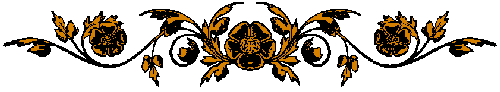
**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN MÔN NGÔN NGỮ LẬP TRÌNH PYTHON**

**ĐỀ TÀI: XÂY DỰNG BOT DISCORD QUẢN LÍ SERVER**

**Nhóm 01 - Họ tên và mã số sinh viên thực hiện đồ án**

**Phan Duy - 3121410003**

**Nguyễn Anh Danh - 3121410103**

**Đỗ Nguyễn Hoàng Tuấn – 3121410554**

**Văn Phú Hiếu - 3121410201**

**Giảng viên hướng dẫn: Trịnh Tấn Đạt**

**Thành phố Hồ Chí Minh – Tháng 5/2023**

**NHẬN XÉT, ĐÁNH GIÁ CỦA GIẢNG VIÊN**

….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**LỜI CẢM ƠN**

Chúng em xin gửi lời cảm ơn chân thành nhất đối với các thầy cô ở khoa Công Nghệ Thông Tin, trường Đại học Sài Gòn đã tạo điều kiện cho chúng em tiếp cận và tìm hiểu để hoàn thành đồ án môn học lần này. Và chúng em cũng xin chân thành cảm ơn thầy Trịnh Tấn Đạt giáo viên giảng dạy đã nhiệt tình hướng dẫn chúng em hoàn thành đồ án lần này.

Trong quá trình thực hiện nghiên cứu và thực hiện làm báo cáo đồ án, do kinh nghiệm thực tế chưa được nhiều, nên bài báo cáo của chúng em có thể vẫn còn những thiếu sót và chưa được hoàn chỉnh nên mong rằng chúng em sẽ nhận được những đóng góp ý kiến đóng góp bổ ích từ thầy để chúng em có thể khắc phục cho những bài báo cáo sau.

Chúng em xin trân trọng cám ơn thầy.

**MỤC LỤC**

**PHẦN 1: MỞ ĐẦU……………………………………………………………. 1**

1. **Lý do chọn đề tài…………………………………………………………….1**

**2. Mục đích - mục tiêu của đề tài……………………………………………...1**

**3. Nội dung đề tài……………………………………………………………….1**

**PHẦN 2: XÂY DỰNG BOT QUẢN LÝ SERVER DISCORD BẰNG THƯ VIỆN DISCORD CỦA NGÔN NGỮ LẬP TRÌNH PYTHON………………2**

1. **Đôi nét về ứng dụng Discord………………………………………………...2**

1.1 Giới thiệu ứng dụng Discord…..……………………………………….…2

1.2 Mục tiêu của ứng dụng Discord……...…………………………………...2

1.3 Khảo sát và phân tích……………………………………………………..2

1.3.1. Khảo sát………………………………………………………………2

1.3.2. Phân tích……………………………………………………………...3

**2. Xây dựng Bot Discord quản lý Server…….………………………………..4**

2.1. Cài đặt thư viện Discord…………………………………………………….4

2.2. Các bước khởi tạo Bot Discord…….……….…………………..……5

2.2.1. Khai báo thư viện…………………………..………………………5

2.2.2. Tạo một Bot mới trên Discord Developer Portal……..……...…….5

2.2.3. Thêm Bot vào máy chủ…………….….…………..……………….7

2.2.4. Khởi tạo Bot………….…………………………………………….8

2.2.5. Tạo sự kiện cho Bot………………………………………………...9

2.2.6. Triển khai Bot………………………………………………….….12

**3. Demo Bot Discord……………………………………………………..……35**

Tài liệu tham khảo……………………………………………………….37

**PHẦN 1: MỞ ĐẦU**

1. **Lý do chọn đề tài:**

Ngày nay, công nghệ thông tin đóng vai trò quan trọng trong sự phát triển của xã hội trong thời đại 4.0. Công nghệ thông tin đã trở thành nhân tố quan trọng, là cầu nối trao đổi giữa các thành phần của xã hội toàn cầu, của mọi vấn đề. Công nghệ thông tin đã và đang góp phần ngày càng quan trọng đưa nước ta ngày càng phát triển nhanh hơn, góp phần hội nhập quốc tế hiệu quả hơn. Việc ứng dụng và phát triển công nghệ thông tin ở nước ta đã góp phần giải phóng sức mạnh vật chất, trí tuệ và tinh thần, thúc đẩy công cuộc đổi mới, phát triển nhanh và hiện đại hóa, tăng cường năng lực cạnh tranh của các doanh nghiệp, nâng cao chất lượng cuộc sống của nhân dân, đảm bảo an ninh quốc phòng góp phần thực hiện thắng lợi sự nghiệp công nghiệp hóa, hiện đại hóa đất nước.

Ngôn ngữ lập trình Python được tạo bởi Guido van Rossum thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu. Do vậy Python cũng được xem như là một ngôn ngữ lập trình tạo ra nhiều ứng dụng có ích trong đời sống nói chung và lập trình game nói riêng.

Trong quá trình tìm hiểu chúng em thấy rất hứng thú với việc quản lý Server Discord và lập trình bằng ngôn ngữ lập trình Python. Giới thiệu sơ lược về ứng dụng Discord:

Discord là một nền tảng giao tiếp phổ biến được phát hành lần đầu tiên vào năm 2015. Nó được tạo bởi Jason Citron, người trước đó đã thành lập OpenFeint, một nền tảng trò chơi xã hội dành cho thiết bị di động. Discord ban đầu được thiết kế như một nền tảng để các game thủ giao tiếp với nhau khi chơi trò chơi trực tuyến. Tuy nhiên, nó nhanh chóng trở nên phổ biến trong các cộng đồng khác và trở thành nền tảng để mọi người kết nối với những người khác có cùng sở thích. Ngày nay, Discord có hơn 250 triệu người dùng đã đăng ký và được sử dụng bởi các cá nhân, doanh nghiệp và tổ chức trên khắp thế giới.

**Mục đích - mục tiêu của đề tài:**

**- Mục đích:**

**+** Nắm chắc được được kỹ năng và kiến thức về lập trình.

+ Tìm hiều về thư viện Discord API trong ngôn ngữ lập trình Python.

+ Cũng cố, áp dụng, nâng cao kiến thức đã được học.

**- Mục tiêu:**

**+** Vận dụng được tính chất của lập trình hướng đối tượng.

+ Sử dụng thư viện Discord API để làm Bot quản lý Server.

1. **Nội dung đề tài:**

**Bao gồm 2 phần:**

* **Mở đầu**
* **Xây dựng ứng dụng Bot quản lý Server Discord bằng thư viện Discord API của Python**

1. Phân tích yêu cầu
2. Thiết kế kiến trúc
3. Lập trình và triển khai
4. Kiểm thử và sửa lỗi

**PHẦN 2: XÂY DỰNG ỨNG DỤNG BOT QUẢN LÝ SERVER DISCORD BẰNG THƯ VIỆN DISCORD API CỦA PYTHON**

**Đôi nét về ứng dụng Discord**

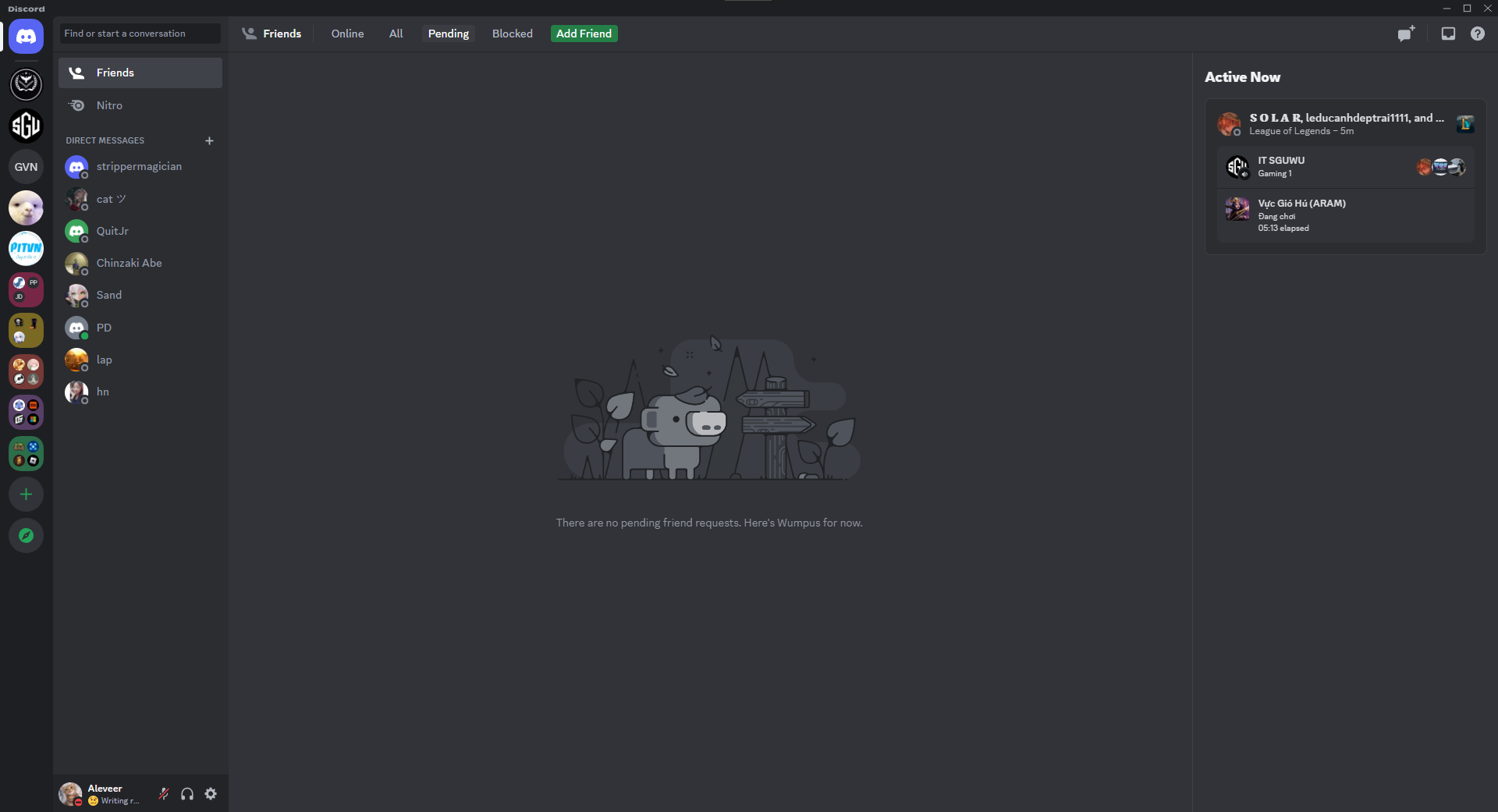
* 1. **Giới thiệu về nền tảng mạng xã hội Discord**

Discord là một nền tảng mạng xã hội được thiết kế đặc biệt cho cộng đồng game thủ và các nhóm chat trực tuyến khác. Nó cho phép người dùng tạo ra các máy chủ (Server) riêng để giao tiếp với bạn bè hoặc thành viên trong nhóm của mình thông qua các kênh (Channel) chat, giọng nói hoặc video.

Ngoài ra, Discord cũng hỗ trợ tính năng tạo Bot để tự động hóa các hoạt động quản lý Server, giúp cho việc quản lý Server trở nên dễ dàng hơn. Để tạo Bot cho Server Discord, ta có thể sử dụng thư viện Discord API của Python.



**Hình 1: Logo Discord**



**Hình 2: Giao diện chính ứng dụng Discord**

* 1. **Mục tiêu của ứng dụng Discord:**

Mục tiêu của ứng dụng Discord là cung cấp cho người dùng một nền tảng giao tiếp trực tuyến hiệu quả và tiện lợi, đặc biệt là trong lĩnh vực game và các hoạt động liên quan đến cộng đồng. Nó cho phép người dùng tạo ra các Server riêng để kết nối với bạn bè hoặc thành viên trong nhóm của mình, chia sẻ thông tin và tương tác với nhau thông qua các kênh chat, giọng nói hoặc video.

**1.3 Tổng quan và phân tích:**

**1.3.1. Khảo sát:**

Discord là một ứng dụng giao tiếp trực tuyến phổ biến, đặc biệt trong cộng đồng game thủ và các nhóm cộng đồng trực tuyến khác.

Một trong những lợi ích của Discord là tính linh hoạt và đa dạng của nó. Người dùng có thể tùy chỉnh các kênh chat và quyền truy cập cho từng thành viên, tạo ra các vai trò khác nhau để quản lý server và tùy chỉnh các cài đặt âm thanh và video cho phù hợp với nhu cầu của mình.

Việc tạo Bot cho máy chủ Discord cũng là một lợi ích lớn, giúp cho việc quản lý Server trở nên dễ dàng hơn và giảm thiểu thời gian và công sức cho các hoạt động quản lý Server. Bot có thể tự động thực hiện các nhiệm vụ như kiểm tra và cập nhật thông tin server, quản lý thành viên, tạo ra các thông báo tự động và nhiều tính năng khác.

Tuy nhiên, Discord cũng có một số hạn chế như việc không thể tùy chỉnh giao diện của ứng dụng hoặc các kênh Chat quá nhiều. Ngoài ra, việc sử dụng Bot cũng có thể gây ra một số vấn đề liên quan đến bảo mật và an toàn thông tin.

**1.3.3. Phân tích:**

- Các hàm trong thư viện Discord API được sử dụng cho việc tạo Bot:

|  |  |
| --- | --- |
| init() | Khởi tạo tất cả các module Discord đã được nhập (trả về một bộ giá trị cho biết việc thành công và thất bại của quá trình khởi tạo) |
| Embed | Là một thư viện bên thứ 3 cung cấp giao diện cho Bot Discord và ứng dụng |
| Discord.ext | Cập nhật màn hình |

1. **Xây dựng Bot Discord quản lý Server:**
   1. **Cài đặt thư viện Discord API:**

**-** Visual Studio Code (64-bit).

- Điều đầu tiên cần làm để lập trình trò chơi trong Python là cài đặt module Discord API. Để thực hiện cài cho module này, có thể nhập câu lệnh như sau trong command Prompt: pip install discord

Sau đó hệ thống sẽ tự động cài đặt thư viện:

**Text

Description automatically generated with medium confidence**

**Hình 3: Cài đặt thư viện Discord**

* 1. **Các bước khởi tạo Bot Discord:**
     1. **Khai báo thư viện:**

Sau khi cài đặt xong thư viện Discord, ta tiến hành khai báo các thư viện cần dùng:

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

Description automatically generated**

**Hình 4: Khai báo thư viện**

**2.2.2 Tạo một Bot mới trên Discord Developer Portal**

Đầu tiên, chúng ta cần đăng nhập vào Discord Developer Portal bằng tài khoản Discord của mình. Sau đó, bạn sẽ thấy giao diện chính của trang Web. Nhấp vào nút "New Application" để bắt đầu quá trình tạo bot mới.

Text

Description automatically generated

**Hình 5: Giao diện chính của Discord Developer Portal**

Sau khi nhấp vào nút "New Application", chúng ta sẽ được yêu cầu đặt tên cho ứng dụng của mình. Tên này không cần phải giống với tên Bot, nhưng nó sẽ được hiển thị trong danh sách các ứng dụng của bạn trên Discord Developer Portal.

A screenshot of a computer

Description automatically generated with medium confidence

**Hình 6: Giao diện tạo ứng dụng khi bấm vào “New Application”**

**Graphical user interface, text, application

Description automatically generated**

**Hình 7: Giao diện chính sau khi được tạo**

**2.2.3 Thêm Bot vào máy chủ cần quản lý**

Tiếp theo, ta cần cài đặt phân quyền cho Bot mới và tùy chỉnh Bot sao cho phù hợp với mục đích của mình. Và sau khi cài đặt phân quyền cho Bot, ta sẽ thêm vào Server mình cần quản lý, và trong trường hợp này, chúng ta sẽ thêm Bot vào Server Discord thông qua Oauth2 URL Generator:

Text

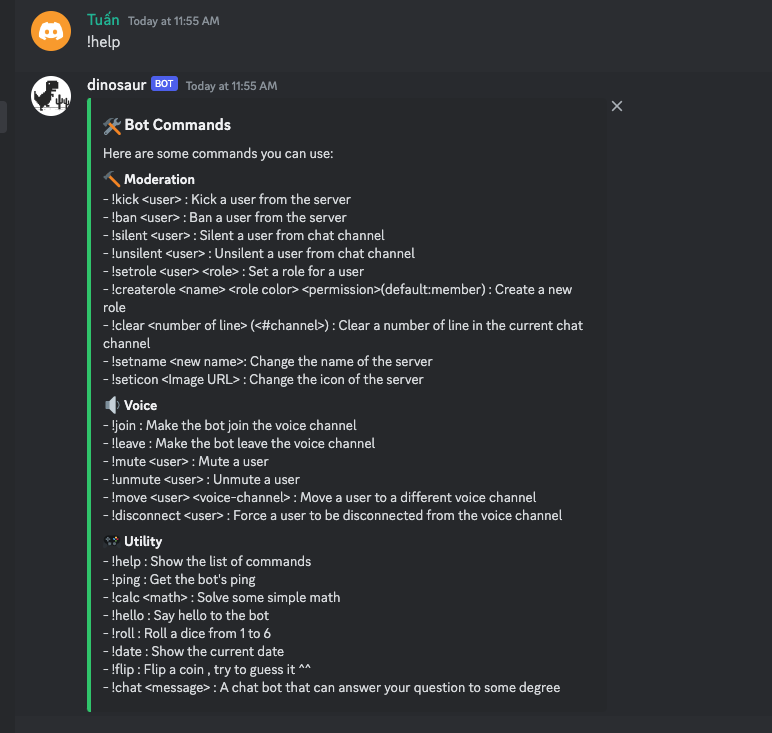
Description automatically generated

**Hình 8: Sau khi cung cấp quyền hạn cho Bot, Web sẽ cung cấp đường dẫn để thêm Bot vào Server Discord.**

Ngoài ra, sau khi thêm thành công vào máy chủ, chúng ta có thể quản lý Bot thông qua việc sử dụng Discord API và sử dụng Token của Bot. Thông qua 2 thư viện này, chúng ta có thể triển khai các tính năng cho Bot của mình, bao gồm xử lý tin nhắn, phẩn hồi các lệnh và tương tác người dùng.

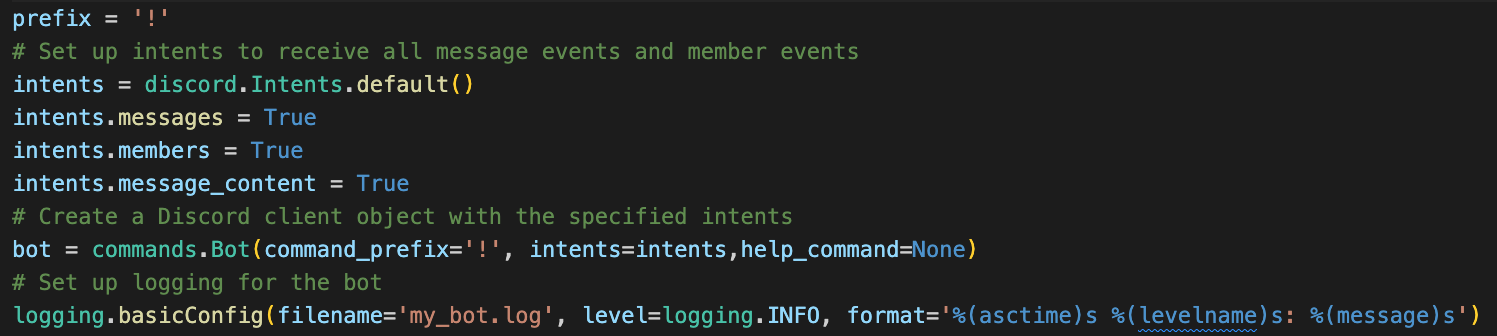
* + 1. **Khởi tạo bot:**

Để khởi tạo Bot thì đầu tiên ta cần nhập và cài đặt các module cần thiết như asyncio, dotenv, os, discord, logging và một số module tùy chỉnh khác. Sau đó, mã lập trình cài đặt một Bot với các intent Discord cụ thể và cấu hình Logging. Nó định nghĩa một số chức năng bất đồng bộ để nhận và gửi phản hồi cho các tin nhắn của Bot. Cuối cùng, nó chạy Bot Discord với mã thông báo Bot và cài đặt một số trình xử lý sự kiện cho Bot lắng nghe. Bot có thể xử lý các tin nhắn của người dùng, các sự kiện tham gia và các sự kiện tham gia thành viên. Nó cũng có thể thực thi các lệnh cụ thể khi người dùng nhập chúng. Ví dụ 1 lệnh cụ thể khi ta nhập lệnh “!help”, lệnh này sẽ yêu cầu người dùng có vai trò cụ thể để thực thi. Lệnh “!help” ở đây ta cho nó vai trò là hiện những dòng lệnh và chức năng những dòng là mà Bot đang có.



**Hình 9: Ví dụ cho lệnh “!help”**

Để khởi tạo được Bot ta cần sử dụng thư viện Discord.py để tạo ra các sự kiện và chức năng cho nó có thể nhận và phản hồi lại tin nhắn:



**Hình 10: Thiết lập bot discord bằng thư viện Discord.py**

* Dòng đầu tiên thiết lập một tiền tố lệnh ('!') cho bot.
* Khối mã thứ hai thiết lập intents để bot nhận tất cả các sự kiện tin nhắn và sự kiện thành viên. intents cụ thể được thiết lập để nhận nội dung tin nhắn.
* Khối mã thứ ba khởi tạo một đối tượng bot Discord với tiền tố lệnh và intents được chỉ định. "help\_command=None" vô hiệu hóa lệnh trợ giúp mặc định.
* Khối mã thứ tư thiết lập logging cho Bot, với tên tệp 'my\_bot.log', mức logging là INFO và định dạng cụ thể cho các thông báo trong Log.

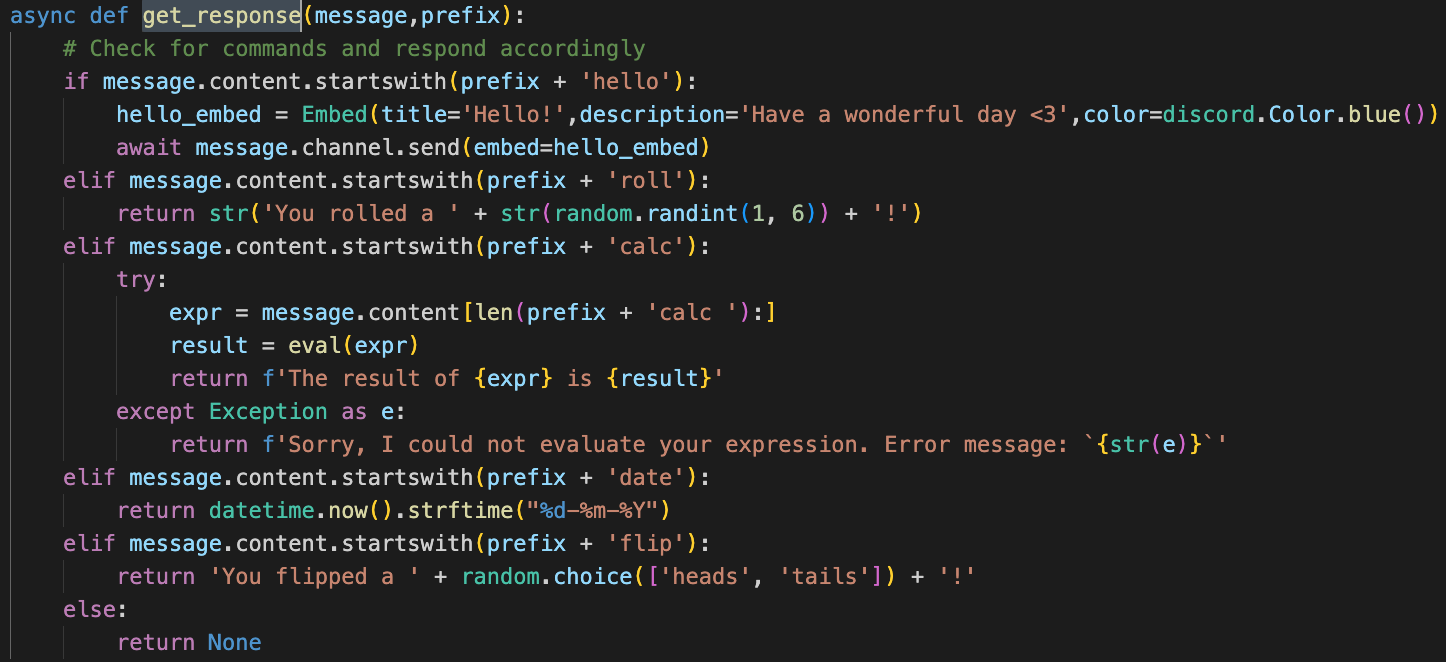
**2.2.5: Tạo sự kiện cho bot**

Sau khi thiết lập tố lệnh cho con bot ta sẽ tạo ra hàm get\_response và hàm send\_message.

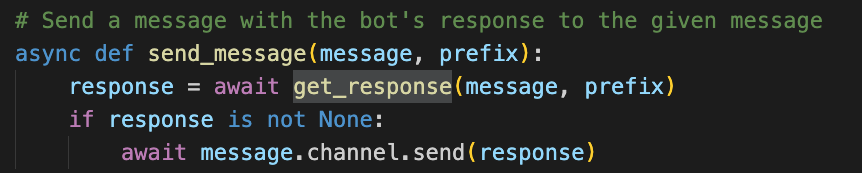
Hàm get\_response được sử dụng để lấy phản hồi cho một tin nhắn được cung cấp. Nó sẽ kiểm tra các tin nhắn bắt đầu với các lệnh được thiết lập trước đó (được định nghĩa bởi prefix) và trả về một phản hồi tương ứng. Nếu tin nhắn không bắt đầu bằng bất kỳ lệnh nào được định nghĩa, nó sẽ trả về None.

Các câu lệnh gồm có:

* "!hello": để Bot chào hỏi và cho một lời chúc mừng.
* "!roll": để tung một xúc xắc và trả lại số được tung ra.
* "!calc": để tính giá trị của một biểu thức được cung cấp bởi người dùng và trả lại kết quả.
* "!date": để trả về ngày hiện tại.
* "!flip": để tung một đồng xu và trả về mặt ngửa hoặc mặt sấp.

  
 **Hình 11: Hàm lấy phản hồi của tin nhắn**

Hàm send\_message được sử dụng để gửi một tin nhắn phản hồi từ Bot cho một tin nhắn được cung cấp. Nó sử dụng hàm get\_response để lấy phản hồi tương ứng và gửi nó đến kênh tin nhắn đã chỉ định nếu có phản hồi được trả về.



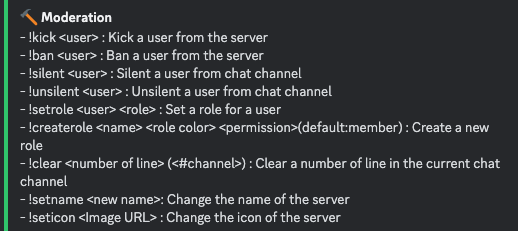
**Hình 12: Hàm gửi tin nhắn cho Bot phản hồi**

Sau khi tạo xong sự kiện gửi và phản hồi tin nhắn ta sẽ tạo các sự kiện và chức năng quản lý của Bot. Bot này gồm có 3 chức năng chính:

* Moderation (Quản lý server).
* Ultility (Tiện ích).
* Voice (Quản lý kênh thoại).

**1) Moderation (Quản lý server)**

- Chức năng này dùng để kết nối và thao tác trực tiếp đến với máy chủ Discord, đồng thời cung cấp chức năng quản lý cho người sử dụng Bot. Các chức năng đó gồm:



**Hình 13: Những chức năng chính trong mục Moderation**

!kick <user> : Đá một người dùng ra khỏi máy chủ.

Text

Description automatically generated

**Hình 14: Đoạn chương trình Python thực thi chức năng Kick thành viên**

* Đây là một hàm kick\_user được định nghĩa như một lệnh trong Discord Bot. Hàm này có 3 tham số là self, ctx và member (là một đối tượng discord.Member).
* Hàm này kiểm tra xem người dùng gọi lệnh có quyền kick thành viên hay không.
* Nếu không, nó sẽ thông báo cho người dùng rằng họ không có quyền để kick thành viên. Nếu có, hàm sẽ kiểm tra xem người dùng có đề cập một thành viên để kick hay không.
* Nếu không, Bot sẽ gửi một tin nhắn trong kênh yêu cầu người dùng đề cập một đối tượng thành viên để kick.
* Nếu người dùng muốn kick chính bot thì hàm sẽ gửi một tin nhắn cho người dùng nói rằng bot không thể kick chính nó.
* Nếu người dùng không phải là chủ sở hữu Server và Member có một Role bằng hoặc cao hơn Role của người dùng, hàm sẽ gửi một tin nhắn cho người dùng nói rằng họ không thể kick thành viên đó.
* Nếu không, hàm sẽ kick thành viên đã được đề cập với lý do "Kicked by" và gửi một tin nhắn tới kênh nói rằng thành viên đã bị kick bởi tác giả.
* Kết quả trả về của hàm kick\_user chính là code tương ứng với định nghĩa của hàm này.

!ban <user> : Cấm một người dùng truy cập vào máy chủ

Text

Description automatically generated

**Hình 15: Đoạn chương trình Python thực thi chức năng Ban thành viên**

* Đây là một đoạn code Python định nghĩa một lệnh ban\_user cho Bot Discord.
* Hàm này có ba tham số là self, ctx và member. Tham số member là một đối tượng discord.Member, là người dùng cần ban. Tham số reason là một chuỗi, là lý do ban.
* Đầu tiên, hàm kiểm tra xem người dùng có đề cập đến thành viên cần ban hay không. Nếu không, Bot sẽ gửi một tin nhắn riêng cho người dùng yêu cầu họ chỉ định một người dùng để ban và return.
* Tiếp theo, hàm kiểm tra xem người dùng có quyền ban thành viên đó hay không. Nếu người dùng không phải là chủ sở hữu Server và thành viên có một role bằng hoặc cao hơn role của người dùng, thì Bot không thể ban thành viên đó và sẽ gửi một tin nhắn trong kênh nói với người dùng rằng Bot không thể ban user này và return.
* Nếu người dùng có quyền ban, hàm tiếp tục bằng cách thực hiện việc ban member. Nếu không có lý do được cung cấp, lý do mặc định sẽ là 'No reason specified.'. Sau khi ban thành viên, hàm sẽ gửi một tin nhắn trong kênh nói với người dùng rằng thành viên đã bị ban với tên thành viên. Nếu lý do được cung cấp, hàm sẽ gửi một tin nhắn khác với lý do ban.
* Nếu Bot không có quyền ban hoặc có lỗi nào đó xảy ra trong quá trình ban, hàm sẽ gửi một tin nhắn trong channel tương ứng để thông báo cho người dùng.
* Kết quả trả về của hàm ban\_user chính là code tương ứng với định nghĩa của hàm này.

!silent <user> : Tắt tiếng một người dùng trên kênh chat

Text

Description automatically generated

**Hình 16: Đoạn chương trình thực thi chức năng cấm thành viên trò chuyện**

* Đoạn code này định nghĩa một lệnh "silent" cho bot Discord. Hàm này có ba tham số là self, ctx và user, trong đó user là một đối tượng thành viên của Discord cần bị mute.
* Đầu tiên, hàm này sẽ tạo hoặc lấy role "Muted". Role "Muted" này sẽ được gán vào user để mute người dùng.
* Nếu người dùng muốn mute không có role "Muted", hàm sẽ tạo một role mới. Sau đó, hàm này sẽ chạy một vòng lặp for để set quyền cho channel cho phép chỉ có role "Muted" mới không thể gửi tin nhắn.
* Nếu người dùng muốn mute là chủ sở hữu server hoặc có role cao hơn người dùng gửi lệnh "silent", hàm sẽ gửi một tin nhắn cho người dùng để thông báo rằng họ không thể mute người dùng đó.
* Nếu người dùng muốn mute một người dùng đã bị mute rồi, hàm sẽ gửi một tin nhắn cho người dùng nói rằng người dùng này đã bị mute.
* Nếu người dùng muốn mute một người dùng mới, hàm sẽ gán role "Muted" cho user và gửi một tin nhắn cho người dùng để thông báo rằng người dùng đã bị mute.
* Kết quả trả về của hàm này là code tương ứng với định nghĩa của hàm "silent".

!unsilent <user> : Bật tiếng lại cho một người dùng trên kênh chat

Text

Description automatically generated

**Hình 17: Đoạn chương trình thực thi chức năng cho phép thành viên trò chuyện**

* Đoạn code trên định nghĩa một lệnh gọi là "unsilent" cho phép người dùng có quyền "kick\_members" unmute (xóa role "Muted" khỏi) một người dùng được chỉ định.
* Hàm đầu tiên kiểm tra xem vai trò "Muted" có tồn tại trong Server hay không. Nếu không tồn tại, hàm sẽ gửi một tin nhắn đến kênh chat và rời khỏi hàm.
* Tiếp theo, hàm kiểm tra xem người dùng đã có vai trò "Muted" hay chưa. Nếu chưa có, hàm sẽ gửi một tin nhắn đến kênh chat và rời khỏi hàm.
* Cuối cùng, nếu người dùng đã có vai trò "Muted", hàm sẽ xóa nó và gửi tin nhắn xác nhận đến kênh chat.

!setrole <user> <role> : Thiết lập một vai trò cho một người dùng.

Text

Description automatically generated

**Hình 18: Đoạn chương trình thực thi đặt vai trò cho thành viên**

Đây là đoạn code cho phép Bot gán một vai trò cho thành viên của server. Các bước cụ thể như sau:

* Nhận vào 3 tham số: ctx là context của lệnh, member là thành viên cần được gán vai trò và role\_name là tên của vai trò.
* Tìm vai trò theo tên được cung cấp bằng câu lệnh role = discord.utils.get(ctx.guild.roles, name=role\_name).
* Kiểm tra xem vai trò có tồn tại không bằng câu lệnh if role is None:. Nếu không tồn tại, Bot sẽ gửi một tin nhắn đến kênh Chat thông báo về điều đó và rời khỏi hàm bằng lệnh return.
* Kiểm tra xem Bot có quyền gán vai trò không bằng câu lệnh if not ctx.guild.me.guild\_permissions.manage\_roles:. Nếu không có quyền, Bot sẽ gửi một tin nhắn thông báo và rời khỏi hàm.
* Kiểm tra xem người gửi lệnh có quyền gán vai trò cho người khác không bằng cách kiểm tra các vai trò của người gửi lệnh và người được gán vai trò. Nếu người gửi lệnh không có quyền gán vai trò cho người được chỉ định, bot sẽ gửi một tin nhắn và rời khỏi hàm.
* Nếu các bước kiểm tra trên đều đúng, Bot sẽ thực hiện lệnh await member.add\_roles(role) để gán vai trò cho thành viên và gửi một tin nhắn xác nhận đến kênh chat.
* Nếu Bot không có quyền gán vai trò, tin nhắn lỗi sẽ được gửi đến kênh chat để thông báo về vấn đề này.

!createrole <name> <role color> <permission> (mặc định: thành viên) : Tạo một vai trò mới.

A picture containing text, screenshot, software

Description automatically generated

**Hình 19: Đoạn chương trình thực thi tạo vai trò mới**

* Đây là một đoạn mã Python cho một lệnh Bot Discord được gọi là "createrole". Nó tạo một vai trò mới với tên, màu sắc và quyền được chỉ định và sau đó gửi một thông báo trở lại kênh để cho biết liệu việc tạo vai trò đã thành công hay không. Lệnh này chỉ được sử dụng bởi các người dùng có quyền quản trị.
* Chức năng đầu tiên kiểm tra xem vai trò có cùng tên đã tồn tại trong guild hay chưa. Nếu đã tồn tại, hàm trả về một thông báo lỗi, cho biết rằng vai trò đã tồn tại.
* Bước tiếp theo là kiểm tra xem tham số quyền có phải là một chuỗi hay không. Nếu nó là một chuỗi, thì nó được chuyển đổi thành một đối tượng Quyền. Nếu không phải là một chuỗi, thì một đối tượng Quyền mặc định được tạo ra.
* Cuối cùng, mã cố gắng tạo ra một vai trò mới với các thông số được chỉ định và gửi một tin nhắn trở lại kênh để cho biết liệu hoạt động đã thành công hay không. Nếu bot không có quyền tạo vai trò, hoặc nếu có lỗi HTTP, mã xử lý những tình huống này và gửi một thông báo lỗi trở lại kênh.

!clear <số dòng> (<#kênh>) : Xóa một số dòng trên kênh chat hiện tại

Text

Description automatically generated

**Hình 20: Đoạn chương trình thực thi xóa số dòng chat trong kênh**

* Đây là một lệnh của Bot Discord cho phép người dùng có quyền "kick\_members" xóa một lượng tin nhắn xác định trong một kênh cụ thể hoặc kênh hiện tại.
* Lệnh này được kích hoạt bằng cách nhập "!clear" và theo sau đó là số lượng tin nhắn cần xóa.
* Nếu số lượng tin nhắn không hợp lệ, bot sẽ gửi một tin nhắn thông báo lỗi. Nếu không có kênh nào được chỉ định, Bot sẽ mặc định đến kênh hiện tại.
* Lệnh này sử dụng phương thức "purge", xóa các tin nhắn từng đoạn để tránh xung đột với giới hạn tốc độ được đặt bởi Discord. Ngoài ra, Bot sẽ đợi 2 giây trước khi xóa lô tin nhắn tiếp theo.

!setname <tên mới> : Thay đổi tên của máy chủ

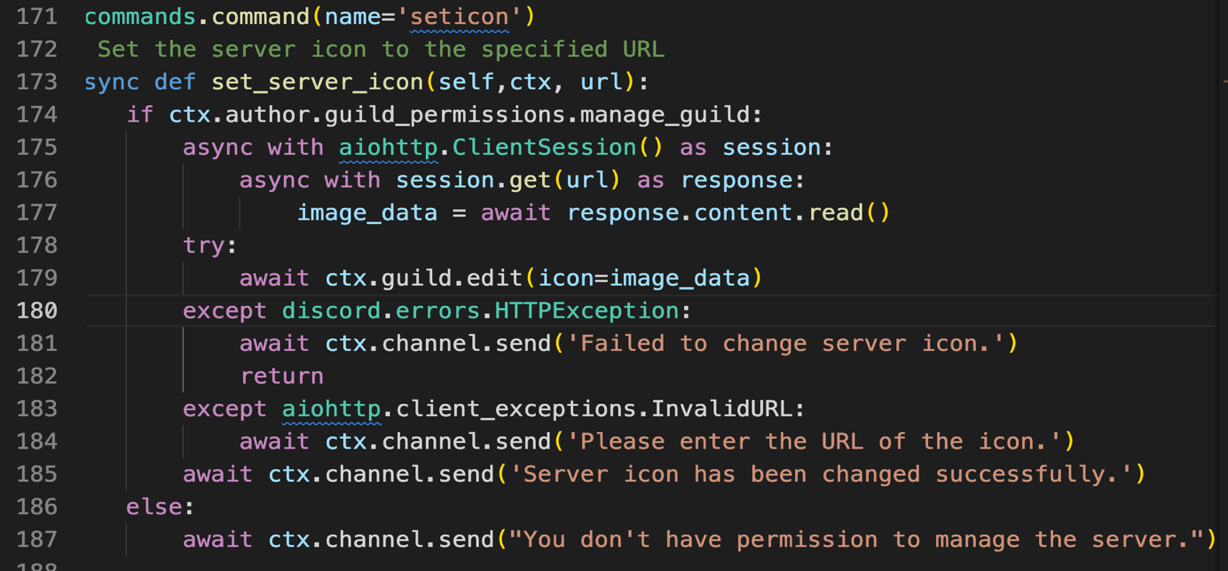
A screen shot of a computer program

Description automatically generated with medium confidence

**Hình 21: Đoạn chương trình thực thi chức năng thay đổi tên máy chủ**

* Đây là một đoạn code của Bot Discord cho phép người dùng có quyền "manage\_guild" của Server đổi tên server bằng cách sử dụng lệnh "!setname" và theo sau là tên mới của Server.
* Nếu người dùng có quyền, Bot sẽ cập nhật tên mới cho Server và gửi thông báo thành công, nếu không, bot sẽ gửi một tin nhắn lỗi phù hợp.
* Đoạn code này sử dụng exception handling để bắt lỗi xảy ra trong quá trình cập nhật tên Server, đảm bảo Bot có thể xử lý các lỗi và gửi thông báo phù hợp cho người dùng.

!seticon <URL ảnh> : Thay đổi biểu tượng của máy chủ



**Hình 22: Đoạn chương trình thực thi chức năng đổi ảnh đại diện cho Server**

* Đây là một đoạn mã tạo ra một chức năng bất đồng bộ gọi là "set\_server\_icon" với 2 tham số đầu vào là "ctx" (context) và "url". Đây là một lệnh cho Bot Discord cho phép người dùng có quyền "manage\_guild" để thay đổi biểu tượng của máy chủ thành một URL cụ thể bằng cách sử dụng lệnh "!seticon" theo sau là URL đó.
* Sử dụng module aiohttp, hàm này sẽ lấy dữ liệu hình ảnh từ URL được chỉ định và thử chỉnh sửa biểu tượng máy chủ bằng cách sử dụng "ctx.guild.edit". Nếu quá trình chỉnh sửa thất bại do lỗi HTTP, Bot sẽ gửi một thông báo nói rằng việc thay đổi biểu tượng máy chủ đã thất bại. Nếu URL hình ảnh không hợp lệ, nó sẽ gửi một tin nhắn yêu cầu URL hợp lệ. Nếu việc thay đổi biểu tượng máy chủ thành công, bot sẽ gửi một thông báo cho người dùng.
* Nếu người dùng nhập lệnh không có quyền "manage\_guild", bot sẽ gửi một thông báo nói rằng họ không có quyền quản lý máy chủ.
* Decorator "@commands.command" ở đầu đoạn mã làm cho hàm này trở thành một lệnh mà người dùng có thể gọi. Thuộc tính "name" chỉ định tên lệnh, trong trường hợp này là "seticon".

**2) Utility**

Utility là 1 tổng hợp của nhiều chức năng đa dụng khác của Bot :

A black screen with white text

Description automatically generated with low confidence

**Hình 23: Danh sách các chức năng ở mục Utility**

!help: Hiển thị danh sách các lệnh.

A screen shot of a computer program

Description automatically generated with low confidence

**Hình 24: Đoạn chương trình thực thi chức năng xem danh sách các lệnh**

Lệnh sẽ được kích hoạt bằng cách người dùng gõ "!help" trong kênh văn bản và gửi một thông điệp nhúng chứa danh sách các lệnh có sẵn, được phân loại thành ba nhóm: hỗ trợ, giọng nói và tiện ích. Mỗi nhóm sẽ hiển thị một danh sách các lệnh với mô tả ngắn gọn.

* Đoạn mã định nghĩa một hàm được gọi là "help\_command" với hai tham số là "self" và "ctx". Tham số "self" liên quan đến phiên bản của bot và là bắt buộc cho tất cả các chức năng được xác định trong lớp bot. Tham số "ctx" là một đối tượng ngữ cảnh chứa thông tin về lệnh, chẳng hạn như người gửi lệnh và kênh nó được gửi ở đâu.
* Hàm tạo một đối tượng nhúng mới sử dụng thư viện Discord.py để định dạng thông điệp dưới dạng thông điệp nhúng. Đối tượng Embed chứa một tiêu đề, mô tả và màu sắc, được thiết lập để hiển thị tiêu đề "Bot Commands", mô tả ngắn gọn và màu xanh lá cây.
* Hàm sau đó thêm ba trường vào đối tượng nhúng, một cho mỗi nhóm lệnh. Mỗi trường có một tên (tên nhóm) và một giá trị (danh sách các lệnh trong nhóm đó). Giá trị là một chuỗi chứa các lệnh và mô tả của chúng, cách nhau bằng dấu xuống dòng.
* Cuối cùng, hàm gửi đối tượng nhúng làm thông điệp đến kênh nơi lệnh được gửi bằng cách sử dụng chức năng "ctx.send". Thông điệp được hiển thị dưới dạng thông điệp nhúng với tiêu đề, mô tả và các trường được xác định trong đối tượng Embed.

!ping: Gọi phản hồi của Bot

A picture containing text, screenshot, font

Description automatically generated

**Hình 25: Đoạn chương trình Python thực thi chức năng lấy thông tin phản hồi từ Bot**

* Lệnh này có tên là "ping", được xác định bằng cách sử dụng decorator @commands.command(name='ping').
* Hàm "ping" có hai tham số, "self" và "ctx". Tham số "self" chỉ phiên bản của Bot và được yêu cầu cho tất cả các chức năng được định nghĩa trong lớp Bot. Tham số "ctx" là một đối tượng ngữ cảnh chứa thông tin về lệnh như người gửi lệnh và kênh nó được gửi ở đâu.
* Nếu người dùng nhập lệnh "ping", Bot sẽ gửi một thông điệp để hiển thị Pong bằng cách sử dụng hàm ctx.send("Pong \U0001F3B3"). Kết quả sẽ hiển thị dưới dạng "Pong 🎳" trong kênh đã gửi lệnh.

!calc <phép tính>: Giải quyết một số phép tính đơn giản.

!hello: Chào hỏi với bot.

!roll: Lắc xúc xắc từ 1 đến 6.

!date: Hiển thị ngày hiện tại.

!flip: Lật đồng xu

A screen shot of a computer program

Description automatically generated with low confidence

**Hình 26: Đoạn chương trình Python thực thi những chức năng khác của Bot**

* Hàm async Python này có hai tham số: message và prefix. Trong hàm, nó kiểm tra tiền tố của nội dung tin nhắn và phản hồi theo nội dung tin nhắn.
* Nếu tiền tố cộng với chuỗi "hello" nằm ở đầu nội dung tin nhắn, nó sẽ gửi một tin nhắn nhúng chào mừng người dùng.
* Nếu tiền tố cộng với chuỗi "roll" nằm ở đầu nội dung tin nhắn, nó sẽ trả về một số nguyên được tạo ngẫu nhiên trong khoảng từ 1 đến 6.
* Nếu tiền tố cộng với chuỗi "calc" nằm ở đầu nội dung tin nhắn, nó sẽ tính toán biểu thức toán học sau tiền tố "calc" và trả về kết quả.
* Nếu tiền tố cộng với chuỗi "date" nằm ở đầu nội dung tin nhắn, nó sẽ trả lại ngày và giờ dưới định dạng của "DD-MM-YY".
* Cuối cùng, nếu tiền tố cộng với chuỗi "flip" nằm ở đầu nội dung tin nhắn, nó sẽ trả về một kết quả đầu hoặc đuôi được tạo ngẫu nhiên.
* Nếu không có trường hợp nào của nội dung tin nhắn phù hợp, nó sẽ trả về None.

!chat <tin nhắn>: Một bot trò chuyện có thể trả lời câu hỏi của bạn một cách cơ bản.

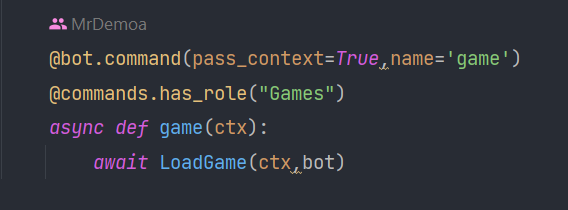
A picture containing text, screenshot, software

Description automatically generated

**Hình 27: Đoạn chương trình Python thực thi chức năng Chat với Bot**

* Khi được thực thi, Bot sẽ lấy tin nhắn được cung cấp bởi người dùng, thêm tên người dùng làm lời nhắc và gửi phản hồi đó đến API OpenAI. API sẽ tạo ra một phản hồi cho lời nhắc đó, mà Bot sau đó sẽ gửi lại cho người dùng trên Discord.
* Nếu tin nhắn bị thiếu hoặc trống, Bot sẽ yêu cầu người dùng cung cấp tin nhắn để trò chuyện.
* Nếu có lỗi trong cuộc gọi API, Bot sẽ gửi một thông báo lỗi đến người dùng.

!game: Một trò chơi Tic Tac Toe đơn giản cho 2 người chơi.

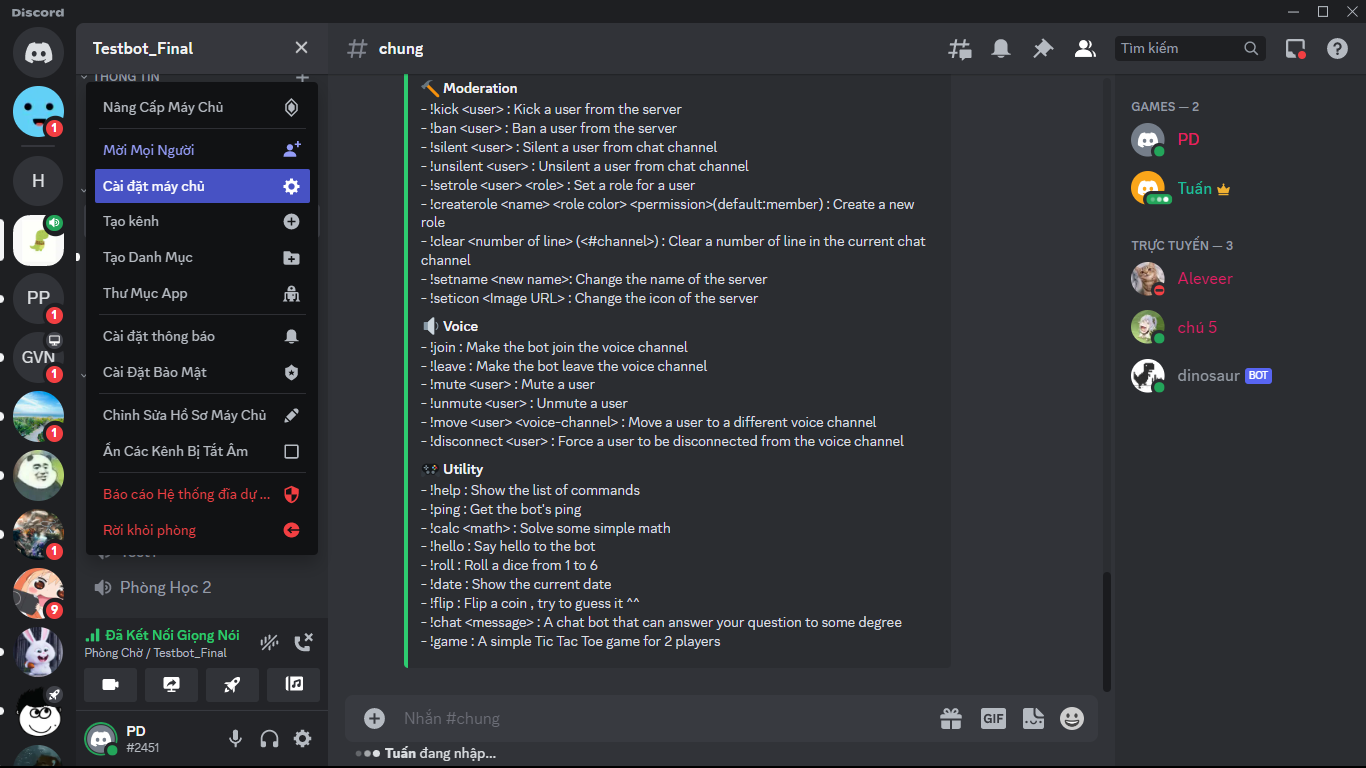


**Hình 28: Đoạn chương trình Python thực thi gọi hàm chức năng chơi Game với Bot**

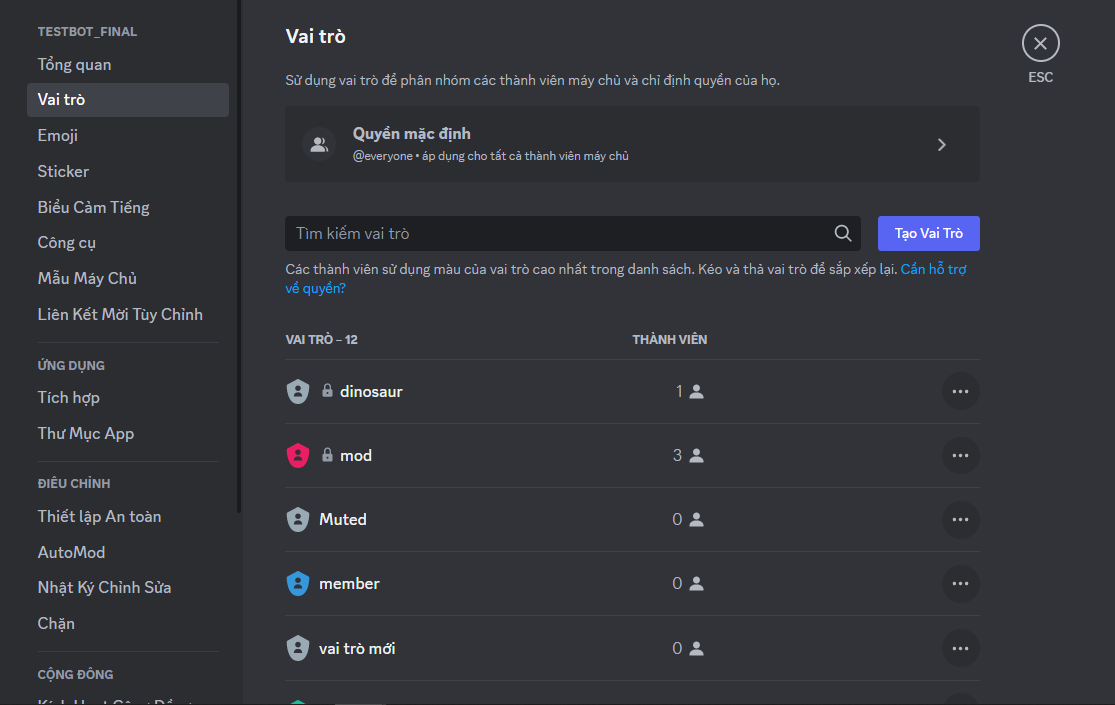
* @bot.command(pass\_context=True,name=’game’) để đăng ký câu lệnh !game cho Bot Discord.
* @bot.command.has\_role(“Game”) để kiểm tra vai trò của người dùng có thuộc role Game hay không. Ở đây nếu chúng ta muốn chơi Tic–Tac–Toe thì buộc người dùng phải tham gia vào role Game ở trên Server Discord.

Để tạo Role Game trên Sever Discord:

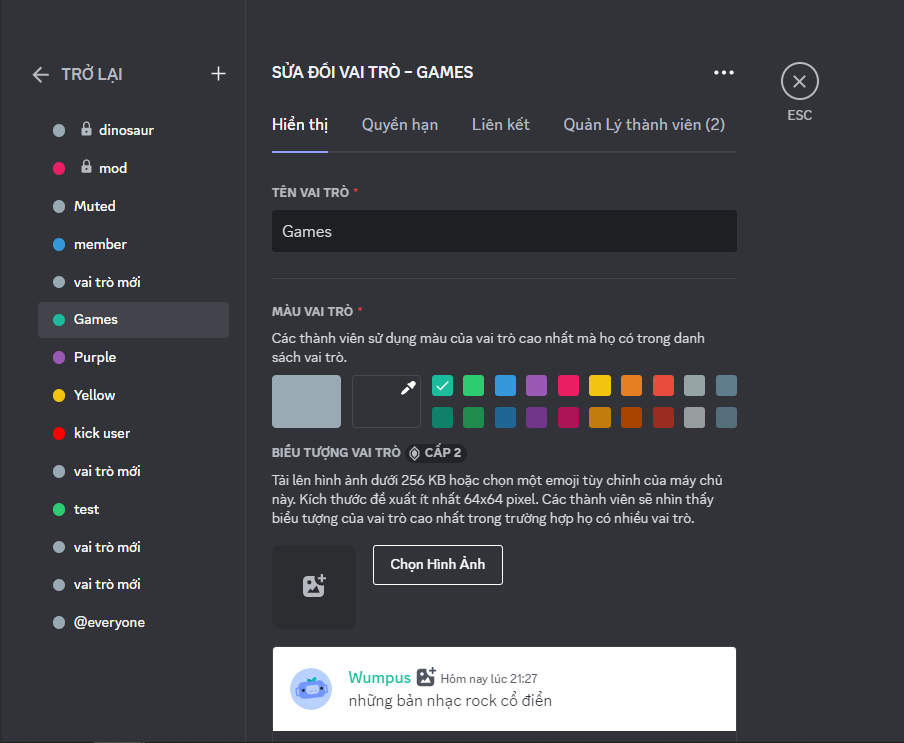
* Đầu tiên vào phần cài đặt máy chủ:



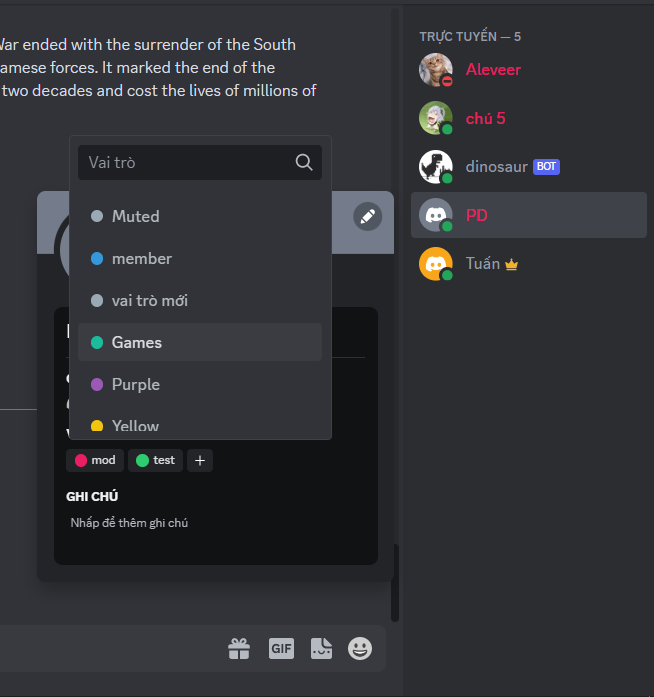
* Vào phần vai trò chọn tạo vai trò:



* Tại đây, chúng ta sẽ đặt tên và màu cho vai trò cũng như quyền hạn, quản lý thành viên:



* Khi Admin muốn thêm role Games cho thành viên thì click chuột trái vào ảnh đại diện bên góc phải Chọn dấu “+” chọn Games hoặc !setRole @User <Role>

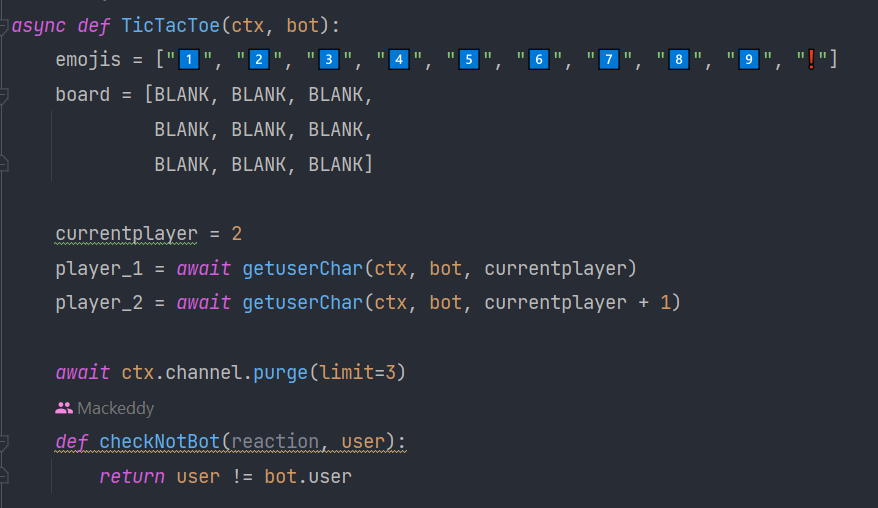


* Tiếp theo sẽ tạo hàm game để gọi đến hàm LoadGame bên file game.py

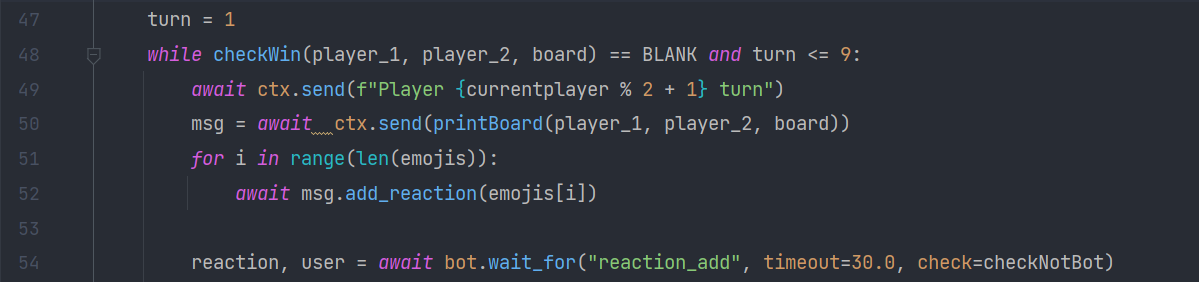


* Tạo tiêu đề Tic Tac Toe

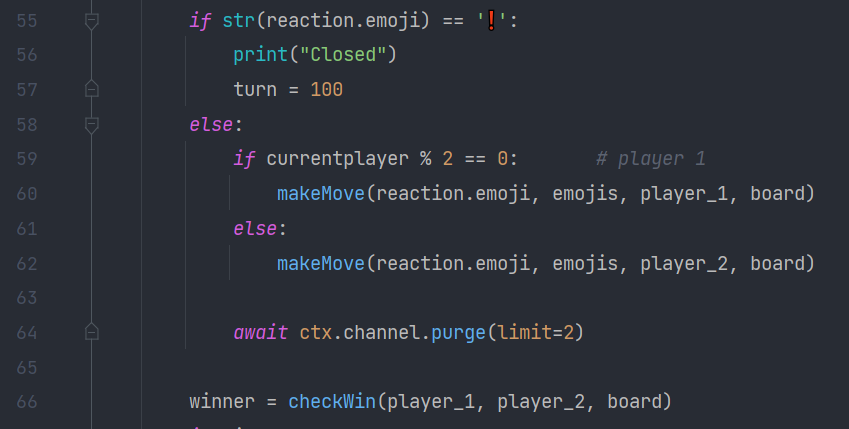
Tạo biến msg để lưu tin nhắn thông báo trò chơi bắt đầu. Thêm icon 🎮 (msg.add\_reaction) khi người dùng click vào thì trò chơi bắt đầu (gọi hàm Tic Tac Toe)



* Hàm TicTacToe truyền vào hai tham số ctx và bot. Ctx là một đối tượng Context của bot discord, cung cấp thông tin về lệnh được gọi, kênh chat,…. Còn bot cung cấp các thuộc tính để tương tác với discord API và xử lý sự kiện Discord.
* Sau đó hàm getuserChar dùng để người chơi chọn icon để đánh dấu đường đi của mình

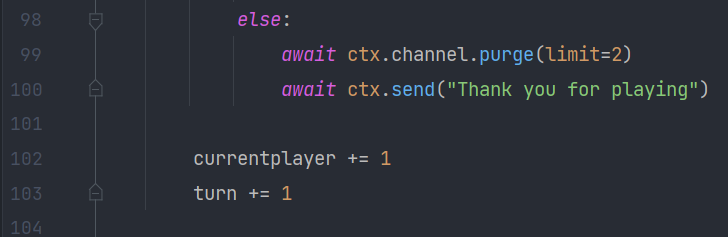


* Thiết lập biến turn = 1 để đếm lượt chơi.
* Đưa vào vòng lập nếu chưa có người thắng và số lượt bé hơn hoặc bằng 9(do Tic Tac Toe 3x3 ô) thì trò chơi vẫn tiếp tục.
* Mỗi lượt sẽ hiển thị đang tới lượt của player nào (dòng 49) và bảng 3x3 đánh dấu trò chơi (printBoard()).
* Dòng 51 và 52 sẽ hiển thị các icon số từ 1 tới 9 để người chơi chọn đánh dấu đường đi của mình sao cho phù hợp. Với mỗi icon đã được người chơi chọn thì lần sau sẽ được xóa đi.

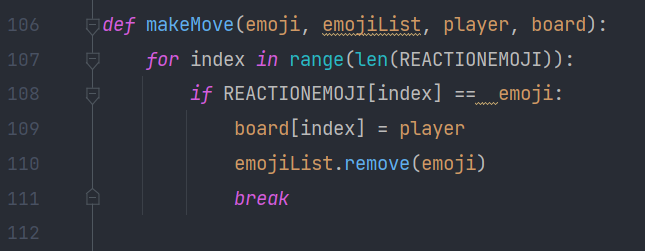




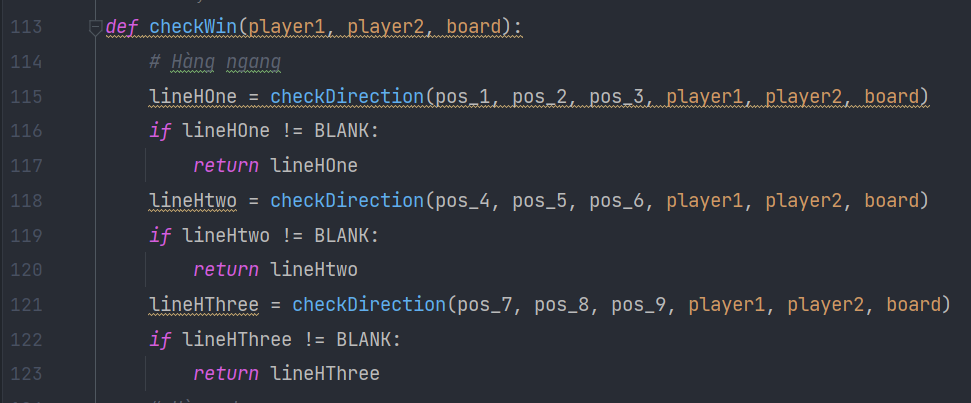


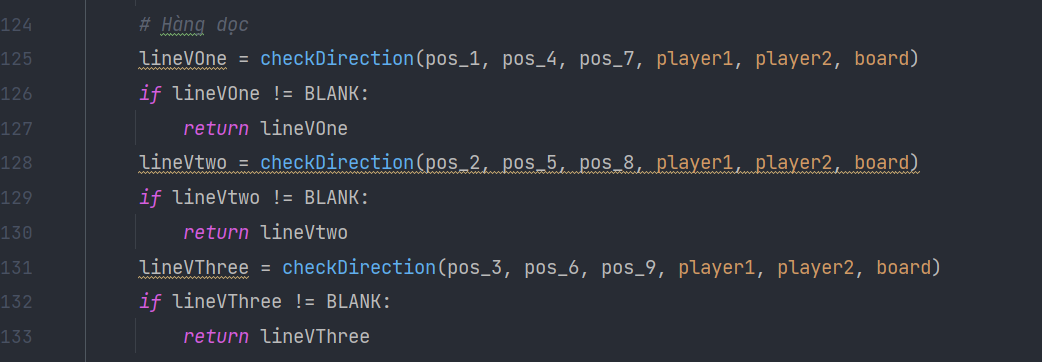


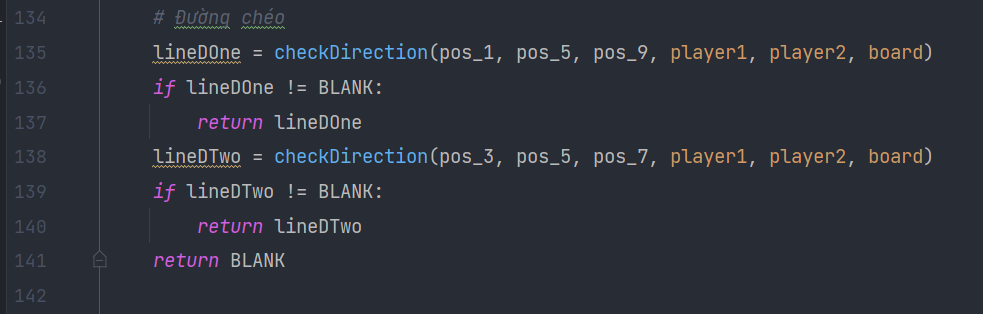
* Dòng 55 sẽ kiểm tra nếu người dùng click vào icon ! Thì sẽ cập nhật biến turn = 100. Đồng thời chương trình sẽ nhảy qua if kế tiếp (dòng 67)và kiểm tra nếu winner != BLANK thì sẽ thông báo tin nhắn hỏi rằng có muốn chơi lại không (dòng 68). Người chơi sẽ chọn 1 trong 2 icon ở dòng 70 và 71. Nếu chọn dấu tick xanh thì sẽ chơi lại và tiếp tục vòng lặp. Nếu không sẽ dừng trò chơi và hiện ra thông báo cảm ơn.
* Dòng 55 nếu người chơi không chọn icon ! Thì sẽ tiếp tục. Lúc này đã tìm ra được winner nên chương trình sẽ chuyển đến elif (dòng 84) do turn đã lớn hơn 9. Lúc này người chơi sẽ được hỏi là muốn tiếp tục chơi không và có 2 sự lựa chọn như trên.

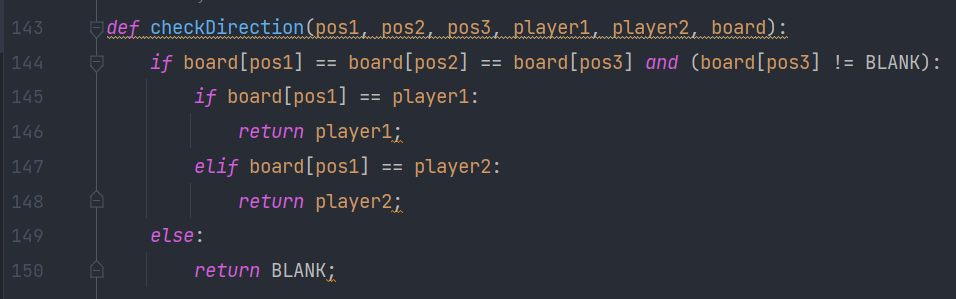


* Hàm makeMove sẽ xóa đi emoji mà người chơi đã chọn:



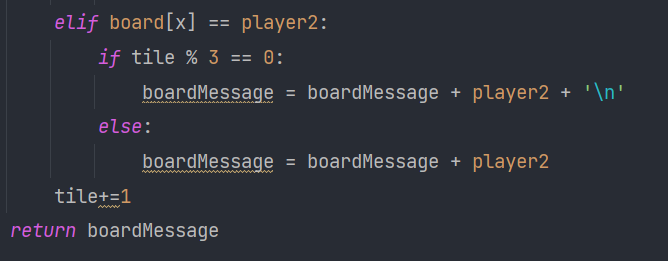






* Hàm checkwin sẽ kiểm tra xem ai là người đã chiến thắng. Hàm này sẽ check lần lượt các hàng ngang, hàng dọc, đường chéo nếu 3 ô trên cùng một hàng dều giống nhau và do player nào đánh dấu thì player đó chiến thắng. Nếu 3 ô trên một hàng không trùng thì trả về BLANK (hàm check direction).





* Hàm này dùng để in bảng 3x3 cùng với các icon lên màn hình để người dùng tương tác.



* Hàm getUserChar cho người chơi chọn icon mình sẽ đánh dấu trong game.

**3) Voice**

- Cuối cùng chính là tính năng voice, tính năng này có khả năng quản lý trực tiếp những thành viên đang trong hội thoại, gồm có các chức năng như:

A black screen with white text

Description automatically generated with low confidence

**Hình 48: Những chức năng chính của Bot ở mục Voice**

!join: Cho phép Bot tham gia kênh thoại.

A screen shot of a computer code

Description automatically generated with low confidence

**Hình 49: Đoạn chương trình Python cho phép thành viên vào Voice**

* Đầu tiên, command này được định nghĩa với tên 'join' bằng decorator '@commands.command(name='join')'.
* Sau đó, hàm join\_voice\_channel được định nghĩa với một tham số ctx, đại diện cho context của message gửi đến Bot.
* Hàm này kiểm tra xem tác giả ctx.author có đang được kết nối đến kênh hội thoại không bằng cách sử dụng ctx.author.voice. Nếu không, Bot sẽ trả về một tin nhắn cho tác giả cho biết rằng họ chưa kết nối đến kênh hội thoại nào.
* Nếu tác giả đang kết nối đến một kênh hội thoại, Bot sẽ lấy kênh đó bằng cách sử dụng ctx.author.voice.channel. Sau đó, Bot sử dụng await channel.connect() để kết nối vào kênh hội thoại đó, và gửi một tin nhắn để thông báo là đã kết nối thành công đến kênh đó.

!leave: Cho phép bot rời khỏi kênh thoại.

A screen shot of a computer

Description automatically generated with low confidence

**Hình 50: Đoạn chương trình Python cho Bot rời khỏi kênh hội thoại.**

* Hàm leave được định nghĩa với một tham số ctx, đại diện cho context của message gửi đến Bot. Hàm này bắt đầu bằng cách lấy kênh mà tác giả đang kết nối đến bằng cách sử dụng ctx.author.voice.channel.
* Sau đó, hàm này sử dụng await ctx.voice\_client.disconnect() để ngắt kết nối Bot với kênh đó. Cuối cùng, Bot sẽ gửi một tin nhắn đến kênh text được chỉ định trong ctx.send() để thông báo rằng Bot đã ngắt kết nối thành công. Tin nhắn này chứa tên của kênh hội thoại mà Bot đã ngắt kết nối ra khỏi bằng cách sử dụng f-string.

!mute <người dùng>: Tắt tiếng cho một người dùng.

A screen shot of a computer program

Description automatically generated with low confidence

**Hình 51: Đoạn chương trình Python thực thi tắt tiếng người dùng của Bot**

* Lệnh này sẽ mute (tắt tiếng) người dùng được chỉ định trong kênh giọng nói mà họ đang tham gia.
* Hàm mute\_voice được định nghĩa với ba tham số: self - đại diện cho chính Bot, ctx - đại diện cho context của tin nhắn gửi đến Bot, member - đại diện cho người dùng mà Bot sẽ tắt tiếng.
* Trước khi tắt tiếng một người dùng, hàm sẽ kiểm tra xem người dùng đó có đang kết nối đến kênh giọng nói không. Nếu không, Bot sẽ gửi một tin nhắn để thông báo cho tác giả rằng người dùng không đang kết nối đến kênh giọng nói nào cả.
* Sau đó, hàm sẽ kiểm tra xem người dùng đó có quyền cao hơn tác giả message không. Nếu có, tác giả message sẽ không thể tắt tiếng người dùng đó và bot sẽ gửi một tin nhắn để thông báo điều này.
* Nếu cả hai điều kiện trên đều thỏa mãn, hàm sử dụng member.edit(mute=True) để mute người dùng đó và gửi một tin nhắn để thông báo cho tác giả message rằng người dùng đã được tắt tiếng trong kênh giọng nói nào đó. Tin nhắn này chứa tên của kênh giọng nói mà người dùng đã bị tắt tiếng bằng cách sử dụng f-string.

!unmute <người dùng>: Bật tiếng lại cho một người dùng.

A picture containing text, screenshot, font

Description automatically generated

**Hình 52: Đoạn chương trình Python thực thi bật tiếng cho người dùng của Bot**

* Command này sẽ unmute (bật tiếng) người dùng được chỉ định trong kênh giọng nói mà họ đang tham gia.
* Hàm unmute\_user\_vc được định nghĩa với hai tham số: self - đại diện cho chính Bot, ctx - đại diện cho context của tin nhắn gửi đến bot, member - đại diện cho người dùng mà bot sẽ unmute.
* Trước khi unmute một người dùng, hàm sẽ kiểm tra xem người dùng đó có đang kết nối đến kênh giọng nói không. Nếu không, Bot sẽ gửi một tin nhắn để thông báo cho tác giả rằng người dùng không đang kết nối đến kênh giọng nói nào cả.
* Nếu người dùng đang kết nối đến kênh giọng nói, hàm sử dụng member.edit(mute=False) để unmute người dùng đó và gửi một tin nhắn để thông báo cho tác giả rằng người dùng đã được unmute trong kênh giọng nói nào đó. Tin nhắn này chứa tên của kênh voice mà người dùng đã được unmute bằng cách sử dụng f-string. Để sử dụng lệnh này, người sử dụng cần có quyền kick\_members trên Discord server.

!move <người dùng> <kênh thoại>: Di chuyển một người dùng đến kênh thoại khác.

A picture containing text, screenshot, font

Description automatically generated

**Hình 53: Đoạn chương trình Python thực thi di chuyển người dùng của Bot**

* Dòng đầu tiên @commands.command() khai báo rằng đây là một lệnh của Bot.
* Tiếp theo, có hàm move với các tham số ctx (context) là ngữ cảnh của command và member là người dùng cần được di chuyển và channel là kênh thoại đích.
* Trong hàm move, đầu tiên kiểm tra xem người dùng đã có quyền di chuyển thành viên không bằng cách kiểm tra .ctx.author.guild\_permissions.move\_members.
* Nếu người dùng không phải chủ sở hữu của Server và member.top\_role lớn hơn hoặc bằng ctx.author.top\_role thì người dùng sẽ không được phép di chuyển thành viên đó.
* Nếu người dùng không đang trong kênh thoại hoặc đã rời khỏi kênh thoại thì sẽ có thông báo phản hồi về việc di chuyển không thành công.
* Nếu đủ điều kiện để di chuyển, hàm member.move\_to(channel) sẽ được sử dụng để di chuyển người dùng đến kênh thoại mới và có thông báo về việc di chuyển thành công.
* Cuối cùng, nếu người dùng không có quyền di chuyển thành viên, một thông báo phản hồi tương ứng sẽ được gửi.

Nếu trong quá trình di chuyển xảy ra lỗi, hàm move\_error sẽ được thực thi để phản hồi lại với người dùng. Hàm này xử lý các lỗi có thể xảy ra trong quá trình thực hiện command, bao gồm: thiếu đối số cần thiết, sai đối số truyền vào hoặc thiếu quyền di chuyển thành viên.

!disconnect <người dùng>: Buộc một người dùng bị ngắt kết nối khỏi kênh thoại.

A screen shot of a computer program

Description automatically generated with medium confidence

**Hình 54: Đoạn chương trình Python thực thi ngắt người dùng khỏi kênh hội thoại của Bot**

* ctx: là context của lệnh, chứa thông tin về message và server context.
* member: là thành viên trong Server được truyền vào để ngắt kết nối giữa thành viên này và voice channel.
* Command kiểm tra quyền hạn của người gọi command thông qua:
  + ctx.author.guild\_permissions.kick\_members
  + ctx.author.guild\_permissions.administrator
  + Nếu có quyền hạn, command sẽ thực hiện việc ngắt kết nối giữa thành viên và kênh hội thoại, và trả về một thông báo xác nhận. Nếu không có quyền hạn, Bot sẽ trả về một thông báo lỗi. Nếu không tìm thấy voice client trong server, Bot sẽ thử kết nối đến kênh hội thoại của thành viên được truyền vào.

**2.2.6. Triển khai Bot**

**1. Triển khai Bot trên máy chủ:**

* Chọn một dịch vụ lưu trữ phù hợp với nhu cầu, ví dụ như Heroku, AWS, Google Cloud Platform, Azure, DigitalOcean, Linode, Vultr, v.v.
* Tạo một tài khoản và đăng nhập vào dịch vụ lưu trữ. Sau đó, tạo một ứng dụng mới và cấu hình các thông số cần thiết, ví dụ như phiên bản Python, biến môi trường, cổng kết nối, v.v. Các dịch vụ lưu trữ thường cung cấp giao diện quản lý để bạn có thể dễ dàng cấu hình.
* Tải mã nguồn của Bot lên dịch vụ lưu trữ và cài đặt các thư viện cần thiết. Chúng ta có thể sử dụng Git để tải mã nguồn từ kho lưu trữ của hoặc tải mã nguồn trực tiếp từ máy tính cá nhân lên dịch vụ lưu trữ. Sau đó, cài đặt các thư viện cần thiết bằng lệnh pip install.
* Cấu hình các biến môi trường cần thiết, ví dụ như token của Bot, cổng kết nối, v.v. Biến môi trường là các giá trị được sử dụng trong mã nguồn của Bot để cấu hình hoạt động của nó. Bạn có thể cấu hình các biến môi trường thông qua giao diện quản lý của dịch vụ lưu trữ hoặc thông qua tệp tin .env.
* Khởi động Bot và kiểm tra xem Bot đã hoạt động chính xác hay chưa. Chúng ta có thể sử dụng các công cụ như Log để theo dõi hoạt động của Bot.

**2. Chạy Bot cục bộ trên máy tính:**

* Cài đặt Python và pip trên máy tính cá nhân. Python là ngôn ngữ lập trình được sử dụng để viết Bot và pip là công cụ quản lý thư viện.
* Tải mã nguồn của Bot về máy tính của và cài đặt các thư viện cần thiết bằng lệnh pip install. Mã nguồn của Bot có thể được lưu trữ trên kho lưu trữ như Github hoặc Bitbucket, hoặc được tải trực tiếp từ máy tính.
* Cấu hình các biến môi trường cần thiết, ví dụ như token của Bot, cổng kết nối, v.v. Bạn có thể cấu hình các biến môi trường thông qua tệp tin .env.
* Khởi động Bot và kiểm tra xem Bot đã hoạt động chính xác hay chưa. Chúng ta có thể sử dụng các công cụ như Log để theo dõi hoạt động của Bot.
* Lưu ý: Khi triển khai Bot trên dịch vụ lưu trữ, chúng ta cần phải trả phí cho dịch vụ đó. Trong khi đó, khi chạy Bot cục bộ trên máy tính, chúng ta không cần phải trả phí nhưng Bot chỉ có thể hoạt động khi máy tính đang chạy và có kết nối Internet.

**3. Demo Bot**

Sau đây chúng ta sẽ chạy thử 1 vài lệnh để thấy chức năng:

* Người dùng Tuấn đã sử dụng set role test trong Server:

A picture containing text, screenshot, font

Description automatically generated

**Hình 55: Người dùng thực thi lệnh !createrole trong Server**

* Lệnh roll ngẫu nhiên ra số 3:

A picture containing text, screenshot, font, logo

Description automatically generated

**Hình 56: Người dùng thực hiện chức năng !roll trên Server**

A screenshot of a phone

Description automatically generated with medium confidence

**Hình 57: Người dùng thực thi chức năng !setname trên Server**

* Lệnh !setname đã thay đổi thành công tên của server thành “Testbot\_Demo”.

A screenshot of a phone

Description automatically generated with medium confidence

**Hình 58: Người dùng thực thi chức năng !date trên Server**

* Lệnh !date để hiện lên ngày tháng hôm nay.

A screenshot of a phone

Description automatically generated with medium confidence

**Hình 59: Người dùng thực thi chức năng !calc trên Server**

* Lệnh !calc tính ra kết quả.

A screenshot of a phone

Description automatically generated with medium confidence

**Hình 60: Người dùng thực thi chức năng !flip trên Server**

* Lệnh !flip cho ta được mặt ngửa.

A screenshot of a chat

Description automatically generated with low confidence

**Hình 61: Người dùng thực thi chức năng !setrole trên Server**

* Người dùng Tuấn đã !set roll cho người dùng PD thành role “test”

A picture containing text, screenshot, font

Description automatically generated

**Hình 62: Người dùng thực thi chức năng !chat trên Server**

* Lệnh !chat đã có thể trực tiếp trò chuyện và hỏi nhiều vấn đề với Bot thông qua OpenAI API.

**Tài liệu tham khảo**

Trang điện tử

[1] <https://www.youtube.com/watch?v=c42T5wKSztQ&list=WL&index=2>

[2] <https://www.youtube.com/watch?v=SPTfmiYiuok>

[3] <https://www.youtube.com/playlist?list=PL-7Dfw57ZZVRB4N7VWPjmT0Q-2FIMNBMP>

Tài liệu khác

[1] Slide bài giảng Ngôn ngữ lập trình Python của thầy Trịnh Tấn Đạt.