МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ДЮГ Дмитро Анатолійович

Метод інтеграції штучного інтелекту до Telegram-боту / Method of Integrating Artificial Intelligence into a Telegram Bot

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КНм-21
Д. А. Дюг
_____
Науковий керівник:
д.т.н., професор М.П. Комар
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
Завідувач кафедри
_____ М.П. Комар

ТЕРНОПІЛЬ - 2023

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ДЮГ Дмитро Анатолійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Метод інтеграції штучного інтелекту до Telegram-боту / Method of Integrating Artificial Intelligence into a Telegram Bot
керівник роботи д.т.н., професор М.П. Комар
затверджені наказом по університету від 8 грудня 2022 року № 491.

2. Строк подання студентом закінченої кваліфікаційної роботи 1 грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити
- огляд предметної області;
- аналіз основних можливостей використання ChatGPT;
- огляд основних можливостей управління чат-ботом у Telegram;
- постановка задачі;
- концептуальні основи інтеграції штучного інтелекту до Telegram-боту;
- метод інтеграції штучного інтелекту до Telegram-боту;
- засоби розробки та інтеграції штучного інтелекту до Telegram-боту;
- розробка чат-боту;
- розробка системи адміністрування Telegram-бота;
- тестування чат-боту та системи адміністрування.

5. Перелік графічного матеріалу у роботі
- схема процесу створення Telegram-боту;
- схема алгоритму роботи Telegram-боту;

- схема логіки чат-боту;
- структура проєкту чат-бота.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 8 грудня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області та постановка задачі дослідження	12.2022 р. – 03.2023 р.	
2	Методи та засоби інтеграції штучного інтелекту до Telegram-боту	03.2023 р. – 05.2023 р.	
3	Реалізація та тестування чат-боту та системи адміністрування	05.2023 р. – 11.2023 р.	
4	Повне завершення та представлення кваліфікаційної роботи на кафедру	01.12.2023 р.	

Студент _____ Д.А. Дюг
підпис

Керівник роботи _____ д.т.н., професор М.П. Комар
підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Метод інтеграції штучного інтелекту до Telegram-боту» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 79 сторінок і містить 18 ілюстрацій, 2 таблиці, 2 додатки та 47 використаних джерел.

Мета кваліфікаційної роботи полягає в дослідженні та розробці методів інтеграції штучного інтелекту з Telegram-ботом, що включає аналіз потенціалу штучного інтелекту для покращення функціональності бота, розробку і впровадження відповідних алгоритмів, а також оцінку ефективності взаємодії з користувачами.

Методи досліджень: теоретичний аналіз, системний аналіз, моделювання та симуляція, експериментальні методи.

Результати дослідження: досліджено методи адаптації алгоритмів машинного навчання для специфічних потреб і можливостей Telegram-ботів, що включає інноваційні підходи до персоналізації взаємодій бота з користувачами, покращення точності та контекстуальної релевантності відповідей бота, а також розробку нових сценаріїв використання штучного інтелекту в чат-ботах.

Результати роботи можуть бути використані для створення більш розумних, ефективних та адаптивних ботів, які здатні автоматично вирішувати задачі користувачів, забезпечувати персоналізовані відповіді та покращувати взаємодію з клієнтами у різних сферах, від обслуговування клієнтів до маркетингу та освіти.

Ключові слова: ІНТЕГРАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ, TELEGRAM-БОТ, CHATGPT.

ABSTRACT

Qualification work on the topic «Method of Integrating Artificial Intelligence into a Telegram Bot» for Master's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 79 pages and it contains 17 figures, 2 tables, 2 annexes and 47 sources.

The goal of the qualification work is to explore and develop methods for integrating artificial intelligence with a Telegram bot. This includes analyzing AI's potential to improve bot functionality, developing and implementing appropriate algorithms, and evaluating user interaction effectiveness.

Research methods involve theoretical analysis, system analysis, modeling, simulation, and experimental methods. The results include adapting machine learning algorithms for specific needs of Telegram bots, innovating in personalizing user interactions, improving accuracy and contextual relevance of responses, and developing new AI scenarios in chatbots.

These findings can be used to create smarter, more efficient, and adaptive bots capable of automating tasks, providing personalized responses, and enhancing customer interactions in various fields from customer service to marketing and education.

Keywords: INTEGRATION, ARTIFICIAL INTELLIGENCE, TELEGRAM BOT, CHATGPT.

ЗМІСТ

ВСТУП

Актуальність теми. Штучний інтелект (ШІ) став трансформуючою силою у сучасному світі, переконструюючи різні аспекти життя, від особистих до професійних сфер. Ця стаття досліджує актуальність і вплив ШІ в сучасних умовах, користуючись інсайтами з різних джерел. ШІ вказує на створення та розвиток комп'ютерів та машин, здатних виконувати завдання, обробляти інформацію та приймати рішення незалежно від введення або взаємодії з людьми. В суті, ШІ-системи можуть мислити самостійно, реплікуючи когнітивні функції людини на базовому рівні.

Інтеграція ШІ в повсякденне життя є глибокою та різноманітною. Вона впливає на наші домівки, робочі місця та громадські простори. Від алгоритмів соціальних мереж, які моніторять контент, до медичного ШІ, який допомагає в діагностиці хвороб, ШІ є невід'ємною частиною сучасного способу життя.

ШІ у повсякденних технологіях:

– розваги: сервіси стрімінгу, такі як Netflix, використовують ШІ для аналізу перегляду та рекомендацій контенту;

– розпізнавання голосу: цифрові асистенти, такі як Amazon's Alexa, використовують ШІ для розуміння та виконання голосових команд;

– охорона здоров'я: ШІ-системи в галузі охорони здоров'я діагностують хвороби, визначаючи патерни в даних пацієнтів;

– автономні транспортні засоби: самохідні автомобілі використовують ШІ для обробки даних з сенсорів для безпечної навігації.

Також, ШІ важливий у розвитку технологій, і дані науковці та інженери зосереджені на його еволюції. ШІ-системи використовують машинне навчання для визначення патернів та тенденцій, що допомагає їм приймати рішення та виконувати завдання ефективніше.

У сфері бізнесу ШІ відіграє ключову роль в різних відділах:

– маркетинг: ШІ-системи передбачають вподобання споживачів для спрямованої розстановки продуктів;

– логістика: ШІ оптимізує маршрути доставки та управляє ресурсами, такими як дрони доставки;

– стратегічне планування: ШІ допомагає в прогнозуванні продажів та формуванні стратегій бізнесу;

– виробництво: ШІ допомагає в управлінні ланцюгом постачання та масштабуванні виробництва в залежності від попиту.

Мета і задачі дослідження. Мета кваліфікаційної роботи полягає в дослідженні та розробці методів інтеграції штучного інтелекту з Telegram-ботом. Це включає аналіз потенціалу ШІ для покращення функціональності бота, розробку і впровадження відповідних алгоритмів, а також оцінку ефективності взаємодії з користувачами. Робота зосереджується на розширенні можливостей бота, підвищенні якості обслуговування та забезпеченні більш глибокої персоналізації взаємодії.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– огляд предметної області;

– аналіз основних можливостей використання ChatGPT;

– огляд основних можливостей управління чат-ботом у Telegram;

– постановка задачі;

– концептуальні основи інтеграції штучного інтелекту до Telegram-боту;

– метод інтеграції штучного інтелекту до Telegram-боту;

– засоби розробки та інтеграції штучного інтелекту до Telegram-боту;

– розробка чат-боту;

– розробка системи адміністрування Telegram-бота;

– тестування чат-боту та системи адміністрування.

Об'єкт дослідження. Об'єктом дослідження у кваліфікаційній роботі є процес інтеграції та взаємодії алгоритмів штучного інтелекту з чат-ботом у месенджері Telegram.

Предмет дослідження. Предметом дослідження у кваліфікаційній роботі є методи та технології, що застосовуються для інтеграції штучного інтелекту у чат-боти.

Методи дослідження.

1. Теоретичний аналіз: вивчення наукової літератури, статей та публікацій для зрозуміння основ штучного інтелекту, алгоритмів обробки природної мови та технологій чат-ботів.

2. Практичне моделювання: створення та тестування моделей ШІ для інтеграції з Telegram-ботом, використання програмного забезпечення для емуляції взаємодії між ботом та користувачами.

3. Експериментальне дослідження: реалізація та тестування інтегрованого рішення у реальних умовах для оцінки ефективності, точності відповідей та користувацького досвіду.

4. Аналіз даних: збір та аналіз даних з взаємодій користувачів для оцінки якості інтеграції ШІ з ботом і виявлення можливих шляхів поліпшення.

Наукова новизна одержаних результатів. Наукова новизна кваліфікаційної роботи полягає у розробці методу адаптації алгоритмів машинного навчання для специфічних потреб і можливостей Telegram-ботів, що включає інноваційні підходи до персоналізації взаємодій бота з користувачами, покращення точності та контекстуальної релевантності відповідей бота, а також розробку нових сценаріїв використання штучного інтелекту в чат-ботах.

Практичне значення отриманих результатів. Практичне значення кваліфікаційної роботи полягає у розвитку та оптимізації інтерактивних чат-ботів для поліпшення користувацького досвіду в Telegram. Результати роботи можуть бути використані для створення більш розумних, ефективних та адаптивних ботів, які здатні автоматично вирішувати задачі користувачів, забезпечувати персоналізовані відповіді та покращувати взаємодію з клієнтами у різних сферах, від обслуговування клієнтів до маркетингу та освіти.

Публікації та апробація КР. Результати кваліфікаційної роботи апробовані та опубліковані у матеріалах:

– XI науково-технічної конференції «Інформаційні моделі, системи та технології», Тернопіль, Тернопільський національний технічний університет імені І. Пулюя, 2023 р.

– науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі», Тернопіль, Західноукраїнський національний університет, 2023 р.

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Огляд предметної області

Штучний інтелект – це система, яка імітує поведінку людини за допомогою комп'ютерної системи. Термін "штучний інтелект" відноситься до новітньої галузі комп'ютерних наук, яка прагне розробити машини, здатні автоматизувати людські дії та процеси [1, 2]. Спеціалізовані застосування ШІ включають експертні системи, обробку природної мови та машинне зір. ШІ дозволяє машинам навчатися з досвіду, адаптуватися до змінних вхідних даних та виконувати завдання, подібні до людських [3-6]. Чат-боти є одним з найпоширеніших методів інтерпретації мови через обробку природної мови (NLP) [7, 8].

NLP – це інструмент для вивчення взаємодії людини з комп'ютером за допомогою людської мови. Цей тип комп'ютера полегшує спілкування між людьми та комп'ютерами, дозволяючи їм спілкуватися на своїх рідних мовах. NLP має багато цілей, які можуть допомогти покращити спілкування, такі як машинний переклад та допомога у спілкуванні людини з машиною, наприклад, з розмовними агентами та іншими [9-11].

Telegram Messenger - це додаток для обміну повідомленнями, який працює через інтернет, та представлений у вигляді веб-додатку та застосунків для популярних платформ (Android, iOS, Windows Phone, Windows NT, MacOS і Linux.) .Цей додаток дозволяє користувачам безкоштовно відправляти текстові повідомлення через Wi-Fi або мобільний інтернет, при умові наявності достатнього тарифного плану.

Telegram зв'язується з номером телефону користувача через відправку перевірочного коду (автоматично зчитується в Android-додатках). Він дозволяє спілкуватися як із окремими особами, що вже встановили додаток, так і створювати групові чати на до 200 учасників із можливістю запрошення нових

учасників. Розробники вважають цю функцію однією з ключових особливостей програми.

Telegram набув популярності в Гонконзі під час протестів завдяки сильному шифруванню тексту і захисту від перехоплення, на відміну від Facebook, де інформація легше ставала доступною публічно.

Безпека є головною особливістю програм для передачі даних. Це означає, що всі взаємодії, включаючи переписку, групи та зашифровані медіа, залишаються конфіденційними і недоступними для сторонніх, навіть у разі перехоплення.

Чат-боти – це комп'ютерні програми, створені для імітації розмови між двома людьми, які стали популярними завдяки широкому використанню месенджерів та розуміння природної мови. Технологія чат-ботів існує з 1965 року і продовжує розвиватися. Чат-боти можна розробляти за допомогою різних технологій, включаючи технічні правилові штучні інтелекти, мову розмітки штучного інтелекту (AIML), Chat Fuel, Chat Scripts, архітектуру управління неструктурованою інформацією (UIMA), розуміння мови (LUIS) та Google dialog flows [12-14].

Чат-бот – це програма, яка моделює імітацію бесіди з користувачем через текстові або голосові інтерфейси. Ці боти можуть бути інтегровані в різні месенджери, веб-сайти або мобільні додатки і зазвичай використовуються для забезпечення автоматизованої взаємодії з користувачами, швидкого відповіді на запитання, надання інформації чи допомоги у вирішенні простих завдань.

Чат-боти, які є порівняно новітнім винаходом в технологічному світі, продовжують бути предметом дебатів щодо їх типології та різновидів. З популярних типів можна виділити наступні:

– боти з попередньо встановленим сценарієм або швидкою відповіддю: ці чат-боти функціонують на основі ієрархічної структури рішень, проводячи користувача через серію стандартних питань до отримання відповіді. Схожі на них програми з меню, де користувачі вибирають із фіксованого переліку опцій, щоб уточнити свій запит.

– боти на основі ключових слів: ці боти є більш розвинутими і працюють, аналізуючи введення користувача, намагаючись відповідати, виходячи з вибраних ключових слів. Хоча цей метод має свої недоліки, як наприклад, проблеми з повторюваними словами або зайвими запитаннями.

– гібридні боти: вони поєднують елементи меню та розпізнавання ключових слів. Це дозволяє користувачам отримувати прямі відповіді на запитання, а також користуватися меню бота для додаткових опцій у разі неефективності ключових слів.

– контекстуальні боти: це одні з найскладніших, оскільки вони використовують машинне навчання та ШІ для запам'ятовування розмов і взаємодій, щоб з часом вдосконалюватися. Вони не покладаються на ключові слова, а скоріше використовують інформацію, отриману від клієнтів, для поліпшення своїх відповідей та функцій.

– голосові боти: Вважаються майбутнім у світі чат-ботів, використовують голосові команди користувачів для відповідей або виконання завдань. Ці боти розроблені з використанням технологій перетворення тексту в мову та розпізнавання голосу, як-от Amazon Alexa та Siri від Apple.

З розвитком цих технологій спостерігається зменшення конфліктних ситуацій при взаємодії з клієнтами. У перехідний період від активного голосового спілкування до переважно текстових інтеракцій, чат-боти ефективно заповнюють прогалину, залишену традиційними телефонними дзвінками.

OpenAI – це некомерційна організація, яка прагне розвивати технологію ШІ для добра людства. OpenAI зосереджується на дослідженні ШІ та розробці безпеки, приватності, та справедливості, надаючи ресурси для допомоги іншим у створенні кращого ШІ. Одним з найпопулярніших продуктів є ChatGPT.

ChatGPT – це новітня технологія, що дозволяє користувачам взаємодіяти з текстовими машинами. Вона використовує алгоритм GPT-3 для розуміння контексту та генерації відповідних відповідей. ChatGPT може застосовуватися для різноманітних цілей, включаючи допомогу користувачам у пошуку інформації, відповідях на питання та виконанні завдань. Технологію також

можна використовувати для створення чат-ботів, які допомагають у різних ситуаціях. ChatGPT став однією з найпопулярніших технологій в Індонезії і був впроваджений різними компаніями для покращення обслуговування клієнтів, підвищення продуктивності та ефективності [15].

Telegram Bot – це спеціальний акаунт Telegram, який призначений для автоматичного оброблення повідомлень. Користувачі можуть взаємодіяти з ботами та відправляти командні повідомлення через приватні або групові чати. Для налаштування акаунтів ботів не потрібен додатковий телефонний номер [16, 17]. Цей акаунт служить лише як інтерфейс для коду, що виконується на сервері. Боти Telegram можуть бути створені відповідно до потреб користувачів та інтегровані з іншими сервісами для розумних будинків, соціальних служб, індивідуальних інструментів та інших віртуальних реалізацій [18–20].

Інтеграція ChatGPT у Telegram-бота є зручною, оскільки користувачам не потрібно відкривати сайт ChatGPT, а доступ до нього можна здійснити просто через Telegram. Таким чином, потенційні студенти чи широка громадськість можуть легко знаходити інформацію. Це стає можливим завдяки простоті доступу та використання технології ChatGPT. Отже, дослідження зосереджується на інтеграції ChatGPT у Telegram-бот. Основною задачею дослідження є пошук швидких і точних відповідей на ключові слова, щоб забезпечити швидку та релевантну реакцію.

1.2 Аналіз основних можливостей використання ChatGPT

ChatGPT, розроблений OpenAI, є передовою моделлю штучного інтелекту для обробки мови. Він є частиною серії GPT (Generative Pre-trШIned Transformer), що представляє собою значний крок у технології обробки природної мови (NLP). ChatGPT розроблено для генерації текстових відповідей, схожих на людські, що робить його відмінно підходящим для розмовних застосувань. Його архітектура базується на глибокому моделі навчання, який був навчений на різноманітних текстах з Інтернету [36, 38].

1.2.1 Розуміння та генерація мови

ChatGPT відмінно розуміє контекст і генерує зв'язні та контекстуально відповіді. Він здатен підтримувати контекст протягом серії повідомлень, забезпечуючи послідовність в розмові.

Перший крок у обробці мови — це токенізація, де вхідний текст розбивається на менші одиниці, які називаються токенами. Ці токени можуть бути словами, фразами або навіть частинами слів. Далі відбувається процес нормалізації — цей процес включає в себе перетворення всього тексту в стандартний формат, наприклад, переведення його в нижній регістр, щоб забезпечити однорідність. Також це може включати видалення пунктуації, спеціальних символів і виправлення помилок у написанні.

ChatGPT побудований на типі нейронної мережі, яка називається Трансформер, яка спеціально розроблена для обробки послідовних даних, таких як текст. Трансформер використовує механізми самоуваги для визначення важливості різних частин вхідного тексту. Кожен токен перетворюється у числову форму, відому як вкладення. Ці вкладення захоплюють семантичну та синтаксичну інформацію про токен. Модель використовує вивчені вкладення як стартову точку для обробки мови, що дозволяє їй розуміти контекст та значення. Механізм самоуваги дозволяє моделі розглядати всю послідовність вхідних даних одночасно, що допомагає їй розуміти контекст. Наприклад, слово "банк" може бути інтерпретоване по-різному в залежності від навколишніх слів. Архітектура Трансформера складається з кількох шарів, кожен з яких виконує серію операцій над вхідним текстом.Під час проходження даних через ці шари модель поетапно будує більш складне розуміння мови.

Моделі штучного інтелекту, такі як ChatGPT, використовують контекст, наданий в попередньому тексті, для вирішення амбігвітностей у мові, таких як омоніми або сарказм. Модель передбачає наступне слово в послідовності, розглядаючи ймовірність різних слів на основі наданого контексту. Це важливо для генерації послідовних та відповідних відповідей.

Також на основі розуміння та генераціїї природної мови ChatGPT можна використовувати для перекладу тексту. Спочатку модель визначає мову-джерело. Це може бути явно вказано користувачем або виведено моделлю з вхідного тексту. Використовуючи свою архітектуру Transformer, ChatGPT аналізує структуру речення, граматику та контекст мови-джерела. Це розуміння є важливим для точного перекладу, оскільки воно передбачає не лише перетворення слова в слово, а й розуміння ідіоматичних виразів, контексту та культурних відтінків.

Модель використовує свій досвід для відображення контексту та значення з мови-джерела на цільову мову. Це передбачає пошук еквівалентних виразів та структур на цільовій мові, які передають те саме значення, що і на мові-джерелі.

Модель формує перекладений текст на цільовій мові, забезпечуючи його точність відображення значення, тона та відтінків оригінального тексту.

Фактори, які впливають на якість перекладу:

– якість та обсяг навчальних даних моделі на обох мовах, як джерело, так і цільова, має значний вплив на точність перекладу$

– хоча ChatGPT є потужною мовною моделлю, важливо зауважити, що її основний інтерфейс не передбачений для перекладу. Тому, навіть якщо вона може здійснювати переклади, вони не завжди можуть відповідати якості спеціалізованої моделі для перекладу.

1.2.2 Адаптивне навчання

ChatGPT можна тонко налаштувати для конкретних завдань, таких як відповіді на питання або переклад, шляхом тренування на спеціалізованих наборах даних [36, 38].

Процес адаптивного навчання ChatGPT, або, загалом, моделей у серії GPT (Generative Pre-trШned Transformer), є складним механізмом, який базується на двох основних етапах: переднавчання та налаштування.

Розглянемо, як працює цей процес:

1. Переднавчання. Подача великої кількості даних — спочатку модель викладається величезному набору даних, що містить різноманітні джерела тексту. Цей набір даних включає книги, статті, веб-сайти та інші форми письмової мови, що надають широкий спектр людських знань та використання мови. Наглядній Індекс — модель вивчає, як передбачати наступне слово в реченні, дивлячись на слова, що передують йому. Цей процес, відомий як мовне моделювання, дозволяє ШІ засвоїти основну граматику, синтаксис та велику кількість інформації в різних галузях. Розвиток контекстуального розуміння — подібно, як модель обробляє ці дані, вона навчається не лише значенню окремих слів, але і тому, як значення змінюється в різних контекстах. Ця здатність є критичною для того, щоб модель могла генерувати послідовні та контекстно відповідні відповіді.

2. Налаштування. Конкретні дані та завдання — після переднавчання модель проходить процес налаштування. Під час цього етапу вона навчається на більш конкретному наборі даних, який настроюється під завдання, які вона буде виконувати. Це може включати в себе дані розмов, тексти з певної галузі або завдання з вирішення проблем. Моделі подаються конкретні вхідні дані і навчається бажаним виходам, що допомагає їй засвоїти нюанси конкретного застосування, чи то реакції в розмовах, технічних пояснень або творчого письма. Для застосувань, таких моделей ШІ як ChatGPT, процес налаштування спрямований на розуміння розмов та генерацію відповідей. Це включає в себе навчання моделі на наборах даних розмов, вчить її розуміти та генерувати відповіді, які не лише контекстно відповідні, але й зацікавлюють та відповідають на вхід користувача.

Завдяки адаптивності модель може бути підтримана під конкретні застосування або галузі, що підвищує її ефективність в спеціалізованих областях.

Загалом, процес адаптивного навчання ChatGPT включає складне поєднання великомасштабного збору даних, контекстного навчання та постійного вдосконалення. Цей процес дозволяє моделі не лише розуміти та генерувати тексти, схожі на людські, але і адаптуватися до конкретних потреб

користувачів та контексту розмови, що робить її дуже ефективною для широкого спектру застосувань в обробці природної мови.

ChatGPT може проаналізувати велику кількість текстової інформації та надати узагальнений звіт про зміст, зберігаючи контекст. Процес аналізу текстів на сторінних ресурсах, полягає у використанні інструменту "browser", який дозволяє відкривати та переглядати веб-сторінки. ШІ аналізує текст, вибираючи важливі фрагменти інформації, а потім цитує ці фрагменти для використання у відповідях. Цей процес допомагає забезпечити точність і актуальність інформації.

1.2.3 Розв'язання проблем на текстовій основі

Модель може допомогти у розв'язанні задач, викладених у текстовому вигляді, таких як математичні задачі, питання з програмування чи логічні головоломки (рисунок 1.1).

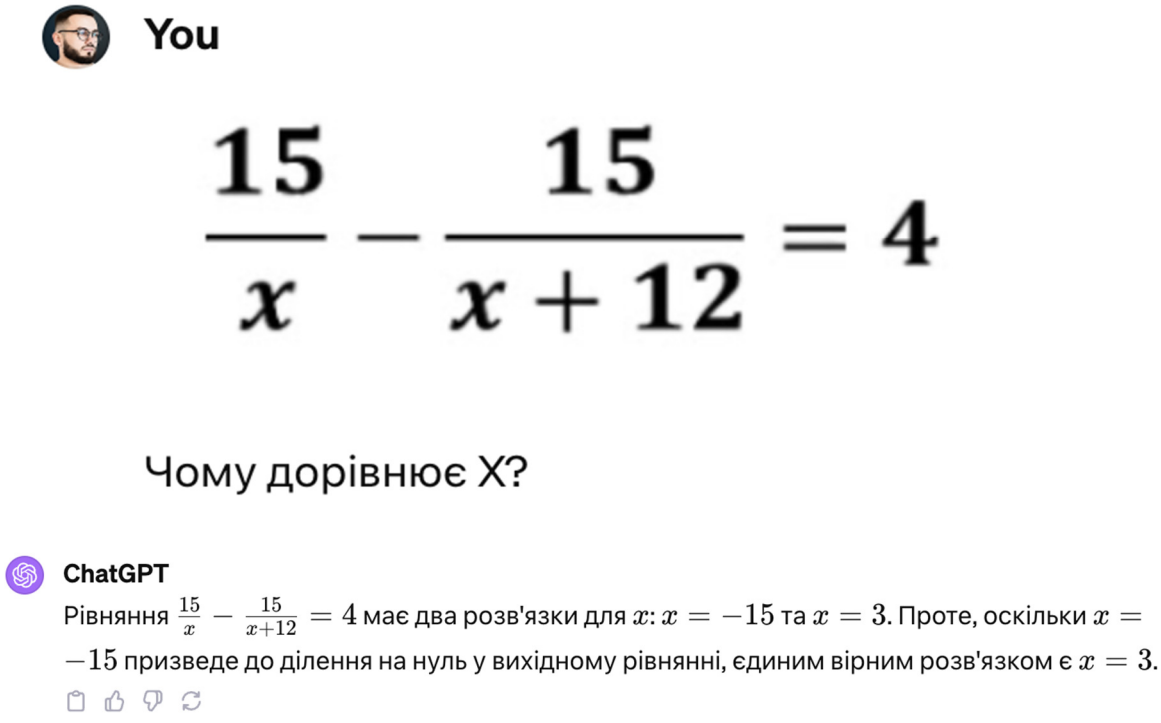

Рисунок 1.1 – Приклад розв'язання задач, викладених у текстовому вигляді

Останнє оновлення ChatGPT 4 отримало інтеграцію з іншою моделлю ШІ “DALL-E”. Що дає можливість чату GPT розрізняти текст із зображень. Це дозволяє пришвидшити взаємодію користувачі з інтерфейсом ШІ.

Після розпізнавання тексту на зображенні штучний інтелект застосовує відповідні математичні правила та алгоритми для вирішення рівняння. Це може включати алгебраїчну обробку, калькуль, або інші математичні методи, в залежності від характеру рівняння. З метою навчання або з метою ясності ChatGPT-4 часто може надавати розбивку на кроки в процесі вирішення задачі, пояснюючи, як він прийшов до відповіді.

1.2.4 Генерація коду на різних мовах програмування

Процес генерації відповідей за допомогою ChatGPT-4 відбувається у декілька етапів:

– початковий запит: коли ви надсилаєте запит, цей текст аналізується моделлю для зрозуміння основної ідеї та контексту запитання;

– обробка інформації: ChatGPT-4, тренована на величезній кількості текстових даних, використовує ці дані для створення відповіді. Модель має вбудовані знання про світ і мову, отримані під час тренування;

– генерація відповіді: модель використовує алгоритми машинного навчання, щоб створити відповідь, яка відповідає поставленому запиту. Вона створює відповідь шляхом передбачення наступних слів і фраз на основі контексту запиту;

– оптимізація відповіді: перед тим, як відповідь відправляється користувачу, модель перевіряє її на точність і релевантність, щоб переконатися, що вона максимально відповідає запитанню;

– відправлення відповіді: готова відповідь відправляється користувачу.

Генерація коду у ChatGPT обмежена даними, якими володіє модель. Цей спосіб взаємодії можна використовувати для аналізу коду, його оптимізації та пошуку проблем. Зважаючи на це можна зробити висновок, що ChatGPT не є

заміною професійним розробникам, а скоріше допоміжним інструментом, який може полегшити та покращити процес розробки програмного забезпечення.

1.3 Огляд основних можливостей управління чат-ботом у Telegram

Розглянемо основні можливості управління чат-ботом у Telegram [35]:

– відправка повідомлень: боти можуть відправляти текстові повідомлення, зображення, відео, аудіо, стікери та файли. Це базова функція, яка є основою для більшості взаємодій з користувачами.

– клавіатури та інлайн-кнопки: боти можуть використовувати спеціальні клавіатури для підтримки взаємодії з користувачем. Це включає звичайні клавіатури, які з'являються замість стандартної клавіатури смартфона, та інлайн-кнопки, які з'являються у самому повідомленні.

– реагування на команди: боти можуть бути налаштовані реагувати на певні команди, які починаються зі знака /, наприклад /start або /help.

– редагування та видалення повідомлень: боти можуть редагувати або видаляти повідомлення, які вони вже відправили.

– обробка відповідей (callback) від інлайн-кнопок: ця функція дозволяє ботам обробляти відповіді від користувачів на інлайн-кнопки, що додає інтерактивність.

– робота з групами і каналами: боти можуть бути додані в групи або канали, де вони можуть відправляти повідомлення, реагувати на повідомлення членів групи або виконувати адміністративні функції.

– інлайн-режим: ця функція дозволяє користувачам викликати бота в будь-якому чаті, вводячи його ім'я та запит. бот потім може відправити відповідь, яку можна вставити у чат.

– вебхуки: замість регулярного опитування telegram api, боти можуть використовувати вебхуки для отримання оновлень. це забезпечує більш ефективну та швидку взаємодію.

– обробка платежів: Telegram дозволяє ботам приймати платежі від користувачів безпосередньо в месенджері.

– кастомізація профілю бота: можна налаштувати ім'я, опис, фото та інші настройки бота через API.

– статистика та аналітика: для публічних ботів можливий доступ до статистики взаємодії користувачів з ботом.

1.4 Постановка задачі

Використання чат-ботів на основі штучного інтелекту, таких як ChatGPT, може значно покращити взаємодію між бізнесом та клієнтами, особливо у процесі первинної комунікації та консультацій.

Основні аспекти, які допоможуть покращити досвід комунікації користувачів з бізнесом:

– автоматичне відповіді на запити: чат-бот може миттєво відповідати на стандартні запити клієнтів. Наприклад, на питання про робочі години, ціни, послуги, характеристики продуктів чи політику повернення.

– кваліфікація потенційних клієнтів: чат-бот може збирати попередню інформацію від клієнтів, таку як їх потреби, бюджет, переваги, що допомагає визначити, наскільки вони підходять як потенційні покупці.

– надання інформації про продукт або послугу: бот може надавати детальну інформацію про продукти або послуги, що сприяє залученню клієнтів та підвищенню їхньої зацікавленості.

– перенаправлення до відділу продажів або підтримки: у випадку складних запитів чи потреби у персональній консультації, бот може перенаправити клієнта до відповідного відділу або спеціаліста.

Основна проблема для зручності управління Telegram-ботом полягає у тому, що Telegram немає нативної системи з візуальним інтерфейсом для управління ботом. Але за допомогою Telegram API можливо створити кастомну панель для управління ботом. Тому одна із цілей цього проєкту створити

Telegram-бот до якого інтегрований Chat GPT та розробити кастомну панель управління чат-ботом.

Основний фукнціонал, який буде підтримувати система адміністрування чат-ботом:

1. Автоматичне видалення користувачів з бази даних боту в разі неподовження платної підписки. Цей функціонал буде зручний для бізнесу в сфері освітніх та інформаційних проєктів. Наприклад: освітня компанія хоче відкрити доступ до своїх освітніх матеріалів, але не має бюджету для створення кастомної повноцінної LMS-платформи. Тому розробка такого функціоналу на базі Telegram-боту буде більш вигіднішою у фінансовому плані, а розробка займе менше часу.

2. Видалення користувачів. Адміністратор боту повинен мати можливість власноруч видаляти підписників з Telegram-боту за допомогою зручного візуального інтерфейсу.

3. Продовдження підписки. Адміністратор боту може самостійно вказати кількість днів, на протязі яких користувач буде мати доступ до бота.

4. Бан користувачів. Це зручний функціонал, коли користувачу треба обмежити доступ до матеріалів у боті, але при цьому не видаляти його з бази підписників. Це треба, для того, щоб адміністратор бота міг бачити контакт необхідної людини і написати в особисті повідомлення, щоб вирішити певні питання.

5. Аналітика. Основні показники повинні показуватись на окремій сторінці:

- всього користувачів — показує загальну кількість підписників;
- активних користувачів — користувачі, які регулярно взаємодіють з ботом;
- всього оплат — загальна сума оплат, яка була проведена за допомогою Telegram-бота;
- сторінка входу адміністратора — доступ до адмін-панелі.

Отже, мета роботи полягає в дослідженні та розробці методів інтеграції штучного інтелекту з Telegram-ботом. Це включає аналіз потенціалу ШІ для покращення функціональності бота, розробку і впровадження відповідних алгоритмів, а також оцінку ефективності взаємодії з користувачами. Робота зосереджується на розширенні можливостей бота, підвищенні якості обслуговування та забезпеченні більш глибокої персоналізації взаємодії.

Висновки до розділу 1

1. Штучний інтелект та його інтеграція у Telegram-боти відкривають нові можливості для автоматизації та ефективної комунікації. Використовуючи технології обробки природної мови та різні типи чат-ботів, від простих із фіксованими сценаріями до складних контекстуальних систем, можна створити інтелектуальні та гнучкі інтерфейси для комунікації. Створення Telegram-боту дозволяє вирішувати різноманітні задачі, зокрема у поліпшенні обслуговування клієнтів. Чат-боти можуть бути створені та налаштовані за індивідуальними потребами, інтегровані з широким спектром сервісів, що підкреслює їх універсальність та масштабованість.

2. Інтеграція ChatGPT у Telegram-боти відкриває нові можливості для поліпшення комунікації між бізнесом та клієнтами, особливо у первинній взаємодії та консультаціях. Автоматизація відповідей, кваліфікація потенційних клієнтів, надання інформації про продукти та послуги, а також здатність перенаправляти клієнтів до спеціалізованих відділів збільшують ефективність та задоволеність користувачів. Розробка спеціалізованої адміністративної панелі для управління Telegram-ботом дозволить краще контролювати його функціональність, включаючи видалення користувачів, управління підписками, банування користувачів та аналітику, забезпечуючи гнучке та ефективне управління ботом.

2 МЕТОДИ ТА ЗАСОБИ ІНТЕГРАЦІЇ ШТУЧНОГО ІНТЕЛЕКТУ ДО TELEGRAM-БОТУ

2.1 Концептуальні основи інтеграції штучного інтелекту до Telegram-боту

Концептуальні основи інтеграції ШІ до Telegram-боту включають:

1. Синергія між ШІ та чат-ботами: використання можливостей ШІ для покращення взаємодії бота з користувачами, забезпечення більш точних і релевантних відповідей.

Синергія між ШІ та чат-ботами полягає у використанні передових можливостей ШІ для значного покращення взаємодії між ботами та користувачами. ШІ дозволяє чат-ботам розуміти запити користувачів більш глибоко та контекстно, реагуючи на них більш точно і релевантно. Це означає, що боти можуть аналізувати великі обсяги даних, вчитися на зразках мови та взаємодіяти з користувачами майже як люди, здатні до природного мовлення і підтримки бесіди. Це підвищує задоволеність користувачів, оскільки вони отримують швидкі, точні та персоналізовані відповіді.

2. Автоматизація та персоналізація: боти з ШІ можуть автоматично відповідати на запитання, надавати рекомендації та адаптуватися до потреб користувача.

Автоматизація та персоналізація в контексті ШІ-ботів означають, що такі боти здатні автоматично реагувати на запитання та надавати індивідуальні рекомендації, адаптуючись до унікальних потреб і переваг кожного користувача. Використовуючи алгоритми машинного навчання і обробки природної мови, боти аналізують попередні запити та взаємодії, щоб зрозуміти контекст та наміри користувачів. Це дозволяє їм не тільки відповідати на запитання, але й пропонувати персоналізовані рішення, поради чи продукти, що значно покращує користувацький досвід.

3. Постійне навчання: ШІ може вчитися на основі взаємодії з користувачами, постійно покращуючи якість своїх відповідей.

Постійне навчання в ШІ чат-ботах полягає в їхній здатності адаптуватися та вдосконалюватися через взаємодію з користувачами. Використовуючи алгоритми машинного навчання, боти аналізують дані з попередніх розмов, вчаться розпізнавати шаблони та контекст, що дозволяє їм ставати більш точними та релевантними в своїх відповідях. Це неперервний процес, який не тільки покращує взаємодію з користувачами, але й допомагає боту адаптуватися до нових ситуацій та запитів, забезпечуючи кращий користувацький досвід.

4. Інтеграція API: використання API для підключення штучного інтелекту до Telegram-бота, що дозволяє легко інтегрувати різні сервіси ШІ.

Інтеграція API для з'єднання штучного інтелекту з Telegram-ботом дозволяє легко інтегрувати різні сервіси ШІ. Це включає в себе використання веб-API для забезпечення зв'язку між чат-ботом Telegram та платформами ШІ, такими як OpenAI, Google Cloud ШІ або IBM Watson. API дозволяє боту надсилати запити до ШІ та отримувати відповіді, трансформуючи ці дані у формат, зрозумілий для кінцевих користувачів. Це спрощує процес розробки, оскільки розробники можуть використовувати готові рішення ШІ, не створюючи їх з нуля.

Це створює основу для створення інтелектуальних, ефективних і корисних чат-ботів для Telegram.

2.2 Метод інтеграції штучного інтелекту до Telegram-боту

Метод інтеграції штучного інтелекту до Telegram-боту може бути описаний у вигляді наступних кроків:

1. Обрати платформу ШІ: Вибрати платформу штучного інтелекту (наприклад, OpenAI для використання GPT-3 чи GPT-4), яка найкраще відповідає потребам вашого бота.

2. Створення Telegram-бота: використати Telegram Bot API для створення нового бота, отримати унікальний токен.

3. Розробка інтерфейсу інтеграції: написати код, який забезпечуватиме взаємодію між Telegram-ботом та API обраної платформи ШІ.

4. Обробка запитів: реалізувати логіку бота для прийому та обробки повідомлень від користувачів, а також для відправлення цих повідомлень до ШІ для обробки.

5. Отримання та надсилання відповідей: отримувати відповіді від ШІ та надсилати їх користувачам.

6. Тестування та налагодження: провести тестування бота, переконатися, що інтеграція працює належним чином, і внести необхідні корективи.

7. Запуск та моніторинг: запустити бота для широкої аудиторії та здійснювати постійний моніторинг його роботи для виявлення та вирішення проблем.

Цей процес вимагає технічних знань у програмуванні та розуміння основ штучного інтелекту, а також обізнаності у використанні API Telegram та обраної платформи ШІ.

В основу методу інтеграції штучного інтелекту до Telegram-боту покладено водоспадну модель розробки програмного забезпечення [21-23]. Використання водоспадної моделі для інтеграції штучного інтелекту до Telegram-боту передбачає послідовний підхід, де кожен етап розробки чітко визначений та виконується один за одним. Це включає аналіз вимог, проектування системи, впровадження, верифікацію та обслуговування. Кожен етап базується на результаті попереднього, що забезпечує структурованість та організованість процесу розробки (рисунок 2.1).

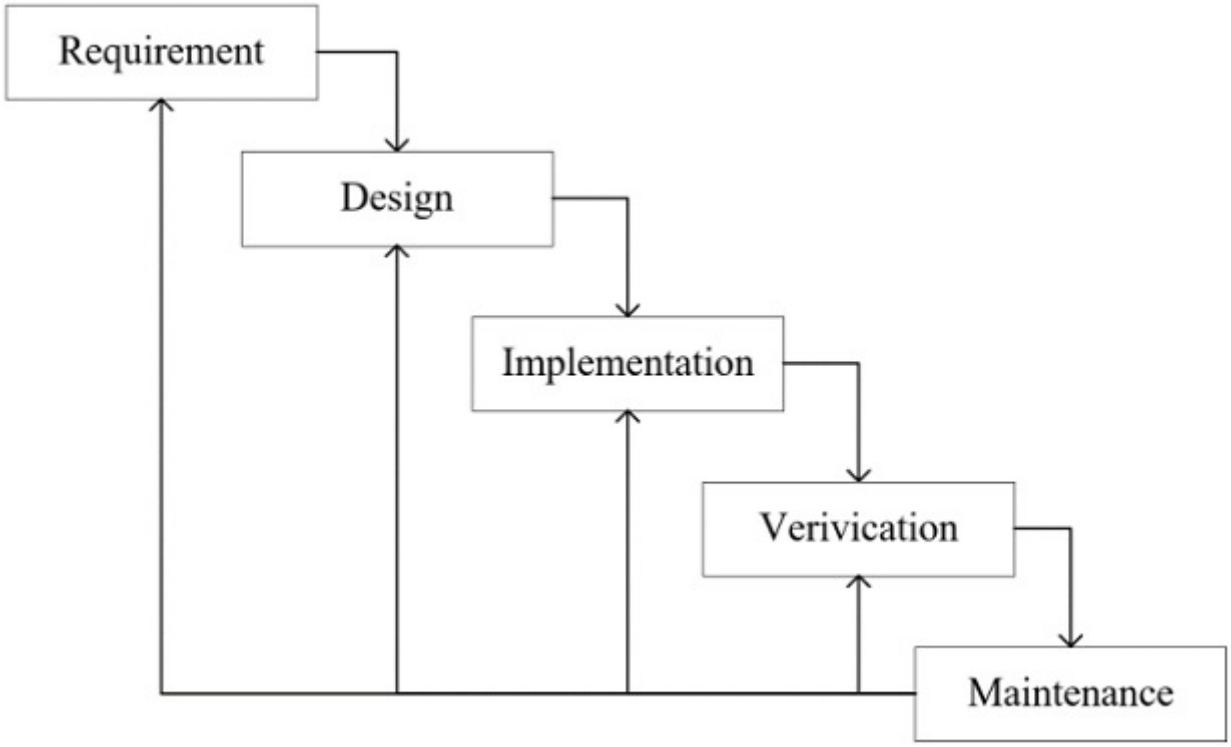

Рисунок 2.1 – Схема водоспадної моделі розробки ПЗ

Кожен етап розробки за водоспадною моделлю включає:

– аналіз вимог: визначення потреб користувачів та системних вимог. Це включає збір деталей про функціональність бота та очікування стосовно інтеграції ШІ;

– проектування системи: створення архітектури та дизайну системи на основі зібраних вимог. Це включає визначення, як ШІ буде інтегровано з Telegram-ботом;

– впровадження: розробка та кодування системи, включаючи інтеграцію ШІ та програмування бота;

– верифікація: тестування системи для перевірки відповідності специфікаціям та виявлення помилок;

– обслуговування: підтримка та оновлення системи після запуску для забезпечення її ефективної роботи.

Процес створення Telegram-бота представлено на рисунку 2.2.

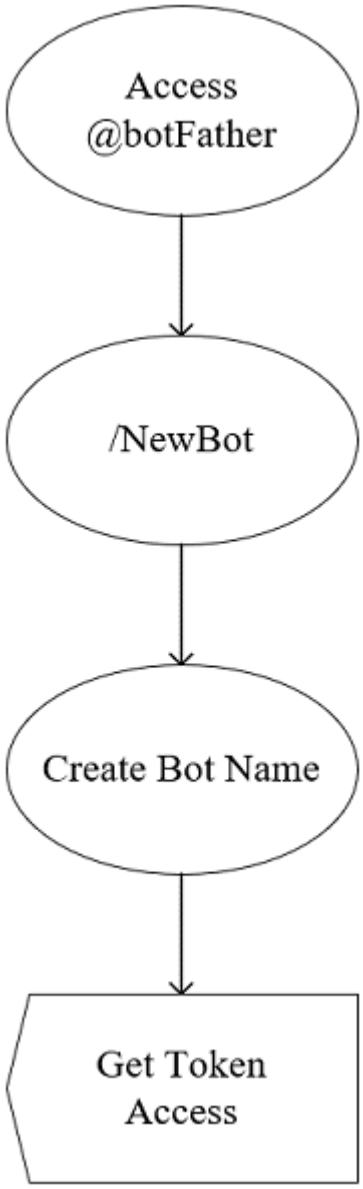

Рисунок 2.2 – Схема процесу створення Telegram-боту

Алгоритм роботи Telegram-бота представлено на рисунку 2.3.

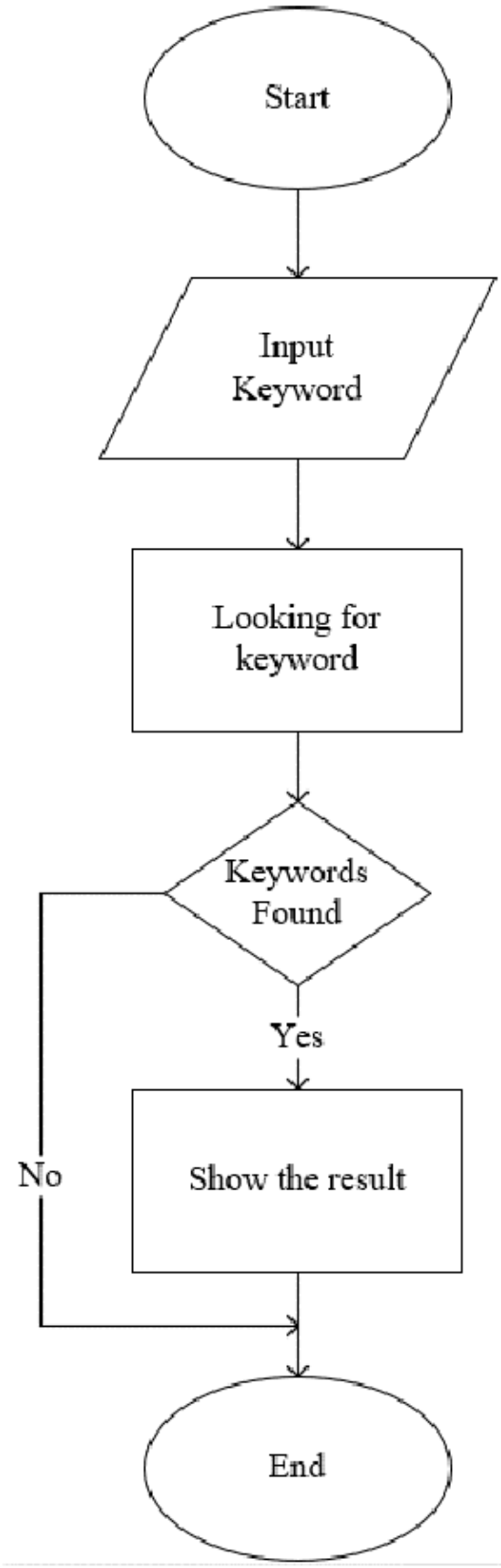

Рисунок 2.3 – Схема алгоритму роботи Telegram-боту

2.3 Засоби розробки та інтеграції штучного інтелекту до Telegram-боту

2.3.1 PHP

PHP є мовою скриптів, інтегрованою у HTML, яка переважно використовується для створення динамічних веб-додатків. Її синтаксис в значній мірі походить від мови C. Расмус Лердорф створив PHP у 1994 році для моніторингу відвідуваності своїх онлайн-резюме. Код PHP легко інтегрується в HTML, що полегшує створення динамічних частин веб-сторінок. Як скриптова мова, PHP потребує наявності відповідного інтерпретатора. Це технологія з відкритим вихідним кодом, яка сумісна з більшістю операційних систем і веб-серверів [39].

PHP здобула велику популярність в Інтернеті, будучи основою WordPress та використовуючись у Facebook. Ця мова грає ключову роль у веб-інфраструктурі, тісно пов'язана з HTML і CSS. Расмус Лердорф випустив першу офіційну публічну версію PHP у 1995 році, спочатку позиціонуючи її як набір інструментів для особистих домашніх сторінок. З появою JavaScript у 1996 році як клієнтської мови, PHP закріпилась у ролі серверної мови. Важливим етапом у її розвитку стало випуск версії 3, коли мову було перейменовано на Hypertext Preprocessor, що сприяло її широкій популярності.

PHP пропонує кілька ключових переваг для веб-розробки, які включають:

- Підвищення Швидкості Розробки: Завдяки багатству інструментів, методів та готових кодових фрагментів, що надаються PHP фреймворками, розробники можуть швидше реалізовувати складні бізнес-вимоги.
- Спрощення Підтримки Веб-Додатків: PHP фреймворки, що підтримують архітектуру MVC (Model-View-Controller), дозволяють розробникам ефективно організовувати та підтримувати веб-додатки.
- Ефективність Роботи з Базами Даних: PHP фреймворки забезпечують легку інтеграцію з різними реляційними базами даних, спрощуючи виконання операцій з базами даних без необхідності складного написання SQL-коду.

- Економічність Розробки: Використання відкритих PHP фреймворків може суттєво скоротити витрати на розробку, надаючи потужні функції та інструменти для ефективної розробки веб-додатків.

Робота PHP заснована на моделі "запит-відповідь". У реальності, цей процес включає більш складну послідовність дій, які можна ілюструвати діаграмою процесів (рисунок 2.4).

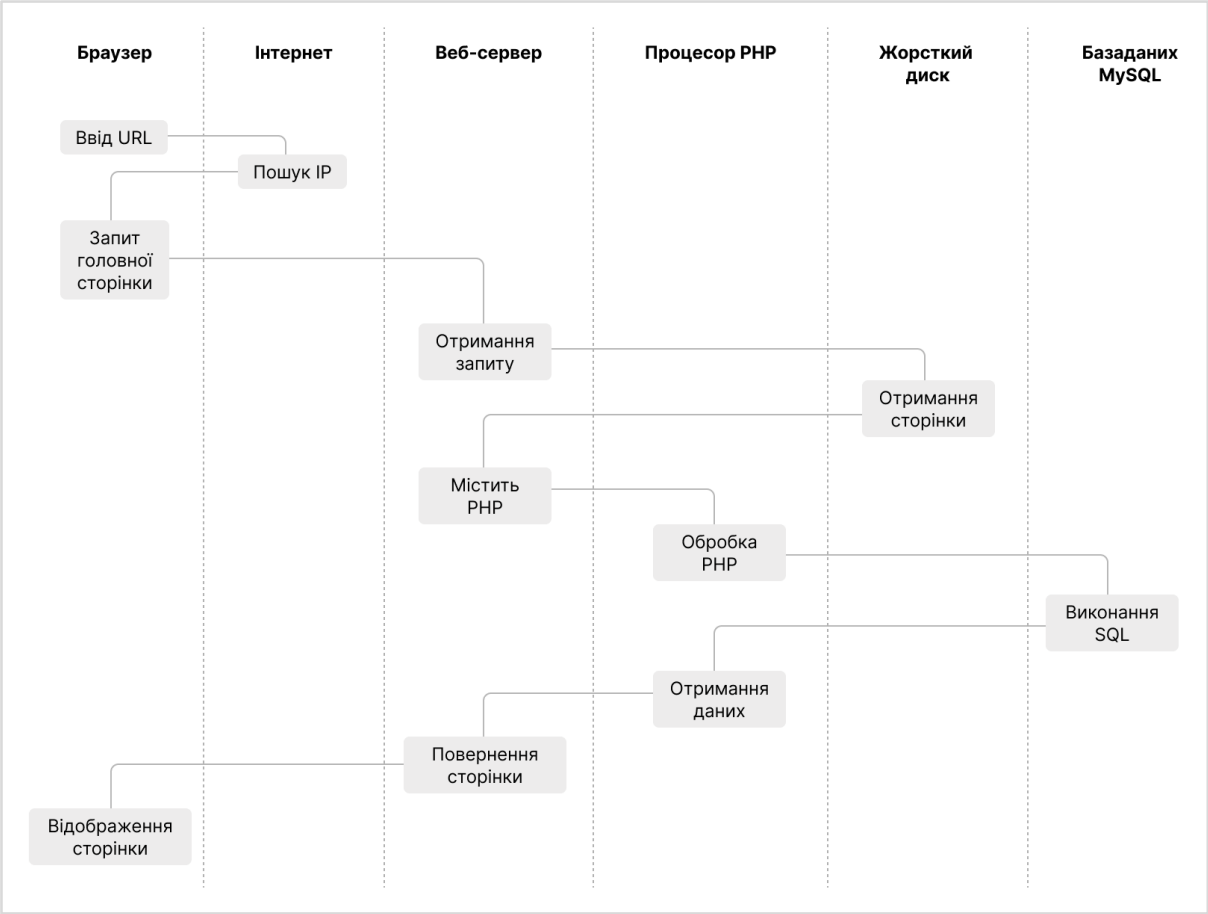

Рисунок 2.4 - Модель запиту із інтерпретацією PHP

2.3.2 Python

Python, створена в кінці 1980-х Гвідо ван Россумом, є широко використовуваною мовою програмування для різноманітних цілей. Вона відзначається динамічною типізацією, багатими структурами даних і різноманітністю вбудованих можливостей, що робить її ідеальною для складних проектів і легких скриптів одночасно. Python можна адаптувати для виконання системних завдань у більшості операційних систем і для запуску коду,

написаного на C++. Це універсальна мова, яка знайшла застосування в безлічі областей [34].

Python підтримує величезну спільноту розробників і має багатий набір відкритого коду. Мова була визнана найкращою за рейтингом IEEE Spectrum у 2018 році та найбільш затребуваною у 2019 році. Розробники можуть знайти рішення для багатьох проблем у великій спільноті користувачів. Існує безліч IDE та інструментів для розробки, а також тисячі доступних пакетів для розширення можливостей мови.

Основні переваги використання Python включають:

- легкість у сприйнятті та чистота синтаксису;
- відсутність необхідності використання дужок;
- автоматизоване управління пам'яттю;
- гнучкість динамічної типізації;
- розгорнута підтримка онлайн-спільноти;
- великий вибір доступних бібліотек.

Python характеризується динамічною типізацією, що означає визначення типу змінної під час її виконання. Серед основних типів даних, Python підтримує цілі числа без обмеження довжини та комплексні числа. Ця мова також має розширену бібліотеку для роботи з текстовими рядками, у тому числі в форматі Unicode.

Що стосується колекцій, Python пропонує кортежі (tuples), списки (масиви), словники (асоціативні масиви) і від версії 2.4, множини.

Класи в Python підтримують множинне успадкування та метапрограмування. Усі типи даних, включаючи базові, є частиною системи класів, з можливістю успадкування навіть від цих базових типів (рисунок 2.5).

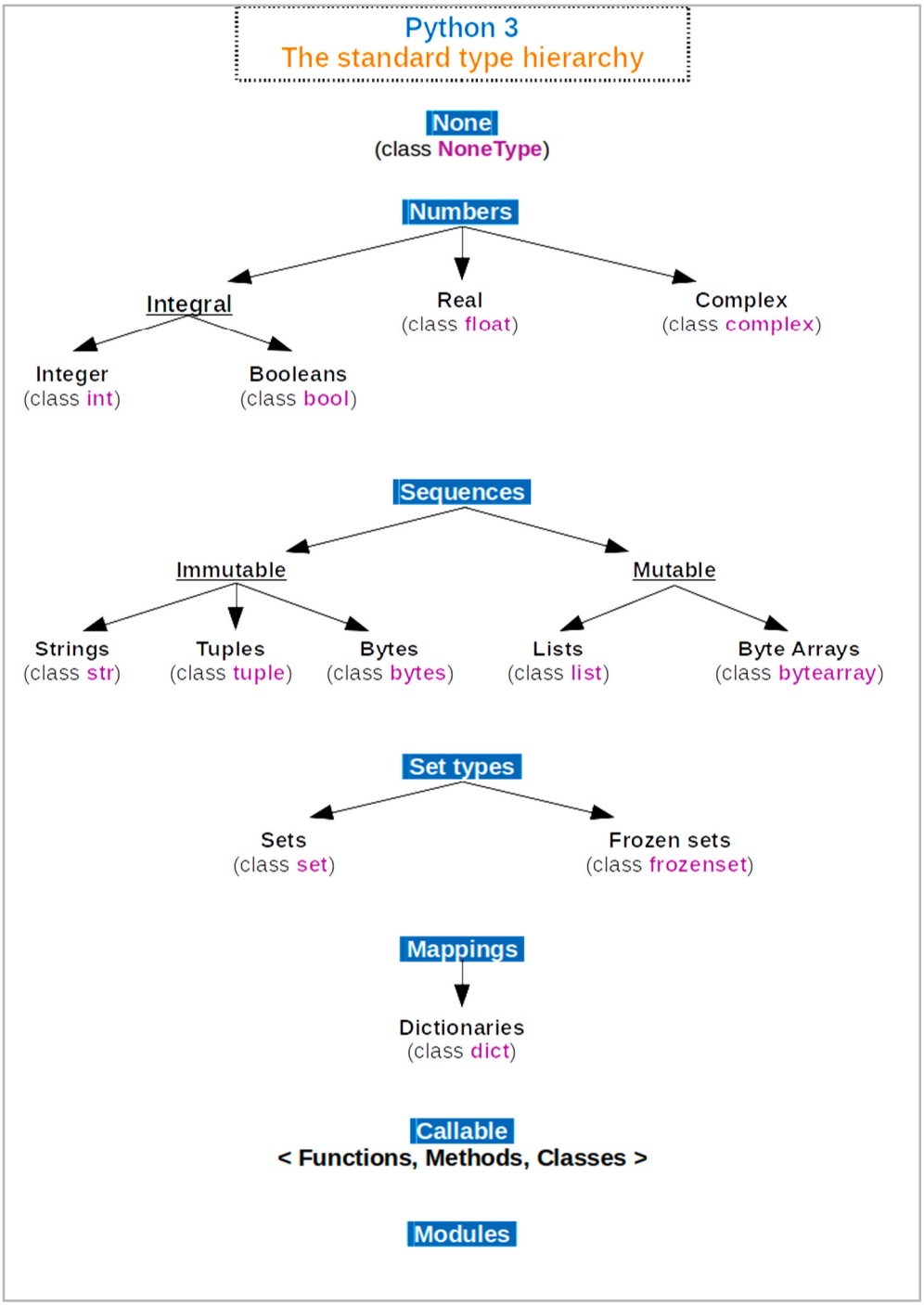

Рисунок 2.5 - Стандартна ієрархія типів Python

2.3.3 Node.js

Node.js — це платформа на стороні сервера, яка дозволяє використовувати JavaScript для створення швидких та масштабованих мережевих додатків. Вона побудована на основі движка V8 JavaScript, який також використовується в Google Chrome. Node.js використовує подієву, неблокуючу модель вводу-виводу, яка робить її легкою та ефективною, особливо для додатків, які працюють з даними в реальному часі через розподілені мережі [40-42].

Node.js змінив парадигму розробки веб-додатків, пропонуючи єдину мову програмування (JavaScript) для клієнтської та серверної частин. Ця універсальність сприяє швидкому розвитку додатків, зменшуючи витрати на навчання та контекстне перемикання для розробників.

Основна концепція та архітектура Node.js полягає у тому, що тут використовується подієво орієнтовану модель та неблокуючий ввід-вивід, що дозволяє оптимізувати пропускну спроможність та масштабування. Асинхронне програмування в Node.js здійснюється за допомогою зворотних викликів, промісів та асинхронних функцій.

Node.js підходить для створення масштабованих мережевих додатків, таких як веб-сервери, API, реалтаймові додатки (наприклад, чати) та IoT рішення. Він забезпечує високу продуктивність завдяки використанню подієвої моделі та ефективному управлінню пам'яттю.

Як вже було вказано, однією з ключових переваг Node.js є його неблокуючий, подієво-керований ввід-вивід, який забезпечує високу ефективність в реальному часі. Ця платформа спеціалізується на конкретних потребах, і зрозуміти це критично важливо. Не підходить Node.js для задач, що вимагають інтенсивних процесорних ресурсів, оскільки таке використання нівелює більшість його переваг. Однак, платформа ідеально підходить для створення швидких додатків завдяки здатності обробляти велику кількість одночасних з'єднань з високою пропускною здатністю в короткі проміжки часу, забезпечуючи тим самим високу масштабованість.

Node.js використовує однопотоковий підхід з неблокуючими вводами-виводами, що дозволяє управляти тисячами одночасних з'єднань в рамках одного подієвого циклу. Це відрізняє Node.js від інших мов програмування, які на кожен запит створюють новий потік, споживаючи при цьому значні обсяги оперативної пам'яті системи. Node.js демонструє здатність обробляти до 1.1 мільйона одночасних з'єднань і більше 500 тисяч одночасних з'єднань через веб-сокети, що робить його неперевершеним лідером у розробці швидких додатків.

В якості порівняльного аналізу можна розглянути Apache, який створює новий потік для кожного запиту, і Nginx, що працює за принципом неблокуючого вводу-виводу, аналогічно Node.js. Варто відзначити, що Apache зі збільшенням кількості запитів споживає більше пам'яті, тоді як Nginx використовує неблокуючий ввід-вивід (рисунок 2.6), обробляючи запити значно швидше і ефективніше.

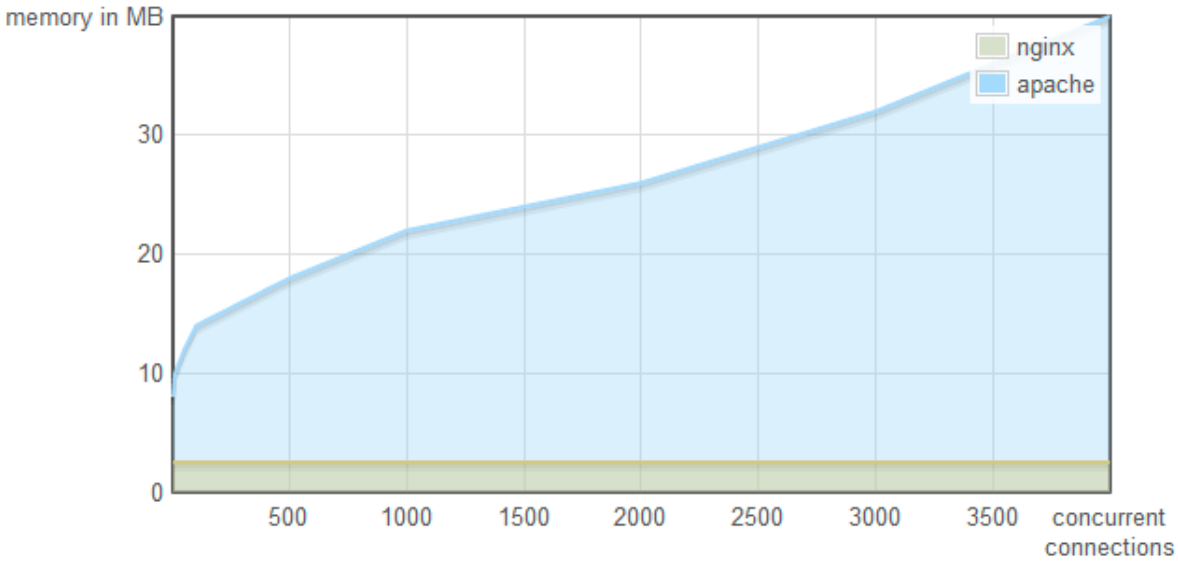

Рисунок 2.6 – Виділення пам’яті на кількість запитів Nginx та Apache

На прикладі (рисунок 2.7) видно, що Nginx здатний обробляти значно більшу кількість запитів за секунду порівняно з Apache, що є вагомою перевагою цієї платформи. Це також сприяє збільшенню потенційної кількості користувачів.

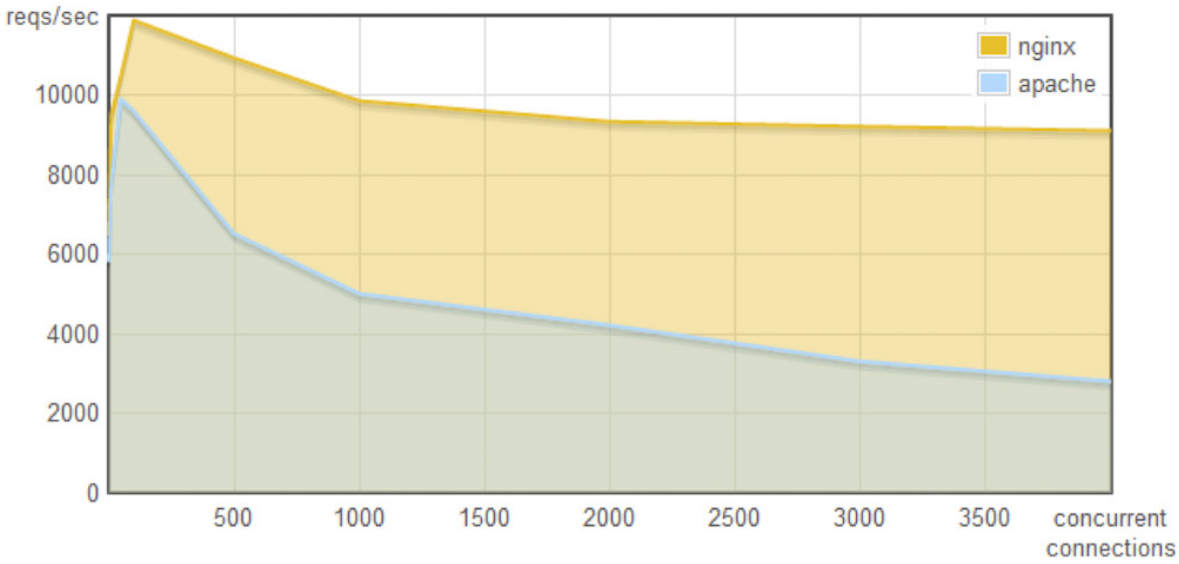

Рисунок 2.7 – Кількість оброблених запитів за секунду Nginx та Apache

Після аналізу трьох технологій – PHP, Python і Node.js – для розробки Telegram-бота, можна зробити висновок, що Node.js є найбільш оптимальним варіантом. Хоча PHP і Python мають свої переваги, Node.js вирізняється своїми унікальними можливостями, які роблять його ідеальним для створення чат-ботів. Node.js використовує JavaScript, що дозволяє розробникам використовувати одну мову для різних аспектів проекту, знижуючи складність розробки та час навчання. Його неблокуюча, подієво-керована архітектура забезпечує високу продуктивність, особливо в реалтаймових додатках, таких як чат-боти. Ця особливість дозволяє Node.js ефективно обробляти велику кількість одночасних запитів, що є ключовим для чат-ботів, які працюють з великою кількістю користувачів. Крім того, Node.js має широку підтримку спільноти та багатий набір інструментів і бібліотек, що полегшує інтеграцію з різними сервісами та платформами, включаючи Telegram. Це робить його надзвичайно гнучким у використанні та розширенні можливостей чат-ботів. Узагальнюючи, Node.js є найкращим вибором для розробки чат-ботів Telegram, завдяки його масштабованості, високій продуктивності, і легкості інтеграції з різними платформами та сервісами. Його використання дозволяє створити ефективний та

швидкий чат-бот, здатний виконувати складні завдання та обробляти велику кількість запитів одночасно.

Висновки до розділу 2

1. Інтеграція штучного інтелекту у Telegram-боти спирається на концепції синергії між ШІ та чат-ботами, автоматизації та персоналізації відповідей, постійного навчання від взаємодії з користувачами, та інтеграції API. Це дозволяє створювати боти, які розуміють і відповідають на запити користувачів більш точно та контекстно, надаючи персоналізовані рекомендації і покращуючи загальний досвід користувачів. Використання API спрощує процес інтеграції та розширює можливості бота, роблячи його ефективним інструментом для бізнесу та повсякденного використання.

2. Метод інтеграції штучного інтелекту в Telegram-боти включає послідовність кроків, заснованих на водоспадній моделі розробки ПЗ. Процес починається з вибору відповідної платформи ШІ, створення та налаштування Telegram-бота через API, розробки інтерфейсу для інтеграції ШІ та бота, та обробки та відправлення запитів до ШІ. Після тестування та налагодження, бот запускається та постійно моніториться. Цей підхід забезпечує структурований і організований розвиток, що дозволяє боту ефективно реагувати на потреби користувачів і адаптуватися до нових умов.

3. Аналіз PHP, Python та Node.js для розробки Telegram-бота показує, що Node.js є найбільш оптимальним варіантом. Використання JavaScript у Node.js знижує складність розробки, а його неблокуюча, подієво-керована архітектура забезпечує високу продуктивність у реалтаймових застосуваннях, таких як чат-боти. Node.js ефективно обробляє велику кількість одночасних запитів і має широку підтримку спільноти, що робить його гнучким для розширення можливостей чат-ботів. Вибір Node.js для чат-ботів Telegram забезпечує масштабованість, високу продуктивність та легкість інтеграції.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ЧАТ-БОТУ ТА СИСТЕМИ АДМІНІСТРУВАННЯ

3.1 Розробка чат-боту

Перший етап у розробці чат-боту – це розробка логіки взаємодії. Основний функціонал, який повинен підтримувати чат-бот: Інформування користувача про освітні програми та послуги, відповіді на питання за допомогою інтеграції ChatGPT, автоматичне видалення користувачів з прив’язаного чату та Telegram-каналу (рисунок 3.1).

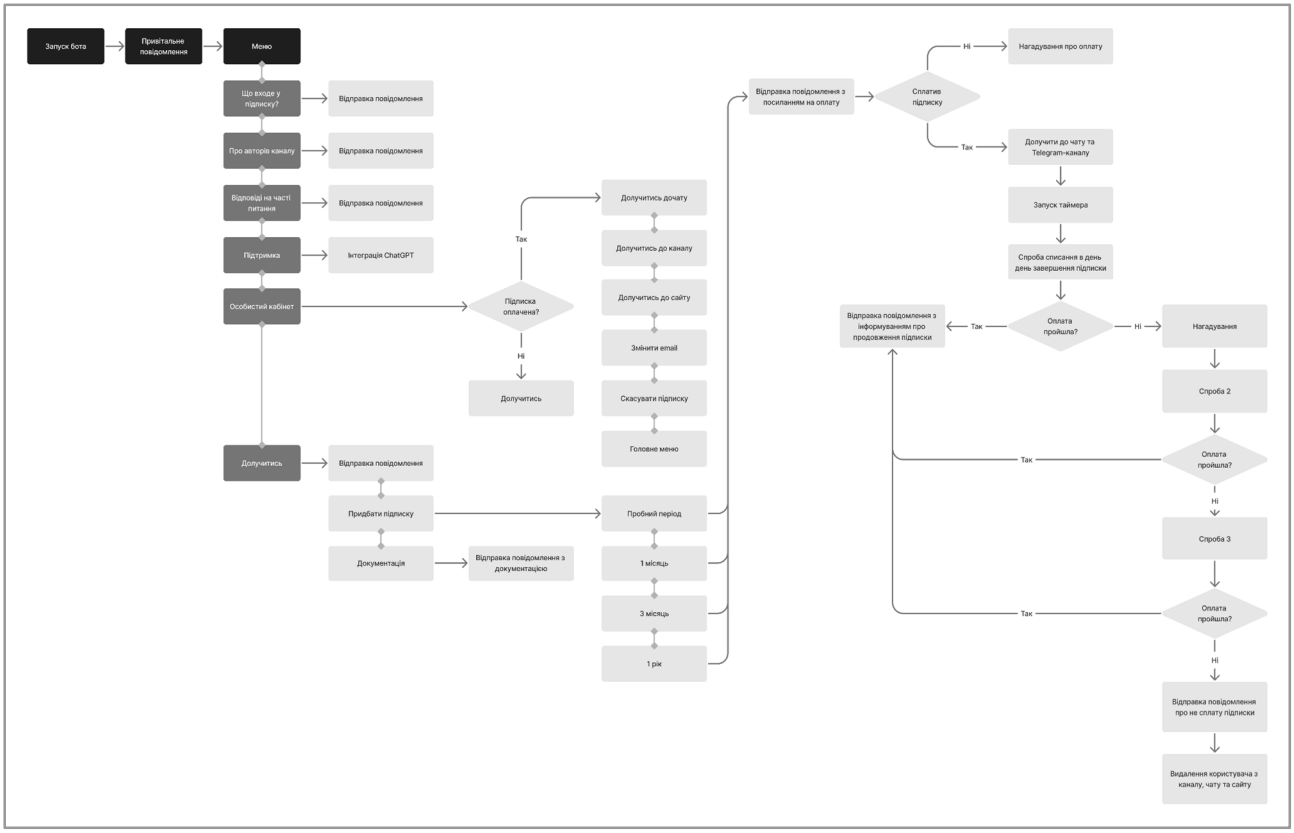

Рисунок 3.1 – Логіка чат-бота

Наступний крок це створення програми, яка автоматизує певні процеси, а також створення інтеграції ChatGPT з Telegram-ботом (рисунок 3.2).

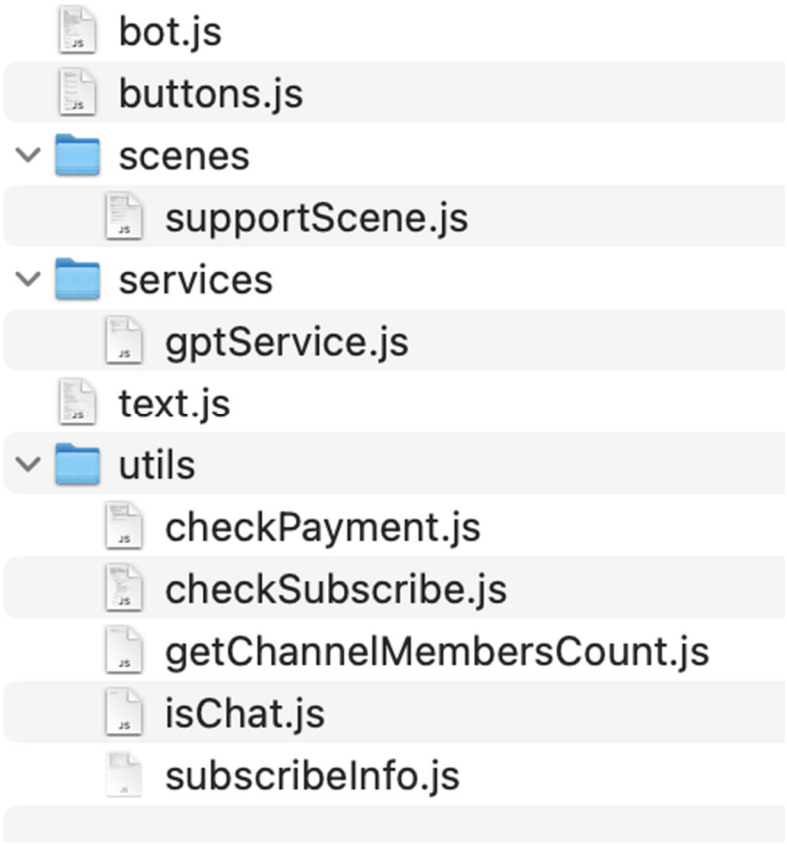

Рисунок 3.2 – Структура проєкту чат-бота

Файл “bot.js” створює Telegram бота з використанням Node.js та бібліотеки Telegraf. Основна мета бота — надавати інтерактивний інтерфейс для користувачів, дозволяючи їм взаємодіяти з різними функціями через команди та кнопки. Давайте розглянемо основу програми чат-боту. Імпорт бібліотек та модулів:

- const { Telegraf, Markup, Scenes, session } = require("telegraf"); - імпортує основні компоненти Telegraf.
- const { startMenuButtons, authorInfoButtons, ... } = require("./buttons"); - імпортує кнопки для інтерфейсу бота.
- const { User } = require("../database/models/models"); - імпортує модель користувача для роботи з базою даних.

Створення та конфігурація бота:

- const bot = new Telegraf(process.env.BOT_TOKEN); - ініціалізує новий екземпляр бота з токеном.

- bot.use(session()); - додає підтримку сесій.
- const stage = new Scenes.Stage([supportScene]); - створює нову сцену для діалогів.
- bot.use(stage.middleware()); - включає обробник сцени.

Команда /start:

- bot.start(async ctx => { ... }); - обробник команди /start.

Реєстрація користувачів:

- Усередині bot.start, код перевіряє наявність користувача в базі даних та реєструє нового при необхідності.

Обробка повідомлень:

- bot.hears("Головне меню", async ctx => { ... }); - обробник для команди "Головне меню".
- Аналогічні обробники для інших команд, наприклад, "Що я отримаю?", "Про авторів", "FAQ" тощо.

Управління підписками та оплатою:

- bot.hears("Придбати підписку", async ctx => { ... }); - обробник для придбання підписки.
- Функції generateValidOrderId і getPayment для генерації та отримання інформації про платіж.

Зміна налаштувань користувача:

- bot.hears("Змінити emШІ", async ctx => { ... }); - обробник для зміни електронної пошти користувача.

Підключення до чату та каналу:

- bot.hears("Долучитись до чату", async ctx => { ... }); - обробник для отримання посилання на чат.

Технічна підтримка:

- bot.hears("Технічна підтримка", async ctx => { ctx.scene.enter("supportScene"); }); - перехід у режим техпідтримки.

Використання фотографій та Markdown:

- Використання методу replyWithPhoto для відправлення фотографій.

- Використання Markdown у методі replyWithMarkdown.

Експорт бота:

- module.exports = { bot }; - експортує об'єкт бота.

Скрипт “buttons.js” реалізує набір функцій для створення кнопок у інтерфейсі користувача Telegram бота. Кожна функція повертає різні комплекти кнопок, що відображаються користувачам у різних сценаріях взаємодії з ботом.

Скрипт “supportScene.js” створює сцену для технічної підтримки у Telegram боті. Сцена використовується для обробки запитань користувачів, взаємодії з сервісом генерації відповідей (ChatGptService) та управління історією обміну повідомленнями.

Скрипт “gptService.js” реалізує клас ChatGptService у Node.js, який використовується для взаємодії з OpenШІ API, зокрема для генерації відповідей від моделі GPT-4. Цей клас ефективно використовує API OpenШІ для генерації відповідей на основі наданих повідомлень, що може бути корисно для реалізації функціоналу чат-ботів або інших програм, яким потрібно генерувати текстові відповіді. Ключові елементи, які використовуються для інтеграції ChatGPT з Telegram-ботом:

Імпорт бібліотеки axios:

- const axios = require("axios"); - імпорт бібліотеки axios, яка використовується для здійснення HTTP-запитів.

Оголошення класу ChatGptService:

- Клас ChatGptService призначений для здійснення запитів до OpenШІ API.

Конструктор класу:

- constructor() - визначає початкові налаштування класу. Встановлює URL API та API ключ з змінних середовища.

Метод generateResponse:

- generateResponse(messages) - метод приймає масив messages (повідомлень), який використовується як вхідні дані для GPT-4.

Запит до OpenШІ API:

- Налаштування заголовків запиту (headers) з вказівкою типу контенту та авторизаційного токена.
- Підготовка data, яка включає модель (model: "gpt-4"), масив повідомлень (messages) та параметр temperature.

Виконання HTTP-запиту:

- Використання axios.post для відправки запиту на API, включаючи URL, дані та заголовки.
- Обробка відповіді з API: витягування відповіді моделі з відповіді сервера (response.data.choices[0].message.content.trim()).
- Обробка помилок: у разі помилки виводиться в лог і повертається рядок "error".

Експорт класу ChatGptService:

- module.exports = { ChatGptService }; - це дозволяє іншим частинам програми імпортувати і використовувати ChatGptService.

Файл “text.js” містить набір різних текстових повідомлень. Ці повідомлення, використовуються в чат-боті для комунікації з користувачами. Кожен ключ у об'єкті texts представляє окреме повідомлення, яке може бути відправлено користувачам відповідно до логіки взаємодії користувача з чат-ботом.

Скрипт “checkPayment.js” визначає функцію checkPayment, яка призначена для перевірки статусу платежу в Telegram-боті. Вона використовує бібліотеку croner для запланованого виконання перевірки платежу, а також взаємодіє з кількома утилітами бази даних та платіжною системою.

Скрипт “checkSubscribe.js” визначає функцію checkSubscribe, яка призначена для автоматичної перевірки та управління підписками користувачів у Telegram-боті. Використовуються заплановані завдання (за допомогою бібліотеки croner) для перевірки статусу підписок та обробки повторюваних платежів.

Скрипт “getChannelMembersCount.js” визначає функцію getChannelMembersCount, яка призначена для отримання кількості учасників Telegram-каналу.

Скрипт “isChat.js” визначає функцію isChat, яка призначена для перевірки, чи повідомлення надійшло з чату, а не від індивідуального користувача у Telegram.

Скрипт “subscribeInfo” визначає функцію subscribeInfo, яка призначена для інформування користувача Telegram-бота про статус його підписки.

3.2 Розробка системи адміністрування Telegram-бота

GUI, що розшифровується як Graphical User Interface (графічний інтерфейс користувача), є формою інтерфейсу користувача, яка дозволяє користувачам взаємодіяти з електронними пристроями через графічні символи та візуальні індикатори, а не використовуючи командний рядок або чистий текст. Графічний інтерфейс користувача для системи адміністрування Telegram-бота передбачає наявність наступного функціоналу:

- Можливість входу/виходу в систему адміністрування.
- Збір статистичних даних користувачів з чат-боту Telegram.
- Список користувачів та певні дії з користувачами чат-боту (Додати кількість днів підписки, бан, видалення).

Першим етап у створення системи адміністрування чат-ботом — це розробка інформаційної архітектури (рисунок 3.3). Інформаційна архітектура — це практика організації контенту таким чином, щоб користувач міг легко переміщуватися по продукту (інтерфейсу) та інтуїтивно виконувати свої завдання. Також артефакт інформаційної архутектури у вигляді блок-схеми дозволяє швидше організувати та структурувати контент інтерфейсу, виявити недоліки у юзабіліті та усунити їх. Для створення системи адміністрування був обраний тип інформаційної архутектури “Flat” (рисунок 3.4). Цей тип

архітектури дозволяє максимально швидко отримати доступ до потрібної інформації

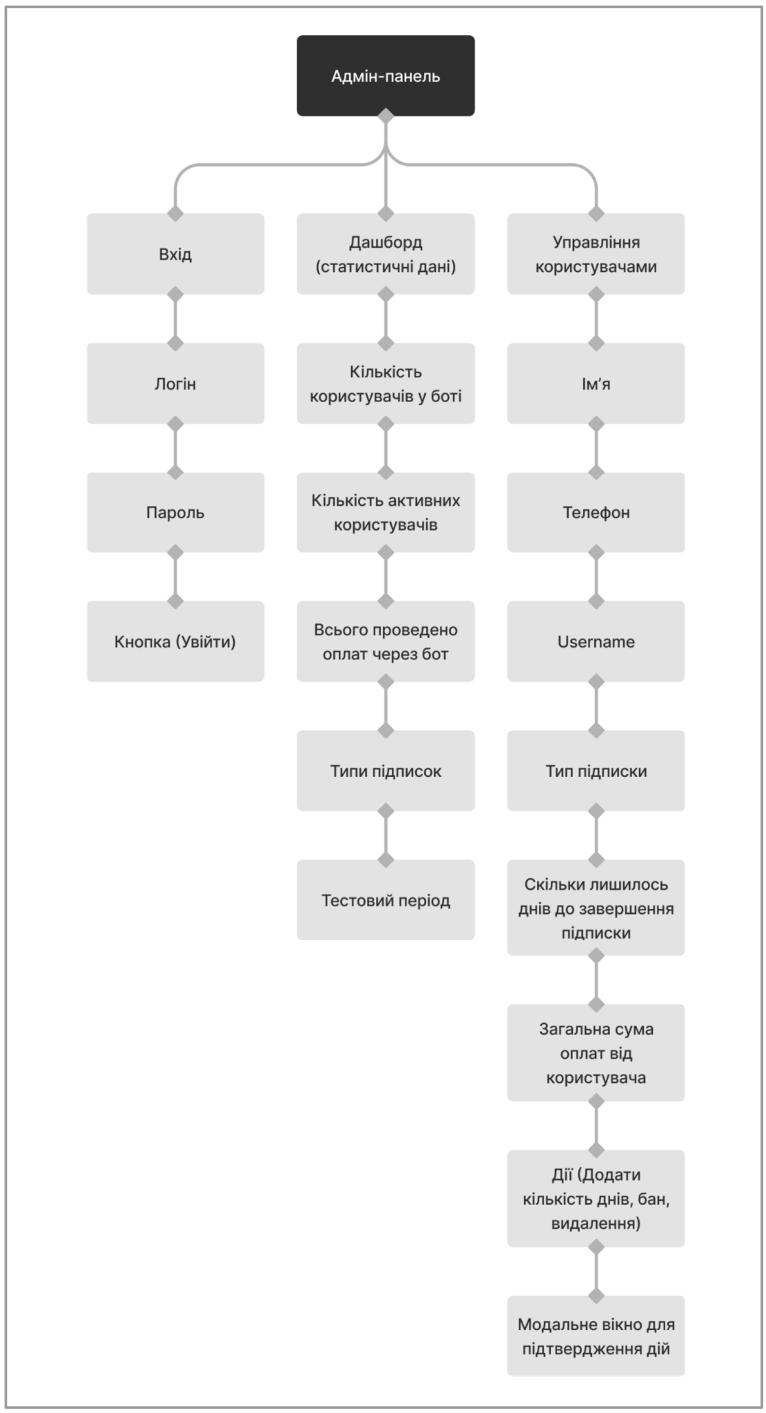

Рисунок 3.3 – Артефакт інформаційної архітектури системи адміністрування

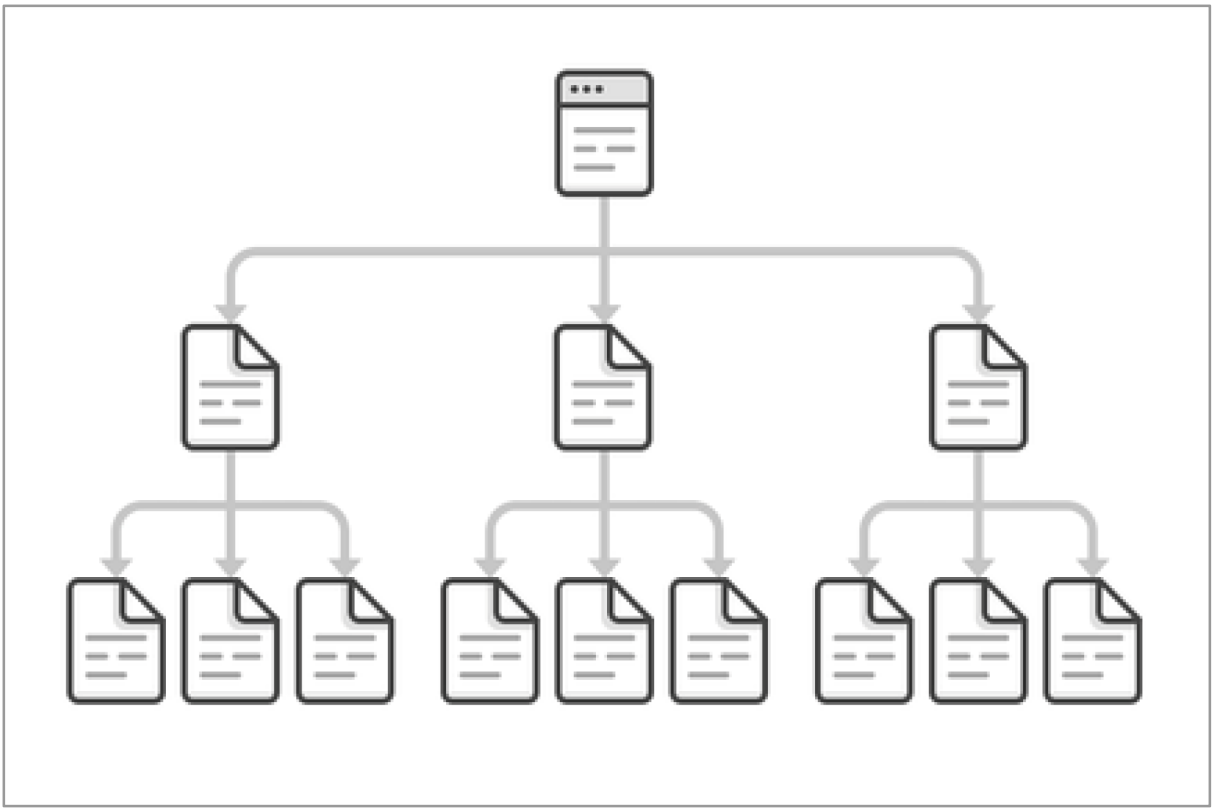

Рисунок 3.4 – Плоский тип інформаційної архітектури, “Flat”

Наступний крок при створенні системи управління — розробка дизайн-макету системи адміністрування. Дизайн-макет розроблено згідно з технічних вимог та артефакту інформаційної архітектури.

Сторінка входу складається з наступних функціональних елементів: поля для вводу даних (логін та пароль), кнопка авторизації (рисунок 3.5).

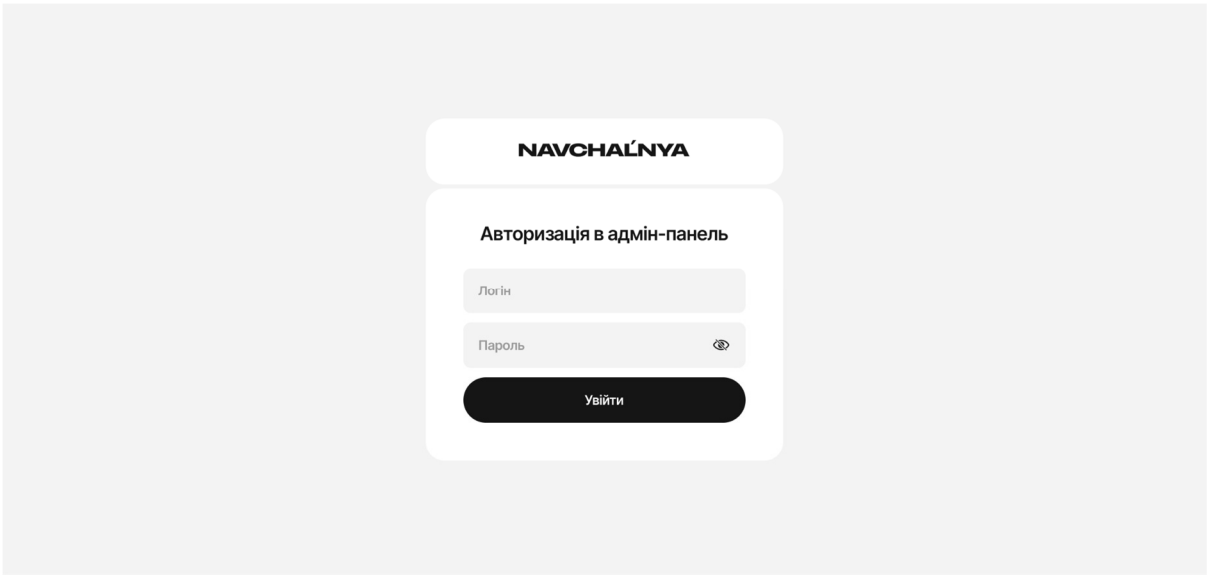

Рисунок 3.5 – Сторінка входу в панель адміністрування

Сторінка аналітики (рисунок 3.6) відображає статистичні дані, такі, як: кількість користувачів у боті, кількість активних користувачів, сума оплат, типи платних підписок на бота, користувачі з тестовим періодом.

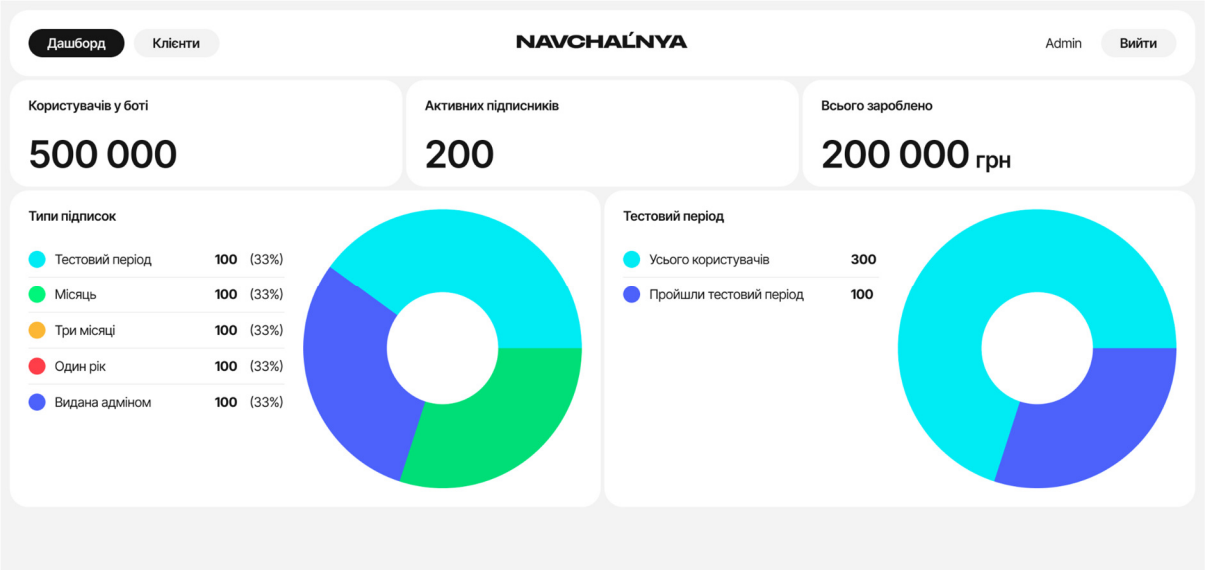

Рисунок 3.6 – Сторінка аналітики

Сторінка управління користувачами (рисунок 3.7) містить інфомацію про користувачів у вигляді списку.

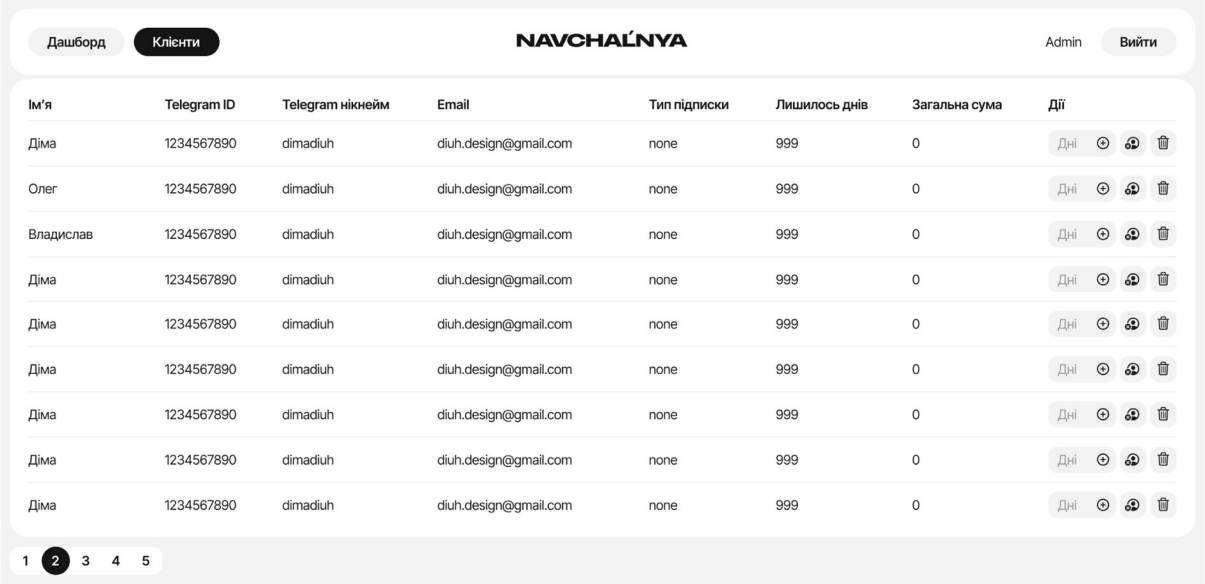

Рисунок 3.7 – Сторінка управління користувачами

Наступний крок у розробці системи адміністрування — це Front-end. Для цього етапу була обрана JS-бібліотека React. React, розроблена Facebook (нині Meta Platforms), є однією з найпопулярніших JavaScript бібліотек, що використовується для розробки інтерфейсів користувача, особливо для односторінкових веб-додатків (SPA). Зі своєю декларативною парадигмою та зосередженістю на компонентах, React забезпечує високу продуктивність, гнучкість та легкість у використанні, що робить її важливим інструментом для front-end розробників.

React має ряд переваг, які дозволяють розробникам швидко та ефективно виконувати свої задачі:

- Декларативність. React забезпечує декларативний стиль програмування, що робить код більш зрозумілим та легшим для налагодження. Розробники описують, якими мають бути компоненти UI, а React відповідає за ефективне їх відображення.
- Компоненти: React використовує компонентний підхід до розробки, де інтерфейс користувача будується з використанням взаємозамінних та повторно використовуваних компонентів. Це не лише сприяє кращій організації коду, але й полегшує тестування та підтримку коду.
- Гнучкість та інтеграція: React легко інтегрується з іншими бібліотеками та фреймворками, що дозволяє розробникам використовувати її у різноманітних проектах і сценаріях. React та Node.js часто використовуються разом у веб-розробці, але важливо розуміти, що вони виконують різні функції та використовуються для різних частин веб-додатку. Таке поєднання React та Node.js є досить популярним у сучасній веб-розробці, оскільки воно дозволяє створювати ефективні та масштабовані веб-додатки.

Структура проєкту (рисунок 3.8) складається з 3-х компонентів React: “Clients.js”, “Login.js” та “МШІn.js”

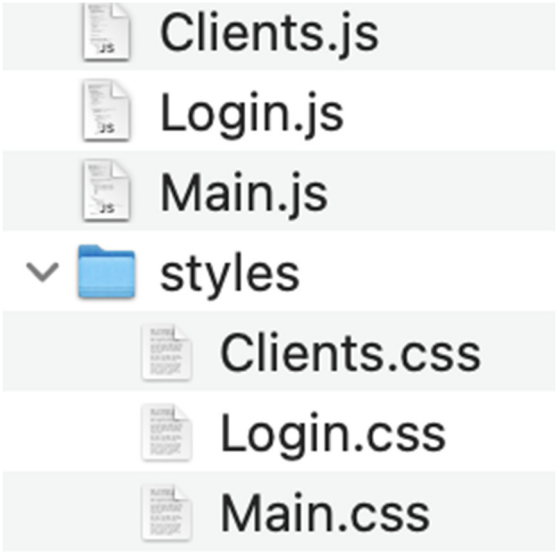

Рисунок 3.8 – Структура проєкту

“Clients.js” використовує реактивні хуки React для відображення та управління даними користувачів у таблиці, імпортованій з react-bootstrap. Він включає логіку для завантаження даних користувачів через API, а також функції для їх видалення, блокування та оновлення підписок. Компонент також реалізує сортування та пагінацію для кращої навігації по даних. Використовуючи observer з mobx-react-lite, він реактивно відстежує зміни стану. Для представлення інформації про кожного користувача використовується компонент UserRow.

“Login.js” використовується для авторизації адміністратора в адмін-панелі. Він включає форму входу з полем для логіну та пароля, де користувачі можуть взаємодіяти з інтерфейсом для введення своїх облікових даних. Використовуючи бібліотеку react-bootstrap для стилізації, компонент також надає можливість перегляду або приховування пароля. Компонент взаємодіє з сервером через функцію login з adminAPI для авторизації та використовує mobx-react-lite для реактивного управління станом. У разі успішної авторизації, відбувається перехід на головну маршрутну сторінку (MAIN_ROUTE) за допомогою хука useNavigate з react-router-dom. Якщо авторизація не вдається, користувачу виводиться відповідне повідомлення про помилку.

“MainIn.js” відображає статистику користувачів у додатку та інформацію про підписки за допомогою кругових діаграм. Він використовує хуки useState та

useEffect для зберігання та завантаження даних про користувачів і членів каналу через API. Компонент використовує mobx-react-lite для реактивного управління станом та react-chartjs-2 для візуалізації даних у вигляді діаграм. Дані фільтруються та агрегуються для відображення різних аспектів, таких як кількість активних підписників, загальна зароблена сума, типи підписок та інформація про тестовий період користувачів.

Також компоненти React стилізуються за допомогою відповідних таблиць стилів CSS.

Наступний крок — Back-end розробка. Структура серверної частини предствлена на рисунку 3.9.

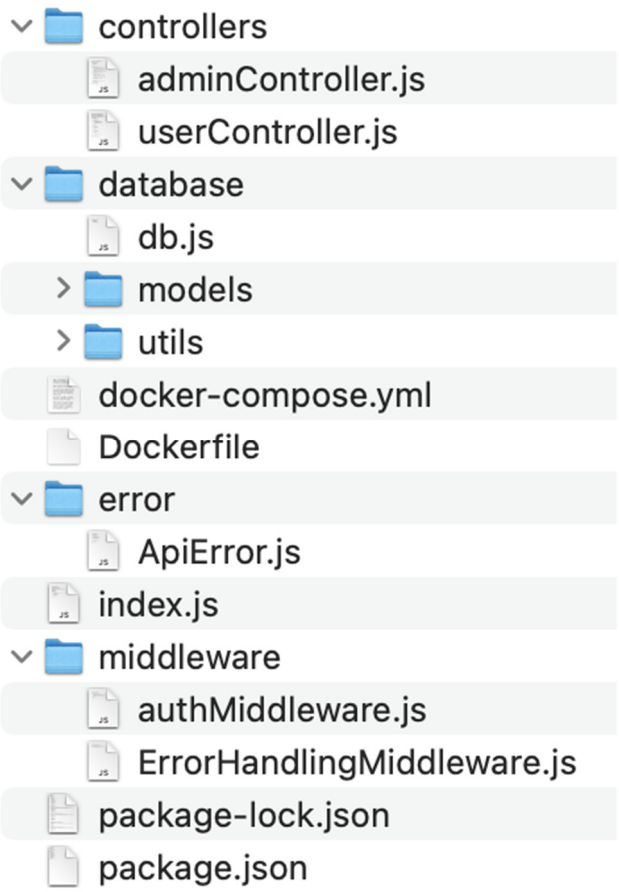

Рисунок 3.9 – Структура серверної частини панелі адміністрування

“AdminController.js” — клас контролера в Node.js, який управляє процесами реєстрації, входу та перевірки адміністраторів у веб-додатку. Використовуючи модель Admin з бази даних та бібліотеку bcrypt для хешування

паролів, контролер надає методи для реєстрації нових адміністраторів з перевіркою унікальності логіна та валідності секретного ключа. Також реалізовано функціонал входу, перевіряючи логін та пароль. У випадку успішної авторизації або реєстрації генерується JSON Web Token (JWT) з використанням jsonwebtoken. Додатково, клас містить методи для перевірки токена та отримання списку всіх адміністраторів. Залежність ApiError використовується для обробки помилок. Контролер експортується для подальшого використання в додатку.

“UserController.js”— контролер для управління користувачами в додатку. Він імпортує необхідні утиліти та модель User з бази даних. Клас UserController включає кілька асинхронних методів: getAll для отримання списку всіх користувачів, addSubscribeDays для додавання кількості днів до підписки користувача, deleteUser для видалення користувача, blockUser для блокування користувача, та getChannelMembers для отримання кількості учасників каналу. Методи взаємодіють з базою даних для здійснення операцій та повертають результати через HTTP відповіді. Код також обробляє можливі винятки та помилки, виводячи їх у консоль. Контролер експортується для використання в інших частинах додатку.

“db.js” — створює новий екземпляр Sequelize, який є ORM (Object-Relational Mapping) для Node.js, для підключення до бази даних PostgreSQL. Він імпортує Sequelize з відповідної бібліотеки та використовує змінні середовища для конфігурації з'єднання з базою даних, включаючи назву бази даних (DB_NAME), користувача (DB_USER), пароль (DB_PASSWORD), діалект (в цьому випадку 'postgres'), хост (DB_HOST) та порт (DB_PORT). Цей екземпляр Sequelize експортується для подальшого використання в додатку, дозволяючи здійснювати операції з базою даних через моделі та запити Sequelize. У цьому контексті, Sequelize використовується для створення з'єднання з базою даних. Конфігурація Sequelize включає інформацію про базу даних, таку як ім'я бази даних, користувач, пароль, тип бази даних (діалект), хост та порт. Це дозволяє додатку ефективно взаємодіяти з базою даних, виконуючи операції створення,

читання, оновлення та видалення даних (CRUD) у спосіб, який є більш зручним та інтуїтивно зрозумілим для JavaScript-розробників.

Тека “models” містить скрипт моделі для бази даних у застосунку Node.js, використовуючи Sequelize ORM. Модель User визначає структуру таблиці користувачів з полями, такими як ідентифікатор, ідентифікатор Telegram, ім'я, прізвище, нікнейм, електронна пошта, статус підписки, кількість залишених днів підписки, тип підписки, загальну суму, останню суму, останню транзакцію, статус пройденого тестового періоду, статус блокування, індекс оплати та повідомлення підтримки. Модель Admin визначає структуру таблиці адміністраторів з полями ідентифікатора, логіна та пароля. Обидві моделі використовують типи даних, надані Sequelize, та експортуються для використання в інших частинах додатку.

Тека “utils” містить список утиліт, які використовують модель User для виконання операцій, таких як оновлення, видалення або отримання інформації про користувачів. Вони здійснюють такі дії, як додавання днів до підписки, блокування або видалення користувачів, оновлення інформації про користувачів, включаючи статус підписки та тип підписки, а також отримання різних даних про користувачів, наприклад, їхній Telegram ID чи статус тестового періоду. Це забезпечує гнучке управління користувацькими даними в рамках додатку.

3.3 Тестування чат-боту та системи адміністрування

Останнім кроком у процесі втілення проекту є перевірка фінального продукту. Цей етап включає оцінку та аналіз роботи розробленого програмного забезпечення, щоб визначити, наскільки добре воно виконує задумані функції. Тестування дозволяє виявити та усунути потенційні помилки у роботі продукту.

У таблиці 3.1 представлено методику тестування завершеного продукту за допомогою серії тестових випадків.

Таблиця 3.1 - Test cases для Telegram-бота

Опис Test Case	Послідовність кроків	Очікуваний результат	Результат
Перевірка роботи технічної підтримки на основі ChatGPT	1. Запустити чат-бота 2. Натиснути кнопку “Головне меню” 3. Натиснути кнопка “Технічна підтримка” 4. Написати питання	Чат-бот з інтеграцією ChatGPT дасть відповідь на задане питання	Пройдено
Перевірка автоматичного додавання користувача до чату та телеграм-каналу після оплати підписки	1. Запустити чат-бота 2. Оплатити підписку або тестовий період 3. Перейти в особистий кабінет 4. Долучитись до чату та Telegram-каналу	Чат-бот автоматично додасть користувача до прив’язаного чату та Telegram-каналу	Пройдено
Перевірка зміни електронної пошти користувача	1. Запустити чат-бота 2. Перейти в особистий кабінет 3. Натиснути кнопка змінити email 4. Ввести новий email	Зміна електронної пошти користувача, яку можна перевірити в адмін-панелі	Пройдено

Так, як дані Telegram-бота інтегровані з системою адміністрування, її теж необхідно протестувати. У таблиці 3.2 представлено методику тестування завершеного продукту за допомогою серії тестових випадків.

Таблиця 3.2 - Test cases для системи адміністрування

Опис Test Case	Послідовність кроків	Очікуваний результат	Результат
Перевірка додавання днів підписки адміністратором	1. Увійти до системи адміністрування, як адміністратор 2. Перейти на сторінку “клієнти” та додати 1 день підписки будь-якому користувачу 3. Підтвердити дію у модальному вікні	Система нарахує вказану кількість днів підписки до Telegram-каналу та чату. Також ця інформація повинна відображатись у розділі “Особистий кабінет” у Telegram-боті	Пройдено
Перевірка блокування користувача	1. Увійти до системи адміністрування, як адміністратор 2. Перейти на сторінку “клієнти” та натиснути кнопку “Block User” 3. Підтвердити дію у модальному вікні	Бот перестане реагувати на будь-які дії користувача	Пройдено
Перевірка роботи збору аналітики	1. Увійти до системи адміністрування, як адміністратор 2. Перейти на сторінку “Дашборд” та звірити дані зі сторінкою “Клієнти”	При зміні даних, які відображає аналітика вони повинні змінюватись на сторінці “Дашборд”	Пройдено

Також було проведено тестування на швидкість завантаження системи за допомогою сервісу https://pagespeed.web.dev/. Результати тесту відображення на рисунку 3.10.

За результатами тестування можна побачити що загальна ефективність дорівнює 91, що являється відмінним показником.

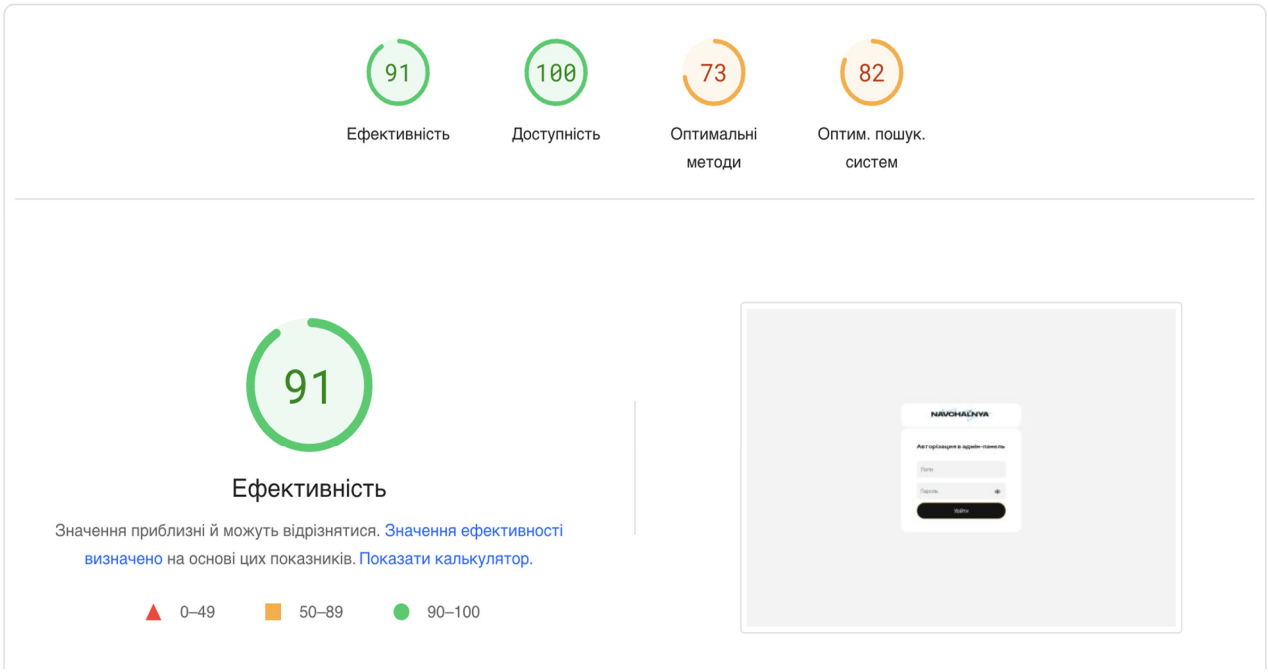

Рисунок 3.10 – Результати тестування системи адміністрування на швидкість завантаження

У третьому розділі детально описано процес розробки чат-бота, інтеграції з моделлю ChatGPT та створення системи адміністрування для Telegram-бота. Основні аспекти включають розробку логіки взаємодії, створення програми, налаштування інтеграції з ChatGPT, розробку інтерфейсу та обробки платежів та підписок.

Висновки до розділу 3

1. Розробка чат-бота базується на Node.js і бібліотеці Telegraf, забезпечуючи різні функції, такі як інформування користувачів, автоматичне видалення, обробка підписок та інші. Окрім того, була розроблена веб-панель адміністрування з використанням React для зручного керування користувачами та аналізу даних.

2. Система адміністрування включає функції для моніторингу та управління користувачами, збору аналітичних даних, обробки підписок і платежів. Важливою складовою є інтеграція з OpenШІ API для використання можливостей ChatGPT у відповідях чат-бота, що значно розширює функціонал і поліпшує взаємодію з користувачами.

3. Тестування чат-боту та системи адміністрування є ключовим кроком у перевірці функціональності та надійності фінального продукту. Воно включає оцінку роботи чат-бота за допомогою серії тестових випадків для перевірки різних функцій, таких як технічна підтримка, управління підписками та блокування користувачів. Аналогічно тестується система адміністрування для перевірки її ефективності, швидкості завантаження та аналітики. Це дозволяє виявити та виправити помилки, забезпечуючи гладке та ефективне використання чат-бота та адміністративної системи.

ВИСНОВКИ

У цій кваліфікаційній роботі роботі була розроблена та реалізована система чат-бота Telegram з інтеграцією моделі штучного інтелекту ChatGPT та створено систему адміністрування для керування цим ботом. Цей проект об'єднує передові технології в області штучного інтелекту та веб-розробки, демонструючи значний потенціал у сфері автоматизованої комунікації та взаємодії з користувачами.

Основні висновки:

1. Штучний інтелект та його інтеграція у Telegram-боти відкривають нові можливості для автоматизації та ефективної комунікації. Використовуючи технології обробки природної мови та різні типи чат-ботів, від простих із фіксованими сценаріями до складних контекстуальних систем, можна створити інтелектуальні та гнучкі інтерфейси для комунікації. Створення Telegram-боту дозволяє вирішувати різноманітні задачі, зокрема у поліпшенні обслуговування клієнтів. Чат-боти можуть бути створені та налаштовані за індивідуальними потребами, інтегровані з широким спектром сервісів, що підкреслює їх універсальність та масштабованість.

2. Інтеграція ChatGPT у Telegram-боти відкриває нові можливості для поліпшення комунікації між бізнесом та клієнтами, особливо у первинній взаємодії та консультаціях. Автоматизація відповідей, кваліфікація потенційних клієнтів, надання інформації про продукти та послуги, а також здатність перенаправляти клієнтів до спеціалізованих відділів збільшують ефективність та задоволеність користувачів. Розробка спеціалізованої адміністративної панелі для управління Telegram-ботом дозволить краще контролювати його функціональність, включаючи видалення користувачів, управління підписками, банування користувачів та аналітику, забезпечуючи гнучке та ефективне управління ботом.

3. Інтеграція штучного інтелекту у Telegram-боти спирається на концепції синергії між ШІ та чат-ботами, автоматизації та персоналізації

відповідей, постійного навчання від взаємодії з користувачами, та інтеграції API. Це дозволяє створювати боти, які розуміють і відповідають на запити користувачів більш точно та контекстно, надаючи персоналізовані рекомендації і покращуючи загальний досвід користувачів. Використання API спрощує процес інтеграції та розширює можливості бота, роблячи його ефективним інструментом для бізнесу та повсякденного використання.

4. Метод інтеграції штучного інтелекту в Telegram-боти включає послідовність кроків, заснованих на водоспадній моделі розробки ПЗ. Процес починається з вибору відповідної платформи ШІ, створення та налаштування Telegram-бота через API, розробки інтерфейсу для інтеграції ШІ та бота, та обробки та відправлення запитів до ШІ. Після тестування та налагодження, бот запускається та постійно моніториться. Цей підхід забезпечує структурований і організований розвиток, що дозволяє боту ефективно реагувати на потреби користувачів і адаптуватися до нових умов.

5. Основні етапи роботи включали глибокий аналіз можливостей ChatGPT, визначення структури та функціональності чат-бота, інтеграцію з Telegram, розробку бекенду та фронтенду для системи адміністрування. В ході роботи було використано ряд технологій, включаючи Node.js, React, Sequelize ORM та інші, що забезпечили гнучку та ефективну розробку.

6. В результаті проекту було створено повнофункціонального Telegram-бота, що відрізняється зручністю використання, високою адаптивністю та можливістю обробки складних запитів користувачів. Інтеграція з ChatGPT дозволила реалізувати продуману і гнучку систему відповідей, здатну обслуговувати широкий спектр запитів. Система адміністрування надала ефективні інструменти для управління контентом, користувачами, підписками та аналітикою.

7. Тестування чат-боту та системи адміністрування є ключовим кроком у перевірці функціональності та надійності фінального продукту. Воно включає оцінку роботи чат-бота за допомогою серії тестових випадків для перевірки різних функцій, таких як технічна підтримка, управління підписками та

блокування користувачів. Аналогічно тестується система адміністрування для перевірки її ефективності, швидкості завантаження та аналітики. Це дозволяє виявити та виправити помилки, забезпечуючи гладке та ефективне використання чат-бота та адміністративної системи.

8. Цей проект демонструє значний потенціал використання штучного інтелекту в розробці чат-ботів, відкриваючи нові перспективи для автоматизації комунікацій, підвищення ефективності обслуговування клієнтів та оптимізації робочих процесів. Реалізація такого проекту є свідченням інноваційного підходу до вирішення актуальних завдань у сфері цифрових технологій та штучного інтелекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. I. Fauzan, “Artificial Intelligence (AI) Pada Proses Pengawasan dan Pengendalian Kepegawaian - Sebuah Eksplorasi Konsep Setelah Masa Pandemi Berakhir,” J. Civ. Serv., vol. 14, no. 1, pp. 31–42, 2020.

2. Guntoro, L. Costaner, and Lisnawita, “Aplikasi Chatbot untuk Layanan Informasi dan Akademik Kampus Berbasis Artificial Intelligence Markup Language (AIML),” Digit. Zo. J. Teknol. Inf. dan Komun., vol. 11, no. 2, pp. 291–300, 2020, doi: 10.31849/digitalzone.v11i2.5049.

3. A. R. Taufani and H. A. Rosyid, “Sistem Tutorial Berbasis Kecerdasan Buatan Pada Proses Pengambilan Keputusan Perawatan dan Perbaikan Gitar,” J. RESTI (Rekayasa Sist. dan Teknol. Informasi), 2019, vol. 3, no. 1, pp. 79–86.

4. X. Liu, “Artificial intelligence and modern sports education technology,” Proc. - 2010 Int. Conf. Artif. Intell. Educ. ICAIE 2010, pp. 772–776, 2010, doi: 10.1109/ICAIE.2010.5641441.

5. X. Xu, H. Li, W. Xu, Z. Liu, L. Yao, and F. Dai, “Artificial intelligence for edge service optimization in Internet of Vehicles: A survey,” Tsinghua Sci. Technol., vol. 27, no. 2, pp. 270–287, 2022, doi: 10.26599/TST.2020.9010025.

6. F. Fourati and M.-S. Alouini, “Artificial intelligence for satellite communication: A review,” Intell. Converg. Networks, vol. 2, no. 3, pp. 213–243, 2021, doi: 10.23919/icn.2021.0015.

7. E. Nila and I. Afrianto, “Rancang Bangun Aplikasi Chatbot Informasi Objek Wisata Kota Bandung Dengan Pendekatan Natural Language Processing,” Komputa J. Ilm. Komput. dan Inform., vol. 4, no. 1, pp. 49–54, 2015, doi: 10.34010/komputa.v4i1.2410.

8. P. Dewonoto Laut Santoso, I. Riski, N. Kholik, M. Raffi Akbar, and A. Saifudin, “Penerapan Artificial Intelligence dalam Aplikasi Chatbot sebagai Media Informasi dan Pembelajaran mengenai Kebudayaan Bangsa,” J. Inform. Univ. Pamulang, 2021, vol. 6, no. 3, pp. 579–589.

9. Muliyono, “Identifikasi Chatbot dalam Meningkatkan Pelayanan Online Menggunakan Metode Natural Language Processing,” J. Inform. Ekon. Bisnis, vol. 3, pp. 142–147, 2021, doi: 10.37034/infeb.v3i4.102.

10. D. Rahayu, M. Mukrodin, and R. Hariyono, “Penerapan Artificial Intelligence Dalam Aplikasi Chatbot Sebagai Helpdesk Objek Wisata Dengan Permodelan Simple Reflex-Agent (Studi Kasus : Desa Karangbenda),” Smart Comp Jurnalnya Orang Pint. Komput., vol. 9, no. 1, pp. 7–21, 2020, doi: 10.30591/smartcomp.v9i1.1813.

11. C. A. Oktavia, “Implementasi Chatbot Menggunakan Dialogflow dan Messenger Untuk Layanan Customer Service Pada E-Commerce,” J I M P - J. Inform. Merdeka Pasuruan, vol. 4, no. 3, pp. 36–40, 2020, doi: 10.37438/jimp.v4i3.230.

12. Y. S. Wijaya, Rahmaddeni, and F. Zoromi, “Chatbot Designing Information Service for New Student Registration Based on AIML and Machine Learning,” JAIA-Journal Artif. Intell. Appl., vol. 1, no. 1, pp. 1–10, 2020.

13. M. U. Fahri, “Implementasi Channel Bot Telegram (Real Time) COVID-19 di Kalimantan Barat dengan Memanfaatkan API,” J. Ilmu Komput. dan Desain Komun. Vis., vol. 5, no. 2, pp. 77–84, 2020.

14. A. Suparno, “Chat Bot sebagai Implementasi Pemanfaatan Teknologi Artificial Intelligence dengan Channel Telegram,” J. Media Apl., vol. 12, no. 2, pp. 47–55, 2020.

15. A. Gilson et al., “How Does ChatGPT Perform on the United States Medical Licensing Examination? The Implications of Large Language Models for Medical Education and Knowledge Assessment,” JMIR Med. Educ., vol. 9, pp. 1–9, 2023, doi: 10.2196/45312.

16. M. I. Kurniawan, U. Sunarya, and R. Tulloh, “Internet of Things : Sistem Keamanan Rumah berbasis Raspberry Pi dan Telegram Messenger,” ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron., vol. 6, no. 1, p. 1, 2018, doi: 10.26760/elkomika.v6i1.1.

17. H. Supriyono, F. Suryawan, R. M. A. Bastomi, and U. Bimantoro, “Sistem Monitoring Suhu dan Gas Amonia untuk Kandang Ayam Skala Kecil,” ELKOMIKA

J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron., vol. 9, no. 3, p. 562, 2021, doi: 10.26760/elkomika.v9i3.562.

18. H. K. Astuti and N. D. Kuswytasari, "Efektifitas Pertumbuhan Jamur Tiram Putih (Pleurotus ostreatus) dengan Variasi Media Kayu Sengon (Paraserianthes falcataria) dan Sabut Kelapa (Cocos nucifera)," J. Sains dan Seni Pomits, vol. 2, no. 2, pp. 144–148, 2013.

19. A. D. Mulyanto, "Pemanfaatan Bot Telegram Untuk Media Informasi Penelitian," J. Ilmu Komput. dan Teknol. Inf., vol. 12, no. 1, p. 49, 2020, doi: 10.18860/mat.v12i1.8847.

20. N. Triwahyuni and S. Beta, "Running Text Information System Design Internet-Based for Small Outlets," J. Appl. Inf. Commun. Technol., vol. 7, no. 2, p. 2022, 2022.

21. H. A. Bukhori, B. Rahayudi, and W. H. N. Putra, "Optimasi Business Process Improvement Berbantuan Metode FLASH dengan Integrasi API Trello," J. RESTI (Rekayasa Sist. dan Teknol. Informasi), vol. 1, no. 1, pp. 19–25, 2017.

22. D. Nataliana, I. Syamsu, and G. Giantara, "Sistem Monitoring Parkir Mobil menggunakan Sensor Infrared berbasis RASPBERRY PI," ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron., vol. 2, no. 1, p. 68, 2014, doi: 10.26760/elkomika.v2i1.68.

23. R. A. Khan, S. U. Khan, H. U. Khan, and M. Ilyas, "Systematic Literature Review on Security Risks and its Practices in Secure Software Development," IEEE Access, vol. 10, pp. 5456–5481, 2022, doi: 10.1109/ACCESS.2022.3140181.

24. D. Saptono, T. M. Sampurna, and T. W. R. N, "Implementasi Algoritma Gunning Fog Index Pada Uji Keterbacaan (Readability Test) Bahasa Indonesia Menggunakan Bahasa Pemrograman Python," vol. 2013, no. November, 2013.

25. Martanto, R. D. Wihadi, R. D. Agusulistyo, and Tjendro, "Penampil Gelombang Tegangan dan Arus Berbasis Arduino Due untuk Generator AC Tiga Fasa," ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron., vol. 8, no. 2, p. 336, 2020, doi: 10.26760/elkomika.v8i2.336.

26. S. A. Sena, A. Muttaqin, and A. Setyawan, “Perancangan dan Pembuatan Application Interface Server untuk Arduino,” J. Tek. Elektro, Fak. Tek. Univ. Brawijaya, vol. 1, no. 4, pp. 1–6, 2013.

27. University of Liverpool. (2021). Чому штучний інтелект так важливий у сьогоднішн ому світі? https://online.liverpool.ac.uk/

28. Harvard John A. Paulson School of Engineering and Applied Sciences. (б.д.). Сучасність і майбутнє ШІ. https://seas.harvard.edu/

29. Nextechar. Важливість штучного інтелекту в сучасному світі. https://www.nextechar.com/

30. Brookings Institution. Як штучний інтелект трансформує світ. https://www.brookings.edu/

31. World Economic Forum. (2021). 5 способів, які ШІ добре виконує у світі прямо зараз. https://www.weforum.org/

32. Vaswani, A., et al. "Attention Is All You Need." Advances in Neural Information Processing Systems, 2017.

33. Brown, T.B., et al. "Language Models are Few-Shot Learners." OpenAI Blog, 2020.

34. Python Software Foundation. Python Documentation.

35. Telegram. Telegram Bot API Documentation.

36. OpenAI. OpenAI API Documentation.

37. Flask Documentation. Flask Web Framework.

38. Radford, A., et al. "Improving Language Understanding by Generative Pre-Training." OpenAI Blog, 2018.

39. A Quick History of PHP. https://medium.com/quick-code/the-history-of-php-ffb920ba4555

40. Why the Hell Would I Use Node.js? A Case-by-case Tutorial https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

41. Node.js Foundation. (2023). Node.js Documentation. https://nodejs.org.

42. Кошова, О. П., Ольховська, О. В., Тацій, Д. С., Олексійчук, Ю. Ф., & Черненко, О. О. (2023). Розробка веб-додатків та сервісів на платформі

Node.JS. Таврійський науковий вісник. Серія: Технічні науки, (2), 78-89. https://doi.org/10.32782/tnv-tech.2023.2.9.

43. "React документація" - React Official Documentation

44. "Основи React" - Ден Абрамов і Ендрю Кларк, книга "Learning React: A Hands-On Guide to Building Web Applications Using React and Redux".

45. Дюг Д.А. Інтеграція Chatgpt до Telegram-боту. Збірник праць VIII науково-практичної конференції вчених і студентів “Інтелектуальні комп’ютерні системи та мережі”, С. 80. https://ki.wunu.edu.ua/conference/archive/2023_2.pdf.

46. Дюг Д.А. Метод інтеграції Chatgpt до Telegram-бота. Матеріали XI науково-технічної конференції «Інформаційні моделі, системи та технології», Тернопіль, Тернопільський національний технічний університет імені І. Пулюя, 13-14 грудня 2023 р. С. 44.

47. Комар М.П., Саченко А.О., Васильків Н.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за другим (магістерським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2021. 32 с.

Додаток А

Лістинг програмного забезпечення

Програмний код створення Telegram-боту:

```
const { Telegraf, Markup, Scenes, session } = require("telegraf");
const {
  startMenuButtons,
  authorInfoButtons,
  joinInfoButtons,
  accountButtons,
  subscribeButtons,
  onlyMenuButtons,
  aboutChannelButtons,
  cancelSubscribeButtons,
  faqButtons,
} = require("./buttons");
const { User } = require("../database/models/models");
const { createUser } = require("../database/utils/createUser");
const { generateValidOrderId } = require("../payments/utils/generateValidOrderId");
const { getPayment } = require("../payments/utils/getPayment");
const { checkPayment } = require("./utils/checkPayment");
const { isChat } = require("./utils/isChat");
const { subscribeInfo } = require("./utils/subscribeInfo");
const { getSubscribe } = require("../database/utils/getSubscribe");
const { texts } = require("./text");
const { getTestPeriod } = require("../database/utils/getTestPeriod");
const { getEmail } = require("../database/utils/getEmail");
const { setEmail } = require("../database/utils/setEmail");
//const { getWebflowId } = require("../database/utils/getWebflowId");
//const { createNewUser } = require("../webflow/utils/createNewUser");
//const { setWebflowId } = require("../database/utils/setWebflowId");
const { setLastTransaction } = require("../database/utils/setLastTransaction");
const { getIsBlocked } = require("../database/utils/getIsBlocked");
const fs = require("fs");
const bot = new Telegraf(process.env.BOT_TOKEN);
const { supportScene } = require("./scenes/supportScene");

bot.use(session());
const stage = new Scenes.Stage([supportScene]);
bot.use(stage.middleware());

bot.start(async ctx => {
  console.log(ctx.message);

  if (isChat(ctx.chat.id, ctx.message.from.id)) return;
  await ctx.replyWithPhoto(
```

```
    { source: fs.createReadStream("./static/hello-msg.png") },
    startMenuButtons()
  );
  await ctx.replyWithMarkdown(
    `Привіт, ${ctx.from.first_name}!` + texts.enterMessage,
    startMenuButtons()
  );

  const id = ctx.from.id;
  const user = await User.findOne({ where: { telegramId: id.toString() } });
  if (!user) {
    const surname = ctx.from.last_name ? ctx.from.last_name : "";
    const username = ctx.from.username ? ctx.from.username : "";
    createUser(id.toString(), ctx.from.first_name, surname, username);
  }
});

bot.use(async (ctx, next) => {
  const chatType = ctx.chat.type;

  if (chatType !== "group" && chatType !== "supergroup" && chatType !== "channel") {
    return next();
  }

  if (ctx.from.id === ctx.botInfo.id) {
    return next();
  }

  let newMember;
  let isSubscribeNewMember = false;
  const isSubscribe = await getSubscribe(ctx.from.id);

  if (ctx.message.new_chat_member) {
    newMember = ctx.message.new_chat_member;
    isSubscribeNewMember = await getSubscribe(newMember.id.toString());
  }

  if (ctx.chat.id.toString() === process.env.CHAT_ID) {
    if (!isSubscribe) {
      await bot.telegram.banChatMember(process.env.CHAT_ID, id);
      await bot.telegram.unbanChatMember(process.env.CHAT_ID, id);
      await bot.telegram.banChatMember(process.env.CHANNEL_ID, id);
      await bot.telegram.unbanChatMember(process.env.CHANNEL_ID, id);
      return next();
    }
    if (newMember && !isSubscribeNewMember) {
      await bot.telegram.banChatMember(process.env.CHAT_ID, newMember.id);
      await bot.telegram.unbanChatMember(process.env.CHAT_ID, newMember.id);
      await bot.telegram.banChatMember(process.env.CHANNEL_ID, newMember.id);
```

```
      await bot.telegram.unbanChatMember(process.env.CHANNEL_ID, newMember.id);
      return next();
    }
    if (newMember && isSubscribeNewMember) {
      return next();
    }
  }

  return next();
});

bot.use(async (ctx, next) => {
  const id = ctx.from.id.toString();

  if (!(await getIsBlocked(id))) {
    return next();
  }
});

bot.hears("Головне меню", async ctx => {
  const isSubscribe = await getSubscribe(ctx.from.id.toString());

  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  await ctx.replyWithPhoto(
    { source: fs.createReadStream("./static/main-menu.png") },
    startMenuButtons(isSubscribe)
  );
});

bot.hears("Що я отримаю?", async ctx => {
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  ctx.replyWithMarkdown(texts.aboutChannelMessage, aboutChannelButtons());
});

bot.hears("Про авторів", async ctx => {
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;
  ctx.replyWithMarkdown(texts.aboutAuthorsMessage, authorInfoButtons());
});

bot.hears("FAQ", async ctx => {
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;
  ctx.replyWithMarkdown(texts.faqMessage, faqButtons());
});

bot.hears("Технічна підтримка", async ctx => {
  ctx.scene.enter("supportScene");
});
```

```
bot.hears("Долучитись", async ctx => {
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  await ctx.replyWithPhoto(
    { source: fs.createReadStream("./static/add-subscr.png") },
    joinInfoButtons()
  );
});

bot.hears("Хочу в ком'юніті", async ctx => {
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  await ctx.replyWithPhoto(
    { source: fs.createReadStream("./static/add-subscr.png") },
    joinInfoButtons()
  );
});

bot.hears("Особистий кабінет", async ctx => {
  const isSubscribe = await getSubscribe(ctx.from.id.toString());

  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  await ctx.reply(`Добрий день, ${ctx.from.first_name}`, accountButtons(isSubscribe));

  await subscribeInfo(ctx);
});

bot.hears("Долучитись до чату", async ctx => {
  const isSubscribe = await getSubscribe(ctx.from.id.toString());
  if (!isSubscribe) return;

  if (isChat(ctx.chat.id, ctx.message.from.id)) return;
  try {
    const { invite_link } = await ctx.telegram.createChatInviteLink(process.env.CHAT_ID, {
      member_limit: 1,
    });
    await ctx.replyWithPhoto({ source: fs.createReadStream("./static/access-chat.png") });
    await ctx.reply(`${invite_link}`, accountButtons(true));
  } catch (e) {
    console.log(e);
    ctx.reply("Помилка при створенні посилання, спробуйте пізніше.");
  }
});

bot.hears("Долучитись до каналу", async ctx => {
  const isSubscribe = await getSubscribe(ctx.from.id.toString());
  if (!isSubscribe) return;
```

```
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;
  try {
    const { invite_link } = await ctx.telegram.createChatInviteLink(process.env.CHANNEL_ID, {
      member_limit: 1,
    });
    await ctx.replyWithPhoto({ source: fs.createReadStream("./static/access-channel.png") });
    await ctx.reply(`${invite_link}`, accountButtons(true));
  } catch (e) {
    console.log(e);
    ctx.reply("Помилка при створенні посилання, спробуйте пізніше.", onlyMenuButtons());
  }
});

bot.hears("Долучитись до сайту", async ctx => {
  const tgId = ctx.from.id.toString();
  const isSubscribe = await getSubscribe(ctx.from.id.toString());
  if (!isSubscribe) return;

  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  if (!(await getEmail(tgId))) {
    await ctx.replyWithPhoto(
      { source: fs.createReadStream("./static/ask-email.png") },
      onlyMenuButtons()
    );
    return;
  }

  // if (!(await getWebflowId(tgId))) {
  //   const data = await createNewUser(await getEmail(tgId));
  //   await setWebflowId(tgId, data._id);
  // }
  await ctx.replyWithPhoto({ source: fs.createReadStream("./static/access-site.png") });
  ctx.replyWithMarkdown(
    "Отримуй повідомлення на email та [долучайся](https://navchalnya.fooror.com/log-in)! ",
    accountButtons(isSubscribe)
  );
});

bot.hears("Змінити email", async ctx => {
  ctx.reply("Введіть адресу Вашої електронної пошти.", onlyMenuButtons());
});

bot.hears(/^[^\s@]+@[^\s@]+\.[^\s@]+$/, async ctx => {
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  await setEmail(ctx.from.id.toString(), ctx.message.text);
  await ctx.replyWithPhoto(
```

```
    { source: fs.createReadStream("./static/update-email.png") },
    onlyMenuButtons()
  );
});

bot.hears("Документація", async ctx => {
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  ctx.replyWithMarkdown(texts.documentationMessage, subscribeButtons());
});

bot.hears("Придбати підписку", async ctx => {
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  ctx.replyWithMarkdown(texts.subscribeAmountMessage, subscribeButtons());
});

bot.hears("Пробний період", async ctx => {
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  if (await getTestPeriod(ctx.from.id.toString())) {
    ctx.replyWithPhoto(
      { source: fs.createReadStream("./static/cancel-test.png") },
      subscribeButtons()
    );
    return;
  }
  const orderId = await generateValidOrderId();
  const paymentLink = await getPayment(orderId, 100);

  const inlinePayButton = Markup.inlineKeyboard([Markup.button.url("Сплатити", paymentLink)]);
  ctx.replyWithMarkdown(texts.testPeriodMessage, inlinePayButton);
  checkPayment(ctx.from.id.toString(), orderId, "test", ctx);
});

bot.hears("1 Місяць", async ctx => {
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  const orderId = await generateValidOrderId();
  const paymentLink = await getPayment(orderId, 49000);
  const inlinePayButton = Markup.inlineKeyboard([Markup.button.url("Сплатити", paymentLink)]);
  await ctx.replyWithPhoto(
    { source: fs.createReadStream("./static/payment-link.png") },
    inlinePayButton
  );
  checkPayment(ctx.from.id.toString(), orderId, "month", ctx);
});

bot.hears("3 місяці", async ctx => {
```

```
  if (isChat(ctx.chat.id, ctx.message.from.id)) return;

  const orderId = await generateValidOrderId();
  const paymentLink = await getPayment(orderId, 139000);
  const inlinePayButton = Markup.inlineKeyboard([Markup.button.url("Сплатити", paymentLink)]);
  await ctx.replyWithPhoto(
    { source: fs.createReadStream("./static/payment-link.png") },
    inlinePayButton
  );
  checkPayment(ctx.from.id, orderId, "threeMonths", ctx);
});

bot.hears("1 Рік", async ctx => {
  if (isChat(ctx.chat.id, ctx.message.from.id)) {
    return;
  }
  const orderId = await generateValidOrderId();
  const paymentLink = await getPayment(orderId, 539000);
  const inlinePayButton = Markup.inlineKeyboard([Markup.button.url("Сплатити", paymentLink)]);
  await ctx.replyWithPhoto(
    { source: fs.createReadStream("./static/payment-link.png") },
    inlinePayButton
  );
  checkPayment(ctx.from.id, orderId, "year", ctx);
});

bot.hears("Скасувати підписку", async ctx => {
  ctx.reply(
    "Ви впевнені, що хочете скасувати підписку? Після закінчення терміну дії підписки
списання коштів не відбудеться, але вартість підписки може збільшитись згодом.",
    cancelSubscribeButtons()
  );
});

bot.hears("Так, скасувати підписку", async ctx => {
  await setLastTransaction(ctx.from.id.toString(), "");
  ctx.replyWithPhoto({ source: fs.createReadStream("./static/ok.png") }, startMenuButtons());
});

bot.hears("Ні, залишитися.", async ctx => {
  ctx.replyWithPhoto(
    { source: fs.createReadStream("./static/cancel-smile.png") },
    startMenuButtons()
  );
});

module.exports = { bot };
```

Програмний код інтеграції ChatGPT до Telegram-боту:

```
const axios = require("axios");

class ChatGptService {
  constructor() {
    this.gptUrl = "https://api.openai.com/v1/chat/completions";
    this.apiKey = process.env.GPT_TOKEN;
  }

  generateResponse(messages) {
    const headers = {
      "Content-Type": "application/json",
      Authorization: `Bearer ${this.apiKey}`,
    };

    const data = {
      model: "gpt-4",
      messages: messages,
      temperature: 0.6,
    };

    return axios
      .post(this.gptUrl, data, { headers })
      .then(response => response.data.choices[0].message.content.trim())
      .catch(err => {
        this.logger.error(err);
        return "error";
      });
  }
}

module.exports = { ChatGptService };
```

Додаток Б

Публікації автора за темою кваліфікаційної роботи

ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

WEST UKRAINIAN NATIONAL UNIVERSITY

Комп'ютерна інженерія

ФКІТ FCIT

ЗБІРНИК ПРАЦЬ

VIII НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ МОЛОДИХ ВЧЕНИХ І СТУДЕНТІВ «ІНТЕЛЕКТУАЛЬНІ КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ»

5 ГРУДНЯ 2023

Наукове товариство ЗУНУ

KI.WUNU.EDU.UA/CONFERENCE/

ТЕРНОПІЛЬ

2023

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Голова оргкомітету: Березький О.М. д.т.н., професор кафедри комп'ютерної інженерії

Члени оргкомітету:

Мельник Г.М., к.т.н., доцент, Західноукраїнський національний університет (керівник)
Піцун О.Й. , к.т.н., доцент, Західноукраїнський національний університет
Савка Н.Я. к.т.н., доцент, Західноукраїнський національний університет
Лящинський П.Б., аспірант, Західноукраїнський національний університет

ПРОГРАМНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Березький О. М. д.т.н., професор, Західноукраїнський національний університет (голова)
Антощук С.Г. , д.т.н, професор, Національний університет «Одеська політехніка»
Батько Ю.М. к.т.н., доцент, Західноукраїнський національний університет
Возна Н.Я. д.т.н., професор, Західноукраїнський національний університет
Говорущенко Т.О. , д.т.н., професор, Хмельницький національний університет
Дубчак Л.О. зав. кафедри КІ, к.т.н., доцент, Західноукраїнський національний університет
Ізонін І.В. к.т.н., доцент, НУ "Львівська політехніка"
Литвиненко В.І. д.н.т, професор, Херсонський національний технічний університет
Лупенко С.А., д.н.т, професор, Тернопільський національний технічний університет імені Івана Пулюя
Мельник Г.М. к.т.н, доцент, Західноукраїнський національний університет
Піцун О.Й., к.т.н. доцент, Західноукраїнський національний університет
Субботін С.О., д.т.н., професор, Національний університет «Запорізька політехніка»
Цмоць І.Г., д.т.н., професор, НУ "Львівська політехніка"
Яровий А.А., д.т.н., професор, Вінницький національний технічний університет
Яцків В.В., д.т.н., професор, Західноукраїнський національний університет

Дюг Д.А.
магістрант 2 курсу ФКІТ ЗУНУ
Науковий керівник д.т.н. Комар М.П., кафедра ІОСУ ЗУНУ

ІНТЕГРАЦІЯ CHATGPT ДО TELEGRAM-БОТУ

Вступ. Штучний інтелект (ШІ) перетворився на ключовий елемент трансформації в сучасному світі, реформуючи все, від особистого до професійного життя. Це дослідження фокусується на важливості та впливі ШІ в сучасних реаліях, використовуючи знання з різноманітних джерел. ШІ вказує на створення та вдосконалення комп'ютерів та машин, які здатні самостійно виконувати завдання, обробляти дані та приймати рішення без безпосередньої участі людини. Системи штучного інтелекту здатні на незалежне мислення, імітуючи людські когнітивні функції на базовому рівні.

Глибока інтеграція ШІ у повсякденне життя охоплює широкий спектр застосувань. Це включає в себе все, від алгоритмів у соціальних мережах, які формують контент, до медичного ШІ, що сприяє в діагностиці захворювань. Штучний інтелект стає невіддільною частиною нашого сучасного життя.

Постановка задачі. Об’єкт дослідження – процес інтеграції чат-боту Telegram з ШІ. Предмет дослідження – засоби інтеграції чат-боту Telegram з моделлю ШІ ChatGPT. Мета дослідження полягає у розробці та аналізі ефективності чат-боту Telegram, інтегрованого з моделлю ШІ ChatGPT.

Основний матеріал. ChatGPT має широкі можливості для застосування у повсякденному житті та у бізнесі прямо зараз, а саме: обробка природної мови (NLP), розв’язання логічних проблем, задач та прикладів на текстовій основі, генерація коду на різних мовах програмування, а також адаптивне навчання [1].

Telegram – один з найбільш популярних месенджерів не тільки в Україні, але і у світі, багато великих компаній використовують його як канал спілкування зі своїми клієнтами, в тому числі, як засіб технічної підтримки. Виходячи з цього комбінація Telegram і ChatGPT – це чудова можливість для бізнесу оптимізувати процес комунікації з клієнтами.

Для розробки Telegram-бота було обрано Node.js. Це платформа на стороні сервера, яка дозволяє використовувати JavaScript для створення швидких та масштабованих мережевих додатків [2].

Дослідження зосереджується на різних аспектах взаємодії чат-бота з користувачами, включаючи його здатність розуміти та адекватно реагувати на запити людей, обробку природної мови, генерацію змістовних та контекстуально відповідних відповідей, а також його можливості в автоматизації обслуговування клієнтів та наданні технічної підтримки. Особлива увага приділяється аналізу того, як чат-бот може вдосконалювати комунікаційні процеси між бізнесом та його клієнтами, а також оцінці його потенціалу в підвищенні ефективності взаємодії.

Висновки. Штучний інтелект (ШІ) став ключовою трансформуючою силою у сучасному світі, впливаючи на різні аспекти життя. Це дослідження акцентує на інтеграції ШІ у чат-боти Telegram, зокрема на використанні моделі ChatGPT, для покращення комунікаційних процесів у бізнесі. Основна увага зосереджена на розвитку та аналізі ефективності чат-ботів, які можуть автоматизувати взаємодію з клієнтами та надавати підтримку за допомогою обробки природної мови. Вибір Node.js для розробки підкреслює перевагу використання швидких і масштабованих технологій у сучасних цифрових рішеннях.

Список літератури

1. Text Generation Models [Електронний ресурс]. – Режим доступу: https://platform.openai.com/docs/guides/text-generation.

2. Кошова О. П., Ольховська О. В., Тацій Д. С., Олексійчук Ю. Ф., Черненко О. О. Розробка веб-додатків та сервісів на платформі Node.js. Таврійський науковий вісник. Серія: Технічні науки. 2023. №2. С. 78-89.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 001
М34

ПРОГРАМНИЙ КОМІТЕТ

Голова: Приймак Микола – професор кафедри комп'ютерних систем та мереж, д.т.н., професор.

Співголови: Марущак Павло – проректор з наукової роботи, докт. техн. наук, професор.
Баран Ігор – канд. техн. наук, доцент, декан факультету ФІС.
Науковий секретар: Семенишин Галина – старший викладач.

Члени: Василь Кривень - завідувач кафедри математичних методів в інженерії д.ф.-м.н., професор; Галина Осухівська – завідувач кафедри комп'ютерних систем та мереж, к.т.н., доцент; Микола Карпінський - професор кафедри кібербезпеки, д.т.н., професор; Жанна Баб'як - завідувач кафедри української та іноземних мов, к.пед. н., доцент; Ярослав Литвиненко – професор кафедри комп'ютерних наук, д.т.н., професор; Михайло Петрик - завідувач кафедри програмної іженерії, д.ф.-м.н., професор; Наталія Загородна – завідувач кафедри кібербезпеки, к.т.н., доцент.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

Голова: Скоренький Юрій Любомирович – канд. техн. наук, доцент кафедри фізики.
Члени: доцент кафедри комп'ютерних наук, к.т.н. В. Никитюк; доцент кафедри програмної інженерії, к.т.н. Д. Михалик; доцент кафедри кібербезпеки, к.т.н. М. Стадник; асистент Н. Шаблій; ст. викладач Л. Джиджора.

М34 Матеріали XI науково-технічної конфції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 13-14 грудня 2023 р.). – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2023. – 257 с.

Адреса оргкомітету: ТНТУ ім. І. Пулюя, м. Тернопіль, вул. Руська, 56, 46001,
тел. (0352) 52-41-33, факс (0352) 254983.
E-mail: **conffis2023@gmail.com**

Редагування, оформлення, верстка: Семенишин Г.М.

СЕКЦІЇ КОНФЕРЕНЦІЇ, ЯКІ ПРЕДСТВЛЕНІ В ЗБІРНИКУ

- Математичне моделювання;
- Інформаційні системи та технології;
- Комп'ютерні системи та мережі;
- Програмна інженерія та моделювання складних розподілених систем;
- Новітні фізико-технічні та освітні технології.

В збірнику надруковано тези доповідей XI науково-технічної конференції «Інформаційні моделі, системи та технології» (Тернопіль, 13-14 грудня 2023 р.) за такими науковими напрямками: математичне моделювання; інформаційні системи та технології; комп'ютерні системи та мережі; програмна інженерія та моделювання складних розподілених систем; новітні фізико-технічні та освітні технології.

Розрахований на науковців, викладачів та студентів вузів.

За зміст тез та дотримання норм академічної доброчесності відповідальність несе автор.

УДК 004.6+004.9
Дмитро Дюг
Західноукраїнський національний університет

МЕТОД ІНТЕГРАЦІЇ CHATGPT ДО TELEGRAM-БОТА

Dmytro Diuh

CHATGPT INTEGRATION METHOD TO TELEGRAM BOT

Основною метою цієї роботи є розробка методології для інтеграції ChatGPT, передової моделі розмовного штучного інтелекту, розробленої OpenAI, в бот Telegram. Ця інтеграція має на меті покращити можливості ботів Telegram, використовуючи вдосконалені здібності обробки природної мови та розуміння ChatGPT. Головна увага приділяється створенню безперешкодного, ефективного та зручного інтерфейсу, який сприяє ефективній комунікації та взаємодії між користувачами Telegram та ботом, що працює на основі ChatGPT. [1]

Дослідження буде вивчати технічні та практичні аспекти цієї інтеграції, а саме:

1. Створення Telegram-боту: Процес ініціації Telegram-бота починається з взаємодії з BotFather, офіційним ботом Telegram для створення та управління іншими ботами. Під час реєстрації нового бота, BotFather генерує унікальний авторизаційний токен. Цей токен є ключем для взаємодії з Telegram Bot API, дозволяючи програмі ідентифікувати та виконувати дії від імені бота.
2. Налаштувапння серверного середовища: ефективна робота Telegram-бота вимагає стабільного серверного середовища, яке може підтримувати постійне з'єднання з Telegram API та ChatGPT. Сервер повинен бути налаштований з врахуванням аспектів масштабованості, надійності та швидкості відповіді, що включає оптимізацію ресурсів, балансування навантаження та запобігання збоям.
3. Інтеграція з ChatGPT через OpenAI API: це вимагає отримання API ключа, та реалізації HTTP запитів до OpenAI для виклику функцій ChatGPT. Належне управління API запитами є критично важливим для забезпечення стабільності та відповідності часових обмежень у відповідях.
4. Обробка команд і повідомлень користувачів: бот повинен ефективно обробляти вхідні дані від користувачів, включаючи текстові повідомлення та команди. Це включає розпізнавання природної мови, аналіз контексту, та формування відповідних запитів до ChatGPT. Алгоритми обробки повинні бути оптимізовані для швидкого реагування та точності в інтерпретації запитів.
5. Розробка та інтеграція системи управління: для зручного адміністрування чат-боту необхідно розробити систему управління за допомогою якої буде можливо керувати користвачами, які запустили бота, та збирати статистичні дані щодо використання бота.

Літсратура

1. Кошова, О. П., Ольховська, О. В., Тацій, Д. С., Олексійчук, Ю. Ф., & Черненко, О. О. (2023). РОЗРОБКА ВЕБ-ДОДАТКІВ ТА СЕРВІСІВ НА ПЛАТФОРМІ NODE.JS. Таврійський науковий вісник. Серія: Технічні науки, (2), 78-89. https://doi.org/10.32782/tnv-tech.2023.2.9