ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА

«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«РОЗРОБКА ТА ВПРОВАДЖЕННЯ ЧАТ-БОТУ З ПРОГРАМУВАННЯ ДЛЯ СТУДЕНТІВ СПЕЦІАЛЬНОСТІ КОМП'ЮТЕРНІ НАУКИ «ПРОГРАМУВАННЯ: BOT- CHALLENGE»

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавра

Виконавець роботи Ломака Богдан Анатолійович
_____ «____»_____ 2023 р.

Науковий керівник к.ф.-м.н Ольховська Олена Володимирівна
_____ «____»_____ 2023 р.

Рецензент д. ф.-м. н., доц. Тетяна Миколаївна Барболіна

ПОЛТАВА 2023 р.

ЗАТВЕРДЖУЮ
Завідувач кафедри _____Олена ОЛЬХОВСЬКА

«___» _____ 202__ р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка та впровадження чат-боту з програмування для студентів спеціальності Комп'ютерні науки «Програмування: Bot-Challenge»
зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня бакалавр
Прізвище, ім'я, по батькові Ломака Богдан Анатолійович
Затверджена наказом ректора № 144-Н від «01» вересня 2022 р.
Термін подання студентом роботи «___»_____ 202__ р.
Вихідні дані до кваліфікаційної роботи: чат-бот для проходження тестів з програмування.
Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
1.1. Постановка основних вимог до змісту.
1.2. Постановка задачі для реалізації чат бота: Bot-Challenge.
РОЗДІЛ 2. ТЕОРЕТИЧНА ЧАСТИНА
2.1. Бот в Телеграм: основні принципи і особливості роботи.
2.2. Як боти в Телеграм спростили життя мільйонів людей. Застосування ботів в повсякденному житті.
2.3. Алгоритм бота в Телеграм.
РОЗДІЛ 3. ІНФОРМАЦІЙНИЙ ОГЛЯД
3.1. Огляд відомих та популярних чат-ботів
3.2. Огляд чат-ботів для навчального процесу
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. План розробки Bot-Challenge
4.2. Розробка Bot-Challenge і бази даних для нього
4.3. Візуальне оформлення Bot-Challenge
4.4. Представлення результатів тестування і аналізу роботи розробленого тренажера
ВИСНОВКИ
СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ
Перелік графічного матеріалу: 39 ілюстрацій.

Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі	Ольховська О.В.		
Інформаційний огляд	Ольховська О.В.		
Теоретична частина	Ольховська О.В.		
Практична частина	Ольховська О.В.		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій та стандартів та звіт керівнику		
3. Постановка задачі		
4. Інформаційний огляд джерел бібліотек та інтернету		
5. Теоретична частина		
6. Практична частина		
7. Закінчення оформлення		
8. Доповідь студента на кафедрі		
9. Доробка (за необхідністю), рецензування		

Дата видачі завдання «__» _____ 202___ р.

Здобувач вищої освіти _____ Ломака Богдан Анатолійович
(підпис)

Науковий керівник _____ к.ф.-м.н Ольховська О.В.
(підпис) (науковий ступінь, ініціали та прізвище)

Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 2023 р.

Секретар ЕК_____ _____

Затверджую
Зав. кафедрою _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 202__ р.

Погоджено
Науковий керівник _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА
«____» _____ 202__ р.

План

кваліфікаційної роботи на тему
«Розробка та впровадження чат-боту з програмування для студентів спеціальності Комп'ютерні науки «Програмування: Bot-Challenge»
зі спеціальності 122 Комп'ютерні науки
освітня програма 122 «Комп'ютерні науки»
ступеня бакалавр
Прізвище, ім'я, по батькові Ломака Богдан Анатолійович

ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ
1.1. Постановка основних вимог до змісту.
1.2. Постановка задачі для реалізації чат бота: Bot-Challenge.
РОЗДІЛ 2. ТЕОРЕТИЧНА ЧАСТИНА
2.1. Бот в Телеграм: основні принципи і особливості роботи.
2.2. Як боти в Телеграм спростили життя мільйонів людей. Застосування ботів в повсякденному житті.
2.3. Алгоритм бота в Телеграм.
РОЗДІЛ 3. ІНФОРМАЦІЙНИЙ ОГЛЯД
3.1. Огляд відомих та популярних чат-ботів
3.2. Огляд чат-ботів для навчального процесу
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА
4.1. План розробки Bot-Challenge
4.2. Розробка Bot-Challenge і бази даних для нього
4.3. Візуальне оформлення Bot-Challenge
4.4. Представлення результатів тестування і аналізу роботи розробленого тренажера
ВИСНОВКИ
СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

Здобувач вищої освіти _____ Богдан ЛОМАКА
«____» _____ 202__ р.

РЕФЕРАТ

Записка: 71 сторінок, 39 ілюстрацій, 1 додаток, 9 літературних джерел.

СИСТЕМА ДИСТАНЦІЙНОГО НАВЧАННЯ, ЧАТ-БОТ, ОБ'ЄКТНА МОДЕЛЬ ДОКУМЕНТА (DOM).

Мета роботи – розробка бота в телеграм для проходження тестів з програмування: «Програмування: Bot-Challenge».

Об'єкт розробки – чат-бот для проходження тестів з програмування.

Методи дослідження – використання редактору коду «PyCharm» мову програмування Python та бібліотек OS, RANDOM, AIOGRAM, TOKENAPI, BDATA, TEXT.

Зроблено огляд систем дистанційного навчання, виділено їх позитивні та негативні сторони. Виконано опис проектних рішень, інструментів та підходів до розробки системи дистанційного навчання. Обрано методологію створення програмного забезпечення. Побудовано діаграму прецедентів. Розроблено чат-бот для проходження тестів з програмування. Розроблено інструкцію з користування чат-бота.

ЗМІСТ

ВСТУП 6
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ 8
1.1. Постановка основних вимог до змісту. 8
1.2. Постановка задачі для реалізації чат бота: Bot-Challenge. 9
РОЗДІЛ 2. ТЕОРЕТИЧНА ЧАСТИНА 10
2.1. Бот в Телеграм: основні принципи і особливості роботи. 10
2.2. Як боти в телеграм спростили життя мільйонів людей. Застосування ботів в повсякденному житті. Приклади. 14
2.3. Алгоритм бота в Телеграм. 19
РОЗДІЛ 3. ІНФОРМАЦІЙНИЙ ОГЛЯД 20
3.1. Огляд відомих та популярних чат-ботів. 20
3.2. Огляд чат-ботів для навчального процесу. 22
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА 24
4.1. План розробки Bot-Challenge. 24
4.2. Розробка Bot-Challenge і бази даних для нього. 25
4.3. Візуальне оформлення Bot-Challenge. 51
4.4. Представлення результатів тестування і аналізу роботи розробленого тренажера. 54
ВИСНОВКИ 58
СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ 59
ДОДАТОК А. ВИХІДНІ КОДИ 60

ВСТУП

Актуальність. У наш час Інтернет-технологій багато аспектів нашого життя переноситься в мережу, прискорюючи тим самим темпи розвитку інформаційного суспільства. Важливим елементом в освіті є дистанційне навчання, а саме проходження навчальних тестів, а також самовдосконалення різноманітними способами.

Зважаючи на те, що на разі дуже швидкий рівень життя, потрібно отримувати нові знання як найшвидше, тому що з появою Інтернета кожного року все менше і менше людей звертаються до паперових носії інформації. Також багато людей зараз є користувачами різноманітних соціальних мереж, зокрема Телеграму і ми вирішили об’єднати це все в одне ціле, створивши телеграм-бота, в якому студенти, зможуть проходити різноманітні тести на тему самої перспективної науки на даний момент, а саме програмування, можна не лише зекономити час, а й додати інтересу до проходження тестів, так як все буде в одному місці і не потрібно буде за різними тестами переходити на різні сайти.

Основною перевагою дистанційного проходження тестів над очною формою є можливість проходження тестів з будь-якого місця, де є доступ до Інтернету, а завдяки тому що це відбувається на платформі Телеграм з’єднання з користувачем більш стабільне. Завдяки тому, що бот реалізований в вигляді саме чат-боту, то це дає студентам можливість отримувати знання зручним для них способом, що дозволяє їм підлаштовувати навчання до свого графіку, стилю життя і особистих потреб. Крім того, дистанційна форма навчання дозволяє студентам самостійно визначати темп і глибину свого навчання, що забезпечує більш гнучкий підхід до навчання.

Мета кваліфікаційної роботи - розробка і реалізація чат-боту для студентів спеціальності Комп’ютерні науки «Програмування: Bot-Challenge».

Об’єктом роботи є процес дистанційного навчання комп’ютерним наукам.

Предметом - програмний продукт, що реалізує чат-бот для студентів на мові програмування Python.

Перелік використаних методів полягає у застосуванні мови Python та бібліотек OS, RANDOM, AIOGRAM, TOKENAPI, BDATA, TEXT.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

1.1. Постановка основних вимог до змісту

Матеріли курсового проекту повинні бути оформлені відповідно до методичних рекомендацій з виконання роботи студентами за освітньою програмою «Комп’ютерні науки». Зміст роботи повинен відповідати завданню затвердженим керівником та завідувачем кафедри.

Розглянемо основні положення викладення матеріалу:

- чіткість та логічна послідовність викладення матеріалу;
- переконливість аргументації;
- обґрунтованість рекомендацій та пропозицій.
- У роботі повинні бути відображеними:
- актуальність тематики та відповідність до сучасного стану науки, техніки і питань виробництва;
- обґрунтування вибраного напрямку досліджень, методів розв’язку задачі та їх порівнянь оцінки;
- аналіз та узагальнення існуючих результатів;
- характер і зміст виконаних теоретичних досліджень та розрахунків, методи досліджень;
- контрольні приклади;
- оцінка повноти розв’язку поставленої задачі;
- наукова або практична цінність виконаної роботи, виклад наукової новизни, якщо вона є.

1.2. Постановка задачі для реалізації чат бота: Bot-Challenge

Основним завданням роботи є розробка алгоритму чат бота: Bot-Challenge. Планується, що алгоритм чат-бота, який розробляється в рамках роботи буде запрограмовано та використовуватиметься, як складова дистанційного курсу «Комп'ютерні науки».

Розглянемо основні завдання роботи:

- провести вибір та обгрунтування мови програмування;
- розробити логічну схему взаємодії з ботом;
- розглянути теоретичні відомості та специфіку застосування чат-бота
- розробити алгоритм чат-бота.

Алгоритм, що розробляється є джерелом отримання навичок студентами, а отже, повинен бути зручним у користуванні. Тому необхідно розробити можливість студента звернутися до теоретичного матеріалу з теми, що допоможе йому у проходженні завдання.

РОЗДІЛ 2. ТЕОРЕТИЧНА ЧАСТИНА

2.1. Бот в Телеграм: основні принципи і особливості роботи

Технологічний світ не стоїть на місці, постійно з'являються нові цікаві тренди, деякі з них здатні серйозно вплинути на всю галузь IT в цілому. Якщо зовсім недавно всі тільки й говорили про додатки, то останнім часом на перший план виходять чат-боти, яким пророкують велике майбутнє у маркетингу та комунікаціях.

Що ж таке "чат-бот"? Це спеціалізована програма, яка базується на платформі обміну повідомленнями, дозволяючи користувачами взаємодіяти зі сторонніми сервісами через знайомий інтерфейс чату.

Тобто для того, щоб отримати певну інформацію, людині не потрібно залишати межі месенджера, достатньо відправити спеціальну команду, яка інтерпретується відповідним чином з боку бота. Наприклад, можна швидко отримати інформацію про погоду, або дізнатися про останні новини цікавого вам видання.

Особливу популярність боти набули завдяки творцям месенджера Telegram, які одними з перших надали широкі можливості для розробників у цьому плані. Фактично незалежні програмісти отримали можливість створення інтерактивних інтерфейсів із справді широкими можливостями.

Це свого роду електронні помічники, які дозволяють користувачеві отримати потрібну інформацію швидко, з мінімумом зусиль і не залишаючи вікно улюбленого месенджера.

Боти в Телеграм – це спеціальні програми, що виконують різні функції та спрощують життя їхніх користувачів. Написані для платформи Telegram, вони призначені для виконання різних функцій: від отримання новин до пошуку інформації і навіть торгівлі акціями!

Головним завданням бота є автоматична відповідь після введеної користувачем команди. При цьому, працюючи безпосередньо через інтерфейс Telegram, програма імітує події живого користувача, за рахунок чого користування таким роботом набагато зручніше і зрозуміло.

Саме тому багато компаній, що розвивають бізнес через інтернет, використовують можливості ботів з кількох причин:

1. Вони дозволяють задіяти черговий канал комунікації з цільовою аудиторією (в Україні телеграм користується близько 3 мільйонів людей).

2. Вони швидко виконують однакову роботу, дозволяючи розвантажити найманих співробітників, тим самим заощаджуючи гроші компанії[1].

Боти можуть виконувати різноманітні завдання, такі як пошук в інтернеті чи державних реєстрах, покупки, платежі, розваги, модерація груп, тестування студентів тощо. У спілкуванні беруть участь користувач Telegram та комп'ютерна програма від стороннього розробника[2].

Види ботів у Telegram

У Telegram використовується один загальний вигляд роботів, яких від традиційних користувачів відрізняє лише наявність приставки «bot» в імені. Самі боти діляться на кілька напрямків:

- Чат-боти. Представляють собою найпростіший чат, що імітує спілкування на задану користувачем тематику.
- Боти-інформатори. Окремий вид ботів, головна мета яких – інформування користувача про ті чи інші події (новини, заходи, публікації тощо).
- Ігрові боти. Боти, в яких можна пограти у різні ігри.
- Боти-помічники. Боти, розроблені різними онлайн-сервісами як доповнення до основної веб-версії[1].
- Користувач може взаємодіяти з ботом за допомогою елементів інтерфейсу месенджера: надсилання повідомлень, натискання на команди та

кнопки, використання онлайн-режиму. Telegram надає три способи взаємодії користувача з ботом: приватний чат (класичний спосіб), група й так званий онлайн-режим:

- Найпоширеніший спосіб — приватний чат. Бот може ініціювати діалог з користувачем лише у двох випадках: авторизація в сторонньому застосунку через Telegram і подання заявки на приєднання до чату.
- Деякі боти можуть бути учасниками груп. Наприклад, у групах бот може підтримувати розмову, модерувати повідомлення або бути ведучим гри.
- Онлайн-режим нагадує інтерфейс пошукової системи. Користувач уводить у поле для введення повідомлень запит, що починається з короткого імені бота. Далі користувач може вибрати й надіслати один з результатів[2].

Чат-бот (англ. chatbot) - це програма, яка імітує реальну розмову з користувачем. Чат-боти дозволяють спілкуватися за допомогою текстових або аудіо повідомлень на сайтах, месенджерах, мобільних додатках або по телефону.

Чат-боти використовують машинне навчання виявлення моделей спілкування. Завдяки постійній взаємодії з людьми вони навчаються наслідувати реальні розмови і реагують на усні або письмові запити, допомагаючи знайти відповіді. Оскільки чат-боти використовують штучний інтелект (ІІ), то розуміють мову, а чи не просто команди. Таким чином, після кожного діалогу вони стають розумнішими. Варто відзначити, що окрім чат-ботів з ІІ, є й ті, які працюють на основі запрограмованих сценаріїв множинним вибором, наприклад, опція А веде до опції В і таке інше[7].

Режим приватності. Ботов часто додають до груп, щоб отримувати різну інформацію - новини, повідомлення і т.д. Саме тому роботи мають режими приватності.

Робот з увімкненим режимом приватності не отримуватиме всіх повідомлень, а лише повідомлення, що задовольняють цим умовам:

- Повідомлення, що починаються з символу косої межі "/"
- Повідомлення, що містять @згадування бота
- Відповіді на повідомлення бота
- Службові повідомлення (про додавання користувача, зміни зображення групи тощо)

Це добре з усіх боків: по-перше, деякі люди спатимуть спокійно, не побоюючись, що їх прослуховуватимуть. По-друге, режим приватності позбавляє розробників необхідності обробляти сотні непотрібних повідомлень з групових чатів.

Режим приватності включений за замовчуванням у всіх роботах. Він може бути вимкнений – тоді бот почне отримувати всі повідомлення, як і звичайний користувач. Всім учасникам конференції видно поточний статус режиму приватності у списку учасників групи.

Рекомендується відключати режим приватності лише у випадках нагальної потреби. У переважній більшості випадків, запиту примусової відповіді повідомлення бота буде достатньо.

2.2. Як боти в телеграм спростили життя мільйонів людей. Застосування ботів в повсякденному житті. Приклади.

Роботи виконують дії за текстовими командами користувача, за принципом «відповідь» після натискання кнопки «Старт». Так, наприклад, Харківський Університет спільно з українською ІТ-компанією розробили чат-бот з математики для учнів 9-11 класів (рис. 2.1) :

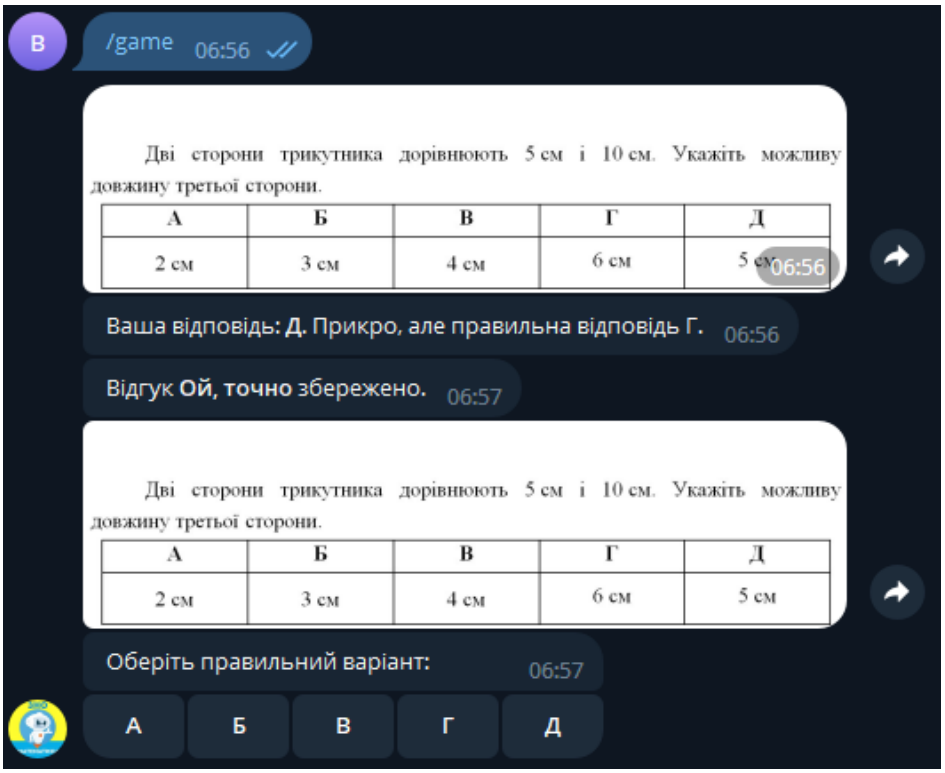

Рисунок 2.1 - Чат-бот з математики для учнів 9-11 класів

Це дуже спрощує користувальницький досвід. І дозволяє учням більш якісно підготуватися до ЗНО.

Телеграм-боти мають безліч очевидних плюсів:

- Доступні 24/7;
- Миттєва відповідь користувачу;

- Зручність у користуванні, спілкування за принципом «питання-відповідь» та текстові завдання під силу давати навіть зовсім недосвідченому користувачеві месенджера;
- Не вимагають встановлення додаткових програм, програм тощо. Все спілкування з ботом ведеться безпосередньо через месенджер;
- Безпека особистих даних – роботи працюють виключно за заданими командами.

Необмежені можливості – віртуального помічника можна запрограмувати на відправку новин, розповідь анекдотів, нагадування важливої інформації, пошук закладів, бронювання столиків у ресторані, замовлення квитків тощо[3].

Навіщо і кому потрібні чат-боти?

Боти дозволяють мінімізувати витрати, пов'язані зі щоденною та однотипною взаємодією з великою кількістю користувачів. Як і в інших сферах бізнесу та виробництва, автоматизація робочого процесу доцільна в тому випадку, якщо завдання та цілі цього процесу можуть бути описані та конкретизовані.

Однак було б невірним розглядати чат-ботів виключно у вузькому значенні – як роботів, які відповідають лише на обмежене коло питань. Сучасні алгоритми вміють взаємодіяти з користувачем і у складніших випадках. Так, наприклад, вони можуть розуміти і відповідати на питання про те саме, але сформульовані по-різному: «де можна отримати товар?», «де найближчий пункт видачі?», «яка адреса магазину?» та ін. Таким чином, чат-боти корисні навіть там, де діалог з клієнтом може бути варіативним і нелінійним.

У тих випадках, коли користувач формулює запит у неясній для робота формі, а також коли його алгоритми не дозволяють дати корисну та однозначну відповідь, програма має можливість переключити співрозмовника на реальну людину. Завдяки цьому вдається досягти

зниження навантаження на операторів у 3 рази, відтинаючи типові питання, які становлять до 70%, в автоматичному режимі[4].

Також є дуже багато варіацій різних Телеграм ботів, таких як:

- Чат-бот поштової служби
- Чат-бот продуктових магазинів
- Чат-бот Арт-заводу
- Чат-бот клініки жіночого здоров'я
- Чат-бот NBA

Взагалі боти в Телеграмі наскільки гнучкі, що скільки б люди не придумали компаній і професій, на кожну із них можливо зробити Телеграм бота, який допоможе в розвитку цієї компанії!

Як боти в телеграм спростили життя мільйонів людей. 51% користувачів очікують від брендів негайної відповіді у будь-який час доби. Тому бізнеси запускають цілодобові служби підтримки та витрачають на це великі гроші. Але більшість питань користувачів є типовими, і на них цілком може відповідати правильно налаштована програма — наприклад, чат-бот. І користувачів це влаштовує: у середньому чат-ботам відповідає 30–40% користувачів, а добре налаштовані чат-боти отримують відповіді у 90% випадків.

За даними SalesForce, 69% користувачів віддають перевагу спілкуванню з ботами, тому що вони можуть отримати відповіді від бренду зі зручною для себе швидкістю. Роботу не складно відповісти 100 користувачам одночасно, а менеджера доведеться чекати. Чат-бот створює відчуття, що бізнес завжди з тобою на зв'язку. І якщо робот вирішує всі проблеми користувача, присутність людини не потрібна.

Якщо боти такі вигідні, чому їх не запровадила більшість бізнесів? Брендам не вистачає прикладів застосування чат-ботів, вони переживають про приватність користувачів і те, що бот даватиме неточні відповіді[6].

Можливості чат-ботів для бізнесу. Додатковий канал продажу. Статистика показує, що люди проводять у месенджерах близько 80% часу використання смартфона. Тому вигідніше длдя бізнесів дивитися у бік чат-ботів, як додаткового каналу продажу.

З погляду продажів, чат-бот схожий на додаток. Найголовніше тут – аудиторія, яка визначає вибір месенджера. Наприклад, з географії — в одних регіонах України популярніший WhatsApp, Viber, в інших — Telegram.

Важливо пам'ятати, що чат-бот це не додаток і не сайт. Тому не потрібно переносити, наприклад, додавання товарів у кошик та оформлення замовлення в бот. Чат-бот - це історія про швидкість, простоту та зручність. Чим простіший діалог, тим вищі продажі.

Чат-бот — хороший спосіб протестувати, чи потрібно бізнесу робити мобільний додаток. Одночасно можна протестувати і маркетинг, і розробку.

Поріг входу в бот у користувача нижче, ніж додаток: Telegram у нього вже стоїть, запустити бот - це два кліки. При цьому сам діалог в роботі можна зробити коротким, скоротити кількість етапів вирви продажів і збільшити швидкість їх проходження. За таких умов чат-бот зможе легко випередити програму. А бізнес зрозуміє, чи має аудиторія запит на розробку повноцінного додатку.

Збір фідбеку користувачів. Робот не може відповісти на запит, до якого сценарій не готовий. Але ця слабкість допомагає збирати зворотний зв'язок від користувачів та покращувати продукт. Можна навчити робота запам'ятовувати всі нестандартні повідомлення від користувачів на відкриті запитання. Потім аналізувати їх і навчати на їх основі нейронну мережу, яка підбирає відповідні за змістом відповіді. У результаті діалог виходить природнішим, а бот намагається привести користувача до угоди[6].

У середині червня 2015 року творець месенджера Telegram Павло Дуров оголосив про запуск платформи для створення чат-ботів, здатних реагувати на команди користувачів, взаємодіючи при цьому із зовнішніми

сервісами. І, схоже, навіть самі розробники не очікували такого успіху, який їхня платформа отримала вже за лічені місяці.

У квітні 2016 року було анонсовано перше масштабне оновлення — Bot Platform 2.0, підготовлене з урахуванням побажань розробників та потреб користувачів, яке значно розширює функціонал ботів. Додано функцію обміну музикою та відео, варіанти швидких відповідей, ідентифікацію розташування користувача, глибоку інтеграцію з іншими службами на основі телефонних номерів (при авторизації) та ряд інших корисних можливостей.

А найцікавіше – Павло Дуров оголосив про створення спеціального фонду, розміром $1 мільйон доларів, в рамках якого творці дійсно цікавих ботів для Telegram отримуватимуть гранти по $25 000 за свій внесок у розвиток екосистеми.

Такий інтерес до цієї теми не залишився непоміченим і з боку інших платформ. Зокрема про свої наміри реалізувати підтримку чат-ботів у Messenger заявив глава Facebook Марк Цукерберг. Найкращого піару для нового тренду на технологічному ринку складно і придумати[8].

2.3. Алгоритм бота в Телеграм.

Алгоритм роботи бот-утиліт досить простий. Повідомлення, команди та запити, надіслані користувачами, надсилаються на програмне забезпечення, запущене на серверах розробників. Посередницький анонімний сервер Telegram обробляє шифрування та здійснює зворотний зв'язок між утилітою та користувачем.

Взаємодія між користувачем та ботом виглядає так:

Користувач бота віддає йому команду -> Бот передає команду на сервер -> Програма на сервері обробляє отриманий від бота запит -> Сервер віддає відповідь боту -> Бот виводить у відповідь екран програми користувача.

І цей цикл повторюється щоразу, коли ви натискаєте на кнопки і взаємодієте з будь-яким телеграм-ботом.

Ви спілкуєтесь із серверами за допомогою простого HTTPS-інтерфейсу, який є спрощеною версією API Telegram. Інакше цей інтерфейс можна назвати програмним каталогом чи бот-алгоритмом.

РОЗДІЛ 3. ІНФОРМАЦІЙНИЙ ОГЛЯД

3.1. Огляд відомих та популярних чат-ботів

Оглянемо популярні та відомі чат-боти в телеграм які зараз найбільш актуальні.

@Hotovyi_do_vsioho_bot — дуже корисний чат-бот. Він дає рекомендації, як діяти в умовах війни: як спланувати переїзд із сім'єю чи захистити будинок; як реагувати на сирену та повідомлення з різних каналів зв'язку; де шукати перевірену інформацію; як убезпечити будинок і зібрати тривожну валізку, аптечку та інше (рис. 3.1):

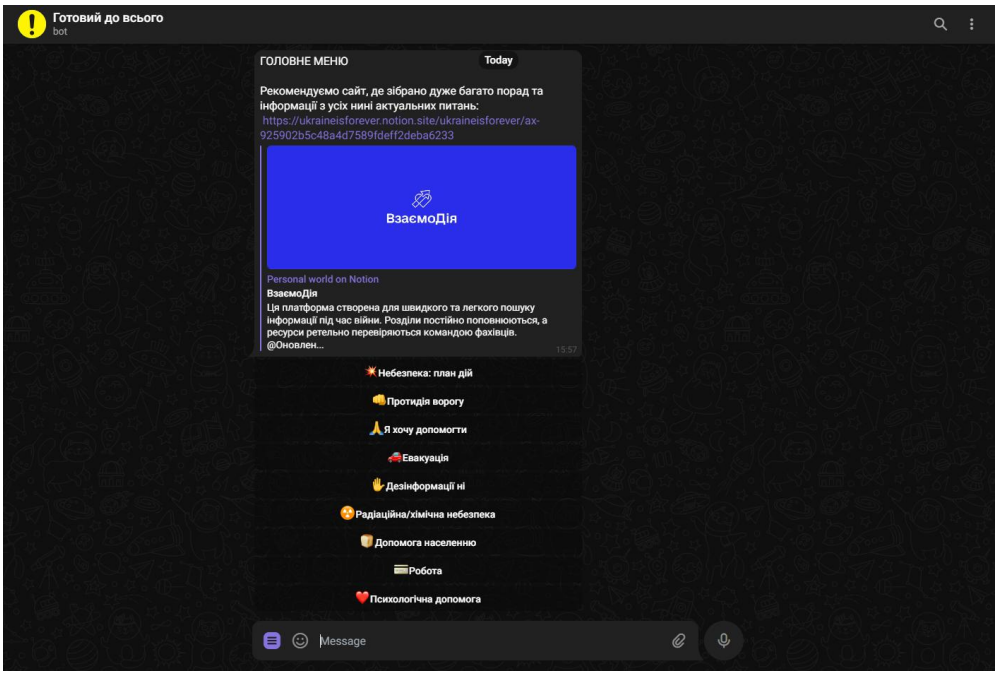

Рисунок 3.1 - Чат-бот @Hotovyi_do_vsioho_bot

@dytyna_ne_sama_bot — бот для допомоги дітям у воєнний час. Передбачений для випадків, якщо батьки загубили дитину, а також якщо бажають тимчасово прихистити дитину у своїй сім'ї. Окрім того, за допомогою бота можна звертатися, якщо ви знайшли дитину без супроводу

дорослих, вам відомі міжнародні організації, які готові прихистити українських дітей, ви маєте інші питання щодо дітей. Бот створили представники дитячого фонду ООН ЮНІСЕФ, Офісу Президента України, Міністерства соціальної політики України (рис.3.2):

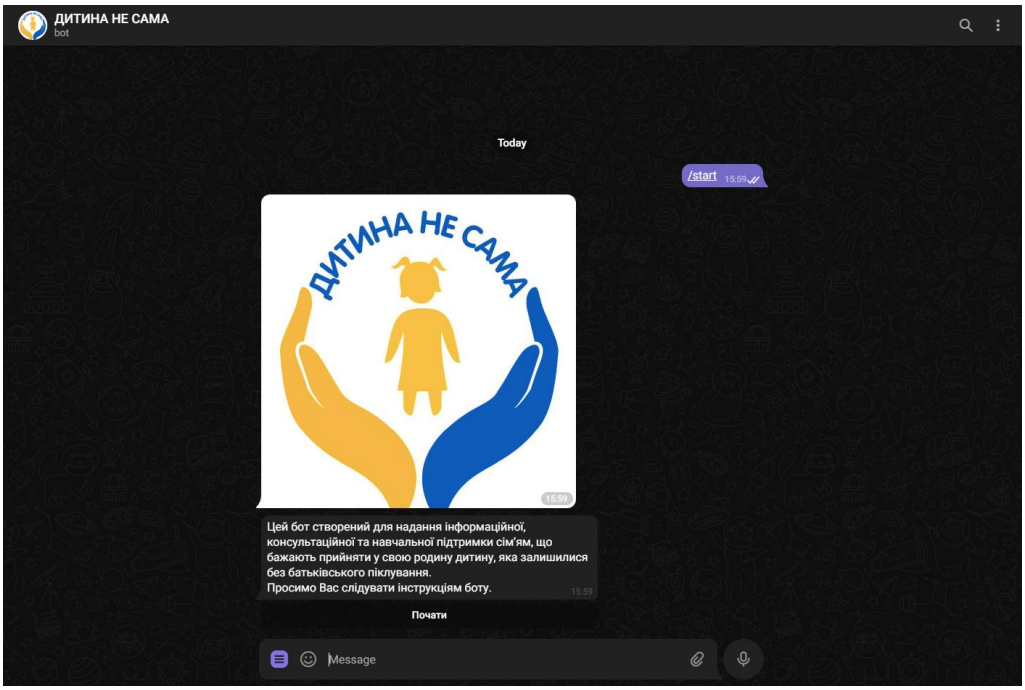

Рисунок 3.2 - Чат-бот @dytyna_ne_sama_bot

Також якщо ви шукаєте розваги з друзями в Telegram, тоцікавий Telegram-бот @GameBot (рис. 3.3) ідеально підходить для вас. Це ігровий бот, у якому ви можете грати в ігри зі своїми друзями. Дуже простий і корисний бот Telegram для розваги з друзями, ви можете легко грати з чатами.

Зараз цей цікавий бот Telegram пропонує три ігри:

- Математична битва: Запитання «Так» або «Ні» на основі математичних тестів.
- Корсари: Ви повинні ухилятися від гарматних яд, щоб перейти на наступний рівень.

- LumberJackBot: ви повинні рубати ліс і врятуватися від гілок, щоб торкнутися.

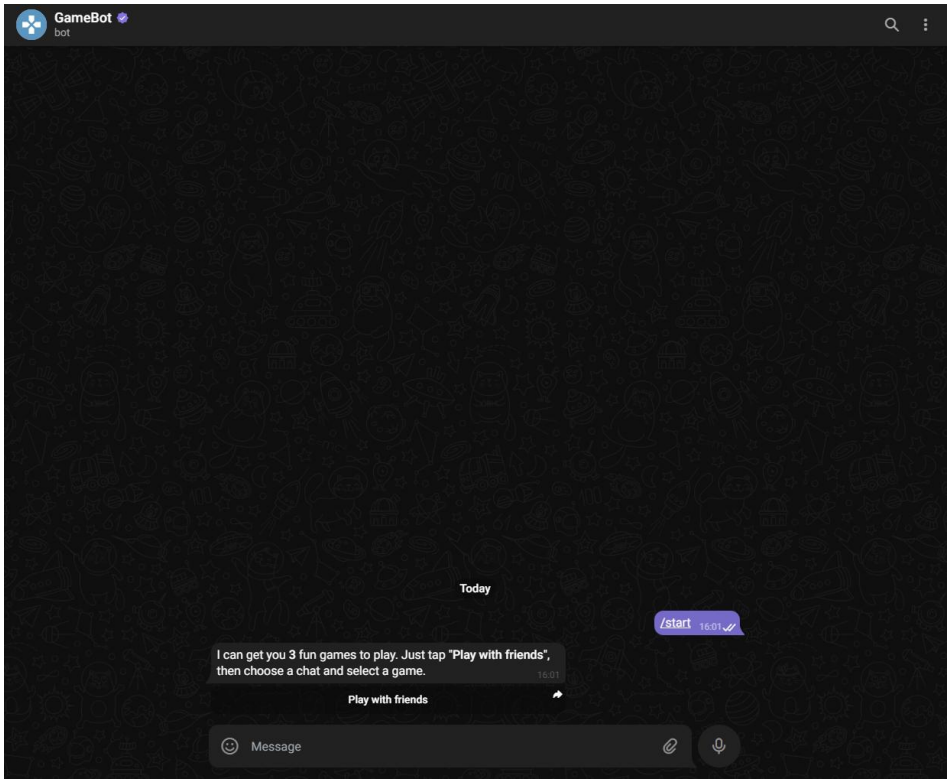

Рисунок 3.3 - Чат-бот @GameBot

3.2. Огляд чат-ботів для навчального процесу

Наразі найвідомішим чат-ботом для навчання в Україні являється @ EducationUABot.

Основне завдання бота - зручна форма інформування про актуальний стан освітнього процесу в Україні і світі. Його можна буде використовувати і після війни, змінюючи тематичне наповнення.

Освітній чат-бот @EducationUaBot допоможе знайти необхідну інформацію щоб повернутися до навчання, де б ви не знаходилися в Україні чи за кордоном. Завдяки Освітньому чат-боту, переміщені учасники освітнього процесу (здобувачі освіти, педагогічні працівники) можуть оперативно знайти інформацію про заклади освіти всіх рівнів у різних

населених пунктах, можливості щодо продовження навчання чи викладання під час війни в кожному регіоні, відновлення особистих документів про освіту, тощо.

Інформація, яку можна знайти в Освітньому чат-боті (рис.3.4):

- Як знайти садочок, школу або коледж за кордоном;
- Як долучитися до онлайн-навчання в Україні;
- Як відновити особисті чи документи про освіту;
- Як продовжувати викладати та працювати під час війни і багато інших корисних матеріалів.

Зручність і практичність використання Освітнього чат-боту підтверджується кількістю авторизованих користувачів - 32825 користувачів, з них 12 133 - шукали інформацію про освіту за кордоном, 11 735 - в Україні.

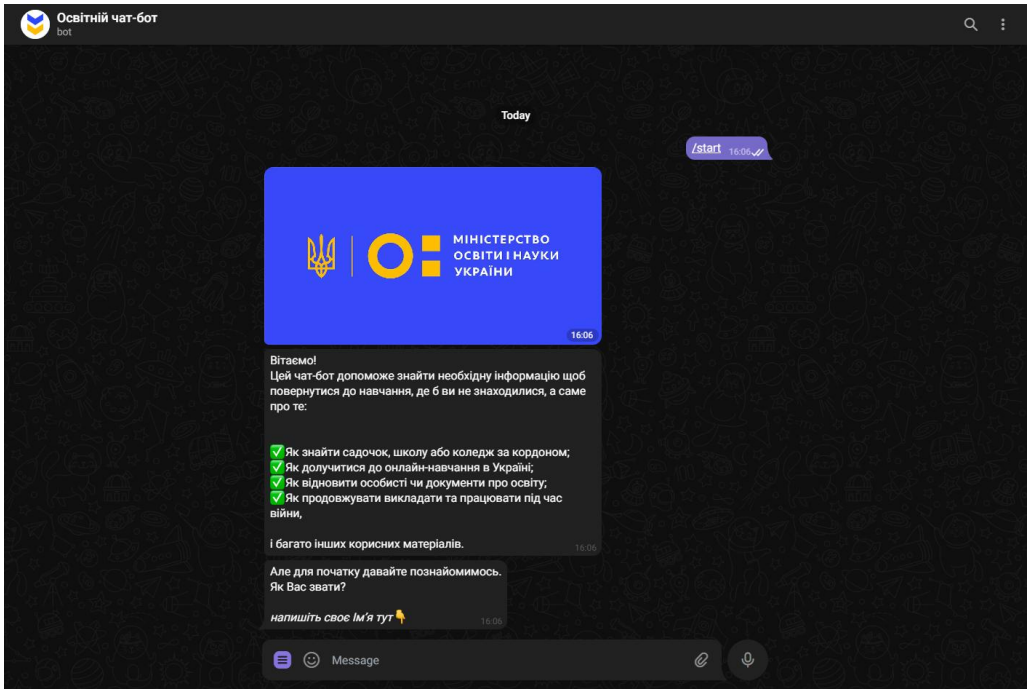

Рисунок 3.4 - Освітній чат-бот

Наявність такого бота ще раз доказує, що боти в телеграм являються дуже зручними і зрозумілими в використанні.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. План розробки Bot-Challenge

Для початку я склав наступний план створення такого бота:

1. Вибрати мову програмування та фреймворк:
    - Вирішити, якою мовою програмування я хочу створити свого чат-бота
    - Вибрати найбільш підходящий фреймворк або бібліотеку.
2. Налаштувати оточення розробки:
    - Встановити вибрану мову програмування та необхідні інструменти розробки.
    - Вивчити інструкції щодо настроювання оточення, надані фреймворком або бібліотекою.
3. Створити нового бота та отримати токен:
    - Знайти робота «@BotFather» в Telegram.
    - Дотримуватися інструкцій «@BotFather» для створення нового бота та отримання унікального токена.
4. Реалізувати функціональність робота:
    - Визначити команди та дії, які мій бот виконуватиме у відповідь на повідомлення від користувачів.
    - Написати код для реалізації функціональності бота, використовуючи вибрану мову програмування та фреймворк.
5. Налаштувати обробники та взаємодію з Telegram API:
    - Використовувати API-ключ і токен вашого бота для налаштування обробників вхідних повідомлень та інших подій.
    - Реалізувати логіку обробки команд та взаємодії з користувачами через Telegram API.
6. Провести тестування та налагодження:

- Надіслати тестові повідомлення та перевірити відповіді бота.
- Виправити можливі проблеми та помилки.

4.2. Розробка Bot-Challenge і бази даних для нього

Вибрати мову програмування та фреймворк:

- Вирішити, якою мовою програмування я хочу створити свого чат-бота.
- Вибрати найбільш підходящий фреймворк або бібліотеку.

Писати я вирішив на Python так як він має величезну поширеність у сфері розробки ботів: Python широко використовується для створення чат-ботів та автоматизації завдань завдяки своїй гнучкості та зручності у використанні. Це означає, що є безліч прикладів, навчальних матеріалів та посібників, які допоможуть вам у розробці та дозволять вам легко знайти допомогу, якщо виникнуть проблеми.

Я вибрав фреймворк aiogram для написання свого чат-бота з кількох причин:

- Зручність використання: aiogram пропонує простий та інтуїтивно зрозумілий інтерфейс, що полегшує розробку чат-бота в Telegram. Він надає безліч зручних функцій та методів, що дозволяють взаємодіяти з API Telegram, обробляти повідомлення, створювати клавіатури та багато іншого.
- Великий набір функціональності: aiogram пропонує широкий набір функціональності, що дозволяє створити повнофункціональний чат-бот. Він підтримує обробку текстових повідомлень, зображень, аудіо та інших медіафайлів, а також надає можливість роботи з клавіатурами, онлайн-режимом, розсилками та багато іншого.
- Підтримка асинхронності: aiogram заснований на асинхронному програмуванні з використанням asyncio, що дозволяє створювати

ефективні та масштабовані чат-боти. Асинхронність дозволяє обробляти кілька запитів одночасно, роблячи бота більш чуйним та швидким.

Налаштувати оточення розробки:

- Встановити вибрану мову програмування та необхідні інструменти розробки.
- Вивчити інструкції з налаштування оточення, надані фреймворком чи бібліотекою.

Я встановив Python, обрану мною мову програмування, з офіційного сайту python.org[9]. Я вибрав останню стабільну версію Python, щоб використовувати всі останні можливості та покращення.

Для керування пакетами та залежностями я використовував інструмент встановлення пакетів pip, який постачається разом із Python. Я оновив pip до останньої версії, щоб гарантувати сумісність із бібліотеками, які я збираюся використовувати.

Для роботи з фреймворком aiogram, я створив віртуальне оточення Python, використовуючи venv інструмент. Віртуальне оточення допомагає ізолювати проект від системної установки Python та дозволяє легко керувати залежностями.

Я активував віртуальне оточення та встановив aiogram, виконавши команду «pip install aiogram». Це завантажило та встановило останню версію фреймворку aiogram у мойому проекті.

Для вивчення інструкцій з настроювання оточення та використання aiogram, я відвідав офіційний сайт aiogram, де знайшов докладну документацію, приклади коду та посібника. Я прочитав основні розділи, що описують основні концепції та функціональність фреймворку, а також вивчив приклади коду для кращого розуміння того, як використовувати aiogram у своєму проекті.

В результаті роботи з налаштування оточення розробки, я встановив Python, оновив інструмент установки пакетів pip, створив і активував віртуальне оточення, встановив фреймворк aiogram і вивчив документацію, необхідну для використання aiogram у розробці мого чат-бота.

Створити нового бота та отримати токен:

- Знайти робота «@BotFather» в Telegram.
- Дотримуватися інструкцій «@BotFather» для створення нового бота та отримання унікального токена.

Я відкрив програму Telegram і в пошуковому рядку знайшов бота з ім'ям «@BotFather».

Після знаходження «@BotFather», я відкрив його чат і дотримувався інструкцій, наданих ботом.

Дотримуючись інструкцій «@BotFather», я створив нового бота, вказавши його ім'я та опис.

Після успішного створення бота, «@BotFather» надав мені унікальний токен для доступу до створеного бота.

Я зберіг отриманий токен, оскільки він використовуватиметься в моєму коді для взаємодії з API Telegram та управління ботом.

В результаті виконання цього пункту плану, я успішно створив новий робот і отримав унікальний токен, який знадобиться для подальшої розробки мого чат-бота.

Реалізувати функціональність робота:

- Визначити команди та дії, які мій бот виконуватиме у відповідь на повідомлення від користувачів.
- Написати код для реалізації функціональності бота, використовуючи вибрану мову програмування та фреймворк.

Для зручної взаємодії користувача з ботом я встановив кілька команд:

/start - Так як з цієї команди користувач починає знайомство з ботом, то тут бот вітає і просить ввести Ім'я користувача, надалі зберігаючи його для відображення користувача в рейтингу.

/game - Команда яка активує тренувальну гру з користувачем, не рахуючи його рейтинг.

/assessment - Команда активує гру вже на оцінювання і рахує всі відповіді користувача записуючи їх.

/help - Команда допомагає користувачеві зрозуміти, як працюють інші команди.

/stats - Команда, що виводить особисту статистику користувача.

/rating - Команда для виведення рейтингу всіх користувачів, що показує кількість балів за правильні відповіді та відсоток правильних відповідей до вирішених завдань.

Далі присано безпосередньо до написання коду.

```
import os
import random
from aiogram import Bot, types
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton
from aiogram.dispatcher import Dispatcher
from aiogram.utils import executor
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup
from WSG import TOKEN
```

Насамперед імпортую необхідні модулі та класи з aiogram.

Імпортую модуль «os» для роботи з файлом, в якому зберігаються картинки із завданнями. Загалом модуль os використовується для роботи з файлами та шляхами в операційній системі, що дозволяє програмі взаємодіяти з файловою системою і виконувати операції над файлами та директоріями.

Імпортую модуль «random», для створення випадковості та різноманітності у грі із завданнями та варіантами відповідей.

Імпортую необхідні модулі та класи з aiogram:

1. «Bot» і «types» з «aiogram»:
   - «Bot» клас надає інтерфейс для роботи з Telegram Bot API. Він використовується для взаємодії з ботом, відправлення та отримання повідомлень, оновлень та інших операцій.
   - «types» модуль містить класи, що відносяться до типів даних, що використовуються в «aiogram», такими як «Message», «CallbackQuery», «ReplyKeyboardMarkup» і т.д. Ці класи представляють різні типи об'єктів в Telegram API, і модуль «types» надає зручні методи для роботи з ними.
2. «ReplyKeyboardMarkup» та «KeyboardButton» з «aiogram.types»:
   - «ReplyKeyboardMarkup» клас представляє клавіатуру з кнопками, яку можна використовувати для створення інтерфейсу користувача в чаті. Він використовується для створення та відправлення користувацьких клавіатур з певними кнопками.
   - «KeyboardButton» клас представляє окрему кнопку на клавіатурі. Він використовується для створення та налаштування кнопок, які будуть відображатися в»ReplyKeyboardMarkup».
3. «Dispatcher» з «aiogram.dispatcher»:
   - «Dispatcher» клас надає механізм для обробки вхідних оновлень та маршрутизації їх до відповідних обробників. Він використовується для налаштування та реєстрації обробників команд, повідомлень, «коллбеків» та інших подій, які можуть відбуватися в роботі.
4. «executor» з «aiogram.utils»:
   - «executor» модуль містить функцію «start_polling()», яка запускає цикл отримання оновлень від Telegram Bot API та передачу їх у

«Dispatcher» для обробки. Вона використовується для запуску бота та початку прийому та обробки повідомлень.

5. «MemoryStorage» з «aiogram.contrib.fsm_storage.memory»:
   - `MemoryStorage` клас надає просте сховище станів (state storage) у пам'яті для використання з «aiogram». Він використовується для збереження станів чату та даних користувача між оновленнями. Цей код, швидше за все, використовується для реалізації кінцевого автомата (FSM) і збереження станів для кожного користувача.
6. «FSMContext» з «aiogram.dispatcher»:
   - «FSMContext» клас представляє контекст кінцевого автомата (FSM) для поточного користувача. Він використовується для доступу та управління станом та даними користувача в рамках кінцевого автомата.
7. «State» та «StatesGroup» з «aiogram.dispatcher.filters.state»:
   - «State» клас представляє окремий стан у кінцевому автоматі (FSM). Він використовується визначення станів та його зв'язку з певними обробниками.
   - «StatesGroup» клас представляє групу станів у кінцевому автоматі (FSM). Він використовується для об'єднання пов'язаних станів групи для зручнішої роботи з ними.

Ці модулі та класи використовуються для налаштування та роботи з ботом, обробки оновлень, управління станами користувача та створення інтерфейсу користувача за допомогою клавіатур та кнопок.

Далі зроблені налаштування бота та створення диспетчера (dispatcher) з використанням бібліотеки aiogram.

```
bot = Bot(token=TOKEN)
storage = MemoryStorage()
dp = Dispatcher(bot, storage=storage)
```

1. «Bot(token=TOKEN)»:
   - Тут створюється екземпляр класу «Bot» з бібліотеки «aiogram». Конструктор класу приймає один аргумент «token», який представляє токен вашого бота, виданий BotFather Telegram. Токен використовується для автентифікації бота та взаємодії з Telegram Bot API.
2. «MemoryStorage()»:
   - Тут створюється екземпляр класу «MemoryStorage» з модуля «aiogram.contrib.fsm_storage.memory». «MemoryStorage» надає просте сховище станів у пам'яті для використання з «aiogram».
3. «Dispatcher(bot, storage=storage)»:
   - Тут створюється екземпляр класу «Dispatcher» із модуля «aiogram.dispatcher». Конструктор класу приймає два аргументи: «bot» та «storage».
   - Аргумент «bot» представляє екземпляр класу «Bot» і вказує, який бот використовуватиметься для відправлення та отримання повідомлень та оновлень.
   - Аргумент «storage» представляє сховище станів (state storage), яке буде використовуватись для збереження станів та даних користувача. Тут передається створений раніше екземпляр «MemoryStorage».

Створений диспетчер («dp») використовується для реєстрації та налаштування обробників команд, повідомлень, «коллбеків» та інших подій, а також для запуску обробки оновлень за допомогою функції «executor.start_polling()».

Далі створюю в директорії файл BData, в ньому я зберігатиму в змінній “IMAGES_DIR” шлях до файлу із завданнями.

Тут же створюю словник із правильними відповідями у змінній “CORRECT_ANSWERS”

```
CORRECT_ANSWERS = {
    '1B5.jpg': 'B',
    '2B4.jpg': 'B',
```

Також імпортую ці змінні до main

Тепер потрібно отримати список картинок нашого файлу BData скористаємося функцією модуля «os». І перемішаємо список для цікавішої гри.

```
all_images = os.listdir(IMAGES_DIR)
random.shuffle(all_images)
```

Функція os.listdir() із модуля os використовується для отримання списку файлів та папок у зазначеній директорії IMAGES_DIR. IMAGES_DIR представляє шлях до директорії, в якій знаходяться зображення. В результаті виконання os.listdir (IMAGES_DIR) буде повернено список імен файлів у зазначеній директорії.

Функція random.shuffle() з модуля random використовується для перемішування елементів у списку all_images у випадковому порядку. Після виконання цього рядка коду елементи у списку all_images будуть перемішані у випадковому порядку.

Створимо функцію get_random_image(), яка повертає випадкову картинку.

Далі я зіткнувся з проблемою, в картинках із завданнями є кілька варіантів відповідей та їх кількість для різних картинок відрізняється. Я вирішив в імені файлу картинки додати цифру яка визначає кількість варіантів відповідей. Наприклад, якщо в картинці два варіанти відповіді A і B, то файл буде називатися так image2.jpeg.

Для отримання цієї цифри я створив наступну функцію

```
def get_num_of_options(image_path):
    filename = os.path.basename(image_path)
    num_str = filename.split('.')[0][-1]
    return int(num_str)
```

У цьому коді функція «get_num_of_options()» виконує такі дії:

1. «filename = os.path.basename(image_path)»: Цей рядок використовує функцію «os.path.basename()» для отримання імені файлу з повного шляху «image_path». Наприклад, якщо «image_path» містить значення «path/to/image.jpg», то «filename» буде містити значення «image.jpg».

2. «num_str = filename.split('.')[0][-1]»: Цей рядок розділяє ім'я файлу «filename» по точці і вибирає першу частину (до першої точки). Потім, за допомогою індексу «[-1]», вилучається останній символ цієї частини. Наприклад, якщо «filename» містить значення «image_3.jpg», то «num_str» буде містити значення «3».

3. «return int(num_str)»: Цей рядок перетворює рядкове значення «num_str» у ціле число за допомогою функції «int()» і повертає його.

Таким чином, функція get_num_of_options() використовується для отримання числа з імені файлу image_path. Вона отримує останній символ першої частини імені файлу, розділеного по точці, і повертає його у вигляді цілого числа.

Тепер настав час написати обробник команди старт.

Але спочатку напишемо клас, у якому зберігатимуться стани, такі як ім'я користувача, стан гри на рейтинг, ігри без рейтингу та стан рейтингу.

```
class GameStates(StatesGroup):
    GAME = State()
    RATING = State()
    NAME = State()
    NO_RATING = State()
```

Даний клас «GameStates» визначає стани чату за допомогою StatesGroup з aiogram.dispatcher.filters.state. Розглянемо кожний рядок докладніше:

1. «GAME = State()»: Цей рядок визначає стан «GAME» всередині класу «GameStates». Стан GAME може використовуватися для відстеження стану гри, наприклад, коли користувач знаходиться в режимі гри.

2. «RATING = State()»: Цей рядок визначає стан «RATING» усередині класу «GameStates». Стан `RATING` може використовуватись для відстеження стану рейтингу або оцінки, наприклад, коли користувач хоче оцінити гру або переглянути рейтинг інших гравців.

3. «NAME = State()»: Цей рядок визначає стан «NAME» всередині класу «GameStates». Стан «NAME» може використовуватися для відстеження стану запиту імені та прізвища користувача, наприклад, коли бот запитує у користувача його дані.

4. «NO_RATING = State()»: Цей рядок визначає стан «NO_RATING» усередині класу «GameStates». Стан «NO_RATING» може використовуватися для відстеження стану, коли користувач не має рейтингу або оцінки, наприклад, якщо він не оцінив гру.

Клас «GameStates» дозволяє визначити різні стани чату, які можуть бути використані для керування потоком бота та виконання відповідних дій залежно від поточного стану.

Напишемо обробник команди старт.

```
@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    await message.answer("Вітання! Перш ніж ми почнемо, зареєструйся. Напиши мені своє ім'я та прізвище")
    await GameStates.NAME.set()
```

У цьому хендлері відбувається обробка команди «/start» від користувача. Давайте розглянемо кожний рядок докладніше:

1. «@dp.message_handler(commands=['start'])»: Цей рядок вказує, що даний хендлер повинен обробляти лише повідомлення-команди з текстом

«/start». «@dp.message_handler» є декоратором, який реєструє функцію «start()» як обробник повідомлень.

2. «async def start(message: types.Message):» Цей рядок визначає асинхронну функцію «start()», яка приймає один аргумент «message» типу «types.Message». «types.Message» представляє повідомлення, отримане від користувача.

3. «await message.answer("Вітання! Перш ніж ми почнемо, зареєструйся. Напиши мені своє ім'я та прізвище")»: Цей рядок надсилає повідомлення у відповідь користувачу з текстом «Вітання! Перш ніж ми почнемо, зареєструйся. Напиши мені своє ім'я та прізвище". Функція «message.answer()» використовується для надсилання повідомлень від бота.

4. «await GameStates.NAME.set()»: Цей рядок використовується для встановлення стану чату за допомогою стану FSM (Finite State Machine). «GameStates.NAME» представляє стан чату, і виклик методу «.set()» встановлює поточний стан чату в «GameStates.NAME». FSM дозволяє боту відстежувати стани чату та виконувати відповідні дії залежно від поточного стану.

Таким чином, даний хендлер обробляє команду «/start», відправляє вітальне повідомлення користувачу та встановлює стан чату в «GameStates.NAME», щоб просити користувача ввести своє ім'я та прізвище.

Тепер потрібно обробити та зберегти Ім'я користувача, для цього створюємо новий хендлер.

```python
@dp.message_handler(state=GameStates.NAME)
async def process_name(message: types.Message, state: FSMContext):
    name = message.text
    user_id = message.from_user.id
    if user_id not in users_data:
        users_data[user_id] = UserData(name=name)
    else:
        users_data[user_id].name = name
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton('/help'))
    await message.answer(f"Радий знайомству, {name}! Ознайомся з командами, натисни /help!", reply_markup=keyboard)
    await state.finish()
```

Цей хендлер обробляє повідомлення, коли користувач перебуває у стані `GameStates.NAME`. Розглянемо кожний рядок докладніше:

1. «@dp.message_handler(state=GameStates.NAME)»: Цей рядок вказує, що цей хендлер повинен бути виконаний тільки в стані «GameStates.NAME». Тобто, він буде викликатись лише тоді, коли користувач надсилає повідомлення, перебуваючи у стані запиту імені.

2. «async def process_name(message: types.Message, state: FSMContext)»: Цей рядок визначає асинхронну функцію «process_name», яка приймає два аргументи: «message» (тип «types.Message2) - об'єкт повідомлення користувача, та «state» (тип «FSMContext2) - об'єкт контексту стану.

3. «name = message.text»: Цей рядок отримує текст повідомлення, надісланого користувачем, і присвоює його змінній «name».

4. «user_id = message.from_user.id»: Цей рядок отримує ідентифікатор користувача, що надіслав повідомлення, з об'єкта «message» і надає його змінній «user_id».

5. «if user_id not in users_data: ... else: ...»: Цей блок перевіряє, чи є ідентифікатор користувача у словнику «users_data». Якщо ідентифікатор відсутній, створюється новий об'єкт «UserData» з ім'ям користувача і додається до словника «users_data». В іншому випадку, якщо ідентифікатор вже існує, ім'я користувача в об'єкті UserData оновлюється.

6. «keyboard = ReplyKeyboardMarkup(resize_keyboard=True)»: Цей рядок створює новий об'єкт клавіатури «ReplyKeyboardMarkup», який міститиме кнопку «/help».

7. «keyboard.add(KeyboardButton(«/help»))»: Цей рядок додає кнопку «/help» у клавіатуру «keyboard».

8. «await message.answer»(f«Радій знайомству, {name}!») клавіатуру «keyboard» у повідомлення.

9. «await state.finish()»: Цей рядок завершує поточний стан користувача та скидає його назад у вихідний стан.

Таким чином, даний хендлер обробляє повідомлення, що містить ім'я користувача, зберігає його в об'єкті UserData, відправляє вітальне повідомлення з кнопкою «/help» і завершує стан запиту імені, переводячи користувача в наступний стан.

Команду help обробимо пізніше, зараз напишемо основну команду assessment та функцію до неї.

Пишемо обробник команди /assessment.

```python
@dp.message_handler(commands=['assessment'])
async def start_game(message: types.Message):
    user_id = message.chat.id
    if user_id not in users_data:
        users_data[user_id] = UserData(name="")
    else:
        users_data[user_id].reset_rating()
    users_data[user_id].images = all_images.copy()
    users_data[user_id].task_count = 0
    await send_next_question(user_id)
```

Цей хендлер обробляє команду «/assessment» від користувача. Давайте розглянемо кожний рядок докладніше:

1. «@dp.message_handler(commands=['assessment'])»: Цей рядок вказує, що цей хендлер повинен бути виконаний тільки при отриманні команди «/assessment». Тобто, він буде викликатись лише тоді, коли користувач надсилає це конкретне повідомлення.

2. «async def start_game(message: types.Message)»: Цей рядок визначає асинхронну функцію «start_game», яка приймає один аргумент «message» (тип «types.Message») - об'єкт повідомлення користувача.

3. «user_id = message.chat.id»: Цей рядок отримує ідентифікатор чату (або користувача) з об'єкта «message» і надає його змінній «user_id».

4. «if user_id not in users_data: ... else: ...»: Цей блок перевіряє, чи є ідентифікатор користувача у словнику «users_data». Якщо ідентифікатор відсутній, створюється новий об'єкт UserData з порожнім ім'ям і додається до

словника users_data. В іншому випадку, якщо ідентифікатор вже існує, скидається рейтинг користувача за допомогою методу «reset_rating()» об'єкта «UserData».

5. «users_data[user_id].images = all_images.copy()»: Цей рядок надає користувачеві копію списку всіх зображень («all_images») зі змінної «users_data». Таким чином, кожному користувачеві надається копія списку зображень для оцінки.

6. 2users_data[user_id].task_count = 0»: Цей рядок встановлює лічильник завдань користувача в нуль. Цей лічильник використовуватиметься для відстеження кількості завдань, виконаних користувачем.

7. «await send_next_question(user_id)»: Цей рядок викликає асинхронну функцію «send_next_question2 та передає їй ідентифікатор користувача («user_id»). Функція «send_next_question» надсилатиме користувачеві наступне питання для оцінки.

Таким чином, даний хендлер обробляє команду «/assessment», ініціалізує або скидає рейтинг користувача, створює копію списку зображень для користувача, скидає лічильник завдань користувача та надсилає наступне питання для оцінки користувача.

Напишемо функцію send_next_question, в цій функції відправлятимемо рандомну картинку із завданням, формуватимемо інлайн клавіатуру з варіантами відповідей. також у цій функції будемо вести підрахунок у відправлених завдань щоб гра була не вічною.

```python
async def send_next_question(chat_id):
    user_data = users_data[chat_id]
    if user_data.task_count >= 20:
        await finish_game(chat_id)
        return

    image_path = os.path.join(IMAGES_DIR, user_data.images.pop(0))

    # Відправляємо зображення із завданням
    with open(image_path, 'rb') as photo:
        message = await bot.send_photo(chat_id, photo)

    # Отримуємо кількість варіантів відповідей з імені файлу
    num_of_options = get_num_of_options(image_path)

    # Зберігаємо правильну відповідь, пов'язану з надісланою картинкою
    correct_answer = CORRECT_ANSWERS.get(os.path.basename(image_path))

    # Зберігаємо правильну відповідь та кількість варіантів відповідей у стан машини
    await GameStates.GAME.set()
    await dp.current_state(chat=chat_id).update_data(correct_answer=correct_answer, num_of_options=num_of_options)
```

```python
# Формуємо варіанти відповідей у вигляді інлайн-кнопок
keyboard = types.InlineKeyboardMarkup(row_width=num_of_options)  # Вказуємо кількість кнопок у рядку
options = ['A', 'B', 'C', 'D', 'E'][:num_of_options]  # Допустимі варіанти відповідей
keyboard.add(*[types.InlineKeyboardButton(text=option, callback_data=option) for option in options])  # Додаємо всі кнопки одночасно
keyboardst = ReplyKeyboardMarkup(resize_keyboard=True)
keyboardst.add(KeyboardButton('/finish'))
user_data.task_count += 1
# Зберігаємо ідентифікатор повідомлення з інлайн-клавіатурою для подальшого видалення
if chat_id not in users_data:
    users_data[chat_id] = UserData()
users_data[chat_id].last_message_id = message.message_id

await bot.send_message(chat_id, 'Виберіть правильний варіант:', reply_markup=keyboard)
await bot.send_message(chat_id, 'Щоб вийти з гри натисніть /finish', reply_markup=keyboardst)
```

Ця функція «send_next_question`» надсилає наступне питання для оцінки користувача. Розберемо кожний рядок докладніше:

1. «user_data = users_data[chat_id]»: Отримуємо об'єкт «UserData» зі словника «users_data» за заданим ідентифікатором «chat_id».

2. «if user_data.task_count >= 20: ...»: Перевіряємо, якщо кількість виконаних завдань («task_count») користувача досягла або перевищила 20, то викликаємо функцію «finish_game» для завершення гри та повертаємось з функції.

3. «image_path = os.path.join(IMAGES_DIR, user_data.images.pop(0))»: Формуємо шлях до наступного зображення для оцінки, використовуючи перший елемент зі списку зображень користувача («user_data.images»). Метод «pop(0)» витягує та видаляє перший елемент зі списку.

4. «with open(image_path, «rb») as photo: ...»: Відкриваємо зображення на шляху «image_path» в режимі читання бінарного файлу і зберігаємо його в змінній «photo».

5. «message = await bot.send_photo(chat_id, photo)»: Відправляємо повідомлення з фотографією («photo») користувачеві за допомогою методу «send_photo» бота. Повертається об'єкт повідомлення, який зберігається у змінній «message».

6. «num_of_options = get_num_of_options(image_path)»: Отримуємо кількість варіантів відповідей з імені файлу за допомогою функції «get_num_of_options». Результат зберігається у змінній «num_of_options».

7.«correct_answer=CORRECT_ANSWERS.get(os.path.basename(image_path))»: Отримуємо правильну відповідь для відправленої картинки, використовуючи ім'я файлу («os.path.basename(image_path)») як ключ у словнику «CORRECT_ANSWERS» . Результат зберігається у змінній «correct_answer».

8. «await GameStates.GAME.set()»: Встановлюємо стан машини «GameStates» у стан «GAME» за допомогою методу «set()».

9.»awaitdp.current_state(chat=chat_id).update_data(correct_answer=correct_answer, num_of_options=num_of_options)»: Оновлюємо дані поточного стану машини («dp.current_state(chat=chat_id)») з новими значеннями «corre num_of_options» за допомогою методу «update_data()».

10.«keyboard=types.InlineKeyboardMarkup(row_width=num_of_options)»: Створюємо об'єкт клавіатури з інлайн-кнопками («InlineKeyboardMarkup») і вказуємо ширину рядка кнопок («row_width») рівну «num_of_options».

11. «options = [«A», «B», «C», «D», «E»][:num_of_options]»: Створюємо список допустимих варіантів відповідей («options») у вигляді символів «A», «B», «C», «D», «E» і обмежуємо його довжину до «num_of_options».

12. «keyboard.add(*[types.InlineKeyboardButton(text=option, callback_data=option) for option in options])»: Додаємо всі інлайн-кнопки в

клавіатуру одночасно за допомогою методу «add()». Кожна кнопка створюється з текстом зі списку «options» та відповідним значенням «callback_data».

13. «keyboardst = ReplyKeyboardMarkup(resize_keyboard=True)»: Створюємо об'єкт клавіатури зі звичайними кнопками («ReplyKeyboardMarkup») і вказуємо параметр «resize_keyboard=True», щоб клавіатура автоматично масштабувалася під розмір екрану.

14. «user_data.task_count += 1»: Збільшуємо значення лічильника завдань користувача на 1.

15. «if chat_id not in users_data: ...»: Перевіряємо, якщо ідентифікатор чату («chat_id») відсутній у словнику «users_data», то створюємо новий об'єкт «UserData».

16. «users_data[chat_id].last_message_id = message.message_id»: Зберігаємо ідентифікатор повідомлення з інлайн-клавіатурою («message.message_id») у полі «last_message_id» об'єкта «UserData» для подальшого видалення.

17. «await bot.send_message(chat_id, «Виберіть правильний варіант:», reply_markup=keyboard)»: Надсилаємо повідомлення з текстом «Виберіть правильний варіант:» та вкладеною клавіатурою «keyboard» користувачу за допомогою методу «send_message» бота.

18. «await bot.send_message(chat_id, «Щоб вийти з гри натисніть /finish», reply_markup=keyboardst)»: Відправляємо повідомлення з текстом «Щоб вийти з гри натисніть /finish» та вкладеною клавіатурою «keyboardst» користувачу за допомогою методу «send_message» бота.

Таким чином, функція «send_next_question» надсилає користувачеві наступне питання для оцінки, включаючи зображення із завданням, варіанти відповідей у вигляді інлайн-кнопок та звичайну клавіатуру для завершення гри.

Так наступним кроком буде перевірка відповіді користувача і тому підрахунок балів за правильну відповідь потрібно написати наступний клас:

```
class UserData:
    def __init__(self, correct_answers=0, total_answers=0, name=""):
        self.correct_answers = correct_answers
        self.total_answers = total_answers
        self.name = name

    def reset_rating(self):
        self.correct_answers = 0
        self.total_answers = 0
```

Клас UserData представляє дані користувача, пов'язані з його оцінкою та ім'ям. Розберемо кожен метод докладніше:

1. «__init__(self, correct_answers=0, total_answers=0, name=«»)»: Конструктор класу, який ініціалізує об'єкт «UserData» із заданими значеннями. Параметри «correct_answers», «total_answers» та «name» мають значення за замовчуванням і можуть бути передані під час створення об'єкта. При створенні об'єкта UserData ці параметри присвоюються відповідним полям об'єкта.

2. «reset_rating(self)»: Метод «reset_rating» скидає рейтинг користувача, встановлюючи значення «correct_answers» та «total_answers» в 0. Він викликається, коли користувач починає нову гру або хоче скинути свої результати.

Таким чином, клас «UserData» надає методи для управління даними користувача, включаючи ініціалізацію об'єкта з певними значеннями та скидання рейтингу користувача. Цей клас використовується для зберігання інформації про кожного користувача, включаючи кількість правильних та загальних відповідей, а також ім'я користувача.

Тепер візьмемося обробкою відповідей від функції «send_next_question».

```python
@dp.callback_query_handler(lambda c: True, state=GameStates.GAME)
async def process_callback(callback_query: types.CallbackQuery, state: FSMContext):
    await bot.answer_callback_query(callback_query.id)

    selected_option = callback_query.data

    # Отримуємо правильну відповідь та кількість варіантів відповідей зі стану машини
    data = await state.get_data()
    correct_answer = data.get('correct_answer')
    num_of_options = data.get('num_of_options')

    if selected_option == correct_answer:
        await bot.send_message(callback_query.from_user.id, f'Ваша відповідь <b>"{selected_option}"</b>:\n✅Правильна відповідь✅!!!', parse_mode='HTML')
        user_id = callback_query.from_user.id
        if user_id in users_data:
            users_data[user_id].correct_answers += 1
        else:
            users_data[user_id] = UserData(correct_answers=1, total_answers=1)
    else:
        await bot.send_message(callback_query.from_user.id, f'Ваша відповідь <b>"{selected_option}"</b>:\n❌Неправильна відповідь!❌', parse_mode='HTML')
        user_id = callback_query.from_user.id
        if user_id not in users_data:
            users_data[user_id] = UserData(correct_answers=0, total_answers=1)
        else:
            users_data[user_id].total_answers += 1
    await bot.delete_message(callback_query.message.chat.id, callback_query.message.message_id)
    await send_next_question(callback_query.from_user.id)
```

Цей хендлер обробляє коллбек-запити, які генеруються при натисканні на кнопки варіантів відповідей під час гри. Розглянемо цей хендлер поетапно:

1. «@dp.callback_query_handler(lambda c: True, state=GameStates.GAME)»: Цей декоратор вказує, що функція «process_callback» оброблятиме коллбек-запити. Він застосовується до функції «process_callback2, щоб вказати, що вона буде викликатися за будь-якого коллбек-запиту в стані «GameStates.GAME».

2. «await bot.answer_callback_query(callback_query.id)»: Функція «answer_callback_query» використовується для надсилання підтвердження обробки колббек-запиту. В даному випадку вона викликається для поточного коллбек-запиту «callback_query», щоб бот відправив відповідь клієнту.

3. «selected_option = callback_query.data»: Отримуємо обраний користувачем варіант відповіді з «callback_query.data». Значення «data» в коллбек-запиті містить інформацію про вибрану кнопку.

4. «data = await state.get_data()»: Використовуючи «state.get_data()», отримуємо дані, збережені в стані машини (FSM). В даному випадку ми отримуємо правильну відповідь («correct_answer») та кількість варіантів відповідей («num_of_options»).

5. Перевіряємо обраний варіант відповіді з правильною відповіддю:

- Якщо вибраний варіант збігається з правильною відповіддю, надсилаємо користувачеві повідомлення про правильну відповідь та збільшуємо кількість правильних відповідей («correct_answers») для даного користувача в об'єкті «users_data».
- Якщо вибраний варіант не збігається з правильною відповіддю, відправляємо користувачеві повідомлення про неправильну відповідь та збільшуємо загальну кількість відповідей («total_answers») для даного користувача в об'єкті «users_data».

6. «await bot.delete_message(callback_query.message.chat.id, callback_query.message.message_id)»: Вилучаємо повідомлення з варіантами відповідей, яке було надіслано раніше, щоб оновити екран із новим питанням.

7. «await send_next_question(callback_query.from_user.id)»: Викликаємо функцію «send_next_question», щоб надіслати наступне запитання користувачеві.

Таким чином, даний хендлер обробляє вибір користувачем варіанта відповіді, порівнює його з правильною відповіддю, оновлює статистику відповідей користувача в об'єкті «users_data» та надсилає наступне питання.

Далі напишемо команду /game і всі необхідні функції для неї зробимо абсолютно такими ж як і для команди /assessment, відмінність буде лише в тому, що ми тут не будемо вважати рейтинг користувача, і гра може тривати нескінченно, поки користувач сам не натисне стоп

Напишемо команду /finish для переривання гри на рейтинг та /stop для зупинки тренувальної гри.

Для початку напишемо функцію яка у на викликається у функції «send_next_question», коли користувач вирішує всі 20 задач.

```
async def finish_game(chat_id):
    user_data = users_data[chat_id]
    # Очищуємо лічильник завдань
    user_data.task_count = 0
    # Очищаємо список фотографій
    user_data.images = []
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton('/finish'))
    await bot.send_message(chat_id, 'Ви вирішили 20 завдань. Щоб вийти з режиму, натисніть /finish', reply_markup=keyboard)
```

Функція «finish_game» використовується для завершення гри та надсилання повідомлення користувачу про завершення гри. Розглянемо цю функцію:

1. «user_data = users_data[chat_id]»: Отримуємо дані користувача по ідентифікатору чату («chat_id») з об'єкта «users_data». Дані користувача зберігаються в екземплярі класу UserData.

2. «user_data.task_count = 0»: Обнулюємо лічильник завдань («task_count») в об'єкті «user_data», щоб гра почала рахувати завдання заново.

3. «user_data.images=[]»: Очищаємо список фотографій («images») в об'єкті «user_data». Таким чином, список фотографій буде порожнім для наступної гри.

4. «keyboard = ReplyKeyboardMarkup(resize_keyboard=True)»: Створюємо екземпляр клавіатури з однією кнопкою.

5. «keyboard.add(KeyboardButton('/finish'))»: Додаємо кнопку «/finish» у створену клавіатуру.

6. «await bot.send_message(chat_id, «Ви вирішили 20 завдань. Щоб вийти з режиму, натисніть /finish», reply_markup=keyboard)»: Відправляємо користувачеві повідомлення про завершення гри. У повідомленні вказується, що користувач завершив 20 завдань і пропонується натиснути кнопку «/finish» для виходу з режиму гри. Клавіатура «keyboard» додається до повідомлення для зручності користувача.

Таким чином, функція «finish_game» оновлює дані користувача, очищає список фотографій і відправляє повідомлення з кнопкою для завершення гри.

Напишемо обробник команди /finish.

```python
@dp.message_handler(commands=['finish'], state=GameStates.GAME)
async def stop_game(message: types.Message, state: FSMContext):
    user_id = message.from_user.id
    if user_id in users_data:
        await state.finish()
        keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
        keyboard.add(KeyboardButton('/rating'))
        keyboard.add(KeyboardButton('/help'))
        await message.answer('Ви вийшли з режиму гри, натисніть /rating, щоб переглянути загальну статистику', reply_markup=keyboard)
        await show_user_rating(user_id)
    else:
        await message.answer("Ви ще не розпочали гру.")
```

Хендлер «stop_game» використовується для зупинки гри за командою «/finish» та виведення користувачеві відповідного повідомлення. Розберемо цей хендлер:

1. «user_id = message.from_user.id»: Отримуємо ідентифікатор користувача («user_id») з об'єкта повідомлення «message». Цей ідентифікатор використовується для перевірки наявності даних користувача.

2. «if user_id in users_data:»: Перевіряємо, чи є дані користувача у словнику «users_data».

3. «await state.finish()»: Завершуємо стан машини (FSM) для даного користувача за допомогою методу «finish()». Це дозволяє очистити стан та підготувати машину до нових станів.

4. «keyboard = ReplyKeyboardMarkup(resize_keyboard=True)»: Створюємо екземпляр клавіатури зі змінним розміром.

5. «keyboard.add(KeyboardButton(«/rating»))»: Додаємо кнопку «/rating» у створену клавіатуру.

6. «keyboard.add(KeyboardButton(«/help»))»: Додаємо кнопку «/help» у створену клавіатуру.

7. «await message.answer(«Ви вийшли з режиму гри, натисніть /rating, щоб переглянути загальну статистику», reply_markup=keyboard)»: Відправляємо користувачеві повідомлення про вихід з режиму гри та пропонуємо натиснути кнопку «/rating» для перегляду загальної статистики. Клавіатура «keyboard» додається до повідомлення для зручності користувача.

8. «await show_user_rating(user_id)»: Викликаємо функцію «show_user_rating(user_id)», яка відображає рейтинг користувача.

Таким чином, хендлер «stop_game» зупиняє гру для користувача, виводить повідомлення про вихід із режиму гри та пропонує користувачеві подивитися свій рейтинг.

Пишемо такий же обробник для тренувальної гри.

```
@dp.message_handler(commands=['stop'], state=GameStates.NO_RATING)
async def stop_game(message: types.Message, state: FSMContext):
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton('/assessment'))
    keyboard.add(KeyboardButton('/help'))
    await message.answer('Ви закінчили гру. Натисніть /assessment, щоб брати участь у грі на рейтинг', reply_markup=keyboard)
    await state.finish()
```

Напишемо функцію для показу особистої статистики.

```
async def show_user_rating(user_id):
    user_data = users_data[user_id]
    if user_data.correct_answers + user_data.total_answers != 0:
        percentage = (user_data.correct_answers / (user_data.correct_answers + user_data.total_answers)) * 100
    else:
        percentage = 0
    rating_text = f"Ваш рейтинг, {user_data.name}, Ваш ID {user_id}:\n\n"
    rating_text += f"Правильно вирішено {user_data.correct_answers} з {user_data.correct_answers + user_data.total_answers} завдань. Відсоток правильних відповідей:{percentage:.2f}%\n"
    rating_text += f"\n Якщо хочеш змінити ім'я, введи команду /start"
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton('/rating'))
    keyboard.add(KeyboardButton('/help'))
    keyboard.add(KeyboardButton('/start'))
    await bot.send_message(user_id, rating_text,reply_markup=keyboard)
```

Функція «show_user_rating» використовується для відображення рейтингу користувача. Давайте розберемо цю функцію:

1. «user_data = users_data[user_id]»: Отримуємо дані користувача зі словника «users_data» за його ідентифікатором «user_id». Це дозволяє отримати інформацію про правильні відповіді, загальну кількість відповідей та ім'я користувача.

2. «if user_data.correct_answers + user_data.total_answers != 0:»: Перевіряємо, що сума правильних відповідей та загальної кількості відповідей не дорівнює нулю. Це робиться для запобігання розподілу на нуль та помилки у розрахунку відсоткового співвідношення.

3. «(user_data.correct_answers / (user_data.correct_answers + user_data.total_answers)) * 100»: Розраховуємо відсоток правильних відповідей, ділячи кількість правильних відповідей на суму правильних та загальних відповідей і помножуючи на 100.

4. «rating_text = f"Ваш рейтинг, {user_data.name}, Ваш ID {user_id}:\n\n"»: Формуємо текст рейтингу, включаючи ім'я користувача та його ідентифікатор.

5. «rating_text += f«Правильно вирішено {user_data.correct_answers} з {user_data.correct_answers + user_data.total_answers} задач. Відсоток правильних відповідей:{percentage:.2f}%\n»»: Додаємо інформацію про кількість правильних відповідей та загальну кількість завдань, а також відсоток правильних відповідей до тексту рейтингу.

6. «rating_text += f«\n Якщо хочеш змінити ім'я, введи команду /start»»: Додаємо повідомлення про те, як змінити ім'я користувача.

7. «keyboard = ReplyKeyboardMarkup(resize_keyboard=True)»: Створюємо екземпляр клавіатури зі змінним розміром.

8. «keyboard.add(KeyboardButton(«/rating»))»: Додаємо кнопку «/rating» у створену клавіатуру.

9. «keyboard.add(KeyboardButton(«/help»))»: Додаємо кнопку «/help» у створену клавіатуру.

10. «keyboard.add(KeyboardButton(«/start»))»: Додаємо кнопку «/start» у створену клавіатуру.

11. «await bot.send_message(user_id, rating_text, reply_markup=keyboard)»: Відправляємо користувачеві повідомлення з рейтингом, використовуючи текст «rating_text», та відображаємо клавіатуру «keyboard» для зручності користувача.

Таким чином, функція «show_user_rating» формує та відправляє користувачеві повідомлення з його рейтингом, включаючи інформацію про правильні відповіді, загальну кількість відповідей, відсоток правильних відповідей та пропонує кнопки для навігації.

Тепер напишемо обробник та логіку виведення команди /rating.

```python
@dp.message_handler(commands=['rating'])
async def show_rating(message: types.Message, state: FSMContext):
    await GameStates.RATING.set()

    # Сортування користувачів за балами за правильні відповіді у спадному порядку
    sorted_users = sorted(users_data.items(), key=lambda x: x[1].correct_answers, reverse=True)

    if sorted_users:
        rating_text = "Рейтинг:\n\n"
        for rank, (user_id, user_data) in enumerate(sorted_users, start=1):
            try:
                percentage = (user_data.correct_answers / (user_data.correct_answers + user_data.total_answers)) * 100
            except ZeroDivisionError:
                percentage = 0.0

            rating_text += f"{rank}. {user_data.name} (ID: {user_id}): Бали - {user_data.correct_answers}. Відсоток правильних відповідей:{percentage:.2f}%\n"

        await message.answer(rating_text)
    else:
        await message.answer("Рейтинг пустий")

    await state.finish()
```

Давайте розберемо хендлер «show_rating» у нашому коді:

1. «await GameStates.RATING.set()»: Встановлюємо стан «RATING» у машині станів («FSMContext2), щоб відстежувати поточний стан гри.

2. «sorted_users = sorted(users_data.items(), key=lambda x: x[1].correct_answers, reverse=True)»: Сортуємо користувачів в «users_data2 за кількістю правильних відповідей («correct_answers») у порядку зменшення. Це дозволяє відобразити рейтинг користувачів на основі їхнього успіху.

3. «if sorted_users:»: Перевіряємо, чи є користувачі у відсортованому списку. Якщо так, то виконуємо такі дії, інакше надсилаємо повідомлення «Рейтинг порожній».

4. «rating_text = «Рейтинг:\n\n»»: Створюємо текстову змінну «rating_text» та ініціалізуємо її заголовком 2Рейтинг:\n\n».

5. «for rank, (user_id, user_data) in enumerate(sorted_users, start=1):»: Ітеруємося по відсортованому списку користувачів та отримуємо ранг (порядковий номер) користувача, його ідентифікатор та дані користувача.

6. «try:»: Починаємо блок обробки винятків для розподілу на нуль.

7. «percentage=(user_data.correct_answers/(user_data.correct_answers + user_data.total_answers)) * 100»: Розраховуємо відсоток правильних відповідей для користувача, використовуючи кількість правильних відповідей та загальну кількість відповідей.

8. «except ZeroDivisionError:»: Обробляємо виняток «ZeroDivisionError», який виникає, якщо загальна кількість відповідей користувача дорівнює нулю. І тут встановлюємо відсоток рівним 0.0.

9. «rating_text += f«{rank}. {user_data.name} (ID: {user_id}): Балі - {user_data.correct_answers}. Відсоток правильних відповідей:{percentage:.2f}%\n»»: Додаємо інформацію про користувача у текст рейтингу, включаючи ранг, ім'я, ідентифікатор, кількість правильних відповідей та відсоток правильних відповідей.

10. «await message.answer(rating_text)»: Надсилаємо повідомлення з текстом рейтингу користувачеві.

11. «await state.finish()»: Завершуємо поточний стан гри у машині станів.

Таким чином, хендлер «show_rating» відображає рейтинг користувачів на основі їх правильних відповідей. Він сортує користувачів за кількістю правильних відповідей та формує текстове повідомлення з інформацією про рейтинг кожного користувача.

Основна структура робота готова, додамо пару команд, такі як «/help», «/contact», «/stats».

У команді «help» виводимо користувачеві інструкцію до всіх наших команд.

У команді «contact» надішлемо користувачеві свої дані

У команді «stats» викличемо асинхронну функцію «show_user_rating» і покажемо особисту статистику користувачеві.

Пишемо код для запуску бота.

```
if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)
```

Код «if __name__ == «__main__»:» використовується в Python для визначення точки входу до програми. Цей блок умови буде виконуватися

лише в тому випадку, якщо скрипт запускається безпосередньо, а не імпортується як модуль.

В даному випадку код «executor.start_polling(dp, skip_updates=True)» запускає нескінченний цикл, який очікує та обробляє вхідні повідомлення та події від користувачів. Він використовує `dp` (диспетчер) для обробки цих повідомлень та подій. Прапор «skip_updates=True» вказує, що потрібно пропускати оновлення, пропущені під час простою або відключення бота.

В результаті, коли ви запускаєте скрипт безпосередньо, функція «executor.start_polling(dp, skip_updates=True)» починає прослуховувати та обробляти вхідні повідомлення та події, що дозволяє роботі взаємодіяти з користувачами в реальному часі.

4.3. Візуальне оформлення Bot-Challenge

Стартова сторінка бота:

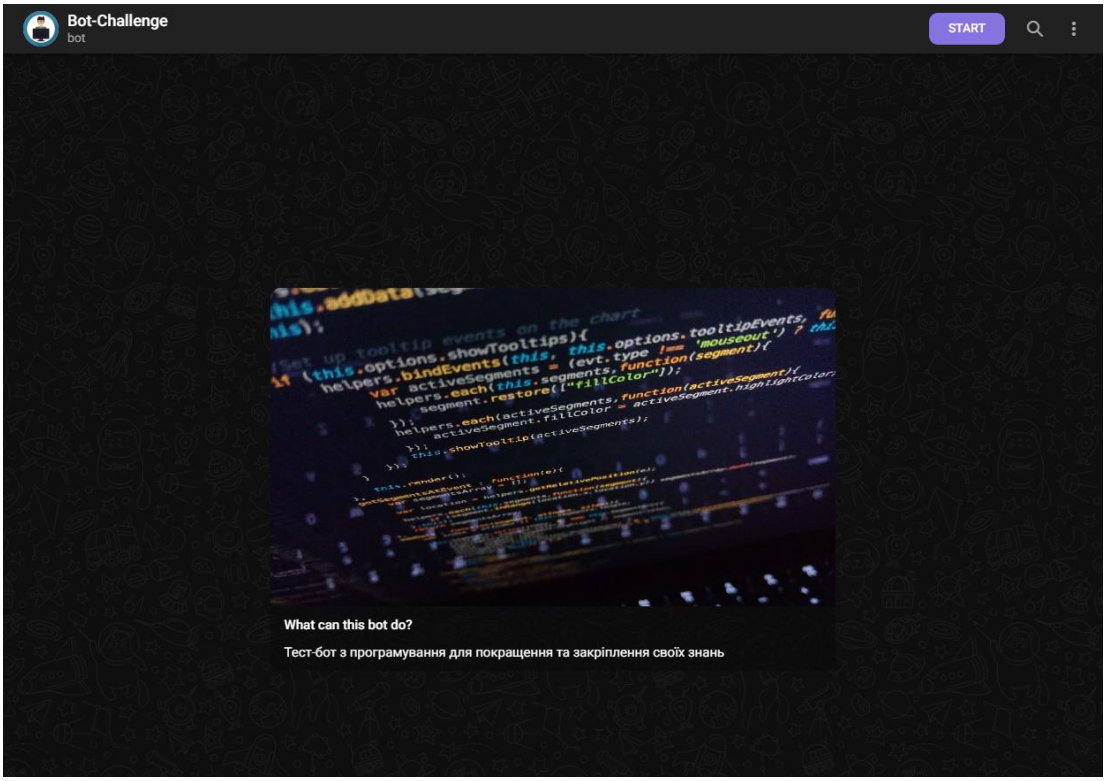

Рисунок 4.1 - Стартова сторінка бота

Інформаційна частина бота:

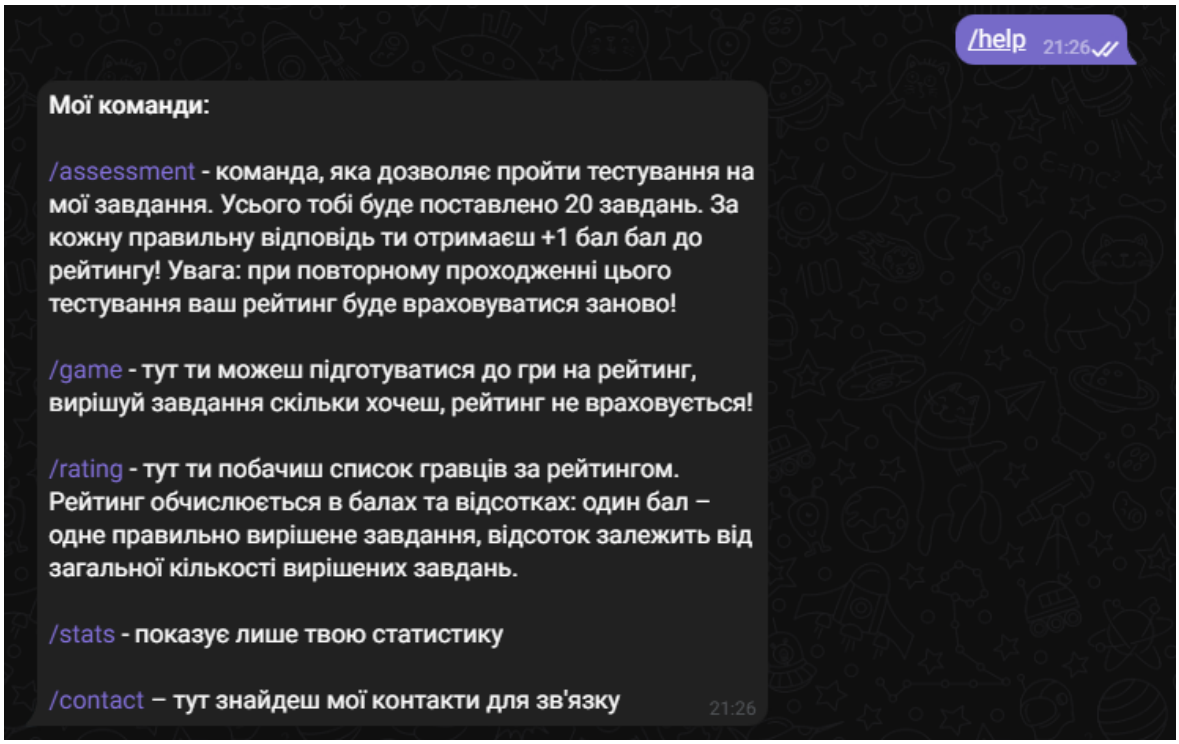

Рисунок 4.2 – Інформаційна частина бота

Вигляд тестування:

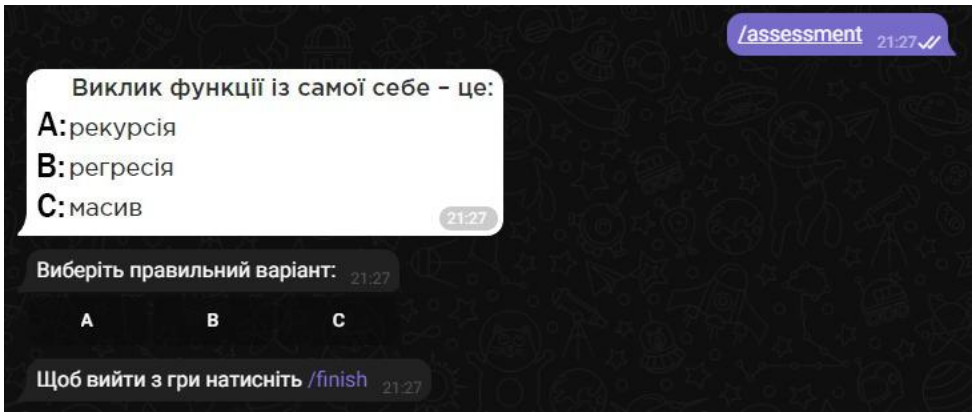

Рисунок 4.3 – Вигляд тестування в боті

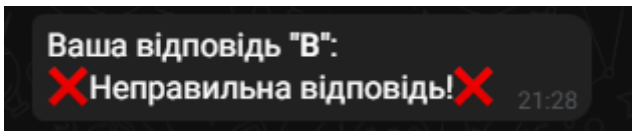

Рисунок 4.4 – Вигляд неправильної відповіді

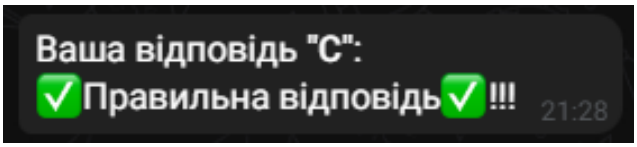

Рисунок 4.4 – Вигляд правильної відповіді

Вигляд таблиці рейтингу:

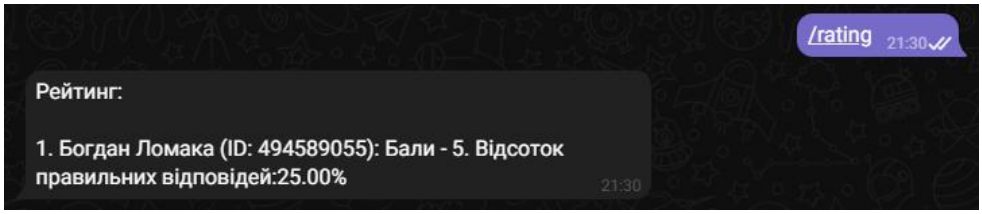

Рисунок 4.5 – Вигляд таблиці рейтингу

Вигляд статистики користувача:

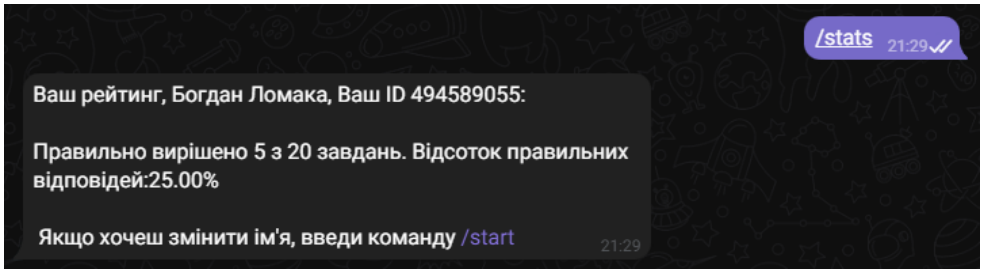

Рисунок 4.6 – Вигляд статистики користувача

4.4. Представлення результатів тестування і аналізу роботи розробленого тренажера

Після того як користувач зайшов на бота через посилання, йому показує головну сторінку бота і після натискання на кнопку «START» в телеграмі, бот повинен почати свою роботу і запросити у користувача його ім’я та прізвище:

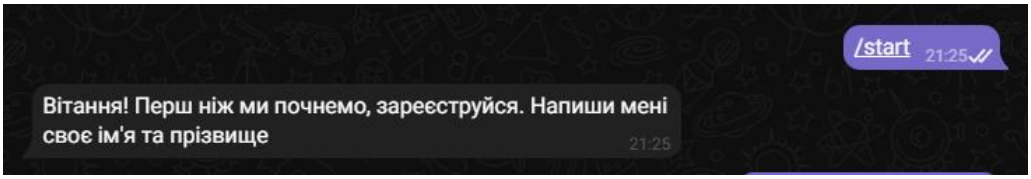

Рисунок 4.7 – Вигляд реєстрації в боті

Після того як користувач вписав свої дані, бот його записує в свою базу даних і пропонує користувачу дізнатись про всі запити які гравець може використовувати, для того щоб гравцю було легше орієнтуватися і розпочати гру:

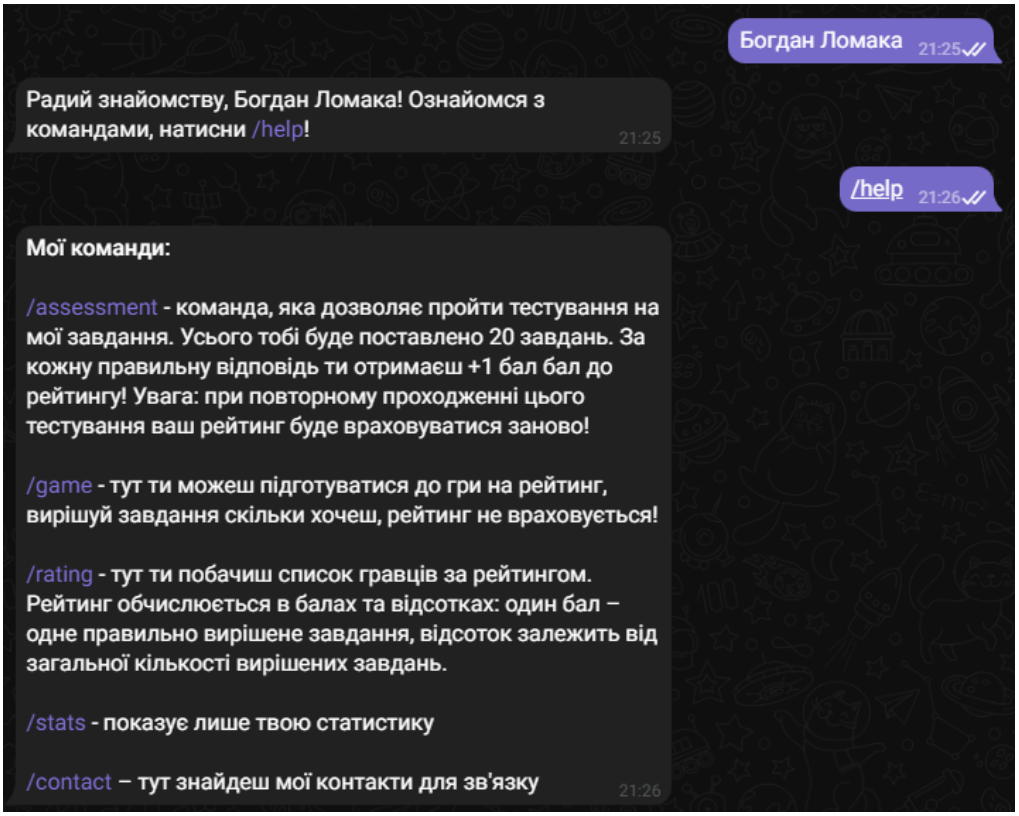

Рисунок 4.8 – Інформаційна частина бота

Після того як користувач огляне запити він може вибрати любу з них, натискаючи на запит «/assessment» він розпочинає безпосередньо проходити сам тест на бали. Тест містить в собі 20 запитань і кожного разу при проходженні цього тесту запитання подаються в випадковій послідовності. При завершені тесту йому запропонує переглянути свою особисту статистику а також переглянути рейтинг всіх гравців:

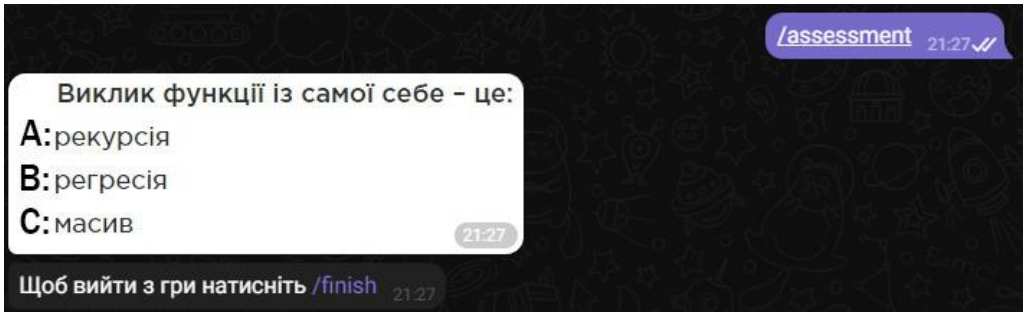

Рисунок 4.8 – Вигляд тестування в боті

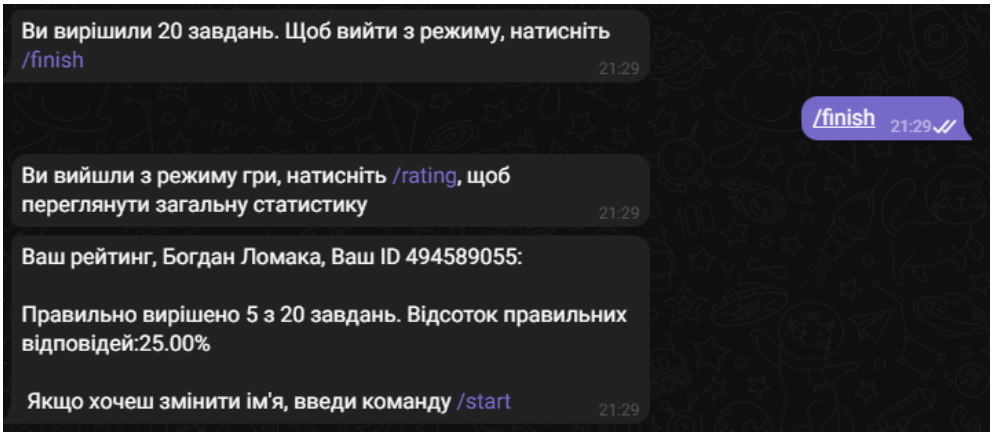

Рисунок 4.8 – Вигляд завершення тесту

Якщо користувач натисне на запит «/game», то гравець зможе безкінечно проходити тестування яке не впливає на його рейтинг а лише підготовлює його до тесту на рейтинг:

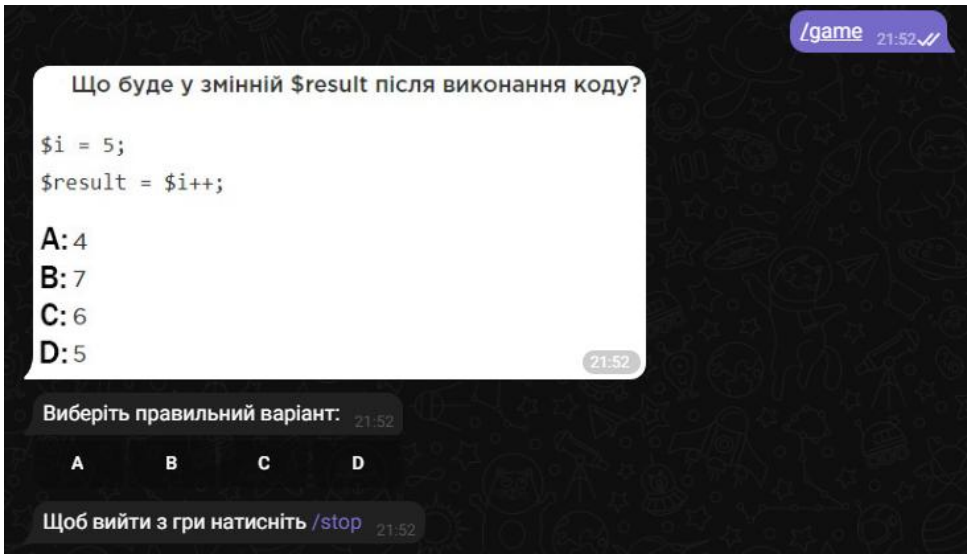

Рисунок 4.9 – Вигляд гри в боті

Натискаючи на запит «/rating», гравець зможе подивитись своє місце у топі а також переглянути результати інших гравців які пройшли тестування:

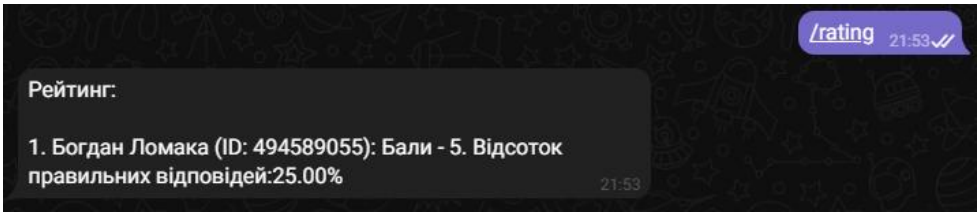

Рисунок 4.10 – Вигляд таблиці рейтингу

Якщо гравець захоче переглянути свою особисту статистику то йому потрібно натиснути на запит «/stats»:

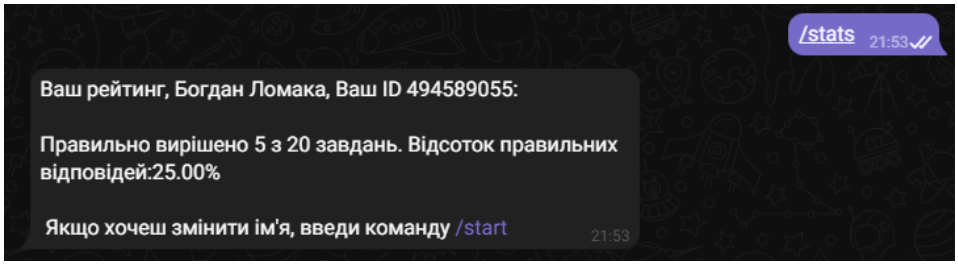

Рисунок 4.11 – Вигляд статистики користувача

Також якщо повторно ввести запит «/start», гравець може змінити своє ім’я при тому що його статистика збережеться.

Якщо користувач зіткнувся з якоюсь проблемою, або захоче запропонувати свої ідеї що до покращення роботи та фунціоналу бота, він може натиснути на запит «contact», та переглянути контакти по яким він може зв’язатися з розробником бота:

Рисунок 4.11 – Вигляд контактних даних

Тестування бота не показало ніяких помилок.

ВИСНОВКИ

Впровадження чат-боту в навчальний процес дозволить підвищити ефективність навчання за допомогою віртуальних тренажерів. Саме тому в рамках курсового проекту ставилися задачі по розробці чат-боту для студентів спеціальності Комп’ютерні науки «Програмування: Bot-Challenge», та його програмування.

Основні результати роботи:

- сформульовані основні вимоги до чат-боту, що розробляється;
- проведено огляд переваг та недоліків та відомого програмного забезпечення для дистанційного навчання;
- розглянуто доступні у мережі програми-тренажери;
- складено алгоритм тренажера з теми чат-бот для студентів спеціальності Комп’ютерні науки «Програмування: Bot-Challenge».

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Консультація з розробки ботів в телеграм [Електронний ресурс]. Режим доступу до ресурсу: URL: https://sharkdevelop.com - Назва з екрану.
2. Що таке Telegram? [Електронний ресурс]. Режим доступу до ресурсу: URL: https://uk.wikipedia.org/wiki/Telegram - Назва з екрану.
3. Запитання і відповіді стосовно Телеграм ботів [Електронний ресурс]. Режим доступу до ресурсу: URL: https://stackoverflow.com/questions/tagged/telegram-bot - Назва з екрану.
4. Навіщо і кому потрібні чат-боти? [Електронний ресурс]. Режим доступу до ресурсу: URL: https://ideadigital.agency/ru/blog/chat-bot/ - Назва з екрану.
5. Офіційна документація Telegram Bot API [Електронний ресурс]. Режим доступу до ресурсу: URL: https://core.telegram.org/bots/api - Назва з екрану.
6. Чи ефективні чат-боти? [Електронний ресурс]. Режим доступу до ресурсу: URL: https://www.unisender.com/ru/blog/kuhnya/chat-boty-vnedrenie/ - Назва з екрану.
7. Дізнайтесь про переваги чат-бота [Електронний ресурс]. Режим доступу до ресурсу: URL: https://sendpulse.ua/ru/support/glossary/chatbot - Назва з екрану.
8. Як користуватися ботами в Telegram [Електронний ресурс]. Режим доступу до ресурсу: URL: https://lemarbet.com/razvitie-internet-magazina/boty-v-telegram/ - Назва з екрану.
9. Офіційний сайт Python [Електронний ресурс]. Режим доступу до ресурсу: URL: https://www.python.org – Назва з екрану.

ДОДАТОК А. ВИХІДНІ КОДИ

```
import os
import random
from aiogram import Bot, types
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton
from aiogram.dispatcher import Dispatcher
from aiogram.utils import executor
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup
from TOKENAPI import TOKEN
from BData import CORRECT_ANSWERS, IMAGES_DIR
from TEXT import HELP_TEXT

bot = Bot(token=TOKEN)
storage = MemoryStorage()
dp = Dispatcher(bot, storage=storage)

all_images = os.listdir(IMAGES_DIR)
random.shuffle(all_images)

class GameStates(StatesGroup):
    GAME = State()
    RATING = State()
    NAME = State()
    NO_RATING = State()

users_data = {}
```

```
class UserData:
    def __init__(self, correct_answers=0, total_answers=0, name=""):
        self.correct_answers = correct_answers
        self.total_answers = total_answers
        self.name = name

    def reset_rating(self):
        self.correct_answers = 0
        self.total_answers = 0

def get_random_image():
    images = os.listdir(IMAGES_DIR)
    image = random.choice(images)
    return os.path.join(IMAGES_DIR, image)

def get_num_of_options(image_path):
    filename = os.path.basename(image_path)
    num_str = filename.split('.')[0][-1]
    return int(num_str)

async def finish_game(chat_id):
    user_data = users_data[chat_id]
    # Очищуємо лічильник завдань
    user_data.task_count = 0
    # Очищаємо список фотографій
    user_data.images = []
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton('/finish'))
    await bot.send_message(chat_id, 'Ви вирішили 20 завдань. Щоб вийти з режиму, натисніть /finish', reply_markup=keyboard)
```

```
async def send_next_question(chat_id):
    user_data = users_data[chat_id]
    if user_data.task_count >= 20:
        await finish_game(chat_id)
        return

    image_path = os.path.join(IMAGES_DIR, user_data.images.pop(0))

    # Відправляємо зображення із завданням
    with open(image_path, 'rb') as photo:
        message = await bot.send_photo(chat_id, photo)

    # Отримуємо кількість варіантів відповідей з імені файлу
    num_of_options = get_num_of_options(image_path)

    # Зберігаємо правильну відповідь, пов'язану з надісланою картинкою
    correct_answer = CORRECT_ANSWERS.get(os.path.basename(image_path))

    # Зберігаємо правильну відповідь та кількість варіантів відповідей у стан машини
    await GameStates.GAME.set()
    await dp.current_state(chat=chat_id).update_data(correct_answer=correct_answer,
num_of_options=num_of_options)

    # Формуємо варіанти відповідей у вигляді інлайн-кнопок
    keyboard = types.InlineKeyboardMarkup(row_width=num_of_options)  # Вказуємо
кількість кнопок у рядку
    options = ['A', 'B', 'C', 'D', 'E'][:num_of_options]  # Допустимі варіанти відповідей
    keyboard.add(*[types.InlineKeyboardButton(text=option, callback_data=option) for option in
options])  # Додаємо всі кнопки одночасно
    keyboardst = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboardst.add(KeyboardButton('/finish'))
    user_data.task_count += 1
```

```
    # Зберігаємо ідентифікатор повідомлення з інлайн-клавіатурою для подальшого видалення
    if chat_id not in users_data:
        users_data[chat_id] = UserData()
    users_data[chat_id].last_message_id = message.message_id

    await bot.send_message(chat_id, 'Виберіть правильний варіант:', reply_markup=keyboard)
    await bot.send_message(chat_id, 'Щоб вийти з гри натисніть /finish', reply_markup=keyboardst)

async def send_next_question_no_rating(chat_id):
    image_path = get_random_image()

    # Відправляємо зображення із завданням
    with open(image_path, 'rb') as photo:
        message = await bot.send_photo(chat_id, photo)

    # Отримуємо кількість варіантів відповідей з імені файлу
    num_of_options = get_num_of_options(image_path)

    # Зберігаємо правильну відповідь, пов'язану з надісланою картинкою
    correct_answer = CORRECT_ANSWERS.get(os.path.basename(image_path))

    # Зберігаємо правильну відповідь та кількість варіантів відповідей у стан машини
    await GameStates.NO_RATING.set()
    await dp.current_state(chat=chat_id).update_data(correct_answer=correct_answer, num_of_options=num_of_options)

    # Формуємо варіанти відповідей у вигляді інлайн-кнопок
    keyboard = types.InlineKeyboardMarkup(row_width=num_of_options)  # Вказуємо кількість кнопок у рядку
    options = ['A', 'B', 'C', 'D', 'E'][:num_of_options]  # Допустимі варіанти відповідей
```

```
    keyboard.add(*[types.InlineKeyboardButton(text=option, callback_data=option) for option in options])  # Додаємо всі кнопки одночасно

    keyboardst = ReplyKeyboardMarkup(resize_keyboard=True)

    keyboardst.add(KeyboardButton('/stop'))

    # Зберігаємо ідентифікатор повідомлення з інлайн-клавіатурою для подальшого видалення

    if chat_id not in users_data:

        users_data[chat_id] = UserData()

    users_data[chat_id].last_message_id = message.message_id


    await bot.send_message(chat_id, 'Виберіть правильний варіант:', reply_markup=keyboard)

    await bot.send_message(chat_id, 'Щоб вийти з гри натисніть /stop', reply_markup=keyboardst)


@dp.message_handler(commands=['start'])

async def start(message: types.Message):

    await message.answer("Вітання! Перш ніж ми почнемо, зареєструйся. Напиши мені своє ім'я та прізвище")

    await GameStates.NAME.set()



@dp.message_handler(state=GameStates.NAME)

async def process_name(message: types.Message, state: FSMContext):

    name = message.text

    user_id = message.from_user.id

    if user_id not in users_data:

        users_data[user_id] = UserData(name=name)

    else:

        users_data[user_id].name = name

    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)

    keyboard.add(KeyboardButton('/help'))
```

```
    await message.answer(f"Радий знайомству, {name}! Ознайомся з командами, натисни /help!", reply_markup=keyboard)
    await state.finish()


@dp.message_handler(commands=['assessment'])
async def start_game(message: types.Message):
    user_id = message.chat.id
    if user_id not in users_data:
        users_data[user_id] = UserData(name="")
    else:
        users_data[user_id].reset_rating()
    users_data[user_id].images = all_images.copy()
    users_data[user_id].task_count = 0
    await send_next_question(user_id)


@dp.message_handler(commands=['finish'], state=GameStates.GAME)
async def stop_game(message: types.Message, state: FSMContext):
    user_id = message.from_user.id
    if user_id in users_data:
        await state.finish()
        keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
        keyboard.add(KeyboardButton('/rating'))
        keyboard.add(KeyboardButton('/help'))
        await message.answer('Ви вийшли з режиму гри, натисніть /rating, щоб переглянути загальну статистику', reply_markup=keyboard)
        await show_user_rating(user_id)
    else:
        await message.answer("Ви ще не розпочали гру.")


@dp.callback_query_handler(lambda c: True, state=GameStates.GAME)
async def process_callback(callback_query: types.CallbackQuery, state: FSMContext):
```

```
    await bot.answer_callback_query(callback_query.id)

    selected_option = callback_query.data

    # Отримуємо правильну відповідь та кількість варіантів відповідей зі стану машини
    data = await state.get_data()
    correct_answer = data.get('correct_answer')
    num_of_options = data.get('num_of_options')

    if selected_option == correct_answer:
        await bot.send_message(callback_query.from_user.id, f'Ваша відповідь
<b>"{selected_option}"</b>:\n✅Правильна відповідь✅!!!', parse_mode='HTML')
        user_id = callback_query.from_user.id
        if user_id in users_data:
            users_data[user_id].correct_answers += 1
        else:
            users_data[user_id] = UserData(correct_answers=1, total_answers=1)
    else:
        await bot.send_message(callback_query.from_user.id, f'Ваша відповідь
<b>"{selected_option}"</b>:\n❌Неправильна відповідь!❌', parse_mode='HTML')
        user_id = callback_query.from_user.id
        if user_id not in users_data:
            users_data[user_id] = UserData(correct_answers=0, total_answers=1)
        else:
            users_data[user_id].total_answers += 1
    await bot.delete_message(callback_query.message.chat.id,
callback_query.message.message_id)
    await send_next_question(callback_query.from_user.id)

@dp.message_handler(commands=['game'])
async def start_game(message: types.Message):
```

```
    await send_next_question_no_rating(message.chat.id)


@dp.message_handler(commands=['stop'], state=GameStates.NO_RATING)
async def stop_game(message: types.Message, state: FSMContext):
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton('/assessment'))
    keyboard.add(KeyboardButton('/help'))
    await message.answer('Ви закінчили гру. Натисніть /assessment, щоб брати участь у грі на рейтинг', reply_markup=keyboard)
    await state.finish()


@dp.callback_query_handler(lambda c: True, state=GameStates.NO_RATING)
async def process_callback_no_ratting(callback_query: types.CallbackQuery, state: FSMContext):
    await bot.answer_callback_query(callback_query.id)
    selected_option = callback_query.data
    # Отримуємо правильну відповідь та кількість варіантів відповідей зі стану машини
    data = await state.get_data()
    correct_answer = data.get('correct_answer')
    num_of_options = data.get('num_of_options')
    if selected_option == correct_answer:
        await bot.send_message(callback_query.from_user.id, f'Ваша відповідь <b>"{selected_option}"</b>:\n✅Правильна відповідь✅!!!', parse_mode='HTML')
    else:
        await bot.send_message(callback_query.from_user.id, f'Ваша відповідь <b>"{selected_option}"</b>:\n❌Неправильна відповідь!❌', parse_mode='HTML')
     # Видаляємо інлайн-клавіатуру
    await bot.delete_message(callback_query.message.chat.id, callback_query.message.message_id)


    # Скидаємо стан машини
    await state.finish()
```

```
    # Відправляємо наступну картинку із завданням

    await send_next_question_no_rating(callback_query.from_user.id)

@dp.message_handler(commands=['rating'])
async def show_rating(message: types.Message, state: FSMContext):
    await GameStates.RATING.set()

    # Сортування користувачів за балами за правильні відповіді у спадному порядку
    sorted_users = sorted(users_data.items(), key=lambda x: x[1].correct_answers, reverse=True)
    if sorted_users:
        rating_text = "Рейтинг:\n\n"
        for rank, (user_id, user_data) in enumerate(sorted_users, start=1):
            try:
                percentage = (user_data.correct_answers / (user_data.correct_answers +
user_data.total_answers)) * 100
            except ZeroDivisionError:
                percentage = 0.0

            rating_text += f"{rank}. {user_data.name} (ID: {user_id}): Бали -
{user_data.correct_answers}. Відсоток правильних відповідей:{percentage:.2f}%\n"

        await message.answer(rating_text)
    else:
        await message.answer("Рейтинг пустий")

    await state.finish()

async def show_user_rating(user_id):
    user_data = users_data[user_id]
```

```
    if user_data.correct_answers + user_data.total_answers != 0:
        percentage = (user_data.correct_answers / (user_data.correct_answers + user_data.total_answers)) * 100
    else:
        percentage = 0
    rating_text = f"Ваш рейтинг, {user_data.name}, Ваш ID {user_id}:\n\n"
    rating_text += f"Правильно вирішено {user_data.correct_answers} з {user_data.correct_answers + user_data.total_answers} завдань. Відсоток правильних відповідей:{percentage:.2f}%\n"
    rating_text += f"\n Якщо хочеш змінити ім'я, введи команду /start"
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton('/rating'))
    keyboard.add(KeyboardButton('/help'))
    keyboard.add(KeyboardButton('/start'))
    await bot.send_message(user_id, rating_text,reply_markup=keyboard)


@dp.message_handler(commands=['stats'],)
async def stats_commands(message: types.Message, state: FSMContext):
    user_id = message.from_user.id
    if user_id in users_data:
        await state.finish()
        await show_user_rating(user_id)
    else:
        await message.answer("Ви ще не розпочали гру.")


@dp.message_handler(commands=['contact'])
async def contact_commands(message: types.Message):
    await message.answer('Мої контакти:\nТелеграм: @lilbord\nПошта: loomlxd@gmail.com')


@dp.message_handler(commands=['help'])
async def contact_commands(message: types.Message):
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
```

```
    keyboard.add(KeyboardButton('/assessment'))
    keyboard.add(KeyboardButton('/game'))
    keyboard.add(KeyboardButton('/rating'))
    keyboard.add(KeyboardButton('/stats'))
    keyboard.add(KeyboardButton('/contact'))
    await message.answer(HELP_TEXT, reply_markup=keyboard, parse_mode='HTML')


@dp.message_handler(state=GameStates.GAME)
async def handle_text_message(message: types.Message):
    await bot.send_message(message.chat.id, "Будь ласка, виберіть відповідь за допомогою кнопок.")


@dp.message_handler(state=GameStates.NO_RATING)
async def handle_text_no_rat_message(message: types.Message):
    await bot.send_message(message.chat.id, "Будь ласка, виберіть відповідь за допомогою кнопок")


if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)
```