МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ТЕКСТОВОЇ DISCORD RPG ГРИ»**

Виконав: студент 4 курсу, групи 6.1229
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми комп'ютерні науки
(назва освітньої програми)
Д. О. Жеребцов
(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)

" 30 " січня 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ СТУДЕНТОВІ

Жеребцову Дмитру Олександровичу

1. Тема роботи Розробка текстової Discord RPG гри

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 ботів
4. Проєктування та реалізація гри

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	10.02.2023	
	теоретичних джерел		
4.	Розробка першого та другого розділів	18.02.2023	
5.	Розробка третього розділу	23.03.2023	
6.	Оформлення та нормокнтроль	12.06.2023	
	кваліфікаційної роботи бакалавра		
7.	Захист кваліфікаційної роботи	20.06.2023	

Студент _____ Д.О. Жеребцов
(підпис) (ініціали та прізвище)

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

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка текстової Discord RPG гри»: 45 с., 24 рис., 8 джерел.

БОТ, КОМП'ЮТЕРНА ГРА, DISCORD, DISNAKE, PYTHON, ROLE-PLAYING GAME

Мета кваліфікаційної роботи: створення RPG гри із застосуванням мови програмування Python для серверу Discord.

Об'єкт дослідження – текстові ігри на тему RPG на базі платформи Discord.

Метод дослідження – аналіз, описовий, порівняльний.

У кваліфікаційній роботі представлено текстову гру – бот для Discord на тему RPG. У процесі розробки було розглянуто усі кроки які потрібно зробити для того, щоб почати створювати своїх власних ботів для реалізації ігор на платформі Discord із застосуванням мови програмування Python та бібліотеки Disnake.

SUMMARY

Bachelor’s qualifying theses « Development of a Text’s Discord RPG-game»: 45 pages, 24 figures, 8 references.

BOT, COMPUTER GAME, DISCORD, DISNAKE, PYTHON, ROLE-PLAYING GAME.

The purpose of the qualification work: creating an RPG game using the Python programming language for the Discord server.

The object of the research is text games on the RPG theme based on the Discord platform.

The research method is analysis, descriptive, comparative.

The qualifying paper presents a text game – bot for Discord on the theme of RPG. In the development process, all the steps that need to be taken in order to start creating your own bots for implementing games on the Discord platform using the Python programming language and the Disnake library were considered.

ЗМІСТ

ВСТУП

Discord – це популярна платформа для спілкування, яка надає можливість взаємодіяти з іншими людьми через текстові, голосові та відео-канали. Однією з переваг Discord є можливість використання ботів, які дозволяють автоматизувати різні функції та робити багато корисних речей для користувачів.

Боти для Discord можуть бути корисними для спільнот, які використовують платформу для спілкування, гри та інших видів діяльності. Наприклад, деякі боти можуть допомогти з управлінням сервером, модерацією та фільтрацією контенту, або навіть створенням власних команд.

Також боти можуть бути корисними для тих, хто хоче грати в текстові RPG ігри. Деякі боти дозволяють створювати та грати в різноманітні RPG - ігри, такі як Dungeons & Dragons або Pathfinder. Зазвичай ці боти використовуються для розіграшу віртуальних битв, розігрування діалогів та інших елементів гри.

Текстові ігри є одним з найстаріших типів відеоігор. Вони базуються на використанні текстових описів дій та інтеракцій з гравцем. З часом, текстові ігри стали популярнішими та складнішими, включаючи графічні елементи та складну логіку гри. Одним з типів текстових ігор є Discord (Role-Playing Game), які базуються на імітації фантастичних світів та розвитку персонажів гравців [7]. Для розробки та гри в текстові ігри однією з популярних платформ є Discord – сервіс для спілкування онлайн за допомогою відео, голосу або тексту [4].

В роботі, що пропонується, представлено розробку текстової гри на тему RPG для Discord з використанням бібліотеки Disnake [3].

1 РЕАЛІЗАЦІЯ DISCORD БОТУ

1.1 Поняття дискорд-боту

Дискорд-боти є автоматизованими програмами, які додають різні функціональні можливості до платформи Discord [5]. Дискорд – це популярна програма для спілкування та організації групових чатів, яка широко використовується геймерами, спільнотами, командами та іншими користувачами з усього світу.

Боти – це програмні рішення, які дозволяють автоматизувати певні завдання на різних платформах, включаючи соціальні мережі, чат-платформи, веб-сайти та інші. Боти можуть бути корисними для різних галузей, від маркетингу та реклами до спілкування з користувачами та автоматизації рутинних завдань. Боти можуть бути розроблені для виконання різних завдань, таких як:

– відповіді на запитання користувачів;

– автоматичні повідомлення та сповіщення;

– обробка замовлень та операцій;

– створення звітів та аналітики;

– моніторинг та аналіз даних;

Одним з найбільш популярних способів розробки ботів є використання мов програмування Python, особливо для розробки ботів для чат-платформи Discord.

Створення бота на базі Python для Discord може бути досить простим, особливо якщо ви маєте досвід у програмуванні. Для розробки бота вам необхідно вивчити деякі ключові концепції, такі як API Discord та фреймворки для розробки ботів на Python. Також важливо розуміти вимоги безпеки та обмеження використання ботів на серверах Discord.

Деякі переваги використання дискорд-ботів:

Автоматизація: Боти дозволяють автоматизувати багато рутинних завдань, спрощуючи вам життя. Вони можуть виконувати автоматичне привітання нових користувачів, надсилати повідомлення з правилами чату, керувати музикою тощо.

Модерація: Багато дискорд-ботів надають функції модерації для збереження порядку на сервері. Вони можуть автоматично видаляти спам, фільтрувати ненормативну лексику, керувати доступом до певних каналів та надавати інструменти для керування членами сервера.

Музика: Дискорд-боти можуть відтворювати музику з різних джерел, таких як YouTube або Spotify, безпосередньо в голосових каналах Discord. Вони надають можливість створювати плейлисти, керувати гучністю та додавати інші функції, пов'язані з музикою.

Розваги: Деякі боти пропонують різні розважальні функції, такі як ігри, квести, генерація мемів або зображень. Вони можуть створювати команди, які дозволяють користувачам грати та взаємодіяти між собою.

Налаштування: Дискорд-боти дозволяють адміністраторам серверів налаштувати різні параметри та функції для того, щоб відповідати потребам їхньої спільноти. Вони надають багато налаштувань для керування ролевою моделлю, правами доступу та іншими аспектами сервера.

Загалом, боти є потужним інструментом, який може спростити та автоматизувати багато рутинних завдань та покращити ефективність вашого бізнесу або проекту.

1.2 Поняття комп’ютерної гри. RPG

Комп’ютерні ігри – це форма розваги, що надає гравцям можливість взаємодіяти з віртуальним середовищем, керуючи персонажами та виконуючи різноманітні завдання та виклики. RPG є одним з найпопулярніших жанрів

комп’ютерних ігор, який здобув широке визнання та популярність серед гравців усього світу.

Рольові ігри відрізняються від інших жанрів своєю основною механікою – можливістю керувати персонажем або групою персонажів, розвивати їх навички та характеристики, взаємодіяти з іншими персонажами та вирішувати завдання в рамках вигаданого світу гри. Цей жанр виник у традиційних настільних рольових іграх, де гравці сідали за стіл з книгою правил та кидком кубика, але з розвитком комп’ютерної технології RPG стали доступними на екранах комп’ютерів та ігрових консолей.

Основні риси рольових ігор включають наявність фантастичного або вигаданого світу, систему розвитку персонажів, наявність квестів та завдань для виконання, а також можливість вибору та впливу на хід сюжету гри. Гравець може визначати вигляд свого персонажа, його характеристики, навички та способи боротьби з ворогами або розв’язання проблем. RPG часто пропонують великий вибір варіантів і можливостей, що робить гру цікавою та непередбачуваною.

Насамперед, рольові ігри відрізняються за типом гри та підходом до розвитку персонажів. Деякі RPG базуються на постійному просуванні гравця по сюжетній лінії та виконанні завдань, що призводять до розвитку персонажа і розкриття історії. Інші RPG пропонують відкритий світ, де гравець може вільно переміщатися та взаємодіяти з ним, виконуючи завдання або досліджуючи оточуючий світ.

2 СТВОРЕННЯ НОВОГО ДОДАТКУ НА ПЛАТФОРМІ DISCORD

2.1 Етапи розробки боту

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 [1], JavaScript, Java та інших.

Етапи, які потрібні для початку розробки свого боту для Discord:

Етап 1 Створення Discord-бота

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

Етап 2 Встановлення Python

Другим кроком є встановлення Python на комп’ютері. Можна завантажити Python з офіційного сайту Python.org.

Етап 3 Встановлення бібліотеки Disnake

Disnake – це бібліотека для розробки ботів для Discord на Python, яка є розширеною версіє. бібліотеки Discord.py [3]. Можна встановити бібліотеку Disnake за допомогою менеджера пакетів Python, наприклад pip: для цього виконати наступну команду у терміналі:

pip install disnake

Етап 4 Підключення бота до API Discord

Для підключення бота до API Discord можна скористатися токеном, який отримано на першому кроці. Необхідно створити новий файл Python та додати наступний код:

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

TOKEN = ‘токен_що отримано’

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)
```

Етап 5 Після додавання коду, необхідно вставити унікальній токен у відповідний рядок.

Цей код підключить вашого 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.

2.2 Важливі аспекти розробки боту

Також є багато важливих 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 ПРОЄКТУ

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 над проектом, використано мову програмування Python [2].

В процесі розробки розглянуто такі теми:

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

3.1 Створення боту на порталі Discord Developers

Для створення боту на порталі Discord Developers [5] необхідно перейти на портал Dicord Developers за посиланням https://discord.com/developers/applications/, де пропонується створити свого бота натиснувши на кнопку “New Application” (див. рис. 3.1).

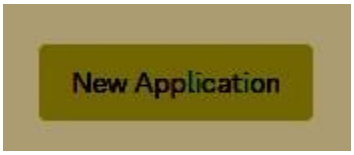

Рисунок 3.1 – Кнопка «New Application»

Після цього дати назву боту, яка буде відображатися на платформі Discord (див. рис. 3.2).

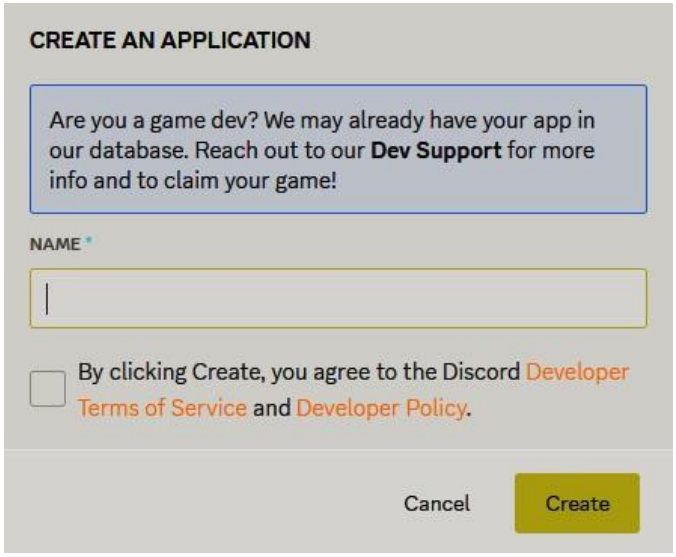

Рисунок 3.2 – Вікно назви бота

Відкривається сторінка, де можна редагувати та налаштовувати бота (див. рис. 3.3).

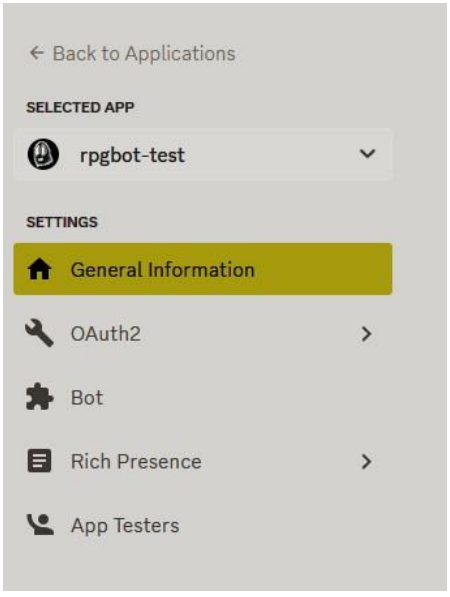

Рисунок 3.3 – Налаштування бота

Тепер можна налаштовувати аватар, назву та права, які будуть доступні для нього.

Для цього потрібно знайти вкладку Bot (див. рис. 3.4) – де знаходиться токен, який ми будемо додавати до нашого коду (див. рис. 3.5).

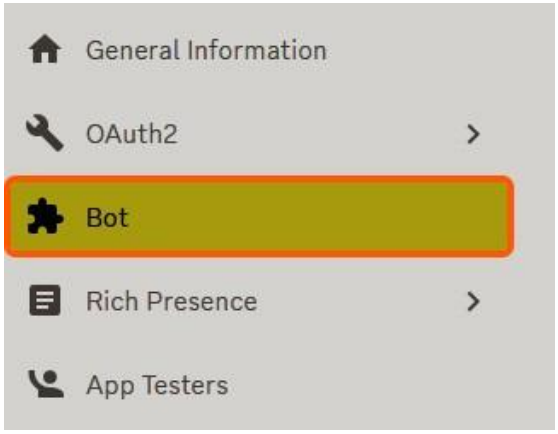

Рисунок 3.4 – Інформаційна кнопка «Bot»

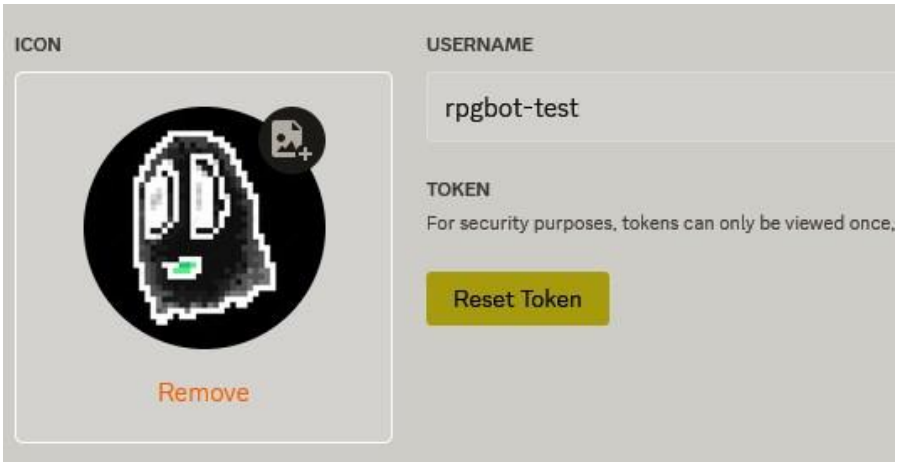

Рисунок 3.5 – Вікно з кнопкною «Reset Token»

Тепер потрібно взяти посилання, щоб додати нашого бота на сервер. Для цього ми переходимо на вкладку “OAuth2” (див. рис. 3.6), де обираємо, що це бот (див. рис. 3.7), та копіюємо посилання на нього (див. рис. 3.8).

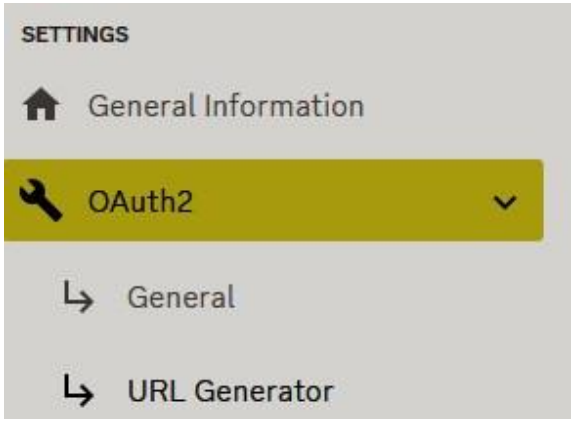

Рисунок 3.6 – Кнопка «OAuth2»

Рисунок 3.7 – Перелік вибору

Рисунок 3.8 – Посилання на бота

Переходячи за посиланням, обирається сервер, на який буде додано розроблений (див. рис. 3.9).

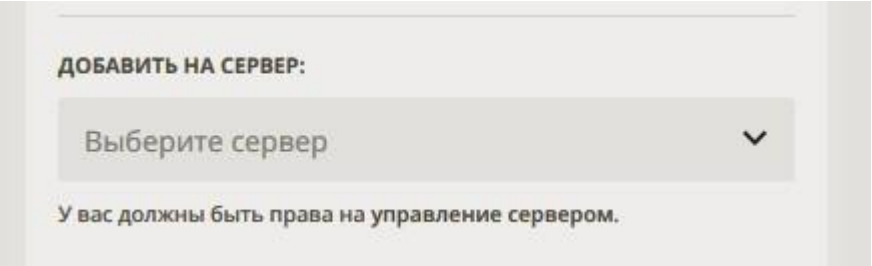

Рисунок 3.9 – Кнопка з вибором сервера

3.2 Створення файлу програми для бота

Після додавання бота на наш сервер, потрібно створити файл, при запуску якого наш бот буде авторизовуватись на сервер. Для цього нам потріібно створити файл з розширенням .py (див. рис. 3.10).

Рисунок 3.10 – Створення проекту

Потрібно встановити основну бібліотеку, за допомогою якої, ми будемо реалізовувати код, це бібліотека Disnake.

Disnake – 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 (термінал), та ввести таку команду (див. рис. 3.11):

```
pip install disnake
```

Рисунок 3.11 – Команда завантаження бібліотеки «disnake»

Бібліотека встановлена, можемо приступати до наступних дій. Потрібно створити блок коду, який буде відповідати за запуск боту, у якому буде знаходитись наш токен (див. рис. 3.12).

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

bot.run('TOKEN')
```

Рисунок 3.12 – Код для запуску бота

Замість напису TOKEN треба вставити той токен який береться з сайту Discord Developers[5].

Перевіряємо нашого бота, запускаємо програму. У терміналі з’явилося таке повідомлення (див. рис. 3.13).

```
C:\Users\dimaa\AppData\Local\Pro
Bot rpgbot-test#8425 online!
```

Рисунок 3.13 – Індикація запуску бота

3.3 Створення допоміжних команди для боту

Боту, що розроблюється, можуть знадобитись такі команди, як “инфо”, яка буде видавати основну інформацію про автора, та самого бота.

Код для цієї команди:

```
@bot.command(name='інфо')
async def bot_info(ctx):
    # Створюємо Embed-повідомлення
    embed = disnake.Embed(title='Інформація про бота', color=310062)

    # Додаємо поля в Embed
    embed.add_field(name='Автор', value='Жеребцов Дмитро Олександрович', inline=True)
    embed.add_field(name='Навчальний заклад', value='Запорізький Національний Університет', inline=True)
    embed.add_field(name='Версія', value='1.0', inline=True)
    embed.add_field(name='Данні', value=f"Математичний факультет, комп'ютерні науки, группа 6.1229", inline=True)

    # Надсилаємо Embed-повідомлення
    await ctx.send(embed=embed)
```

При виконанні цієї команди отримуємо такий результат (див. рис. 3.14):

Рисунок 3.14 – Команда «!інфо»

Також знадобиться команда, яка буде показувати усі доступні у цьому боті команди, вона буде викликатися як “!команды”.

```
# Создаем команду для просмотра списка команд
@bot.command(name='команди')
async def show_commands(ctx):
    # Список усіх команд
    command_list = [f'`{command.name}`' for command in bot.commands]

    # Відправляємо список команд у канал
    await ctx.send(f'Список доступних команд: {", ".join(command_list)}')
```

3.4 Основні команди та класи для боту на тему RPG

Перед тим як створювати команди та класи для гри, необхідно визначитися з призначенням класів та команд. В першу чергу, потрібен клас персонажу Character (див. рис. 3.15):

Клас Character представляє гравця або персонажа. Він має властивості, такі як ім'я (name), здоров'я (health), атака (attack), захист (defense). Клас також містить початкові значення максимального здоров'я (max_health) і максимального захисту (max_defense).

```
class Character:
    def __init__(self, name, health, attack, defense):
        self.name = name
        self.max_health = health
        self.health = health
        self.attack = attack
        self.max_defense = defense
        self.defense = defense
```

Рисунок 3.15 – Клас персонажа «Character»

Клас Enemy представляє ворога (див. рис. 3.16). Він має аналогічні властивості, що і у класу Character. Клас також має метод take_damage, який віднімає від здоров'я ворога отриманий урон.

У коді створюються екземпляри класу Enemy для кожного ворога у грі. Ви можете додавати нових ворогів, використовуючи цей шаблон.

```
class Enemy:
    def __init__(self, name, health, attack, defense):
        self.name = name
        self.max_health = health
        self.health = health
        self.attack = attack
        self.max_defense = defense
        self.defense = defense

    def take_damage(self, damage):
        self.health -= damage
        if self.health < 0:
            self.health = 0
```

Рисунок 3.16 – Клас ворогів «Enemy»

Клас Trader представляє торговця (див. рис. 3.17). Він має властивості, такі як ім'я (name), здоров'я (health), атака (attack), захист (defense).

```
class Trader:
    def __init__(self, name, health, attack, defense):
        self.name = name
        self.health = health
        self.attack = attack
        self.defense = defense

trader = Trader("Торговец", 50, 10, 5)
```

Рисунок 3.17 – Клас торговця «Trader»

Клас Boss представляє боса або головного ворога у грі (див. рис. 3.18). Він також має аналогічні властивості, що і у класу Character.

```
class Boss:
    def __init__(self):
        self.name = "Горний Голем"
        self.health = 200
        self.attack = 20
        self.defense = 5
```

Рисунок 3.18 – Клас боса «Boss»

Також потрібно створити список з іменами та характеристиками ворогів, який буде опиратися на наш клас Enemy.

```
# Створюємо екземпляри класу Enemy для кожного ворога
enemies = [
    Enemy("Гоблін", 60, 10, 5),
    Enemy("Орк", 75, 15, 10),
    Enemy("Троль", 90, 20, 15),
    Enemy("Дракон", 150, 30, 20),
```

```
    Enemy("Демон", 100, 25, 25),
]
```

Також потрібно створити умову, за якою не можна запускати інші команди, поки не застосовано першу команду:

```
game_started = False
```

3.4.1 Команда !грати

Тепер можна створювати першу команду для створення імені, та вибору класа персонажа. Команда буде викликатись як !грати.

```
@bot.command(name='грати')
async def start_game(ctx):
    global game_started
    global character

    if game_started:
        await ctx.send("Гра вже почата.")
        return
    await ctx.send("Привіт! Давай почнемо гру!.")
    await ctx.send("Дайте назву вашому персонажу.")
    name_msg = await bot.wait_for('message', check=lambda message: message.author == ctx.author)
    name = name_msg.content
    await ctx.send(f"Супер, ім'я персонажа – {name}.")

    await ctx.send("Тепер оберіть клас для вашого персонажа.")
```

```
        await ctx.send("1. Воїн")
        await ctx.send("2. Лучник")
        await ctx.send("3. Маг")

        class_msg = await bot.wait_for('message', check=lambda message: message.author == ctx.author)
        if class_msg.content == "1":
            character_class = "Воїн"
            character = Character(name, 200, 40, 10)
        elif class_msg.content == "2":
            character_class = "Лучник"
            character = Character(name, 130, 60, 5)
        elif class_msg.content == "3":
            character_class = "Маг"
            character = Character(name, 140, 50, 2)
        else:
            await ctx.send("Такого класу не існує, спробуйте ще раз.")
            return

        await ctx.send(f"Чудово, ваш персонаж {name} – {character_class}.")

        await ctx.send("Тепер ви готові до бою! Скористайтеся командою !бій, щоб зустрітися з випадковим ворогом.")
        game_started = True
```

Алгоритм дій:

а) спочатку, перевіряється, чи гра вже розпочата. Якщо так, то повідомлення "Гра вже почата" надсилається користувачеві, і функція завершується;

б) якщо гра ще не почата, відбувається привітання з користувачем шляхом надсилання повідомлення "Привіт! Давай почнемо гру!";

в) користувачу пропонується надати ім'я свого персонажа через повідомлення "Дайте назву вашому персонажу." Код очікує на повідомлення з ім'ям від автора контексту (ctx.author);

г) після отримання повідомлення з ім'ям персонажа, зберігається отримане ім'я в змінну name;

д) користувачеві пропонується вибрати клас свого персонажа. Класи представлені через повідомлення "1. Воїн", "2. Лучник", "3. Маг";

е) код очікує на повідомлення з вибраним класом від автора контексту;

ж) залежно від отриманого повідомлення з вибраним класом, змінна character_class присвоюється відповідне значення ("Воїн", "Лучник" або "Маг"), а також створюється об'єкт character класу Character з відповідними параметрами;

з) якщо отримане повідомлення не відповідає жодному класу, надсилається повідомлення "Такого класу не існує, спробуйте ще раз." і функція завершується;

и) надсилається повідомлення, що підтверджує успішний вибір імені та класу персонажа, наприклад, "Чудово, ваш персонаж [ім'я] – [клас персонажа]";

к) користувачеві надсилається повідомлення "Тепер ви готові до бою! Скористайтеся командою !бій, щоб зустрітися з випадковим ворогом";

л) змінна game_started встановлюється в True для позначення того, що гра розпочалася.

Отже, цей код дозволяє користувачам почати гру, вибрати ім'я та клас персонажа, та отримати підтвердження про початок гри. Після цього користувачі можуть використовувати інші команди для гри, такі як команда !бій.

3.4.2 Команда !бій

Наступний крок – формування команди, у якій вже створений персонаж буде зустрічатися з випадковим ворогом. Система буде покроковою, гравець та ворог обирають дію: піти в атаку, або захищатися. Ця команда викликається як команда !бій:

```
@bot.command(name='бій')
async def start_battle(ctx):
    global game_started
    global current_enemy
    if current_enemy is None:
        enemy = random.choice(enemies)
    else:
        enemy = current_enemy

    if not game_started:
        await ctx.send("Спочатку почніть гру використавши команду !грати.")
        return

    character.health = character.max_health
    enemy.health = enemy.max_health
    character.defense = character.max_defense

    await ctx.send("Бій починається!")
    while True:
        current_enemy = random.choice(enemies)
        # Хід гравця
```

```
            embed = disnake.Embed(title=f"{character.name} ({character.health} HP) vs {enemy.name} ({enemy.health} HP)",
                                  color=disnake.Color.green())
            embed.add_field(name="Оберіть дію:", value="1. Атаковать\n2. Защищаться")
            msg = await ctx.send(embed=embed)

            def check(m):
                return m.author == ctx.author and m.channel == ctx.channel and m.content in ["1", "2"]

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

            if choice == 1:
                damage = character.attack
                await ctx.send(f"{character.name} атакує {enemy.name} і наносить {damage} урону!")
                current_enemy.health -= damage
            else:
                character.defense()
                await ctx.send(f"{character.name} захищаєтсья.")

            if current_enemy.health <= 0:
                await ctx.send(f"{enemy.name} помер!")
                break

            # Хід ворога
            embed = disnake.Embed(title=f"{character.name} ({character.health} HP) vs {enemy.name} ({enemy.health} HP)",
```

```
                color=disnake.Color.red())
            embed.add_field(name="Хід ворога:", value=f"{enemy.name} атакує!")
            await ctx.send(embed=embed)

            damage = current_enemy.attack
            await ctx.send(f"{enemy.name} атакує {character.name} і наносить {damage} урону!")
            character.health -= damage

            if character.health <= 0:
                await ctx.send(f"{character.name} загинув.")
                game_started = False
                return

        await ctx.send("Бій закінчено. Ви перемогли!")
        await ctx.send("Тепер ви можете відправитись у нову подорож за допомогою команди !подорож.")
```

Алгоритм дій:

а) спочатку перевіряється, чи гра вже почата (змінна game_started). Якщо гра не почата, надсилається повідомлення "Спочатку почніть гру використавши команду !грати." і функція завершується;

б) ініціалізується змінна enemy, яка представляє ворога. Якщо current_enemy є пустим, то ворог вибирається випадковим чином зі списку enemies. Інакше, використовується поточний ворог, який був зазначений попередньо;

в) здоров'я та захист гравця та ворога (об'єкти character і enemy) відновлюються до початкових значень;

г) надсилається повідомлення "Бій починається!";

д) запускається безкінечний цикл битви (while True), який триває до тих пір, поки гравець або ворог не помре;

е) хід гравця:
   – надсилається вбудоване повідомлення embed, яке показує інформацію про гравця та ворога;
   – користувачеві пропонується обрати дію: атакувати (вибір "1") або захищатися (вибір "2");
   – за допомогою функції check перевіряється, чи повідомлення з вибраним варіантом належить користувачу, який запустив команду, і чи воно знаходиться в тому ж каналі;
   – вибір гравця зберігається в змінну choice;
   – якщо вибір дорівнює 1, ворог отримує урон від атаки гравця, а повідомлення про це надсилається;
   – якщо вибір дорівнює 2, гравець збільшує свій захист, а повідомлення про це надсилається;

ж) перевіряється, чи ворог помер після ходу гравця. Якщо так, надсилається повідомлення про смерть ворога і цикл битви переривається;

з) хід ворога:
   – надсилається вбудоване повідомлення embed, яке показує інформацію про гравця та ворога;
   – ворог атакує гравця, і повідомлення про це надсилається;
   – гравець отримує шкоду від атаки ворога;

и) перевіряється, чи гравець помер після ходу ворога. Якщо так, надсилається повідомлення про смерть гравця, змінна game_started встановлюється в False, і функція завершується;

к) після кожного ходу перевіряється, чи була завершена битва. Якщо так, надсилається повідомлення про перемогу гравця, і функція завершується;

л) на кінці битви надсилаються повідомлення "Бій закінчено. Ви перемогли!" та "Тепер ви можете відправитись у нову подорож за допомогою команди !подорож."

Цей код дозволяє користувачам розпочати бій з випадковим ворогом, виконувати ходи атаки та захисту, і залежно від результату битви надсилаються відповідні повідомлення.

3.4.3 Команда !подорож

Наступний крок – створення команди, у якій можна відправитись у одну з двох локацій, Ліс або Гори. На локації Ліс гравець зустрічає торговця, який запропонує йому відгадати загадку: якщо гравець відгадує, торговець розповість, що знаходиться на локації Гори, якщо гравець не вгадає, торговець розізлиться та вступить з гравцем у бій.

Код для першої частини (Ліс):

```
@bot.command(name='подорож')
async def adventure(ctx):
    global game_started
    game_started = True
    tries = 0
    if not game_started:
        await ctx.send("Спочатку почніть гру за допомогою команди !грати")
        return

    await ctx.send("Ви вирушаєте у нову подорож...")
    # Вибір локации
    embed = disnake.Embed(title="Оберіть локацію:",
                          description="1. Ліс\n2. Гори")
```

```
msg = await ctx.send(embed=embed)

def check(m):
    return m.author == ctx.author and m.channel == ctx.channel and m.content in ["1", "2"]

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

if choice == 1:  # Ліс
    await ctx.send("Вы потрапили у ліс.")
    await asyncio.sleep(1)
    await ctx.send(
"Ви зустрічаєте Торговця, який пропонує вам відгадати його загадку. Якщо ви вгадаєте, він розповість вам, що чекає на вас у горах, якщо ні – ви будете битися з ним.")

    # Задаємо загадку
    riddle = "Що збільшується, але ніколи не зменшується?"
    await ctx.send(f"Загадка: {riddle}")

    # Перевіряємо відповідь користувача
    def check_answer(m):
        return m.author == ctx.author and m.channel == ctx.channel

    while tries < 3:
        answer_msg = await bot.wait_for('message', check=check_answer)
        answer = answer_msg.content.lower()
        if answer == "вік" or answer == "роки":
            await ctx.send(
```

```
                "Ви правильно відповіли на загадку! Торговець розповідає вам, що в горах ховається страшне чудовисько, з яким треба бути обережним.")
                return
            else:
                tries += 1
                await ctx.send(f"Ваша відповідь. У вас залишилось {3 – tries} спроби.")

        # Якщо не вгадали загадку за 3 спроби – починаємо битися з торговцем
        await ctx.send("Ви не вгадали загадки за 3 спроби! Торговець дістав зброю та нападає на вас")
        # Код для бійки з торговцем
        await ctx.send("Бій починається!")
        while True:
            # Хід гравця
            embed = disnake.Embed(
                title=f"{character.name} ({character.health} HP) vs {trader.name} ({trader.health} HP)",
                color=disnake.Color.green())
            embed.add_field(name="Выберите действие:", value="1. Атаковати\n2. Захищатися")
            msg = await ctx.send(embed=embed)

            def check(m):
                return m.author == ctx.author and m.channel == ctx.channel and m.content in ["1", "2"]

            choice_msg = await bot.wait_for('message', check=check)
```

```
            choice = int(choice_msg.content)

            if choice == 1:
                damage = character.attack
                await ctx.send(f"{character.name} атакує {trader.name} та наносить {damage} урону!")
                trader.health -= damage
            else:
                character.defense()
                await ctx.send(f"{character.name} захищається.")

            if trader.health <= 0:
                await ctx.send(f"{trader.name} загинув!")
                break

            # Хід торговця
            embed = disnake.Embed(
                title=f"{character.name} ({character.health} HP) vs {trader.name} ({trader.health} HP)",
                color=disnake.Color.red())
            embed.add_field(name="Хід торговця:", value=f"{trader.name} атакує!")
            await ctx.send(embed=embed)

            damage = trader.attack
            await ctx.send(f"{trader.name} атакує {character.name} та наносить {damage} урону!")
            character.health -= damage

            if character.health <= 0:
```

```
        await ctx.send(f"{character.name} загинув.")
        game_started = False
        return
```

Алгоритм дій:

а) спочатку перевіряється, чи гра вже почата (змінна game_started). Якщо гра не почата, надсилається повідомлення "Спочатку почніть гру за допомогою команди !грати" і функція завершується;

б) ініціалізується змінна tries, яка відслідковує кількість спроб відповіді на загадку;

в) надсилається повідомлення "Ви вирушаєте у нову подорож...";

г) гравець обирає локацію:
   – надсилається вбудоване повідомлення embed, яке пропонує вибрати локацію: ліс (вибір "1") або гори (вибір "2");
   – за допомогою функції check перевіряється, чи повідомлення з вибраним варіантом належить користувачу, який запустив команду, і чи воно знаходиться в тому ж каналі;
   – вибір користувача зберігається в змінну choice;

д) якщо вибір дорівнює 1 (ліс), надсилається повідомлення "Ви потрапили у ліс";

е) гравець зустрічає торговця:
   – надсилається послідовність повідомлень, що описують зустріч з торговцем та його пропозицію;
   – задається загадка за допомогою змінної riddle;
   – гравець намагається відповісти на загадку, із збереженням кількості спроб в змінну tries;
   – якщо гравець правильно відповів на загадку, надсилається повідомлення про правильну відповідь та інформація про чудовисько, що чекає у горах;

- якщо гравець не вгадав загадку за 3 спроби, надсилається повідомлення про невдалу спробу та розпочинається бій з торговцем;

ж) починається бій з торговцем:
- бій відбувається у циклі while True;
- надсилається вбудоване повідомлення embed, що відображає інформацію про гравця та торговця;
- гравець обирає дію: атакувати (вибір "1") або захищатися (вибір "2");
- залежно від вибору гравця відбувається атака або захист;
- перевіряється, чи торговець помер після ходу гравця. Якщо так, надсилається повідомлення про смерть торговця і цикл битви переривається;

з) хід торговця:
- надсилається вбудоване повідомлення embed, що відображає інформацію про гравця та торговця;
- торговець атакує гравця, і повідомлення про це надсилається;
- гравець отримує шкоду від атаки торговця;

и) перевіряється, чи гравець помер після ходу торговця. Якщо так, надсилається повідомлення про смерть гравця, змінна game_started встановлюється в False, і функція завершується.

Якщо обирається другу локація Гори, гравець зустрічається з великим монстром «Горний Колос», з яким вступає у бійку:

```
elif choice == 2: # Гори
    await ctx.send("Ви потрапили в гори.")
    await asyncio.sleep(1)
    await ctx.send("Ви зіткнулися із Горним Големом! Можливо.. вам не варто було приходити сюди..?")
    await asyncio.sleep(1)
```

```
            # Битва з босом

            boss = Boss()

            character.health = character.max_health
            boss.health = boss.health
            character.defense = character.max_defense

            await ctx.send("Бій з Горним Големом починається!")

            while True:
                # Хід гравця
                embed = disnake.Embed(
                    title=f"{character.name} ({character.health} HP) vs {boss.name} ({boss.health} HP)",
                    color=disnake.Color.green())
                embed.add_field(name="Оберіть дію:", value="1. Атакувати\n2. Захищатися")
                msg = await ctx.send(embed=embed)

                def check(m):
                    return m.author == ctx.author and m.channel == ctx.channel and m.content in ["1", "2"]

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

                if choice == 1:
                    damage = character.attack
```

```
                await ctx.send(f"{character.name} атакує {boss.name} і наносить {damage} урону!")
                boss.health -= damage
            else:
                character.defense()
                await ctx.send(f"{character.name} захищається.")

            if boss.health <= 0:
                await ctx.send(f"{boss.name} помер!")
                break

            # Хід боса
            boss_damage = boss.attack
            await ctx.send(f"{boss.name} атакує {character.name} і наносить {boss_damage} урону!")
            character.health -= boss_damage

            if character.health <= 0:
                await ctx.send(f"{character.name} помер! Гра закінчена.")
                game_started = False
                break

            tries += 1

        if game_started:
            await ctx.send(f"Вітаємо! Ви перемогли Горного Колоса у {tries} спроб!")
            await asyncio.sleep(1)
            await ctx.send("Тепер вам доступні всі команди.")
            await asyncio.sleep(1)
```

```
            await ctx.send("Ви можете продовжити грати або закінчити гру командою !завершити.")
        else:
            await ctx.send("Гру закінчено.")
    else:
        await ctx.send("Некоректний вибір локації.")
        return
```

Алгоритм дій:

а) гравець обрав локацію 2 (гори);

б) надсилається повідомлення "Ви потрапили в гори";

в) надсилається послідовність повідомлень, що описують зустріч з персонажем "Горний Голем";

г) ініціалізуються змінні для здоров'я гравця, боса та оборони гравця;

д) починається бій з ворогом "Горий Голем":

- бій відбувається у циклі while True;
- надсилається вбудоване повідомлення embed, що відображає інформацію про гравця та боса;
- гравець обирає дію: атакувати (вибір "1") або захищатися (вибір "2");
- залежно від вибору гравця відбувається атака або захист;
- перевіряється, чи бос помер після ходу гравця. Якщо так, надсилається повідомлення про смерть боса і цикл битви переривається;

е) хід боса:

- надсилається вбудоване повідомлення embed, що відображає інформацію про гравця та боса;
- бос атакує гравця, і повідомлення про це надсилається;
- гравець отримує урон від атаки боса;

ж) перевіряється, чи гравець помер після ходу боса. Якщо так, надсилається повідомлення про смерть гравця, змінна game_started встановлюється в False, і функція завершується;

з) якщо гра продовжується (змінна game_started дорівнює True), надсилаються повідомлення про перемогу гравця, кількість спроб та доступність всіх команд. Змінна game_started залишається True;

и) якщо гра закінчена (змінна game_started дорівнює False), надсилається повідомлення "Гру закінчено";

к) якщо вибір локації не є коректним (ні 1, ні 2), надсилається повідомлення "Некоректний вибір локації".

Після перемоги над Колосом відкривається доступ до усіх команд та гравець може або продовжити гру, або закінчити її.

3.4.4 Команда !завершити

Якщо потрібно завершити гру, гравець може скористатися командою !завершити:

```
@bot.command(name='завершити')
async def end_game(ctx):
    global game_started, character
    if not game_started:
        await ctx.send("Гра ще не почата.")
        return

    await ctx.send(f"Гра завершена.")
    game_started = False
    character = None
```

Алгоритм дій

а) перевіряється, чи гра вже розпочата (змінна game_started);

б) якщо гра не розпочата, надсилається повідомлення "Гра ще не почата";

в) якщо гра розпочата, надсилається повідомлення "Гра завершена";

г) змінна game_started встановлюється в False, що означає завершення гри;

д) змінна character встановлюється в None, що скидає інформацію про персонажа (якщо така була).

3.4.5 Демонстрація гри

Демонстрація гри представлена на рис. 3.19 –3.24.

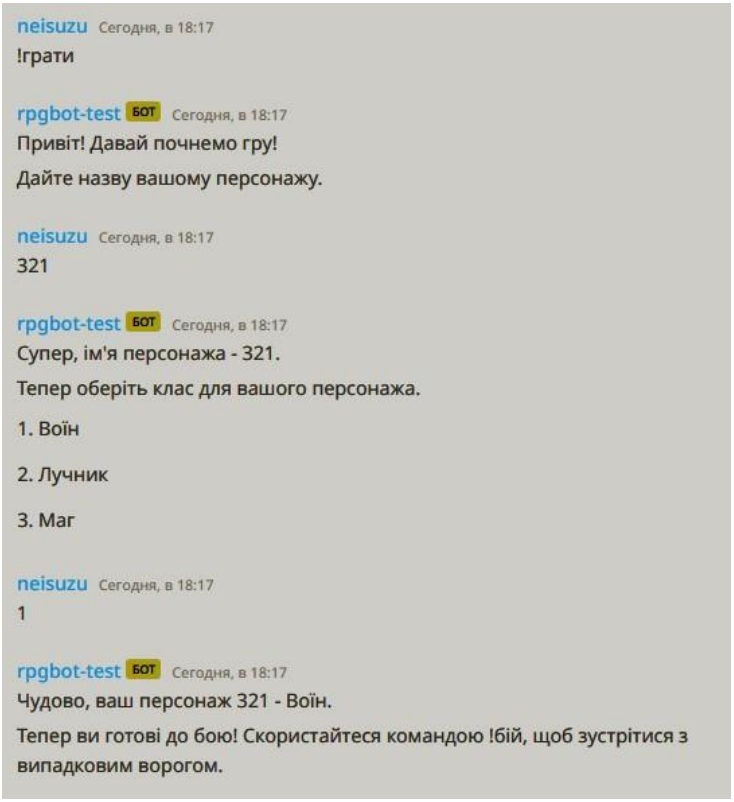

Рисунок 3.19 – Команда «!грати»

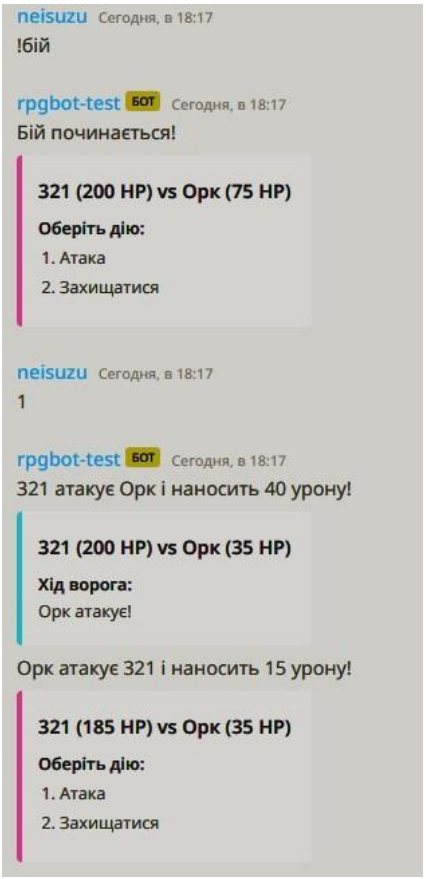

Рисунок 3.20 – Команда «!бій»

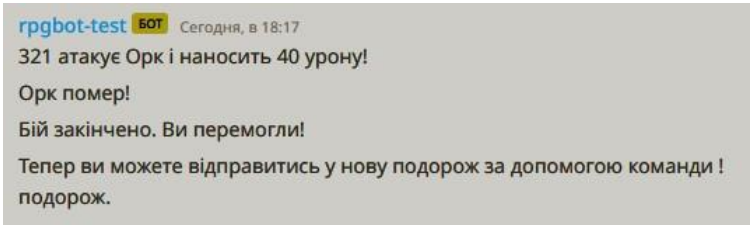

Рисунок 3.21 – Повідомлення перемоги

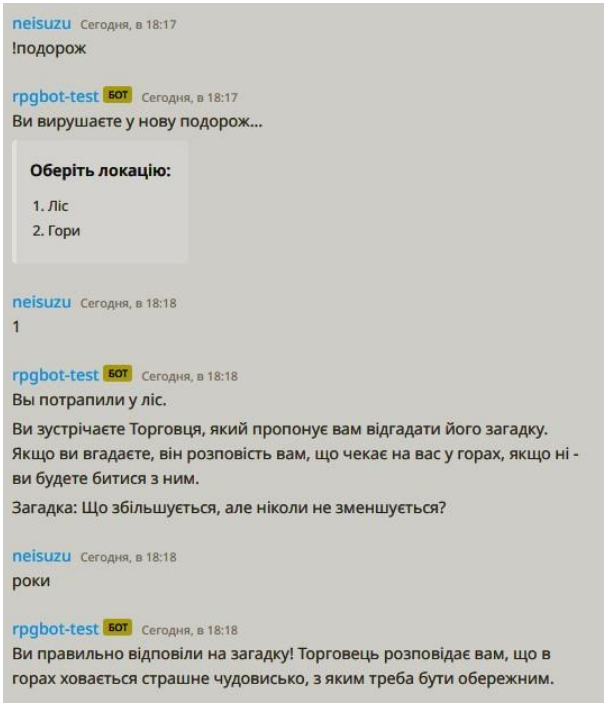

Рисунок 3.22 – Команда «!подорож» та локація «ліс»

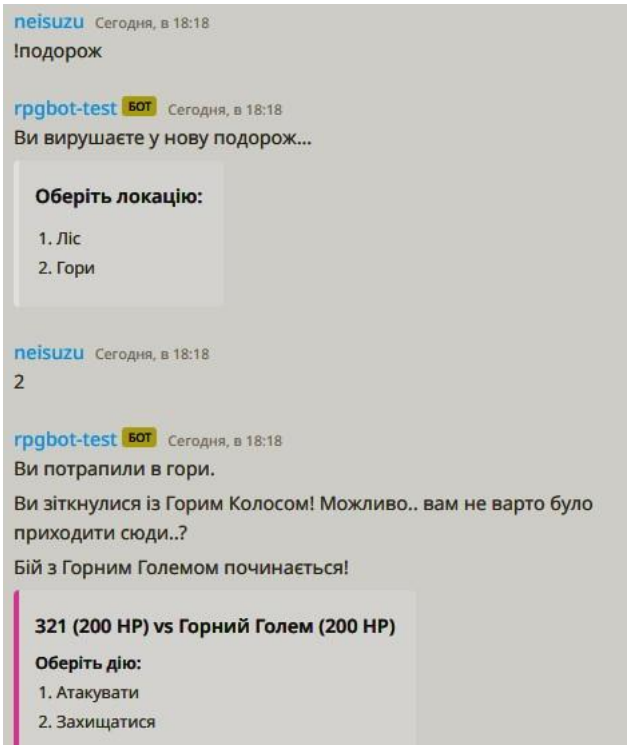

Рисунок 3.23 – Команда «!подорож» та локація «гори»

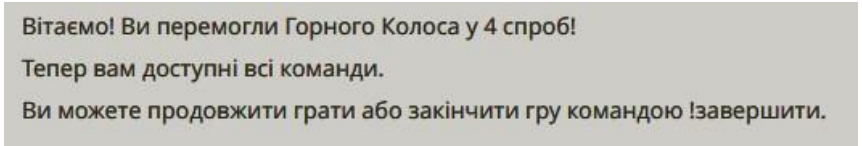

Рисунок 3.24 – Повідомлення після перемоги

ВИСНОВКИ

У кваліфікаційній роботі представлено текстову гру – бот для Discord на тему RPG. У процесі розробки було розглянуто усі кроки які потрібно зробити для того, щоб почати створювати своїх власних ботів для реалізації ігор на платформі Discord із застосуванням мови програмування Python та бібліотеки Disnake.

В процесі реалізації проєкту:

– створено обліковий запис боту;
– створено та налаштовано права боту;
– створено кодову частину для активації бота;
– розроблено допоміжні команди;
– розроблено основні команди для реалізації RPG частини боту;
– здійснено тестування та налагодження гри.

Результати кваліфікаційної роботи:

– розроблено бот для Discord на тему RPG;
– створено систему персонажу та вибору його класу;
– реалізовано основні команди та активності;
– прописані умови для використання команд;
– проведено тестування та налагодження проєкту.

Розробка текстової гри на тему RPG для Discord є досить складною задачею, яка потребує уваги до деталей та вміння працювати з різними технологіями та бібліотеками. Проте, результат розробки – захоплююча текстова гра, яка зможе зацікавити користувачів Discord.

ПЕРЕЛІК ПОСИЛАНЬ

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

2. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Чернігів : ФОП Баликіна С.М., 2020. 180 с.

3. Офіційна документація Disnake. URL : https://disnake.readthedocs.io/en/latest/ (дата звернення: 15.02.2023).

4. Офіційна документація Discord. URL : https://discord.readthedocs.io/en/latest/ (дата звернення: 21.02.2023).

5. Портал Discord Developer Portal з інформацією для розробників ботів. URL : https://discord.com/developers/docs/intro (дата звернення: 06.03.2023).

6. Яковенко А.В. Основи програмування. Python. Частина 1. Київ : КПІ ім. Ігоря Сікорського, 2018. 195 с.

7. Michael M. Basics of Game Design. Boca Raton, Florida : CRC Press, 2016. 400 p.

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