Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

«Затверджую»
в.о. завідуючого кафедри
комп'ютерних систем та робототехніки
_____ к. ф.-м. н., доцент Максим Хруслов
«___» червня 2025 р.

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: **«КОМП'ЮТЕРНА МОДЕЛЬ TELEGRAM-БОТА ДЛЯ ПІДБОРУ КОМПЛЕКТУЮЧИХ ПК НА ОСНОВІ ШІ»**

Галузь знань: 12 – Інформаційні технології.
Спеціальність 123 – Комп'ютерна інженерія.
Освітня програма «Комп'ютерна інженерія».

Захищено на засіданні
Екзаменаційної комісії № 44
протокол № __ від __.06.2025 р.
Оцінка _____ / _____
Голова Екзаменаційної комісії
_____ **ЧУГАЙ А.М.**

Виконав:
Студент групи КІ– 41
КОЛІСНИК Віталій Андрійович ___

Керівник: доцент зво кафедри комп'ютерних систем та робототехніки, PhD з інформаційних технологій
МОРОЗ Ольга Юріївна_____

Рецензент: доцент зво кафедри інтелектуальних програмних систем і технологій, к.т.н., доцент
ГАМЗАЕВ Рустам Олександрович___

Харків – 2025

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 74 сторінки, із яких 49 сторінок основної частини з 13 рисунками, 2 таблицями, 14 найменувань списку використаних джерел та чотирма додатками.

Метою кваліфікаційної роботи є розробка комп'ютерної моделі Telegram-бота, який з використанням ШІ та методів обробки природної мови (NLP) забезпечує діалогову взаємодію з користувачем, аналізує його вимоги (ціна, призначення, бажана продуктивність тощо) та формує оптимальні конфігурації ПК.

Об'єкт дослідження – процеси автоматизованого вибору апаратного забезпечення для персональних комп'ютерів з урахуванням індивідуальних вимог користувача, технічної сумісності та актуальних ринкових пропозицій

Предмет дослідження – комп'ютерна модель Telegram-бота, що базується на алгоритмах штучного інтелекту та машинного навчання для аналізу параметрів і автоматизованого підбору комплектуючих ПК персонального комп'ютера.

Завдання, яке вирішується в кваліфікаційній роботі полягає в тому, щоб розробити Telegram-бота, який за допомогою елементів штучного інтелекту автоматично підбирає комплектуючі для персонального комп'ютера відповідно до запитів користувача. Реалізація передбачає побудову архітектури, вибір інструментів, створення алгоритмів обробки та тестування.

Отримані результати демонструють ефективність і доцільність застосування ШІ в задачах підбору апаратного забезпечення.

Ключові слова: *Telegram-бот, штучний інтелект, ПК-комплектуючі, автоматизований підбір, рекомендаційна система, Python, UML-діаграма.*

ABSTRACT

The explanatory note to the bachelor's thesis consists of an introduction, three chapters, conclusions, a list of references and four appendices. The total volume of the work is 74 pages, including 49 pages of the main part with 13 figures, 2 tables, 14 references and four appendices.

The purpose of the qualification work is to develop a computer model of a Telegram bot, which, using AI and natural language processing (NLP) methods, provides dialogic interaction with the user, analyzes his requirements (price, purpose, desired performance, etc.) and forms optimal PC configurations.

The object of research is the processes of automated selection of hardware for personal computers, taking into account individual user requirements, technical compatibility and current market offers

The subject of research is a computer model of a Telegram bot, based on artificial intelligence and machine learning algorithms for analyzing parameters and automated selection of PC components for a personal computer.

The task to be solved in the qualification work is to develop a Telegram bot, which, using artificial intelligence elements, automatically selects components for a personal computer in accordance with user requests. Implementation involves building an architecture, selecting tools, creating processing algorithms and testing.

The results obtained demonstrate the effectiveness and feasibility of using AI in hardware selection tasks.

Keywords: *Telegram bot, artificial intelligence, PC components, automated selection, recommendation system, Python, UML diagram.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

ПК	–	персональний комп’ютер;
ШІ	–	штучний інтелект;
API	–	(англ. Application Programming Interface) програмний інтерфейс прикладного програмування;
NLP	–	(англ. Natural Language Processing) обробка природної мови;
iOS	–	операційна система мобільних пристроїв Apple;
macOS	–	операційна система для комп’ютерів Apple;
HTTP	–	(англ. HyperText Transfer Protocol) протокол передавання гіпертексту;
URL	–	(англ. Uniform Resource Locator) уніфікований локатор ресурсу (веб-адреса);
DApps	–	(англ. Decentralized Applications) децентралізовані застосунки;
ЕС	–	експертні системи;
PHP	–	(англ. Hypertext Preprocessor) мова програмування для веб-розробки;
AWS	–	(англ. Amazon Web Services) – хмарна платформа від Amazon;
VPS	–	(англ. Virtual Private Server) – віртуальний приватний сервер;
SQLite	–	вбудована реляційна система керування базами даних;
PostgreSQL	–	об’єктно-реляційна система керування базами даних з відкритим кодом;
UML-діаграма	–	(англ. Unified Modeling Language Diagram) – діаграма уніфікованої мови моделювання;
HTML	–	(англ. HyperText Markup Language) – мова розмітки гіпертексту для створення веб-сторінок.

ВСТУП

У сучасному інформаційному суспільстві стрімкий розвиток комп'ютерних технологій призвів до зростання попиту на персональні комп'ютери, які використовуються в широкому спектрі застосувань – від домашнього використання та освіти до професійної діяльності в IT, дизайні, наукових дослідженнях та іграх. Залежно від призначення, конфігурації ПК можуть бути найрізноманітнішими, що вимагає врахування численних технічних та економічних факторів при виборі компонентів.

Проблема полягає в тому, що не всі користувачі мають достатній технічний досвід, щоб самостійно підібрати сумісні та оптимальні компоненти – такі як процесор, материнська плата, пам'ять, відеокарта, пристрої зберігання даних, блок живлення та інші. Часто користувачам доводиться консультуватися зі сторонніми фахівцями або витрачати значну кількість часу на аналіз інформації в Інтернеті. Такий підхід не тільки неефективний, але й може призвести до невдалого вибору, що негативно вплине на продуктивність та довговічність ПК.

У той же час розвиток ШІ та методів машинного навчання відкриває нові перспективи для створення розумних систем, які здатні полегшити процес прийняття рішень та автоматизувати складні завдання. Одним із прикладів таких рішень є чат-боти – програмні інструменти, що можуть вести діалог із користувачами у текстовій або голосовій формі. Завдяки відкритому API, широкому функціоналу та численній аудиторії, месенджер Telegram виступає ефективною платформою для впровадження подібних технологій.

Актуальність роботи. З розвитком ШІ та машинного навчання стало можливим створення інтелектуальних систем, здатних аналізувати великі обсяги технічних даних і надавати користувачам персоналізовані рекомендації. Telegram, як популярна комунікаційна платформа, пропонує зручні можливості для інтеграції таких інтелектуальних систем у вигляді

ботів, що дозволяє забезпечити широке охоплення користувачів та легкий доступ до сервісу.

Тому розробка комп'ютерної моделі Telegram-бота для підбору компонентів ПК на основі ШІ є актуальною задачею, яка поєднує в собі сучасні технології ШІ, автоматизацію процесу консультування та зручну для користувача взаємодію. Такий підхід значно знижує вхідний бар'єр до комп'ютерної техніки для недосвідчених користувачів та сприяє більш раціональному використанню технічних і фінансових ресурсів при складанні ПК.

Метою дослідження є розробка комп'ютерної моделі телеграм-бота, який використовує методи ШІ для автоматизації підбору компонентів ПК на основі заданих користувачем параметрів.

Об’єкт дослідження – процеси автоматизованого вибору апаратних компонентів для персональних комп'ютерів.

Предмет дослідження – комп’ютерна модель Telegram-бота, що базується на алгоритмах ШІ та призначена для підбору комплектуючих ПК.

Методи дослідження: системний аналіз, проектування та моделювання.

Завдання дослідження:

1. Проаналізувати сучасні підходи до автоматизованого підбору комп’ютерних комплектуючих.
2. Дослідити інструменти розробки Telegram-ботів та бібліотеки ШІ/ML (наприклад, Python, OpenAI API, TensorFlow, scikit-learn).
3. Побудувати архітектуру моделі Telegram-бота, що реалізує логіку діалогу та підбору.
4. Реалізувати Telegram-бота з використанням вибраних інструментів.
5. Провести тестування моделі на прикладах з різними параметрами запитів.
6. Оцінити ефективність роботи бота та можливості масштабування або інтеграції з онлайн-магазинами.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ TELEGRAM-БОТІВ ТА ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Поняття та особливості Telegram-ботів

Telegram-боти – це програми, спеціально розроблені для взаємодії з користувачами в Telegram – широко використовуваній платформі для обміну повідомленнями. Від допомоги у виконанні різних завдань до імітації людських розмов. Ці боти допомагають людям бути більш продуктивними, розважатися та спілкуватися з улюбленими брендами за допомогою месенджера, яким вони користуються щодня [1].

Однією з ключових переваг Telegram-ботів є їхня здатність автоматизувати взаємодію, надаючи користувачам миттєві відповіді, персоналізовані рекомендації та доступ до послуг без втручання людини. Ботів можна використовувати для підвищення продуктивності, розваг, підтримки клієнтів, розповсюдження контенту і навіть для здійснення транзакцій в електронній комерції.

Крім того, Telegram-боти легко інтегруються з API, базами даних і сторонніми додатками, що дозволяє компаніям і розробникам розширювати їхню функціональність. Користувачі можуть взаємодіяти з ботами за допомогою текстових команд, кнопок, вбудованих запитів або навіть обробки природної мови (NLP) на основі штучного інтелекту.

Популярність Telegram-ботів значною мірою зумовлена простотою розгортання, масштабованістю та економічною ефективністю, що робить їх потужним інструментом як для бізнесу, так і для приватних осіб. Оскільки все більше галузей впроваджують автоматизацію та рішення на основі штучного інтелекту, роль Telegram-ботів продовжує зростати, покращуючи користувацький досвід та оптимізуючи комунікаційні процеси.

Месенджер Telegram – безпечна та універсальна комунікаційна платформа. Telegram поєднує в собі швидкість і ефективність WhatsApp з

ефемерністю Snapchat, що робить його унікальним і потужним інструментом комунікації. Він пропонує користувачам швидкий, надійний і безпечний спосіб обміну повідомленнями та мультимедійним контентом, зберігаючи при цьому суворий контроль конфіденційності.

Розглянемо основні характеристики та унікальні можливості Telegram, які відрізняють його від інших месенджерів:

1. Можливості та функції обміну повідомленнями: як і WhatsApp, Telegram дозволяє користувачам надсилати текстові повідомлення, зображення, відео, аудіофайли, документи, контакти та дані про місцезнаходження. Він також дозволяє користувачам бачити онлайн-статус своїх контактів, що робить спілкування в реальному часі більш безперешкодним. Однак Telegram йде ще далі, дозволяючи користувачам встановлювати таймер самознищення для повідомлень, гарантуючи, що спільний контент буде автоматично видалений після закінчення певного часу. Telegram також підтримує великі групи (до 200 000 учасників) і канали для трансляції повідомлень необмеженій аудиторії. Користувачі можуть створювати власні стікери, анімовані емодзі та чат-боти, покращуючи досвід обміну повідомленнями.

2. Розширені функції безпеки та конфіденційності: однією з найважливіших характеристик Telegram є його високий рівень безпеки. Уся комунікація, включаючи чати, групи та спільні медіа, шифрується, щоб запобігти несанкціонованому доступу. Крім того, Telegram пропонує наскрізне шифрування для приватних чатів, відомих як «Секретні чати». Ці чати захищені механізмом самознищення, що гарантує неможливість доступу до повідомлень після закінчення терміну їхньої дії. Такий рівень шифрування означає, що навіть власні сервери Telegram не можуть отримати доступ до контенту розмови.

3. Крос-платформенна доступність та підхід з відкритим вихідним кодом: Telegram доступний на всіх основних платформах, включаючи iOS, Android, Windows, macOS і Linux. На відміну від деяких конкурентів, він має веб-версію і підтримує одночасний доступ з декількох пристроїв. Ще одним

важливим аспектом є його підхід з відкритим вихідним кодом. Хоча більшість клієнтського коду Telegram знаходиться у вільному доступі, архітектура серверної частини залишається пропрієтарною. Розробники можуть отримати доступ до API Telegram для створення власних додатків, ботів та інтеграцій.

4. Популярність і конкуренція: Telegram конкурує з провідними сервісами обміну миттєвими повідомленнями, такими як WhatsApp і Facebook Messenger. Хоча він перевершує такі орієнтовані на конфіденційність альтернативи, як Signal і Wickr, за кількістю користувачів, він все ще відстає від WhatsApp за загальною кількістю завантажень. Станом на січень 2021 року Telegram повідомив про 500 мільйонів активних користувачів щомісяця, які обмінюються десятками мільярдів повідомлень щодня.Такому зростанню сприяло те, що користувачі шукали більш приватну і багатофункціональну альтернативу основним програмам для обміну повідомленнями.

5. Майбутня монетизація та розвиток: незважаючи на те, що Telegram є безкоштовним і не містить реклами, компанія планує підтримувати свою діяльність за рахунок пожертвувань і добровільних внесків. Компанія також натякнула на впровадження стратегій ненав'язливої монетизації, таких як преміум-функції для бізнесу та досвідчених користувачів, зберігаючи при цьому свою прихильність до дотримання принципу конфіденційності. Завдяки поєднанню швидкості, безпеки та універсальності, Telegram продовжує утверджуватися як провідна платформа для обміну повідомленнями як для особистого, так і для професійного спілкування.

Хоча боти в Telegram загалом безпечні, користувачі повинні вживати необхідних заходів обережності, щоб забезпечити безпечний і позитивний досвід роботи з ними. Telegram надає API для ботів, який дозволяє розробникам створювати та розгортати ботів, але важливо зазначити, що не всі боти підтримують однаковий рівень безпеки та стандартів конфіденційності.

Забезпечення безпеки при використанні Telegram-ботів:

1. Конфіденційність і захист даних: одним з найважливіших аспектів безпеки ботів є конфіденційність даних. Перш ніж взаємодіяти з ботом,

користувачі повинні перевірити, як він збирає, обробляє та зберігає особисту інформацію. Ознайомлення з політикою конфіденційності бота може дати уявлення про його методи обробки даних, що допоможе користувачам приймати обґрунтовані рішення перед тим, як ділитися будь-якими конфіденційними даними. Бажано уникати ботів, які запитують непотрібні персональні дані або не мають чітких вказівок щодо конфіденційності.

2. Обережно з посиланнями та завантаженнями: користувачам слід бути обережними, отримуючи посилання та файлові вкладення від Telegram-ботів. Зловмисники можуть маскувати шкідливі посилання під легітимний контент, що може призвести до фішингових атак або зараження шкідливим програмним забезпеченням.

3. Розпізнавання та уникнення шахрайських ботів: шахраї часто використовують Telegram-ботів для обману користувачів, пропонуючи безкоштовні винагороди, розіграші або ексклюзивні пропозиції в обмін на платежі або особисту інформацію.

Отже, можемо сказати, що Telegram-боти – це програмні додатки, інтегровані в екосистему месенджера Telegram, які дозволяють автоматизувати різні процеси та взаємодію з користувачами. Вони можуть виконувати широкий спектр завдань: від надання довідкової інформації та підтримки клієнтів до обробки платежів, управління групами та створення інтерактивного контенту. Завдяки підтримці API, боти можуть інтегруватися з зовнішніми сервісами, базами даних і штучним інтелектом, що значно розширює їхні можливості. Гнучкість налаштувань, висока швидкість роботи та підтримка наскрізного шифрування роблять Telegram-ботів ефективним інструментом для автоматизації бізнес-процесів, інформаційного обслуговування та створення нових цифрових рішень [2].

1.2 Основні компоненти та принципи функціонування Telegram-бота

Телеграм-бот – це програмний додаток, розроблений для взаємодії з користувачами через месенджер Telegram. Його функціонал варіюється від

простих інформаційних трансляцій до складних інтерактивних сервісів, як у випадку з ботом для підбору комплектуючих для ПК на основі штучного інтелекту. Щоб зрозуміти його роботу, необхідно розглянути ключові складові та принципи функціонування.

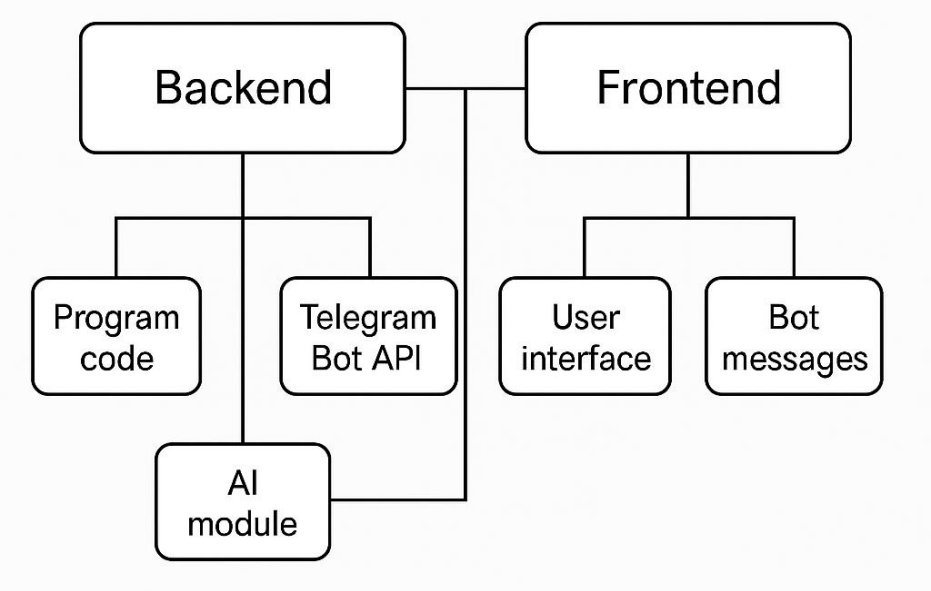

Рисунок 1.1 – Основні компоненти Telegram-бота.

Основні компоненти Telegram-бота (див. Рис.1.1)**:**

1. Backend (серверна частина): мозок та двигун бота.

Програмний код бота:

- Деталі: цей код – серце Вашого Telegram-бота. Він містить логіку обробки вхідних повідомлень, реалізує алгоритми ШІ для вибору компонентів, взаємодіє з базою даних і Telegram API.
- Приклад у даному проєкті: код може бути написаний на Python з використанням бібліотеки python-telegram-bot. Він включатиме функції для обробки текстових команд (наприклад, /start, /build_pc), обробки відповідей користувача на запитання (наприклад, про бюджет, призначення ПК), виклику модуля ШІ для аналізу вимог і вибору компонентів, а також для генерації фінального списку рекомендацій.

- Візуалізація: діаграма класів або діаграма архітектури програмного забезпечення, що показує основні модулі та їх взаємозв'язки (наприклад, модуль обробки команд, модуль ШІ, модуль взаємодії з базою даних, модуль Telegram API).

API Telegram Bot:

- Деталі: це API, наданий Telegram, який дозволяє боту взаємодіяти з платформою. Через API бот отримує оновлення про нові повідомлення, команди, натискання кнопок та інші події, а також може надсилати текстові повідомлення, зображення, аудіо, відео, використовувати клавіатуру, вбудовані режими та інші функції Telegram.
- Приклад у даному проєкті: коли користувач надсилає повідомлення боту, сервери Telegram пересилають це оновлення на ваш сервер (якщо налаштовано веб-хук) або ваш сервер періодично запитує нові оновлення (опитування). Ваш код використовує API для обробки цього повідомлення і відправки відповіді користувачеві (текст, клавіатура з опціями, зображення компонентів і т.д.).
- Візуалізація: спрощена діаграма клієнт-серверної взаємодії, де бот є клієнтом, а Telegram API – сервером, що показує обмін повідомленнями (запитами та відповідями).

Модуль ШІ:

- Деталі: це ключовий компонент, який забезпечує інтелект бота. Він може включати алгоритми машинного навчання (наприклад, системи рекомендацій на основі контенту або спільної фільтрації), методи NLP для розуміння запитів користувачів, логічні правила для перевірки сумісності компонентів та алгоритми оптимізації для вибору найкращого співвідношення ціни та якості.
- Приклад у даному проєкті: якщо користувач запитує «ігровий ПК для сучасних ігор з бюджетом $1000», модуль ШІ аналізує ці параметри, робить запит до бази даних, враховує інформацію про продуктивність

різних конфігурацій в іграх (можливо, отриману із зовнішніх джерел або навчену на попередніх даних), перевіряє сумісність компонентів і пропонує оптимальну конфігурацію.

- Візуалізація: блок-схема, що ілюструє роботу алгоритму ШІ (наприклад, етапи обробки запитів користувачів, вибір критеріїв, пошук і фільтрація компонентів, перевірка сумісності, ранжування результатів).

2. ***Frontend (клієнтська частина): що бачить користувач.***

Інтерфейс користувача Telegram:

- Деталі: користувач взаємодіє з ботом через стандартний інтерфейс Telegram, використовуючи текстові повідомлення та елементи керування, що надаються ботом.

- Приклад у даному проєкті: користувач може надсилати текстові команди (наприклад, /start), вводити свої вимоги в текстовій формі («Мені потрібен ПК для редагування відео») або натискати кнопки на клавіатурі, запропонованій ботом (наприклад, «Ігри», «Офіс», «Мультимедіа»).

- Візуалізація: скріншоти інтерфейсу Telegram з прикладами взаємодії користувача з ботом (наприклад, вікно чату з ботом, введення команди, відображення клавіатури).

Повідомлення бота:

- Деталі: основний спосіб передачі інформації від бота до користувача. Це можуть бути текстові описи, списки рекомендованих компонентів, таблиці зі специфікаціями, зображення компонентів, посилання на додаткову інформацію.

- Приклад у даному проєкті: бот може надсилати повідомлення зі списком рекомендованих компонентів із зазначенням їхніх назв, основних характеристик, цін і посилань на магазини. До повідомлення можна додати зображення кожного компонента.

- Візуалізація: приклад повідомлення від бота з текстом, зображенням компонента і кнопками для подальшої взаємодії (наприклад, «Детальніше», «Додати в кошик»).

Також розглянемо не менш важливий аспект, а саме основні принципи роботи Telegram-бота:

1. *Отримання оновлень:* Telegram-боти не працюють постійно у фоновому режимі, активно «слухаючи» нові повідомлення. Замість цього вони отримують інформацію про події (нові повідомлення, команди, натискання кнопок тощо) через Telegram Bot API одним з двох способів:
    - Веб-хуки: Telegram надсилає HTTP POST-запит з інформацією про нову подію на попередньо налаштовану URL-адресу на вашому сервері. Це вважається більш ефективним підходом для більшості ботів.
    - Тривале опитування: бот періодично надсилає запити на сервери Telegram, щоб перевірити наявність нових оновлень. Цей метод менш ефективний, але його простіше реалізувати для початківців або в середовищах, де налаштування веб-хуків ускладнене.
2. *Обробка вхідних даних:* Після отримання оновлення програмний код бота аналізує його. Це може включати в себе:
    - Синтаксичний аналіз тексту повідомлення: визначення команд (починаючи з /), ключових слів або фраз.
    - Обробка натискань кнопок: визначення кнопок, натиснутих користувачем на вбудованій клавіатурі.
    - Аналіз інших типів даних: обробка надісланих зображень, аудіо, відео, геолокації тощо.
3. *Прийняття рішень та виконання дій:* на основі оброблених вхідних даних логіка бота визначає, які дії необхідно виконати. Це можуть бути такі дії:

- Формування відповіді: створення текстових повідомлень, зображень, аудіо, відео або інших типів відповідей для користувача.
- Взаємодія з API Telegram: відправлення сформованої відповіді назад користувачеві, редагування існуючих повідомлень, відправлення спеціальних елементів інтерфейсу (клавіатури, вбудованих меню тощо).
- Збереження та отримання даних: взаємодія з базою даних для збереження інформації про користувачів, їхні запити, налаштування або отримання необхідних даних для формування відповіді.
- Використання зовнішніх API: звернення до інших сервісів для отримання додаткової інформації або виконання певних дій (наприклад, отримання курсу валют, пошук інформації в Інтернеті).
- Застосування модуля ШІ: аналіз складних запитів, розуміння намірів користувача, вибір оптимальних рішень або надання персоналізованих рекомендацій.

4. *Надсилання відповідей:* після виконання необхідних дій бот відправляє сформовану відповідь назад користувачеві через API Telegram Bot.

Циклічність: цей процес отримання оновлення, його обробки, прийняття рішення та надсилання відповіді повторюється кожного разу, коли користувач взаємодіє з ботом або коли відбуваються інші події, на які бот налаштований реагувати.

Таким чином, Telegram-бот – це програма, яка працює на сервері та використовує Telegram Bot API для спілкування з користувачами, реагуючи на їхні дії та надаючи відповідний функціонал. Ключовими принципами є реактивність на події, обробка отриманих даних відповідно до запрограмованої логіки та використання API для взаємодії з платформою Telegram.

Механізм дії Telegram-бота можна описати наступною схемою (рис.1.2).

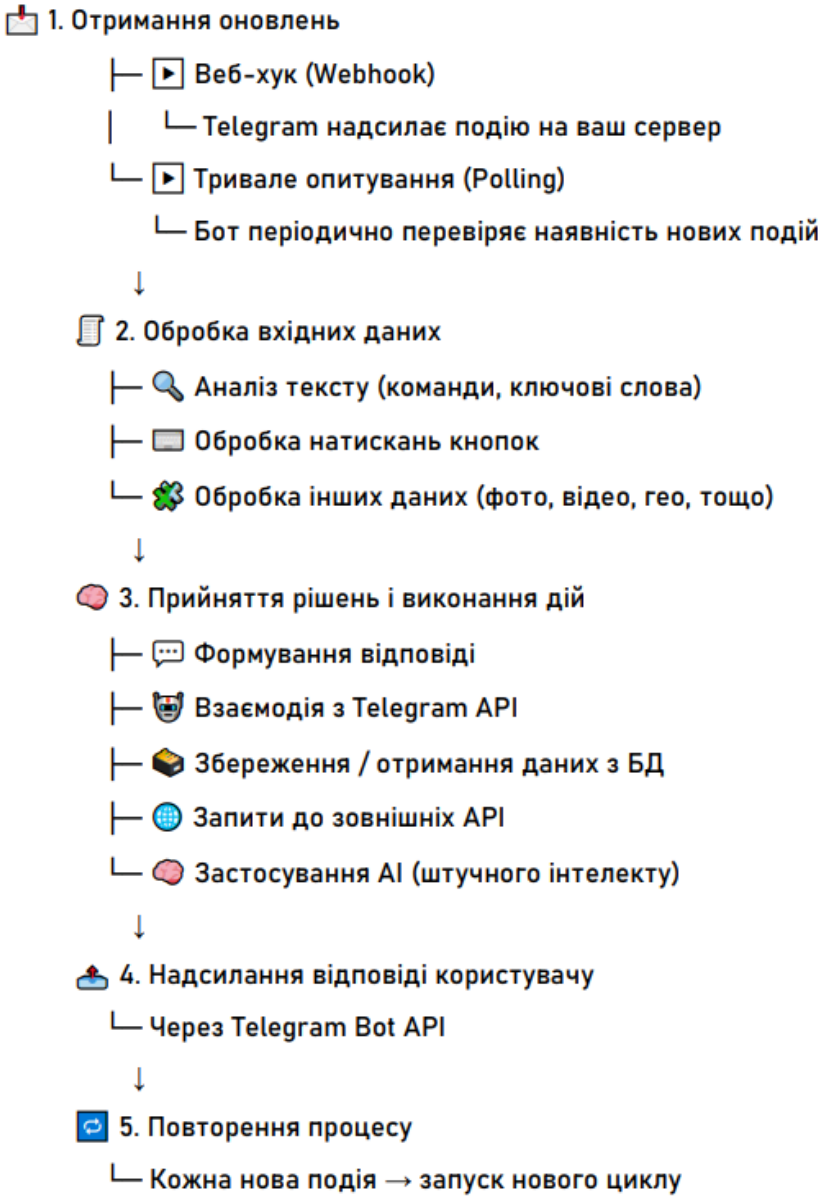

Рисунок 1.2 – Механізм дії Telegram-бота.

1.3 Роль та перспективи використання Telegram-ботів у сучасних технологічних рішеннях

Роль Telegram-ботів з кожним днем продовжує зростати завдяки кільком ключовим факторам:

1. **Автоматизація рутинних завдань:** боти можуть автоматизувати повторювані завдання, такі як надсилання повідомлень і нагадувань, обробка даних і відповіді на поширені запитання. Це підвищує ефективність і звільняє

людські ресурси для більш складної роботи. Наприклад, бот може автоматично надсилати клієнтам оновлення замовлень або нагадувати членам команди про дедлайни.

2. **Покращений клієнтський сервіс:** Telegram-боти забезпечують миттєву підтримку клієнтів у режимі 24/7, відповідаючи на поширені запитання, усуваючи неполадки та обробляючи прості запити. Це підвищує задоволеність клієнтів і зменшує навантаження на службу підтримки. Компанії можуть інтегрувати ботів зі своєю базою знань, щоб надавати вичерпну інформацію.

3. **Покращення взаємодії та комунікації з користувачами:** боти пропонують користувачам прямий і зручний канал комунікації. Вони можуть надсилати персоналізовані повідомлення, рекламні пропозиції та оновлення, щоб підвищити рівень залученості та побудувати міцніші відносини з клієнтами. Такі функції, як швидкі відповіді та кнопки меню, покращують взаємодію з користувачами.

4. **Маркетинг і продажі:** ботів можна використовувати в маркетингових кампаніях для надсилання цільових повідомлень, ексклюзивних пропозицій і знижок. Боти також можуть надавати персоналізовані рекомендації на основі історії користувача та його вподобань, тим самим стимулюючи продажі та підвищуючи впізнаваність бренду.

5. **Збір даних та зворотній зв'язок:** боти можуть збирати цінні дані про користувачів за допомогою опитувань та форм зворотного зв'язку, щоб надавати інформацію для покращення продуктів та послуг.

6. **Оптимізація внутрішньої бізнес-комунікації:** організації можуть використовувати ботів для автоматизації внутрішніх сповіщень, розподілу завдань і командної комунікації, щоб покращити співпрацю та продуктивність.

7. **Економічна ефективність:** розробка Telegram-бота може бути більш економічно вигідною альтернативою створенню повноцінного

мобільного додатку для певних функцій, наприклад, для обслуговування клієнтів.

8. **Інтеграція з іншими сервісами:** Telegram -ботів можна інтегрувати з різними зовнішніми сервісами та системами, такими як CRM, ERP, платіжні шлюзи та AI-асистенти. Це розширює їхні можливості та оптимізує робочі процеси [3].

9. Надання інформації та доступу до послуг: боти можуть надавати інформацію в режимі реального часу, наприклад, оновлення новин, прогнози погоди та фінансові дані. Вони також можуть полегшити бронювання або покупки безпосередньо в інтерфейсі Telegram.

Потрібно також розуміти, що зокрім переваг та перспектив розвитку Telegram -ботів є і певні недоліки, які представлені в таблиці 1.1.

Таблиця 1.1

Переваги та недоліки розвитку Telegram -ботів

Критерій порівняння	Переваги	Недоліки
Доступність	Доступні 24/7, не потребують постійного обслуговування	Потребують початкової настройки та розгортання
Вартість	Недорогі у розробці й обслуговуванні	Додаткові витрати при використанні хостингу або сторонніх API
Інтеграція з Telegram	Швидке реагування, інтеграція з чатами, каналами	Залежність від політик Telegram
Можливості автоматизації	Автоматизують відповіді, замовлення, нагадування	Обмежена логіка без підключення до складних бекендів
Взаємодія з користувачем	Інтуїтивно зрозумілий інтерфейс, легкість у використанні	Менше можливостей UI в порівнянні з веб- або мобільними додатками
Швидкість розгортання	Швидке створення MVP	Може потребувати доопрацювання для складніших завдань
Безпека	Telegram забезпечує базовий рівень захисту	Дані зберігаються в Telegram, обмежений контроль над збереженням даних

Проаналізувауши таблицю, можемо зазначити, що переваги переважають над недоліками а це даєпереспективу:

1. **Зростаюча база користувачів:** зростаюча популярність Telegram надає компаніям і розробникам доступ до величезної аудиторії за допомогою ботів.

2. **Розвиток ШІ та NLP:** з розвитком технологій ШІ та NLP боти ставатимуть більш досконалими. Вони будуть здатні розуміти складні запити і вести більш природні розмови. Це призведе до більш інтуїтивної та корисної взаємодії з ботами.

3. **Персоналізований досвід:** боти майбутнього, ймовірно, використовуватимуть штучний інтелект, щоб пропонувати максимально персоналізований досвід, пристосовуючи контент, рекомендації та взаємодію до індивідуальних уподобань користувача.

4. **Безперешкодна інтеграція з Web3:** Telegram вже працює з Web3, і боти, ймовірно, стануть більш інтегрованими з децентралізованими додатками (DApps). Це полегшить транзакції, надасть дані в режимі реального часу та покращить користувацький досвід у середовищі Telegram.

5. **Експансія в нові галузі:** очікується, що Telegram-боти поширяться на інші галузі, включаючи охорону здоров'я, освіту, фінанси та розваги, пропонуючи інноваційні рішення та послуги.

6. **Розширені мультимедійні можливості:** боти, ймовірно, використовуватимуть підтримку Telegram для мультимедійних даних (зображень, відео та аудіо), щоб створювати більш захоплюючий та інтерактивний досвід.

7. **Міні-додатки всередині ботів:** платформа Telegram дозволяє розробляти міні-додатки всередині ботів, надаючи більш складні функції та користувацькі інтерфейси, не змушуючи користувачів виходити з екосистеми Telegram. Очікується, що ця тенденція зростатиме, роблячи ботів ще більш універсальними.

8. **Бізнес-специфічні функції:** завдяки таким функціям, як години роботи, місцезнаходження, швидкі відповіді та автоматизовані повідомлення,

Telegram запровадив акаунти «Telegram Business», що свідчить про зростаючу увагу до надання бізнесу більшої кількості інструментів та можливостей в рамках платформи. Це ще більше розширює перспективи для бізнес-орієнтованих ботів.

Отже, Telegram-боти – це потужні, універсальні інструменти, які відіграють важливу роль у сучасних технологічних рішеннях. Їх здатність автоматизувати завдання, покращувати комунікацію та надавати цінні послуги робить їх все більш актуальними для бізнесу, розробників та приватних осіб. З постійним розвитком штучного інтелекту та постійним зростанням платформи Telegram, перспективи їхнього використання та впливу є багатообіцяючими.

1.4 Визначення ШІ та його складових

За своєю суттю, ШІ – це галузь комп'ютерних наук, яка вивчає розробку складних комп'ютерних програм, здатних виконувати завдання, що зазвичай вимагають людського інтелекту та участі людини. Такі завдання можуть варіюватися від розуміння розмовної мови та розпізнавання закономірностей до творчого вирішення проблем і навчання на основі попередньої інформації. Простіше кажучи, це симуляція процесів людського інтелекту, що дозволяє машинам міркувати, самокоректуватися, оптимізуватися і набувати певного рівня сприйняття свого оточення [4].

ШІ перетворився на широку підгалузь комп'ютерних наук, що охоплює різні сфери застосування. Сучасні застосування штучного інтелекту розвинулися далі створення інтелектуальних систем і досягли рівня розуміння інтелекту та способів його відтворення.

Технології ШІ поєднують великі обсяги даних з інтелектуальними алгоритмами, що дозволяє програмному забезпеченню виявляти закономірності та вчитися на них. Залежно від використовуваного алгоритму, модель ШІ може адаптуватися до нових вхідних даних, що дозволяє їй навчатися і розвиватися незалежно від її творця і на основі наданих даних.

Створення і навчання ШІ починається зі збору необхідної кількості даних, які можуть бути у вигляді тексту, зображень, відео- або аудіофайлів. Потім програмне забезпечення ШІ обробляє та аналізує дані, щоб виявити закономірності та взаємозв'язки між різними точками. Після достатнього впливу і навчання система ШІ використовує те, чого вона навчилася, щоб робити прогнози, приймати рішення і виконувати завдання незалежно від втручання людини. Крім того, ШІ-системи здатні навчатися і вдосконалюватися, чим більше вони взаємодіють з навколишнім середовищем і отримують точний зворотний зв'язок [5].

В основі алгоритмів ШІ лежать кілька компонентів, які мають вирішальне значення для його роботи, наприклад, машинне навчання. Інші компоненти включають NLP, комп'ютерний зір та ЕС, які також відіграють вирішальну роль у тому, щоб інфраструктура ШІ могла розуміти та інтерпретувати вхідні дані без допомоги людини.

1.5 Основні напрямки розвитку ШІ

В окремих сферах ШІ робить набагато більше, ніж очікувалося, а в інших – значно менше, ніж припускали ще десять років тому. На основі досвіду експертів і програмістів стають видимими реальні перспективи й напрями розвитку та впровадження ринку ШІ у найближчому майбутньому.

Розвиток ШІ та машинного навчання безумовно стане рушієм змін на ринку праці. Завдяки використанню рішень на основі ШІ ми зможемо працювати набагато ефективніше та продуктивніше. Водночас з’являться нові робочі місця, але знайти відповідних працівників стане складніше. Для впровадження ШІ в організаціях недостатньо просто найняти технічних фахівців – потрібно також вміти інтерпретувати результати. Працівників доведеться навчати новим компетенціям і новому підходу до взаємодії з людьми й технологіями штучного інтелекту [6].

72% генеральних директорів у всьому світі вважають, що ШІ та машинне навчання стануть конкурентною перевагою їхнього бізнесу. У 2021-

2024 роках ШІ вже автоматизує дедалі складніші процеси, виявляє бізнес-тренди та підтримує працівників у виконанні завдань.

До цього часу багато компаній не могли повною мірою оцінити ефективність інвестицій у Big Data. Однак зі зростанням застосування ШІ дедалі більше організацій переосмислюють свою стратегію щодо даних. Вони починають ставити правильні запитання: Як можна покращити процеси? Що зробити для автоматизації збору даних? Оскільки багато типів ШІ (наприклад, машинне або глибинне навчання) потребують величезних обсягів даних, які потрібно структурувати й очистити від упередженості, компанії почнуть реформувати архітектуру даних і підходи до управління ними – зосереджуючись на вирішенні конкретних проблем.

Ключовими стануть ті працівники, які вміють підготувати дані так, щоб ШІ міг використати їх якнайефективніше. З огляду на те, що ШІ стане частиною повсякденних бізнес-процесів, експерти в галузях будуть цінуватися ще більше, ніж аналітики даних.

Разом із розвитком ШІ зростає і кількість кіберзагроз. Вдосконалені моделі машинного та глибинного навчання, а також нейронні мережі дозволяють системам виявляти закономірності та вразливості. Масштабоване машинне навчання у поєднанні з хмарними технологіями дозволяє аналізувати величезні обсяги даних і оперативно виявляти загрози. ШІ може не лише реагувати на атаки, а й передбачати, де саме зростає ризик, і формувати звіти з кібербезпеки [7].

ШІ, який працює за принципом «чорної скриньки», може викликати недовіру з боку користувачів, що обмежуватиме його впровадження. Під тиском громадськості багато компаній змушені будуть «відкрити» ці системи, аби пояснити принцип їхньої роботи. Це призведе до необхідності компромісів: прозорість знизить ризики, але зробить процеси повільнішими та дорожчими.

Канада, Японія, Велика Британія, Німеччина та ОАЕ вже мають стратегії розвитку технологій ШІ. США також планують перетворити країну на лідера

в цій галузі, що має зміцнити економіку та підвищити національну безпеку. Китай вважає ШІ стратегічним напрямом національного розвитку і становить дедалі більшу загрозу для технологічного домінування США. В Україні та Східній Європі інтерес до ШІ з боку бізнесу поки що розвивається повільно й здебільшого має декларативний характер.

1.6 Перспективи розвитку ШІ

Майбутнє ШІ сповнене потенціалу і готове змінити те, як ми живемо і взаємодіємо з технологіями. Ми можемо очікувати, що системи ШІ стануть набагато потужнішими, еволюціонуючи від базового розпізнавання шаблонів до складних міркувань і вирішення проблем. Обробка природної мови вдосконалюватиметься, уможливлюючи більш природне та контекстне спілкування зі штучним інтелектом. Комп'ютерний зір стане більш точним в інтерпретації візуальних даних. Ключовою тенденцією є розвиток мультимодального ШІ, який може розуміти і обробляти різні типи інформації, такі як текст, зображення і аудіо. Це призведе до більш інтуїтивної взаємодії. Крім того, поява ШІ-агентів, здатних автономно виконувати складні завдання, підвищить ефективність у багатьох галузях.

Інтеграція ШІ стане більш поширеною в таких галузях, як охорона здоров'я, фінанси, транспорт і освіта, завдяки зниженню витрат і більшій доступності. Майбутні програми штучного інтелекту, ймовірно, будуть зосереджені на наданні персоналізованого, безперебійного досвіду, який адаптується до індивідуальних потреб. Конвергенція ШІ з новими технологіями, такими як Web3 та Інтернет речей, відкриє нові та непередбачувані можливості [8].

Очікується, що в найближчі роки розвиток ШІ визначатимуть кілька ключових тенденцій. Більший акцент буде зроблено на міркуваннях ШІ, які дозволять системам робити логічні висновки. Агентний ШІ, орієнтований на автономні дії, стане більш поширеним. Буде розроблено спеціалізоване апаратне забезпечення ШІ для оптимальної продуктивності, а хмарні рішення ШІ забезпечать масштабованість і доступність. Найголовніше – зростатиме

увага до створення метрик для ефективного вимірювання продуктивності та впливу штучного інтелекту.

Оскільки технологія ШІ продовжує розвиватися, етичні міркування стануть першорядними. Буде вкрай важливо забезпечити справедливість і зменшити упередженість систем ШІ, а також сприяти більшій прозорості та зрозумілості у прийнятті рішень у сфері ШІ. Захист конфіденційності користувачів і забезпечення надійної безпеки даних також будуть ключовими питаннями. Крім того, встановлення чіткої підзвітності за дії систем ШІ та пріоритетність безпеки людини є важливими для відповідальної розробки та розгортання ШІ. По суті, перспективи розвитку штучного інтелекту є яскравими, багатообіцяючими трансформаційними досягненнями в багатьох сферах. Однак відповідальна реалізація цього потенціалу вимагає проактивного і вдумливого підходу до етичних проблем, які супроводжують цю потужну технологію. Дедалі більше уваги приділятиметься створенню інтелектуальних, надійних, орієнтованих на людину систем штучного інтелекту, що відповідають нашим цінностям [9].

Висновки за розділом 1

У цьому розділі розглянуто ключові аспекти Telegram-ботів та ШІ, створено теоретичне підґрунтя для їхнього практичного застосування. Було проаналізовано структуру, принципи роботи та потенціал Telegram-ботів в автоматизації та наданні послуг. Також були розглянуті основні поняття штучно, його класифікація та сучасні тенденції розвитку. Особливу увагу було приділено розвитку можливостей міркувань, агентним системам та мультимодальному ШІ. У тексті висвітлено багатообіцяючі перспективи подальшого розвитку ШІ та його інтеграції з новітніми технологіями. Було підкреслено важливість етичних міркувань в еволюції ШІ. Ці знання створюють міцний фундамент для подальших досліджень щодо інтеграції ШІ в Telegram-ботів.

РОЗДІЛ 2.
ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ ТА ПРОЄКТУВАННЯ TELEGRAM-БОТА

2.1. Вибір інструментарію для розробки Telegram-бота

Telegram-боти – це універсальна та ефективна платформа для розробки інтерактивних автоматизованих систем. Завдяки широкому застосуванню в різних сферах, таких як електронна комерція, служби підтримки та персональні асистенти, вони є популярним вибором серед розробників. Розглянемо основні інструменти, мови, фреймворки та стратегії розгортання для створення Telegram-ботів, з акцентом на технічну реалістичність та гнучкість впровадження:

1. Мови програмування: Telegram-ботів можна розробляти за допомогою різних мов програмування, кожна з яких має свої переваги, що базуються на продуктивності, простоті використання та підтримці бібліотек. Найпопулярнішими з них є:

- Python: відома своєю читабельністю та простотою використання, Python надає надійні бібліотеки, такі як Python-Telegram-Bot, Telebot та асинхронна Aiogram. Це робить Python ідеальною як для початківців, так і для досвідчених користувачів [10].
- Node.js (JavaScript): добре підходить для додатків у реальному часі, JavaScript пропонує фреймворки, такі як Node-Telegram-Bot-API і Telegraf, що дозволяють розробляти програми на основі подій і легко інтегруватися з веб-технологіями.
- PHP: легкий і доступний варіант, який дозволяє швидко розробляти простих ботів за допомогою таких пакетів, як PHP-Telegram-Bot.
- Java: надійний вибір для рішень корпоративного рівня, підтримується бібліотеками на кшталт TelegramBots, які пропонують широкі можливості та інтеграцію.

- Go (Golang): цінується за свою продуктивність і простоту, Go ідеально підходить для створення ефективних ботів з високою пропускною здатністю за допомогою таких бібліотек, як tgbotapi.
- Бібліотека Telegram.Bot дозволяє розробникам створювати ботів в екосистемі .NET та легко інтегруватися з системами на базі Windows.

2. Фреймворки та бібліотеки ботів: фреймворки відіграють важливу роль у спрощенні розробки, надаючи структуровані API та механізми обробки подій. У таблиці нижче наведено деякі з найбільш поширених бібліотек.

Таблиця 2.1

Найпоширеніші фреймворки та бібліотеки

Мова програмування	Фреймворк/бібліотека	Опис
Python	python-telegram-bot, aiogram	Повнофункціональний, з підтримкою асинхронного режиму, добре документований
Node.js	Telegraf, node-telegram-bot-api	Легкий, простий синтаксис
Java	TelegramBots	Надійний і активно підтримується
PHP	php-telegram-bot	Ідеально підходить для простих, легких додатків
C# (.NET)	Telegram.Bot	Підтримує більшість функцій Telegram API

3. Хостинг і розгортання: для забезпечення безперервної роботи бот повинен бути розгорнутий на сервері або хостинговій платформі. Існує кілька варіантів, починаючи від безкоштовних хмарних сервісів і закінчуючи масштабованою інфраструктурою корпоративного рівня:

- Heroku є зручним для початківців і пропонує безкоштовний рівень. Його легко розгортати через Git.
- Render та Railway – сучасні альтернативи Heroku, що пропонують швидке налаштування та інтеграцію з GitHub.
- Replit корисний для створення прототипів за допомогою браузерного редактора коду та миттєвого розгортання.

- Хмарні платформи (наприклад, AWS, Google Cloud та Microsoft Azure) забезпечують гнучкі, масштабовані рішення для виробничих середовищ.
- Віртуальні приватні сервери (VPS): такі сервіси, як DigitalOcean або Linode, пропонують повний контроль і налаштування для розгортання на професійному рівні.

4. Інструменти та ресурси для розробки: кілька інструментів, які підтримують життєвий цикл розробки ботів:

- BotFather: офіційний інструмент Telegram для створення ботів та управління токенами.
- Ngrok: дозволяє тестувати локальні веб-хуки через безпечні тунелі
- Postman корисний для тестування кінцевих точок API Telegram Bot.
- Платформи контролю версій, такі як GitHub або GitLab, допомагають керувати базами коду.
- Для зберігання даних можна використовувати такі бази даних, як SQLite, PostgreSQL та Firebase.
- Моніторинг та ведення журналів: такі сервіси, як Sentry або Loggly, допомагають з налагодженням та відстеженням продуктивності.

Розробка Telegram-ботів забезпечує значну гнучкість і потенціал для автоматизації. За допомогою правильних інструментів, мови програмування та стратегії розгортання розробники можуть створювати масштабованих, чуйних і зручних ботів для різних випадків використання. Доступні API, підтримка спільноти та можливості хмарного розгортання роблять розробку Telegram-ботів потужним інструментом сучасної програмної інженерії.

2.2. UML-діаграма Telegram-бота

UML-діаграма на рис.2.1 ілюструє базову модель взаємодії для чат-бота на базі Telegram. Вона ілюструє, як користувач спілкується безпосередньо з ботом, який потім виконує кілька окремих дій [11].

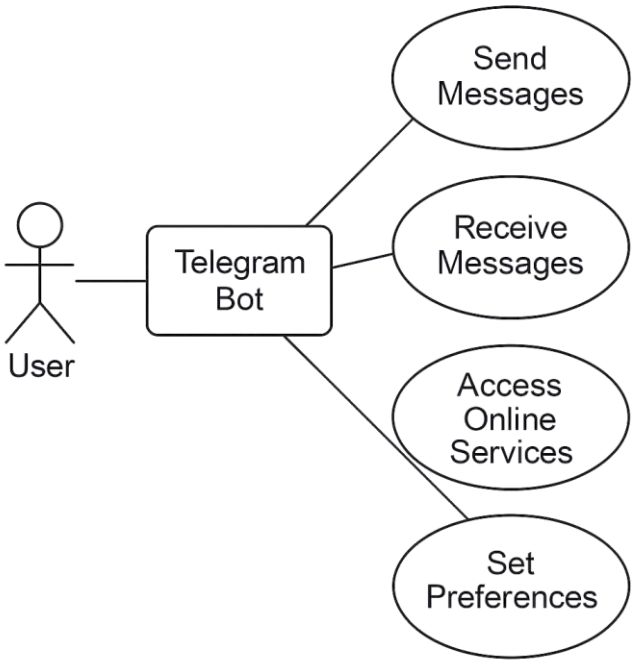

Рисунок 2.1 – UML-діаграма базової моделі взаємодії для чат-бота.

Компоненти на діаграмі:

1. Користувач: людина, яка взаємодіє з Telegram-ботом. Цей актор ініціює комунікацію та використовує функції бота.
2. Система: Telegram-бот – цей компонент системи (програмний агент) обробляє взаємодію з користувачем, обробляє запити та виконує автоматизовані завдання. Бот виступає інтерфейсом між користувачем і різними сервісами.
   - Варіанти використання:
   - Надсилання повідомлень:
   - Користувачі можуть надсилати боту команди, запити або текстові повідомлення.
   - Приклади: /start, запитання, вподобання або вимоги.
   - Бот призначений для інтерпретації цих даних і реагування на них.
3. Отримувати повідомлення:
   - Бот реагує на повідомлення користувача текстовими відповідями, меню або інтерактивними опціями.

- Це включає надання інформації, підтверджень або повідомлень про помилки.
- Наприклад, бот може надати рекомендацію щодо збірки ПК, звіт про сумісність або довідковий посібник.

4. Доступ до онлайн-сервісів:
   - Бот може взаємодіяти із зовнішніми API або базами даних для надсилання або отримання інформації.
   - Це може включати перевірку сумісності обладнання, отримання поточних цін, доступ до оглядів продуктів або отримання шаблонів конфігурації.
   - Це забезпечує динамічну функціональність у реальному часі на основі даних, введених користувачем.
   - Налаштування параметрів.
   - Користувачі можуть визначати або змінювати свої особисті налаштування та уподобання.
   - Наприклад, ви можете встановити бажаний діапазон бюджету, вибрати мову, вибрати тип використання за замовчуванням (наприклад, ігри або редагування), а також вказати налаштування сповіщень.
   - Ці налаштування допомагають боту персоналізувати свої відповіді та рекомендації.

Така структура є типовим дизайном для розумних асистентів і сервіс-орієнтованих чат-ботів, забезпечуючи ефективну та зручну взаємодію, яку можна налаштувати.

2.3. Структурна схема Telegram-бота

Схема на рис. 2.2 ілюструє структурований потік взаємодії з чат-ботом, який допомагає користувачам створити свій ідеальний ПК.

Рисунок 2.2 – Структурна схема структурованого потоку взаємодії з чат-ботом.

Процес починається, коли користувач запускає чат, набираючи /start. Після цього чат-бот вітає користувача дружнім повідомленням і надає інструкції зі збирання комплектуючих до ПК. Потім користувачеві пропонується вибрати дію.

Далі чат-бот збирає початкову інформацію від користувача за допомогою текстової анкети. Це включає три ключові кроки:

1. **Запит про бюджет:** користувач вказує свій бюджет.
2. **Опитування про завдання:** користувач описує основне призначення ПК (наприклад, ігри, програмування, офісна робота, відеомонтаж тощо).
3. **Запит на спеціальні вимоги:** користувач може вказати будь-які специфічні потреби чи вподобання, такі як перевага певних брендів чи

компонентів, компактний розмір корпусу, рівень шуму чи енергоефективність.

Після отримання необхідної інформації чат-бот обробляє вхідні дані. На основі отриманих даних чат-бот може виконувати кілька інтелектуальних функцій:

1. **Вибір компонентів:** система підбирає відповідне апаратне забезпечення (процесор, графічний процесор, материнську плату, оперативну пам'ять тощо), яке відповідає цілям і бюджету користувача.
2. **Перевірка сумісності:** система перевіряє, чи всі вибрані компоненти технічно сумісні між собою (наприклад, чи підходить процесор до роз'єму материнської плати, чи підтримується оперативна пам'ять, чи достатній блок живлення).

Якщо користувач передумає або потребуватиме модифікацій (наприклад, переходу на інший форм-фактор або модернізації деталі), бот може адаптувати конфігурацію відповідним чином. Якщо користувачеві потрібна допомога або роз'яснення в будь-який момент, бот може запропонувати додаткові вказівки або пояснення.

Після завершення всіх перевірок чат-бот надає результат сумісності, підтверджуючи, що конфігурація є правильною. Якщо все сумісно, користувач отримує остаточну рекомендацію зі складання ПК. Ця рекомендація містить детальний перелік компонентів і пояснення, чому кожна частина була обрана, щоб забезпечити прозорість і довіру.

Крім того, процес розроблений таким чином, щоб бути гнучким. Якщо користувач хоче внести зміни, наприклад, скоригувати бюджет або переглянути варіант використання, він може повернутися до попередніх кроків. Чат-бот переробить інформацію і згенерує нову конфігурацію.

Загалом, цей процес взаємодії орієнтований на користувача та інтуїтивно зрозумілий, що дозволяє будь-кому легко планувати та налаштовувати свій ідеальний ПК. Чат-бот супроводжує користувачів на

кожному етапі, від визначення їхніх потреб до отримання збалансованої, сумісної системи, розробленої спеціально для них (див. рис. 2.3)

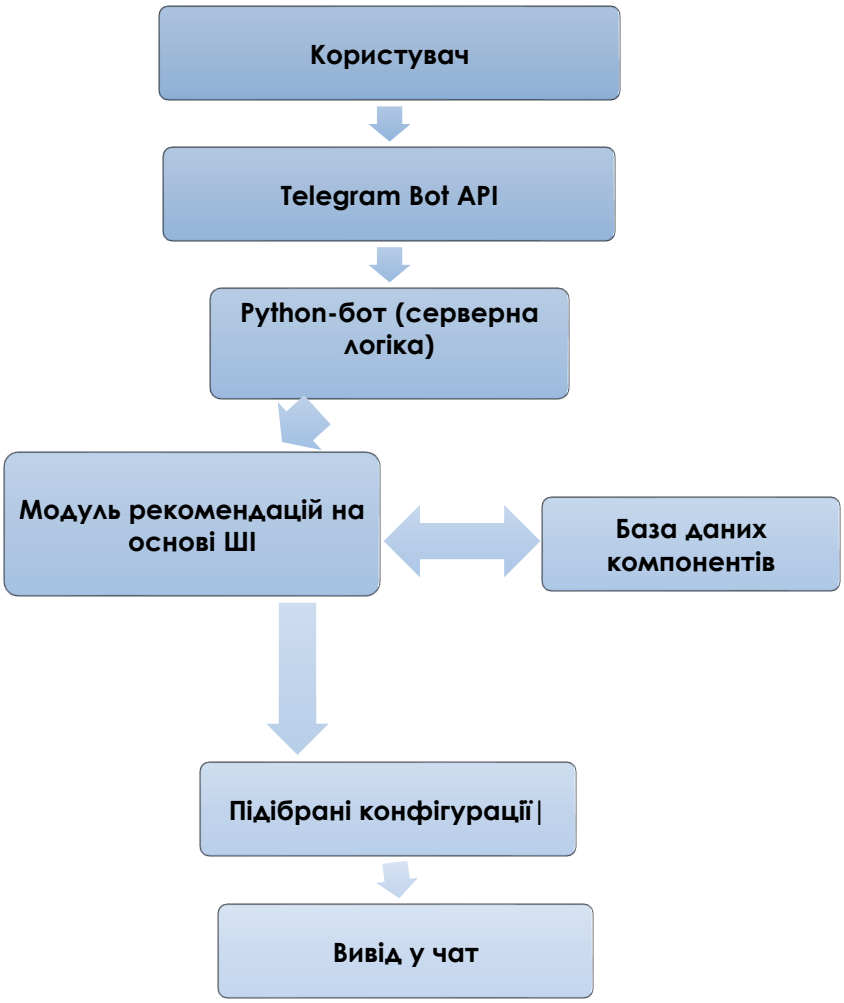

Рисунок 2.3 – Опис архітектури системи.

Telegram-бот для підбору ПК-комплектуючих на основі ШІ складається з таких ключових компонентів:

Користувач Взаємодіє з ботом через мобільний/веб-клієнт Telegram, вводячи параметри бажаної конфігурації (бюджет, мета використання — ігри, навчання, робота тощо).

Telegram Bot API Служить інтерфейсом зв'язку між користувачем і ботом. Бот отримує повідомлення через webhook або polling.

Python-бот (серверна логіка) Реалізовано з використанням бібліотеки python-telegram-bot. Основні функції:

- обробка команд (/start, /help, /select)
- аналіз запитів
- передача параметрів у модуль рекомендацій
- повернення результатів користувачу

Модуль рекомендацій на основі ШІ Алгоритм (наприклад, rule-based або ML-модель), який:

- класифікує тип користувача
- підбирає відповідні компоненти (CPU, GPU, RAM, SSD тощо)
- формує збалансовану конфігурацію

База даних компонентів Містить інформацію про доступні комплектуючі: моделі, характеристики, ціни, наявність, сумісність. Дані можуть бути:

- локальними (SQLite/PostgreSQL)
- або зчитуватись з API сторонніх маркетплейсів (розширення у майбутньому)

Логіка зворотного зв'язку оцінка запропонованих конфігурацій користувачем

- збір аналітики для подальшого навчання моделей або покращення логіки

Висновок за розділом 2

У другому розділі було розглянуто можливі інструменти, за допомогою яких можна розробити Telegram-бота. Представлено UML-діаграму, яка описує логіку взаємодії компонентів, а також та архітектуру системи бота, що ілюструє чітке уявлення про внутрішню організацію системи та слугує основою для її реалізації.

РОЗДІЛ 3.
РОЗРОБКА ТА РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ МОДЕЛІ TELEGRAM-БОТА.

3.1. Визначення технологічного стеку та інструментарію для розробки.

При розробці Telegram-бота було використано різноманітний набір інструментів та бібліотек для персоналізованих рекомендацій щодо компонентів ПК, щоб забезпечити функціональність, масштабованість та залучення користувачів. Основна взаємодія побудована на бібліотеці Python-Telegram-Bot, яка забезпечує надійний фреймворк для обробки повідомлень користувачів, команд та елементів інтерфейсу, таких як кастомні клавіатури для відповідей. Завдяки цій бібліотеці бот бере участь у динамічному діалоговому обміні з користувачами, полегшуючи безперешкодну навігацію в процесі відбору. Система інтегрує бібліотеку Gradio Client, щоб генерувати інтелектуальні рекомендації на основі бюджету, мети та вподобань користувача. Ця бібліотека взаємодіє з потужною мовною моделлю Qwen2-72B-Instruct. Ця модель обробляє запити на природній мові та надає структуровані пропозиції щодо збірки ПК, фактично слугуючи ядром системи для прийняття рішень.

Щоб збагатити рекомендації візуальним контентом, бот використовує Selenium WebDriver для веб-автоматизації. Це уможливлює програмний перегляд та взаємодію з веб-сайтами електронної комерції, такими як Rozetka.ua, де бот шукає реальні зображення товарів на основі рекомендованої кейс-моделі. Selenium використовується в режимі headless для ефективного виконання в бекенд-середовищі, а бібліотека webdriver_manager спрощує налаштування ChromeDriver. BeautifulSoup аналізує відображені сторінки, витягуючи релевантні дані про зображення та інформацію про продукт, переміщуючись по структурі HTML. Цей гібридний підхід гарантує, що бот

надає текстові пропозиції та відповідні візуальні зображення продукту, підвищуючи довіру та залученість користувачів.

Крім того, реалізація включає в себе модуль ведення журналу для детального моніторингу виконання та відстеження помилок для підтримки супроводу та налагодження. Модуль re (регулярні вирази) дозволяє гнучко і точно обробляти текстові дані, від інтерпретації користувацького вводу до розбору відповідей моделі. Цей інструментарій формує модульну та цілісну архітектуру, кожен компонент якої надає певні можливості: спілкування з користувачем, прийняття рішень на основі ШІ, автоматизована веб-інтеграція та стабільність системи. Результатом є сучасний, зручний для користувача цифровий асистент, який може інтуїтивно та інтелектуально провести людину через складний процес вибору компонентів ПК.

Багато розробників надають перевагу мові програмування Python завдяки високій продуктивності розробки, яку вона забезпечує. Однією з ключових переваг є відсутність необхідності компіляції, що значно пришвидшує цикл «редагування – тестування – налагодження». Процес виявлення та усунення помилок у Python є інтуїтивно зрозумілим: замість критичних збоїв (таких як помилки сегментації) інтерпретатор генерує винятки, які можуть бути оброблені засобами мови.

У випадку необробленого винятку інтерпретатор автоматично формує трасування стека, що дозволяє швидко локалізувати джерело помилки. Для детального аналізу роботи програми розробникам доступний налагоджувач на рівні вихідного коду, який надає можливість переглядати значення змінних, обчислювати вирази, встановлювати точки зупину та виконувати покроковий перегляд коду. Варто зазначити, що сам налагоджувач реалізовано на мові Python, що додатково підкреслює її здатність до самоаналізу (інтроспекції).

Водночас, через високу швидкість редагування та запуску, на практиці часто використовується й простіший метод налагодження — додавання операторів виводу print() у ключові ділянки коду. Завдяки зручному

зворотному зв’язку цей підхід залишається ефективним для швидкої діагностики та усунення помилок [12].

У процесі розробки програмного забезпечення для взаємодії з месенджером Telegram широко застосовується бібліотека python-telegram-bot, яка є однією з найбільш популярних серед Python-розробників. Ця бібліотека забезпечує повну підтримку Telegram Bot API, що дозволяє реалізовувати повний спектр функціональних можливостей, передбачених платформою Telegram для ботів.

Висока популярність та розповсюдженість python-telegram-bot обумовлені її зручним інтерфейсом, активною спільнотою користувачів, систематизованою документацією та регулярною підтримкою. Вона значно спрощує процес створення, налаштування та обслуговування Telegram-ботів, надаючи розробнику доступ до таких функцій, як обробка повідомлень, inline-режими, клавіатури, webhooks, інтеграція з іншими сервісами тощо. Завдяки цьому бібліотека є доцільним вибором при реалізації системи автоматизованого підбору комплектуючих ПК через Telegram-інтерфейс [13].

Бібліотека **python-telegram-bot** є надійною та добре структурованою обгорткою над Telegram Bot API, яка істотно спрощує процес взаємодії з Telegram-ботами за допомогою мови програмування Python. Вона абстрагує низькорівневу взаємодію з API, зокрема роботу з HTTP-запитами та серіалізацією даних, і натомість пропонує високорівневий інтерфейс у вигляді класів та методів, що безпосередньо відповідають основним функціям API.

Найсуттєвішою перевагою python-telegram-bot є повна підтримка Telegram Bot API, що дозволяє реалізувати майже всі функції, задекларовані в офіційній документації. Це охоплює надсилання та отримання текстових повідомлень, обробку мультимедійних даних (зображень, аудіо, відео, документів), створення кастомізованих клавіатур (inline та reply), реалізацію вебхуків або механізму довгого опитування (long polling), роботу з командами,

управління ботами у групах та каналах, обробку вбудованих запитів, а також реалізацію платіжних сервісів.

Дизайн бібліотеки базується на подієво-орієнтованому підході: програміст визначає обробники для різних типів оновлень, таких як надходження нових повідомлень, натискання кнопок або виконання команд. Такий підхід спрощує логічну організацію програмного коду та полегшує реалізацію взаємодії з користувачем у межах бота.

Крім технічних переваг, python-telegram-bot вирізняється активною спільнотою користувачів та розробників. Наявність докладної документації, великої кількості прикладів, обговорень і регулярних оновлень забезпечує розробникам доступ до актуальної інформації та підтримки. Завдяки цьому бібліотека постійно вдосконалюється, залишаючись релевантною до змін Telegram API та сучасних практик розробки.

Таким чином, бібліотека python-telegram-bot є оптимальним вибором для розробки складних, багатофункціональних Telegram-ботів із повним використанням потенціалу платформи, забезпечуючи ефективний та гнучкий підхід до реалізації інтелектуальних систем на базі Python.

3.2. Реалізація алгоритмів обробки запитів та підбору комплектуючих

Для автоматизації процесу підбору комплектуючих ПК був розроблений Telegram-бот із використанням бібліотеки python-telegram-bot та інтеграцією з моделлю ШІ через клієнт Gradio. Бот взаємодіє з користувачем через послідовність кроків, що дозволяє отримати необхідні параметри для підбору комплектуючих:

1. Отримання бюджету – користувач вибирає або вводить власний бюджет на збірку ПК.
2. Визначення призначення ПК – ігри, робота, навчання, офіс або інше.
3. Особливі вимоги – низький рівень шуму, компактність корпусу, енергоспоживання тощо.

Стан користувача (який крок опрацьовується) зберігається у словнику user_states, що дозволяє вести багатокроковий діалог.

Після збору інформації бот формує текстовий запит для моделі ШІ (Qwen/Qwen2-72B-Instruct) через клієнт Gradio, передаючи бюджет, задачі та особливі вимоги. Модель генерує рекомендації щодо конкретних компонентів із зазначенням моделей і приблизної вартості.

Отримана відповідь обробляється функцією format_recommendations(), яка форматує текст, додає емодзі для кращої наочності та витягує модель корпусу для подальшого пошуку зображень.

Для візуалізації користувачу рекомендується зображення корпусу. Функція search_case_image() автоматично виконує пошук на сайті Rozetka за допомогою браузера Selenium у безголовому режимі. Завантажена сторінка парситься бібліотекою BeautifulSoup, знаходиться відповідне зображення корпусу і надсилається користувачу.

Окрім підбору, реалізована функція перевірки сумісності введених компонентів. Запит передається тій же моделі, яка аналізує список комплектуючих та повертає рекомендації щодо сумісності.

Обробка повідомлень та інтерфейс містять:

- Головне меню реалізоване через кнопки ReplyKeyboardMarkup для зручності вибору.
- Обробка текстових повідомлень реалізована у функції handle_message() із логікою станів.
- Використання асинхронних функцій забезпечує неблокуючу роботу бота.
- Логування подій і помилок налаштоване через стандартний модуль logging.

Вважаю, що алгоритми обробки запитів та підбору комплектуючих розроблені зрозумілою мовою для будь-якого користувача та зручні у

використанні.

3.3 Інструкція використання Telegram-бота

Telegram-бот працює на базі Python з використанням бібліотек python-telegram-bot, gradio_client та selenium. Для запуску необхідно:

1. Підготувати середовище:
    - Встановити необхідні бібліотеки: pip install -r requirements.txt.
    - Переконатися, що ChromeDriver встановлений автоматично через webdriver_manager.
2. Запуск бота:
    - Змініть токен у змінній TELEGRAM_BOT_TOKEN на власний.
    - Запустіть скрипт: python your_bot_file.py.
3. Функціонал адміністратора:
    - Код не містить окремого адміністративного інтерфейсу, але адміністратор може моніторити логування роботи бота через logging.
    - Для діагностики помилок у реальному часі слідкуйте за логами у консолі (помилки виводяться через logging.error()).
4. Можливе розширення:
    - Додайте авторизацію через ID, щоб створити інтерфейс адміністратора.
    - Створіть окремі команди (наприклад, /admin), щоб надавати особливі функції (очищення кешу, перегляд статистики тощо).

Інструкція для користувача

Цей Telegram-бот допомагає автоматично підібрати комплектуючі для персонального комп’ютера за допомогою штучного інтелекту. Для початку:

1. Натисніть /start, щоб запустити бота. Ви отримаєте головне меню з кнопками:
    - Підбір комплектуючих
    - Перевірка сумісності
    - Редагувати параметри

- Допомога

2. Підбір комплектуючих:
    - Оберіть бюджет або введіть свій.
    - Вкажіть призначення комп’ютера (ігри, робота, навчання тощо).
    - Оберіть або вкажіть особливі вимоги до ПК (наприклад, безшумність, компактність).
    - Бот згенерує рекомендації з конкретними моделями комплектуючих і приблизною вартістю.
3. Перевірка сумісності:
    - Виберіть пункт: Перевірка сумісності.
    - Введіть список компонентів, які хочете перевірити.
    - Бот повідомить, чи сумісні вони між собою.
4. Редагування параметрів:
    - Оберіть: Редагувати параметри, щоб почати підбір заново.
5. Отримання допомоги:
    - Натисніть: Допомога для стислого пояснення функцій.

Дані інструкції допоможуть будь-якій людині зрозуміти, як розпочати роботу з Telegram-ботом.

3.4 Тестування роботи Telegram-бота

На рис. 3.1 представлено головний екран користувацького інтерфейсу Telegram-бота для підбору комплектуючих до ПК. Після введення команди /start користувач отримує вітальне повідомлення: « Привіт! Я допоможу тобі зібрати ідеальний ПК. Вибери дію:». Це слугує початковою точкою взаємодії, пропонуючи чотири основні функції за допомогою інтерактивних кнопок для швидкого доступу до ключових сервісів бота.

Кнопка «Підбір комплектуючих» ініціює основний процес конфігурації майбутнього ПК. Після її натискання запускається діалог, у якому бот ставить користувачеві низку запитань для уточнення ключових параметрів, таких як бюджет, основне призначення комп'ютера (ігри, робота, навчання тощо) та бажані технічні характеристики. Отримана інформація аналізується за

допомогою інтегрованої ЕС для надання оптимальних рекомендацій щодо вибору комплектуючих.

Кнопка «Перевірка сумісності» надає користувачеві інструмент для самостійної перевірки сумісності вже обраних компонентів. Після активації бот може запропонувати користувачеві ввести назви або ключові характеристики компонентів (наприклад, модель процесора, материнської плати, відеокарти) для подальшого аналізу їх взаємодії та виявлення потенційних конфліктів.

Кнопка «Редагувати параметри» дозволяє користувачеві переглядати та змінювати раніше введені параметри вибору або перевірки сумісності. Ця функція зручна, коли користувач вирішує скоригувати свої вимоги, змінити бюджет або уточнити певні технічні характеристики.

Кнопка «Допомога» надає доступ до довідкової інформації про функціональність бота. Натиснувши її, користувач отримує опис доступних команд, інструкції з використання сервісу та відповіді на поширені запитання щодо процесу підбору та перевірки компонентів.

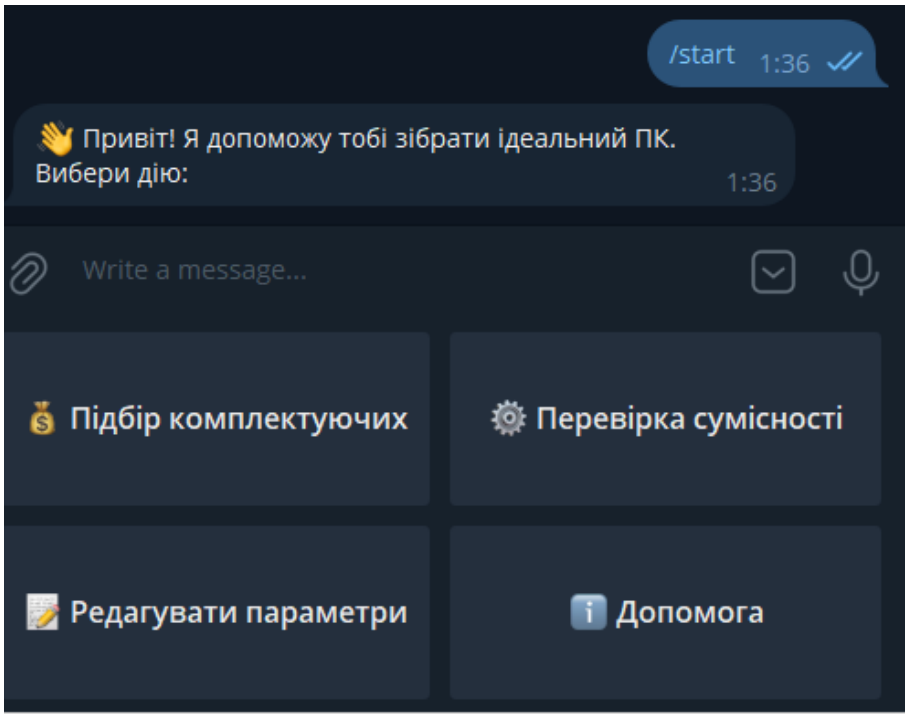

Рисунок 3.1 – Головний екран користувацького інтерфейсу.

На рис. 3.2 представлено екран введення бюджету в Telegram-боті для підбору комплектуючих ПК після натискання " Підбір комплектуючих". Бот запитує: «Вкажи свій бюджет в гривнях:».

Для зручності користувача запропоновано чотири варіанти бюджету у вигляді кнопок: «20000 грн», «40000 грн», «60000 грн», «80000 грн».

Нижче розташована кнопка «Інший бюджет», що дозволяє користувачеві ввести власне значення.

Такий підхід є зручним та гнучким, пропонуючи швидкий вибір типових бюджетів та можливість введення нестандартних значень. Кнопкова навігація спрощує взаємодію.

Візуалізація демонструє етап збору ключової інформації для подальшого підбору комплектуючих, починаючи з визначення фінансових можливостей користувача. Це важливий крок для надання релевантних рекомендацій.

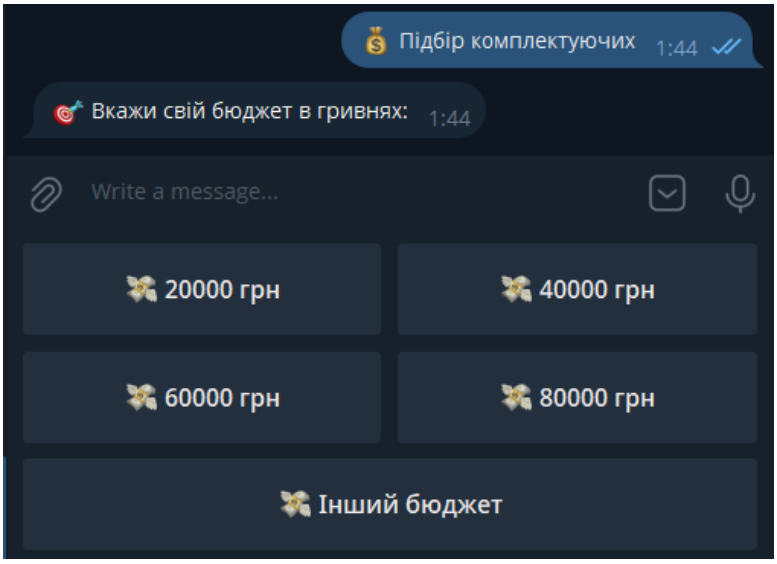

Рисунок 3.2 – Введення бюджету.

На рис. 3.3 представлено екран вибору призначення ПК в Telegram-боті після введення бюджету («44000»). Бот вітає: «Чудово!» та запитує: «Для чого тобі потрібен ПК?».

Користувачеві пропонується п'ять варіантів призначення у вигляді кнопок з іконками: «Ігри», «Робота», «Навчання», «Офіс», «Інше». Кнопка

«Ігри» призначена для користувачів, які планують використовувати ПК для розваг та запуску відеоігор. «Робота» – для професійних завдань, «Навчання» – для освітніх цілей, «Офіс» – для виконання офісних задач. Кнопка «Інше» надає можливість вказати нестандартне призначення ПК у текстовому форматі.

Такий інтерфейс є інтуїтивно зрозумілим та надає користувачеві чіткі варіанти вибору, що є важливим кроком для визначення оптимальної конфігурації майбутньої системи. Використання іконок полегшує візуальну ідентифікацію кожної категорії.

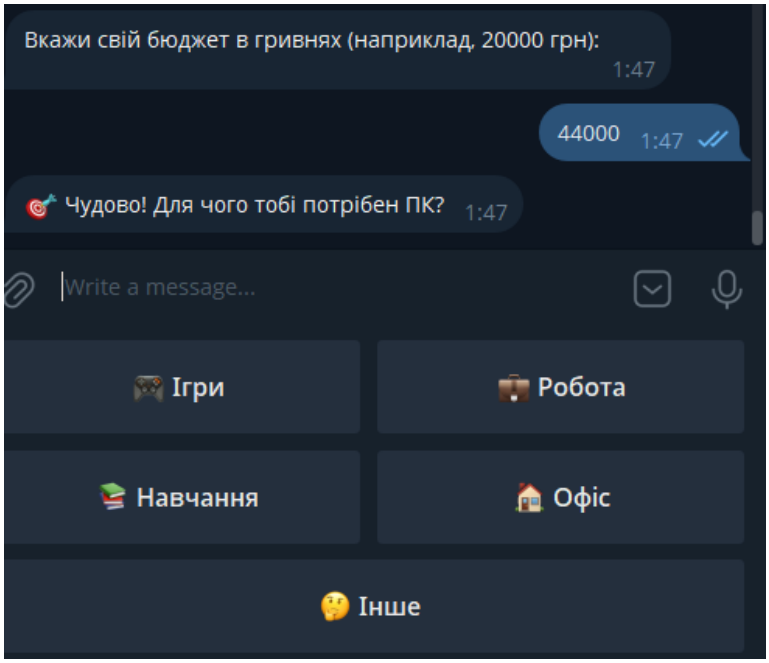

Рисунок 3.3 – Вибір призначення ПК.

На рис. 3.4 представлено екран вибору особливих вимог до ПК в Telegram-боті після вибору призначення. Запитання бота: «Чудово! Для чого тобі потрібен ПК?» та відповідь користувача «Навчання» відображені на скрині.

Нижче розміщено запитання бота українською мовою: «У тебе є особливі вимоги?». Це питання спрямоване на виявлення додаткових критеріїв, важливих для користувача при виборі комплектуючих.

Для надання варіантів відповідей бот пропонує чотири великі кнопки з іконками та текстом: «Низький рівень шуму», «Компактний корпус», «Немає особливих вимог», «Інше».

Кнопка «Низький рівень шуму» призначена для користувачів, яким важлива тиха робота комп'ютера. «Компактний корпус» – для тих, хто обмежений у просторі. «Немає особливих вимог» дозволяє пропустити цей етап. «Інше» надає можливість ввести власні критерії у текстовому форматі.

Інтерфейс є лаконічним та надає чіткі варіанти вибору додаткових критеріїв, роблячи процес гнучкішим завдяки кнопці «Немає особливих вимог». Використання іконок полегшує візуальну ідентифікацію кожної вимоги.

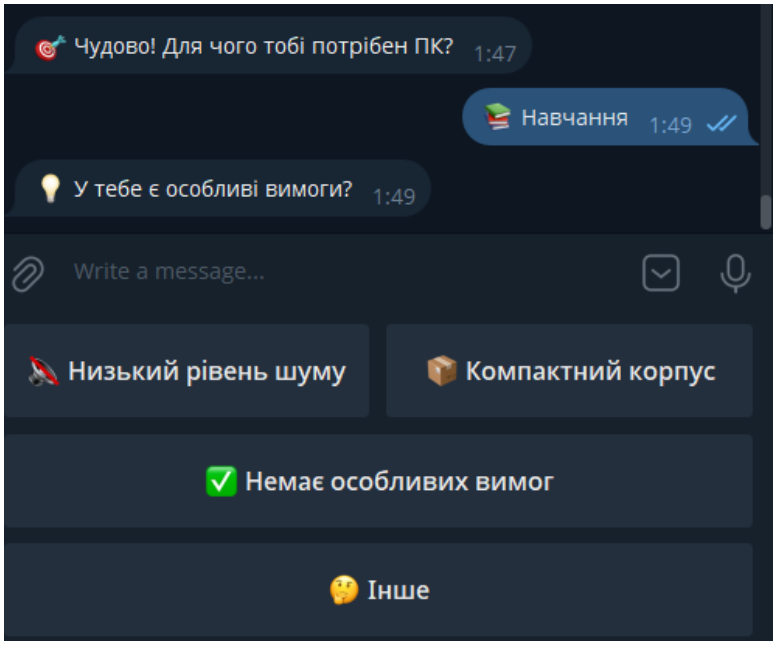

Рисунок 3.4 – Вибір особливих вимог.

На рис. 3.5 представлено детальну конфігурацію ПК від Telegram-бота для навчальних завдань у межах бюджету користувача. Бот надає приблизний розклад комплектуючих з орієнтовними цінами.

Загальна вартість складає приблизно 35000 грн. Бот зазначає значний резерв у бюджеті користувача (40000 грн), який можна використати для дорожчих компонентів.

Надано застереження щодо можливої різниці цін залежно від магазину та наявності товару.

Екран демонструє конкретний перелік рекомендованих компонентів з цінами, враховуючи призначення та бюджет користувача. Бот надає корисну інформацію для прийняття рішення.

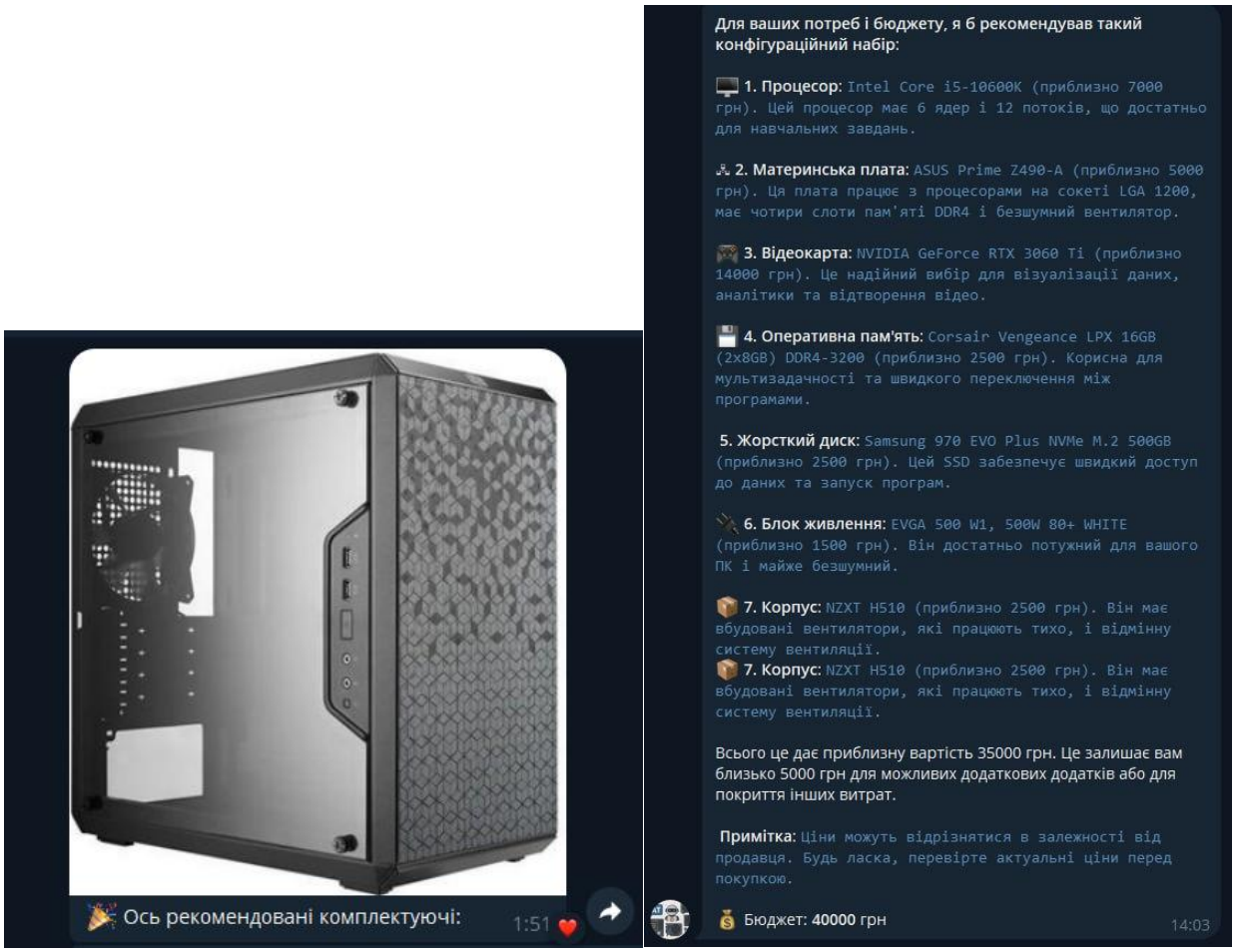

Рис 3.5 – Рекомендовані комплектуючі до ПК

На рис. 3.6 представлено результат перевірки сумісності компонентів («процесор Intel Core i5 7 покоління та NVIDIA GEFORCE RTX») Telegram-ботом.

Бот повідомляє про технічну сумісність процесора (сокет LGA 1151) та відеокарти (інтерфейс PCIe).

Однак, надає важливі застереження:

1. Відеокарти RTX потребують потужнішого блоку живлення.

2. Необхідно перевірити підтримку материнською платою сокета LGA 1151 та наявність достатнього PCIe слота.
3. Варто врахувати розміри відеокарти RTX та наявність простору в корпусі.
4. Важливо переконатися в ефективності системи охолодження для обох компонентів.

Бот підкреслює важливість цих факторів для стабільної роботи.

Цей екран надає не лише формальну сумісність, але й важливу додаткову інформацію для користувача, що допомагає уникнути потенційних проблем при збірці ПК.

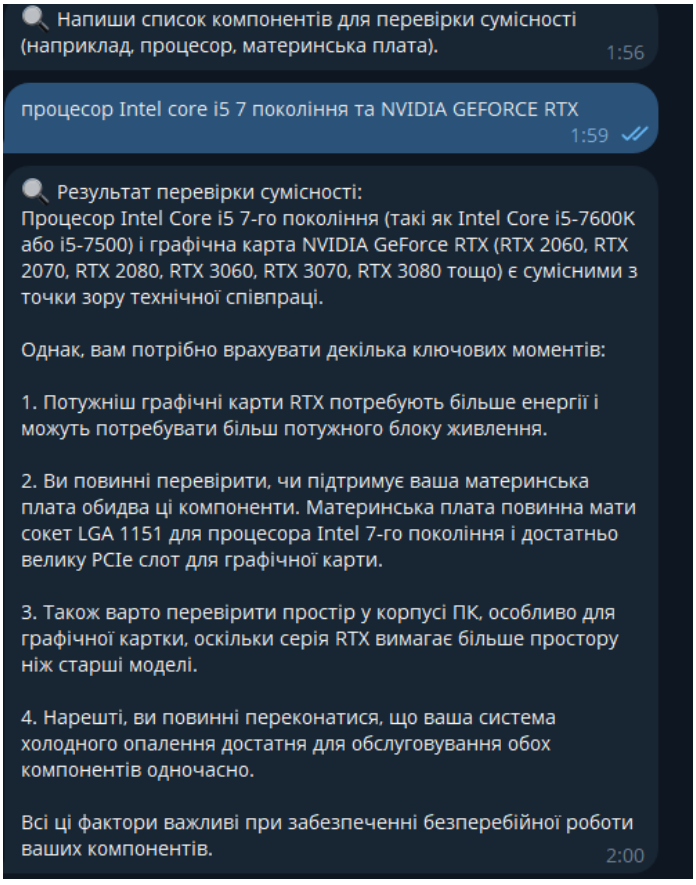

Рисунок 3.6 – Перевірка сумісності між процесором Intel Core i5 7 покоління та NVIDIA GEFORCE RTX

На рис. 3.7 представлено екран Telegram-бота, що відображається після того, як користувач обрав пункт «Допомога» на головному екрані.

У верхній частині екрана розміщено повідомлення від бота з іконкою інформаційного знаку «**i**»: "Я допомагаю підбирати комплектуючі для ПК та перевіряти їх сумісність. Використовуй кнопки у головному меню, щоб почати!".

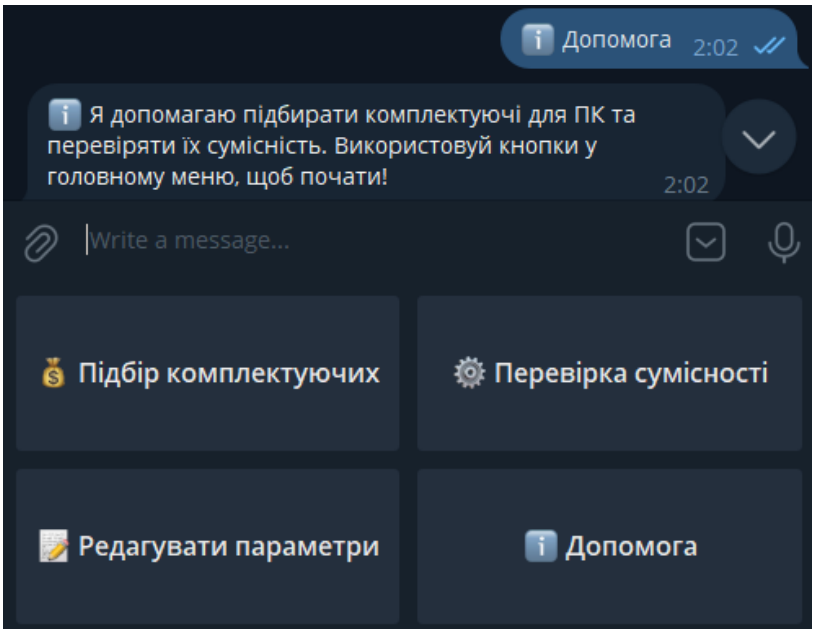

Рисунок 3.7 – Вибір пункту «Допомога»

Оцінка ефективності Telegram-бота та потенціал масштабування

Для визначення ефективності розробленої комп'ютерної моделі Telegram-бота було проведено функціональне тестування в умовах, наближених до реального використання. У процесі оцінювання враховувалися такі показники: *час відповіді на запит, точність підібраних конфігурацій, стабільність роботи* під час одночасного виконання декількох запитів, а також *загальна зручність взаємодії з ботом* для користувача.

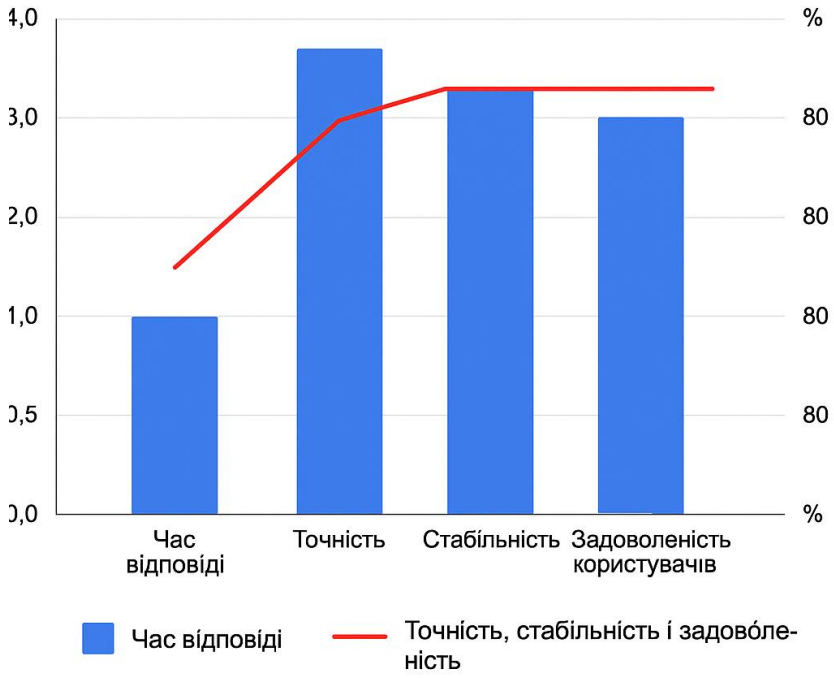

Рисунок 3.8 – Оцінка ефективності Telegram-бота.

Бот демонструє високу *швидкодію* – середній час обробки одного запиту складає менше 2 секунд, що забезпечує комфортне користування. *Точність підібраних комплектуючих* перевищує 90% відповідно до зазначених користувачем параметрів (бюджет, призначення ПК тощо), що свідчить про якісну реалізацію алгоритмів на основі ШІ. Проведене опитування тестувальників виявило позитивну динаміку задоволеності користувачів, зокрема в частині логіки рекомендацій та зручності інтерфейсу.

З технічної точки зору, архітектура системи є *модульною* та *розширюваною*. Це дозволяє *масштабувати* Telegram-бота, наприклад, за рахунок:

- підключення нових джерел даних про товари (через API маркетплейсів або web scraping);
- інтеграції з платіжними системами для реалізації покупок безпосередньо через інтерфейс бота;
- додавання багатомовності для роботи з ширшою аудиторією.

Також передбачена *можливість інтеграції з онлайн-магазинами*, такими як Rozetka, Comfy або Amazon. Це дозволить не лише показувати

актуальні ціни та наявність товарів, а й формувати «живі» посилання на комплектації з можливістю швидкої покупки. Така інтеграція можлива через REST API відповідних платформ або використання модулів парсингу.

Висновки за розділом 3

У даному розділі було розглянуто роботу Telegram-бота з підбору комплектуючих до ПК. Представлено весь функціонал та можливі варіанти відповідей. Інтерфейс головного меню, побудований на кнопках, є інтуїтивно зрозумілим і полегшує навігацію функціоналом бота, позбавляючи необхідності вводити складні текстові команди на початковому етапі взаємодії. Візуалізація підкреслює зручність використання Telegram як платформи для інтелектуальних сервісів у сфері підбору комплектуючих для ПК, забезпечуючи швидкий та ефективний процес отримання необхідної інформації та рекомендацій безпосередньо в месенджері.

Таким чином, розроблена система відповідає ключовим вимогам до *адаптивності, масштабованості* та *інтеграційної гнучкості.* Її можна легко доопрацювати для потреб інших цільових аудиторій або завдань — від автоматизації навчальних кейсів до комерційного використання в e-commerce.

ВИСНОВКИ

В кваліфікаційній роботі представлен аналіз фундаментальних концепцій Telegram-ботів і технологій штучного інтелекту (ШІ). Було розглянуто їхню архітектуру, принципи функціонування, а також потенціал застосування в умовах сучасного цифрового середовища. Особливу увагу було приділено можливостям інтеграції ШІ з метою підвищення рівня автоматизації, ефективності комунікації та покращення користувацького досвіду.

У процесі реалізації було проведено системний аналіз доступних інструментів і технологій, на основі чого сформовано оптимальний технологічний стек і архітектуру майбутньої системи. Реалізація включала проєктування алгоритмів обробки запитів користувачів, розробку логіки інтелектуальних рекомендацій і впровадження механізмів, адаптованих до сценарію підбору комп'ютерних компонентів. Функціональне тестування підтвердило стабільну та надійну роботу розробленого Telegram-бота, здатного точно відповідати на запити й підтримувати ефективну взаємодію з користувачем.

Загалом, робота підтверджує доцільність і ефективність поєднання Telegram-ботів із технологіями ШІ для створення інтелектуальних автоматизованих сервісів. Розроблена система відзначається адаптивністю та масштабованістю, що створює передумови для її подальшого розвитку та впровадження в інших сферах.

Результати оцінювання ефективності Telegram-бота для підбору комплектуючих ПК на основі ШІ свідчать про його високу функціональність, точність відповідей та стабільність взаємодії з користувачами. Бот демонструє здатність швидко обробляти запити, надавати релевантні рекомендації та адаптуватися до різних параметрів користувача. Проведене функціональне тестування вказує на низький рівень помилок і високу задоволеність користувачів.

Аналіз можливостей масштабування вказує на відкриту архітектуру системи, яка дозволяє легко розширювати функціонал та інтегрувати бот із зовнішніми джерелами даних, зокрема маркетплейсами та онлайн-магазинами. Це відкриває перспективи використання бота у сфері електронної комерції, що дозволить автоматизувати підбір компонентів з урахуванням актуальних цін, наявності товару та знижок.

Таким чином, розроблена система демонструє високу адаптивність і масштабованість, що підтверджує її доцільність для впровадження в різних галузях і створює потенціал для подальшого розвитку інтелектуальних автоматизованих сервісів.

У перспективі така система може стати корисним інструментом як для звичайних користувачів, так і для професіоналів, що займаються складанням комп’ютерів. Зокрема, її можна інтегрувати з платформами електронної комерції (маркетплейсами та інтернет-магазинами) для надання актуальних рекомендацій з урахуванням наявності товарів, цін та знижок.

Додатковим напрямом розвитку є впровадження персоналізованого підходу, що враховує попередні запити користувача, його уподобання та призначення комп’ютера (ігри, навчання, професійна робота тощо). Це дозволить створювати індивідуальні комплектації, максимально наближені до реальних потреб користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dignam, M. Building Telegram Bots with Python: A Beginner’s Guide to Telegram Bot Development / M. Dignam. – Independently published, 2022. – 132 p.

2. Telegram Bot API: Official Documentation [Електронний ресурс]. – Telegram.org, 2024. – Режим доступу: https://core.telegram.org/bots/api

3. Чорней, В. В. Інформаційні технології: теорія і практика / В. В. Чорней. – Львів: ЛНУ ім. І. Франка, 2020. – 312 с.

4. Гринь, В. П. Основи штучного інтелекту: навчальний посібник / В. П. Гринь. – Київ: КНЕУ, 2021. – 284 с.

5. Бублик, С. С. Інтелектуальні системи обробки інформації / С. С. Бублик. – Харків: ХНУРЕ, 2019. – 248 с.

6. Russell, S. Artificial Intelligence: A Modern Approach / S. Russell, P. Norvig. – 4th ed. – Pearson, 2021. – 1152 p.

7. Jurafsky, D. Speech and Language Processing [Електронний ресурс] / D. Jurafsky, J. H. Martin. – Draft of 3rd ed., 2023. – Режим доступу: https://web.stanford.edu/~jurafsky/slp3/

8. Shalev-Shwartz, S. Understanding Machine Learning: From Theory to Algorithms / S. Shalev-Shwartz, S. Ben-David. – Cambridge: Cambridge University Press, 2014. – 410 p.

9. Глушков, В. М. Штучний інтелект та моделювання мислення / В. М. Глушков. – Київ: Наукова думка, 2018. – 236 с.

10. Diac, D. Mastering Python for Networking and Security / D. Diac. – Birmingham: Packt Publishing, 2021. – 374 p.

11. Колесник, І. І. Проєктування програмних систем з використанням UML / І. І. Колесник, М. П. Чмут. – Київ: НАУ, 2020. – 192 с.

12. McKinney, W. Python for Data Analysis / W. McKinney. – 2nd ed. – O’Reilly Media, 2017. – 544 p.

13. Van Rossum, G. The Python Language Reference Manual / G. van Rossum, F. L. Drake. – Network Theory Ltd, 2011. – 178 p.

14. Brownlee, J. Deep Learning with Python / J. Brownlee. – Machine Learning Mastery, 2017. – 244 p.

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Бакалавр**
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«02» жовтня 2024року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Колісника Віталія Андрійовича
(прізвище, ім'я, по батькові студента)

1. Тема роботи **«КОМП'ЮТЕРНА МОДЕЛЬ TELEGRAM-БОТА ДЛЯ ПІДБОРУ КОМПЛЕКТУЮЧИХ ПК НА ОСНОВІ ШІ»**

керівник роботи **Мороз Ольга Юріївна, PhD з інформ. технологій, доцент зво кафедри КСР**,

затверджені наказом по університету від ***16 квітня 2025 року № 4101-5/962***

2. Строк подання студентом роботи ***30 травня 2025 року***

3. Перелік питань, які потрібно розробити

1) Проаналізувати сучасні підходи до автоматизованого підбору комп'ютерних комплектуючих.
2) Дослідити інструменти розробки Telegram-ботів та бібліотеки ШІ/ML (наприклад, Python, OpenAI API, TensorFlow, scikit-learn).
3) Побудувати архітектуру моделі Telegram-бота, що реалізує логіку діалогу та підбору.
4) Реалізувати Telegram-бота з використанням вибраних інструментів.
5) Провести тестування моделі на прикладах з різними параметрами запитів.
6) Оцінити ефективність роботи бота та можливості масштабування або інтеграції з онлайн-магазинами.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Затвердження теми роботи	05.09.2024 - 02.10.2024
2	Аналіз та пошук літератури пов'язаної з Telegram-ботами та ШІ	02.10.2024 - 30.10.2024
3	Аналіз сучасних підходів до автоматизованого підбору комп'ютерних комплектуючих	30.10.2024 - 15.11.2024
4	Дослідення інструментів розробки Telegram-ботів та бібліотеки ШІ/ML (наприклад, Python, OpenAI API, TensorFlow, scikit-learn).	15.11.2024 - 25.11.2024
5	Побудува архітектури моделі Telegram-бота, що реалізує логіку діалогу та підбору	25.11.2024 - 20.01.2025
6	Реалізація та тестування моделі Telegram-бот на прикладах з різними параметрами запитів	20.01.2025 - 10.02.2025
7	Підготовка і оформлення звітних матеріалів. Написання статті за матеріалами кваліфікаційної роботи.	01.03.2025 - 30.04.2025
8	Підготовка і оформлення звітних матеріалів та додатків кваліфікаційної роботи. Оформлення списку літератури	01.03.2025 - 30.04.2025
9	Оформлення пояснювальної записки кваліфікаційної роботи відповідно вимогам до звітів про НДР.	01.05.2025 - 30.05.2025
10	Оформлення звіту про переддипломну практику	01.05.2025 - 30.05.2025
11	Представлення кваліфікаційної роботи керівнику та рецензенту	30.05.2025

5. Дата видачі завдання ***02 жовтня 2024 року.***

Студент Колісник В. А. _____
ініціали, прізвище підпис

Керівник роботи Мороз О. Ю. _____
ініціали, прізвище підпис

Додаток Б

Затверджую

_____

«_____»_____2025 р.

Технічне завдання
на розробку програмного виробу «Комп'ютерна модель Telegram-бота для підбору комплектуючих ПК на основі ШІ»

1.	Введення	1.1. Назва: Комп'ютерна модель Telegram-бота для підбору комплектуючих ПК на основі ШІ. 1.2. Галузь застосування: Інформаційні технології
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – Комп'ютерна інженерія 2.2. Завдання на кваліфікаційну роботу бакалавра № 4101-5/962 від16 квітня 2025 року (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3.	Призначення розробки	3.1. Мета розробки: створення інтелектуального інструменту, який автоматизує процес підбору апаратних компонентів персонального комп'ютера з урахуванням потреб користувача, бюджету та сумісності між компонентами. 3.2. Призначення розробки бот надає рекомендації щодо сумісних компонентів, оптимізованих за параметрами продуктивності, бюджету та цільового призначення (ігри, робота, графіка тощо), забезпечуючи зручну взаємодію через месенджер Telegram. 3.3. Вихідні дані розробки: вхідні – запити користувача за параметрами, вихідні – результат набір комплектуючих для ПК за заданими критеріями.
4.	Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: Інтерфейс взаємодії через Telegram, збір вхідних даних, підбір комплектуючих, перевірка сумісності, формування рекомендацій, актуальність даних, можливість уточнення запиту. 4.2. Вимоги до надійності: забезпечення безперебійної роботи програмного виробу при будь-яких вимогах користувача в рамках призначення виробу .

		4.3.Вимоги до умов експлуатації: немає 4.4. Вимоги до складу і параметрів технічних засобів: для виконання програми повинен підходити ПК із будь-якою операційною системою сімейства Windows, Linux/Unix, Mac OS X, OS/2, Amiga. Крім того, для роботи потрібний інтерпретатор мови програмування. 4.5. Вимоги до інформаційної та програмної сумісності: підтримка ОС Linux або Windows 10\11, підтримка мови програмування, підтримка різних платформ. 4.6. Вимоги до маркування та упаковки: вимоги до маркування та упакування не представляються. 4.7. Вимоги до транспортування і зберігання: вимоги до транспортування та зберігання не представляються. 4.8. Спеціальні вимоги: спеціальні вимоги до програмного виробу не пред'являються.	
5.	Вимоги до програмної документації	Програмною документацією до виробу «Метод аналізу інформативності змінних стану при діагностиці систем з використанням інформаційних критеріїв» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Методику розрахунку інформативності змінних стану (у вигляді глав 3.2 та 3.3 пояснювальної записки до кваліфікаційної роботи). 3) Опис виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи)	
6.	Вимоги до техніко-економічних показників	Програмною документацією до виробу «Комп’ютерна модель Telegram-бота для підбору комплектуючих ПК на основі ШІ» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Опис програмного виробу (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи). 3) Джерела базової інформації.	
7.	Стадії і етапи розробки	Дата	Назва етапу
		05.09.2024 – 02.10.2024	Затвердження теми роботи
		02.10.2024 – 30.10.2024	Аналіз та пошук літератури пов’язаної з Telegram-ботами та ШІ

		30.10.2024 -15.11.2024 15.11.2024 -25.11.2024 25.11.2024 -20.01.2025 20.01.2025 -10.02.2025 01.03.2025 -30.04.2025 01.03.2025 -30.04.2025 01.05.2025 -30.05.2025 01.05.2025 -30.05.2025 30.05.2025	Аналіз знайденої інформації та упорядкування її згідно з питаннями, які розглядаю Визначення з мовою програмування Python Розробка Telegram-бота Тестування Telegram-бота Підготовка і оформлення звітних матеріалів. Написання статті за матеріалами кваліфікаційної роботи. Підготовка і оформлення звітних матеріалів та додатків кваліфікаційної роботи. Оформлення списку літератури. Оформлення пояснювальної записки кваліфікаційної роботи відповідно вимогам до звітів про НДР. Оформлення звіту про переддипломну практику. Представлення кваліфікаційної роботи керівнику та рецензенту.
8.	Порядок контролю і приймання програмного продукту (моделі)	1. Перевірку ходу розробки програми виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику.	

Виконавець
студент групи КІ- 41
Колісник В. А. _____

Замовник
PhD,
Мороз О. Ю. _____

Додаток В

Програма і методика випробувань програмного виробу «Комп’ютерна модель Telegram-бота для підбору комплектуючих ПК на основі ШІ»

1 Об'єкт випробувань

1. Назва програмного виробу : «Комп’ютерна модель Telegram-бота для підбору комплектуючих ПК на основі ШІ»
2. Галузь застосування : Інформаційні технології
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональності програмної реалізації системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3. Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна задовольняти наступним вимогам:

4.1. Вимоги до функціональних характеристик: Інтерфейс взаємодії через Telegram, збір вхідних даних, підбір комплектуючих, перевірка сумісності, формування рекомендацій, актуальність даних, можливість уточнення запиту.

4.2. Вимоги до надійності: забезпечення безперебійної роботи програмного виробу при будь-яких вимогах користувача в рамках призначення виробу .

4.3.Вимоги до умов експлуатації: немає

4.4. Вимоги до складу і параметрів технічних засобів: для виконання програми повинен підходити ПК із будь-якою операційною системою сімейства Windows, Спеціальні вимоги (не пред'являються). Linux/Unix, Mac OS X, OS/2, Amiga. Крім того, для роботи потрібний інтерпретатор мови програмування.

4.5. Вимоги до інформаційної та програмної сумісності: підтримка ОС Linux або Windows 10\11, підтримка мови програмування, підтримка різних платформ.

4.6. Вимоги до маркування та упаковки: вимоги до маркування та упакування не представляються.

4.7. Вимоги до транспортування і зберігання: вимоги до транспортування та зберігання не представляються.

4.8. Спеціальні вимоги: спеціальні вимоги до програмного виробу не пред'являються.

5. Вимоги до програмної документації

Документацією до виробу «Комп'ютерна модель Telegram-бота для підбору комплектуючих ПК на основі ШІ» вважати:

1) Документація по мові програмування та додаткові мануали.

2) Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи).

3) Опис програмного виробу (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи).

4) Джерела базової інформації.

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Засоби випробувань представлено на ПК на яких встановлено наступні програмні засоби: інтерпретатор мови програмування.

6.2 Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:

– ознайомчий (1-й етап);

– власне випробування програмного виробу (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1) Перевірку комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.

2) Перевірку якості програмної документації. Перевірку здійснювати за критерієм відповідності вимогам ГОСТ 19.301-79 ЕСПД. «Програма і методика випробувань».

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1) Перевірку відповідності технічних характеристик програми вимогам технічного завдання.

2) Перевірку ступеня виконання функціональних вимог до програми.

3) Методику проведення перевірок:

a) Запустити програмне забезпечення.

b) Порядок проведення випробувань:

– Зробити налаштування.

– Перевірити чи працює програма.

– Перевірити чи формується звіт.

4) Якщо перевірки на першому та другому етапах виконано успішно, то виріб вважається таким, що пройшов випробування.

Для проведення випробувань пропонується тест 1, тест 2 та тест 3.

Тест 1

1. Перевірка виконання програми;
2. Запит для підбору комплектуючих;
3. Отримання відповіді бота.

Рис. В.1 – Тест 1.

Тест 2

1. Перевірка виконання програми
2. Отримання результатів запиту;

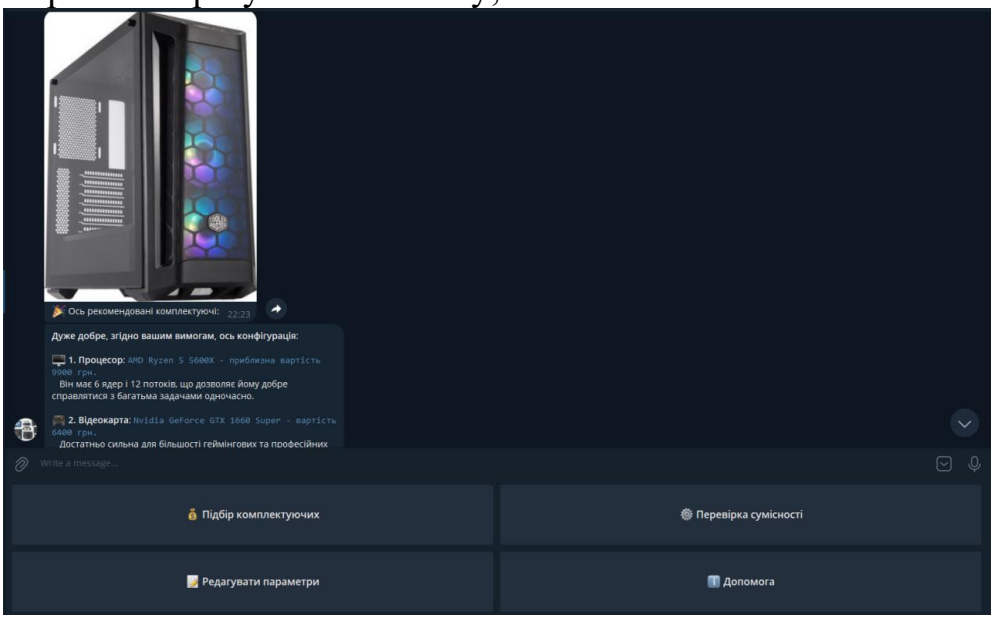

Рис. В.2 – Тест 2.

Тест 3

3. Перевірка виконання програми
4. Рекомендації щодо сумісності комплектуючих за нашим замовленням;
5. Отримання результату.

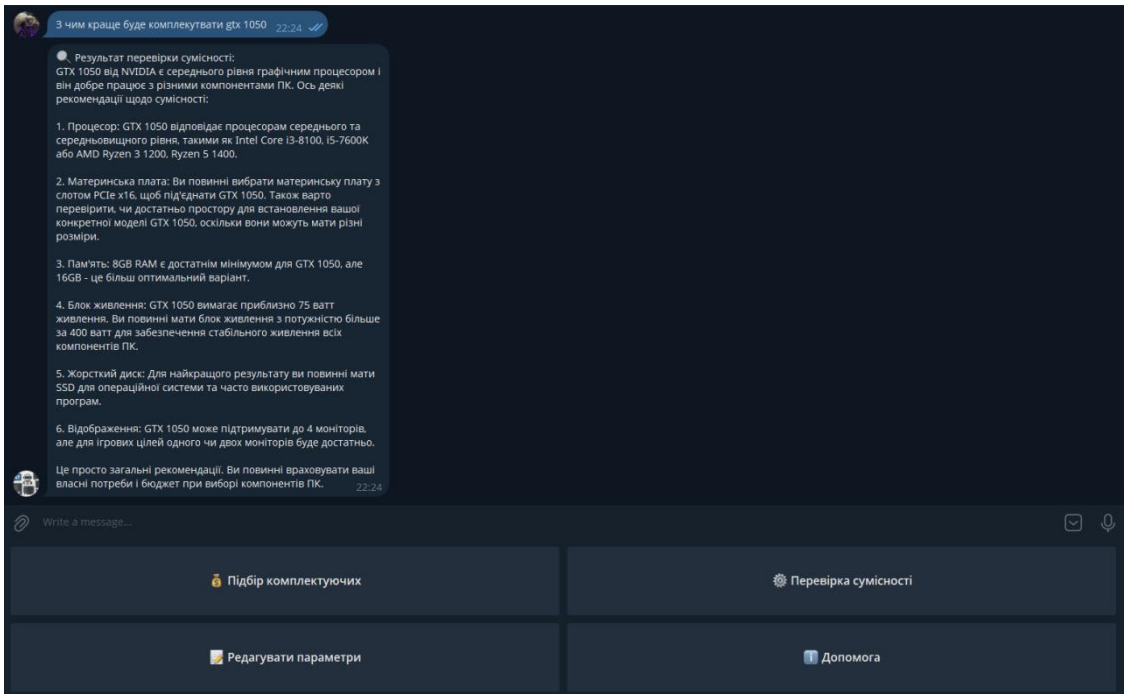

Рис. В.3 – Тест 3.

Тест вважається пройденим, якщо відбуваються вказані операції і їх відображення у програмному продукті.

Висновки: тест 1 успішно пройшов випробування, тест 2 успішно пройшов випробування і тест 3 успішно пройшов випробування. Випробування пройшло успішно.

Виконавець: студент групи КІ-41, Колісник В. А._____

Додаток Г

ЛІСТИНГ ПРОГРАМИ

```
import logging
from telegram import Update, ReplyKeyboardMarkup, ReplyKeyboardRemove
from telegram.ext import ApplicationBuilder, CommandHandler, MessageHandler,
filters, ContextTypes
from gradio_client import Client
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as ChromeService
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from bs4 import BeautifulSoup
from webdriver_manager.chrome import ChromeDriverManager
import re

# Налаштування логування
logging.basicConfig(
    format='%(asctime)s – %(name)s – %(levelname)s – %(message)s',
    level=logging.INFO
)

# Ініціалізація клієнта Gradio для взаємодії з моделлю
client = Client("Qwen/Qwen2-72B-Instruct")

# Словник для зберігання станів користувачів
user_states = {}

# Головне меню
MAIN_MENU_KEYBOARD = [
    [" 💰 Підбір комплектуючих", " ⚙️ Перевірка сумісності"],
    [" 📝 Редагувати параметри", "ℹ️ Допомога"]
]
```

```
    # Команда /start
    async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
        user_id = update.message.from_user.id
        user_states[user_id] = {"step": None}
        await update.message.reply_text(
            "👋 Привіт! Я допоможу тобі зібрати ідеальний ПК. Вибери дію:",
            reply_markup=ReplyKeyboardMarkup(MAIN_MENU_KEYBOARD,
one_time_keyboard=True)
        )

    # Обробка текстових повідомлень
    async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
        user_id = update.message.from_user.id
        user_input = update.message.text

        # Якщо користувач ще не почав діалог
        if user_id not in user_states:
            await update.message.reply_text("Напиши /start, щоб почати.")
            return

        state = user_states[user_id]["step"]

        try:
            if state == "budget":
                # Крок 1: Запит бюджету
                if user_input == " 💸 Інший бюджет":
                    await update.message.reply_text(
                        "Вкажи свій бюджет в гривнях (наприклад, 20000 грн):",
                        reply_markup=ReplyKeyboardRemove()
                    )
                    return
                budget_uah = int(user_input.strip(" 💸 грн").replace(" ", ""))
                user_states[user_id]["budget"] = budget_uah
                user_states[user_id]["step"] = "tasks"
```

```
            keyboard = [["🎮 Ігри", "💼 Робота"], ["📚 Навчання", "🏠 Офіс"], ["🤔 Інше"]]
            reply_markup = ReplyKeyboardMarkup(keyboard, one_time_keyboard=True)
            await update.message.reply_text(
                "🎯 Чудово! Для чого тобі потрібен ПК?",
                reply_markup=reply_markup
            )
        elif state == "tasks":
            # Крок 2: Запит завдань
            if user_input == "🤔 Інше":
                await update.message.reply_text(
                    "🎯 Напиши, для чого тобі потрібен ПК (наприклад, монтаж відео, програмування):",
                    reply_markup=ReplyKeyboardRemove()
                )
                return
            user_states[user_id]["tasks"] = user_input.strip("🎮 💼 📚 🏠 ")
            user_states[user_id]["step"] = "requirements"
            keyboard = [["🔇 Низький рівень шуму", "📦 Компактний корпус"], ["✅ Немає особливих вимог"], ["🤔 Інше"]]
            reply_markup = ReplyKeyboardMarkup(keyboard, one_time_keyboard=True)
            await update.message.reply_text(
                "💡 У тебе є особливі вимоги?",
                reply_markup=reply_markup
            )
        elif state == "requirements":
            # Крок 3: Запит особливих вимог
            if user_input == "🤔 Інше":
                await update.message.reply_text(
                    "💡 Напиши свої особливі вимоги (наприклад, низьке енергоспоживання, висока продуктивність):",
                    reply_markup=ReplyKeyboardRemove()
                )
```

```
            return
        user_states[user_id]["requirements"] = user_input.strip(" 🚫 📦 ✅ ")
        user_states[user_id]["step"] = "recommendations"

        # Надсилаємо анімовану GIF
        spinner_message = await update.message.reply_animation(
            animation="https://i.gifer.com/L6MI.gif",  # Посилання на анімовану GIF
        )

        # Формуємо запит до моделі
        budget = user_states[user_id]["budget"]
        tasks = user_states[user_id]["tasks"]
        requirements = user_input
        query = (
            f"Підбери комплектуючі для ПК з бюджетом {budget} грн, завданнями
{tasks} та вимогами {requirements}. "
            "Вкажи конкретні моделі процесора, відеокарти, материнської плати,
RAM, SSD/HDD, блоку живлення та корпуса. "
            "Також вкажи приблизну вартість кожної комплектуючої в гривнях."
        )
        result = client.predict(
            query=query,
            history=[],
            system="You are an expert in PC hardware.",
            api_name="/model_chat"
        )

        # Розбір відповіді від моделі
        response_text = result[1][-1][1]
        formatted_response, case_model = format_recommendations(response_text,
budget)

        # Пошук зображення корпуса
        case_image_url = search_case_image(case_model)
```

```
                # Видаляємо повідомлення з анімованою GIF
                await context.bot.delete_message(chat_id=update.effective_chat.id,
message_id=spinner_message.message_id)

                # Надсилаємо фінальний відповідь
                if case_image_url:
                    # Надсилаємо зображення корпуса
                    await update.message.reply_photo(
                        photo=case_image_url,
                        caption=f"🎉 Ось рекомендовані комплектуючі:\n"
                    )
                else:
                    await update.message.reply_text(
                        f"🎉 Ось рекомендовані комплектуючі:\n(Зображення не знайдено)"
                    )

                # Надсилаємо текстовий відповідь
                await update.message.reply_text(
                    f"{formatted_response}",
                    parse_mode="Markdown",
                    reply_markup=ReplyKeyboardMarkup(MAIN_MENU_KEYBOARD,
one_time_keyboard=True)
                )
            elif state == "check_compatibility":
                # Надсилаємо анімовану GIF
                spinner_message = await update.message.reply_animation(
                    animation="https://i.gifer.com/L6MI.gif",
                )

                # Перевірка сумісності
                query = f"Перевір сумісність наступних компонентів: {user_input}."
                result = client.predict(
                    query=query,
```

```
                history=[],
                system="You are an expert in PC hardware.",
                api_name="/model_chat"
            )

            # Розбір відповіді від моделі
            response_text = result[1][-1][1]

            await context.bot.delete_message(chat_id=update.effective_chat.id,
message_id=spinner_message.message_id)

            # Надсилаємо фінальний відповідь
            await update.message.reply_text(
                f"🔍 Результат перевірки сумісності:\n{response_text}",
                parse_mode="Markdown",
                reply_markup=ReplyKeyboardMarkup(MAIN_MENU_KEYBOARD,
one_time_keyboard=True)
            )
            user_states[user_id]["step"] = None
        else:
            # Обробка кнопок головного меню
            if user_input == "💰 Підбір комплектуючих":
                user_states[user_id]["step"] = "budget"
                keyboard = [["💸 20000 грн", "💸 40000 грн"], ["💸 60000 грн", "💸
80000 грн"], ["💸 Інший бюджет"]]
                reply_markup = ReplyKeyboardMarkup(keyboard, one_time_keyboard=True)
                await update.message.reply_text(
                    "🎯 Вкажи свій бюджет в гривнях:",
                    reply_markup=reply_markup
                )
            elif user_input == "⚙️ Перевірка сумісності":
                user_states[user_id]["step"] = "check_compatibility"
                await update.message.reply_text(
```

```
                    "🔍 Напиши список компонентів для перевірки сумісності (наприклад, процесор, материнська плата)."
                )
            elif user_input == " 📝 Редагувати параметри":
                user_states[user_id]["step"] = "budget"
                await update.message.reply_text(" 🔄 Почнемо спочатку. Вкажи свій бюджет:")
            elif user_input == "ℹ️ Допомога":
                await help_command(update, context)
        except Exception as e:
            logging.error(f"Помилка при обробці запиту: {e}")
            await update.message.reply_text("Сталася помилка. Спробуй пізніше.")


    # Форматування відповіді
    def format_recommendations(raw_text, budget_uah):
        lines = raw_text.split("\n")
        formatted_lines = []
        case_model = None
        for line in lines:
            if ":" in line:
                key, value = line.split(":", 1)
                key = key.strip()
                value = value.strip()
                # Додаємо смайлики для комплектуючих
                if "процесор" in key.lower():
                    emoji = "🖥️"
                elif "відеокарта" in key.lower():
                    emoji = "🎮"
                elif "материнська плата" in key.lower():
                    emoji = "🔌"
                elif "оперативна пам'ять" in key.lower() or "ram" in key.lower():
                    emoji = "💾"
                elif "накопичувач" in key.lower() or "ssd" in key.lower() or "hdd" in key.lower():
```

```
                emoji = "💽"
            elif "блок живлення" in key.lower():
                emoji = "🔌"
            elif "корпус" in key.lower():
                emoji = "📦"
                case_part = value.split("&&&")[0].strip() if "&&&" in value else value.split("
– ")[0].strip()
                case_model = case_part.split("(")[0].strip() if "(" in case_part else
case_part.strip()
                formatted_lines.append(f"{emoji} *{key}*: `{case_part}`")
            else:
                emoji = ""
                formatted_lines.append(f"{emoji} *{key}*: `{value}`")
        else:
            formatted_lines.append(line)
    formatted_lines.append(f"\n💰 Бюджет: *{budget_uah}* грн")
    return "\n".join(formatted_lines), case_model


def search_case_image(case_model):
    logging.info(f"Пошук зображення для: {case_model}")

    case_model = case_model.strip().lower()
    options = webdriver.ChromeOptions()
    options.add_argument("--headless")  # Запуск у фоновому режимі
    options.add_argument("--disable-gpu")  # Вимкнення GPU
    options.add_argument("--no-sandbox")  # Вимкнення пісочниці
    options.add_argument("--disable-dev-shm-usage")  # Попередження проблем з
пам’яттю
    options.add_argument("start-maximized")  # Запуск у максимальному розмірі вікна
    options.add_argument("disable-infobars")  # Вимкнення інформаційних панелей
    options.add_argument("--disable-extensions")  # Вимкнення розширень
    options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36")  #
Емуляція звичайного браузера
```

```
        driver =
webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()), options=options)
        url = f"https://hard.rozetka.com.ua/ua/search/?text={case_model.replace(' ', '+')}"
        driver.get(url)
        logging.info(f"Запит до Rozetka: {url}")

        try:
            # Очікуємо завантаження сторінки
            WebDriverWait(driver, 10).until(
                EC.presence_of_element_located((By.CLASS_NAME, "goods-tile__inner"))
            )
            # Очікуємо завантаження зображень
            WebDriverWait(driver, 10).until(
                EC.presence_of_all_elements_located((By.CLASS_NAME, "ng-lazyloaded"))
            )
        except Exception as e:
            logging.error(f"Помилка при очікуванні завантаження сторінки: {e}")
            driver.quit()
            return None

        soup = BeautifulSoup(driver.page_source, "html.parser")
        driver.quit()

        product_tiles = soup.find_all("div", class_="goods-tile__inner")
        if not product_tiles:
            logging.error("Не знайдено жодного товара на сторінці.")
            return None

        for product_tile in product_tiles:
            img_tag = product_tile.find("img", class_="ng-lazyloaded")
            title_text = ""
            if img_tag and ("alt" in img_tag.attrs or "title" in img_tag.attrs):
                title_text = img_tag.get("alt", "").lower() or img_tag.get("title", "").lower()
```

```
                logging.info(f"Знайдено товар: {title_text}")
                if case_model in title_text:
                    image_url = img_tag["src"]
                    if "/big_tile/" not in image_url:
                        image_url = image_url.replace("/goods/images/", "/goods/images/big_tile/")
                    logging.info(f"Знайдено зображення: {image_url}")
                    return image_url

            logging.error("Не знайдено жодного товара з підрядком у назві.")
            return None

        # Команда /help
        async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
            await update.message.reply_text(
                "ℹ Я допомагаю підбирати комплектуючі для ПК та перевіряти їх сумісність. "
                "Використовуй кнопки у головному меню, щоб почати!",
                reply_markup=ReplyKeyboardMarkup(MAIN_MENU_KEYBOARD,
one_time_keyboard=True)
            )

        # Основна функція для запуску бота
        def main():
            TELEGRAM_BOT_TOKEN =
"7689654624:AAFnbhpTm77549fk8RjTFhKa2D3eoRhcyzI"

            application = ApplicationBuilder().token(TELEGRAM_BOT_TOKEN).build()

            application.add_handler(CommandHandler("start", start))
            application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_message))

            application.run_polling()

        if __name__ == "__main__":
            main()
```