МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА DISCORD-БОТА ДЛЯ АВТОМАТИЧНОЇ ВИДАЧІ РОЛЕЙ»**

Виконав: студент 4 курсу, групи 6.1229
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми комп'ютерні науки
(назва освітньої програми)
Р. О. Черновол
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії, доцент, к.т.н. Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 комп'ютерні науки
(шифр і назва)

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Чопоров С.В.
(підпис)

" ___ " _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ СТУДЕНТОВІ

Черноволу Руслану Олександровичу

1. Тема роботи Розробка Discord-бота для автоматичної видачі ролей

Керівник роботи Матвіїшина Надія Вікторівна, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація DISCORD бота
4. Проєктування та реалізація бота
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 30.01.2023

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етпів кваліфікаційної роботи	Строк виконання етапів роботи	Примітки
1.	Розробка плану роботи	30.01.2023	
2.	Збір вихідних даних	20.01.2023	
3.	Обробка методичних та теоретичних джерел	12.02.2023	
4.	Розробка першого та другого розділів	20.02.2023	
5.	Розробка третього розділу	24.03.2023	
6.	Оформлення та нормокнтроль кваліфікаційної роботи бакалавра	20.05.2023	
7.	Захист кваліфікаційної роботи	20.06.2023	

Студент _____ Р.О. Черновол
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.В. Матвіїшина
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка Discord-бота для автоматичної видачі ролей»: 34 с., 16 рис., 7 джерел.

БОТ, РОЗПОДІЛ РОЛЕЙ, API, DISCORD, DISNAKE, PYTHON

Об’єкт дослідження – досліджування процесу створення Discord бота для автоматичної видачі ролей, включаючи керування боту, створення спеціальних команд для взаємодії з користувачами.

Мета роботи: розробка боту автоматичної видачі ролей для популярного месенджера Discord з використанням розширеного Discord API.

Метод дослідження – аналіз, описовий, порівняльний.

Результатом кваліфікаційної роботи є бот для видачі ролей, розроблений для сервісу Discord, і який має наступний функціонал: модерація сервера, управління правами на сервері, можливість запуску двох міні ігор. Проведено тестування боту на сервері Discord.

Розроблений Discord бот може бути корисним для спрощення управління сервером, підвищення взаємодії з користувачем та автоматизації багатьох завдань.

SUMMARY

Bachelor's qualification theses "Development an Autorole Discord Bot": 34 pages, 16 figures, 7 sources.

BOT, DISTRIBUTION OF ROLES, API, DISCORD, DISNAKE, PYTHON

The object of the study is to investigate the process of creating a Discord bot for automatic role assignment, including managing the bot, creating special commands for interacting with users.

The goal of the work: the development of a bot for the automatic issuance of roles for the popular Discord messenger using the extended Discord API.

The research method is analysis, descriptive, comparative.

The result of the qualification work is a bot for issuing roles, developed for the Discord service, and which has the following functionality: server moderation, rights management on the server, the ability to run two mini games. The bot has been tested on the Discord server.

A developed Discord bot can be useful for simplifying server management, improving user interaction, and automating many tasks

ЗМІСТ

ВСТУП

Discord бот – це програма, що допомагає автоматизувати багато процесів на сервері Discord, такі як розсилання повідомлень, обробка команд та взаємодія з користувачами. Боти можуть бути розроблені на різних мовах програмування та платформах, але часто використовується Python та Node.js.

Discord боти є дуже популярними серед адміністраторів серверів та користувачів, які хочуть забезпечити більш ефективну та зручну роботу з сервером. Вони можуть допомогти у підтримці порядку, зборі та аналізі статистики, автоматизації деяких процесів, розвагах та багато іншого.

Крім цього, для розробки власних ботів на Discord можна використовувати різні API, такі як Discord API, який дозволяє взаємодіяти з сервером та його користувачами, та інші зовнішні сервіси для забезпечення додаткової функціональності. Розробка власного бота на Discord може бути чудовою можливістю для практики програмування та створення корисних інструментів для спільноти.

Розробка Discord ботів полегшує та допомагає людям користуватися самим Discord. Discord – це один з найвідоміших та використовуваних додатків для спілкування людей на відстані, особливо має великий попит у геймерів, оскільки дуже сильно полегшує користування таким додатком, аніж інші додатки. Discord боти – це програмні додатки, які дозволяють автоматизувати та спростити різні завдання, пов'язані з керуванням Discord-серверами та спілкуванням з користувачами. Боти являють собою автоматичні облікові записи на Discord, які можуть виконувати певні завдання на сервері та взаємодіяти з користувачем через текстові повідомлення. Боти можуть використовуватися для різних завдань, від модерації до ігор та повідомлень.

Discord боти розробляються для автоматизації та спрощення різних завдань, пов'язаних з керуванням Discord-серверами та спілкуванням з користувачами.

Наприклад, боти можуть використовуватися для:

- модерації: боти можуть видаляти повідомлення з образливим текстом, спамом, посиланнями на небажаний контент тощо;
- управління: боти можуть керувати правами доступу користувачів, створювати та видаляти канали, змінювати налаштування сервера тощо;
- розваги: боти можуть створювати ігри, опитування, голосування та інші інтерактивні елементи на сервері;
- повідомлень: боти можуть відстежувати новини, оновлення, події та сповіщати користувачів про них;
- інформації: боти можуть надавати інформацію про погоду, новини, курси валют та інші події.

Боти можуть бути розроблені як відкритими проектами, доступними для всіх, так і закритими, доступними лише для певних користувачів. Розробка Discord бота може бути корисна для покращення управління сервером, підвищення взаємодії з користувачем та автоматизації багатьох завдань.

1 ОСОБЛИВОСТІ СЕРВІСУ DISCORD

1.1 Популярні сервіси для спілкування

Месенджери, такі як Telegram, Viber, Skype та інші (рис. 1.1), відіграють велику роль у сучасному спілкуванні, забезпечуючи миттєвий обмін повідомленнями в реальному часі через Інтернет. Вони дозволяють передавати текстові повідомлення, звукові сигнали, зображення та відео. Свою популярність вони здобули завдяки можливості проводити як текстові, так і голосові чати, а також організовувати конференції для багатьох учасників. Більшість месенджерів пропонують мобільні додатки, що дозволяє порівняти їх зі стандартною функцією надсилання SMS, але без додаткової плати за повідомлення, а також здатність відправляти мультимедійний контент.

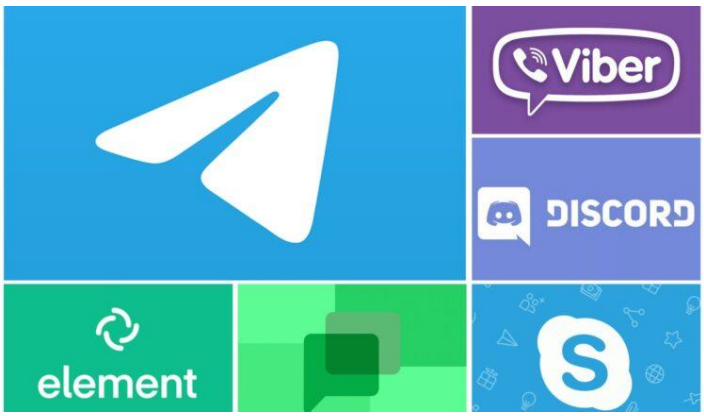

Рисунок 1.1 – Приклади месенджерів

Один з таких месенджерів – сервіс Discord, започаткований 6 березня 2016 року, який швидко здобув популярність серед геймерських спільнот та розробників. Discord поєднує найкращі функції сучасних месенджерів для спілкування. Кожен користувач може створити власний сервер, що називається гільдією. Гільдія представляє собою ізольовану колекцію користувачів і каналів на сервері Discord, пов'язану з певним географічним

регіоном. Власник сервера має необмежені права. На сервері можна створювати безліч текстових і голосових чатів, запрошувати інших людей за допомогою тимчасових або постійних посилань, а також налаштовувати рівні доступу. Однак, однією з ключових особливостей Discord є можливість створення та підключення ботів до сервера.

1.2 Етапи розробки боту на Discord

Розробка Discord бота – це процес створення програмного забезпечення, яке допомагає користувачам спілкуватися та керувати своїм сервером на Discord. Основними етапами розробки бота є:

- концептуалізація: визначення функцій бота та його поведінки на сервері;
- проектування: розробка структури бота та його алгоритмів;
- розробка: програмування бота з використанням мов програмування, таких як Python [1] або JavaScript;
- тестування: перевірка бота на наявність помилок та недоліків, виправлення помилок;
- випуск: випуск бота на сервер.

У розробці Discord бота зазвичай задіяні програмісти та фахівці зі знанням мов програмування та роботи з API Discord [4]. Для розробки бота використовуються різноманітні інструменти, такі як бібліотеки та фреймворки для розробки ботів, редактори коду та середовища розробки.

Discord боти можуть мати різні функції, такі як автоматичне привітання нових користувачів, нагадування про події, створення опитувань та голосувань, автоматичне розсилання повідомлень та інше. Користувачі можуть розробляти власні боти або використовувати готові боти з відкритих джерел.

Discord боти допомагають покращити спілкування на сервері та забезпечують більш зручний та ефективний спосіб керування сервером. Однак, як і в будь-якій галузі, існують певні проблеми, пов'язані з використанням ботів на Discord. Наприклад, деякі боти можуть бути зловмисними та можуть використовуватися для збирання конфіденційної інформації користувачів. Залежно від потреб користувача, боти на Discord можуть зробити досвід користування більш приємним та зручним. Проте, важливо пам'ятати про безпеку та використовувати тільки надійні та перевірені боти на Discord.

Щоб створити Discord [3] бота, можна використовувати різні мови програмування, такі як Python, JavaScript, Java, C++, C# та інші. Для зручності розробки ботів були створені спеціальні бібліотеки та фреймворки, такі як Discord.py, Discord.js, JDA, Discord4J та інші.

Одним із переваг використання Discord ботів є їхній широкий спектр можливостей та легкість використання. Користувачі можуть легко налаштувати бота під свої потреби, додавши до нього різноманітні плагіни та розширення. Крім того, багато геймерів використовують Discord ботів для автоматизації рутинних задач та покращення взаємодії з іншими гравцями.

Загалом, Discord боти є потужним інструментом для геймерів та розробників, які шукають швидкий та простий спосіб створити бота для спілкування та автоматизації геймплею.

2 СТВОРЕННЯ ПРОЕКТУ

2.1 Характеристики Discord боту

Discord бот – це програмний інструмент, який дозволяє створювати автоматизовані процеси в сервісі Discord. Серед особливостей Discord бота можна виділити:

- можливості автоматизації: Discord бот дозволяє створювати різні автоматизовані процеси, такі як реагування на конкретні повідомлення, вітання нових користувачів, автоматичне створення каналів та інші;
- легкість використання: Discord бот має простий та зрозумілий інтерфейс, що дозволяє легко створювати та редагувати автоматизовані процеси;
- кросплатформність: Discord бот може бути використаний на різних платформах, таких як Windows, Mac, Linux, Android, iOS та інші;
- підтримка різних мов програмування: Discord бот може бути написаний на різних мовах програмування, таких як Python, JavaScript, Java, C# та інші [7];
- велика спільнота: Discord боти мають велику спільноту розробників, де можна знайти багато корисної інформації та підказок щодо розробки бота;
- бібліотеки: Для розробки Discord ботів можна використовувати різні бібліотеки, які спрощують процес розробки та дозволяють легко працювати з API Discord;
- можливості розширення: Discord бот дозволяє створювати власні команди та функції для розширення можливостей бота.

Одна з головних переваг Discord ботів полягає у їхній простоті у використанні. Це дозволяє користувачам безпроблемно взаємодіяти з ботом та отримувати необхідну інформацію або виконувати необхідні дії.

За допомогою вбудованих функцій, таких як відправка повідомлень, опитування, розклади та інші, боти можуть допомогти користувачам ефективніше управляти своїм часом та діями на Discord.

Крім того, Discord боти мають велику спільноту розробників, яка надає багато корисної інформації та ресурсів для тих, хто хоче навчитися розробляти свої власні боти. У спільноті Discord ботів можна знайти відповіді на багато питань, які стосуються розробки та налаштування ботів, а також поділитися своїми ідеями та досвідом з іншими розробниками. Крім того, більшість розробників ботів надають відкритий доступ до свого вихідного коду, що дозволяє іншим користувачам адаптувати та вдосконалювати ботів за своїми потребами.

Також Discord боти мають широкий спектр функцій, які можна додати до бота для покращення його функціональності. Наприклад, можна додати інтерактивні ігри, музичні програвачі, різноманіт

Discord – є безкоштовною програмою, яку можна використовувати для навчання та створення ботів. Among Us - розрахована на багато користувачів гра, в якій гравці грають за членів команди на космічному кораблі, один з яких є зрадником.

Одна з основних переваг Discord бота полягає в тому, що він легкий для використання і має зрозумілий інтерфейс. Крім того, є безліч корисних ресурсів та документації, що допоможуть розробникам швидко навчитися роботі з ботом та зрозуміти його можливості.

До найпопулярніших Discord ботів можна віднести:

– MEE6 – бот для управління сервером, який дозволяє автоматизувати деякі рутинні завдання, такі як вітання нових користувачів, автоматичне видалення спаму та інші;
– Dyno – бот, що надає багато корисних функцій для управління сервером, таких як автоматичні повідомлення про нові ролі, заборона на куріння в текстових каналах, автоматичні повідомлення про нові повідомлення та інші;

– Groovy – бот для відтворення музики з YouTube та інших популярних сервісів, який дозволяє відтворювати музику в голосових каналах та змішувати її з іншими звуками;
– Pokétwo – бот, який додає в Discord елементи гри Pokémon, такі як можливість ловити покемонів та проводити битви між ними.

Загалом, Discord боти стали незамінним інструментом для управління сервером та підвищення взаємодії користувачів.

2.2 Налаштування конфігурацій проекту

Найголовнішим аспектом при налаштуванні Discord бота є зрозуміння, яку аудиторію він буде обслуговувати. Для розробки бота використовується Discord API [4], що забезпечує взаємодію з платформою Discord та дозволяє створювати функціональні боти для спілкування з користувачами. Для налаштування бота необхідно визначити його параметри та додаткові функції, що будуть відповідати потребам користувачів. Основні параметри, які необхідно налаштувати, це токен бота, його доступ до серверів та каналів, а також можливості взаємодії з користувачами. Також, варто враховувати особливості платформи Discord [3], що можуть вплинути на роботу бота, такі як обмеження на кількість запитів до API та можливості інтеграції з іншими сервісами.

Основні особливості Discord бота:

– вибір мови програмування: Discord боти можуть бути написані на різних мовах програмування, таких як JavaScript, Python, Ruby, C ++ та інші;
– авторизація бота: для того, щоб бот міг працювати на сервері, потрібно авторизувати його за допомогою спеціального токена;
– реакції та повідомлення: бот може реагувати на певні повідомлення та відповідати на них, а також відправляти повідомлення до користувачів на сервері;

– команди: бот може виконувати різні команди, які програміст заздалегідь заложив в його коді, наприклад, вітати нових користувачів, відправляти повідомлення на певний час або запускати ігри на сервері;
– інтеграція з іншими сервісами: бот може бути інтегрований з іншими сервісами, наприклад, з YouTube, Spotify або GitHub [5], що дозволяє зручно отримувати інформацію та здійснювати дії безпосередньо на сервері.

Боти для Discord можуть бути написані різними мовами програмування, включаючи JavaScript, Python та Ruby. Вони можуть бути запущені на різних платформах, таких як Linux, Windows або MacOS. Але, як і у випадку з WebGL, для Discord ботів найкраще використовувати платформу Node.js, оскільки вона надає кращу підтримку та більше можливостей.

2.3 Створення боту для Discord

Першим кроком є створення нового додатку на платформі Discord та отримання токена доступу. Для цього потрібно створити обліковий запис розробника на Discord і перейти до розділу "Додатки". Тут можна створити новий додаток та отримати токен доступу до API Discord.

Наступним кроком є вибір мови програмування та бібліотеки, яка буде використовуватися для розробки бота. Discord надає документацію та допоміжні бібліотеки для різних мов програмування, таких як Python, JavaScript, Java та інших.

Етапи, необхідні для початку розробки власного боту для Discord

а) створіть Discord-бота

Перш за все, потрібно зареєструвати нового бота на платформі Discord, якщо це не зроблено. Для цього можна скористатися інструкцією на сайті Discord, або ж знайти відповідну інформацію в Інтернеті. Після реєстрації бота отримано токен, який знадобиться для підключення до API Discord.

б) встановлення Python

Другим кроком є встановлення Python на комп'ютері; можна завантажити Python з офіційного сайту Python.org.

в) встановлення бібліотеки Disnake [2]

Disnake – це бібліотека для розробки ботів для Discord на Python, яка є форком бібліотеки Discord.py. Користувач може встановити бібліотеку Disnake за допомогою менеджера пакетів Python, наприклад pip. Для цього виконайте наступну команду у терміналі:

pip install disnake

г) підключіть бота до API Discord

Для підключення бота до API Discord ви можете скористатися токеном, який ви отримали на першому кроці. Створіть новий файл Python та додайте наступний код:

```
python
import disnake
from disnake.ext import commands

TOKEN = 'ваш_токен'

bot = commands.Bot(command_prefix='!')

@bot.event
async def on_ready():
    print(f'Logged in as {bot.user.name} ({bot.user.id})')

bot.run(TOKEN)
```

д) після додавання коду, потрібно вставити унікальній токен у відповідний рядок.

Цей код підключить бота до API Discord та дозволить йому слідкувати за різними подіями, такими як підключення до сервера або приймання

повідомлень в чаті. Команда command_prefix встановлює префікс для команд бота, які будуть виконуватися при виклику з цим префіксом.

Після написання коду можна перевірити, чи все працює, відлагодити його та перевірити, чи бот відповідає на всі запити користувачів.

Також є багато не маловажних деталей в розробці ботів, які напряму впливають на створення та реалізацію.

- обробка подій – важлива деталь розробки бота, оскільки це дозволяє боту реагувати на дії користувачів. Python має багато бібліотек для обробки подій, наприклад, discord.py, яка є однією з найбільш популярних бібліотек для створення ботів Discord на Python;
- безпека – інша важлива деталь, яку потрібно враховувати під час розробки бота. Для забезпечення безпеки можна використовувати різноманітні механізми, наприклад, обмеження доступу до деяких функцій, використання SSL-шифрування для захисту передачі даних, перевірку введених користувачем даних тощо;
- функціональність – важливо зрозуміти, які функції потрібні користувачам та які функції можуть бути використані для поліпшення користувацького досвіду. Бот може виконувати різні функції, наприклад, вітати нових користувачів на сервері, реагувати на команди користувачів, відправляти повідомлення тощо;
- оптимізація – важливо забезпечити оптимальну роботу бота. Це може бути досягнуто за допомогою правильного розподілу завдань та використання асинхронних запитів. В більшості бібліотек для роботи з Discord, таких як discord.py, є можливість використання асинхронної розробки.

3 РЕАЛІЗАЦІЯ ПРОЕКТУ

Створення ботів для Discord на Python – це процес розробки програмного забезпечення, яке взаємодіє з API Discord для створення робочих ботів, які можуть виконувати різні завдання на сервері. Python є однією з популярних мов програмування для створення ботів для Discord, адже вона має простий синтаксис, широкий набір бібліотек та інструментів розробки, які значно полегшують розробку ботів.

Під час роботи з цим проектом, використовується така мова програмування як Python.

Під час розробки розглянуто такі теми:

– створення бота на порталі https://discord.com/developers/applications/;
– створення файлу програми, встановлення та додаєм бібліотеку disnake;
– створення допоміжних команд для боту;
– створення основних команд для бота;
– перевірка роботи усіх команд та систем

3.1 Створення бота на порталі Discord Developers

Щоб почати працювати у Discord, потрібно зареєструватися на офіційному сайті розробника, завантажити Discord. Після завантаження додатку, необхідно зайти на сайт для розробників (https://discord.com/developers/applications), та пройти авторизацію, а потім натиснути на “New application” у правому верхньому боці (рис. 3.1).

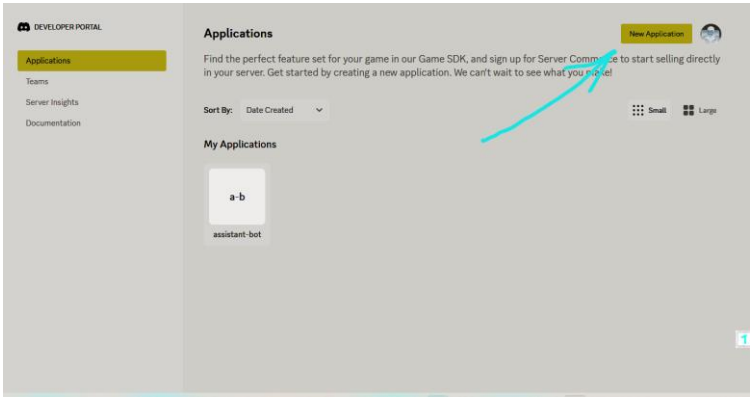

Рисунок 3.1 – Створення проекту

Після того як натиснули “New application” необхідно дати назву боту (наприклад, assistant-bot) (рис. 3.2).

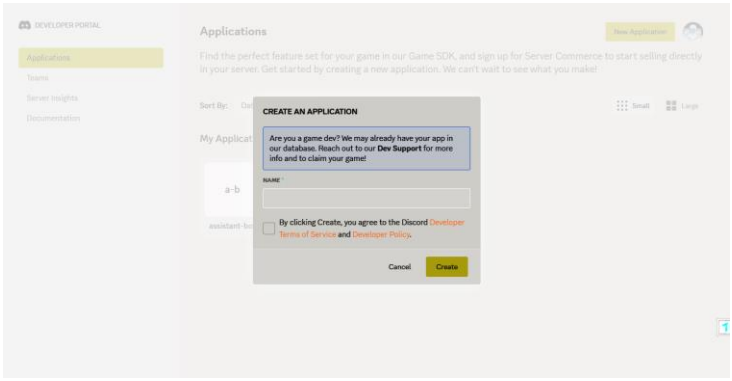

Рисунок 3.2 – Назва боту

Після того як назвали бота та прийняли умови, користувач зможе здійснити налаштування, в яких можливо змінювати його ім’я, або поставити фото (рис. 3.3).

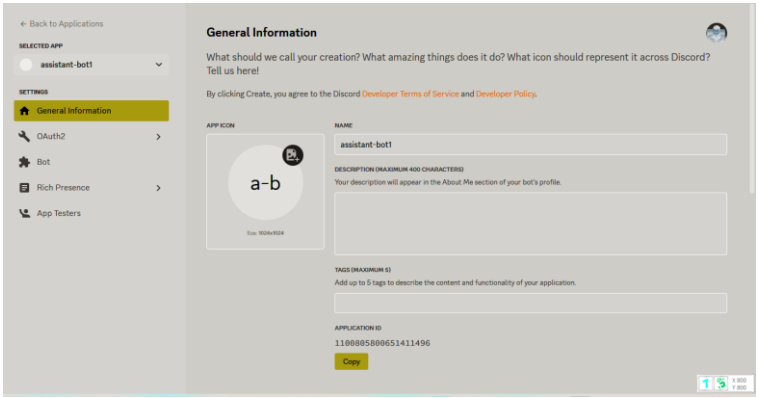

Рисунок 3.3 – Початкове вікно бота

3.2 Створення файлу програми для боту

Після додавання боту на сервер, потрібно створити файл, при запуску якого бот буде авторизовуватись на сервері. Для цього потрібно створити файл з розширенням .py (рис. 3.4).

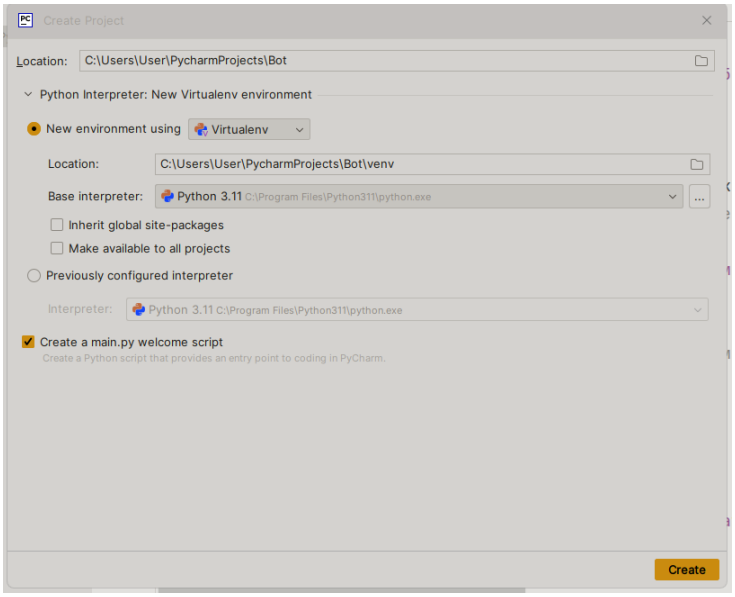

Рисунок 3.4 – Створення файлу

Нам потрібна основна бібліотека, за допомогою якої, ми будемо реалізовувати код, це бібліотека Disnake.

Disnake [2]– це асинхронна бібліотека для створення ботів для Discord на мові програмування Python. Вона є покращенням популярної бібліотеки discord.py і містить більшість її функцій та додаткові інструменти для розробки.

disnake забезпечує підтримку всіх основних функцій Discord API, таких як керування каналами, ролями, серверами, повідомленнями та користувачами. Крім того, вона пропонує додаткові можливості для створення віджетів, інтерактивних повідомлень та багато іншого.

Основною перевагою disnake є її швидкодія та підтримка асинхронної розробки, що дозволяє легко і швидко реагувати на події в Discord API та

запускати декілька операцій одночасно. Також disnake має документацію та активну спільноту, що допомагає розробникам у вирішенні проблем та розв'язанні питань.

Для того щоб встановити цю бібліотеку, нам потрібно відкрити командну строку, та ввести таку команду (рис. 3.5):

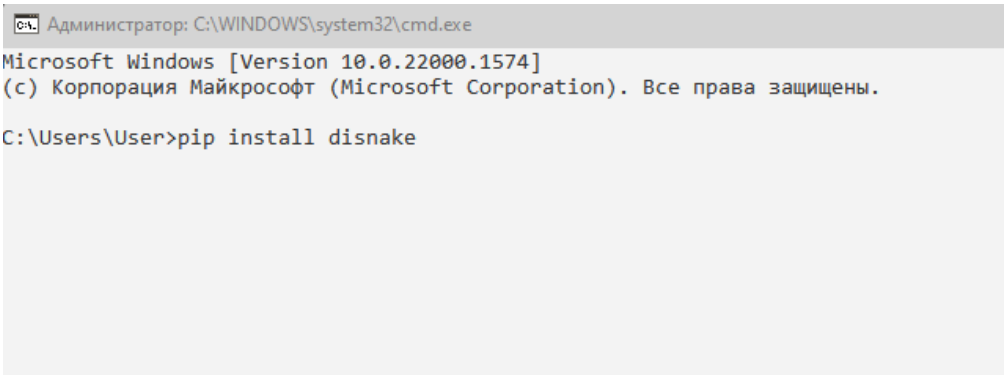

Рисунок 3.5 – Завантаження бібліотеки disnake

Якщо бібліотека встановлена, можна приступати до наступних дій.

Потрібно створити блок коду, який буде відповідати за запуск боту, у якому буде знаходитись наданий токен (рис. 3.6).

```
# Bot online!
@bot.event
async def on_ready():
    print(f'Bot {bot.user} online!')

bot.run('TOKEN')
```

Рисунок 3.6 – Створення коду для підключення бота

Замість напису TOKEN треба вставити той токен який було взято з сайту до цього.

Перевіряємо бота, запускаємо програму. У терміналі з’явилося таке повідомлення (рис. 3.7).

```
Bot assistant-bot#0826 is ready to work!
```

Рисунок 3.7 – Відображення готовності роботи бота

3.3 Створення інформаційні команди для боту

Боту, що розробляється, може знадобитись така команда, як “бот”, яка буде видавати основну інформацію про автора, та самого бота (Рис. 3.8).

Код для цієї команди:

```
@bot.command(name='бот')
async def bot_info(ctx):
    # створюємо embed повідомлення
    embed = disnake.Embed(title="інформація про бота", description="Нижче наведена інформація про бота:")
    embed.set_author(name="Черновол Руслан Олександрович")
    embed.add_field(name="Discord", value="Це найпопулярніша програма у світі", inline=False)
    embed.add_field(name="Для отримання більшої інформації для бота напишіть", value="!команда", inline=False)
    embed.add_field(name="Заклад", value="Запорізький Національний Університет", inline=False)
    embed.add_field(name="Дані", value="Математический факультет, компьютерные науки, група 6.1229", inline=False)
    embed.add_field(name="Опис",
                    value="Це командний Bot створений мовою Python за допомогою бібліотеки disnake для сервера Discord.",
                    inline=False)
```

```
    # відправляємо embed повідомлення до каналу
    await ctx.send(embed=embed)
```

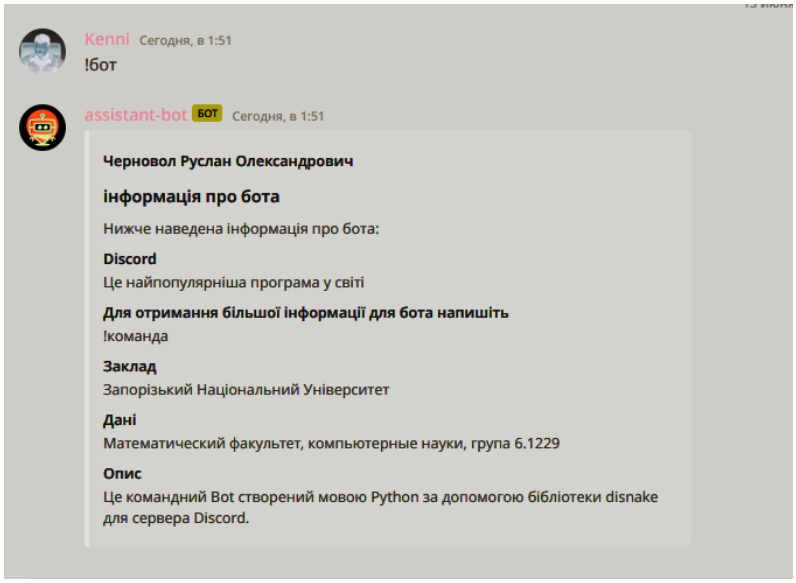

Рисунок 3.8 – Виконання команди “бот”

Також необхідною є команда, яка буде показувати усі доступні у цьому боті команди, вона буде викликатися як “!команди” (рис 3.9).

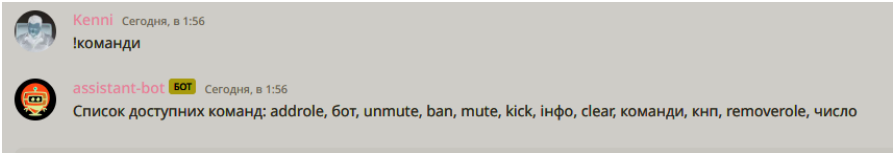

Рисунок 3.9 – Виконання команди “Команди"

```
    # Створюємо команду для перегляду списку команд
    @bot.command(name='команди')
    async def show_commands(ctx):
        # Список всіх команд
        command_list = [f'{command.name}' for command in bot.commands]

        # Відправляємо список команд в канал
```

```
        await ctx.send(f'Список доступних команд: {", ".join(command_list)}')
```

3.4 Створюємо основні команди для боту

Для роботи боту потрібно створити основні команди які він буде виконувати:

– Привітання нових користувачів.

– Видача ролей людям, для облегшення пошуку співрозмовників.

– Запуск міні ігор.

– Модерацію (очищення чату, або навіть можливість виганяти людей за порушення правил серверу).

Для вітання нових користувачів розроблено команду :

```
@bot.event
async def on_member_join(member):

    channel = member.guild.system_channel
    if channel is not None:
        # Відправляємо привітальне повідомлення
        await channel.send(f'Привіт, {member.mention}! Ласкаво просимо до нашого серверу!')
```

Код, який дає змогу боту видавати ролі користувачам згідно з тим, як вони реагують на пост на сервері (рис. 3.10):

```
# Словник відповідності емодзі та ролей
emoji_to_role = {
    '🤜': 1100476709574279229,  # ID ролі для 🤜 емодзі
```

```
    '🏎️': 1100477003800531125,  # ID ролі для 🏎️ емодзі
    '□': 1100477213570236507   # ID ролі для □ емодзі
}

# ID повідомлення, на яке потрібно реагувати
message_id = 1100494282424586452

@bot.event
async def on_raw_reaction_add(payload: disnake.RawReactionActionEvent):
    if str(payload.emoji) in emoji_to_role.keys() and payload.message_id == message_id:
        guild = bot.get_guild(payload.guild_id)
        member = guild.get_member(payload.user_id)
        role_id = emoji_to_role[str(payload.emoji)]
        role = guild.get_role(role_id)
        await member.add_roles(role)
```

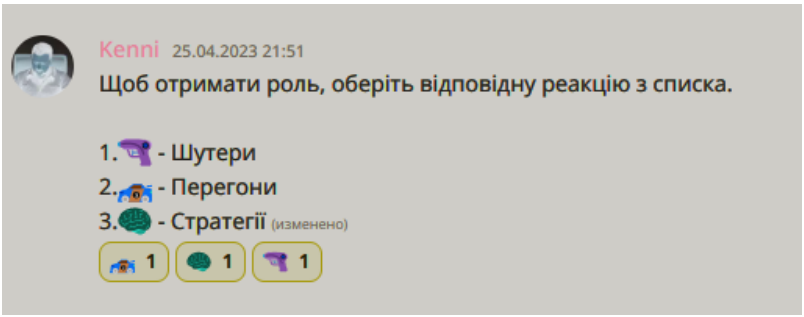

Рисунок 3.10 – Повідомлення для вибору ролей користувачами

Потім для легкості було б не погано додати йому таку можливість видавати ролі з допомогою команди !addrole (рис.3.11):

```
@bot.command(name='addrole')
```

```
async def add_role(ctx, role: disnake.Role, member: disnake.Member = None):
    # Перевіряємо, чи має користувач право на керування ролями
    if not ctx.author.guild_permissions.manage_roles:
        await ctx.send('У вас немає прав на управління ролями!')
        return

    # Якщо користувач не вказав користувача, використовуємо автора повідомлення
    if member is None:
        member = ctx.author

    # Додаємо роль користувачеві
    await member.add_roles(role)
    await ctx.send(f'Роль "{role.name}" надана користувачу {member.mention}!')
```

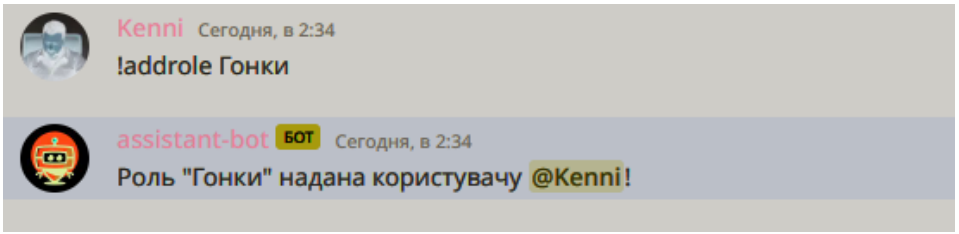

Рисунок 3.11 – Виконання команди “addrole”

Та додамємо до цього команду !removerole щоб можна було і забрати роль (рис.3.12):

```
@bot.command(name='removerole')
async def remove_role(ctx, role: disnake.Role, member: disnake.Member = None):
```

```
    # Проверяем, имеет ли пользователь право на управление ролями
    if not ctx.author.guild_permissions.manage_roles:
        await ctx.send('У вас нет прав на управление ролями!')
        return

    # Если пользователь не указал пользователя, используем автора сообщения
    if member is None:
        member = ctx.author

    # Удаляем роль у пользователя
    await member.remove_roles(role)
    await ctx.send(f'Роль "{role.name}" удалена у пользователя {member.mention}!')
```

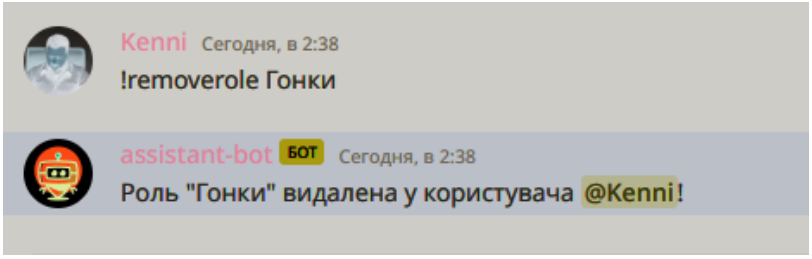

Рисунок 3.12 – Виконання команди “removerole”

Для розваг додано дві відомі міні гри, такі як вгадай число від 1 до 100 та камінь, ножиці, папір (рис.3.13, 3.14):

```
@bot.command(name='кнп')
async def rps(ctx):
    player_score = 0
    bot_score = 0
    choices = ['камінь', 'папір', 'ножиці']
```

```
        while player_score < 5 and bot_score < 5:
            bot_choice = random.choice(choices)
            await ctx.send('Камінь, папір чи ножиці?')
            try:
                player_choice = await bot.wait_for('message', timeout=30.0,
check=lambda m: m.author == ctx.author and m.content.lower() in choices)
            except asyncio.TimeoutError:
                await ctx.send('Вибач, але ти надто довго реагував.')
                return
            player_choice = player_choice.content.lower()
            await ctx.send('Я вибрав {}'.format(bot_choice))
            if bot_choice == player_choice:
                await ctx.send('Ничья!')
            elif (bot_choice == 'камінь' and player_choice == 'ножиці' or
                  bot_choice == 'папір' and player_choice == 'камінь' or
                  bot_choice == 'ножиці' and player_choice == 'папір'):
                bot_score += 1
                await ctx.send('Я переміг, мій рахунок зараз
{}.'.format(bot_score))
            else:
                player_score += 1
                await ctx.send('Ти переміг, твій рахунок зараз
{}.'.format(player_score))
        if player_score > bot_score:
            await ctx.send('Вітаю, ти переміг!')
        else:
            await ctx.send('Вибач, я виграв! Пощастить наступного разу.')
    @bot.command()
    async def guess(ctx):
        """Гра вгадай число"""
```

```
            # Генеруємо випадкове число
            number = random.randint(1, 100)

            # Починаємо гру
            await ctx.send('Вгадай число від 1 до 100!')

            # Перевіряємо відповідь користувача та даємо підсказки
            for i in range(5):
                def check(m):
                    return m.author == ctx.author and m.channel == ctx.channel
and m.content.isdigit()

                guess = await bot.wait_for('message', check=check)
                guess = int(guess.content)

                if guess < number:
                    await ctx.send('Занадто маленьке число, спробуй ще раз.')
                elif guess > number:
                    await ctx.send('Занадто велике число, спробуй ще раз.')
                else:
                    await ctx.send(f'Вітаю, ты вгадав число {number}!')
                    break
            else:
                await ctx.send(f'Нажаль, ти не вгадав число {number}. Спробуй
ще раз.')
```

а) камінь, ножиці, папір.

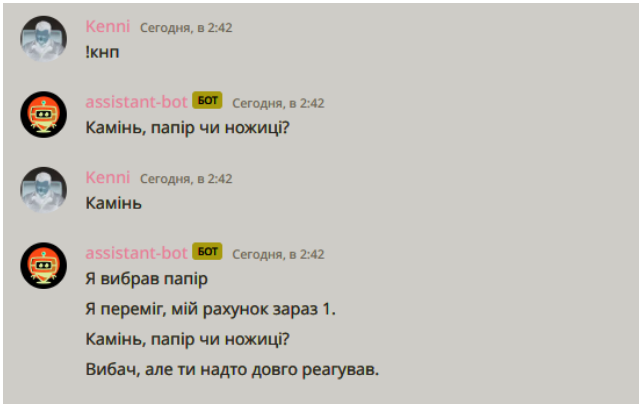

Рисунок 3.13 – Виконання команди “кнп”

б) вгадай число від 1 до 100.

Рисунок 3.14 – Виконання команди “число”

Боту додано роль модератора, щоб він міг за допомогою команд керувати чатом, виганяти, або блокувати людей (рис.3.15):

```
@bot.command()
@commands.has_permissions(kick_members=True)
async def kick(ctx, member: disnake.Member, *, reason=None):
    """Натиснути користувача."""
```

```
    await member.kick(reason=reason)
    await ctx.send(f'Користувач {member} був кікнутий із сервера.')

@bot.command()
@commands.has_permissions(ban_members=True)
async def ban(ctx, member: disnake.Member, *, reason=None):
    """Забанити користувача."""
    await member.ban(reason=reason)
    await ctx.send(f'Користувач {member} був забанений на сервері.')

@bot.command()
@commands.has_permissions(manage_messages=True)
async def clear(ctx, amount: int):
    """Видалити певну кількість повідомлень."""
    await ctx.channel.purge(limit=amount + 1)
    await ctx.send(f'Видалено {amount} повідомлень.')

@bot.command()
@commands.has_permissions(manage_messages=True)
async def mute(ctx, member: disnake.Member):
    """Заборонити користувачеві писати в чат."""
    role = disnake.utils.get(ctx.guild.roles, name='Muted')
    if not role:
        role = await ctx.guild.create_role(name='Muted')

    await member.add_roles(role)
    await ctx.send(f'Користувач {member} був замьючен.')

@bot.command()
```

```
    @commands.has_permissions(manage_messages=True)
    async def unmute(ctx, member: disnake.Member):
        """Дозволити користувачеві писати в чат."""
        role = disnake.utils.get(ctx.guild.roles, name='Muted')
        if not role:
            return

        await member.remove_roles(role)
await ctx.send(f'Користувач {member} був размьючен.')
```

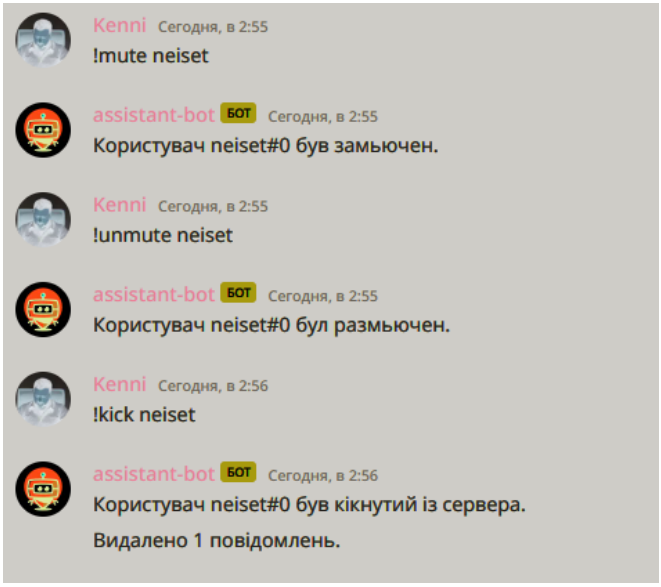

Рисунок 3.15 – Виконання команд модерації

ВИСНОВКИ

У цьому дослідженні було проаналізовано існуючі боти та їх методи розробки, що послужило основою для створення розширення Discord API. Використовуючи це розширення, був розроблений бот з декількома перевагами, такими як:

- більш гнучке налаштування бота: не потрібно перезавантажувати систему для простих змін, таких як включення або відключення компонентів;
- проста процедура створення та підключення модулів до бота;
- можливість складати функціональні модулі залежно від потреб сервера завдяки модульності;
- прозора система подій, що робить сервер більш безпечним;
- система дозволів, що дозволяє модулям взаємодіяти між собою.

Для бота були реалізовані модулі для модерації сервера, управління ботом. Бот був використаний на реальному сервері Discord для автоматизації модерування.

В майбутньому можна додати нові функціональні модулі та можливість використовувати штучний інтелект.

ПЕРЕЛІК ПОСИЛАНЬ

1. Васильєв О.М. Програмування мовою Python. Тернопіль : Навчальна книга – Богдан, 2019. 504 с.

2. Офіційна документація Disnake. URL : https://disnake.readthedocs.io/en/latest/ (дата звернення: 18.03.2023).

3. Офіційна документація Discord. URL : https://discord.readthedocs.io/en/latest/ (дата звернення: 29.03.2023).

4. Discord API. URL : https://discord.com/developers/docs/intro (дата звернення: 15.04.2023).

5. Discord Bot на GitHub. URL : https://github.com/topics/discord-bot.

6. Guild Resource. URL : https://discordapp.com/developers/docs/resources/guild (дата звернення: 23.04.2023).

7. Zelle J. Python Programming: An Introduction to Computer Science, 3rd Ed. Portland : Franklin, Beedle & Associates, 2016. 552 p.