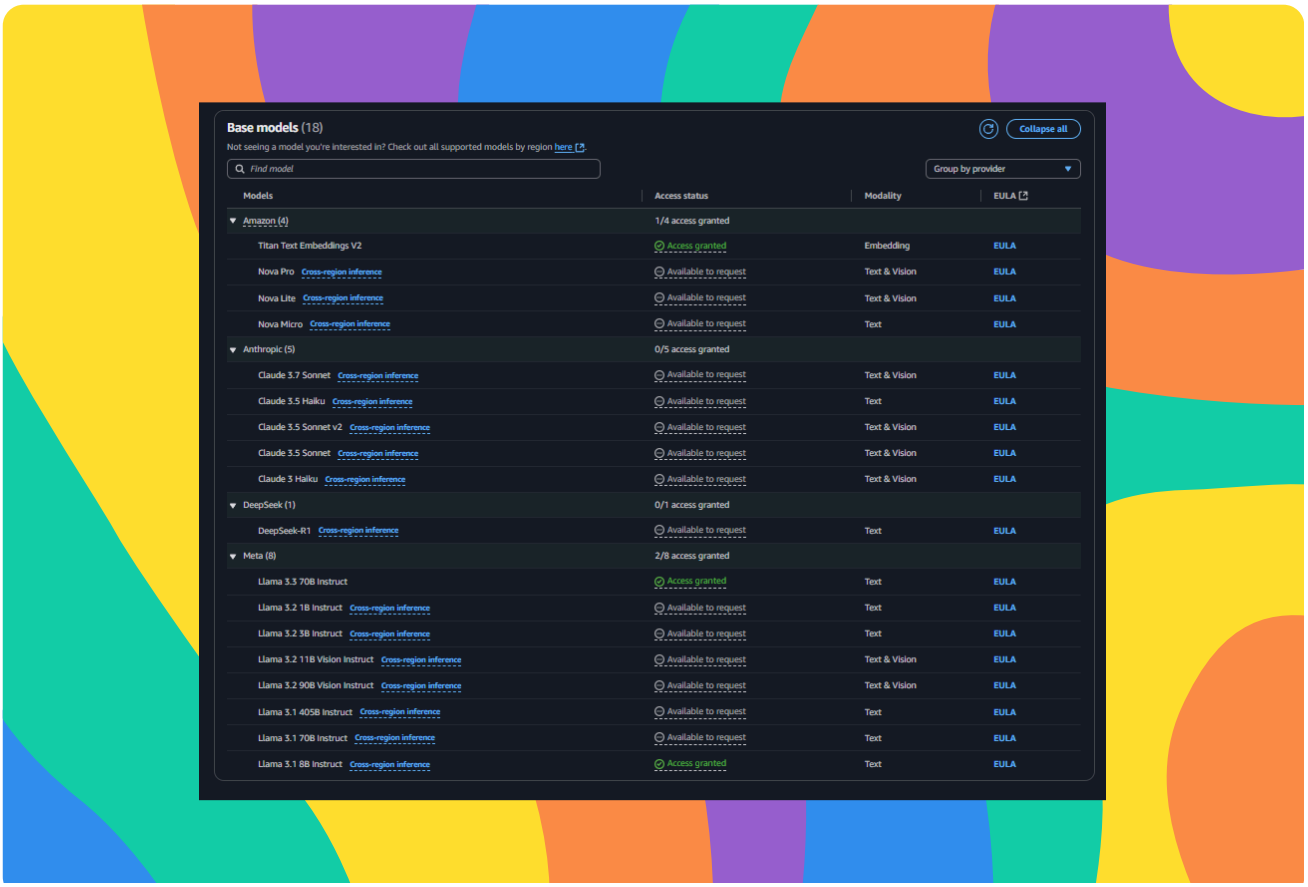


jyothesh karnam
NextWork Student

AI Models

AI models are important for my chatbot because they're the translator of my Knowledge Base's search results into human-like text. Without AI models, my chatbot would only respond with chunks of text from my documents – which isn't the best experience!

To access an AI model in Bedrock, I had to visit the "Model Access" page and request access explicitly! AWS needs explicit access because some AI model providers have extra forms/rules if you wanted to use them, and AWS needs to check availability.

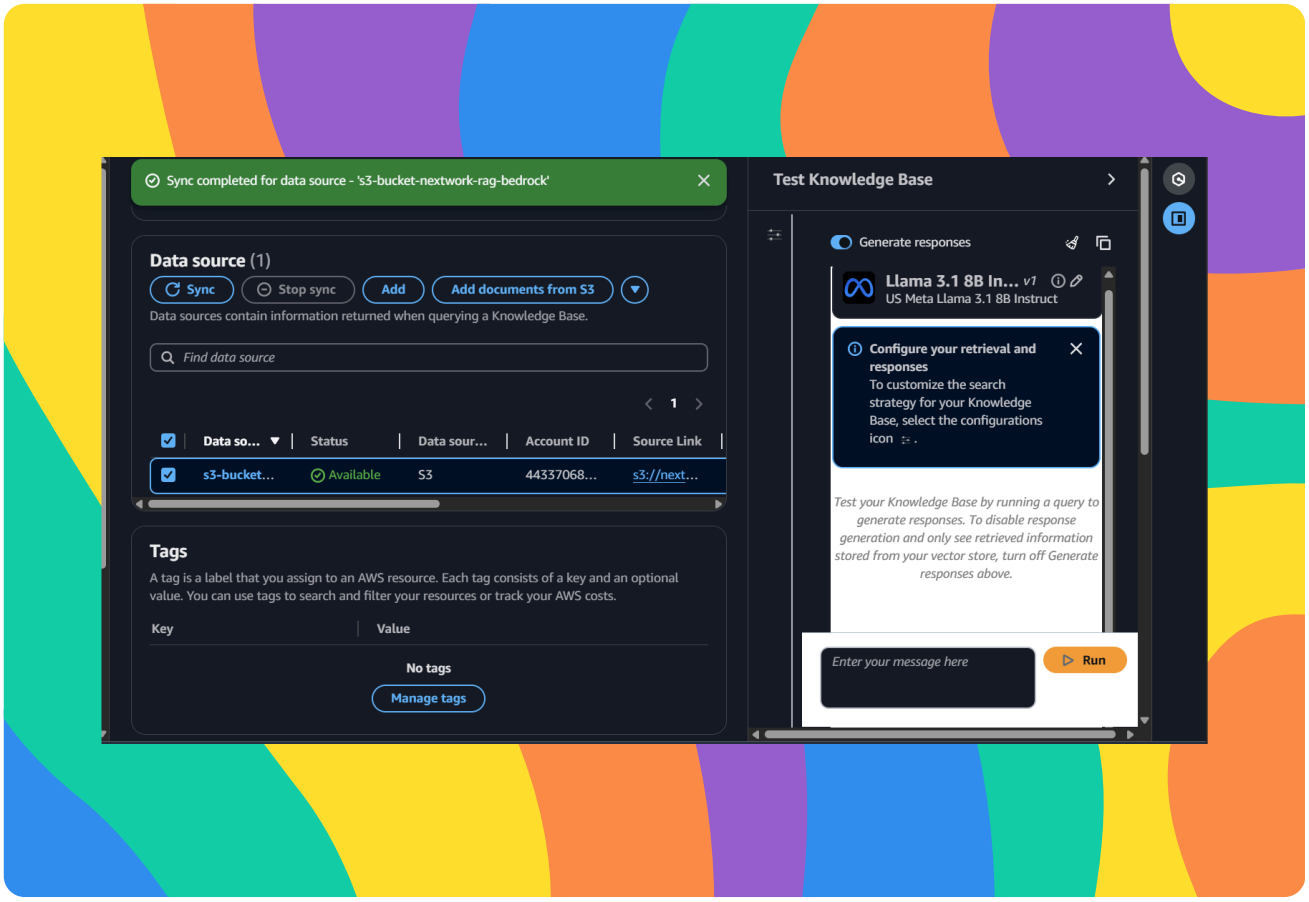


jyothesh karnam
NextWork Student

Syncing the Knowledge Base

Even though I've already connected my S3 bucket when creating the Knowledge Base, I still need to sync my data, which will actually move the data from S3 and into my Knowledge Base + OpenSearch Serverless. Currently, the Knowledge Base is empty!

The sync process involves three key steps: Ingesting (i.e. Bedrock takes the data from S3), Processing (i.e. Bedrock chunks and embeds the data) and Storing (i.e. Bedrock stores the processed data in my vector store, OpenSearch Serverless).



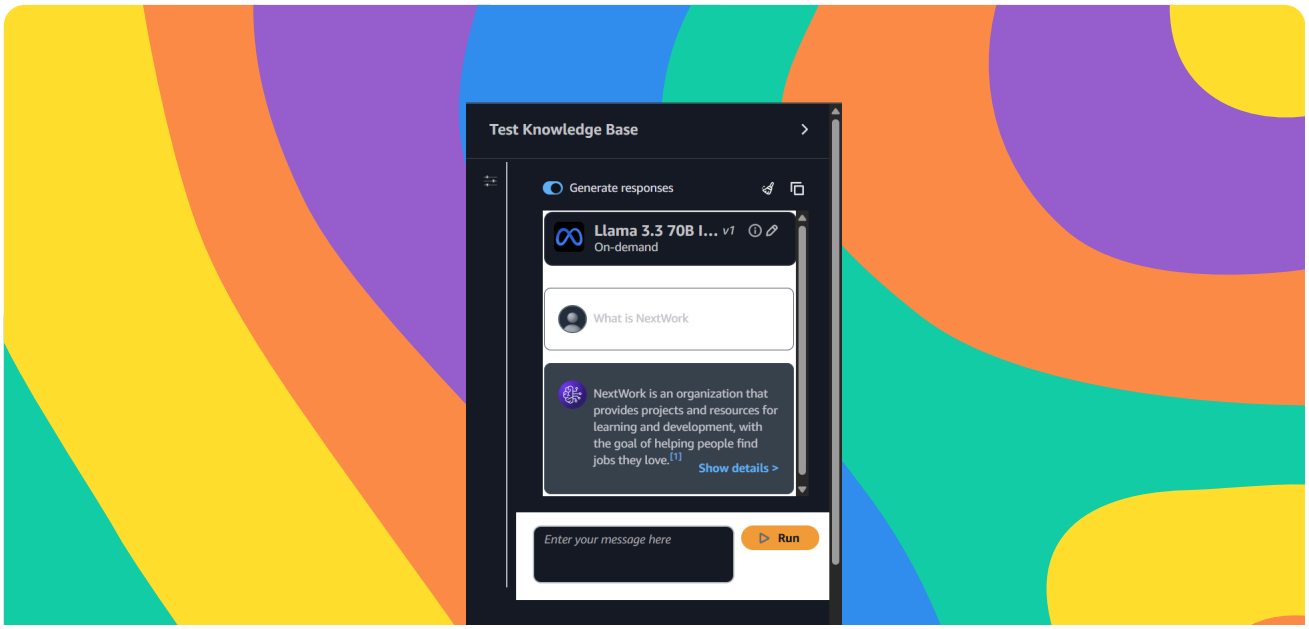
jyothesh karnam
NextWork Student

Testing My Chatbot

initially tried to test my chatbot using Llama 3.1 8B as the AI model, but it triggered an error – it was not available on demand. I had to switch to Llama 3.3 70B because it was offered on-demand by AWS (since it's a newer, efficient model).

When I asked about topics unrelated to my Knowledge Base's data, my chatbot responds that it cannot help me with this request. This proves that the chatbot only knows the information that I gave it – it won't know even common knowledge outside.

I can also turn off the Generate Responses setting to see the raw chunks of data directly from my Knowledge Base. When I tested this, my chatbot gave a list of 5 paragraphs to answer a question, whereas the AI model will convert it to a sentence.



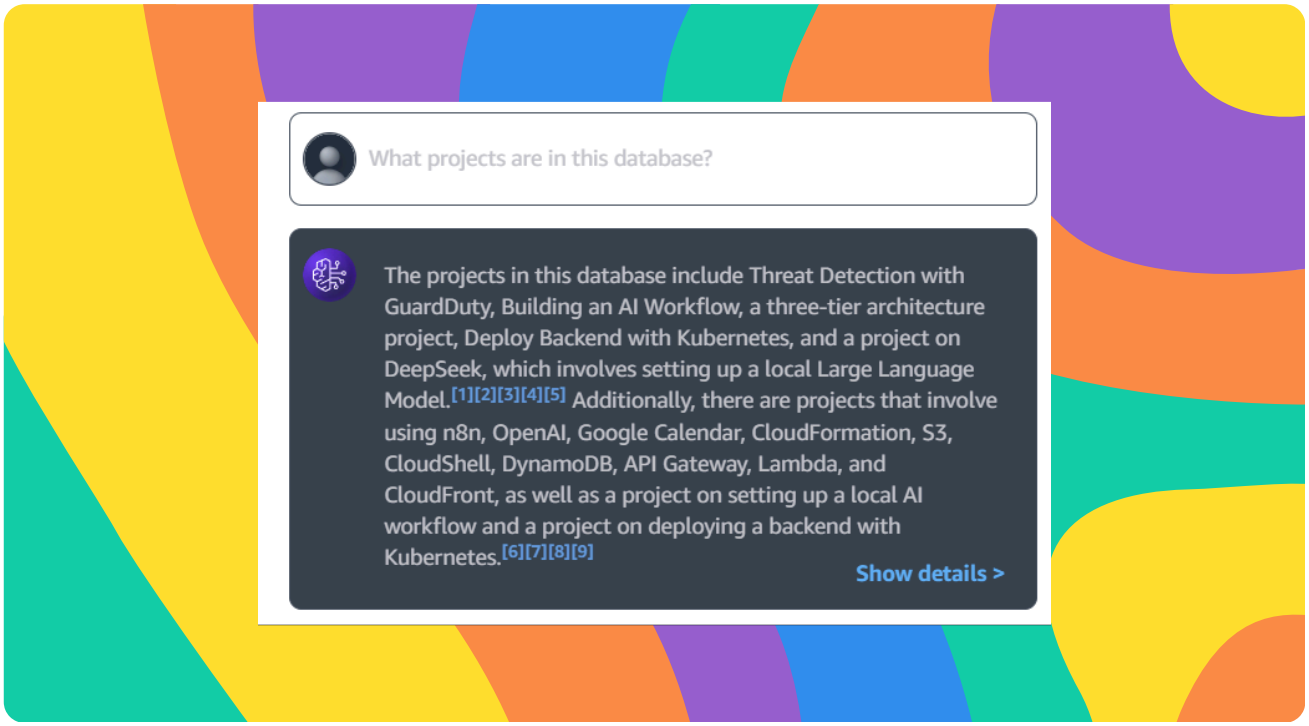
jyothesh karnam
NextWork Student

Upgrading My Chatbot

In a project extension, I looked into increasing the number of source chunks, which are the number of chunks of text that the Knowledge Base will return for a query. This will improve my chatbot's responses by giving my AI model more data to use.

I also added a custom prompt that tells my chatbot to always respond by mentioning skills that the student's learnt, even if the student didn't ask for it directly. I also gave context that the database contains documentation for projects.

After adjusting the source chunks and the generation prompt, my chatbot's response became so much more detailed! The chatbot would talk about skills I've learnt, and used 10 or more examples within the response.



NextWork.org

Everyone should be in a job they love.

Check out nextwork.org for more projects