CONTENTS

CONTENTS 1

1.0 Tempo 2

2.0 COMMANDS 2

2.1 Punishments 2

2.1.1 /ban 3

2.1.2 /kick 3

2.1.3 /warn 3

2.1.4 /unban 3

2.2 Logging 3

2.2.1 /case 4

2.2.2 /cases 4

2.2.3 /deletecase 4

2.3 Assorted 4

2.3.1 /config 4

2.3.2 /help 5

2.3.3 /ping 5

2.3.4 /server 5

2.3.5 /user 5

3.0 Events/features 5

3.1 guildCreate 5

3.2 guildDelete 6

3.3 guildMemberAdd 6

3.4 guildMemberRemove 6

3.5 interactionCreate 6

3.6 Ready 6

4.0 ERROR HANDLING 6

**FIGURES**

[Figure 1‑1: Tempo Moderation Logo 2](#_Toc129278316)

# Tempo

Tempo Moderation, part of the Tempo Tools family, is the main, and currently the only functioning bot. It is written in JavaScript, using Node.js and Discord.Js, It utilizes MySQL to allow logging of data for various purposes across the bot.

Its purpose is to allow users to easily moderate their servers, and log such actions that they may perform for further usage. This started as a passion project, me wanting to code something, but I now wish to be able to publish the bot, and get it used by users in replacement of other bots.

A picture containing icon

Description automatically generated

Figure 1‑1: Tempo Moderation Logo

# COMMANDS

The following is a list of commands that the bot currently supports, the usage and assorted information related to each one.

Key:

/commandname

<required>

[optional]

## Punishments

This section will contain all punishment based commands that can be utilized to inform users of infractions caused against the server/individuals.

### /ban

/ban is a straight forward command, allowing users with sufficient permission to ban other users from the server, providing the user with a ban notification from the bot, which would include why the ban was issued, and also automatic logging of the punishment on Tempo’s systems for further usage.

/ban <userid/user@> <reason>

The requirement for this command is for the punishing user to have the “Ban Members” permission, the bot to be higher ranked than the punished user, and for the actioning user to be higher ranked than the punished user. You are unable to ban users who are not in the server, but I may consider adding it in the future.

### /kick

/kick is a straight forward command, allowing users with sufficient permission to kick other users from the server, providing the user with a kick notification informing them of why they were removed, alongside automatic logging of the punishment.

/kick <userid/user@> <reason>

The requirements for this command are for the punishing user to have the “Kick Members” permission, the bot to be higher ranked than the punished user, and for the actioning user to be higher ranked than the punished user.

### /warn

/warn is a simple command, allowing users to warn other users of any infractions that have been caused. It notifies the user of their warn alongside automatic logging by Tempo.

/warn <userid/user@> <reason>

To execute this command, you need to have the “Moderate Members” permission alongside being higher ranked than the user alongside the bot.

### /unban

/unban allows for users to revoke bans from users, allowing them to re-join the server. It will automatically log the unban to tempos systems for further usage.

/unban <userid/user@> <reason>

To run this command, you are required to have the “Ban Members” permission.

## Logging

This section will contain all logging commands relevant to the aforementioned topic, punishments.

### /case

This command allows you to find a specific case based on the case ID, which can be found in the log channel when a punishment is made, or by using /cases.

/case <caseid>

To run this command, you are needed to have the “Moderate Members” permission.

### /cases

/cases works similar to how /case works, but instead it fetches all punishments from the user mentioned. It will then return all cases found, with the id, the type, the date, the reason and the moderator who punished the user.

/cases <userid/user@>

This command also requires the “Moderate Members” permission to be ran.

### /deletecase

This command is the opposite of /case, with it removing a specific case instead of finding it. It delete the case, and display all information about the punishment.

/deletecase <caseid>

This command requires the “Moderate Members” permission

## Assorted

These commands don’t really belong anywhere else but are useful in my personal opinion.

### /config

This command allows the server owner to allocate a welcome/leave channel and a log channel.

/config <option> <set>

Current config options:

Join – Welcome/leave channel

Log – bot logging channel

This requires the server owner to run it, as otherwise it does not work. May update to require administrator if needed.

### /help

This command returns all available commands and their usage. It is similar to this document.

/help [commandname]

If you run the command without a command name, it will return all available command names for the user to rerun the command with the relevant name. Has no required permissions

### /ping

This command returns the ping, not too sure how stable it is/accurate it is, but it does a thing. No required permissions

### /server

This is a command that returns the server name and total members, needs to be updated to contain more information but unsure what to add.

### /user

This is my favourite command, and allows you to fetch information about a user

It will return username, pfp, user id, mention, account age, user join date and user roles. If the user is not in the server, it returns the user tag, user id, mention and pfp

/user <userid/user@>

It requires the “Moderate Members” permission to run

# Events/features

This section contains all features that are currently available, to users or to me.

I plan on making analytical saving of these events for further research.

## guildCreate

This event is triggered by the bot joining a new server. Upon it being triggered, the bot updates its status to reflect the new server count. It also sends a message into the global log channel informing me of a new server join. It also creates a record of the server in its storage system for further use.

## guildDelete

This event is triggered by the bot leaving a server. Upon it leaving, it will update its status and remove the server information from its systems. It will not remove any punishment records.

## guildMemberAdd

This event is triggered by a user joining a server that the bot is in. This allows for the bot to send Welcome messages, and log the user joining in the server log channel and global log channel.

## guildMemberRemove

This event is triggered by a user leaving a server that the bot is in, or the bot being removed from a server. This allows it to send a leave message to the server, and recording it in the server log channel and global log channel.

## interactionCreate

This is triggered when a command is used in a server. Nothing too special here other than to figure out who caused the bot to crash when it was used.

## Ready

This event is triggered when the bot is turned on, and sets the status, checks the backend systems for the servers and removes servers it was removed from while offline, and add new servers that it joined while offline.

# ERROR HANDLING

The bot has multiple systems setup to automatically log when an event occurs, and even more to allow for error logging. When an error occurs, it will spit out the error, including who, what, where and why.