Discord API • Part II

Interact with Midjourney using Discord API • Part II

9 min read • October 16, 2023

Table of contents

Introduction

This article builds upon Part I and covers Discord rate limits and the Midjourney moderation system, two of the most intricate topics. We've provided a working code for your convenience.

Before we proceed, make sure you have a Discord account with an active Midjourney subscription. The $10 Basic Plan will suffice.

Follow these simple steps to obtain:

- Discord server id number, referenced in this article as `server_id`
- Discord channel id number, referenced in this article as `channel_id`
- Discord token, referenced in this article as `discord_token`

Discord rate limits

Discord's original documentation suggests using response headers to detect cases where the rate limit is exceeded and adjust accordingly. While this approach is appropriate for complex or commercial applications, we'll opt for a simpler yet equally efficient method for this article.

We'll introduce a 350-millisecond pause (sleep) before each Discord API call. Since our code operates with a single thread, this will be sufficient to remain comfortably under the limit. Moreover, our small program's overall performance will not be impacted, as Midjourney's response time consistently exceeds the proposed 350 milliseconds.

Midjourney moderation system

Midjourney performs both pre- and post-moderation of /imagine command prompts, and you'll need a reliable way to detect both.

When you execute a POST request for an imagine interaction with your prompt in a Discord channel (as covered in Part I), you'll encounter ***seven*** possible cases:

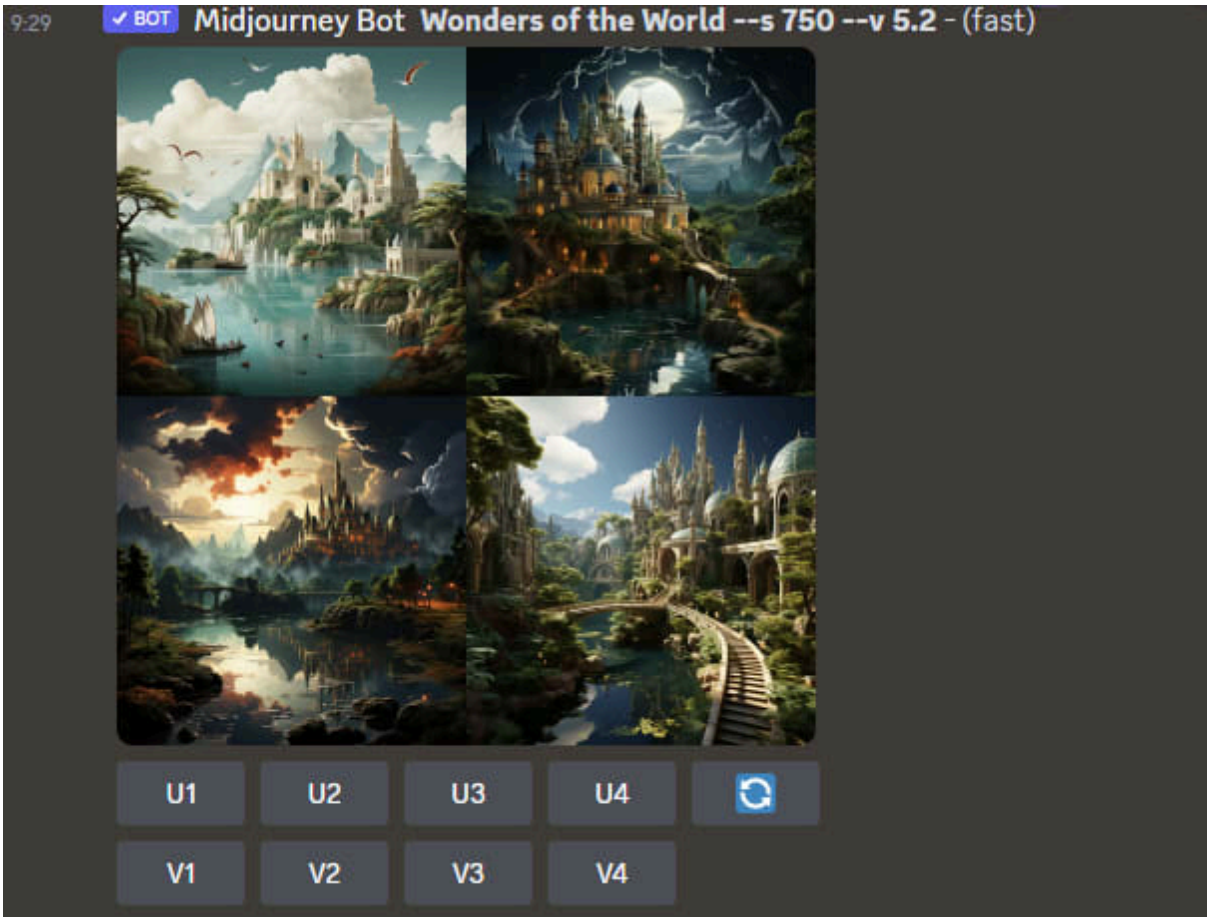
1 **Happy path**

After a few seconds, the posted message will appear in the channel with the original message `id`, indicating its status as `(Waiting to start)` in the `content` field and with the `type` field set to 0.

GET `https://discord.com/api/v10/channels/channel_id/messages`

```
[
 {
  "id": "<Discord original message id>",
  "type": 0,
  "content": "**Wonders of the World…** - <@Discord user id> (Waiting to start)",
  "channel_id": "<Discord channel id>",
```

Once Midjourney completes the generation, the original message will be deleted, and a final message with a new `id` containing the generation results will be posted by Midjourney.

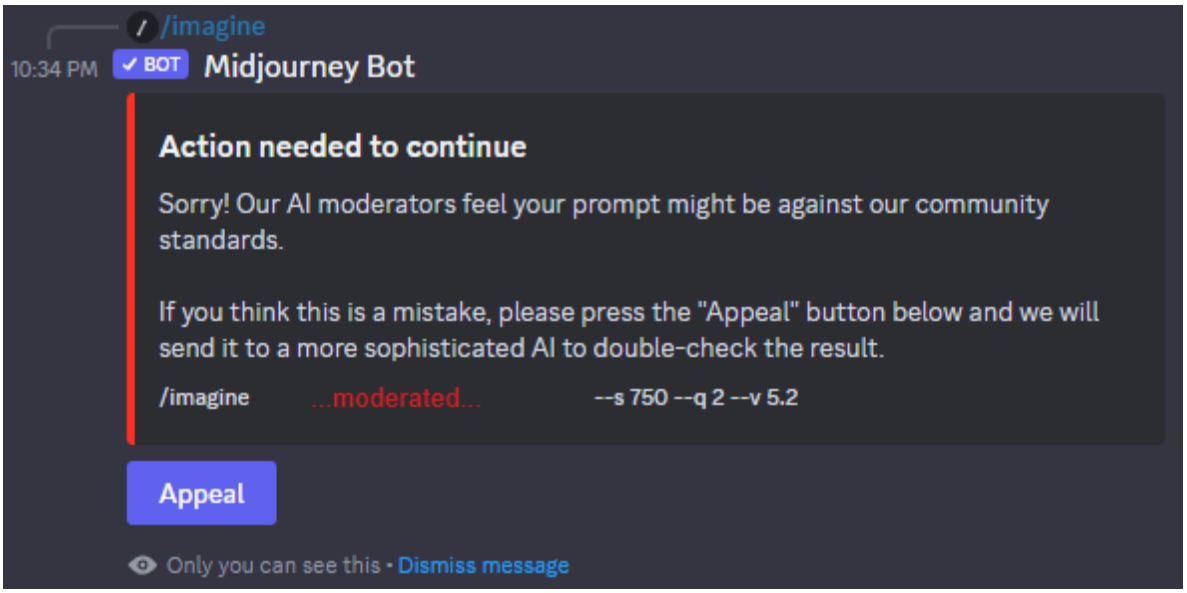


GET https://discord.com/api/v10/channels/channel_id/messages

```
[
 {
  "id": "<Discord final message id>",
  "type": 0,
  "content": "**Wonders of the World…** - <@Discord user id> (fast)",
  "channel_id": "<Discord channel id>",
```

2 **Pre-moderation**

This is the most straightforward case: the posted message will never appear in the channel. You will see something like this.



Messages marked “Only you can see this” can be retrieved by Discord Gateway API[1]

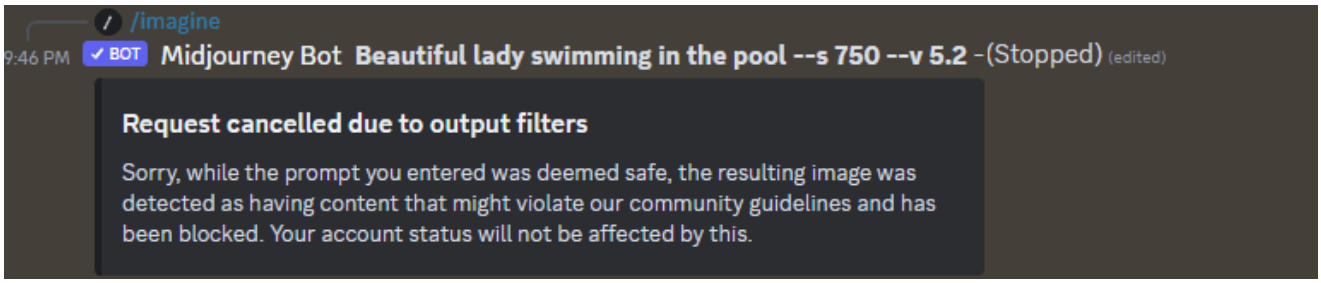
3 Post-moderation

The post will appear in the channel just like in the “Happy path” case.

GET `https://discord.com/api/v10/channels/channel_id/messages`

```
[
 {
  "id": "<Discord original message id>",
  "type": 0,
  "content": "**Beautiful lady swimming in the pool…** - <@Discord user id> (Waiting to start)",
  "channel_id": "<Discord channel id>",
```

Generation may even start, and you will see the progress, but suddenly it will stop with a message similar to the one below.



GET `https://discord.com/api/v10/channels/channel_id/messages`

```
[
 {
  "id": "<Discord original message id>",
  "type": 0,
  "content": "**Beautiful lady swimming in the pool…** - <@Discord user id>
(Stopped)",
  "channel_id": "<Discord channel id>",
```

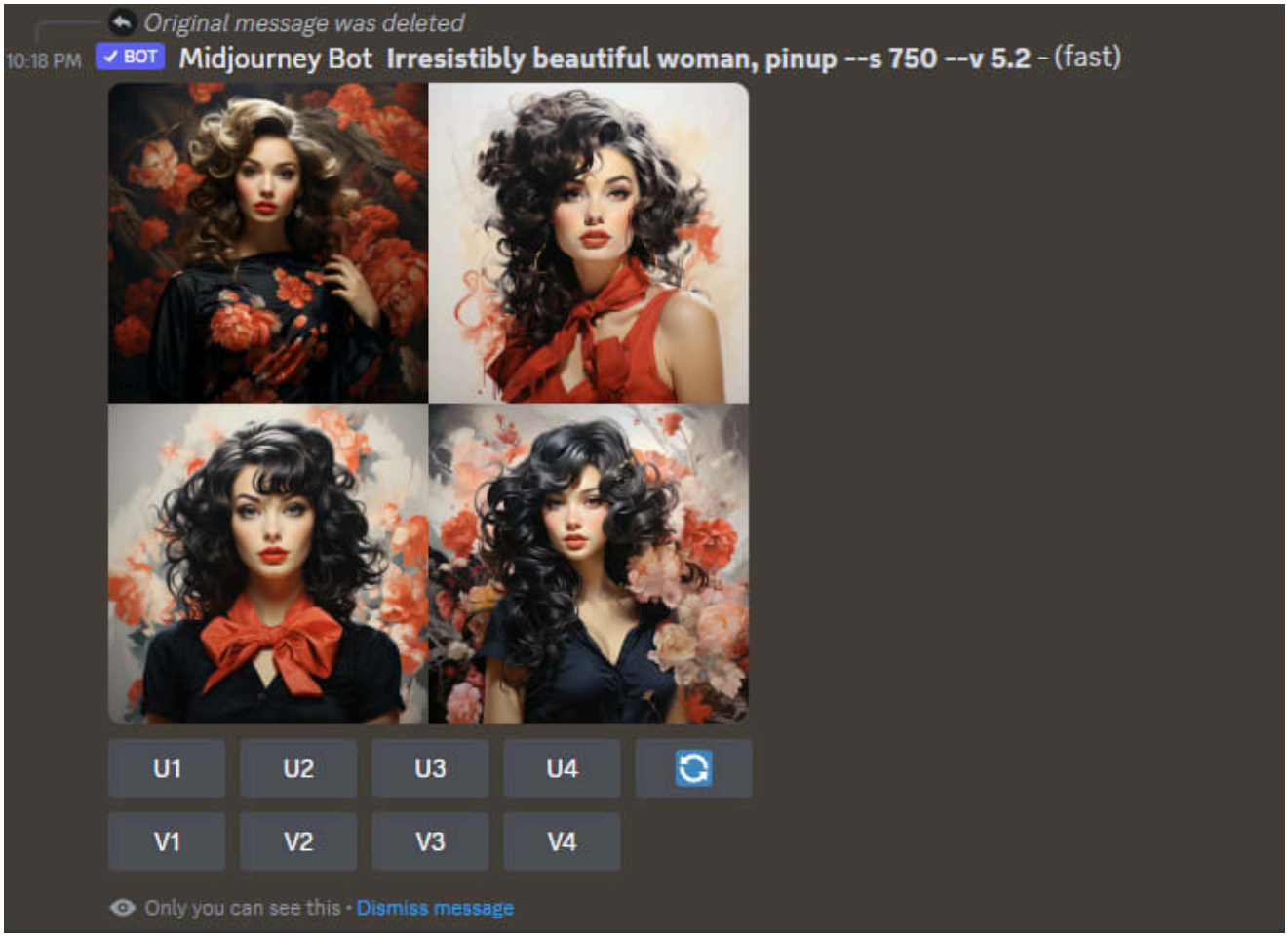
4 Ephemeral moderation

Certain words in your prompt trigger “soft moderation”. Generation starts and completes successfully, but after completion, the message will display “Original message was deleted”. The message is no longer returned by the GET `https://discord.com/api/v10/channels/channel_id/messages` endpoint, even though it was initially present. For a better understanding, refer to the sequence below.

GET `https://discord.com/api/v10/channels/channel_id/messages`

```
[
 {
  "id": "<Discord original message id>",
  "type": 0,
  "content": "**Irresistibly beautiful woman, pinup…** - <@Discord user id> (Waiting
to start)",
  "channel_id": "<Discord channel id>",
```

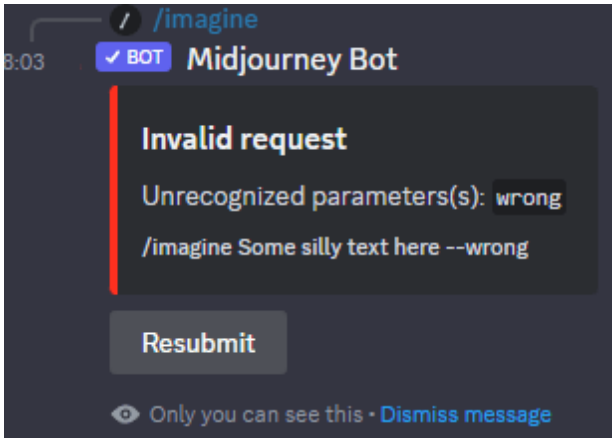
The generated message was deleted.



Messages marked “Only you can see this” can be retrieved by Discord Gateway API[1]

5 **Invalid request**

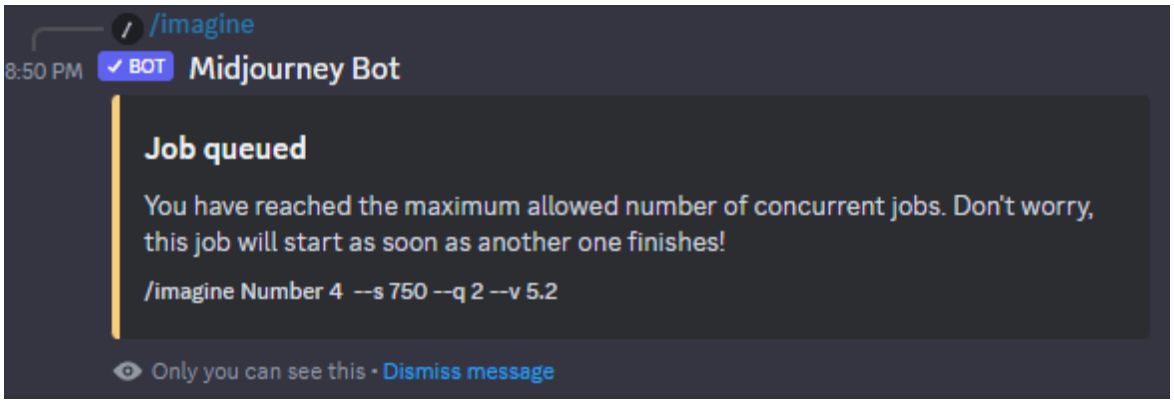
This occurs when you specify an incorrect parameter. Like the “Pre-moderation” case, the posted message will not appear in the channel. It’s advisable to include code for checking the prompt syntax before posting. We plan to cover the topic of prompt syntax verification in one of our follow-up articles.



Messages marked “Only you can see this” can be retrieved by Discord Gateway API[1]

6 **Job queued**

Depending on your Midjourney Subscription Plan, you can have anywhere from 3 (Basic & Standard) to 15 (Pro & Mega) concurrent job executions. Once this limit is reached, job requests will be queued, and messages associated with these jobs will not immediately appear in the GET `https://discord.com/api/v10/channels/channel_id/messages` until Midjourney has completed one of the executing jobs. This essentially means that the job has been placed into Midjourney's internal query.

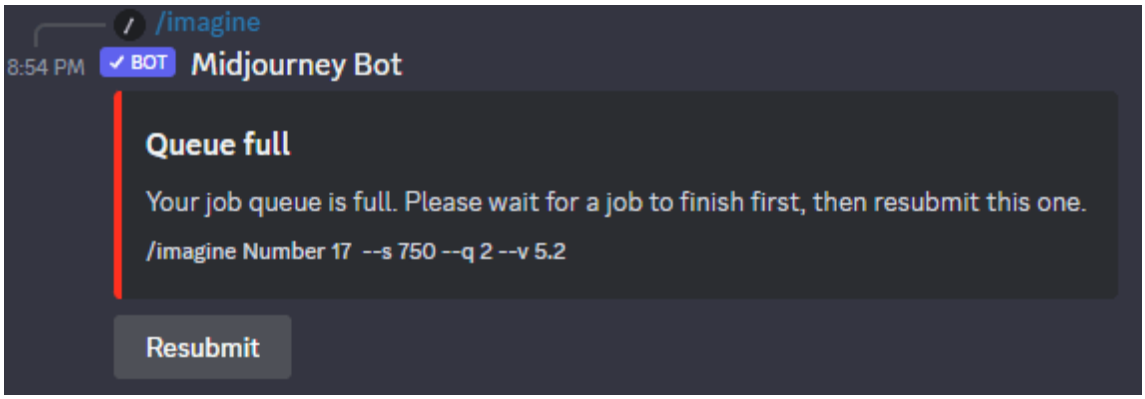


Messages marked "Only you can see this" can be retrieved by Discord Gateway API[1]

You generally want to minimize encountering the "Job queued" status, as there is substantial evidence that Midjourney may throttle your generation requests in such cases and use it as an indicator of a Midjourney ToS violation (…may not use automated tools to access, interact with, or generate Assets through the Services…).

7 **Queue full**

Midjourney job queue can hold up to 10 jobs, after that you will get a "Queue full" message. You will have to wait and try again.



Messages marked "Only you can see this" can be retrieved by Discord Gateway API[1]

You generally want to minimize encountering the “Queue full” status, as there is substantial evidence that Midjourney may throttle your generation requests in such cases and use it as an indicator of a Midjourney ToS violation (…may not use automated tools to access, interact with, or generate Assets through the Services…).

Discord REST API (HTTPS) vs Discord Gateway API (WebSocket)

As you noticed in the previous paragraph, certain messages [1] can still be retrieved using the Discord Gateway API. Unlike the Discord REST API we’re using in this article, the Discord Gateway API requires the use of WebSocket API to facilitate real-time communication between your client application and the Discord server. This adds another level of complexity and, more importantly, may increase the chances of getting banned.

In the vast majority of cases, you can build your logic without the use of the Discord Gateway API. We plan to cover corner cases where the use of the Discord Gateway API is the only feasible solution in later articles.

Midjourney /imagine generation logic

Taking into account what we learned from the previous paragraphs, we can implement the following simple strategy:

1. Ensure there is at least one available concurrent job slot. To achieve this, you can either use a separate Midjourney account for all your automation work or carefully manage the number of manually initiated jobs, keeping them to a maximum of two. This approach will help avoid encountering both “Job queued” and “Queue full” cases.

2. Before posting an image prompt, execute a GET request to `https://discord.com/api/v10/channels/channel_id/messages` to retrieve the `id` of the very first record. By default, this endpoint returns the 50 most recent messages from the channel, sorted in descending order, with the most recent message listed first.

3. After posting the image prompt using a POST request to `https://discord.com/api/v10/interactions`, continuously check the channel every 3 to 5 seconds by making a GET request to `https://discord.com/api/v10/channels/channel_id/messages`. Compare the `id` of the very first record with the value obtained in the previous step. If, after 30 seconds, you do not see a new record in the channel, you can assume you are either dealing with a “Pre-moderation” or “Invalid request” case, and your prompt may require revision.

4. Once you have determined that a new message has indeed appeared in your channel, save the `id` of this message. Continue calling GET requests to `https://discord.com/api/v10/channels/channel_id/messages` every 20 to 30 seconds to check for the following remaining cases:

- Happy path: when the generation is completed, the message with the original `id` is no longer present in the channel. Instead, a new message is present with a non-empty components array. You can obtain the generated image from the `attachments[0].url field`.
- Post-moderation: when the message with the original `id` is still present, and its `content` field ends with `(Stopped)`.
- Ephemeral moderation: when the `id` of the very first record matches the value obtained before posting the image prompt.

Working JavaScript code

```
// bash
// USEAPI_DISCORD="..." USEAPI_SERVER="..." USEAPI_CHANNEL="..." node index.js

// Node 18+

const imagine_prompt = "Hologram cat in neon lights";

const discordAPI = `https://discord.com/api/v10`;
const MidjourneyAppId = `936929561302675456`;

// Load Discord settings from environment
const discord = process.env.USEAPI_DISCORD ?? 'Discord token';
const server = process.env.USEAPI_SERVER ?? 'Discord server id number';
const channel = process.env.USEAPI_CHANNEL ?? 'Discord channel id number';

console.info({ discord, server, channel, imagine_prompt });

const sleep = (ms = 0) => new Promise(resolve => setTimeout(resolve, ms));

const DiscordHeaders = (token) => ({
    "Content-Type": "application/json",
    "Authorization": `${token}`,
});

// https://discord.com/developers/docs/resources/channel#get-channel-messages
const GetDiscordChannelMessages = async (discord, channel) => {
    const response = await fetch(
        `${discordAPI}/channels/${channel}/messages`,
        { headers: DiscordHeaders(discord) });

    return response;
}

// Midjourney Imagine https://discord.com/api/v10/channels/channel_id/application-
commands/search?type=1&include_applications=true&query=imagine
const PostDiscordImagine = async (discord, server, channel, prompt) => {
    const data = {
        "type": 2,
        "application_id": MidjourneyAppId,
        "guild_id": server,
        "channel_id": channel,
        "session_id": (new Date()).getTime(),
        "data": {
            "version": "1118961510123847772",
            "id": "938956540159881230",
            "name": "imagine",
            "type": 1,
            "options": [
                {
                    "type": 3,
```

```
                    "name": "prompt",
                    "value": prompt
                }
            ],
            "application_command": {
                "id": "938956540159881230",
                "application_id": MidjourneyAppId,
                "version": "1118961510123847772",
                "default_permission": true,
                "default_member_permissions": null,
                "type": 1,
                "nsfw": false,
                "name": "imagine",
                "description": "Create images with Midjourney",
                "dm_permission": true,
                "options": [
                    {
                        "type": 3,
                        "name": "prompt",
                        "description": "The prompt to imagine",
                        "required": true
                    }
                ]
            },
            "attachments": []
        }
    };

    const response = await fetch(`${discordAPI}/interactions`, {
        method: "POST",
        body: JSON.stringify(data),
        headers: DiscordHeaders(discord)
    });

    return response;
}

const demo = async () => {
    const getBeforeMessages = await GetDiscordChannelMessages(discord, channel);

    if (getBeforeMessages.status !== 200) {
        console.error(`Discord /messages status ${getBeforeMessages.status}`, await
getBeforeMessages.json());
        process.exit(1);
    }

    const beforeMessages = await getBeforeMessages.json();

    const beforeIds = new Set();
    beforeMessages.forEach(msg => beforeIds.add(msg.id));

    await sleep(350);
```

```
    const postImagine = await PostDiscordImagine(discord, server, channel,
imagine_prompt);

    if (postImagine.status != 204) {
        console.error(`Discord /interactions status ${postImagine.status}`, await
postImagine.json());
        process.exit(1);
    }

    const maxPostedAttempts = 10;

    let attempt = 1;
    let postedMessage;

    // Check for message to appear in the channel for 20 seconds total
    do {
        await sleep(2000);

        const getMessages = await GetDiscordChannelMessages(discord, channel);

        if (getMessages.status !== 200) {
            console.error(`Discord /messages status ${getMessages.status}`, await
getMessages.body());
            process.exit(1);
        }

        const messages = await getMessages.json();

        // New (Waiting to start) imagine interaction
        postedMessage = messages.find(msg => !beforeIds.has(msg.id));

        if (postedMessage) {
            console.log(`Found new message ${postedMessage.id}`,
postedMessage.content);
            break;
        }

        attempt++;

    } while (attempt <= maxPostedAttempts);

    if (postedMessage === undefined) {
        console.error(`Posted message not found due to moderation or an invalid
prompt`);
        process.exit(1);
    }

    const termStopped = `> (Stopped)`;
    const maxGeneratedAttempts = 60;

    attempt = 1;
```

```
    let generatedMessage;

    // Check for posted message to appear in the channel with new id for 10 minutes
max
    do {
        // Wait for 10 seconds before checking on message progress
        await sleep(10000);

        const getMessages = await GetDiscordChannelMessages(discord, channel);

        if (getMessages.status !== 200) {
            console.error(`Discord /messages status ${getMessages.status}`, await
getMessages.body());
            process.exit(1);
        }

        const messages = await getMessages.json();

        const progress = messages.find(message =>
            message.id == postedMessage.id &&
            // Not completed
            !message.components?.length &&
            // Not stopped
            !message.content.endsWith(termStopped));
        if (progress) {
            console.log(`#${attempt} ${progress.id} progress`, progress.content);
        } else {
            const completed = messages.find(message =>
                !beforeIds.has(message.id) &&
                message.id !== postedMessage.id &&
                // Either completed or stopped
                (!!message.components?.length ||
message.content.endsWith(termStopped))
            );

            generatedMessage = (completed && !!completed.components?.length) ?
completed : undefined;

            break;
        }

        attempt++;

    } while (attempt <= maxGeneratedAttempts);

    if (generatedMessage === undefined) {
        console.error(`Message not found due to post-moderation or ephemeral
moderation`);
        process.exit(1);
    }
```

```
    // Successful generation
    console.info(`Completed`, generatedMessage.content);
    console.info(`Download URL`, generatedMessage.attachments[0].url);
}

demo();
```

Cross posted

- hashnode.dev
- dev.to
- medium.com
- habr.com