📄 Project Documentation: Genie API Chatbot Integration using Amplify

❓ Overview

This prototype demonstrates how to build a chatbot frontend using React, AWS Amplify (Gen 2), and Databricks Genie API. It covers backend configuration, API integration, authentication, and overcoming CORS limitations via Amplify Functions.

✔️ Step-by-Step Progress

1️⃣ Create an IAM Identity User

- Go to the **IAM Console** and create a new user with programmatic access.
- Attach a **custom policy** with the minimum permissions required for Amplify and Genie integration.

2️⃣ Required IAM Permissions

Below are the services and required permissions that need to be allowed:

Service	Access Level	Resource Scope
CloudWatch Logs	Limited: Write	LogGroupName: `/aws/amplify/*`, Region: `us-east-1`
Amplify	Limited: List	All resources
AppSync	Limited: Read, List, Write	All resources
CloudFormation	Limited: Read, List	Multiple
STS	Limited: Write	Multiple
Systems Manager	Limited: Read, List	Multiple

3️⃣ Deploy a Basic Amplify App (Todo App)

Follow the Amplify docs to deploy your first app:

📘 Reference: https://docs.amplify.aws

Deployed a simple **Todo list** application using the Amplify Data client and React frontend.

4️⃣ Add Authentication (Amazon Cognito)

Used Amplify Auth for user authentication:

- Open or create the auth file

  `amplify/auth/resource.ts`

📘 Guide: Set Up Auth with Amplify

Amplify Auth is powered by **Amazon Cognito**, which handles:

- User sign-up/sign-in
- Account recovery
- Secure token management

5️⃣ Integrated Weather API (Using Axios)

Tested API integration using `axios` in the React frontend:

- Successfully fetched real-time weather data.
- This proved external APIs can be integrated, given no CORS restriction.

6️⃣ Genie API Exploration

Studied and tested Genie APIs from Databricks:

📘 Reference: Databricks Genie API Docs

APIs used:

API	Method	Endpoint
Start Conversation	POST	`/api/2.0/genie/spaces/{space_id}/start-conversation`
Create Conversation Msg	POST	`/api/2.0/genie/spaces/{space_id}/conversations/{conversation_id}/messages`
Get Conversation Msg	GET	`/api/2.0/genie/spaces/{space_id}/conversations/{conversation_id}/messages/{message_id}`

🔑 **Note**: All APIs require an **access token from Databricks**.(Setting → Developer → Access tokens → Generate Access token)

7️⃣ Tested Genie API in Postman

- Confirmed correct request/response behavior.
- Understood payload structures.
- Verified access token usage.
- Tested end-to-end chat message flows.

8️⃣ CORS Issue with Genie API

- Databricks **does not allow:**

  ```
  Access-Control-Allow-Origin: "*"
  ```

- Therefore, **frontend cannot call Genie APIs directly**.

9️⃣ Solution: Use Amplify Function as Proxy

To overcome CORS limitations:

- Created an **Amplify Function** (AWS Lambda).
- This acts as a **proxy** between frontend and Genie API.
- It securely injects the Databricks token and forwards the request.

🔟 Amplify Function Setup (callGenie)

- Wrote `callGenie` function in backend.
- Added logic to accept input, hit Genie endpoint, and return response.

  🛑 **Issue**: CloudFront link (API URL) not automatically generated.