Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти Перший (бакалаврський)

AI-агент для персоналізованого консультування представників велосипедної спільноти

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-1
Денис НИШКУР
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: ас. Єгор КОРНІЄНКО
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти Перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Нишкуру Денису Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи AI-агент для персоналізованого консультування представників велосипедної спільноти

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025

3. Вхідні дані до роботи 1) Мова програмування Python; 2) Середовище PyCharm; 3) Технології розробки Telegram-ботів; 4) Документація та API-інтерфейси;

4. Перелік питань, що потрібно опрацювати у роботі 1) Аналіз предметної області; 2) Огляд та вибір технологій реалізації; 3) Інтеграція з мовною моделлю штучного інтелекту; 4) Розробка архітектури системи; 5) Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій Слайд-презентації - 9

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	28.04.2025-05.05.2025	
2	Вибір технології розробки	10.05.2025-14.05.2025	
3	Розробка telegram-бота	15.05.2025-06.06.2025	
4	Впровадження та тестування	07.06.2025-09.06.2025	
5	Оформлення матеріалів кваліфікаційної робо-	10.06.2025-15.06.2025	
	ти		
6	Подання кваліфікаційної роботи керівникові	16.06.2025	
	та її попередній захист		
7	Подання кваліфікаційної роботи на рецензу-	18.06.2025	
	вання		

Дата видачі завдання " 21 " квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ Єгор КОРНІЄНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 77 с., 12 рис., 2 табл., 2 дод., 22 джерел.

AI-АГЕНТ, КОНСУЛЬТАНТ,PYTHON,ШТУЧНИЙ ІНТЕЛЕКТ, API, ВЕЛИКА МОВНА МОДЕЛЬ, РОЗПІЗНАВАННЯ ГОЛОСУ

Метою кваліфікаційної роботи є дослідження процесу створення інтелектуального агента для персоналізованого консультування представників велосипедної спільноти. Було вивчено, як сучасні AI-моделі, зокрема чат-боти на базі великих мовних моделей, можуть ефективно взаємодіяти з користувачами, враховуючи їх інтереси, запити й індивідуальні особливості.

У ході виконання кваліфікаційної роботи було розроблено інтелектуального агента для персоналізованого консультування велосипедистів. Основною метою було створення зручного інтерфейсу взаємодії у форматі Telegram-бота, який здатен надавати поради щодо тренувань, зберігати індивідуальні дані про користувача. У роботі досліджено сучасні архітектури великих мовних моделей та принципи інтеграції штучного інтелекту до месенджерів. Реалізовано функціонал введення статистики, голосової взаємодії та генерації відповідей на основі попереднього досвіду користувача. Проведено тестування системи за участі представників велосипедної спільноти, що дозволило отримати зворотний зв’язок та сформувати пропозиції щодо подальшого вдосконалення. Отримані результати демонструють ефективність використання AI-технологій для підтримки та розвитку активних спільнот.

ABSTRACT

Bachelor’s thesis: 77 pages, 12 figures, 2 tables, 2 appendices, 22 sources.

AI-AGENT, CONSULTANT, PYTHON, ARTIFICIAL INTELLIGENCE, API, LARGE LANGUAGE MODEL, SPEECH RECOGNITION

The major goal of this thesis is to explore the process of developing an intelligent agent for personalized consulting of members of the cycling community. The study examines how modern AI models, particularly chatbots based on large language models, can effectively interact with users by taking into account their interests, requests, and individual characteristics.

In order to develop an intelligent agent for personalized consulting of cyclists, this thesis involved the creation of a user-friendly interaction interface in the form of a Telegram bot capable of providing training advice and storing individual user data. The study explored modern architectures of large language models and the principles of integrating artificial intelligence into messaging platforms. Functionality was implemented for logging user statistics, enabling voice interaction, and generating responses based on the user's previous experience. The system was tested with members of the cycling community, which provided feedback and suggestions for further improvement. The obtained results demonstrate the effectiveness of AI technologies in supporting and fostering active communities.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ШІ – штучний інтелект

ІАЦ – інформаційно-аналітичний центр

API – інтерфейс прикладного програмування (англ., application programming interface)

ASR – автоматичне розпізнавання мови (англ., automatic speech recognition)

CNN – згорткова нейронна мережа (англ., convolutional neural network)

CTC – конекціоністська тимчасова класифікація (англ., connectionist temporal classification)

DNN – глибока нейронна мережа (англ., deep neural network)

E2E – наскрізний (англ., end-to-end)

GDPR – загальний регламент про захист даних (англ., general data protection regulation)

GMM – модель гауссової суміші (англ., Gaussian mixture model)

GPU – графічний процесор (англ., graphics processing unit)

GRU – рекурентний блок з керуванням затворами (англ., gated recurrent unit)

HMM – прихована марковська модель (англ., hidden markov model)

HTTP – протокол передачі гіпертексту (англ., hypertext transfer protocol)

IDE – інтегроване середовище розробки (англ., integrated development environment)

LLM – велика мовна модель (англ., large language model)

LSTM – довга короткочасна пам’ять (англ., long short-term memory)

NLP – обробка природної мови (англ., natural language processing)

RNN – рекурентна нейронна мережа (англ., recurrent neural network)

SQL – структурована мова запитів (англ., structured query language)

SSL – рівень захищених сокетів (англ., secure sockets layer)

TPU – тензорний блок обробки (англ., tensor processing unit)

VoIP – голос через інтернет протокол (англ., voice over internet protocol)

ВСТУП

У сучасному цифровому ландшафті, що характеризується стрімким зростанням обсягів інформації, критично важливою стає здатність до її фільтрації, персоналізації та адаптації під індивідуальні потреби користувача. Це особливо актуально для вузькоспеціалізованих спільнот, таких як велосипедна, де учасники постійно шукають оперативні та точні поради щодо широкого кола питань: від нюансів тренувального процесу та вибору оптимального спорядження до особливостей технічного обслуговування велосипеда чи пошуку безпечних та цікавих маршрутів. Існуючі традиційні джерела інформації, такі як форуми, блоги чи загальні довідники, часто є розрізненими, містять узагальнені дані або вимагають значних зусиль та часу на пошук релевантних і достовірних відомостей. Це створює інформаційну перевантаженість та може призвести до упущень важливих деталей.

Паралельно з цим, інтенсивний розвиток технологій штучного інтелекту, особливо у сфері обробки природної мови та генерації тексту, відкриває унікальні можливості для створення інтелектуальних помічників. Ці передові системи здатні не лише ефективно аналізувати величезні масиви текстових даних, але й розуміти складні запити користувачів, вести осмислений діалог та надавати високоточні, контекстно-залежні відповіді. Такий прогрес у розробці AI-агентів робить можливим автоматизоване експертне консультування, значно підвищуючи ефективність та зручність доступу до спеціалізованих знань у реальному часі.

Актуальність теми дослідження обумовлюється зростаючою потребою у персоналізованих та доступних джерелах знань для динамічних спільнот, а також швидким розвитком технологій ШІ, що дозволяють задовольнити цю потребу. Розробка спеціалізованих AI-агентів, інтегрованих у повсякденні комунікаційні платформи, такі як месенджери, може кардинально змінити підхід до отримання та використання інформації.

Метою даної кваліфікаційної роботи є дослідження та розробка AI-агента, який буде забезпечувати персоналізоване консультування для представників велосипедної спільноти. Це дозволить вирішити проблему інформаційної перевантаженості, мінімізувати час на пошук достовірних даних та надати доступ до експертних знань у зручному та ефективному діалоговому форматі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття та класифікація AI-агентів

Поняття агентів штучного інтелекту (AI agents) є одним з центральних у сучасній когнітивній та обчислювальній інженерії. На відміну від традиційних систем ШІ, які реалізують фіксовані алгоритми для вирішення конкретних завдань, AI-агенти володіють здатністю автономно сприймати, аналізувати та діяти в динамічному середовищі, адаптуючи свою поведінку на основі зворотного зв’язку та накопиченого досвіду.

Згідно з сучасними дослідженнями, AI-агент — це система, яка може самостійно виконувати завдання від імені користувача або іншої системи, приймаючи рішення, будуючи плани, використовуючи доступні інструменти, і часто взаємодіючи з навколишнім середовищем [1]. Їхні можливості включають, зокрема, обробку природної мови, планування, моделювання поведінки, навчання на основі досвіду, інтеграцію з іншими програмними модулями. Визначальними рисами AI-агентів є:

- автономність – здатність працювати без постійного втручання людини;
- проактивність – здатність ініціювати дії для досягнення цілей;
- соціальна взаємодія – здатність співпрацювати з іншими агентами або людьми.

Еволюція агентів штучного інтелекту була прискорена нещодавніми проривами у великих мовних моделях (LLM), які надали основу для більш досконалих можливостей міркування. Сучасні агенти ШІ використовують ці розвинутості мовні моделі як основні компоненти, доповнюючи їх спеціалізованими модулями для пам'яті, планування, використання інструментів та взаємодії з навколишнім середовищем. Ця інтеграція дозволяє агентам виконувати складні завдання, які були б складними або

неможливими для традиційних систем ШІ, від врегулювання фінансових звітів до надання покрокових інструкцій для польових техніків на основі контекстуального розуміння інформації про продукцію.

Історія розвитку штучного інтелекту демонструє циклічний характер повернення до базових концепцій, закладених ще на ранніх етапах становлення галузі. Наприкінці 1980-х років сформувалася амбіційна мета — створення систем, здатних до загального інтелекту, порівняного або навіть вищого за людський. Уже на початку 1990-х років було відкрито і активно досліджено низку ключових алгоритмів, які стали фундаментом сучасного ШІ: штучні нейронні мережі, навчання з підкріпленням, символічне міркування, еволюційні обчислення тощо.

Попри на дані результати, які ці методи демонстрували у вузькоспеціалізованих задачах, на той момент залишалися дві нерозв’язані проблеми: загальність (тобто здатність адаптуватися до широкого спектру завдань) і масштабованість (ефективна робота на великих обсягах даних та в складних середовищах)[2]. Спроби інтегрувати окремі успішні методи в єдину когерентну систему стикалися з технічними та концептуальними обмеженнями. А масштабування таких систем до рівня вирішення реальних задач, на яких моделі не були попередньо навчено, виявилося практично недосяжним.

Як наслідок, у подальшому розвиток штучного інтелекту змістився у бік агентно-орієнтованих систем, які дозволяють створювати модульні, автономні компоненти з чітко визначеною поведінкою, здатні до адаптації, взаємодії з навколишнім середовищем та навчання. Саме агентний підхід відкрив можливості до більш гнучкої архітектури, яка може поєднувати різні форми інтелектуальної активності в межах однієї системи, та забезпечувати поступовий рух до створення узагальнених і адаптивних ШІ-рішень.

1.2 Компоненти AI-агентів на основі Великих Мовних Моделях

Сучасні агенти штучного інтелекту (AI Agents), побудовані на основі великих мовних моделях (LLM), реалізують складну багатокомпонентну архітектуру. Основними структурними елементами є: механізми планування, пам’ять, використання інструментів, саморефлексія, адаптація, інтерфейси взаємодії, а також модулі вибору дій[3]. Компонент планування відповідає за розбиття складного завдання на низку підзадач та визначення логічної послідовності їх виконання. У випадку LLM-агентів планування реалізується через підходи “Chain-of-Thought” (ланцюжок міркувань) і “Tree-of-Thought” (дерево рішень), що дозволяють моделі поетапно формувати аргументовані дії та обґрунтовувати вибір кожного кроку. Сучасні агенти здатні адаптувати побудований план у режимі реального часу, враховуючи зворотний зв’язок або помилки, виявлені в ході виконання. Це значно підвищує гнучкість і стійкість поведінки агента. На рисунку 1.1 демонструється, як агент формує, аналізує та коригує план дій.

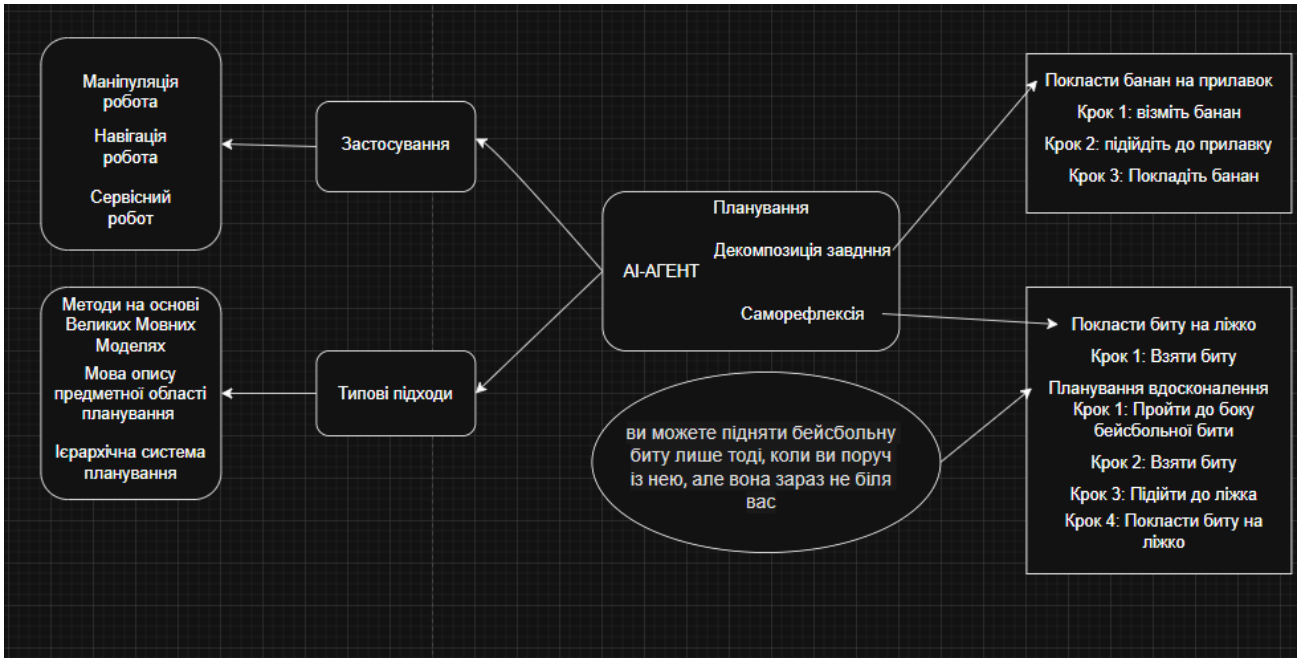

Рисунок 1.1 – Огляд компоненту планування ШІ

Пам’ять є невіддільним елементом когнітивної системи агента. У контексті LLM-агентів розрізняють кілька рівнів пам’яті:

- Training memory – знання, закладені у параметрах моделі під час передтренування[4];

- Short-term memory — оперативна пам’ять, яка охоплює контекст поточної сесії[5];

- Long-term memory — накопичена інформація про попередні взаємодії, яку можна вибірково використовувати при наступних зверненнях[6];

- Episodic та Semantic memory — пам’ять подій (історія користувача) і абстрактних знань (факти, правила)[7].

LLM-агенти використовують зовнішні сховища (бази даних, текстові архіви) для реалізації довготривалої пам’яті, а також можуть навчатися або оновлювати цю інформацію у процесі взаємодії з користувачем. На рисунку 1.2 зображено аналогію між структурою пам’яті людини та пам’яттю LLM-агента.

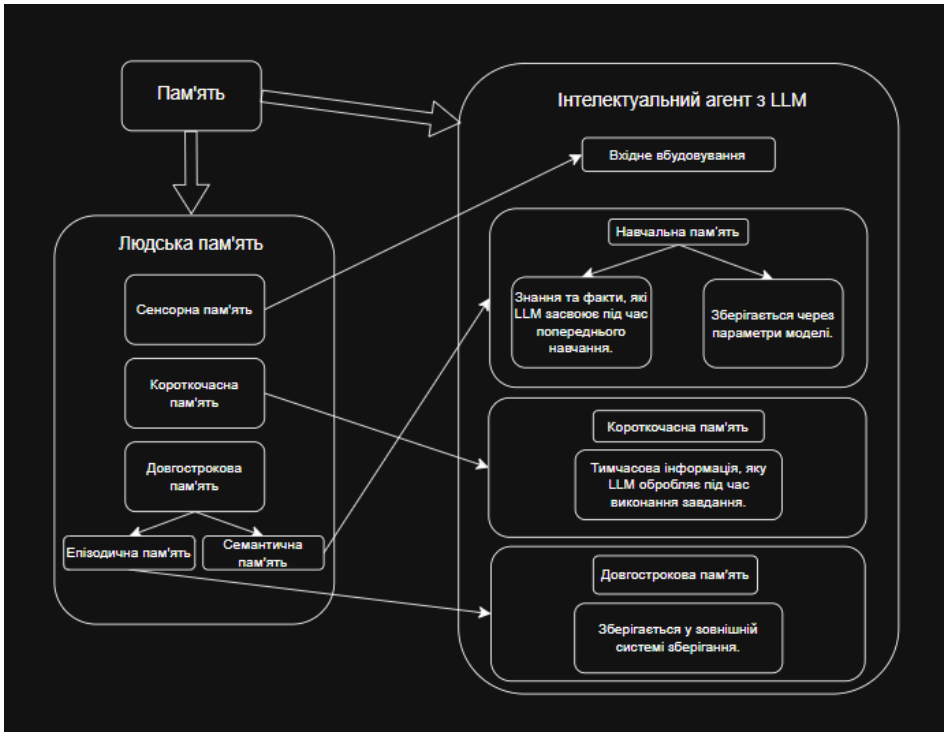

Рисунок 1.2 – Структура пам’яті

LLM-моделі, за своєю природою, мають фіксований набір знань, обмежений навчальним корпусом. Для подолання цього обмеження агенти отримують можливість використовувати зовнішні інструменти — API, бази знань, плагіни, калькулятори, сервіси веб-пошуку. Це розширює їхню функціональність, дозволяє отримувати актуальну інформацію та виконувати дії поза межами мовної моделі. Агенти можуть самостійно обирати, який інструмент залучити, та інтегрувати кілька одночасно для вирішення комплексного завдання.

Важливою ознакою інтелектуального агента є здатність до самоспостереження та самовиправлення. LLM-агенти можуть здійснювати рефлексивний аналіз власних дій, оцінювати впевненість у відповідях, виявляти помилки, переглядати проміжні результати та коригувати хід міркувань. Це реалізується через промпт-архітектури, де модель "перечитує" попередні дії та задає собі запитання про ефективність кожного кроку.

Агент не лише генерує текст — він також приймає рішення про наступну дію. Це може бути текстова відповідь, уточнення, виконання зовнішнього запиту або навіть ініціація нового завдання. LLM-агенти застосовують принципи оцінки корисності, витрат, ризику та контекстної доречності для вибору оптимальної стратегії дій. Адаптація відбувається за рахунок використання підкріплення (reinforcement learning), самонавчання або участі людини у навчанні (human-in-the-loop).

Для ефективної комунікації з користувачем агент повинен підтримувати природні модальності: текст, голос, зображення. LLM-агенти використовують механізми обробки природної мови (NLP) для розуміння намірів користувача, структурування відповідей, а також генерації пояснень до своїх дій — що критично важливо для довіри й зрозумілості.

Важливою складовою сучасного агентного дизайну є безпека, що включає запобігання шкідливим діям, уникнення небажаного контенту, дотримання приватності користувача. Агентні архітектури реалізовують модулі фільтрації, етичні обмеження та механізми відповідальності, що

регулюють допустимі межі поведінки.

1.3 Огляд інтегрованого середовища розробки PyCharm

PyCharm — це інтегроване середовище розробки (IDE), розроблене компанією JetBrains, що спеціалізується на підтримці програмування мовою Python. Однією з основних переваг даного середовища є його безпосередня оптимізація під потреби професійних Python-розробників, включаючи як традиційну серверну логіку, так і сучасні напрямки штучного інтелекту, веброзробки, аналізу даних та автоматизації. Крім того, PyCharm підтримує сумісність з найактуальнішими версіями Python-інтерпретатора, забезпечуючи розробнику доступ до новітніх можливостей мови. IDE надає такі основні функціональні можливості[8]:

- інтелектуальне автодоповнення коду - перевірка синтаксису та семантики в режимі реального часу, підсвічування помилок, а також швидка навігація по структурах проєкту;

- гнучке керування середовищами розробки - включаючи підтримку віртуальних середовищ (venv, pipenv, conda), що дозволяє ізолювати залежності окремих проєктів;

- вбудований відлагоджувач (debugger) - який дає змогу виконувати покрокове налагодження, переглядати вміст змінних та зупиняти виконання коду у визначених точках (breakpoints);

- інтеграція з Git та іншими системами контролю версій - з візуальним інтерфейсом для комітів, порівняння змін, злиття гілок та перегляду історії;

- розширена підтримка роботи з базами даних - включаючи SQL, PostgreSQL, MySQL, SQLite та інші — безпосередньо з вікна середовища;

- вбудовані засоби тестування — підтримка фреймворків pytest, unittest, nose тощо, з автоматичним запуском, переглядом звітів та повторенням тестів;

Окрім зазначеного, PyCharm забезпечує ефективне профілювання

продуктивності програм під час виконання. Розробник має змогу аналізувати використання пам’яті, часу виконання окремих фрагментів коду, а також взаємодію між модулями. Це дозволяє не лише знаходити вузькі місця у логіці виконання, а й оптимізувати роботу з ресурсами. Таким чином, PyCharm є потужним інструментом, що охоплює всі ключові аспекти розробки, налагодження, тестування та підтримки Python-додатків, забезпечуючи стабільність, масштабованість та високу ефективність розробницького процесу.

1.4 Розпізнавання голосу як компонент інтелектуальної взаємодії

Розпізнавання мови (англ. Automatic Speech Recognition, ASR) — це процес автоматичної інтерпретації усного мовлення людини та перетворення його у текстову форму. Упродовж останніх десятиліть ця галузь пройшла значний шлях — від простих систем на основі правил до глибоких нейронних архітектур, здатних до багатомовного, контекстного та шумостійкого розпізнавання.

Перші спроби реалізувати машинне розпізнавання людського голосу датуються ще 1952 роком, коли компанія Bell Laboratories презентувала систему Audrey, здатну розпізнавати окремі цифри, продиктовані чоловічим голосом. Система мала низку обмежень, зокрема потребу в паузах між словами, прив’язаність до конкретного диктора та низьку точність при змінних умовах.

У 1962 році компанія IBM представила систему Shoebox, яка вже могла розпізнавати до 16 англійських слів, що вважалося проривом. Протягом 1970-х років розпізнавання мовлення стало об’єктом стратегічного інтересу урядів (зокрема Міноборони США), що сприяло появі перших систем зі словниковим запасом у 1000+ слів.

У XXI столітті розпізнавання мовлення зазнало радикальних змін завдяки впровадженню глибоких нейронних мереж (Deep Neural Networks,

DNN), що забезпечили новий рівень точності, адаптивності та стійкості до шуму. Класичні статистичні методи, зокрема приховані марковські моделі (HMM) у поєднанні з Gaussian Mixture Models (GMM), протягом тривалого часу були домінуючими в архітектурі систем розпізнавання мови. Проте вони мали низку обмежень: необхідність ручного налаштування ознак, слабку узагальнюваність, вразливість до акцентів і шуму, а також високу чутливість до контексту.

З появою потужних обчислювальних ресурсів (GPU, TPU), масштабних аудіокорпусів та алгоритмів оптимізації, стало можливим тренування глибоких нейромереж, здатних автоматично навчатися з необроблених або слабо оброблених звукових сигналів.

Сучасні системи автоматичного розпізнавання мовлення базуються переважно на таких типах архітектур:

- RNN (Recurrent Neural Network) — дозволяють обробляти послідовності змінної довжини, зберігаючи часові залежності[9];

- LSTM (Long Short-Term Memory) — варіант RNN, здатний утримувати інформацію про далекі залежності у часі, зменшуючи ефект "забування"[10];

- GRU (Gated Recurrent Units) — спрощена версія LSTM з меншою кількістю параметрів, що краще підходить для реального часу[11];

- CNN (Convolutional Neural Networks) — ефективні для обробки спектрограм і локальних патернів у звуковому сигналі[12];

- Transformer — архітектура, побудована на механізмах багатоголового уваги (multi-head attention), здатна обробляти глобальний контекст без рекурсій[13];

- Conformer — сучасний гібрид CNN і Transformer, який поєднує переваги обох підходів і демонструє state-of-the-art (SOTA) результати у багатьох тестових наборах[14];

На відміну від традиційних ASR-систем, які потребували окремого навчання акустичної моделі, мовної моделі та декодера, сучасні end-to-end

моделі поєднують усі ці компоненти в єдину нейронну мережу. Найбільш поширені підходи включають:

- CTC (Connectionist Temporal Classification) — дозволяє зіставляти вхідні аудіо-фрейми з текстовим виходом без потреби у жорсткій сегментації[15];
- Seq2Seq з увагою (attention) — дозволяє точно зіставляти елементи аудіо та тексту за допомогою механізмів динамічного фокусування на релевантних частинах вхідного сигналу[16];
- Transducer-моделі (RNN-T) — поєднують переваги CTC та Seq2Seq, дозволяючи обробляти мовлення в потоковому режимі (streaming)[17];

З метою узагальнення основних характеристик популярних моделей автоматичного розпізнавання мовлення, а також для наочного порівняння їх архітектурних підходів, функціональних можливостей і особливостей застосування, було сформовано таблицю 1.1.

Таблиця 1.1 – Відомі сучасні моделі ASR

Модель	Розробник	Архітектура	Особливості
DeepSpeech	Mozilla	RNN + CTC	Відкрите ПЗ, легке навчання, можна запускати локально
Wav2Vec 2.0	Meta AI	Transformer	Навчається в self-supervised режимі, потребує менше анотованих даних
Whisper	OpenAI	Encoder-Decoder Transformer	Висока точність, багатомовність, адаптація до шуму, слабкого мовлення
Julius	Nagoya Univ.	HMM + GMM	Легка, працює в реальному часі, але менш точна ніж нейронні аналоги
Conformer-	Google	Conformer +	SOTA на LibriSpeech, підтримує

Transducer		RNN-T	потокову обробку

Перехід від класичних статистичних підходів до нейромережевих дозволив системам розпізнавання мови досягти безпрецедентного рівня ефективності, гнучкості та природності взаємодії. Завдяки архітектурам Transformer, Conformer і моделям, що підтримують end-to-end навчання, стало можливим впровадження ASR у мобільні додатки, чат-боти, розумні пристрої та навіть вбудовані системи.

1.5 Аналіз можливостей інтеграції ШІ у месенджери

Месенджери стали важливою складовою повсякденного цифрового життя, перетворившись з інструментів для обміну повідомленнями на багатофункціональні платформи, які підтримують боти, платіжні системи, мікросервіси, канали та інші інтерактивні інструменти. Одним із найперспективніших напрямів розвитку є інтеграція технологій штучного інтелекту (ШІ) у середовище месенджерів.

Інтеграція великих мовних моделей, таких як GPT, Claude або Gemini, у середовище месенджера дозволяє надати нову якість взаємодії — природномовний діалог, контекстну підтримку, генерацію персоналізованих порад на основі попередньої історії взаємодії. На практиці це може виглядати як персональний віртуальний тренер у чаті, що враховує не лише мету користувача, але й умови навколишнього середовища, доступність обладнання, рівень підготовки, навіть настрій користувача, якщо додано відповідну аналітику.

Окрім цього, використання месенджера дозволяє інтегрувати мультимодальні можливості: голосове введення, зображення (наприклад, фото велосипеда для оцінки стану), геолокацію для прокладки маршрутів чи аналізу поїздок, підключення до інших додатків (наприклад, трекерів чи планувальників).

Проте, існують і технічні виклики: обмеження API, необхідність оптимізації моделей для швидкої відповіді, безпека персональних даних, складність у реалізації повноцінної персоналізації без збереження історії взаємодії. Також не всі моделі підтримують українську мову на достатньому рівні якості. Незважаючи на ці труднощі, інтеграція ШІ у месенджери відкриває широкі можливості для створення зручних, доступних і ефективних сервісів персоналізованого консультування велосипедистів. Вона дозволяє використовувати звичний інтерфейс, мінімізує бар’єри входу для нових користувачів та сприяє створенню динамічної спільноти навколо цифрового помічника.

1.5.1 Telegram

Telegram, як один із найбільш технологічно просунутих месенджерів[18], пропонує широкий спектр інструментів для інтеграції штучного інтелекту (ШІ), що робить його популярним серед розробників та бізнесів. Основним механізмом взаємодії є Telegram Bot API, який надає детальну документацію (Telegram, 2025) для створення ботів з функціями обробки природної мови, машинного навчання та автоматизації. Боти можуть інтегруватися з NLP-моделями, такими як GPT або Dialogflow, для аналізу текстових і голосових повідомлень, а також підтримують вебхуки (Webhooks) для ефективного отримання оновлень у реальному часі.

Однією з ключових переваг Telegram є його відкрита архітектура, що дозволяє розробникам використовувати різноманітні бібліотеки (наприклад, python-telegram-bot або aiogram) для створення складних інтерактивних ботів. Крім того, Telegram підтримує inline-режим, коли боти можуть генерувати контент прямо в чаті, що корисно для швидких відповідей або пошуку даних. Також важливим аспектом є можливість роботи з мультимедіа, включаючи обробку зображень, аудіо та відео за допомогою комп’ютерного зору або аудіоаналітики.

Проте інтеграція ШІ в Telegram має певні обмеження. Наприклад, API має ліміти на кількість запитів (до 30 повідомлень на секунду для одного бота)[19], що може бути недостатньо для високонавантажених систем. Крім того, відсутність вбудованих інструментів для навчання моделей безпосередньо в месенджері вимагає використання зовнішніх хмарних сервісів, таких як AWS або Google Cloud. Також варто враховувати проблеми безпеки, оскільки Telegram не надає повного E2E-шифрування для ботів, що може обмежити роботу з конфіденційними даними.

Незважаючи на це, Telegram залишається одним із найкращих варіантів для інтеграції ШІ завдяки своїй гнучкості, масштабованості та активної розробницької спільноти. Висока швидкість роботи API, підтримка різних мов програмування та можливість створення кастомних інтерфейсів роблять його ідеальним вибором для розробників, які хочуть впроваджувати інтелектуальні чат-боти, віртуальних асистентів або автоматизовані системи обслуговування клієнтів.

1.5.2 Viber

Viber, як один із провідних месенджерів на європейському ринку[9], розроблений компанією Rakuten, пропонує унікальні можливості для інтеграції штучного інтелекту через свою бот-платформу. Цей месенджер відзначається стабільною роботою на всіх платформах, високоякісним VoIP-зв'язком та повноцінною підтримкою української мови, що робить його особливо привабливим для українських користувачів та розробників. Платформа Viber for Business надає офіційний API для створення інтерактивних ботів, який дозволяє реалізовувати широкий спектр функцій - від простих інформаційних систем до складних діалогових інтерфейсів з елементами машинного навчання.

Технічна реалізація ботів у Viber базується на вебхуках, що забезпечує обробку повідомлень у реальному часі з обов'язковим використанням SSL-

шифрування для захисту даних. Боти можуть обробляти текстові повідомлення, мультимедійний контент (фото, відео, стікери), а також використовувати інтерактивні елементи інтерфейсу - кастомні кнопки, каруселі, опитування та багатошагові діалоги. Важливою особливістю архітектури Viber є те, що боти можуть надсилати повідомлення лише підписаним користувачам, що вносить певні обмеження у сценарії взаємодії .

Серед ключових переваг платформи слід відзначити офіційну підтримку української мови на всіх рівнях - від клієнтського додатка до API, що є рідкісним явищем серед міжнародних месенджерів. Viber також надає зручні інструменти для роботи з rich-контентом, що дозволяє ефективно представляти інфографіку, новинні матеріали або маркетингові пропозиції. Розробники мають доступ до базової статистики - кількість підписників, активність взаємодії з кнопками та інші метрики, що дозволяє оцінювати ефективність бота.

Однак платформа має ряд істотних обмежень для розробників ШІ-рішень[20]. Найбільш суттєвим є те, що бот не може ініціювати діалог з користувачем - взаємодія можлива лише після явної підписки користувача. Відсутність підтримки inline-ботів та можливості інтеграції в групові чати значно обмежує сценарії використання. Технічні обмеження включають ліміт у 60 повідомлень на хвилину та відсутність вбудованого механізму збереження історії діалогів, що вимагає додаткової реалізації на стороні сервера.

1.5.3 WhatsApp

WhatsApp як найпопулярніший у світі месенджер із понад 2 мільярдами активних користувачів, пропонує унікальні можливості для інтеграції ШІ через WhatsApp Business API. На відміну від інших месенджерів, WhatsApp реалізує жорстко контрольовану модель доступу до свого API, що з одного боку обмежує свободу розробників, а з іншого - забезпечує високий рівень

безпеки та стабільності роботи ботів. Офіційний доступ до API надається виключно через акредитованих партнерів Meta, що значно ускладнює процес розробки для малих компаній та стартапів.

Технічна архітектура WhatsApp для інтеграції ШІ базується на вебхуках і cloud-based рішеннях, що дозволяє обробляти текстові повідомлення, медіафайли, локації та контакти. Особливістю платформи є підтримка end-to-end шифрування для всіх типів повідомлень, що робить її ідеальним вибором для роботи з конфіденційними даними. WhatsApp Business API підтримує створення шаблонів повідомлень, які мають бути затверджені Meta перед використанням - це дозволяє запобігти спаму, але значно збільшує час розгортання ботів.

Однак платформа має серйозні обмеження для розробників ШІ-рішень. Основні з них включають відсутність публічного API для індивідуальних розробників, високу вартість інтеграції (від 0.09 за повідомлення залежно від регіону), а також жорсткі обмеження на частоту відправки повідомлень (до 80 повідомлень на секунду для enterprise-рівня)[21]. Відсутність підтримки inline-ботів і складних інтерактивних елементів інтерфейсу значно обмежує можливості створення складних діалогових систем.

Для інтеграції ШІ в WhatsApp розробники змушені використовувати проміжні сервери та хмарні рішення, що збільшує затримки у відповідях бота. Найбільш поширеними підходами є використання платформи Dialogflow від Google або власних NLP-рішень, інтегрованих через REST API. Важливо відзначити, що WhatsApp не надає власних інструментів для машинного навчання, тому вся обробка природної мови має відбуватися на сторонніх серверах.

1.5.4 Порівняльна оцінка розглянутих месенджерів

Розглянуті платформи — Telegram, Viber, WhatsApp, Facebook

Messenger і Discord — мають різні можливості щодо інтеграції ботів і штучного інтелекту. Кожен із месенджерів має свої переваги та обмеження, які визначають їхню доцільність для певних цілей, зокрема — створення AI-агентів, чат-ботів і консультативних систем.

Telegram є однією з найгнучкіших платформ для розробки ботів. Він має відкритий Bot API, дозволяє швидко інтегруватися зі сторонніми сервісами, підтримує команди, inline-режим, кнопки, меню тощо. Особливістю є зручна підтримка збереження контексту діалогу, що важливо для інтеграції зі штучним інтелектом. Telegram не накладає суворих обмежень на кількість повідомлень чи формат взаємодії, а розгортання бота не потребує модерації.

Viber має потужну бізнес-орієнтовану платформу з підтримкою ботів, але більше сфокусований на комунікації з брендами. Підключення Viber Bot API можливе лише після реєстрації публічного аккаунта. Хоча боти підтримують обробку повідомлень, кнопки, відповіді тощо, платформа має суворіші обмеження щодо контролю з боку Viber. Вона більше підходить для push-сповіщень, обслуговування клієнтів і маркетингових кампаній, ніж для гнучкої AI-інтеграції.

WhatsApp від Meta надає доступ до Business API, що є платним та орієнтованим на середній і великий бізнес. Хоча платформа підтримує інтеграцію зі сторонніми сервісами й базовий рівень автоматизації, вона не дозволяє створення повноцінних чат-ботів без офіційного схвалення та використання платного шлюзу. WhatsApp використовують переважно для односпрямованого інформування клієнтів або підтримки через прості сценарії.

Так було складено загальну таблицю порівняння аналізованих мессенджерів, що наведено у таблиці 1.2.

Таблиця 1.2 – порівняльна таблиця мессенджерів.

Характеристика	Telegram	Viber	WhatsApp
Відкритість API	Відкрите	Відкрите (реєстрація)	Обмежене
Модерація бота	Ні	Так	Так
Комунікаційні формати	Текст, медіа, кнопки	Текст, кнопки	Текст
Гнучкість у інтеграції	Висока	Середня	Низька
Використання в спільноті	Висока	Низька	Низька
Підтримка ШІ	Так	Обмежено	Через платні шлюзи

Для реалізації інтелектуального агента, що виконує роль персонального консультанта у спортивній (зокрема велосипедній) спільноті, було проаналізовано низку популярних месенджерів. Враховуючи вимоги до функціональності, гнучкість API, технічні можливості інтеграції з мовною моделлю, а також потреби цільової аудиторії, найбільш доцільним вибором став саме Telegram.

Ця платформа вирізняється високим рівнем відкритості та зручністю з точки зору розробника. Інтерфейс Bot API не потребує обов’язкової модерації чи верифікації при створенні бота, що суттєво спрощує та прискорює процес його розробки, тестування та подальшої підтримки. У порівнянні з WhatsApp або Viber, які мають обмеження щодо кількості повідомлень, вимагають проходження модераційних процедур або

передбачають оплату за API-доступ, Telegram пропонує більш лояльні та відкриті умови для реалізації гнучкої логіки взаємодії з користувачем.

Головною перевагою, є підтримка розширеного функціоналу для побудови інтерфейсу взаємодії з користувачем: кнопки, меню, callback-запити, inline-режим, групи, канали та інше. Це дозволяє створити інтуїтивно зрозумілий і зручний інтерфейс для користувача, що особливо важливо при побудові консультаційного сервісу.

Telegram забезпечує зручну інтеграцію з мовними моделями, такими як OpenAI GPT або Google Gemini, завдяки можливості надсилання HTTP-запитів. Крім того, підтримка форматованого тексту (Markdown, HTML) надає широкі можливості для оформлення відповідей, а функціонал збереження контексту дозволяє реалізовувати багатокрокові діалоги з урахуванням попередньої історії взаємодії.

Крім того, Telegram відомий високим рівнем безпеки, що є важливим аспектом при роботі з персональними даними користувачів. Боти Telegram працюють у зашифрованому середовищі, а сам месенджер регулярно оновлюється та підтримує багаторівневу валідацію даних.

Багато представників цільової аудиторії (спортсмени, тренери, учасники велосипедних спільнот) вже активно використовують цей месенджер у повсякденному житті. Це означає, що їм не потрібно встановлювати нове програмне забезпечення — бот доступний одразу, просто через посилання або QR-код. Такий підхід значно підвищує залученість користувачів і зменшує поріг входу.

1.6 Постановка задачі

У ході дослідження було встановлено, що попри зростання інтересу до велоспорту та активного способу життя, велосипедна спільнота в Україні досі не має у своєму розпорядженні універсального та інтелектуального інструмента консультування, який би поєднував зручність доступу,

персоналізовану взаємодію та адаптивну обробку запитів користувача. Існуючі рішення — переважно це інформаційні вебресурси, тематичні форуми — мають низку обмежень: фрагментарність інформації, відсутність інтерактивності, необхідність ручного пошуку й обробки даних, а також — обмежена підтримка української мови та локального контенту.

Разом із цим, сучасні досягнення в галузі штучного інтелекту, зокрема розвиток великих мовних моделей (LLM), дозволяють створювати інтелектуальні системи, здатні до ведення природної мовної взаємодії, формування відповідей з урахуванням контексту запиту та надання релевантної інформації у реальному часі. На основі цього постає необхідність у створенні інтелектуального агенту, який би реалізовував функції персонального консультанта для велосипедистів, мав зручний інтерфейс взаємодії у звичному для користувача середовищі, наприклад, месенджері Telegram, і використовував сучасні можливості генеративного штучного інтелекту. З огляду на поставлену мету, у процесі роботи передбачається виконання таких завдань:

- аналіз сучасного стану проблеми консультування у велосипедній спільноті та вивчення існуючих рішень на основі ШІ;
- дослідження можливостей інтеграції великих мовних моделей, , через API-сервіси до інфраструктури чат-бота;
- проєктування архітектури Telegram-бота з урахуванням функцій обробки запитів, генерації відповідей, можливості голосової взаємодії та збереження даних;
- розробка прототипу системи, що забезпечує діалогову взаємодію з користувачем у контексті його запитів щодо тренувань, безпеки, технічного стану обладнання тощо;
- перевірка працездатності системи в умовах реального використання з боку представників цільової аудиторії та формулювання висновків щодо ефективності обраного підходу.
- запровадження напрямку подальших досліджень та покращення

розробленого програмного рішення.

2 ПРОЄКТУВАННЯ TELEGRAM-БОТА З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Вимоги до функціоналу системи

У процесі розробки інтелектуального Telegram-бота для представників велосипедної спільноти важливо сформулювати чіткі вимоги до його функціоналу. Ці вимоги поділяються на дві основні категорії: функціональні, які описують, що саме має робити система, та нефункціональні, що визначають її якість, надійність і технічні обмеження. Також важливу роль відіграє розуміння обмежень, пов’язаних із використанням зовнішніх сервісів, таких як Telegram API та мовна модель, яка виконує роль консультанта.

Основною функцією Telegram-бота є персоналізоване консультування користувачів з питань, пов’язаних із велоспортом, тренуваннями, екіпіруванням, технічним обслуговуванням велосипеда, а також участю у заходах. Бот має надавати відповіді на запити, враховуючи індивідуальні особливості користувача, такі як рівень підготовки, тип велосипеда, стиль катання, цілі використання велосипеда тощо. Щоб забезпечити це, з перших хвилин взаємодії бот ініціює анкетування, яке включає запитання про вік, стать, досвід, місце проживання, фізичну активність, рівень знань і очікування. Ці дані стають основою для подальшого формування відповідей, підказок і рекомендацій.

Крім консультування, бот повинен забезпечувати зручну й логічну навігацію у взаємодії. Для цього необхідна підтримка різних сценаріїв роботи: початкове знайомство, повторне використання, оновлення особистої інформації, запит поради або інформації, перегляд збережених даних. Однією з ключових вимог до системи є збереження історії діалогу, що дозволяє моделі враховувати попередні звернення користувача, формувати більш точні відповіді, а також повертатися до раніше обговорених тем. Цей

функціонал забезпечує безперервність і природність спілкування.

Telegram-бот повинен підтримувати як вільне введення тексту, так і інтерактивні елементи інтерфейсу, такі як кнопки, меню, швидкі відповіді. Це дає змогу користувачу не лише вводити питання вручну, а й швидко переходити до стандартних дій: оновлення профілю, перегляд рекомендацій, виклик допомоги тощо. Важливо, щоб усі повідомлення, команди й реакції були представлені українською мовою, зрозуміло й структуровано, що зробить взаємодію інтуїтивною навіть для користувачів з мінімальним досвідом у спілкуванні з ботами.

Окрім базового функціоналу, система повинна відповідати ряду нефункціональних вимог. Серед них особливо важливими є швидкодія, стабільність, доступність, захист персональних даних та зручність використання. Telegram-бот повинен працювати безперервно, забезпечуючи високу доступність сервісу. Відповіді на запити не мають перевищувати кількох секунд, що забезпечується через ефективну обробку повідомлень, асинхронну архітектуру, а також можливість кешування відповідей або шаблонів.

Захист персональних даних користувача є ще одним критично важливим аспектом. Усі зібрані відомості мають зберігатися у захищеній формі з урахуванням сучасних вимог до безпеки. Це включає шифрування бази даних, обмеження доступу до інформації, журналювання дій та контроль за доступом. Оскільки бот працює з персоніфікованими порадами, важливо гарантувати, що зібрана інформація не потрапить до сторонніх осіб або третіх сервісів без згоди користувача.

Система проєктується з урахуванням реальних технічних обмежень. Наприклад, Telegram API має обмеження на розмір повідомлень, кількість запитів у певний період часу, формат передаваних файлів. Так само, при роботі з мовною моделлю виникають обмеження на кількість токенів, які можна передати в одному запиті, а також обмеження по частоті використання, що регламентується тарифами на обслуговування. Це вимагає

від розробника уважного підходу до побудови діалогу, ефективної роботи з контекстом, фільтрації надлишкової інформації та оптимізації формулювання запитів.

Зрештою, вся система має бути готовою до масштабування, тобто до можливого зростання кількості користувачів, запитів, обсягів даних. Для цього необхідно заздалегідь закласти можливість використання сучасних хмарних рішень, таких як AWS або Google Cloud, або налаштувати власну серверну інфраструктуру з розподіленням навантаження. Загалом, вимоги до функціоналу Telegram-бота мають забезпечити його здатність бути не просто чат-асистентом, а справжнім інтелектуальним помічником для велосипедистів, який адаптується під конкретного користувача, працює швидко, точно й безпечно.

2.2 Структура система та її компоненти

Інтелектуальний Telegram-бот для велосипедної спільноти є багатокомпонентною системою, яка складається з кількох логічно взаємопов’язаних модулів. Його структура визначає, яким чином здійснюється обробка повідомлень користувача, формування відповідей, зберігання даних і взаємодія з зовнішніми сервісами, зокрема з мовною моделлю на основі штучного інтелекту. Розуміння архітектури та внутрішньої будови бота є ключем до успішного його проєктування, підтримки й масштабування.

У центрі всієї системи перебуває модуль взаємодії з Telegram API, який забезпечує отримання вхідних повідомлень від користувачів і надсилання відповідей. Через цей модуль бот реєструє старт нових діалогів, опрацьовує команди (наприклад, /start, /help, /profile), обробляє кнопки й текстові запити. Саме цей компонент виступає "вхідною точкою" у систему, виконуючи роль контролера, що координує дії інших підсистем.

Другим ключовим елементом є модуль обробки повідомлень, який

відповідає за аналіз отриманого тексту, визначення його наміру (інтенту), контексту, тематики та етапу діалогу. У простих випадках, таких як "Оновити профіль" або "Допомога", цей модуль може самостійно сформувати відповідь або викликати відповідний сценарій. Якщо ж запит потребує складної обробки, модуль формує структуру запиту до мовної моделі, передає її далі та потім обробляє отриману відповідь, адаптуючи її під Telegram.

Одним з найважливіших елементів є модуль генерації відповідей, який реалізує зв’язок з мовною моделлю. Цей компонент формує prompt (підказку) на основі запиту користувача, контексту попереднього діалогу та профілю. Він відповідає за підготовку запиту, обробку відповіді моделі, фільтрацію та редагування результату, з урахуванням обмежень платформи та зручності подачі інформації для користувача. Тут також реалізовані механізми повторного запиту, обробки помилок та обмеження на кількість символів.

Інтегральною частиною системи виступає база даних, у якій зберігаються профілі користувачів, історія діалогів, налаштування, статус взаємодії та статистика. Структура цієї бази має бути оптимізованою для швидкої вибірки даних під час кожної нової взаємодії, а також забезпечувати захист персональної інформації. Зберігання історії діалогів дозволяє мовній моделі враховувати попередні питання користувача й підтримувати логічну послідовність відповідей. Крім того, база використовується для контролю дій користувача, наприклад: чи завершено анкету, чи були вже дані поради, які маршрути цікавили, тощо.

Окремо слід виділити модуль сценаріїв — умовну "логіку діалогу", яка визначає, що робити в тій чи іншій ситуації. Наприклад, якщо користувач обрав оновлення профілю, бот повинен покроково запитати нову інформацію. Якщо запит містить незрозумілі або конфліктні дані, бот має коректно відреагувати, запитавши уточнення. Саме сценарний модуль забезпечує природність спілкування, плавність переходів між темами та стійкість до помилок.

Уся система побудована за принципом модульності, що дозволяє легко оновлювати окремі компоненти, змінювати логіку взаємодії, адаптувати API або підключати нові мовні моделі. Такий підхід також забезпечує можливість тестування кожного модуля окремо — наприклад, перевірити лише генерацію prompt'ів, роботу бази даних або обробку помилкових запитів без необхідності запуску всього бота.

Компоненти системи активно взаємодіють між собою. Після отримання повідомлення від Telegram, воно спочатку проходить через модуль обробки, який вирішує, чи потрібно звертатися до ШІ. Якщо так — формується запит, який надсилається до мовної моделі. Отримана відповідь передається до генератора повідомлень, де форматуються репліки з урахуванням платформи Telegram (наявність емодзі, розмір повідомлення, кнопки тощо). Водночас результати записуються до бази даних, щоб зберегти контекст для майбутніх діалогів.

Загальна структура Telegram-бота — це збалансована система, у якій поєднується зручний інтерфейс взаємодії, потужна мовна модель, гнучка логіка сценаріїв та надійне зберігання даних. Саме така архітектура дозволяє створити ефективного асистента, який буде корисним не лише для новачків у велоспорті, але й для досвідчених користувачів, що шукають персоналізовану та актуальну підтримку.

2.3 Архітектура взаємодії з мовною моделлю

Однією з основних переваг сучасних інтелектуальних систем є здатність генерувати осмислені, релевантні відповіді на запити користувача в режимі реального часу. У межах проєкту Telegram-бота для консультування представників велосипедної спільноти така функціональність реалізується завдяки інтеграції з мовною моделлю Google Gemini. Архітектура цієї взаємодії передбачає чіткий обмін запитами та відповідями між Telegram-ботом і зовнішнім API, що забезпечує високу якість комунікації.

Взаємодія починається з фази підготовки запиту, що надсилається до моделі. Кожне повідомлення користувача обробляється ботом і перетворюється у структурований prompt — текстову інструкцію або контекст, у якому формулюється завдання для штучного інтелекту. У prompt можуть включатися не тільки останні запити користувача, а й збережена історія діалогу, його профіль (рівень досвіду, інтереси, обрана мова спілкування), а також додаткова інформація про очікуваний формат відповіді. Таким чином забезпечується персоналізація, яка є критично важливою для ефективного консультування.

Інженерія запитів (prompt engineering) відіграє ключову роль у забезпеченні якості відповідей. Належне формулювання інструкцій дозволяє спрямувати модель на очікувану тематику, стиль та глибину відповіді. Наприклад, у випадку запиту про технічне обслуговування велосипеда, prompt може містити уточнення: "Поясни, як налаштувати задній перемикач передач на гірському велосипеді для початківця, простою мовою". Таке завдання дозволяє моделі згенерувати не просто загальну відповідь, а чітко структуровану інструкцію відповідно до рівня користувача.

Лістинг 2.1 – Приклад prompt`у для AI-агента

```
data = {
            "model": MODEL_NAME,
            "messages": [
                {"role": "system", "content": "Будь ласка,
надавай доброзичливі та корисні поради для велосипедної
спільноти, враховуючи профіль користувача, щоб допомогти
покращити його навички катання, освоєння нових трюків та
розвитку різних стилів катання. Використовуй нейтральний та
теплий тон, уникаючи звертань на ти, аби забезпечити комфортне
спілкування для всіх користувачів. Пам'ятай про рівень та
інтереси користувача, щоб твої поради були максимально кориснимиа відповідними."},
                {"role": "user", "content": full_message}
            ]
        }
```

Після створення prompt-а, бот передає його до API Google Gemini. Важливо враховувати вимоги до формату запиту: як правило, він повинен

містити текстову інструкцію, ідентифікатор сесії або користувача, а також параметри генерації, такі як довжина відповіді, тональність, мова тощо. Система обробки відповіді, отриманої від моделі, повинна забезпечити коректне форматування (розділення на блоки, додавання роздільників або маркування списків) для зручного читання в інтерфейсі Telegram.

Важливим компонентом архітектури є збереження діалогового контексту. Це реалізується шляхом ведення сесії користувача, у межах якої зберігаються ключові повідомлення, запити, теми розмови, а також сформовані поради. Такий підхід дозволяє будувати багатокрокові діалоги, уточнювати інформацію, повертатися до попередніх тем, а також здійснювати адаптивне консультування. Наприклад, якщо користувач запитував про тренування на витривалість, а пізніше — про харчування, система може поєднати ці теми в контексті загального плану підготовки.

З технічного боку, для кожної взаємодії з моделлю застосовується оптимізація обсягу запитів: не вся історія надсилається до моделі щоразу, а лише найбільш релевантні фрагменти, що дозволяє зменшити навантаження на API і прискорити обробку. У випадках тривалих або складних відповідей може використовуватися фрагментація — поділ генерації на кілька частин із подальшим склеюванням відповіді.

Також реалізовано систему обробки помилок: якщо модель не відповіла або API повернув технічну помилку, бот надсилає користувачу повідомлення з проханням повторити запит або дочекатися відновлення з’єднання. Крім того, контролюється довжина запитів, формат тексту, можливі обмеження на кількість одночасних сесій.

Загалом, архітектура взаємодії з Google Gemini побудована таким чином, щоб забезпечити максимальну ефективність, гнучкість та адаптивність. Завдяки цьому Telegram-бот може виконувати функції інтелектуального помічника, що консультує велосипедистів за запитами різного рівня складності, зберігаючи при цьому індивідуальний підхід і високий рівень точності.

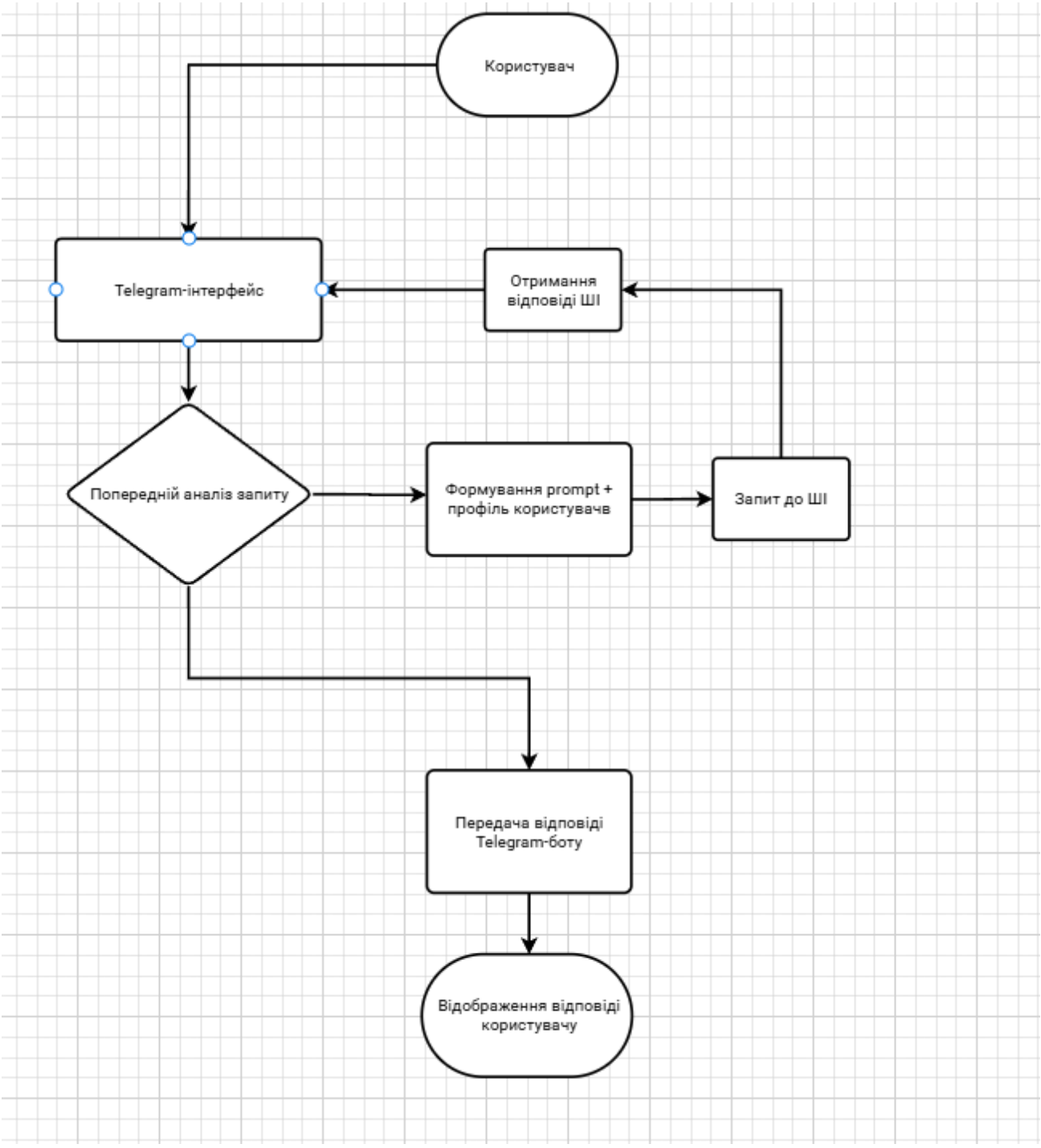

Рисунок 2.1 – Структурна схема логіки обробки запиту

Представлена схема ілюструє логічну послідовність обробки запиту користувача в системі Telegram-бота, яка поєднує використання локальної бази даних і зовнішнього сервісу штучного інтелекту.

Процес починається з ініціації повідомлення користувачем у Telegram. Повідомлення передається через Telegram-інтерфейс, який відповідає за

приймання вхідної інформації. Далі виконується попередній аналіз запиту, що включає розпізнавання голосових повідомлень (за потреби), перевірку формату, вилучення ключових слів і підготовку даних для подальшої обробки.

На основі отриманого запиту та інформації про користувача з бази даних формується prompt — запит у форматі, придатному для передачі до мовної моделі ШІ. Далі відбувається розгалуження логіки: система визначає, чи потребує запит взаємодії з мовною моделлю, чи відповідь можна сформувати на основі локальних даних користувача. У першому випадку ініціюється запит до зовнішньої ШІ-моделі (через API), яка генерує відповідь з урахуванням наданого контексту. У другому — здійснюється формування відповіді безпосередньо з бази даних, наприклад, якщо користувач хоче переглянути або змінити свій профіль.

Отримана відповідь (незалежно від джерела) передається назад через Telegram-бот, який виконує функцію інтерфейсу взаємодії з користувачем, і відображається у вигляді повідомлення в чаті. Подібна архітектура дозволяє об'єднати гнучкість ШІ з ефективністю локального збереження даних, забезпечуючи персоналізований, адаптивний та економічно обґрунтований спосіб консультування користувачів.

2.4 Зберігання та обробка даних користувача

Робота інтелектуального Telegram-бота передбачає не лише генерацію відповідей у реальному часі, а й збереження контексту взаємодії, персональних налаштувань користувача та історії запитів. Саме тому критично важливою є система зберігання та обробки даних, яка забезпечує персоналізацію, безперервність комунікації й адаптацію рекомендацій до потреб конкретного учасника велосипедної спільноти.

Структура профілю користувача є базовим компонентом системи. Профіль містить ключову інформацію, таку як ім’я або псевдонім

користувача, вік (за бажанням), рівень велосипедної підготовки (початківець, аматор, професіонал), тип інтересів (шосейний спорт, MTB, туризм, щоденні поїздки), обраний стиль спілкування (формальний, неформальний), а також мову взаємодії. Окрім цього, профіль може містити дані про цілі користувача — наприклад, "підготовка до марафону", "покращення витривалості", "пошук велозаходів у регіоні".

Ці дані збираються під час першої взаємодії бота з користувачем через діалогову анкету або налаштовуються поступово в процесі спілкування. Зручна реалізація оновлення профілю дозволяє користувачу в будь-який момент змінити свої цілі, рівень або інтереси, що миттєво впливає на якість порад і релевантність відповідей.

Лістинг 2.2 – приклад зберігання даних користувача.

```
def create_or_update_db():
    conn = sqlite3.connect('user_profiles.db')
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS profiles (
        user_id INTEGER PRIMARY KEY,
        name TEXT,
        level TEXT,
        interests TEXT,
        style TEXT
    )''')
    conn.commit()
    conn.close()
```

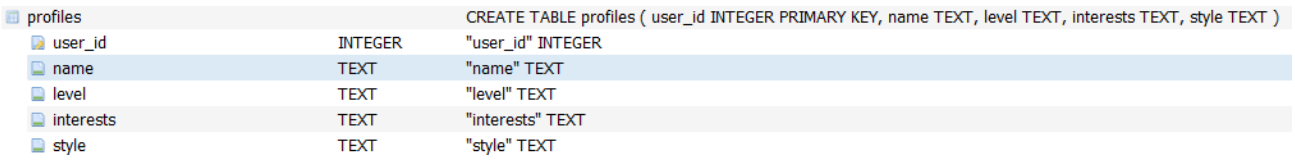

Рисунок 2.2 – таблиця БД в середовищі DB Browser for SQLite

Збереження історії діалогів також є критично важливим аспектом. Уся історія запитів і відповідей зберігається у вигляді сесій, пов’язаних з конкретним користувачем. Це дозволяє:

- - відновлювати попередні теми розмови;

- - формувати рекомендації на основі минулих дій;
- - аналізувати зміну інтересів і поведінки користувача;
- - забезпечувати безперервний діалог без втрати контексту.

Діалоги можуть зберігатися в базі даних у вигляді структури JSON, що містить кожен запит, відповідь моделі, дату і час, а також ключові параметри (наприклад, тональність відповіді або тематичну категорію). Це дає змогу легко фільтрувати або аналізувати історію для покращення роботи бота.

Обробка даних користувача здійснюється згідно з принципами безпеки та приватності. Зокрема, важливо реалізувати:

- обмеження доступу до бази даних — лише авторизовані системні модулі мають право читати або змінювати записи;
- шифрування даних при збереженні або передачі — застосовується як мінімум SSL для API-запитів та алгоритми хешування для чутливої інформації;
- видалення або деактивація профілю на вимогу користувача — відповідно до норм GDPR або локальних законодавств;
- обмеження журналів збереження — дані, що не використовуються, можуть автоматично архівуватися або анонімізуватися.

Додатково, бот може здійснювати аналіз взаємодії на основі зібраних даних: виявляти часті теми, рекомендовані тренувальні плани, корисні поради, які були збережені користувачем як “обрані”. Також можливе формування коротких звітів про активність, зокрема скільки разів користувач консультувався на певну тему, або яку кількість порад отримав за тиждень.

У перспективі можлива реалізація адаптивної поведінки бота: наприклад, якщо користувач часто звертається з технічними питаннями, бот може автоматично пропонувати підбір відеоінструкцій або додаткових статей. Якщо ж з’являється значна перерва у взаємодії, система може ініціювати "перевірку стану" користувача, запропонувавши поновити тренування чи оновити профіль.

Таким чином, ефективна система зберігання та обробки даних у

Telegram-боті не лише підтримує персоналізований підхід, а й забезпечує надійну основу для розширення функціоналу, адаптації порад і формування глибшого взаємозв’язку між користувачем та інтелектуальним помічником.

3 ТЕХНІЧНА РЕАЛІЗАЦІЯ AI-АГЕНТА ДЛЯ КОНСУЛЬТУВАННЯ

3.1 Середовище та інструменти розробки

Для розробки Telegram-бота з використанням штучного інтелекту була обрана мова програмування Python. Цей вибір зумовлений багатьма перевагами, які надає Python для створення сучасних програмних продуктів, особливо у сфері штучного інтелекту, машинного навчання та розробки чат-ботів.

Python є однією з найпопулярніших мов програмування завдяки своїй простоті, читабельності та гнучкості. Його синтаксис дозволяє швидко писати та підтримувати код, що значно прискорює процес розробки. Для реалізації складних функцій, таких як обробка природної мови, робота з API штучного інтелекту, а також інтеграція з Telegram, Python має потужний набір бібліотек і фреймворків, що забезпечують високу продуктивність та стабільність роботи.

Однією з ключових причин вибору Python є наявність спеціалізованих бібліотек, таких як TeleBot (pyTelegramBotAPI) для створення Telegram-ботів, Requests для роботи з HTTP-запитами, SQLite3 для роботи з базами даних, SpeechRecognition і Pydub для обробки голосових повідомлень. Ці інструменти дозволили реалізувати весь необхідний функціонал у межах одного технологічного стеку, що значно полегшило розробку та підтримку проєкту.

Крім того, Python має активну та велику спільноту розробників, що забезпечує широкий доступ до документації, прикладів та готових рішень, які можна адаптувати під власні потреби. Це дозволяє швидко розв’язувати можливі проблеми та впроваджувати сучасні підходи і технології у проєкт.

Для розробки коду було обране інтегроване середовище розробки PyCharm. PyCharm — це потужний і зручний IDE, що підтримує всі основні можливості для роботи з Python, включаючи автодоповнення коду, підсвітку

синтаксису, налагодження, рефакторинг і інтеграцію з системами контролю версій. Використання PyCharm значно підвищує продуктивність розробника, дозволяє швидко виявляти помилки та робити зміни у коді без зайвих зусиль.

Особливо важливим є той факт, що PyCharm має вбудовані інструменти для роботи з віртуальними середовищами, що дозволяє ізолювати залежності проєкту і підтримувати стабільність середовища розробки. Це особливо корисно при роботі з різними бібліотеками для AI та обробки даних, де версії пакетів можуть бути критичними для коректної роботи.

PyCharm також забезпечує інтеграцію з системою контролю версій Git, що дозволяє вести облік змін у коді, працювати у команді і безпечно впроваджувати нові функції. Це особливо важливо для дипломної роботи, де важливо мати чітку структуру розробки і можливість повернутись до попередніх версій у разі потреби.

Таким чином, вибір Python як основної мови програмування та PyCharm як середовища розробки був зумовлений їхньою простотою, потужністю та широкими можливостями, які значно спростили процес створення інтелектуального Telegram-бота. Цей технологічний стек дозволив швидко реалізувати необхідний функціонал, забезпечити якість коду і підготувати надійний інструмент для консультацій велосипедистів із застосуванням штучного інтелекту.

3.2 Архітектура та структура проєкту

Для реалізації функціоналу Telegram-бота з використанням штучного інтелекту було використано кілька ключових бібліотек Python, кожна з яких була обрана з урахуванням її специфічних переваг, сумісності з іншими компонентами проєкту та ефективності у вирішенні поставлених завдань.

Першою і основною бібліотекою є pyTelegramBotAPI (telebot), яка забезпечує зручний інтерфейс для створення Telegram-ботів. Ця бібліотека

відома своєю простотою у використанні, широким функціоналом та стабільністю. Вона дозволяє швидко організувати обробку повідомлень, створення інтерактивних меню, клавіатур, а також підтримує всі сучасні можливості Telegram API. Завдяки цьому розробник може зосередитись на логіці роботи бота, не витрачаючи час на низькорівневу інтеграцію з Telegram.

Для роботи з HTTP-запитами використовується бібліотека requests. Вона є однією з найпопулярніших і найпростіших у застосуванні бібліотек для відправлення запитів до зовнішніх API. Requests підтримує роботу з методами GET, POST, PUT та іншими, що дає змогу легко взаємодіяти з сервісом OpenRouter для отримання відповідей від мовної моделі штучного інтелекту. Лаконічний і зрозумілий синтаксис requests підвищує швидкість розробки і надійність коду.

Для збереження даних користувачів вибрана стандартна бібліотека sqlite3, що надає можливість працювати з вбудованою базою даних SQLite. Це легковажний, але водночас потужний інструмент, який не потребує налаштування окремого серверу баз даних, що значно спрощує розгортання і тестування. SQLite ідеально підходить для зберігання інформації про профілі користувачів у невеликих і середніх проєктах, забезпечуючи надійність і швидкість доступу до даних.

Для розпізнавання голосових повідомлень використовується бібліотека SpeechRecognition, яка є однією з найпопулярніших у Python для конвертації аудіо у текст. Вона підтримує різні бекенди, зокрема Google Web Speech API, що дає змогу отримувати високоякісне розпізнавання мови українською мовою без необхідності власної складної інфраструктури. Це важливо для зручності користувачів, які можуть спілкуватися з ботом голосом, що підвищує інтерфейс і доступність бота.

Для конвертації та обробки аудіофайлів застосовується бібліотека pydub, яка дозволяє легко працювати з різними форматами звуку, конвертувати аудіо з одного формату в інший (наприклад, з OGG у WAV), а

також редагувати аудіофайли. Ця бібліотека має простий API і забезпечує сумісність із популярними аудіокодеками, що дозволяє надійно інтегрувати голосове розпізнавання в Telegram-бота.

Кожна з цих бібліотек була обрана за її спеціалізацію, надійність, активний розвиток та простоту інтеграції у спільний проєкт. Вони доповнюють одна одну, утворюючи ефективний інструментарій для створення інтелектуального агента, здатного підтримувати живе спілкування, персоналізовані консультації та обробку голосових повідомлень у рамках Telegram.

3.3 Організація бази даних та управління профілями користувачів

У проєкті для зберігання даних про користувачів і їхні профілі використовується реляційна база даних SQLite. Вибір саме SQLite був зумовлений кількома ключовими перевагами. По-перше, SQLite є вбудованою базою даних, яка не потребує окремого серверного середовища чи складної конфігурації, що робить її ідеальним варіантом для невеликих і середніх проєктів, зокрема для Telegram-ботів, які працюють на локальних або хмарних серверах без спеціального адміністрування баз даних. По-друге, SQLite забезпечує високу швидкість операцій читання і запису, що особливо важливо для швидкого реагування бота на запити користувачів.

База даних містить одну основну таблицю profiles, у якій зберігаються такі поля: унікальний ідентифікатор користувача user_id, ім’я name, рівень катання level, інтереси interests та стиль катання style. Така структура дозволяє зберігати всі необхідні персональні дані для формування контексту консультацій. Ідентифікатор користувача використовується як первинний ключ, що гарантує унікальність записів і дозволяє легко знаходити профілі конкретних користувачів.

Управління профілями реалізовано через набір функцій, які відповідають за створення, перегляд, оновлення і збереження інформації.

Створення профілю відбувається за допомогою поетапного опитування користувача через послідовність повідомлень у Telegram. Користувач вводить своє ім’я, рівень підготовки, інтереси і стиль катання, що потім зберігається у базі даних. Такий покроковий підхід дозволяє мінімізувати помилки при введенні та забезпечує збір повної і структурованої інформації.

Перегляд профілю надає користувачу можливість переконатися у правильності введених даних, що підвищує довіру до бота і робить взаємодію більш прозорою. Функція редагування профілю дозволяє оновлювати інформацію, що є важливим для підтримки актуальності даних, особливо враховуючи те, що рівень навичок і інтереси можуть змінюватися з часом.

Важливим аспектом є безпека і цілісність даних. Використання SQLite дозволяє локально контролювати збереження інформації, а використання SQL-запитів з параметризованими значеннями (INSERT OR REPLACE) запобігає ін’єкціям і забезпечує коректне оновлення профілів.

Зберігання та управління профілями користувачів у базі даних є ключовим компонентом для персоналізації консультацій на основі ШІ. Дані профілю безпосередньо передаються до API штучного інтелекту, що дозволяє створювати відповіді, які враховують індивідуальні особливості користувача — рівень підготовки, інтереси та стиль катання. Це підвищує якість взаємодії і задоволеність користувачів.

Таким чином, застосування SQLite як системи управління базою даних забезпечує простоту розгортання, надійність роботи, швидкий доступ до інформації і можливість масштабування функціоналу в подальшому, якщо це буде потрібно. Управління профілями реалізовано таким чином, щоб максимально полегшити роботу користувачів і забезпечити персоналізований підхід до кожного велосипедиста.

3.4 Інтеграція API штучного інтелекту та обробка запитів користувачів

Однією з ключових складових функціональності розробленого

Telegram-бота є інтеграція із зовнішнім сервісом штучного інтелекту — API OpenRouter, що надає доступ до сучасної мовної моделі Gemini. Цей API дозволяє створювати діалоги з користувачем на природній мові, що забезпечує високу якість персоналізованих консультацій з урахуванням індивідуальних особливостей велосипедиста.

Інтеграція реалізована за допомогою HTTP-запитів, які надсилаються бібліотекою requests. Використання requests було обрано через її простоту, стабільність, та широку підтримку у спільноті Python-розробників. Ця бібліотека забезпечує ефективний обмін даними між ботом та сервером OpenRouter, підтримуючи сучасні стандарти безпеки, такі як HTTPS і авторизацію токеном.

При отриманні повідомлення від користувача, бот здійснює попередній запит до локальної бази даних SQLite, щоб отримати інформацію про профіль користувача — ім’я, рівень підготовки, інтереси та стиль катання. Ці дані формують контекст, який є критично важливим для коректної роботи ШІ-моделі, адже дозволяють надати поради, що максимально відповідають персональним потребам користувача. Завдяки цьому рекомендації бота виходять більш цільовими і практичними.

Дані профілю і текстове повідомлення користувача об’єднуються у спеціальний формат запиту, що складається зі списку словників з ролями — «system» і «user». Системне повідомлення містить інструкції для моделі, які регламентують тон і стиль відповіді, а також завдання — надавати доброзичливі, інформативні поради щодо катання на велосипеді BMX з урахуванням індивідуальних параметрів. Користувацьке повідомлення містить безпосередньо текст запиту, що моделює реальний діалог.

Після формування запиту відбувається надсилання HTTP POST-запиту з відповідними заголовками, де присутній ключ API для аутентифікації. Сервер OpenRouter обробляє цей запит, виконує генерацію відповіді за допомогою моделі Gemini і повертає JSON-об'єкт із варіантами відповідей. Розроблений бот аналізує отриманий результат, вибирає відповідь і надсилає

її користувачу у вигляді повідомлення Telegram.

Особлива увага приділяється обробці помилок і виключень, що може виникати при роботі з мережею або API. У разі недоступності сервісу, помилок аутентифікації або невірного формату відповіді бот інформує користувача про проблему, що покращує користувацький досвід та дозволяє уникнути повного "зависання" або некоректної роботи системи.

Вибір API OpenRouter з моделлю Gemini було зроблено через кілька причин. По-перше, ця платформа підтримує передові алгоритми генерації природної мови, які дозволяють створювати живі, емпатійні діалоги з користувачами. По-друге, OpenRouter пропонує гнучкий інтерфейс для розробників із детальною документацією, що значно полегшує інтеграцію та подальше масштабування функціоналу. По-третє, Gemini-модель має можливість тонкого налаштування та підготовки інструкцій, що дозволяє адаптувати стиль спілкування і зміст відповідей саме під цільову аудиторію велосипедистів.

Таким чином, використання API штучного інтелекту не лише підвищує функціональні можливості бота, але й створює цінність для кінцевого користувача, забезпечуючи персоналізований, контекстно-залежний сервіс консультування, який можна порівняти з живим тренером чи наставником. Це значно розширює можливості цифрового помічника і сприяє розвитку велосипедної спільноти через інтерактивний і технологічно просунутий інструмент.

3.5 Обробка голосових повідомлень і їх інтеграція у роботу бота

У сучасному світі голосові технології набирають все більшої популярності, оскільки вони забезпечують зручний та швидкий спосіб взаємодії з цифровими сервісами, особливо у мобільних додатках та месенджерах. Для розробленого Telegram-бота, який консультує велосипедистів, підтримка голосових повідомлень є важливою функцією,

оскільки це дозволяє користувачам швидко отримувати відповіді, не відволікаючись на набір тексту. Це особливо актуально для активних велосипедистів, які можуть спілкуватися з ботом під час руху або тренувань.

Для реалізації розпізнавання голосових повідомлень була обрана бібліотека speech_recognition — одна з найпопулярніших і зручних у використанні Python-бібліотек, що забезпечує доступ до декількох сервісів розпізнавання мови, серед яких Google Speech Recognition, яка має високу точність та підтримує українську мову. Основною перевагою speech_recognition є простота інтеграції у вже існуючий код, а також стабільність роботи і можливість швидко адаптуватися до різних аудіоформатів.

Telegram надсилає голосові повідомлення у форматі OGG, який на відміну від WAV або MP3, менш поширений для обробки мовою Python, особливо для розпізнавання голосу. Тому було використано бібліотеку pydub, яка дозволяє легко конвертувати аудіофайли з формату OGG у WAV. Конвертація у WAV необхідна, оскільки модуль розпізнавання голосу приймає саме цей формат для аналізу аудіо. pydub відзначається простотою у використанні, гнучкістю та стабільністю, що зробило її ідеальним вибором для цього завдання.

Після отримання та конвертації аудіофайлу, розпізнавання відбувається за допомогою Google Speech Recognition API, який аналізує аудіо, перетворюючи мовлення користувача у текст. Цей текст далі обробляється в рамках основного функціоналу бота — передається у модуль генерації відповіді на основі штучного інтелекту, що забезпечує релевантні поради велосипедисту з урахуванням його профілю.

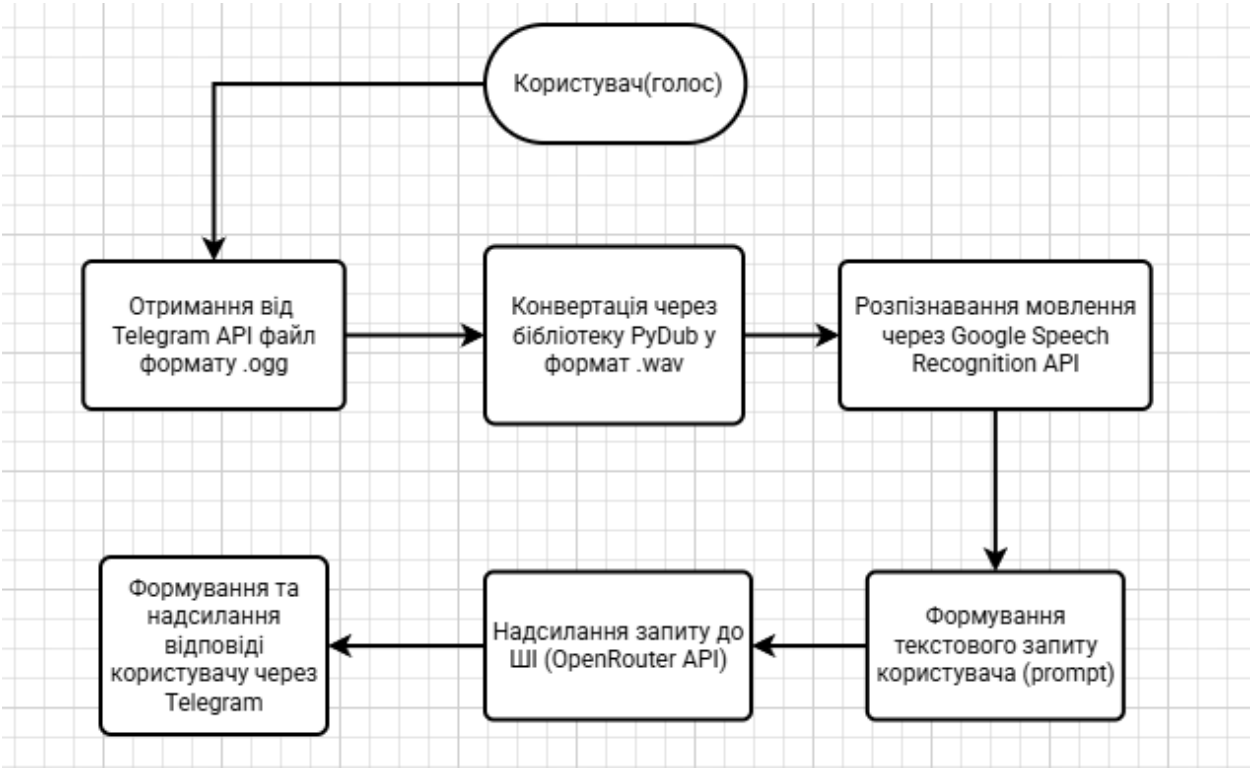

Рисунок 3.1 – структурна схема алгоритму обробки голосового запиту

Однією з важливих особливостей реалізації є грамотне управління тимчасовими файлами. Після завершення конвертації і розпізнавання аудіо, тимчасові файли видаляються, щоб не перевантажувати файлову систему та не зменшувати продуктивність сервера. Це також покращує безпеку, оскільки не зберігаються особисті голосові дані користувачів.

Обрана стратегія обробки голосових повідомлень оптимально поєднує легкість впровадження, високу точність розпізнавання і швидкість роботи. Відсутність необхідності у побудові власних моделей розпізнавання мови або складних алгоритмів дозволила зосередитись на основному функціоналі бота — персоналізованих порадах для велосипедистів. Важливим фактором є також підтримка української мови сервісом Google, що забезпечує якісне розпізнавання аудіо, адаптованого до цільової аудиторії.

Загалом, інтеграція голосових повідомлень значно підвищує доступність та зручність користування ботом. Вона створює комфортний користувацький досвід, дозволяючи велосипедистам отримувати

консультації навіть у ситуаціях, коли набір тексту не є зручним або безпечним, наприклад під час їзди на велосипеді. Таким чином, ця функція є важливою складовою успішної реалізації проєкту, спрямованою на покращення взаємодії користувачів з ботом та підвищення їх залученості.

4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Основний функціонал Telegram-бота

Розроблений інтелектуальний Telegram-бот "BMX Sensei" є багатофункціональним інструментом для персоналізованого консультування представників велосипедної спільноти, зокрема тих, хто цікавиться BMX. Застосунок забезпечує зручну та інтуїтивно зрозумілу взаємодію через інтерфейс месенджера Telegram.

На рисунку 4.1 наведено початковий екран взаємодії з ботом. При першому запуску або команді `/start` бот вітає користувача, представляючись як "вело-гід", та пропонує писати або надсилати голосові повідомлення з питаннями щодо BMX, трюків та тренувань. Це свідчить про підтримку як текстового, так і голосового введення запитів, що підвищує зручність користування.

Рисунок 4.1 – Початковий екран взаємодії з BMX Sensei

На цьому ж екрані бот надає короткий опис своїх можливостей, що включає:

- поради по трюках: консультування щодо виконання різноманітних трюків на БМХ;

- тренування: надання рекомендацій та порад щодо тренувального процесу для велосепидистів;

- бмх-стиль: інформація та консультація стосовно різних стилів та напрямків у БМХ;

- інтелектуальні відповіді від ШІ: забезпечення розумних та контекстно-залежних відповідей, що базуються на великих мовних моделях;

- персональний профіль для кращих порад: можливість створення та використання для надання більш точних та індивідуальних рекомендацій.

Бот заохочує користувачів розпочати взаємодію командою /start, наголошуючи на інтерактивному та динамічному характері спілкування.

На рисунку 4.2 продемонстровано реакцію бота на команду /start, яка активує додаткові інтерактивні елементи у вигляді кнопок. Ці кнопки надають користувачеві прямий доступ до функцій керування профілем:

- створити профіль: дозволяє новому користувачеві або тому, хто ще не має профілю, ініціювати його створення;

- переглянути профіль: надає можливість існуючим користувачам переглянути збережені дані свого персонального профілю;

- редагувати профіль: забезпечує функціонал для внесення змін до наявного профілю користувача;

Рисунок 4.2 – Інтерактивні кнопки для керування профілем

Ці функції керування профілем є ключовими для реалізації персоналізованого консультування, оскільки профіль користувача може містити інформацію про його рівень досвіду, інтереси, фізичні параметри, що дозволить ШІ-агенту адаптувати свої рекомендації та відповіді.

На рисунку 4.3 детально представлено процес створення персонального профілю користувача. Після вибору опції "Створити профіль" бот послідовно запитує у користувача необхідну інформацію:

- ім'я: бот просить ввести ім'я користувача (наприклад, "Денис");
- рівень: запитується рівень підготовки велосипедиста (наприклад, "Початківець"). Це дозволяє адаптувати складність порад та рекомендацій;
- інтереси: користувач може вказати свої інтереси в BMX (наприклад, "бажання вчитись");
- стиль катання: запитується бажаний стиль катання (наприклад, "Стріт").

Рисунок 4.3 – Процес створення персонального профілю

Після успішного введення всіх даних бот підтверджує, що "Профіль створено!". Наявність такого профілю забезпечує базу для персоналізації взаємодії, дозволяючи AI-агенту надавати консультації, що максимально відповідають індивідуальним потребам та цілям кожного користувача.

На рисунку 4.4 продемонстровано можливість перегляду створеного профілю, а також розширення функціоналу після його успішного створення. Після натискання кнопки "Переглянути профіль" бот виводить всю збережену інформацію про користувача: ім'я, рівень, інтереси та стиль катання.

Рисунок 4.4 – Перегляд персонального профілю

На рисунку 4.5 показано процес редагування вже існуючого профілю користувача, що забезпечує гнучкість та актуальність збережених даних. Після вибору опції "Редагувати профіль" (доступної з головного меню профілю), бот послідовно запитує нові значення для кожного поля профілю:

- нове ім'я: користувач може змінити своє ім'я (наприклад, з "Денис" на "Влад");
- новий рівень: можливість оновити рівень підготовки (наприклад, з "Початківець" на "Профі");
- нові інтереси: оновлення інтересів (наприклад, з "Бажання вчитись" на "Вдосконалення результатів");
- новий стиль: зміна бажаного стилю катання (наприклад, зі "Стріт" на "Парк").

Рисунок 4.5 – Процес редагування та оновлення персонального профілю

Після успішного введення оновлених даних бот підтверджує, що "Профіль оновлено!". Можливість миттєвого перегляду оновленого профілю (після натискання "Переглянути профіль") підтверджує успішність змін. Ця функція є важливою для підтримки актуальності персональних даних користувача, що, у свою чергу, дозволяє AI-агенту надавати максимально релевантні та адаптовані консультації протягом усього періоду взаємодії.

На рисунку 4.6 представлено приклад інтелектуального консультування, де користувач ініціює запит до бота, вводячи текстове повідомлення (наприклад, "Привіт, підкажи мені про екіпірування"). Бот, використовуючи можливості інтелектуальної обробки природної мови та доступні дані профілю користувача, формує розгорнуту відповідь, яка надається у вигляді структурованого тексту.

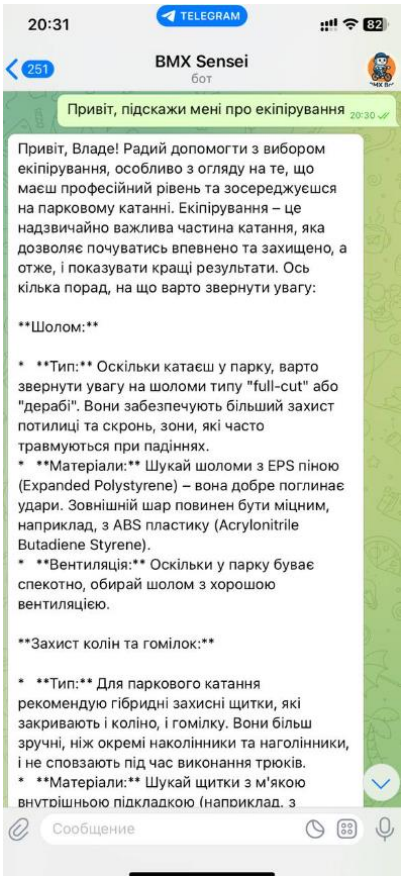

Рисунок 4.6 – Приклад персоналізованої консультації щодо екіпірування

Формат відповіді залишається послідовним, демонструючи здатність бота надавати вичерпну інформацію, розділяючи її на керовані частини. Такий підхід забезпечує читабельність та зручність сприйняття довгих відповідей у вікні чату Telegram. Завершується відповідь загальним висновком, який підкреслює індивідуальний підхід до вибору екіпірування.

На рисунку 4.7 демонструється функціонал розпізнавання та обробки голосових повідомлень. Користувач надсилає боту голосове повідомлення (наприклад, "Привіт, друже"), що відображається у чаті як іконка голосового повідомлення з тривалістю. Бот обробляє це голосове повідомлення, перетворюючи його на текст, та формує відповідь, що відображається у текстовому форматі. Це значно підвищує зручність взаємодії, дозволяючи користувачам спілкуватися з ботом природним чином, без необхідності

введення тексту вручну.

Рисунок 4.7 – Приклад обробки голосового повідомлення та текстової відповіді

ВИСНОВКИ

В рамках кваліфікаційної роботи було проведено дослідження предметної області, що є фундаментом для розробки AI-агента для персоналізованого консультування представників велосипедної спільноти. Це дослідження включало всебічний аналіз актуальних вимог та потреб користувачів, а також оцінку існуючих технологічних рішень для створення діалогових систем.

Зокрема, було детально вивчено принципи функціонування та архітектури сучасних інтелектуальних систем, здатних до обробки природної мови та взаємодії з користувачами. Проаналізовано ефективність різних підходів до формування баз знань та механізмів їх інтеграції з алгоритмами прийняття рішень. Особливу увагу було приділено особливостям розгортання та функціонування AI-агентів у середовищі популярних комунікаційних платформ, а також розглянуто стратегії забезпечення їх масштабованості та надійності. Це дозволило виявити ключові технічні та функціональні виклики, які необхідно було подолати під час розробки.

На основі проведеного аналізу було сформовано чітке уявлення про вимоги до AI-агента, який мав би не лише надавати інформацію, але й здійснювати персоналізоване консультування, враховуючи індивідуальні запити та потреби користувачів з велосипедної спільноти. Виходячи з цих вимог та з огляду на популярність та функціональність сучасних месенджер-платформ, було розроблено AI-агента на базі Telegram та мови програмування Python з використанням її спеціалізованих бібліотек. Вибір Telegram як платформи обґрунтований його широкою аудиторією, зручністю інтеграції та підтримкою різноманітного функціоналу для чат-ботів. Python же, з його багатим екосистемою бібліотек для обробки природної мови (NLP), машинного навчання та взаємодії з API, виявився оптимальним вибором для реалізації складної логіки агента.

Призначення розробленого AI-агента полягає у забезпеченні миттєвого

та персоналізованого доступу до релевантної інформації для представників велосипедної спільноти. Агент здатен надавати консультації з широкого спектру питань, таких як: вибір велосипеда та спорядження, маршрути для катання, правила дорожнього руху для велосипедистів, основи ремонту та обслуговування, інформація про локальні велоподії та спільноти.

Переваги розробленого AI-агента над існуючими рішеннями або традиційними способами отримання інформації полягають у його цілодобовій доступності, масштабованості, здатності обробляти велику кількість одночасних запитів та забезпечувати персоналізовану взаємодію без залучення людських ресурсів. Він усуває необхідність пошуку інформації на численних веб-ресурсах або очікування відповіді від адміністраторів спільнот. Завдяки використанню алгоритмів NLP, агент інтерпретує природні запити користувачів, надаючи точні та контекстуально релевантні відповіді, що значно покращує користувацький досвід.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Naveen K. AI Agents: Evolution, Architecture, and Real-World Applications [Електронний ресурс] – Режим доступу: https://doi.org/10.48550/arXiv.2503.12687 (дата звернення: 16.03.2025). – Назва з екрана.

2. Babak H. AI and Agents [Електронний ресурс] – Режим доступу: https://doi.org/10.1002/aaai.12170 (дата звернення: 04.04.2024). – Назва з екрана.

3. Penguy Z. An In-depth Survey of Large Language Model-based Artficial Inteligience Agents [Електронний ресурс] / Zijian J., Ning C. – Режим доступу: 2309.14365 (Дата звернення: 23.09.2023). – Назва з екрана.

4. Susanne M. Improving fluid intelligence with training on working memory [Електронний ресурс] – Режим доступу: Improving fluid intelligence with training on working memory | PNAS (дата звернення: 13.05.2008). – Назва з екрана.

5. Sunny K. AI Agents: Short-Term memory [Електронний ресурс] – Режим доступу: AI Agents: Short-Term vs. Long-Term Memory - Data Magic AI Blog (дата звернення: 27.02.2025). – Назва з екрана.

6. Feng Li Long Term Memory : The Foundation of AI Self-Evlolution [Електронний ресурс] – Режим доступу: Long Term Memory : The Foundation of AI Self-Evolution (дата звернення: 21.10.2024). – Назва з екрана.

7. Chad DeChant Episodic memory in AI agents poses risks that should be studied and mitigated [Електронний ресурс] – Режим доступу: https://doi.org/10.48550/arXiv.2501.11739 (дата звернення: 20.01.2025). – Назва з екрана.

8. Pycharm: the Python IDE for Professional Developers [Електронний ресурс] // JetBrains. – Режим доступу: PyCharm: The only Python IDE you need

9. Cole Stryker. What is a recurrent neural network? [Електронний

ресурс]. – Режим доступу: What is a Recurrent Neural Network (RNN)? | IBM (Дата звернення: 04.10.2024). – Назва з екрану.

10. What is LSTM – Long Short Term Memory? [Електронний ресурс]. - Режим доступу: What is LSTM – Long Short Term Memory? | GeeksforGeeks (Дата звернення: 28.05.2022). – Назва з екрану.

11. Cathrine Jeeva. Gated Recurrent Unit Networks. [Електронний ресурс]. - Режим доступу: Gated Recurrent Unit (GRU) - Scaler Topics (Дата звернення: 04.05.2023). – Назва з екрану.

12. Zoumana Kelta. An introduction to Convolutional Neural Networks (CNNs) [Електронний ресурс]. - Режим доступу: An Introduction to Convolutional Neural Networks: A Comprehensive Guide to CNNs in Deep Learning | DataCamp (Дата звернення: 14.11.2023). – Назва з екрану.

13. Apoorv Nandan. Automatic speech recognition with transformer [Електронний ресурс]. - Режим доступу: Automatic Speech Recognition with Transformer (Дата звернення: 13.01.2021). – Назва з екрану.

14. Anmol Gulati. Conformer: Convoluiton-augmented transformer for speech recognition // James Qin, Chung-Cheng Chiu, Niki Parmar, Wei Han. [Електронний ресурс]. - Режим доступу: [2005.08100] Conformer: Convolution-augmented Transformer for Speech Recognition (Дата звернення: 16.05.2020). – Назва з екрану.

15. Connectionist Temporal Classification [Електронний ресурс]. - Режим доступу: Connectionist Temporal Classification | GeeksforGeeks (Дата звернення: 27.12.2023). – Назва з екрану.

16. Murali Kartrick. Analysis of Multilingual Sequence-to-Sequence speech recognition systems / Takaaki Hori, Matthew Wiesner, Shiniji Watanabe. [Електронний ресурс]. - Режим доступу: [1811.03451] Analysis of Multilingual Sequence-to-Sequence speech recognition systems (Дата звернення: 07.11.2018). – Назва з екрану.

17. University of Toronto. RNN-T: RNN-Transducer [Електронний ресурс]. - Режим доступу: RNN-T: RNN Transducer (Дата звернення:

14.11.2012). – Назва з екрану.

18. Fabio D. Most popular Messaging Apps [Електронний ресурс] – Режим доступу: Most Popular Messaging Apps (2025) (Дата звернення: 24.04.2025). – Назва з екрану.

19. Telegram Bot Features [Електронний ресурс] // Telegram. – Режим доступу: Telegram Bot Features

20. Viber API Documentation [Електронний ресурс] // Viber. – Режим доступу: Viber REST API | Viber Developers Hub

21. Платформа WhatsApp Business [Електронний ресурс] // WhatsApp. – Режим доступу: Платформа WhatsApp Business

22. Платформа Messenger [Електроний ресурс] // Facebook. – Режим доступу: Платформа Messenger