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

import {

PutCommand,

ScanCommand,

UpdateCommand,

DeleteCommand,

DynamoDBDocumentClient,

} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient();

const docClient = DynamoDBDocumentClient.from(client);

export const handler = async (event, context) => {

console.log("Incoming event:", event);

try {

switch (event.httpMethod) {

case "GET":

return await handleGetRequest();

case "POST":

return await handlePostRequest(event, context);

case "PATCH":

return await handlePatchRequest(event);

case "DELETE":

return await handleDeleteRequest(event);

default:

return {

statusCode: 400,

body: JSON.stringify({ message: "Unsupported method" }),

};

}

} catch (error) {

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

return {

statusCode: 500,

body: JSON.stringify({ message: "Server error", error: error.message }),

};

}

};

//

// GET → fetch all tasks

//

const handleGetRequest = async () => {

const command = new ScanCommand({

TableName: "TodoTable",

});

const response = await docClient.send(command);

return {

statusCode: 200,

body: JSON.stringify(response.Items),

};

};

//

// POST → create a new task

//

const handlePostRequest = async (event, context) => {

const { name, completed } = JSON.parse(event.body);

const command = new PutCommand({

TableName: "TodoTable",

Item: {

id: context.awsRequestId, // unique id per request

name,

completed,

},

});

await docClient.send(command);

return {

statusCode: 200,

body: JSON.stringify({ message: "Task created successfully" }),

};

};

//

// PATCH → update an existing task

//

const handlePatchRequest = async (event) => {

const { id, name, completed } = JSON.parse(event.body);

const command = new UpdateCommand({

TableName: "TodoTable",

Key: { id },

ExpressionAttributeNames: {

"#n": "name",

},

UpdateExpression: "SET #n = :n, completed = :c",

ExpressionAttributeValues: {

":n": name,

":c": completed,

},

ReturnValues: "ALL\_NEW",

});

const response = await docClient.send(command);

return {

statusCode: 200,

body: JSON.stringify({

message: "Task updated successfully",

task: response.Attributes,

}),

};

};

//

// DELETE → delete a task

//

const handleDeleteRequest = async (event) => {

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

const command = new DeleteCommand({

TableName: "TodoTable",

Key: { id },

ReturnValues: "ALL\_OLD",

});

const response = await docClient.send(command);

return {

statusCode: 200,

body: JSON.stringify({

message: "Task deleted successfully",

task: response.Attributes,

}),

};

};