**NOTES** [things to do]

* **~~Look into pooled connections for fast querying~~**
* **At each step, consider what data you can leverage for analytics and AI purpose. Perhaps a recommendation system for joining a certain ‘group’ in a server?**
* [**https://www.pinecone.io/learn/vector-database/**](https://www.pinecone.io/learn/vector-database/)
  + **Let server admins enable “AI” by answering questions on what type of server they run and doing so enables access to other servers that have answered these questions. Can use this to return recommendation servers to others in your server.**
* **/preset add (preset for project, team, study)**
* **/preset remove (all, project, team, study)**
* **/set prefix (Admin only)**
  + **/set permission for what roles can invoke /create [command]**
  + **/set permission for what roles can invoke /form [command]**
  + **/set variables from the ‘group\_categories’ table**
    - **i.e. /set group [group\_category] [some variable]**
* **A group\_category is a string or phase an admin set. Each group\_category is essentially a pointer to a "Category" in a discord server.**

- we will write all the bot's slash commands. I will go through one-by-one which ones I am writing.

**- we will write necessary functions to update appropriate database tables for said slash commands, with error checking**

**- We will decide how to implement a 30 day grace period on\_guild\_leave, so that if the guild doesn't re-add the bot, it will remove all the data for that guild\_id in all other tables.**

**Index of** [commands]**:**

**Global Key:**

[] = required paramater

() = optional parameter

**Command Prefixes:**

/create [command] [database dev-guide]

/disband [command] [database dev-guide]

/form [command] [database dev-guide]

/set [command] [database dev-guide]

**Database Tables:**

‘group\_categories’

‘groups’

**Server Owner Commands:**

/wipe

/recover

**Admin Commands:**

/create group

/create poll

/disband group

**User Commands:**

/form

/disband

**/create** [command]

**/create Command List:**

* /create group
* /create poll

**/create Admin Command(s):**

**1. Create a Group Type**

**Command:**  
/create group [group\_category] [post\_channel\_id]

(emoji, default=👍) (std\_admin\_msg, default=“”)

**Example Admin Usage:**

/create [team] [#find-teams]

(👍) (\*\*This is a team group type.\*\*)

**Function:**

Only a server admin can call this method. When invoked, a server [category\_id] is created with the name [group\_category]. If set, a specific (emoji) and (std\_admin\_msg) is stored for this [group\_category]. Anytime a user invokes /form group command, it will refer to the [group\_category] created by the admin, and gather the necessary [category\_id], [post\_channel\_id], (emoji), and (std\_admin\_msg) to initialize a user group.

**2. Create a Poll**

**Command:**

/create poll [question]

(choices) (existing channel\_ids)

**Example Admin Usage:**

/create poll [What group do you want to join?]

(a, b, c) (#a, #b, #c)

**Function:**

A server admin can initiate a poll in any channel. When invoked, a message containing the poll will be posted. If the admin specifies choices, the question will populate with reaction emotes of letters 🇦, 🇧, 🇨 ... and so on, otherwise it will populate with 👍 and 👎. Moreover, the admin can optionally map these choices to a specific channel\_id. By doing so, users can directly join the corresponding channel by reacting with the appropriate choice.

**/disband** [command]

**Admin Command(s):**

**1. Disband a Group**

**Command:**

/disband [channel\_id] (retype confirmation message)

**Example Admin Usage:**

/disband [#jacks-team] (I understand this will delete #jacks-team permanently)

**Function:**

Only a server admin can execute this command. It can be called anywhere inside the server. The admin must retype the confirmation message as shown to confirm deletion. When invoked, the entire channel data is archived, and sent to the [group\_owner\_id\_id] and admin in a private DM. Then, the associated [text\_channel\_id] and [voice\_channel\_id] are removed from the respective [category\_id]. The post with [post\_id] is deleted from the [post\_channel\_id] location. The admin can choose to be notified that a group has been disbanded in the form of a log.

**User Command(s):**

**1. Disband a Group**

**Command:**

/disband (confirmation message)

**Example User Usage:**

/disband (I understand this will delete #jacks-team permanently)

**Function:**

Only [group\_owner\_id\_id] can call this method inside the [channel\_id] where the appropriate group exists. The user must retype the confirmation message as shown to confirm deletion. When invoked, entire channel data is archived, and sent to the [group\_owner\_id\_id] in a private DM. Then, the associated [text\_channel\_id] and [voice\_channel\_id] are removed from the respective [category\_id]. The post with [post\_id] is deleted from the [post\_channel\_id] location. The admin can choose to be notified that a group has been disbanded in the form of a log.

**/form** [command]

**/form Command List:**

* /form group

**/form Admin Command(s):**

**/form User Command(s):**

**1. Form a Group**

**Command:**

/form group [group\_category] [group\_name] (user message)

**Example User Usage:**

/form group [team] [Jack’s Team]

(Read this resource: xyz.com)

**Function:**

FILL THIS OUT

**Standard Message Format:**

Welcome to our [group\_category] “[group\_name]”!

Please react with [emoji, default=👍] to board this group.

(std\_admin\_msg)

(user message)

[group\_category] by @user

**Example Output:** font size and spacing may vary

Welcome to my team “Jack’s Team”!

Please react with 👍 to board this group.

**This is a team group type.**

Read this resource: xyz.com

Team by @user

**DATABASE TABLES**[database dev-guide]

1. **Table Name:** ‘guilds’

|  |  |
| --- | --- |
| guild\_id | join\_date |
|  |  |

**Table Relationships:**

[guild\_id] (PRIMARY Key)

Uses Indexing to ensure that guild\_id exists where any command is being called.   
  
**TERMINAL COMMAND:**CREATE TABLE guilds (

guild\_id BIGINT PRIMARY KEY,

join\_date TIMESTAMP);

1. **Table Name:** ‘group\_categories’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| guild \_id | category\_id | category\_name | post \_channel\_id | emoji | admin\_msg |
|  |  |  |  |  |  |

**Table Relationships:**

[guild \_id] (Foreign Key)

Ensures every ‘group\_category’ is associated with an existing server.

[guild\_id, category\_id] (Composite Primary Key)

Ensures within a specific server, duplicate group\_categories cannot exist.

**POSTGRESQL COMMAND:**

CREATE TABLE group\_categories (

guild\_id BIGINT REFERENCES guilds(guild\_id),

category\_id BIGINT NOT NULL,  
category\_name TEXT NOT NULL,

post\_channel\_id BIGINT NOT NULL,

emoji TEXT DEFAULT '👍',

admin\_msg TEXT DEFAULT '',

PRIMARY KEY (guild\_id, category\_id));

1. **Table Name:** ‘groups’

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| guild \_id | category\_id | group\_ owner\_id | Post \_id | text\_ channel\_id | Voice\_ channel\_id | Created\_date |
|  |  |  |  |  |  |  |

**Table Relationships:**

[guild\_id] (Foreign Key)

Ensures every group\_category created is for an existing server.

[guild\_id, category\_id] (Composite Primary Key)

Ensures that a category\_id is defined for that server. Also links to ‘group\_categories’ Table giving access to details like [category\_name],[post\_channel\_id], [emoji], and [std\_admin\_msg] when user forms a group.

[text\_channel\_id] (Foreign Key)  
Links to ‘text\_channel\_id’ in the ‘groups’ table, indicating the variables associated with type for deletion process.

**POSTGRESQL COMMAND:**

CREATE TABLE groups (

guild\_id BIGINT REFERENCES guilds(guild\_id),

category\_id BIGINT NOT NULL,

group\_owner\_id BIGINT NOT NULL,

post\_id BIGINT NOT NULL,

text\_channel\_id BIGINT UNIQUE NOT NULL,

voice\_channel\_id BIGINT NOT NULL,

created\_date TIMESTAMP NOT NULL,

FOREIGN KEY (guild\_id, category\_id) REFERENCES group\_categories(guild\_id, category\_id));

**/create** [database dev-guide]

Relevant Data: [server\_id], [group\_category], [group\_category\_color, default=], [emoji, default=👍], [std\_admin\_msg, default=]

**/create Command List:**

* /create group
* /create poll

**/create Admin Command(s):**

1. **Create a Group Type**

**Command:**

/create group [group\_category] [post\_channel\_id]

(emoji, default=👍) (std\_admin\_msg, default=“”)

**Method:**

When an admin invokes the /create group command, the bot stores predefined data such as server\_id, group\_category, category\_id, post\_channel\_id, emoji (defaulting to "👍" if not specified), and std\_admin\_msg (empty by default) in a dedicated table ‘group\_categories’. Later, when a user uses the /form group command, the bot refers to this table to fetch details about the group type in the server, using them to generate user posts and server channels accordingly.

**Considerations:**

~~- Admin creates duplicate group\_category.~~

* ~~Solution: Print “This group already exists in this server!” error after checking it’s duplicate.~~

- Admin deletes Category without bot.

* Solution: If a Category is deleted without bot, the next time a user invokes /form group with that group\_category, it will delete any Table rows with the [server\_id, group\_category] (Composite Primary Key) and print error “Sorry, this group doesn’t exist within the server.”

**Data Variables:**

[server\_id] – The server where command was invoked.

[group\_category] – The type/category of group set by admin.

[category\_id] – The category\_id generated from creation of a group\_category.

[emoji, default=👍] – Reaction role emoji associated with posts with group\_category.

[std\_admin\_msg, default=“”] – Standard message by admin for posts with group\_category.

**User Command(s):**

**/disband** [database dev-guide]

[group\_owner\_id\_id\_id], [group\_category], [group\_category\_color, default=], [group\_name],

[emoji, default=👍], [std\_admin\_msg, default=], [post\_id], [role\_id], [channel\_id]

**Admin Command(s):**

1. **Disband a Group**

**Command:**

/disband [channel\_id]

**Method:**

In the database table ‘group\_categories’, any rows with [text\_channel\_id] (foreign key) are removed.

**Table Name:** ‘form\_group\_category’

**Data Variables:**

[server\_id] – Reference to server where command was invoked.  
[group\_owner\_id\_id\_id] – Reference to user who created the group.  
[channel\_id] – ID of the private channel created for the group.

**Table Relationships:**

[server\_id] (Foreign Key)   
Ensures every ‘channel\_id’ being deleted is associated with an existing server.

[text\_channel\_id] (Foreign Key)  
Links to ‘text\_channel\_id’ in the ‘groups’ table, indicating the variables associated with type for deletion process.

**User Command(s):**

1. **Disband a Group**

**Command:**

/disband

**Method:**

In the database table ‘form\_group\_category’, any rows with [text\_channel\_id] (foreign key) are removed.

**Data Variables:**

[server\_id] – Reference to server where command was invoked.  
[group\_owner\_id\_id\_id] – Reference to user who created the group.  
[channel\_id] – ID of the private channel created for the group.

**Table Relationships:**

[server\_id] (Foreign Key)   
Ensures every ‘channel\_id’ being deleted is associated with an existing server.

[text\_channel\_id] (Foreign Key)  
Links to ‘text\_channel\_id’ in the ‘groups’ table, indicating the variables associated with type for deletion process.

**/form** [database dev-guide]

**Admin Command(s):**

**User Command(s):**

**1. Form a Group**

**Command:**

/form group [group\_category] [group\_name] (user message)

**Method:**   
When a user invokes the /form group command, the system logs the relevant details into a table. This table captures details like the server's ID, group type, group name, group owner, post ID, emoji, and the IDs of the created text and voice channels, along with the date of creation. When a user executes the command, it leads to the creation of new text and voice channels under the specified category in the server, both named after the group name provided. The invoking user is tagged as the group owner. Concurrently, a message gets posted with an emoji reaction, setting up a reaction-based access control. Users who react to this post with the correct emoji gain access to the channels. To manage inactive groups, a created\_date timestamp is noted. Groups with 30 days of inactivity alert the admin and the group owner, giving a 7-day grace period to respondto the bot before automated removal.

**Data Variables:**

[server\_id] – The server where command was invoked.

[group\_category] – Links [category\_id] from ‘group\_categories’ table.

[group\_owner\_id\_id] – Records the user who created the group.  
[group\_name] – Name of the text and voice channels.   
[post\_id] – ID of the post with the reaction role.  
[text\_channel\_id] – ID of the private text channel created for the group.  
[voice\_channel\_id] – ID of the private voice channel created for the group.

[created\_date] – Date the group was created.