Cloud Kinetics – Cloud Engineer Intern Assessment

Retrieval-Augmented Generation Chatbot Implementation

Student: Do Vu Bao Phuc

University: Swinburne Vietnam

Student ID: SWH00946

Submission Date: April 3rd, 2025

GitHub Repository: itsyoboygin/Cloud_Kinetics_Intern_Assessment

1. Assessment Overview

This assessment is about implementing a Retrieval-Augmented Generation (RAG) Chatbot using AWS, Bedrock in particular. The objective of the project is to create a chatbot capable of answering questions on a predefined knowledge base, maintaining conversational memory, and sending both questions and responses to a Telegram group. From a technical standpoint, it should allow Markdown (.md) file uploads to Amazon S3 to update its knowledge base, users' authentication via Cognito and automated syncs with the new knowledge base when a file is uploaded.

I have done this assessment to the best of my ability, and the objectives that have been accomplished go as follows:

- Configured User Pool for users' authentication (email/phone-based sign-up) and App Clients using AWS Cognito.
- Set up AWS AppSync for managing and retrieving data for the chatbot.
- Deployed a Lambda function that extracts text from .md files and stores data in Amazon OpenSearch for retrieval (including a test case).
- Hosted a S3 Static Website integrated with AWS CloudFront for https browing.

2. AWS Cognito & AppSync Setup

For the whole project, I will be using Asia Pacific (Sydney) Region, or ap-southeast-2. With Cognito, users will be asked to fill in their email or username for authentication. After signing up, a verification mail will be sent directly from Cognito to verify their identity.

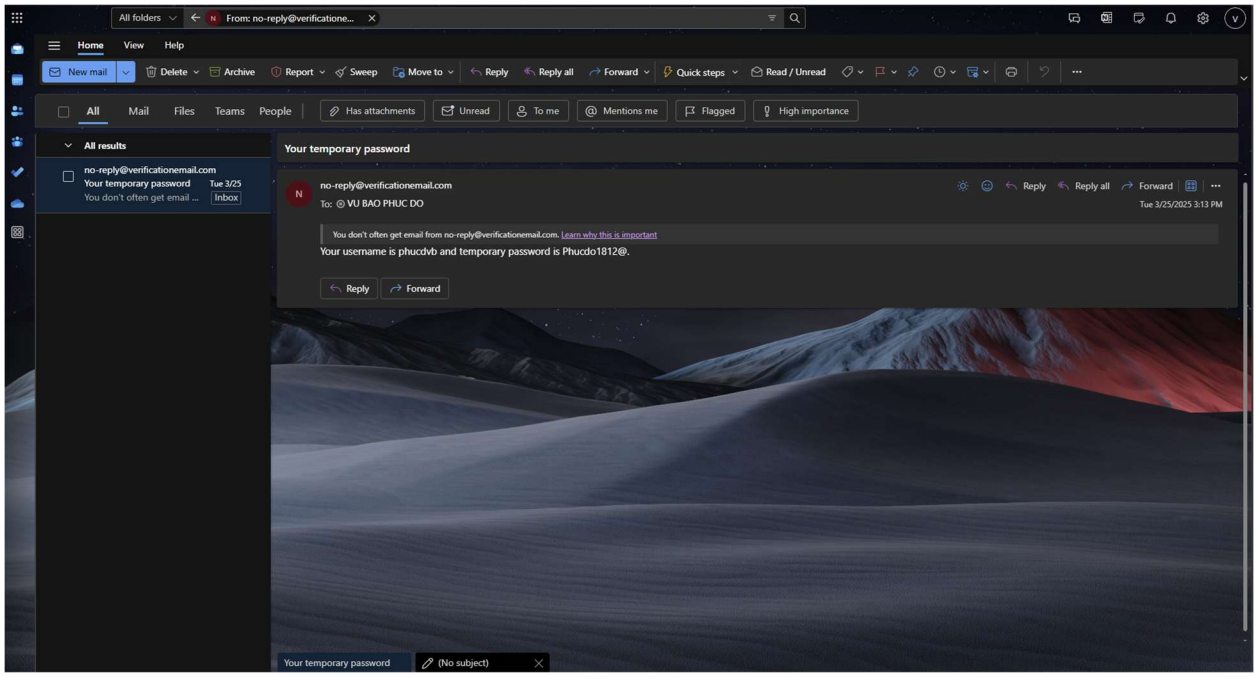

Figure 1. Verification Email

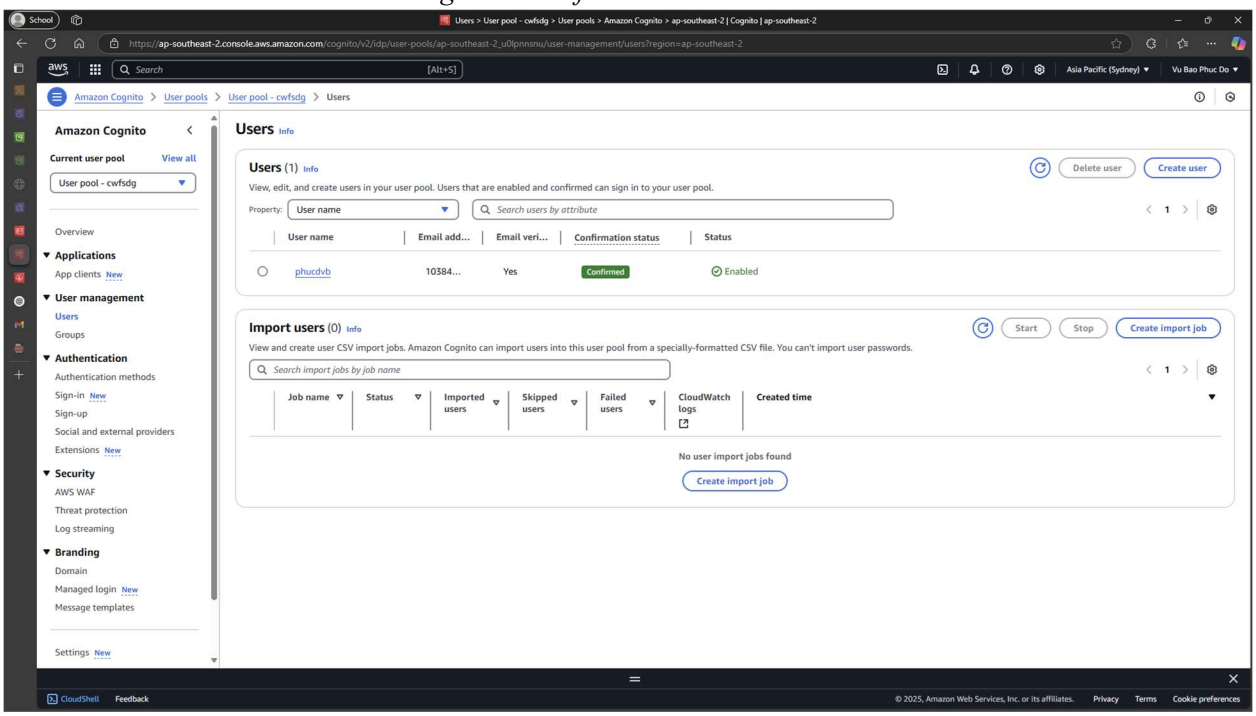

Figure 2. User Confirmed

Next, I created an App Client to generate tokens for authentication. This allows users to sign-in using their username and password.

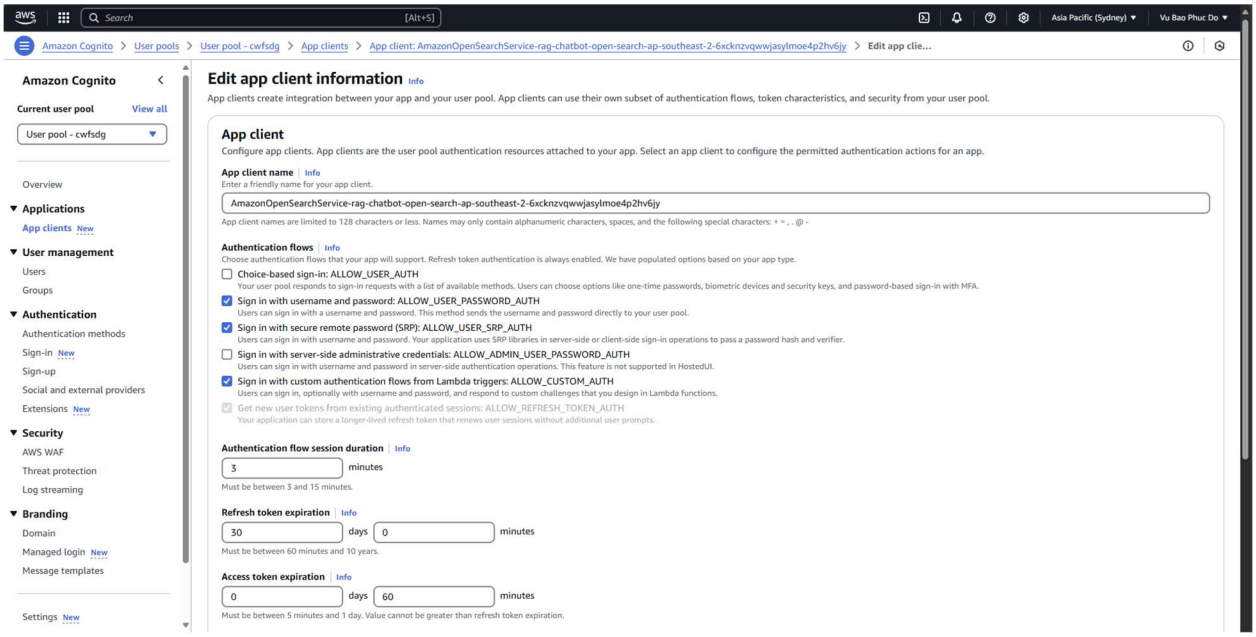

Figure 3. App Client Settings

Having done with Cognito, I integrated Cognito with an AWS AppSync API to restrict acess to authenticated users. I defined a schema that allows the creation of s new user

using their email and username. The directive @aws_cognito_user_pool indicates that only authenticated users (via AWS Cognito) can perform the mutation. For data sources, I attached a DynamoDB called Users to store users's data such as chatbot messages, personal data or any other relevant information if existed. A Lambda function was also attached to it for storing markdown files (which will be demonstrated later on).

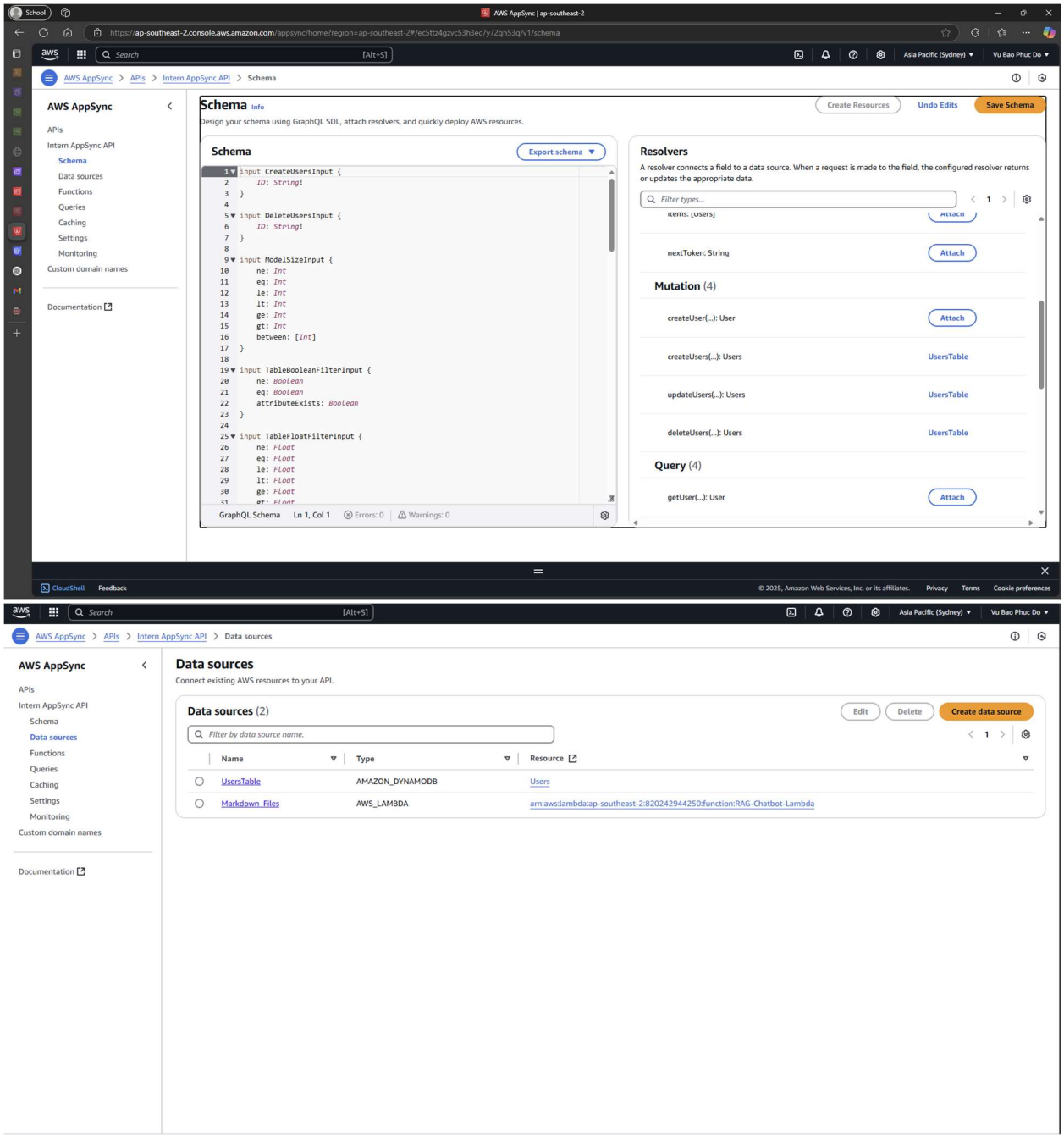

Figure 4. AWS AppSync Setup

3. Developing the Lambda Function

IAM Role

The role attached to the Lambda function has four permissions policies as shown in the figure below. AmazonOpenSearchServiceFullAccess grants full access to Amazon OpenSearch Service, AmazonS3ReadOnly allows lambda to read objects from S3 buckets, and AWSLambdaBasicExecutionRole provides permissions for writing logs to CloudWatch. The Lambda_S3_OpenSearch policy, which was manually created, defines limited write access to OpenSearch and read access to an S3 bucket, rag-chatbot-knowledge in particular. These policies serve lambda to read, process and store content from .md files to OpenSearch. Execution details is logged to CloudWatch for monitoring and debugging.

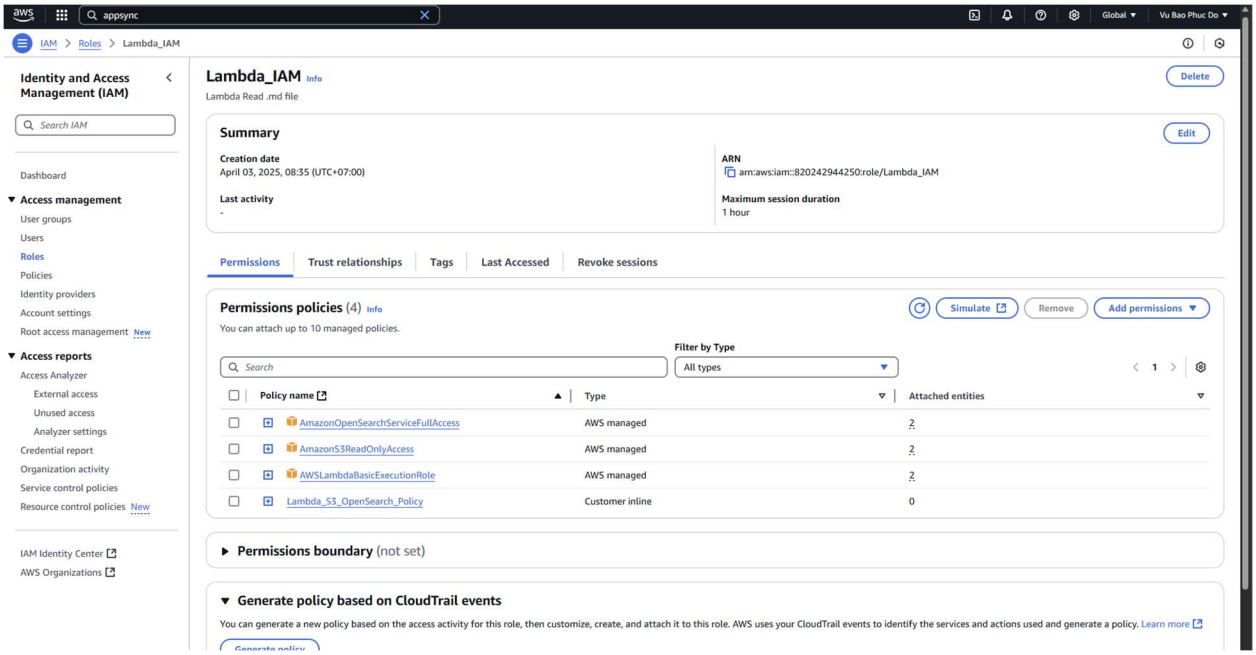

Figure 5. IAM Role for Lambda Function

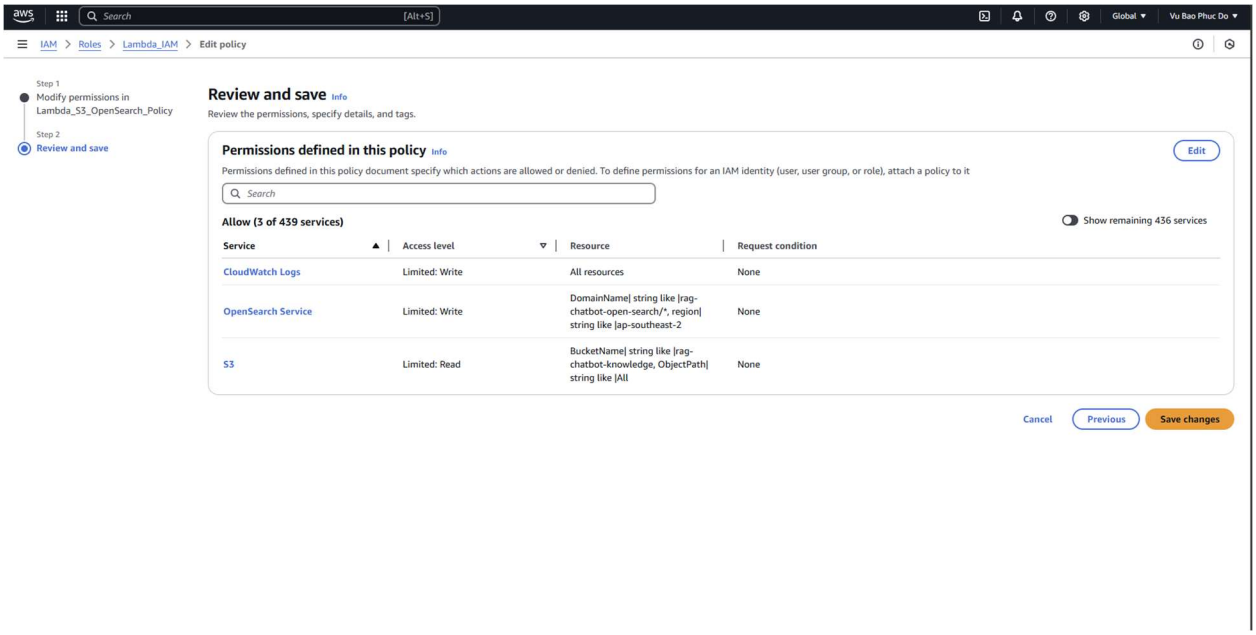

Figure 6. Policy details

RAG-Chatbot-Lambda Function

When developing this function, I used boto3 library to interact with AWS services. I aimed to make the process of reading and extracting .md files fully automated, once a new object is added to the S3 bucket triggered by Amazon S3 events. As shown in the figures below, I pushed a file called test_document.md to my S3 bucket, then executed the test event to see if it would work. Potentially, it can be extended to further process and store the extracted data in OpenSearch.

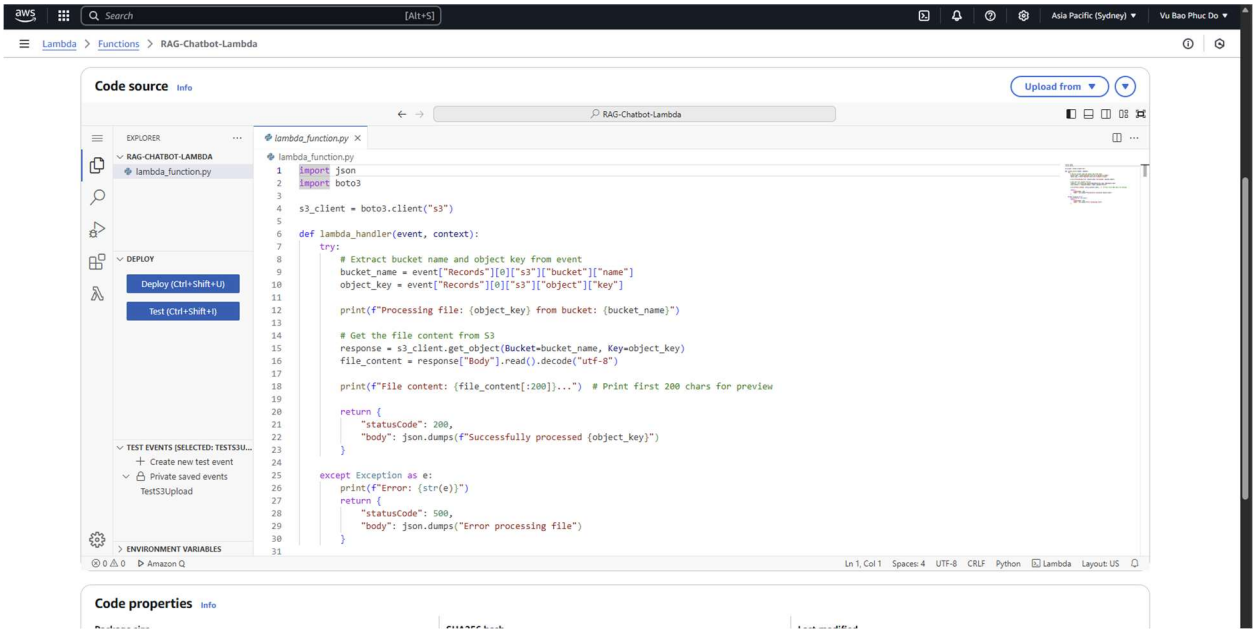

Figure 7. Lambda function coding

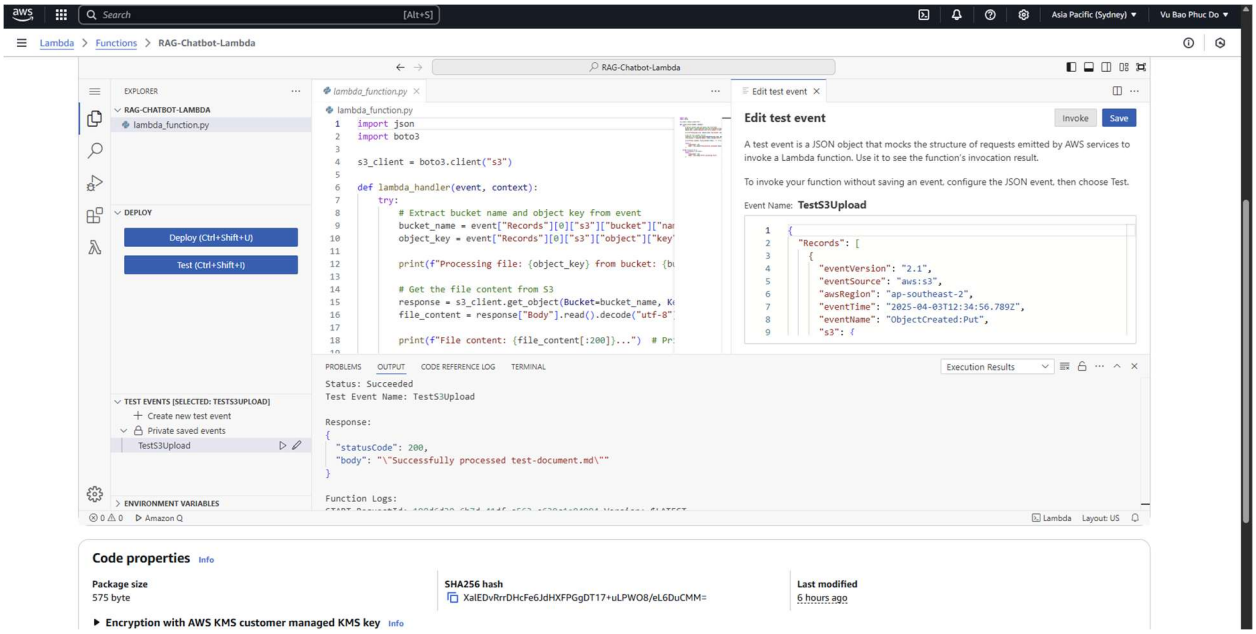

Figure 8. Event test

Figure 9. Test ouptut

4. Unresolved Issues

User Logging In

After having set up Lambda, I created a simple index page for users logging in to test if the database is connected. Amazon CloudFront was also integrated to ensure secure networking. Apparently, the database was working but the index.html file stayed stuck. The issue is that there was no authentication token being found and therefore, the system cannot redirect to OpenSearch domain.

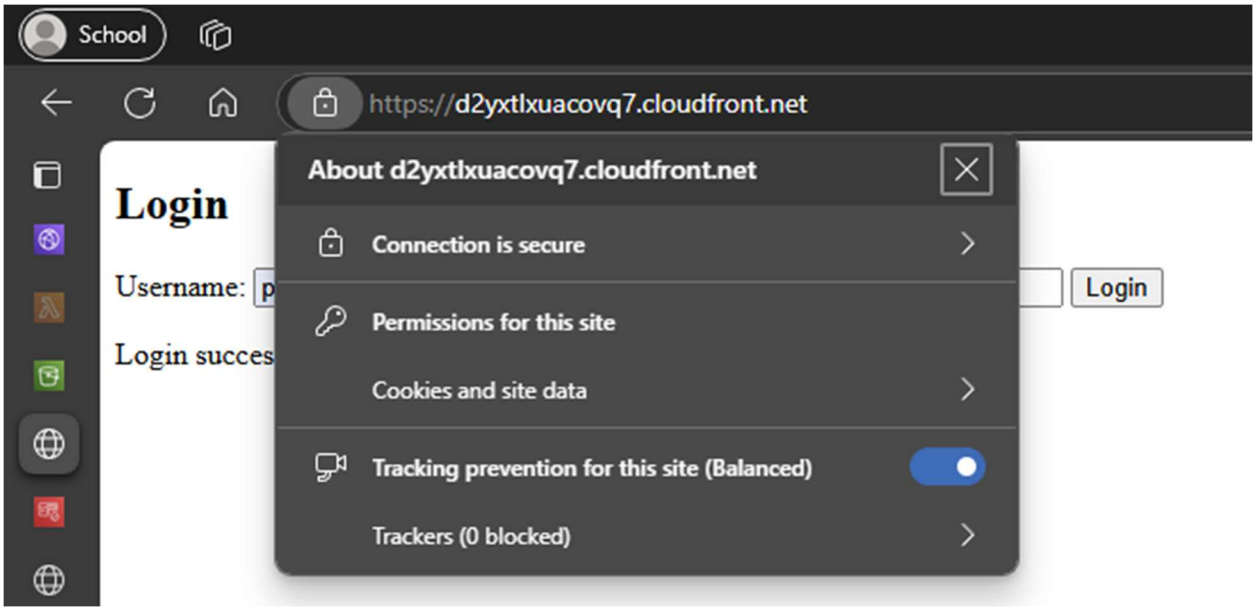

Figure 10. CloudFront's integration to S3 Hosted Static Website

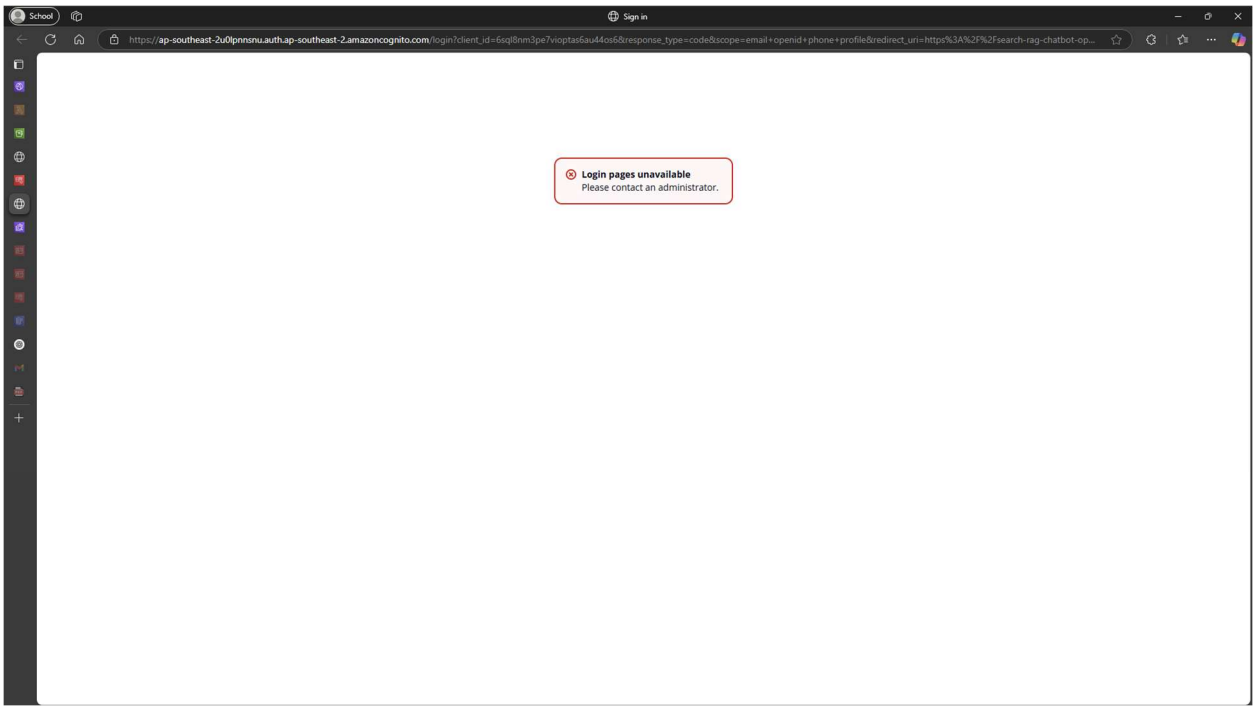

Figure 11. Failed redirection to OpenSearch domain