Ari

Computing ~~Mini~~ Project

~~Connecting a Discord bot to DynamoDB~~

*\*cough\* Ehem sorry, that’s the wrong title.*

*…*

*Here we go. All set.*

**The writing of a Javascript discord bot, and how it can intake and send information to DynamoDB.**

At first, my project seemed to be an easy one. I had found an online guide by Brian Morrison II about how to create/write discord bots – and what needed to be done to connect said discord bot with a DynamoDB instance, thereby storing information into a database. (The guide can be found here: <https://brianmorrison.me/blog/storing-info-in-a-database-with-discord-bots/>). I was fairly excited about this concept, as we had used DynamoDB in class – and I have already spun up other amazon servers for school projects before, and spreading out on amazon utilities seems incredibly useful for me.

**The Beginning**

To start, I created the table I plan on sending and storing information in. We additionally tag it for organization, assisting with later tracking and finding (if there was more then one database on the account.)

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

We then setup an Identity and Access Management (IAM) user for our bot with full access to DynamoDB. We save the AWS token for later use, we’ll be putting it in our ‘.env’ file to connect our discord bot to this database.

**…of the end**

Getting to the next part of the guide, I need to make the discord bot that can connect with and provide data to the database. Brian Morrison’s guide talks about how to setup a ‘survey’ command on a bot, and the bot-code to do so. We create a new bot at discord.com/developers, pulling its Application ID (to combine with the AWS tokens in the ‘.env’ folder.) We also give the bot ‘administrator’ permission so it can take adequate actions on discord. We also provide it ‘Scopes’, which at the time I didn’t know I needed – but will soon discover I required. Scopes are a the method modern discord bot api uses to authorize actions. As I was about to discover while following the guide, since when it was written – discord.js (the API) had updated from v12 to v13 since.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, application, Teams

Description automatically generated

Starting on the bot, I install all the nodejs packages required to setup and run the bot. However, I quickly run into a couple bugs – things not working, or simply working differently then expected. For example, client.on.message instead wanting me to use client.on.messageCreate – and similar version incompatibility. I get through the simple setup (getting the bot running, simple ping-pong messages, ect. However, when I get through the guide to the segment on setting up a !survey command, nothing responds to me.

Graphical user interface, text, application, chat or text message

Description automatically generated

This issue continues, until I reach out to the creator of the blog Brian Morrison directly on discord and him and others point me towards the Discord.js refactor. The Discord.js v13 update enforced a ton of new requirements, including privileged intent for message content (requiring bots to directly be authorized to work with others, slash commands, and the ability for folders to house files and commands. Because of all these changes, Brian Morrison’s code and guide was out of date. I found the official discord.js making a bot guide – *and with these two guides combined, I have the power to control the ~~world~~* ***server!***

<https://discord.com/blog/slash-commands-are-here> <https://discordjs.guide/additional-info/changes-in-v13.html>

<https://discordjs.guide/creating-your-bot/>

**Interactions, commands, async replies, and more.**

With a working discord.js v13 bot in hand, I could start tackling the real reason I was there – to make a survey command that can intake data and send it to DynamoDB. I started with refactoring Brian Morrison’s code to utilize ‘interaction channels’, one of the v13 changes made it so instead of directly publishing to a channel. ‘Interactions’ are created when someone runs a slash command. The bot is required to follow-up to a response within 3 seconds or leave the channel open with a ‘delay period’. One option we have is ‘awaitMessages()’, that waits for a response and allows us to send to the same channel and response. Specifically, the command ‘interaction.channel.followUp()’ sends a response to the same channel the command originally started in. That is to say – ‘interaction’ pulls information from the channel the command was first run in (IE: that opens up the interaction). Interaction.channel pulls the specific channel location – and .followUp() can send a response to the same location.

I almost was finished at writing a working /survey command. However, I ran into a major async awaitMessages() issue – I couldn’t for the life of me get chained questions to function. I would get a single response functioning, but I had to fully nest each question until I could get it functioning in any regard.

A screenshot of a computer screen

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, chat or text message

Description automatically generated

At this point I had a working bot, a working database, but for some reason my code wasn’t sending to the database. As seen in the above image, it was fully functional – but it was also late at night, and I decided to rest for the night. I send a query off to Brian Morrison, seeing if I was missing something obvious- and if they could help. And in the morning, I got a response back! Apparently, I had an index for my storage rather than an object, and thus when it was sent to DynamoDB, it wouldn’t take the items. I merged in his branch which made the fix, as well as his helpful rearrangement of my crazy nesting to a cleaner organization.Graphical user interface, text, application, Word

Description automatically generated