МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

Тема: «Телеграм бот на базі технології ChatGPT-3.5»

Виконавець: Ренат КВАШУК

Керівник: к.т.н., доцент Олена ТОЛСТІКОВА

Нормоконтролер: к.т.н., доцент Олена ТОЛСТІКОВА

КИЇВ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*
Кафедра *комп'ютерних інформаційних технологій*
Спеціальність *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)

«____»_____2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
Квашука Рената Олеговича
(ПІБ випускника)

1. Тема роботи: «Телеграм бот на базі технології ChatGPT-3.5» затверджена наказом ректора № 1976/ст від 29.09.2023р.
2. Термін виконання роботи: з 02 жовтня 2023 року по 31 грудня 2023 року.
3. Вихідні дані до роботи: телеграм бот на базі технології ChatGPT-3.5.
4. Зміст пояснювальної записки: 1. Аналіз та поняття технології ChatGPT. 2. Засоби проектування телеграм-бота. 3. Розробка та тестування телеграм-бота.
5. Перелік обов'язкового ілюстративного матеріалу: 1. Вигляд телеграм боту. 2. Чат. Кнопка «Старт». 3. Початок роботи. 4. Повідомлення про вибір мови. 5. Обробка запиту. 6. Обробка та надання відповіді. 7. «Меню». 8. Блок схема принципу роботи телеграм-бота.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Аналіз предметної області та огляд аналогів. Написання розділу «Аналіз та поняття технології ChatGPT»	02.10.2023-16.10.2023	
2.	Вибір та опис використаних технологій. Написання розділу «Засоби проектування телеграм-бота»	17.10.2023-30.10.2023	
3.	Написання розділу «Розробка та тестування телеграм-бота»	31.10.2023-14.11.2023	
4.	Загальне редагування та друк пояснювальної записки	15.11.2023-20.11.2023	
5.	Проходження нормоконтролю, перепліт пояснювальної записки.	16.11.2023-20.10.2023	
6.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	19.12.2023-22.12.2023	

7. Дата видачі завдання 02.10.2023 р.

Керівник кваліфікаційної роботи _____ Олена ТОЛСТІКОВА
(підпис керівника)

Завдання прийняв до виконання _____ Ренат КВАШУК
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Телеграм бот на базі технології ChatGPT-3.5» містить: 93 сторінки, 33 рисунки, 16 інформаційних джерел, 1 додаток.

Об'єктом досліджень – створення телеграм-бота з використанням технології обробки живої мови OpenAI.

Предметом досліджень – аналіз і проектування телеграм-бота та його компонентів для опрацювання запитів.

Мета кваліфікаційної роботи – отримати кінцевий продукт у вигляді телеграм-боту, побудованого на базі штучного інтелекту для демонстрації технології обробки природної мови.

Методи дослідження – мова програмування JavaScript, інтегроване середовище розробки WebStorm**.**

Результати кваліфікаційної роботи рекомендується використовувати для демонстрації роботи технології обробки природної мови, та в подальшому інтеграції у компанії та підприємства.

ТЕЛЕГРАМ БОТ, ОБРОБКА ПРИРОДНОЇ МОВИ, БЕЗПЕКА, JAVASCRIPT, WEBSTORM , OPENAI API.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ……7

ВСТУП…….8

РОЗДІЛ 1. АНАЛІЗ ТА ПОНЯТТЯ ТЕХНОЛОГІЇ CHATGPT ……....11

1.1. Етапи розвитку телеграм-ботів..…….....……..……...….....11

1.2. Чат-бот та його основні функціональні можливості……..…..12

1.3. Використання чат-ботів……..…14

1.4. Основні технології та алгоритми, що застосовуються при створенні телеграм-ботів на базі штучного інтелекту…...……17

1.5. Можливості та приклади впровадження телеграм-ботів на базі штучного інтелекту в різних сферах діяльності…...……..18

1.6. Аналіз впливу телеграм-ботів на процес спілкування в онлайн-середовищах: переваги та недоліки…….....20

1.7. Технології ChatGPT: Розвиток та Перспективи. Історія розвитку ChatGPT…...……...21

1.8. Принципи роботи ChatGPT…….22

1.9. Вплив ChatGPT на сучасне спілкування……26

ВИСНОВКИ ДО 1 РОЗДІЛУ ……...……32

РОЗДІЛ 2. ЗАСОБИ ПРОЕКТУВАННЯ ТЕЛЕГРАМ-БОТА....…….34

2.1. Опис предметної області ……...……...……..34

2.2. Опис вимог телеграм-бота…..…….....35

2.3. Мова програмування JavaScript ……...……..36

2.4. Дослідження популярних фреймворків для створення Telegram - бота з використанням мови JavaScript…….....41

2.5. Розпізнавання мовлення від Google Cloud ……48

2.6. Середовище розробки Webstorm ……..63

ВИСНОВКИ ДО 2 РОЗДІЛУ ……...……....65

РОЗДІЛ 3. РОЗРОБКА ТА ТЕСТУВАННЯ ТЕЛГРАМ-БОТА….……66

3.1. Принцип роботи телеграм бота……...……...…….66

3.2. Проєктування телеграм-бота…….....73

ВИСНОВКИ ДО 3 РОЗДІЛУ ……...……....90

ВИСНОВКИ……..91
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ…….93
ДОДАТОК А…….94

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

API (Application Programming Interface)	–	Програмний інтерфейс.
ChatGPT (Chat Generative Pre-trained Transformer)	–	Модель генерації тексту від OpenAI.
IDE (Integrated Development Environment)	–	Інтегроване середовище розробки
NodeJS (Node JavaScript)	–	Середовище виконання JavaScript.
URL (Uniform Resource Locator)	–	Визначник місцезнаходження сайту.
GCloud (Google Cloud)	–	Облачна платформа Google Cloud.
AI (Artificial Intelligence)	–	Штучний інтелект.
SDK (Software Development Kit)	–	Набір для розробки програмного забезпечення.
JSON (JavaScript Object Notation)	–	Відмінний об'єктовий формат.
CLI (Command Line Interface)	–	Командний рядок інтерфейсу.
DB (Database)	–	База даних.

ВСТУП

Сучасний світ інформаційних технологій надає безмежні можливості для розробки інноваційних програмних рішень, спрямованих на полегшення та покращення спілкування користувачів з комп'ютерами та електронними пристроями. Однією з передових технологій в цьому контексті є штучний інтелект, який здатний аналізувати тексти, взаємодіяти з користувачами та надавати інформаційну підтримку.

Кваліфікаційна робота присвячена створенню та дослідженню телеграм-бота, побудованого на основі передових технологій штучного інтелекту, зокрема моделі GPT-3.5. Цей бот призначений для автоматизованої обробки повідомлень в месенджері Telegram та надання інтелектуальних відповідей на запити користувачів.

Дана кваліфікаційна робота розглядає технічні аспекти розробки та функціональні можливості розробленого телеграм-бота на базі технології ChatGPT-3.5. Досліджуються методи взаємодії бота з користувачами, його здатність адаптуватися до різноманітних завдань та роль в поліпшенні спілкування в онлайн-середовищі.

Метою роботи є розробка та впровадження Телеграм бота, який використовує технологію ChatGPT-3.5 для забезпечення ефективної та інтерактивної взаємодії з користувачами. Проект передбачає створення бота, який здатний розуміти та відповідати на різноманітні запитання, а також навчатися від користувачів та адаптуватися до нових сценаріїв використання.

Для досягнення поставленої мети необхідне рішення наступних завдань:

1. Розробка архітектури бота;
2. Інтеграція ChatGPT-3.5;
3. Навчання та адаптація бота;

Об'єктом досліджень є створення телеграм-бота з використанням технології обробки живої мови OpenAI.

Предметом досліджень є аналіз і проектування телеграм-бота та його компонентів для опрацювання запитів.

Актуальність теми кваліфікаційної роботи: зростання популярності та використання месенджерів та соціальних мереж створює потребу в інтелектуальних ботах, які можуть забезпечити користувачам не тільки інформаційну підтримку, але й інтерактивну взаємодію. Технологія ChatGPT-3.5 визначається своєю здатністю адаптуватися до контексту та вивчати з інтеракцій, що робить її ідеальним інструментом для створення інтелектуальних ботів у месенджері Telegram. Використання цієї технології може значно поліпшити якість обслуговування користувачів та забезпечити новий рівень персоналізації взаємодії.

Відповідно до поставленої мети роботи визначено основні **завдання дослідження**:

– визначити структуру бота, яка включає в себе модулі взаємодії з користувачем, обробки запитань, інтеграції з ChatGPT-3.5 та генерації відповідей.

– розробити механізми комунікації з API ChatGPT-3.5

– впровадити методи оптимізації взаємодії для забезпечення ефективності та швидкості відповідей.

– розробити телеграм-бота для демонстрації роботи технології обробки живої мови;

Для досягнення мети дослідження та виконання поставлених завдань використано метод системно-структурного аналізу наукової літератури в контексті створення інтелектуального Телеграм бота на основі технології ChatGPT-3.5. Цей метод дозволив детально розглянути особливості та аспекти використання ChatGPT-3.5 в порівнянні з іншими методиками розпізнавання природної мови та створення інтелектуальних ботів.

Для систематизації отриманих висновків та формулювання докладних результатів дослідження були використані методи аналізу, формалізації, абстрагування та узагальнення. Це дозволило здійснити глибокий огляд наявних підходів до створення інтелектуальних ботів, які базуються на технології ChatGPT-3.5.

Крім того, використання цих методів дозволило виявити ключові аспекти архітектури бота, ефективність його взаємодії з API ChatGPT-3.5 та можливості навчання та адаптації до потреб користувачів. Такий підхід створив основу для подальших етапів дослідження та розробки інтелектуального Телеграм бота.

Наукова новизна дослідження виявляється у створенні інтелектуального Телеграм бота, який забезпечить мобільний та легкий доступ до технології обробки природної мови. Цей бот може бути потенційно інтегрований у різноманітні пристрої, від смартфонів до систем відеоспостереження, надаючи користувачам доступ до інтелектуального асистента у будь-який час і в будь-якому місці.

Дослідження передбачає великий попит на проект через його доступність, легкість використання та зрозумілість для кінцевого користувача. Створений інструмент може стати цінним активом для підприємств та компаній, що прагнуть покращити свою кібербезпеку та захистити конфіденційну інформацію від неправомірного доступу. Такий бот відкриває нові перспективи у впровадженні ефективних засобів кіберзахисту та може відгукнутися на потреби широкого кола користувачів та підприємств.

РОЗДІЛ 1
АНАЛІЗ ТА ПОНЯТТЯ ТЕХНОЛОГІЇ CHATGPT

1.1. Етапи розвитку телеграм-ботів

Початки розвитку телеграм-ботів. Спочатку телеграм-боти з'явилися на початку 2010-х років, коли Telegram набув популярності. Перші боти були обмеженими за можливостями та функціями, використовувалися переважно для надсилання повідомлень та стандартних операцій.

Запуск Bot API та розширення можливостей. У 2015 році Telegram запустив Bot API, що відкрило нові горизонти для розробників та розширило можливості телеграм-ботів. Це дозволило створювати більш складні та інтерактивні боти з широким спектром функціональних можливостей, включаючи створення ігор, опитувань та обробку зображень.

Інтеграція з зовнішніми сервісами та API. Пізніше боти отримали можливість інтеграції з зовнішніми сервісами та API, що розширило їхню функціональність. Вони стали спроможними виконувати різноманітні завдання, включаючи роботу з геолокацією, роботу з базами даних, оплату послуг та інтерактивність.

Використання штучного інтелекту. Важливим кроком у розвитку телеграм-ботів стало використання штучного інтелекту та мовних моделей, таких як GPT (Generative Pre-trained Transformer). Це підвищило інтелектуальну здатність ботів та зробило їх більші інтерактивними та здатними розуміти природну мову користувачів.

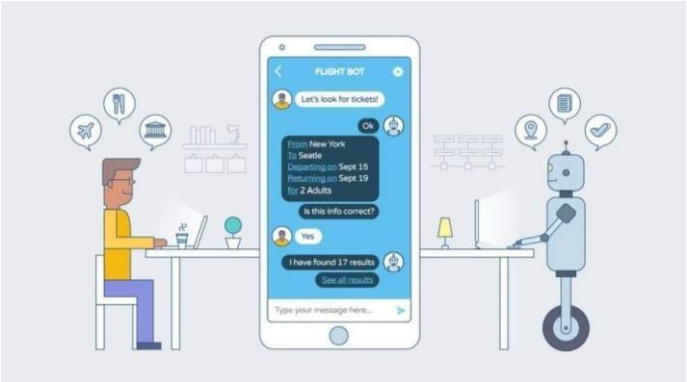
Сучасний стан та вплив телеграм-ботів. Сьогодні телеграм-боти стали важ-ливою частиною цифрового спілкування. Вони використовуються у бізнесі для автоматизації процесів, у навчанні для створення навчальних платформ, у медицині для консультування пацієнтів, у розважальній індустрії

та багатьох інших сферах. Телеграм-боти покращують ефективність спілкування та надають доступ до нових сервісів та можливостей.

1.2. Чат-бот та його основні функціональні можливості

Чат-бот (англ. chatbot) - це програмний продукт, який дає змогу моделювати реальну розмову з клієнтом [2]. Цей віртуальний компаньйон представляє собою автоматизовану систему спілкування з користувачами та допомагає підприємцям стати ближче зі своїми покупцями (рис. 1.1). Окрім того, швидкість опрацювання заявки, яку залишив клієнт, чат-ботом допомагає завойовувати його лояльність та бажання купувати саме в цій компанії.

Більшість ботів запрограмовані на певні ланцюги розвитку подій, що ведуть до певного результату та надання запрограмованих даних користувачеві. Але досягнення в області штучного інтелекту (ШІ) в



комбінації з розвитком месенджерів сприяють їх розвитку.

Рис. 1.1. Наглядний приклад взаємодії клієнта з чат-ботом

Основною ціллю використання цього програмного продукту є допомога

клієнтам якомога швидше досягти своєї кінцевої мети: отримання додаткової інформації чи продаж товару або послуг. Зрештою, бота можна запрограмувати так, щоб він міг цілодобово повторювати успіх агентів по роботі з клієнтами.

Для правильного підходу потрібно зрозуміти, які є способи інтегрування чат-бота у необхідний проект чи бізнес та які приховані від звичайного користувача «підводні камені» має в собі цей метод.

Основними базовими питаннями можуть бути:

- визначення очікуваного результату - потрібно розуміти, що використання чат-бота не може, поки що, замінити повністю використання людей в обслуговуванні покупців, але може звести деякі задачі до вагомого мінімуму. Потрібно звернути увагу, що для користувача платформою важливо вести «живу» бесіду, тому не варто намагатися видати діалог з ботом за реальнулюдину і дати змогу переключитись на менеджера;

- ґрунтовно пізнати тему роботи з клієнтами - необхідно персоналізувати взаємодію з покупцем та централізувати інформацію про обслуговування користувачів, щоб вони мали змогу легко себе ідентифікувати для повторного використання «віртуального» менеджера. В іншому випадку це може призвести тільки до негативного досвіду від роботи з платформою;

- спрямувати увагу на дані - зберігання та збір, а потім і аналіз даних надає можливість відслідковувати тенденції. Це допоможе бренду створювати клієнтські профілі, які допомагають ще більше персоналізувати підходи до комунікації з клієнтами, а також отримувати дані по навігації сайтом,мобільним додатком або ж по самому чат-боту;

- продумати дизайн та структуру - створення ботів має певні шаблони та правила, на які потрібно звернути увагу для покращення досвіду їх використання у покупців. Орієнтація на знання розробки сайтів та мобільних додатків є помилковою у будуванні свого віртуального помічника, тому варто приділити особливу увагу цій темі перед початком роботи над ним. Необхідно розглянути питання: «Як має звучати ваша компанія з уст клієнта?», ретельно обміркувати особистий бренд.

1.3. Використання чат-ботів

Використання чат-ботів має певні плюси для автоматизації сервісу обслуговування покупців, а саме: забезпечення компанією технічної підтримки та цілодобового сервісного обслуговування для клієнтів.

Оформлення замовлень в режимі онлайн, а тим більше 24/7, вагомо виділяє компанію серед інших конкурентів;

Перенесення на чат-бота відповіді на популярні питання зменшує час очікування відповіді для користувача, що також позитивно впливає на довіру до магазину чи будь-якого бізнесу;

Допомагають охоплювати більше клієнтів. Використання месенджерів присутнє у вагомої кількості людей, тому застосування такого інструменту, як чат-бот, є цілком доцільним з точки зору збільшення об'єму продажів. Щоб залучити якомога більший об'єм нового трафіку клієнтів чат- бот повинен відповідати необхідним для цього вимогам:

- працювати без перебоїв та помилок;
- відповідати особистому бренду або компанії;
- забезпечувати використання лінгвістики та правил комунікування;
- мати чіткі та зрозумілі задачі для виконання;
- надавати достовірну інформацію та FAQ;
- працювати на різних пристроях.

Плідна співпраця з «живими» менеджерами. Чат-боти опрацьовують обмежений спектр питань, тому при необхідності і при виникненні ситуації, з якою програма матиме труднощі, варто визначити людину-агента та перенаправити її на рішення даного питання (рис. 1.2). Тому сміливо можна сказати, що боти в певній мірі звільняють консультанта від дрібної роботи та допомагають переключитися на більш складні та термінові задачі;

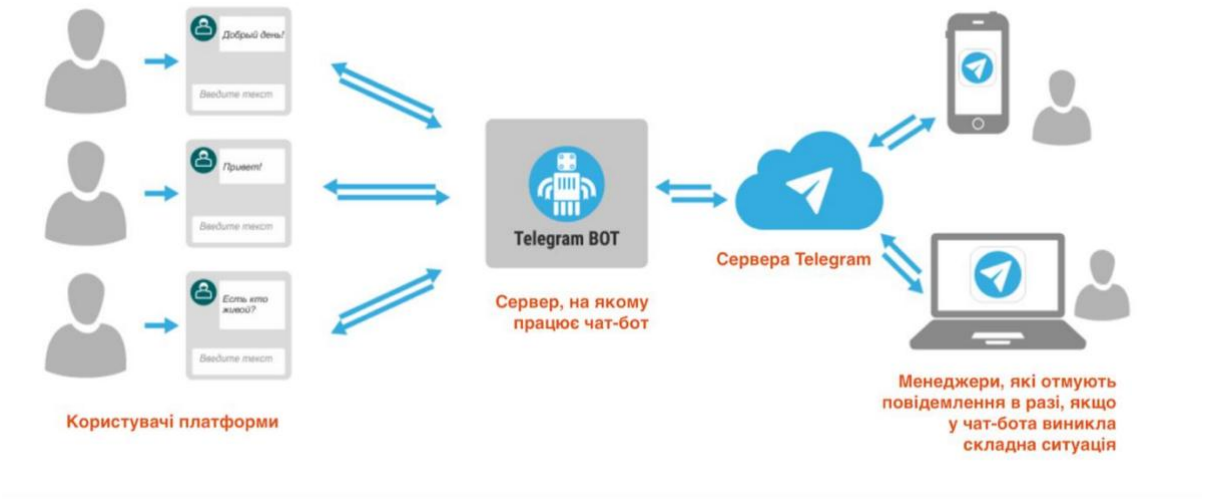


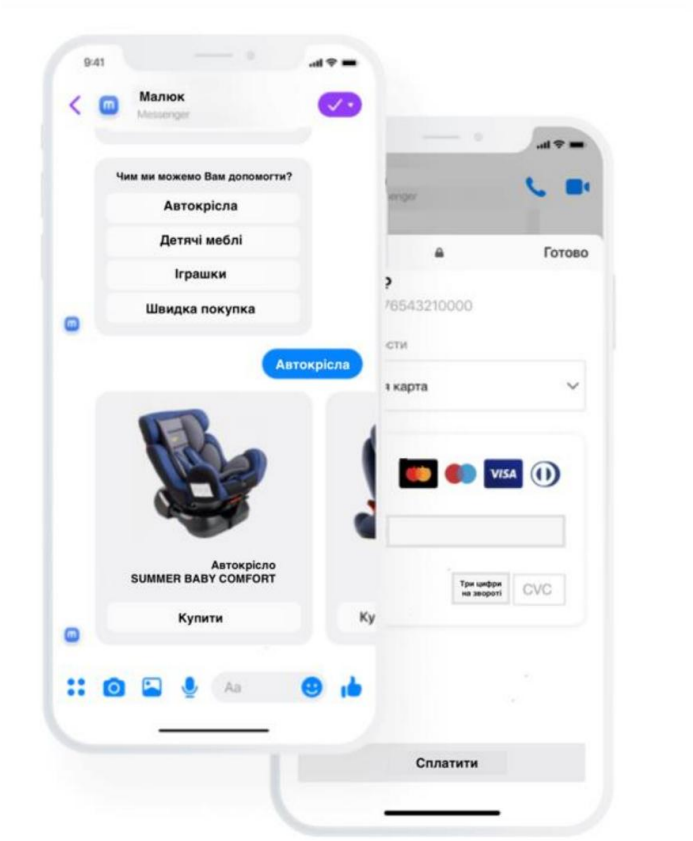
Рис. 1.2. Передання чат-ботом складної ситуації менеджеру

Допомагають зменшити операційні витрати на співробітників компанії. Для більшості компаній, а тим більше молодих, мати цілодобовупідтримку є занадто затратним, а саме чат-бот може вирішити це питання;

Надають сучасні та прогресивні інструменти для інтернет- маркетингу та продажів. Добре продумана структура віртуального менеджера може сприяти заключенню угод та закриттю продажів за допомогою використання надійних маркетингових стратегій. І це надає додаткові можливості:

− чат-бота можна налаштовувати на встановлення індивідуальної взаємодії з певним клієнтом, даючи надійні та перевірені поради, орієнтуючись на минулий досвід співпраці та продажів даному покупцеві;

− чат-боти надають змогу надавати повний супровід по воронці продажів за певними маркетинговими рекомендаціями та не заставляють клієнта очікувати в разі виникнення певних питань;



– є можливість налаштувати чат-бота певним чином, щоб він мав змогу передбачати всі кроки, які клієнт може зробити на шляху отримання бажаного товару, та його поведінку при покупці. Це дає змогу відправити потрібні акційні пропозиції тільки відповідним користувачам, що збільшує цільове попадання для заключення угоди (рис. 1.3);

Рис. 1.3. Приклад інтеграції онлайн-оплати в чат-бота

На даному етапі розвитку прикладного програмного інтерфейсу месенджерів можна підключити платформи для здійснення онлайн-оплат в роботу бота; Покращення роботоспроможності та функціональності відбувається за рахунок аналізу оцінки та оптимізації. Роботоспроможність та продуктивність покращуються за рахунок спостережень за такими напрямками; досвід користувача, якість та зручність використання, його лінгвістичні вміння. Покращення чат-бота має такі аспекти:

– привітальні та запрошувальні розсилки;

- додавання нової та сезонної продукції;
- різноманітні інформаційні блоки для поширення бренду;
- актуальні та вигідні знижки.

Наразі використання чат-ботів є хорошою альтернативою великому відділу продажів та технічній підтримці, тому може здаватись привабливою ідеєю автоматизації. Проте даний метод несе свої ризики і потрібно якомога відповідальніше поставитись до створення віртуального помічника. До мінусів використання ботів належать:

- не всі бізнес-моделі підтримують оптимізацію за допомогою даного методу. Для початку потрібно проаналізувати свою компанію, продукт та аудиторію, тому що не завжди впровадження даного ПЗ призводить до покращення продажів чи комунікації з покупцями;
- втрата сенсу відгуків від клієнтів через використання віртуальної техпідтримки, яка не відразу реагує на актуальні проблеми користувачів;
- одним з недоліків є невміння розпізнавати команди, які не були закладені в основу логіки бота. Люди можуть висловлюватись не точно по відношенню до товару чи послуги, в такому разі це буде викликати труднощі у наданні допомоги;
- відсутність емоцій є важливим бар'єром у спілкуванні з клієнтами. «Холодний розум» не завжди приносить позитивні наслідки, тому боти втрачають можливість діяти в залежності від емоцій, що може відштовхувати талякати покупців;
- необхідність в постійній оптимізації. Від розвитку та оптимізації сервісу буде залежати його відповідність вимогам клієнтів та його надійна працездатність. Слідкування за правильністю інформації, що надається клієнтам, і належність представлення бренду користувачам є трудомісткою по об'єму роботи задачею. По мірі змін в актуальності ринку та товарів, необхідно аналізувати досвід комунікації з користувачами та виявити нагальні питання, які варто вирішити.

1.4. Основні технології та алгоритми, що застосовуються при створенні телеграм-ботів на базі штучного інтелекту

Глибоке навчання (Deep Learning). Глибоке навчання є ключовою техноло-гією у створенні телеграм-ботів. Воно використовує нейронні мережі з багатьма шарами для аналізу та вивчення складних зв'язків у великих обсягах даних. Глибоке навчання використовується для розпізнавання образів, обробки природної мови, генерації тексту та багатьох інших завдань, які є важливими для телеграм-ботів.

Обробка природної мови (NLP). Обробка природної мови - це галузь шту-чного інтелекту, яка дозволяє комп'ютерам розуміти та взаємодіяти з людьми через природну мову. Вона використовується в телеграм-ботах для аналізу текстової інформації, розпізнавання мовних команд, відповіді на запити користувачів та автоматичного створення текстового контенту.

Машинне навчання (Machine Learning). Машинне навчання використовується для навчання телеграм-ботів адаптуватися до нових даних та ситуацій. Алгоритми машинного навчання дозволяють ботам покращувати свою ефективність та точність в реальному часі, враховуючи змінні умови.

Алгоритми розпізнавання образів. Ці алгоритми використовуються для аналізу та інтерпретації зображень, які можуть надходити до телеграм-бота че-рез камеру смартфону або завантажуватися користувачами. Вони дозволяють ботам розпізнавати об'єкти, текст, людей та інше на зображеннях.

Алгоритми маркетингового аналізу та рекомендацій. Ці алгоритми допомагають телеграм-ботам аналізувати поведінку користувачів та надавати рекомендації щодо послуг, продуктів чи контенту, які можуть бути цікавими для конкретного користувача.

Алгоритми обробки даних та бази даних. Ці алгоритми дозволяють телеграм-ботам ефективно зберігати, оновлювати та використовувати великі

обсяги даних. Вони грають важливу роль у забезпеченні швидкості та надійності роботи ботів.

1.5. Можливості та приклади впровадження телеграм-ботів на базі штучного інтелекту в різних сферах діяльності

Телеграм-боти на базі штучного інтелекту знайшли широке застосування в різних галузях діяльності, сприяючи автоматизації процесів та поліпшенню обслуговування клієнтів.

В бізнесі: Покращення обслуговування клієнтів. Телеграм-боти в бізнесі використовуються для автоматизації обробки запитів клієнтів та надання інформації про продукти та послуги. Приклади включають ботів для онлайн-консультацій, обробки замовлень та відстеження доставки, а також ботів для надання інформації про статус рахунку чи тарифів.

В освіті: Навчання та підтримка студентів. У сфері освіти телеграм-боти використовуються для надання навчального матеріалу, відповідей на питання студентів та навіть проведення тестів і оцінювання. Вони сприяють покращенню доступу до навчання та забезпечують індивідуалізовану підтримку для студентів.

В медицині: Консультування пацієнтів. Телеграм-боти в медицині можуть надавати користувачам інформацію про симптоми, лікування та надавати пора-ди щодо медичних питань. Вони також можуть служити для запису на прийом до лікаря, нагадування про прийом ліків та ведення медичних журналів.

У розважальній індустрії: Розважальні застосунки. У розважальній індустрії телеграм-боти використовуються для створення ігор, віртуальних асистентів та розважального контенту. Вони можуть пропонувати користувачам різноманітні розваги, від грального застосунків до новин та розважальних історій.

У фінансовій галузі: Моніторинг та фінансовий аналіз. У фінансовій галузі телеграм-боти можуть надавати користувачам інформацію про стан їхніх фінансів, бути інструментами для моніторингу ринків та навіть виконувати операції з фінансами через інтеграцію з банківськими системами.

В інших галузях: Персоналізовані рішення. Телеграм-боти можуть бути використані в різних інших галузях, включаючи галузі послуг, технологічні стартапи, громадську діяльність та багато інших. Вони надають можливість створювати персоналізовані рішення для користувачів та автоматизувати рутинні завдання.

1.6. Аналіз впливу телеграм-ботів на процес спілкування в онлайн-середовищах: переваги та недоліки

Впровадження телеграм-ботів у онлайн-середовища суттєво вплинуло на спосіб, яким люди спілкуються та виконують різноманітні завдання в цифровому просторі.

Переваги використання телеграм-ботів:

− 24/7 Доступність: Телеграм-боти доступні цілодобово, що дозволяє користувачам отримувати відповіді та послуги у будь-який час, навіть під час не-обхідності негайного вирішення проблеми;

− Швидкість та Ефективність: Боти можуть автоматизувати багато рутинних завдань, що допомагає ефективно обробляти запити користувачів та змен-шити час очікування;

− Персоналізація: Телеграм-боти можуть адаптуватися до потреб кожного користувача, надаючи персоналізовані поради, рекомендації та інформацію;

− Масштабованість: Боти можуть обслуговувати велику кількість користувачів одночасно без втрати якості обслуговування;

− Зручність: Використання ботів може бути більш зручним для користувачів, оскільки вони можуть спілкуватися з ботом в звичному месенджері.

Недоліки використання телеграм-ботів:

1. Брак Глибокого Розуміння: Незважаючи на значний прогрес у сфері обробки природної мови (NLP), багато телеграм-ботів все ще мають обмежену здатність до глибокого розуміння складних запитів та контексту.

2. Відсутність Емоційного Спілкування: Боти не здатні до емоційного спілкування та сприйняття емоцій користувачів, що робить їх менш придатними для деяких видів спілкування.

3. Загроза Приватності: Використання телеграм-ботів може вимагати передачі особистих даних, що може підвищити ризик порушення приватності користувачів.

4. Залежність від Технологій: Ефективність телеграм-ботів залежить від розвитку технологій штучного інтелекту та обробки даних.

Телеграм-боти вплинули на спосіб, яким ми спілкуємося та виконуємо завдання в онлайн-середовищах, надаючи численні переваги, такі як доступність цілодобово, швидкість, персоналізація та інші. Однак вони також мають недоліки, такі як обмежений рівень розуміння та відсутність емоційного спілкування. Для ефективного використання телеграм-ботів важливо збалансувати їхні переваги та недоліки та враховувати потреби користувачів та конкретні сфери використання.

1.7. Технології ChatGPT: Розвиток та Перспективи. Історія розвитку ChatGPT

Історія розвитку технології ChatGPT є захоплюючою подорожжю в світі штучного інтелекту та обробки природної мови. Започнемо з початкового витоку та прослідкуємо ключові моменти, що привели до створення системи ChatGPT.

Початок шляху: GPT-1. Шлях розвитку ChatGPT віддаляє свій старт з моменту створення моделі GPT-1 (Generative Pre-trained Transformer) в 2018 році компанією OpenAI. Ця модель була однією з перших, яка використовувала архітектуру Transformer та навчалася на великій кількості тексту з Інтернету. Однак GPT-1 була відносно обмеженою у своїй здатності генерувати природну мову та розуміти контекст.

Пошук кращого рішення: GPT-2. У 2019 році OpenAI представила модель GPT-2, яка вразила світ своєю вражаючою мовною здатністю. GPT-2

мала 1,5 мільярда параметрів, що робило її найбільшою моделлю на той момент. Вона змогла генерувати текст, який був важко відрізнити від тексту, написаного людиною. Це відкрило безліч застосувань, включаючи створення автоматизованих текстів, розважальний контент та різноманітні додатки для обробки мови.

Від величезної моделі до ChatGPT: GPT-3. Зі створенням GPT-3 в 2020 році відбувся значний прорив в розвитку технології. GPT-3 мала неймовірний розмір - 175 мільярдів параметрів - що робило її найбільшою та найздатнішою моделлю на той момент. Основна особливість GPT-3 полягала в її здатності гене-рувати текст на великому спектрі мов та мовленнєвих стилів. Це сприяло розви-тку багатьох текстових додатків, зокрема ChatGPT.

Поява ChatGPT: Застосування GPT-3 в чат-ботах. ChatGPT стала логічним кроком у розвитку GPT-3. Ця технологія використовує базову модель GPT-3 для створення текстових інтерфейсів для спілкування з користувачами. Перші ітерації ChatGPT дозволили створювати чат-ботів, які можуть відповідати на запити та виконувати завдання, засновані на текстовому вводі.

Пошук оптимального використання: Підвищення ефективності. Упродовж останніх років дослідники та інженери активно працювали над підвищенням ефективності ChatGPT. Це включає в себе оптимізацію для конкретних галузей, підвищення здатності розуміти контекст та розвиток механізмів фільтрації контенту для захисту від недоречних або шкідливих відповідей.

Історія розвитку ChatGPT - це історія становлення потужного інструменту у сфері обробки природної мови та штучного інтелекту. Від попередніх версій GPT до створення ChatGPT було зроблено значні кроки у напрямку реалізації більш природної мови та розуміння контексту. Ця технологія має великий потенціал у різних сферах, включаючи бізнес, освіту та медицину, і її подальший розвиток обіцяє надзвичайно цікаві перспективи.

1.8. Принципи роботи ChatGPT

Архітектура глибокого навчання

Глибоке навчання є підгалуззю машинного навчання, яка стала особливо популярною завдяки своїм вражаючим досягненням в різних сферах, таких як візуальне розпізнавання, обробка природної мови, автономні автомобілі, медична діагностика і багато інших. Глибоке навчання використовує нейронні мережі, які мають багато шарів (глибокість), що дозволяє їм вивчати складні залежності у вхідних даних та робити передбачення або приймати рішення.

Основні компоненти архітектури глибокого навчання:

1. Нейронні мережі: Нейронні мережі є основною будівельною одиницею глибокого навчання. Кожен нейрон приймає вхідні дані, обчислює вагову суму цих даних та передає результат через функцію активації. Нейрони організовані у різні шари, включаючи вхідний шар, внутрішні (приховані) шари та вихідний шар.

2. Глибина мережі: Однією з ключових характеристик глибокого навчання є наявність багатошарових мереж, де кожен шар представляє собою набір нейронів, які обробляють вхідні дані і надсилають їх до наступного шару. Глибока мережа має здатність розпізнавати та вивчати більш складні та абстрактні функції, що робить її ефективною у завданнях, таких як розпізнавання об'єктів на зображеннях.

3. Функції активації: Функції активації визначають активність нейронів у мережі. Популярні функції активації включають ReLU (Rectified Linear Activation), сигмоїду та тангенс гіперболічний. Вони регулюють передачу сигналу між нейронами та дозволяють мережі навчатися.

4. Зворотне поширення помилки: Це ключовий метод тренування глибоких мереж. Під час тренування мережа порівнює вихідні дані з очікуваними результатами і коригує ваги нейронів, щоб мінімізувати помилки. Зворотне поширення помилки допомагає мережі навчатися та покращувати свою точність.

5. Функція втрати: Функція втрати визначає різницю між виходом мережі та очікуваними результатами. Одна з популярних функцій втрати - середньоквадратична помилка (MSE) для завдань регресії і перехресна ентропія (Cross-Entropy) для завдань класифікації.

Глибоке навчання використовується в різних галузях, і його потенціал лише посилюється завдяки розвитку архітектур та алгоритмів. Таким чином, воно стає важливим інструментом для розв'язання складних завдань та вирішення реальних проблем у сучасному світі.

Методи тренування та навчання

Методи тренування та навчання глибоких нейронних мереж в архітектурі глибокого навчання включають в себе різні стратегії та алгоритми, які дозволяють мережам вчитися з великим обсягом даних та покращувати свою продуктивність.

Зворотне поширення помилки (Backpropagation): Цей метод є одним із найпоширеніших та основних для тренування нейронних мереж. Він базується на ідеї коригування ваг нейронів відповідно до градієнту функції втрати. Мережа порівнює свій вихід із правильними відповідями, і градієнти обчислюються з кінця до початку (зворотні градієнти), щоб оновити ваги таким чином, щоб зменшити помилку.

Методи оптимізації: Для швидкого та ефективного тренування глибоких мереж використовуються різні методи оптимізації, такі як стохастичний градієнтний спуск (SGD), адам, RMSprop, та інші. Ці методи дозволяють швидше знаходити оптимальні ваги мережі.

Регуляризація: Глибокі мережі, особливо з великою кількістю параметрів, схильні до перенавчання. Регуляризація, така як L1 та L2 регуляризація, допомагає зменшити цей ефект шляхом додавання штрафу за великі значення ваг.

Підвищення швидкості навчання (Learning Rate Scheduling): Швидкість навчання (learning rate) грає важливу роль у тренуванні мережі. Її можна

поступово зменшувати під час навчання, що допомагає у збільшенні стійкості та точності моделі.

Функції активації: Вибір функцій активації, таких як ReLU, сигмоїда, тангенс гіперболічний та інші, може впливати на швидкість та якість навчання мережі.

Пакетне навчання (Batch Training): Замість оновлення ваг на кожному прикладі, моделі можуть бути треновані на пакетах (батчах) даних. Це сприяє більш стабільному та швидкому навчанню.

Перенавчання (Overfitting) та розріджувачі (Dropout): Для попередження перенавчання, що може виникнути при тренуванні на великій кількості даних, використовують методи, такі як dropout, які випадково вимикають частину нейронів у процесі навчання.

Ці методи тренування та навчання використовуються для побудови та налаштування глибоких мереж у великих і складних завданнях машинного навчання, дозволяючи моделям досягти високої точності та ефективності в різних доменів, включаючи візуальне розпізнавання, обробку природної мови, рекомендації та багато інших.

Генерація тексту та розуміння запитів

Глибоке навчання має велике значення для генерації тексту та розуміння запитів у різних сферах застосування. Генерація тексту виконується за допомогою наступних систем:

Рекурентні нейронні мережі (RNN) та LSTM: Ці архітектури мереж дозволяють моделям генерувати послідовний текст, працюючи зі зв'язками між словами або символами в тексті. Вони широко використовуються для завдань, таких як машинний переклад та генерація тексту.

Згорткові нейронні мережі (CNN): Ці мережі використовуються для генерації тексту, зокрема для створення більш деталізованих описів зображень.

Трансформери (Transformer): Моделі на основі трансформерів, такі як GPT (Generative Pre-trained Transformer), стали дуже популярними для генерації тексту. Вони можуть генерувати послідовний текст, будуючи його по одному слову (або токену) за раз. Розуміння запитів виконується наступним чином:

Обробка природної мови (NLP): Моделі глибокого навчання, особливо трансформери, дозволяють розуміти та відповідати на запити користувачів. Наприклад, системи підтримки клієнтів можуть використовувати NLP для автоматичного розуміння запитів і надання рекомендацій або відповідей.

Машинний переклад: Глибоке навчання застосовується для автоматичного перекладу мов. Моделі, такі як Google Translate, використовують нейронні мережі для генерації перекладів.

Автоматичне аналіз тексту: Моделі глибокого навчання можуть аналізувати текст, щоб витягнути інформацію, таку як ключові слова, емоційний тон, тема та інше. Це використовується в моніторингу соціальних медіа, аналізі відгуків та інших сферах.

Розпізнавання мови: Глибоке навчання допомагає розпізнавати та розуміти мову в мовленні та тексті. Це застосовується у системах голосового управління, в системах розпізнавання голосу та інших.

Загальний вплив глибокого навчання на генерацію тексту та розуміння запитів полягає в тому, що воно дозволяє створювати більш точні та ефективні системи, які здатні автоматизувати багато рутинних завдань, пов'язаних з обробкою та аналізом текстової інформації. Це розширює можливості вирішення завдань у різних сферах, від інтернет-сервісів до медицини та бізнесу.

1.9. Вплив ChatGPT на сучасне спілкування

Використання у чат-ботах та веб-сервісах

Глибоке навчання і його архітектури мають широке застосування у чат-ботах та веб-сервісах. Чат-боти використовують моделі глибокого навчання

для створення автоматичних відповідей на запити користувачів. Вони можуть аналізувати текст запиту та генерувати відповідь, яка краще відповідає контексту та потребам користувача.

Також вони використовуються для надання підтримки клієнтам, відповіді на запити та розв'язання рутинних питань. Вони можуть взаємодіяти з користувачами через текстові повідомлення, голосовий ввід або інші інтерфейси.

Чат-боти можуть використовувати моделі рекомендацій на основі глибокого навчання, щоб надавати користувачам персоналізовані поради, продукти або послуги. Моделі глибокого навчання можуть аналізувати текстові повідомлення користувачів для визначення їхнього настрою та емоційного стану. Це допомагає ботам реагувати адекватно на користувачів.

Глибоке навчання використовується для аналізу вмісту на веб-сайтах та підбору релевантного контенту для користувачів. Це може включати рекомендації статей, відео, товарів та іншого контенту. Моделі глибокого навчання використовуються для автоматичної індексації та класифікації вмісту на веб-сервісах. Наприклад, вони можуть автоматично розпізнавати категорії новин, тегувати фотографії або аналізувати відгуки користувачів.

Веб-сервіси використовують алгоритми ранжування на основі глибокого навчання для покращення результатів пошуку та надання рекомендацій. Це допомагає користувачам знаходити потрібну інформацію швидше та з більшою точністю. Моделі глибокого навчання можуть використовуватися для аналізу даних та передбачення трендів на веб-сервісах. Наприклад, вони можуть передбачати популярність певних товарів або прогнозувати поведінку користувачів.

Застосування глибокого навчання у чат-ботах та веб-сервісах дозволяє створювати більш інтелектуальні та ефективні системи, які можуть краще взаємодіяти з користувачами та надавати персоналізовану інформацію та послуги.

Вплив на медіа та журналістику

Глибоке навчання дозволяє створювати персоналізований контент для читачів та глядачів на основі їхніх індивідуальних інтересів та поведінки в мережі. Це може включати індивідуальні рекомендації статей, новинних заголовків, телепрограм та іншого контенту.

Журналісти та редактори використовують глибоке навчання для аналізу даних та відстеження популярних тем і тем, що генерують обговорення. Це допомагає визначати та прогнозувати тенденції в інформаційному просторі.

Глибоке навчання допомагає автоматизувати обробку мультимедійного контенту, такого як відео та зображення. Моделі можуть виявляти об'єкти на фотографіях, визначати рухи відомих осіб на відео та інше.

Журналісти та медіа можуть використовувати авторські системи та чат-ботів на основі глибокого навчання для автоматизованого створення та публікації контенту, відчуття новин та взаємодії з читачами.

Глибоке навчання допомагає створювати інтерактивний та іммерсивний контент для медіа, включаючи віртуальну реальність та розширену реальність. Це може включати в себе ігри, інтерактивні візуалізації новин та інше.

Глибоке навчання допомагає створювати механізми автоматизованого пошуку та агрегації інформації для журналістів. Це полегшує процес пошуку даних та джерел для створення матеріалів.

Загалом, глибоке навчання реформує та змінює спосіб, яким медіа створює, редагує, поширює та споживає інформацію. Воно допомагає журналістам і медійним компаніям розширювати можливості у сфері збору, аналізу та поширення новин та контенту, а також відкриває нові можливості для інтерактивності та залучення аудиторії. Однак важливо зберігати етичні та редакційні стандарти, щоб забезпечити якість та об'єктивність інформації, яку споживачі отримують.

Етичні питання та виклики

Застосування глибокого навчання в медіа та журналістиці породжує численні етичні питання та виклики, які потребують уважного розгляду та розв'язання.

Достовірність та дезінформація: Моделі глибокого навчання можуть легко створювати фейкові новини, текстовий контент та відео. Це створює ризик поширення дезінформації. Журналісти повинні бути відповідальними за перевірку та достовірність інформації, а також розробляти та використовувати інструменти для виявлення фейків.

Приватність і захист даних: Збір та аналіз великих обсягів даних для тренування глибоких моделей може порушувати приватність користувачів. Журналісти та медіа повинні дотримуватися найвищих стандартів захисту даних та враховувати правила GDPR та інших регуляторів.

Байєсівська фільтрація та ехо-камера: Алгоритми рекомендацій на основі глибокого навчання можуть створювати "бульбашки" фільтрації, де користувачі бачать лише інформацію, що відповідає їхнім поглядам і переконанням. Це може погіршувати політичний та культурний діалог. Журналісти повинні допомагати розширювати горизонти інформації та створювати інтерактивні платформи для різних поглядів.

Автоматизація та робочі місця: Використання авторських систем та ботів може вплинути на робочі місця журналістів. З одного боку, це може полегшити рутинні завдання, але з іншого боку, це може призвести до втрати робочих місць. Важливо розглядати соціальні та економічні наслідки автоматизації.

Біаси та адекватність: Моделі глибокого навчання можуть вивчати біаси з навколишнього світу та відображати їх у своїй роботі. Це може викликати проблеми з адекватністю та справедливістю інформації. Журналісти повинні активно відстежувати та коригувати такі біаси.

Етика роботи з даними: Збір та використання великих обсягів даних для глибокого навчання повинні відбуватися в межах етичних норм. Журналісти

повинні дотримуватися правил щодо джерел даних, конфіденційності та безпеки.

Вплив на професію журналіста: Використання глибокого навчання та автоматизації може змінити професійні обов'язки журналістів. Важливо надавати можливість журналістам навчатися та адаптуватися до нових технологій.

Реклама та монетизація: Глибоке навчання використовується для персоналізації реклами, що може впливати на споживачів. Журналісти повинні розглядати етичні аспекти реклами та монетизації для збереження незалежності та об'єктивності інформації.

Всі ці етичні питання та виклики вимагають уважної уваги та обговорення серед журналістів, медіа компаній, а також владних та регуляторних організацій. Розробка етичних стандартів та рекомендацій є важливою частиною забезпечення відповідального та об'єктивного використання глибокого навчання у медіа та журналістиці.

Перспективи розвитку та застосування

Глибоке навчання має значний потенціал у сферах медицини та освіти, відкриваючи нові можливості для покращення діагностики, лікування, навчання та навчальних процесів.

Діагностика та обробка зображень: Моделі в медицині можуть аналізувати зображення, такі як рентгенограми, МРТ-знімки та знімки з СТ-сканерів, для виявлення патологій та хвороб. Вони можуть допомагати лікарям точніше та швидше ставити діагнози. Прогнозування захворювань: Аналітичні моделі можуть передбачати ризик розвитку захворювань, таких як рак, діабет, серцево-судинні захворювання і багато інших. Це допомагає вчасно вживати профілактичні заходи. Оптимізація лікування: Моделі в медицині аналізують клінічні дані та інформацію про пацієнтів для визначення найкращих методів лікування та доз ліків. Роботи в хірургії:

Роботизовані системи допомагають хірургам виконувати операції з більшою точністю та мінімальними пошкодженнями.

Персоналізована медицина: Моделі враховують генетичні дані та інші фактори здоров'я пацієнтів для створення індивідуалізованих планів лікування.

Сфера освіти: Персоналізоване навчання: Аналітичні моделі створюють індивідуалізовані програми навчання для кожного учня на основі його навчальних потреб та стилю навчання. Автоматизована оцінка та звітування: Моделі оцінюють роботу учнів та генерують звіти для вчителів та батьків, спрощуючи процес відстеження прогресу. Навчання мовам та переклад: Мовні моделі допомагають учням вивчати іноземні мови та автоматично перекладати тексти. Симуляція та віртуальна реальність: Моделі створюють навчальні симулятори та віртуальні навчальні середовища, що полегшують засвоєння складного матеріалу. Автоматизація адміністративних завдань: Аналітичні моделі автоматизують адміністративні завдання, такі як розклади та документація. Розвиток навичок: Вчителі використовують моделі для створення програм та додатків, які допомагають учням розвивати навички в різних областях.

Ці можливості показують, як аналітика та моделі можуть покращувати якість медичних послуг та освітніх процесів, роблячи їх більш доступними та індивідуалізованими.

Розширення мовної підтримки та культурних адаптацій

Багатомовність: Моделі можуть бути навчені працювати з різними мовами, що сприяє міжнародному спілкуванню та доступу до інформації для глобальної аудиторії.

Локалізація та персоналізація: Системи можуть адаптуватися до конкретних культурних особливостей та індивідуальних уподобань користувачів, роблячи взаємодію більш природною та зрозумілою.

Мовний аналіз: Глибоке навчання допомагає розуміти контекст та смисл мови, враховуючи варіанти вимови, акценти та мовні нюанси.

Автоматичний переклад: Мовні моделі здатні автоматично перекладати тексти та аудіо мовами, сприяючи міжнародному спілкуванню та доступу до інформації в різних мовах.

Підтримка мовних меншин: Моделі допомагають забезпечити доступ до інформації та послуг для мовних меншин та груп, які раніше мали обмежений доступ.

Адаптація до локальних реалій: Моделі можуть розуміти локальні терміни, вирази та культурні особливості, що робить інтеракцію більш адаптованою до конкретного регіону.

Сприяння багатокультурному розумінню*: Розширення мовної підтримки допомагає сприяти взаєморозумінню та діалогу між різними культурами та мовами.

Ці можливості демонструють, як глибоке навчання допомагає зробити інтернет та інформацію більш доступними та доступними для різних мов та культур, сприяючи глобальному спілкуванню та розумінню.

ВИСНОВКИ ДО РОЗДІЛУ 1

В даному розділі було розглянуто та проаналізувано кілька важливих аспектів сучасної інформаційної та технологічної сфери, включаючи технології ChatGPT, розвиток та перспективи їх використання.

Перш за все, була досліджена роль та можливості технологій ChatGPT. Ця розвинена система глибокого навчання дозволяє створювати інтелектуальні програми та ботів, здатних розуміти та генерувати людську мову. Розвиток ChatGPT відбувався з кожним поколінням, і його застосування варіюється від генерації тексту до покращення спілкування з користувачами.

Далі, розглянута роль Телеграм-ботів на базі штучного інтелекту. Такі боти стали необхідним інструментом в бізнесі, освіті, медицині, розважальній індустрії та багатьох інших галузях. Вони забезпечують покращену підтримку клієнтів, надають інтерактивне навчання, консультують пацієнтів та забезпечують нові можливості для спілкування.

Завдяки цьому дослідженню, ми можемо визначити значущість та перспективи використання технологій ChatGPT та Телеграм-ботів на базі штучного інтелекту в нашому сучасному світі. Ці інновації відкривають нові можливості для покращення комунікації, розвитку бізнесу та навчання, і вони залишаються актуальними та обіцяючими напрямками для подальших досліджень та розробок.

Таким чином, технології ChatGPT, Телеграм-боти на базі штучного інтелекту та фреймворки для їх розробки відіграють важливу роль у сучасному цифровому світі, і їхнє використання продовжує розширюватися, створюючи нові можливості та перспективи для інновацій та покращення.

Технологія ChatGPT, розроблена на основі глибокого навчання та нейронних мереж, суттєво розширила можливості природної мови обробки, дозволяючи створювати різноманітні застосування у сфері комунікації та інтерактивного спілкування. За роки розвитку ця технологія зазнала численних покращень і вдосконалень, і її потенціал у багатьох галузях, включаючи медицину, освіту, бізнес та розваги, є величезним.

Телеграм-боти, які базуються на штучному інтелекті, стали значущими інструментами для автоматизації та покращення спілкування в онлайн-середовищах. Вони знаходять застосування в різних сферах, від обслуговування клієнтів у бізнесі до підтримки студентів у освіті. Можливості розвитку телеграм-ботів на базі штучного інтелекту ще далеко не вичерпані, і ця технологія обіцяє надавати більше інновацій та покращень у майбутньому.

За результатами дослідження можна стверджувати, що технології ChatGPT та телеграм-боти мають великий потенціал для трансформації

способу, яким ми спілкуємося та взаємодіємо з інформацією. Проте важливо враховувати етичні аспекти та забезпечувати безпеку в їхньому використанні.

Майбутнє цих технологій залежить від творчості, досліджень та впровадження в практику, і вони можуть стати важливими компонентами нашої цифрової епохи, допомагаючи нам досягти нових вершин в спілкуванні та інтеракції з технологіями.

РОЗДІЛ 2
ЗАСОБИ ПРОЕКТУВАННЯ ТЕЛЕГРАМ-БОТА

2.1.Опис предметної області

Тема кваліфікаційної роботи зосереджується на створенні інтелектуального Телеграм Бота, який базується на передовій технології обробки природної мови — ChatGPT-3.5. Ця технологія, розроблена компанією OpenAI, представляє собою потужну модель штучного інтелекту, здатну генерувати людино-подібні текстові відповіді та виконувати завдання на рівні людського розуміння. Предметна область дослідження включає:

1. Розгляд особливостей та можливостей штучного інтелекту, які привнесла технологія ChatGPT-3.5.
2. Вивчення методів взаємодії з API ChatGPT-3.5 для отримання текстових відповідей.
3. Аналіз потреб користувачів та визначення функціональних вимог до бота.
4. Розробка модулів для розпізнавання облич, побудови масок, ідентифікації особливостей та інших функцій.
5. Вивчення аспектів інтерфейсного дизайну для створення зручного та естетичного інтерфейсу бота.
6. Аналіз та вибір мови програмування JavaScript для реалізації бота у середовищі Telegram.
7. Розгляд можливостей використання розробленого бота у різних сценаріях, включаючи кібербезпеку, розваги, та інші області.

2.2. Опис вимог телеграм-бота

Дане дослідження спрямоване на розробку інноваційного рішення у сфері обробки природної мови, розвиток функціональних та нефункціональних можливостей телеграм-бота та вивчення можливостей практичного використання отриманих результатів у різних сферах.

Застосунок для демонстрації роботи технології обробки природної мови повинен відповідати наступним функціональним вимогам:

1. Забезпечити взаємодію бота з платформою Telegram для прийому та надсилання повідомлень.

2. Реалізувати систему аутентифікації для користувачів бота та обробку авторизованих запитань.

3. Розробити функцію обробки запитань користувачів та генерації відповідей з використанням технології ChatGPT.

4. Забезпечити можливість ведення контекстних розмов та зберігання історії діалогу для кращого розуміння контексту.

5. Додати можливість вибору мови взаємодії з ботом та генерації відповідей на обраній мові.

6. Розробити систему команд та інтерактивних сценаріїв для більш гнучкого та структурованого взаємодії з ботом.

7. Впровадити функцію пошуку інформації на основі запитань користувачів та надання відповідей.

8. Реалізувати систему фільтрації та модерації контенту для уникнення нецензурних виразів та відсутності необгрунтованої інформації.

9. Додати можливість користувачам налаштовувати параметри роботи бота та персоналізувати його відповіді.

10. Забезпечити користувачів оповіщеннями та повідомленнями від бота про події чи важливу інформацію.

У результаті аналізу функціональних вимог та огляду аналогів, сформовано наступні нефункціональні вимоги:

– застосунок повинен мати зручний та мінімалістичний інтерфейс;

– застосунок повинен бути розроблений на мові програмування JavaScript;

2.3. Мова програмування JavaScript

Історія JavaScript

Перші передумови для появи цієї мови з'явилися в 1992 році, коли була розпочата розробка вбудованої скриптової мови Cmm (С мінус мінус). Пізніше його перейменували в ScriptEase, оскільки назва С мінус мінус мала негативний відтінок. Загалом. До того, як мова отримала сучасну назву, її назва змінювалася ще кілька разів.

У 1995 році Брендан Айх отримав завдання ввести мову програмування в браузер Netscape. Спочатку мова називалася Mocha, а потім LiveScript. Нарешті він отримав свою сучасну назву - JavaScript. Тут розробники пішли на хитрість. У той час, коли вони вдосконалювали LiveScript, мова Java була досить популярною. Щоб залучити більше розробників до роботи з новою мовою, було вирішено використовувати в її назві Java. Кінцевим результатом є JavaScript.

Остання версія мови ES6 була випущена в 2015 р. З її появою мова отримала друге життя. З'явилися нові стандарти, а також можливість роботи з константами. Сам код також змінився. Мова дотримується принципу скорочення коду з більшою функціональністю.

JavaScript

JavaScript — це кросплатформна об’єктно-орієнтована мова сценаріїв, яка є невеликою та легкою. У середовищі виконання JavaScript можна пов’язувати з об’єктами середовища виконання та забезпечувати програмний контроль над ними.

JavaScript включає стандартну бібліотеку об’єктів, таких як масив, дата і математика, а також основний набір мовних елементів, таких як оператори та керуючі конструкції. Основний JavaScript можна розширити для різних цілей, додавши до нього нові об’єкти, наприклад:

JavaScript на стороні клієнта розширює ядро мови, надаючи об’єкти для керування браузером та його об’єктною моделлю документа (DOM). Наприклад, розширення на стороні клієнта дозволяють програмі розташовувати елементи у формі HTML і обробляти події користувача, такі як кліки миші, введення форми та навігація сторінкою.

JavaScript на стороні сервера розширює основну мову, надаючи об’єкти для запуску JavaScript на сервері. Наприклад, розширення на стороні сервера дозволяє програмі підключатися до бази даних, підтримувати безперервність інформації між викликами програми або маніпулювати файлами на сервері.

JavaScript і Java

JavaScript і Java схожі в деяких аспектах, але принципово відрізняються в інших. Javascript подібний до Java, але не має статичної типізації та суворої перевірки типів. JavaScript слід переважно синтаксису Java у виразах, конвенціях про іменування та основному потокі керування конструкціями, тому його було перейменовано з LiveScript на JavaScript.

На відміну від скомпільованої системи класів Java, створеної на основі оголошень, JavaScript підтримує систему виконання, засновану на невеликій кількості типів даних, які представляють числові, логічні та рядкові значення. У JavaScript є об’єктна модель на основі прототипу замість більш загальної моделі на основі класів. Модель об’єктів на основі прототипів передбачає динамічне успадкування, тобто. те, що успадковується, може відрізнятися від окремих об’єктів. JavaScript також підтримує функції без особливих декларативних вимог. Функції можуть бути властивостями об'єктів, які виконуються як методи вільного типу.

У порівнянні з Java, JavaScript є мовою дуже вільної форми. Вам не потрібно оголошувати змінні, класи та методи. Вам не потрібно турбуватися про те, чи є методи публічними, приватними чи захищеними, і вам не потрібно впроваджувати інтерфейси. Змінні, параметри та типи, що повертаються функціями, не вводяться явно.

Java — це мова програмування на основі класів, розроблена для швидкого виконання та безпеки типів. Захист типу означає, наприклад, що ви не можете привнести цілочисельний тип до типу посилання на об'єкт або отримати доступ до приватної пам'яті, змінивши байт-код Java. Класово-орієнтована модель Java означає, що програми складаються виключно з класів та їхніх методів.

Спадкування класів і сувора типізація Java зазвичай вимагають тісно пов'язаних ієрархій об'єктів. Ці вимоги роблять програмування на Java більш складним, ніж програмування на JavaScript.

Дух JavaScript походить від невеликих динамічно типізованих мов, таких як HyperTalk і dBASE. Ці мови сценаріїв пропонують інструменти програмування для ширшої аудиторії із простішим синтаксисом, спеціалізованою вбудованою функціональністю та мінімальними вимогами до створення об’єктів.

JavaScript і специфікація ECMAScript

JavaScript стандартизовано європейською асоціацією Ecma International, діяльність якої присвячена стандартизації інформаційно-комунікаційних систем (спочатку ECMA була абревіатурою Європейської асоціації виробників комп'ютерів). Ця стандартизована версія JavaScript, яка називається ECMAScript, працює однаково у всіх програмах, які підтримують цей стандарт. Компанії можуть використовувати стандарт відкритої мови для розробки власної реалізації JavaScript. Стандарт ECMAScript задокументований у специфікації ECMA-262.

Стандарт ECMA-262 також затверджений ISO (Міжнародною організацією зі стандартизації) як ISO-16262. Ви можете знайти специфікацію на веб-сайті Ecma International. Специфікація ECMAScript не описує об'єктну модель документа (DOM), яка стандартизована Консорціумом World Wide Web (W3C). DOM визначає спосіб доступу до об’єктів HTML-документу з вашого сценарію.

Особливості JavaScript

Основні особливості цієї мови програмування:

Динамічне введення тексту. Тобто тип даних буде визначено лише тоді, коли змінній або константі буде присвоєно її значення.

Гнучка робота з функціями. У JS можна не тільки виконувати функції, а й повертати функції з функцій, передавати функції як параметри іншим функціям і призначати функції як значення змінних.

JavaScript підтримується всіма сучасними браузерами.

Об'єктно-орієнтоване програмування. Тобто це така методологія програмування, в якій вся програма представляється у вигляді набору об’єктів.

Крім того, важливою особливістю JavaScript є його розвинена інфраструктура. Сьогодні розробники можуть працювати з великою кількістю бібліотек і фреймворків (найпопулярніші з них React, Angular і Vue), декількома конструкторами, допоміжними бібліотеками (наприклад, Lodash) і генераторами статичних сайтів.

Що стосується сфери застосування, то, перш за все, мова JavaScript широко використовується в веб-розробці. І це працює в поєднанні з HTML і CSS. За допомогою JS можна створювати будь-які програми для браузера. Наприклад, кредитний калькулятор, який ви бачите на більшості веб-сайтів банків, також створений за допомогою JavaScript.

Більше того, вся візуальна частина цього калькулятора являє собою комбінацію HTML + CSS. Тобто кнопки, діаграми, повзунки є статичними елементами. За допомогою JS все анімується, а також проводяться всі основні обчислення.

Іншим прикладом є форма підписки або реєстрації. Він був створений у HTML+CSS. Однак взаємодія з сервером забезпечується саме завдяки JS. Крім того, JavaScript можна використовувати для створення наступних програм і додатків:

1. Розробка мобільного програмного забезпечення (за допомогою React Native).

2. Розробка серверних рішень за допомогою Node.js.

3. Створення настільних додатків. JS використовується, наприклад, у програмах, створених Adobe.

4. Програмування побутової техніки та платіжних терміналів.

Переваги та недоліки JS

Популярність JavaScript пояснюється багатьма факторами, включаючи велику кількість переваг, серед яких:

1. Незамінний при розробці веб-сайтів і додатків. Як зазначалося вище, JS підтримується всіма сучасними браузерами. Крім того, мова легко інтегрується з макетом і сервером.

2. Висока швидкість і продуктивність. Ця мова дозволяє частково обробляти веб-сторінки на стороні користувача. Це дозволяє витрачати менше часу на відкриття, а також зменшує навантаження на сервери.

3. Велика кількість інструментів і багата інфраструктура. Якщо спочатку багато працювали виключно з самою мовою, то сьогодні, завдяки наявності безлічі бібліотек, з’явилися зручні та доступні інструменти для всіх.

4. Відносна простота. Написання програм зазвичай займає менше часу. При цьому кількість коду також зазвичай менша в порівнянні з багатьма іншими мовами.

5. Широкі можливості для веб-сторінок. За допомогою JS ви можете оживити будь-яку веб-сторінку. Крім того, JS значно покращує зручність використання програм і веб-сайтів.

6. Відносна легкість навчання. Навіть ті, хто ніколи раніше не стикався з програмуванням, можуть почати вивчати JS. Більше того, візуалізація багатьох дій додає ентузіазму учням.

Що стосується недоліків, то до них можна віднести наступні моменти:

1. Немає можливості завантажувати та читати файли.

2. Слабкий набір тексту. Усі недоліки коду виявляються на етапі роботи програми, що в деяких випадках може бути незручно. Крім того, невідповідності, які можуть бути в коді JS, ігноруються самою мовою.

Рівень безпеки. Досить легко ввести шкідливий код у мову програмування, як-от JS.

2.4. Дослідження популярних фреймворків для створення Telegram - бота з використанням мови JavaScript

Telegraf.js: Розробка Телеграм-ботів на JavaScript

З ростом популярності месенджера Telegram багато розробників зацікавилися можливістю створення власних телеграм-ботів. Однак, розробка та управління ботами може бути викликом. Однією з найпопулярніших платформ для створення та керування телеграм-ботами на мові JavaScript є Telegraf.js.

Основи Telegraf.js

Telegraf.js - це фреймворк для створення та керування телеграм-ботами на мові JavaScript. Він базується на популярній платформі Node.js, що

дозволяє розробникам використовувати свої знання JavaScript для створення ботів. Основні переваги Telegraf.js включають:

1. Простий та Інтуїтивний API. Telegraf.js надає зручний та легкий у використанні API, що спрощує створення різноманітної функціональності для бота. Розробники можуть швидко створювати відповіді на повідомлення та обробляти події.

2. Обробка Різних Типів Повідомлень. Telegraf.js підтримує обробку різних типів повідомлень, включаючи текстові повідомлення, фотографії, аудіо та багато інших. Це дозволяє створювати багатофункціональні боти.

3. Команди та Клавіатури. Фреймворк дозволяє створювати команди, які викликають певні дії бота, а також інтерактивні клавіатури, що полегшують спілкування з користувачами.

4. Зберігання Даних. Telegraf.js надає можливість зберігати дані та стан бота за допомогою вбудованої підтримки локальної бази даних. Це дозволяє зберігати історію спілкування та іншу корисну інформацію.

5. Міжнародна Підтримка. Фреймворк дозволяє легко налаштовувати бота для роботи з різними мовами та локалізацією. Це особливо корисно для глобальних проектів.

6. Розширюваність. Telegraf.js - це розширюваний фреймворк, і ви можете легко розширювати його функціональність за допомогою плагінів та middleware. Це дозволяє додавати нові можливості боту з часом.

Telegraf.js - це потужний та зручний фреймворк для розробки та управління телеграм-ботами на JavaScript. Він надає розробникам інструменти для швидкого створення та розширення функціональності бота, а також дозволяє робити ботів більш інтерактивними та корисними для користувачів. Цей фреймворк є відмінним вибором для тих, хто бажає створити та управляти власними телеграм-ботами з легкістю.

BotUI: створення розмовних інтерфейсів з використанням JavaScript

Розмовні інтерфейси користувача (Conversational User Interfaces, CUIs) стають все популярнішими для створення привабливих та інтерактивних користувацьких досвідів. Одним із інструментів, що спрощує розробку CUI, є BotUI, бібліотека на мові JavaScript, яка дозволяє розробникам створювати інтерфейси, схожі на чат, у веб-додатках. У цій статті ми розглянемо функції та можливості BotUI та як його можна використовувати для побудови розмовних інтерфейсів.

BotUI - це відкрита бібліотека на мові JavaScript, яка надає простий та гнучкий спосіб додавати інтерфейси, схожі на чат, до ваших веб-додатків. Вона призначена для того, щоб допомогти розробникам створювати інтерактивні та динамічні розмови з користувачами.

Основні функції BotUI:

1. BotUI пропонує інтуїтивний та дружелюбний API, який дозволяє розробникам визначати розмови та взаємодії з користувачами з легкістю. Він надає методи для додавання повідомлень, кнопок та дій до потоку розмови.

2. Ви можете настроювати вигляд вашого чат-інтерфейсу за допомогою CSS, щоб він відповідав фірмовому стилю та дизайну вашого додатка. Ця гнучкість дозволяє створювати послідовність користувацького досвіду.

3. BotUI підтримує інтерактивні елементи, такі як кнопки та поля введення, що дозволяє користувачам приймати рішення та надавати введення під час розмови. Це дозволяє створювати форми, опитування та інтерактивні взаємодії.

4. Розробники можуть визначати обробники подій для взаємодій користувача, таких як натискання кнопок чи відправка форм. Це дозволяє контролювати логіку вашого додатка на основі відповідей користувача.

5. BotUI розроблено з урахуванням асинхронних операцій, що робить його підходящим для сценаріїв, де дані потрібно отримувати з сервера або зовнішніх API під час розмови.

6. Бібліотека підтримує локалізацію, що дозволяє створювати багатомовні чат-інтерфейси для вашого додатка.

7. BotUI можна інтегрувати з іншими бібліотеками та фреймворками JavaScript, що дозволяє використовувати його разом з вашим поточним

```
var botui = new BotUI('my-botui-app');

botui.message.bot({ // показати повідомлення від бота
  content: 'Привіт! Як я можу вам допомогти сьогодні?'
}).then(function () {
  return botui.action.button({ // показати запит на кнопки
    action: [
      {
        text: 'Почати',
        value: 'start'
      },
      {
        text: 'Дізнатися більше',
        value: 'learn'
      }
    ]
  });
}).then(function (res) {
  if (res.value === 'start') {
    botui.message.bot({
      content: 'Чудово! Давайте розпочнемо.'
    });
    // Обробка подальшої розмови на основі вибору користувача
  } else if (res.value === 'learn') {
    botui.message.bot({
      content: 'Ви можете дізнатися більше про наші послуги на нашому веб-сайті.
    });
  }
});
```

технологічним стеком.

Побудова Розмовних Інтерфейсів за Допомогою BotUI: Щоб почати використовувати BotUI, вам потрібно включити бібліотеку у свій проект. Після цього ви можете визначити розмову, додаючи повідомлення, кнопки та дії до інтерфейсу чату. Ось приклад простої розмови з BotUI:

Рис. 2.1. BotUI

У даному прикладі ми створюємо просту розмову, де бот вітає користувача, пропонує кнопки для вибору та реагує на вибір користувача.

BotUI можна застосовувати в різних сферах, включаючи:

1. Чат-боти для Підтримки Клієнтів: Створюйте чат-ботів, які допомагають клієнтам, надаючи відповіді на часто задавані питання або керуючи їх під час усунення неполадок.

2. Інтерактивні Форми: Створюйте розмовні форми, що збирають введення користувачів у форматі чату.

3. Рекомендації Продуктів: Розробляйте ботів, які рекомендують продукти чи послуги на основі вподобань та потреб користувачів.

4. Онбординг та Навчання: Надавайте користувачам інструкції під час проходження процесу введення до додатку або надавайте навчальні матеріали та пояснення.

5. Збір Даних: Збирайте дані від користувачів за допомогою інтерактивних розмов.

BotUI - це потужний інструмент для створення розмовних інтерфейсів користувача на мові JavaScript. Його дружелюбний API, настроюваний дизайн та підтримка інтерактивних елементів роблять його цінним додатком до вашого інструментарію для веб-розробки. Незалежно від того, чи ви створюєте чат-ботів для підтримки клієнтів, інтерактивних форм, чи систем рекомендацій продуктів, BotUI допоможе вам забезпечити захоплюючі та інтерактивні взаємодії для ваших користувачів.

Node-Telegram-Bot-API: розробка Телеграм-Ботів з використанням Node.js

Node.js є однією з найпопулярніших платформ для розробки серверних додатків, і вона також використовується для створення Телеграм-ботів завдяки бібліотеці Node-Telegram-Bot-API. У цій статті ми розглянемо, як використовувати Node-Telegram-Bot-API для створення Телеграм-ботів та які можливості вона надає розробникам.

Node-Telegram-Bot-API - це бібліотека для Node.js, яка дозволяє розробникам легко створювати та керувати Телеграм-ботами. Вона надає простий інтерфейс для спілкування з [Telegram Bot API](https://core.telegram.org/bots/api), що дозволяє розробникам створювати і налаштовувати ботів для взаємодії з користувачами у Телеграмі.

Основні Функції Node-Telegram-Bot-API :

1. Сприйняття Повідомлень. Бібліотека дозволяє боту сприймати повідомлення від користувачів, включаючи текстові повідомлення, зображення, аудіо та інші мультимедійні вміст.

2. Відправлення Повідомлень. Розробники можуть відправляти повідомлення від бота до користувачів, що дозволяє створювати різноманітні відповіді та взаємодію з користувачами.

3. Реакція на Команди. Бот може реагувати на команди, які користувачі вводять, і виконувати відповідні дії. Команди дозволяють взаємодіяти з ботом та отримувати інформацію чи послуги.

4. Клавіатури та Кнопки. Бібліотека підтримує створення клавіатур та кнопок для взаємодії з користувачами. Це дозволяє створювати інтерактивні діалоги та меню.

5. Мультимедіа та Вміст. Node-Telegram-Bot-API дозволяє ботам відправляти зображення, аудіо, відео та інший мультимедійний вміст у чатах.

6. Розсилка Повідомлень. Розробники можуть створювати розсилки повідомлень для великої кількості користувачів, що дозволяє сповіщати користувачів про важливі події або оновлення.

7. Сповіщення та Оголошення. Бібліотека дозволяє створювати

```
const TelegramBot = require('node-telegram-bot-api');
const token = 'YOUR_BOT_TOKEN';
const bot = new TelegramBot(token, { polling: true });

bot.on('message', (msg) => {
  const chatId = msg.chat.id;
  if (msg.text === '/start') {
    bot.sendMessage(chatId, 'Привіт! Я ваш Телеграм-бот. Я готовий взаємодіяти
  }
});
```

сповіщення та оголошення для користувачів, що дозволяє ботам повідомляти про новини або важливі інформаційні повідомлення.

Приклад Використання Node-Telegram-Bot-API для створення бота, який вітає користувачів та реагує на команду "/start":

Рис. 2.2. Node-Telegram-Bot-API

В цьому прикладі бот очікує команду "/start" від користувача та вітає його, відправляючи повідомлення.

Node-Telegram-Bot-API може бути використана для створення різноманітних Телеграм-ботів, таких як:

1. Боти для Клієнтської Підтримки: Створюйте ботів для надання відповідей на запитання клієнтів та вирішення їх проблем.

2. Новинні та Інформаційні Боти: Розробляйте ботів для розсилки новин, оголошень та інформаційних повідомлень.

3. Ігрові Боти: Створюйте ігрові боти для розваги та взаємодії з користувачами у чатах.

4. Боти для Автоматизації Завдань: Використовуйте ботів для автоматизації певних завдань, які можуть бути виконані через Телеграм.

5. Сповіщення та Оголошення: Використовуйте ботів для розсилки сповіщень та важливих оголошень.

Node-Telegram-Bot-API - це потужна бібліотека для розробки Телеграм-ботів з використанням Node.js. Вона надає розробникам зручний спосіб створення різноманітних ботів для взаємодії з користувачами у Телеграмі. Незалежно від того, чи ви розробляєте бота для клієнтської підтримки, інформаційного сервісу чи розваг, Node-Telegram-Bot-API допоможе вам реалізувати ваші ідеї та завдання.

2.5. Розпізнавання мовлення від Google Cloud

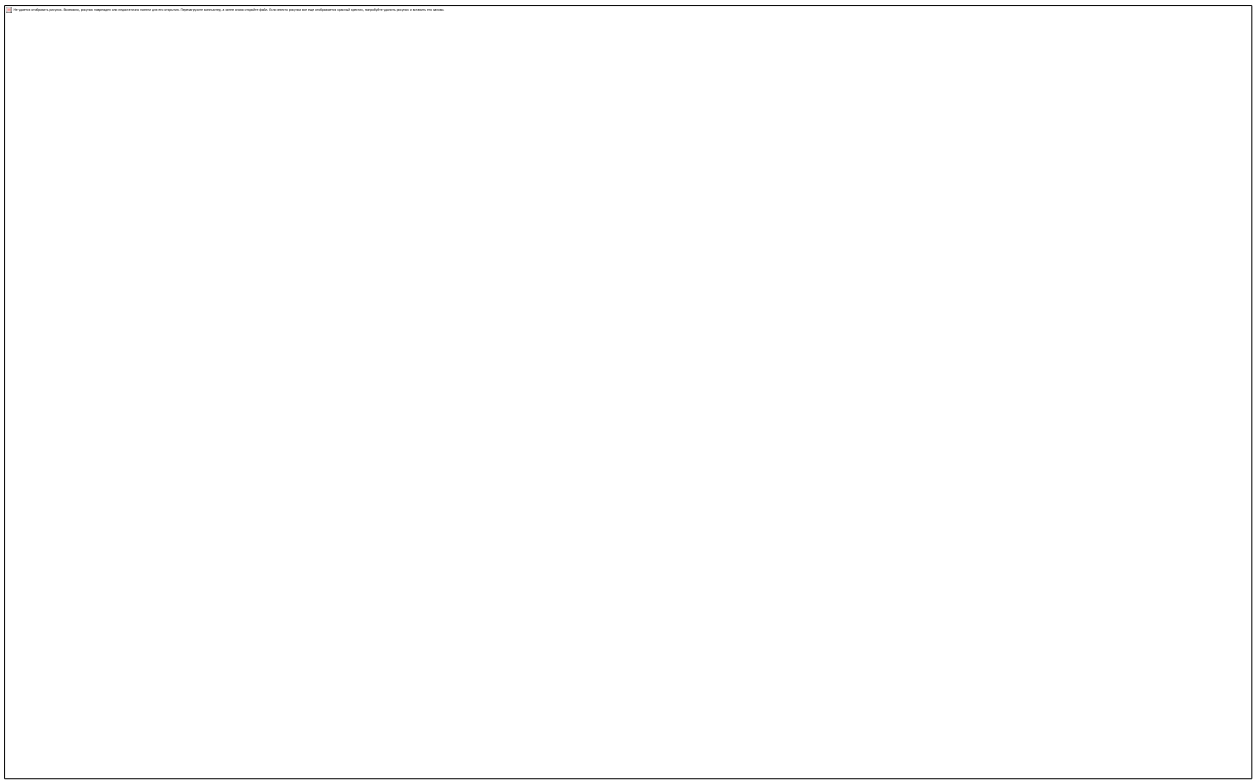


Рис. 2.3. Google Cloud.

Google Cloud Speech-To-Text — це сучасний інструмент для автоматичного перетворення та транскрипції мовлення в текст. Це корисні сервіси мовлення від Google дозволяють розробникам використовувати автовідповідачі в кол-центрах, дозволяють IoT-пристроям спілкуватися з

користувачами та перетворюють текстові повідомлення на голосовий формат.

У 2016 році Speech-to-Text, який раніше називався Cloud Speech API, був випущений. У перші роки існування Google використання API збільшувалося кожні шість місяців. Це рішення базується на алгоритмах глибокого навчання нейронної мережі Google для автоматичного розпізнавання мови (ASR).

Розгортання ASR у хмарі за допомогою API або навіть локально за допомогою локального перетворення мови на текст, яке інтегрує технології розпізнавання мовлення Google у ваше локальне рішення. Ви можете взяти під контроль свою інфраструктуру, використовуючи технологію розпізнавання мовлення з високозахищеними мовними даними, відповідаючи на необхідні правила розміщення даних і відповідаючи вимогам.

Google Speech-to-Text пропонує низку моделей машинного навчання для розпізнавання мови, адаптованих до різних типів використання. Ці моделі включають транскрипцію телефонних дзвінків, транскрипцію аудіо з відео, транскрипцію довгого або короткого контенту та багато іншого. Для кожного типу клієнти можуть вибрати модель, яка найкраще відповідає потребам їхнього бізнесу.

Виділяються деякі з найбільш поширених моделей машинного навчання для розшифрування аудіофайлів:

Latest Long: Ця модель може бути корисною для розшифровки довгої форми контенту. Якщо ваша мова недоступна, він може бути кращим інструментом для транскрипції деяких виступів або розмов. Ви навіть можете використовувати його замість моделі відео.

Latest short: як і в попередній моделі, з цією версією ви можете легко перетворити мову на текст, що містить контент, за кілька секунд.

Відео: за допомогою цієї моделі ви зможете перетворити відеокліпи на текст. Вона співпрацювала з відео з різними спікерами. Якщо ви хочете

транскрибувати високоякісний звук, записаний професійним мікрофоном, ця модель ідеально підходить. Ви можете використовувати модель за замовчуванням, наведену нижче, якщо на вашому відео є лише один спікер.

Телефонні розмови Так як ця модель очевидна, розпізнавання мовлення є чудовим варіантом для аналізу телефонних дзвінків. У цьому місці ви можете транскрибувати звук телефонної розмови.

ASR: Command & Search: ця модель перетворює короткі звуки, наприклад голосові команди, у текст. Якщо ця модель не працює на вашій мові або регіоні, ви можете скористатися моделлю Latest Short, яка також працює.

ASR: За мовчанням: ця модель транскрибує будь-яке аудіо, тому ви можете використовувати її, якщо ваш контент не відповідає попереднім характеристикам. Важливо пам’ятати, що, наприклад, якщо використовувати цю модель для транскрипції вашого відео, якість буде нижчою, ніж у випадку використання «ідеального методу».

Медична розмова: ця модель є логічною та корисною в медичному секторі. Ви можете використовувати її для розшифровки записів або розмов з медичним працівником.

Ключові переваги розпізнавання мови

1. Висока мовна адаптивність сервісу пропонує унікальні докази підвищення точності транскрипції. Крім того, ви можете використовувати класи для автоматичного перетворення вимовлених чисел на адреси, роки, валюти та багато інших речей. Наприклад, для кращого читання функція перетворення мови в тексті вкаже «23», коли хтось говорить «двадцять три».

2. Просте порівняння якості: інтерфейс цього інструменту простий для використання та розуміння. Таким чином, ви можете спробувати кілька конфігурацій, щоб оптимізувати якість транскрипції.

3. За допомогою глобального словника Google Speech-to-Text, який підтримує понад 125 мов, розпізнавання голосу в більшості країн є високопродуктивним.

4. Стійкість до шуму: цей сервіс гарантує, що в певних умовах вам не потрібне додаткове шумозаглушення. Розпізнавання мови Google Cloud може це зробити.

5. У результаті фільтрації ненормативної лексики в аудіоконтенті ви не будете турбуватися про неправильне, недоречне або непрофесійне мовлення.

6. Автоматична пунктуація Google Speech-to-Text також використовує нову нейронну мережу LSTM для автоматичної пунктуації мовних транскрипцій. В тексті модель може автоматично включати коми, знаки запитання та тире.

Інфраструктура

Робочі навантаження генеративного штучного інтелекту підвищили вимоги до сценаріїв використання машинного навчання, таких як великі мовні моделі (LLM), генеративний штучний інтелект і дифузійні моделі. Щоб поповнити портфоліо опцій GPU для навчання та визначення моделей ML, було оголошено приватну попередню версію суперкомп'ютерів A3 із графічним процесором NVIDIA H100.

Контейнери

Драйвер Cloud Storage FUSE Container Storage Interface (CSI) на GKE дозволяє програмам Kubernetes монтувати сегменти Cloud Storage як локальні файлові системи. Це спрощує процес доступу до даних, що зберігаються в хмарному сховищі Google, через семантику файлів. Драйвер доступний у загальнодоступному попередньому перегляді.

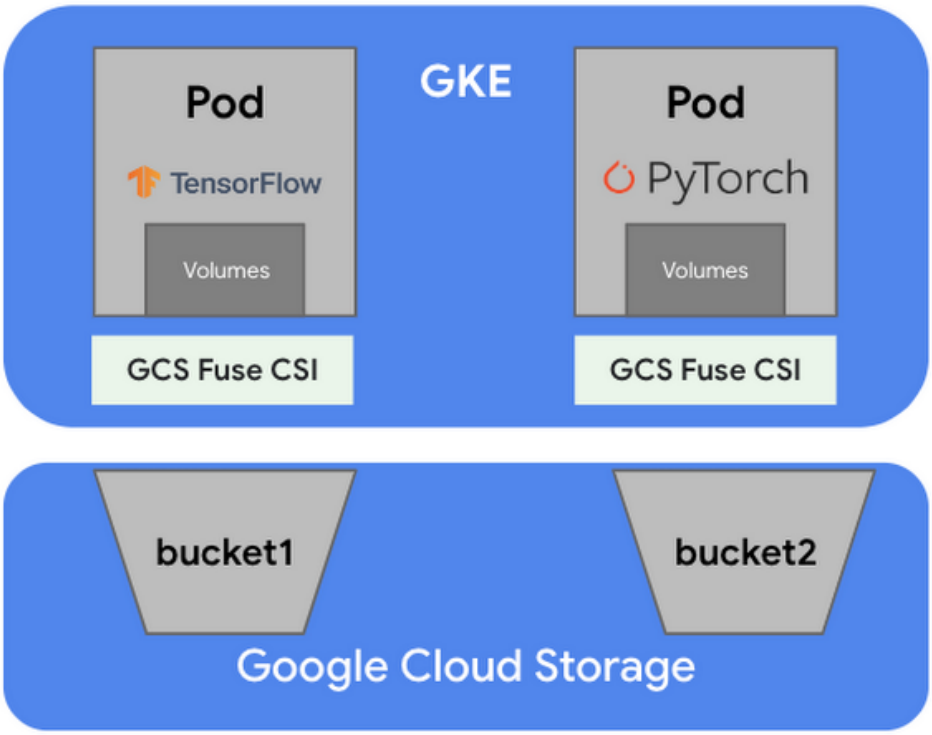


Рис. 2.4. Cloud Storage FUSE

Переглянемо серію з двох частин, яка демонструє, як створити резервну копію робочих навантажень GKE. Серія охоплює резервне копіювання для GKE, яке є рідним рішенням для резервного копіювання Kubernetes для вашого кластера GKE.

Ідентичність і безпека

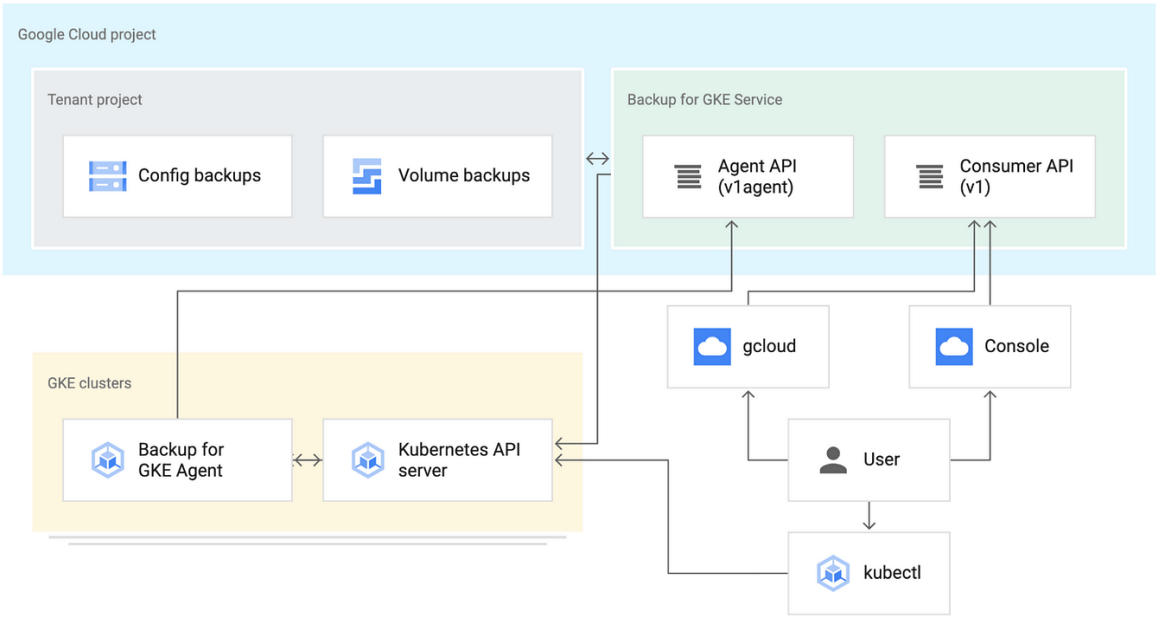
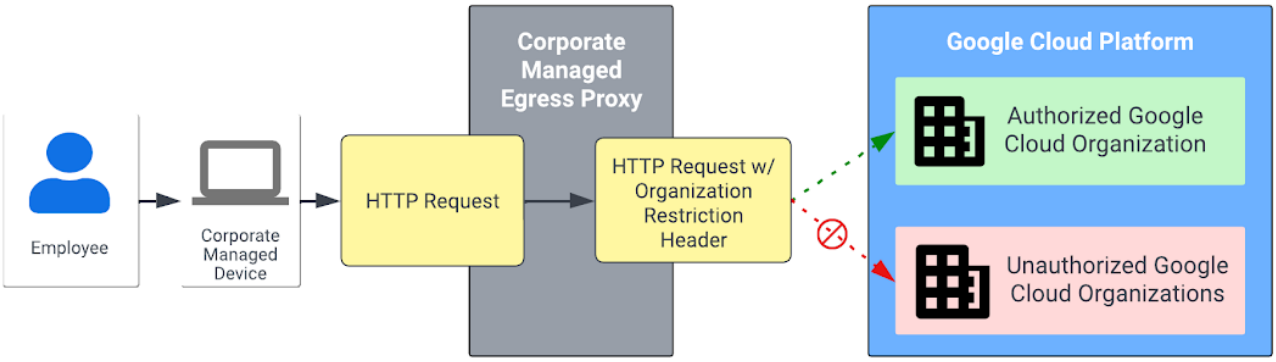


Рис. 2.5. Cloud Storage

Організаційні обмеження — це новий засіб безпеки Google Cloud, який дає змогу адміністраторам керувати обмеженням доступу користувачів лише до ресурсів і даних у спеціально авторизованих організаціях Google Cloud. Він працює, обмежуючи доступ до трафіку, що надходить із корпоративних



керованих пристроїв.

Рис. 2.5. Організаційні обмеження.

Security Command Center полегшив розуміння ідентифікації активів у великих складних середовищах. Тепер він підтримує SQL-подібні запити, щоб отримати детальну інформацію про те, де розташовані активи, їх конфігурацію та перерахування активів на основі типу ресурсу, зв'язку ресурсу, конфігурації операційної системи та метаданих організаційної політики. Існує також бібліотека запитів, яка містить попередньо визначені запити, щоб полегшити отримання інформації про активи.

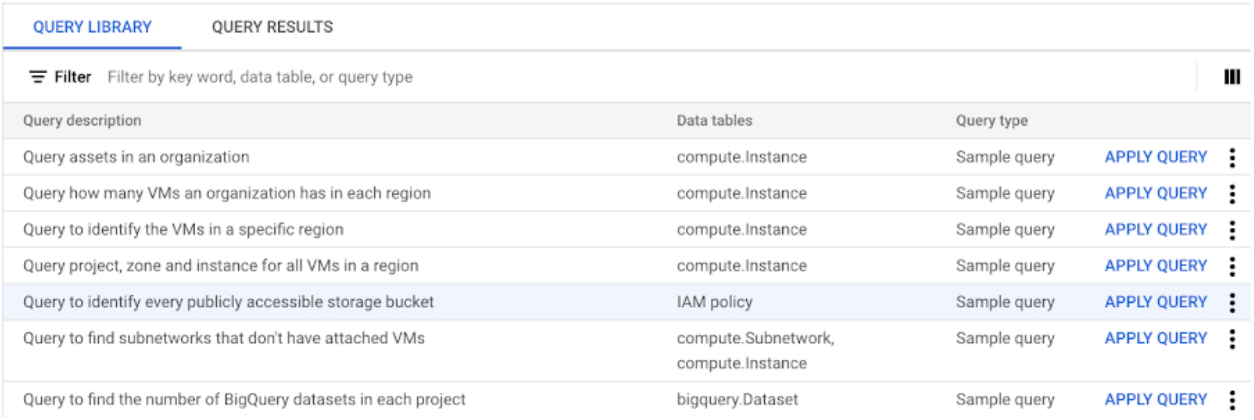
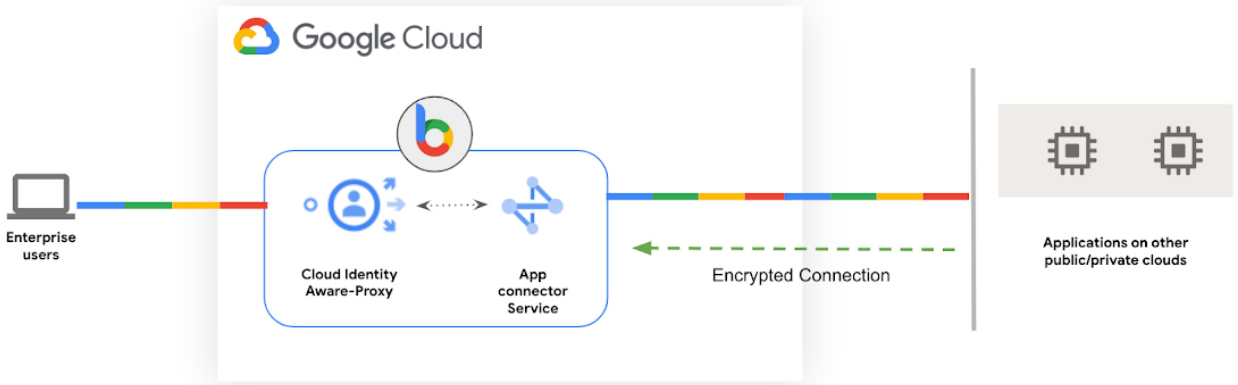


Рис. 2.6. Security Command Center.

Рис. 2.7. Google Cloude Server.

Досвід роботи з програмою BeyondCorp Enterprise App було покращено. Він має новий покроковий робочий процес для вбудованих веб-додатків і автоматичних балансувальників навантаження та серверних служб. App Connector також дозволив розширити контроль доступу Zero Trust до веб-програм, розміщених у сторонніх хмарах.



Мережа

Google Cloud Load Balancing отримав дві нові функції, які допомагають підвищити загальну надійність Cloud Load Balancer для ваших робочих навантажень:

- додаткове сегментування, вбудоване в наш Global External Load Balancer;
- використання регіональних зовнішніх балансувальників навантаження як варіант відновлення після відмови.

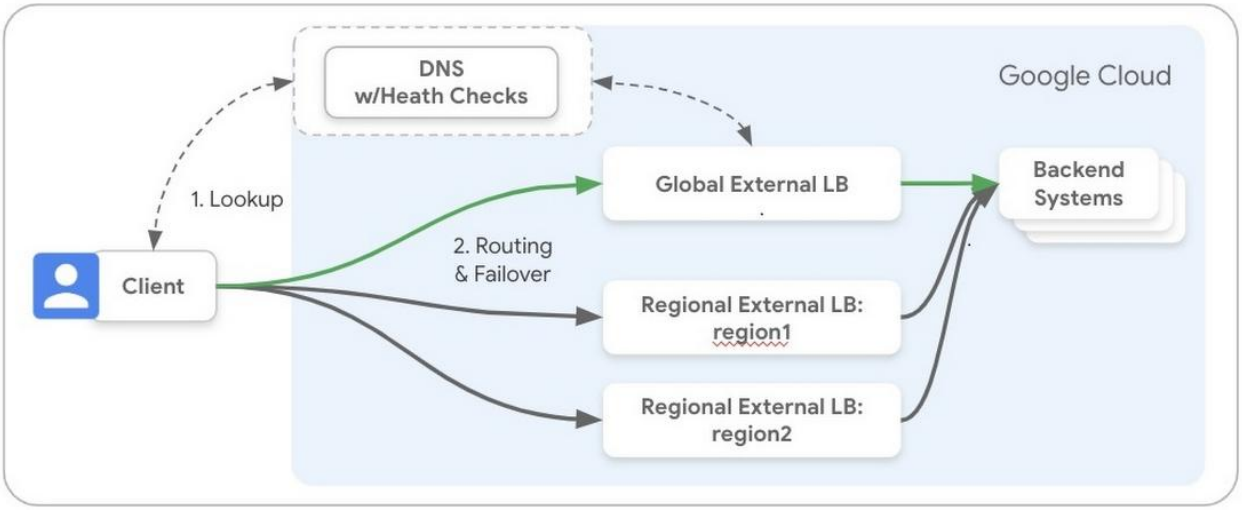


Рис. 2.8. Google Cloud Load Balancing

Посібники з еталонної архітектури для мережевих шаблонів містять нову публікацію, в якій висвітлюються ключові мережеві шаблони, які слід застосовувати, коли справа доходить до доставки додатків для Інтернету.

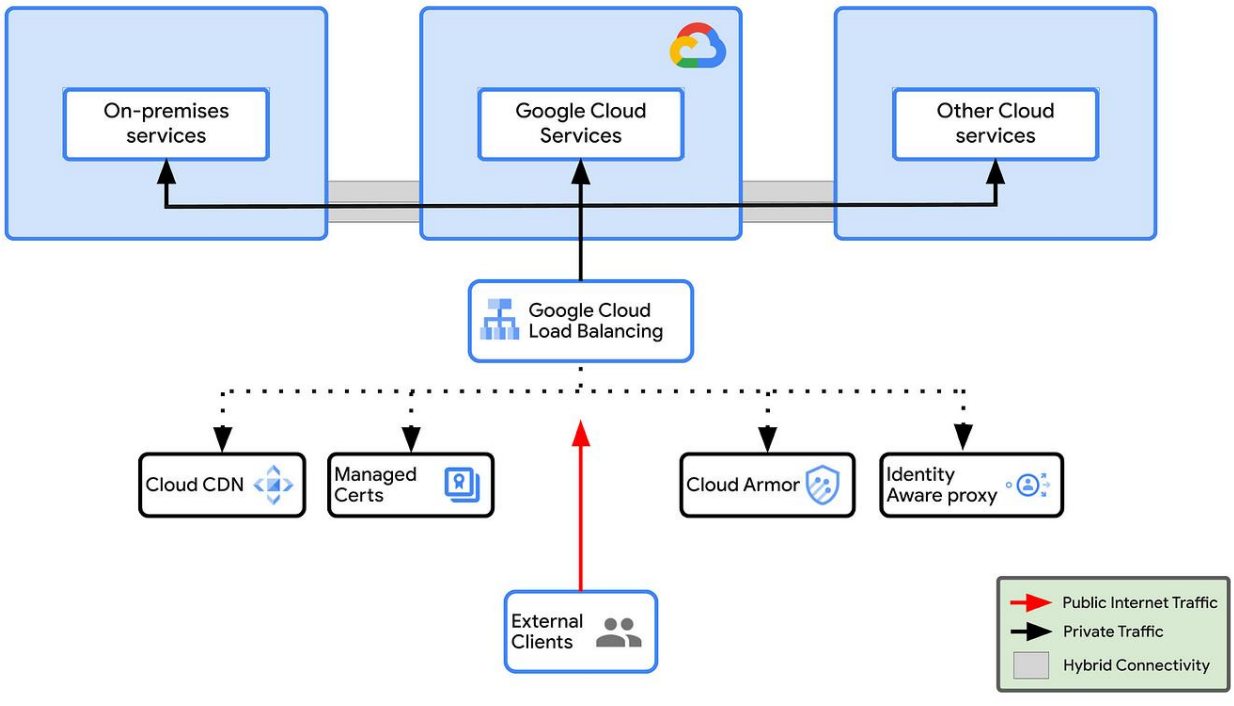
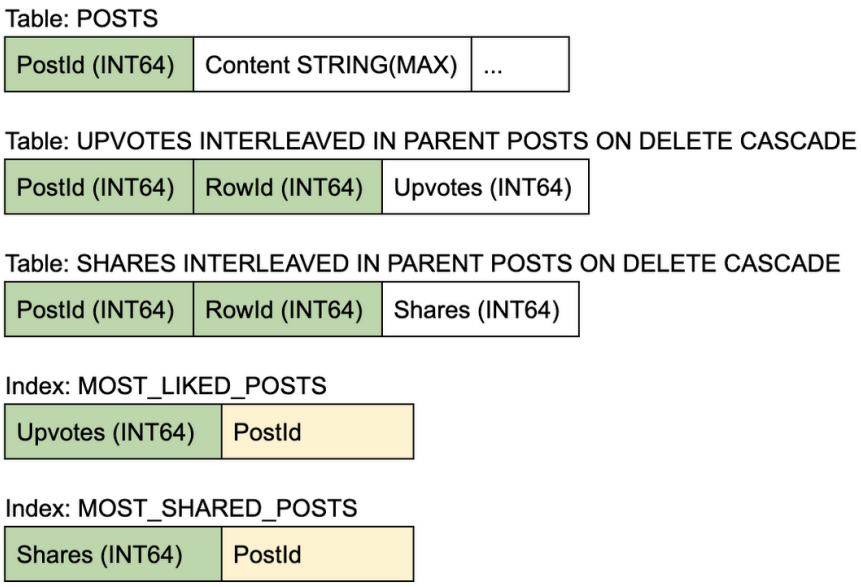


Рис. 2.9. Google Cloud Архітектура

Зразки планів запитів — це візуальний інструмент, який дозволяє розробникам і адміністраторам баз даних візуалізувати плани виконання запитів для історичних виконання запитів. У дописі в блозі наголошується, як можна використовувати цю функцію, щоб зрозуміти зміну затримки під час виконання певного запиту (це допоможе вам відповісти, що змінилося, що спричинило зміни затримки).

Рис. 2.10. Google Cloud Архітектура

Hotspotting визначається як ситуація, коли занадто багато запитів надсилається до одного рядка бази даних, насичуючи ресурси сервера, на якому розміщений рядок, і спричиняє високі затримки кінцевого



користувача.

Рис. 2.11. Hotspotting.

Віддалені функції в BigQuery – це функція, яка дозволяє користувачам розширювати BigQuery SQL за допомогою власного спеціального коду, написаного та розміщеного в Cloud Functions або Cloud Run. Спеціальний код може бути написаний на одній із кількох мов програмування, які підтримуються цими середовищами виконання.

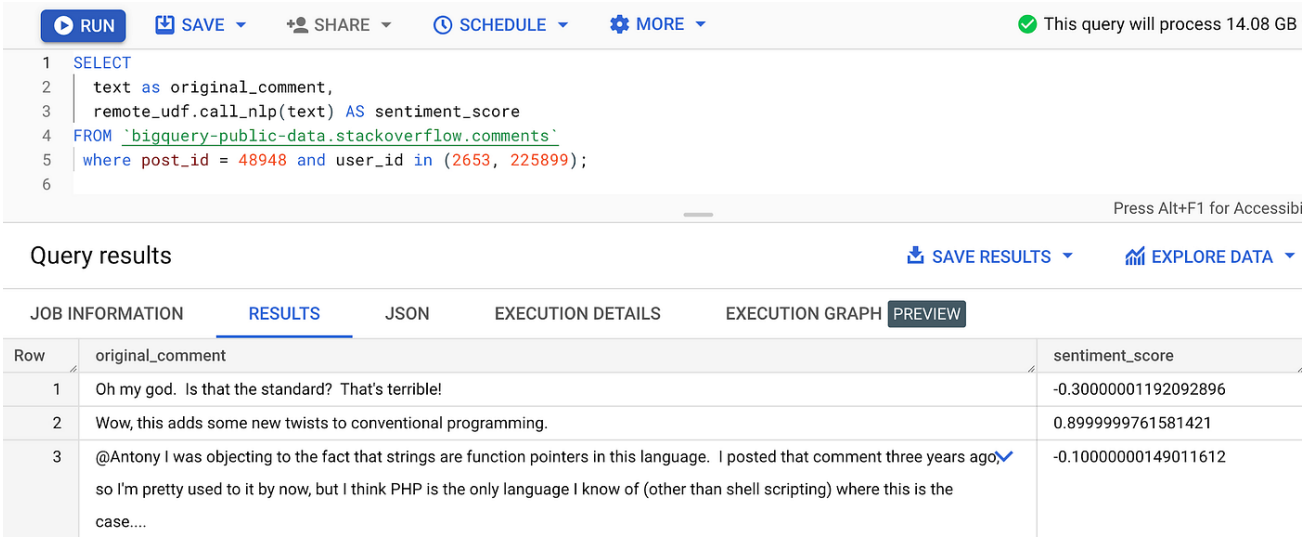


Рис. 2.12. BigQuery

Розробники та практики

Cloud Tasks – один із цікавих сервісів Google Cloud. Це дозволяє створювати незалежні завдання, які можна додавати до черги, і ці завдання зберігаються до виконання. Виконання спрямовується до програми App Engine або довільної кінцевої точки HTTP (працює на Compute Engine, Google Kubernetes Engine, Cloud Run, Cloud Functions або локальній системі), яка обробляє завдання.

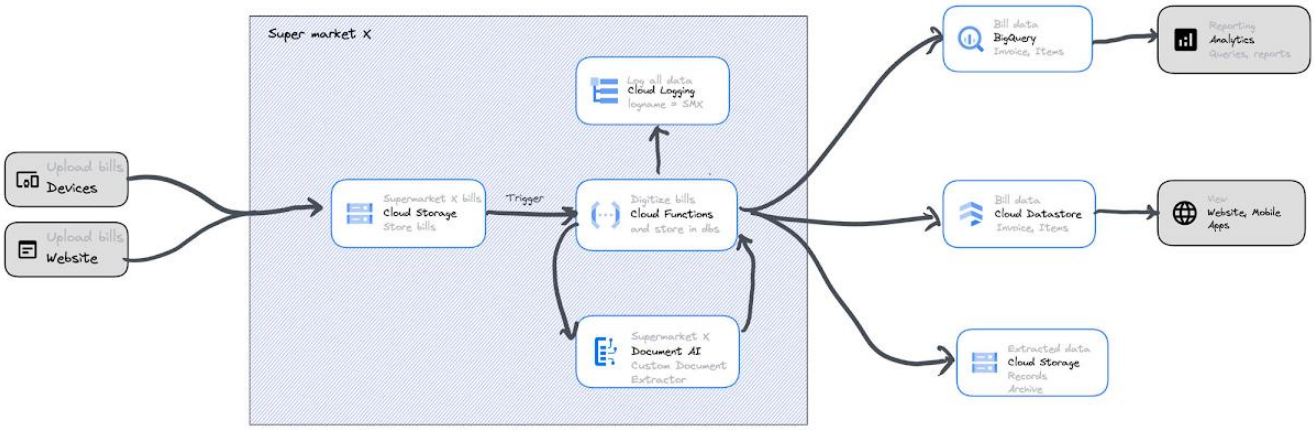


Рис. 2.13. Cloud Tasks

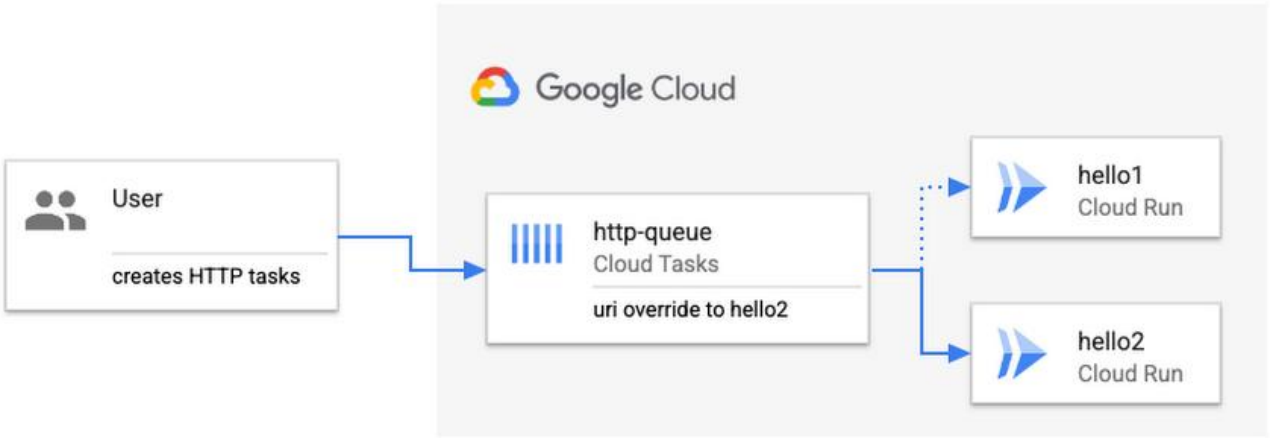


Рис. 2.14. Cloud Tasks HTTP

Популярність Cloud Run продовжує зростати, і організації прагнуть розробляти та розгортати внутрішні програми на платформі. Запуск внутрішніх програм означатиме, що вони не доступні через загальнодоступний Інтернет, а найкращі практики щодо безпеки та роботи в мережі повністю інтегровані в нього. Він охоплює 3 загальні шаблони проектування: внутрішні веб-програми, внутрішні API та мікросервіси, що охоплюють спільний VPC.

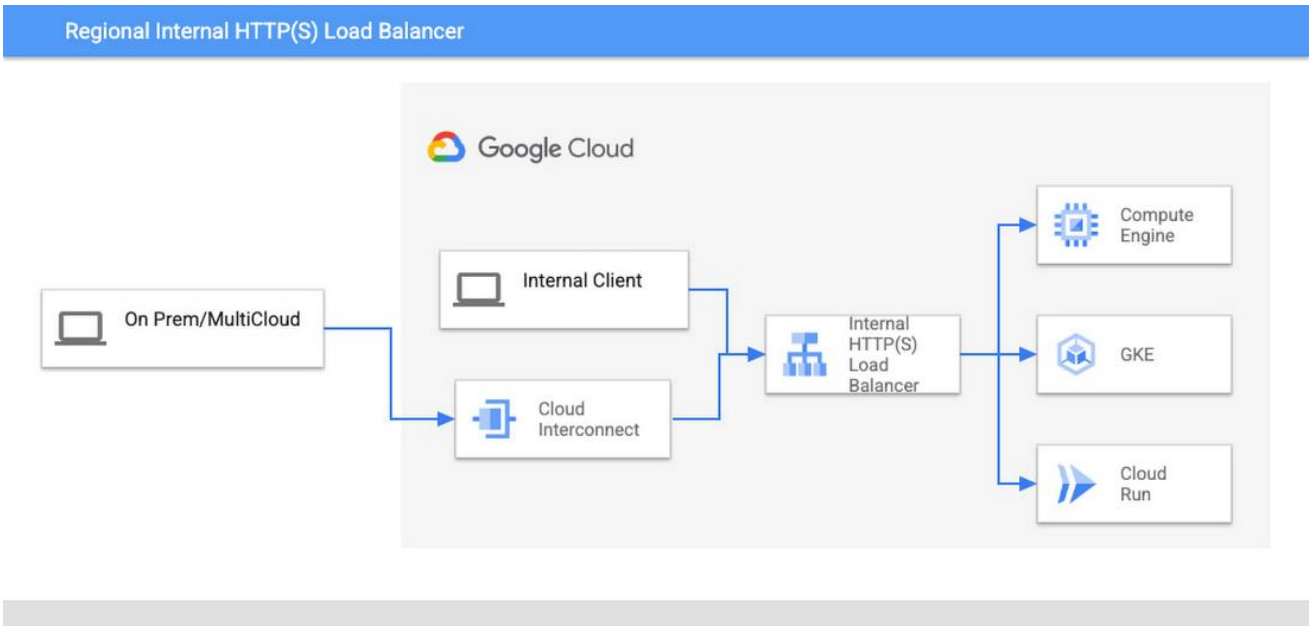


Рис. 2.15. Cloud Tasks HTTPS

DevOps і SRE

Cloud Deploy продовжує регулярно випускати функції. Cloud Deploy є впевненою платформою безперервної доставки для Google Kubernetes Engine,

PHASES SUMMARY

Filter Enter property name or value

Phases and Jobs	Status	Started	Duration	Completed
canary-25	Successful	4/7/23, 8:11 PM	00:03:23	4/7/23, 8:14 PM
Deploy	Successful	4/7/23, 8:11 PM	00:01:40	4/7/23, 8:12 PM
Verify	Successful	4/7/23, 8:12 PM	00:01:39	4/7/23, 8:14 PM
canary-50	Successful	4/7/23, 8:15 PM	00:03:26	4/7/23, 8:18 PM
Deploy	Successful	4/7/23, 8:15 PM	00:01:47	4/7/23, 8:16 PM
Verify	Successful	4/7/23, 8:17 PM	00:01:32	4/7/23, 8:18 PM
stable	Successful	4/7/23, 8:19 PM	00:03:04	4/7/23, 8:22 PM
Deploy	Successful	4/7/23, 8:19 PM	00:01:46	4/7/23, 8:20 PM
Verify	Successful	4/7/23, 8:20 PM	00:01:10	4/7/23, 8:22 PM

Cloud Run і Anthos.

Рис. 2.15. Cloud Deploy

Перевірка часу безвідмовної роботи – це чудова функція, доступна в Google Cloud Monitoring, яка дозволяє контролювати доступність вашої програми без інструментів. Він відстежує доступність ресурсів і затримку через регулярні проміжки часу і може бути першим портом звернення, коли у вашій програмі назріває проблема.

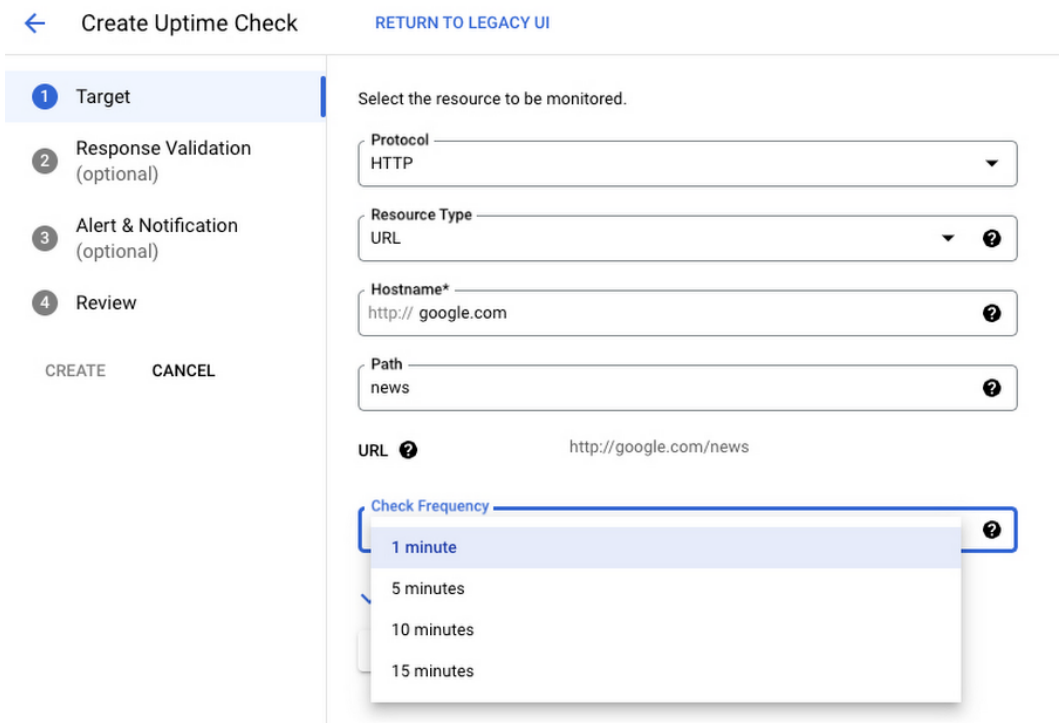


Рис. 2.16. Google Cloud Monitoring.

Варіанти використання

Голосові роботи в контакт-центрах Dialogflow змінюють досвід обслуговування клієнтів за допомогою динамічного генерування мовлення замість відтворення статичного аудіо. Це дає можливість скористатися високоякісними синтезованими голосами, щоб надати абонентам відчуття знайомства та можливості налаштувати їх відповідно до своїх потреб.

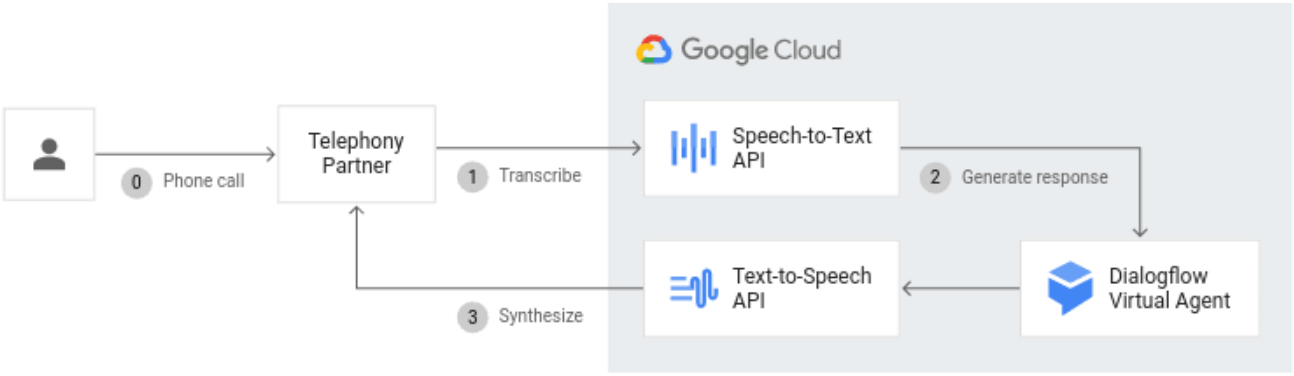


Рис. 2.17. Голосові роботи

Також є можливість дозволити своїм пристроям розмовляти людськими голосами як програми для читання тексту, щоб створити природне спілкування з користувачами. Можна створювати наскрізний голосовий користувальницький інтерфейс, щоб покращити взаємодію з користувачем завдяки простій і привабливій взаємодії.

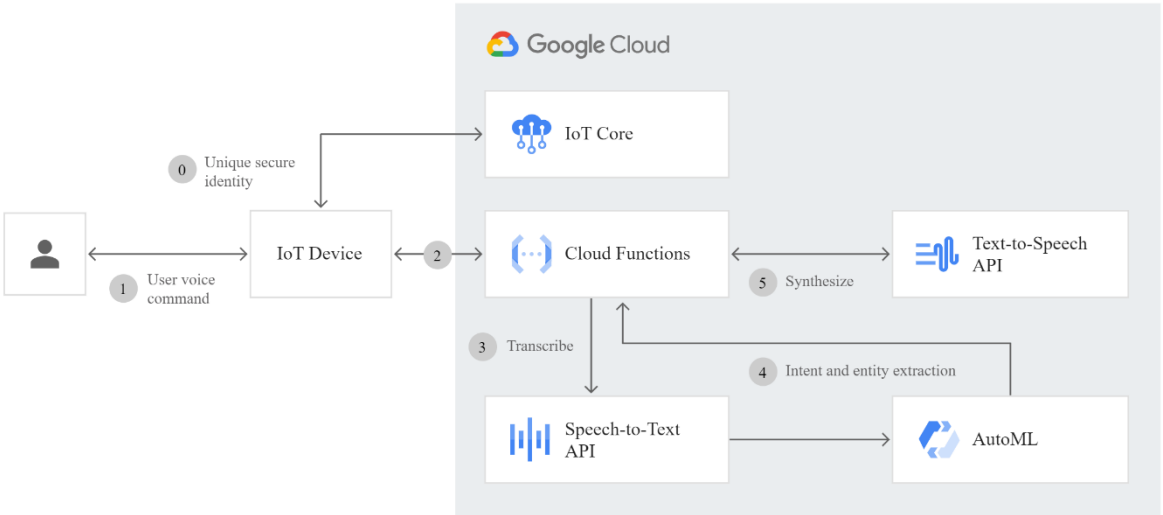
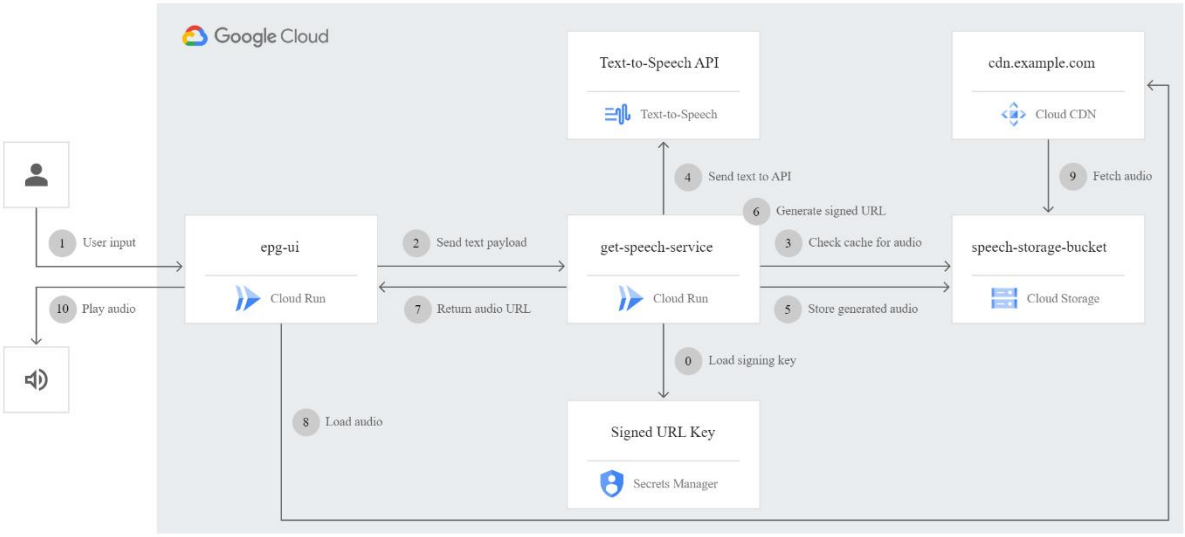


Рис. 2.18. IoT

В контексті технологічного аналізу та оптимізації взаємодії з користувачами, можемо зрозуміти, що інтеграція електронних телегідів (EPG) з механізмами текст-до-мовлення (TTS) може служити ефективним інструментом для покращення доступності контенту. Даний підхід дозволяє



не лише оптимізувати взаємодію з користувачами, але й відповідати сучасним стандартам доступності інформації. Для ілюстрації може бути проведено демонстраційний аналіз можливостей TTS в контексті EPG, підкреслюючи його потенційний вплив на покращення користувацького досвіду та відповідність стандартам доступності.

Рис. 2.19. Text-To-Speech

2.6. Середовище розробки Webstorm

WebStorm — інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей. WebStorm постачається з перед-установленим плагінами JavaScript (такими як для Node.js), котрі доступні для PhpStorm безоплатно (рис. 2.20).

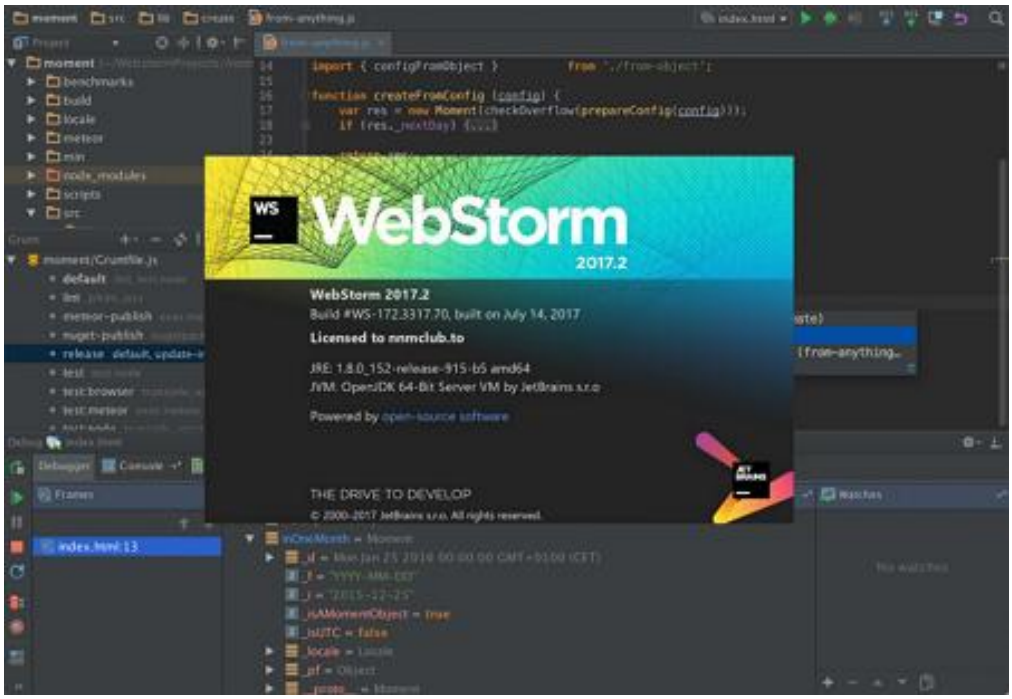


Рис. 2.20. Вікно завантаження програми JetBrains WebStorm

WebStorm підтримує мови JavaScript, CoffeeScript, TypeScript та Dart.

WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) — в таких конструкціях підтримується коректний рефакторинг.

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі проектування та розробки телеграм-бота були визначені та описані функціональні та нефункціональні вимоги до застосунку. Було розглянуто принцип дії, переваги та недоліки архітектурних рішень на основі використання OpenAI для обробки текстового контенту та Google Text-to-Speech для генерації аудіовідповідей.

Програма була розроблена з використанням сучасного стеку технологій для телеграм-ботів та інтеграції з іншими сервісами. Для розробки бота використовувалось середовище WebStorm, що дозволяє ефективно створювати додатки різної складності.

Використання Google Text-to-Speech для генерації аудіовідповідей та OpenAI API для обробки текстового контенту дозволило розробити телеграм-бот з ефективним та стабільним інтерфейсом для користувачів. За допомогою Google Text-to-Speech було забезпечено можливість генерації аудіовідповідей на основі текстового контенту, що робить взаємодію з ботом більш динамічною та комфортною.

OpenAI API використовувався для обробки текстових повідомлень користувачів, що надає можливість генерації адаптованих та інформативних відповідей. Це робить досвід використання бота більш інтелектуальним та відповідальним на запитання користувачів.

Важливо відзначити, що програма відповідає як описаним вимогам, так і загальноприйнятим стандартам для телеграм-ботів, забезпечуючи надійність та зручність використання. В ході розробки було уникнуто типових проблем та враховано можливі аспекти, пов'язані з обробкою голосового та текстового контенту, забезпечуючи якісний та сучасний продукт для користувачів.

РОЗДІЛ 3
РОЗРОБКА ТА ТЕСТУВАННЯ ТЕЛЕГРАМ-БОТА

3.1. Принцип роботи телеграм бота

Для початку, щоб розпочати роботу з даним телеграм-ботом, потрібно його знайти через «Пошук», в самому месенджері Телеграм, увівши його індивідуальний тег **@MasterGPTChatBot**, після чого ми побачимо телеграм-бота «Repeat & Reply» (рис. 3.1).

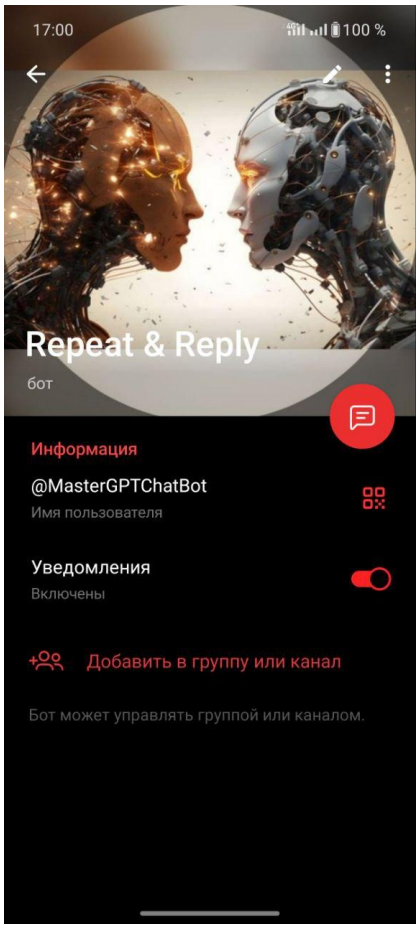


Рис. 3.1. Вигляд телеграм боту.

Коли користувач перший раз відкриває телеграм-бота, перше, що він бачить – це чат в якому його зустрічає короткий опис даного боту, та кнопка «Старт» для того, щоб розпочати роботу з ним (рис. 3.2).

Рис. 3.2. Чат. Кнопка «Старт»

Після того як користувач натиснув кнопку «Старт», він може обрати мову обробки запитів «українська» або «англійська», а також, надсилається повідомлення, що бот очікує голосове або текстове повідомлення від користувача (рис. 3.3).

Рис. 3.3. Початок роботи

В залежності від того, яку мову було обрано, будуть надаватися відповіді саме нею. Тепер користувач вже може надіслати запит в текстовому або голосовому форматі (рис. 3.4).

Рис. 3.4. Повідомлення про вибір мови

Після того, як користувач сформує та відправить своє повідомлення, телеграм-бот надасть відповідь «Повідомлення прийнято. Очікую відповіді від сервера» (рис. 3.5).

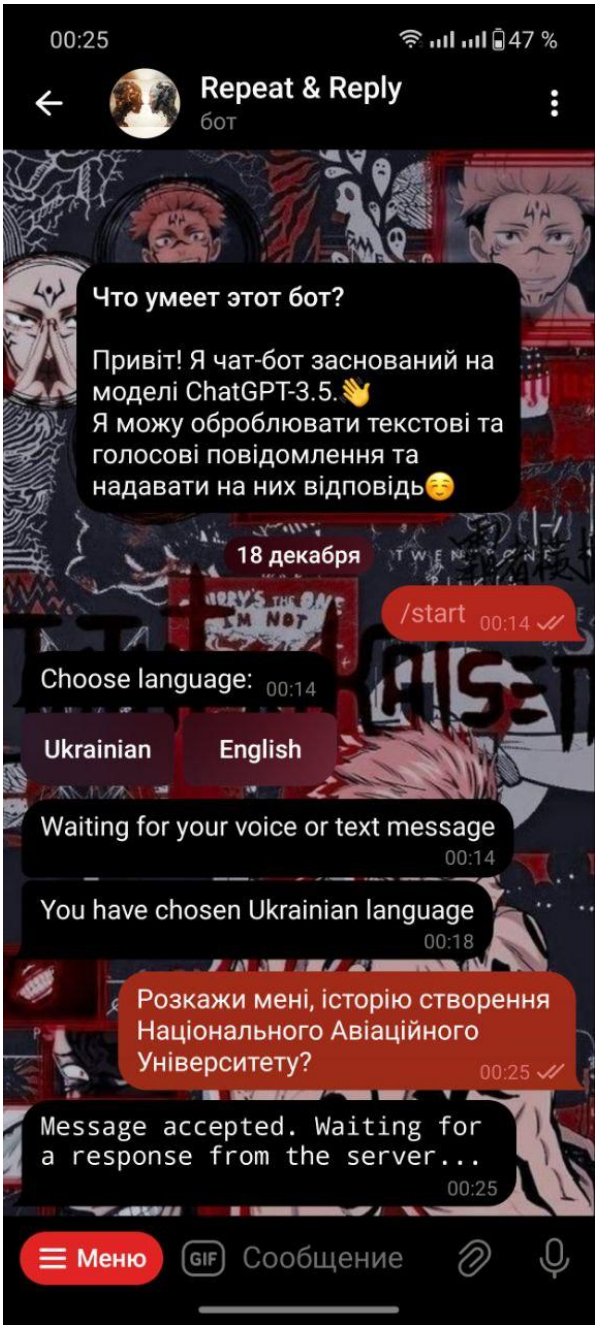


Рис. 3.5. Обробка запиту

Обробивши запит користувача, телеграм-бот надає відповідь у двох форматах: аудіо та текстове повідомлення (рис. 3.6).

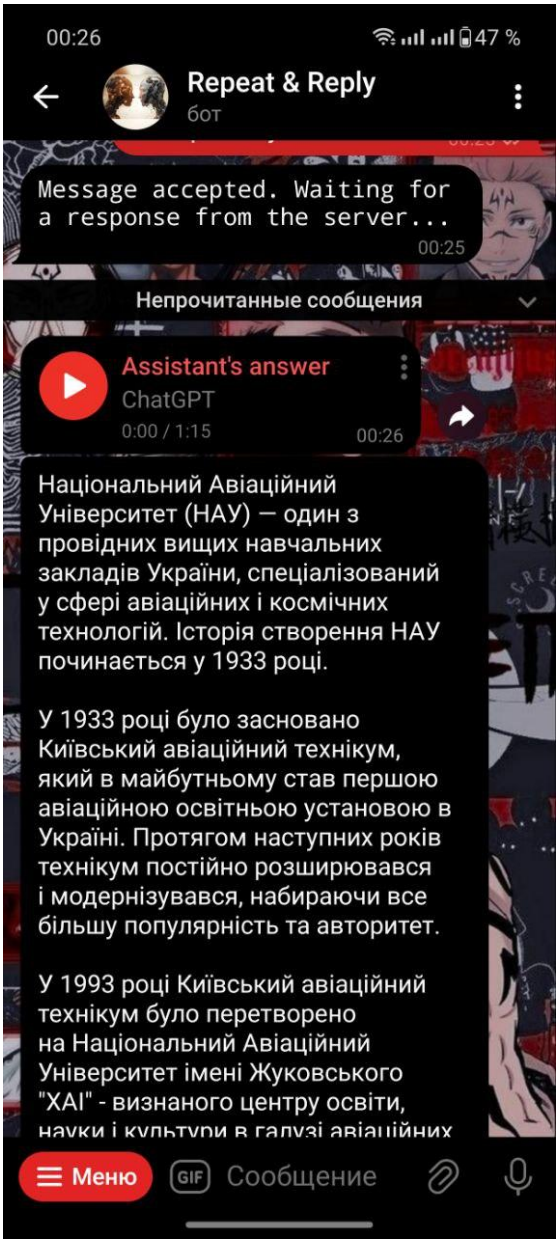


Рис. 3.6. Обробка та надання відповіді.

Користувач може змінити мову обробки запитів, натиснувши у відповідне «Меню». На даний момент у бота є 4 команди (рис. 3.7).

- /start – початок роботи;
- /en – зміна мови обробки запитів на англійську;
- /uk - зміна мови обробки запитів на українську;
- /language – меню мов.

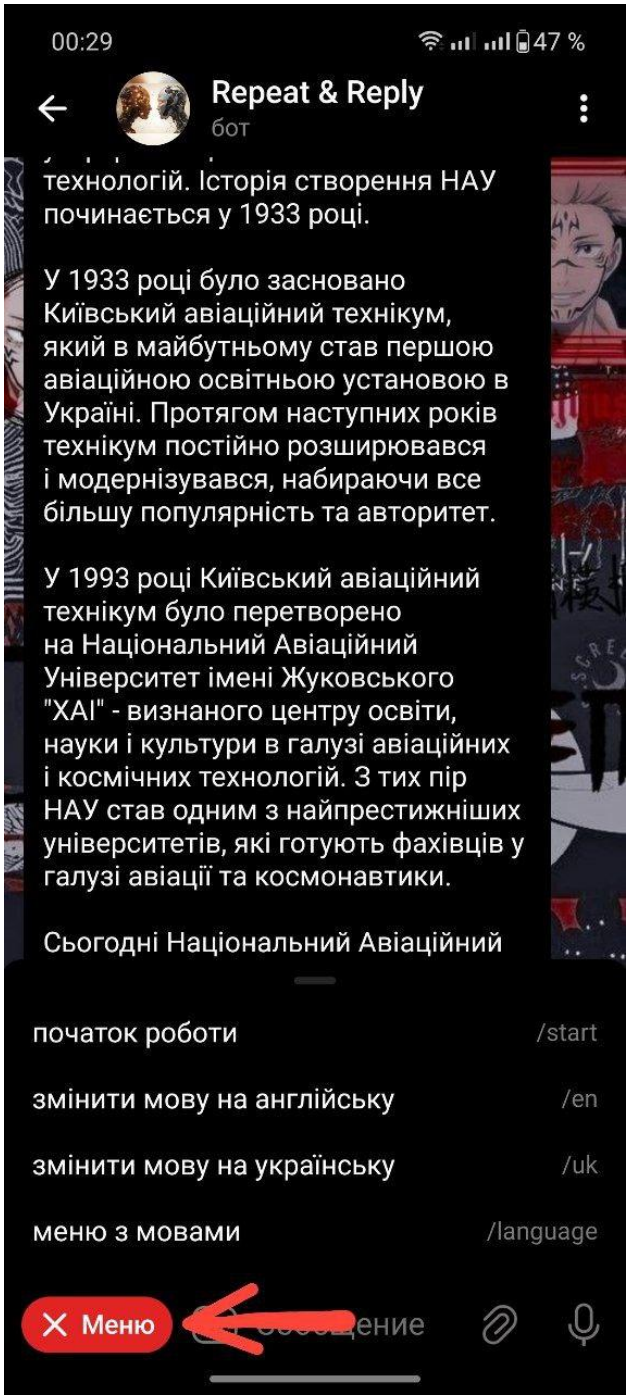


Рис. 3.7. Вигляд «Меню»

Алгоритм роботи телеграм-бота

1. Отримання повідомлення від користувача: коли користувач надсилає овідомлення до телеграм бота, бот отримує це повідомлення та готується передати його для обробки.

2. Відправлення запиту до OpenAI API: бот використовує OpenAI API для взаємодії з моделлю GPT-3.5. Він формує запит, який містить текстове повідомлення користувача, і відправляє цей запит до API.

3. Обробка запиту GPT-3.5: OpenAI API приймає запит від бота і передає його моделі GPT-3.5. Модель обробляє текст і генерує відповідь на основі навчення на величезному обсязі текстових даних.

4. Обробка запиту Text-To-Speech Google: після обробки запиту GPT-3.5 надсилає згенеровану відповідь до Text-To-Speech через Google API.

5. Формування відповіді: після обробки запиту Text-To-Speech, аудіо відповідь надсилається користувачу в телеграм-бот через Telegram API.

6. Надсилання відповіді користувачеві: бот отримує відповідь від GPT-3.5 та Text-To-Speech, і надсилає її користувачеві в телеграм.

3.2. Проєктування телеграм-бота

Main.js

// Імпорти необхідних залежностей

```
import { Telegraf, session, Markup } from 'telegraf';
import { message } from 'telegraf/filters';
import { code } from 'telegraf/format';
import config from 'config';
import { ogg } from './ogg.js';
import { openai } from './openai.js';
import { removeFile } from './utils.js';
import { initCommand, processTextToChat, INITIAL_SESSION } from
'./logic.js';
import { changeLanguage } from './logic.js';
```

Цей блок відповідає за імпорт необхідних бібліотек та функцій:

− **Telegraf**: основний клас для роботи з ботом Telegram у фреймворку Telegraf;

− **session**: мідлвара для роботи з сесіями, що дозволяє зберігати дані між повідомленнями;

– **Markup**: клас для генерації клавіатур та інших інтерактивних елементів;

– **message**: фільтр для обробки повідомлень типу "message";

– **code**: функція для стилізації тексту як коду в Telegram;

– **config**: модуль для роботи з конфігураційними файлами;

– **ogg**, **openai**, **removeFile**: функції та об'єкти з інших модулів, які використовуються в основному коді;

– **initCommand**, **processTextToChat**, **INITIAL_SESSION**, **changeLanguage**: функції та об'єкти з модуля "logic.js", які також використовуються в основному коді.

Цей блок коду служить для підключення всіх необхідних інструментів та функцій, які будуть використовуватися в подальшому коді бота.

// Ініціалізація бота та встановлення сесії

```
const bot = new Telegraf(config.get('TELEGRAM_TOKEN'));

bot.use(session());
```

У цьому блоку ініціалізується об'єкт **bot**, який представляє собою бота Telegram на основі класу **Telegraf**. Також встановлюється сесія за допомогою функції **session()**:

– **new Telegraf(config.get('TELEGRAM_TOKEN'))**: Створюється новий екземпляр бота на основі токену, отриманого з конфігураційного файлу. Токен використовується для з'єднання бота з платформою Telegram.

– **bot.use(session())**: Встановлює мідлвару сесій для обробки та збереження даних сесії між повідомленнями. Це дозволяє зберігати стан бесіди та інші дані для кожного користувача.

Цей блок коду є необхідним для належного налаштування та функціонування бота згідно з платформою Telegram.

// Визначення команд для зміни мови

```
bot.command('en', async (ctx) => {
  await changeLanguage(ctx, 'en-GB');
});

bot.command('uk', async (ctx) => {
  await changeLanguage(ctx, 'uk-UA');
});
```

У цьому блоку коду визначаються команди **/en** та **/uk**, які викликають функції для зміни мови. Кожна з цих команд викликає відповідну функцію **changeLanguage**:

– **bot.command('en', async (ctx) => {...})**: Визначає команду **/en**, яка викликається при написанні користувачем **/en**. У функції обробки команди викликається **changeLanguage** з параметром **'en-GB'**, що вказує на зміну мови на англійську.

– **bot.command('uk', async (ctx) => {...})**: Аналогічно, визначає команду **/uk**, яка викликає функцію **changeLanguage** з параметром **'uk-UA'**, змінюючи мову на українську.

Цей блок коду дозволяє користувачам змінювати мову за допомогою команд **/en** та **/uk**.

// Визначення команди /start та відправлення повідомлення користувачеві

```
bot.command('start', async (ctx) => {
  ctx.session ??= INITIAL_SESSION;
  await ctx.reply('Choose language:', Markup.inlineKeyboard([
    Markup.button.callback('Ukrainian', 'language_uk'),
    Markup.button.callback('English', 'language_en'),
  ]));
  await ctx.reply('Waiting for your voice or text message');
});
```

У цьому блоку коду обробляється команда **/start**. Коли користувач викликає команду **/start**, спочатку ініціалізується сесія (якщо вона ще не існує)

за допомогою **ctx.session ??= INITIAL_SESSION;**. Після цього бот відправляє повідомлення користувачеві за допомогою **ctx.reply**.

Повідомлення містить клавіатуру для вибору мови та текст "Waiting for your voice or text message", що повідомляє користувачеві, що бот готовий отримати його голосове або текстове повідомлення.

Цей блок коду допомагає встановити зв'язок з користувачем, дозволяючи йому обрати мову та надсилати повідомлення після виклику команди **/start.**

// Обробка вибору мови "Ukrainian" через клавіатуру

```
bot.action('language_uk', async (ctx) => {
  ctx.session.chosenLanguage = 'uk-UA';
  await ctx.reply('You have chosen Ukrainian language');
});
```

У цьому блоку коду обробляється вибір мови "Ukrainian" через клавіатуру. При натисканні кнопки "Ukrainian" викликається ця функція. Обрана мова (українська) зберігається в сесії за допомогою ctx.session.chosenLanguage = 'uk-UA';. Після цього бот відправляє повідомлення, що підтверджує обрану мову.

Цей блок коду дозволяє користувачеві вибрати мову "Ukrainian" і зберігає цей вибір в сесії.

// Обробка вибору мови "English" через клавіатуру

```
bot.action('language_en', async (ctx) => {
  ctx.session.chosenLanguage = 'en-GB';
  await ctx.reply('You have chosen English language');
});
```

У цьому блоку коду обробляється вибір мови "English" через клавіатуру. При натисканні кнопки "English" викликається ця функція. Обрана мова (англійська) зберігається в сесії за допомогою ctx.session.chosenLanguage = 'en-GB';. Після цього бот відправляє повідомлення, що підтверджує обрану мову.

Цей блок коду дозволяє користувачеві вибрати мову "English" і зберігає цей вибір в сесії.

// Обробка події надходження голосового повідомлення

```
bot.on(message('voice'), async (ctx) => {
  ctx.session ??= INITIAL_SESSION;
  try {
    await ctx.reply(code('I received a message. Waiting for a
response from the server...'));
    const link = await
ctx.telegram.getFileLink(ctx.message.voice.file_id);
    const userId = String(ctx.message.from.id);
    const oggPath = await ogg.create(link.href, userId);
    const mp3Path = await ogg.toMp3(oggPath, userId);

    removeFile(oggPath);

    const text = await openai.transcription(mp3Path);

    removeFile(mp3Path);

    // Відправляємо текстову відповідь
    await processTextToChat(ctx, text, 'text');

    // Визначаємо тип відповіді (текст чи аудіо)
    const responseType = ctx.session.responseType || 'text';

    if (responseType === 'audio') {
      // Відправляємо аудіо відповідь
      await processTextToChat(ctx, text, 'audio');
    }
  } catch (e) {
    console.log(`Error while voice message`, e.message);
  }
});
```

У цьому блоку коду обробляється подія надходження голосового повідомлення. При отриманні голосового повідомлення бот висилає повідомлення про отримання і розпочинає обробку повідомлення.

Основні етапи обробки голосового повідомлення:

1. Отримання посилання на аудіофайл голосового повідомлення.

2. Отримання ідентифікатора користувача та створення шляху для тимчасового збереження аудіофайлів.

3. Створення OGG-файлу та його перетворення у MP3 для подальшого використання OpenAI.

4. Транскрибування аудіофайлу з використанням OpenAI.

5. Відправлення текстової відповіді на отримане голосове повідомлення.

6. Визначення типу відповіді (текст чи аудіо) та відправлення відповіді в залежності від обраного типу.

Цей блок коду взаємодіє з OpenAI для транскрибування голосового повідомлення та відправляє відповідь на отримане повідомлення.

// Обробка події надходження текстового повідомлення

```
bot.on(message('text'), async (ctx) => {
  ctx.session ??= INITIAL_SESSION;
  try {
    await ctx.reply(code('Message accepted. Waiting for a response
from the server...'));

    // Визначаємо тип відповіді (текст чи аудіо)
    const responseType = ctx.session.responseType || 'text';

    // Обробляємо текстове повідомлення
    await processTextToChat(ctx, ctx.message.text, responseType);
  } catch (e) {
    console.log(`Error while voice message`, e.message);
  }
});
```

У цьому блоку коду обробляється подія надходження текстового повідомлення. При отриманні текстового повідомлення бот висилає повідомлення про прийняття та розпочинає обробку текстового повідомлення.

Основний етап обробки текстового повідомлення:

1. Визначення типу відповіді (текст чи аудіо).

2. Обробка текстового повідомлення та відправлення відповіді в залежності від обраного типу.

Цей блок коду взаємодіє з іншими частинами програми для обробки та відправлення відповіді на отримане текстове повідомлення.

// Запуск бота

```
bot.launch();

// Обробники для зупинки бота при SIGINT та SIGTERM
process.once('SIGINT', () => bot.stop('SIGINT'));
process.once('SIGTERM', () => bot.stop('SIGTERM'));
```

У цьому блоку коду викликається метод **launch()**, який запускає бота. Після цього додаються обробники сигналів **SIGINT** та **SIGTERM** для коректного завершення роботи бота при виході з програми.

Цей блок завершує основний файл **main.js** та запускає бота для взаємодії з користувачами в Telegram.

Logic.js

// Імпорт модулів для взаємодії з OpenAI та конвертації тексту в аудіо

```
import { openai } from './openai.js';
import { textConverter } from './text.js';

export const INITIAL_SESSION = {
  messages: [],
};
```

У цьому блоку коду вказано імпорт необхідних модулів для взаємодії з OpenAI та конвертації тексту в аудіо. Також експортується константа **INITIAL_SESSION**, яка містить початковий стан сесії.

// Функція для зміни мови користувача

```
export async function changeLanguage(ctx, languageCode) {
  ctx.session.chosenLanguage = languageCode;
  await ctx.reply(`You have selected language: ${languageCode ===
```

```
'en-GB' ? 'English' : 'Ukrainian'}`);
  await ctx.reply('Waiting for your voice or text message');
}
```

У цьому блоку коду визначена функція **changeLanguage**, яка встановлює обрану користувачем мову в сесії та відправляє відповідне повідомлення.

// Функція для ініціалізації сесії

```
export async function initCommand(ctx) {
  ctx.session = { ...INITIAL_SESSION };
  await ctx.reply('Waiting for your voice or text message');
}
```

У цьому блоку коду визначена функція **initCommand**, яка ініціалізує сесію та відправляє відповідне повідомлення.

// Функція для обробки текстових повідомлень та взаємодії з OpenAI

```
export async function processTextToChat(ctx, content) {
  try {
    ctx.session.messages = ctx.session.messages || [];
    ctx.session.messages.push({ role: openai.roles.USER, content
});

    // Змінюємо виклик функції textToSpeech, додаючи параметр
обраної мови
    const response = await openai.chat(ctx.session.messages);
    const source = await
textConverter.textToSpeech(response.content,
ctx.session.chosenLanguage);

    await ctx.sendAudio(
        { source },
        { title: "Assistant's answer", performer: 'ChatGPT' }
    );

    // Також можна відправити текстову відповідь:
    await ctx.reply(response.content);
  } catch (e) {
    console.log('Error while processing text to gpt', e.message);
  }
}
```

У цьому блоку коду визначена функція **processTextToChat**, яка відповідає за обробку текстових повідомлень від користувача та взаємодію з OpenAI.

Openai.js

// Імпорт класів та функцій з OpenAI та модуля для роботи з конфігурацією

```
import { Configuration, OpenAIApi } from 'openai'
import config from 'config'
import { createReadStream } from 'fs'
```

У цьому блоку коду виконується імпорт необхідних модулів для роботи з OpenAI та конфігурацією.

// Клас для взаємодії з OpenAI

```
class OpenAI {
  roles = {
    ASSISTANT: 'assistant',
    USER: 'user',
    SYSTEM: 'system',
  }
```

У цьому блоку коду оголошується клас **OpenAI**, який містить об'єкт **roles**, що містить ролі для системи чату.

Конструктор класу OpenAI:

```
constructor(apiKey) {
  const configuration = new Configuration({
    apiKey,
  })
  this.openai = new OpenAIApi(configuration)
}
```

У цьому блоку коду описується конструктор класу **OpenAI**, який приймає ключ API для OpenAI та ініціалізує об'єкт OpenAIApi.

Метод для чат-взаємодії з OpenAI:

```
async chat(messages) {
  try {
```

```
    const response = await this.openai.createChatCompletion({
      model: 'gpt-3.5-turbo',
      messages,
    })
    return response.data.choices[0].message
  } catch (e) {
    console.log('Error while gpt chat', e.message)
  }
}
```

У цьому блоку коду описується асинхронний метод **chat**, який використовує OpenAI для чат-взаємодії. Метод використовує модель **gpt-3.5-turbo** та повертає відповідь.

Метод для отримання транскрипції аудіофайлу:

```
async transcription(filepath) {
    try {
      const response = await this.openai.createTranscription(
        createReadStream(filepath),
        'whisper-1'
      )
      return response.data.text
    } catch (e) {
      console.log('Error while transcription', e.message)
    }
  }
}

// Створення об'єкту класу OpenAI з ключем API
export const openai = new OpenAI(config.get('OPENAI_KEY'))
```

У цьому блоку коду описується асинхронний метод **transcription**, який використовує OpenAI для отримання транскрипції аудіофайлу. Також створюється об'єкт **openai** з ключем API, який експортується для використання в інших частинах програми.

Openai.js

// Імпорт бібліотек та модулів

```
import axios from 'axios'
import ffmpeg from 'fluent-ffmpeg'
import installer from '@ffmpeg-installer/ffmpeg'
import { createWriteStream } from 'fs'
import { dirname, resolve } from 'path'
import { fileURLToPath } from 'url'
```

У цьому блоку коду виконується імпорт необхідних бібліотек та модулів, таких як **axios**, **fluent-ffmpeg**, **@ffmpeg-installer/ffmpeg**, а також модулів Node.js, таких як **createWriteStream**, **dirname**, **resolve**, і **fileURLToPath**.

Оголошення класу OggConverter:

```
class OggConverter {
  constructor() {
    ffmpeg.setFfmpegPath(installer.path)
  }

  toMp3(input, output) {
    try {
      const outputPath = resolve(dirname(input), `${output}.mp3`)
      return new Promise((resolve, reject) => {
        ffmpeg(input)
          .inputOption('-t 30')
          .output(outputPath)
          .on('end', () => resolve(outputPath))
          .on('error', (err) => reject(err.message))
          .run()
      })
    } catch (e) {
      console.log('Error while creating mp3', e.message)
    }
  }

  async create(url, filename) {
    try {
      const oggPath = resolve(__dirname, '../voices',
`${filename}.ogg`)
      const response = await axios({
        method: 'get',
        url,
        responseType: 'stream',
      })
      return new Promise((resolve) => {
        const stream = createWriteStream(oggPath)
        response.data.pipe(stream)
        stream.on('finish', () => resolve(oggPath))
```

```
      })
    } catch (e) {
      console.log('Error while creating ogg', e.message)
    }
  }
}
```

У цьому блоку коду оголошується клас **OggConverter**, який включає конструктор та дві функції: **toMp3** і **create**.

Реалізація функції toMp3:

```
toMp3(input, output) {
  try {
    const outputPath = resolve(dirname(input), `${output}.mp3`)
    return new Promise((resolve, reject) => {
      ffmpeg(input)
        .inputOption('-t 30')
        .output(outputPath)
        .on('end', () => resolve(outputPath))
        .on('error', (err) => reject(err.message))
        .run()
    })
  } catch (e) {
    console.log('Error while creating mp3', e.message)
  }
}
```

У цьому блоку коду реалізується функція **toMp3**, яка призначена для конвертації аудіо у форматі ogg в mp3. Здійснюється використання бібліотеки **fluent-ffmpeg**. Важливо зазначити, що ця функція повертає обіцянку (Promise) і в разі успіху повертає шлях до нового mp3-файлу, а в разі помилки видає повідомлення про помилку.

Реалізація функції create:

```
async create(url, filename) {
  try {
    const oggPath = resolve(__dirname, '../voices',
`${filename}.ogg`)
    const response = await axios({
      method: 'get',
      url,
```

```
      responseType: 'stream',
    })
    return new Promise((resolve) => {
      const stream = createWriteStream(oggPath)
      response.data.pipe(stream)
      stream.on('finish', () => resolve(oggPath))
    })
  } catch (e) {
    console.log('Error while creating ogg', e.message)
  }
}
```

У цьому блоку коду реалізується функція **create**, яка використовується для створення ogg-файлу на основі вказаного URL. Також повертає обіцянку (Promise) зі шляхом до нового ogg-файлу в разі успіху. В разі помилки виводиться повідомлення про помилку.

Openai.js

// Імпорт бібліотек та модулів

```
import { dirname, resolve } from 'path'
import { fileURLToPath } from 'url'
import { readFileSync } from 'fs'
import jwt from 'jsonwebtoken'
import axios from 'axios'
```

У цьому блоку коду виконується імпорт необхідних бібліотек та модулів, таких як **dirname**, **resolve**, **fileURLToPath**, **readFileSync**, **jsonwebtoken**, і **axios.**

Оголошення класу TextConverter:

```
class TextConverter {
  async getToken() {
    const key = JSON.parse(
        readFileSync(resolve(__dirname, '../tgbot-key.json'),
'utf-8')
    )

    const token = jwt.sign(
        {
          iss: key.client_email,
          scope: 'https://www.googleapis.com/auth/cloud-platform',
          aud: 'https://www.googleapis.com/oauth2/v4/token',
```

```
          exp: Math.floor(Date.now() / 1000) + 60 * 60,
          iat: Math.floor(Date.now() / 1000),
        },
        key.private_key,
        { algorithm: 'RS256' }
    )

    const response = await axios.post(
        'https://www.googleapis.com/oauth2/v4/token',
        {
          grant_type: 'urn:ietf:params:oauth:grant-type:jwt-
bearer',
          assertion: token,
        }
    )

    return response.data.access_token
  }

  async textToSpeech(text, languageCode = 'uk-UA') {
    try {
      const url =
'https://texttospeech.googleapis.com/v1/text:synthesize'

      const data = {
        input: { text },
        voice: {
          languageCode: languageCode,
          name: `${languageCode}-Wavenet-A`,
        },
        audioConfig: { audioEncoding: 'MP3' },
      }

      const accessToken = await this.getToken()

      const response = await axios({
        url,
        method: 'POST',
        data,
        headers: {
          Authorization: `Bearer ${accessToken}`,
          'Content-Type': 'application/json',
        },
      })

      return Buffer.from(response.data.audioContent, 'base64')
    } catch (e) {
      console.error('Error while text 2 speech', e.message);
    }
  }
}
```

У цьому блоку коду оголошується клас **TextConverter**, який включає дві асинхронні функції: **getToken** і **textToSpeech**.

Реалізація функції getToken:

```
async getToken() {
  const key = JSON.parse(
      readFileSync(resolve(__dirname, '../tgbot-key.json'), 'utf-
8')
  )

  const token = jwt.sign(
      {
        iss: key.client_email,
        scope: 'https://www.googleapis.com/auth/cloud-platform',
        aud: 'https://www.googleapis.com/oauth2/v4/token',
        exp: Math.floor(Date.now() / 1000) + 60 * 60,
        iat: Math.floor(Date.now() / 1000),
      },
      key.private_key,
      { algorithm: 'RS256' }
  )

  const response = await axios.post(
      'https://www.googleapis.com/oauth2/v4/token',
      {
        grant_type: 'urn:ietf:params:oauth:grant-type:jwt-bearer',
        assertion: token,
      }
  )

  return response.data.access_token
}
```

У цьому блоку коду реалізується функція **getToken**, яка використовується для отримання токену автентифікації для використання сервісів Google Cloud. Здійснюється читання ключів з файлу **tgbot-key.json**, підпис JWT-токену та взяття токену від Google Cloud.

Реалізація функції textToSpeech:

```
async textToSpeech(text, languageCode = 'uk-UA') {
  try {
    const url =
'https://texttospeech.googleapis.com/v1/text:synthesize'

    const data = {
      input: { text },
      voice: {
        languageCode: languageCode,
        name: `${languageCode}-Wavenet-A`,
      },
      audioConfig: { audioEncoding: 'MP3' },
    }

    const accessToken = await this.getToken()

    const response = await axios({
      url,
      method: 'POST',
      data,
      headers: {
        Authorization: `Bearer ${accessToken}`,
        'Content-Type': 'application/json',
      },
    })

    return Buffer.from(response.data.audioContent, 'base64')
  } catch (e) {
    console.error('Error while text 2 speech', e.message);
  }
}
```

У цьому блоку коду реалізується функція textToSpeech, яка використовується для синтезу аудіо з тексту за допомогою сервісу Text-to-Speech Google Cloud. Важливо зазначити, що ця функція повертає об'єкт Buffer з аудіоданими у форматі base64.

Utils.js

// Імпорт бібліотек та модулів

```
import { unlink } from 'fs/promises'
```

У цьому блоку коду виконується імпорт необхідної функції **unlink** з модуля **fs/promises**.

Оголошення асинхронної функції removeFile:

```
export async function removeFile(path) {
  try {
    await unlink(path)
  } catch (e) {
    console.log('Error while removing file', e.message)
  }
}
```

У цьому блоку коду оголошується асинхронна функція **removeFile**, яка використовується для видалення файлу з вказаним шляхом.

Ця функція використовує **unlink**, яке є асинхронною версією функції видалення файлу у Node.js. Вона приймає аргумент **path** - шлях до файлу, який потрібно видалити. Якщо виникає помилка під час видалення файлу, вона виводиться у консоль. В іншому випадку файл успішно видаляється.

Рис. 3.8. Блок схема принципу роботи телеграм-бота

ВИСНОВКИ ДО РОЗДІЛУ 3

В результаті розробки телеграм-бота був створений функціонал, спрямований на зручне та ефективне спілкування з користувачами. Для цього було використано широкий спектр інструментів та сервісів, які дозволяють взаємодіяти з текстовим та голосовим контентом.

Початок відбувався з реалізації можливості вибору мови для забезпечення більш комфортного досвіду користувачів. Це забезпечує можливість обирати мову комунікації, надаючи більше гнучкості.

Очікування голосових та текстових повідомлень відбувається в неперервному режимі, забезпечуючи миттєвий зворотний зв'язок. Голосові повідомлення обробляються, конвертуються у текстовий формат, і подаються для дальшої обробки та відповіді.

Текстові повідомлення піддаються обробці через OpenAI, що забезпечує генерацію контекстно-залежних відповідей. Використання Google Text-to-Speech дає можливість конвертувати текстові відповіді в аудіоформат для більш різноманітного взаємодії з користувачем.

Проект враховує можливість зміни мови у будь-який момент, що робить його ще більш дружелюбним для різномовних користувачів.

Даний підхід до обслуговування помилок включає в себе виявлення та логування проблем, що допомагає вдосконалити якість та стабільність роботи бота.

Розроблений бот став результатом вдалої інтеграції технологій та готових рішень для створення користувацького дружнього та функціонального інтерфейсу у месенджері Telegram.

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено глибокий аналіз сучасних технологій, зокрема технологій ChatGPT, їхніх розширених можливостей та впливу на сучасний діджитал-ландшафт. ChatGPT, використовуючи глибоке навчання та нейронні мережі, демонструє значний потенціал у сферах природної мови обробки, комунікацій та інтерактивної взаємодії. Його застосування, що поширюється від генерації тексту до інтелектуальних систем підтримки, підкреслює його важливість у багатьох сферах життя, включаючи медицину, освіту, та розваги.

Окрема увага була приділена телеграм-ботам на базі штучного інтелекту, які вже знайшли своє застосування в різних галузях, від бізнесу до освіти. Їх потенціал у сфері автоматизації та підтримки користувачів підкреслюється шляхом надання інтерактивних та персоналізованих рішень.

Загальний висновок роботи вказує на те, що технології ChatGPT та телеграм-боти на базі штучного інтелекту мають потужний вплив на еру цифрових технологій, проте їхнє успішне впровадження та розвиток вимагають уваги до етичних стандартів та безпеки. Майбутнє даних технологій може бути яскравим та інноваційним, але також вимагає постійного вдосконалення та адаптації до змінюваних умов.

У дослідженні сучасних технологій, зокрема технології ChatGPT та впровадженні телеграм-бота на її базі, було виявлено значущий вплив цих рішень на сучасний цифровий ландшафт. ChatGPT, з його глибоким навчанням та нейронними мережами, відкриває широкі можливості в сферах природної мови обробки та інтерактивної комунікації. Спільне використання Google Text-to-Speech та OpenAI API у телеграм-боті дозволило створити інтелектуальний інструмент з ефективним інтерфейсом, який задовольняє високі стандарти якості та надійності. Проект було реалізовано з використанням передових технологій, таких як мова програмування JavaScript, що підтверджує його сучасний та перспективний характер. З урахуванням усіх аспектів розробки та архітектурних

рішень, телеграм-бот відповідає всім вимогам та стандартам, забезпечуючи користувачам комфортну та інтелектуальну взаємодію.

Після завершення розробки телеграм-бота було досягнуто високої ефективності та зручності взаємодії з користувачами завдяки впровадженню передових технологій та інтеграції широкого спектру інструментів. Основний акцент зроблено на комфорт користувачів, надаючи їм можливість вибору мови комунікації та швидкість обробки їхніх повідомлень. Інтеграція з Google Text-to-Speech дозволила забезпечити більш глибоке взаєморозуміння за рахунок конвертації текстових відповідей в аудіоформат. Крім того, система ефективно виявляє та логує помилки, що забезпечує стабільність та надійність бота. У результаті розробки створений продукт інтегрує передові технології з практичною користю для користувачів у месенджері Telegram, підкреслюючи важливість сучасних рішень у впровадженні високоякісних телекомунікаційних сервісів.

Отже, виконання даного проекту не лише дало можливість застосувати теоретичні знання, отримані протягом навчання, але й покращило навички роботи з реальними технологіями та API, підняло рівень ефективності в розробці та взаємодії з різними сервісами для створення функціонального та інтелектуального телеграм-бота. Також було описано принцип роботи телеграм бота та складено схему опису алгоритму його роботи.

Результатом виконання кваліфікаційної роботи є застосунок для демонстрації роботи технології обробки живої мови. Даний телеграм бот відповідає описаним та загальним вимогам сучасних телеграм ботів, повністю працездатний та готовий до інтегрування в систему підприємств та компаній.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Language Models are Few-Shot Learners. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020).
2. Sequence to Sequence Learning with Neural Networks. Sutskever, I., Vinyals, O., & Le, Q. V. (2014).
3. Hands-On Chatbots and Conversational UI Development, Srini Janarthanam. (2017).
4. OpenAI. URL: https://www.openai.com/ (21.12.2023).
5. Telegram. URL: https://core.telegram.org/bots (20.12.2023)
6. Cloude Google Text-To_speech. URL: https://cloud.google.com/text-to-speech/#documentation (19.12.2023)
7. Node-Telegram-Bot-API. URL: https://github.com/yagop/node-telegram-bot-api (21.12.2023)
8. BotUI. https://botui.org/ (21.12.2023)
9. MDN Web Docs (Mozilla Developer Network). URL: https://developer.mozilla.org/uk/ (19.11.2023)
10. JavaScript.info. URL: https://javascript.info/ (18.09.2023)
11. W3Schools JavaScript Tutorial. URL: https://www.w3schools.com/js/ (19.09.2023)
12. JavaScript Design Patterns. URL: https://addyosmani.com/resources/essentialjsdesignpatterns/book/ (18.10.2023)
13. Eloquent JavaScript by Marijn Haverbeke. URL: https://eloquentjavascript.net/
14. JavaScriptWeekly. URL: https://javascriptweekly.com/
15. FreeCodeCamp. URL: https://www.freecodecamp.org/
16. JavaScript Design Patterns. URL: https://refactoring.guru/design-patterns/javascript

ДОДАТОК А

```
// Імпорти необхідних залежностей
import { Telegraf, session, Markup } from 'telegraf';
import { message } from 'telegraf/filters';
import { code } from 'telegraf/format';
import config from 'config';
import { ogg } from './ogg.js';
import { openai } from './openai.js';
import { removeFile } from './utils.js';
import { initCommand, processTextToChat, INITIAL_SESSION,
changeLanguage } from './logic.js';

// Створення екземпляру бота з використанням токену з
конфігураційного файлу
const bot = new Telegraf(config.get('TELEGRAM_TOKEN'));

// Використання сесій для зберігання даних між повідомленнями
bot.use(session());

// Команда для зміни мови на англійську
bot.command('en', async (ctx) => {
  await changeLanguage(ctx, 'en-GB');
});

// Команда для зміни мови на українську
bot.command('uk', async (ctx) => {
  await changeLanguage(ctx, 'uk-UA');
});

// Команда /start, яка викликає ініціалізацію бота
bot.command('start', async (ctx) => {
  ctx.session ??= INITIAL_SESSION;
  // Клавіатура для вибору мови
  await ctx.reply('Choose language:', Markup.inlineKeyboard([
    Markup.button.callback('Ukrainian', 'language_uk'),
    Markup.button.callback('English', 'language_en'),
  ]));
  // Повідомлення для користувача після вибору мови
  await ctx.reply('Waiting for your voice or text message');
});

// Обробник для вибору української мови через клавіатуру
bot.action('language_uk', async (ctx) => {
  ctx.session.chosenLanguage = 'uk-UA';
  await ctx.reply('You have chosen Ukrainian language');
});

// Обробник для вибору англійської мови через клавіатуру
bot.action('language_en', async (ctx) => {
  ctx.session.chosenLanguage = 'en-GB';
  await ctx.reply('You have chosen English language');
});
```

```
// Обробник для голосових повідомлень
bot.on(message('voice'), async (ctx) => {
  ctx.session ??= INITIAL_SESSION;
  try {
    await ctx.reply(code('I received a message. Waiting for a
response from the server...'));
    const link = await
ctx.telegram.getFileLink(ctx.message.voice.file_id);
    const userId = String(ctx.message.from.id);
    const oggPath = await ogg.create(link.href, userId);
    const mp3Path = await ogg.toMp3(oggPath, userId);

    removeFile(oggPath);

    const text = await openai.transcription(mp3Path);

    removeFile(mp3Path);

    // Відправляємо текстову відповідь
    await processTextToChat(ctx, text, 'text');

    // Визначаємо тип відповіді (текст чи аудіо)
    const responseType = ctx.session.responseType || 'text';

    if (responseType === 'audio') {
      // Відправляємо аудіо відповідь
      await processTextToChat(ctx, text, 'audio');
    }
  } catch (e) {
    console.log(`Error while voice message`, e.message);
  }
});

// Обробник для текстових повідомлень
bot.on(message('text'), async (ctx) => {
  ctx.session ??= INITIAL_SESSION;
  try {
    await ctx.reply(code('Message accepted. Waiting for a response
from the server...'));

    // Визначаємо тип відповіді (текст чи аудіо)
    const responseType = ctx.session.responseType || 'text';

    // Обробляємо текстове повідомлення
    await processTextToChat(ctx, ctx.message.text, responseType);
  } catch (e) {
    console.log(`Error while voice message`, e.message);
  }
});

// Запускаємо бота
bot.launch();
```

```
// Обробники для зупинки бота при SIGINT та SIGTERM
process.once('SIGINT', () => bot.stop('SIGINT'));
process.once('SIGTERM', () => bot.stop('SIGTERM'));
```