Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка чат-бота для месенджера Discord на платформі Node.js, з використанням бази даних MongoDB

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Мац О. І.
(підпис) (прізвище та ініціали)

Керівник _____ Липак Г. І.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г. В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І. О.
(підпис) (прізвище та ініціали)

Рецензент _____ Стадник М. А.
(підпис) (прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«22» червня 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Мацу Олегу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка чат-бота для месенджера Discord на платформі Node.js з використанням бази даних MongoDB

Керівник роботи Липак Галина Ігорівна, к.н.с.к, доцент. кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 16 » березня 2022 року № 4/7-161

2. Термін подання студентом завершеної роботи 23.06.2022 р.

3. Вихідні дані до роботи Літературні та Інтернет-джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ ФУНКЦІОНУВАННЯ ЧАТ БОТІВ. 1.1 Поняття та сфери застосування ботів. 1.2 Класифікація ботів. 1.2.1 Чат боти, які здатні виокремлювати команди. 1.2.2 Чат боти, які здатні підтримувати діалог. 1.3 Приклади застосування ботів. 1.4 Приклад бота в месенджері Discord – MEE6. РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЕКТУ. 2.1 Початкові налаштування. 2.2 Початок роботи з порталом розробника Discord. 2.3 Встановлення Node.js. 2.4 Встановлення модулів Discord.js. 2.5 Встановлення FFMPEG. 2.6 Створення бази даних та налаштування сервера. 2.7 Написання сценарію поведінки для бота. 2.7.1 Написання базового коду та його налагодження. 2.7.2 Створення модераторського модуля. 2.7.3 Створення зображень з привітанням для нових учасників. 2.7.4 Створення модуля для програвання музики. 2.7.5 Створення системи економіки та рівневої системи. 2.8 Висновки до другого розділу. РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ. 3.1 Діяльність. Її види та розуміння в безпеці праці. 3.2 Вимоги безпеки до робочих місць для виконання робіт за персональним комп'ютером.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., к.т.н., доцент кафедри МТ		

7. Дата видачі завдання 24 січня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	24.01.2022	*Виконано*
2.	Підбір джерел про програми-боти, месенджер Discord	04.01.2022-30.01.2022	*Виконано*
	та СУБД MongoDB		
3.	Переклад та опрацювання джерел про JavaScript	31.01.2022-06.02.2022	*Виконано*
	та платформу Node.js		
4.	Виконання дослідження щодо ролі ботів у соціальних	07.02.2022-13.02.2022	*Виконано*
	мережах		
5.	Оформлення розділу «ТЕОРЕТИЧНІ ЗАСАДИ	14.02.2022-06.03.2022	*Виконано*
	ФУНКЦІОНУВАННЯ ЧАТ БОТІВ»		
6.	Оформлення розділу «ПРАКТИЧНА РЕАЛІЗАЦІЯ	07.03.2022-03.04.2022	*Виконано*
	ПРОЕКТУ»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	04.04.2022-17.04.2022	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	18.04.2022-01.05.2022	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	02.05.2022-15.05.2022	*Виконано*
10.	Нормоконтроль	16.05.2022-22.05.2022	*Виконано*
11.	Перевірка на плагіат	09.06.2022	*Виконано*
12.	Попередній захист кваліфікаційної роботи	10.06.2022	*Виконано*
13.	Захист кваліфікаційної роботи	23.06.2022	

Студент _____ Мац О. І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Липак Г. І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка чат-бота для месенджера Discord на платформі Node.js з використанням бази даних MongoDB // Кваліфікаційна робота освітнього рівня «Бакалавр» // Мац Олег Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-42 // Тернопіль, 2022 // С. – 54, рис. – 62, табл. – 2, додат. – 11, бібліогр. – 12.

Ключові слова: бот, месенджер, JavaScript, Node.js, сценарій поведінки, команда, модуль, функція, сервер.

В дипломній роботі розглянуто теоретико-методологічні та практичні аспекти створення чат бота на платформі Node.js.

Робота складається із вступу, трьох розділів, висновку, списку посилань на використану літературу і додатків.

У вступі обґрунтовується актуальність теми та формулюються задачі подальшого дослідження.

У першому розділі описуються загальні відомості про ботів та їхнє призначення.

У другому розділі описується інструментальне забезпечення, яке потрібне для реалізації проекту, а також усі етапи створення проекту.

Третій розділ присвячений охороні праці та безпеці життєдіяльності.

ANNOTATION

Development of a chatbot for the Discord messenger on the Node.js platform, using the MongoDB database // qualification work of the educational level «Bachelor» // Mats Oleh Ihorovich // Ternopil Ivan Pului National Technical University, faculty of computer information systems and software engineering, cathedra of computer sciences, group SNs-42 // Ternopil, 2022 // p. – 54, img. – 62, spreadsheets. – 2, appendix – 11, refferences – 12.

Keywords: bot, messenger, JavaScript, Node.js, action script, command, module, function, server.

In this qualification work methodological and practical aspects of creating a chat bot on the platform Node.js are described.

Paper work consist of introduction, three chapters, conclusion, list of references and appendix.

In the introduction, relevance of the topic is proved and the tasks of further research are formulated.

General information about bots and their purpose is described in the first chapter.

In the second chapter, tools needed for project implementation, as well as all stages of project creation are described.

The third section is devoted to the safety of work and health.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Месенджер – програма для обміну повідомленнями, яка має дружній для користувача інтерфейс та підтримує протоколи передачі даних

ЧБ – чат бот. Програма що імітує справжню розмову у вигляді текстових повідомлень.

ПЗ – програмне забезпечення.

Email – електронна пошта. Спосіб обміну інформацією, у вигляді цифрових листів.

API – Application Programming Interface. Набір методів (програм, протоколів та засобів) для взаємодії компонентів ПЗ.

B2B – Business-To-Business (бізнес для бізнесу).

B2C – Business-To-Consumer (бізнес для користувача).

IDE – Integrated Development Environment (інтегроване середовище розробки).

NLP – Natural-language processing (обробка природної мови). Напрям штучного інтелекту, математичної лінгвістики та інформатики. Вивчає «синтез природної мови», тобто перетворення будь-яких фраз вжитих людиною, у формальні команди.

NER – Named Entry Recognition (ідентифікація названого об’єкту) є підзавданням вилучення інформації, яке має на меті знайти та класифікувати названі об’єкти, згадані в неструктурованому тексті, у попередньо визначені категорії (імена людей, організації, вирази часу, кількості, грошові оцінки, відсотки тощо).

RSS – Rich Site Summary (короткий зміст сайту). XML-формат що використовують для інформації, яка часто оновлюється (наприклад новинних сайтів).

ЗМІСТ

ВСТУП

Актуальність теми. У процесі розвитку людини, більша частина знань отримується нею за допомогою обміну інформацією з іншими людьми. Це відбувається напряму – особисті зустрічі та спілкування, або дистанційно – за допомогою електронного листування або спеціальних програм – месенджерів. В більшості месенджерів на відміну від приватних повідомлень присутня функція групових чатів.

Там де з’являється велике скупчення людей, необхідно підтримувати порядок, виконувати очищення чату, видаляти заборонені слова, тощо. Цим займаються модератори, але ці завдання піддаються алгоритмізації, тому їх виконання можна автоматизувати.

Мета і задачі дослідження. Метою роботи є створення чат бота на платформі Node.js.

Об’єктом дослідження є чат-бот, як програмне забезпечення, що автоматизовує певні дії.

Предметом дослідження є методика створення та впровадження чат-бота для месенджера Discord.

Практичне значення одержаних результатів. Практичне значення проекту полягає в автоматизації деяких завдань модераторів на серверах з великою кількістю учасників.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ ФУНКЦІОНУВАННЯ ЧАТ БОТІВ

1.1 Поняття та сфери застосування ботів

Бот – це спеціальна програма, що автоматизовано (з певною частотою або за розкладом) виконує певні дії через ті ж самі інтерфейси, що й звичайний користувач. В більшості випадків боти призначені для виконання одноманітної та монотонної роботи, з максимально можливою швидкістю (набагато вищою за можливості людини). В зв'язку з великим поширенням ботів, на веб-сайтах створено механізм (файл robots.txt) що обмежує дії та поведінку ботів [8].

Також, ботів застосовують в умовах, коли необхідна краща реакція, ніж в людини (тобто в іграх, для інтернет-аукціонів, тощо) або, що менш звично, для імітації дій людини (тобто, боти для чатів тощо) [8].

На деяких ресурсах (новинних сайтах або форумах) ботів використовують для агрегації медіа з інших ресурсів. Такі боти спостерігають за іншими ресурсами та створюють єдину стрічку з медіа-файлів та новин [6].

Бот-модератор – це окремий аккаунт, який працює за спеціальною програмою або скриптом, призначеним для автоматичного або напівавтоматичного модерування чатів. Таких ботів зазвичай використовують для виконання так званої «брудної роботи». Найчастіше цією «брудною роботою» є слідкування за перепискою на наявність заборонених слів, відповіді на певні запитання користувачів, видалення непотрібних/старих повідомлень або реклами, тощо. Це завдання, які піддаються алгоритмізації, тобто дії для вирішення яких можна розбити в якусь послідовність. При цьому їх виконання занадто монотонне для людини.

У месенджері Discord статус «Бот» є додатковим статусом, який знаходиться в стороні від основної ієрархії користувачів.

Всі права бота спрямовані на збільшення швидкості роботи та дистанції від інших учасників (щоб не заважати користувачам).

1.2 Класифікація ботів

Існує багато різних ботів, які ми зустрічаємо в повсякденному житті. При відвідуваннях веб-сайтів можна зустріти чат-ботів, які можуть відповісти на питання про товари/послуги, або просто розібратись з навігацією сайту. Коли виникнуть якісь проблеми або складні запитання, цей бот може направити запит до справжнього співробітника сайту. Багато магазинів використовують ботів для розсилки інформації про товари, а також для можливості їх замовлення через месенджер [6].

Класифікацію ботів можна проводити за кількома параметрами. За способами отримання інформації від користувачів, ботів класифікують як:

1. Текстові – отримують інформацію від користувачів у текстовій формі, яку обробляють та виділяють з неї команди;
2. Голосові – записують людське мовлення, перетворюють його на текст, а тоді аналізують, виокремлюючи команди.

Текстові боти є простими у розробці та швидшими у роботі, за рахунок того, що затримка обробки інформації мінімальна, на відміну від голосових даних. Проте затримка обробки голосових даних зменшується з розвитком обчислювальних потужностей та систем штучного інтелекту [6].

Голосові помічники є доволі зручними у більшості випадків, оскільки вони можуть імітувати повноцінного співрозмовника. Крім того, вони можуть сприймати та розуміти людську мову. Зазвичай такі боти здатні синтезувати свої відповіді у вигляді аудіо-повідомлень. Проте на алгоритм розпізнавання мовлення потрібно витратити багато зусиль, і навіть це не гартує його коректної роботи для цілої мови, адже людям притаманні також акценти [6].

Розвинуті програми-співрозмовники називають – «віртуальний асистент». За голосом, вони можуть впізнавати співрозмовника, та враховувати це відповідаючи йому. Наприклад, звертаючись до дитини, помічник видаватиме тільки придатний для дітей контент. Так само розробляють засоби для аналізу

емоцій користувача, на основі його голосу. На основі тону та слів користувача, бот може визначати, як краще відповісти [6].

Ідеальною програмою-співрозмовником вважають ту, що може пройти тест Тюрінга. Для того щоб пройти тест, програма повинна протягом 5 хвилин вести розмову з людиною. Якщо в кінці тесту людина не зможе точно визначити з ким спілкується (з людиною чи програмою), то програма вважається такою що пройшла тест [6].

За призначенням чат-ботів (ЧБ) класифікують:

1. ЧБ для розмов на абстрактні теми;
2. ЧБ орієнтовані для розмов на певну тему.

Перший вид орієнтований на діалог із користувачем на загальні теми, не маючи нічого конкретного на меті. Другий тип, орієнтований на допомогу користувачу з повсякденними завданнями та неформальне спілкування (створення будильників та нагадувань; постійні оновлення про певну тему, наприклад погоду; керування основним пристроєм та розумним будинком, тощо)

При використанні бота також важливий інтерфейс взаємодії із користувачем. В той час як месенджер (або інша програма для спілкування) має чітко визначений інтерфейс та способи взаємодії з ним, в бота, інтерфейс також включає в себе методи взаємодії із отриманою інформацією.

Підтримка команд та можливості виокремлення команд із повідомлення користувача є ключовими методами взаємодії з чат-ботами, на даний момент.

Прості боти підтримують лише такий метод взаємодії. Вони мають невеликий набір доступних команд, які можуть опрацювати та здійснити пов’язану із цими командами дію. Інколи, для зручності, такі команди подаються у візуальному інтерфейсі месенджера. Для їх надсилання достатньо натиснути по відповідному написі/іконці на екрані. На рисунку 1.1 подано приклад таких кнопок в месенджері Telegram.

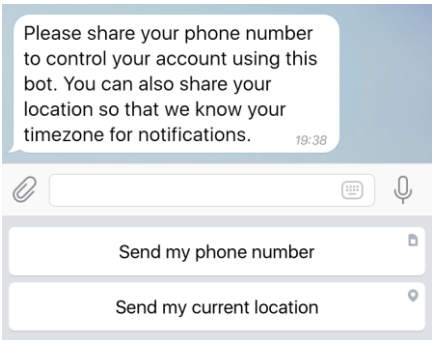

Рисунок 1.1 – Приклад кнопок- команд для бота в інтерфейсі Telegram

Загальноприйнятою концепцією в написанні фіксованих команд ботів, є використання символу префікс-символів. В Telegram таким символом є «/» (слеш). Наприклад команда «/help» реалізована для багатьох ботів, зазвичай вона надсилає у відповідь список підтримуваних команд або іншу інформацію потрібну для початку взаємодії з ботом. В командах такого типу, зазвичай, не допускається наявності пробілу в команді, навіть якщо вона складається з декількох слів. До прикладу, команда повинна виглядати так: «/showhelp».

Також в команди можна додавати певну інформацію – параметри. Наприклад, можна зробити команду: «/setalarm». В кінець команди слід дописати конкретну годину, і бот створить нагадування. Розробляти таких ботів досить просто, а щоб додати нову функцію, достатньо просто дописати нову команду у вихідний код.

1.2.1 Чат боти, які здатні виокремлювати команди

Функція виокремлення команд із звичайних повідомлень користувача є доволі популярною. З підтримкою такого функціоналу, бот зможе розуміти різні варіації команд, наприклад: «Поставиш будильник на 12:30 23-го червня?» чи «Встанови будильник на завтра о 12:30». Для реалізації такого функціоналу необхідне використання NLP (засобів обробки природної мови). Наше повсякденне спілкування часто є неточним, та в залежності від теми спілкування, одне й те ж слово може мати різні значення. Наприклад, слово «set» в англійській мові має 430 значень.

При використанні засобів обробки природної мови, з тексту виокремлюються корисні фрагменти інформації, а на їх основі формується команда, яку бот повинен виконати. Також для цього використовують технологію розпізнавання названого об’єкта (NER), щоб точно визначати об’єкт, що має на увазі користувач.

Це дозволить боту розуміти, що Львів – це місто, 2022 рік – це 2022-й рік за григоріанським календарем, а Богдан – це ім’я.

Також існують боти, які виокремлюють ключові слова з речень користувача. У таких ботів просто є готові фрази для відповідей на певну кількість питань із набором ключових слів. Наприклад, слово «тариф» надіслане боту на сайті оператора зв’язку, змісить його надіслати попередньо заготовану для цього слова відповідь з інформацією про доступні тарифи. Якщо такому боту поставити складне запитання, яке не відноситься до його сфери, – він або поставить уточнююче запитання, або запропонує з’єднати з оператором (людиною).

1.2.2 Чат боти, які здатні підтримувати діалог

Ще одним видом чат-ботів є боти з можливістю розуміти контекст діалогу. Наприклад, ось один із можливих діалогів (Л – людина, Б – бот):

Л – Слід додати яйця у список покупок.

Б – Додав.

Розмова на іншу тему або перерва в діалозі.

Л – Також слід придбати пляшку миючого засобу.

Б – Миючий засіб доданий в список покупок.

В результаті, користувач матиме список покупок з двома згаданими предметами. Також варто звернути увагу на те, що чим більше різноманітних фраз/відповідей бот має у своєму «арсеналі», тим ближче розмова нагадуватиме людську. В таких випадках, щоб забезпечити розуміння ботом діалогу а також

гарантувати доречність відповідей, використовують нейронні мережі та машинне навчання.

Також нейроні мережі використовують для персоналізації контенту або реклами. Наприклад, бот аналізує нещодавні речення або пошукові запити користувача на якусь тему, і за потреби, запропонує доречніший продукт або послугу.

Крім того, боти можуть мати можливість інтеграції зі сторонніми сервісами. Наприклад, бот Google Assistant, від компанії Google, може відкривати програми сторонніх розробників, інколи навіть виконувати певні дії в них. Наприклад, на рисунку 1.2, наведено взаємодію бота зі сторонньою програмою для програвання музики – Spotify.

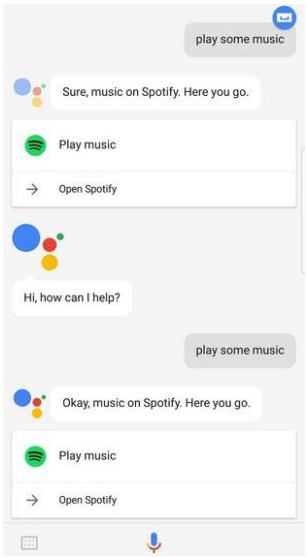

Рисунок 1.2 – Використання Google Assistant для запуску програми Spotify

Для реалізації такої взаємодії, розробники обох додатків повинні створити спеціальний API у своєму продукті. Взаємодію бота практично неможливо реалізувати, якщо в цільовому додатку відсутні певні інтерфейси взаємодії.

1.3 Приклади застосування ботів

Найчастіше чат ботів використовують у службах підтримки користувачів різних компаній. Вони можуть відповідати на більшість питань клієнтів, без необхідності втручання людини. Крім цього існує багато інших застосувань для ботів. Існують боти, що дозволяють бронювати столики в ресторані, керувати банківськими рахунками, вести анонімку кореспонденцію, тощо.

Найновіший ЧБ, який став дуже популярним – це безкоштовний бот-юрист. Він ще не скоро зможе замінити справжнього юриста, але вже зараз може допомогти користувачам створювати справи для оскарження штрафів за неправильне паркування, отримувати компенсації за приховані витрати при покупках/подорожах, або надавати відповіді на різні юридичні запитання.

Прикладом можна вважати факт коли бот, створений компанією Microsoft, Xiaoice отримав фото співрозмовника з підвернутою ногою, він спитав наскільки було боляче було людині при отриманні травми.

Одним з найвідоміших сервісів, який широко застосовує ботів, є – Reddit. Це великий форум з тисячами публікацій від користувачів. Для дуже великим публікацій, Reddit створили бота AutoTLDR (Too Long, Do not Read). За декілька хвилин бот створює коротку анотацію для довгої статті, розміром 450-700 символів, яку розміщує на форумі або в коментарях до статті.

Одним з найпопулярніших ботів для туристів є Mezi. Достатньо сказати боту коли й куди ви хочете поїхати, і він допоможе вам купити квиток, забронювати готель та навіть столик у ресторані. Цей бот агрегує сотні сайтів бронювання і надає зручний інтерфейс для взаємодії з усіма ними. Інтерфейс спілкування бота Mezi зображено на рисунку 1.3.

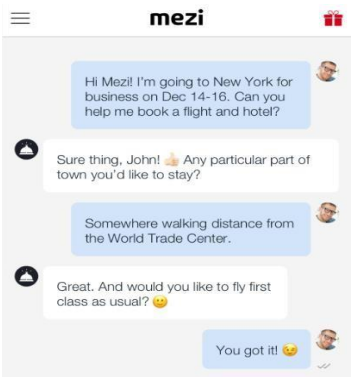

Рисунок 1.3 – Інтерфейс спілкування з ботом Mezi

Також популярності набувають ЧБ для замовлення їжі. З їх допомогою можна замовити собі їжу без потреби в телефонних дзвінках. Через зручний інтерфейс бот дозволяє обрати собі їжу, вказати адресу доставки та здійснити оплату. Прикладом може слугувати бот Foodie від компанії Facebook (Рисунок 1.4).

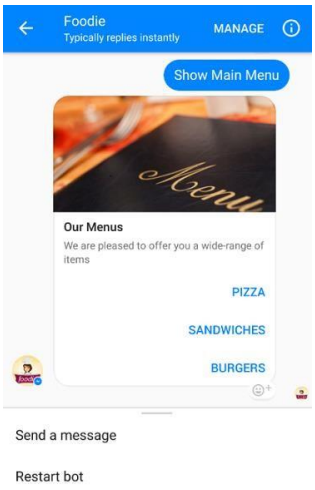

Рисунок 1.4 – Інтерфейс бота для доставки їжі Foodie у Facebook Messenger

Іншим популярним видом ботів є боти-агрегатори. На основі RSS стрічок такі боти можуть отримувати інформацію від різних інформаційних ресурсів та публікувати її в певному каналі. Як приклад можна розглянути бота MonitoRSS в месенджері Discord. Через спеціальне меню на сайті, бот дозволяє додати декілька RSS стрічок для моніторингу, та публікуватиме оновлення до зазначеного текстового каналу на сервері, де кожен користувач зможе побачити оновлення.

1.4 Приклад бота в месенджері Discord – MEE6

Одним з прикладів ЧБ для месенджера Discord є бот MEE6. Це один з найпопулярніших ботів, що має досить велику кількість функцій і присутній на багатьох популярних серверах.

Логотип цього бота виглядає як усміхнене блакитне коло з руками і ногами. Бот також має додаток у вигляді преміум, який коштує $11.95 на місяць, $49.99 на рік і $89.90 назавжди.

За допомогою власних команд, які можна задати власноруч, MEE6 може відповідати, писати в особисті повідомлення, забирати / видавати ролі. Також можна включити функцію «random» – тоді він буде відповідати різними відповідями з різним шансом.

Бот підтримує рівневу систему, основні команди якої подано в таблиці 1.1. Завдяки їй можна налаштувати привітання для учасника, якщо той набрав потрібний рівень. Привітання може приходити в особисті повідомлення або прямо на сервер. Цю функцію можна відключити. Також можна зробити так, щоб бот видавав роль за певний рівень, але на серверах це працює тільки якщо придбаний преміум план.

Таблиця 1.1 – Команди рівневої системи.

Команда	Дія
!rank	Показує рівень учасника
!levels	Дає посилання на сторінку з рівнями учасників сервера.

Завдяки модулю привітань, бот може надсилати повідомлення з привітанням новому учаснику. Так само, як і в випадку з рівнями, він може відправити привітання в особисті повідомлення або на сервер. Це привітання можна відредагувати в будь–який момент, що робить функцію ще зручнішою. Бот також може видавати ролі новачкам.

Завдяки модулю «Search Anything» бот може знайти що–небудь в інтернеті. Цю функцію не можна редагувати, але можна відключити команди для пошуку. Самі команди подано в таблиці 1.2.

Таблиця 1.2 – Основні команди пошуку

Команда	Дія
!imgur	Шукає зображення на imgur.com
!twitch	Шукає трансляції на Twitch.tv
!urban	Шукає сленгові слова в Urban Dictionnary
!youtube	Шукає відео на YouTube

Завдяки модулю «Moderator» ботові можна надати повноваження модератора і автоматизувати частину модераторських обов'язків. Існує також опція «auto–moderator»: після кожного порушення MEE6 буде видаляти повідомлення і/або надсилати попередження автору повідомлення. В цьому модулі також можна прибрати деякі команди як і в «Search Anything» (Додаток А).

1.5 Висновки до першого розділу

В даному розділі детально розглянуто ботів – програми які дозволяють автоматично спілкуватися з користувачами. Проаналізовано вже існуючих ботів та наведено приклади їх використання. Дані програми мають широкий спектр застосування, в тому числі їх використовують для потреб бізнесу. Інформаційних та розважальних ботів можна монетизовувати різними способами, в залежності від платформи спілкування та цільової аудиторії.

РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЕКТУ

2.1 Початкові налаштування

Для початку роботи необхідно створити обліковий запис в системі Discord. Для цього необхідно перейти на офіційний сайт додатку. На рисунках 2.1 та 2.2 показано адресу сайту та його вигляд.

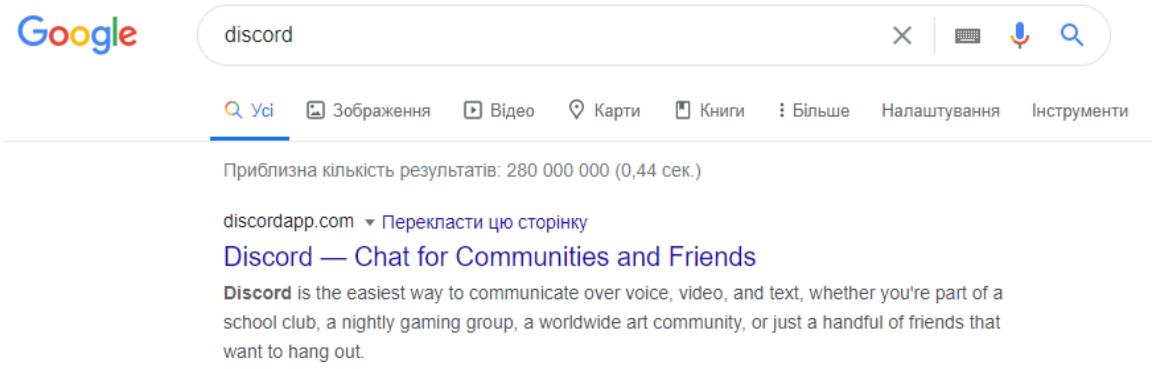

Рисунок 2.1 – Адреса та назва сайту у пошуковій видачі

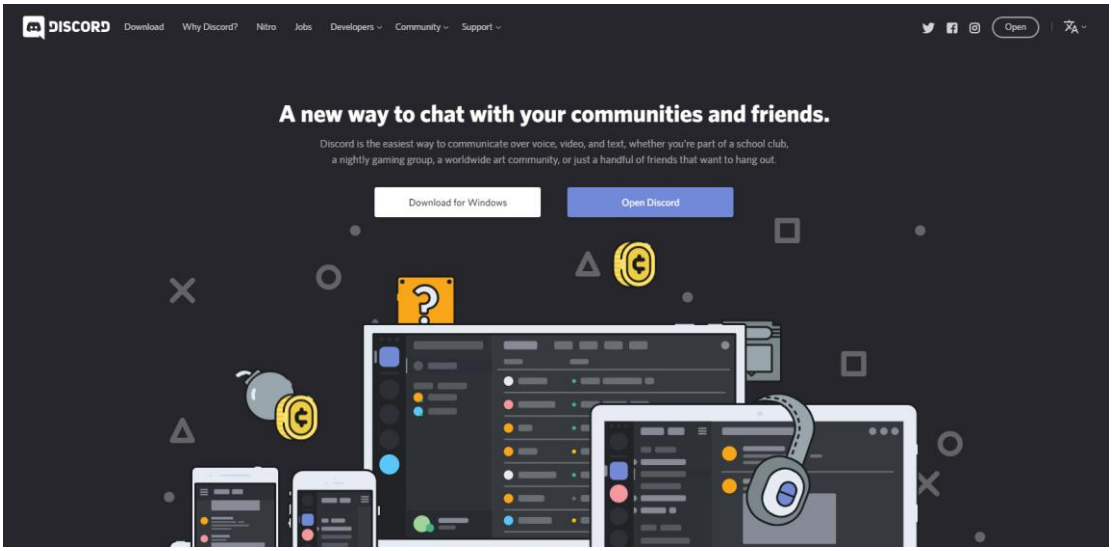

Рисунок 2.2 – Вигляд головної сторінки сайту

Після натискання кнопки «Login» у правому верхньому кутку, потрібно вибрати реєстрацію та ввести свої дані як показано на рисунку 2.3.

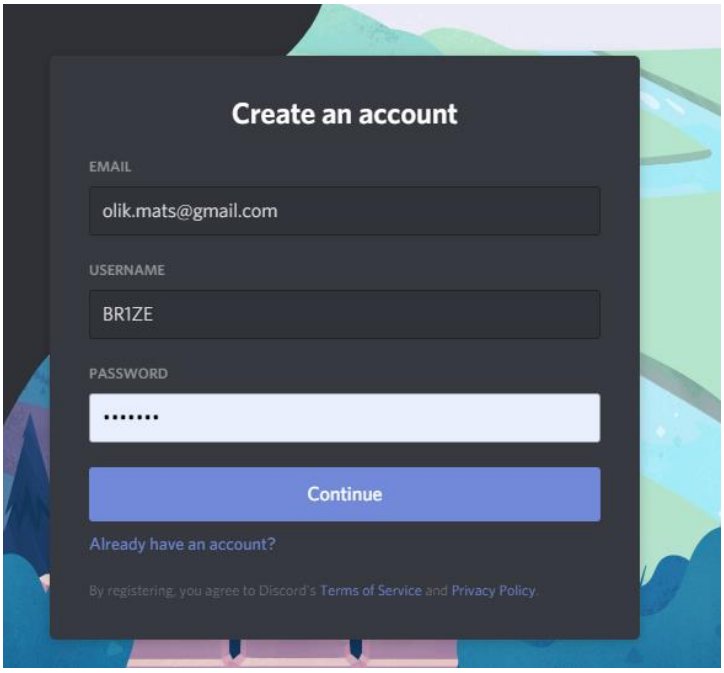

Рисунок 2.3 – Форма реєстрації.

Після реєстрації відкривається сторінка веб-клієнта додатку. На рисунку 2.4 показано його інтерфейс.

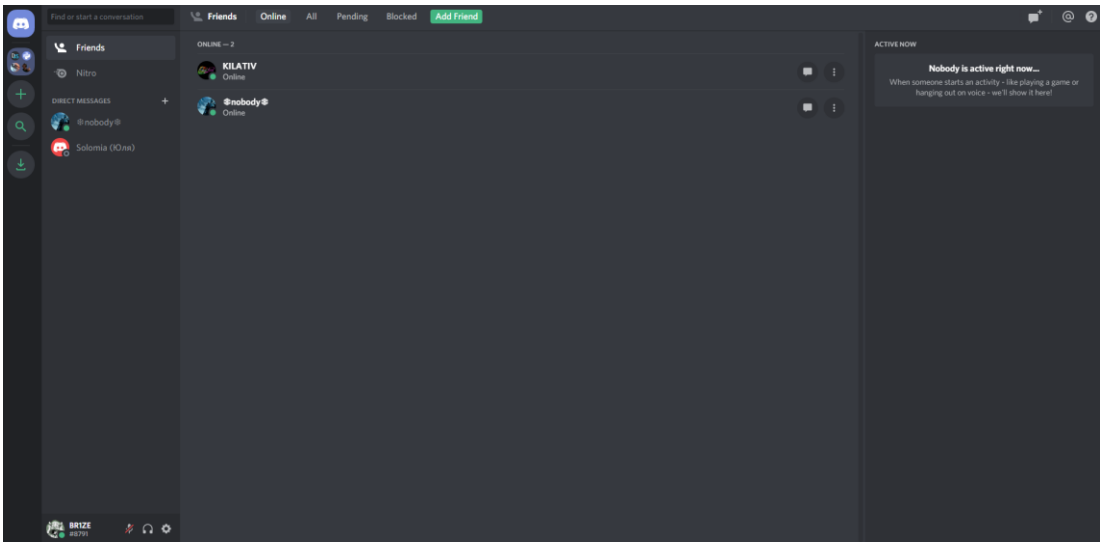

Рисунок 2.4 – Інтерфейс веб–клієнта Discord

Для зручності можна завантажити та встановити додаток на комп’ютер. Після відкриття виконуваного файлу, додаток швидко завантажить все необхідне та одразу ж відкриється. Як видно на рисунку 2.5, інтерфейс додатку не відрізняється від веб–клієнта.

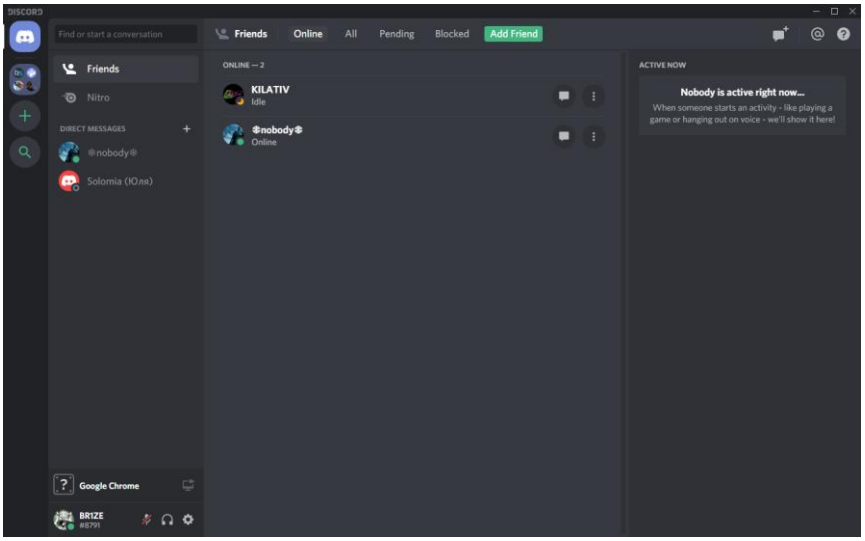

Рисунок 2.5 – Інтерфейс додатку встановленого на ПК

Тепер необхідно створити сервер, на якому буде тестуватись майбутній бот, для цього тиснемо на кнопку, зображену на рисунку 2.6, обираємо пункт «Створити», як на рисунку 2.7, вводимо назву та створюємо сервер (рисунок 2.8).

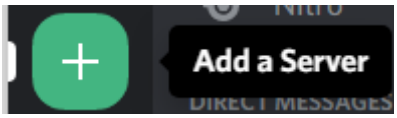

Рисунок 2.6 – Кнопка додавання нового сервера

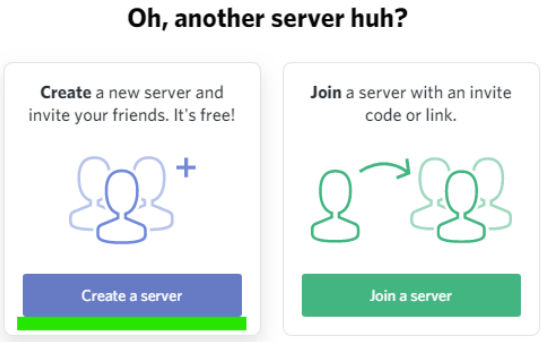

Рисунок 2.7 – Вибір пункту створення сервера(підкреслено зеленим)

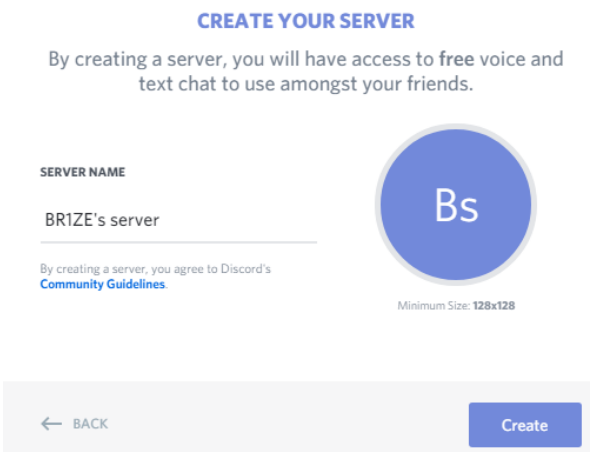

Рисунок 2.8 – Вікно вибору назви сервера та його логотипу

Вигляд створеного сервера показано на рисунку 2.9.

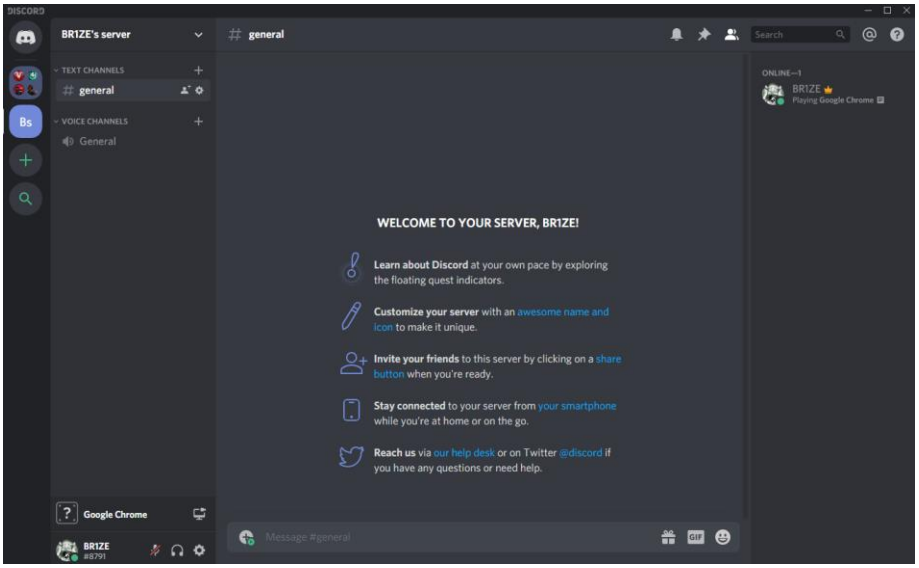

Рисунок 2.9 – Вигляд новоствореного сервера.

Інтерфейс додатку можна умовно поділити на 4 частини: панель навігації між серверами (рисунок 2.10), панель навігації між каналами (рисунок 2.11), вікно обраного чату (рисунок 2.12) та панель зі списком учасників (рисунок 2.13).

Рисунок 2.10 – Панель навігації між серверами

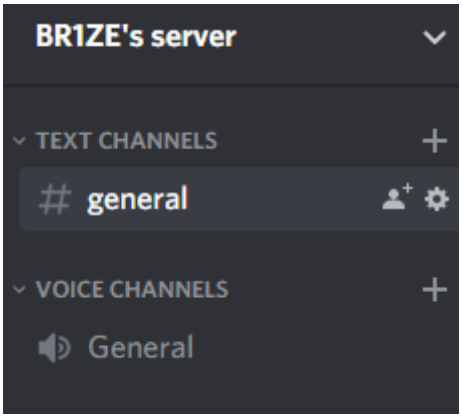

Рисунок 2.11 – Список каналів сервера

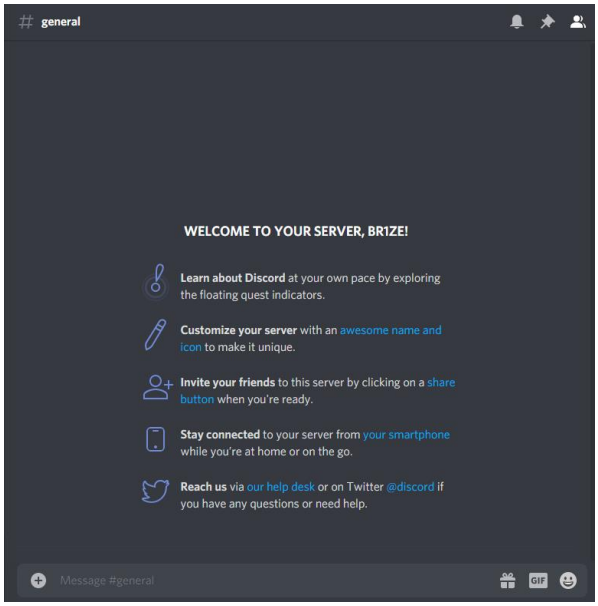

Рисунок 2.12 – Вікно обраного текстового каналу

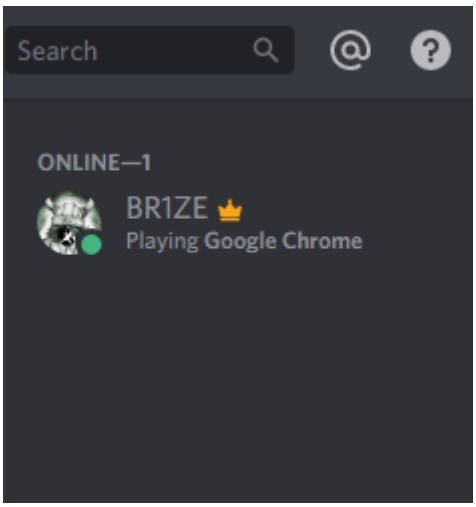

Рисунок 2.13 – Панель зі списком учасників

Інтерфейс програми, як і веб-додатку, є зручним та простим в освоєнні. Також варто зауважити що всі елементи взаємодії структуровані, а документація для розробників містить відповіді на будь-які запитання.

2.2 Початок роботи з порталом розробника Discord

Для початку роботи на порталі розробника, необхідно ввійти туди під своїм обліковим записом Discord. На рисунках 2.14 та 2.15 показано адресу сайту та його вигляд.

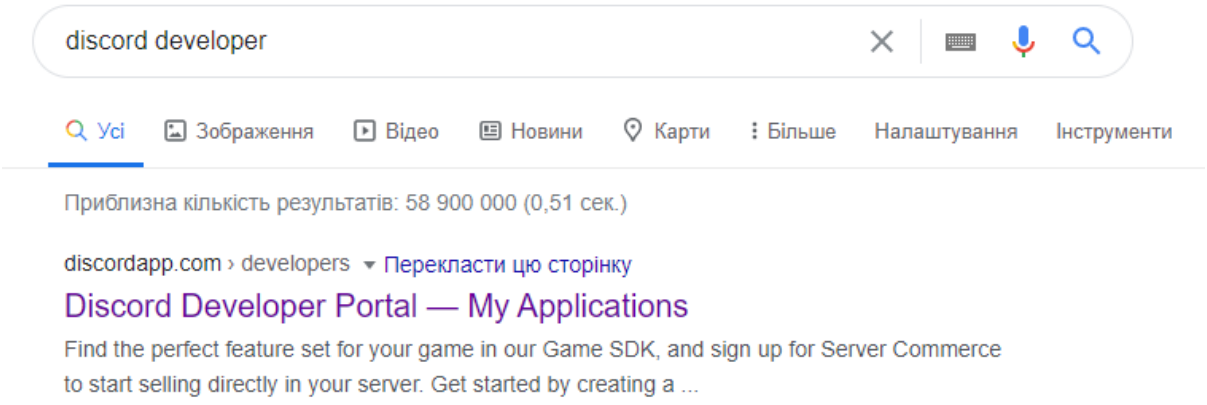

Рисунок 2.14 – Результат пошукової видачі та адреса порталу

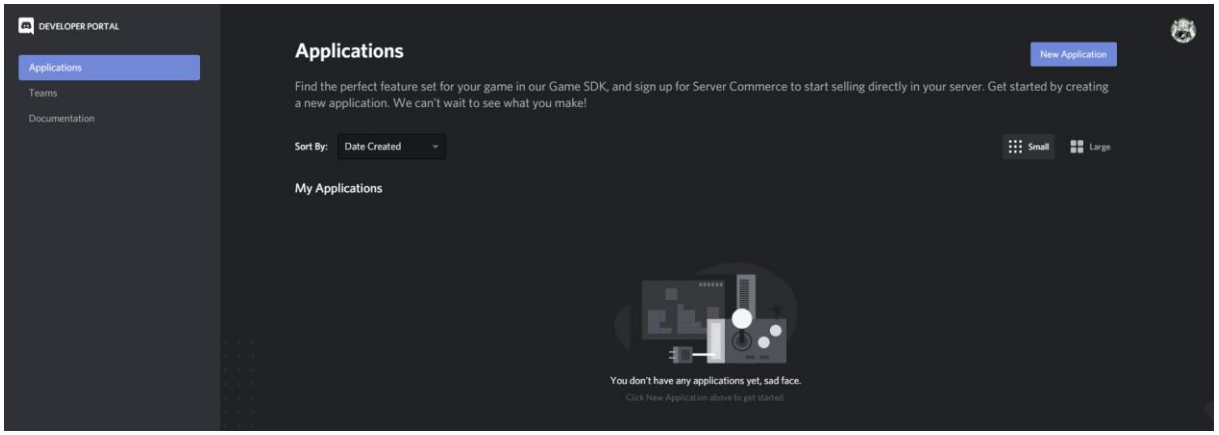

Рисунок 2.15 – Початкова сторінка порталу

Тепер необхідно створити новий додаток, це можна зробити, натиснувши на кнопку «New Application». Після натискання слід ввести назву додатку та створити його, у віконечку, що зображено на рисунку 2.16

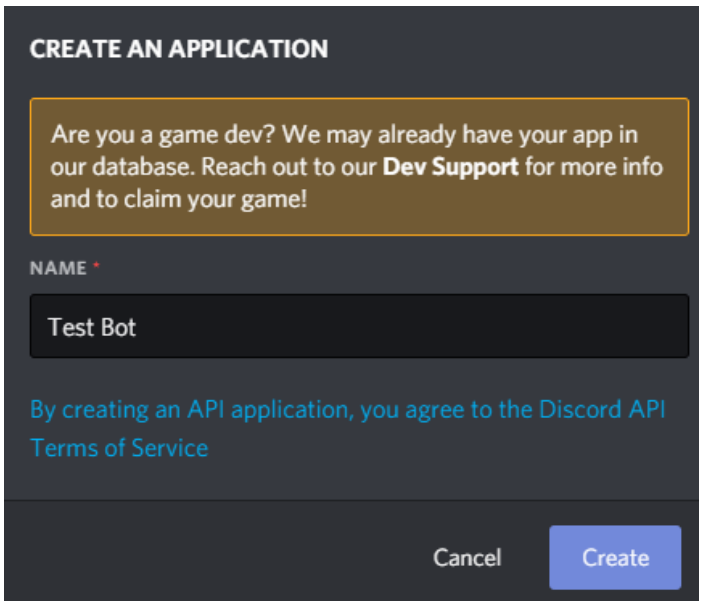

Рисунок 2.16 – Вікно створення нового додатку

Після вдалого створення відбудеться перехід на сторінку додатку (рисунок 2.17).

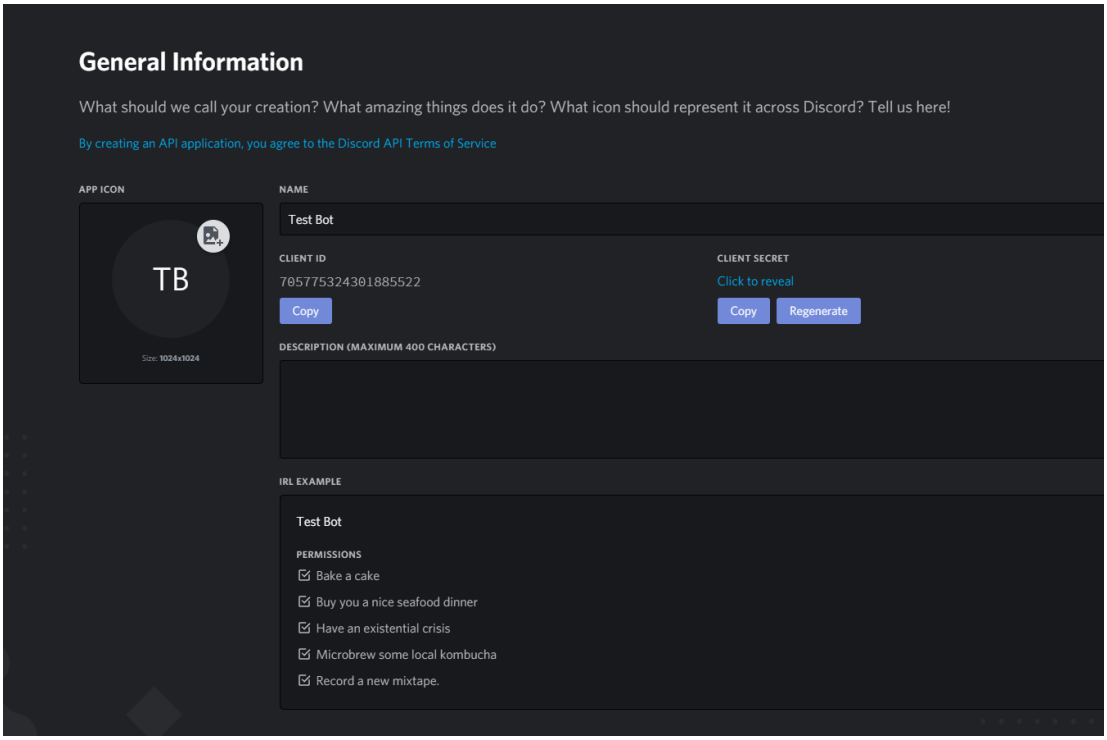

Рисунок 2.17 – Сторінка створеного додатку

Тепер необхідно створити обліковий запис бота, для цього необхідно перейти в розділ «Bot» (рисунок 2.18) додатку та відповідною кнопкою додати бота (кнопку зображено на рисунку 2.19), заповнивши його ім’я та завантаживши фотографію.

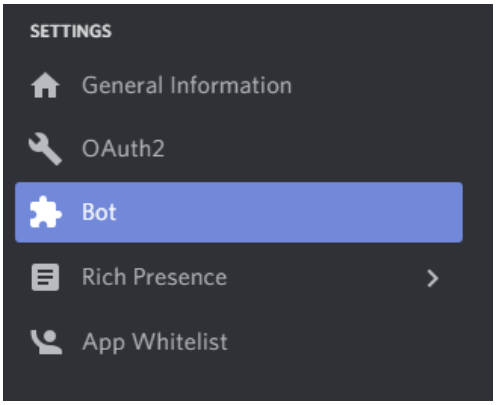

Рисунок 2.18 – Розділ «Bot» створеного додатку

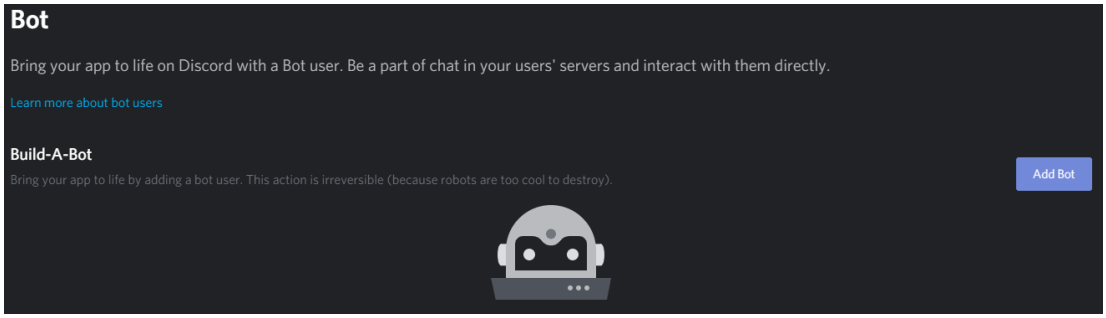

Рисунок 2.19 – Кнопка додавання нового бота

Профіль створеного бота зображено на рисунку 2.20. З цієї сторінки можна змінити ім’я та зображення бота, скопіювати або створити новий автентифікаційний токен.

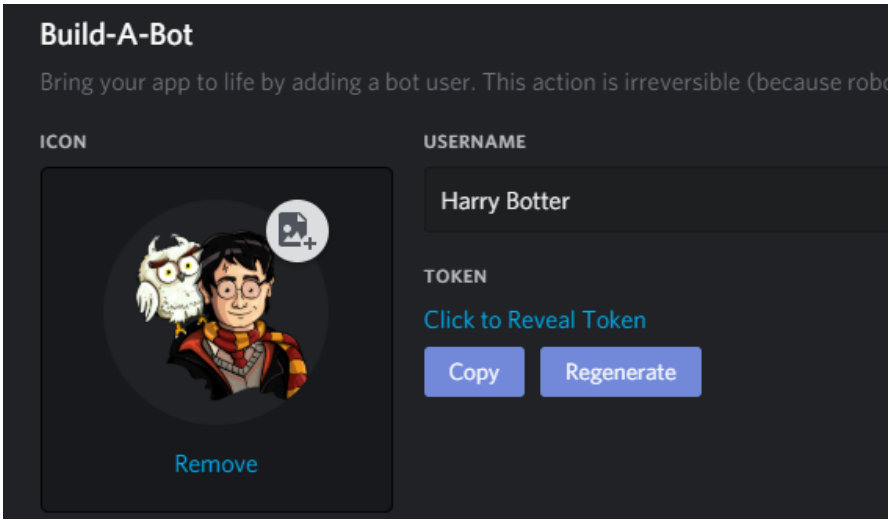

Рисунок 2.20 – Вигляд створеного бота на сайті

Тепер необхідно запросити бота на сервер, для цього існує спеціальна форма запрошення у вигляді посилання.

Саме посилання виглядає наступним чином:

discord.com/oauth2/authorize?client_id=BOT_ID&scope=bot&permissions =NUMBER,

де BOT_ID – це ідентифікаційний номер облікового запису бота, а NUMBER – параметр, що відповідає за обсяг повноважень бота на сервері, куди його запрошено, який задається числом.

ID бота можна скопіювати у вкладці «General Information» на сторінці нашого додатку (рисунок 2.21).

Ідентифікатор для визначення необхідних прав можна знайти у вкладці «Bot» (рисунок 2.22).

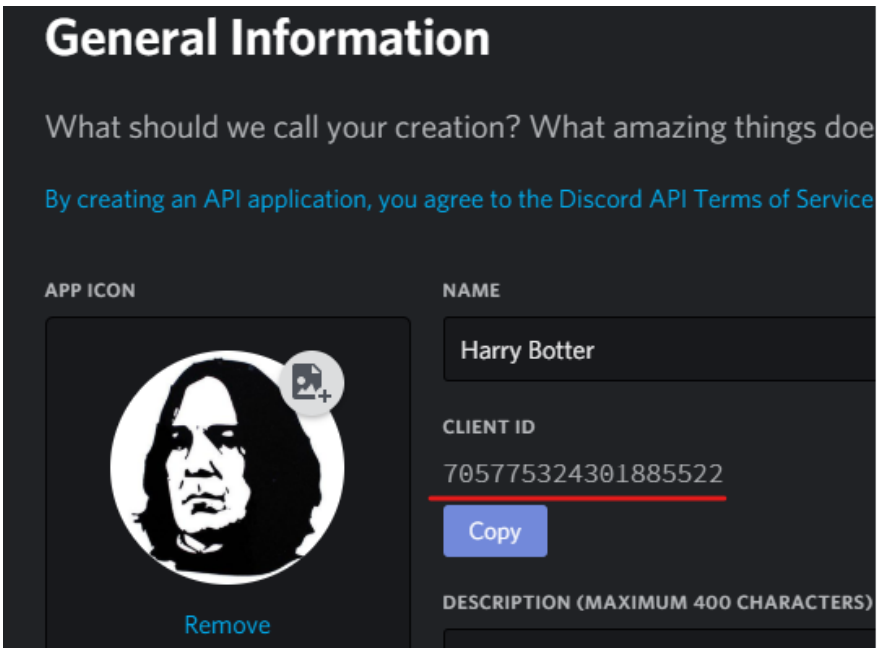

Рисунок 2.21 – Розташування ідентифікатора облікового запису

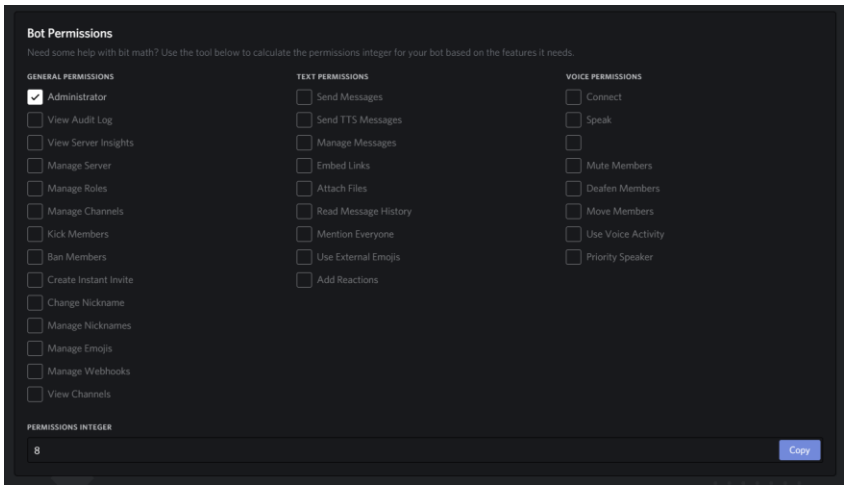

Рисунок 2.22 – Панель визначення ідентифікатора необхідних прав

Після переходу за створеним посиланням необхідно обрати сервер, на який буде запрошено бота, перевірити його повноваження та підтвердити дію, як зображено на рисунку 2.23. На рисунку 2.24 зображено долучення бота на сервер.

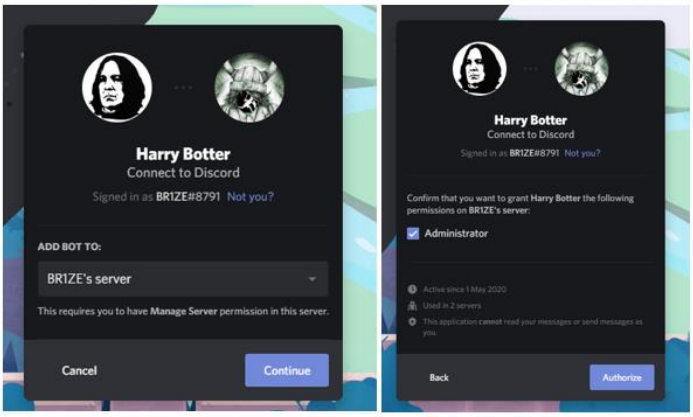

Рисунок 2.23 – Сторінка вибору сервера та підтвердження запрошення

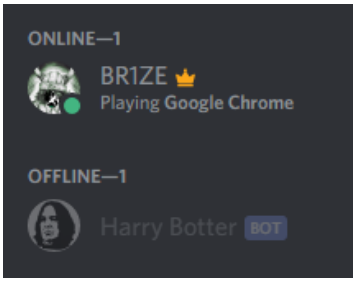

Рисунок 2.24 – Бот знаходиться на сервері

Таким чином, акаунт для бота створено й додано на тестовий сервер. Жодних інших дій на порталі розробника здійснювати не треба.

2.3 Встановлення Node.js

Для встановлення Node.js варто завантажити інсталяційний файл з офіційного сайту – nodejs.org. Сайт зображено на рисунку 2.25. Завантажувати рекомендовано версію з довгою підтримкою (LTS), оскільки, практично, всі модулі створюються саме для неї, тож це забезпечить максимально стабільну роботу.

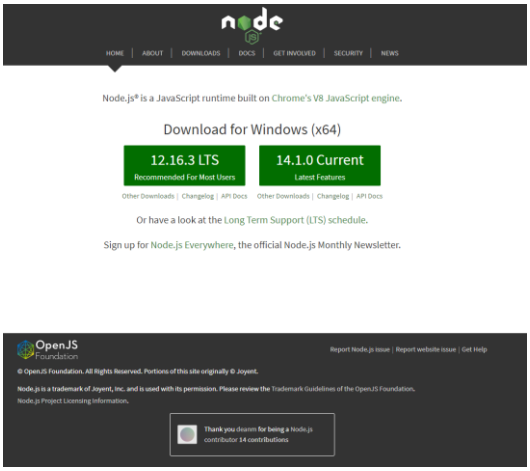

Рисунок 2.25 – Офіційний сайт Node.js

Після відкриття завантаженого файлу запуститься звичайний інсталятор, в якому потрібно обрати місце для встановлення та необхідні компоненти. Інсталятор зображений на рисунках 2.26 та 2.27.

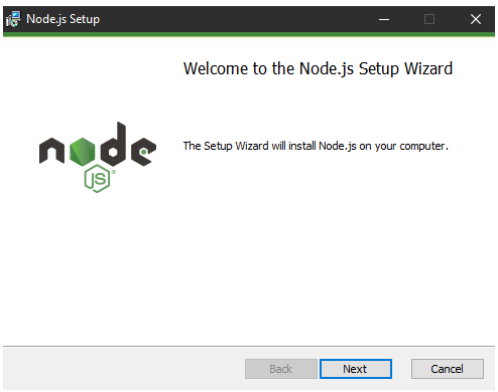

Рисунок 2.26 – Початкова сторінка інсталятора

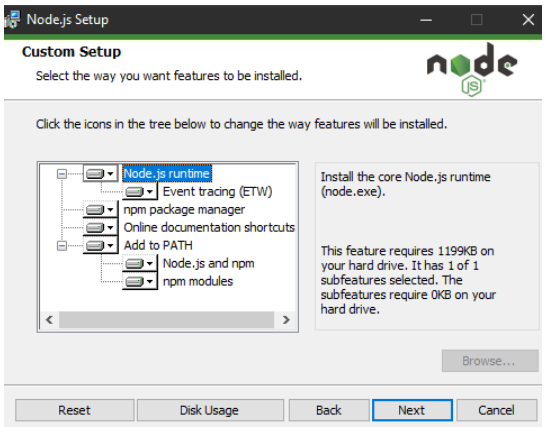

Рисунок 2.27 – Сторінка вибору компонентів

Після завершення установки, Node буде розміщений у вказаній раніше папці та готовий до роботи.

2.4 Встановлення модулів Discord.js

Список модулів які необхідні для роботи створюваного бота:

– Discord.js;
– Discord.js–music–v11;
– Ytdl–core;
– Canvas;
– YT.js.

Всі ці модулі розповсюджуються за допомогою технології npm (Node Package Manager), тому, якщо на комп'ютері встановлено Node, всі функції npm повинні працювати [7]. Необхідні модулі можна знайти на сайті www.npmjs.com (головна сторінка сайту показана на рисунку 2.28). На сторінці кожного окремого модуля є унікальна команда для його встановлення, його опис, властивості та приклад використання з базовим зразком коду (рисунок 2.29 та додаток Б)

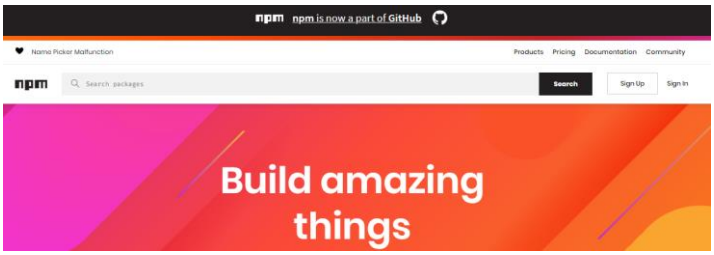

Рисунок 2.28 – Головна сторінка сайту npm з рядком пошуку

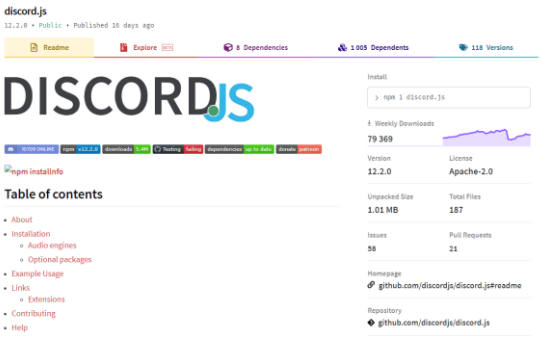

Рисунок 2.29 – Сторінка модуля Discord.js

Для встановлення модулів необхідно відкрити командний рядок (бажано в режимі адміністратора), відкрити директорію, в яку необхідно встановити

модулі, та ввести команду встановлення відповідного модуля. Для модуля Discord.js команда зображена на рисунку 2.30. Результат виконання команди та вигляд директорії представлено на рисунках 2.31-2.33.

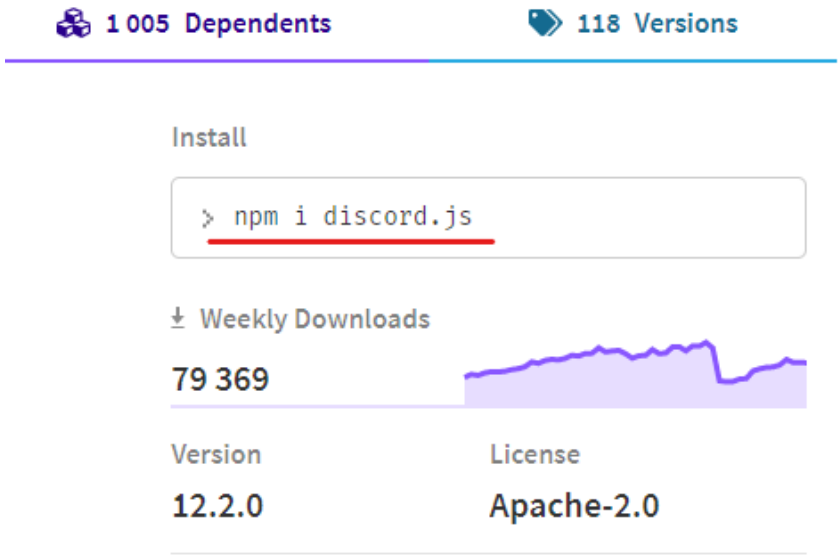

Рисунок 2.30 – Команда встановлення модуля Discord.js

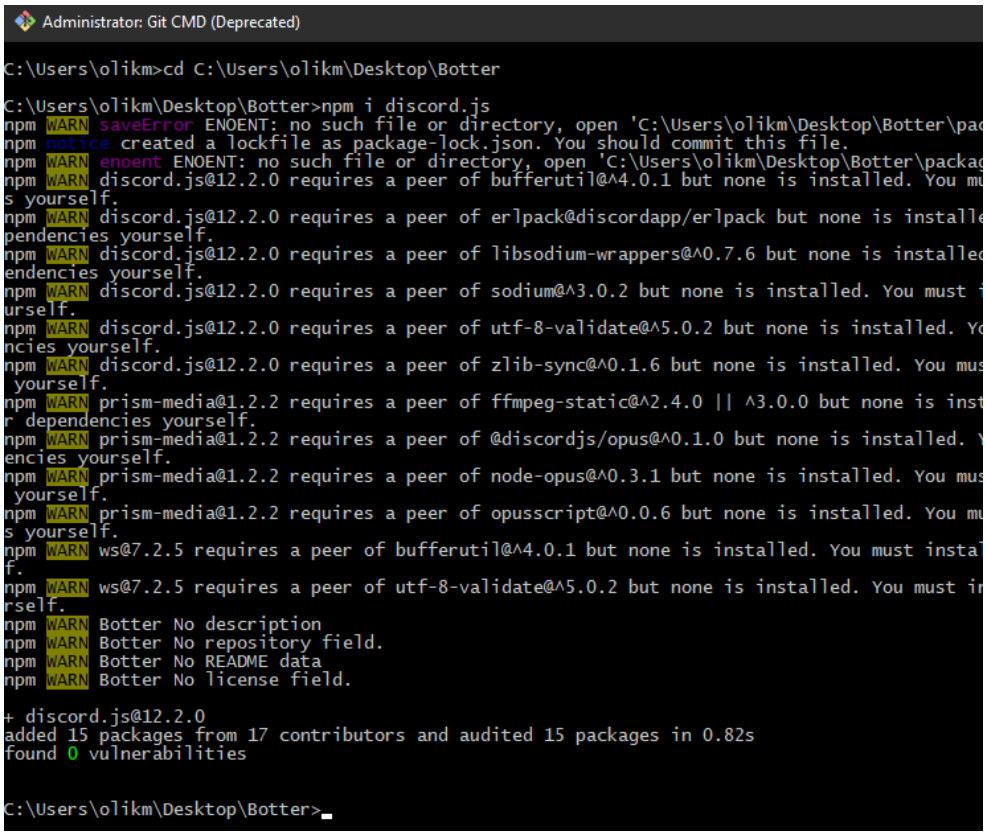

Рисунок 2.31 – Результат виконання команди «npm i discord.js»

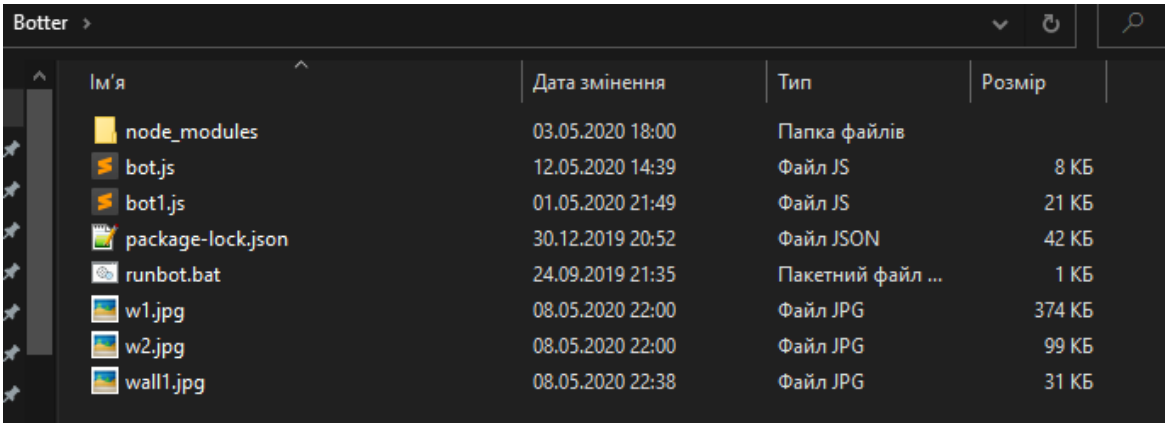

Рисунок 2.32 – Вигляд директорії після виконання команди

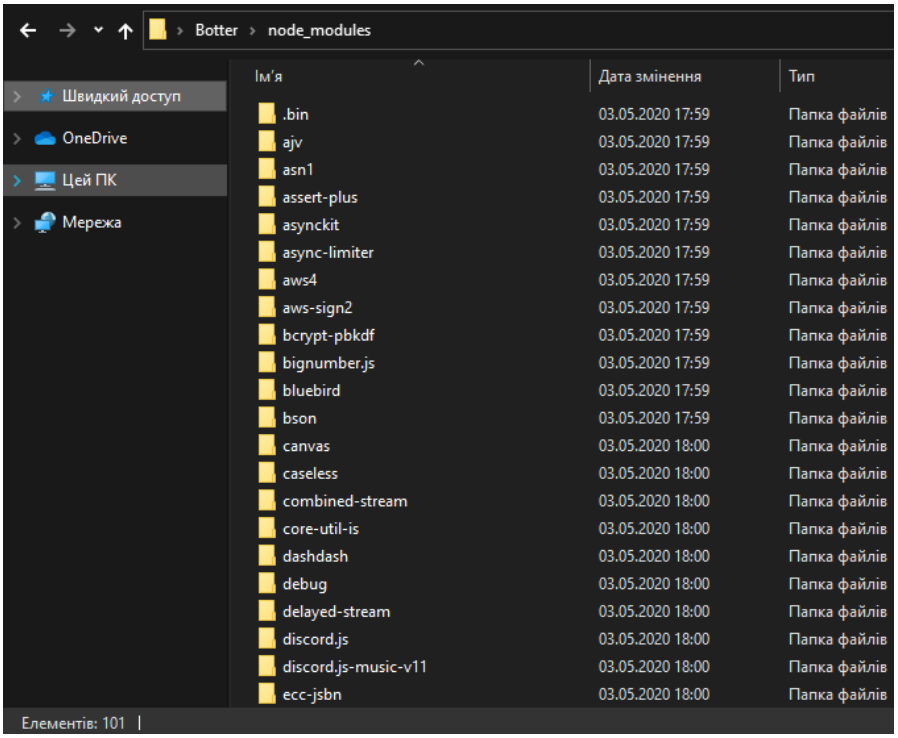

Рисунок 2.33 –Автоматично створена директорія node_modules)

Інші модулі встановлюються у такий самий спосіб.

Варто зауважити, що деякі модулі використовують спільні бібліотеки, тому слід звертати увагу на версії всіх бібліотек, адже репозиторій одного модуля, може містити старішу версію певної бібліотеки, з якою інші модулі можуть не функціонувати як слід [11].

2.5 Встановлення FFMPEG

FFmpeg – це комплекс вільних комп'ютерних програм та програмних бібліотек для маніпуляцій з цифровими відео- та аудіо-матеріалами – запис, конвертація та пакування у різні формати контейнерів.

Проект славиться наявністю різних аудіо та відео кодеків.

Інтерфейс командного рядка має інтуїтивний вигляд.

FFmpeg було розроблено під Linux, але він успішно працює й у Apple Mac OS X та Microsoft Windows.

Для встановлення комплексу програм FFMPEG також необхідно перейти на офіційний сайт проекту (www.ffmpeg.org/download.html). На сторінці необхідно обрати пакет для необхідної платформи (рисунок 2.34).

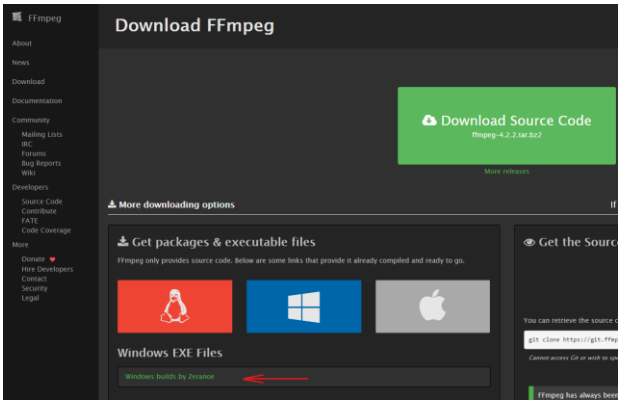

Рисунок 2.34 – Сторінка вибору платформи на офіційному сайті

При виборі платформи Windows відбувається переадресація на піддомен автора версії FFMPEG, який зображено на рисунку 2.35. Після вибору необхідних параметрів почнеться завантаження архіву з необхідними даними та декількома виконуваними файлами [10].

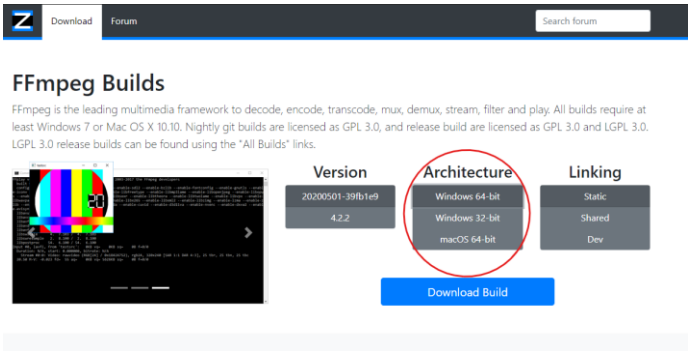

Рисунок 2.35 – Піддомен автора версії FFMPEG

Для коректної роботи в майбутньому, весь вміст завантаженої папки варто помістити в корінь диску C та додати шлях до папки як змінну оточення системи [10].

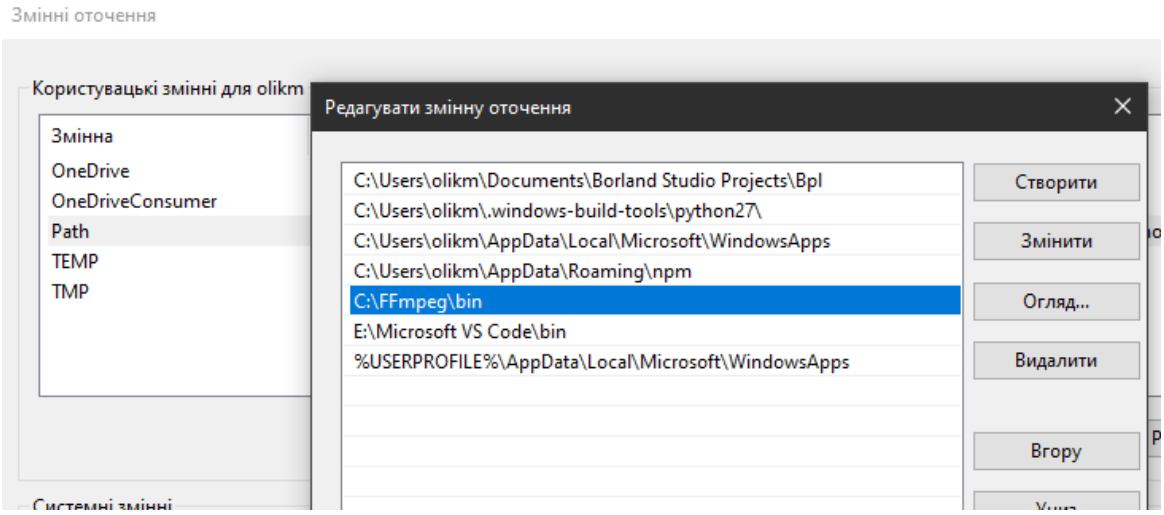

Рисунок 2.36 – Вигляд шляху для змінної оточення

Щоб додати змінну оточення у Windows, потрібно перейти у «Властивості системи» >> «Додаткові настройки системи» >> «Додатково» >> «Змінні оточення» [10].

2.6. Створення бази даних та налаштування сервера

MongoDB – це система керування базами даних (СКБД) з відкритим вихідним кодом, якій не потрібен опис схеми таблиць. MongoDB є документо-орієнтованою, тобто в основі її роботи лежать деревоподібні документні сховища [12].

MongoDB дозволяє зберігати документи в JSON-подібному форматі. Мова для формування запитів є досить гнучкою, можна індексувати збережені атрибути. Також база даних забезпечує ефективне зберігання бінарних об'єктів з великим об’ємом[12].

Для початку роботи потрібно завантажити та встановити публічну версію сервера MongoDB з офіційного сайту. Сайт зображено на рисунку 2.37.

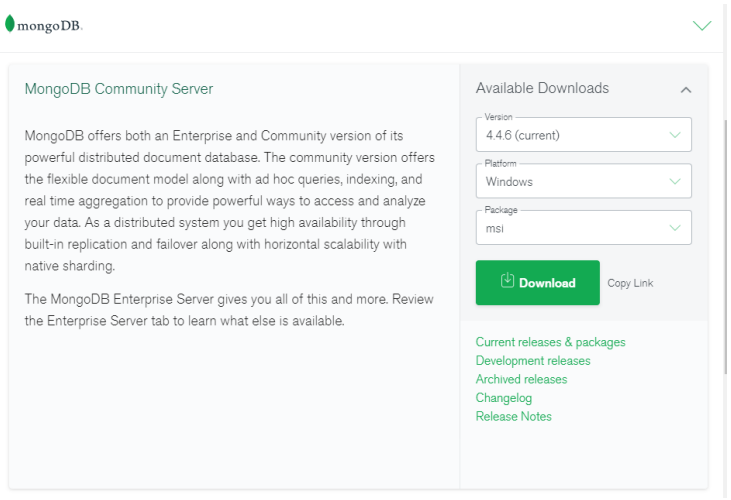

Рисунок 2.37 – Офіційний сайт MongoDB

Після здійснення установки, для запуску сервера необхідно відкрити командний рядок та перейти в директорію програми.

Для запуску сервера потрібно ввести команду mongod.exe (рисунок 2.38).

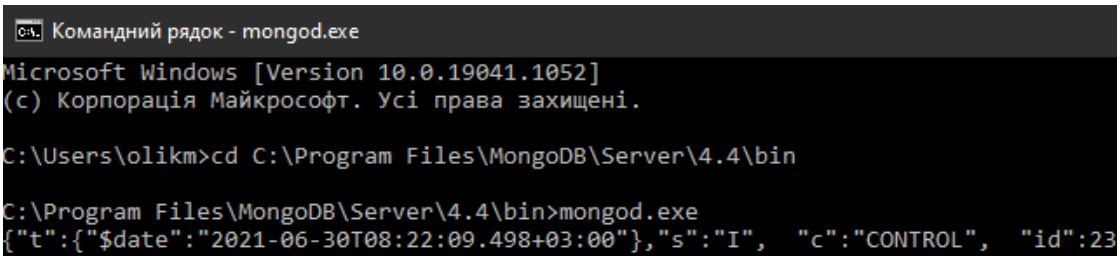

Рисунок 2.38 – Командний рядок з запуском сервера

Після запуску сервера, з декількох останніх рядків, можна виокремити необхідну для підключення інформацію, а саме адресу та порт підключення. Дані зображено на рисунку 2.39.

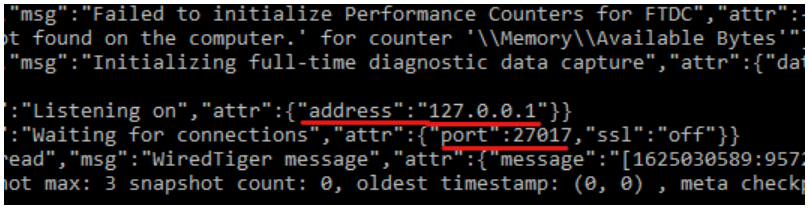

Рисунок 2.39 – Дані необхідні для підключення

Також з’явиться програма MongoDB Compass, яка дозволяє керувати базою даних за допомогою графічного інтерфейсу. Вигляд запису показано на рисунку 2.40.

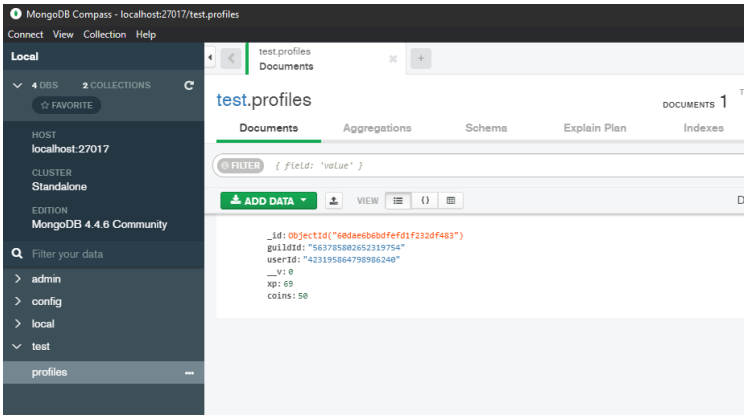

Рисунок 2.40 – Вигляд програми MongoDB Compass

Для підключення нашого сервера до скрипта, необхідно в файлі конфігурації прописати шлях до сервера. Вигляд файлу кофігурації показано на рисунку 2.41 [5].

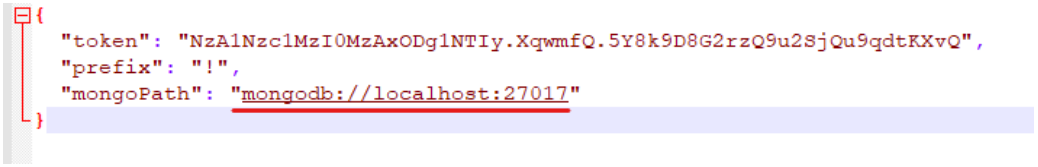

Рисунок 2.41 – Шлях до сервера у файлі конфігурації

Після цього, жодних подальших дій з базою даних виконувати не потрібно, оскільки її вміст буде редагуватись безпосередньо ботом. Сервер просто повинен працювати (на локальному ПК або сервері) та бути доступним у мережі Інтернет.

2.7 Написання сценарію поведінки для бота

2.7.1 Написання базового коду та його налагодження

Для того, щоб почати писати сценарій, необхідно створити файл, в якому він розміщуватиметься. Для цього в основній директорії проекту слід створити порожній файл з розширенням .js і дати йому назву, для прикладу буде використано файл bot.js [2].

Вигляд місця розташування файлу та його вміст показано на рисунку 2.42.

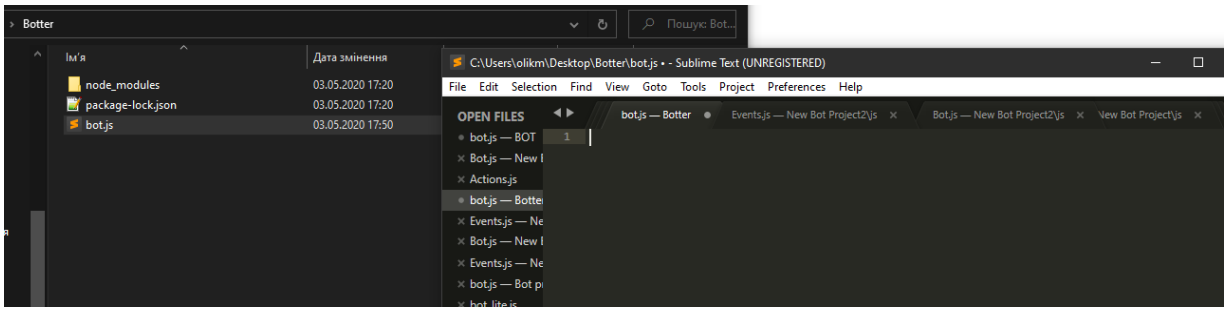

Рисунок 2.42 – Директорія файлу bot.js та його вміст

Базовий набір коду для початку роботи подано на сторінці модуля discord.js (додаток Б)

Але поки що сценарій не запуститься. Для його запуску необхідно вписати в файл секретний авторизаційний токен, за допомогою якого бот ввійде в свій, раніше створений, обліковий запис. Токен необхідно вставити в спеціальний рядок замість слова «token», як зображено на рисунку 2.43.

```
client.login('token');
```

Рисунок 2.43 – Рядок коду де потрібно вставити токен

Сам токен необхідно скопіювати з порталу розробника, в розділі «Бот» створюваного додатку. Кнопка копіювання показана на рисунку 2.44, а вигляд готового коду в додатку В.

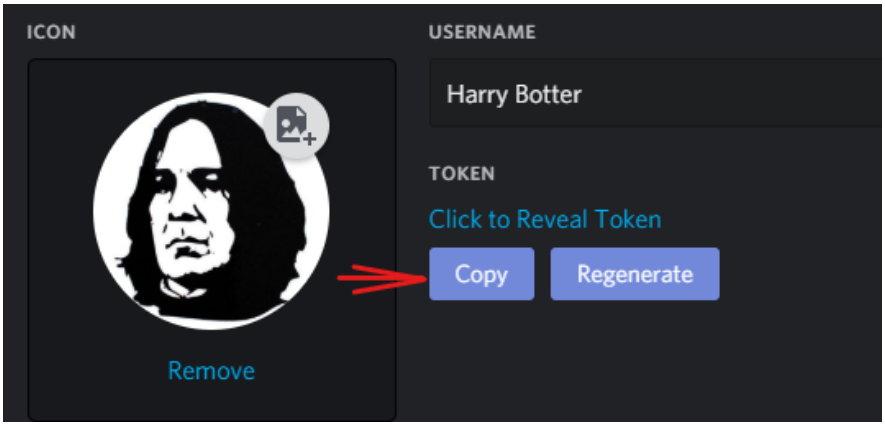

Рисунок 2.44 – Розташування авторизаційного токену та кнопка копіювання

Тепер можна перевірити роботу сценарію. Щоб його запустити необхідно викликати командний рядок в директорії де розміщується файл та ввести команду «node bot.js» [2]. Результат виконання команди показано на рисунках 2.45 та 2.46.

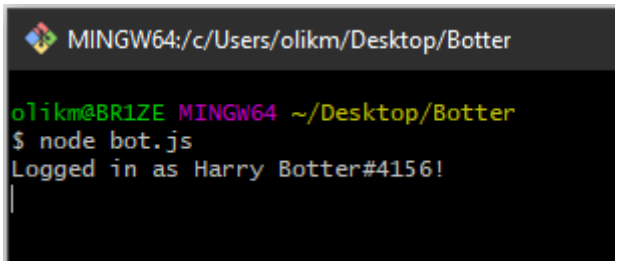

Рисунок 2.45 – Результат виконання у командному рядку

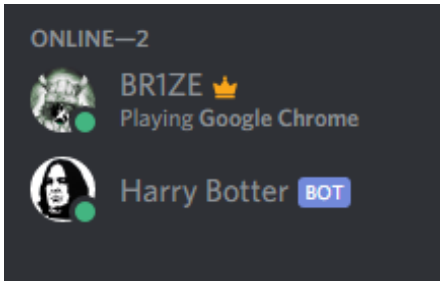

Рисунок 2.46 – У списку учасників бот відображається в онлайн статусі

Тепер можна перевірити команду, яка міститься в коді. При введенні слова «ping», бот повинен надіслати у відповідь слово «pong» (рисунок 2.47).

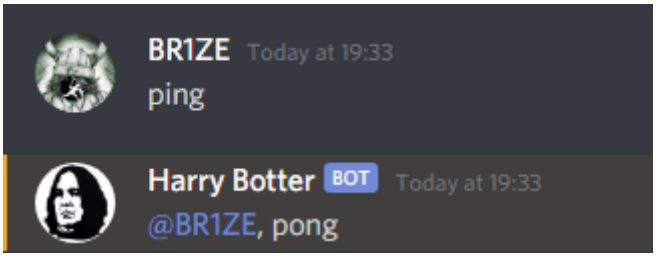

Рисунок 2.47 – Відповідь на команду «ping»

Рядок з токеном завжди повинен бути останнім, але це не дуже зручно, якщо його знадобиться редагувати, а він буде використовуватись в коді декілька разів. Тому його слід замінити змінною, а значення змінної повинне задаватись на початку коду. Результат такої заміни показано на рисунку 2.48.

Рисунок 2.48 – Заміна рядка з токеном на відповідну змінну

Також, для швидкого запуску бота при необхідності постійних тестувань, можна створити .bat файл, який миттєво запускатиме сценарій, при його відкритті. Для цього створюємо пустий текстовий документ та змінюємо його розширення з .txt на .bat, в створеному файлі вписуємо код, поданий в лістингу 2.1. Результат виконання подано на рисунку 2.49.

Лістинг 2.1. – Код файлу runbot.bat

```
@ECHO off

node bot.js
GOTO end

:end
PAUSE
```

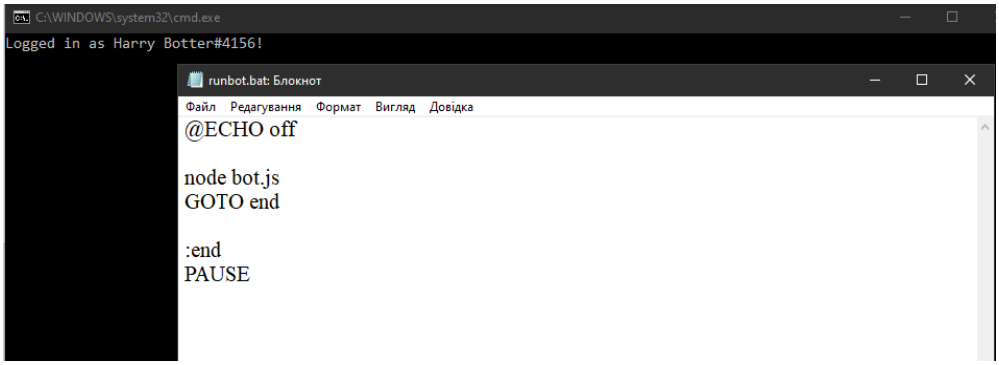

Рисунок 2.49 – Вигляд файлу runbot.bat та результат його виконання

Коли завантажено всі необхідні модулі та налаштовано роботу додатка з базовим кодом, можна переходити до написання різноманітного функціоналу.

2.7.2. Створення модераторського модуля

Модераторський модуль складатиметься з чотирьох команд, використовувати які, зможуть лише учасники з відповідними повноваженнями правами.

Команда «бану» є дуже важливою, вона дозволяє назавжди «забанити» учасника, який не дотримується правил поведінки.

Проте перед її написанням, всередині контейнера client.on{} слід задекларувати умови, при яких не потрібно виконувати команду. В лістингу 2.2. наведно команду, яка перевіряє повідомлення. В разі якщо: канал надсилання – особисті повідомлення, автор команди – бот, або повідомлення не починається з префіксу – повідомлення буде проігноровано [1].

Лістинг 2.2 – Перевірка повідомлення за трьома параметрами

```
    if(message.author.bot || message.channel.type == "dm" || !message.content.startsWith(prefix))
    return
```

Сама команда повинна виглядати наступним чином: !ban @nik, де nik – згадка імені учасника, якого потрібно забанити. Також командою передбачено деякі помилки введення та сповіщення в разі їх виявлення, а також сповіщення

про успішне виконання. Код команди представлено в додатку Д, а результати виконання на рисунках 2.50 та 2.51.

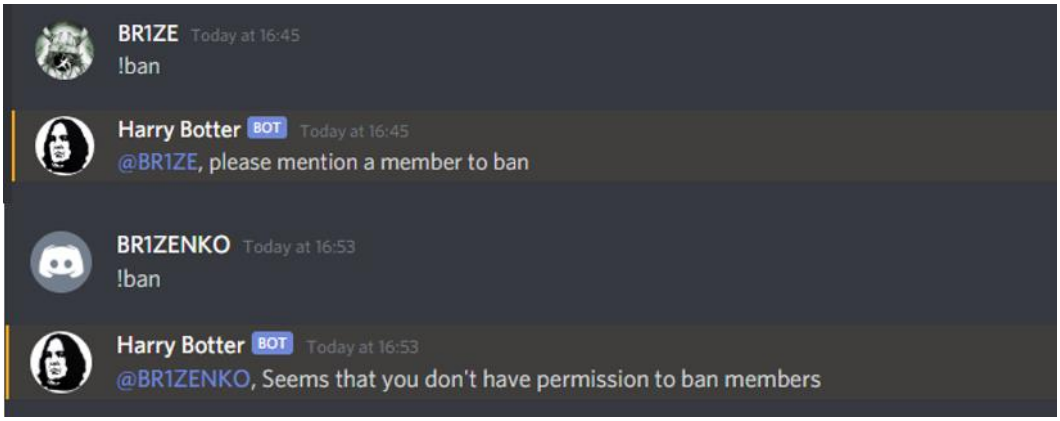

Рисунок 2.50 – Реакції бота на команду без згадки та спробу здійснити команду без повноважень

Рисунок 2.51 – Відповідь в разі успішного виконання команди

Також важливою опцією є швидке очищення повідомлень. Вона часто буває корисною, коли винувату в спамі людину вже покарано, але текстові канали потребують модерації. Для використання команди достатньо її ввести, вказавши кількість повідомлень, що необхідно видалити. Важливо пам’ятати що боти можуть видаляти лише повідомлення не старіші за 14 днів, таке обмеження було зроблено через особливість роботи бази даних, що використовується в Discord.

Для моніторингу було додано команду, яка вестиме лог використання цієї команди в командному рядку.

Декларування асинхронної функції та команду для логування подано в лістингу 2.3, а сама функція наведена в додатку Е. Результати роботи команди представлено на рисунках 2.52 – 2.54.

Лістинг 2.3 – Опис асинхронної функції та команда для логування

```
client.on('message', async message => {}
    console.log(message.author.username + " deleted "+ howmanydelete + " messages");
```

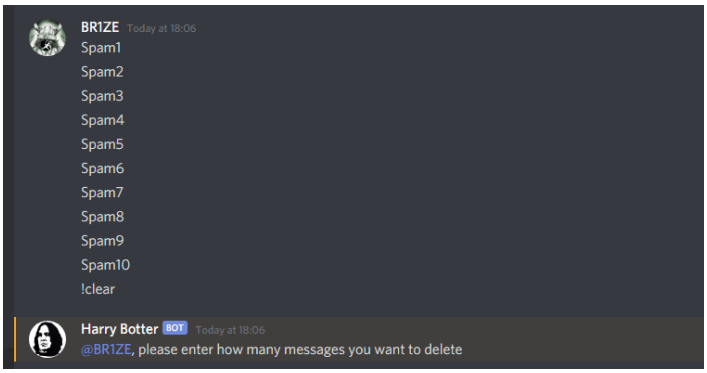

Рисунок 2.52 – Результат введення команди без параметру

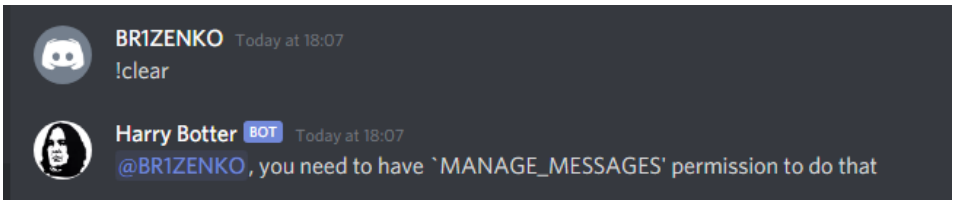

Рисунок 2.53 – Результат введення команди без прав

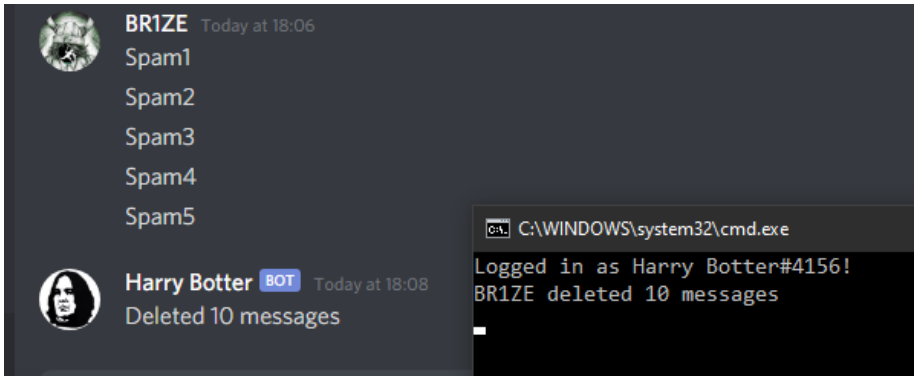

Рисунок 2.54 – Результат введення команди

Необхідно розуміти, що видалення великої кількості повідомлень це не миттєва операція, тому для коректної роботи скрипта, дану функцію необхідно описати, як асинхронну(async). Виконання даної функції переводить бота в

режим очікування, до її завершення. Ця операція триває декілька секунд, але даний спосіб допомагає уникнути різних конфліктів у коді [9].

2.7.3 Створення зображень з привітанням для нових учасників

Функція привітань є приємним доповненням до функціоналу бота. Код цієї функції створює оригінальне зображення з привітанням для кожного учасника при його долученні до сервера. Код функції подано в додатку Ж, а результат роботи зображено на рисунку 2.55.

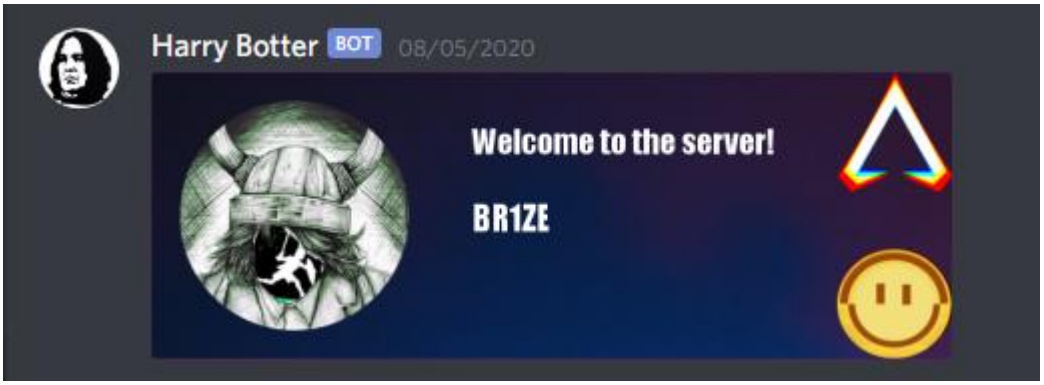

Рисунок 2.55 – Вітальне зображення для нових учасників

Ця функція є опціональною, оскільки не несе жодної практичної користі. Проте, більшість ботів в Discord мають таку функцію, інколи реалізовану у вигляді звичайного текстового повідомлення.

2.7.4 Створення модуля для програвання музики

Програвання музики однією з найпопулярніших функцій сучасних ботів, адже вона дозволяє колективно прослуховувати улюблені композиції та водночас спілкуватись.

Модуль складається з декількох частин: функцій програвання та керування програванням. Код функції подано в додатках И, К та Л, а результати роботи на рисунках 2.56 та 2.57.

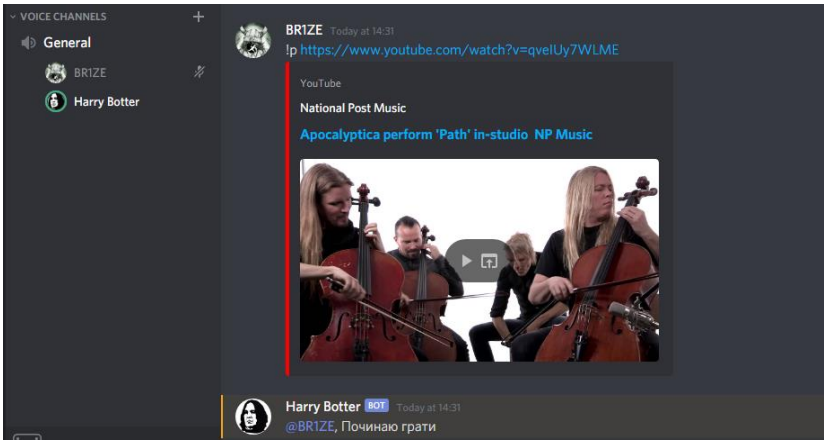

Рисунок 2.56 – Результат роботи функції

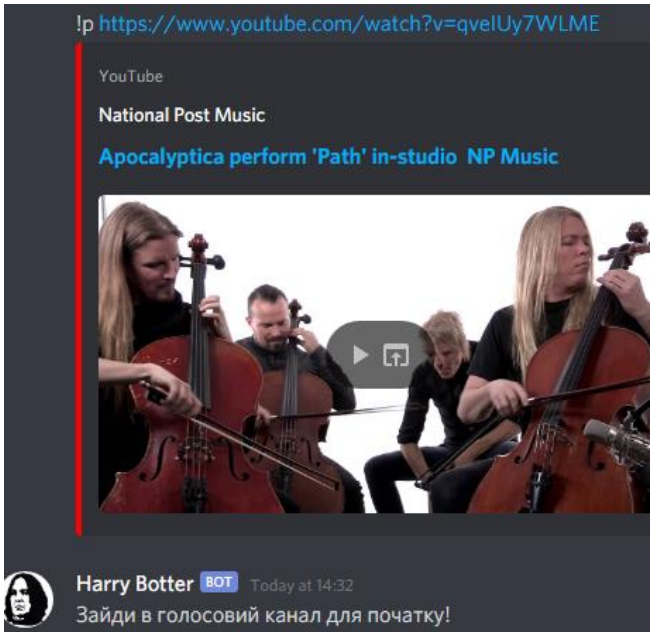

Рисунок 2.57 – Відповідь при відсутності користувача в голосовому каналі

Варто зауважити, що якщо команду програвання введе користувач який не зайшов в жоден голосовий канал сервера, бот відмовить у виконанні команди. Це зроблено з метою економії ресурсів у випадку зловживання, та уникнення спаму голосових композицій.

2.7.5 Створення системи економіки та рівневої системи

Система економіки є дуже популярною на великих серверах. Завдяки ній, кожен учасник сервера може накопичувати та використовувати віртуальну валюту. Для реалізації функції необхідно створити профіль для кожного

учасника в базі даних. Код функції подано в додатку М, а результат роботи зображено на рисунку 2.58.

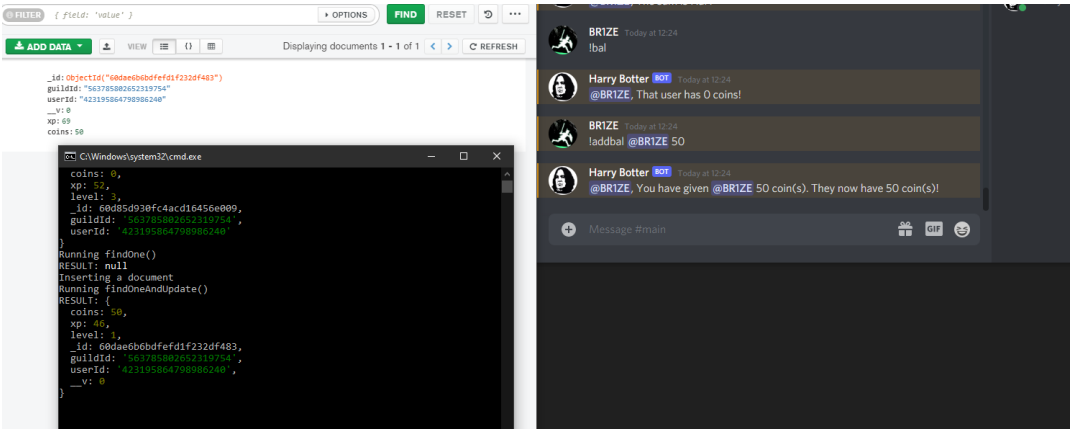

Рисунок 2.58 – Результат роботи функції

Якщо в базі немає користувача що виконує команду, для нього створюється профіль за готовим шаблоном. При повторному виконання команд, існуючий профіль буде зчитуватись або редагуватись.

2.8 Висновки до другого розділу

В ході виконання роботи було створено повністю робочого чат-бота з такими функціями:

- Бан учасників;
- Кік учасників з сервера;
- Вітання учасників при вході на сервер;
- Підтримка рівневої системи;
- Підтримка економічної системи з віртуальною валютою;
- Видалення вказаної кількості повідомлень.

Для функціонування бота необхідна база даних MongoDB, для підключення використовується посилання на сервер (локальне або глобальне) яке задається в файлі конфігурації. Жодних додаткових налаштувань БД, виконувати не потрібно.

За потреби, код можна модифікувати для сервера кожного клієнта.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Діяльність. Її види та розуміння в безпеці праці

Діяльність – це форма активного відношення людини до навколишнього природного середовища, яке вона цілеспрямовано перетворює це процес створення людиною умов для свого існування та розвитку.

Залежно від мети, змісту та форм розрізняють три основні різновиди діяльності: гру, навчання та працю.

Гра – це вид поведінки тварин та діяльність людини, ціллю якої є саме здійснення діяльності, а не практичні результати, котрі досягаються з її допомогою.

Навчання – це діяльність, безпосередньою ціллю якої є засвоєння людиною певних знань, навиків та вмінь.

Праця – свідома діяльність людини, спрямована на створення матеріальних і духовних благ. Вона є необхідною умовою існування та розвитку людини.

Людині незалежно від віку властиві всі три різновиди діяльності, проте в різні періоди життя вони виявляються по-різному за метою, змістом, формою та значенням. У дошкільному віці провідним різновидом діяльності є гра, у шкільному – навчання, а в зрілому – праця.

Праця відзначається значною різноманітністю. За характером роботи її можна поділити на три основні види: фізична праця, механізовані форми фізичної праці і розумова праця.

Доля фізичної і психічної складових у різних видах трудової діяльності неоднакова: під час фізичної праці переважає м’язова діяльність, а під час розумової – психічна. Але жоден з видів діяльності не відбувається без її регулювання центральною нервовою системою.

Під фізичною працею розуміють виконання людиною енергетичних функцій у виробничій системі. Ця праця вимагає значної м’язової активності. За характером роботи м’язів фізична робота поділяється на динамічну і статичну. Динамічна робота здійснюється при переміщенні тіла людини, її рук, ніг, пальців у просторі, статична – при утриманні вантажу, при виконанні роботи стоячи або сидячи.

Особливістю статичної роботи є її виражена втомлювальна дія, що зумовлена довготривалим скороченням і напруженням м’язів, безперервним збудженням нервових центрів, в той час як динамічна робота характеризується ритмічним скорочення м’язів, що сприяє повноцінному їх кровопостачанню і газообміну, почерговим збудженням і гальмування нервових центрів, що регулюють діяльність м'язів, що, у свою чергу, призводить до меншої втомленості.

Динамічну фізичну роботу, за якої задіяні більше 2/3 м'язів людини, прийнято називати загальною, при участі в роботі від 2/3 до 1/3 м'язів (тулуба або рук чи ніг) – регіональною, при участі в роботі менше 1/3 м'язів (наприклад, набір тексту на комп'ютері) – локальною.[1]

Механізовані форми фізичної праці виконуються людиною-оператором, їх здійснення забезпечується поєднанням фізичних і розумових функцій. Діяльність людини-оператора може відбуватися як у штатних (детермінованих), так і позаштатних (недетермінованих) обставинах. При детермінованих обставинах працюючому заздалегідь відомі алгоритми дій, він керується відповідними правилами, інструкціями, працює за жорстким технологічним графіком. У недетермінованих обставинах можливі збої у технологічному процесі, неполадки у роботі устаткування та ін., які усуваються за відповідними інструкціями.

Розумова праця людини, на відміну від фізичної, супроводжується меншими витратами енергетичних запасів (витрати енергії складають від 2500 до 3000 Ккал на добу), але це не означає, що вона є легкою.

Розумова праця забезпечується активністю головного мозку – під час розумової діяльності значно активуються аналітичні та синтетичні функції центральної нервової системи, ускладнюється прийом і переробка інформації, виникають функціональні зв’язки між окремими нервовими центрами, нові комплекси умовних рефлексів, зростає роль уваги, пам'яті, зорового та слухового аналізаторів.

Інтенсивна розумова праця викликає значне зростання потреб мозку в кисні. Будь-яка розумова діяльність супроводжується певним нервовопсихічним напруженням, малорухливістю, вимушеною позою тощо.

Розумова праця характеризується напруженістю, яка визначається обсягом інформаційного навантаження.

До розумової діяльності належать деякі види операторської праці, праця керівників виробничих процесів, творча працю, праця слідчих, суддів, лікарів, викладачів.

3.2 Вимоги безпеки до робочих місць для виконання робіт за персональним комп'ютером

Сучасний розвиток технічного та технологічного стану виробництва передбачає постійну автоматизацію та оптимізацію виробничих процесів. Сьогодні, напевно, важко уявити компанію, господарська діяльність в якій би здійснювалась без використання комп’ютерної техніки. Через масовий характер робіт, що виконуються працівниками за допомогою комп’ютера, законодавством України чітко врегульовано норми та вимоги до використання комп’ютерної техніки на підприємстві, безпосередньо й охорона праці при роботі з комп’ютером.

Приміщення, в яких планується установка та подальша робота з комп’ютером, повинні відповідати проектній документації будинку, погодженій з уповноваженими державними органами. Крім того, роботодавець повинен враховувати санітарні нормативи освітлення, вимоги до параметрів

мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів.

Правила поширюються на умови й організацію праці при роботі з візуальними дисплейними терміналами (ВДТ) усіх типів вітчизняного та зарубіжного виробництва на основі електронно-променевих трубок (ЕПТ), що використовуються в електронно-обчислювальних машинах (ЕОМ) колективного використання та персональних ЕОМ (ПЕОМ). Так, наприклад, роботодавцю заборонено установлювати комп'ютери в приміщеннях, розташованих у підвалах будинків. Для уникнення можливих аварій та замикань, поряд з приміщеннями, де вестиметься робота з комп'ютером (над чи під ними), також не дозволяється проведення робіт, що потребують здійснення надмірно вологих технологічних процесів. Відповідне приміщення повинно бути укомплектоване системами центрального або індивідуального опалення, кондиціонування чи вентиляції повітря. Але при установці зазначених систем, необхідно переконатись, що батареї опалення, водопровідні труби, вентиляційні кабелі тощо, надійно сховані під захисними щитками, які перешкоджатимуть можливому потраплянню робітника під напругу.[2]

У кожній кімнаті, де обладнуватимуться робочі місця співробітників, що працюватимуть на комп'ютері, повинні бути наявні елементи природного та штучного освітлення. При цьому, на вікнах слід встановити легко регульовані жалюзі чи штори, які дозволять працівникам коригувати рівень освітлення в приміщенні. Бажано розмістити комп'ютери в кімнаті таким чином, щоб світло потрапляло на екрани моніторів з півдня чи північного сходу. З метою досягнення максимального рівня безпечності і охорони праці при роботі з комп'ютером, виробничі приміщення необхідно обладнати аптечками першої медичної допомоги, системами автоматичної пожежної сигналізації і вогнегасниками. В приміщенні, в якому разом працюють 5 або більше комп'ютерів, на видимому місці установлюється службовий вимикач, який у разі потреби дозволить повністю відключити електричне живлення кімнати.

Роботодавець, який використовує найману працю робітників, повинен забезпечити відповідність їхніх робочих місць комфортним та безпечним умовам.

Площа одного робочого місця повинна бути не менше 6 м2, а обсяг – не менше 20 м3. При розміщенні робочих місць необхідно дотримуватись таких вимог:

− природне світло повинно падати збоку, переважно зліва;

− відстань від робочого місця до стін зі світловим прорізами повинна складати не менше 1 м;

− відстань між бічними поверхнями відеотерміналів має бути не меншою за 1,2 м;

− відстань між тильною поверхнею одного відеотермінала та екрана − іншого не повинна бути меншою 2,5 м, а прохід між рядами робочих − місць – не меншим одного метра.

Висота робочої поверхні столу для відеотермінала має бути в межах 68-80 см, а ширина повинна забезпечувати можливість використання операцій у зоні досяжності моторного поля (розміри столу: висота – 72,5 см, ширина – 60-140 см, глибина – 80-100 см).

Робоче сидіння (сидіння, стілець, крісло) працівника на обчислювальній техніці повинно бути підйомно-поворотним, плоским, спереду закругленим, а для усунення статичного напруження м’язів рук улаштоване стаціонарними або змінними підлокітниками.

Екран відеотермінала та клавіатура мають розташовуватися на оптимальній відстані від очей працівника, але не ближче 60 см, з урахуванням розміру алфавітно-цифрових знаків та символів [2].

При потребі високої концентрації уваги під час виконання робіт з високим рівнем напруженості суміщені робочі місця з відеотерміналами та персональними ЕОМ необхідно відділяти одне від одного перегородками висотою 1,5 - 2 м.

Організація робочого місця з ЕОМ для управління технологічними обладнаннями має передбачати:

− достатній простір для людини-оператора;

− розташування екрана відеотермінала в робочій зоні, яке забезпечувало б зручність зорового спостереження у вертикальній площині під кутом плюс-мінус 30° від лінії зору оператора;

− можливість повертання екрана відеотермінала навколо горизонтальної та вертикальної осі.[1]

3.3 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи описано діяльність, її види, роль та розуміння з точки зору безпеки життєдіяльності та охорони праці.

Проаналізовано основні види праці, їхні відмінності та вимоги до людини (працівника) який займається цим видом праці.

Описано вимоги до робочого місця при роботі з персональним комп’ютером, які встановлені законодавством України та державними санітарними нормами.

Охорона праці має тісний зв’язок з такими науками, як безпека життєдіяльності, гігієна і фізіологія праці, психологія, ергономіка, інженерна психологія, соціологія, екологія та технічна естетика. Вона використовує досягнення цих наук для обґрунтування і створення оптимальних умов праці та безпеки життєдіяльності людини.

ВИСНОВКИ

В ході виконання роботи було створено повністю робочого чат-бота з такими функціями:

– Бан учасників.

– Кік учасників з сервера.

– Вітання учасників при вході на сервер.

– Підтримка рівневої системи.

– Підтримка економічної системи з віртуальною валютою.

– Видалення вказаної кількості повідомлень.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Подано визначення поняття «бот», його призначення та класифікацію.

– Розглянуто поширені види ботів та приклади їх застосування. Зокрема, досліджено популярного бота для месенджера Discord – MEE6.

– Висвітлено роль ботів в мережі Інтернет, соціальних мережах та месенджерах.

– Проаналізовано використання ботів для потреб бізнесу, а також особистого використання для потреб окремих користувачів.

В другому розділі кваліфікаційної роботи:

– Розроблено чат-бота для месенджера Discord на платформі Node.js. Також використано базу даних MongoDB для постійного зберігання інформації про користувачів.

– Спроектовано та реалізовано модульну систему функцій бота, при якій функції (команди) незалежні між собою.

– Протестовано всі функції та результати їх роботи.

У розділі «Безпека життєдіяльності, основи хорони праці» подано визначення поняття «діяльність», описано її види та розуміння в безпеці праці. Описано вимоги безпеки до робочих місць для виконання робіт за персональним комп'ютером.

ПЕРЕЛІК ДЖЕРЕЛ

1. Атенсио Л. Функціональне програмування на JavaScript. Як поліпшити код JavaScript-програм / Л. Атенсио. – Київ: Вільямс, 2020. – 304 с.

2. Браун Е. Вивчаємо JavaScript: керівництво по створенню сучасних веб-сайтів / Е. Браун. – Київ: Вільямс, 2017. – 368 с.

3. Браун І. Веб розробка з використанням Node та Express / І. Браун. – Хмельницький: Пітер, 2016. – 336 с.

4. Даккет Дж. Javascript і jQuery. Інтерактивна веб-розробка / Дж. Даккет – Київ: Ексмо, 2016. – 640 с.

5. Дейлі Б. Розробка веб-додатків за допомогою Node.js, MongoDB і Angular: вичерпне керівництво з використання стека MEAN / Б. Дейлі. – Львів: Руслан, 2020. – 656 с.

6. Джанарсанам С. Розробка чат-ботів та розмовних інтерфейсів / С. Джанарсанам. – Київ: ДМК, 2018. – 340 с

7. Пауерс Ш. Вивчаємо Node.js / Ш. Пауерс. – Київ: Пітер, 2018. – 400 с.

8. Бот (програма) [Електронний ресурс] // Wikipedia. – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Робот_(програма).

9. Сімпсон К. Ви не знаєте JS. Асинхронна обробка та оптимізація / К. Сімпсон – Київ: Пітер, 2019. – 352 с.

10. Негус К. FFmpeg / К. Негус – Дніпро: Вайлі, 2020. – 912 с.

11. Янг А. Node.js в дії / А. Янг. – Київ: Пітер, 2018. – 512 с.

12. MongoDB [Електронний ресурс] // Wikipedia. – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MongoDB

ДОДАТКИ

Додаток А

Модераторські команди бота MEE6

Команда	Дія
!ban	Банить учасника.
!tempban	Банить учасника на визначений термін.
!clear	Видаляє вказану кількість повідомлень в каналі.
!infractions	Показує кількість порушень учасника.
!kick	Виганяє учасника з серверу.
!mute	Забирає можливість написання повідомлень в учасника.
!tempmute	Забирає можливість написання повідомлень в учасника на вказаний термін.
!role–info	Видає інформацію про вказану роль(її повноваження, доступ до каналів, обмеження…)
!server–info	Виводить інформацію про сервер.
!slowmode	Вмикає повільний режим в чаті, при якому для кожного користувача(крім адміністратора) вводиться обмеження на швидкість написання повідомлень.
!unmute	Повертає можливість написання повідомлень для учасника.
!user–info	Виводить інформацію про вказаного учасника.
!warn	Виносить попередження вказаному учаснику.

Додаток Б

Базовий набір коду для використання Discord.js

```
const Discord = require('discord.js');
const client = new Discord.Client();

client.on('ready', () => {
  console.log(`Logged in as ${client.user.tag}!`);
});

client.on('message', msg => {
  if (msg.content === 'ping') {
    msg.reply('pong');
  }
});

client.login('token');
```

Додаток В

Набір коду зі сторінки модуля Discord.js

```
const Discord = require('discord.js');
const client = new Discord.Client();
client.on('ready', () => {
  console.log(`Logged in as ${client.user.tag}!`);
});
client.on('message', msg => {
  if (msg.content === 'ping') {
    msg.reply('pong');
  } });
client.login('token');
```

Додаток Г

Мінімальний код з яким можна запустити бота

```
const Discord = require('discord.js');
const client = new Discord.Client();
client.on('ready', () => {
  console.log(`Logged in as ${client.user.tag}!`);
});
client.on('message', msg => {
  if (msg.content === 'ping') {
    msg.reply('pong');
  } });
client.login('NzA1Nzc1MzI0MzAxODg1NTIy.Xrp7ug.Na9UkkAZ5u0bWOOV8XGz-7xqBuw');
```

Додаток Д

Кінцевий код команди ban

```
      client.on('message', message => {
         if(message.author.bot   ||   message.channel.type   ==   "dm"   ||
!message.content.startsWith(prefix)) return

        let args = message.content.slice(prefix.length).trim().split(' ')
       if(message.content.startsWith(prefix + "ban")){

if(!message.member.hasPermission("BAN_MEMBERS"))return(message.reply("See
ms that you don't have permission to ban members"))
          let toban = message.mentions.members.first()

          if(!toban)return(message.reply("please   mention   a   member   to
ban"))
          let reason = args.splice(2).join(" ")

          const ban_embed = new RichEmbed()
          .setTitle(message.author.username    +    '    banned    '    +
toban.user.username)
          .setDescription('Reason :' + reason)
          .setColor("RANDOM")
          .setTimestamp()

          toban.ban(reason)
          message.channel.send(ban_embed)
        }
      });
```

Додаток Е

Кінцевий код функції видалення повідомлень

```
      client.on('message', async message => {
          if(message.author.bot    ||    message.channel.type    ==    "dm"    ||
!message.content.startsWith(prefix))return
        let args = message.content.slice(prefix.length).trim().split(' ')
        if(message.content.startsWith(prefix + "clear")){
      if(!message.member.hasPermission("MANAGE_MESSAGES"))return(message.reply("
you need to have \`MANAGE_MESSAGES\` permission to do that"))
          let howmanydelete = args[1]
      if(!howmanydelete)return(message.reply("please enter how many messages you
want to delete"))
          if(howmanydelete  <  1)return(message.reply("please  enter  number  more
than 1"))
          if(howmanydelete > 100)return(mesasge.reply("please enter number lower
than 100"))
          await message.channel.bulkDelete(howmanydelete)
          message.channel.send(`Deleted \`${howmanydelete}\` messages`)
          console.log(message.author.username + " deleted "+ howmanydelete + "
messages");
        }
      });
```

Додаток Ж

Код функції створення зображень

```
client.on('guildMemberAdd', async member => {
  console.log(member.user.username + ' + 1 Member!')
const canvas = Canvas.createCanvas(700, 250);
const ctx = canvas.getContext('2d');
const applyText = (canvas, text) => {
    const ctx = canvas.getContext('2d');
    let fontSize = 70;

    do {
         ctx.font = '30px impact';
    } while (ctx.measureText(text).width > canvas.width - 300);

    return ctx.font;
};
const background = await Canvas.loadImage('./wall1.jpg');
ctx.drawImage(background, 0, 0, canvas.width, canvas.height);

ctx.strokeStyle = '#74037b';
    ctx.strokeRect(0, 0, canvas.width, canvas.height);

    ctx.font = '28px impact';
    ctx.fillStyle = '#ffffff';
    ctx.fillText('Welcome to the server!', canvas.width / 2.5,
canvas.height / 3.5);


    ctx.font = applyText(canvas, member.displayName);
    ctx.fillStyle = '#ffffff';
    ctx.fillText(member.displayName, canvas.width / 2.5, canvas.height /
1.8);

    ctx.beginPath();
    ctx.arc(125, 125, 100, 0, Math.PI * 2, true);
    ctx.closePath();
    ctx.clip();

    const avatar = await Canvas.loadImage(member.user.displayAvatarURL);

    ctx.drawImage(avatar, 25, 25, 200, 200);


    const attachment = new Discord.Attachment(canvas.toBuffer(),
'text.jpg');
   member.guild.channels.get('Channel_ID').send(attachment)
  });
```

Додаток И

Код створення черги програвання та команд керування

```
const queue = new Map();
client.once('ready', () => {
    console.log('Ready to play!');
});
client.once('reconnecting', () => {
    console.log('Reconnecting!');
});
client.once('disconnect', () => {
    console.log('Disconnect!');});});
client.on('message', async message => {
    if (message.author.bot) return;
    if (!message.content.startsWith(prefix)) return;
    const serverQueue = queue.get(message.guild.id);

    if (message.content.startsWith(prefix +'p')) {
         execute(message, serverQueue);
         return;
    } else if (message.content.startsWith(prefix+'skip')) {
         skip(message, serverQueue);
         return;
    } else if (message.content.startsWith(prefix+'stop')) {
         stop(message, serverQueue);
         return;
    } else {
    return;
```

Додаток К

Код функції програвання

```
async function execute(message, serverQueue) {
    const args = message.content.split(' ');

    const voiceChannel = message.member.voiceChannel;
    if (!voiceChannel) return message.channel.send('Зайди в голосовий
канал для початку!');
    const permissions = voiceChannel.permissionsFor(message.client.user);
    if (!permissions.has('CONNECT') || !permissions.has('SPEAK')) {
        return message.channel.send('Я не маю прав!');
    }

    const songInfo = await ytdl.getInfo(args[1]);
    const song = {
        title: songInfo.title,
        url: songInfo.video_url,
    };

    if (!serverQueue) {
        const queueContruct = {
            textChannel: message.channel,
            voiceChannel: voiceChannel,
            connection: null,
            songs: [],
            volume: 5,
            playing: true,
        };

        queue.set(message.guild.id, queueContruct);

        queueContruct.songs.push(song);

        try {
            var connection = await voiceChannel.join();
            queueContruct.connection = connection;
            play(message.guild, queueContruct.songs[0]);
        } catch (err) {
            console.log(err);
            queue.delete(message.guild.id);
            return message.channel.send(err);
        }
    } else {
        serverQueue.songs.push(song);
        console.log(serverQueue.songs);
        return message.channel.send(`${song.title} додав до черги!`);
    }

}
```

Додаток Л

Код функцій керування програванням

```
      function skip(message, serverQueue) {
          if (!message.member.voiceChannel) return message.channel.send('Зайди в
голосовий канал для початку!');
          if (!serverQueue) return message.channel.send('Нема пісень які я можу
пропустити!');
          serverQueue.connection.dispatcher.end();
      }

      function stop(message, serverQueue) {
          if (!message.member.voiceChannel) return message.channel.send('Зайди в
голосовий канал для початку!');
          serverQueue.songs = [];
          serverQueue.connection.dispatcher.end();
      }

      function play(guild, song) {
          const serverQueue = queue.get(guild.id);

          if (!song) {
               serverQueue.voiceChannel.leave();
               queue.delete(guild.id);
               return;
          }

          const dispatcher = serverQueue.connection.playStream(ytdl(song.url))
               .on('end', () => {
                    console.log('Music ended!');
                    serverQueue.songs.shift();
                    play(guild, serverQueue.songs[0]);
               })
               .on('error', error => {
                    console.error(error);
               });
          dispatcher.setVolumeLogarithmic(serverQueue.volume / 5);
      }
```

Додаток М

Код функції створення профілю користувача в базі даних

```
const mongoose = require('mongoose')

const reqString = {
  type: String,
  required: true,
}

const profileSchema = mongoose.Schema({
  guildId: reqString,
  userId: reqString,
  coins: {
    type: Number,
    default: 0,
  },
  xp: {
    type: Number,
    default: 0,
  },
  level: {
    type: Number,
    default: 1,
  },
})

module.exports = mongoose.model('profiles', profileSchema)
```