**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ**

# КВАЛІФІКАЦІЙНА РОБОТА
# на тему: «ЧАТ-БОТ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЗАЦІЇ ОБСЛУГОВУВАННЯ КЛІЄНТІВ»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
*(код, найменування спеціальності)*
освітньо-професійної програми Інформаційні системи та технології
*(назва)*

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

_____ Максим ЦИБУЛЬКО
*(підпис)* *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: Максим ЦИБУЛЬКО
здобувач вищої освіти
група ІСД-42

Керівник: Вікторія ЖЕБКА
*науковий ступінь, вчене звання* д.т.н., професор

Рецензент:
*науковий ступінь, вчене звання*

**Київ 2025**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти бакалавр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

**ЗАТВЕРДЖУЮ**
Завідувач кафедрою ІПЗАС

_____ Каміла СТОРЧАК
«_____» _____ 2025 р.

**ЗАВДАННЯ**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Цибулько Максим Олегович
*(прізвище, ім'я, по батькові здобувача)*

1. Тема кваліфікаційної роботи: Чат-бот з елементами штучного інтелекту для автоматизації обслуговування клієнтів

керівник кваліфікаційної роботи Вікторія ЖЕБКА, д.т.н., професор,
*(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)*
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023р. №145

2. Строк подання кваліфікаційної роботи «__» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, чат-бот з елементами штучного інтелекту для автоматизації обслуговування клієнтів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Поставновка завдання: визначення потреби, вимог до функціональності та розроблення інтелектуального чат-бота для автоматизації клієнтського обслуговування.

Реалізація та тестування: розробка програмного коду, вибір технологічного стеку, збірка, тестування продуктивності та аналіз результатів.

Перелік графічного матеріалу: *презентація*

6. Дата видачі завдання «19» жовтня 2023 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Визначення потреб та вимог до чат-бота. Формулювання цілей та обґрунтування вибору технологій. | 19.10-05.11.25 | |
| 2 | Аналіз наукової та технічної літератури. Сучасні підходи до розробки інтелектуальних чат-ботів. | 05.11-12.11.25 | |
| 3 | Огляд, порівняння та аналіз технологій: Google Assistant, Google Search API, Telegram API, NLP-моделі. | 13.11-18.11.25 | |
| 4 | Розробка рекомендацій щодо створення та інтеграції інтелектуального чат-бота. | 19.11-23.11.25 | |
| 5 | Реалізація чат-бота та аналіз результатів тестування. | 24.11-03.12.25 | |
| 6 | Оформлення результатів дослідження. Написання та оформлення кваліфікаційної роботи. | 04.12-10.12.25 | |
| 7 | Розробка демонстраційних матеріалів | 11.12-20.12.25 | |
| 8 | Підготовка доповіді до захисту. | 21.12-29.12.25 | |

Здобувач вищої освіти _____ Максим ЦИБУЛЬКО
*(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Вікторія ЖЕБКА
*(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

# РЕФЕРАТ

Текстова частина бакалаврської роботи: 64 сторінки, 31 рисунок, 39 джерел.

*Об'єкт дослідження* – інтелектуальна інформаційна система для автоматизації обслуговування клієнтів на основі чат-бота.

*Предмет дослідження* – методи розробки та впровадження штучного інтелекту в чат-боти для підвищення ефективності автоматизованої взаємодії з користувачами.

*Мета роботи* – розробка та реалізація інтелектуального чат-бота, здатного автоматизувати процес обслуговування клієнтів, використовуючи технології штучного інтелекту, Google Assistant та Google Search API.

*Методи дослідження*:

1. Аналіз літературних джерел та нормативних документів – дослідження сучасних технологій розробки чат-ботів, штучного інтелекту та їх застосування у бізнесі.
2. Методи якісного та кількісного аналізу – вивчення алгоритмів обробки природної мови (NLP), моделей машинного навчання та оцінка їх ефективності у сфері автоматизації клієнтської підтримки.
3. Емпіричні методи – аналіз існуючих рішень у сфері чат-ботів, інтегрованих в бізнес-процеси компаній для обслуговування клієнтів.
4. Методи моделювання – розробка концептуальної, логічної та програмної архітектури чат-бота, його тестування на працездатність та ефективність.
5. Порівняльний аналіз – оцінка можливостей та функціональності різних платформ для розробки чат-ботів (Google Assistant, Google Search API, Telegram API, Dialogflow тощо).

*Галузь використання* – автоматизовані системи взаємодії з клієнтами у сферах електронної комерції, фінансових послуг, логістики, технічної підтримки та інших сервісних галузях.

КЛЮЧОВІ СЛОВА: ШТУЧНИЙ ІНТЕЛЕКТ, АВТОМАТИЗАЦІЯ, GOOGLE ASSISTANT, GOOGLE SEARCH API, МОДЕЛЮВАННЯ, КЛІЄНТСЬКЕ ОБСЛУГОВУВАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ

## ABSTRACT

*Text part of the Bachelor's thesis:* 64 pages, 31 figures, 39 sources.

*Object of research* – an intelligent information system for automating customer service based on a chatbot.

*Subject of research* – methods of developing and implementing artificial intelligence in chatbots to improve the efficiency of automated user interaction.

*Aim of the thesis* – to develop and implement an intelligent chatbot capable of automating the customer service process using artificial intelligence technologies, Google Assistant, and the Google Search API.

*Research methods:*

1. Analysis of literary sources and regulatory documents – studying modern technologies for chatbot development, artificial intelligence, and their application in business.
2. Qualitative and quantitative analysis methods – examination of natural language processing (NLP) algorithms, machine learning models, and evaluation of their effectiveness in the field of customer support automation.
3. Empirical methods – analysis of existing chatbot solutions integrated into business processes for customer service.
4. Modeling methods – development of the conceptual, logical, and software architecture of the chatbot, testing its performance and efficiency.
5. Comparative analysis – evaluation of the capabilities and functionality of various chatbot development platforms (Google Assistant, Google Search API, Telegram API, Dialogflow, etc.).

*Field of application* – automated customer interaction systems in the fields of e-commerce, financial services, logistics, technical support, and other service industries.



**KEYWORDS:** ARTIFICIAL INTELLIGENCE, AUTOMATION, GOOGLE ASSISTANT, GOOGLE SEARCH API, MODELING, CUSTOMER SERVICE, NATURAL LANGUAGE PROCESSING

**ЗМІСТ**

# ВСТУП

*Актуальність дослідження.* У сучасному світі автоматизація обслуговування клієнтів стає критично важливою складовою ефективного бізнесу. З розвитком цифрових технологій компанії прагнуть зменшити витрати на персонал, підвищити якість сервісу та забезпечити безперервний зв'язок із клієнтами. Одним із найбільш перспективних рішень для цього є інтелектуальні чат-боти, які можуть автоматично відповідати на запити користувачів, виконувати типові завдання та надавати персоналізовані консультації. Зростання попиту на інтелектуальних віртуальних помічників спостерігається у сферах електронної комерції, фінансових послуг, технічної підтримки, логістики та інших сервісних індустрій. Використання чат-ботів дозволяє підвищити рівень взаємодії з клієнтами, зменшити навантаження на працівників та підвищити оперативність обслуговування. Однак, незважаючи на значний розвиток технологій, існує низка викликів, пов'язаних із коректністю розпізнавання мовлення, підтримкою різних мов, інтеграцією в бізнес-процеси та забезпеченням безпеки даних.

Тому тема дослідження, пов'язана з розробкою інтелектуального чат-бота для автоматизації обслуговування клієнтів, є актуальною та має практичне значення. У роботі буде проведено аналіз сучасних технологій, розроблено архітектуру чат-бота, а також реалізовано програмний прототип, що демонструє можливості автоматизації взаємодії з користувачами.

*Предмет дослідження* – методи розробки та впровадження штучного інтелекту в чат-боти для підвищення ефективності автоматизованої взаємодії з користувачами.

*Мета роботи* – розробка та реалізація інтелектуального чат-бота, здатного автоматизувати процес обслуговування клієнтів, використовуючи технології штучного інтелекту, Google Assistant та Google Search API.

*Наукові завдання:*

– аналіз сучасних технологій та методів створення інтелектуальних чат-ботів, зокрема технологій обробки природної мови (NLP), машинного навчання та штучного інтелекту;

– дослідження платформ та інструментів для розробки чат-ботів, таких як google assistant, google search api, telegram api, dialogflow тощо;

– визначення ключових вимог до системи чат-бота, враховуючи її функціональність, швидкість обробки запитів, інтеграцію з зовнішніми сервісами та зручність використання для клієнтів;

– оцінка ефективності роботи розробленого чат-бота за допомогою тестування та аналізу його продуктивності;

– проведення порівняльного аналізу різних підходів до реалізації чат-ботів та визначення оптимального рішення для автоматизації обслуговування клієнтів;

– формулювання рекомендацій щодо впровадження інтелектуальних чат-ботів у бізнес-процеси для покращення якості обслуговування клієнтів.

*Методи дослідження*:

1. Аналіз літературних джерел та нормативних документів – дослідження сучасних технологій розробки чат-ботів, штучного інтелекту та їх застосування у бізнесі.

2. Методи якісного та кількісного аналізу – вивчення алгоритмів обробки природної мови (NLP), моделей машинного навчання та оцінка їх ефективності у сфері автоматизації клієнтської підтримки.

3. Методи моделювання – розробка концептуальної, логічної та програмної архітектури чат-бота, його тестування на працездатність та ефективність.

4. Порівняльний аналіз – оцінка можливостей та функціональності різних платформ для розробки чат-ботів (Google Assistant, Google Search API, Telegram API, Dialogflow тощо).

# 1 ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ ЧАТ-БОТІВ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

## 1.1 Інтернет-комунікаційні технології та їх застосування в чат-ботах

Ефективна робота сучасного чат-бота неможлива без надійного технологічного підґрунтя, яке забезпечує обмін інформацією між користувачем, ботом та зовнішніми сервісами. Саме інтернет-комунікаційні технології формують основу функціонування чат-ботів: вони забезпечують доставку повідомлень, обробку запитів у режимі реального часу, з'єднання з базами даних, API, хмарними платформами тощо. Ці технології є сполучною ланкою між клієнтським інтерфейсом і програмною логікою чат-бота, дозволяючи створювати інтуїтивно зрозумілий і швидкий канал спілкування.

Основними інструментами, що забезпечують комунікацію, є протоколи HTTP(S), WebSocket, REST і GraphQL API, а також хмарні сервіси на кшталт Firebase, Google Cloud Functions, AWS Lambda тощо. Завдяки їм можлива інтеграція чат-бота з месенджерами (Telegram, Facebook Messenger, Viber), голосовими асистентами (Google Assistant, Alexa), CRM-системами, базами знань та іншими корпоративними ресурсами [1].

Крім того, широке застосування отримали технології webhook та polling, які забезпечують зв'язок між клієнтським інтерфейсом (месенджером або сайтом) та серверною частиною чат-бота. Webhook дозволяє миттєво реагувати на події, зменшуючи затримку між дією користувача та відповіддю системи. Polling, у свою чергу, використовується там, де вебхуки недоступні або непрактичні.

У цьому підрозділі буде детально розглянуто ключові інтернет-комунікаційні технології, які застосовуються у розробці чат-ботів, особливості їхньої роботи, переваги та недоліки, а також практичні приклади реалізації таких рішень у реальних проєктах.

Application Programming Interface (API) – це програмний інтерфейс, який дозволяє двом програмам взаємодіяти між собою. У контексті чат-ботів API виступає як "мова спілкування" між ботом і зовнішніми сервісами, такими як месенджери (Telegram API), платформи машинного навчання (OpenAI API), бази даних або CRM-системи.

За допомогою API чат-бот може:

– надсилати та отримувати повідомлення в месенджері;

– отримувати доступ до зовнішніх даних (наприклад, курс валют, погода);

– здійснювати аутентифікацію користувача;

– виконувати логіку взаємодії з бекендом системи (наприклад, перевірку наявності товару в інтернет-магазині).

API спрощує інтеграцію з різними сервісами та дозволяє будувати масштабовану систему, де кожен компонент відповідає за свою частину функціоналу.

Representational State Transfer (REST) – це архітектурний стиль, який використовується для створення API. REST-архітектура базується на використанні HTTP-запитів (GET, POST, PUT, DELETE) для взаємодії між клієнтом (ботом) і сервером [1].

REST API є одним з найпоширеніших способів обміну даними у чат-ботах. Наприклад:

– бот може надсилати GET-запит, щоб отримати список товарів із сайту;

– використовувати POST-запит, щоб зареєструвати нового користувача в системі;

– PUT або DELETE-запити – для оновлення або видалення даних.

REST є зручним, зрозумілим і легко реалізується в більшості мов програмування. Саме тому його найчастіше використовують для взаємодії з серверними частинами чат-ботів або сторонніми API, такими як OpenWeather, Google Search API, Firebase тощо.

Webhook – це механізм, який дозволяє серверу надсилати повідомлення (або інші дані) до чат-бота автоматично при настанні певної події. На відміну від REST API, де бот активно надсилає запити до сервера, webhook працює у зворотному напрямку: сервер сам надсилає дані боту, коли це потрібно [2].

Приклад використання webhook:

– коли користувач надсилає повідомлення в Telegram-бот, Telegram надсилає запит на сервер розробника через webhook;

– цей запит містить усю інформацію про повідомлення, і сервер одразу може на нього відреагувати (наприклад, відповісти або записати лог).

Webhook дозволяє скоротити затримки у відповіді та зменшує навантаження на сервер, оскільки немає потреби постійно перевіряти наявність нових повідомлень (як це відбувається при використанні polling).

WebSocket – це протокол, який забезпечує двостороннє з’єднання між клієнтом і сервером у реальному часі. На відміну від HTTP, де кожен запит ініціюється окремо, WebSocket дозволяє встановити одне з’єднання і використовувати його для постійного обміну даними.

У чат-ботах WebSocket застосовується для:

– онлайн-чату в реальному часі;

– інтеграції з вебінтерфейсами або мобільними додатками;

– реалізації функцій миттєвого сповіщення.

Хоча Telegram і більшість месенджерів використовують webhook, WebSocket корисний у власних чат-системах або тоді, коли потрібна мінімальна затримка та висока швидкість передачі інформації (наприклад, в ігрових ботах, торгових платформах, сервісах підтримки в режимі live).

Інтеграція чат-ботів за допомогою інтернет-комунікаційних технологій:

API забезпечує стандартний інтерфейс, через який бот може отримувати й передавати інформацію до зовнішніх платформ. Наприклад, за допомогою Telegram Bot API, бот може надсилати повідомлення користувачеві, обробляти кнопки, медіафайли чи команди. Крім того, API сторонніх сервісів – таких як Google Maps API або OpenWeather API – дозволяють чат-боту надавати корисну

інформацію в діалозі з користувачем (наприклад, показати погоду або прокласти маршрут) [3].

REST API конкретизує спосіб доступу до ресурсів (даних або функцій), який є уніфікованим, передбачуваним та легко реалізується. Наприклад, при замовленні товару через чат-бот користувач заповнює форму, а бот надсилає POST-запит до сервера інтернет-магазину для збереження замовлення. Коли користувач хоче переглянути статус замовлення – виконується GET-запит. REST дозволяє реалізувати інтеграцію з внутрішніми сервісами компанії (бухгалтерія, логістика, підтримка) у вигляді простої структури запитів [3].

Webhook забезпечує миттєву реакцію бота на події у зовнішньому середовищі. Це особливо важливо для інтеграції з месенджерами, такими як Telegram чи Viber. Наприклад, коли користувач надсилає повідомлення в Telegram, платформа миттєво передає запит на сервер бота через webhook. Сервер обробляє цей запит, виконує відповідну дію (наприклад, звертається до бази даних) і формує відповідь, яка надсилається назад користувачу. Таким чином, webhook діє як “тригер”, що запускає логіку обробки повідомлень.

WebSocket застосовується там, де важлива миттєвість та постійне з’єднання – наприклад, у власних корпоративних чатах або веб-застосунках, що вимагають live-підтримки. За допомогою WebSocket чат-бот може отримувати дані без затримки, підтримувати реальний діалог, відображати статуси в реальному часі, передавати оновлення без перезавантаження сторінки. Це дозволяє створювати інтерактивні інтерфейси та забезпечує зручність для кінцевого користувача.

### 1.2 Методи та технології створення інтелектуальних чат-ботів

Створення інтелектуальних чат-ботів – складний, багаторівневий процес, який поєднує в собі сучасні досягнення у галузі штучного інтелекту, машинного навчання, обробки природної мови (NLP), а також використання програмних фреймворків, сервісів і хмарних платформ. Головна мета такого чат-бота – не просто реагувати на заздалегідь прописані ключові слова, а розуміти контекст,

інтерпретувати наміри користувача, адаптуватися до нових сценаріїв і навчатися з часом.

В основі інтелектуального чат-бота лежать моделі машинного навчання, які дозволяють розпізнавати текст, класифікувати запити, формувати відповіді, а також синтезувати діалог. Найбільш поширеним напрямом у цьому контексті є використання алгоритмів Natural Language Processing (NLP), що дозволяють боту розуміти людську мову на глибшому рівні. Такі методи, як tokenization (розбиття тексту на складові), named entity recognition (розпізнавання іменованих сутностей), sentiment analysis (аналіз емоційного забарвлення), є ключовими для побудови гнучкої логіки спілкування.

Залежно від рівня складності та цілей, розробники обирають різні платформи та підходи до реалізації чат-бота. Серед найпопулярніших технологій — використання готових фреймворків, таких як Dialogflow від Google, Rasa (відкритий фреймворк з підтримкою NLP та ML), Microsoft Bot Framework, а також інтеграція з LLM-моделями на кшталт GPT-4 через OpenAI API. Ці інструменти значно скорочують час розробки, надаючи готові рішення для обробки мовлення, збереження сесій, налаштування інтентів та сценаріїв [4].

На рис. 1.8 зображено стандартну архітектуру, яка використовується під час створення інтелектуальних чат-ботів. Взаємодія починається з користувача, який надсилає повідомлення через месенджер (наприклад, Telegram). Це повідомлення надходить до сервера бота через Webhook або API Gateway. Далі повідомлення обробляється одним або кількома модулями: NLP-модулем (наприклад, Dialogflow або Rasa) – для розпізнавання інтенції та сутностей, ядром бізнес-логіки – для прийняття рішень, або ж взаємодії з базою даних чи зовнішніми сервісами (наприклад, Google Search API, OpenWeather). Після обробки запиту, система формує відповідь і надсилає її назад до користувача.

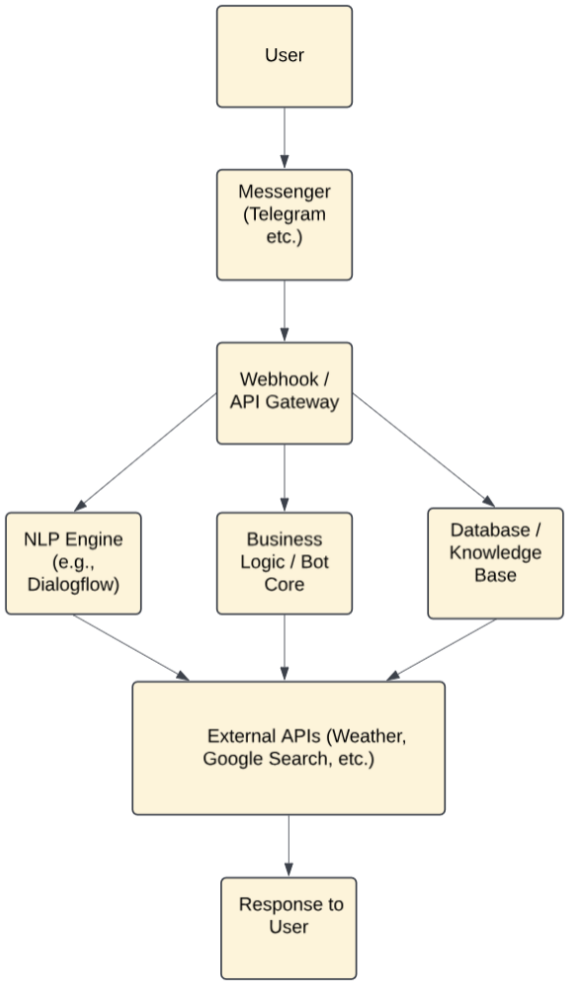



Рисунок 1.8 Типова архітектуру інтелектуального чат-бота

Найпопулярніші рішення фреймворків для створення інтелектуальних чат-ботів:

spaCy – одна з найшвидших та найефективніших бібліотек для обробки природної мови (NLP) на мові Python. Вона дозволяє виконувати такі завдання, як токенізація, лематизація, визначення частин мови, розпізнавання іменованих сутностей (NER), а також побудова залежностей між словами в реченні. У контексті чат-ботів spaCy часто використовується для попередньої обробки тексту перед передачею в ML-моделі або системи класифікації намірів (intent recognition) [5].

Rasa – фреймворк з відкритим кодом для створення AI-ботів з використанням NLP і машинного навчання. Він складається з двох основних компонентів: Rasa NLU (Natural Language Understanding) – для розпізнавання намірів і сутностей у

повідомленнях користувача, та Rasa Core – для побудови діалогової логіки на основі історії спілкування. Rasa є чудовим вибором для тих, хто хоче мати повний контроль над своїм ботом, з можливістю навчання на власних даних, без використання хмарних платформ.

Dialogflow – хмарна платформа від Google, яка забезпечує зручний інтерфейс для побудови чат-ботів із підтримкою природної мови. Вона дозволяє швидко створювати інтенти, інтегрувати різні платформи (Telegram, Viber, Google Assistant, Slack тощо), підключати fulfillment через webhook і використовувати Google Cloud для зберігання даних. Dialogflow особливо зручний для розробників, які бажають швидко реалізувати функціональність без глибоких знань у галузі машинного навчання [6].

GPT (Generative Pre-trained Transformer) – сімейство великих мовних моделей (LLM), розроблених OpenAI, серед яких найвідоміші GPT-3 та GPT-4. Ці моделі забезпечують високоякісне генеративне спілкування та можуть бути інтегровані у чат-ботів через API. GPT-моделі не потребують ручного створення інтентів або сценаріїв – вони навчені на великому обсязі тексту і здатні відповідати на запити контекстуально та граматично правильно. Вони особливо корисні у випадках, коли необхідна імпровізація, нестандартні відповіді або багатотематичне спілкування.

Microsoft Bot Framework – платформа для створення чат-ботів, яка забезпечує розробку, тестування, розгортання та підключення ботів до різноманітних каналів (Skype, Telegram, Teams, Web Chat). Вона добре інтегрується з Azure Cognitive Services, включаючи Language Understanding (LUIS), що дозволяє будувати інтелектуальні рішення на базі Microsoft-екосистеми.

Flask – мікрофреймворк для Python, який вирізняється мінімалізмом та легкістю. Він ідеально підходить для створення невеликих вебсервісів, REST API та webhook-серверів, які потрібні для інтеграції чат-бота з месенджерами, такими як Telegram чи Facebook Messenger. За допомогою Flask легко реалізувати обробку запитів від Telegram API, здійснити маршрутизацію повідомлень, виклик NLP-функцій або з’єднання з базою даних. Через свою простоту Flask дозволяє швидко

прототипувати рішення і виводити їх у production з мінімальними витратами ресурсів [7].

Django – повноцінний фреймворк з широким функціоналом для побудови складних веб-додатків. Він містить вбудовані модулі для роботи з базами даних, авторизації користувачів, адміністрування, створення REST API (через Django REST Framework) та захисту даних. Django буде корисним у проєктах, де чат-бот є частиною більшої системи (наприклад, CRM, портал замовлень або освітня платформа), або коли потрібно реалізувати масштабований, безпечний бекенд із розмежуванням прав доступу та збереженням історії взаємодій [8].

У межах даного проєкту Django/Flask будуть використані для реалізації:

– обробки повідомлень, отриманих через Telegram Webhook;
– з'єднання з NLP-модулями (наприклад, spaCy або GPT);
– обміну даними з базою знань або іншими API;
– логіки відповідей залежно від намірів користувача.

### 1.3 Використання Google Assistant для інтеграції штучного інтелекту

Google Assistant є одним із найпотужніших інтелектуальних голосових асистентів сучасності, що активно застосовується у сфері автоматизації комунікації з користувачами. Його застосування в проєктах, пов’язаних із чат-ботами, відкриває нові перспективи у створенні голосових інтерфейсів, які дозволяють взаємодіяти з ботом у природній, розмовній формі. Завдяки використанню алгоритмів обробки природної мови (NLP), Google Assistant здатен розпізнавати інтенції користувача, класифікувати запити, а також адаптувати відповіді до контексту діалогу. У процесі взаємодії він може аналізувати як голосові, так і текстові повідомлення, що забезпечує універсальність і зручність у використанні на різних типах пристроїв [9].

Однією з ключових переваг використання Google Assistant є можливість створення власних сценаріїв взаємодії через платформу Actions on Google. Це дозволяє розробникам створювати індивідуальні функції, які активуються

голосовими командами, і налаштовувати ботів для виконання конкретних завдань – наприклад, бронювання, пошук інформації, надання рекомендацій тощо. Завдяки тісній інтеграції з іншими сервісами Google, такими як Google Search, Gmail, Calendar, Maps, значно розширюються функціональні можливості чат-бота. Крім того, Google Assistant підтримує багатомовність, що є важливою перевагою для реалізації ботів, орієнтованих на широку аудиторію.

Втім, використання Google Assistant також має низку обмежень. По-перше, система є частиною закритої екосистеми Google, що накладає певні технічні та юридичні обмеження на гнучкість і адаптивність рішення. Наприклад, створення і публікація користувацьких дій потребує проходження обов'язкової модерації зі сторони Google. По-друге, структура діалогів повинна відповідати суворим вимогам платформи, що може обмежити свободу побудови складних сценаріїв і розгалужених логічних структур. Ще одним суттєвим обмеженням є орієнтованість платформи на голосову взаємодію — на багатьох пристроях із Google Assistant відсутні екрани, тому можливості для візуального представлення інформації суттєво знижуються. Це потребує особливої уваги до формулювання відповідей, які повинні бути лаконічними, інформативними та легко сприйманими на слух.

Зважаючи на наведене, Google Assistant доцільно використовувати як один із каналів взаємодії з користувачем у складі багатокомпонентної системи чат-бота. Його функціональність може бути ефективно доповнена окремими серверними модулями, реалізованими за допомогою відкритих фреймворків та бібліотек, що забезпечить ширшу кастомізацію, розширюваність та можливість інтеграції зі сторонніми системами [10].

У процесі взаємодії з чат-ботом на базі Google Assistant користувач ініціює діалог шляхом голосового або текстового запиту. Цей запит надходить до Google Assistant, який обробляє його за допомогою вбудованих алгоритмів обробки природної мови. Далі запит передається до модуля розпізнавання наміру (NLP), який визначає, що саме хотів сказати користувач і яку дію слід виконати. Визначений намір активує відповідну "дію" (Action) у системі Actions on Google –

це набір інструкцій, які можуть включати доступ до зовнішніх API, баз даних або логічних обчислень (рис. 1.9).

Після виконання відповідної дії система формує відповідь, яка передається назад через Google Assistant до користувача у зручному форматі – як правило, голосовому або текстовому, залежно від пристрою. Така архітектура забезпечує швидкий, природний та інтуїтивно зрозумілий спосіб взаємодії, а також дозволяє гнучко масштабувати функціонал чат-бота під конкретні потреби сервісу [11].

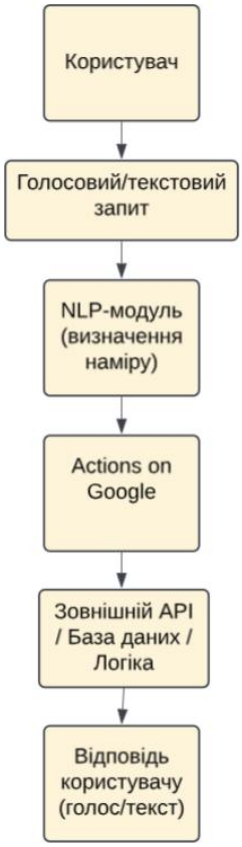



Рисунок 1.9 – Схема взаємодії користувача з Google Assistant у процесі роботи чат-бота

## 1.4 Основні принципи роботи Google Search API та його роль у чат-ботах

Одним із ефективних способів розширення можливостей інтелектуального чат-бота є інтеграція з Google Search API, який дозволяє здійснювати пошук інформації в Інтернеті безпосередньо із середовища бота. Це відкриває можливість

оперативного надання користувачу актуальних відповідей, посилань, статей, зображень або іншого контенту, що знаходиться за межами бази знань самого бота.

Принцип роботи Google Search API базується на передачі запитів до Google Custom Search Engine (CSE) – спеціалізованого механізму, який дозволяє виконувати пошук в Інтернеті або у межах обмеженого переліку сайтів. Для взаємодії з цим сервісом бот надсилає HTTP-запит до певної URL-адреси з параметрами (ключовими словами, мовою, кількістю результатів тощо). У відповідь API повертає структуровані дані у форматі JSON, які містять результати пошуку – з назвами, описами, посиланнями, мініатюрами тощо. Завдяки цьому бот може обробити ці дані, відформатувати їх і надіслати користувачу у вигляді зручного повідомлення [12].

Інтеграція Google Search API у логіку чат-бота дозволяє значно підвищити інформативність та універсальність відповіді. Наприклад, якщо бот отримує запит, на який немає готової відповіді в його базі знань, він може автоматично сформулювати пошуковий запит, надіслати його до Google API, проаналізувати перші результати й надати користувачу релевантну відповідь з посиланням на джерело. Такий підхід дозволяє реалізувати динамічне отримання контенту, зменшує залежність від обмеженого набору сценаріїв та дозволяє реагувати на більш широкий спектр питань.

Google Search API може бути використаний для тематичного або фільтрованого пошуку – наприклад, лише в межах сайту компанії, бази знань, технічної документації або офіційних джерел. Це забезпечує точніші та надійніші відповіді, знижує ризик помилкової інформації та покращує загальний користувацький досвід.

Таким чином, Google Search API виступає як потужний інструмент для підтримки чат-ботів у ситуаціях, коли потрібна миттєва, актуальна та релевантна інформація, яку складно зберігати локально. Його використання робить чат-бота значно гнучкішим, більш інтелектуальним і здатним до самостійного пошуку відповідей у динамічному середовищі [12].

На рисунку 1.10 зображено типовий сценарій, за яким чат-бот інтегрується з Google Search API для обробки складних або нестандартних запитів. Коли користувач надсилає запит, система спочатку перевіряє локальну базу знань або заздалегідь прописані сценарії. Якщо відповідь не знайдено, бот формує динамічний пошуковий запит і надсилає його до Google Search API. Отримані результати у форматі JSON обробляються та форматуються у зрозумілу відповідь, яка повертається користувачеві через інтерфейс бота. Такий підхід дозволяє забезпечити розширену функціональність без збільшення внутрішньої бази знань.

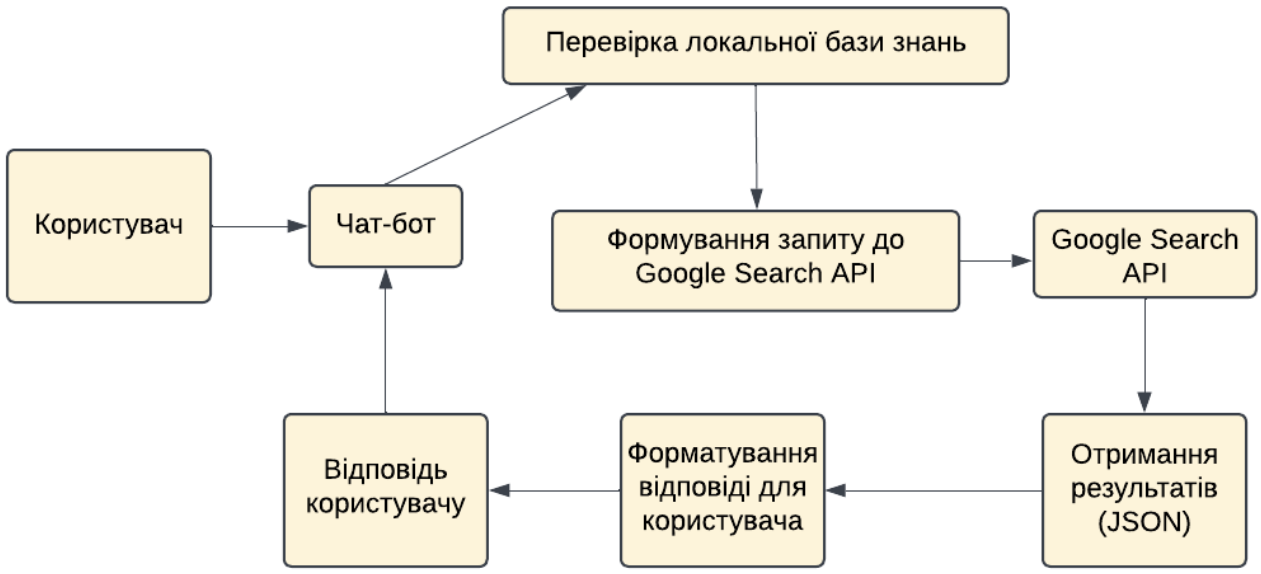



Рисунок 1.10 – Схема використання Google Search API у чат-боті для динамічного пошуку інформації

# 2 РОЗРОБКА АЛГОРИТМУ ТА МОДЕЛЮВАННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ОБСЛУГОВУВАННЯ КЛІЄНТІВ

## 2.1 Постановка проблеми, мета та завдання дослідження

В умовах стрімкого розвитку цифрових технологій компанії дедалі частіше стикаються з необхідністю забезпечення швидкої, зручної та безперервної комунікації з клієнтами. Традиційні канали зв’язку, як-от електронна пошта або кол-центри, часто виявляються недостатньо ефективними через обмежену швидкість реагування, високі витрати на персонал та низьку масштабованість. Водночас зростає попит на автоматизовані рішення, здатні забезпечувати якісну взаємодію з клієнтами в режимі реального часу.

Одним із найбільш перспективних напрямів вирішення цієї проблеми є впровадження чат-ботів із елементами штучного інтелекту. Такі системи можуть ефективно автоматизувати відповіді на типові запити, надавати консультації, супроводжувати користувача під час вибору товару або послуги, а також адаптуватися до специфіки конкретного бізнесу. Проте розробка ефективного інтелектуального чат-бота потребує вирішення низки технічних і концептуальних задач, серед яких – обробка природної мови (Natural Language Processing, NLP), вибір моделей машинного навчання, інтеграція зі сторонніми сервісами (наприклад, Google Assistant або Google Search API), а також забезпечення точності та безпеки обробки даних [13].

Таким чином, постає проблема створення інтелектуального чат-бота, який би не лише автоматизував обслуговування клієнтів, але й мав адаптивну, масштабовану архітектуру, був зручним у використанні, а також забезпечував високу якість взаємодії.

Мета дослідження полягає у розробці та реалізації інтелектуального чат-бота, здатного автоматизувати процес обслуговування клієнтів шляхом інтеграції

сучасних технологій штучного інтелекту, таких як Google Assistant, Google Search API, а також засобів обробки природної мови.

Для досягнення поставленої мети необхідно виконати такі наукові завдання:

– здійснити аналіз сучасних технологій та методів створення інтелектуальних чат-ботів;

– дослідити можливості використання NLP та машинного навчання у сфері автоматизованого обслуговування;

– обґрунтувати вибір технологічного стеку для реалізації системи (мови програмування, API, фреймворки);

– розробити архітектурну модель чат-бота, що забезпечує інтеграцію зі сторонніми сервісами;

– реалізувати функціональний прототип чат-бота з використанням обраних технологій;

– провести тестування розробленої системи, оцінити її ефективність та зручність для користувачів;

– сформулювати рекомендації щодо впровадження подібних рішень у бізнес-процеси різних галузей.

Інтелектуальний чат-бот, який розробляється в межах цієї роботи, є не лише інструментом автоматизації клієнтського обслуговування, а й багатофункціональним цифровим помічником з розширеними можливостями. Його потенціал виходить далеко за межі простого обміну повідомленнями. Такий бот може бути цікавим і корисним з кількох причин:

– персоналізація обслуговування, бот може запам’ятовувати попередні запити користувача, адаптувати відповіді під його потреби та надавати більш релевантну інформацію з часом, створюючи ефект "людського" спілкування.

– підтримка 24/7 без втрати якості, на відміну від живих операторів, бот доступний цілодобово, що особливо важливо для глобальних компаній або клієнтів у різних часових зонах.

– можливість навчання та самовдосконалення, завдяки інтеграції моделей машинного навчання бот може вдосконалювати свої відповіді, вивчаючи нові шаблони спілкування та реагуючи на зміну контексту.

– голосове керування та мультимодальність, через інтеграцію з Google Assistant бот зможе взаємодіяти з користувачами не лише через текст, а й через голосові команди, що значно розширює його аудиторію (наприклад, люди з обмеженими можливостями).

– пошук інформації в реальному часі, завдяки Google Search API бот може надавати актуальні відповіді на запити користувачів, шукаючи інформацію в інтернеті під час діалогу.

– простота масштабування, чат-бот легко адаптується під нові задачі або галузі – достатньо змінити базу знань або налаштувати додаткові сценарії поведінки, без необхідності глобальної переробки всієї системи.

– аналітика та зворотний зв'язок, бот може збирати статистику взаємодії, виявляти слабкі місця у процесах обслуговування, що допомагає бізнесу приймати обґрунтовані рішення щодо покращення сервісу .

– можливість інтеграції з CRM, ERP та іншими системами, завдяки відкритим API, бот може бути легко вбудований у внутрішню інфраструктуру компанії, автоматизуючи ще більше процесів – від обробки замовлень до підтримки клієнтів [14].

## 2.2 Аналіз ринку сучасних чат-ботів та їх роль в автоматизації обслуговування клієнтів

Ринок чат-ботів стрімко зростає та трансформується під впливом розвитку технологій штучного інтелекту, машинного навчання та обробки природної мови. Все більше компаній інтегрують чат-боти у свої бізнес-процеси з метою оптимізації обслуговування клієнтів, підвищення рівня доступності сервісу та зниження витрат на персонал. Сучасні чат-боти здатні не лише відповідати на стандартні запити, а й вести складні діалоги, здійснювати пошук інформації, інтегруватися з зовнішніми

системами (CRM, ERP), а також персоналізувати взаємодію з користувачем. Це робить їх важливим інструментом цифрової трансформації бізнесу та одним із ключових елементів автоматизації клієнтської підтримки.

Аналіз існуючих аналогів є важливим етапом перед початком розробки будь-якого програмного рішення, зокрема інтелектуального чат-бота. Такий аналіз дозволяє вивчити найкращі практики, виявити сильні та слабкі сторони конкурентних рішень, оцінити технологічні підходи, які вже довели свою ефективність на ринку. Завдяки цьому розробник може уникнути типових помилок, обґрунтовано обрати архітектуру системи, сформувати чіткі вимоги до функціональності та знайти потенційні напрямки для вдосконалення або інновацій. Крім того, аналіз аналогів допомагає зорієнтуватися у трендах галузі, визначити очікування кінцевих користувачів і забезпечити конкурентоспроможність майбутнього продукту. У підсумку, це сприяє економії часу й ресурсів під час розробки та підвищує ймовірність створення успішного і затребуваного рішення [15].

Модель ChatGPT від компанії OpenAI набула популярності не лише як веб-сервіс, а й як основа для створення розумних чат-ботів у месенджерах, зокрема Telegram (рис. 1.1). Сотні незалежних розробників створюють ботів, що використовують API ChatGPT для надання консультацій, генерації текстів, навчання, технічної підтримки тощо. Такі боти здатні вести логічні й контекстуальні діалоги, пам’ятати попередні повідомлення та адаптувати відповіді до стилю спілкування користувача. Telegram у цьому випадку виконує роль зручного середовища комунікації, що дозволяє швидко запускати ботів, тестувати їхню роботу та масштабувати аудиторію без потреби у розробці окремого застосунку. Завдяки відкритому Telegram API та підтримці Webhook, розробники можуть реалізовувати інтерактивні інтерфейси з кнопками, меню та формами. ChatGPT-боти в Telegram стали популярними в освіті, бізнесі, копірайтингу, маркетингу й навіть у сфері ментального здоров’я [15].

Рисунок 1.1 – Логотип чат боту ChatGPT

Erica – один із найуспішніших прикладів використання штучного інтелекту в банківському секторі. Вперше запущений у 2018 році, цей віртуальний помічник обслуговує понад 25 мільйонів клієнтів (рис. 1.2). Erica доступна в мобільному додатку банку та дозволяє користувачам здійснювати банківські операції, перевіряти баланс, оплачувати рахунки, отримувати персоналізовані фінансові поради й рекомендації щодо економії коштів. Основною особливістю Erica є її інтеграція зі штучним інтелектом, що дозволяє системі розуміти природну мову, обробляти голосові команди та вчитися на основі запитів користувачів. Попри те, що Erica не працює в Telegram або інших відкритих месенджерах, її архітектура демонструє важливий приклад побудови спеціалізованого чат-бота, який повністю інтегрований у внутрішню інфраструктуру великої фінансової організації.

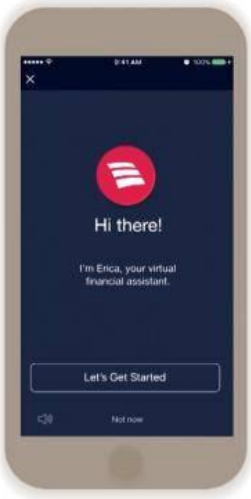



Рисунок 1.2 – чат бот Erica в банківській сфері

Buoy Health – це інтелектуальний чат-бот, який допомагає користувачам визначити, до якого лікаря звернутися або які дії вжити на основі описаних симптомів (рис. 1.3). Сервіс використовує обробку природної мови (NLP) і медичні алгоритми, розроблені спільно з лікарями, щоб забезпечити точні, безпечні рекомендації. Buoy став особливо популярним під час пандемії COVID-19, коли користувачі потребували попередньої оцінки свого стану без фізичного відвідування медичних установ. Бот працює через веб-інтерфейс, але з технічної точки зору його функціонал можна реалізувати і в Telegram, що відкриває можливості для створення медичних помічників у популярних месенджерах. Buoy Health є прикладом спеціалізованого чат-бота, орієнтованого на безпечну, етичну та корисну взаємодію з користувачем у сфері охорони здоров'я.

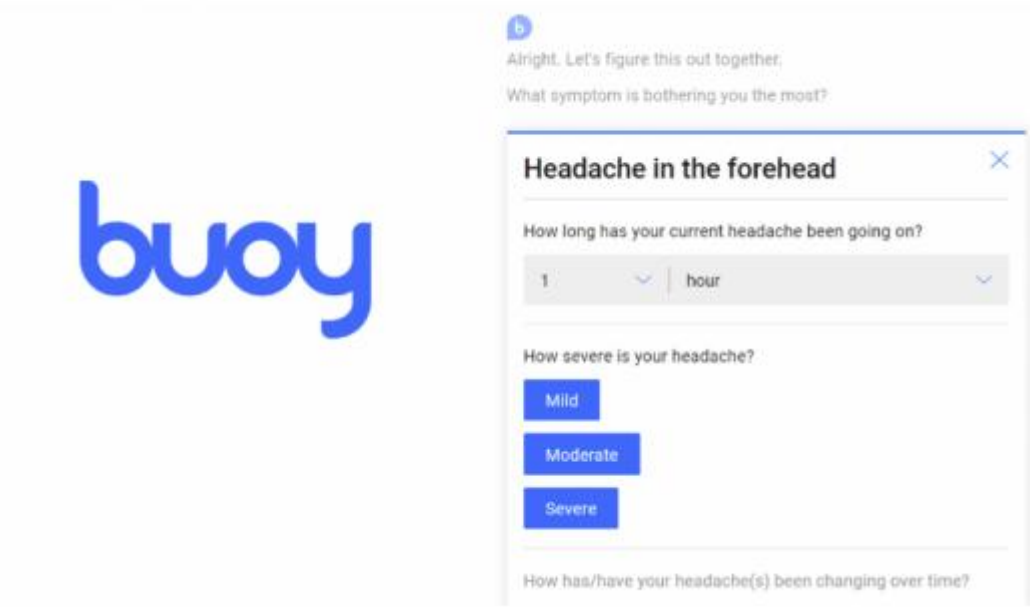



Рисунок 1.3 Логотип Buoy Health

Watson Assistant – платформа для створення інтелектуальних чат-ботів і голосових помічників, розроблена IBM (рис. 1.4). Вона орієнтована переважно на великі компанії й підтримує складні сценарії спілкування, розгалужену логіку, обробку природної мови, інтеграцію з CRM, ERP та сторонніми API. Watson Assistant може бути розгорнутий як у внутрішніх каналах компанії (на вебсайті, в

додатках), так і у месенджерах, включаючи Telegram. Застосування Watson охоплює технічну підтримку, внутрішнє навчання персоналу, сервісні центри, логістику, охорону здоров'я. Завдяки машинному навчанню та можливості тренувати ботів на власних даних, ця платформа дає змогу створювати гнучкі, масштабовані рішення, які відповідають на потреби конкретної організації [16].

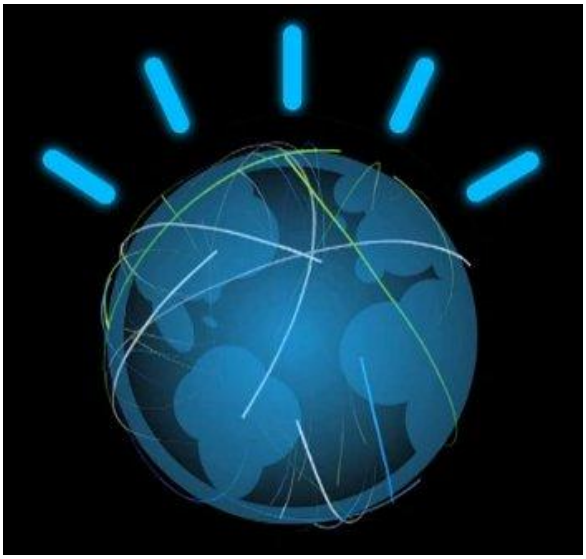


Рисунок 1.4 Логотип Watson Assistant

Віртуальний помічник від Booking.com дозволяє користувачам керувати своїми бронюваннями, отримувати інформацію про готелі, дізнаватися про правила скасування, зворотній трансфер тощо – усе це через інтерактивний чат у мобільному додатку або на сайті (рис. 1.5). Хоча бот не інтегрований із Telegram, його функціональність може бути реалізована і в месенджерах – що активно використовується іншими туристичними сервісами. Бот підтримує багатомовність, що дозволяє обслуговувати клієнтів у всьому світі. Він використовує базу даних Booking, персоналізовані шаблони повідомлень та штучний інтелект для розуміння намірів користувача. Це приклад успішного бот-рішення у сфері тревел-сервісів, де оперативність і точність відповіді критично важливі.

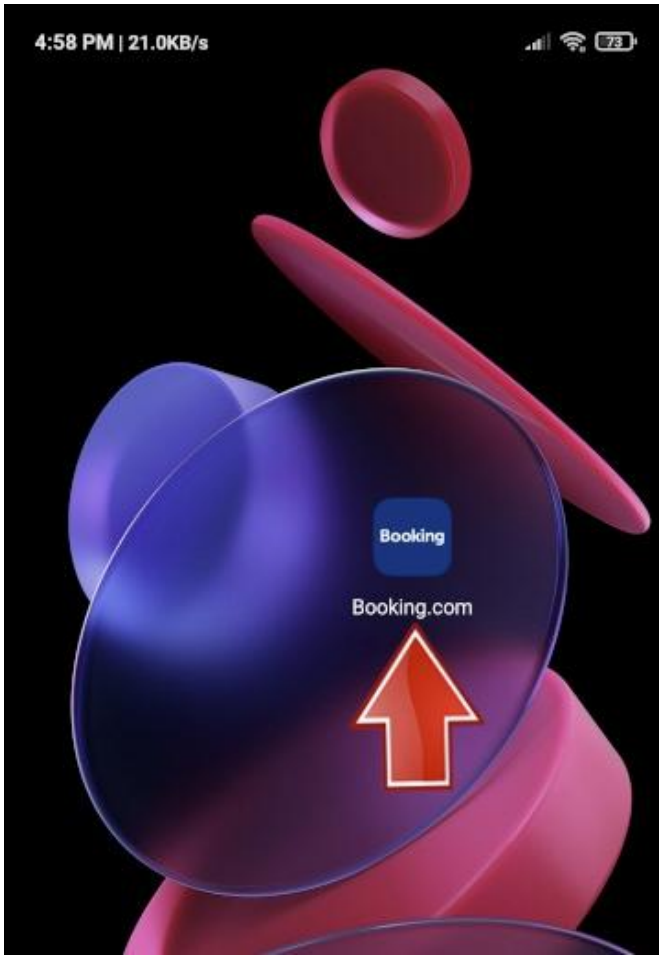



Рисунок 1.5 Додаток чат боту Booking.com

Duolingo Bot був створений як частина мовного додатку Duolingo для забезпечення реалістичної практики мовлення. Користувач може спілкуватися з ботом у вигляді "живої" розмови, на яку бот реагує відповідно до заданої теми, наприклад, замовлення їжі, бронювання готелю, розмова з лікарем (рис. 1.6). Ці боти були реалізовані з використанням AI та NLP, щоб моделювати природний діалог. Хоча зараз ця функція зосереджена всередині мобільного додатку, були експериментальні проєкти зі створення Duolingo-ботів у Telegram. Це приклад того, як чат-боти можуть бути ефективними не лише для підтримки клієнтів, а й для освіти – особливо у випадках, коли потрібна симуляція практичного застосування знань.


Рисунок 1.6 Duolingo Bot

Одним із прикладів успішного використання чат-ботів у сфері туризму є Hipmunk Bot – віртуальний помічник для планування подорожей. Цей бот працював у популярних месенджерах, зокрема Facebook Messenger, і допомагав користувачам знаходити авіарейси, готелі та варіанти подорожей, враховуючи індивідуальні вподобання. Його головною перевагою була простота взаємодії: користувач вводив, звідки він вирушає, а бот пропонував можливі напрямки, бронювання або рекомендації на основі введених даних. Крім цього, бот міг

приймати геолокацію та працювати у форматі розмови, максимально наближеному до спілкування з реальною людиною.

На рисунку 1.7 зображено інтерфейс Hipmunk Bot під час діалогу з користувачем у месенджері. Видно, як бот пропонує надіслати свою локацію або ввести місто вручну, щоб допомогти у плануванні маршруту. Такий підхід значно спрощує взаємодію та знижує бар'єр входу для користувача, який шукає зручний спосіб організації поїздки.

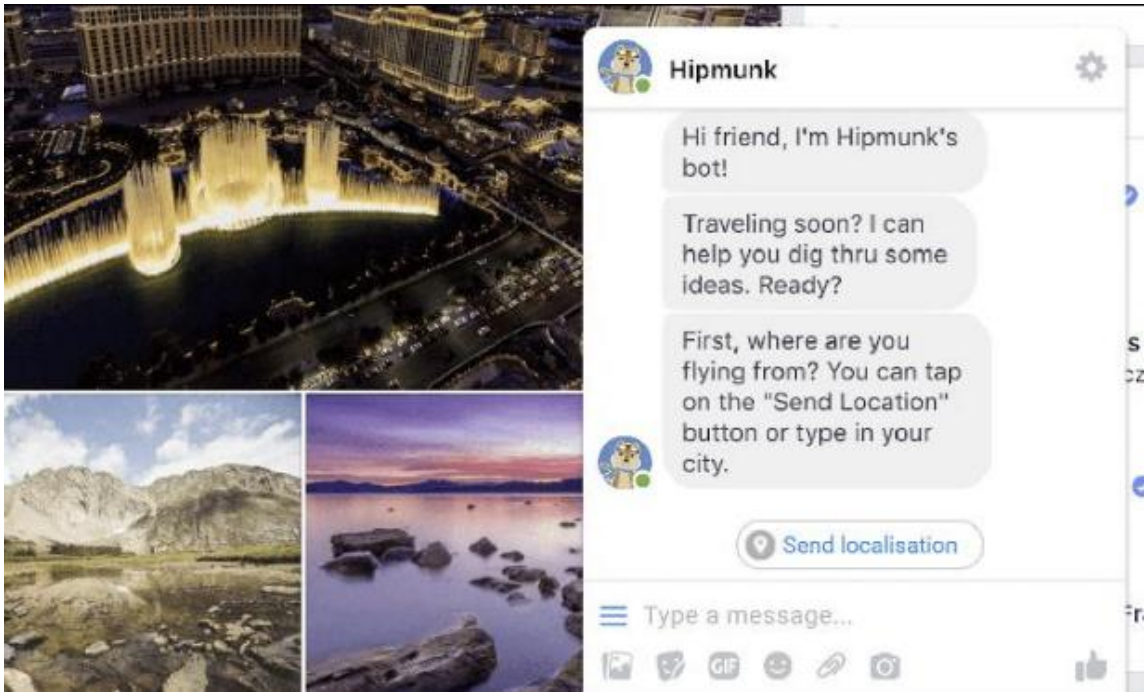



Рисунок 1.7 Інтерфейс чат-бота Hipmunk для планування подорожей

У результаті аналізу сучасного ринку чат-ботів можна побачити, що ці інструменти вже сьогодні відіграють ключову роль в автоматизації обслуговування клієнтів у різних галузях – від електронної комерції до охорони здоров'я. Їхня ефективність значною мірою залежить не лише від вбудованого штучного інтелекту чи функціональних можливостей, а й від того, наскільки зручно та швидко відбувається взаємодія з користувачем у цифровому середовищі. Саме тому важливу роль у розвитку чат-ботів відіграють сучасні інтернет-комунікаційні технології, які забезпечують передачу даних, з'єднання з API сервісів, обробку запитів у реальному часі тощо. Далі розглянемо основні види таких технологій та їхнє практичне застосування у створенні чат-ботів.

## 2.3 Методи та технології створення інтелектуальних чат-ботів

Для ефективної розробки інформаційної системи необхідно попередньо сформувати її концептуальну модель, яка відображатиме основні процеси функціонування системи, послідовність їх виконання, а також можливі альтернативні сценарії розвитку подій. Такий підхід дозволяє краще зрозуміти логіку взаємодії між компонентами системи ще на етапі проєктування. З цією метою доцільно створити контекстну діаграму у нотації IDEF0, яка забезпечує структурований опис функціональності системи на верхньому рівні. Після цього виконується її декомпозиція для деталізації окремих процесів. Крім того, варто розробити діаграми варіантів використання (use case), що демонструють взаємодію користувачів із системою та дозволяють визначити функціональні вимоги до неї.

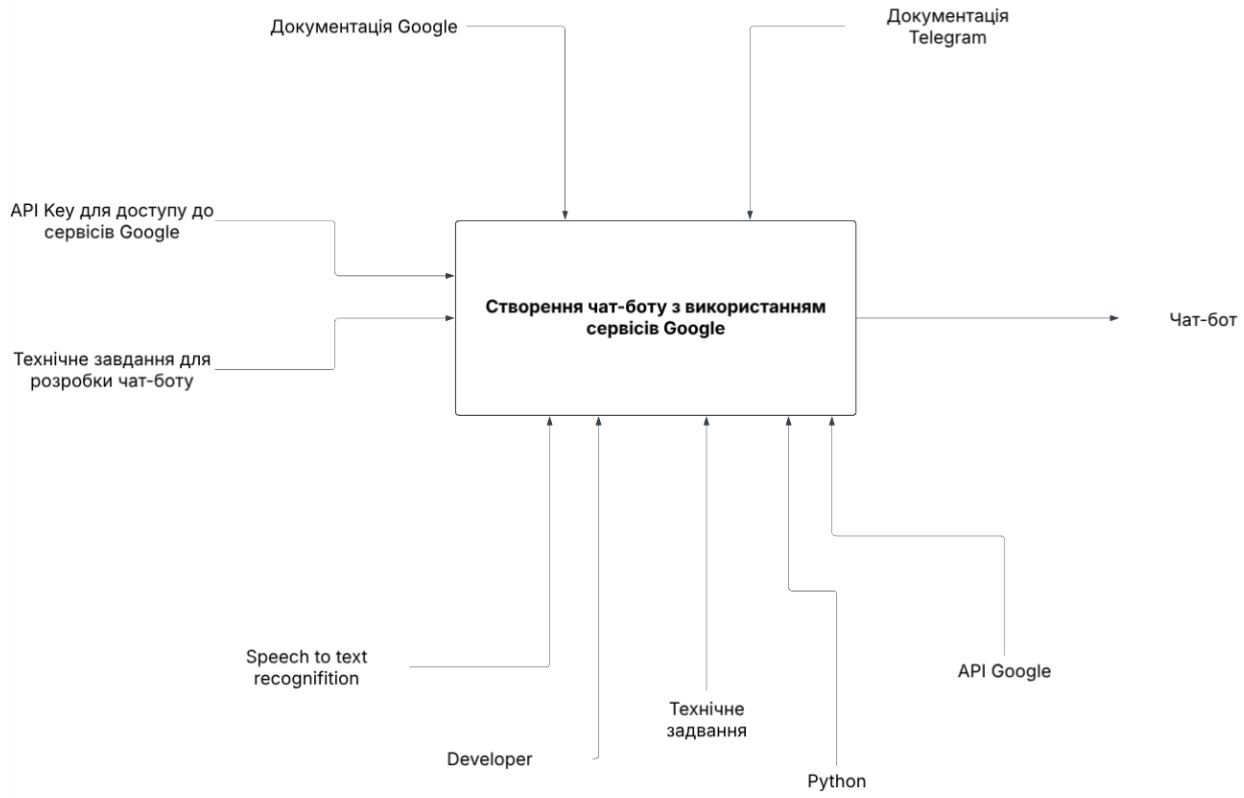



Рисунок 2.4 – Контекстна діаграма (IDEF0), що відображає загальну модель функціонування розроблюваної інформаційної системи

Побудова контекстної діаграми (рис. 2.4) є ключовим етапом у проєктуванні інформаційної системи, яка передбачає створення чат-бота з інтеграцією сервісів

Google. Такий підхід дозволяє узагальнено представити основні функціональні елементи системи, встановити взаємозв'язки між ними, а також визначити вхідні та вихідні дані, механізми та засоби керування.

У процесі створення чат-бота одним з перших кроків є отримання *API key* – унікального ідентифікатора, що використовується для автентифікації запитів до Google-сервісів. На основі контекстної діаграми також формується технічне завдання, що визначає вимоги до майбутньої системи [18].

До основних механізмів, які беруть участь у функціонуванні системи, належать:

– **Сервіс Google Speech-to-Text Recognition** — перетворює голосові запити користувача на текст;

– **Розробник**, який проєктує логіку роботи чат-бота;

– **Технічне забезпечення**, необхідне для розміщення та обробки запитів;

– **Мова програмування Python,** за допомогою якої реалізовано логіку взаємодії;

– **Google API**, що забезпечує доступ до зовнішніх сервісів.

Уся структура системи регламентується офіційною документацією Telegram та Google. Контекстна діаграма дозволяє наочно представити загальний вигляд системи, а її декомпозиція — деталізувати логіку виконання окремих процесів.

Підрівні контекстної діаграми:

1. ***Отримання доступу до Telegram API***
2. ***Інтеграція сервісів Google***
3. ***Формування алгоритмічної структури роботи чат-бота***

Ці три етапи відповідають ключовим крокам розробки веб-орієнтованої інформаційної системи:

– **Перший етап** – встановлення зв'язку з Telegram для виконання основних операцій.

– **Другий етап** – підключення окремих сервісів Google, необхідних для обробки запитів.

– **Третій етап** – побудова логіки функціонування чат-бота та реалізація сценаріїв його роботи.

Опис етапів декомпозиції:

**Етап 1 – Доступ до Telegram:**

– *Вхідні дані:* технічне завдання.

– *Вихідні дані:* встановлений зв’язок із Telegram.

– *Керування:* документація Telegram та Google.

– *Механізми:* Python, розробник, технічне забезпечення.

**Етап 2 – Інтеграція Google-сервісів:**

– *Вхідні дані:* Telegram-зв’язок.

– *Вихідні дані:* встановлений зв’язок із Google-сервісами.

– *Керування:* документація Google API.

– *Механізми:* розробник, Python, сервіс Speech-to-Text.

**Етап 3 – Формування алгоритму чат-бота:**

– *Вхідні дані:* API ключі, зв’язок з Google.

– *Вихідні дані:* функціонуючий чат-бот.

– *Керування:* документація Telegram.

– *Механізми:* Google API, розробник, Python.

На кожному з етапів декомпозиції формуються вихідні дані, які стають вхідними для наступного етапу, забезпечуючи цілісність і логічність розробки.

Інтеграція Google-сервісів:

Процедура підключення Google-сервісів включає такі підетапи:

– з’єднання з Google Assistant;

– використання Google Search для розширеного пошуку;

– отримання відповіді на запит.

**Етап підключення до Google Assistant:**

– *Вхідні дані:* зв’язок із Telegram.

– *Вихідні дані:* результат обробки запиту.

– *Керування:* Python, Google API, розробник.

– *Механізми:* документація Google.

**Етап підключення до Google Search (для масштабованого пошуку):**

– *Вхідні дані:* результат попереднього пошуку.

– *Вихідні дані:* повноцінна відповідь на запит.

– *Керування:* Python, Speech-to-Text, розробник.

– *Механізми:* документація Google.

**Завершальний етап – отримання результату:**

– *Вхідні дані:* масштабований пошук.

– *Вихідні дані:* сформована відповідь користувачу.

– *Керування:* API Google, технологія розпізнавання мовлення.

– *Механізми:* супровідна документація.

Моделювання системи за допомогою UML:

Для побудови логічної моделі системи використовується *Unified Modeling Language* (UML) — уніфікована мова моделювання, яка дозволяє візуалізувати структуру і поведінку програмного забезпечення. Використання UML дає змогу подати загальну архітектуру системи у вигляді діаграм з різних точок зору.

Одним з ключових інструментів є **діаграма прецедентів (Use Case diagram)**, що демонструє сценарії взаємодії користувачів із системою [18].

Головні актори системи:

– **Admin (розробник):** відповідає за налаштування чат-бота, технічну підтримку та управління конфігураціями.

– **User (користувач):** взаємодіє із ботом через доступні функції, реалізовані в системі.

На основі ідентифікованих акторів було сформовано перелік основних варіантів використання, зокрема:

– Використання Telegram;

– Текстовий пошук;

– Голосовий пошук;

– Додавання розширень;

– Зміна правил або конфігурацій;

– Інтеграція з Dialogflow.

На основі визначених акторів системи та сформованого переліку можливих сценаріїв взаємодії з чат-ботом була побудована діаграма варіантів використання (Use Case diagram), яка наочно відображає функціональні можливості системи. Вона представлена на рисунку 2.5.

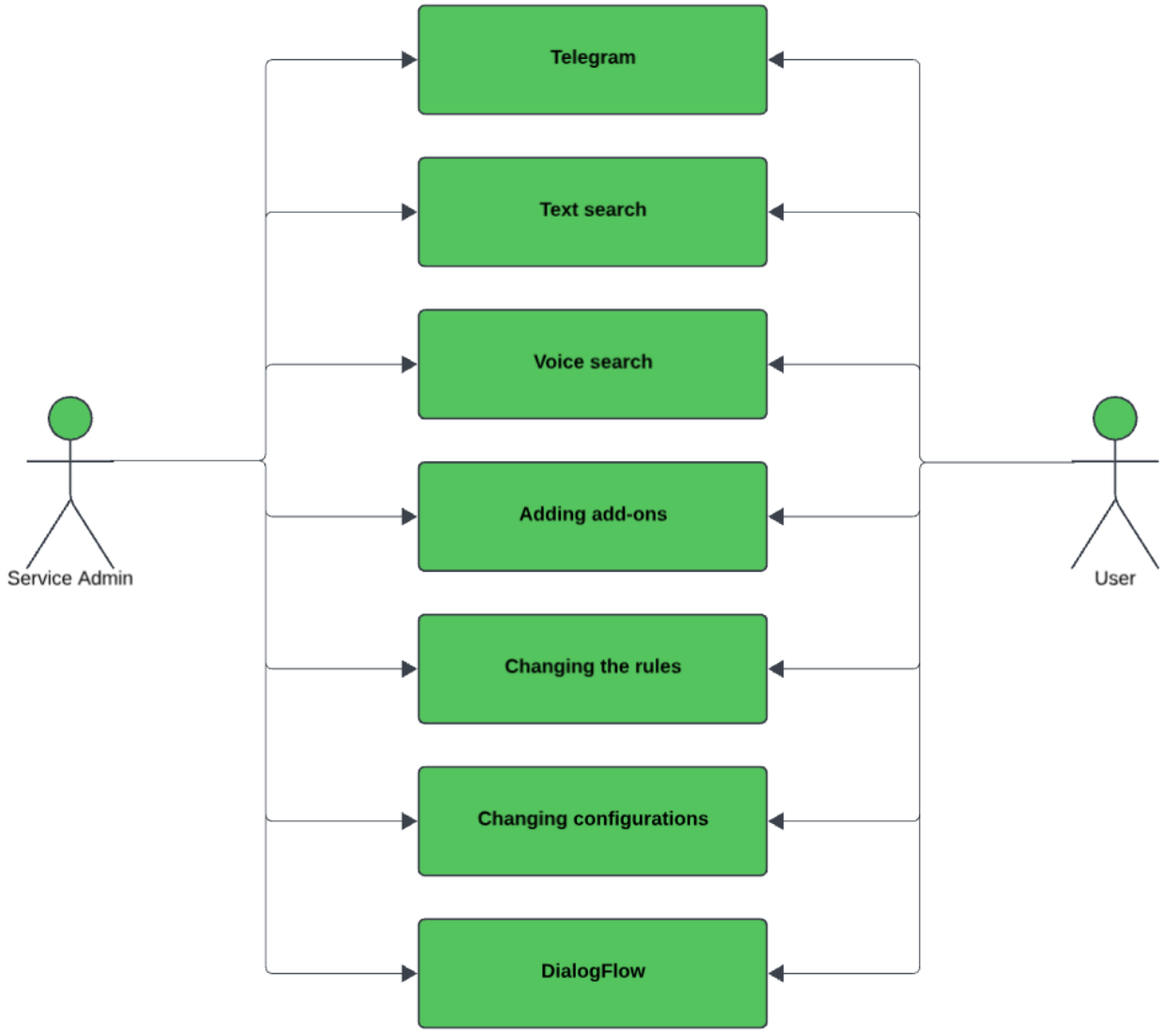



Рисунок 2.5 – Діаграма варіантів використання (Use Case) для взаємодії з чат-ботом

На основі проведеного аналізу типів чат-ботів та їх функціональних можливостей було прийнято рішення зосередитися на створенні інформаційного

чат-бота, основним завданням якого є покращення пошукових процесів шляхом інтеграції з платформою Telegram.

У процесі розробки було розглянуто та проаналізовано низку діаграм, зокрема UML-діаграми та діаграми варіантів використання (Use Case), що дало змогу глибше зрозуміти архітектуру системи та сценарії взаємодії з користувачем.

Зважаючи на те, що більшість чат-ботів мають простий та інтуїтивно зрозумілий інтерфейс, важливо забезпечити наявність чітко визначених правил функціонування та логічної структури виконання дій.

Для досягнення поставленої мети було застосовано метод часткових цілей, відповідно до якого вся система розбивалася на окремі етапи та підетапи. Кожен з них містив детальний опис дій, що забезпечило послідовну реалізацію всіх функціональних компонентів чат-бота [19].

# 3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

## 3.1 Етапи розробки інтелектуальної системи чат-бота

Чат-бот – це не лише інтерфейс взаємодії з користувачем, який дозволяє вести діалог у текстовому або голосовому форматі, але й програмний код, що автоматизує певні інформаційні процеси. В умовах цифрової епохи користувачі щодня взаємодіють із численними програмами, додатками та ботами, зокрема в соціальних мережах.

Поняття "бот" набуло особливої популярності у 2015 році, коли стрімко зросла потреба бізнесу в автоматизації рутинних завдань. Багато консалтингових і комерційних компаній почали активно впроваджувати ботів у свою діяльність. Значну роль у популяризації ботів відіграв месенджер Telegram, який зробив цю технологію доступною для масового користувача. Завдяки ботам з’явилась можливість здійснювати швидкий і точний пошук, отримуючи інформацію безпосередньо в чаті [21].

Головна ідея проєкту полягає у створенні **інформаційної системи в межах іншої інформаційної системи**, яка здійснює пошук запитуваної інформації та повертає результат у зручному для користувача вигляді – у форматі текстового повідомлення. Основна мета — забезпечити користувачу можливість отримання корисної, розважальної чи пізнавальної інформації без необхідності переходити між різними платформами. Система повинна бути універсальною, інтуїтивно зрозумілою та багатофункціональною.

У процесі реалізації проєкту було використано метод **SMART**, що дозволив чітко сформулювати цілі розробки. Методологія SMART базується на таких принципах:

- **S** (Specific) – конкретність;
- **M** (Measurable) – вимірюваність;
- **A** (Achievable) – досяжність;

- **R** (Relevant) – актуальність;
- **T** (Time-bound) – визначеність у часі.

Використання мови програмування Python у поєднанні з технічною документацією Google API дозволило реалізувати функціональну модель чат-бота, яка фактично виконує роль пошукового асистента.

Серед ключових результатів проєкту:

– Інтеграція модуля *Speech-to-Text recognition* для голосового введення даних та їх подальшого перетворення у текст;

– Підключення до *Google Assistant API* та *Google Search API*;

– Паралельна робота голосового асистента й пошукової системи;

– Отримання результатів запиту у зручному вигляді для користувача.

Планування структури робіт

Для організації процесу реалізації проєкту було застосовано метод **структурної декомпозиції робіт** (*Work Breakdown Structure, WBS*). Цей підхід передбачає ієрархічний розподіл усіх завдань проєкту на менші компоненти, що дає змогу краще планувати хід розробки, визначати відповідальних осіб, розподіляти ресурси та оцінювати витрати. Іншими словами, на основі кінцевих результатів формується перелік конкретних дій, що входять до складу проєкту.

WBS-структура представлена на рисунку 3.1.

Організаційна структура проєкту

Організаційна структура проєкту відображає логіку взаємодії між учасниками, їхні ролі та відповідальність, а також загальні принципи управління проєктом. Це концептуальна модель, яка визначає, як формуються управлінські рішення в межах розробки чат-бота.

Організаційна структура компанії або проєкту показує, хто виконує певні функції, хто ухвалює рішення та як забезпечується ефективна взаємодія між учасниками. Діаграма організаційної структури (OBS) наведена на рисунку 3.2 [22].

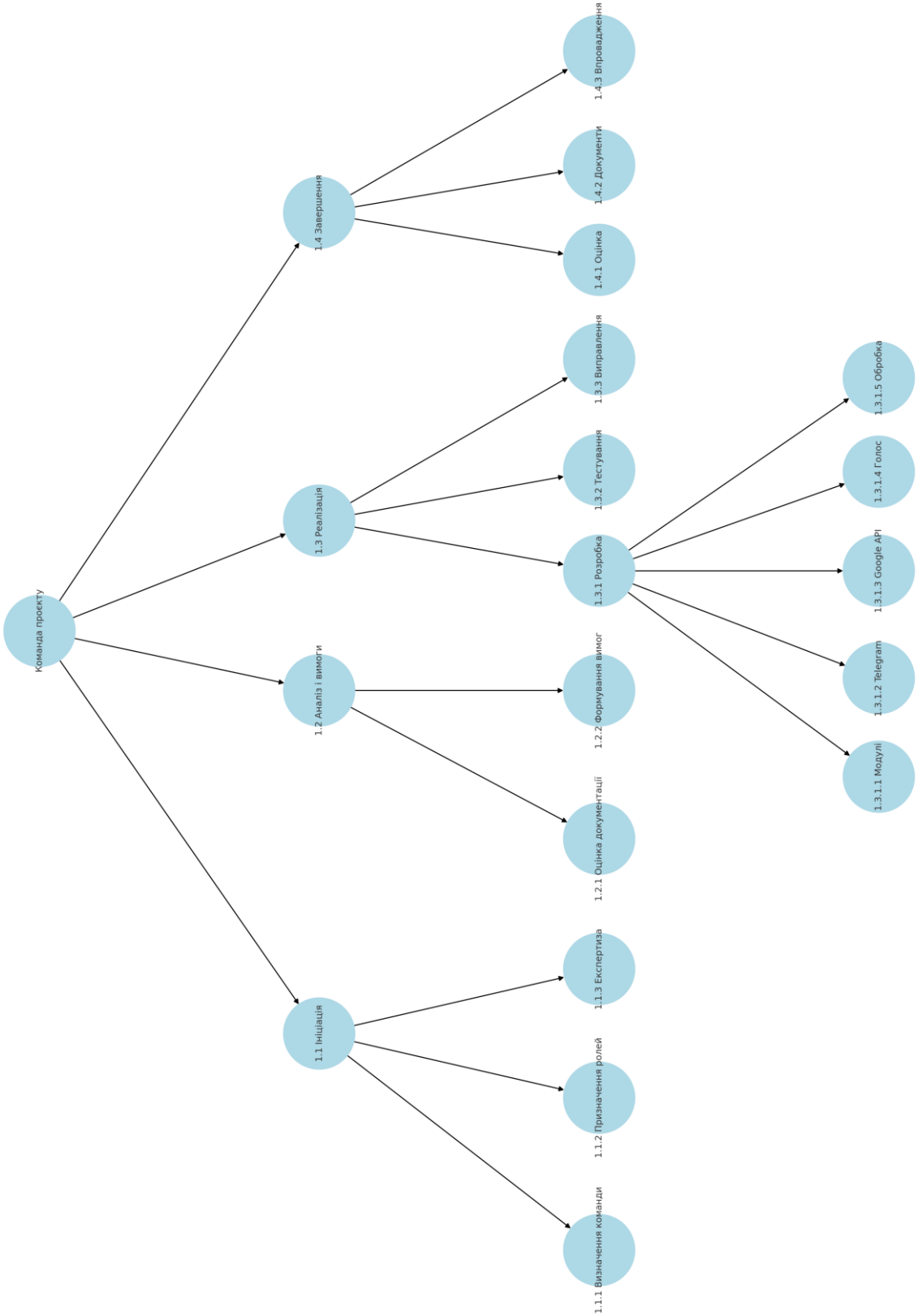



Рисунок 3.1 – Ієрархічна WBS-структура інформаційної системи

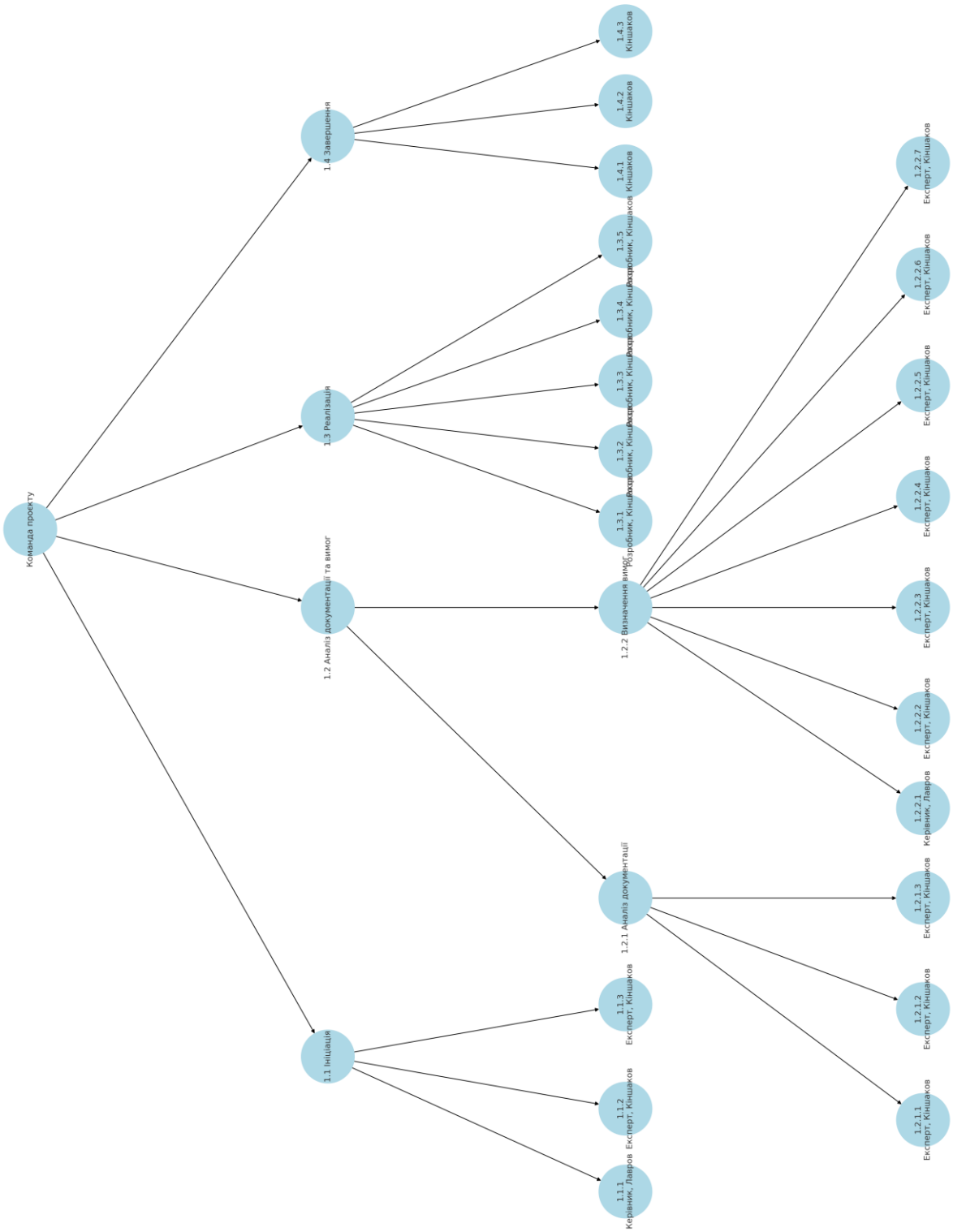



Рисунок 3.2 – Структура відповідальності учасників розробки (OBS)

У процесі управління проєктом важливо чітко розподілити ролі між усіма учасниками та уникнути плутанини у виконанні завдань. З цією метою доцільно використовувати матрицю відповідальності, зокрема матрицю RACI (Responsible, Accountable, Consulted, Informed) – простий, але ефективний інструмент для визначення обов’язків та зон відповідальності в межах процесів чи проєктів.

Матриця RACI дозволяє уникнути неузгодженостей у виконанні завдань, виявити «зони невизначеності» у функціональних обов’язках, а також сприяє прозорому розподілу відповідальності між учасниками команди. Вона може бути корисною як на стадії поточного стану процесу ("AS-IS"), так і при моделюванні майбутнього стану ("TO-BE").

У випадках, коли виникають розбіжності у тлумаченні обов’язків чи повноважень, матриця відповідальності сприяє винесенню спірних моментів на загальне обговорення, що в результаті дозволяє приймати зважені колективні рішення.

Для ефективного планування реалізації проєкту було застосовано метод PERT (Program Evaluation and Review Technique) – методику оцінки та перегляду плану, яка особливо корисна для управління великими, складними та одноразовими проєктами. Методика PERT дозволяє враховувати невизначеність у тривалості завдань, що дає змогу скласти робочий графік навіть за умови неповної інформації.

Особливістю методу PERT є акцент на аналізі часу виконання кожної задачі, а також виявлення критичного шляху та мінімального необхідного часу для реалізації проєкту в цілому.

Для візуалізації проєктної мережі було використано інструмент GanttProject з надбудовою PERT-діаграма, що дозволяє наочно зобразити зв’язки між завданнями, їхню послідовність і залежності. Відповідну мережу подано на рисунках 3.3 та 3.4 [23].

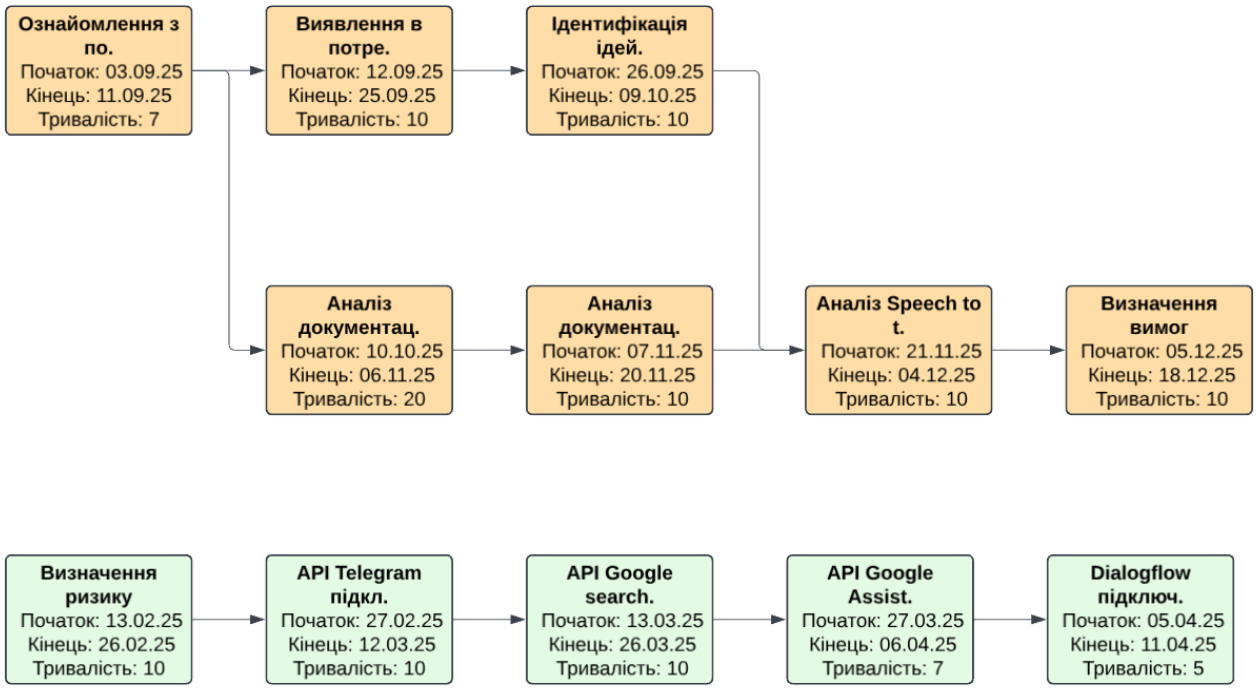



Рисунок 3.3 – Побудова логічної мережі завдань методом PDM

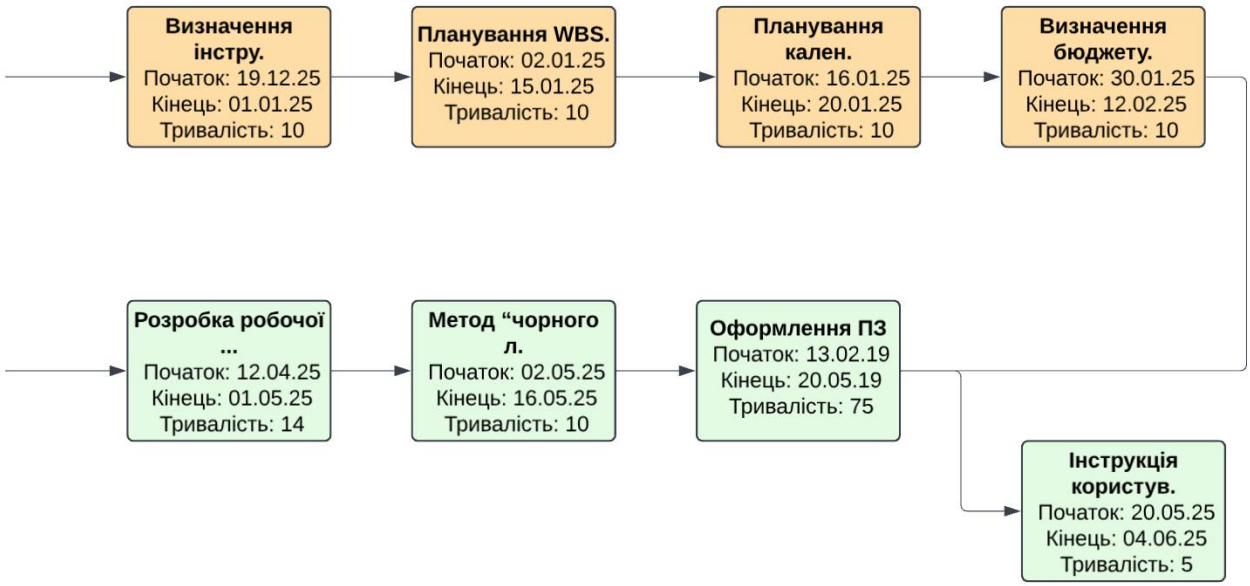



Рисунок 3.4 – Продовження логічної мережі завдань методом PDM

Для ефективного управління часовими ресурсами в межах розробки інформаційної системи було використано діаграму Ганта. Цей інструмент є одним із найпоширеніших способів візуалізації запланованих завдань у часовому просторі. Горизонтальне представлення задач на часовій шкалі дозволяє наочно

відобразити строки початку та завершення кожної задачі, послідовність їх виконання та можливі перетини.

Діаграма Ганта широко застосовується в управлінні проєктами різної складності та масштабів, оскільки забезпечує зручний формат представлення інформації для всієї команди. Завдяки їй менеджери можуть своєчасно реагувати на відхилення від плану, контролювати хід виконання завдань та загальний прогрес проєкту. Візуалізація графіка реалізації проєкту наведена на рисунку 3.5.

*Управління ризиками:*

Після формування повного переліку робіт та визначення відповідальних осіб, було проведено оцінювання можливих ризиків, що можуть вплинути на якість, строки та стабільність реалізації проєкту. Метою цього етапу стало передбачення потенційних проблем і розробка заходів щодо їх уникнення або мінімізації впливу.

У процесі аналізу було виокремлено такі основні ризики:

– R1 – внесення змін до технічного завдання на етапі реалізації;
– R2 – виявлення помилок у коді після завершення етапу розробки;
– R3 – недотримання встановленого календарного плану;
– R4 – зміни у функціоналі сторонніх сервісів Google;
– R5 – тимчасова непрацездатність (хвороба) розробника;
– R6 – некоректне або недостатнє тестування функціоналу;
– R7 – несправності або збої в роботі апаратного забезпечення.

Наступним етапом було проведення експертної оцінки ймовірності настання кожного з ризиків. На основі отриманих даних було сформовано таблицю ризиків (табл. 3.1), яка включає ймовірність виникнення, рівень впливу та рекомендовані заходи реагування [24].

Таблиця 3.1

Рівень ймовірності виникнення ризиків у процесі реалізації проєкту

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 |
|---|---|---|---|---|---|---|---|
| Слабо ймовірний | | | | | | | |
| Малоймовірний | | | | | | | |
| Ймовірний | | | | | | | |
| Дуже ймовірний | | | | | | | |
| Майже неминучий | | | | | | | |

Після оцінки ймовірності виникнення основних ризиків було побудовано таблицю можливих втрат, які можуть бути спричинені реалізацією зазначених загроз. Цей аналіз дозволяє заздалегідь виявити критичні зони впливу та підготувати відповідні заходи для мінімізації шкоди [25].

Оцінка втрат базується на прогнозуванні можливих наслідків у разі реалізації кожного з ризиків, визначених у попередньому етапі. Зокрема, було враховано вплив на часові рамки, якість кінцевого продукту, стабільність функціонування системи, фінансові витрати, а також людський ресурс.

На основі проведеного аналізу була сформована таблиця потенційних втрат, які можуть виникнути як у процесі проєктування, так і на етапі безпосередньої реалізації та впровадження інформаційної системи [24-26].

Таблиця 3.2

Рівень впливу ризиків на результати реалізації проекту

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 |
|---|---|---|---|---|---|---|---|
| Мінімальна | | | | | | | |
| Низька | | | | | | | |
| Середня | | | | | | | |

Продовження таблиці 3.2

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Висока | | | | | | | |
| Максимальна | | | | | | | |

На основі двох попередніх таблиць — оцінки ймовірності виникнення ризиків (табл. 3.1) та рівня їх впливу (табл. 3.2) — була сформована матриця ризиків, яка дозволяє узагальнено оцінити потенційну загрозу для проєкту.

Матриця поєднує два ключові параметри:

Ймовірність виникнення ризику (від "слабо ймовірного" до "майже неминучого");

Рівень наслідків (втрат), які ризик може спричинити (від "мінімальних" до "максимальних").

На основі цього перехрестя формується три рівні ризиків:

Білий (світлий) – незначний ризик (можна ігнорувати або просто відстежувати);

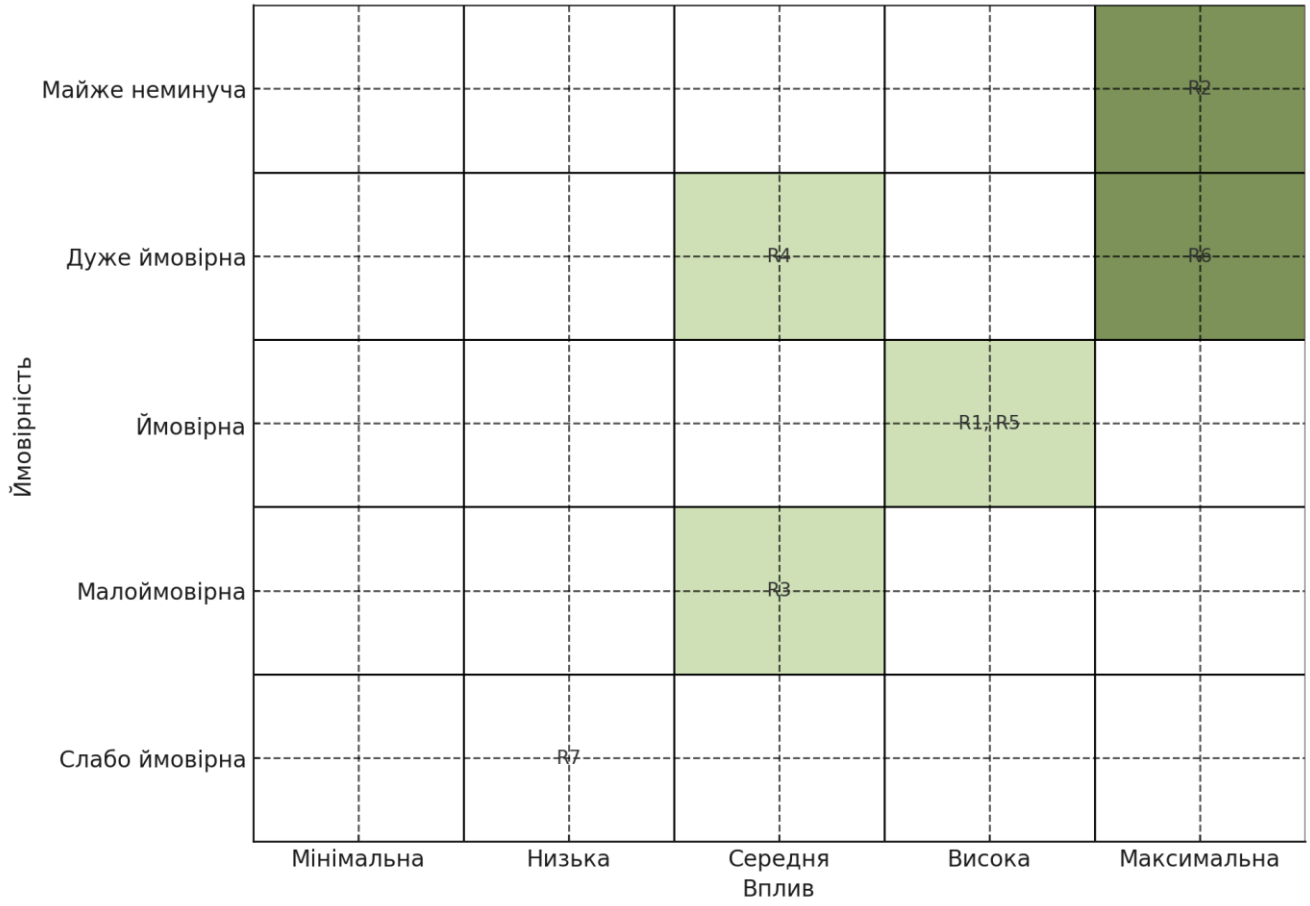
Світло-сірий – помірний ризик (потребує уваги та моніторингу);

Темно-сірий / чорний – критичний ризик (потребує негайної реакції або запобіжних заходів).

Завдяки такій візуалізації команда проєкту може швидко оцінити, які ризики потребують першочергової уваги, та визначити пріоритетність дій щодо їх уникнення або мінімізації [26].

Таблиця 3.3

Матриця оцінки ризиків у координатах “Ймовірність – Наслідки”




На основі проведеного аналізу ймовірності та рівня впливу ризиків (див. табл. 3.5–3.7), було визначено три критичних ризики, які становлять найбільшу загрозу для успішної реалізації проєкту [27]:

– *R2 – пропущені помилки у ході розробки*

Основна причина виникнення даного ризику – людський фактор. Уникнути його можна шляхом ретельного перегляду та доопрацювання коду, а також застосування багаторівневого модульного тестування. Ключовим елементом зниження ризику виступає професійність розробника та використання сучасних інструментів перевірки коду.

– *R4 – зміна функціоналу сервісів Google*

Незважаючи на відносно низьку ймовірність, цей ризик має суттєвий вплив у разі реалізації. У випадку внесення змін до зовнішніх API або припинення

підтримки певного функціоналу з боку Google, проєкт може зазнати затримок. Вирішенням є оперативна адаптація системи, що потребує втручання розробника – зокрема, заміни або оновлення інтеграцій з відповідними сервісами.

– *R6 – некоректне тестування*

Цей ризик може призвести до виявлення помилок уже після завершення розробки, що викликає додаткові витрати часу та негативне враження замовника. Для мінімізації загроз необхідно впровадити контроль якості на кожному етапі тестування, включаючи юніт-тести, інтеграційні тести та приймальне тестування.

Фінальним етапом планування проєкту є визначення бюджету, необхідного для реалізації всіх завдань, передбачених у технічному завданні. Розрахунок бюджету базується на обсязі робіт, кількості фахівців, що залучаються, та їхній кваліфікації.

У процесі розробки були залучені наступні ключові виконавці:

– Розробник – відповідальний за створення архітектури системи, програмування, інтеграцію з зовнішніми сервісами (Google API, Telegram API) та тестування;

– Керівник проєкт**у** – забезпечує контроль над етапами розробки, координацію роботи команди, планування строків;

– Експерт – здійснює перевірку технічної документації, формування вимог, аналіз ризиків та участь у приймальному тестуванні.

У наступному розділі буде представлено деталізовану таблицю з розподілом бюджету відповідно до ролей та обсягу виконаних робіт [28].

## 3.2 Вибір технологічного стеку для розробки

Початковим етапом створення чат-бота стало визначення мови програмування, за допомогою якої реалізовуватиметься основна логіка системи. Враховуючи складність функціоналу, необхідність обробки запитів, інтеграцію з API сторонніх сервісів та гнучкість розробки, було прийнято рішення використовувати мову **Python**.

Python є високорівневою, інтерпретованою мовою загального призначення, яка орієнтована насамперед на **зручність та продуктивність розробника**.

Основними перевагами Python є:

– **Універсальність** – підтримує розробку вебзастосунків, десктопних програм, ігор, систем автоматизації, чат-ботів, наукових та інженерних розрахунків.

– **Простота синтаксису** – код, написаний на Python, є читабельним та зрозумілим навіть для початківців [28].

– **Низький поріг входу** – ідеально підходить для швидкого освоєння розробки.

– **Швидка розробка та зручний супровід** – у порівнянні з Java або C++, Python дозволяє писати менше коду для вирішення тієї самої задачі, що знижує витрати на підтримку проєкту.

– **Велика екосистема** – тисячі готових бібліотек і фреймворків, серед яких особливо актуальні для даного проєкту: ***telebot, requests, speech_recognition, google-api-python-client.***

Підтримка з боку IT-індустрії

Python є однією з найбільш підтримуваних мов сучасності. Її активно використовують провідні світові компанії, серед яких:

– **Google** – інтеграція з Google Cloud та Google Assistant;

– **Telegram** – надає офіційну підтримку API для Python;

– **Facebook, Mozilla, Dropbox, RedHat, Yandex** – використовують Python у своїх внутрішніх та зовнішніх продуктах.

Особливо важливим аргументом на користь вибору Python стало те, що **обидва ключові сервіси, що використовуються в системі – Telegram і Google – надають офіційні API з повною підтримкою мови Python**. Це значно спрощує процес інтеграції та скорочує час розробки [29-30].

### 3.3 Використання мови Python у створенні чат-ботів

**Python** – це одна з найпопулярніших мов програмування, яка була створена Гвідо ван Россумом у 1991 році. Вона належить до мов загального призначення і активно використовується у найрізноманітніших сферах: **веб-розробці**, **автоматизації процесів**, **наукових обчисленнях**, **розробці програмного забезпечення**, **робототехніці**, **аналітиці даних**, а також для створення **чат-ботів**.

Основні особливості Python:

- **Портативність**: Python працює на більшості сучасних операційних систем – Windows, macOS, Linux.
- **Простий та інтуїтивно зрозумілий синтаксис**, що нагадує англійську мову, завдяки чому зменшується час на розробку програмного забезпечення.
- **Інтерпретована мова**: код виконується одразу після написання, що значно пришвидшує процес створення прототипів і тестування.
- **Гнучка парадигма програмування**: підтримує процедурне, об'єктно-орієнтоване та функціональне програмування.
- **Широкий набір бібліотек та фреймворків**: серед найпопулярніших – Django та Flask для веб-розробки, NumPy і Pandas для обробки даних, Telebot для створення Telegram-ботів тощо.
- **Можливість автоматизації**: дозволяє створювати скрипти для автоматизації рутинних завдань (наприклад, перейменування файлів, надсилання листів, парсинг контенту).

Сфери застосування:

- **Веб-розробка (back-end)**: Python широко використовується для створення серверної частини веб-сайтів та веб-застосунків. Серед типових задач – обробка HTTP-запитів, взаємодія з базами даних, реалізація бізнес-логіки, маршрутизація та забезпечення безпеки. Найпоширенішими фреймворками є **Django** і **Flask**.
- **Автоматизація**: Python дозволяє автоматизувати виконання повторюваних завдань – від простих дій до складних сценаріїв перевірки, перетворення форматів даних, парсингу сайтів або перевірки помилок у програмному коді.

– **Тестування програмного забезпечення**: Python використовується для написання тестів, контролю якості та налагодження систем. Приклади інструментів – **Green**, **Requestium** тощо [29].

– **Наукові обчислення та обробка даних**: мова широко застосовується в Data Science, Machine Learning і Big Data-завданнях.

– **Розробка чат-ботів**: завдяки бібліотекам ***pyTelegramBotAPI, aiogram, telebot***, а також зручній інтеграції з Google API, Python є одним з найзручніших рішень для реалізації функціоналу інтерактивних ботів.

Підтримка великих компаній

Python активно використовується такими міжнародними компаніями як:

– **Google**

– **Facebook**

– **Dropbox**

– **Spotify**

– **Netflix**

– **Yandex**

– **Telegram**

Ці компанії обрали Python завдяки його гнучкості, швидкості розробки та величезній спільноті підтримки. Зокрема, **Google** та **Telegram** надають офіційні Python-бібліотеки для роботи зі своїми API, що стало ключовим фактором у виборі мови для даного проєкту [30].

### 3.4 Вибір платформи та середовища для розгортання

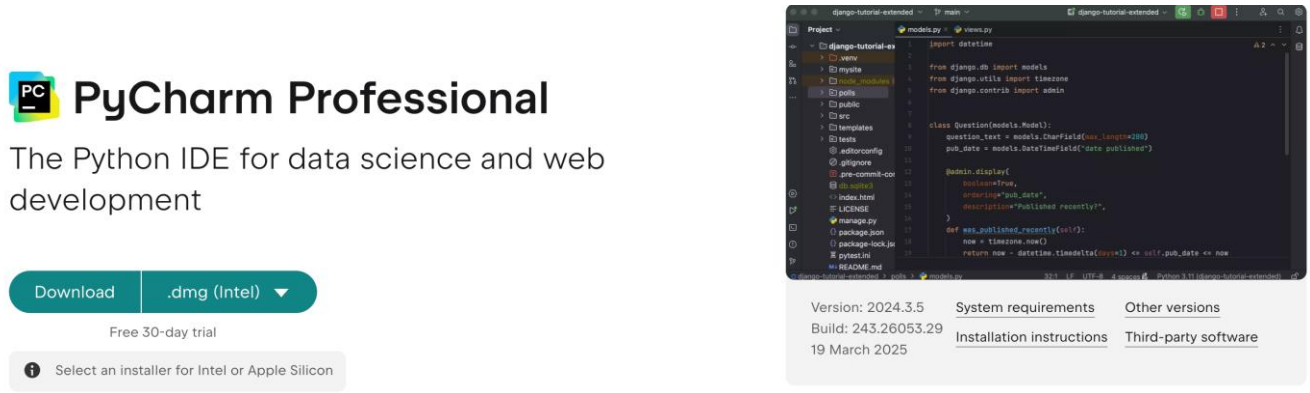



Рисунок 3.3 – IDE PyCharm для розробки продукту

1. Google Assistant

**Google Assistant** – це інтелектуальний голосовий помічник, який дозволяє здійснювати управління пристроєм голосом або текстовими командами. Помічник здатен надавати персоналізовану інформацію та виконувати широкий спектр дій:

– керування пристроями та «розумним» будинком;

– відображення персональних даних (календар, нагадування, повідомлення);

– пошук інформації в Інтернеті;

– управління медіа (музика, відео);

– встановлення будильників, таймерів;

– відкриття додатків;

– переклад тексту в реальному часі;

– розпізнавання голосових профілів користувачів.

Асистент працює у тісному зв’язку з екосистемою Android, підтримує безперервні розмови без необхідності постійно повторювати фразу активації "Hey Google", і може підлаштовувати відповіді відповідно до особистості співрозмовника.

2. Dialogflow

**Dialogflow** – це інструмент Google для розпізнавання природної мови та створення розмовних інтерфейсів (чат-ботів, голосових помічників, IVR-систем). Завдяки використанню технологій **машинного навчання**, Dialogflow:

– розпізнає **наміри користувача (intents)**;

– підтримує **предметні сутності** (час, дати, числа, адреси);

– дозволяє створювати та навчати власних асистентів;

– підтримує понад **20 мов** та інтеграції з **14 платформами** (Telegram, Facebook Messenger, Slack, інші);

– забезпечує **багатомовність**, що робить його придатним для глобального використання.

Dialogflow працює на базі Google Cloud Platform, постійно вдосконалюється, має доступ до SLA (Service Level Agreement) і підтримку хмарних сервісів [31].

3. Google Cloud Speech-to-Text

**Cloud Speech-to-Text** – це сервіс Google для розпізнавання мови в текст. Він підтримує:

– **реальний час** транскрипції з мікрофона або аудіопотоку;

– розпізнавання аудіофайлів з локального сховища;

– **багатомовність** (до 4 мов у одному запиті);

– автоматичне форматування (дати, числа, номери телефонів);

– високу точність завдяки використанню **глибинних нейронних мереж**.

Сервіс активно використовується у сфері штучного інтелекту та автоматизації, є ключовим елементом для реалізації голосових чат-ботів, таких як той, що описується в межах даного проєкту.

4. Google Search

**Google Search** – найбільша у світі пошукова система, яка обробляє сотні терабайт інформації з веб-ресурсів. У межах реалізації чат-бота сервіс використовується для:

– виконання пошукових запитів на основі голосового або текстового вводу;

– повернення швидких відповідей за допомогою системи **PageRank**;

– доступу до **структурованої інформації** (погода, рейси, календарі, новини тощо);

– використання **пошукових операторів** для гнучкої фільтрації результатів;

– **безпечного пошуку** (SafeSearch) для виключення небажаного контенту.

Завдяки поєднанню вищезазначених сервісів, проєкт забезпечує **інтелектуальну, багатофункціональну та зручно масштабовану** систему взаємодії з користувачем у формі чат-бота, здатного обробляти як текстові, так і голосові запити в реальному часі [32].

### 3.5 Структурна модель функціонування чат-бота

Описаний у другому розділі функціонал чат-бота є унікальним за своєю структурою та водночас інтуїтивно зрозумілим для користувача. Його архітектура побудована таким чином, щоб забезпечити максимально просту взаємодію з користувачем при високому рівні технічної реалізації.

Для наочного розуміння принципу функціонування інформаційної системи, розроблено структурно-логічну схему, яка демонструє основні етапи обробки вхідних запитів, взаємодію з сервісами Google, обробку даних та виведення результату у зручній для користувача формі [33-34].

Вказана схема подана на рисунку 3.4 та відображає ключові компоненти системи, їхню взаємодію та послідовність виконання основних процесів.

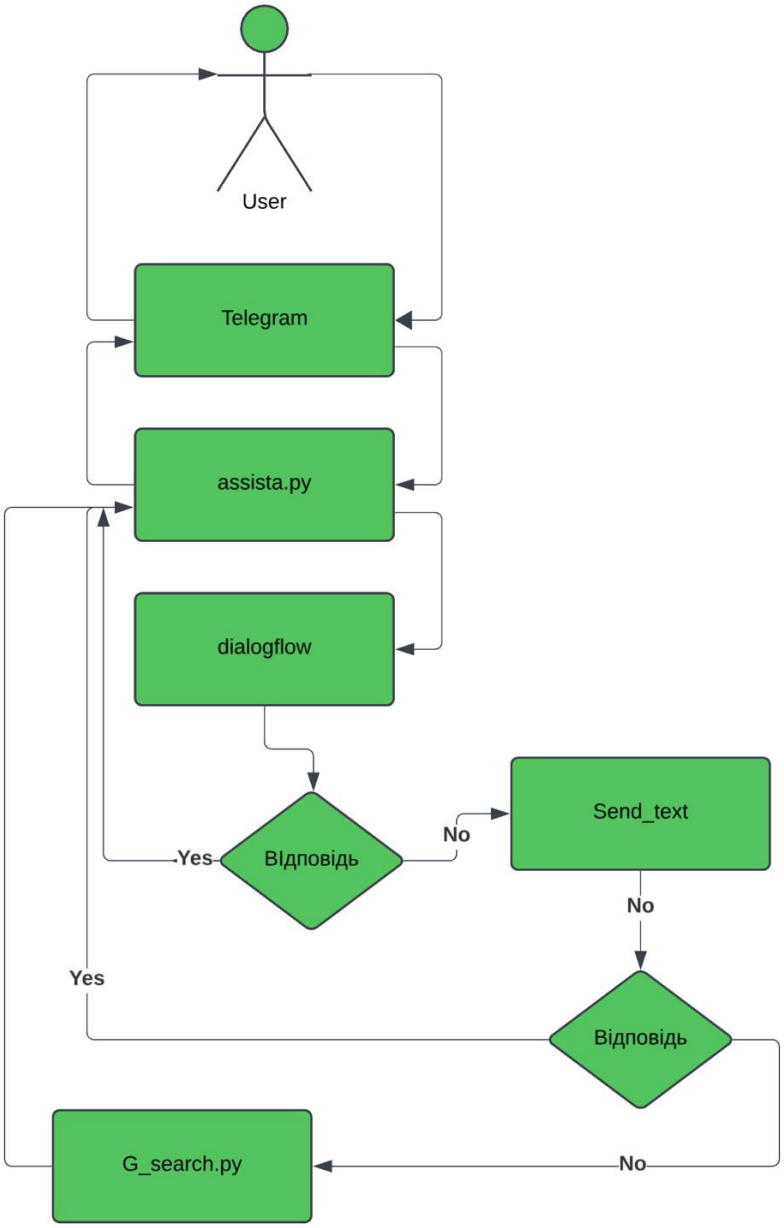



Рисунок 3.4 – Діаграма прецедентів

Аналізуючи функціональну схему (див. рис. 3.10), можна впевнено стверджувати, що реалізований чат-бот має **унікальну архітектуру взаємодії** між внутрішніми модулями та зовнішніми сервісами. Особливу увагу при цьому заслуговує **інтеграція прикладних програмних інтерфейсів (API)**, які є ключовим елементом усієї системи [34].

На сьогоднішній день використання API є одним із найважливіших підходів у розробці інформаційних систем, адже саме вони:

– забезпечують ефективну взаємодію між різними сервісами та платформами;

– спрощують реалізацію складного функціоналу;

– дозволяють масштабувати систему без значної переробки коду;

– економлять ресурси розробника.

У даному проєкті реалізація чат-бота здійснюється на основі **клауд-месенджера Telegram**, який підтримує відкритий API та має зручну архітектуру для інтеграції зі сторонніми сервісами, зокрема Google.

Отримання доступу до Telegram API

Першим кроком розробки було створення бота в середовищі Telegram, зокрема – через офіційного бота ***@BotFather***. Процедура передбачає:

1. Створення нового бота;
2. Присвоєння йому унікального імені та username;
3. Генерацію **унікального токена доступу (API key);**
4. Збереження ключа для подальшого використання в коді Python.

На **рисунку 3.5** показано процес отримання API ключа Telegram, що слугує ідентифікатором для взаємодії з платформою через бібліотеки типу telebot або aiogram.

Цей токен надалі використовується для обробки вхідних повідомлень, відправлення відповідей, а також для реалізації всього механізму взаємодії користувача з ботом у режимі реального часу [35].

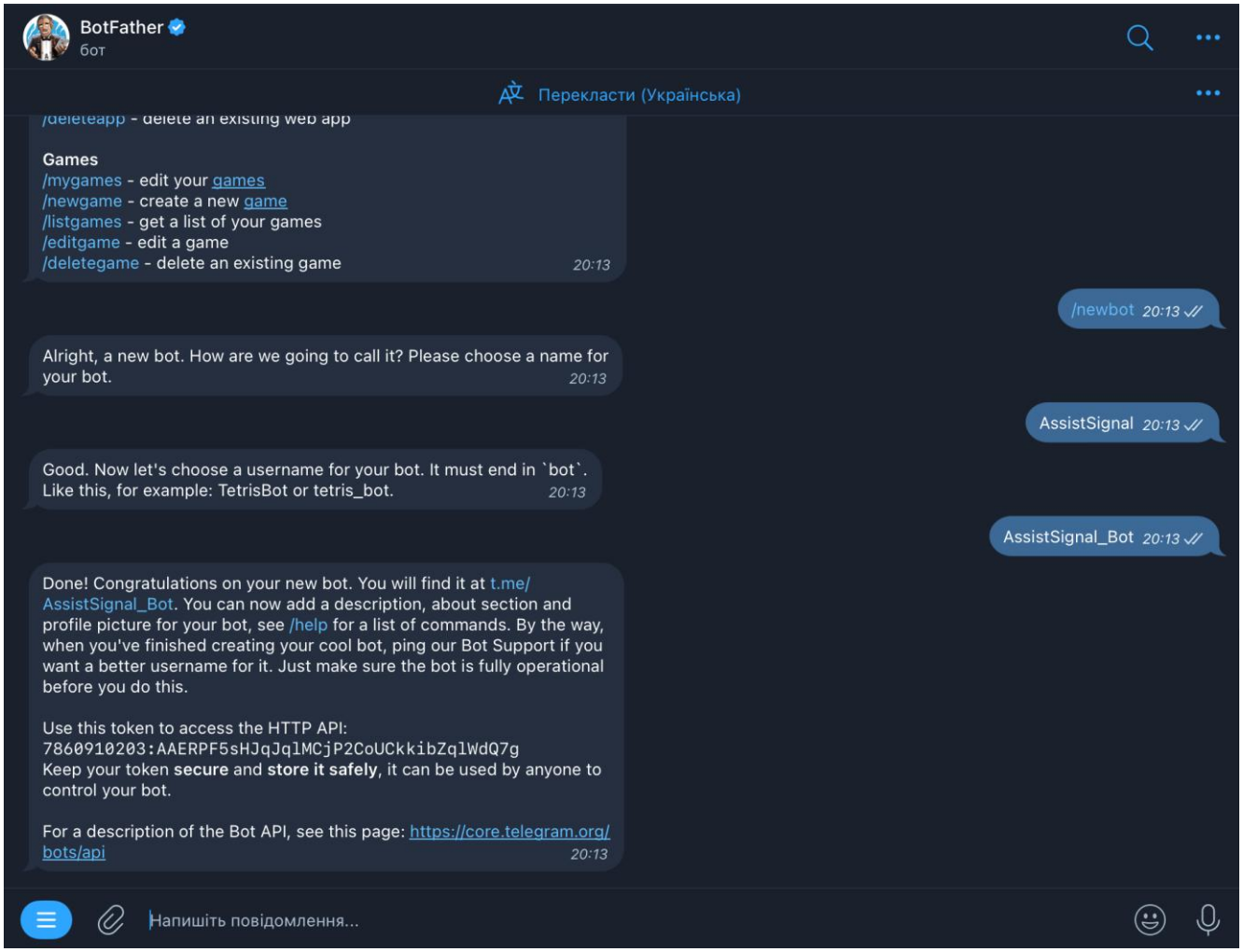



Рисунок 3.5 – Генерація унікального токена доступу до Telegram API

Після створення Telegram-бота та отримання API-токена, наступним етапом стала реалізація скрипта, який дозволяє автоматизувати процес обробки запитів користувача шляхом інтеграції Telegram з сервісами компанії Google.

Цей скрипт написаний мовою програмування Python, що була обрана за її зручність, універсальність та високу сумісність з API Google і Telegram. Основна мета скрипта — забезпечити взаємодію між користувачем, месенджером Telegram, а також хмарними інструментами Google (Dialogflow, Assistant, Speech-to-Text, Search API тощо) [36].

В рамках цього скрипта реалізовані такі функціональні блоки:

– приймання вхідних повідомлень (текстових або голосових);
– обробка тексту та перетворення голосу в текст;
– надсилання запитів до Google Search / Dialogflow;

– отримання структурованої відповіді;

– повернення результату користувачу в чаті.

На рисунку 3.5 зображено структуру проєкту, реалізовану засобами мови Python. Дана структура відображає основні модулі, функції та залежності, необхідні для запуску та стабільної роботи чат-бота.

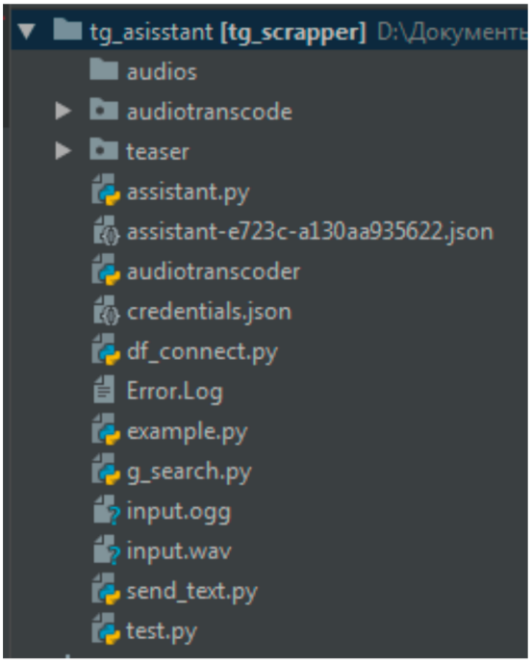



Рисунок 3.6 – Архітектура директорій і файлів проєкту Telegram-бота

Тека audios призначена для збереження голосових повідомлень користувача. Каталог audiotranscode використовується для трансформування аудіофайлів та зміни їх формату за допомогою відповідної бібліотеки. На рисунку також наведено фрагмент коду, що забезпечує збереження отриманого аудіоповідомлення [36].

```
at = audiotranscodec.Audiotranscodec()
at.transcode('{}.ogg'.format(name), '{}.wav'.format(name))
```

Рисунок 3.7 – Конвертація аудіофайлу з формату .ogg у .wav за допомогою Python

У поточній структурі проєкту тека teaser наразі не використовується, проте за необхідності її можна застосовувати для збереження тезисних описів або проміжних текстових даних. Файл assistant.py виконує роль інтеграційного модуля між сервісами Telegram та Google. Він слугує основною частиною системи, яка

відповідає за обробку вхідних повідомлень, взаємодію з API та надсилання відповідей. Для реалізації цієї взаємодії використовується бібліотека telebot, яка забезпечує ефективне з'єднання з Telegram API. Нижче представлено фрагмент коду, який виконує ініціалізацію та активацію Telegram-бота.

```
bot = telebot.TeleBot('7860910203:AAERPF5sHJqJqlMCjP2CoUCkkibZqlWdQ7g')
```

Рисунок 3.8 – Ініціалізація бота

Чат-бот також реалізує функцію вітання користувача при початку взаємодії. Такий механізм дозволяє створити зручне та дружнє середовище спілкування. Нижче наведено фрагмент коду, який відповідає за надсилання привітального повідомлення при активації бота:

```
@bot.message_handler(commands=['start'], content_types=['text'])
def send_welcome(mes):
    bot.send_message(mes.chat.id, "Hello, I'm Scrapper, your personal assistant.")
```

Рисунок 3.9 – Обробка команди /start

Основна логіка чат-бота відповідає за приймання та обробку повідомлень користувача, а також за генерацію відповідей. Цей функціонал реалізується через обробник повідомлень, що фіксує текстові та голосові запити. Нижче наведено фрагмент коду, який дозволяє опрацьовувати усі типи вхідних повідомлень [37]:

```
@bot.message_handler(content_types=['voice', 'text'])
def get_voice(mes):
    bot.send_message(mes.chat.id, 'Ok, let me check what you wrote.')
```

Рисунок 3.10 – Обробка текстових і голосових повідомлень

Функція bot.polling() забезпечує запуск основного циклу роботи бота, під час якого він очікує та обробляє вхідні повідомлення. У разі виникнення помилок, вони зберігаються у лог-файлі, що дозволяє здійснювати подальший аналіз. Нижче представлено фрагмент коду, який відповідає за запуск Telegram-бота та обробку винятків:

```
def telegram_polling():
    try:
        bot.polling(none_stop=True, timeout=60)
    except:
        traceback_error_string = traceback.format_exc()
        with open("Error.Log", "a") as myfile:
            myfile.write("\n\n\n" + time.strftime("%c") + "\n<<ERROR polling>>\n" + traceback_error_string + "\n<<ERROR polling>>\n")
        bot.stop_polling()
        time.sleep(10)
        telegram_polling()
```

Рисунок 3.11 – Основний цикл обробки повідомлень

Файли з розширенням .json у проєкті використовуються як службові – вони містять конфіденційні ключі доступу та параметри авторизації для взаємодії з сервісами Google, зокрема Google Cloud та Dialogflow. Файл df_connect.py виконує роль посередника між Telegram-ботом та платформою Dialogflow, забезпечуючи передачу текстових запитів і обробку відповідей від інтелектуального помічника. Нижче наведено фрагмент коду, що реалізує підключення та відправку запиту до Dialogflow [37]:

```
import dialogflow_v2 as dialogflow

def dialogflow_request(text, lang_code):
    session_client = dialogflow.SessionsClient()
    session = session_client.session_path('assistant-c723c', 'assistant-c723c')
    text_input = dialogflow.types.TextInput(text=text, language_code=lang_code)
    query_input = dialogflow.types.QueryInput(text=text_input)
    response = session_client.detect_intent(session=session, query_input=query_input)

    if response.query_result.intent.display_name == 'Default Fallback Intent':
        return None

    return response.query_result.fulfillment_text
```

Рисунок 3.12 – Підключення до Dialogflow

Файл Error.Log виконує роль системного журналу для фіксації помилок, які можуть виникати під час роботи чат-бота. Це дозволяє відстежити, у який саме момент сталася помилка, та вчасно локалізувати проблему. Нижче представлено фрагмент коду, який реалізує механізм запису помилок у лог-файл:

```
except:
    traceback_error_string = traceback.format_exc()
    with open("Error.Log", "a") as myfile:
        myfile.write("\n\n\n" + time.strftime("%c") + "\n<<ERROR polling>>\n" + traceback_error_string + "\n<<ERROR polling>>\n")
```

```
Traceback (most recent call last):

File "/usr/local/lib/python3.5/dist-packages/urllib3/connectionpool.py", line 387, in _make_request
    six.raise_from(e, None)

File "<string>", line 2, in raise_from

File "/usr/local/lib/python3.5/dist-packages/urllib3/connectionpool.py", line 383, in _make_request
    httplib_response = conn.getresponse()

File "/usr/lib/python3.5/http/client.py", line 1197, in getresponse
    response.begin()

File "/usr/lib/python3.5/http/client.py", line 297, in begin
    version, status, reason = self._read_status()

File "/usr/lib/python3.5/http/client.py", line 258, in _read_status
    line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")

File "/usr/lib/python3.5/socket.py", line 575, in readinto
    return self._sock.recv_into(b)

File "/usr/local/lib/python3.5/dist-packages/urllib3/contrib/pyopenssl.py", line 294, in recv_into
    raise timeout('The read operation timed out')

socket.timeout: The read operation timed out
```

Рисунок 3.13 – Фрагмент коду логування помилок у файл Error.Log

Файли test.py та example.py виконують роль допоміжних скриптів для тестування функціональності окремих компонентів системи. Вони застосовуються для виявлення суперечностей між модулями, перевірки коректності роботи функцій та налагодження окремих частин проєкту.

Файл g_search.py реалізує функціонал надсилання текстових запитів до Google Search API. Він отримує текстове повідомлення користувача, передає його у систему пошуку та повертає відповідь у вигляді списку результатів. Нижче наведено фрагмент відповідного коду [39]:

```
def search_result(query):
    print('GOT QUERY:', query)
    search_results = google.search(query, num_page)
    return_results = []

    for result in search_results:
        return_results.append('{}\n{}\nURL: {}'.format(result.name, result.description, result.link))
        print(result.name, result.link, result.description)

    result = {'response': return_results}
    return result
```

Рисунок 3.14 – Фрагмент коду надсилання запиту до Google Search API (файл g_search.py)

Файл send_text.py відповідає за взаємодію з Google Assistant API. Його основна функція — відправлення запиту користувача та отримання відповіді у текстовому вигляді. Перед передачею запиту виконується ініціалізація параметрів: налаштовується формат аудіо, мовні та діалогові параметри, а також інформація про модель. Нижче подано фрагмент коду, який демонструє конфігурацію запиту до Google Assistant [39]:

```
def iter_assist_request():
    config = embedded_assist_pb2.AssistConfig(
        audio_out_config = embedded_assist_pb2.AudioOutConfig(
            encoding='LINEAR16',
            sample_rate_hertz=16000,
            volume_percentage=0,
        ),
        dialog_state_in = embedded_assist_pb2.DialogStateIn(
            language_code=ego.language_code,
            conversation_state=ego.conversation_state,
            is_new_conversation=ego.is_new_conversation,
        ),
        device_config = embedded_assist_pb2.DeviceConfig(
            device_id=ego.device_id,
            device_model_id=ego.device_model_id,
        ),
        text_query=txt_query,
    )
```

Рисунок 3.15 – Фрагмент коду конфігурації запиту до Google Assistant API (файл send_text.py)

Наступним етапом після формування конфігурації є встановлення з’єднання з Google Assistant API та передача сформованого запиту. У результаті виклику асистента ми отримуємо відповідь, яку можна обробити або вивести користувачу. Нижче наведено фрагмент коду, який відповідає за передачу запиту до Google Assistant API та отримання результату [39]:

```
grpc_channel = google.auth.transport.grpc.secure_authorized_channel(
    credentials, http_request, api_endpoint
)
logging.info('Connecting to %s', api_endpoint)

with SampleTextAssist(lang, device_model_id, device_id, context,
                      grpc_channel, grpc_deadline) as assist:
    while True:
        query = user_input
        reply_text, reply_html = assist.assist(text_query=query)
        return reply_text
```

Рисунок 3.16 – Фрагмент коду надсилання запиту та отримання відповіді від Google Assistant API

# ВИСНОВКИ

У ході виконання даної кваліфікаційної роботи було розглянуто сучасні підходи до створення чат-ботів з елементами штучного інтелекту, а також проаналізовано їх значення в контексті автоматизації процесів обслуговування користувачів. Особливу увагу було приділено інтернет-комунікаційним технологіям, які забезпечують інтеграцію чат-ботів з платформами Google та месенджером Telegram.

Проаналізовано ринок актуальних рішень, що функціонують у сфері клієнтського сервісу, а також обґрунтовано вибір технологій, що дозволяють реалізувати високоефективного віртуального асистента. Зокрема, виокремлено такі технології, як Google Assistant, Google Search API, Dialogflow та Google Speech-to-Text, які стали основою для побудови розумної відповіді чат-бота.

Розроблено функціональну архітектуру інформаційної системи, створено UML-діаграми, контекстні схеми та виконано декомпозицію основних процесів системи. Це дозволило побудувати логічну структуру програмного продукту та забезпечити зручність подальшого масштабування і підтримки.

Було продемонстровано реалізацію програмного продукту мовою Python із застосуванням бібліотек telebot, dialogflow, google-search, а також реалізовано обробку голосових запитів користувачів. Також створено скрипти для конвертації аудіофайлів, обробки команд користувача, логування помилок і обміну інформацією з API.

Виокремлено основні ризики, що можуть впливати на якість і строки реалізації проєкту, та побудовано відповідні матриці ймовірності, втрат і ризиків. На основі аналізу було виділено критичні моменти, які потребують особливої уваги на етапах розробки та тестування.

Реалізовано чат-бота, здатного здійснювати пошук інформації за допомогою голосових і текстових запитів, що значно підвищує зручність взаємодії кінцевого користувача з інформаційними сервісами. Результати цієї роботи підтвердили ефективність обраного підходу до створення інтелектуального помічника.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Артим В.І. Інтелектуальний чат-бот для підтримки клієнтів у сфері туристичного бізнесу // Кваліфікаційна робота. – Чорноморський національний університет імені Петра Могили, 2024. – 53 с.
2. Завальнюк М.Є., Ковалюк О.О. Використання чат-ботів у бізнесі // Матеріали конференції. – Вінницький національний технічний університет, 2023.
3. Крупа А. Технологія чат-бот як чинник комп'ютерно-посередницької комунікації цифрового суспільства // Humanities Studies, 2022.
4. Боднар Д.В. Розробка чат-бота для автоматизації обслуговування клієнтів // Кваліфікаційна робота. – Тернопільський національний технічний університет імені Івана Пулюя, 2024.
5. Савченко І.В. Використання чат-ботів у торгівлі будівельними матеріалами // Науковий вісник, 2023.
6. Нікітченко М.О. Дослідження і аналіз можливостей чат-боту зі штучним інтелектом // Науковий журнал, 2023.
7. Козлов С.І., Якушко А.В. Використання технологій чат-ботів в умовах цифрової трансформації бізнесу // Матеріали конференції. – НТУУ "КПІ", 2022.
8. Лаврова О.В. Особливості застосування чат-ботів на основі штучного інтелекту у фінансовому секторі // Економіка і регіон, 2022.
9. Крупа А.Г. Управління ІТ-інфраструктурою підприємства як чинник ефективності цифрового менеджменту. Збірник наукових праць студентів, аспірантів, докторантів і молодих вчених «Молода наука-2022» : у 5 т. / Запорізький національний університет. Запоріжжя : ЗНУ, 2022. Т.5. С. 59-60.
10. Крупа А.Г. Формування концепції цифрової політики європи як чинник стабільного розвитку. Стратегічні пріоритети розвитку підприємництва,торгівлі та біржової діяльності: матеріали ІІІ-ої Міжнародної науково-практичної конференції, Запоріжжя,11-12 травня 2022 року. Запоріжжя : НУ «Запорізька політехніка», 2022. С. 317-319.

11. Крупа А.Г. Місце і роль чат-ботів у цифровому суспільстві. Економіко-правові та соціально-технічні напрями еволюції цифрового суспільства: матеріали міжнародної науково-практичної конференції: у 2 т. Том 2. Дніпро: Університет митної справи та фінансів, 2022. С. 443-445.

12. Крупа А.Г. Цифрова трансформація промисловості у країнах ЄС як чинник удосконалення суспільства INDUSTRY 4.0. XV Міжнародна науково-практична конференція «International scientic innovations in human life». 2022. С. 313-322.

13. Teslenko, Tatyana & Zadoia, Viacheslav (2021). Breakthrough technologies as a factor of formation of information economy in the conditions of digitalization. Humanities studies: Collection of Scientic Papers. Zaporizhzhia : Zaporizhzhia National University.7 (84). 48-57.

14. Teslenko, Tatyana & Zadoia, Viacheslav. Breakthrough technologies as a factor of formation of information economy in the conditions of digitalization. Humanities studies: Collection of Scienti c Papers. Zaporizhzhia: Zaporizhzhia National University, 2021. 7 (84), P. 48-57.

15. Петренко О.М. Використання штучного інтелекту в чат-ботах для підвищення якості клієнтського сервісу // Збірник наукових праць Харківського національного економічного університету, 2023.

16. Sharma, Ruchir (2018). Advanced countries. Waiting for a new"economic miracle" / trans. from English Andriy Ishchenko. Kyiv: Nash format. 296.

17. Schwab, Claus (2019). The fourth industrial revolution, Shaping the fourth industrial revolution. Kharkiv: Family Leisure Club. 426.

18. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. – 4th ed. – Pearson, 2020. – 1136 p.

19. Jurafsky D., Martin J. H. Speech and Language Processing. – 3rd ed. – Draft, 2023. – 1320 p.

20. Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press, 2016. – 775 p.

21. VanderPlas J. Python Data Science Handbook. – O’Reilly Media, 2016.

22. Chollet F. Deep Learning with Python. – 2nd ed. – Manning Publications, 2021.

23. Brownlee J. Deep Learning for Natural Language Processing. – Machine Learning Mastery, 2017.

24. Raschka S., Mirjalili V. Python Machine Learning. – Packt Publishing, 2022.

25. Dey L. Natural Language Processing with Transformers. – O'Reilly, 2022.

26. Ghosh S. Developing Bots with Python. – Apress, 2017.

27. Zimmerman J. Building Chatbots with Python. – Packt Publishing, 2021.

28. Brown T. et al. Language Models are Few-Shot Learners. // NeurIPS, 2020.

29. Devlin J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. // arXiv:1810.04805, 2018.

30. Radford A. et al. Improving Language Understanding by Generative Pre-Training. // OpenAI, 2018.

31. Vaswani A. et al. Attention is All You Need. // NeurIPS, 2017.

32. Vinyals O. et al. A Neural Conversational Model. // ICML, 2015.

33. Serban I. et al. Building End-to-End Dialogue Systems Using Generative Hierarchical Neural Network Models. // AAAI, 2016.

34. Zhang Y. et al. DialoGPT: Large-Scale Generative Pretraining for Conversational Response Generation. // Microsoft Research, 2019.

35. Gao J. et al. Neural Approaches to Conversational AI. // Foundations and Trends in Information Retrieval, 2019.

36. Huang M. et al. A Survey on Dialogue Systems: Recent Advances and New Frontiers. // ACM Computing Surveys, 2020.

37. Luger E., Sellen A. "Like Having a Really Bad PA": The Gulf between User Expectation and Experience of Conversational Agents. // CHI Conference, 2016.

38. Wollny, S., Schneider, J., Di Mitri, D., Weidlich, J., Rittberger, M., & Drachsler, H. (2021). Are we there yet?—A systematic literature review on chatbots in education. Frontiers in Artificial Intelligence, (4), 654924.

Deng, X., & Y u, Z. (2023). A meta-analysis and systematic review of the effect of chatbot technology use in sustainable education. Sustainability, 15(4), 2940.

# ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

**Державний університет інформаційно -комунікаційних технологій**

**Кафедра інженерії програмного забезпечення автоматизованих систем**

Кваліфікаційна робота на тему:

«ЧАТ-БОТ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЗАЦІЇ ОБСЛУГОВУВАННЯ КЛІЄНТІВ»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
освітньо -професійної програми Інформаційні системи та технології

Виконав: ЦИБУЛЬКО Максим ІСД -42
Науковий керівник роботи: Вікторія ЖЕБКА

**КИЇВ - 2025**

***Актуальність дослідження*** У сучасному світі автоматизація обслуговуванняклієнтів стає критично важливою складовою ефективного бізнесу. З розвитком цифрових технологій компанії прагнуть зменшити витрати на персонал, підвищити якість сервісу та забезпечити безперервнийзв'язокіз клієнтами.

***Метою роботи*** є розробка та реалізація інтелектуального чат-бота, здатного автоматизувати процес обслуговуванняклієнтів, використовуючитехнології штучногоінтелекту,Google Assistantта Google Search API.

***Об'єктом дослідження*** інтелектуальна інформаційна система для автоматизації обслуговуванняклієнтів на основі чат-бота.

***Предмет дослідження*** – методи розробки та впровадження штучного інтелекту в чат-боти для підвищення ефективності автоматизованої взаємодії з користувачами

Для досягнення цієї мети в роботі необхідно виконати наступні **завдання:**

- аналіз сучасних технологій та методів створення інтелектуальних чат-ботів, зокрема технологій обробки природноїмови (NLP), машинного навчання та штучногоінтелекту;
- дослідження платформ та інструментів для розробки чат-ботів, таких як google assistant, google search api, telegram api, dialogflow тощо;
- визначення ключових вимог до системи чат-бота, враховуючиї функціональність, швидкість обробки запитів, інтеграцію з зовнішніми сервісами та зручність використання для клієнтів;
- оцінка ефективності роботирозробленогочат-бота за допомогою тестуваннята аналізу його продуктивності;
- проведення порівняльногоаналізу різних підходів до реалізації чат-ботів та визначення оптимального рішення для автоматизації обслуговуванняклієнтів;

формулювання рекомендацій щодо впровадження інтелектуальних чатботів у бізнес-процеси для покращення якості обслуговування клієнтів.

## Методита технологіїстворенняінтелектуальнихчат-ботів

3

Створення інтелектуальних чат-ботів – складний, багаторівневий процес, який поєднує в собі сучасні досягнення у галузі штучного інтелекту, машинного навчання, обробки природної мови (NLP), а також використання програмних фреймворків, сервісів і хмарних платформ. Головна мета такого чат-бота – не просто реагувати на заздалегідь прописані ключові слова, а розуміти контекст, інтерпретувати наміри користувача, адаптуватися до нових сценаріїв і навчатися з часом.

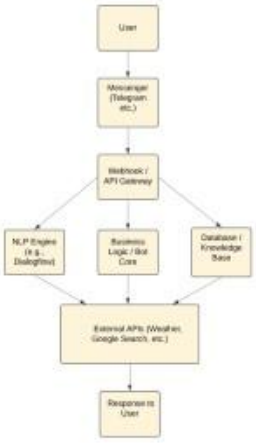



Рисунок 1– Типова архітектуру інтелектуального чат
-

Рисунок 2 – Схема взаємодії користувача з Google Assistant у процесі роботи чатбота

## Основні принципи роботи Google Search API тайого роль у чат-ботах

4

Інтеграція чат-бота з Google Search API дозволяє отримувати актуальну інформацію з Інтернету у реальному часі, формуючи відповіді на запити, що виходять за межі внутрішньої бази знань. Це підвищує гнучкість і ефективність бота у взаємодії з користувачем.

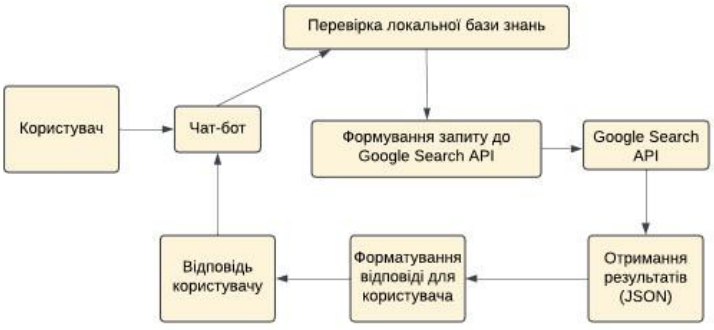



Рисунок 3– Схема використання Google Search API у чат-боті
для динамічного пошуку інформації

## Постановка проблеми, мета та завдання дослідження



Розробка інтелектуального чат-бота для автоматизації обслуговування клієнтів із використанням NLP, Google Assistant та Google Search API.

🔍 **Основні завдання:**

• Аналіз сучасних технологій чат -ботів

• Використання NLP і ML для обробки запитів

• Розробка архітектури з інтеграцією сторонніх сервісів

• Реалізація та тестування прототипу

💡 **Переваги бота:**

• Персоналізація обслуговування

• Робота 24/7 без втрати якості

• Пошук у реальному часі ( Google Search API)

• Голосове керування ( Google Assistant)

• Можливість масштабування та інтеграції з CRM

## Аналіз ринку сучасних чат-ботів та їх роль в автоматизації обслуговування клієнтів



Рисунок 4– Логотип чат боту ChatGPT


Рисунок 5– чат бот Erica в банківській сфері

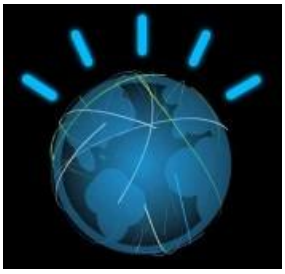


Рисунок 7– Логотип Watson Assistant


Рисунок 6 – Логотип Buoy Health

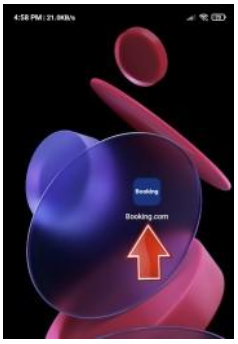



Рисунок 8– Додаток чат боту Booking.com

Рисунок 9– Duolingo Bot


Рисунок 11 –Контекстна діаграма (IDEF0), що відображає загальну модель функціонування розроблюваної інформаційної системи


Рисунок 12 – Діаграма варіантів використання (Use Case) для взаємодії з чат-ботом

## Етапи розробки інтелектуальної системи чат-бота



Чат-бот — це програмна система, що забезпечує зручнувзаємодію з користувачему текстовому або голосовому форматі та виконує автоматизований пошук інформації. Метою проєкту є створення багатофункціонального асистента, який через інтеграцію з Google Search API та Google Assistant надає результатипошуку у зручному вигляді. Для організації розробкивикористано методології SMART та WBS, що дозволило чітко структурувати завдання та забезпечити ефективне управління проєктом Реалізована система підтримує голосове введення, працює в реальному часі та легко масштабується.

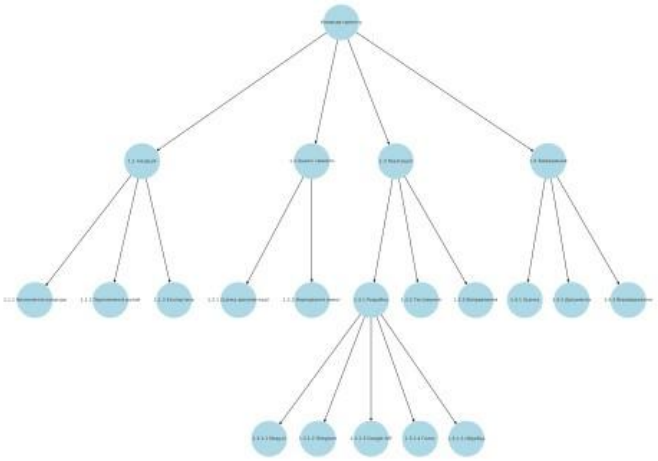


Рисунок 13– Ієрархічна WBS-структура інформаційної системи

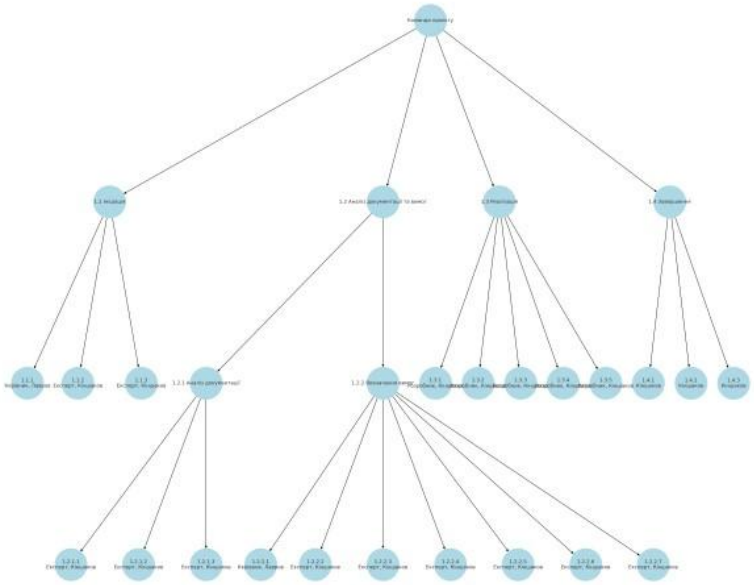


Рисунок 14– Структура відповідальності учасників розробки (OBS)

Етапи розробки інтелектуальної системи чат- бота

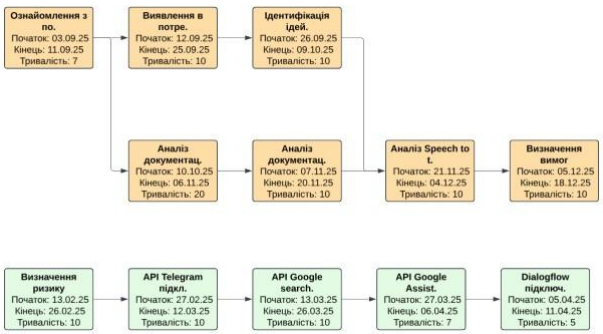



Рисунок 15– Побудова логічної мережі завдань методом PDM


Рисунок 16– Продовження логічної мережі завдань методом PDM

## Етапи розробки інтелектуальної системи чат-бота

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 |
|---|---|---|---|---|---|---|---|
| Слабо ймовірний | | | | | | | |
| Малоймовірний | | | | | | | |
| Ймовірний | | | | | | | |
| Дуже ймовірний | | | | | | | |
| Майже неминучий | | | | | | | |

Таблиця 1 Рівень ймовірності виникнення ризиків у процесі реалізації проєкту

## Структурна модель функціонування чат- бота

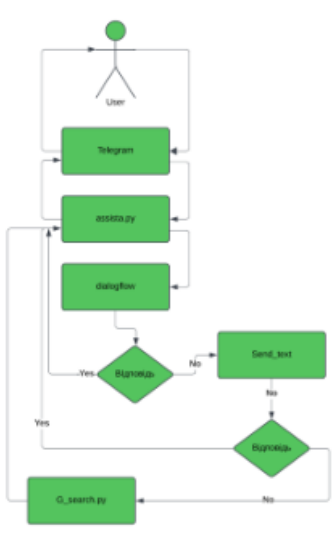



Рисунок 17 – Діаграма прецедентів

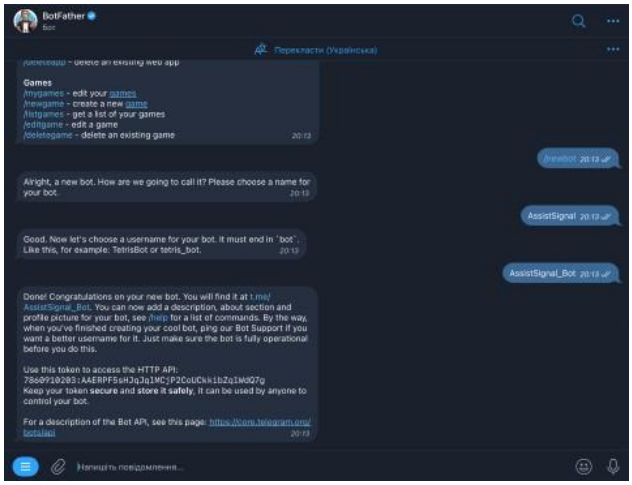



Рисунок 18 – Генерація унікального токена доступу до Telegram API

**Структурна модель функціонування чат- бота**

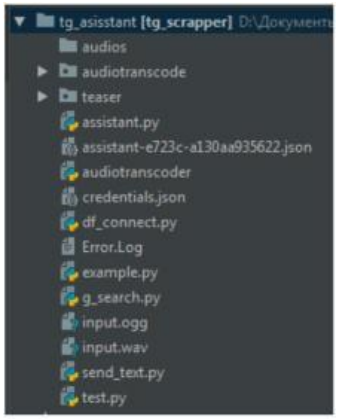



Рисунок 19 – Архітектура директорій і файлів проєкту Telegram-бота

```
at = audiotranscodec.Audiotranscoder()
at.transcode('{}.ogg'.format(name), '{}.wav'.format(name))
```

Рисунок 20 – Конвертація аудіофайлу з формату .ogg у .wav за допомогою Python

```
bot = telebot.TeleBot('7860910203:AAERPF5sHJqJqlMCjP2CoUCkkibZqlWdQ7g')
```

Рисунок 21 – Ініціалізація бота

```
@bot.message_handler(commands=['start'], content_types=['text'])
def send_welcome(mes):
    bot.send_message(mes.chat.id, "Hello, I'm Scrapper, your personal assistant.")
```

Рисунок 22 – Обробка команди / start

```
@bot.message_handler(content_types=['voice', 'text'])
def get_voice(mes):
    bot.send_message(mes.chat.id, 'Ok, let me check what you wrote.')
```

Рисунок 23 – Обробка текстових і голосових повідомлень

```
import dialogflow_v2 as dialogflow

def dialogflow_request(text, lang_code):
    session_client = dialogflow.SessionsClient()
    session = session_client.session_path('assistant-c723c', 'assistant-c723c')
    text_input = dialogflow.types.TextInput(text=text, language_code=lang_code)
    query_input = dialogflow.types.QueryInput(text=text_input)
    response = session_client.detect_intent(session=session, query_input=query_input)

    if response.query_result.intent.display_name == 'Default Fallback Intent':
        return None

    return response.query_result.fulfillment_text
```

Рисунок 25 – Підключення до Dialogflow

```
def telegram_polling():
    try:
        bot.polling(none_stop=True, timeout=60)
    except:
        traceback_error_string = traceback.format_exc()
        with open("Error.Log", "a") as myfile:
            myfile.write("\n\n\n" + time.strftime("%c") + "\n<<ERROR polling>>\n" + traceback_error_string + "\n<<ERROR polling>>\n")
        bot.stop_polling()
        time.sleep(10)
        telegram_polling()
```

Рисунок 24 – Основний цикл обробки повідомлень

**Структурна модель функціонування чат- бота**

```
except:
    traceback_error_string = traceback.format_exc()
    with open("Error.Log", "a") as myfile:
        myfile.write("\n\n\n" + time.strftime("%c") + "\n<<ERROR polling>>\n" + traceback_error_string + "\n<<ERROR polling>>\n")
```

```
Traceback (most recent call last):

File "/usr/local/lib/python3.5/dist-packages/urllib3/connectionpool.py", line 387, in _make_request
    six.raise_from(e, None)

File "<string>", line 2, in raise_from

File "/usr/local/lib/python3.5/dist-packages/urllib3/connectionpool.py", line 383, in _make_request
    httplib_response = conn.getresponse()

File "/usr/lib/python3.5/http/client.py", line 1197, in getresponse
    response.begin()

File "/usr/lib/python3.5/http/client.py", line 297, in begin
    version, status, reason = self._read_status()

File "/usr/lib/python3.5/http/client.py", line 258, in _read_status
    line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")

File "/usr/lib/python3.5/socket.py", line 575, in readinto
    return self._sock.recv_into(b)

File "/usr/local/lib/python3.5/dist-packages/urllib3/contrib/pyopenssl.py", line 294, in recv_into
    raise timeout('The read operation timed out')

socket.timeout: The read operation timed out
```

Рисунок 25– Фрагмент коду логуванняпомилок у файл Error.Log

```
def search_results(query):
    print('GOT QUERY: ', query)
    search_results = google.search(query, num_page)
    return_results = []

    for result in search_results:
        return_results.append({'title': result.name, 'description': result.description, 'url': result.link})
        print(result.name, result.link, result.description)

    result = {'response': return_results}
    return result
```

Рисунок 26– Фрагмент коду надсилання запиту до Google Search API (файл g_search.py)

```
grpc_channel = google.auth.transport.grpc.secure_authorized_channel(
    credentials, http_request, api_endpoint
)
logging.info('Connecting to %s', api_endpoint)

with SampleTextAssist(lang, device_model_id, device_id, context,
                      grpc_channel, grpc_deadline) as assist:
    while True:
        query = user_input
        reply_text, reply_html = assist.assist(text_query=query)
        return reply_text
```

Рисунок 28 –Фрагмент коду надсилання запиту та отримання відповіді від Google AssistantAPI

```
def iter_assist_request():
    config = embedded_assist_pb2.AssistConfig(
        audio_out_config = embedded_assist_pb2.AudioOutConfig(
            encoding='LINEAR16',
            sample_rate_hertz=16000,
            volume_percentage=0,
        ),
        dialog_state_in = embedded_assist_pb2.DialogStateIn(
            language_code=ego.language_code,
            conversation_state=ego.conversation_state,
            is_new_conversation=ego.is_new_conversation,
        ),
        device_config = embedded_assist_pb2.DeviceConfig(
            device_id=ego.device_id,
            device_model_id=ego.device_model_id,
        ),
        text_query=txt_query,
    )
```

Рисунок 27– Фрагмент коду конфігурації запиту до Google AssistantAPI (файл send_text.py)

# ВИСНОВКИ



- Розглянуто сучасні підходи до створення чат-ботів із елементами штучного інтелекту та їхню роль в автоматизації обслуговування користувачів. Проаналізовано інтернет-комунікаційні технології, що забезпечують інтеграцію з Google-сервісами та месенджером Telegram, зокрема Google Assistant, Google Search API, Dialogflow та Google Speech-to-Text.
- Виконано обґрунтування вибору технологічного стеку, розроблено архітектуру інформаційної системи, створено UML-діаграми, контекстні схеми та реалізовано логіку роботи голосових і текстових запитів. Продемонстровано практичну реалізацію чат-бота мовою Python із використанням відповідних бібліотекі API.
- Визначено ризики, які можуть впливати на якість та строки реалізації проєкту, побудовано відповідні матриці для їх аналізу та контролю. Реалізована система підтвердила ефективність обраного підходу та забезпечила зручну й інтуїтивну взаємодію з користувачем у режимі реального часу.
- Запропоноване рішення демонструє практичну цінність чат-бота як цифрового асистента з голосовим та текстовим введенням, що має потенціал до масштабування та подальшого вдосконалення.

Дякую за увагу!
Доповідь закінчено