Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Інтелектуальний помічник для складання та модернізації ПК

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-21-1
Є. В. Штирляєв

Керівник
кваліфікаційної роботи А. М. Стрюк

Завідувач кафедри А. М. Стрюк

Кривий Ріг
2025

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

А. М. Стрюк

«___» _____ 20__ р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-21-1 Штирляєву Єгору Валерійовичу

1. Тема: Інтелектуальний помічник для складання та модернізації ПК затверджено наказом по КНУ № 7 від «31» січня 2025 р

2. Термін подання студентом закінченої роботи: «31» травня 2025р.

3. Вихідні дані по роботі: Розроблювана система повинна взаємодіяти з Telegram-ботом, зберігати та оновлювати дані користувача, перевіряти сумісність комплектуючих, аналізувати конфігурацію ПК і формувати рекомендації щодо модернізації.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): Опис функціоналу бота, принципи взаємодії з користувачем, розробка структури збереження даних, реалізація Telegram-бота з використанням підключеного ШІ для аналізу сумісності та надання рекомендацій, тестування роботи системи.

5. Перелік ілюстративного матеріалу: архітектурні схеми системи, блок-схеми роботи модулів, схема взаємодії компонентів, знімки Telegram-інтерфейсу

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Аналіз аналогів та формування вимог	27.03.2025 – 28.03.2025
2	Збір даних про комплектуючі ПК	29.03.2025 – 09.04.2025
3	Проєктування структури Telegram-бота	10.04.2025 – 18.04.2025
4	Реалізація збереження та оновлення даних користувача	19.04.2025 – 23.04.2025
5	Розробка перевірки сумісності компонентів ШІ	24.04.2025 – 30.04.2025
6	Інтеграція з ШІ-сервісом	01.04.2025 – 04.04.2025
7	Тестування бота та виправлення помилок	05.04.2025 – 13.05.2025
8	Підготовка 1–2 розділів записки	03.05.2025 – 20.05.2025
9	Підготовка 3–4 розділів записки	21.05.2025 – 25.05.2025
10	Оформлення записки	26.05.2025 – 30.05.2025

Дата видачі завдання: «28» Січня 2025 р.

Студент Є. В. Штирляєв

Керівник роботи А. М. Стрюк

РЕФЕРАТ

СКЛАДАННЯ ПК, ІНТЕЛЕКТУАЛЬНИЙ ПОМІЧНИК, ШТУЧНИЙ ІНТЕЛЕКТ, ТЕЛЕГРАМ-БОТ, КОМПЛЕКТУЮЧІ, СУМІСНІСТЬ, АВТОМАТИЗАЦІЯ, ОНОВЛЕННЯ СИСТЕМИ

Пояснювальна записка містить 30 сторінок, 17 рисунків, 8 використані джерела.

Метою кваліфікаційної роботи є створення Telegram-бота - інтелектуального помічника для користувачів, які хочуть зібрати або оновити персональний комп'ютер. Основна функція бота - надання порад щодо вибору комплектуючих з урахуванням сумісності, потреб і бюджету користувача.

Особливістю проекту є використання вбудованої моделі штучного інтелекту, яка веде діалог з користувачем, надає пояснення й рекомендації. Логіка ШІ не змінюється - вона працює як окремий модуль у межах системи.

Користувач може ввести наявну конфігурацію ПК або описати бажані характеристики. Бот аналізує ці дані, перевіряє сумісність компонентів (CPU, GPU, RAM, накопичувачів тощо) та пропонує рекомендації. Також можна змінювати окремі параметри, переглядати або скинути конфігурацію.

Інтерфейс реалізовано у вигляді Telegram-бота, що забезпечує простоту та доступність з будь-якого пристрою.

У перспективі передбачено розширення функцій - інтеграцію з базами цін, рейтингами та візуальними інструментами порівняння. Проєкт створено з урахуванням подальшого розвитку - у майбутньому планується розширення функціоналу, зокрема інтеграція з базами даних цін, рейтингами комплектуючих або інструментами візуального порівняння.

ABSTRACT

PC BUILDING, INTELLIGENT ASSISTANT, ARTIFICIAL INTELLIGENCE, TELEGRAM BOT, COMPONENTS, COMPATIBILITY, AUTOMATION, SYSTEM UPGRADE

The explanatory note contains 30 pages, 17 figures, 8 and referenced sources.

The aim of this qualification project is to develop a Telegram bot that serves as an intelligent assistant for users looking to build or upgrade a personal computer. The main function of the bot is to provide recommendations for selecting components based on compatibility, user needs, and budget.

A key feature of the project is the use of a built-in artificial intelligence model that communicates with users in a conversational format, offering explanations and suggestions. The AI logic itself remains unchanged and functions as an independent module within the system.

The user can enter their current PC configuration or describe the desired specifications. The bot analyzes this information, checks the compatibility of components (CPU, GPU, RAM, storage, etc.), and offers upgrade suggestions. It also allows for editing individual parameters, viewing the current setup, or resetting it to start a new selection.

The interface is implemented as a Telegram bot, ensuring accessibility from any device and a user-friendly interaction format.

Future improvements include functionality expansion such as integration with pricing databases, component ratings, and visual comparison tools.

Зміст

ВСТУП 8
1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ РОБОТИ 9
1.1 Аналіз професійної галузі проблеми 9
1.2 Аналіз існуючих аналогів 10
1.3 Формулювання актуальності та завдань роботи 11
2 ФУНКЦІОНАЛЬНОЇ СХЕМИ І АЛГОРИТМУ 13
2.1 Розробка математичних моделей 13
2.1.1 Основні позначення 13
2.1.2 Опис функціонування системи 14
2.1.3 Інтерпретація як експертної системи 14
2.2 Розробка структурної та функціональної схем програмного комплексу 15
2.3 Розробка структурних схем баз даних UML 20
2.4. Проектування структур таблиць бази даних і вибір типів даних полів 22
2.5 Розробка UML-діаграм класів програмного комплексу 23
2.5.1 Структура програмного комплексу 23
2.5.1 Взаємозв’язки між компонентами 24
2.6 Розробка блок-схем алгоритмів програмного комплексу 25
2.6.1 Введення характеристик ПК 25
2.6.2 Оновлення окремої характеристики 27
2.6.3 Оцінка конфігурації ПК 28
2.6.4 Обробка довільних повідомлень 30
2.7 Розробка моделі візуального інтерфейсу програмного комплексу 31
2.7.1 Особливості реалізації 34
3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ПРОЕКТУ 35
3.1 Розробка основних класів програмних модулів проекту 35
3.2 Розробка візуального інтерфейсу програмного комплексу 36
ВИСНОВКИ 38

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 39
ПОВНИЙ ТЕКСТ ПРОГРАМНИХ МОДУЛІВ 40
Файл config.py 40
Файл data_manager.py 41
Файл main.py 42
СКРІНШОТИ РОБОТИ ІНТЕРФЕЙСУ 53

ВСТУП

У сучасному світі персональний комп’ютер є незамінним інструментом для вирішення широкого спектра завдань - від повсякденного спілкування до професійної обробки даних. Однак разом з різноманітністю задач зростає і кількість доступних комплектуючих, що робить процес оновлення або складання ПК досить складним, навіть для досвідчених користувачів. Що вже казати про новачків.

Більшість людей, звісно, хотіли б уникнути зайвих витрат на несумісне обладнання або комплектуючі з невдалим співвідношенням ціни та якості. Неправильний вибір може обернутися не лише фінансовими втратами, а й розчаруванням.

Саме тому було створено інтелектуального помічника - Telegram-бота з інтегрованим штучним інтелектом, який спрощує процес підбору та модернізації комп’ютера. На даному етапі він виконує такі ключові функції:

а) Допомога у виборі комплектуючих відповідно до заданого бюджету.

б) Консультації щодо складання ПК «з нуля» або оновлення існуючої системи за заданими критеріями.

в) Оцінка поточної конфігурації та рекомендації для підвищення її продуктивності.

г) Відповіді на технічні питання користувачів у форматі звичайного чату.

Завдяки використанню потужної ШІ-моделі бот здатен вести діалог природною мовою, що робить його доступним та зрозумілим навіть для людей без технічного досвіду. Він допомагає економити час, зменшує ризик помилок і підвищує комфорт від користування комп’ютером.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ РОБОТИ

1.1 Аналіз професійної галузі проблеми

В умовах стрімкого розвитку цифрових технологій процес підбору комплектуючих до персонального комп’ютера стає дедалі складнішим. Користувачам, незалежно від рівня підготовки, доводиться орієнтуватися в безмежній кількості моделей і технічних характеристик, що часто викликає плутанину і призводить до помилкових рішень при складанні або оновленні ПК. Серед основних проблем, з якими зіштовхуються користувачі:

а) Складність самостійного підбору сумісних компонентів.

б) Необхідність володіння великим обсягом технічних знань для розуміння характеристик і нюансів різних комплектуючих.

в) Високі ризики придбання несумісного обладнання, що може призвести до зайвих витрат.

г) Обмежена кількість україномовних сервісів з простим та інтуїтивно зрозумілим інтерфейсом.

На сьогоднішній день частковими розв’язаннями цієї проблеми виступають онлайн-сервіси для підбору конфігурацій. Проте переважна більшість таких сайтів доступна лише англійською мовою та має складний або перевантажений інтерфейс, що створює додаткові бар’єри для користувачів. Альтернативою можуть бути інформаційні ресурси на кшталт YouTube чи Google, або консультації в магазинах побутової техніки, як-от Comfy, Foxtrot, Allo тощо. Втім, якість таких порад часто залишає бажати кращого.

Проблема залишається актуальною й надалі, адже дедалі більше людей прагнуть зібрати власний комп’ютер самостійно - як з міркувань економії, так і задля особистого контролю над якістю. При цьому не всі готові звертатися до сторонніх фахівців, особливо враховуючи випадки недобросовісного ставлення з боку деяких «майстрів».

Саме тому виникла ідея створення інтелектуального Telegram-бота, який допоможе вирішити ці проблеми швидко, зручно та зрозуміло, навіть для недосвідчених користувачів.

1.2 Аналіз існуючих аналогів

Серед існуючих аналогів можна виокремити декілька популярних рішень, зокрема веб-сайти PCPartPicker та BuildMyPC. Обидва сервіси пропонують гнучкі можливості для підбору комплектуючих і мають досить потужний функціонал. Проте вони не позбавлені суттєвих недоліків, які обмежують їхню ефективність для українських користувачів:

а) Вартість комплектуючих базується на даних з іноземних магазинів, що робить підрахунки неактуальними для українського ринку.

б) Обидва сервіси є англомовними. Хоча вони підтримують багато мов, українська серед них відсутня.

в) Інтерфейс і логіка підбору можуть бути складними для користувача, який не має глибоких знань у сфері ПК-компонентів.

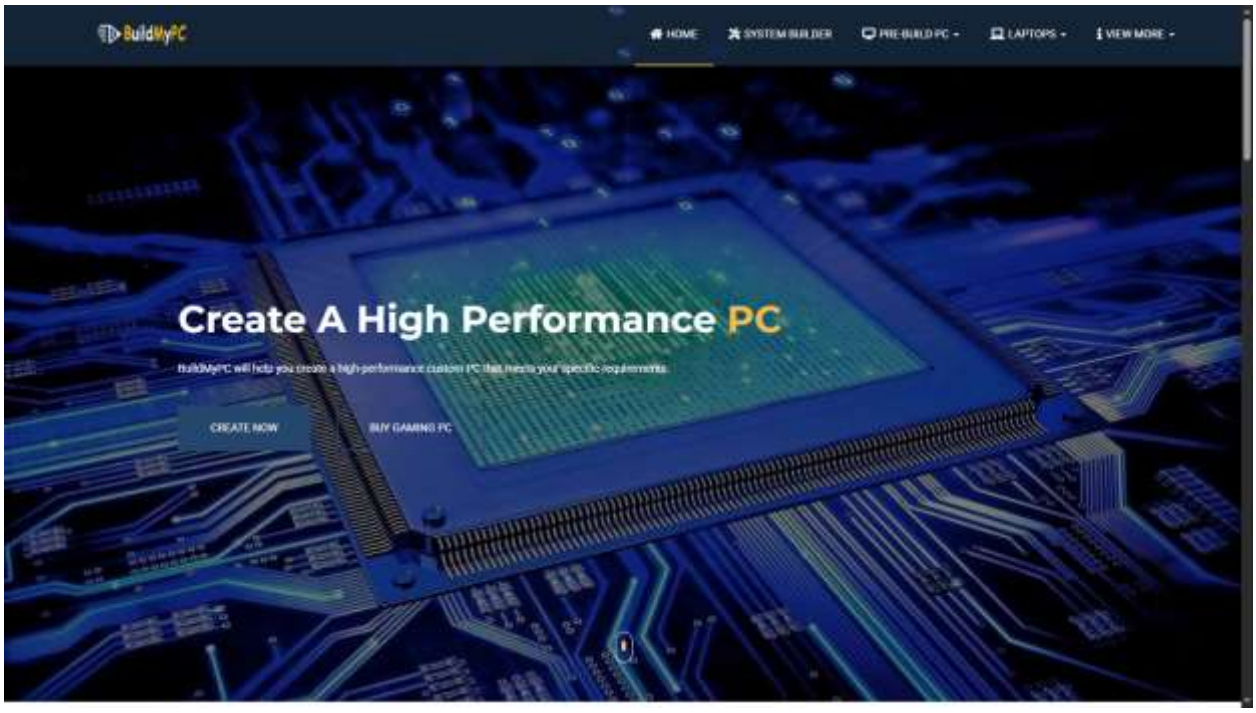

Рисунок 1.1 Головна сторінка BuildMyPC

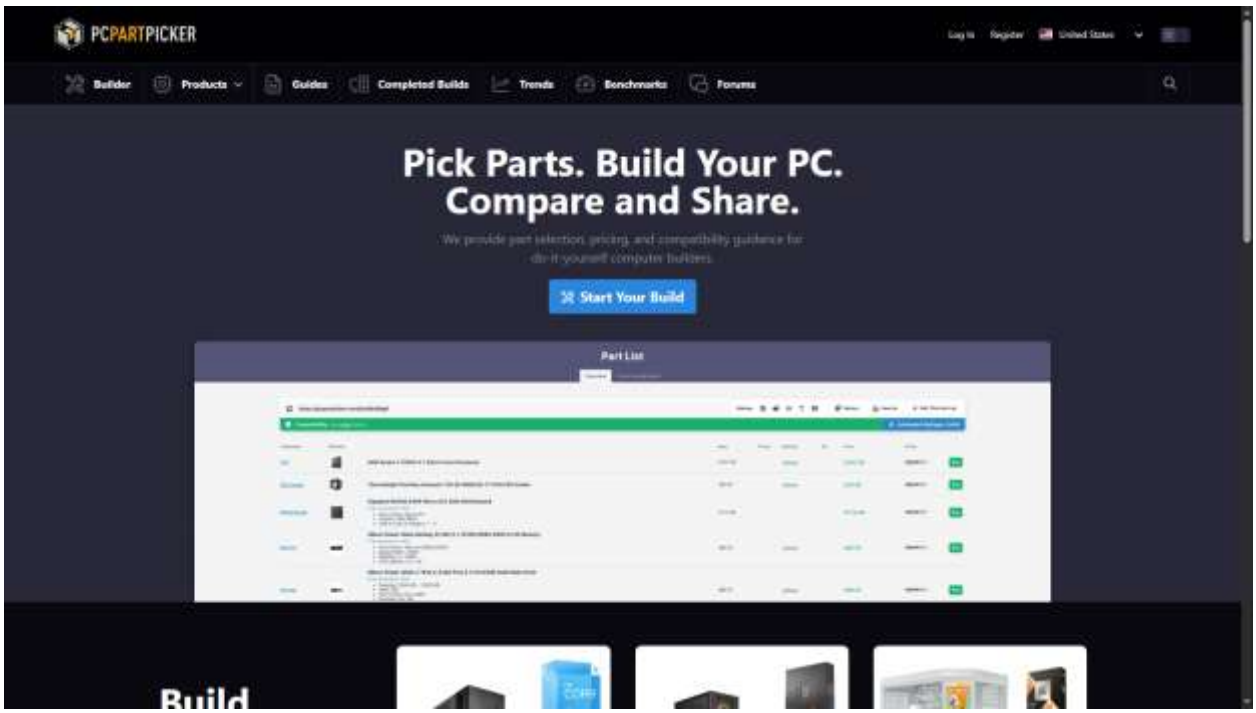

Рисунок 1.2 Головна сторінка PCPartPicker

Окрім веб-ресурсів, варто згадати і про подібні Telegram-боти, наприклад: @BuildPCBot та @PC_Bilder_Bot. Утім, ці проєкти більше не підтримуються й фактично припинили роботу.

На відміну від них, створений мною бот краще підходить для недосвідчених користувачів, які бажають зібрати або оновити свій ПК. Він має підтримку української мови, простий інтерфейс та зрозумілу логіку взаємодії. Головна мета - зробити процес максимально доступним і комфортним навіть для тих, хто раніше не мав справи з підбором комплектуючих.

Завдяки такому рішенню з’являється можливість зайняти ще малозаповнену нішу - інтелектуальних, україномовних Telegram-помічників у сфері комп’ютерної техніки. Це не лише підвищить зручність користування, а й сприятиме зростанню технічної обізнаності серед широкого кола користувачів.

1.3 Формулювання актуальності та завдань роботи

Актуальність теми: У наш час комп’ютер - це основний інструмент для виконання багатьох завдань. Кількість комплектуючих постійно зростає, і

навіть досвідченому користувачу іноді складно правильно підібрати деталі для свого ПК або зібрати його з нуля. Ніхто не хоче переплачувати за несумісне або неякісне обладнання, і це зрозуміло.

На сьогодні існує кілька сайтів для підбору комплектуючих, наприклад PCPartPicker або BuildMyPC. Вони мають великий вибір, але у них є суттєві мінуси: ціни не з українських магазинів, відсутність української мови та складний інтерфейс для новачків. Також були Telegram-боти, які допомагали з підбором, але багато з них перестали працювати.

Тому розробка простого і зручного Telegram-бота українською мовою, який допомагатиме користувачам підбирати ПК, буде актуальною ще довгий час. Бот допоможе тим, хто хоче зібрати свій ПК самостійно та заощадити гроші, не витрачаючи час на складні пошуки.

Мета та завдання дослідження: Метою цієї роботи є створення Telegram-бота, який допомагає користувачам підібрати збірку ПК з хорошим співвідношенням ціни та якості, враховуючи їхні потреби. Для цього потрібно:

а) Зібрати інформацію про потреби користувача.

б) Виявити слабкі місця у вже існуючій збірці (якщо вона є).

в) Налаштувати взаємодію бота з ШІ для отримання оптимальних рекомендацій.

г) Реалізувати прототип бота з простим і зручним інтерфейсом.

д) Провести тестування та оцінити ефективність бота.

Цей бот стане корисним і простим помічником для користувачів, які хочуть легко та швидко підібрати або оновити свій ПК.

2 ФУНКЦІОНАЛЬНОЇ СХЕМИ І АЛГОРИТМУ

2.1 Розробка математичних моделей

Telegram-бот «Інтелектуальний помічник для складання та модернізації ПК» є інтерактивною системою, яка дозволяє користувачеві ввести характеристики свого персонального комп’ютера та поставити запитання, пов’язане з доцільністю оновлення або придатністю системи до певних завдань. Отримані дані обробляються модулем штучного інтелекту (ШІ), який генерує відповідь.

У цьому розділі наведено формалізований опис функціонування системи, визначено її складові у вигляді математичних множин та функцій, а також представлено логіку обробки даних.

2.1.1 Основні позначення

Для формалізації логіки роботи системи та зручності подальшого опису використовуються наступні множини, функції та позначення, які відображають структуру даних і процес обробки запитів користувача.

$$U = \{u_1, u_2, \ldots, u_n\}$$

U - множина всіх користувачів системи

$$C = \{(cpu, gpu, ram, storage)\}$$

C - множина можливих конфігурацій ПК, де кожна конфігурація є кортежем із чотирьох параметрів: cpu - модель центрального процесора, gpu - модель графічного процесора, ram - обсяг оперативної пам’яті (у ГБ), storage - тип накопичувача (наприклад, HDD, SSD)

D = {"Що покращити?", "Чи піде гра?", "Який компонент найслабший?", …}

D - множина текстових запитів користувача

$$M = \{(u, c) \mid u \in U, c \in C\}$$

M - пам’ять системи, яка зберігає лише пари користувач-конфігурація

R - множина результатів, які повертає модуль ШІ у відповідь на запити. Кожен елемент $r \in R$ - це сформульована відповідь: порада, оцінка або висновок.

2.1.2 Опис функціонування системи

Користувач $u \in U$ вводить свою технічну конфігурацію $c \in C$ та текстовий запит $d \in D$.

$$M \leftarrow M \cup \{(u, c)\}$$

Система зберігає лише технічну конфігурацію у пам’яті. Запит d не зберігається.

$$AI: C \times D \longrightarrow R$$

$$r = AI(c, d), \text{ де } r \in R$$

ШІ аналізує як конфігурацію ПК, так і зміст запиту, щоби сформувати відповідь r.

$$u \longleftarrow r$$

Сформована відповідь надсилається користувачу

2.1.3 Інтерпретація як експертної системи

Система Telegram-бота функціонує як експертна система з класичною трирівневою архітектурою:

$$F = \{(u, c) \mid (u, c) \in M\}$$

Містить лише збережені дані: ID користувача та його конфігурацію.

Реалізована у вигляді логіки функції AI, яка кодує знання про сумісність компонентів, ігрові вимоги та оптимізацію. Функція AI(c, d) поєднує технічні характеристики з мовним запитом і формує релевантну відповідь на основі внутрішньої логіки ШІ.

2.2 Розробка структурної та функціональної схем програмного комплексу

Структурна схема програмного комплексу відображає взаємозв'язки між основними його компонентами. До складу системи входять такі ключові модулі:

а) Користувач Telegram - взаємодіє з ботом через інтерфейс месенджера;

б) Telegram-бот (PCBuddyBot) - реалізований з використанням aiogram, відповідає за обробку повідомлень, керування станами та генерацію відповідей;

в) Локальне сховище (user_pc_info) - зберігає інформацію про характеристики ПК користувача під час сесії;

г) Зовнішній API (intelligence.io.solutions) - використовується для надсилання характеристик ПК користувача та отримання оцінки або допомоги на основі моделей штучного інтелекту.

Функціональна схема (сценарій роботи) бота базується на поетапній взаємодії з користувачем:

а) Ініціалізація: при запуску бот відправляє привітальне повідомлення та головне меню з варіантами дій.

б) Введення характеристик ПК: Якщо дані вже існують, бот пропонує їх оновити. Інакше - запускається послідовне введення (CPU → GPU → RAM → SSD/HDD).

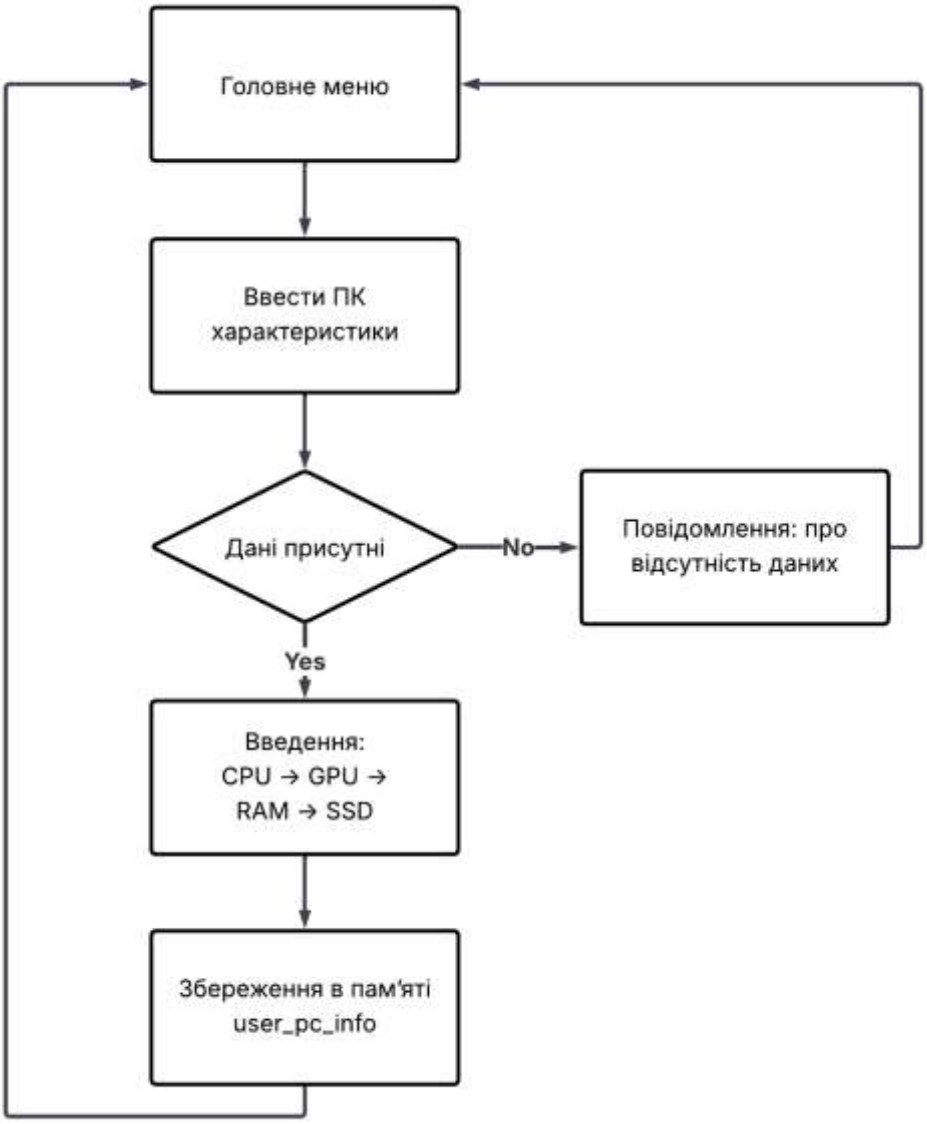

Рисунок 2.1 функціональна схема введення характеристик ПК

в) Оцінка характеристик: бот відправляє дані до API, отримує оцінку та формує коротку відповідь.

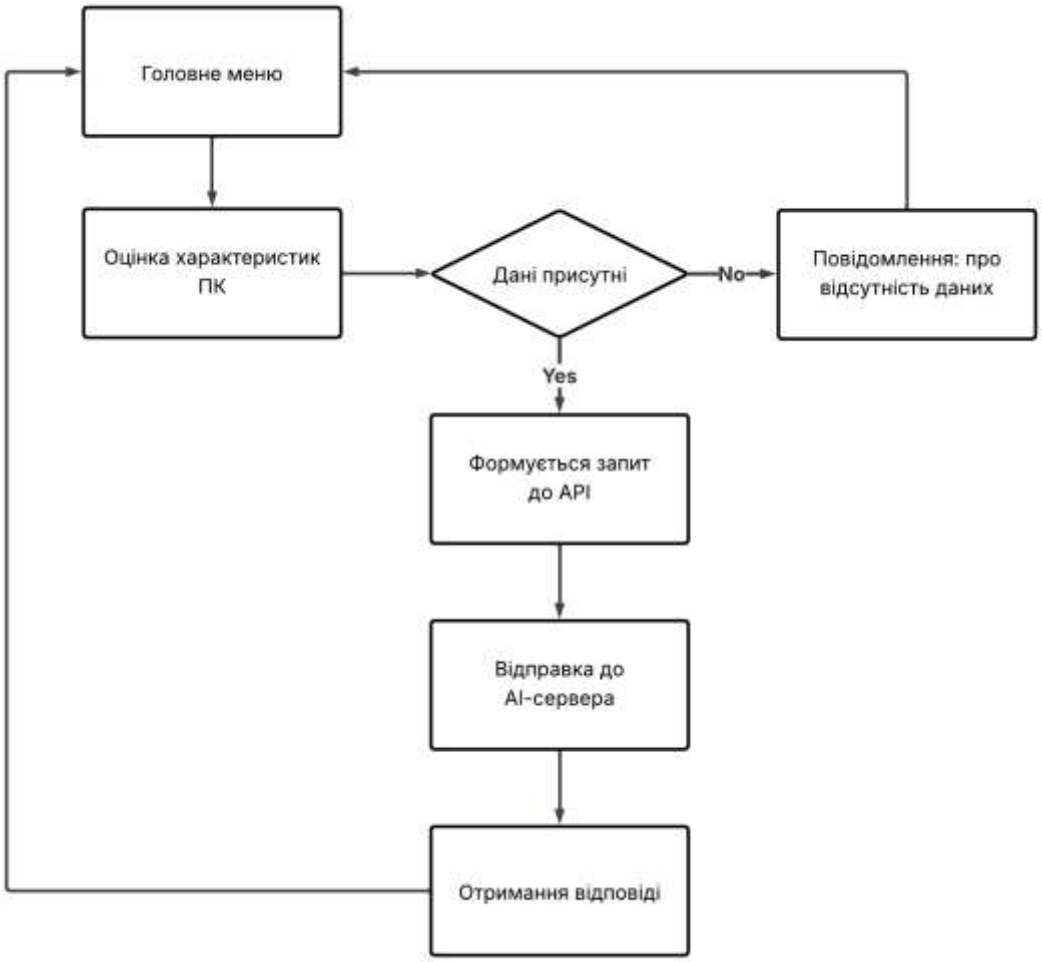

Рисунок 2.2 функціональна схема оцінки характеристик ПК

г) Оновлення окремої характеристики: користувач обирає параметр (CPU, GPU тощо), вводить нове значення, і воно зберігається.

Рисунок 2.3 функціональна схема оновлення окремої характеристики характеристик ПК

д) Видалення інформації: бот повністю очищає збережені дані користувача.

Рисунок 2.4 функціональна схема видалення інформації характеристик ПК

е) Запит довільного повідомлення: бот передає текст та наявні характеристики до зовнішнього API, який генерує відповідь згідно з інструкцією.

Рисунок 2.5 функціональна схема довільного запиту користувача

Таким чином, реалізовано чітку FSM-архітектуру з використанням aiogram.fsm, що дозволяє надійно керувати станами взаємодії. Уся логіка побудована з урахуванням зручності користувача та дотримання стандартів UI/UX у межах Telegram.

2.3 Розробка структурних схем баз даних UML

У структурній моделі даних телеграм-бота «Інтелектуальний помічник для складання та модернізації ПК» передбачено збереження базової інформації про користувача та пов’язані з ним апаратні характеристики персонального комп’ютера. Дані зберігаються у вигляді ключ-значення в оперативній пам’яті програми, однак логічна модель відповідає класичному реляційному підходу, який може бути реалізований у будь-якій СКБД.

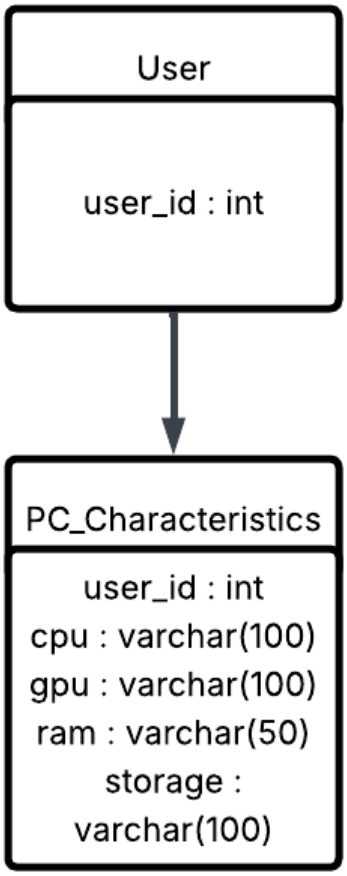

Рисунок 2.6 Структурна схема бази дани

У структурі системи передбачено дві ключові сутності - користувач та характеристики персонального комп’ютера, що зберігаються відповідно у таблицях User та PC_Characteristics. Ці сутності утворюють логічне ядро бази даних, а їхній взаємозв’язок демонструється на відповідній схемі (див. скріншоти нижче).

Таблиця User є основною - саме з неї починається будь-яка взаємодія користувача з Telegram-ботом. Вона містить унікальний ідентифікатор user_id, який автоматично зчитується з Telegram-акаунта. Це дозволяє уникнути додаткової авторизації або створення облікових записів: ідентифікація відбувається прозоро для користувача. Таким чином, кожен рядок таблиці представляє одного реального користувача.

Другу таблицю - PC_Characteristics - можна розглядати як електронну "анкету" системного блоку користувача. Вона зберігає технічні параметри, які вводяться вручну:

а) cpu - модель або назва центрального процесора,

б) gpu - модель графічного процесора (відеокарти),

в) ram - обсяг оперативної пам’яті,

г) storage - тип і обсяг накопичувача (наприклад, SSD 512 ГБ чи HDD 1 ТБ).

Кожен користувач має лише одну актуальну конфігурацію ПК. Тому між таблицями User і PC_Characteristics встановлено один-до-одного (1:1) зв’язок. Це означає, що кожен user_id в таблиці PC_Characteristics є одночасно і зовнішнім ключем, який посилається на відповідний запис у таблиці User, і первинним ключем - що гарантує унікальність записів і цілісність даних.

Такий підхід до структурування бази даних дозволяє зберігати індивідуальні апаратні характеристики для кожного користувача окремо. Він є достатньо гнучким, щоби реалізувати його як у пам’яті програми на етапі прототипування, так і у повноцінній реляційній базі даних у рамках промислової реалізації. Крім того, структура легко розширюється - наприклад, можна додати нові поля (типу ОС, монітор або блок живлення), не змінюючи логіку взаємозв’язків між таблицями.

2.4. Проектування структур таблиць бази даних і вибір типів даних полів

Структура бази даних - це основа будь-якої інформаційної системи. У межах проєкту Telegram-бота, що допомагає користувачам обирати або модернізувати ПК, було спроєктовано просту, але ефективну модель бази даних. Вона забезпечує зберігання актуальних характеристик комп'ютера для кожного користувача бота.

Модель містить дві таблиці: User та PC_Characteristics. Таблиця User зберігає унікальні Telegram ID (user_id), які бот автоматично зчитує при першій взаємодії. Це дозволяє точно ідентифікувати кожного користувача без додаткової реєстрації.

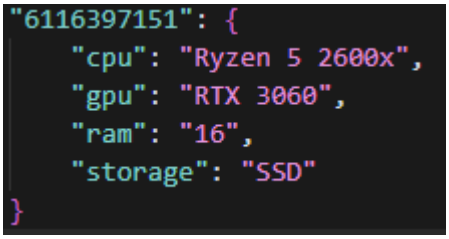

Рисунок 2.7 Приклад таблиці з даними користувачів

Таблиця PC_Characteristics фіксує поточні характеристики ПК, які користувач вводить вручну: процесор (cpu), відеокарта (gpu), оперативна пам'ять (ram) і накопичувач (storage). Кожен користувач має лише одну активну конфігурацію, тому між таблицями встановлено зв'язок «один до одного» (1:1). Для реалізації цього зв'язку поле user_id у таблиці PC_Characteristics є зовнішнім ключем, що посилається на таблицю User.

Типи даних обрано з урахуванням формату введення та вимог Telegram:

а) user_id - BIGINT, щоб зберігати великі числові значення.

б) cpu, gpu, storage - VARCHAR(100) для назв моделей.

в) ram - VARCHAR(50), що дозволяє вводити, наприклад, "16", "16 GB" або "32GB DDR4".

2.5 Розробка UML-діаграм класів програмного комплексу

На основі структурного аналізу архітектури Telegram-бота, що виконує роль інтелектуального помічника для збирання та модернізації персонального комп’ютера, було спроєктовано UML-діаграму класів. Вона наочно відображає основні компоненти системи, їх атрибути, методи та взаємозв’язки. Створення такої діаграми є важливим етапом об'єктно-орієнтованого проєктування, оскільки дозволяє формалізувати логіку взаємодії між класами й забезпечити модульність та масштабованість програмного комплексу.

Telegram-бот реалізований за допомогою Aiogram, що базується на асинхронній моделі роботи. Основні модулі розподілено за функціональними класами, кожен з яких відповідає за певний аспект функціональності бота: введення та оновлення даних користувача, зберігання станів, виклик зовнішніх API тощо.

2.5.1 Структура програмного комплексу

а) Основний модуль (файл запуску): Містить функцію main() для запуску бота, створення об'єктів Bot і Dispatcher, а також скидання webhook. Використовує глобальний словник user_pc_info для зберігання характеристик ПК кожного користувача.

б) PCInfoStates: Клас станів FSM, описує етапи поетапного введення характеристик ПК - CPU, GPU, RAM, SSD/HDD, підтвердження, оновлення.

в) Обробники повідомлень: Набір асинхронних функцій, які відповідають за взаємодію з користувачем: старт, введення, перегляд, оновлення, видалення та оцінка характеристик ПК. Включають використання FSM, клавіатури та форматування тексту.

г) Допоміжні функції: format_pc_info(info) - форматує характеристики в текст для відповіді. cancel_keyboard() - створює клавіатуру з кнопкою скасування.

д) Інтеграція з зовнішнім API: У функціях evaluate_pc_specs() та filter_messages() виконується HTTP-запит до моделі ШІ через API для оцінки або відповіді на запити користувача.

е)

2.5.1 Взаємозв’язки між компонентами

а) main() ініціалізує Bot, Dispatcher і запускає обробку - композиція.

б) Обробники використовують PCInfoStates, cancel_keyboard, format_pc_info - асоціація.

в) Оцінка характеристик викликає зовнішнє API - залежність.

г) Дані користувача зберігаються в user_pc_info - асоціація з глобальним сховищем.

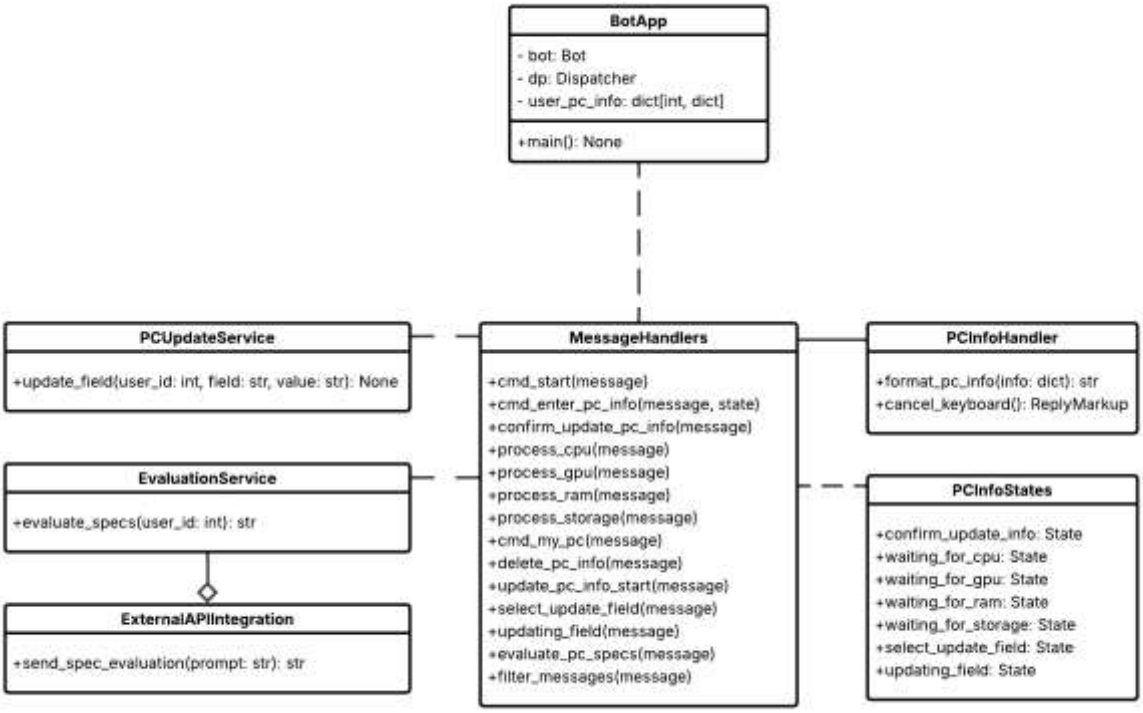

Рисунок 2.8 UML-діаграма класів

Розробка UML-діаграми класів надала змогу формалізувати внутрішню архітектуру Telegram-бота, описати логіку обробки користувацьких даних та забезпечити структурованість програмного коду. Завдяки чітко визначеним зв’язкам між компонентами досягається високий рівень модульності, що

забезпечує зручність у підтримці, тестуванні та подальшому масштабуванні програмного комплексу.

2.6 Розробка блок-схем алгоритмів програмного комплексу

Блок-схеми є важливою частиною розробки програмного забезпечення. Вони дозволяють візуально відобразити логіку роботи системи, виявити помилки ще на етапі проєктування та спростити спілкування між учасниками команди. У рамках цієї дипломної роботи блок-схеми допомогли структурувати поведінку Telegram-бота, який виконує роль інтелектуального помічника користувача при роботі з конфігурацією персонального комп’ютера.

Основна функціональність бота охоплює такі сценарії: введення, оновлення, перегляд і видалення характеристик ПК, а також оцінка конфігурації за допомогою зовнішнього API.

2.6.1 Введення характеристик ПК

Сценарій починається після натискання кнопки «Ввести характеристики ПК». У функції cmd_enter_pc_info бот перевіряє, чи є у користувача збережені характеристики. Якщо так - через PCInfoStates.confirm_update_info бот уточнює, чи потрібно їх оновити. Якщо ні - запускається покрокове введення: CPU → GPU → RAM → SSD/HDD. Кожен етап обробляється окремими функціями (process_cpu, process_gpu тощо), а користувач має можливість скасувати процес. По завершенню дані зберігаються у внутрішньому словнику user_pc_info, а бот повідомляє про успішне збереження.

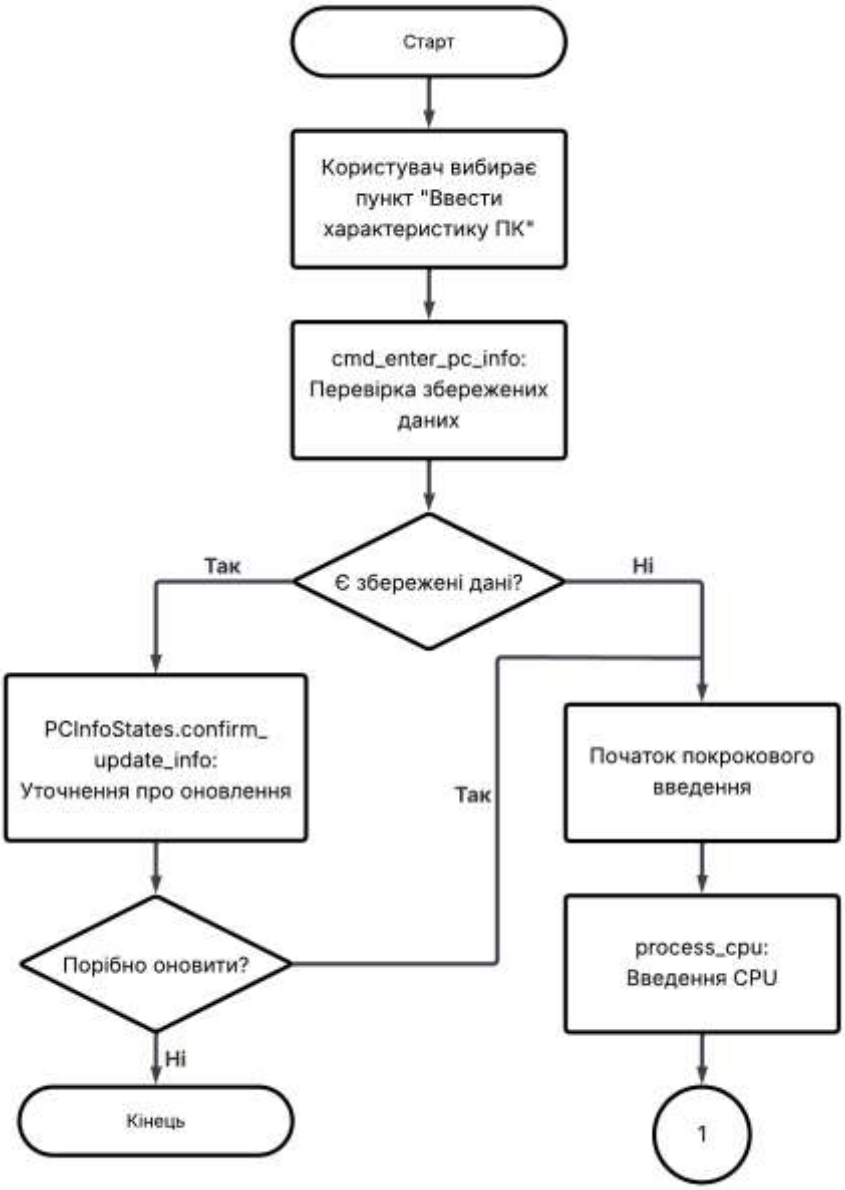

Рисунок 2.9 Блок-схема введення характеристик ПК (частина 1)

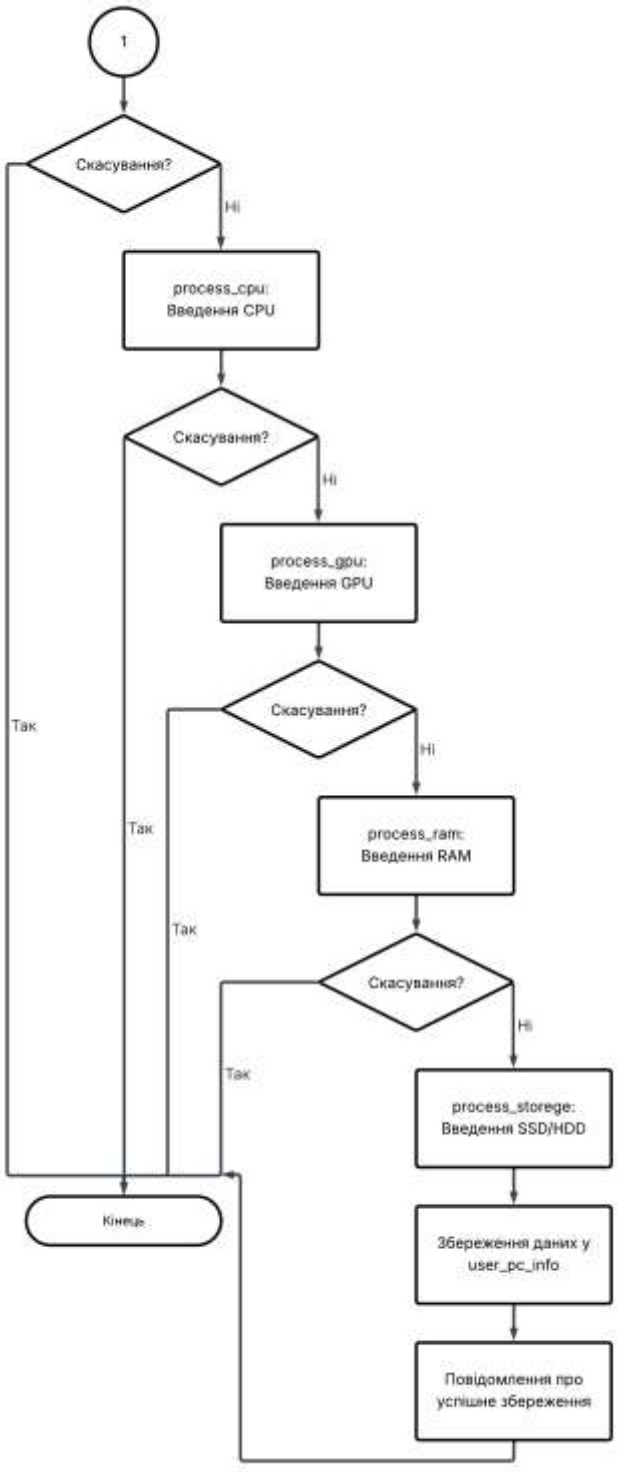

Рисунок 2.10 Блок-схема введення характеристик ПК (частина 2)

2.6.2 Оновлення окремої характеристики

Після вибору пункту «Оновити характеристику» запускається функція update_pc_info_start, яка запитує в користувача, що саме він хоче змінити. Далі

функція select_update_field перевіряє введене значення, і якщо воно коректне - передає керування у updating_field, де приймається нове значення. У разі успішного оновлення бот повідомляє користувача про завершення дії. Якщо користувач вводить недійсну характеристику або скасовує процес - бот відповідно реагує.

Рисунок 2.11 Блок-схема оновлення окремої характеристики

2.6.3 Оцінка конфігурації ПК

Цей сценарій обробляється у функції evaluate_pc_specs. Після натискання кнопки «Оцінка характеристик ПК» бот перевіряє, чи є в користувача збережені дані. Якщо так - формує запит із вказаними

характеристиками (CPU, GPU, RAM, SSD/HDD) у вигляді тексту, який надсилається до зовнішнього API. Отримана відповідь коротко узагальнюється і надсилається користувачу у зручному форматі. Якщо дані відсутні, бот пропонує їх спочатку ввести.

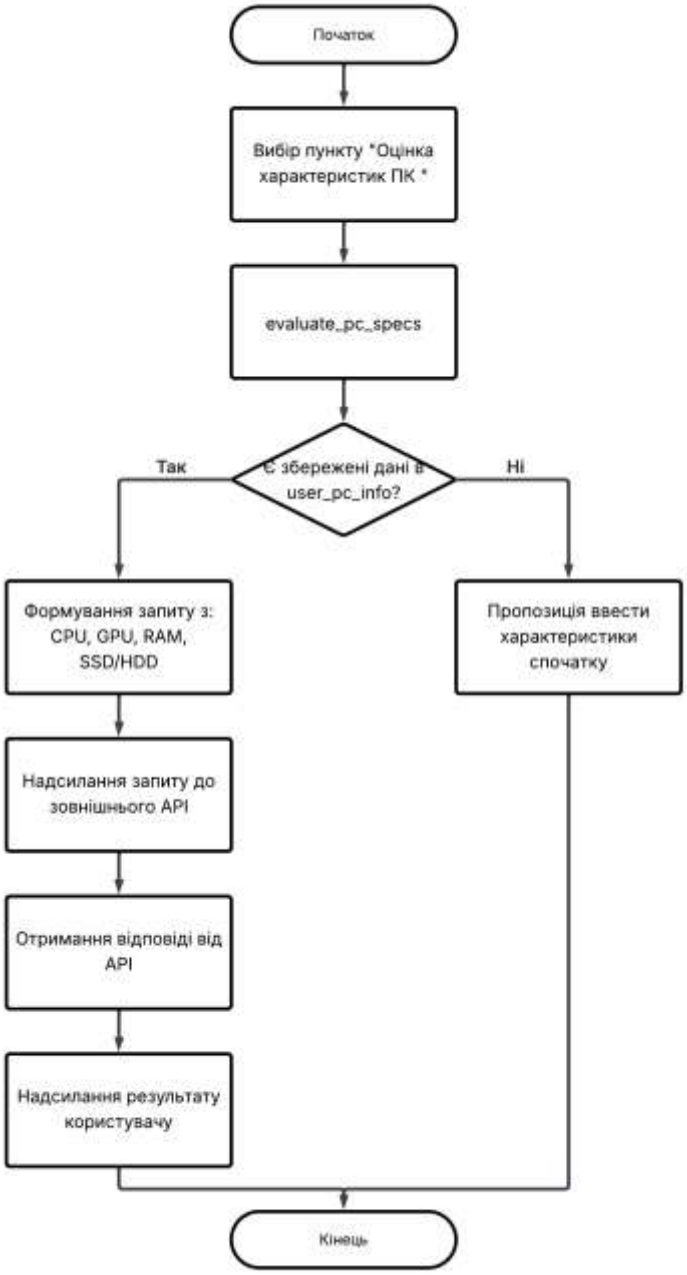

Рисунок 2.12 Блок-схема оцінки конфігурації ПК

2.6.4 Обробка довільних повідомлень

Функція filter_messages відповідає за всі інші повідомлення, що не є командами або натисканням кнопок. Бот перевіряє, чи є збережені дані, і якщо так - включає їх у запит до API. У відповідь користувач отримує релевантну інформацію, яка враховує його конфігурацію ПК. Якщо ж повідомлення не стосується теми комп’ютерів, бот ввічливо просить сформулювати запит в межах дозволеної тематики.

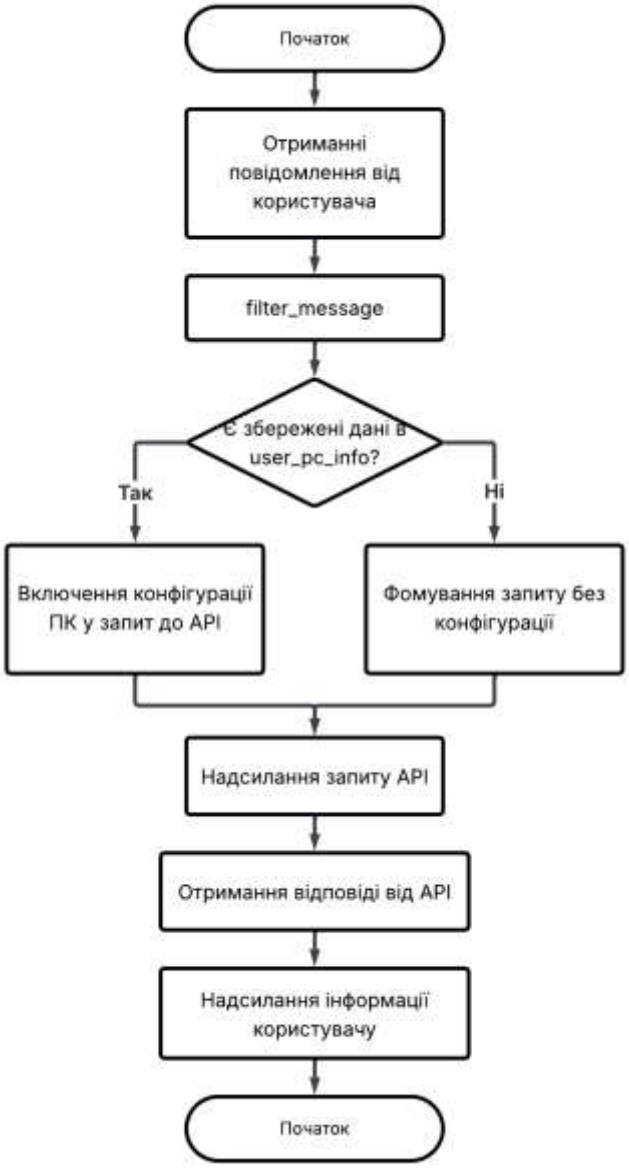

Рисунок 2.13 Блок-схема обробки довільних повідомлень

2.7 Розробка моделі візуального інтерфейсу програмного комплексу

Візуальний інтерфейс користувача Telegram-бота реалізовано за допомогою стандартних інструментів месенджера: кнопок швидкого доступу (ReplyKeyboardMarkup) та текстових повідомлень. Мета інтерфейсу - забезпечити просту та інтуїтивно зрозумілу взаємодію користувача з програмним комплексом. Основні елементи інтерфейсу:

а) «Мій ПК» – перегляд збережених характеристик;

б) «Ввести характеристики ПК» – покрокове введення;

в) «Оновити характеристику» – зміна одного з параметрів;

г) «Видалити характеристики»;

д) «Оцінка характеристик ПК».

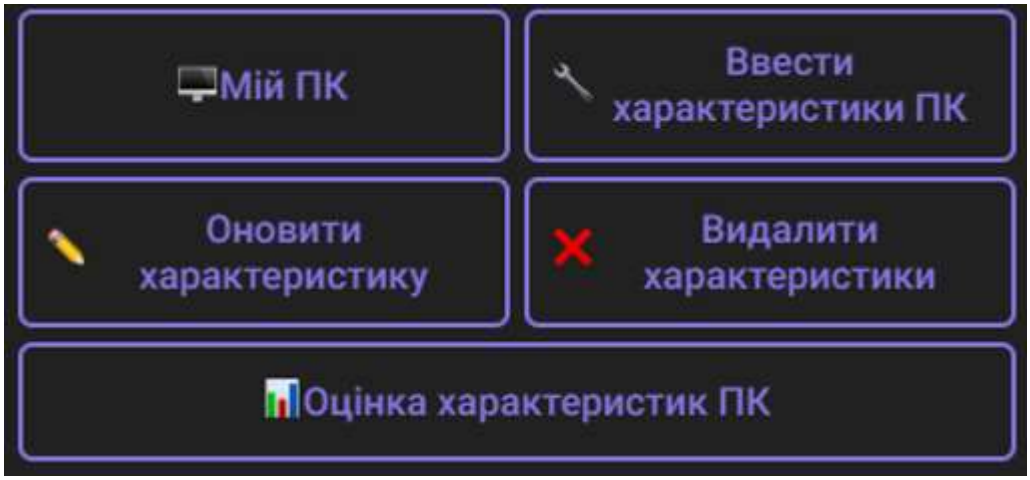

Рисунок 2.14 Головна клавіатура меню

Діалог введення характеристик: запускається при першому використанні або оновленні. Користувач поетапно вводить CPU, GPU, RAM і SSD/HDD. На кожному кроці доступна кнопка скасування.

Оновлення окремої характеристики: після вибору бот уточнює, який саме компонент слід оновити. Далі користувач вводить нове значення, яке зберігається у даних.

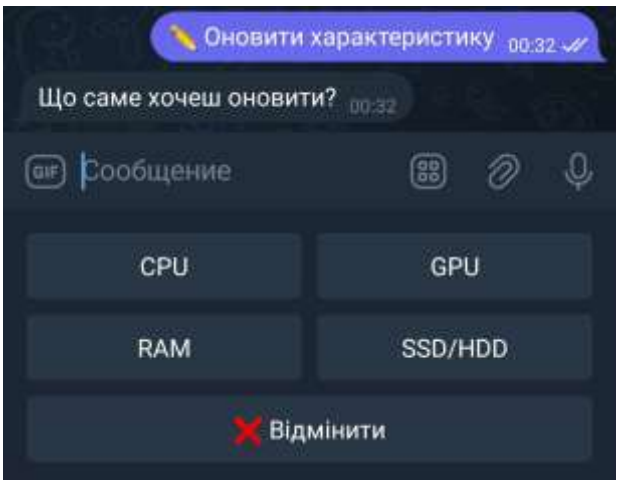

Рисунок 2.15 Оновлення окремої характеристики

Підтвердження дій: якщо користувач вже має збережені дані, бот пропонує підтвердити бажання їх перезаписати («Так» / «Ні»).

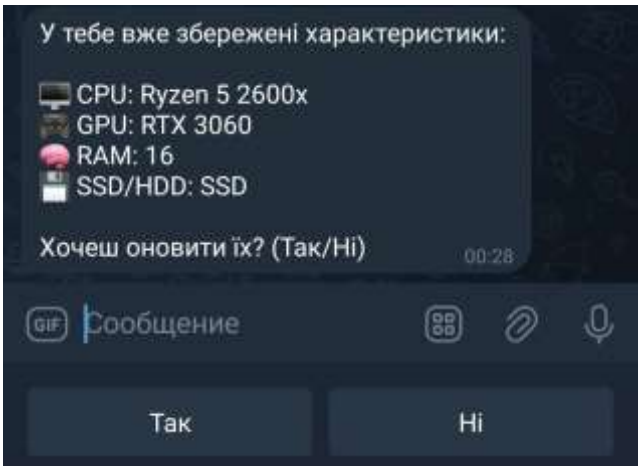

Рисунок 2.16 Підтвердження дій

Вивід результатів оцінки: після надсилання запиту на аналіз бот виводить сформовану відповідь від зовнішнього API у вигляді тексту.

Повідомлення про помилки або інструкції: у випадку помилки бот надає чіткі підказки (наприклад, про неправильний формат вводу або відсутність характеристик).

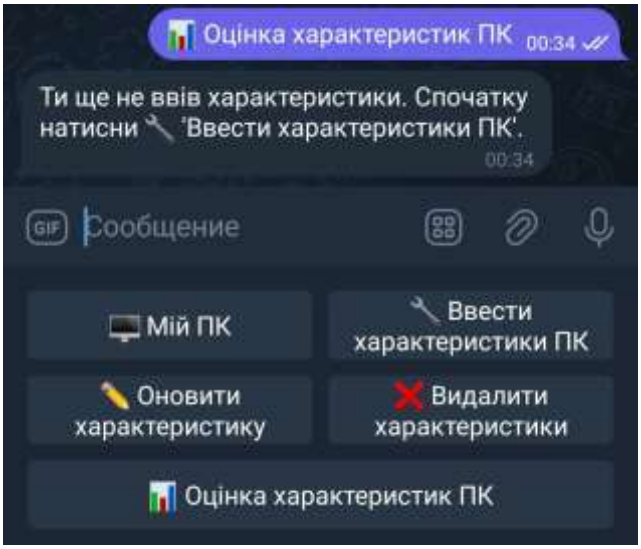

Рисунок 2.17 Повідомлення про помилки

2.7.1 Особливості реалізації

а) Інтерфейс побудований на основі FSM-моделі (машини станів), що дозволяє адаптувати логіку діалогу під поточний стан користувача.

б) Відповіді максимально стислі та структуровані, що відповідає особливостям мобільної платформи Telegram.

в) Користувач має можливість швидко повернутися до головного меню або перервати взаємодію в будь-який момент.

Інтерфейс Telegram-бота побудовано на принципах простоти, послідовності та логічності. Застосування клавіатур і поетапного введення забезпечує зручну взаємодію, навіть у межах текстового середовища. Реалізоване рішення є ефективним для мобільного використання й не потребує графічного інтерфейсу.

3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ПРОЕКТУ

3.1 Розробка основних класів програмних модулів проекту

У процесі створення Telegram-бота «Інтелектуальний помічник для складання та модернізації ПК» було реалізовано низку важливих компонентів, що формують основу його логіки. Архітектура базується на принципах модульності, чіткого розділення відповідальностей (SRP) та простоти подальшого супроводу.

Одним із ключових елементів став клас PCInfoStates. Він визначає всі стани finite state machine (FSM), що використовується ботом для поетапного збору або оновлення інформації про комп'ютер. Завдяки цьому підходу бот не «губиться» у діалозі, а чітко знає, на якому кроці взаємодії перебуває користувач. Наприклад, стани waiting_for_cpu, waiting_for_gpu, waiting_for_ram та waiting_for_storage відповідають очікуванню відповідної характеристики. Є також проміжні стани - confirm_update_info для підтвердження оновлення і select_update_field та updating_field для точкових змін.

Для тимчасового зберігання інформації використовується глобальний словник user_pc_info. Він не є класом у класичному розумінні, але виконує надзвичайно важливу роль - фіксує характеристики кожного користувача за його user_id. Структура проста: зберігається словник із параметрами ПК: CPU, GPU, RAM і накопичувач.

Користувацький інтерфейс реалізовано через змінну main_keyboard, що є об'єктом класу ReplyKeyboardMarkup. Це головне меню, яке бачить користувач одразу після запуску бота. У ньому - три логічних ряди кнопок, що дозволяють легко отримати доступ до основних функцій: перегляд характеристик, їх введення, оновлення, видалення та оцінка. Кожна з цих кнопок ініціює певну логіку через обробники повідомлень.

Основна робота з ботом реалізована у вигляді функцій-хендлерів, прив'язаних до тексту повідомлень або кнопок. Вони відповідають за конкретні сценарії. Наприклад, cmd_start() виводить вітання та головне меню,

cmd_enter_pc_info() запускає процес введення характеристик, а функції process_cpu(), process_gpu() тощо - обробляють кожен етап цього процесу. Для оновлення існує окрема гілка: update_pc_info_start() ініціює зміну, select_update_field() дає обрати характеристику, а updating_field() приймає нове значення. Функція evaluate_pc_specs() звертається до зовнішнього API для аналізу конфігурації, а filter_messages() опрацьовує всі інші повідомлення - вона діє як "розумна" відповідь на вільний текст користувача, з урахуванням вже введених параметрів ПК.

Крім основної логіки, у боті є також декілька допоміжних функцій. Наприклад, format_pc_info() відповідає за те, щоб красиво і зрозуміло оформити технічну інформацію у вигляді повідомлення. А cancel_keyboard() створює просту клавіатуру з кнопкою «Відмінити», яка використовується у всіх випадках, коли користувач має змогу зупинити дію.

Загалом, кожен компонент у системі виконує свою чітку роль. Така структура дозволяє зручно масштабувати бота, швидко вносити зміни, а також легко зрозуміти логіку його роботи навіть новому розробнику.

3.2 Розробка візуального інтерфейсу програмного комплексу

Візуальний інтерфейс програмного комплексу реалізовано у вигляді текстового та кнопкового середовища, що працює безпосередньо у вікні Telegram. Для цього використовуються можливості бібліотеки Aiogram, зокрема система ReplyKeyboardMarkup, яка дозволяє створювати інтерактивні клавіатури прямо в чаті. Такий підхід дає змогу повністю відмовитися від класичного графічного інтерфейсу, зберігаючи при цьому зручність та інтуїтивну зрозумілість взаємодії.

Основною ідеєю при створенні інтерфейсу була проста й послідовна логіка, зрозуміла навіть недосвідченому користувачеві. Після запуску бота або завершення певної дії на екрані з'являється головна клавіатура. Вона містить всі необхідні кнопки: для перегляду характеристик комп’ютера, їх введення, оновлення, видалення, а також відправлення на оцінку. Кожна кнопка

відкриває відповідний сценарій, і користувач завжди бачить, що саме він може зробити на кожному етапі.

Введення інформації про ПК реалізовано покроково. Бот чітко формулює запит: спочатку просить вказати процесор, потім відеокарту, далі оперативну пам’ять та накопичувач. Користувач у будь-який момент може скасувати дію, натиснувши кнопку «Відмінити». Це значно знижує ризик помилок та підвищує зручність.

Оновлення даних відбувається подібним чином. Спочатку бот запитує, який саме параметр потрібно змінити (наприклад, GPU або RAM), а потім приймає нове значення. Якщо користувач вводить некоректну відповідь або обирає невідповідну опцію, система надає підказку або повідомлення про помилку з поясненням, як діяти далі.

Всі результати та повідомлення бот формулює у текстовому форматі - це можуть бути підтвердження успішного збереження даних, попередження про відсутність введеної інформації, або відповідь від зовнішнього API після оцінки конфігурації.

Окремої уваги заслуговує механізм роботи FSM (finite state machine) - саме завдяки йому бот "пам’ятає", на якому кроці зараз знаходиться користувач, і реагує відповідно. Це дозволяє реалізувати логічно цілісні й контрольовані сценарії, де кожна дія веде до очікуваного результату.

Інтерфейс Telegram-бота вийшов простим, логічним і мобільним. Він не перевантажений зайвими елементами, всі дії виконуються через зрозумілі кнопки та короткі текстові інструкції. У результаті навіть користувач без спеціальних знань легко орієнтується в системі та може повноцінно з нею працювати.

ВИСНОВКИ

Під час виконання цієї дипломної роботи було реалізовано Telegram-бота, який допомагає користувачам підбирати комплектуючі для персонального комп’ютера, оцінювати вже наявні конфігурації та отримувати поради щодо оновлення системи. Основна ідея полягала в тому, щоб зробити процес підбору максимально зручним, простим і доступним для кожного - навіть для тих, хто не має спеціальних технічних знань.

Розробка охопила не лише програмування функціоналу, а й моделювання логіки взаємодії з користувачем, розробку структури даних, інтеграцію зі штучним інтелектом та побудову інтерфейсу у форматі діалогу. Користувачі можуть вводити характеристики свого ПК, редагувати їх, отримувати оцінку продуктивності - і все це без складних меню, без зайвих налаштувань, просто через звичайний чат у Telegram.

Проєкт вийшов функціональним і досить гнучким. Його структура дозволяє легко розширювати можливості в майбутньому - наприклад, підключити базу цін, виводити візуальні графіки або зробити повноцінну інтеграцію з магазинами.

Робота над ботом підтвердила, що сучасні інструменти - такі як штучний інтелект, месенджери та FSM - дають змогу створювати справді корисні продукти, здатні автоматизувати складні процеси й зробити їх доступними для широкої аудиторії. І якщо хоча б декільком людям цей бот допоможе зібрати свій перший ПК або уникнути помилки при покупці - значить, ця робота була виконана не даремно.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

IO.net. Децентралізована хмарна інфраструктура для ШІ [Електронний ресурс]. – Режим доступу: https://id.io.net

Aiogram Documentation. Офіційна документація бібліотеки для створення Telegram-ботів на Python [Електронний ресурс]. – Режим доступу: https://docs.aiogram.dev

Telegram Bot API. Офіційна документація Telegram API для ботів [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api

BuildMyPC – Інтелектуальний сервіс для складання ПК [Електронний ресурс]. – Режим доступу: https://buildmypc.net

PCPartPicker – Сервіс добору сумісних компонентів для ПК [Електронний ресурс]. – Режим доступу: https://pcpartpicker.com

Мэтиз Е. *Изучаем Python* [Електронний ресурс]. – Режим доступу: https://homeread.net/book/izuchaem-python-erik-metiz#tx

How-To Geek. 10 Mistakes Beginners Make When Building PCs [Електронний ресурс]. – Режим доступу: https://www.howtogeek.com/882866/beginner-pc-building-mistakes

GamersNexus. Common PC Build Mistakes – System Build Preparation [Електронний ресурс]. – Режим доступу: https://gamersnexus.net/guides/834-common-pc-building-mistakes

ПОВНИЙ ТЕКСТ ПРОГРАМНИХ МОДУЛІВ

Файл config.py

```
from typing import Dict, List
BOT_TOKEN = '7949627803:AAHedOEqRneykHhW7ECMX_5Bl8xQ7N5CwKg'
API_URL =
"https://api.intelligence.io.solutions/api/v1/chat/completions"
API_KEY = "Bearer io-v2-
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJvd25lciI6IjAzZWRmOTk3LTZ
jYmMtNDZlZS1iOWZiLWNmY2Y1YjZmYjQ2MCIsImV4cCI6NDg5OTQxODA1MH0.iIG
k_FzytA9Fl_6tgfhHWuTk2iGzm_s2l2YEno5WOgA7STlRcIkqqtL6LJO44ZFMwSK
9hRpw8UeqNMM9ZcdxOQ"
MAIN_KEYBOARD_BUTTONS: List[List[str]] = [
    ["📄 Мій ПК", "🔧 Ввести характеристики ПК"],
    ["➖ Оновити характеристику", "❌ Видалити
характеристики"],
    ["📊 Оцінка характеристик ПК"]
]
WELCOME_MESSAGE = '''Привіт! Я PCBuddyBot — твій надійний
помічник у збиранні та модернізації комп'ютера.
Разом ми підберемо найкращі комплектуючі для твоїх завдань
або покращимо твій ПК.
Я допоможу зробити правильний вибір навіть тоді, коли ти не
розумієшся на "залізі".
Обери дію нижче! 🚀'''
EVALUATION_PROMPT = """Ти телеграм-бот, який оцінює ПК
користувача. Відповідай коротко і українською."""
GENERAL_PROMPT = """Ти телеграм-бот, який допомагає
користувачам з ПК. Відповідай коротко і українською. На інші
теми тобі відповідати не можна,треба просити задати питання
пов'язане с ПК, модернізацією та подібним. Якщо в тебе будуть
просити файл, не кидай його, кажи, що ти ограниченый в токенах и
не можешь это делать"""

THINKING_MESSAGES: List[str] = [
```

```
        "🤔 Думаю...",
        "🌪️ Обдумую...",
        "⏳ Момент...",
        "💡 Формую відповідь..."
    ]
```

Файл data_manager.py

```
    import json
    import os
    from typing import Dict, Optional
    import logging
    logger = logging.getLogger(__name__)
    class DataManager:
    def __init__(self, file_path: str = "user_data.json"):
    self.file_path = file_path
    self.data: Dict[int, Dict[str, str]] = {}
    self.load_data()
    def load_data(self) -> None:
    try:
    if os.path.exists(self.file_path):
    with open(self.file_path, 'r', encoding='utf-8') as file:
    self.data = json.load(file)
    logger.info(f"Данные успешно загружены из {self.file_path}")
    else:
    self.data = {}
    logger.info(f"Файл {self.file_path} не найден, создан новый
словарь данных")
    except Exception as e:
    logger.error(f"Ошибка при загрузке данных: {e}")
    self.data = {}
    def save_data(self) -> None:
    try:
    with open(self.file_path, 'w', encoding='utf-8') as file:
    json.dump(self.data, file, ensure_ascii=False, indent=4)
    logger.info(f"Данные успешно сохранены в {self.file_path}")
    except Exception as e:
```

```
    logger.error(f"Ошибка при сохранении данных: {e}")
    def get_user_data(self, user_id: int) -> Optional[Dict[str,
str]]:
    return self.data.get(str(user_id))
    def save_user_data(self, user_id: int, data: Dict[str, str])
-> None:
    self.data[str(user_id)] = data
    self.save_data()
    def delete_user_data(self, user_id: int) -> bool:
    if str(user_id) in self.data:
    del self.data[str(user_id)]
    self.save_data()
    return True
    return False
    def get_all_users(self) -> Dict[int, Dict[str, str]]:
    return {int(k): v for k, v in self.data.items()}
```

Файл main.py

```
    import asyncio
    import logging
    import random
    from typing import Dict, Optional, List
    from aiogram import Bot, Dispatcher, types
    from aiogram.filters import CommandStart
    from aiogram.methods import DeleteWebhook
    from aiogram.types import Message, ReplyKeyboardMarkup,
KeyboardButton, ReplyKeyboardRemove
    from aiogram.fsm.context import FSMContext
    from aiogram.fsm.state import StatesGroup, State
    import requests
    from config import (
        BOT_TOKEN, API_URL, API_KEY, MAIN_KEYBOARD_BUTTONS,
        WELCOME_MESSAGE, EVALUATION_PROMPT, GENERAL_PROMPT,
        THINKING_MESSAGES
    )
    from data_manager import DataManager
```

```
    logging.basicConfig(
        level=logging.INFO,
        format='%(asctime)s - %(name)s - %(levelname)s -
%(message)s'
    )
    logger = logging.getLogger(__name__)
    bot = Bot(BOT_TOKEN)
    dp = Dispatcher()
    data_manager = DataManager()
    class PCInfoStates(StatesGroup):
        confirm_update_info = State()
        waiting_for_cpu = State()
        waiting_for_gpu = State()
        waiting_for_ram = State()
        waiting_for_storage = State()
        select_update_field = State()
        updating_field = State()
    user_used_messages: Dict[int, List[str]] = {}
    def get_main_keyboard() -> ReplyKeyboardMarkup:
        return ReplyKeyboardMarkup(
            keyboard=[[KeyboardButton(text=btn) for btn in row]
for row in MAIN_KEYBOARD_BUTTONS],
            resize_keyboard=True
        )
    def get_cancel_keyboard() -> ReplyKeyboardMarkup:
        return ReplyKeyboardMarkup(
            keyboard=[[KeyboardButton(text="❌ Відмінити")]],
            resize_keyboard=True
        )
    def format_pc_info(info: Dict[str, str]) -> str:
        return (
            f"🖥 CPU: {info.get('cpu', 'Не вказано')}\n"
            f"🎮 GPU: {info.get('gpu', 'Не вказано')}\n"
            f"🧠 RAM: {info.get('ram', 'Не вказано')}\n"
            f"💾 SSD/HDD: {info.get('storage', 'Не вказано')}"
```

```
        )
    def get_random_thinking_message(user_id: int) -> str:
        if user_id not in user_used_messages:
            user_used_messages[user_id] = []
        available_messages = [msg for msg in THINKING_MESSAGES
if msg not in user_used_messages[user_id]]

        if not available_messages:
            user_used_messages[user_id] = []
            available_messages = THINKING_MESSAGES
        message = random.choice(available_messages)
        user_used_messages[user_id].append(message)
        return message
    async def make_api_request(system_prompt: str, user_message:
str, user_id: int) -> str:
        try:
            await asyncio.sleep(random.randint(2, 4))
            thinking_message =
get_random_thinking_message(user_id)
            thinking_msg = await bot.send_message(user_id,
thinking_message)
            headers = {
                "Content-Type": "application/json",
                "Authorization": API_KEY
            }
            data = {
                "model": "deepseek-ai/DeepSeek-R1",
                "messages": [
                    {"role": "system", "content":
system_prompt},
                    {"role": "user", "content": user_message}
                ]
            }
            response = requests.post(API_URL, headers=headers,
json=data)
            response.raise_for_status()
```

```
            data = response.json()
            text = data['choices'][0]['message']['content']
            await bot.delete_message(chat_id=user_id,
message_id=thinking_msg.message_id)
            return text.split('</think>\n\n')[-1]
        except Exception as e:
            logger.error(f"API request error: {e}")
            return "Вибачте, сталася помилка при обробці запиту.
Спробуйте пізніше."
    @dp.message(CommandStart())
    async def cmd_start(message: Message):
        user = message.from_user
        logger.info(f"Пользователь {user.full_name} (ID:
{user.id}) запустил бота")
        await message.answer(WELCOME_MESSAGE, parse_mode='HTML',
reply_markup=get_main_keyboard())
    @dp.message(lambda message: message.text == "🔧 Ввести
характеристики ПК")
    async def cmd_enter_pc_info(message: Message, state:
FSMContext):
        user = message.from_user
        user_id = user.id
        pc_info = data_manager.get_user_data(user_id)
        logger.info(f"Пользователь {user.full_name} (ID:
{user_id}) начал ввод характеристик ПК")
        if pc_info:
            await message.answer(
                f"У тебе вже збережені
характеристики:\n\n{format_pc_info(pc_info)}\n\n"
                "Хочеш оновити їх? (Так/Ні)",
                reply_markup=ReplyKeyboardMarkup(
                    keyboard=[[KeyboardButton(text="Так"),
KeyboardButton(text="Ні")]],
                    resize_keyboard=True
                )
```

```
            )
            await
state.set_state(PCInfoStates.confirm_update_info)
        else:
            await message.answer("Почнемо! Введи, будь ласка,
свій процесор (CPU):", reply_markup=ReplyKeyboardRemove())
            await state.set_state(PCInfoStates.waiting_for_cpu)
    @dp.message(lambda message: message.text == "📊 Оцінка
характеристик ПК")
    async def evaluate_pc_specs(message: Message):
        user = message.from_user
        user_id = user.id
        pc_info = data_manager.get_user_data(user_id)
        logger.info(f"Пользователь {user.full_name} (ID:
{user_id}) запросил оценку характеристик ПК")
        if not pc_info:
            await message.answer("Ти ще не ввів характеристики.
Спочатку натисни 🔧 'Ввести характеристики ПК'.")
            return
        system_prompt = (
            f"{EVALUATION_PROMPT}\n\nХарактеристики
користувача:\n"
            f"CPU: {pc_info.get('cpu', 'Не вказано')}\n"
            f"GPU: {pc_info.get('gpu', 'Не вказано')}\n"
            f"RAM: {pc_info.get('ram', 'Не вказано')}\n"
            f"SSD/HDD: {pc_info.get('storage', 'Не вказано')}\n"
        )
        response = await make_api_request(system_prompt,
message.text, user_id)
        await message.answer(response, parse_mode="Markdown")
    @dp.message(PCInfoStates.confirm_update_info)
    async def confirm_update_pc_info(message: Message, state:
FSMContext):
        if message.text.lower() == "так":
```

```
            await message.answer("Введи свій процесор (CPU):",
reply_markup=ReplyKeyboardRemove())
            await state.set_state(PCInfoStates.waiting_for_cpu)
        else:
            await message.answer("Добре, залишаємо старі
характеристики ✅", reply_markup=get_main_keyboard())
            await state.clear()
    @dp.message(PCInfoStates.waiting_for_cpu)
    async def process_cpu(message: Message, state: FSMContext):
        if message.text == "❌ Відмінити":
            await message.answer("Ввід характеристик скасовано
❌", reply_markup=get_main_keyboard())
            await state.clear()
            return
        await state.update_data(cpu=message.text)
        await message.answer("Тепер введи свою відеокарту
(GPU):", reply_markup=get_cancel_keyboard())
        await state.set_state(PCInfoStates.waiting_for_gpu)
    @dp.message(PCInfoStates.waiting_for_gpu)
    async def process_gpu(message: Message, state: FSMContext):
        if message.text == "❌ Відмінити":
            await message.answer("Ввід характеристик скасовано
❌", reply_markup=get_main_keyboard())
            await state.clear()
            return
        await state.update_data(gpu=message.text)
        await message.answer("Введи об'єм оперативної пам'яті в
gb(RAM):", reply_markup=get_cancel_keyboard())
        await state.set_state(PCInfoStates.waiting_for_ram)
    @dp.message(PCInfoStates.waiting_for_ram)
    async def process_ram(message: Message, state: FSMContext):
        if message.text == "❌ Відмінити":
            await message.answer("Ввід характеристик скасовано
❌", reply_markup=get_main_keyboard())
```

```
            await state.clear()
            return
        await state.update_data(ram=message.text)
        await message.answer("Введи свій накопичувач
(SSD/HDD):", reply_markup=get_cancel_keyboard())
        await state.set_state(PCInfoStates.waiting_for_storage)
    @dp.message(PCInfoStates.waiting_for_storage)
    async def process_storage(message: Message, state:
FSMContext):
        user = message.from_user
        if message.text == "❌ Відмінити":
            logger.info(f"Пользователь {user.full_name} (ID:
{user.id}) отменил ввод характеристик")
            await message.answer("Ввід характеристик скасовано
❌", reply_markup=get_main_keyboard())
            await state.clear()
            return
        data = await state.get_data()
        data['storage'] = message.text
        data_manager.save_user_data(user.id, data)
        logger.info(f"Пользователь {user.full_name} (ID:
{user.id}) сохранил характеристики ПК: {data}")
        await message.answer("Характеристики успішно збережені
✅", reply_markup=get_main_keyboard())
        await state.clear()
    @dp.message(lambda message: message.text == "📄 Мій ПК")
    async def cmd_my_pc(message: Message):

        user = message.from_user
        info = data_manager.get_user_data(user.id)
        logger.info(f"Пользователь {user.full_name} (ID:
{user.id}) запросил просмотр характеристик ПК")
        if info:
            await message.answer(f"Ось твої характеристики
ПК:\n\n{format_pc_info(info)}")
```

```
        else:
            await message.answer("Ти ще не ввів характеристики.
Натисни 🔧 'Ввести характеристики ПК' щоб додати їх!")
    @dp.message(lambda message: message.text == "❌ Видалити
характеристики")
    async def delete_pc_info(message: Message):
        user = message.from_user
        if data_manager.delete_user_data(user.id):
            logger.info(f"Пользователь {user.full_name} (ID:
{user.id}) удалил характеристики ПК")
            await message.answer("Твої характеристики успішно
видалені ❌", reply_markup=get_main_keyboard())
        else:
            logger.info(f"Пользователь {user.full_name} (ID:
{user.id}) попытался удалить несуществующие характеристики")
            await message.answer("Немає характеристик для
видалення!", reply_markup=get_main_keyboard())
    @dp.message(lambda message: message.text == "➖ Оновити
характеристику")
    async def update_pc_info_start(message: Message, state:
FSMContext):
        user_id = message.from_user.id
        pc_info = data_manager.get_user_data(user_id)
        if not pc_info:
            await message.answer("Ти ще не ввів характеристики.
Натисни 🔧 'Ввести характеристики ПК'!")
            return
        keyboard = ReplyKeyboardMarkup(
            keyboard=[
                [KeyboardButton(text="CPU"),
KeyboardButton(text="GPU")],
                [KeyboardButton(text="RAM"),
KeyboardButton(text="SSD/HDD")],
                [KeyboardButton(text="❌ Відмінити")]
```

```
            ],
            resize_keyboard=True
        )
        await message.answer(
            "Що саме хочеш оновити?",
            reply_markup=keyboard
        )
        await state.set_state(PCInfoStates.select_update_field)

    @dp.message(PCInfoStates.select_update_field)
    async def select_update_field(message: Message, state:
FSMContext):
        if message.text == "❌ Відмінити":
            await message.answer("Оновлення скасовано ❌",
reply_markup=get_main_keyboard())
            await state.clear()
            return
        valid_choices = ["CPU", "GPU", "RAM", "SSD/HDD"]
        if message.text not in valid_choices:
            keyboard = ReplyKeyboardMarkup(
                keyboard=[
                    [KeyboardButton(text="CPU"),
KeyboardButton(text="GPU")],
                    [KeyboardButton(text="RAM"),
KeyboardButton(text="SSD/HDD")],
                    [KeyboardButton(text="❌ Відмінити")]
                ],
                resize_keyboard=True
            )
            await message.answer(
                "Будь ласка, використовуй кнопки для вибору
характеристики!",
                reply_markup=keyboard
            )
            return
```

```
        field = message.text.lower()
        await state.update_data(field_to_update=field)
        await message.answer(
            f"Введи нове значення для {message.text}:",
            reply_markup=get_cancel_keyboard()
        )
        await state.set_state(PCInfoStates.updating_field)
    @dp.message(PCInfoStates.updating_field)
    async def updating_field(message: Message, state:
FSMContext):
        if message.text == "❌ Відмінити":
            await message.answer("Оновлення скасовано ❌",
reply_markup=get_main_keyboard())
            await state.clear()
            return
        data = await state.get_data()
        field = data['field_to_update']
        user_id = message.from_user.id

        pc_info = data_manager.get_user_data(user_id)
        if pc_info:
            pc_info[field] = message.text
            data_manager.save_user_data(user_id, pc_info)
            await message.answer(
                f"Характеристика {field.upper()} успішно
оновлена ✅",
                reply_markup=get_main_keyboard()
            )
        else:
            await message.answer(
                "Трапилась помилка! Почни спочатку.",
                reply_markup=get_main_keyboard()
            )
        await state.clear()
    @dp.message()
```

```
    async def filter_messages(message: Message):
        user = message.from_user
        user_id = user.id
        pc_info = data_manager.get_user_data(user_id)
        logger.info(f"Пользователь {user.full_name} (ID:
{user_id}) отправил сообщение: {message.text}")

        system_prompt = GENERAL_PROMPT
        if pc_info:
            system_prompt += (
                f"\n\nОсь характеристики ПК користувача:\n"
                f"CPU: {pc_info.get('cpu', 'Не вказано')}\n"
                f"GPU: {pc_info.get('gpu', 'Не вказано')}\n"
                f"RAM: {pc_info.get('ram', 'Не вказано')}\n"
                f"SSD/HDD: {pc_info.get('storage', 'Не
вказано')}\n"
                "Враховуй ці характеристики у відповідях."
            )
        response = await make_api_request(system_prompt,
message.text, user_id)
        await message.answer(response, parse_mode="Markdown")
    async def main():
        try:
            await bot(DeleteWebhook(drop_pending_updates=True))
            await dp.start_polling(bot)
        except Exception as e:
            logger.error(f"Bot error: {e}")
    if __name__ == "__main__":
        asyncio.run(main())
```

СКРІНШОТИ РОБОТИ ІНТЕРФЕЙСУ

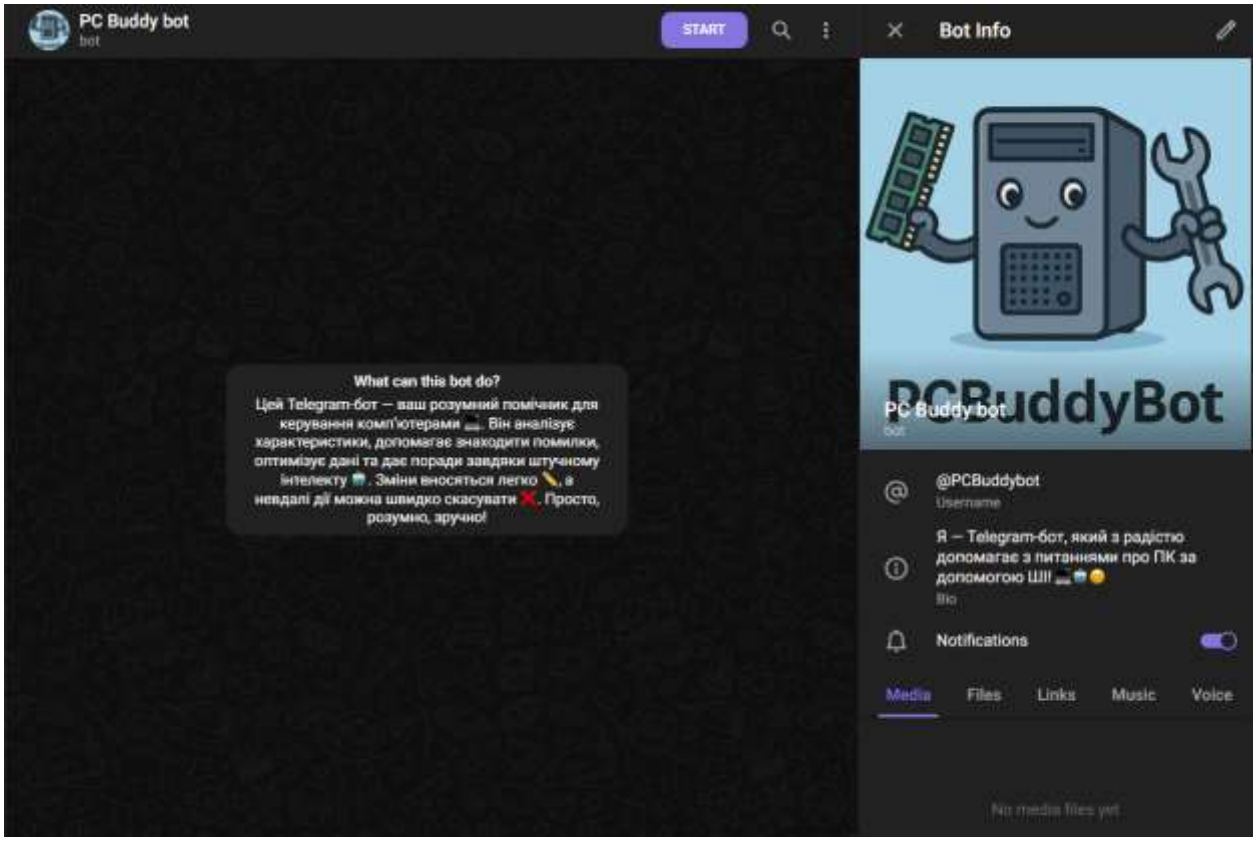

Рисунок 6.1 Стартове меню бота

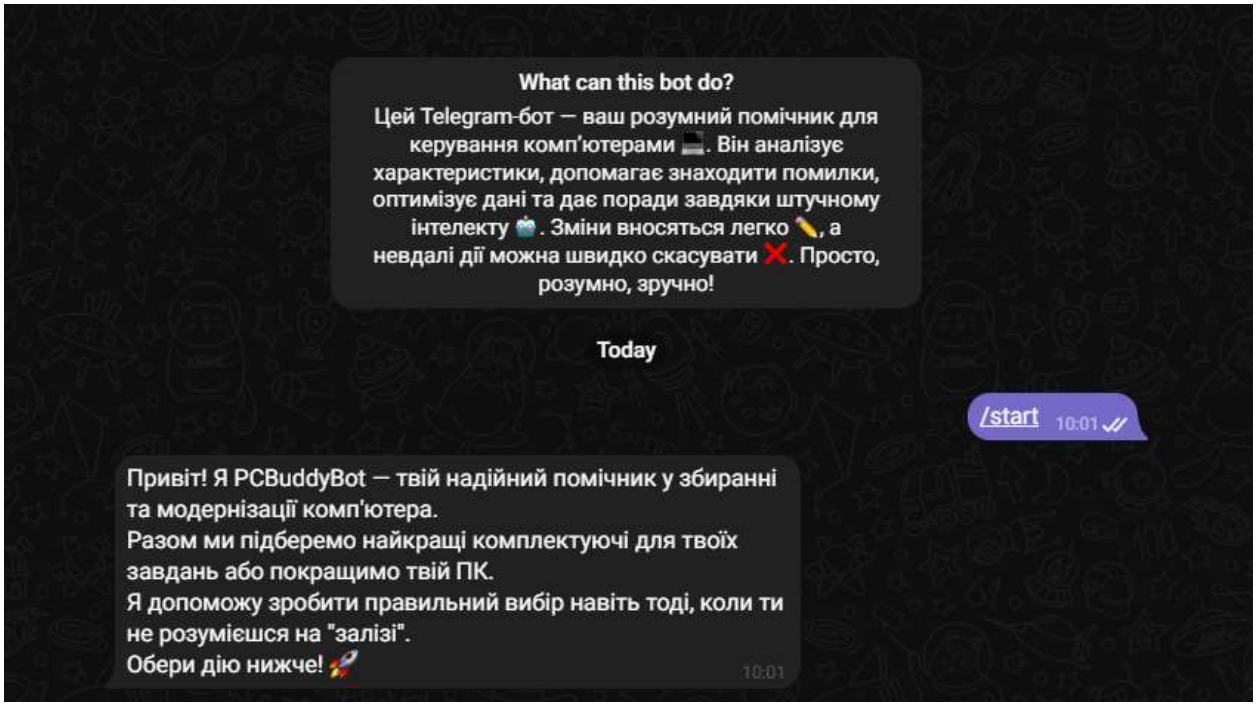

Рисунок 6.2 Реакція бота на команду /start

Рисунок 6.3 Вибір пункту «Мій ПК»

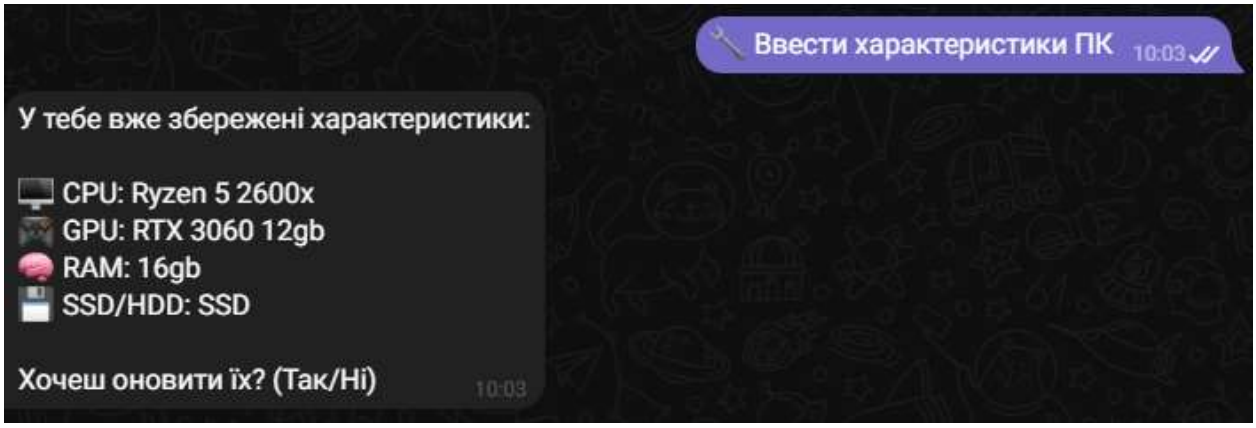

Рисунок 6.4 Вибір пункту «Ввести характеристики ПК»

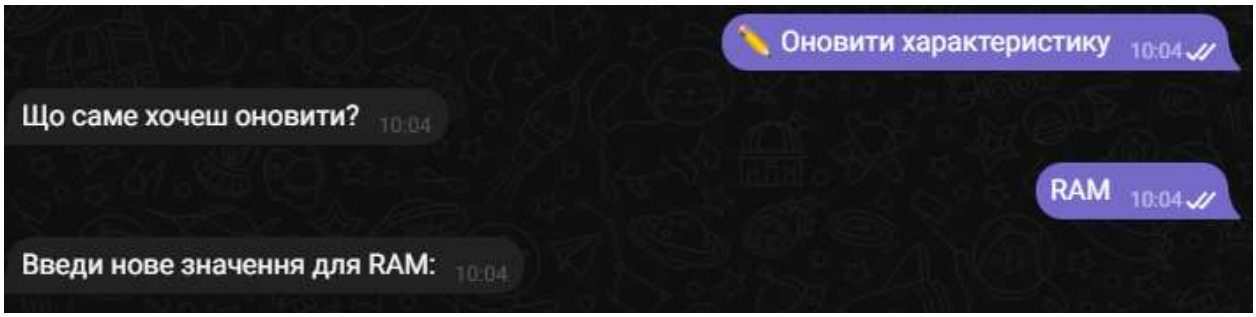

Рисунок 6.5 Вибір пункту «Оновити характристику»

Рисунок 6.6 Відміна у пункті «Оновити характристику»

Рисунок 6.7 Вибір пункту «Оцінка характеристик ПК»

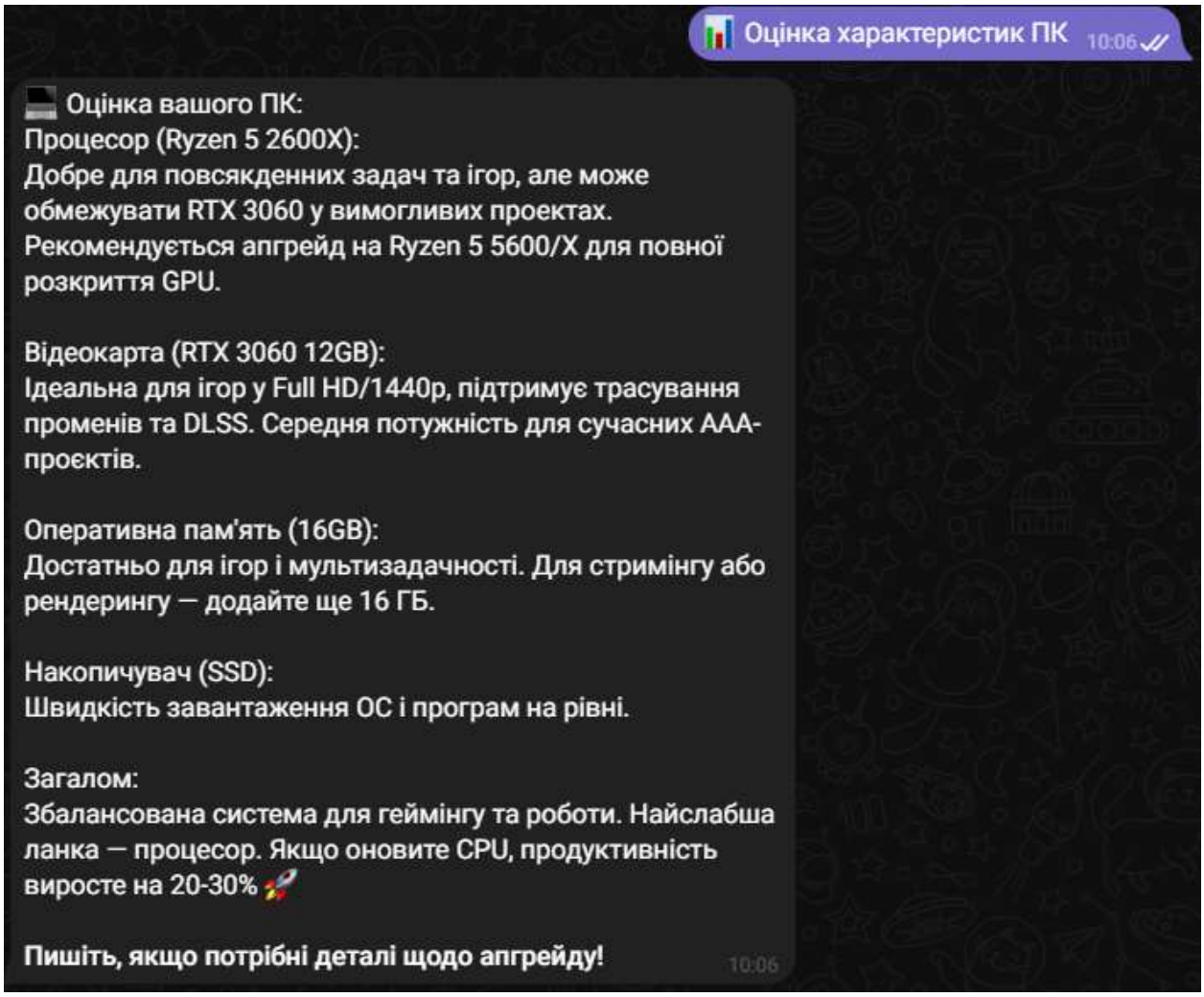

Рисунок 6.8 Результат пункту «Оцінка характеристик ПК»

Рисунок 6.9 Довільне питання боту

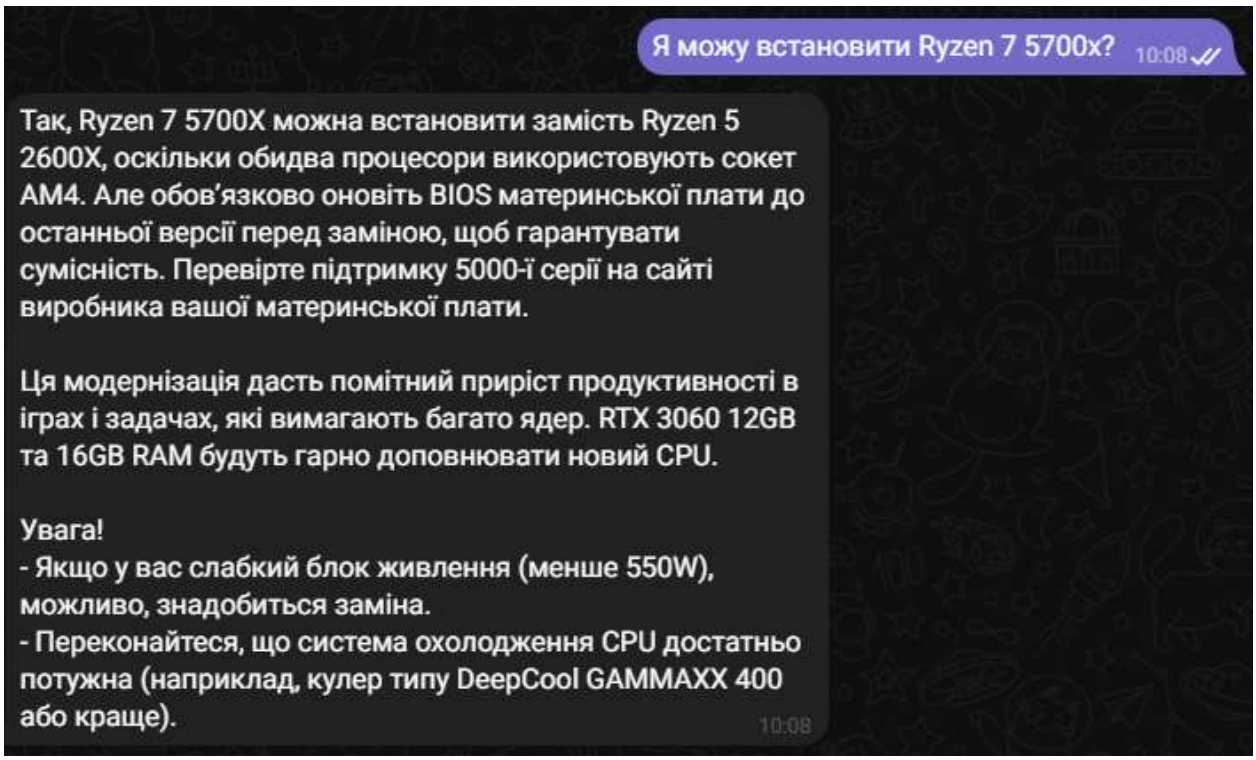

Рисунок 6.10 Результат довільного повідомлення боту

Рисунок 6.11 Вибір пункту «Видалити характеристики»

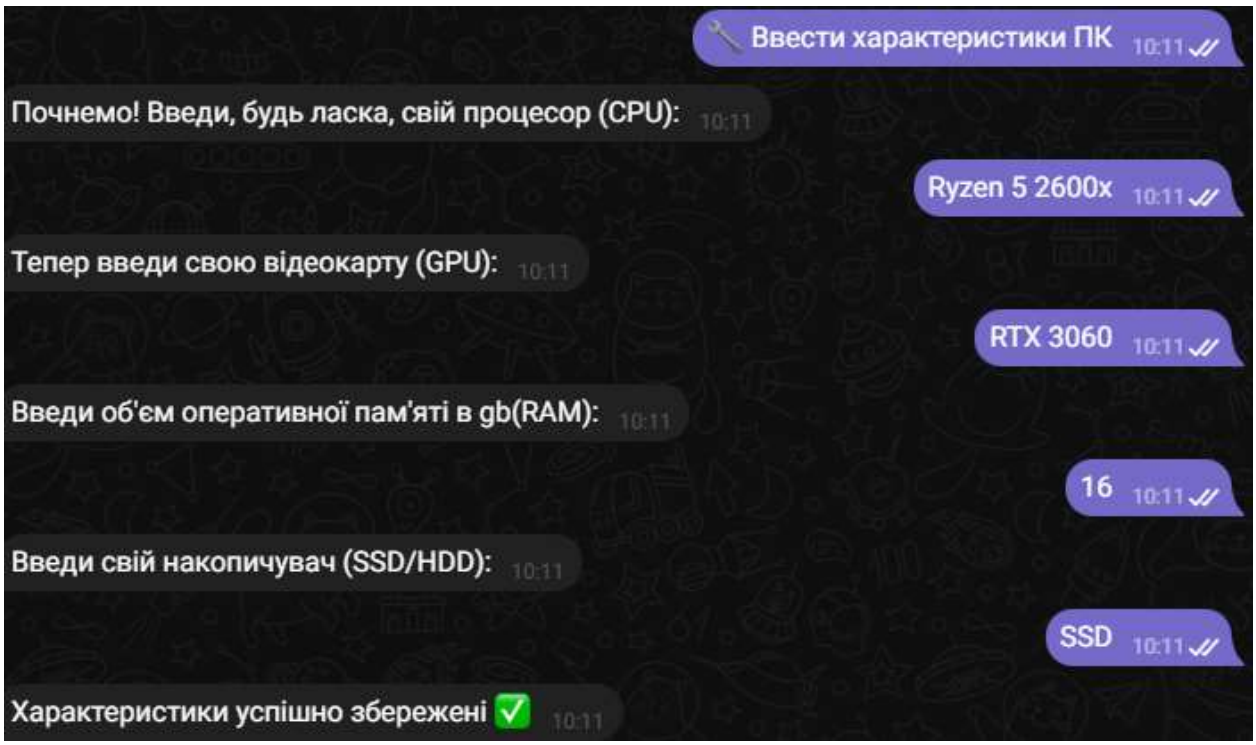

Рисунок 6.12 Введення характеристик