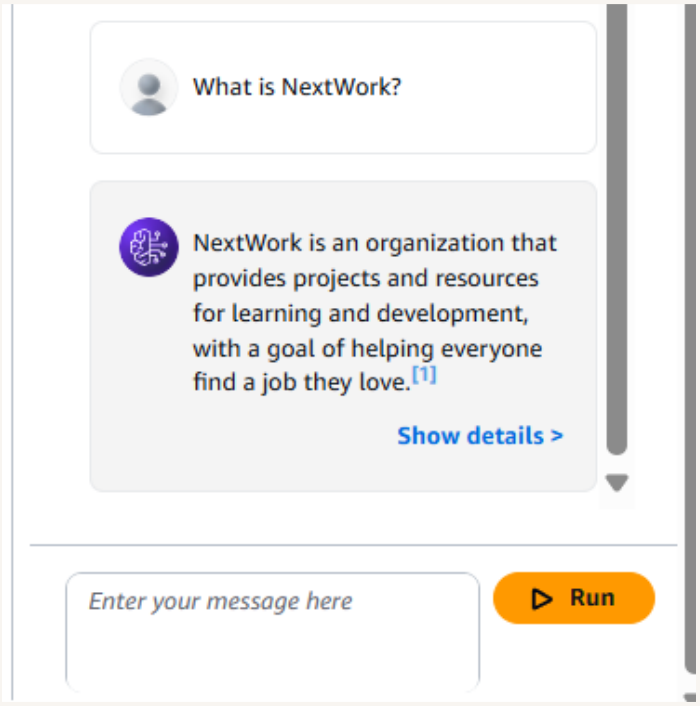
NextWork.org

RAG-based Chatbot with Bedrock

Dineshraj Dhanapathy



DI
Dineshraj Dhanapathy
NextWork Student

Introducing Today's Project!

RAG (Retrieval Augmented Generation) is a method that combines retrieved data with AI to produce informed answers. In this project, I will demonstrate RAG by merging live search results with AI output.

Tools and concepts

Services I used were Amazon Bedrock, S3, and OpenSearch Serverless. Key concepts I learnt include chunking, embeddings, vector stores, and the importance of syncing data for effective AI-powered

Project reflection

This project took me approximately 2 hours. The most challenging part was setting up the Knowledge Base and troubleshooting model responses. It was most rewarding to see the chatbot provide accurate, real-time answers.

I did this project today to deepen my understanding of AI models and enhance my skills in integrating knowledge bases. The project met my goals by successfully building a functional, responsive AI chatbot.

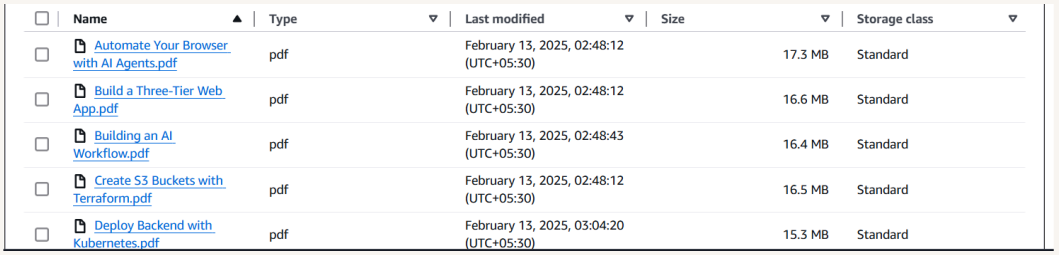
DI
Dineshraj Dhanapathy
NextWork Student

Understanding Amazon Bedrock

Amazon Bedrock is a managed service offering scalable foundation models and tools to build generative AI applications. I'm using Bedrock in this project to integrate and deploy AI-powered solutions.

My Knowledge Base is connected to S3 because S3 is a secure, scalable, and durable storage solution that hosts our data and enables efficient retrieval and processing for AI models for fast access.

In an S3 bucket, I uploaded documents including FAQs, manuals, and guides for our chatbot's Knowledge Base. My S3 bucket is in the same region as my KB because I'm using Ohio East 2 for low latency



Name	Type	Last modified	Size	Storage class
Automate Your Browser with AI Agents.pdf	pdf	February 13, 2025, 02:48:12 (UTC+05:30)	17.3 MB	Standard
Build a Three-Tier Web App.pdf	pdf	February 13, 2025, 02:48:12 (UTC+05:30)	16.6 MB	Standard
Building an AI Workflow.pdf	pdf	February 13, 2025, 02:48:43 (UTC+05:30)	16.4 MB	Standard
Create S3 Buckets with Terraform.pdf	pdf	February 13, 2025, 02:48:12 (UTC+05:30)	16.5 MB	Standard
Deploy Backend with Kubernetes.pdf	pdf	February 13, 2025, 03:04:20 (UTC+05:30)	15.3 MB	Standard

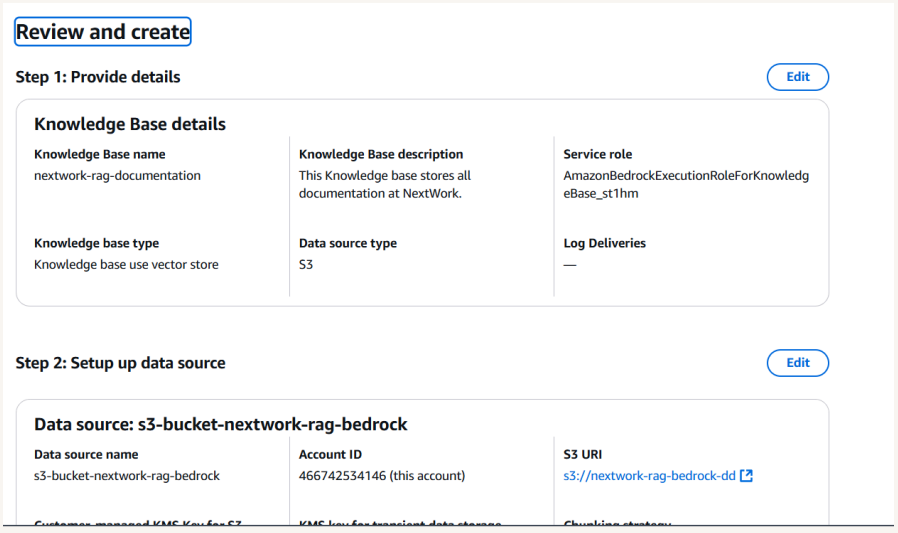
DI
Dineshraj Dhanapathy
NextWork Student

My Knowledge Base Setup

My Knowledge Base uses a vector store, which means it efficiently stores and retrieves embeddings for fast AI responses. When I query my Knowledge Base, OpenSearch will quickly find relevant document chunks

Embeddings are numerical representations of text that help AI understand context and meaning. The embedding model I'm using is Titan Text Embeddings v2 because it provides accurate, efficient semantic search.

Chunking is the process of breaking large text into smaller parts for efficient AI processing. In my Knowledge Base, chunks are set to 300 tokens, ensuring the chatbot retrieves precise and relevant info.



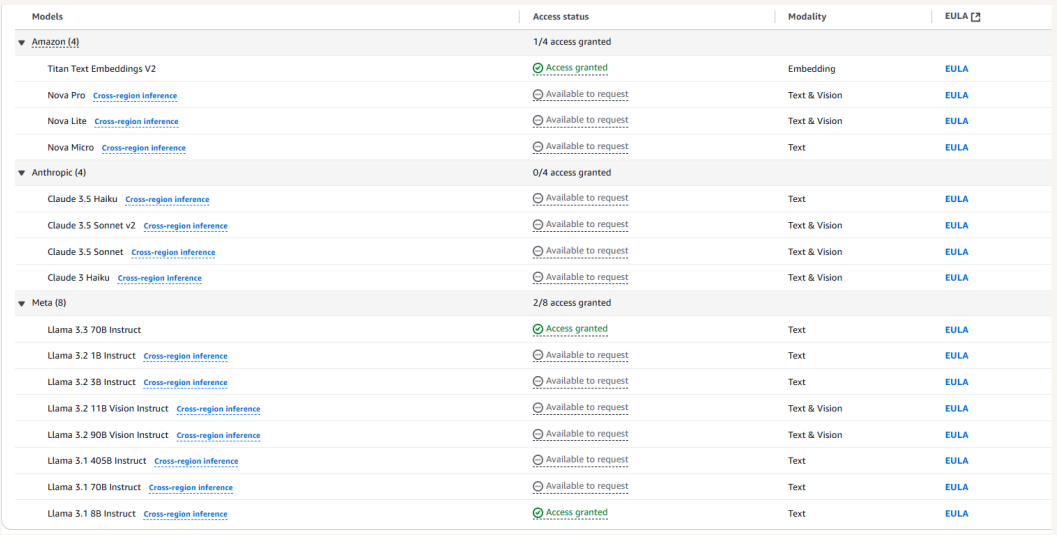
DI

Dineshraj Dhanapathy
NextWork Student

AI Models

AI models are important for my chatbot because they generate intelligent, context-aware responses. Without AI models, my chatbot would only retrieve static text without understanding user intent.

To get access to AI models in Bedrock, I had to request model access in the AWS console. AWS needs explicit access because it ensures compliance, security, and proper usage of foundation models.



Models	Access status	Modality	EULA
Amazon (4)	1/4 access granted		
Titan Text Embeddings V2	Access granted	Embedding	EULA
Nova Pro Cross-region inference	Available to request	Text & Vision	EULA
Nova Lite Cross-region inference	Available to request	Text & Vision	EULA
Nova Micro Cross-region inference	Available to request	Text	EULA
Anthropic (4)	0/4 access granted		
Claude 3.5 Haiku Cross-region inference	Available to request	Text	EULA
Claude 3.5 Sonnet v2 Cross-region inference	Available to request	Text & Vision	EULA
Claude 3.5 Sonnet Cross-region inference	Available to request	Text & Vision	EULA
Claude 3 Haiku Cross-region inference	Available to request	Text & Vision	EULA
Meta (8)	2/8 access granted		
Llama 3.3 70B Instruct	Access granted	Text	EULA
Llama 3.2 1B Instruct Cross-region inference	Available to request	Text	EULA
Llama 3.2 3B Instruct Cross-region inference	Available to request	Text	EULA
Llama 3.2 11B Vision Instruct Cross-region inference	Available to request	Text & Vision	EULA
Llama 3.2 90B Vision Instruct Cross-region inference	Available to request	Text & Vision	EULA
Llama 3.1 405B Instruct Cross-region inference	Available to request	Text	EULA
Llama 3.1 70B Instruct Cross-region inference	Available to request	Text	EULA
Llama 3.1 8B Instruct Cross-region inference	Access granted	Text	EULA

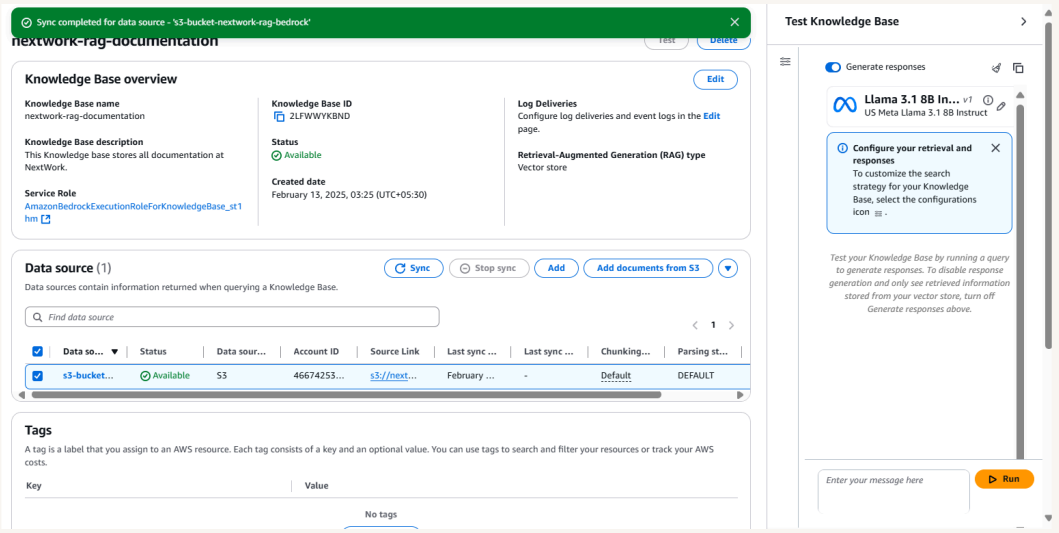
DI

Dineshraj Dhanapathy
NextWork Student

Syncing the Knowledge Base

Even though I already connected my S3 bucket when creating the Knowledge Base, I still need to sync because it ensures the latest data is indexed, updated, and ready for accurate retrieval by the AI.

The sync process involves three steps: Ingesting, where Bedrock retrieves data from S3; Processing, where it chunks and embeds the data; and Storing, where Bedrock stores the processed data in OpenSearch Serverless.



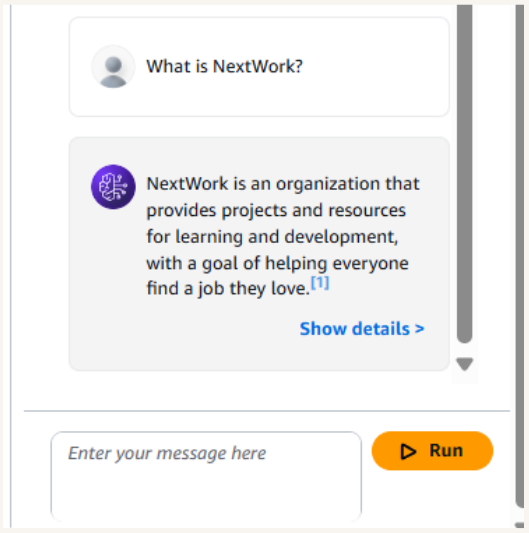
DI
Dineshraj Dhanapathy
NextWork Student

Testing My Chatbot

I initially tried to test my chatbot using Llama 3.1 8B as the AI model, but it lacked the depth needed for complex queries. I had to switch to Llama 3.3 70B because it provides more accurate, nuanced responses.

When I asked about topics unrelated to my data, my chatbot struggled to provide relevant answers. This proves that the chatbot relies on the knowledge base, and without related data, its accuracy drops.

You can also turn off the Generate Responses setting to prevent the AI from automatically generating answers. This allows you to manually adjust or review the responses before they are delivered, giving more control over the interaction.



NextWork.org

Everyone should be in a job they love.

Check out nextwork.org for more projects