Lambda functions – music web app

1. loginHandler-music:

import {DynamoDBClient, QueryCommand} from "@aws-sdk/client-dynamodb";

//Initialise DynamoDB Client

const dynamoClient = new DynamoDBClient({region: "us-east-1"});

export const handler = async (event) => {

console.log("Incoming Event: ", JSON.stringify(event, null, 2));

const headers = {

'Access-Control-Allow-Origin' : '\*',

'Access-Control-Allow-Methods' : 'OPTIONS,POST',

'Access-Control-Allow-Headers' : 'Content-Type',

'Content-Type' : 'application/json',

};

if(event.httpMethod === "OPTIONS"){

return{

statusCode: 200,

headers,

body: JSON.stringify({message: "CORS preflight success"}),

};

}

if(event.httpMethod === "POST"){

try{

//extract email and password from request body

const {email, password} = JSON.parse(event.body);

console.log("Email: ", email, " Password: ", password);

const params = {

TableName: "login",

IndexName : "email-Index", //GSI on email attribute

KeyConditionExpression: "email = :email",

ExpressionAttributeValues: {

":email": {S: email}

},

};

const {Items} = await dynamoClient.send(new QueryCommand(params));

//Check if user with this email exists

if(Items && Items.length > 0 ){

//use first result (since email is unique)

const user = Items[0];

//If the password matches

if(user.password.S === password){

return{

statusCode: 200,

headers,

body: JSON.stringify({

success: true,

message: "Login successful",

user\_name: user.user\_name.S,

}),

};

}else{

//If password doesn't match

return{

statusCode: 400,

headers,

body: JSON.stringify({

success: false,

message: "Invalid email or password",

}),

};

}

}else{

//If user with this email doesn't exist

return{

statusCode: 400,

headers,

body: JSON.stringify({

success: false,

message: "Invalid email or password",

}),

};

}

}catch(error){

console.error("Error during login:", error);

return{

statusCode: 500,

headers,

body: JSON.stringify({

success: false,

message: "Internal server error",

}),

};

}

}else{

//unsupported HTTP method

return{

statusCode: 405,

headers,

body: JSON.stringify({

success: false,

message: "Method not allowed",

}),

};

}

};

1. registerHandler-music:

import {DynamoDBClient, PutItemCommand, QueryCommand} from "@aws-sdk/client-dynamodb";

const dynamoClient = new DynamoDBClient({region: "us-east-1"});

export const handler = async (event) => {

console.log("Incoming Event: ", JSON.stringify(event, null, 2));

const headers = {

'Access-Control-Allow-Origin' : '\*',

'Access-Control-Allow-Methods': 'OPTIONS,POST',

'Access-Control-Allow-Headers': 'Content-Type',

'Content-Type': 'application/json',

};

if(event.httpMethod === 'OPTIONS'){

return {

statusCode: 200,

headers,

body: JSON.stringify({message: "CORS preflight success"}),

};

}

if(event.httpMethod === 'POST'){

try{

const {email, username, password} = JSON.parse(event.body);

if(!email || !username || !password){

return{

statusCode: 400,

headers,

body: JSON.stringify({

success: false,

message: "All fields are required",

}),

};

}

//Validation for email

if(!/^s3\d{7}@student\.rmit\.edu\.au$/.test(email)){

return{

statusCode: 400,

headers,

body: JSON.stringify({

success: false,

message: "Email must be a valid RMIT student email address",

}),

};

}

//Validation for username

if(!/^\w+\d$/.test(username)){

return{

statusCode: 400,

headers,

body: JSON.stringify({

success: false,

message: "Username must end with a digit",

}),

};

}

//Validation for password

if(!/^\d{6}$/.test(password)){

return{

statusCode: 400,

headers,

body: JSON.stringify({

success: false,

message: "Password must be exactly 6 digits",

}),

};

}

//Check if email already exists using email GSI

const queryParams = {

TableName: 'login',

IndexName: 'email-Index',

KeyConditionExpression: 'email = :email',

ExpressionAttributeValues:{

':email' : {S: email},

},

};

const {Items} = await dynamoClient.send(new QueryCommand(queryParams));

//If yes, return this message

if(Items && Items.length > 0){

return{

statusCode: 400,

headers,

body: JSON.stringify({

success: false,

message: "The email already exists",

}),

};

}

//Otherwise, add new user to login table

const newUser = {

TableName : 'login',

Item:{

email: {S: email},

user\_name: {S: username},

password: {S: password},

},

};

await dynamoClient.send(new PutItemCommand(newUser));

return{

statusCode: 200,

headers,

body: JSON.stringify({

success: true,

message: "User registered successfully",

}),

};

}catch(error){

console.error("Error during registration: ", error);

return{

statusCode: 500,

headers,

body: JSON.stringify({

success: false,

message: "Server Error",

}),

};

}

}else{

return{

statusCode: 405,

headers,

body: JSON.stringify({

success: false,

message: "Method Not Allowed",

}),

};

}

};

1. setCurrUser-music

import{DynamoDBClient, PutItemCommand} from "@aws-sdk/client-dynamodb";

const dynamoClient = new DynamoDBClient({region: "us-east-1"});

export const handler = async (event) => {

console.log("Incoming Event: ", JSON.stringify(event,null,2));

const headers = {

'Access-Control-Allow-Origin' : '\*',

'Access-Control-Allow-Methods': 'OPTIONS,POST',

'Access-Control-Allow-Headers': 'Content-Type',

'Content-Type': 'application/json',

};

if(event.httpMethod === 'OPTIONS' ){

return {

statusCode: 200,

headers,

body: JSON.stringify({message: "CORS preflight success"}),

};

}

if(event.httpMethod === 'POST'){

try{

const {email, user\_name} = JSON.parse(event.body);

if(!email | !user\_name){

return {

statusCode: 400,

headers,

body: JSON.stringify({

success: false,

message: "Email is required"

}),

};

}

const params = {

TableName: "currentUser",

Item: {

email: {S: email},

user\_name: {S: user\_name},

},

};

await dynamoClient.send(new PutItemCommand(params));

return{

statusCode: 200,

headers,

body: JSON.stringify({

success: true,

message: "Current user set successfully",

}),

};

}catch(error){

console.log("Error: ", error);

return{

statusCode: 500,

headers,

body: JSON.stringify({

success: false,

message: "Error storing user",

}),

};

}

}else{

return{

statusCode: 405,

headers,

body: JSON.stringify({

success: false,

message: "Method not allowed",

}),

};

}

};

1. getCurrUser-music

import {DynamoDBClient, GetItemCommand} from '@aws-sdk/client-dynamodb';

const dynamoClient = new DynamoDBClient({region: 'us-east-1'});

export const handler = async (event) => {

console.log("Incoming Event: ", JSON.stringify(event, null, 2));

const headers ={

'Access-Control-Allow-Origin' : '\*',

'Access-Control-Allow-Methods': 'OPTIONS,GET',

'Access-Control-Allow-Headers': 'Content-Type',

'Content-Type': 'application/json',

};

if(event.httpMethod === 'OPTIONS'){

return{

statusCode: 200,

headers,

body: JSON.stringify({

succes: true,

message : "CORS pleflight success",

}),

};

}

if(event.httpMethod === 'GET'){

const email = event.queryStringParameters?.email;

if(!email){

return{

statusCode: 400,

headers,

body: JSON.stringify({

succes: false,

message : "Email is required",

}),

};

}

const params = {

TableName: "currentUser",

Key: { email: {S : email} },

};

try{

const {Item} = await dynamoClient.send(new GetItemCommand(params));

if(Item){

return{

statusCode: 200,

headers,

body: JSON.stringify({

success: true,

message : "User found",

user: {

email: Item.email.S,

user\_name: Item.user\_name.S,

},

}),

};

}else{

return{

statusCode: 404,

headers,

body: JSON.stringify({

success: false,

message : "No user found",

}),

};

}

}catch(error){

console.log("Error fetching user: ", error);

return{

statusCode: 500,

headers,

body: JSON.stringify({

success: false,

message : "Internal server error",

}),

};

}

}

return{

statusCode: 405,

headers,

body: JSON.stringify({

success: false,

message : "Method not allowed",

}),

};

}

1. fetchSubs-music

import { DynamoDBClient, QueryCommand } from '@aws-sdk/client-dynamodb';

import { S3Client, HeadObjectCommand, GetObjectCommand } from '@aws-sdk/client-s3';

import { getSignedUrl } from "@aws-sdk/s3-request-presigner";

const BUCKET\_NAME = "music-bucket-a1";

const dynamoClient = new DynamoDBClient({ region: 'us-east-1' });

const s3Client = new S3Client({ region: 'us-east-1' });

async function getArtistImagesFromS3(title, album) {

const keyBase = `${title}\_${album}`.replace(/\s+/g, "\_");

const folderPath = "artists\_images";

for (let i = 1; i <= 3; i++) {

const key = `${folderPath}/${keyBase}\_${i}.jpg`;

try {

const headCmd = new HeadObjectCommand({ Bucket: BUCKET\_NAME, Key: key });

await s3Client.send(headCmd);

const getObjectCmd = new GetObjectCommand({ Bucket: BUCKET\_NAME, Key: key });

return await getSignedUrl(s3Client, getObjectCmd, { expiresIn: 3600 });

} catch (err) {

if (err.name === "NotFound") continue;

else throw err;

}

}

return "";

}

export const handler = async (event) => {

console.log("Incoming Event: ", JSON.stringify(event, null, 2));

const headers = {

'Access-Control-Allow-Origin': '\*',

'Access-Control-Allow-Methods': 'OPTIONS,GET',

'Access-Control-Allow-Headers': 'Content-Type',

'Content-Type': 'application/json',

};

if (event.httpMethod === 'OPTIONS') {

return {

statusCode: 200,

headers,

body: JSON.stringify({ success: true, message: "CORS preflight success" }),

};

}

if (event.httpMethod === 'GET') {

const user\_email = event.queryStringParameters?.user\_email;

if (!user\_email) {

return {

statusCode: 400,

headers,

body: JSON.stringify({ success: false, message: "User Email is required" }),

};

}

const queryParams = {

TableName: 'music\_subscriptions',

KeyConditionExpression: 'user\_email = :user\_email',

ExpressionAttributeValues: {

':user\_email': { S: user\_email },

},

};

try {

const { Items } = await dynamoClient.send(new QueryCommand(queryParams));

if (Items && Items.length > 0) {

const formattedItems = await Promise.all(

Items.map(async (item) => {

const img\_url = await getArtistImagesFromS3(item.music\_title.S, item.music\_album.S);

return {

music\_album: item.music\_album.S,

music\_title: item.music\_title.S,

music\_artist: item.music\_artist.S,

music\_year: item.music\_year.S || "Unknown Year",

music\_img\_url: img\_url,

};

})

);

return {

statusCode: 200,

headers,

body: JSON.stringify({

success: true,

message: "Subscriptions fetched successfully",

subscriptions: formattedItems,

}),

};

} else {

return {

statusCode: 404,

headers,

body: JSON.stringify({ success: false, message: "No subscriptions found for the user" }),

};

}

} catch (error) {

console.error("Error fetching subscriptions: ", error);

return {

statusCode: 500,

headers,

body: JSON.stringify({ success: false, message: "Error fetching subscriptions" }),

};

}

}

return {

statusCode: 405,

headers,

body: JSON.stringify({ success: false, message: "Method not allowed" }),

};

};

1. queryHandler-music

import {DynamoDBClient, ScanCommand} from '@aws-sdk/client-dynamodb';

import {S3Client, HeadObjectCommand, GetObjectCommand} from '@aws-sdk/client-s3';

import {getSignedUrl} from "@aws-sdk/s3-request-presigner";

const TABLE\_NAME = "music";

const BUCKET\_NAME = "music-bucket-a1"

const dynamoClient = new DynamoDBClient({region: 'us-east-1'});

const s3Client = new S3Client({region: 'us-east-1'});

async function getArtistImagesFromS3(title,album){

const keyBase = `${title}\_${album}`.replace(/\s+/g,"\_");

const folderPath = "artists\_images";

for (let i=1;i<=3;i++){

const key = `${folderPath}/${keyBase}\_${i}.jpg`;

try{

const headCmd = new HeadObjectCommand({

Bucket: BUCKET\_NAME,

Key: key,

});

await s3Client.send(headCmd);

const getObjectCmd = new GetObjectCommand({

Bucket: BUCKET\_NAME,

Key: key,

});

const signedUrl = await getSignedUrl(s3Client, getObjectCmd, {expiresIn: 3600});

return signedUrl;

}catch(err){

if(err.name === "NotFound"){

continue;

}else{

throw err;

}

}

}

return "";

}

export const handler = async (event) => {

console.log("Incoming Event: ", JSON.stringify(event, null, 2));

const headers ={

'Access-Control-Allow-Origin' : '\*',

'Access-Control-Allow-Methods': 'OPTIONS,POST',

'Access-Control-Allow-Headers': 'Content-Type',

'Content-Type': 'application/json',

};

if(event.httpMethod === 'OPTIONS'){

return{

statusCode: 200,

headers,

body: JSON.stringify({

succes: true,

message : "CORS pleflight success",

}),

};

}

if(event.httpMethod === 'POST'){

const {title, year, artist, album } = JSON.parse(event.body);

if(!title && !year && !artist && !album){

return{

statusCode: 400,

headers,

body: JSON.stringify({

success: false,

message: "At least one field is required",

}),

};

}

const queryParams = {

TableName: 'music',

ExpressionAttributeNames : {},

ExpressionAttributeValues: {},

FilterExpression: "",

};

let filterConditions = [];

if (title) {

queryParams.ExpressionAttributeNames["#title"] = "title";

queryParams.ExpressionAttributeValues[":title"] = { S: title };

filterConditions.push("#title = :title");

}

if (artist) {

queryParams.ExpressionAttributeNames["#artist"] = "artist";

queryParams.ExpressionAttributeValues[":artist"] = { S: artist };

filterConditions.push("#artist = :artist");

}

if (year) {

queryParams.ExpressionAttributeNames["#year"] = "year";

queryParams.ExpressionAttributeValues[":year"] = { S: year };

filterConditions.push("#year = :year");

}

if (album) {

queryParams.ExpressionAttributeNames["#album"] = "album";

queryParams.ExpressionAttributeValues[":album"] = { S: album };

filterConditions.push("#album = :album");

}

if (filterConditions.length > 0) {

queryParams.FilterExpression = filterConditions.join(" AND ");

}

console.log("DynamoDB Query Params:", queryParams);

try {

const { Items } = await dynamoClient.send(new ScanCommand(queryParams));

if (Items && Items.length > 0) {

const formattedItems = await Promise.all (

Items.map(async (item) => {

const img\_url = await getArtistImagesFromS3(item.title.S, item.album.S);

return{

music\_album: item.album.S,

music\_title: item.title.S,

music\_artist: item.artist.S,

music\_year: item.year.S || "Unknown Year",

music\_img\_url: img\_url,

};

})

);

return{

statusCode: 200,

headers,

body: JSON.stringify({

success: true,

message: "Search successful",

music: formattedItems,

}),

};

}else{

return{

statusCode: 404,

headers,

body: JSON.stringify({

success: false,

message: "No result is retrieved. Please query again.",

}),

};

}

}catch(error){

console.error("Error querying music:", error);

return{

statusCode: 405,

headers,

body: JSON.stringify({

success: false,

message: "Method not Allowed",

}),

};

}

}

}

1. addSubs-music

import { DynamoDBClient, PutItemCommand } from "@aws-sdk/client-dynamodb";

const dynamoClient = new DynamoDBClient({ region: "us-east-1" });

export const handler = async (event) => {

console.log("Incoming Event: ", JSON.stringify(event, null, 2));

const headers = {

'Access-Control-Allow-Origin': '\*',

'Access-Control-Allow-Methods': 'OPTIONS,POST',

'Access-Control-Allow-Headers': 'Content-Type',

'Content-Type': 'application/json',

};

if (event.httpMethod === 'OPTIONS') {

return {

statusCode: 200,

headers,

body: JSON.stringify({ success: true, message: "CORS preflight success" }),

};

}

if (event.httpMethod === 'POST') {

const { user\_email, music\_title, music\_artist, music\_year, music\_album, music\_img\_url } = JSON.parse(event.body);

if (!user\_email || !music\_album || !music\_title || !music\_artist || !music\_year || !music\_img\_url) {

return {

statusCode: 400,

headers,

body: JSON.stringify({ success: false, message: 'Required fields missing' }),

};

}

const music\_id = `${music\_album}\_${music\_title}`;

const params = {

TableName: 'music\_subscriptions',

Item: {

user\_email: { S: user\_email },

music\_id: { S: music\_id },

music\_album: { S: music\_album },

music\_title: { S: music\_title },

music\_artist: { S: music\_artist },

music\_year: { S: music\_year },

music\_img\_url: { S: music\_img\_url },

},

};

try {

console.log('Attempting to add subscription with params:', JSON.stringify(params));

await dynamoClient.send(new PutItemCommand(params));

return {

statusCode: 200,

headers,

body: JSON.stringify({ success: true, message: "Music subscribed successfully" }),

};

} catch (error) {

console.error('Error adding subscription:', error);

return {

statusCode: 500,

headers,

body: JSON.stringify({ success: false, message: "Error adding subscription" }),

};

}

}

return {

statusCode: 405,

headers,

body: JSON.stringify({ success: false, message: "Method not allowed" }),

};

};

1. removeSubs-music

import { DynamoDBClient, DeleteItemCommand } from "@aws-sdk/client-dynamodb";

const dynamoClient = new DynamoDBClient({ region: "us-east-1" });

export const handler = async (event) => {

console.log("Incoming Event: ", JSON.stringify(event, null, 2));

const headers = {

'Access-Control-Allow-Origin': '\*',

'Access-Control-Allow-Methods': 'OPTIONS,DELETE',

'Access-Control-Allow-Headers': 'Content-Type',

'Content-Type': 'application/json',

};

if (event.httpMethod === 'OPTIONS') {

return {

statusCode: 200,

headers,

body: JSON.stringify({

success: true,

message: "CORS preflight success",

}),

};

}

if (event.httpMethod === 'DELETE') {

const { user\_email, music\_album, music\_title } = JSON.parse(event.body);

if (!user\_email || !music\_album || !music\_title) {

return {

statusCode: 400,

headers,

body: JSON.stringify({

success: false,

message: 'Required fields missing'

}),

};

}

const music\_id = `${music\_album}\_${music\_title}`;

const deleteParams = {

TableName: 'music\_subscriptions',

Key: {

user\_email: { S: user\_email },

music\_id: { S: music\_id },

},

};

try {

await dynamoClient.send(new DeleteItemCommand(deleteParams));

return {

statusCode: 200,

headers,

body: JSON.stringify({

success: true,

message: "Subscription removed successfully"

}),

};

} catch (error) {

console.error('Error removing subscription:', error);

return {

statusCode: 500,

headers,

body: JSON.stringify({

success: false,

message: "Error removing subscription"

}),

};

}

}

return {

statusCode: 405,

headers,

body: JSON.stringify({

success: false,

message: "Method not allowed"

}),

};

};

1. logoutHandler-music

import { DynamoDBClient, DeleteItemCommand } from "@aws-sdk/client-dynamodb";

const dynamoClient = new DynamoDBClient({ region: "us-east-1" });

export const handler = async (event) => {

console.log("Incoming Event: ", JSON.stringify(event, null, 2));

const headers = {

'Access-Control-Allow-Origin': '\*',

'Access-Control-Allow-Methods': 'OPTIONS,DELETE',

'Access-Control-Allow-Headers': 'Content-Type',

'Content-Type': 'application/json',

};

if (event.httpMethod === 'OPTIONS') {

return {

statusCode: 200,

headers,

body: JSON.stringify({

success: true,

message: "CORS preflight success",

}),

};

}

if (event.httpMethod === 'DELETE') {

const { email } = JSON.parse(event.body);

if (!email) {

return {

statusCode: 400,

headers,

body: JSON.stringify({

success: false,

message: 'User email is required',

}),

};

}

const deleteParams = {

TableName: 'currentUser', // Your DynamoDB table name

Key: {

email: { S: email },

},

};

try {

await dynamoClient.send(new DeleteItemCommand(deleteParams));

return {

statusCode: 200,

headers,

body: JSON.stringify({

success: true,

message: "User logged out successfully",

}),

};

} catch (error) {

console.error('Error logging out user:', error);

return {

statusCode: 500,

headers,

body: JSON.stringify({

success: false,

message: "Error logging out user",

}),

};

}

}

return {

statusCode: 405,

headers,

body: JSON.stringify({

success: false,

message: "Method not allowed",

}),

};

};