**Software Requirements Specification**

**for**

**<**VibeCheque**>**

**Version 1.0 approved**

**Prepared by <Izge Bayyurt, Liam Hillery, Aiden McGlauflin, Noah Moring, Bryan Sturdivant>**

**<COS-420: Team FireTruck>**

**<10/08/2024>**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 Functional Requirements 4

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| Initial Version | 10/09/24 | Initial | 0.0.1 |
| Updated Document | 10/21/24 | Updated to Finish first draft of Document | 0.0.2 |

# **Introduction**

## **Purpose**

VibeCheque is a Discord bot designed to address the challenges of emotional miscommunication in text-based conversations by leveraging artificial intelligence to analyze and convey the emotional tone of messages. The AI-powered Discord bot will utilize natural language processing and generative AI to bridge the gap between intended and perceived emotions in text-based interactions. This document is intended to describe the software requirements to create the Discord Bot.

## **Document Conventions**

This document was created based on the IEEE template for System Requirement Specification Documents.

The following conventions are used as follows:

| **Convention** | **Description** |
| --- | --- |
| VC | VibeCheque |

## **Intended Audience and Reading Suggestions**

This document is intended for the product manager, developers, designer, product users, and testers. The document is organized to provide an overview of the product, followed by detailed functional and non-functional requirements. It is recommended that readers start with the introduction and overall description sections before moving on to the more technical details.

## **Product Scope**

VibeCheque is a Discord bot designed to improve emotional clarity in text-based communication by helping users convey tone and emotion more effectively. The bot uses AI to assist in detecting the tone of messages and offers options to enhance communication through means like color and sound. The primary objective of VibeCheque is to reduce misunderstandings in digital conversations and to improve personal connections between users. Targeting everyday users on Discord, VibeCheque will initially focus on providing its services exclusively within the Discord platform. Leveraging AI to address the challenges of emotional miscommunication, the bot aims to enhance the quality of interactions within various Discord communities.

## **References**

VibeCheque Github Link: <https://github.com/COS420-Fall24/Team-F.git>

Useful Resources:

* UI Web Package: <https://v2.chakra-ui.com/getting-started>
* Discord Developer Portal: <https://discord.com/developers/applications>
* Design/Mockup Website:<https://www.figma.com/>

# **Overall Description**

## **Product Perspective**

VibeCheque is a new, self-contained product designed to function as a Discord bot aimed at improving emotional clarity in text-based communication. It is not part of an existing product or system, but has been developed independently to address the specific need for improved emotional tone recognition in digital conversations. VibeCheque integrates directly with the Discord platform, leveraging Discord’s existing messaging system to detect tones in user messages by using AI.

The bot interacts solely with the Discord environment, with no current plans to expand to other platforms. All communication and functionality are contained within Discord’s servers, channels, and direct messages. Users will interact with the bot through commands and prompts provided within the discord interface, and the bot will provide tone feedback via colors, sounds, or vibrations in response to messages.

All of this is meant to enhance personal communication by providing clarity to

## **Product Functions**

* Tone Detection
* Tone Indicator Suggestions
* Customizable Message Enhancements
* Manual Tone Tagging
* Mood Check-ins
* User Commands
* Mood Clarification

## **User Classes and Characteristics**

* Casual Users:
  + **Description**: These users are individuals who use Discord for general communication, such as chatting with friends, gaming, or casual community interactions. These users might not have much technical expertise and will use VibeCheque primarily for enhancing everyday conversations.
  + **Frequency of use**: Regular, depending on how often they use Discord.
  + **Functions Used**: Tone detection, customizable message enhancements, and manual tone tagging.
  + **Technical Expertise**: Basic; they should be comfortable with Discord but may not have advanced technical knowledge past that.
  + **Importance**: High - Casual users will make up the majority of VibeCheque’s user base.
* Moderators:
  + **Description:** These users manage Discord servers and communities. They are responsible for setting community standards and monitoring interactions.
  + **Frequency of Use:** Regular to frequent, as they manage multiple conversations and users.
  + **Functions Used:** Tone detection and mood check-ins to help moderate communication, potentially configuring bot settings for their community.
  + **Technical Expertise:** Moderate; familiar with Discord’s server management and bot customization tools.
  + **Importance:** Medium - Important for enabling smooth integration of VibeCheque into communities and ensuring the bot supports community guidelines.
* Administrators**:** 
  + **Description:** Administrators will have access to VibeCheque’s internal settings and configurations. They are responsible for maintaining the bot and managing user privileges.
  + **Frequency of Use:** Occasional; mainly when configuring or troubleshooting the bot.
  + **Functions Used:** Bot configuration, user permission settings, system monitoring.
  + **Technical Expertise:** High; must be familiar with the bot’s setup and functionality as well as Discord’s administrative tools.
  + **Importance:** Low - Important for the initial set up, but their role will diminish after installation.

## **Operating Environment**

VibeCheque will operate within the Discord platform, interacting with Discord’s messaging API and user interface. The bot will be developed using Typescript, running a Node.js environment. The bot will be hosted on a server capable of running Node.js and will require access to the internet to communicate with Discord servers.

## **Design and Implementation Constraints**

VibeCheque is designed exclusively for Discord, limiting the bot’s use to just this platform. Discord also enforces rate limits on its API, which will restrict the number of actions the bot can perform within a given time frame. Tone detection will rely on AI models, which could have limitations in their accuracy, especially when it comes to more ambiguous tones.

## **User Documentation**

VibeCheque will have a landing page that will contain helpful resources for installing and using the bot. We also plan to have help commands that will be accessible directly from Discord, so users will only have to type commands to receive quick instructions of specific bot functions.

## **Assumptions and Dependencies**

In order for VibeCheque to work there will have to be platform stability on Discords end. Users and server administrators will need to grant VibeCheque the necessary permissions for proper operation. Any restrictions on these permissions could limit the bot’s effectiveness. The bot will also rely on third-party AI and machine learning models. These models or APIs will need to be available and functional for the bot to work correctly. Changes in accuracy or availability will limit the bot’s performance.

# **External Interface Requirements**

## **User Interfaces**

Sample messages from a bot:

* All samples not using the bot are work-in-progress

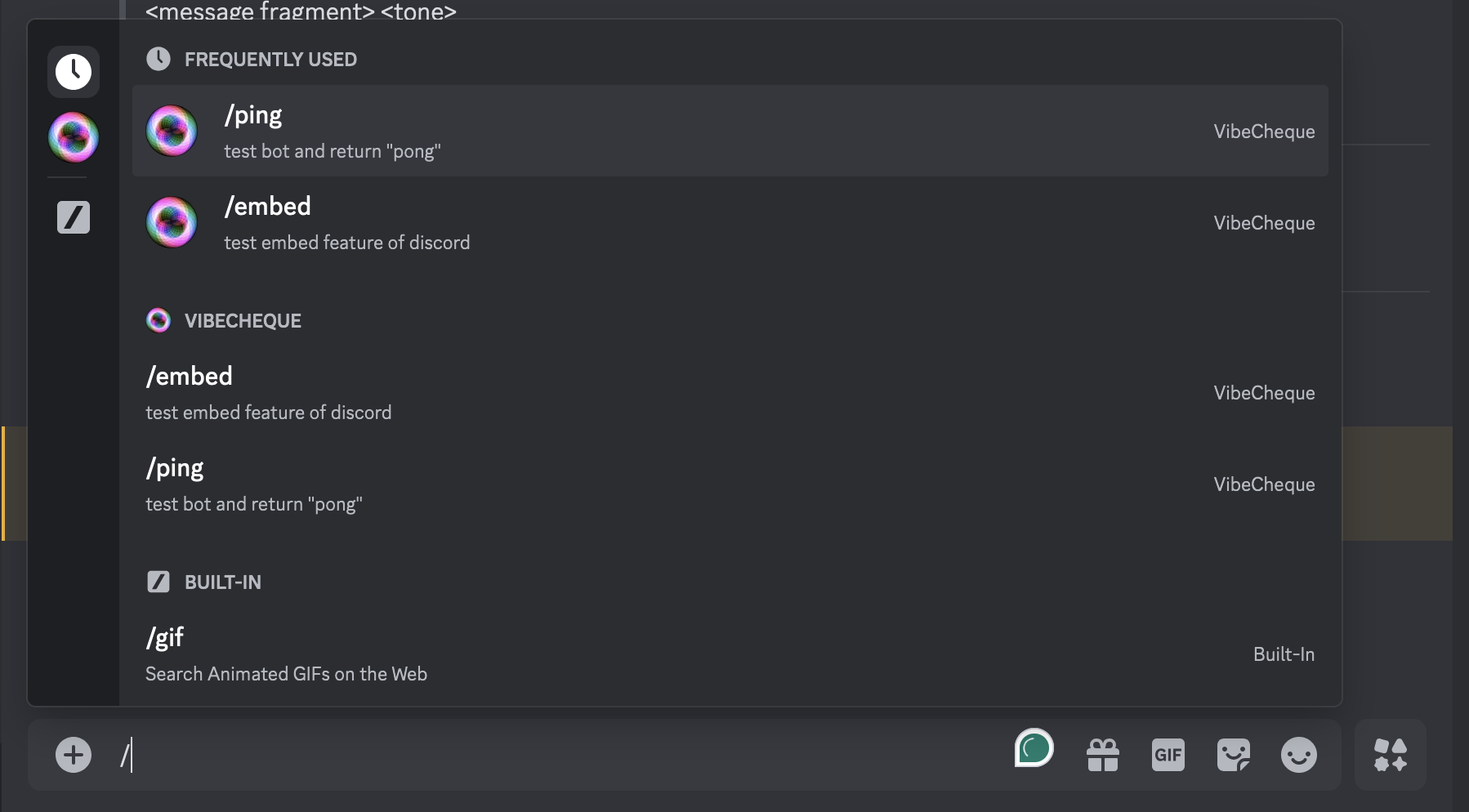


Figure 3.1a-1: The UI response when a user types “/”. The UI shows all available (bot and Discord) slash commands. The slash commands with “VibeCheque” on the right are the bot-specific slash commands. Any command referred to as a “slash command” is accessed in this way.

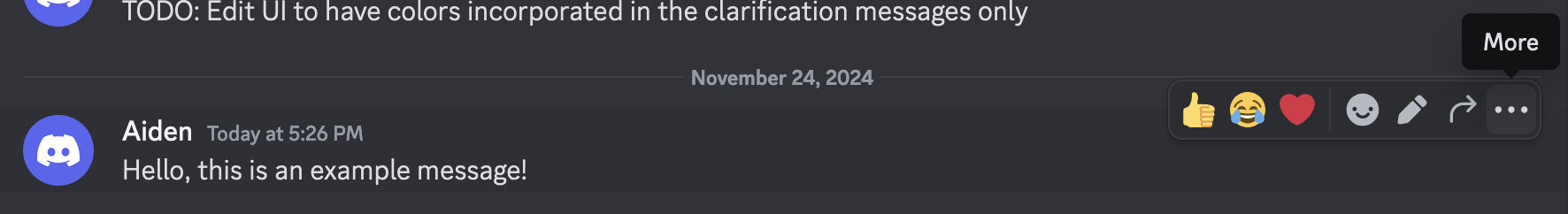


Figure 3.1a-2: When a user hovers over a message, this UI pop-up appears. When more is selected, the result is pictured below in Figure 3.1a-3.

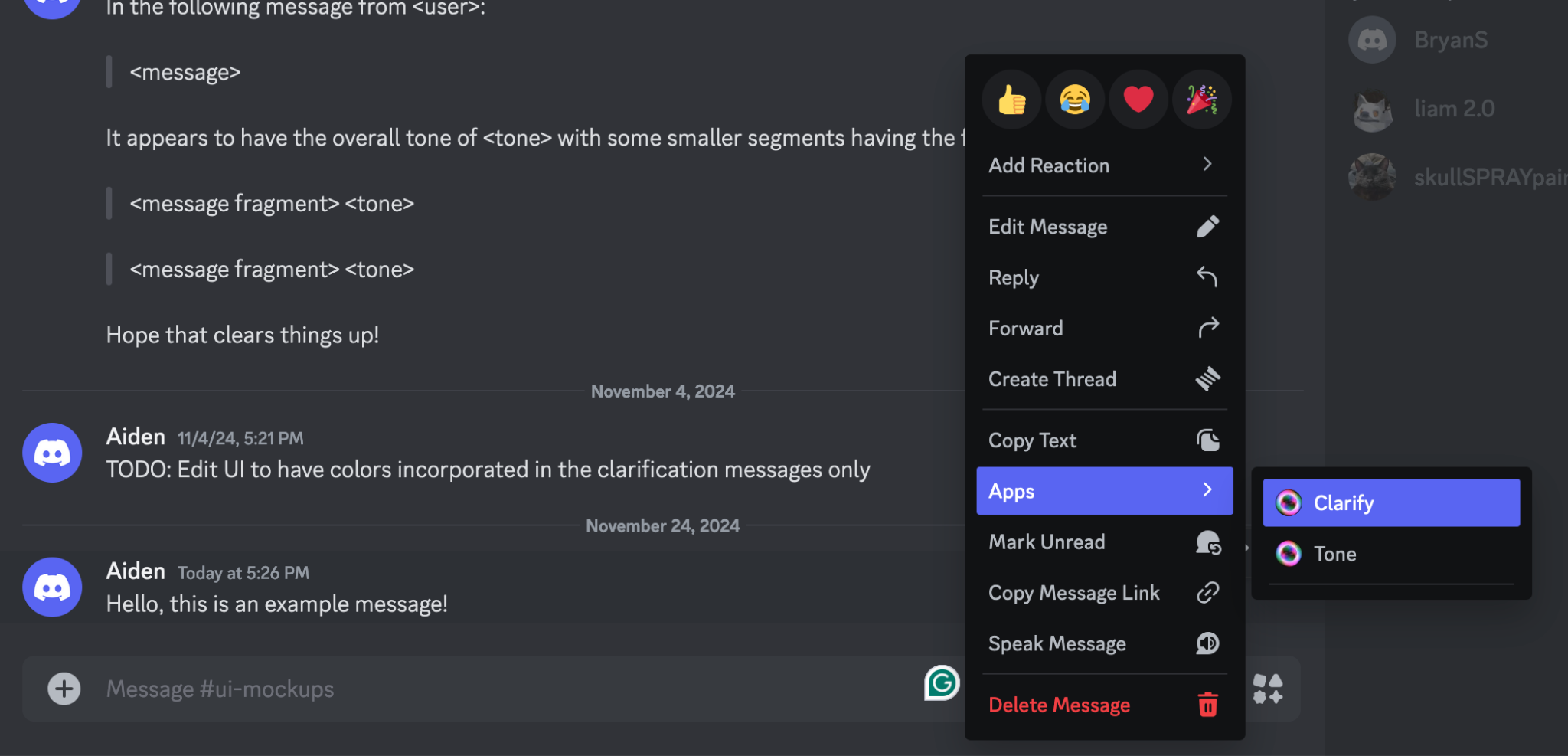
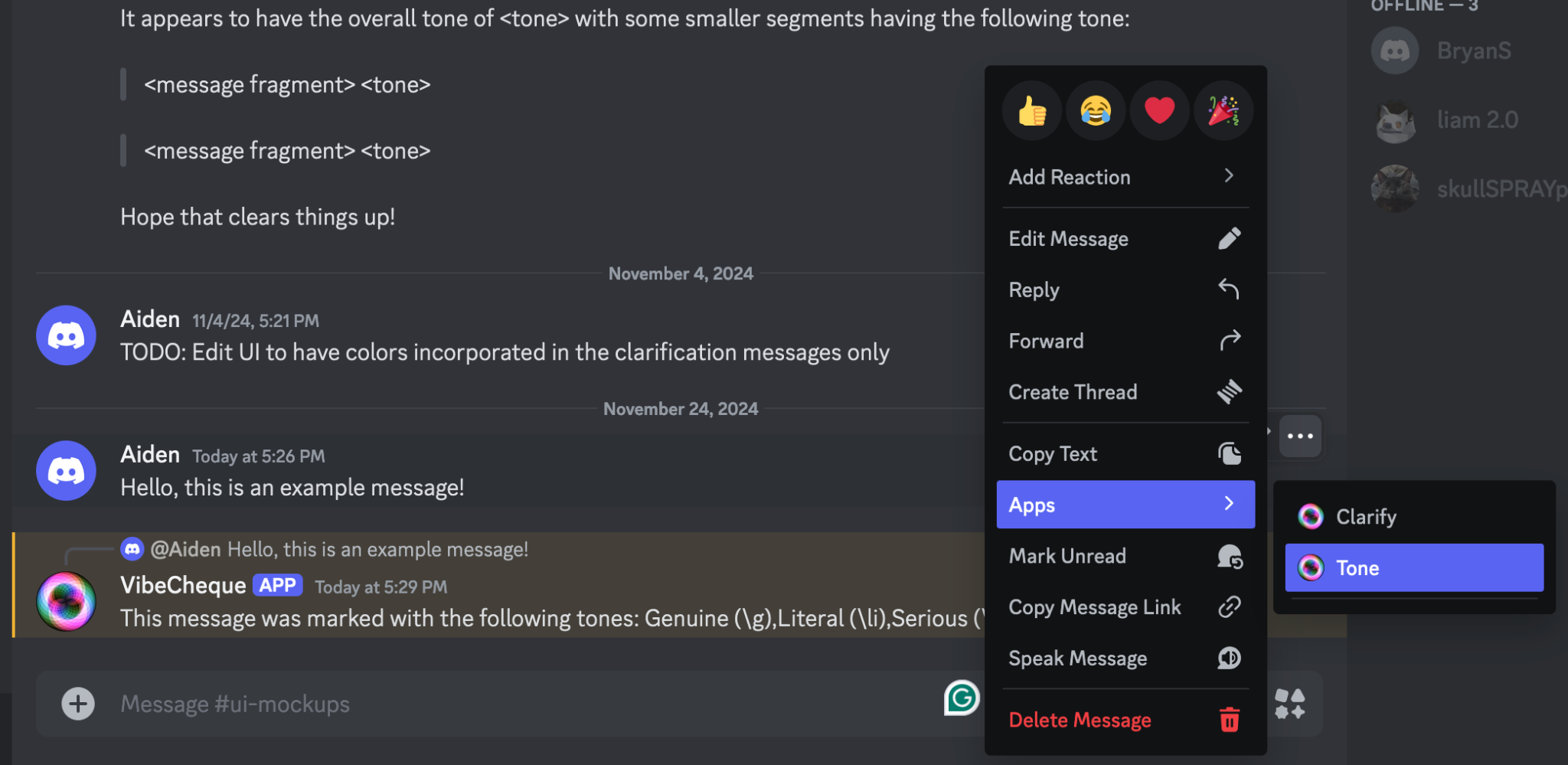


Figure 3.1a-3: The expanded “more” menu has an option for Apps. In this section are the application commands the user can select. Any command referred to as an “application command” or “app command” is accessed through this menu.

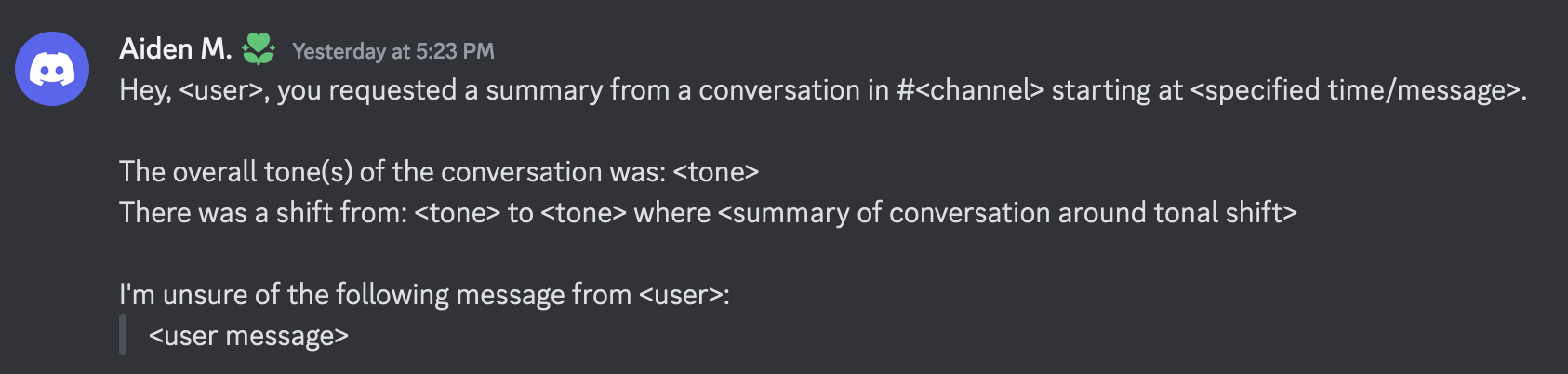


Figure 3.1b: A sample message depicting a bot response to a user requesting a conversational tone summary. The message starts with a greeting to the user, the channel of the conversation, and the starting message or starting time (depending on which the user specified). After the greeting, the overall tone is indicated and any tonal shifts in the conversation with a summary of the messages around the tonal shift. Finally, there’s any messages from specified users where the bot was unable to determine the tone. The unsure messages should be included with any tonal shifts if the occur together.

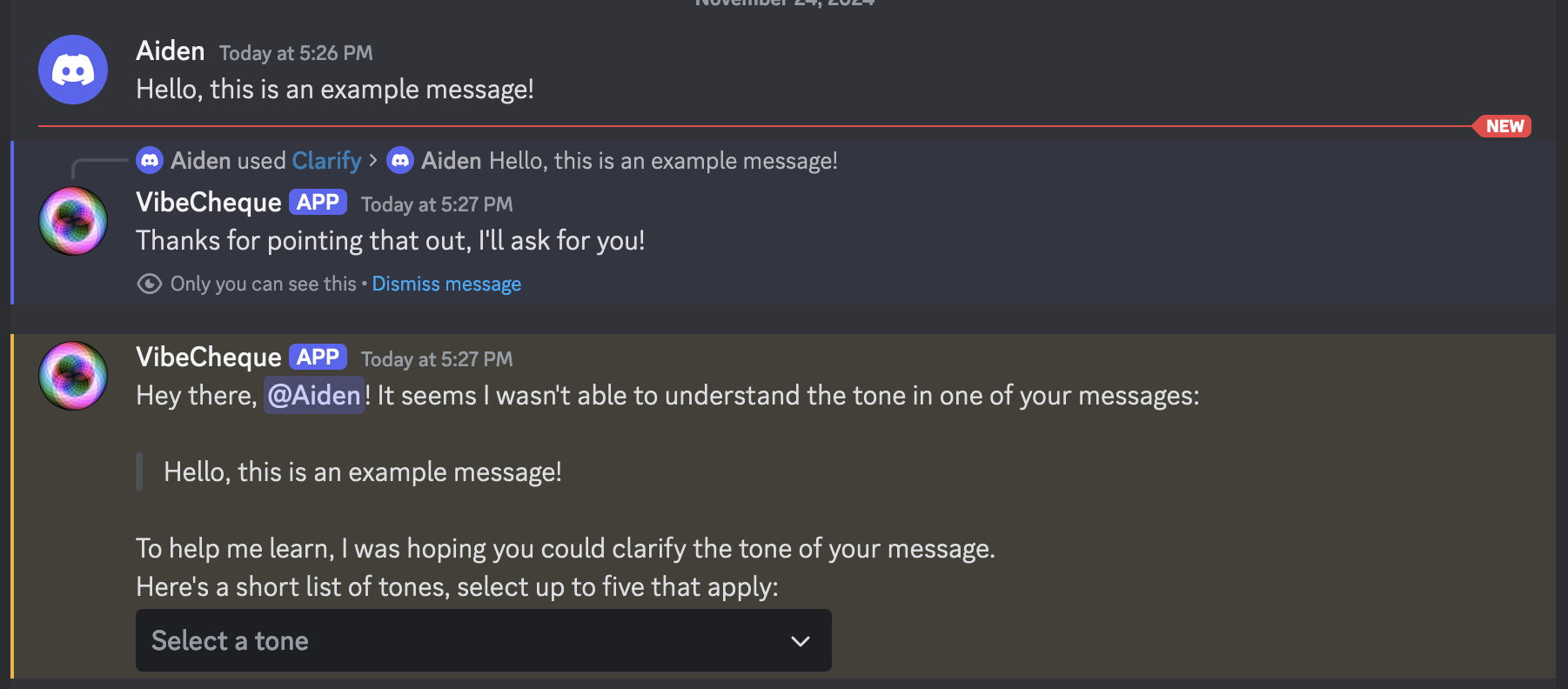


Figure 3.1c-1: The result of running the “Clarify” app command. The first new message is an ephemeral message sent to the user who requested clarification. The ephemeral message will disappear after Dismiss is hit. The next message is sent to the original poster of the first message requesting them to select up to five tones that apply to their message.

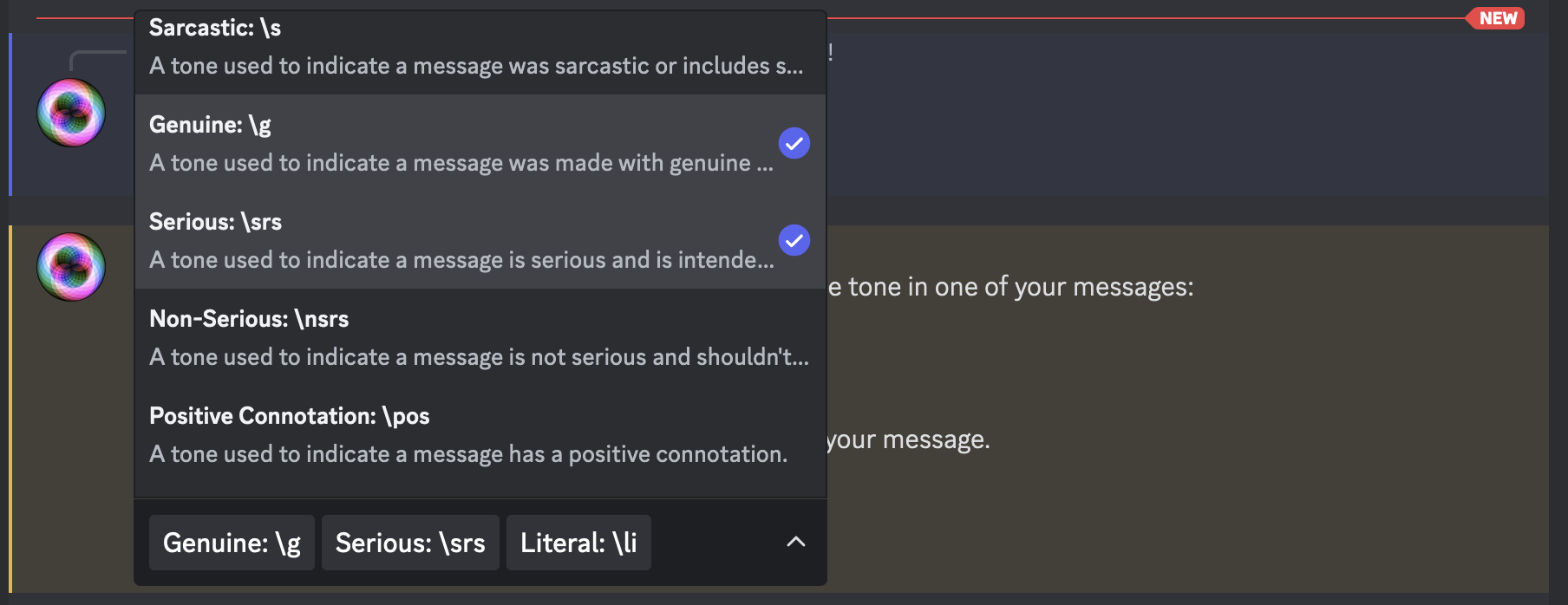
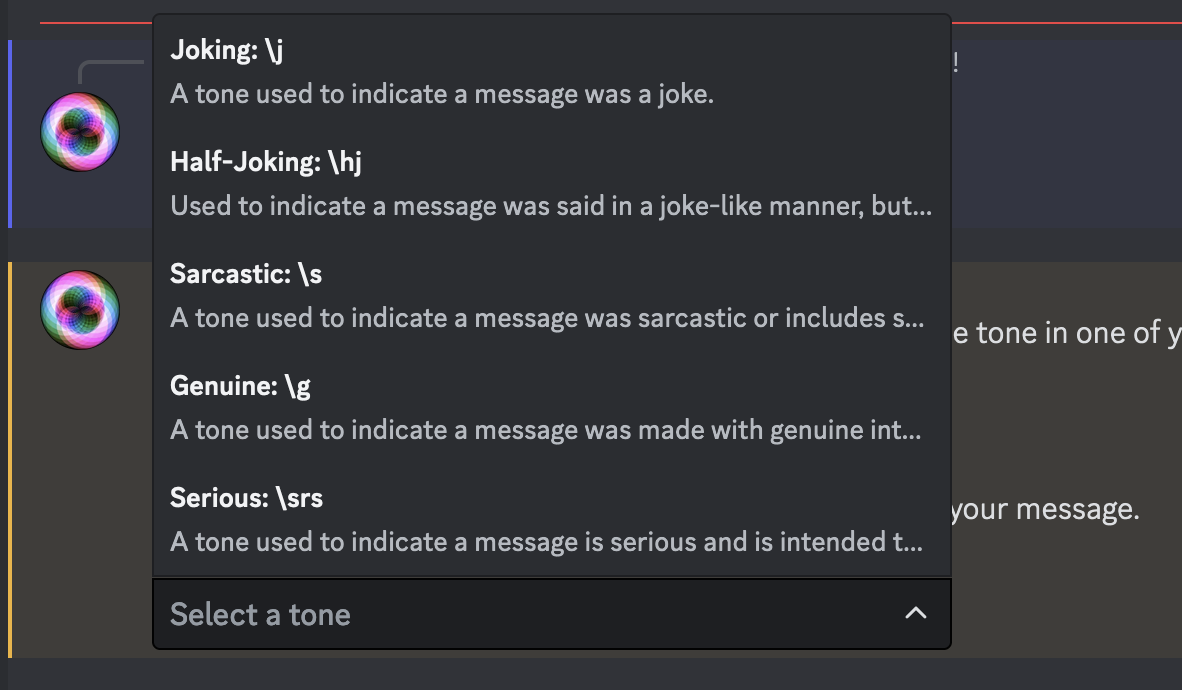


Figure 3.1c-2: The list of tones and their description in the StringSelectMenu object of the discord.js UI. The menu contains all tones and will only close when the user clicks anywhere outside the menu. There’s a filter on the menu that only accepts input from the user of the original message to prevent others from answering.

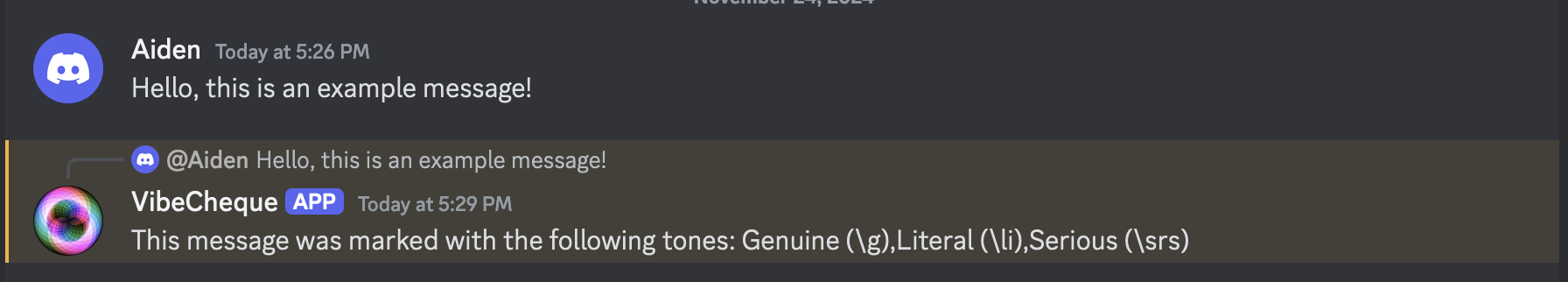


Figure 3.1c-3: The final result of the user requesting clarification after the original poster provides tone. The bot’s clarification message is deleted, and the bot replies to the original message with the tones the poster selected.

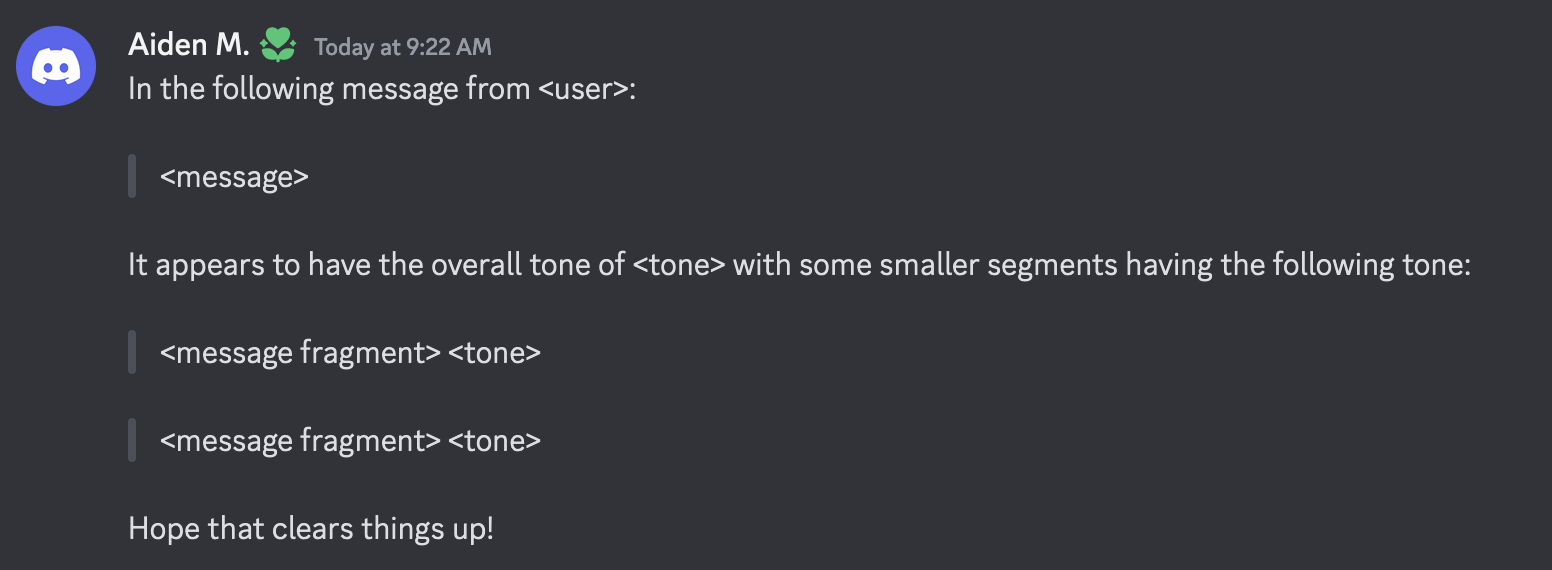


Figure 3.1d: A sample message depicting a message to a user who asked clarification from another user’s message. The message starts by naming the user and then their message. The bot then writes the overall message tone and some smaller segments with a different tone.

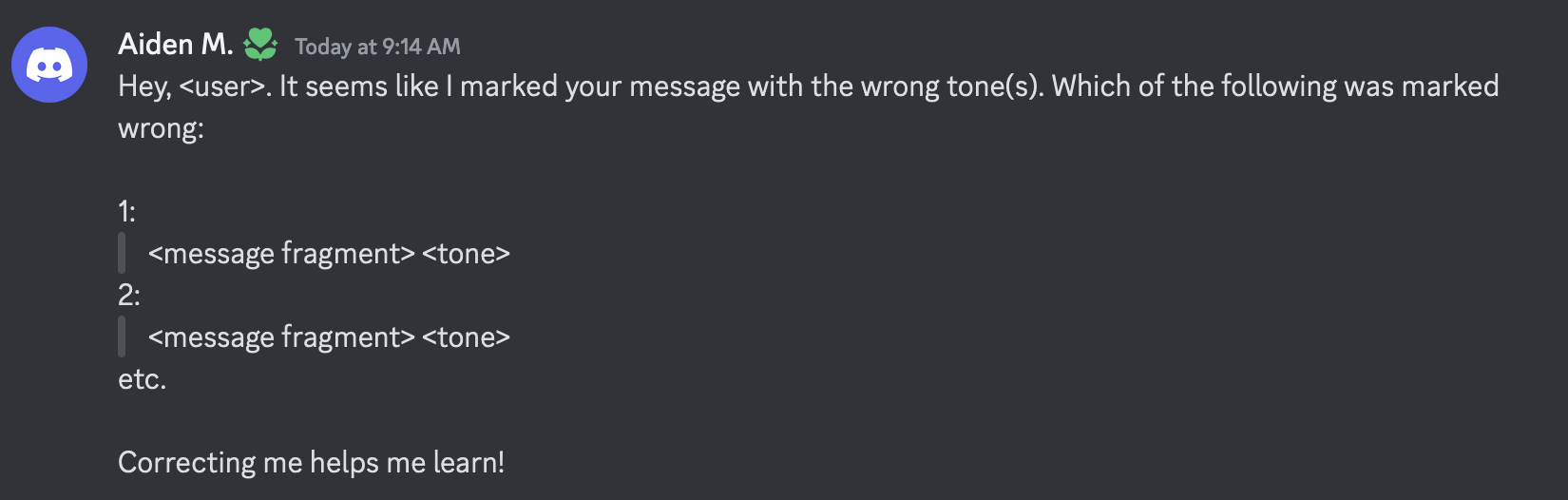


Figure 3.1e: A sample message depicting a situation where the bot marked a user’s message with the wrong tone. If the message has multiple tone indicators, it will list the message fragment with their respective tone, otherwise, it will just show the tone of the message. The user should then respond with the message and the corrected tone.

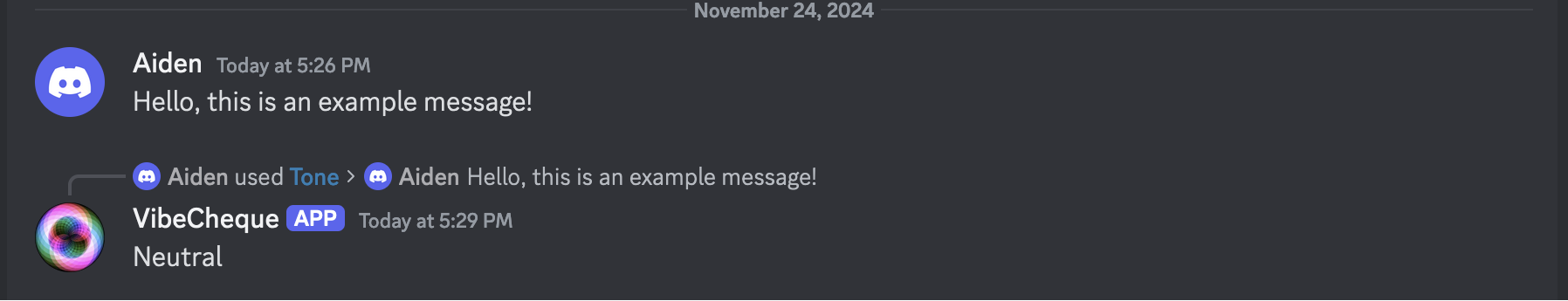


Figure 3.1f: The result of the “Tone” app command. The bot makes an OpenAI API call to generate a one or two-word response of a message’s tone.

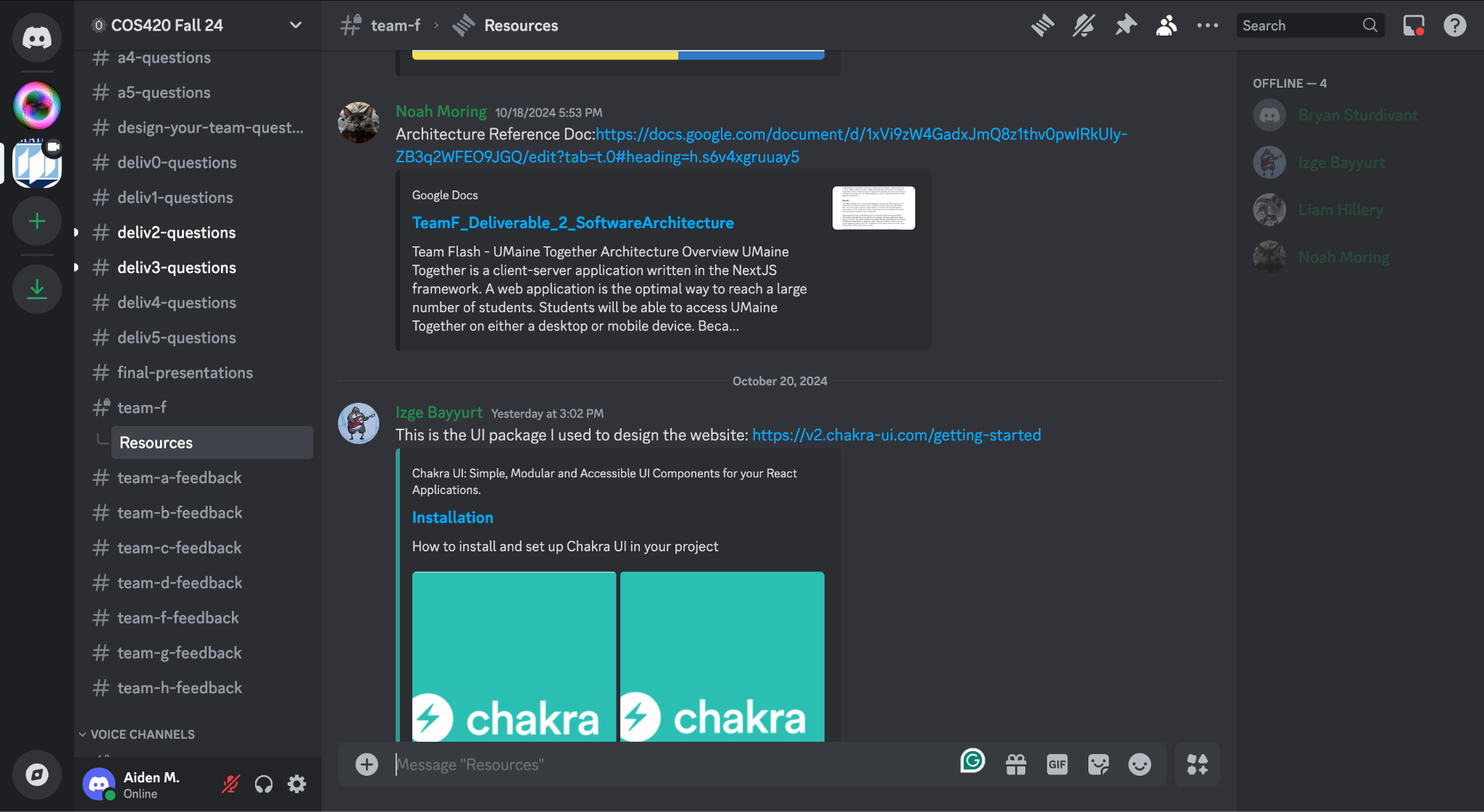


Figure 3.1g: Discord’s UI. (From left to right) The list of the user’s servers, the server the user currently is in and its channels, the messaging interface showing message history and the message field, and the list of users and their current status.

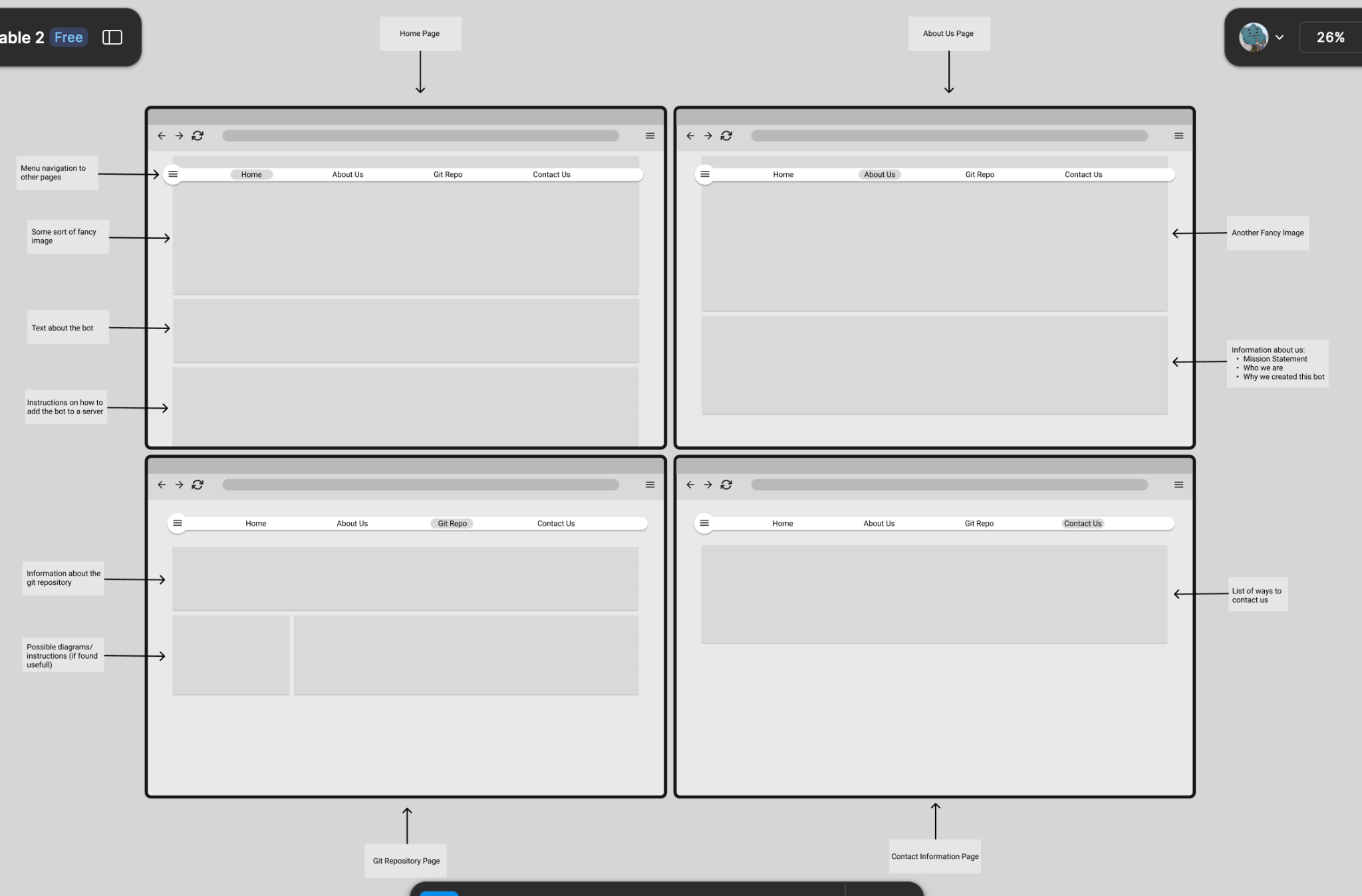


Figure 3.1h: The landing page for the Discord bot. On the top of each page is a navigation menu with a button for each of the four pages on the site. (Top-left) Home page: The home page will contain information about the bot and an installation guide. (Top-right) About us page: The about us page will contain information about the team and why we decided to do this project. (Bottom-left) Git repository (repo) page: The git repo page will contain all necessary information about the git repo. (Bottom-right) Contact us page: The contact us page will contain the contact information and the potential restriction about contacting us (we’re students, don’t expect an immediate response, etc).

## **Hardware Interfaces**

* Supported devices: Anything that can run discord
* A server to support our bot
* HTTP

## **Software Interfaces**

* OpenAI:
* Chakra-ui:
* React:
* Firebase:

A discord account is necessary, and the bot must be in the server for users to interact with it. Users will use slash-commands and messages to interact with the bot.

## **Communications Interfaces**

* HTTP
* Discord
* Discord Message -> API request

# **System Features**

*<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>*

## Functional Requirements

1. **Accessibility:** The system should allow users to specify how they want to receive the tone indicator: color, sound, text indicator.
2. **Integration:** The system shall use the Discord API to interact with the users through the Discord client.
3. **Message Misinterpret Alert:** The system will DM the user when a message they sent may be misinterpreted emotionally. This will allow the user to rephrase for better clarity.
4. **Tone Indicators:** The system will allow users to add easily interpreted tone indicators to their messages to clarify the emotional tone of their writing.
5. **Share Emotion:** The system will allow users to share how they are feeling as a mood update. This update shall overwrite the existing mood of the user, and the mood that is shared shall be visible to other users in the server/DM.
6. **Emotional Compatibility:** The system will check if another user is in a similar mood before connecting with them.
7. **Anonymous Mood Clarification:** The system will allow users to ask the bot to analyze another user’s mood or tone anonymously through message commands, so no one needs to feel awkward.
8. **User Emotional Profile:** The users will be able to access a history of their emotions set by the Share Emotion feature.
9. **Mood Check-in after a period of inactivity:** The system will check in with the user via a mood-check in prompt sent through a DM to update the user's current mood, after a certain period of inactivity.
10. **More or Less Mood Check-ins:** The system will allow users to change how often the mood check-ins occur.
11. **Pre-existing Mood Choices:** The system will allow users to select a mood from a preset of moods when they use Share Emotion functionality.
12. **Tone Indicator Description:** The system will allow users to get a text or audio description of each tone indicator so users can quickly and easily understand another user’s intentions.
13. **Easy to Distinguish Tone Indicator:** The system will provide definitions of tone indicators with example use cases, such that users can choose the appropriate tone for their message.
14. **Custom tone indicators:** The system will allow users to create custom tone indicators which can be seen by other users. The system will allow users to add descriptions to clarify their tone indicators.

# **Other Nonfunctional Requirements**

## **Performance Requirements**

1. **Speed:** VC bot should be able to detect and add a tone within 5 seconds unless service disruptions are present.
2. **Scalability:** App/Bot should be able to work on a minimum 10 servers simultaneously without compromising on other performance requirements.

## **Safety Requirements**

1. **Reporting:** Reports about the bot should be investigated within 2 business days. If the report cannot be resolved within that time frame, the reporter should be notified of the status.

## **Security Requirements**

1. **Security:** The app should use encryption to handle identifiable data.

## **Software Quality Attributes**

1. **Accessibility:** The VC bot must provide accessibility options that comply with the Web Content Accessibility Guidelines(WCAG) to ensure all users can utilize the mood/tone indicators effectively.
2. **Usability:** The bot should provide documentation on how to use the features, and all the features of the bot should be accessible within the Discord client.
3. **Reliability:** The bot must maintain a 99% uptime.
4. **Responsiveness:** The bot should be responsive to actions.
5. **Compatibility:** The bot’s landing page should run on all browsers based on Gecko, WebKit, and Blink web engines.
6. **Mobile Friendliness:** The bot’s landing page should be able to run on mobile browsers.

## **Business Rules:**

# **Other Requirements**

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

**Appendix A: Glossary**

Discord: Commonly used messaging platform.

Discord.js: The javascript library used by discord bots and applications.

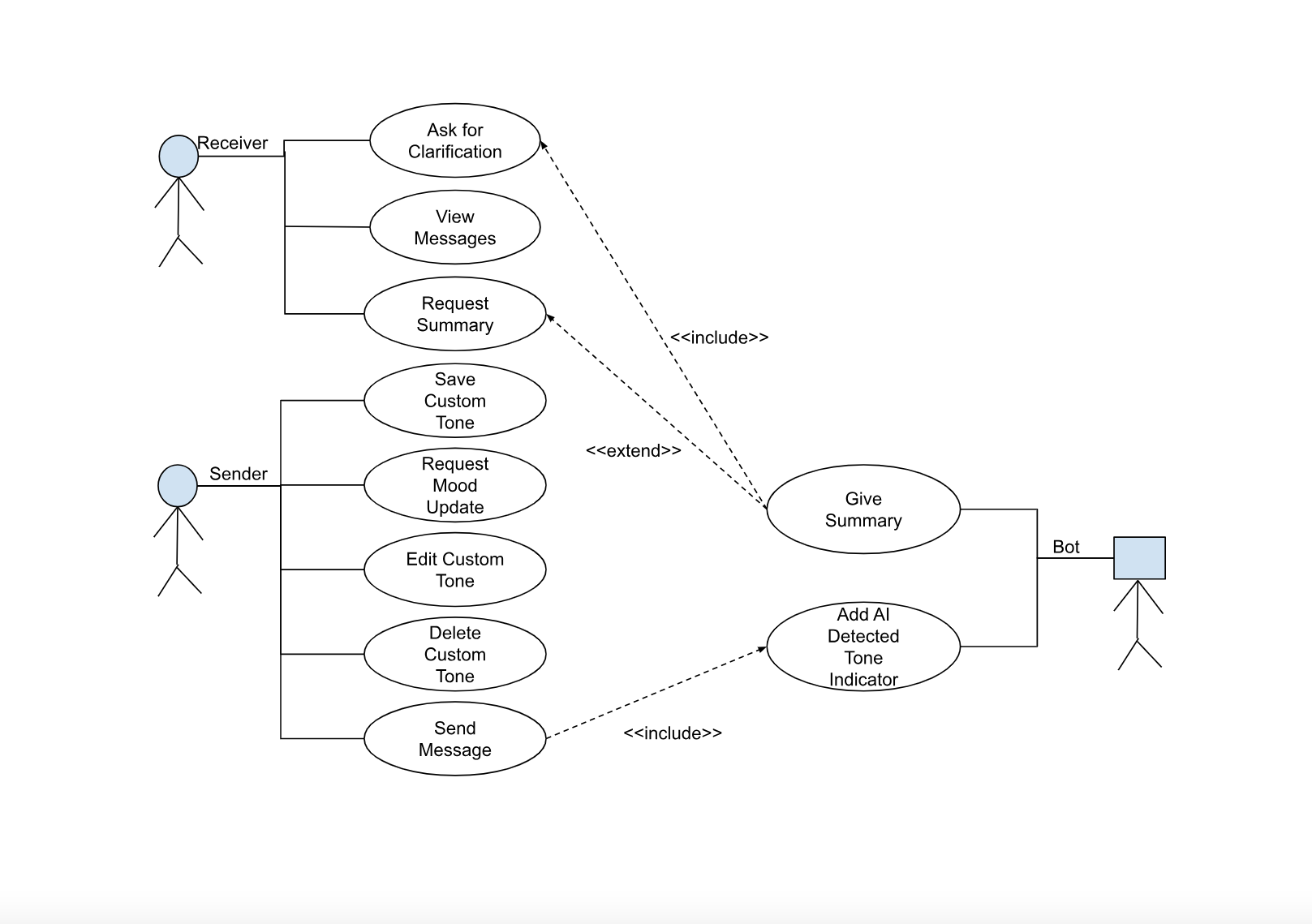
Discord Bot: A discord account used to complete automated account.

Firebase: A backend cloud-computing and application development platform used to help applications scale.

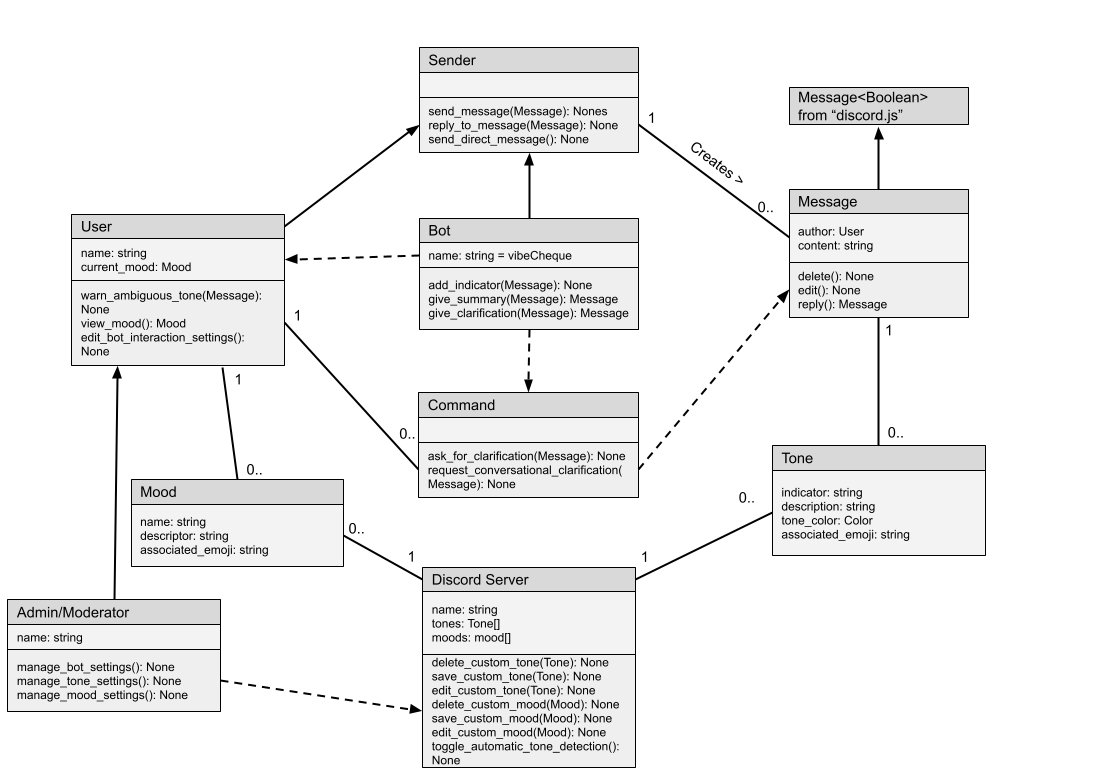
Chakra UI: A react component library used to help developers create user interfaces and web applications.

OpenAI: An API, created by a company of the same name, that enables AI implementation.

**Appendix B: Analysis Models**



(Figure AB.1): Use-Case Diagram

**

(Figure AB.2): Domain Model