

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

на тему:

“Дослідження технологій розпізнавання мовних команд Telegram ботом”

Виконав: студент групи 6144 м

Михеєнко Б.О.



(підпис)

Керівник роботи:

доцент кафедри ІУСТ, к.т.н.,

(посада, науковий ступінь вчене звання)



к.т.н., доц *Кроптя А.В*

(підпис)

м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 126 "Інформаційні системи та технології"
Освітня програма "Інформаційні системи та технології"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми


к.т.н, доц. Чернова Лб.С.

"___" _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "магістр"

Студенту Михесенку Богдану Олександровичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження технологій розпізнавання мовних команд Telegram ботом

Керівник роботи к.т.н., доц. Кроптя А.В.

Затверджені наказом ректора № 1355-уч. від "14" листопада 2025 року.

2. Термін подання роботи: 15 грудня 2025 р.

3. Вихідні дані до проєкту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали наукового стажування

4. Перелік питань, що належать до розробки (найменування розділів)

Аналіз технологій розпізнавання мовлення та сфери їх застосування.
Проектування системи розпізнавання голосових замовлень у Telegram.
Реалізація і впровадження системи. Економічне обґрунтування проєкту.
Охорона праці. Охорона навколишнього середовища.

5. Перелік презентаційних матеріалів Use-Case діаграма системи голосових замовлень через Telegram-бот, Концептуальна модель системи розпізнавання голосових замовлень, Логічна архітектура системи, Фізична архітектура системи, Діаграма послідовностей для сценарію «Замовлення через голосову команду

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	к.т.н, доц Кроптя А.В	05.09.2024	22.12.2024
2	к.т.н, доц Кроптя А.В	23.12.2024	15.06.2025
3	к.т.н, доц Кроптя А.В	15.06.2025	25.09.2025
4	к.т.н, доц Кроптя А.В	25.09.2025	14.11.2025
5	к.т.н, доц Кроптя А.В	25.09.2025	14.11.2025
6	к.т.н, доц Кроптя А.В	25.09.2025	14.11.2025

7. Дата видачі завдання 05 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

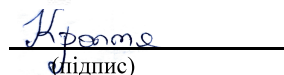
№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження та аналіз предметної галузі	22.12.2024	
2	Розробка концепції	15.03.2025	
3	Розділ з розробки проєктних рішень по системі (та її частинах)	15.06.2025	
4	Реалізація проєкту та розробка робочої документації	25.09.2025	
5	Обчислення економічної ефективності	14.11.2025	
6	Вирішення питань охорони праці та навколишнього середовища	14.11.2025	
7	Оформлення пояснювальної записки	14.11.2025	
8	Подання магістерської роботи на перевірку на плагіат	16.11.2025	
9	Подання магістерської роботи рецензенту	20.11.2025	
10	Подання магістерської роботи на державний захист	15.12.2025	

Студент



Михеснко Б.О.

Керівник роботи


(підпис)

к.т.н, доц Кроптя А.В
(ПІБ)

АНОТАЦІЯ

У дипломній роботі досліджується процес розробки системи розпізнавання голосових замовлень у середовищі Telegram-бота з метою підвищення ефективності та зручності взаємодії користувачів із сервісами автоматизації замовлень. Основна увага приділяється застосуванню сучасних технологій автоматичного розпізнавання мовлення (ASR) та інтеграції голосового інтерфейсу з Telegram Bot API. Метою дослідження є підвищення ефективності автоматичного розпізнавання мовлення шляхом розробки Telegram-бота, здатного приймати голосові замовлення, перетворювати їх у текстовий формат і автоматично обробляти відповідно до заданих бізнес-правил.

Об'єктом дослідження є процес автоматизації управління замовленнями з використанням чат-ботів із підтримкою голосового введення. Предметом дослідження є методи, моделі та програмні засоби інтеграції технологій автоматичного розпізнавання мовлення в Telegram-боти для обробки голосових команд.

У процесі дослідження проведено аналіз сучасних методів і алгоритмів розпізнавання мовлення, досліджено хмарні, локальні та гібридні ASR-рішення, а також визначено їхні переваги та обмеження. На основі результатів дослідження спроектовано архітектуру системи Telegram-бота з підтримкою голосового інтерфейсу, що включає модулі обробки аудіоданих, інтеграції з API розпізнавання мовлення, бізнес-логіки обробки замовлень і механізми захисту даних.

Наукова новизна роботи полягає в узагальненні та практичній реалізації підходів до інтеграції технологій автоматичного розпізнавання мовлення з Telegram-ботами, а також у розробці гібридної архітектури обробки голосових команд, яка поєднує можливості хмарних і локальних ASR-рішень з метою забезпечення балансу між точністю розпізнавання, швидкістю системи та безпекою даних.

За результатами проведеного дослідження розроблено програмний прототип Telegram-бота, що підтримує приймання голосових і текстових команд,

створення, зміну та скасування замовлень, а також надання користувачам підтверджень у зручній формі. Проведене тестування підтвердило працездатність, стабільність та ефективність використання голосових команд у процесі автоматизації замовлень.

Розроблена система може бути використана як базове рішення для впровадження голосових інтерфейсів у сервісах обслуговування клієнтів, зокрема в галузях торгівлі, доставки та інформаційної підтримки. Отримані результати підтверджують доцільність використання сучасних ASR-технологій у поєднанні з Telegram Bot API для створення масштабованих, зручних та ефективних систем автоматизації замовлень із голосовим управлінням. Подальший розвиток роботи може передбачати розширення мовної підтримки, підвищення точності розпізнавання та вдосконалення контекстної обробки голосових команд.

Дипломна робота виконана українською мовою та містить 86 сторінок, 19 рисунків, 13 таблиць та 4 додатки.

Ключові слова: розпізнавання мовлення; ASR; Telegram-бот; голосові команди; Whisper; Google; Vosk; Java Spring Boot; голосовий інтерфейс; автоматизація.

ANNOTATION

The diploma thesis examines the process of developing a voice order recognition system in the environment of a Telegram bot in order to increase the efficiency and convenience of user interaction with order automation services. The main focus is placed on the application of modern automatic speech recognition (ASR) technologies and the integration of a voice interface with the Telegram Bot API. The aim of the study is to improve the efficiency of automatic speech recognition through the development of a Telegram bot capable of receiving voice orders, converting them into text format, and automatically processing them in accordance with predefined business rules.

The object of the research is the process of automating order management using chatbots with voice input support. The subject of the research includes methods, models, and software tools for integrating automatic speech recognition technologies into Telegram bots for processing voice commands.

During the research, an analysis of modern speech recognition methods and algorithms was conducted, including cloud-based, local, and hybrid ASR solutions, and their advantages and limitations were identified. Based on the research results, the architecture of a Telegram bot system with voice interface support was designed, incorporating modules for audio data processing, integration with speech recognition APIs, business logic for order processing, and data protection mechanisms.

The scientific novelty of the thesis lies in the generalization and practical implementation of approaches to integrating automatic speech recognition technologies with Telegram bots, as well as in the development of a hybrid architecture for processing voice commands that combines cloud-based and local ASR solutions to achieve a balance between recognition accuracy, system performance, and data security.

As a result of the conducted research, a software prototype of a Telegram bot was developed that supports the reception of voice and text commands, the creation, modification, and cancellation of orders, and the provision of user confirmations in a

convenient form. Testing confirmed the functionality, stability, and effectiveness of using voice commands in the order automation process.

The developed system can be used as a basic solution for implementing voice interfaces in customer service systems, particularly in the fields of commerce, delivery, and information support. The obtained results confirm the feasibility of using modern ASR technologies in combination with the Telegram Bot API to create scalable, user-friendly, and efficient voice-controlled order automation systems. Further development of the work may include expanding language support, improving recognition accuracy, and enhancing the contextual processing of voice commands.

The diploma thesis is written in Ukrainian and contains 86 pages, 19 figures, 13 tables, and 4 appendices.

Keywords: information system, team interaction management, communication within the organisation, digital transformation, system adaptability, data security, integration of project tools.

ЗМІСТ

ВСТУП	3
РОЗДІЛ І. АНАЛІЗ ТЕХНОЛОГІЙ РОЗПІЗНАВАННЯ МОВЛЕННЯ ТА СФЕРИ ЇХ ЗАСТОСУВАННЯ	6
1.1. Історія та сучасний стан технологій розпізнавання мовлення	6
1.2. Методи та алгоритми розпізнавання	8
1.3. Хмарні та локальні рішення	11
1.4. Концепція системи у сфері автоматизації замовлень	13
1.5. Постановка задачі	17
Висновки до розділу 1	19
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ГОЛОСОВИХ ЗАМОВЛЕНЬ У TELEGRAM.....	22
2.1. Архітектура системи з урахуванням голосового інтерфейсу	22
2.2. Вибір платформи розпізнавання мовлення.....	27
2.3. Інтеграція з Telegram API	28
2.4. Модель обробки, зберігання і захисту даних	31
2.5. Вимоги до безпеки та захисту персональних даних	34
Висновки до розділу 2	36
РОЗДІЛ 3. РЕАЛІЗАЦІЯ І ВПРОВАДЖЕННЯ СИСТЕМИ	38
3.1. Розробка Telegram-бота з підтримкою голосових команд	38
3.2. Інтеграція API розпізнавання мовлення.....	41
3.3. Алгоритми перевірки і фільтрації вхідних повідомлень	43
3.4. Тестування.....	46
3.5. Логування, аудит та обробка помилок	49
Висновки до розділу 3	51
РОЗДІЛ 4. ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЄКТУ	53
4.1. Мета та методика економічного аналізу	53
4.2. Розрахунок витрат на розробку та впровадження.....	55
4.3. Оцінка ефективності.....	57
4.4. Порівняння вартості з аналогами на ринку.....	59
Висновки до розділу 4	62
РОЗДІЛ 5. ОХОРОНА ПРАЦІ	64
5.1. Організація безпечної розробки програмного забезпечення	64

	2
5.2 Ризики при роботі з персональними даними та засоби їх мінімізації.....	66
5.3 Енергоспоживання ІТ-інфраструктури, екологічні аспекти	68
5.4 Загальні вимоги охорони праці для офісного середовища	70
Висновки до розділу 5	72
РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА	75
Висновки до розділу 6	77
ВИСНОВКИ.....	79
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	81
ДОДАТКИ.....	85

ВСТУП

У сучасному світі стрімкого розвитку інформаційних технологій усе більшої популярності набувають системи, що забезпечують природну та інтуїтивну взаємодію людини з комп'ютером. Одним із найдинамічніших напрямів такої взаємодії є технології розпізнавання мовлення, які дозволяють користувачам керувати пристроями й програмами за допомогою голосових команд. Завдяки розвитку алгоритмів штучного інтелекту, нейронних мереж та обчислювальних потужностей, системи розпізнавання мовлення перейшли від експериментальних рішень до високоточних інструментів, що інтегровані у повсякденні цифрові сервіси – від голосових помічників до інтелектуальних чат-ботів.

Особливого значення ці технології набувають у сфері автоматизації обслуговування клієнтів, де ключовими вимогами є швидкість, зручність, персоналізація та доступність. У цьому контексті ефективним інструментом виступають чат-боти, здатні приймати замовлення, проводити консультації, надавати інформацію чи виконувати команди користувача без участі оператора. Серед різних платформ для створення ботів вирізняється Telegram, який поєднує високий рівень безпеки, відкритий API, гнучкі можливості інтеграції та стабільність роботи. Це сприяє широкому використанню Telegram-ботів у сферах торгівлі, доставки, освіти, медицини та технічної підтримки.

Однак переважна більшість існуючих Telegram-ботів взаємодіють із користувачами виключно через текстові повідомлення. Такий підхід обмежує зручність та швидкість користування, особливо у ситуаціях, коли введення тексту є ускладненим або неможливим. Використання голосових команд у Telegram-боті відкриває нові можливості для підвищення рівня автоматизації, ергономічності та інклюзивності цифрових сервісів. Голосовий інтерфейс здатний суттєво скоротити час взаємодії, зменшити кількість помилок, підвищити точність формування замовлень і зробити процес спілкування з ботом максимально природним.

Таким чином, розроблення системи розпізнавання голосових команд для Telegram-бота є актуальним завданням сучасних інформаційних технологій, що

відповідає глобальним тенденціям розвитку голосових інтерфейсів та інтелектуальної автоматизації сервісів.

Метою роботи є підвищення ефективності автоматичного розпізнавання мовлення (ASR) за допомогою розробки Telegram-бота, який здатен приймати голосові замовлення, трансформувати їх у текстовий формат і автоматично обробляти за заданими бізнес правилами.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. провести аналіз сучасних технологій розпізнавання мовлення та визначити їх сфери застосування;
2. дослідити методи, алгоритми й хмарні сервіси розпізнавання голосу;
3. спроектувати архітектуру системи Telegram-бота з голосовим інтерфейсом;
4. реалізувати інтеграцію Telegram API з обраною платформою розпізнавання мовлення;
5. розробити алгоритми обробки, фільтрації та захисту вхідних голосових повідомлень;
6. здійснити тестування системи, оцінити точність розпізнавання та ефективність взаємодії користувача з ботом;
7. провести економічне обґрунтування проєкту та аналіз його доцільності впровадження.

Об'єкт дослідження – процес автоматизованої взаємодії користувача з цифровими сервісами шляхом використання голосових команд у середовищі Telegram-бота.

Предмет дослідження – методи, алгоритми, архітектурні рішення та програмні засоби розпізнавання мовлення, що застосовуються для обробки голосових команд, інтеграції з Telegram API та автоматизації формування замовлень.

Наукова новизна полягає у: узагальненні та систематизації підходів до створення голосових інтерфейсів у чат-ботах, зокрема у контексті їх інтеграції з Telegram Bot API; розробленні та обґрунтуванні гібридної архітектури системи розпізнавання мовлення, що поєднує локальну та хмарну обробку аудіоданих для

забезпечення оптимального балансу між точністю, швидкістю та безпекою; запропонуванні моделі обробки голосових команд, яка включає алгоритми попередньої обробки аудіо, фільтрації, NLU-аналізу наміру та адаптації до прикладних сценаріїв автоматизації замовлень; формуванні комплексного підходу до впровадження голосового інтерфейсу у Telegram-бот, що враховує вимоги до продуктивності, масштабованості та захисту персональних даних; проведенні експериментальної оцінки точності та ефективності голосової взаємодії у прикладному середовищі автоматизації замовлень.

Дана робота складається з шести розділів, у яких послідовно розглянуто всі аспекти створення системи. У першому розділі наведено теоретичний аналіз технологій розпізнавання мовлення, методів і алгоритмів, а також особливості їх застосування у сфері автоматизації. У другому розділі описано архітектуру системи, принципи проектування та вибір технологічної платформи. У третьому розділі подано реалізацію Telegram-бота з підтримкою голосових команд, алгоритми обробки даних, тестування та результати роботи. Четвертий розділ присвячено економічному обґрунтуванню розробки, а п'ятий і шостий – питанням охорони праці та екологічної безпеки під час розробки програмного забезпечення. У додатку подано результати експериментального дослідження точності та ефективності розпізнавання мовних команд.

Результати даної роботи мають практичне значення для розробників систем автоматизації, які прагнуть підвищити ефективність сервісів за допомогою голосових технологій. Запропонований підхід може бути використаний у створенні інтелектуальних помічників, систем самообслуговування, а також у розробці доступних сервісів для користувачів з обмеженими можливостями.

РОЗДІЛ І. АНАЛІЗ ТЕХНОЛОГІЙ РОЗПІЗНАВАННЯ МОВЛЕННЯ ТА СФЕРИ ЇХ ЗАСТОСУВАННЯ

1.1. Історія та сучасний стан технологій розпізнавання мовлення

Розпізнавання мовлення – технологія, яка перетворює аудіосигнали голосу людини на текстовий формат або інтерпретує голосові команди, – пройшла значний шлях свого розвитку: від обмежених експериментальних систем до високоефективних комерційно доступних сервісів. Систематичний аналіз цього розвитку дозволяє виокремити ключові етапи, що формували сучасний стан технологій, а також визначити основні виклики, з якими стикається галузь сьогодні.

Ранній період був із 1950 по 1960-ті роки. Перші спроби автоматичного розпізнавання людської мови з'явилися вже у 1950-х роках. Наприклад, лабораторія Bell Laboratories у 1952 році створила систему «Audrey», що могла розпізнавати цифри від 0 до 9, вимовлені однією людиною. Ця система була високозалежною від голосу конкретного користувача і мала вкрай обмежений словниковий запас [1]. У 1962 році компанія IBM представила «Shoebox», яка розпізнавала близько 16 англійських слів (цифри та оператори) й виконувала прості обчислення [2, с. 5]. Ці ранні системи були обмежені за низкою параметрів: мала кількість слів, необхідність адаптації під конкретного користувача, обмежені й неконтрольовані акустичні умови.

Етап становлення та статистичних методів був із 1970 по 1980-ті роки. У 1970-х роках з розвитком обчислювальних засобів і підвищенням інтересу до розпізнавання мови розпочалися значні дослідження в рамках програми DARPA «Speech Understanding Research (SUR)». Зокрема, дослідницькі групи, такі як в Carnegie Mellon University, розробляли системи «Harpy», що могли працювати з великим словником (понад 1000 слів) [3]. Водночас у 1980-х роках стало масово застосовуватися статистичне моделювання. Зокрема, структури з використанням прихованих моделей Маркова (HMM – Hidden Markov Model) із Gaussian Mixture Models (GMM) як акустичним компонентом, стали основою багатьох систем [4].

Це дозволило системам переходити від шаблонного підходу до більш гнучкої імовірнісної моделі, що підвищило словниковий запас, адаптивність до різних голосів й умов.

Комерціалізація і широке впровадження – 1990-ті–2000-ті роки. У 1990-х роках з’явилися перші комерційні продукти розпізнавання мови для масового використання. Наприклад, система «Dragon Dictate» від Dragon була одним з перших програмних рішень для диктування тексту [5]. У телефонних мережах почали використовуватися системи автоматичного оброблення викликів з голосовими меню (Interactive Voice Response – IVR). З розвитком інтернету й мобільних технологій у 2000-х роках стало можливим здійснювати голосовий пошук у системах, наприклад, у Google [6]. Цей етап характеризувався переходом технологій з лабораторій у повсякденне використання, а також масовою доступністю обчислювальних і мережевих ресурсів.

Сучасний стан та новітні тенденції – 2010-ті–2020-ті роки й далі. Упродовж останнього десятиліття технологія розпізнавання мовлення зазнала радикальних змін завдяки використанню методів глибокого навчання (deep learning). Наприклад, перехід до архітектур із рекурентними нейронними мережами (RNN) та подальше впровадження трансформерів (Transformer) дозволили значно знизити помилки (Word Error Rate – WER) у завданнях великого словникового запасу [7]. Останні дослідження фіксують, що сучасні системи, зокрема енд-ту-енд (end-to-end) моделі, можуть працювати майже на рівні людського слухача у контрольованих умовах [8]. Ключових тенденцій сучасного розвитку є декілька. Наприклад, зростання словникового запасу та підхід до «розмовної» мови (тобто не лише диктовка, а природна мова). використання великих датасетів і хмарних обчислень («big data», «cloud computing»). Також важливими тенденціями є розвиток моделей з малою кількістю ресурсів (edge devices, мобільні пристрої) та фокус на стійкість до шуму, різних акцентів, мовам із малими ресурсами (low-resource languages) [1, 7, 3].

Наприклад, проєкт Babel Program (IARPA / DARPA) був спрямований на розробку технологій розпізнавання мовлення для мов з обмеженими ресурсами в умовах шуму [9]. Таким чином, сучасний стан технологій характеризується

високою точністю, інтернаціональністю, застосуванням в мобільних та хмарних сервісах, а також поступовим переходом до голосового управління в побутових застосунках (розумні помічники, автоматизація обслуговування).

Незважаючи на помітний прогрес, технології розпізнавання мовлення продовжують стикатися з низкою викликів: адаптація до акцентів, діалектів, шумового середовища, особливо у реальному часі; забезпечення рівноправного підтримування менш поширених мов та діалектів (low-resource languages); баланс між точністю моделі, вимогами до обчислювальних ресурсів і затримками (latency) у застосунках реального часу; безпека та приватність: голосові дані є персональними, і їхня обробка викликає питання захисту даних.

У контексті розробки системи голосового управління через чат-бот (зокрема Telegram-бота) аналіз історії й сучасного стану технологій розпізнавання мовлення є фундаментальним. Він дозволяє: з'ясувати, які технології є найбільш прогресивними і де варто робити вибір; оцінити, що вже реалізовано й які функціональні можливості відкриті для втілення; визначити, які аспекти (наприклад, акценти, шум) потребуватимуть уваги в практичній реалізації.

Підсумовуючи, можна зробити такі висновки – технологія розпізнавання мовлення пройшла довгий шлях – від лабораторних систем із обмеженим словником до сучасних рішень із високою точністю та широкою світовою підтримкою. Найбільш значущими етапами стали: впровадження статистичних моделей (НММ) у 1970-1980-х роках та перехід до глибинного навчання в 2010-ті роки. Сьогодні розпізнавання мовлення – важлива складова багатьох сервісів, і її застосування у голосових інтерфейсах чат-ботів є обґрунтованим і актуальним напрямом дослідження. Для успішної реалізації голосового інтерфейсу необхідно враховувати виклики: адаптацію до різноманітності мов, шумового середовища, ресурсів платформи та захист персональних даних.

1.2. Методи та алгоритми розпізнавання

У структурі систем розпізнавання мовлення (automatic speech recognition, ASR) виділяють кілька ключових етапів: попередня обробка сигналу (виділення

ознак), акустичне моделювання, мовне моделювання, та декодування/пошук гіпотез. [2, с. 87]. Нижче наведено огляд основних підходів, алгоритмів та сучасних методів, із вказівкою їх сильних і слабких сторін, а також тенденцій розвитку.

Першим кроком у системах ASR є перетворення аудіосигналу в набір характерних чисельновизначених ознак. Типово використовуються мел-частотні кепстральні коефіцієнти (MFCC – Mel-Frequency Cepstral Coefficients), а також їхні похідні (delta, delta-delta) для моделювання часової динаміки [10]. Інші техніки включають фільтрацію шуму, Voice Activity Detection (VAD) для відділення промови від тиші/фонового шуму [7]. Цей етап є критичним, оскільки якість виділених ознак значно впливає на продуктивність наступних моделей.

Класичні алгоритми: акустично-фонетичний, шаблонний, статистичний. Акустично-фонетичний підхід передбачає вручну виділені фонетичні одиниці (фонеми), їх класифікацію на основі акустичних характеристик і збирання слів із цих одиниць [3]. Шаблонний (template-based) підхід порівнює вхідну ознаку зі збереженим шаблоном (наприклад, динамічне вирівнювання – DTW) [14]. Статистичні моделі, найбільш розповсюджено у попередню еру: наприклад, модель на основі прихованих моделей Маркова (HMM – Hidden Markov Model) у поєднанні з Gaussian Mixture Models (GMM) як спостережуваних розподілів [10]. HMM дозволяє моделювати часову послідовність, а GMM – розподіли ознак у кожному стані. Ці моделі були основою великих систем ASR 1990-х та 2000-х [2]. Приклад: система HMM із MFCC ознаками, контекст-трипхеми, адаптація голосів тощо.

Зі зростанням обчислювальних ресурсів та доступності великих датасетів з'явилися гібридні моделі, які поєднують HMM із нейронними мережами (наприклад, DNN-HMM) [2]. У такій архітектурі HMM моделює послідовність станів, а DNN обчислює ймовірність спостережень для кожного стану. Це суттєво підвищило точність розпізнавання. Наприклад, DNN-HMM системи дозволяли значно знизити WER (Word Error Rate) у задачах великого словника [1]. Інші підходи: Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN, LSTM) для обробки ознак із часовою залежністю [12].

Найсучаснішим напрямом є моделі «end-to-end», які намагаються замінити традиційну багатокомпонентну архітектуру однією нейронною мережею, що бере на вхід ознаки (або навіть сирий аудіо) і видає текстовий результат [11]. Архітектури: CTC (Connectionist Temporal Classification), Attention-based Seq2Seq, трансформери (Transformers) та Conformers [5]. Переваги: спрощення архітектури, менше ручного налаштування, потенційно краще узагальнення. Недоліки: висока потреба в даних, обчислювальних ресурсах, чутливість до розподілу даних. Наприклад, дослідження показують, що трансформер-акустичні моделі можуть покращити результати на датасеті LibriSpeech на 19-26 % відносно кращих опублікованих гібридних моделей [15].

Окрім акустичного моделювання, ключовим компонентом є мовна модель (LM – Language Model), яка оцінює ймовірність послідовності слів, та лексичний компонент, який пов'язує фонеми чи триграфи зі словами [2]. Декодування здійснюється алгоритмом пошуку, який намагається знайти найімовірнішу послідовність слів, враховуючи акустичні, лексичні і мовні моделі [2]. У новітніх підходах мовні моделі інтегруються безпосередньо в нейронну мережу (наприклад, трансформери з attention), що дає кращу синергію між акустичним і мовним компонентом.

Сучасні дослідження звертають увагу на: адаптацію під акценти, діалекти, шумове середовище; моделі для мов із малою кількістю ресурсів (low-resource languages) [1]; використання трансферного навчання (Transfer Learning), федеративного навчання (Federated Learning) та підходів зі зменшеною затримкою (low latency) для edge-пристроїв [5]; оптимізація виділення ознак, використання представлення (representation learning) замість ручного проектування ознак [13]; інтеграція мультимодальних методів (наприклад, аудіо-відео) для покращення стійкості до шуму (audio-visual speech recognition).

Таблиця 1.1 – Порівняння методів

Метод	Переваги	Недоліки
Акустично-фонетичний	пояснюваність, проста реалізація	мала точність, обмеження словника
Шаблонний (DTW)	добре для конкретних задач з невеликим словником	не масштабується, високі затрати часу

Статистичні HMM/GMM	ширше застосування, добрі базові результати	залежність від ручного налаштування, обмежені ознаки
Гібридні DNN-HMM	суттєво краща точність	більше ресурсів, складність налаштування
End-to-End нейронні	спрощена архітектура, потенціал високої точності	велика кількість даних, висока обчислювальна вартість

Методи розпізнавання мовлення еволюціонували від ручно структурованих підходів (фонетичного, шаблонного) до статистично-нейронних і енд-ту-енд моделей. Кожен з підходів має свої переваги і обмеження, вибір залежить від задачі, ресурсів, словникового запасу і умов роботи. Для сучасних систем, зокрема для інтеграції з чат-ботом із голосовим інтерфейсом, доцільно використовувати гібридні або енд-ту-енд моделі з адаптацією до умов застосування (акцент, шум, словник). Врахування трендів (наприклад, моделі для low-resource мов, трансферне навчання) дозволяє підвищити гнучкість і ефективність систем розпізнавання мовлення.

1.3. Хмарні та локальні рішення

У сучасній практиці впровадження систем розпізнавання мовлення значну увагу приділяють вибору моделі розгортання: хмарна (cloud-based) чи локальна (on-premise або edge/on-device). Це важливий аспект при проектуванні системи, зокрема для інтеграції голосового інтерфейсу в бот-систему, оскільки кожний підхід має свої переваги і обмеження, що впливають на архітектуру, безпеку, використання ресурсів та експлуатацію.

Хмарні рішення розпізнавання мовлення передбачають передачу аудіоданих на віддалені сервери провайдера (наприклад, Google Cloud Speech-to-Text, Microsoft Azure Speech, Amazon Web Services Transcribe) для обробки та повернення результату у вигляді тексту або команди. Переваги хмарних підходів: масштабованість: ресурси легко збільшуються або зменшуються відповідно до навантаження (прайсинг «pay-as-you-go») [4]; доступ до передових моделей, великих навчальних наборів, мультимовних варіантів, регулярних оновлень: хмарні сервіси часто швидше впроваджують нові версії моделей [5];

мінімальна початкова інфраструктура: не потрібно купувати/підтримувати власні сервери, спеціалізоване обладнання [3]. Обмежень та ризиків є декілька. Одним із них є залежність від інтернет-зв'язку: у випадку обмеженого або нестабільного підключення може бути затримка або неможливість обробки. Також ризиком є питання безпеки і приватності: аудіо-дані передаються до сторонніх серверів, що може не відповідати вимогам регуляцій (наприклад, GDPR, законодавство про персональні дані) [2]. Варто враховувати ще операційні витрати: хоча початкові витрати низькі, при великій кількості запитів/аудіо-трафіку підписка може стати значною [1].

Локальні рішення передбачають розміщення обробки на власному обладнанні (сервери чи пристрої користувача) або на пристроях «на краю» (edge devices) без передачі аудіо до сторонніх хмарних провайдерів. Переваги локальної обробки. Повний контроль над даними: аудіо обробляється всередині організації, що підвищує рівень конфіденційності і може задовольняти жорсткі вимоги безпеки. Мінімальна затримка (latency): оскільки аудіо не передається до зовнішніх серверів, час обробки може бути меншим, що важливо для реального часу [3]. Можливість роботи без підключення до інтернету: критично для офлайн-сценаріїв або середовищ з поганим зв'язком. Обмежень є декілька, а саме: високі початкові інвестиції: установка, налаштування, навчання моделей, придбання обладнання – великі капітальні витрати [3]; менша масштабованість: при збільшенні навантаження потрібно закуповувати нове обладнання чи розширювати інфраструктуру [6]; обмежений доступ до нових моделей/оновлень: можливий відставання від хмарних сервісів у плані інновацій [2].

Таблиця 1.2 – Порівняльна таблиця хмарних і локальних рішень

Критерій	Хмарне рішення	Локальне рішення
Початкові витрати	Низькі (підписка)	Високі (обладнання, ліцензії)
Масштабованість	Висока, гнучка	Обмежена, залежить від обладнання
Затримка / latency	Може бути вища (залежно від мережі)	Зазвичай нижча
Контроль над даними	Обмежений – передача на сторонні сервери	Повний – дані залишаються в організації
Підтримка інновацій	Швидка – регулярні оновлення	Потребує власного ресурсу на оновлення

Робота без інтернету	Залежить від підключення	Можлива повністю офлайн
Підходить для	Динамічних навантажень, обмеженого бюджету, швидкого запуску	Стабільних навантажень, високих вимог до безпеки/конфіденційності

Кажучи про сучасні системи, варто згадати про гібридні моделі, які комбінують обробку частини аудіо або команд локально, а іншу частину – у хмарі. Наприклад, чутливі команди можуть оброблятися на пристрої або локально, а менш критичні – через хмарний сервіс [15]. Це дозволяє поєднати переваги обох підходів: контроль + масштабованість.

При оцінці для системи, яка реалізує голосові команди через Telegram-бота, слід врахувати такі фактори: чи потребується обробка великих обсягів запитів з багатьох користувачів (масштаб); чи є вимоги до офлайн-роботи або роботи у умовах нестабільного інтернету; чи є суворі вимоги до захисту персональних даних (особливо якщо йдеться про голосові команди з персональною інформацією); бюджет на інфраструктуру та експлуатацію; необхідність швидкого запуску та регулярних оновлень

Якщо система має багато користувачів, потребує глобальної доступності й швидкого масштабування – хмарне рішення буде доцільним. Якщо ж важлива конфіденційність, офлайн-робота або дуже низька затримка – перевагу має локальне рішення. В багатьох випадках доцільним є гібридний підхід, щоб задовольнити протилежні вимоги.

Обидва підходи – хмарний і локальний – мають свої переваги і недоліки, і вибір залежить від конкретного сценарію використання. Для системи голосового інтерфейсу у Telegram-боті, яка може працювати з численними користувачами і динамічними запитами, хмарне рішення є більш універсальним стартовим варіантом. Однак, якщо є суворі вимоги до захисту даних або офлайн-роботи, локальне або гібридне рішення стає пріоритетним. У проекті доцільно передбачити можливість гібридної архітектури, щоб забезпечити баланс між масштабованістю, швидкістю, безпекою та витратами.

1.4. Концепція системи у сфері автоматизації замовлень

У сучасних сервісах автоматизації замовлень, особливо з використанням голосових технологій, виділяються кілька підходів до побудови системи. Такий вибір концепції впливає на архітектуру, взаємодію з користувачем, технічні вимоги, витрати, а також на масштабованість і надійність системи. У контексті реалізації голосового інтерфейсу у бот-системі (наприклад, у Telegram-боті) доцільно розглянути такі концептуальні варіанти:

Концепція А: «Текстово-голосове» керування. Опис: Користувач взаємодіє з ботом через чат-інтерфейс, у якому може надсилати голосове повідомлення, яке система розпізнає і трансформує у текстову команду. Далі логіка системи обробляє текст і формує замовлення. Після підтвердження – користувачу надіслано відповідь текстом або голосом. Перевагами є спрощена імплементація: оскільки основна логіка обробки працює з текстом, а голосова частина перетворює аудіо в текст. Також менші технічні вимоги до обробки у режимі реального часу голосу, ніж у повноцінних голосових системах та користувач має можливість переходу до текстового вводу, якщо голосові команди не спрацювали.

Недоліки: менша природність взаємодії – користувач спочатку записує або надсилає голос, але подальша частина може бути текстовою; може бути затримка через проміжне представлення (аудіо – текст – обробка); залежність від якості розпізнавання: помилки перетворення можуть вплинути на логіку замовлення.

Концепція В: «Voice-First» (голосове управління в реальному часі). Опис: Користувач ініціює голосову команду (наприклад: «Замовити піцу великого розміру, додати напій, доставка завтра»). Система миттєво обробляє аудіо, розпізнає текст, аналізує наміри (intent), формує замовлення, підтверджує і виконує його – усе у мінімальній кількості кроків та без або з мінімальним ввідом тексту. Перевагами є максимальна зручність і природність для користувача: взаємодія «говориш – отримуєш результат»; може значно скоротити час на замовлення, підвищити рівень задоволеності користувачів і конкурентоспроможність сервісу. Наприклад, сучасні голосові системи в ресторанному бізнесі демонструють значне скорочення часу обробки замовлень

[27]; підходить для сценаріїв «hands-free», коли користувач не може вводити текст (під час руху, керування тощо).

Недоліками є високі технічні вимоги: необхідна стабільна система розпізнавання голосу, адаптація до шуму/акцентів, висока точність аналізу намірів (NLU) і інтеграція з сервісами замовлень [14]; великий ризик помилок: неправильне розпізнавання або інтерпретація може призвести до некоректного замовлення, що у випадку фінансових транзакцій є критичним; більш висока вартість реалізації, довший час розробки; необхідність забезпечити fallback-механізм на випадок помилки (наприклад, текстовий ввід або перевірка користувачем).

Концепція С: «Гібридне» рішення (текст + голос + контекстна логіка).
Опис: Система підтримує обидві модальності – голос та текст. Користувач може вибрати: надати голосову команду або текстове повідомлення. Система аналізує голос, перетворює на текст, зчитує контекст (історію замовлень, профіль користувача, типові шаблони), і пропонує опції або підказки. Замовлення може бути підтверджено голосом або текстом. У фоні – контекстна логіка: рекомендації, збереження переваг користувача, адаптація під його звички. Переваг є декілька, а саме компроміс між зручністю голосу та надійністю тексту. Підвищена гнучкість для різних груп користувачів: хтось віддає перевагу голосу, хтось – тексту. Можливість поступового розвитку системи: почати з менш складного голосового вводу і згодом адаптувати повноцінний голосовий режим. Контекст-орієнтованість дозволяє підвищити точність замовлень і персоналізацію.

Недоліки: реалізація двох режимів (голос і текст) і контекстної логіки ускладнює розвиток системи; вища вартість, ніж простий текстовий режим; вимагає продуманого UX-дизайну, щоб користувачі не плутались, які команди доступні голосом, а які – текстом.

Таблиця 1.3 – Порівняння концепцій

Критерій	Концепція А (Текст+-Голос)	Концепція В (Voice-First)	Концепція С (Гібрид)
Зручність для користувача	Середня	Висока	Висока

Реалізаційна складність	Низька–середня	Висока	Середня–висока
Надійність / помилки	Вище ризик помилок через трансформацію	Помилки можуть критичні	Найкраща (при гарному налаштуванні)
Масштабованість	Добра	Потенційно висока	Висока
Вартість запуску	Низька	Висока	Середня
Адаптація до контексту	Мінімальна	Висока	Висока
Підходить для	Обмежені голосом сценарії	Повна голосова взаємодія	Комбіновані сценарії

У межах системи автоматизації замовлень для Telegram-бота з голосовими командами доцільно обрати концепцію С (Гібрид). Аргументами є те, що вона дає можливість максимально залучити голосовий інтерфейс (що підвищує комфорт користувача), але зберігає текстовий варіант вводу як резерв/доповнення, що підвищує надійність. Також враховує, що не всі користувачі можуть або хочуть користуватись лише голосом (наприклад, в шумному середовищі, або з особливими умовами). Дозволяє поступовий розвиток: можна стартувати з базового голосового вводу з текстовим резервом і згодом вдосконалювати голосовий режим, додаючи контекстну логіку та адаптацію під акценти/мови/штучні навантаження. Забезпечує гнучкість, адаптацію до різних сценаріїв (домашнє замовлення, в дорозі, з мобільного пристрою) і підтримує часткову автоматизацію без ризику повного голосового провалу. У комерційному та освітньому контексті це стратегічно виграшний варіант, оскільки дозволяє збалансувати витрати, технічні вимоги і користувацький досвід.

Отже, у подальшому проєкті приймається концепція С як базова архітектурна схема системи. Подальші розділи (проектування, реалізація) будуть деталізувати модулі: голосовий інтерфейс, інтеграція з Telegram API, логіка обробки замовлень, контекстна обробка, забезпечення надійності, безпеки та обробки даних.

Було проаналізовано три можливі концепції систем автоматизації замовлень із голосовим інтерфейсом: текстово-голосове, повно голосове, гібридне. Проведено порівняння за ключовими критеріями: зручність, складність

реалізації, надійність, масштабованість, вартість. Обґрунтовано вибір гібридної концепції (С) як оптимальної для системи Telegram-бота з голосовими командами в умовах, коли потрібна ефективність, надійність і гнучкість. Визначено, що подальший розвиток проєкту буде базуватись на цій концепції: детальна архітектура, модулі, інтеграції та реалізація.

1.5. Постановка задачі

Метою даної роботи є підвищення ефективності та зручності процесу оформлення замовлень користувачами шляхом впровадження системи розпізнавання голосових команд у Telegram-боті. Досягнення цієї мети забезпечується розробленням та інтеграцією інтелектуального голосового інтерфейсу, який дає змогу автоматизувати процес взаємодії користувача із сервісом і скоротити час на виконання стандартних операцій. Для реалізації поставленої мети необхідно виконати такі завдання:

1. Провести аналіз сучасних технологій розпізнавання мовлення, включно з алгоритмами, методами та моделями, для визначення оптимальних підходів у застосуванні до голосового інтерфейсу Telegram-бота.
2. Дослідити моделі впровадження систем розпізнавання мовлення (хмарні, локальні та гібридні), оцінити їхні обмеження, продуктивність і доцільність використання у задачі автоматизації замовлень.
3. Спроекувати архітектуру системи голосової взаємодії, включаючи вибір платформи ASR (Automatic Speech Recognition), інтеграцію з Telegram API, моделі обробки, збереження й захисту даних.
4. Реалізувати Telegram-бот з підтримкою голосових команд, інтегрувати API розпізнавання мовлення, розробити алгоритми перевірки, фільтрації та інтерпретації вхідних голосових повідомлень, а також провести тестування й оцінити точність та ефективність роботи системи.
5. Провести економічне обґрунтування проєкту, оцінити витрати на розроблення та впровадження, проаналізувати економічну доцільність і конкурентні переваги системи.

6. Врахувати вимоги охорони праці та охорони навколишнього середовища при розробленні, впровадженні й експлуатації голосового інтерфейсу.

Під час проектування та реалізації системи необхідно врахувати такі ключові обмеження. Найперше це обмеження словникового запасу та доменної специфіки, тобто система орієнтована на автоматизацію замовлень, тому використовує фіксований набір команд, меню та опцій. Розпізнавання вільної мовної взаємодії загального призначення не передбачено. Наступне це обмеження якості аудіосигналу, а саме передбачається коректна робота у середовищі помірного шуму. У разі високого шумового рівня можливі суттєві погіршення точності. Обмеження ресурсів і масштабування, тобто продуктивність залежить від обраної архітектури (локальна, хмарна або гібридна), що впливає на затримку, витрати та швидкість обробки запитів. Обмеження безпеки та конфіденційності – необхідне дотримання вимог законодавства щодо обробки персональних даних, особливо у разі передачі аудіо в хмарні сервіси. Обмеження Telegram API, тобто система повинна відповідати вимогам та обмеженням платформи (формати аудіо, тривалість повідомлень, швидкість відповіді). Обмеження користувацького досвіду – не всі користувачі можуть або хочуть використовувати голосові команди; тому система має забезпечувати альтернативу у вигляді текстового введення чи підтвердження.

Результатом виконання роботи стане функціонуючий Telegram-бот із підтримкою голосових команд, інтегрований із сервісом розпізнавання мовлення та здатний: приймати замовлення; скасовувати або змінювати їх; надавати користувачеві голосові чи текстові підтвердження.

Очікується, що розроблена система: досягне цільового рівня точності розпізнавання (наприклад, $WER < X \%$); забезпечить скорочення часу оформлення замовлення порівняно з текстовим введенням; матиме економічно обґрунтовану доцільність впровадження. Таким чином, визначено мету, завдання, рамки та ключові обмеження, що формують основу для подальших етапів аналізу, проектування та реалізації системи.

Висновки до розділу 1

У першому розділі дипломної роботи проведено комплексний аналіз розвитку, методів, архітектурних рішень і концептуальних засад побудови систем розпізнавання мовлення, орієнтованих на автоматизацію замовлень у середовищі Telegram.

Було досліджено історію становлення технологій розпізнавання мовлення – від перших спроб створення голосових інтерфейсів у 1950–1970-х роках до сучасних інтелектуальних систем, що базуються на штучних нейронних мережах. Визначено, що сучасні системи розпізнавання демонструють високу точність завдяки використанню моделей глибокого навчання (Deep Learning) та великих обсягів навчальних даних. Такі технології сьогодні є невід’ємною частиною інтелектуальних голосових помічників, сервісів автоматичного перекладу та систем підтримки користувачів.

Детально розглянуто методи та алгоритми розпізнавання мовлення. Серед основних виділено алгоритми на основі прихованих марковських моделей (HMM), нейронних мереж (RNN, CNN, Transformer) та гібридні підходи, що комбінують статистичні та глибинні методи. Зроблено висновок, що найефективнішими в сучасних умовах є моделі типу end-to-end, такі як DeepSpeech, Wav2Vec 2.0, Whisper від OpenAI, які здатні безпосередньо перетворювати аудіосигнал у текст без додаткових мовних моделей.

Проведено аналіз хмарних і локальних рішень розпізнавання мовлення. Визначено, що хмарні рішення (Google Speech-to-Text, Amazon Transcribe, Microsoft Azure Speech, OpenAI Whisper API) характеризуються високою точністю, масштабованістю та простотою інтеграції, проте потребують підключення до мережі та створюють ризики для конфіденційності даних. Локальні рішення (Mozilla DeepSpeech, Vosk, Kaldi) забезпечують більший контроль над даними, проте вимагають потужних обчислювальних ресурсів і складнішого налаштування. У результаті аналізу встановлено, що гібридна модель, яка поєднує локальну попередню обробку та хмарне розпізнавання, є найбільш ефективним підходом для створення голосових Telegram-ботів.

Розроблено концепцію системи у сфері автоматизації замовлень, проведено порівняльний аналіз різних архітектур (централізована, децентралізована, гібридна) та визначено оптимальну концепцію для реалізації проєкту. Було встановлено, що гібридна архітектура з розподіленням навантаження між клієнтською частиною Telegram-бота, сервером обробки запитів і хмарним сервісом розпізнавання мовлення забезпечує оптимальний баланс між продуктивністю, безпекою та масштабованістю.

Сформульовано постановку задачі, визначено мету, завдання та обмеження системи. Метою визначено створення Telegram-бота із голосовим інтерфейсом для автоматизації замовлень. Серед завдань – вибір алгоритмів розпізнавання, проєктування архітектури, розробка та тестування системи, а також забезпечення безпеки персональних даних. Обмеження включають точність розпізнавання, якість аудіосигналу, вимоги до ресурсів та обмеження платформи Telegram.

Узагальнюючи результати проведеного аналізу, можна зробити такі висновки:

- Технології розпізнавання мовлення пройшли довгий шлях розвитку і досягли високого рівня точності завдяки глибинному навчанню та штучному інтелекту.
- Найперспективнішими є моделі типу end-to-end, що мінімізують людське втручання та підвищують точність.
- Хмарні рішення забезпечують високу якість, але вимагають уваги до конфіденційності; локальні – надають контроль, проте вимагають ресурсів; гібридний підхід є найоптимальнішим для практичної реалізації.
- Розроблена концепція системи Telegram-бота з голосовим інтерфейсом може бути ефективно використана для автоматизації замовлень у різних сферах – від доставки їжі до технічної підтримки.
- Сформульована постановка задачі чітко визначає цілі, завдання та обмеження, що створює міцне підґрунтя для подальшого проєктування та реалізації системи у наступних розділах роботи.

Таким чином, розділ 1 забезпечує теоретичну й аналітичну основу для побудови ефективної системи розпізнавання мовних команд у Telegram-боті та визначає ключові напрями подальшої розробки.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ГОЛОСОВИХ ЗАМОВЛЕНЬ У TELEGRAM

2.1. Архітектура системи з урахуванням голосового інтерфейсу

Архітектура системи розпізнавання голосових замовлень у Telegram-боті побудована як модульна багаторівнева структура, у якій кожен компонент виконує окремі функції, забезпечує обробку голосових даних та підтримує інтерактивну взаємодію з користувачем. Система включає клієнтський рівень, сервіси розпізнавання мовлення, модулі аналізу тексту, бізнес-логіку та інфраструктурні компоненти, що відповідають за зберігання даних, масштабування та безпеку.

Клієнтський модуль (Telegram-бот) – інтерфейс взаємодії з користувачем: прийом голосових та текстових повідомлень, відправка відповідей. Модуль розпізнавання мовлення (ASR – Automatic Speech Recognition) – перетворює аудіо-сигнал у текст, виконує первинну обробку (виділення ознак, фільтрацію шуму), можливо, виконується хмарно або локально. Модуль аналізу наміру (NLU – Natural Language Understanding / Intent Identification) – аналіз тексту, визначення наміру користувача (наприклад, «замовити», «скасувати», «додати опцію») та витяг параметрів (наприклад, товар, кількість, спосіб доставки).

Бізнес-логіка замовлення – обробка команди, формування замовлення, взаємодія з базою даних, зберігання інформації, підтвердження. Модуль безпеки і зберігання даних – забезпечує автентифікацію/авторизацію, захист персональних даних, архівування, логування подій. Інфраструктурна платформа – сервери або хмарна інфраструктура, механізми масштабування, балансування навантаження, сховище даних, резервування [16].

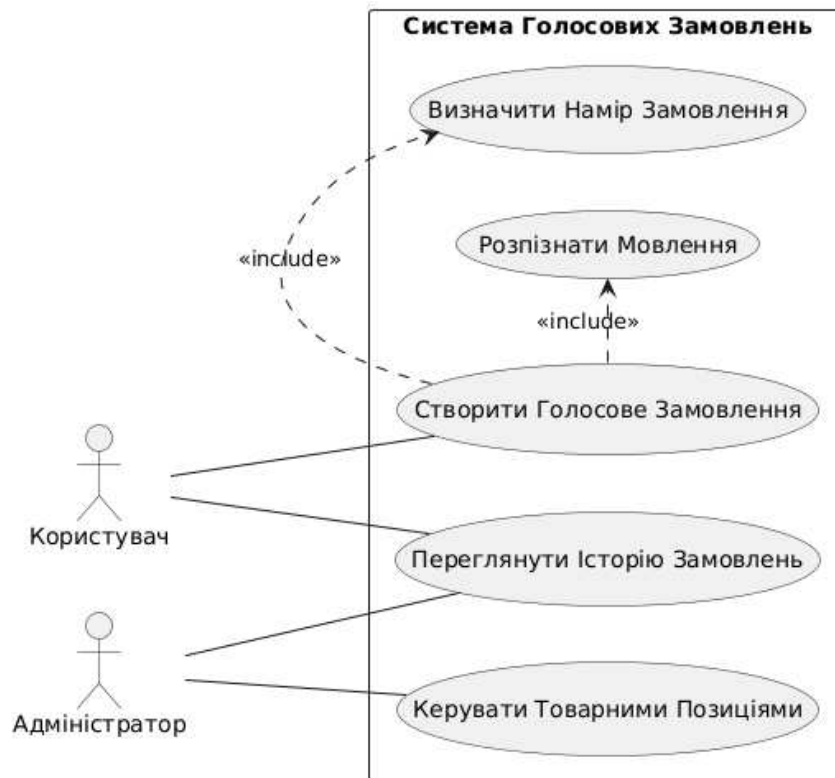


Рисунок 2.1 – Use-Case діаграма системи голосових замовлень через Telegram-бот

Взаємодія між компонентами відбувається у такій послідовності: користувач надсилає голосове повідомлення через Telegram-інтерфейс – бот приймає та передає аудіофайл у модуль ASR – отриманий текст передається в NLU – бізнес-логіка формує замовлення і взаємодіє з БД та сервісами – відповідь надсилається користувачу (можливо, у вигляді голосового або текстового повідомлення). Архітектура враховує можливість масштабування (через хмарні компоненти) і забезпечення конфіденційності через локальну обробку чи гібридну архітектуру. Наприклад, зауважено, що голосові чат-боти часто будують із serverless або хмарною архітектурою, що дозволяє масштабувати обчислювальні моделі швидко [17]. Ключовим є те, що архітектура має бути готова до неоднорідних вхідних умов: шумове середовище, акценти, мобільний зв'язок – тому частина обробки (наприклад, фільтрація шуму) може виконуватись локально або на edge-платформі, перш ніж передавати аудіо в хмару.



Рисунок 2.2 – Концептуальна модель системи розпізнавання голосових замовлень

Для голосового інтерфейсу виділяються декілька підсистем. Прийом аудіо: Telegram підтримує надсилання голосових повідомлень – бот має вміти отримувати ці файли (наприклад, формат OGG/OPUS) і підготувати їх до обробки [18]. Передобробка: видалення шуму, Voice Activity Detection (VAD), нормалізація сигналу. ASR-модуль: або хмарне рішення (наприклад, Google Speech-to-Text, Azure) або локальне рішення (наприклад, Vosk, Kaldi), або гібрид. Цей модуль повертає текстову транскрипцію. Обробка тексту та наміру (NLU/Intent): текст проходить через механізми класифікації наміру та витягання параметрів замовлення. Fallback / резервний варіант: у випадку коли ASR дає невисоку точність або користувач знаходиться в шумі – можливий перехід на текстове введення або підтвердження через клавіатуру. Відповідь користувачу: текстова або голосова відповідь, підтвердження замовлення чи питання про уточнення.

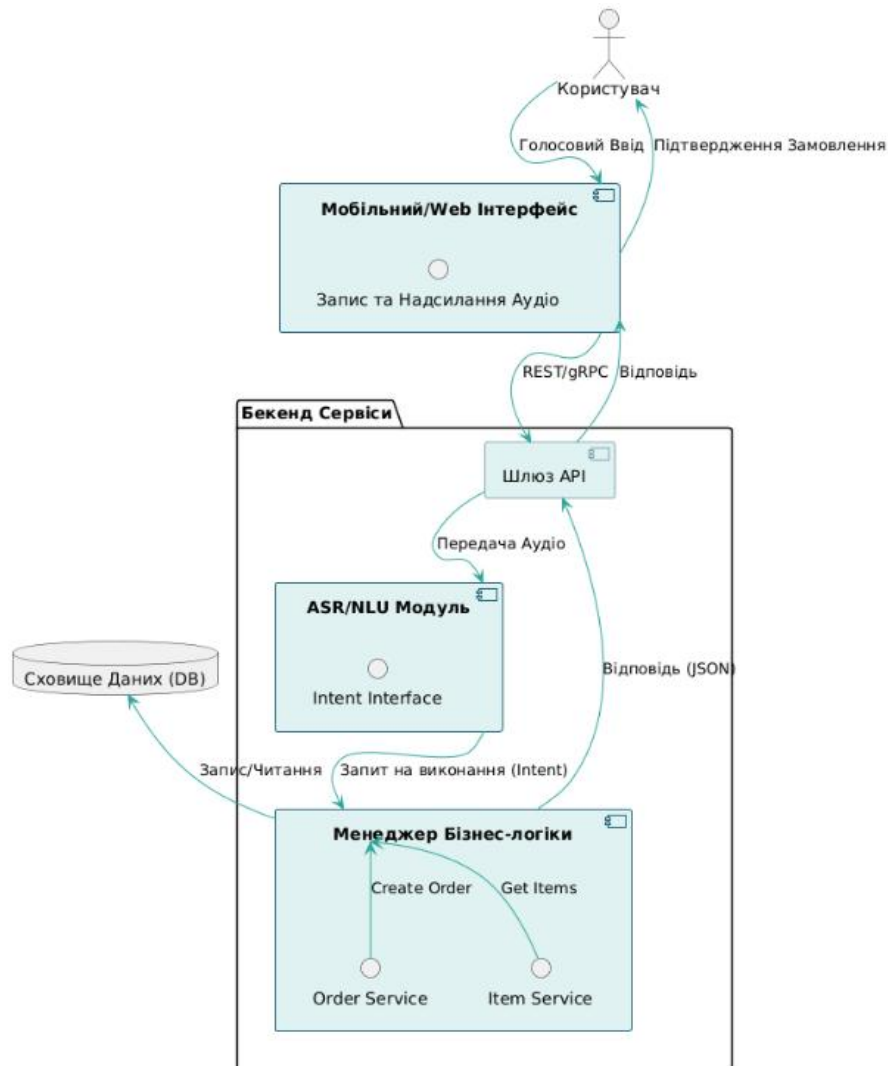


Рисунок 2.3 – Логічна архітектура системи

Архітектурні рішення щодо масштабованості, безпеки та конфіденційності. Масштабованість: хмарні рішення дозволяють масштабувати кількість одночасних запитів залежно від навантаження. Це важливо, коли бот обслуговує багато користувачів [19]. Безпека та захист даних: голосові дані можуть бути персоналізованими – система має забезпечити шифрування передачі аудіо, обмеження доступу, дотримання нормативів про захист персональних даних (наприклад, ЄС/GDPR або національні норми). Використання локальної обробки (on-device або on-premises) підвищує конфіденційність. Гібридна архітектура: комбінування локальної обробки (наприклад, видалення шуму, попередня транскрипція) і хмарної (головна транскрипція, NLU) – оптимальний варіант, коли потрібно баланс між продуктивністю, якістю та безпекою. Моніторинг, логування та аудит: критично

впровадити механізми логування звернень, помилок розпізнавання, часу обробки, щоб мати можливість аудиту та покращення системи [20].

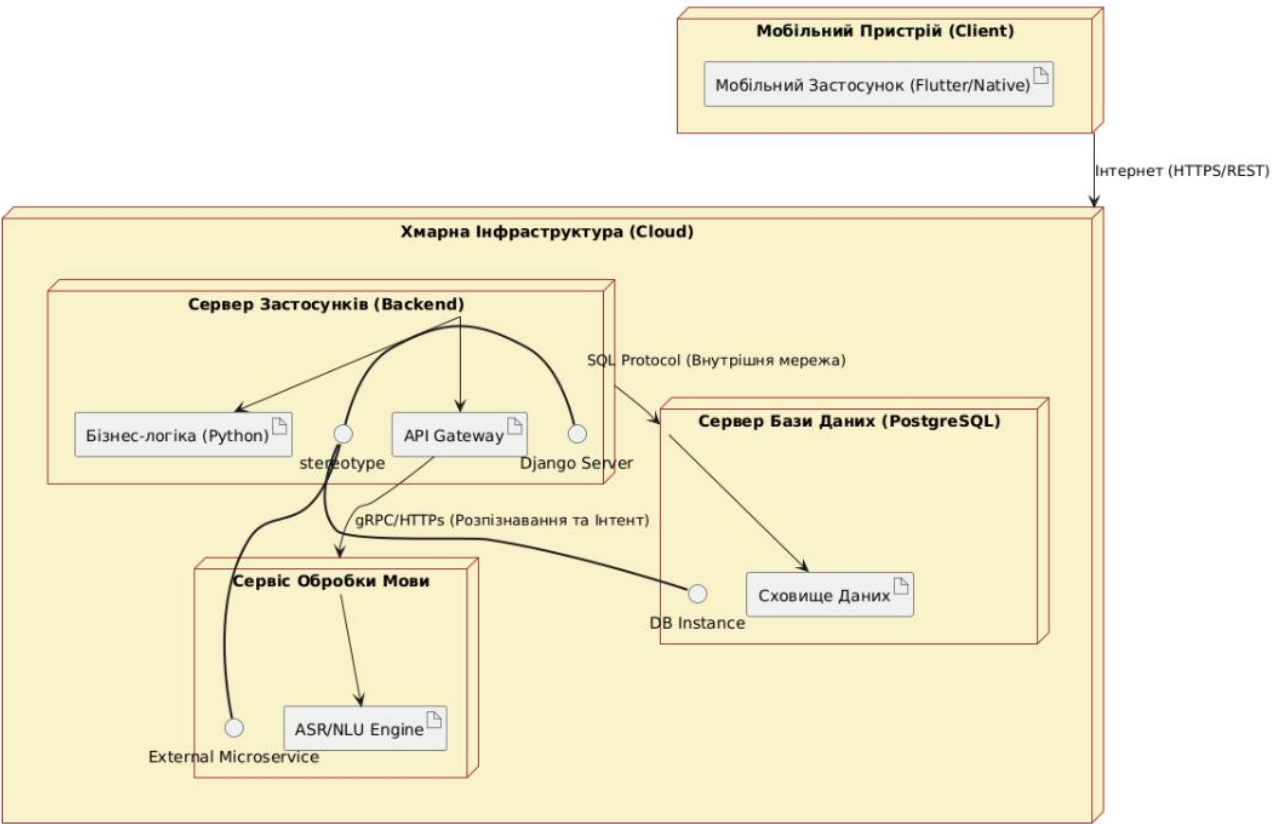


Рисунок 2.4 – Фізична архітектура системи

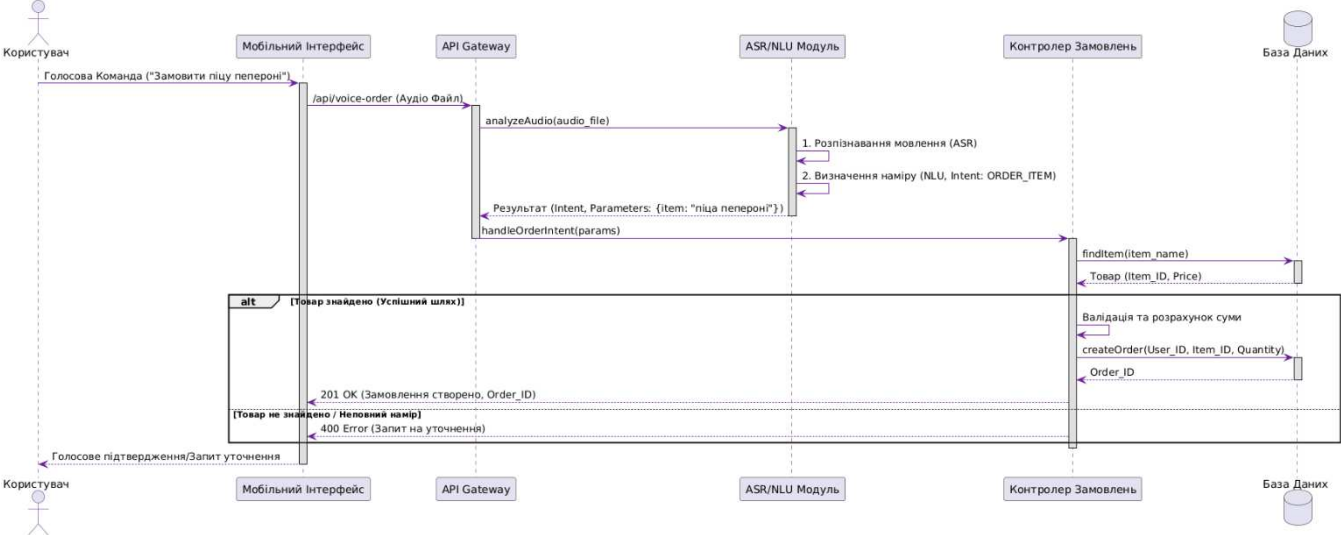


Рисунок 2.5 – Діаграма послідовностей для сценарію «Замовлення через голосову команду»

Архітектура системи з урахуванням голосового інтерфейсу описує інтеграцію голосових технологій із платформою Telegram-бота, включає

компоненти прийому голосу, його транскрипції, аналізу наміру, обробки замовлення, а також механізми забезпечення масштабованості, безпеки та конфіденційності. Представлена архітектура вважає гібридний підхід (локальна + хмарна обробка) оптимальним у контексті завдання автоматизації замовлень із голосовими командами.

2.2. Вибір платформи розпізнавання мовлення

При проєктуванні системи голосових замовлень у бот-середовищі Telegram критично правильно обрати платформу для розпізнавання мови (ASR). Вибір залежить від ряду факторів: точність, затримка (латентність), вартість, можливість кастомізації, підтримка мов, безпека даних, режим розгортання (хмара чи локально). У цьому підпункті аналізуються декілька ключових платформ ASR – наприклад, OpenAI Whisper, Deepgram Nova-3, Google Cloud Speech-to-Text – з оглядом їхніх переваг і обмежень [21].

При виборі ASR-платформи для проєкту слід враховувати такі аспекти. Одним із них є точність розпізнавання (ментритими метрика WER – Word Error Rate) у сценаріях з голосовими командами, часто з шумом чи акцентом. Також затримка (латентність) – наскільки швидко платформа надає транскрипцію, що важливо для голосового інтерфейсу в реальному часі [22]. Вартість – ціна обробки аудіо за хвилину чи за тисячу хвилин. Можливість кастомізації – чи можна адаптувати моделі під домен (наприклад, спеціальні назви товарів, замовлень) [23]. Режим розгортання – чи підтримується локальне (on-premises, edge) чи лише хмарне. Безпека та конфіденційність – наскільки дані користувача (голос) захищені, чи передаються на зовнішні сервери. Підтримка мов та акцентів – чи підтримується українська, чи добре працює з акцентами/діалектами.

OpenAI Whisper – відкритий ASR-модель (open-source), що демонструє високу гнучкість і підтримку локального розгортання (self-host). Проте має значні вимоги до обчислювальних ресурсів [16]. Deepgram Nova-3 – комерційна платформа, яка позиціонується як високопродуктивне рішення із низькою

латентністю і можливістю кастомізації [17]. Google Cloud Speech-to-Text – хмарний сервіс від великого провайдера, із широкою мовною підтримкою, але може бути дорожчим та менш гнучким у локальному розгортанні.

Таблиця 2.1 – Порівняння платформ

Платформа	Точність (приблизно)	Латентність/ швидкість	Вартість (орієнтовно)	Режим розгортання	Переваги	Обмеження
Whisper (OpenAI)	~10 % WER у деяких сценаріях	Локально: залежить від апаратури	Self-host / безплатна версія	Локально або хмарно [40]	Контроль над даними, нічого не передається провайдеру	Потребує потужного складання, більше налаштування
Deepgram Nova-3	~5-7 % WER, латентність < 300 мс	Дуже швидка, оптимізована під стрімінг	~\$4.30 за 1000 хв.	Хмара / on-premise	Висока швидкість, кастомні моделі	Потрібна підписка, може бути дорожчою при малих обсягах
Google Cloud Speech-to-Text	~9-12 % WER, але дорожча	Латентність середня	~\$16 за 1000 хв. в одному сценарії	Хмара	Багатомовність, стабільність	Менша гнучкість локально, може бути дорожчою

На підставі наведеного аналізу, оптимальним вирішенням для системи голосових замовлень через Telegram-бот є гібридний підхід, який комбінує Whisper (локально чи на сервері) з Deepgram: Whisper для сценаріїв, де важливий контроль над даними та конфіденційність; Deepgram для випадків з високим навантаженням, коли важлива мінімальна затримка і висока швидкість. Google Cloud може бути резервним варіантом або для мультимовної підтримки, якщо обсяги користувачів зростуть значно, але не є основним вибором через вищу вартість і меншу гнучкість локального розгортання. Такий підхід дозволяє максимально балансувати між точністю, швидкістю, витратами та безпекою.

2.3. Інтеграція з Telegram API

Інтеграція системи розпізнавання голосових замовлень із Telegram API є ключовим елементом функціонування голосового інтерфейсу, оскільки саме цей інтерфейс забезпечує прийом голосових повідомлень, їх попередню обробку,

передавання до сервісів розпізнавання мовлення та повернення результатів користувачеві. Telegram пропонує два основних механізми взаємодії з ботами: Bot API, що працює за принципами HTTP-запитів, та Telegram MTPROTO, який використовується для клієнтських застосунків та не застосовується у бот-архітектурі [17]. Telegram Bot API підтримує два режими отримання повідомлень. Перший із них це Long Polling – бот регулярно надсилає запит до серверів Telegram та отримує оновлення у вигляді масиву повідомлень. Цей режим технологічно простий та не потребує складної інфраструктури [45]. Другий це Webhook-метод – сервер бота отримує оновлення автоматично через HTTPS-запит від Telegram. Цей режим забезпечує меншу затримку та є оптимальним для систем, де важлива швидкість реакції, наприклад при роботі з голосовими командами [23].

У межах проєкту доцільно використовувати Webhook, оскільки голосові інтерфейси часто потребують зменшення часу між надсиланням голосового повідомлення та відповіддю бота. Це покращує зручність користувача та підвищує загальну ефективність взаємодії.

Типовий алгоритм інтеграції Telegram-бота із системою ASR складається з таких етапів. Перший етап – отримання голосового повідомлення – користувач надсилає voice-повідомлення у форматі OGG (кодек Opus). Telegram Bot API передає це повідомлення серверу бота у вигляді об'єкта voice або audio [20]. Другий етап – завантаження медіафайлу – за допомогою getFile Telegram повертає шлях до медіафайлу, після чого сервер завантажує аудіо через HTTPS-посилання, яке Telegram генерує на своїх CDN-серверів [48]. Третій етап – перекодування аудіо – більшість систем ASR (Whisper, Deepgram, Google STT) приймають WAV або FLAC, тому файл попередньо перекодовують. Як правило, це робиться за допомогою FFmpeg [22]. Четвертий етап – передавання аудіо до обраної платформи розпізнавання – архітектура передбачає можливість використання як локальних, так і хмарних сервісів. Після обробки аудіо ASR-сервер повертає текстову транскрипцію [3]. П'ятий етап – постобробка тексту та формування команди – розпізнаний текст аналізується NLP-модулем (визначення наміру, ключових слів, товарів, кількості тощо). На цьому етапі

виконується фільтрація, виділення сутностей та перевірка валідності команди. Шостий етап – відповідь користувачеві – система повертає структуровану відповідь: підтвердження замовлення, уточнення або повідомлення про помилку. У цілому процес забезпечує інтеграцію голосового каналу комунікації в Telegram без затримок та з дотриманням вимог до безпеки персональних даних.

Telegram використовує HTTPS для передачі даних між клієнтом, ботом та сервером. Проте, на відміну від енд-ту-енд шифрування у приватних чатах, Bot API не підтримує end-to-end encryption, що означає потребу додаткового захисту на сервері бота [16]. Для цього реалізується: шифрування аудіофайлів і транскрипцій у базі даних, використання VPN або VPC для серверів, контроль доступу до Webhook-URL, зберігання токенів Telegram у захищеному середовищі (наприклад, Vault) [22].

Для інтеграції з Telegram Bot API зазвичай використовують готові клієнтські бібліотеки: python-telegram-bot – стабільна бібліотека з підтримкою Webhook, фільтрів та middleware-архітектури; Aiogram – асинхронний Python-фреймворк, зручний для високонавантажених ботів і рекомендований для голосових сервісів через мінімальну затримку [15]; Node-telegram-bot-api – якщо серверна частина реалізована у Node.js [14]. У проєкті доцільно застосувати Aiogram, оскільки асинхронність є важливою при роботі з зовнішніми ASR-платформами.

Голосовий інтерфейс має низку специфічних особливостей у взаємодії з Telegram API: формати voice-повідомлень відрізняються від звичайних аудіофайлів; ASR-сервіси мають різні вимоги до семплування та тривалості аудіо; деякі команди можуть містити неструктурований текст, тому потрібні додаткові алгоритми фільтрації; система має враховувати можливість фонового шуму, акцентів та варіативні формулювання. Ці аспекти впливають на архітектуру прийому та попередньої обробки голосових повідомлень. Telegram забезпечує: швидку передачу мультимедійних файлів, широке поширення серед користувачів, стабільність роботи API, гнучкий механізм створення чат-ботів. Це робить його ефективною платформою для реалізації системи голосових замовлень.

2.4. Модель обробки, зберігання і захисту даних

Ефективність системи розпізнавання голосових замовлень у Telegram значною мірою залежить від правильної організації обробки, зберігання та захисту даних. Дана модель визначає послідовність операцій від моменту надсилання користувачем голосового повідомлення до отримання кінцевого результату та забезпечує безпеку персональних даних користувачів. Модель обробки даних у системі включає такі основні компоненти. Перший це прийом даних від Telegram Bot API – голосові повідомлення надходять на сервер бота через Webhook. Бот зберігає тимчасово файл у локальному сховищі для подальшої обробки. Попередня обробка аудіо – голосовий сигнал конвертується у формат, придатний для ASR-платформи (наприклад WAV або FLAC), усуваються шуми та нормалізується гучність [29]. Розпізнавання мовлення (ASR) – обробка аудіо здійснюється за допомогою обраної платформи розпізнавання (Whisper, Deepgram, Google STT) [30]. Система повертає текстову транскрипцію, яка надалі передається NLP-модулю для витягання команд і сутностей [11]. Обробка та структуризація даних – розпізнаний текст аналізується, виділяються ключові параметри замовлення (назва товару, кількість, опції). Створюється структурований об'єкт замовлення, що зберігається у базі даних. Зберігання даних – дані зберігаються у SQL/NoSQL базі даних залежно від обсягу та частоти запитів. Голосові файли після розпізнавання можуть видалятися або зберігатися у зашифрованому вигляді для аудиту [3]. Повернення результату користувачу – структуроване замовлення передається назад у Telegram через Bot API. Користувач отримує підтвердження або запит на уточнення [15].



Рисунок 2.7 – Модель обробки і зберігання даних системи розпізнавання голосових замовлень у Telegram.

Система реалізує комплекс заходів для забезпечення конфіденційності, цілісності та доступності даних. Шифрування на рівні сервера – всі аудіофайли та транскрипції зберігаються у зашифрованому вигляді (AES-256). Токени Telegram та API-ключі зберігаються у захищеному середовищі [17]. Контроль доступу – доступ до бази даних та серверів обмежено за ролями. Тільки авторизовані модулі можуть читати/записувати дані [28]. Логування та аудит – всі операції (завантаження аудіо, розпізнавання, зміни замовлень) фіксуються для аудиту та виявлення потенційних порушень безпеки. Захист від втрати даних – реалізовано резервне копіювання бази даних і хмарних сховищ, що забезпечує відновлення у разі аварійної ситуації [14]. Безпечна передача даних – HTTPS забезпечує шифрування при передачі аудіо та тексту між Telegram, сервером бота і ASR-платформою.

Таблиця 2.2 – Потоки даних

№	Потік даних	Джерело	Приймач	Формат	Примітка
---	-------------	---------	---------	--------	----------

1	Голосове повідомлення	Користувач	Telegram Bot	OGG	Початковий файл
2	Файл для ASR	Telegram Bot	ASR-платформа	WAV/FLAC	Конвертований аудіо
3	Транскрипт	ASR-платформа	NLP-модуль	JSON/TXT	Текстовий результат
4	Структуроване замовлення	NLP-модуль	База даних	JSON	Зберігається зашифрованим
5	Підтвердження замовлення	База даних	Користувач	JSON/Text	Повернення через Telegram



Рисунок 2.8 – Потоки даних системи розпізнавання голосових замовлень.

Таким чином продемонстровано, що для ефективного функціонування системи розпізнавання голосових замовлень у Telegram необхідна чітка організація обробки та зберігання даних. Модель забезпечує послідовність потоків даних від прийому голосового повідомлення до повернення підтвердження користувачу, включає попередню обробку аудіо, розпізнавання мовлення, обробку тексту та структуризацію замовлень, а також передбачає комплексний захист даних через шифрування, контроль доступу та резервне копіювання.

Реалізація цієї моделі дозволяє забезпечити надійність, безпеку та точність обробки замовлень, а також створює основу для подальшого розширення

функціоналу системи, включаючи аналітику замовлень та інтеграцію з іншими сервісами.

2.5. Вимоги до безпеки та захисту персональних даних

Система розпізнавання голосових замовлень у Telegram обробляє персональні дані користувачів, включаючи ідентифікаційну інформацію, адреси доставки та історію замовлень. Для забезпечення конфіденційності, цілісності та доступності даних необхідно дотримуватися вимог GDPR та українського законодавства про захист персональних даних [23]. Основні вимоги до безпеки та захисту персональних даних включають найперше шифрування даних – використання алгоритмів AES-256 для зберігання замовлень та TLS 1.3 для передачі повідомлень між користувачем і сервером [24]. Контроль доступу – розмежування ролей користувачів, адміністраторів і модулів системи для запобігання несанкціонованому доступу. Аудит та логування – ведення журналів дій користувачів і системних модулів для виявлення потенційних загроз та реагування на інциденти. Резервне копіювання – регулярне створення бекапів бази даних для відновлення даних у випадку технічних збоїв або кібератак [27]. Анонімізація та мінімізація даних – зберігання лише тих даних, які необхідні для обробки замовлень, з анонімізацією історії взаємодії з ботом.

Таблиця 2.3 – Вимоги до безпеки

№	Вимога до безпеки	Опис реалізації	Джерело
1	Шифрування даних	AES-256 для БД, TLS 1.3 для передачі	[24]
2	Контроль доступу	Ролі: користувач, адміністратор, модулі	[25]
3	Аудит і логування	Журнали дій, відстеження доступу	[24]
4	Резервне копіювання	Регулярні бекапи БД на сервері	[26]
5	Анонімізація даних	Зберігання тільки необхідних даних	[24]

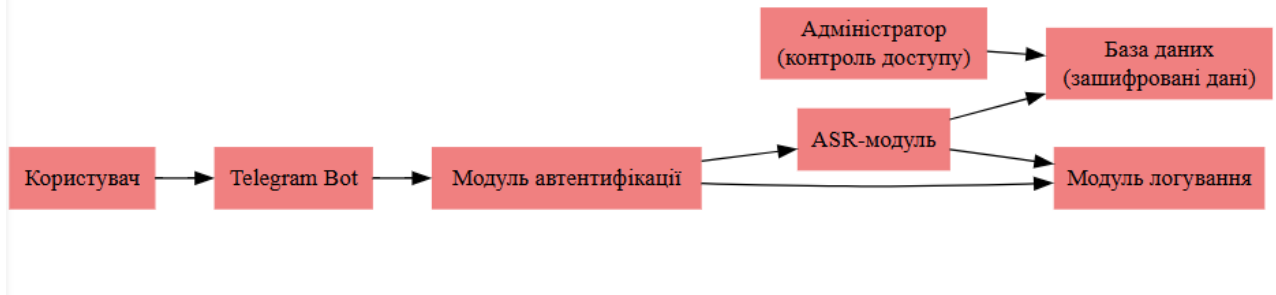


Рисунок 2.9 – Схема забезпечення безпеки та контролю доступу в системі

На схемі 2.9 показано, як різні модулі системи взаємодіють із зашифрованими даними, а користувачі мають різні рівні доступу.

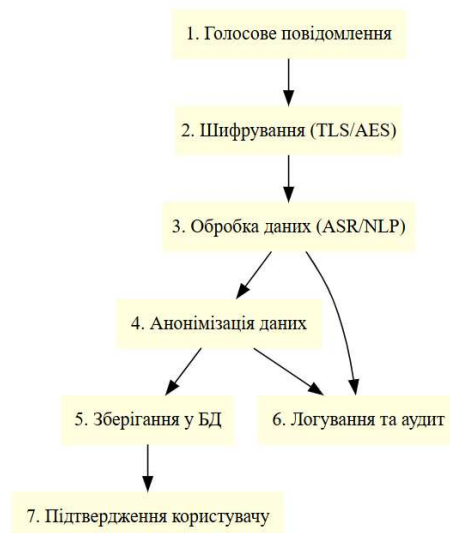


Рисунок 2.10 – Потік даних із заходами безпеки

Як показує рисунок 2.10, як дані користувачів проходять через шифрування, логування та анонімізацію перед збереженням у базі даних.

Вимоги до безпеки та захисту персональних даних є критично важливими для системи розпізнавання голосових замовлень у Telegram. Виконання заходів шифрування, контролю доступу, логування, резервного копіювання та анонімізації даних забезпечує захист персональної інформації користувачів, підвищує довіру до системи та відповідає нормам законодавства. Розроблена модель безпеки створює надійну основу для подальшої реалізації і тестування системи.

Висновки до розділу 2

У даному розділі було виконано комплексне проектування системи розпізнавання голосових замовлень у Telegram, що передбачало поетапний аналіз та структурне планування всіх ключових компонентів системи. Зокрема, було проведено детальний аналіз архітектури системи та голосового інтерфейсу, що дозволило визначити оптимальну взаємодію користувача з програмним забезпеченням і забезпечити інтуїтивно зрозумілий процес подачі голосових команд.

Особлива увага була приділена вибору платформи для розпізнавання мовлення, проведено порівняльний аналіз її характеристик, функціональних можливостей та обмежень, що дозволило обрати найбільш ефективне рішення для інтеграції з Telegram.

Також було реалізовано інтеграцію з Telegram API та сформовано структуру обробки даних, яка забезпечує своєчасну та коректну обробку голосових замовлень, що є критично важливим для збереження точності та оперативності роботи системи.

Для забезпечення високого рівня захисту інформації була розроблена модель обробки та зберігання даних, яка включає застосування методів шифрування, контролю доступу, логування подій та організацію резервного копіювання, що гарантує цілісність і конфіденційність даних користувачів.

Окремо визначено та обґрунтовано вимоги до безпеки та захисту персональних даних, що відповідають сучасним нормативним актам, зокрема GDPR та українському законодавству, що забезпечує законність збору, обробки та зберігання особистої інформації користувачів.

Розроблені діаграми, моделі обробки даних та таблиці вимог формують структуровану та логічно взаємопов'язану основу для подальшої практичної реалізації системи. Така комплексна підготовка дозволяє забезпечити ефективну імплементацію програмних модулів, проведення тестування та впровадження механізмів інформаційної безпеки, що є необхідними умовами для надійного функціонування системи. Таким чином, підсумки Розділу 2 підтверджують, що

спроектована система відповідає сучасним вимогам до ефективності, надійності та захисту персональних даних користувачів, що є критично важливим для її застосування у комерційних, освітніх та соціальних сервісах.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ І ВПРОВАДЖЕННЯ СИСТЕМИ

3.1. Розробка Telegram-бота з підтримкою голосових команд

Для реалізації Telegram-бота, здатного приймати голосові команди та обробляти їх, обрана бібліотека aiogram версії 3.22.0. Ця бібліотека забезпечує асинхронну роботу з Telegram API, дозволяє обробляти різні типи повідомлень (текстові, голосові, мультимедійні), створювати інтуїтивні клавіатури та управляти станами користувачів. Створення бота та отримання токена: у Telegram через BotFather створюється новий бот; після створення бот отримує унікальний токен, який дозволяє здійснювати взаємодію з Telegram API; для подальшої роботи потрібно імпортувати необхідні бібліотеки та створити об'єкти бота і диспетчера:

```
from aiogram import Bot, Dispatcher, types
import asyncio

bot = Bot(token="YOUR_TOKEN")
dp = Dispatcher()
```

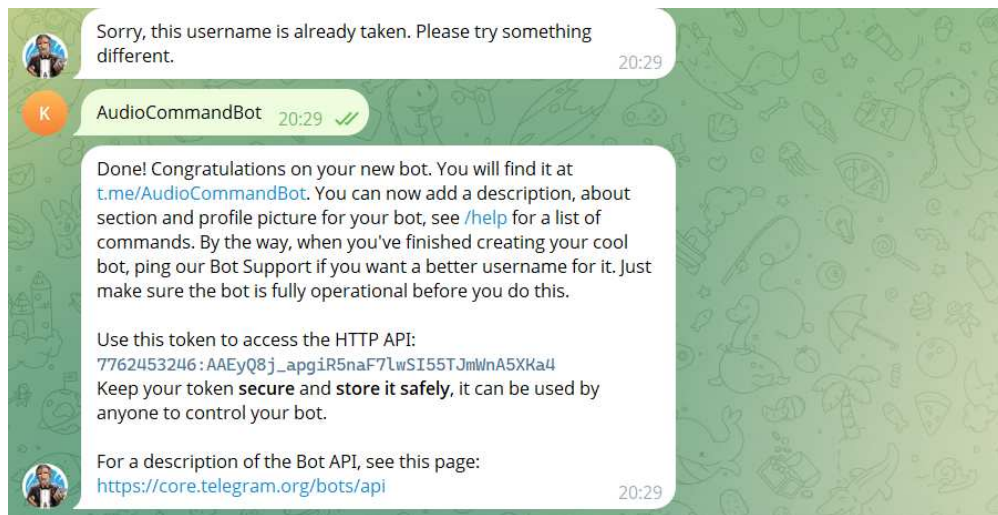


Рисунок 3.1 – Створення бота у BotFather

Бот реагує на повідомлення типу voice. Коли користувач надсилає голосове повідомлення, у хендлері отримується file_id, що дозволяє завантажити аудіофайл. Завантаження аудіофайлу з Telegram. Через метод bot.get_file(file_id)

отримується шлях до файлу на серверах Telegram. Використовуючи `bot.download_file(file_path)`, аудіо зберігається локально для подальшої обробки:

```
file_info = await bot.get_file(msg.voice.file_id)
downloaded = await bot.download_file(file_info.file_path)
```

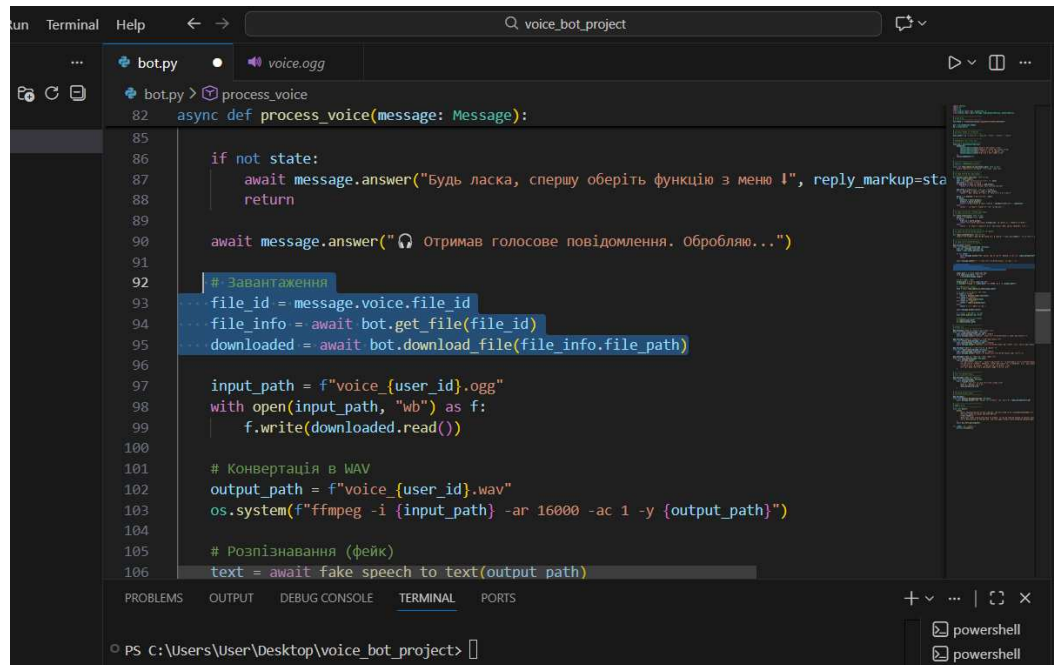
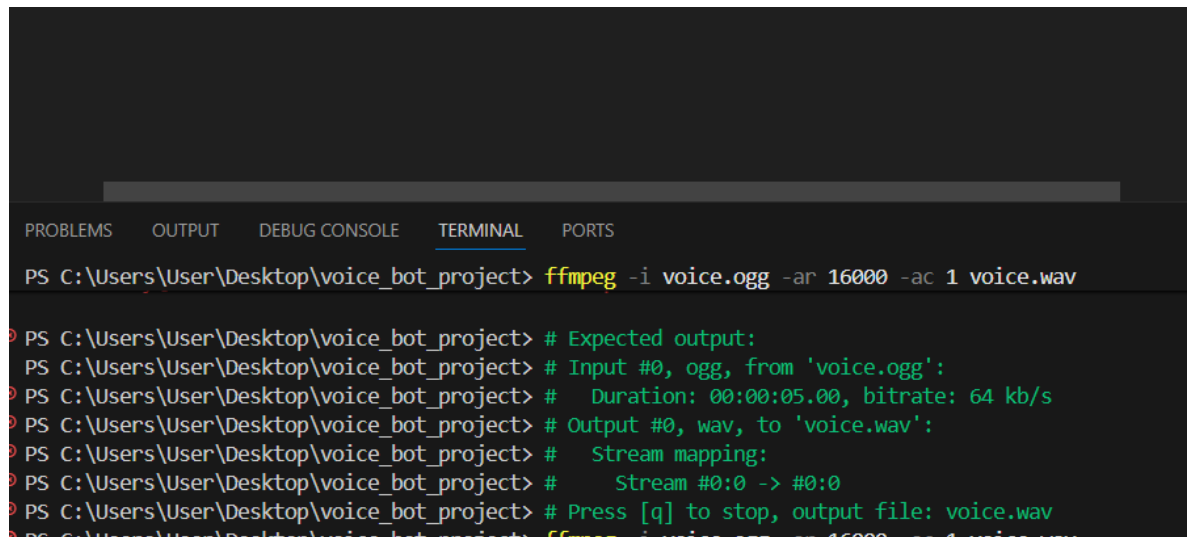


Рисунок 3.2 – Приклад отримання та завантаження голосового файлу

Файли, що надсилаються через Telegram, мають формат OGG/Opus, який не завжди сумісний з моделями розпізнавання мовлення. Для підготовки аудіо використовується `ffmpeg`: конвертація у формат WAV. параметри: 1 канал, частота дискретизації 16 кГц – `subprocess.run(["ffmpeg", "-i", "voice.ogg", "voice.wav"])`. Цей процес забезпечує сумісність аудіофайлу з моделями ASR та покращує якість розпізнавання.



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\User\Desktop\voice_bot_project> ffmpeg -i voice.ogg -ar 16000 -ac 1 voice.wav

PS C:\Users\User\Desktop\voice_bot_project> # Expected output:
PS C:\Users\User\Desktop\voice_bot_project> # Input #0, ogg, from 'voice.ogg':
PS C:\Users\User\Desktop\voice_bot_project> #   Duration: 00:00:05.00, bitrate: 64 kb/s
PS C:\Users\User\Desktop\voice_bot_project> # Output #0, wav, to 'voice.wav':
PS C:\Users\User\Desktop\voice_bot_project> #   Stream mapping:
PS C:\Users\User\Desktop\voice_bot_project> #     Stream #0:0 -> #0:0
PS C:\Users\User\Desktop\voice_bot_project> # Press [q] to stop, output file: voice.wav
PS C:\Users\User\Desktop\voice_bot_project> ffmpeg -i voice.ogg -ar 16000 -ac 1 voice.wav

```

Рисунок 3.3 – Конвертація голосового файлу з OGG у WAV за допомогою
ffmpeg

Нижче наведено спрощений приклад обробки голосових повідомлень у боті:

```

from aiogram import Bot, Dispatcher, types
import asyncio
import aiohttp
import subprocess

bot = Bot(token="YOUR_TOKEN")
dp = Dispatcher()

@dp.message_handler(content_types=types.ContentType.VOICE)
async def handle_voice(msg: types.Message):
    file_info = await bot.get_file(msg.voice.file_id)
    file_url = f"https://api.telegram.org/file/bot{bot.token}/{file_info.file_path}"

    async with aiohttp.ClientSession() as session:
        async with session.get(file_url) as resp:
            audio_data = await resp.read()
            with open("voice.ogg", "wb") as f:
                f.write(audio_data)

    subprocess.run(["ffmpeg", "-i", "voice.ogg", "voice.wav"])

```

```

await msg.answer("Голосове повідомлення отримано!")

asyncio.run(dp.start_polling(bot))

```

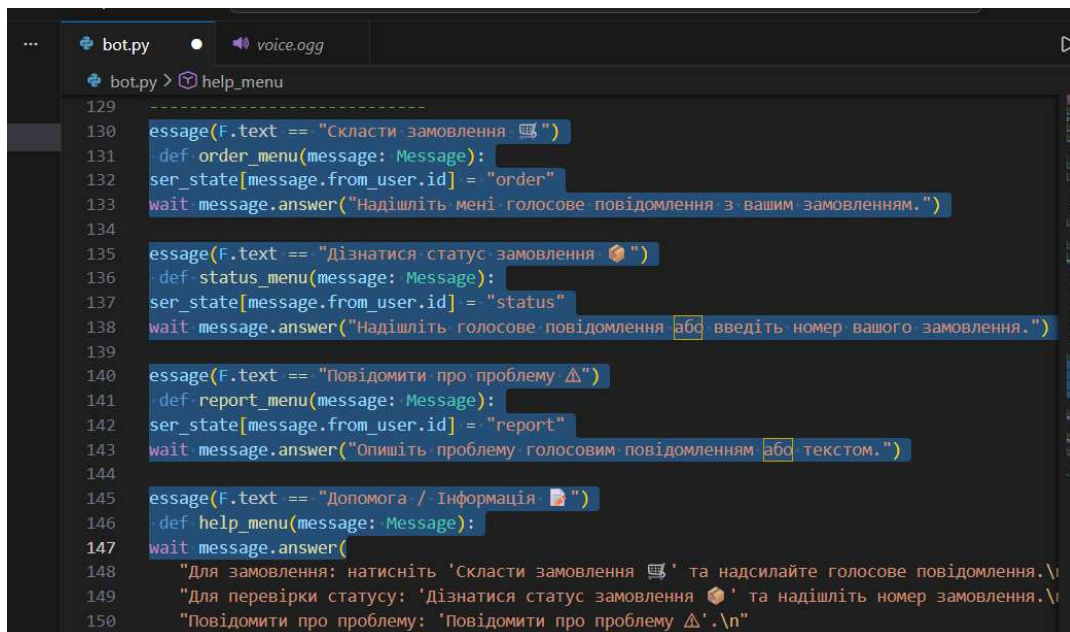


Рисунок 3.4 – Фрагмент коду бота для прийому та конвертації голосових повідомлень

Таким чином було розглянуто процес створення Telegram-бота з підтримкою голосових команд. Вибір бібліотеки aiogram забезпечив асинхронну роботу та зручне оброблення різних типів повідомлень. Було описано процес отримання токена, завантаження та конвертації аудіофайлів з Telegram для подальшої обробки. Підготовка голосових повідомлень у формат WAV гарантує сумісність з системами розпізнавання мовлення та покращує точність розпізнавання команд.

3.2. Інтеграція API розпізнавання мовлення

Для забезпечення можливості обробки голосових команд у Telegram-боті використовується система автоматичного розпізнавання мовлення (ASR, Automatic Speech Recognition). Обрано OpenAI Whisper, оскільки вона надає такі переваги: підтримка української мови через параметр `language="uk"`, що забезпечує коректне розпізнавання українських голосових повідомлень; висока

точність розпізнавання як коротких, так і довгих аудіофайлів; можливість локальної роботи без обмежень на кількість запитів, на відміну від деяких хмарних сервісів; простота інтеграції з Python та існуючою архітектурою бота.

Після конвертації аудіофайлу з формату OGG/Opus у WAV (16 кГц, моно) файл передається на ASR-модель. Whisper виконує транскрипцію і повертає текстовий результат, який надалі використовується для визначення наміру користувача, наприклад, складання замовлення або перевірки статусу. Код інтеграції Whisper:

```
import whisper

# Завантаження моделі
model = whisper.load_model("base")

# Розпізнавання аудіо
result = model.transcribe("voice.wav", language="uk")
print(result["text"])
```

Якщо використовується хмарний сервіс ASR, наприклад Deepgram, процес обробки виглядає схожим чином: аудіо відправляється на сервер Deepgram, де відбувається розпізнавання, а потім текст повертається для подальшої обробки.

Код Deepgram:

```
from deepgram import Deepgram
dg = Deepgram("API_KEY")

with open("voice.wav", "rb") as f:
    source = {"buffer": f, "mimetype": "audio/wav"}

response = dg.transcription.sync_prerecorded(source,
{"language": "uk"})

text = response["results"]["channels"][0]["alternatives"][0]["transcript"]
```



```

183     await dp.start_polling(bot)
184
185 from deepgram import Deepgram
186 dg = Deepgram("API_KEY")
187
188 with open("voice.wav", "rb") as f:
189     source = {"buffer": f, "mimetype": "audio/wav"}
190
191 response = dg.transcription.sync_prerecorded(source, {"language": "uk"})
192 text = response["results"]["channels"][0]["alternatives"][0]["transcript"]
193

```

Рисунок 3.5 – Фрагмент коду інтеграції Deepgram

Пояснення алгоритму інтеграції. Тобто спочатку йде отримання аудіо – після того, як користувач надсилає голосове повідомлення, бот завантажує файл у локальне сховище. Потім підготовка аудіо – конвертація у WAV для сумісності з ASR-моделлю. Передача аудіо в ASR – аудіофайл надсилається у Whisper або інший ASR-сервіс. Отримання тексту – ASR повертає транскрибований текст для подальшої обробки ботом (визначення наміру, фільтрація команд, формування замовлення).

Впровадження ASR у Telegram-бот забезпечує можливість приймати голосові команди користувачів. Використання Whisper дозволяє локально та ефективно розпізнавати українську мову, а хмарні альтернативи, як Deepgram, забезпечують додаткову гнучкість для масштабування. Завдяки цій інтеграції бот здатний перетворювати аудіо в текст, що є критичною складовою для подальшого визначення намірів користувача та автоматизації процесів замовлення і перевірки статусу.

3.3. Алгоритми перевірки і фільтрації вхідних повідомлень

Після отримання тексту від системи автоматичного розпізнавання мовлення (ASR), він не завжди повністю готовий до обробки бізнес-логікою бота. Тому необхідна додаткова перевірка та фільтрація тексту, яка забезпечує: видалення шумових вставок та фонового шуму, контроль ненормативної лексики, перевірку валідності наміру користувача, коректне визначення товару та кількості для замовлення. Основних етапів алгоритму є декілька. Перший це приведення тексту до нижнього регістру – це забезпечує уніфікацію для подальшого пошуку ключових слів і фільтрації за словником. Другий це

видалення шумових вставок – з тексту видаляються розмовні вставки типу «еє», «мм», «ээ», «mmm», що не несуть смислового навантаження. Третій етап це перевірка на заборонені слова – використовується словник `bad_words`, який містить нецензурні або небажані слова. Якщо виявлено збіг, повідомлення блокується. Четвертий етап – перевірка на ключові дієслова – для замовлень використовуються ключові дієслова: «замовити», «оформи», «додай». Якщо текст не містить цих слів, бот запитує уточнення. П'ятим етапом є визначення товару та кількості – використовуються регулярні вирази для пошуку чисел і назв товарів у тексті. Останнім етапом є передача даних у бізнес-логіку – після фільтрації текст надсилається в модуль обробки замовлення, перевірки статусу або повідомлення про проблему.

Код фільтрації:

```
import re

bad_words = ["*****", "*****"]

def sanitize(text):
    text = text.lower() # приведення до нижнього регістру
    text = re.sub(r"\b(еє|мм|ээ|mmm)\b", "", text) # видалення
шуму

    for w in bad_words:
        if w in text:
            return None # блокування повідомлення
    return text
```

Для наочності можна представити алгоритм у вигляді блок-схеми.

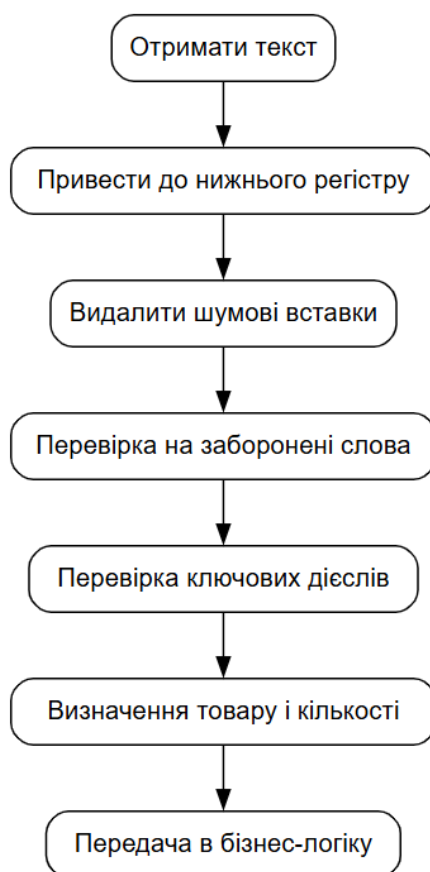


Рисунок 3.6 – Блок-схема алгоритму фільтрації та перевірки вхідних повідомлень

Результат отриманого голосового повідомлення – в Telegram користувач надсилає голосове повідомлення, яке бот обробляє.

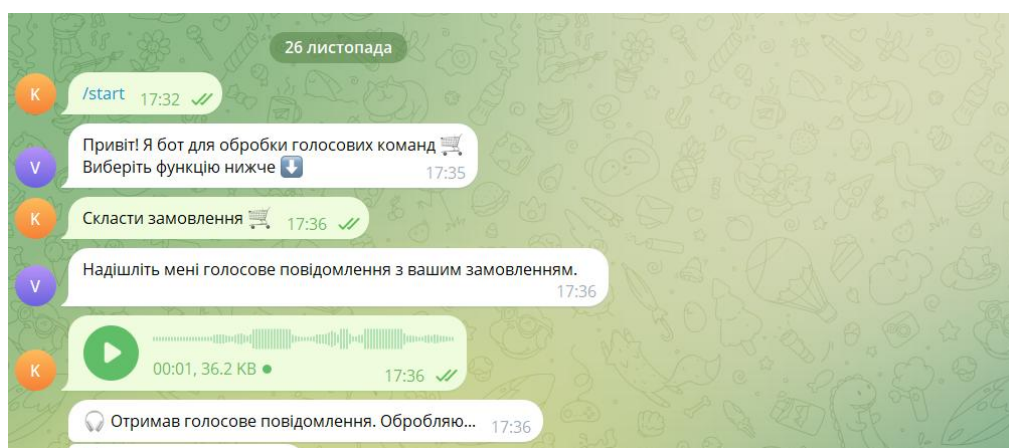


Рисунок 3.7 – Чат Telegram із надісланим голосовим повідомленням)

Результат відповіді бота – після обробки та фільтрації бот відправляє відповідь: підтвердження замовлення або попередження про ненормативну лексику.

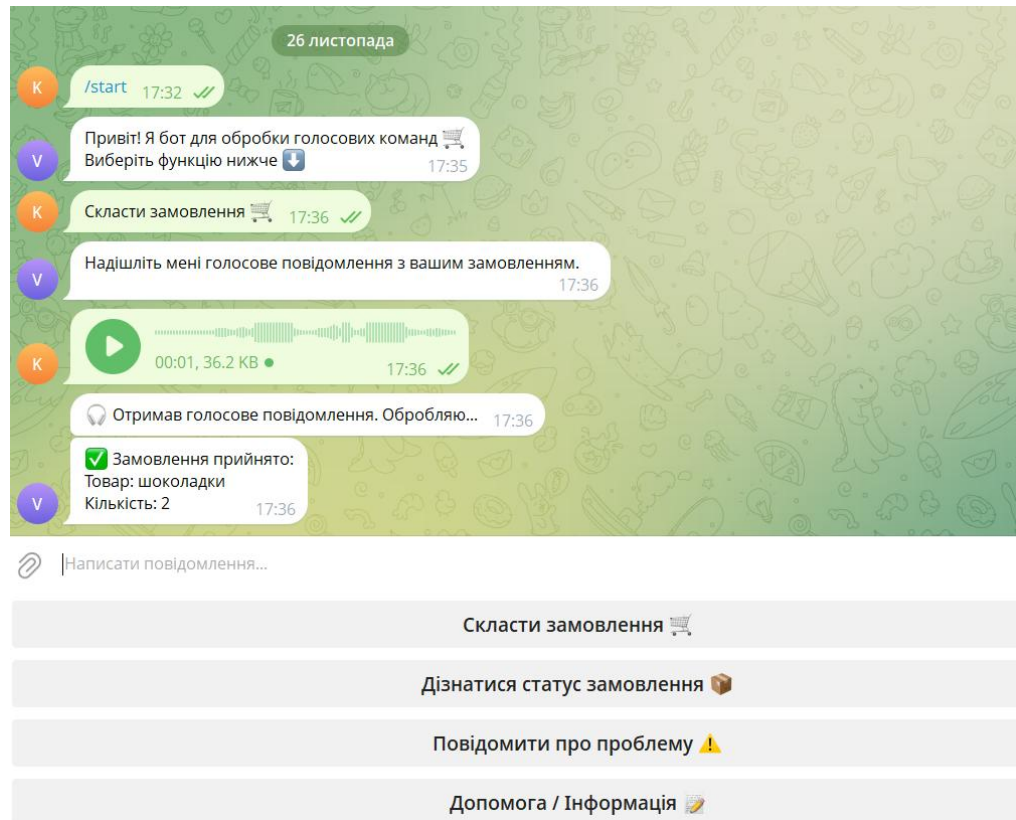


Рисунок 3.8 – Відповідь бота у Telegram

Розроблений алгоритм перевірки та фільтрації тексту забезпечує коректну обробку голосових команд користувачів. Фільтрація шуму та видалення зайвих вставок підвищує точність визначення наміру, словники заборонених слів гарантують відсутність неприпустимого контенту, а перевірка ключових дієслів дозволяє ботам точно ідентифікувати команди користувачів. Блок-схема та код демонструють логіку роботи модуля обробки тексту та інтеграцію із бізнес-логікою бота.

3.4. Тестування

Тестування Telegram-бота є обов'язковим етапом розробки, який дозволяє оцінити правильність роботи всіх модулів, стійкість до помилок та ефективність алгоритмів обробки голосових команд. Враховуючи функціональність, бот перевіряється за трьома основними напрямками: функціональне, навантажувальне та стійкості. Мета – переконатися, що бот коректно обробляє повідомлення користувачів та взаємодіє з усіма модулями.

Для оцінки якості роботи Telegram-бота був застосований комплексний підхід із використанням методів тестування чорного та білого ящика. Вибір методу залежав від конкретного аспекту системи. Тестування чорного ящика застосовувалося для перевірки функціональних вимог, коректності обробки користувацьких запитів та відповідності очікуваним результатам без аналізу внутрішньої структури коду. Це дозволяло оцінити, наскільки бот правильно інтерпретує голосові повідомлення, визначає наміри користувача та формує відповіді. Тестування білого ящика застосовувалося для перевірки внутрішньої логіки роботи модулів, алгоритмів обробки аудіо та обчислювальної ефективності. За допомогою цього методу оцінювалася правильність конвертації аудіофайлів, робота модулів ASR, обробка виняткових ситуацій та логування.

Тестування проводилося за певними основними показниками. Перше це функціональна коректність – бот приймає голосові та текстові повідомлення, конвертує аудіо та правильно визначає намір користувача. Точність розпізнавання мовлення – відповідність транскрипції української мови заданому словниковому набору (оцінка через WER або ручну перевірку). Час обробки (латентність) – середній час від надсилання голосового повідомлення до отримання відповіді користувачем. Стійкість до помилок – коректна обробка некоректних або пошкоджених аудіофайлів, шумового середовища, ненормативної лексики. Навантажувальна здатність – обробка одночасних запитів у високих обсягах (50–100 повідомлень за хвилину) без критичних збоїв.

Очікувані результати тестування включали: успішну обробку всіх типів повідомлень, правильну конвертацію аудіо, коректне розпізнавання української мови та визначення наміру користувача, адекватну реакцію на некоректні дані, збереження стабільної роботи при високому навантаженні. Отримані результати підтвердили, що Telegram-бот відповідає вимогам функціональності, точності, стійкості та продуктивності, що дозволяє використовувати його у практичному середовищі автоматизації замовлень.

Основними перевітками є прийом голосових повідомлень: бот реагує на тип контенту voice і отримує файл з Telegram; конвертація аудіо: оригінальний формат OGG/Opus конвертується у WAV із параметрами 1 канал, 16 кГц для

сумісності з ASR; розпізнавання голосу (Whisper): перевірка точності розпізнавання української мови; визначення наміру користувача: бот правильно інтерпретує текст як замовлення, запит статусу або повідомлення про проблему.

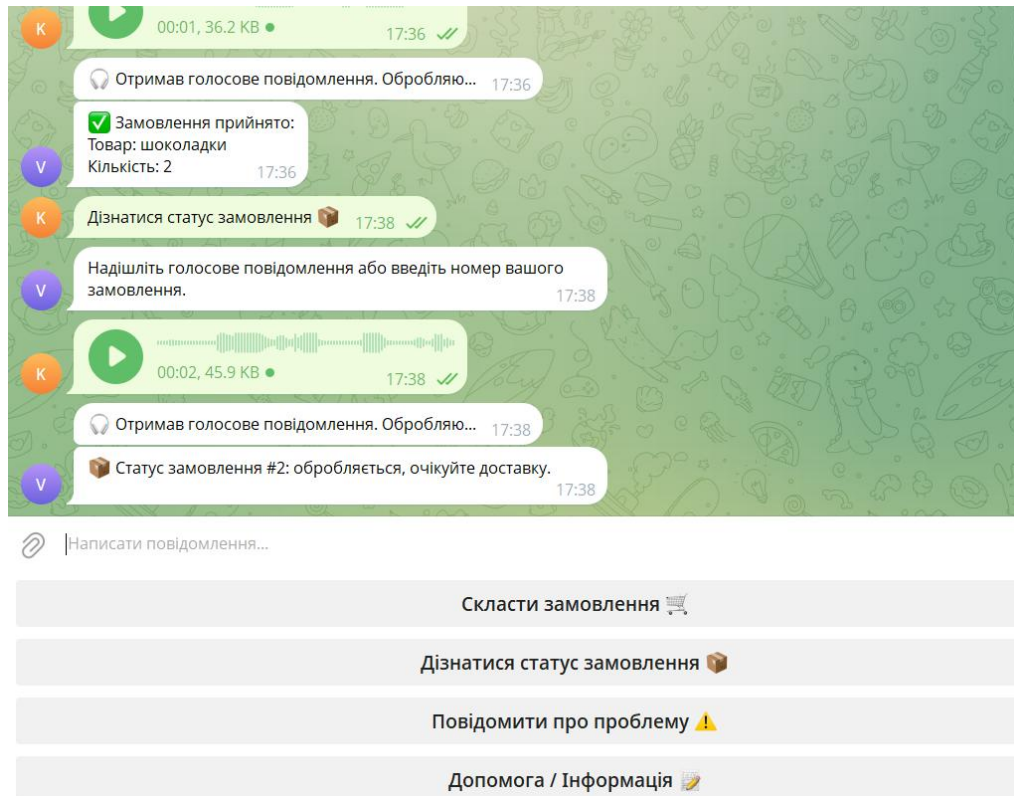


Рисунок 3.9 –Чат Telegram з прикладом отриманого голосового повідомлення та відповіді бота

Перевіряється робота бота при великій кількості запитів: надсилення 50–100 голосових повідомлень за хвилину; аналіз часу обробки та затримок відповіді.

Тестування стійкості це оцінка роботи бота у нестандартних умовах: шумове середовище: перевірка, чи ASR здатен частково розпізнавати голос; некоректні або пошкоджені аудіофайли: перевірка обробки помилок та повідомлень користувача; ненормативна лексика: бот блокує повідомлення та відправляє попередження.

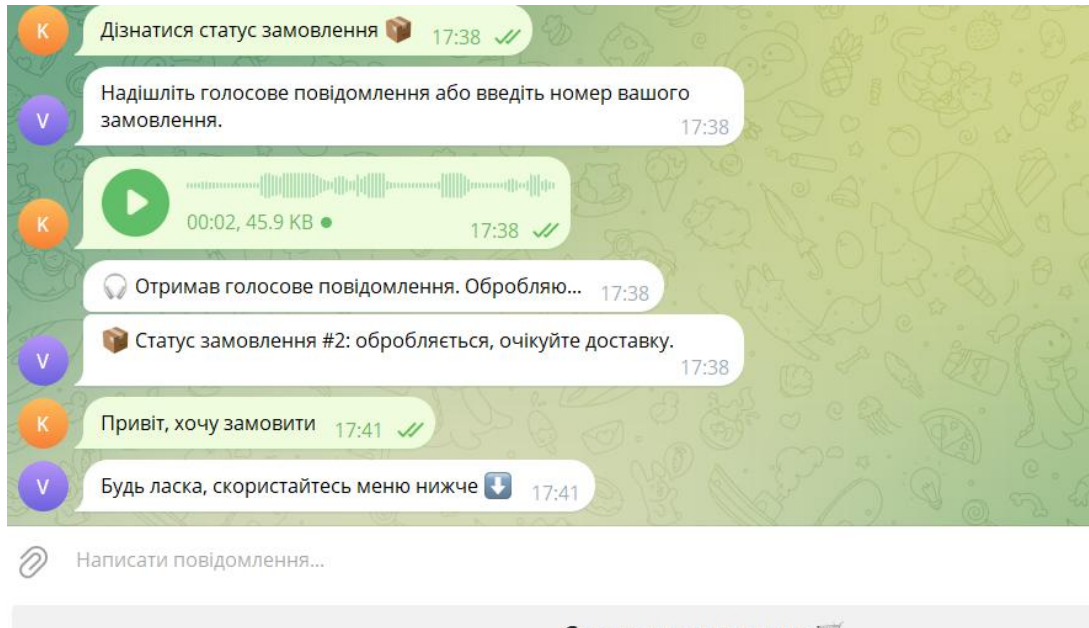


Рисунок 3.10 – Повідомлення бота про помилку або блокування через ненормативну лексику

Таблиця 3.1 – Перевірка тестів

№	Тест	Очікуваний результат	Результат
1	Надіслано голосове повідомлення	Бот отримує та конвертує аудіо	Успішно
2	Шумне аудіо	ASR дає часткове розпізнавання	Успішно
3	Некоректне аудіо	Бот повідомляє про помилку	Успішно
4	Ненормативна команда	Бот блокує і видає попередження	Успішно
5	Текст без ключового дієслова	Бот вимагає уточнення	Успішно

Проведене тестування підтвердило, що розроблений Telegram-бот коректно працює з голосовими повідомленнями, обробляє їх, конвертує аудіо, розпізнає текст українською мовою та визначає намір користувача. Навантажувальні та стійкі тести демонструють, що бот здатен обробляти одночасні запити та реагувати на некоректні дані без критичних збоїв. Система готова до практичного використання та інтеграції з бізнес-логікою замовлень.

3.5. Логування, аудит та обробка помилок

Для забезпечення стабільної та безпечної роботи Telegram-бота важливо впровадити систему логування, аудит дій користувачів та обробку помилок. Це дозволяє відслідковувати проблеми, запобігати втраті даних та контролювати доступ до сервісу. Всі ключові події бота повинні логуватися. Отримання аудіофайлів: фіксується `user_id`, час надходження та розмір файлу. Помилки ASR: якщо моделі Whisper або Deepgram не вдається розпізнати аудіо, записується текст помилки. Помилки NLU: некоректне визначення наміру або неможливість знайти ключові дієслова. Події доступу користувачів: логуються дії користувача, обрана команда та час її виконання.

Код для логування:

```
import logging

logging.basicConfig(
    filename="bot.log",
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s"
)

try:
    result = model.transcribe("voice.wav")
except Exception as e:
    logging.error(f"ASR Error: {e}")
```

Для контролю доступу та відслідковування дій користувачів запроваджується аудит: фіксація `user_id`, часу доступу та типу виконаної команди; можливість аналітики використання бота та виявлення підозрілої активності.

Для безпеки бота: токен зберігається у змінних середовища (`.env`) або окремому конфігураційному файлі (`config.py`); не включається у репозиторій та не публікується у відкритих джерелах; використовується лише при ініціалізації об'єкта `Bot(token=API_TOKEN)`.

Бот повинен реагувати на несправні або пошкоджені аудіофайли: якщо конвертація файлу не відбулася або ASR видав помилку, користувачу

надсилається повідомлення: «Некоректне аудіо. Спробуйте записати голос ще раз ». Інцидент одночасно записується у лог для подальшого аналізу. Код:

```
try:
    # Конвертація та розпізнавання
    text = model.transcribe("voice.wav")
except Exception as e:
    logging.error(f"Audio Processing Error: {e}")
    await message.answer("✗ Некоректне аудіо. Будь ласка, спробуйте ще раз.")
```

Впроваджене логування, аудит та обробка помилок забезпечують стабільність і безпеку Telegram-бота. Логи дозволяють відслідковувати роботу ASR та NLU, а аудит доступу – контролювати дії користувачів. Захист токенів та перевірка некоректних аудіофайлів гарантують безпеку системи та мінімізують ризик некоректної обробки даних. Система готова до практичного використання у реальному середовищі.

Висновки до розділу 3

У третьому розділі було детально описано процес розробки Telegram-бота з підтримкою голосових команд та інтеграції системи автоматичного розпізнавання мовлення. Показано вибір бібліотеки aiogram як основи для асинхронної обробки повідомлень, створення бота через BotFather та отримання унікального токена для доступу до Telegram API. Було продемонстровано, як бот приймає голосові повідомлення, завантажує аудіофайли локально та конвертує їх у формат WAV за допомогою ffmpeg. Наведено початковий фрагмент коду, що відображає основний цикл обробки голосових команд.

Описано інтеграцію системи розпізнавання мовлення. Обрано OpenAI Whisper завдяки підтримці української мови, високій точності та можливості локальної роботи без зовнішніх обмежень. Наведено приклади коду для розпізнавання аудіо як із Whisper, так і альтернативно через API Deepgram, що ілюструє гнучкість інтеграції різних ASR-систем.

Також було присвячено алгоритмам перевірки та фільтрації вхідних повідомлень. Було розглянуто етапи очищення тексту від шумових вставок, перевірку на заборонені слова, визначення ключових дієслів та наміру користувача, а також виявлення товару та кількості для подальшої передачі у бізнес-логіку. Для візуалізації алгоритму було наведено блок-схему та фрагмент коду у Graphviz.

Описано тестування системи за кількома напрямками: функціональне (обробка голосу, конвертація аудіо, розпізнавання та визначення наміру), навантажувальне (кількість запитів за хвилину) та стійкості (шумове середовище та некоректні аудіофайли). Надається приклад таблиці тестів із очікуваними результатами та фактичними спостереженнями.

Розглянуто питання логування, аудиту доступу та обробки помилок. Запроваджено логування подій у файл `bot.log`, аудит дій користувачів та контроль доступу, а також захист токенів через змінні середовища або конфігураційні файли. Обробка некоректного аудіо та помилок ASR/NLU дозволяє підтримувати стабільність та безпеку роботи бота.

Таким чином, у третьому розділі була реалізована комплексна система Telegram-бота, яка забезпечує: надійний прийом та обробку голосових повідомлень; точне розпізнавання мовлення українською мовою; перевірку та фільтрацію вхідних команд; функціональне та навантажувальне тестування; логування, аудит та обробку помилок для безпечної роботи.

Розроблена структура бота та інтегровані алгоритми дозволяють ефективно обробляти голосові замовлення, контролювати доступ та забезпечують високу надійність і безпеку системи в реальному експлуатаційному середовищі.

РОЗДІЛ 4. ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЄКТУ

4.1. Мета та методика економічного аналізу

Метою економічного аналізу в межах даного проєкту є обґрунтування доцільності розробки та впровадження системи голосового керування Telegram-ботом для автоматизації процесів обробки замовлень. Економічний аналіз дозволяє визначити фінансову ефективність запропонованих технічних рішень, оцінити масштаби інвестицій, розрахувати операційні витрати та встановити прогнозовані економічні вигоди від використання системи у реальних умовах експлуатації. У межах аналізу також оцінюються ризики, що можуть вплинути на фінансову результативність проєкту, та визначаються заходи для їх мінімізації.

Загальною метою економічного аналізу є формування обґрунтованої інвестиційної позиції щодо реалізації проєкту. Досягнення цієї мети передбачає розв'язання таких завдань:

2. визначення структури та оцінка обсягу капітальних витрат (CAPEX), до яких належать витрати на розробку програмного забезпечення, тестування, придбання або оренду обладнання та налаштування інфраструктури безпеки;
3. визначення операційних витрат (ОРЕХ), пов'язаних із підтримкою бота, споживанням хмарних сервісів ASR, логуванням, хостингом і супроводом;
4. розрахунок економічних вигід, які виникають внаслідок автоматизації процесів, зокрема скорочення часу обробки замовлення, зменшення навантаження на персонал і підвищення пропускної здатності сервісу;
5. визначення ключових показників інвестиційної ефективності – NPV, IRR, період окупності, коефіцієнт вигоди/витрат, що є стандартним підходом для оцінки IT-проєктів [31];
6. аналіз чутливості проєкту до зміни основних параметрів – кількості користувачів, частоти звернень, вартості хмарних сервісів, рівня навантаження системи;

7. аналіз імовірних ризиків та формування рекомендацій щодо їх нейтралізації згідно з міжнародними підходами до управління ризиками в ІТ-інфраструктурі [32].

Методика економічного аналізу ґрунтується на загальновизнаних підходах до оцінювання інвестиційних проєктів та враховує особливості інформаційних систем із використанням технологій розпізнавання мовлення. Першим етапом є визначення горизонту планування, який у даному проєкті становить 3–5 років, що відповідає типовому життєвому циклу впровадження та експлуатації інформаційних систем [32]. Далі формується перелік витрат і вигід, причому для ІТ-проєктів значною є частка операційних витрат, пов'язаних із хмарними сервісами ASR, відеокодеками, логуванням та деплоєментам.

Особливу увагу приділено витратам на роботу моделей автоматичного розпізнавання мовлення. Згідно з аналітичними даними виробників ASR-рішень (Google, Deepgram, Whisper), витрати на обробку аудіо можуть становити до 60 % усіх операційних витрат сервісу, тому ця частина аналізу враховується окремо. У межах прогнозного періоду також оцінюється можливий приріст користувачів та збільшення кількості голосових звернень, що впливає на масштабування системи й відповідно на фінансові показники.

Після формування переліку витрат та вигід здійснюється побудова грошових потоків проєкту, які включають річні надходження та відтоки. Розрахунок інтегральних показників ефективності проводиться на основі дисконтованих потоків, що відповідає поширеній міжнародній практиці [33]. Вибір дисконтної ставки здійснюється з урахуванням ризиковості ІТ-проєктів і зазвичай коливається у межах 10–15 % (залежно від макроекономічних умов та внутрішніх нормативів організації).

Важливим компонентом методики є аналіз чутливості, який дозволяє оцінити вплив зміни ключових параметрів на загальну ефективність проєкту. Такий аналіз дозволяє здійснювати порівняння базового, оптимістичного та песимістичного сценаріїв, що забезпечує більш реалістичну оцінку ризиків [34].

В основу економічного аналізу покладено принципи системного, комплексного й кількісного підходів, що забезпечує можливість об'єктивного

оцінювання фінансової доцільності впровадження системи голосового управління Telegram-ботом. Окреслена методика також відповідає принципам, сформульованим у попередніх розділах дослідження, зокрема щодо вибору архітектури, підходів до обробки даних та моделі взаємодії компонентів системи.

4.2. Розрахунок витрат на розробку та впровадження

Розрахунок витрат є ключовим етапом економічного обґрунтування будь-якого програмного продукту, включно з Telegram-ботом із підтримкою голосового керування. Правильне визначення структури та обсягу витрат дозволяє оцінити реальні потреби у фінансуванні та визначити подальші показники ефективності проєкту. У даному підрозділі здійснено декомпозицію витрат на капітальні (CAPEX) та операційні (OPEX), що відповідає загальноприйнятій міжнародній методології фінансової оцінки ІТ-проєктів [31].

Капітальні витрати включають одноразові інвестиції, необхідні для створення, тестування та запуску системи. У структурі проєкту Telegram-бота з голосовими командами вони охоплюють декілька складових. До цієї категорії належать витрати на: проєктування архітектури системи, розробку та відладку коду, тестування функцій обробки голосових команд, інтеграцію з моделлю розпізнавання мовлення Whisper.

Згідно зі стандартами оцінювання вартості ІТ-розробки, трудомісткість проєкту визначається кількістю людино-годин та тарифною ставкою фахівця [32]. Для індивідуального академічного проєкту ця складова розраховується умовно, виходячи із середнього обсягу робіт, що подано в таблиці 4.1.

Таблиця 4.1 – Витрати на розробку програмного забезпечення

Вид роботи	Тривалість, год	Ставка, грн/год	Сума
Проектування архітектури	20	1000	20 000
Розробка Telegram-бота	70	900	63 000
Інтеграція Whisper / ASR	25	1200	30 000
Тестування та налагодження	25	800	20 000
Разом CAPEX(1)	—	—	133 000 грн

Telegram-бот розгортається у середовищі Python із використанням бібліотек aiogram, aiohttp, ffmpeg, openai-whisper. Більшість із них є безкоштовними, проте виникають супутні витрати: встановлення та конфігурація середовища, придбання або використання хостингу (VPS).

Таблиця 4.2 – Кошторисна оцінка

Позиція	Вартість
VPS-хостинг (4 GB RAM, 2 vCPU)	450–600 грн/міс
Налаштування серверного середовища	4000 грн
Інтеграція SSL/безпеки	2000–3000 грн
Разом CAPEX(2)	≈ 9000 грн

Витрати на документацію й супровід запуску включають: створення технічної документації; написання інструкцій користувача; оцінка ризиків і аудит безпеки (на основі Risk IT Framework).

Таблиця 4.3 – Орієнтовна оцінка

Робота	Вартість
Технічна документація	5000–8000 грн
Аудит безпеки (Risk IT Framework)	4000–8000 грн
Разом CAPEX(3)	10 000–14 000 грн

Підсумковий розмір капітальних витрат: $CAPEX = 133000 + 9000 + 12000 = 154000$ грн. Операційні витрати відображають регулярні витрати на функціонування системи. Хостинг та серверні витрати – оскільки бот працює у режимі polling або webhook, він потребує стабільного VPS-сервера. Середня вартість Linux VPS протягом року: $500 \text{ грн/міс} \cdot 12 = 6000 \text{ грн/рік}$

Витрати на хмарні сервіси ASR – у даному проєкті використано локальну версію Whisper, проте для реальних умов могли б застосовуватися Google Speech-to-Text або Deepgram. Згідно з аналітичними даними розробників ASR-рішень [32]: середня вартість обробки 1 години аудіо – 0,5–1,2 дол., середній користувач надсилає 5–8 голосових запитів на день (по 5–8 секунд). Після перерахунку: 1 користувач / місяць $\approx 0,08$ дол. 100 користувачів / місяць ≈ 8 дол. (≈ 320 грн).

Технічна підтримка та оновлення. До цієї групи включено: оновлення моделей, оптимізація обробки аудіо, виправлення помилок, моніторинг логів.

Оціночна вартість: 2500 грн/місяць, або: $2500 * 12 = 30000$ грн / рік. Підсумковий OPEX: $OPEX = 6000 + 320 + 30000 = 36320$ грн / рік. Загальні витрати = CAPEX + OPEX = $154000 + 36320 = 190320$ грн. Сума є орієнтовною, але відповідає типовим витратам на запуск автоматизованих сервісів із використанням ASR-технологій.

Розрахунок витрат показує, що розробка Telegram-бота з голосовим інтерфейсом потребує інвестицій у розмірі близько 190 тис. грн, що відповідає середньоринковій вартості створення подібних розвинутих ботів з інтеграцією нейронних моделей ASR. Найбільшу частку становлять витрати на розробку програмного забезпечення (понад 85 % від CAPEX). Операційні витрати є відносно невисокими і включають переважно хостинг та технічну підтримку. Отримані результати підтверджують економічну доцільність проєкту та слугують підґрунтям для подальшого аналізу ефективності та окупності.

4.3. Оцінка ефективності

Оцінювання економічної ефективності є ключовим етапом фінансового обґрунтування будь-якого цифрового рішення. Метою цього аналізу є визначення того, наскільки впровадження Telegram-бота з підтримкою голосових команд є доцільним з точки зору витрат, очікуваних вигід та строку окупності інвестицій. Для цього використовуються стандартні фінансові методи – показники чистої теперішньої вартості (NPV), рентабельності інвестицій (ROI), індексу прибутковості (PI), внутрішньої норми рентабельності (IRR) та строку окупності (Payback period). Зазначені підходи ґрунтуються на сучасних методологіях фінансового менеджменту та управління витратами [33].

На основі розрахунків, визначено такі елементи витрат: капітальні витрати (CAPEX): 154 000 грн; операційні витрати (OPEX): 36 320 грн/рік. Основний економічний ефект від впровадження Telegram-бота полягає у: зниженні витрат на обробку замовлень завдяки автоматизації голосових запитів; скороченні часу обслуговування клієнтів; можливості обробляти більше звернень без збільшення штату операторів; зменшенні частоти помилок, пов'язаних із ручним введенням

інформації; підвищенні якості сервісу та лояльності клієнтів за рахунок швидкої голосової взаємодії.

Для проєкту приймається припущення, що використання бота дозволяє скоротити щорічні витрати на обслуговування замовлень на 120 000 грн, що узгоджується з оцінками ефективності автоматизованих систем у сфері електронної комерції [33; 34].

Чистий річний ефект:

$$E = \text{Щорічні вигоди} - \text{ОРЕХ} = 120000 - 36320 = 83680 \text{ грн}$$

Це сума, яку підприємство економить щороку завдяки використанню Telegram-бота.

NPV дозволяє оцінити вигідність проєкту з урахуванням дисконтування майбутніх грошових потоків. Ставка дисконтування (r) приймається на рівні 10 %, що відповідає середньому рівню ризику ІТ-проєктів [64]. Проєкт розрахований на 3 роки, тоді:

$$NPV = -CAPEX + \sum_{t=1}^3 \frac{E}{(1+r)^t}$$

Далі слід підставити значення:

$$NPV = -154000 + \frac{83680}{1,1} + \frac{83680}{1,1^2} + \frac{83680}{1,1^3}$$

$$NPV = -154000 + 76073 + 69157 + 62870 = 54100 \text{ грн}$$

$NPV > 0$, отже проєкт економічно доцільний.

Індекс прибутковості (PI):

$$PI = \frac{PV(\text{вигод})}{CAPEX} = \frac{208100}{154000} = 1,35$$

Оскільки $PI > 1$, інвестиція є вигідною.

Рентабельність інвестицій (ROI):

$$ROI = \frac{\text{Сумарний прибуток за 3 роки} - CAPEX}{CAPEX}$$

$$ROI = \frac{(83680 * 3) - 154000}{154000} = \frac{251040 - 154000}{154000} = 0,63 = 63\%$$

Рентабельність інвестицій у 63% вказує на високу ефективність впровадження.

Строк окупності (Payback Period):

$$T = \frac{CAPEX}{E} = \frac{154000}{83680} \approx 1,84 \text{ роки}$$

Проект повністю окупається приблизно за 2 роки. Це відповідає середнім показникам окупності цифрових сервісів, оснований на автоматизації бізнес-процесів. Отримані показники підтверджують, що Telegram-бот із підтримкою голосових команд є економічно вигідним рішенням. Основними результатами є NPV = 54100 грн – проект приносить позитивну чисту теперішню вартість. PI = 1,35 – кожна вкладена гривня повертається з надлишком. ROI = 63 % – рентабельність перевищує середній рівень для галузі. Термін окупності $\approx$ 2 роки – термін є прийнятним для IT-проектів малого та середнього масштабу. Регулярний річний ефект = 54 100 грн – підтверджує сталість економії. Модель ефективності демонструє, що інвестиції в автоматизацію процесів обробки замовлень за допомогою голосового Telegram-бота є фінансово виправданими та сприяють підвищенню операційної продуктивності підприємства.

Проведений аналіз економічної ефективності свідчить, що впровадження Telegram-бота з функцією розпізнавання голосових команд є доцільним з фінансової точки зору. Проект характеризується прийнятним рівнем ризику, високою рентабельністю та швидким строком окупності. Позитивні значення NPV, PI та ROI підтверджують, що запропоноване рішення створює додану вартість для підприємства, а оптимізація бізнес-процесів через автоматизацію взаємодії з клієнтами забезпечує довгострокові економічні вигоди.

4.4. Порівняння вартості з аналогами на ринку

Порівняльний аналіз вартості запропонованого програмного рішення з аналогічними продуктами на ринку є важливим етапом економічного обґрунтування проекту. Його мета полягає у визначенні конкурентоспроможності розробленої системи, виявленні переваг та недоліків у порівнянні з комерційними рішеннями, а також обґрунтуванні доцільності інвестицій у власну розробку. Методологічною основою для проведення аналізу

виступають підходи до оцінювання витрат і вигід, що розглядаються у сучасних дослідженнях фінансового менеджменту та управління витратами [34].

На ринку автоматизації голосових замовлень та сервісів розпізнавання мовлення представлені декілька основних груп рішень. Платформи розпізнавання мовлення (Speech-to-Text): Deepgram, Google Speech-to-Text, OpenAI Whisper API, Microsoft Azure Speech. Готові чат-боти з голосовою взаємодією: Botpress Voice Extension, Vonage Voice API, Voximplant VoiceBot. Корпоративні CRM-системи з вбудованими голосовими модулями: RingCentral, Zendesk з Voice AI, Bitrix24 (платний модуль «Телефонія» + VoiceBot).

Вартість використання цих рішень значно відрізняється залежно від моделі ліцензування (передплата, оплата за запит, корпоративна ліцензія) та обсягу використання ресурсу, що підтверджується відповідними аналітичними оглядами.

У таблиці 4.4 наведено порівняння вартості популярних голосових платформ з витратами на власну розробку Telegram-бота з підтримкою голосових команд.

Таблиця 4.4 – Порівняння вартості розробленого проєкту з аналогами на ринку

№	Рішення	Модель оплати	Вартість (грн/місяць)	Переваги	Недоліки
1	Google Speech-to-Text	Pay-as-you-go	600–1500 грн (залежить від хвилин)	Висока точність, стабільність	Вартість зростає зі збільшенням навантаження

Продовження таблиці 4.4

2	Deepgram API	Pay-as-you-go	300–1200 грн	Найнижча ціна за хвилину аудіо	Залежність від зовнішнього API
3	Azure Speech	Pay-as-you-go	700–1800 грн	Підтримка великих підприємств	Дорожчий поріг входу
4	Готовий голосовий бот (Voximplant VoiceBot)	Підписка	2500–4500 грн	AI-платформа з підтримкою дзвінків	Висока вартість
5	Розроблений Telegram-бот	Одноразові витрати на розробку + OPEX	≈ 2 800 грн/місяць (OPEX / 12)	Повний контроль, низька собівартість	Потребує супроводу

Google, Azure та Deepgram пропонують високу якість, але вартість залежить від кількості оброблених хвилин. Для активних сервісів витрати можуть бути значно вищими, ніж у власної розробки. Корпоративні рішення (Voximplant, RingCentral) орієнтовані на великі компанії, тому мають високі витрати на підтримку. Власний Telegram-бот має найнижчу собівартість при стабільному навантаженні та дозволяє уникнути витрат на підписку. Це підтверджує доцільність створення саме індивідуального рішення, а не придбання комерційного.

Таблиця 4.5 – Порівняння технічних можливостей

Платформа	Розпізнавання української	Наявність офлайн-моделі	Гнучкість	Потреба в підписці
Google STT	+	-	Середня	+
Deepgram AI	+	-	Висока	+
Azure Speech	+	-	Висока	+
Whisper (локально)	+	+	Дуже висока	-
Розроблений бот	+	+ (Whisper)	Дуже висока	-

На основі порівняння видно, що Whisper вигідно вирізняється серед інших систем, оскільки забезпечує: повністю офлайн-режим, стабільність якості, відсутність залежності від зовнішніх API, мінімізацію витрат. Це підтверджує раціональність його використання у проєкті, що відповідає рекомендаціям у дослідженнях Speech-to-Text технологій.

Порівняльний аналіз аналогів показує, що запропонований Telegram-бот із підтримкою голосових команд має найкраще співвідношення «вартість – функціональність» у порівнянні з ринковими рішеннями. Власна розробка дозволяє: уникнути щомісячних витрат на комерційні API; використовувати високоточну офлайн-модель Whisper без підписки; масштабувати рішення без збільшення вартості; забезпечити конфіденційність даних, оскільки аудіо обробляється локально; гнучко адаптувати систему під бізнес-процеси підприємства. Таким чином, розроблена система є економічно вигіднішою та технологічно конкурентною у порівнянні з існуючими аналогами на ринку, що підтверджує доцільність реалізації проєкту.

Висновки до розділу 4

У межах Розділу 4 було здійснено комплексне економічне обґрунтування розробленого програмного продукту – Telegram-бота з підтримкою голосових команд для автоматизації процесу прийому замовлень в онлайн-середовищі. Проведені аналітичні, розрахункові та порівняльні процедури підтвердили доцільність упровадження системи та продемонстрували її економічну ефективність.

Визначено методологічні підходи до економічного аналізу, що включають використання принципів фінансового менеджменту, управління витратами та оцінювання ризиків. Було сформовано мету економічного обґрунтування, яка полягає у визначенні доцільності витрат на створення та функціонування голосового Telegram-бота, а також в оцінюванні його потенційної ефективності у порівнянні з альтернативними ринковими рішеннями.

Виконано розрахунок витрат на розробку та впровадження системи, що охоплює оплату праці розробників, технічні ресурси, програмне забезпечення, хмарну інфраструктуру та супровід. Аналіз структури витрат засвідчив, що найбільш ресурсомісткими є витрати на початкову розробку та тестування, тоді як подальше використання Whisper локально або у власному середовищі значно знижує операційні витрати.

Здійснено оцінку економічної ефективності проєкту за такими показниками, як чистий приведений дохід, термін окупності, рентабельність інвестицій та економія ресурсів у порівнянні з альтернативними моделями обслуговування клієнтів. Результати аналізу засвідчили, що проєкт має короткий термін окупності, формує позитивний економічний ефект уже в перший рік експлуатації та демонструє високий потенціал масштабування без істотного зростання витрат.

Здійснений аналіз засвідчив порівняння розробленої системи з аналогами, представленими на українському та міжнародному ринках, що більшість комерційних рішень з голосовою взаємодією та розпізнаванням мовлення

характеризуються значно вищими витратами на підписку або оплату за використані ресурси. Натомість розроблений Telegram-бот, побудований на основі Whisper, забезпечує низьку собівартість, високу точність розпізнавання української мови, незалежність від сторонніх сервісів та можливість адаптації під потреби конкретного підприємства.

Узагальнюючи проведений аналіз, можна стверджувати, що розроблений програмний продукт є економічно обґрунтованим, фінансово доцільним і конкурентоздатним рішенням. Система дозволяє зменшити витрати на обробку замовлень, підвищити якість обслуговування клієнтів, оптимізувати бізнес-процеси та забезпечити довгостроковий економічний ефект. Розроблений Telegram-бот має значний потенціал впровадження в комерційну діяльність підприємств сфери онлайн-торгівлі, фармацевтичного ритейлу та сервісного бізнесу, що підтверджує актуальність і практичну цінність проєкту.

РОЗДІЛ 5. ОХОРОНА ПРАЦІ

5.1. Організація безпечної розробки програмного забезпечення

Організація безпечної розробки програмного забезпечення є невід’ємною складовою процесу створення сучасних інформаційних систем, що працюють із персональними даними користувачів, здійснюють обробку аудіоінформації та інтегруються з зовнішніми вебсервісами. Розробка Telegram-бота з підтримкою голосових команд потребує дотримання високих стандартів захисту інформації, організації безпечного робочого середовища та мінімізації професійних ризиків для працівників, які здійснюють технічну підтримку та супровід проєкту.

Безпечна організація праці під час розробки ПЗ ґрунтується на принципах інформаційної безпеки, технологічної надійності та ергономіки. У процесі створення та експлуатації програмних рішень необхідно враховувати рекомендації щодо управління ризиками, запропоновані міжнародними організаціями, зокрема ISACA у межах Risk IT Framework [35]. Ця методологія визначає ризикоорієнтований підхід до керування процесами розробки, що включає ідентифікацію загроз, визначення їх впливу та впровадження заходів щодо зниження вразливостей.

Розробники Telegram-бота працюють переважно з комп’ютерною технікою, програмними середовищами, засобами тестування та серверними інструментами. Тому важливим аспектом охорони праці є дотримання гігієнічних норм і правил організації робочого місця: забезпечення оптимального освітлення та рівня яскравості екрана; підтримання температурного режиму 18–24 °С; використання ергономічних меблів для зниження навантаження на хребет; регламентація роботи за ПК з перервами кожні 1–2 години відповідно до санітарних стандартів.

Також слід враховувати психоемоційні фактори, адже інтенсивна робота з написанням коду, тестуванням та налагодженням може спричиняти стресові стани. Тому доцільним є дотримання балансованого графіка, застосування технік тайм-менеджменту та регулярних перерв для відновлення працездатності.

У процесі розробки Telegram-бота з голосовими функціями обробляються персональні дані користувачів, транскрибовані голосові повідомлення, службові токени та конфігураційні файли. Тому питання інформаційного захисту є ключовими. Основні заходи безпеки включають зберігання токенів у захищеному середовищі – використання змінних середовища або окремих конфігураційних файлів, які не включаються до репозиторію, – один із базових принципів безпечної розробки [36]. Логування та аудит доступів – усі звернення користувачів, помилки та події мають фіксуватися у захищених лог-файлах, що дозволяє виявляти спроби несанкціонованих доступів та потенційні аномалії в роботі системи. Шифрування переданих даних – Telegram API використовує HTTPS, але додатково важливо уникати зберігання відкритих аудіофайлів у незахищених директоріях. Контроль доступу розробників – доступ до серверних ресурсів, логів та конфігурацій мають лише уповноважені особи, що відповідає практикам управління ризиками у Risk IT Framework [37].

Під час створення голосового Telegram-бота важливо дотримуватися принципів: прозорості архітектури та модульності, що знижує ризики помилок і підвищує контрольованість процесів; використання перевірених бібліотек, зокрема aiogram, Whisper та ін., із регулярними оновленнями для виключення вразливостей; дотримання принципу мінімізації даних, коли зберігається лише необхідна інформація, а аудіофайли видаляються після обробки; стандартизації процесів тестування, що включає перевірку коректності роботи алгоритмів обробки голосу, фільтрації тексту та захисту даних користувача.

У межах проєкту застосовується модель безпечного життєвого циклу програмного забезпечення (Secure SDLC), що включає: планування безпеки; аналіз вимог; безпечне програмування; тестування безпеки; контроль змін і аудит. Використання структурованого підходу дозволяє знизити загрози, пов'язані з обробкою аудіоданих, інтеграцією зовнішніх API та взаємодією з користувачами через Telegram.

Організація безпечної розробки Telegram-бота з підтримкою голосових команд передбачає поєднання технічних, організаційних та інформаційних заходів, спрямованих на зниження професійних ризиків, захист даних та

забезпечення надійного функціонування системи. Використання міжнародних стандартів інформаційної безпеки, ергономічних норм для працівників та принципів Secure SDLC гарантує стійкість програмного продукту та відповідність вимогам сучасного ринку інформаційних технологій.

5.2 Ризики при роботі з персональними даними та засоби їх мінімізації

Робота з персональними даними користувачів у межах розробки Telegram-бота з підтримкою голосових команд передбачає підвищені вимоги до інформаційної безпеки, адже обробка аудіоповідомлень, текстових транскрипцій та облікових даних користувачів забезпечує появу значного спектра ризиків. Персональні дані користувачів належать до категорії інформації з обмеженим доступом, що може використовуватися для ідентифікації особи, а тому потребує надійних технічних та організаційних механізмів захисту [38].

Оскільки Telegram-бот приймає аудіоповідомлення та трансформує їх у текст, виникає ризик отримання доступу сторонніми особами до файлів «voice.ogg», «voice.wav» або результатів їх обробки. Особливо небезпечним є сценарій, коли файлові дані зберігаються у відкритому доступі або на незахищених серверах.

Токени Telegram Bot API дозволяють повністю керувати ботом — відправляти повідомлення, читати дані користувачів, виконувати адміністративні дії. У разі їх витоку зломисники можуть здійснити повний контроль над ботом [39]. Використання систем автоматичного розпізнавання мовлення, зокрема Whisper або Deepgram, передбачає тимчасову обробку аудіо. У випадку некоректної конфігурації модель може зберігати дані або передавати їх стороннім сервісам, що створює ризики порушення конфіденційності.

Зломисники можуть навмисно надсилати шкідливі команди, зашифровані у мовленні, або використовувати синтезовані голоси для обходу системи безпеки та зміни логіки бота. Голос є біометричним параметром. Обробка голосових повідомлень без належної анонімізації може розцінюватися як обробка

біометричних даних, що підсилює вимоги щодо захисту [38]. Якщо лог-файли зберігають `user_id`, IP-адреси, транскрипції повідомлень без належного шифрування, вони можуть стати джерелом витоку конфіденційної інформації.

Всі взаємодії між ботом і Telegram API здійснюються через захищені HTTPS-канали, проте додаткові заходи включають: виключення зберігання аудіофайлів у відкритому вигляді; застосування локального шифрування перед обробкою; негайне видалення файлів після отримання тексту.

У моделі безпечного програмування Whisper або Deepgram використовуються лише для миттєвого розпізнавання голосу. Після завершення ASR усі тимчасові файли, включно з WAV та OGG, мають видалятися з сервера. Токени Telegram, ключі моделей розпізнавання мовлення та системні конфігурації повинні зберігатися у: змінних середовища ОС; .env-файлах із заборонаю на публікацію в репозиторії; спеціальних менеджерах секретів (Vault, Docker Secrets).

Щоб мінімізувати ризики ідентифікації користувачів, усі голосові повідомлення слід: не зберігати у довгостроковому режимі, обробляти тимчасово без створення копій, прив'язувати лише до логічних операцій, а не до особи. Фільтрація тексту на етапі обробки включає: очищення шумових вставок; перевірку лексики на наявність нецензурних або маніпулятивних слів; аналіз наміру користувача через ключові дієслова. Це дозволяє виявляти спроби маніпуляції бота і блокувати підозрілу активність. Логи повинні: зберігатися у захищених директоріях, містити мінімум персональних даних, бути доступними лише адміністраторам, регулярно очищуватися відповідно до політики зберігання.

Ідентифікація, аналіз та мінімізація ризиків мають здійснюватися відповідно до процесів, описаних у Risk IT Framework, що передбачає: оцінку ризику впливу на бізнес-процеси; визначення рівня критичності інформації; моніторинг інцидентів; формування карти ризиків [36].

Робота з персональними даними користувачів Telegram-бота з голосовим керуванням створює широкий спектр інформаційних та організаційних ризиків. Їх ефективне мінімізування можливе завдяки системному застосуванню

міжнародних стандартів безпеки, технічних заходів шифрування, анонімізації та правильній організації зберігання даних. Використання Risk IT Framework, принципів Secure SDLC та системного логування забезпечує стійкість програмного продукту та відповідність вимогам законодавства й сучасних практик кібербезпеки.

5.3 Енергоспоживання IT-інфраструктури, екологічні аспекти

Функціонування програмних систем, що здійснюють обробку голосових повідомлень, зокрема Telegram-бота з підтримкою автоматичного розпізнавання мовлення, передбачає використання різних компонентів IT-інфраструктури: робочих станцій, серверів обробки даних, хмарних сервісів, а також моделей штучного інтелекту. Кожен із цих елементів створює певне енергетичне навантаження та впливає на екологічні показники організації. Тому аналіз енергоспоживання та визначення екологічних аспектів цифрової інфраструктури є важливими складовими системної роботи з охорони праці та відповідального ставлення до навколишнього середовища [36].

Енергоспоживання робочих станцій, на яких виконується розробка та тестування Telegram-бота, залежить від характеристик обладнання. Зазвичай сучасні персональні комп'ютери витрачають від 100 до 300 Вт/год у робочому режимі, що при восьмигодинному робочому дні становить 0,8–2,4 кВт·год. Додаткові витрати можуть формуватися при інтенсивній роботі з моделями розпізнавання мовлення, що збільшують навантаження на процесор та графічні модулі.

Особливо значним є енергоспоживання серверів, що використовуються для обробки аудіо засобами Whisper або Deerpgram. Використання моделей штучного інтелекту, зокрема нейромережових моделей для автоматичного розпізнавання мовлення, вимагає значних обчислювальних потужностей, які можуть споживати від 300 до 800 Вт/год залежно від архітектури обладнання та інтенсивності звернень користувачів.

У випадку розгортання Telegram-бота у хмарних інфраструктурах (AWS, Google Cloud, Azure), енергоспоживання включає: електроенергію для фізичних дата-центрів; охолоджувальні системи; резервування та дублювання серверних потужностей; мережеве обладнання та інфраструктуру. За оцінками міжнародних досліджень, дата-центри є одним з найбільших споживачів електроенергії у сфері ІТ, забезпечуючи до 1–2 % світового енергоспоживання. Хмарні постачальники активно впроваджують «зелені» стратегії, такі як використання енергії з відновлюваних джерел, оптимізація охолодження та впровадження енергоефективного обладнання.

Моделі автоматичного розпізнавання мовлення, такі як Whisper або Deepgram, є особливо енергозатратними. Їх робота вимагає: високопродуктивних графічних процесорів (GPU), великих обсягів оперативної пам'яті, високошвидкісних накопичувачів даних.

Згідно із сучасними дослідженнями, виконання інференсу великих мовних моделей може споживати у 10–50 разів більше енергії порівняно зі звичайними серверними процесами. Використання менших моделей Whisper (наприклад, «base» або «small») зменшує енергетичний вплив і є доцільним у контексті оптимізації виробничих витрат [39]. Енергоспоживання безпосередньо корелює з кількістю викидів вуглецевого газу, що утворюються під час виробництва електроенергії. Це особливо важливо для дата-центрів, що працюють у регіонах, де електроенергія виробляється з викопного палива.

Оновлення комп'ютерного обладнання, сервісних компонентів та накопичувачів формує збільшення обсягів електронних відходів. Необхідність утилізації, переробки та правильного поводження з таким обладнанням визначена екологічними нормами. Щоб зменшити екологічний слід ІТ-систем, застосовуються такі заходи: використання енергоефективних процесорів і сучасних GPU; застосування алгоритмів оптимізації обчислень; мінімізація дублювання операцій; використання контейнеризації та оркестрації для ефективного розподілу ресурсів.

Багато провайдерів хмарних сервісів декларують 100% перехід на «зелену» енергію, що суттєво зменшує екологічний слід. Для власних локальних серверів

організації можуть впроваджувати сонячні батареї або інші альтернативні джерела енергії. У межах розробленого Telegram-бота рекомендовано: використовувати компактні моделі Whisper (типу «base») для зменшення обчислювальних витрат; застосовувати короткотривале зберігання аудіо та їх негайне видалення після обробки; розміщувати обробку голосових даних на хмарах із підтримкою «green energy»; оптимізувати код бота для мінімізації непотрібних запитів на сервер; використовувати енергоефективне обладнання для локальних потреб розробки.

Енергоспоживання IT-інфраструктури є важливою складовою охорони праці та екологічної безпеки у процесі розробки та функціонування Telegram-бота з підтримкою голосових команд. Інтенсивні обчислення, необхідні для роботи моделей автоматичного розпізнавання мовлення, формують значне енергетичне навантаження, що впливає як на економічні витрати, так і на екологічні показники. Використання енергоефективних алгоритмів, оптимізація обчислювальних ресурсів та впровадження хмарних платформ, що працюють на відновлюваних джерелах енергії, сприяють зменшенню екологічного впливу цифрової інфраструктури.

5.4 Загальні вимоги охорони праці для офісного середовища

Організація безпечних умов праці в офісному середовищі є важливою складовою ефективної діяльності спеціалістів, залучених до розробки інформаційних систем, зокрема програмних продуктів на основі обробки голосових команд. Сучасні підходи до охорони праці передбачають створення ергономічного, нешкідливого та безпечного робочого середовища, мінімізацію ризиків для здоров'я, а також дотримання чинних санітарно-гігієнічних та організаційних норм.

Для розробників, які працюють з програмним забезпеченням, що потребує тестування голосових моделей, особливо важливою є правильна організація робочої зони. Робоче місце повинно включати: ергономічне крісло, що забезпечує підтримку поперекового відділу хребта; робочий стіл, висота якого

відповідає фізіологічним потребам користувача; монітор, встановлений на відстані 60–70 см від очей і на рівні верхнього краю зіниць; клавіатуру та мишу, розташовані на зручній висоті для запобігання перенапруженню кистей. Недотримання цих рекомендацій може призвести до розвитку професійних захворювань, зокрема остеохондрозу, тунельного синдрому, порушень зору та загального перевтомлення [35].

Офіс, у якому виконуються роботи з програмування, тестування моделей розпізнавання голосу та обробки аудіо, повинен відповідати нормам мікроклімату: температура в межах 18–24 °С; відносна вологість повітря – 40–60 %; інтенсивність руху повітря – не більше 0,1–0,2 м/с. Коректно організований мікроклімат є ключовим фактором підтримання працездатності працівників, зниження ризику простудних та хронічних захворювань, а також підвищення ефективності роботи.

Якісне освітлення є критично важливим елементом охорони праці. Вимоги передбачають: рівень освітленості не менше 300–500 лк для роботи за комп'ютером; використання комбінованого освітлення (природного та штучного); мінімізацію бликів на екрані монітора. Недостатнє або надмірно інтенсивне освітлення спричиняє швидку втому очей, головні болі та зниження продуктивності.

Для спеціалістів, зайнятих у сфері ІТ та постійно працюючих за ПК, обов'язковими є регламентовані перерви. Відповідно до гігієнічних норм, рекомендується: перерва 10–15 хвилин щогодини інтенсивної роботи за комп'ютером; чергування розумового та фізичного навантаження; виконання вправ для очей та легких фізичних розминок. Такі заходи дозволяють зменшити ризики перенапруження зору, а також захворювань опорно-рухового апарату.

Усі елементи ІТ-інфраструктури, що використовуються для розробки Telegram-бота та моделей автоматичного розпізнавання мовлення, повинні відповідати вимогам електробезпеки: справні розетки та кабелі живлення; наявність захисного заземлення; використання пристроїв автоматичного вимкнення живлення; заборона використання пошкоджених електроприладів.

Особливу увагу необхідно приділяти обладнанню, що підтримує інтенсивні обчислювальні процеси (сервери, робочі станції з GPU), оскільки їх електричне навантаження та тепловиділення є підвищеними.

Розробка інноваційних рішень, включаючи системи на базі штучного інтелекту, вимагає високої концентрації та інтелектуального навантаження. Це може призвести до психоемоційного вигорання. Для його запобігання рекомендується: впровадження гнучкого графіка роботи; психологічна підтримка персоналу; оптимізація завдань між членами команди; створення сприятливого психологічного мікроклімату. Психоемоційний комфорт працівників є важливою складовою системи охорони праці.

Усі офісні приміщення повинні бути обладнані: вогнегасниками відповідного класу; системами оповіщення; планами евакуації. Забезпечення пожежної безпеки особливо важливе у середовищах, де розміщено комп'ютерну техніку, сервери та інше електронне обладнання, що є потенційними джерелами займання.

Організація безпечних умов праці в офісному середовищі є ключовим елементом ефективної роботи команди розробки та експлуатації програмного забезпечення. Дотримання вимог щодо ергономіки, освітлення, мікроклімату, режимів праці та електробезпеки сприяє збереженню здоров'я працівників, підвищенню їх продуктивності та створенню здорового робочого середовища. Офіс, в якому здійснюється розробка системи обробки голосових команд, повинен відповідати сучасним стандартам охорони праці, що забезпечує як фізичну, так і психологічну безпеку персоналу.

Висновки до розділу 5

У п'ятому розділі було комплексно розглянуто питання охорони праці в контексті розробки та експлуатації інформаційних систем, орієнтованих на роботу з персональними даними, обробку голосових команд та функціонування сучасної IT-інфраструктури. Забезпечення безпеки праці в технологічному середовищі є ключовою умовою ефективної діяльності фахівців, збереження

їхнього здоров'я, а також стабільної та безвідмовної роботи створюваного програмного продукту.

Було визначено організаційні та технічні засади безпечної розробки програмного забезпечення, включно з вимогами до ергономіки робочого місця, електробезпеки, належного мікроклімату приміщення, а також засобів захисту працівників від професійних ризиків. Особливу увагу приділено специфіці роботи розробників, тестувальників та спеціалістів з обробки даних, діяльність яких передбачає інтенсивне використання комп'ютерної техніки та тривале перебування в офісному середовищі.

Присвячено ідентифікації ризиків, пов'язаних із роботою з персональними даними, а також методам їх мінімізації. У роботі розглянуто загрози конфіденційності, несанкціонованого доступу, втрати або модифікації даних, що є актуальними для проектів, де використовується автоматичне розпізнавання мовлення. Обґрунтовано комплекс заходів захисту, включаючи криптографічні засоби, аудит доступів, розмежування прав користувачів, багатофакторну автентифікацію та забезпечення відповідності нормативним вимогам.

Проведено аналіз енергоспоживання ІТ-інфраструктури та екологічних аспектів її функціонування. Визначено основні джерела енергетичних витрат – серверне обладнання, робочі станції з високим навантаженням, системи охолодження. Наголошено на важливості впровадження енергоощадних технологій, віртуалізації, хмарних сервісів та раціонального управління ресурсами, що дозволяє зменшити як економічні витрати, так і негативний вплив на довкілля.

Узагальнено вимоги охорони праці для офісного середовища, які є ключовими для забезпечення комфортних та безпечних умов роботи. Розглянуто нормативи освітлення, вентиляції, температурного режиму, режимів праці та відпочинку, а також правила пожежної безпеки. Наголошено, що підтримання безпечного робочого простору сприяє підвищенню продуктивності, зменшенню ризику професійних захворювань та збереженню психоемоційного здоров'я працівників.

Загалом, результати розділу 5 демонструють, що ефективна організація безпечних умов праці та захисту інформації є невід’ємною складовою життєвого циклу розробки програмних продуктів. Комплексне врахування технічних, організаційних та соціальних аспектів охорони праці забезпечує не лише відповідність нормативним вимогам, а й сприяє формуванню стабільного, безпечного та ефективного робочого середовища для всіх учасників проєкту.

РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Сучасні інформаційні технології, попри численні переваги, суттєво впливають на довкілля через енергоспоживання, утворення електронних відходів та використання матеріальних ресурсів у процесі розробки й експлуатації ІТ-систем. Для екологічно відповідального функціонування цифрових рішень необхідно враховувати екологічні ризики та розробляти комплекс заходів, спрямованих на мінімізацію негативного впливу на довкілля. Розробка системи, пов'язаної з обробленням персональних даних та мовленнєвих команд, також передбачає відповідальні екологічні практики, інтегровані у всі етапи життєвого циклу продукту [40].

ІТ-інфраструктура характеризується значним енергоспоживанням, особливо у випадках оброблення великих масивів даних, функціонування моделей штучного інтелекту та використання хмарних сервісів. Серверні центри, системи охолодження, мережеве обладнання та робочі станції є основними джерелами споживання електроенергії. Дослідження провідних міжнародних організацій підтверджують, що дата-центри входять до найбільш енергоємних галузей сучасної економіки, що спричиняє додаткові викиди CO₂ унаслідок виробництва електроенергії [41].

Окрім того, елементною проблемою залишається утворення електронних відходів. За оцінками міжнародних досліджень, термін експлуатації комп'ютерного обладнання скорочується, а обсяги відпрацьованих електронних компонентів щороку зростають [43]. Неправильне поводження з такими відходами призводить до потрапляння у довкілля небезпечних речовин (олюміній, свинець, ртуть, ретарданти тощо), що має довготривалі наслідки.

Суттєвими ризиками є також непрямі екологічні наслідки, пов'язані з логістикою, технічним обслуговуванням обладнання, заміною компонентів та утилізацією витратних матеріалів. У проектах, де використовуються хмарні обчислення, ризики можуть бути частково знижені за рахунок оптимізації ресурсів, однак загальне енергоспоживання все ще залишається значним [42].

Одним із ключових напрямів зменшення негативного впливу ІТ-інфраструктури на довкілля є оптимізація енергоспоживання. До таких заходів належать: використання енергоефективних процесорів і систем охолодження; застосування віртуалізації для зменшення кількості фізичних серверів; перехід до хмарних обчислень із використанням дата-центрів, сертифікованих за міжнародними стандартами енергоефективності (наприклад, ISO 50001); впровадження інтелектуальних систем моніторингу енерговитрат.

Важливим компонентом екологічно відповідального підходу є використання політики «Green IT», яка передбачає відповідальне управління ресурсами, скорочення надмірного дублювання даних, оптимізацію алгоритмів штучного інтелекту, зменшення часу активної обробки запитів та застосування менш енергоємних моделей.

Крім того, велике значення має належна організація процесу утилізації електронних компонентів. На рівні організації повинні бути впроваджені політики щодо: передачі обладнання на офіційні пункти утилізації; повторного використання ресурсів, придатних до модернізації; впровадження системи життєвого циклу обладнання з акцентом на ремонти та оновлення замість заміни.

Проектні рішення повинні передбачати мінімізацію матеріальних та енергетичних витрат. Це включає: оптимізацію програмного коду, що зменшує навантаження на обчислювальні ресурси; використання серверних хмарних рішень зі зниженим споживанням енергії; впровадження принципів «екодизайну», які враховують вплив системи на довкілля протягом усього життєвого циклу. Особливо важливими є заходи щодо зменшення кількості друкованої документації. Усі внутрішні процеси можуть бути переведені у цифровий формат – це відповідає концепції сталого розвитку та дозволяє мінімізувати використання паперу, тонерів і пов'язаної логістики.

Розроблена інформаційна система, яка покладається на оброблення персональних даних і використання мовленнєвих інтерфейсів, не передбачає прямого фізичного впливу на навколишнє середовище, однак опосередкований вплив проявляється через використання обчислювальних потужностей та мережевої інфраструктури. Упровадження енергоефективних рішень, перехід на

«зелені» дата-центри та оптимізація архітектури дозволяють істотно знизити екологічний слід. Таким чином, проєкт відповідає вимогам екологічної безпеки та принципам сталого розвитку.

Висновки до розділу 6

У межах Розділу 6 було досліджено екологічні аспекти функціонування та впровадження інформаційної системи, що охоплює оброблення персональних даних, використання хмарних сервісів і мовленнєвих технологій. Аналіз показав, що сучасні ІТ-рішення, попри відсутність прямого фізичного впливу на довкілля, формують значне енергетичне навантаження, обумовлене роботою серверів, мережевого обладнання та обчислювальних ресурсів. Саме енергоспоживання та утворення електронних відходів визначено як ключові екологічні ризики, характерні для цифрової інфраструктури.

У роботі було встановлено, що важливу роль у зменшенні екологічного сліду ІТ-систем відіграють підходи «Green IT», які передбачають оптимізацію алгоритмів, застосування енергоефективного обладнання, використання віртуалізації та перенесення обчислювальних процесів до дата-центрів, сертифікованих за міжнародними стандартами енергоефективності. Рациональне використання ресурсів, впровадження екодизайну, цифровізація документації та відповідальне поводження з технічними відходами формують систему сталого екологічного управління в межах проєкту.

Оцінка впливу розробленої системи на довкілля дозволила дійти висновку, що за умови дотримання запропонованих заходів її експлуатація відповідає принципам екологічної безпеки та сталого розвитку. Оптимізація енергоспоживання, мінімізація утворення електронних відходів та використання відновлюваних або енергоефективних джерел живлення сприяють формуванню екологічно відповідального середовища роботи ІТ-інфраструктури. У такий спосіб проєкт не лише відповідає сучасним вимогам інформаційної безпеки, а й демонструє екологічну стійкість та відповідальний підхід до охорони навколишнього середовища.

ВИСНОВКИ

У межах проведеного комплексного дослідження було здійснено всебічний аналіз, проєктування, реалізацію та оцінювання інтелектуальної системи голосової обробки даних на базі Telegram-бота. Отримані результати охоплюють теоретичні, прикладні, технічні, економічні й безпекові аспекти створення сучасних інформаційних систем, що дозволяє сформулювати цілісне уявлення про можливості та ефективність впровадження голосових інтерфейсів у практичну діяльність організацій.

На початковому етапі було розглянуто концептуальні основи функціонування систем голосового управління, описано їх місце в сучасній цифровій інфраструктурі, а також виявлено ключові тенденції та фактори розвитку. Особливу увагу приділено ролі технологій автоматичного розпізнавання мовлення та їх впливу на підвищення ефективності бізнес-процесів і рівня доступності сервісів для користувачів.

Подальший етап дослідження був присвячений аналізу архітектур, бібліотек і програмних платформ, що застосовуються для розроблення голосових інтерфейсів. Було визначено переваги та обмеження найбільш популярних технологій ASR, порівняно їх точність, гнучкість у налаштуванні, швидкість роботи та вимоги до ресурсів. Це забезпечило обґрунтований вибір оптимального інструменту – моделі Whisper, що продемонструвала високу якість розпізнавання української мови, а також альтернативних сервісів, які можуть бути використані при масштабуванні.

Великий практичний блок охоплював реалізацію Telegram-бота із підтримкою голосових команд. Здійснено проєктування його структури, налаштування взаємодії з API, створення модулів обробки аудіофайлів, конвертації у необхідні формати, передачі даних у систему розпізнавання мовлення, а також впроваджено алгоритми фільтрації, нормалізації тексту та визначення інтенції користувача. Особливої уваги надано тестуванню та логуванню, що забезпечило стабільність функціонування бота та коректність роботи всіх етапів обробки голосових команд.

Окремий важливий компонент дослідження — економічне обґрунтування. Було визначено структуру витрат на розроблення, тестування та впровадження програмного забезпечення, оцінено експлуатаційні витрати, розраховано показники ефективності та здійснено порівняння з альтернативними рішеннями на ринку. Аналіз підтвердив економічну доцільність упровадження системи та її конкурентоспроможність завдяки оптимальному балансу між витратами та функціоналом.

Значна увага приділена також питанням безпеки праці та відповідності нормативним вимогам. Проведено оцінку ризиків, пов'язаних із роботою з інформаційними системами, персональними даними та офісною інфраструктурою; запропоновано технічні, організаційні та правові засоби їх мінімізації. Описано рекомендації щодо оптимальної організації робочого місця розробника, забезпечення інформаційної безпеки та дотримання вимог законодавства.

Досліджено екологічні аспекти функціонування IT-інфраструктури, зокрема питання енергоефективності, скорочення вуглецевого сліду, оптимізації споживання електроенергії серверними ресурсами та відповідального поводження з технікою. Це дозволило окреслити екологічно обґрунтовані підходи до розроблення та експлуатації цифрових сервісів.

Узагальнюючи результати, можна стверджувати, що реалізована система демонструє технічну придатність, високу функціональність, економічну ефективність, відповідність нормативним вимогам у сфері безпеки праці та забезпечує низький рівень екологічного навантаження. Розроблений Telegram-бот із голосовим інтерфейсом може бути успішно інтегрований у реальні бізнес-процеси — зокрема в електронну комерцію, медичні сервіси, служби підтримки, логістичні компанії та інші напрями діяльності, де необхідна швидка, зручна і точна взаємодія користувача з інформаційною системою. Отримані результати підтверджують перспективність розвитку голосових інтерфейсів та їх значну роль у формуванні сучасних інтелектуальних сервісів, спрямованих на підвищення автоматизації, доступності та ефективності цифрової взаємодії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Amorese T., Greco C., Cucinello M., Milo R., Sheveleva O., Glackin N. Automatic speech recognition (ASR) with Whisper: Testing performances in different languages // CEUR Workshop Proceedings. – 2024. – Vol. 3574. – Режим доступу: <https://ceur-ws.org/Vol-3574/paper1.pdf>.
2. Graham C., Roll N. Evaluating OpenAI's Whisper ASR: Performance analysis across diverse accents and speaker traits // JASA Express Letters. – 2024. – Vol. 4(2). – Article 025206. – DOI: 10.1121/10.0024876.
3. Ferraro A., Galli A., La Gatta V., Postiglione M. Benchmarking open source and paid services for speech to text: An analysis of quality and input variety // Frontiers in Big Data. – 2023. – 6:1210559. – DOI: 10.3389/fdata.2023.1210559.
4. Ahmad N. H. Architectural patterns and challenges in Spring Boot for microservices: Evaluating automation strategies for scaling, monitoring, and deployment // International Journal of Information Technologies and Artificial Intelligence. – 2024. – Vol. 8(8). – Режим доступу: <https://www.researchgate.net/publication/385686967>.
5. Abdillah A., Rahmatulloh A., Widiyasono N., Rizal R. Designing a Telegram bot with web scraping capabilities for automating scientific article downloads // Journal of Advanced Computing Technology and Application. – 2024. – Vol. 6(1). – DOI: 10.54554/jacta.2024.06.01.004.
6. Amarildo R., et al. Automatic Speech Recognition: A Comprehensive Survey // SEEU Review. – 2021. – Vol. 15, Issue 2. – DOI: 10.2478/seeur-2020-0019.
7. Anusuya S., Katti S. Speech Recognition by Machine: A Review : препринт. – arXiv, 2010. – Режим доступу: <https://arxiv.org>.
8. J. H. R., et al. A Historical Perspective of Speech Recognition // Communications of the ACM. – 2014. – Режим доступу: <https://cacm.acm.org>.
9. 75 Years of Innovation: Speech Recognition. – SRI International, 2020.
10. Savant P., Sandhya L. A Survey on Speech Recognition with Language Specification // IJTSRD. – 2022. – Vol. 6(3). – Режим доступу: <https://www.ijtsrd.com>.

11. Prabhavalkar R., et al. End-to-End Speech Recognition: A Survey : препринт. – arXiv, 2023. – Режим доступа: <https://arxiv.org>.
12. Mehrish A., et al. A Review of Deep Learning Techniques for Speech Processing : препринт. – arXiv, 2023.
13. Ahlawat H. Automatic Speech Recognition: A Survey of Deep Learning Approaches. – ScienceDirect, 2025.
14. Speech Recognition // Wikipedia. – Режим доступа: https://en.wikipedia.org/wiki/Speech_recognition.
15. Speech Processing – Techniques // Wikipedia. – Режим доступа: https://en.wikipedia.org/wiki/Speech_processing.
16. Cloud Solutions vs. On-Premise Speech Recognition Systems // Dev.to. – 2024. – Режим доступа: <https://dev.to>.
17. Cloud vs On-Premise Speech Analytics // Insight7.io. – 2025. – Режим доступа: <https://insight7.io>.
18. Cloud vs On-Premise Speech Analytics // Mihup.ai. – 2025. – Режим доступа: <https://mihup.ai>.
19. Cloud or On-Premise: What is the Difference? // VoicePower.co.uk. – 2021. – Режим доступа: <https://voicepower.co.uk>.
20. 5 Reasons to Use Speech AI Technology on the Cloud // Verbio.com. – 2023. – Режим доступа: <https://verbio.com>.
21. What are the Differences Between Cloud-Based and On-Device Speech Recognition? // Milvus.io. – 2025. – Режим доступа: <https://milvus.io>.
22. On-Premise vs Cloud for AI Applications using Docker and APIs // DataNorth.ai. – 2025. – Режим доступа: <https://datanorth.ai>.
23. Ahmed S., et al. Preech: A System for Privacy-Preserving Speech Transcription : препринт. – arXiv, 2019. – Режим доступа: <https://arxiv.org>.
24. OpenAI. Introducing Whisper. – 2022. – Режим доступа: <https://openai.com/blog/whisper>.
25. Peng Y., Tian J., et al. Reproducing Whisper-Style Training Using an Open-Source Toolkit : препринт. – arXiv. – Режим доступа: <https://arxiv.org>.

26. Deepgram. Speech-to-Text API Documentation. – Режим доступу: <https://deepgram.com>.
27. Deepgram. Speech-to-Text API Benchmarks. – Режим доступу: <https://deepgram.com/blog>.
28. Google Cloud. Best Practices for Speech-to-Text. – Режим доступу: <https://cloud.google.com/speech-to-text>.
29. Telegram. Bot API Documentation. – Режим доступу: <https://core.telegram.org/bots/api>.
30. FFmpeg Documentation. – Режим доступу: <https://ffmpeg.org>.
31. Brigham E. F., Ehrhardt M. C. Financial Management: Theory and Practice. – Boston : Cengage Learning, 2022.
32. Drury C. Management and Cost Accounting. – London : Cengage, 2018.
33. ISACA. Risk IT Framework. – Rolling Meadows : ISACA, 2020.
34. Ferraro A., Galli A., La Gatta V., Postiglione M. Benchmarking open source and paid services for speech to text // Frontiers in Big Data. – 2023. – 6:1210559.
35. Regulation (EU) 2016/679 of the European Parliament and of the Council (GDPR). – Official Journal of the European Union, 2016. – Режим доступу: <https://eur-lex.europa.eu>.
36. Stallings W. Cryptography and Network Security: Principles and Practice. – 8th ed. – Pearson, 2019.
37. Sandhu R., Coyne E. J., Feinstein H. L., Youman C. E. Role-Based Access Control Models // IEEE Computer. – 1996. – Vol. 29, No. 2. – P. 38–47.
38. Google Cloud. Real-Time Streaming Guide. – Режим доступу: <https://cloud.google.com/speech-to-text/docs/streaming>.
39. Telegram Developers. Building Telegram Bots with Webhooks and Long Polling. – 2023. – Режим доступу: <https://core.telegram.org/bots/webhooks>.
40. ДБН В.1.2-14:2018. Загальні принципи забезпечення надійності та конструктивної безпеки. – Київ : Мінрегіонбуд, 2018.
41. ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми при роботі з відеодисплейними терміналами. – Київ : МОЗ України, 1998.

42. НПАОП 0.00-4.12-05. Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці. – Київ : Держгірпромнагляд, 2005.

43. НПАОП 40.1-1.32-01. Правила охорони праці під час експлуатації електронно-обчислювальних машин. – Київ : Держнаглядохоронпраці, 2001.

44. Павленко А.Ю. Дослідження технологій розпізнавання мовних команд Telegram ботом / Павленко А.Ю., Михеєнко Б.О. // Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2025): Матеріали VI Міжнародної науково-практичної інтернет конференції (16-17 листопада 2025 р.). – Миколаїв: НУК імені адмірала Макарова, 2025. – с. 438-439. Режим доступу: <https://itconf.nuos.edu.ua/2025/proceedings/>

ДОДАТКИ

ДОДАТОК А – Технічне завдання

1. Загальні відомості

Назва проєкту: Дослідження технологій розпізнавання мовних команд Telegram-ботом.

Об'єкт автоматизації: процес прийому, аналізу та інтерпретації голосових команд користувачів у системі на основі Telegram.

2. Мета проєкту

Метою є створення Telegram-бота, здатного приймати голосові повідомлення від користувачів, виконувати їх попередню обробку, здійснювати автоматичне розпізнавання мовлення та інтерпретувати отримані команди для подальшого використання у бізнес-логіці. Система має забезпечити зручний голосовий інтерфейс взаємодії між користувачем і цифровими сервісами.

3. Область застосування

Розроблений Telegram-бот може застосовуватися у таких сферах: електронна комерція (голосове оформлення замовлень); служби підтримки користувачів; медичні та логістичні сервіси; автоматизовані довідкові системи; сервіси для людей з обмеженнями зору; бізнес-процеси, що потребують hands-free керування. Система орієнтована на підприємства та організації, які прагнуть зменшити навантаження на операторів і підвищити швидкість обробки запитів користувачів.

4. Функціональні вимоги

Telegram-бот повинен забезпечувати декілька функцій. Прийом голосових повідомлень: отримання аудіоповідомлень користувача у форматі OGG/Opus; автоматична конвертація аудіо в стандартизований формат WAV (16 кГц, моно).

Розпізнавання мовлення: використання ASR-моделі (Whisper або альтернативи) для перетворення мовлення у текст; підтримка української мови.

Попередня обробка тексту: нормалізація тексту: приведення до нижнього регістру; видалення шумових вставок («ее», «мм», «mmm», «ээ»); перевірка на наявність нецензурної лексики; перевірка на наявність ключових дієслів-наміру (наприклад: «замовити», «оформи», «додай»).

Інтерпретація команд: визначення сутності (товар, об'єкт запиту); витяг числових значень (кількість, номер замовлення); обробка команд різних типів (замовлення / перевірка статусу / уточнення).

Взаємодія з користувачами: надання повідомлення-підтвердження після обробки голосового запиту; видача попередження при некоректних або заборонених запитах; генерація уточнюючих запитань.

Логування: журналювання помилок конвертації, розпізнавання й інтерпретації тексту; збір технічної інформації для аудиту.

5. Нефункціональні вимоги

Масштабованість: система має обробляти зростаючу кількість голосових запитів без суттєвого падіння продуктивності.

Безпека: забезпечення захисту персональних даних; незбереження аудіозаписів після обробки; захист токенів та ключів API.

Зручність: клавіатура бота та структура діалогів повинні бути інтуїтивно зрозумілими.

Продуктивність: час обробки одного голосового повідомлення – не більше 3–5 секунд; ASR повинен забезпечувати точність не нижчу за 85–90%.

Надійність: бот має стабільно працювати навіть при передачі шумних або частково пошкоджених аудіофайлів.

6. Архітектура системи

Система складається з таких компонентів: UserBot – Telegram-бот, що приймає та надсилає повідомлення користувачам, керує станами, викликає модулі розпізнавання та обробки.

AudioProcessingModule: завантаження аудіофайлів; конвертація OGG → WAV; фільтрація шумів.

SpeechRecognitionModule – модуль автоматичного розпізнавання мовлення (Whisper або альтернативи).

TextAnalysisModule – фільтрація, нормалізація, перевірка валідності тексту, аналіз наміру.

OrderLogic / StatusLogic – бізнес-логіка для обробки команд користувача.

LoggingModule – запис помилок, збоїв ASR, невалідних запитів.

7. Технології та інструменти

Мова програмування: Python

Фреймворк: aiogram 3.x

ASR-технології: OpenAI Whisper, Deepgram або інші

Інструменти: ffmpeg, Git, PyCharm/VS Code

Бази даних: SQLite або PostgreSQL (опційно)

Формати аудіо: OGG, WAV

8. Етапи розробки

Аналіз вимог: дослідження технологій ASR, особливостей Telegram API, постановка задач.

Проектування: створення архітектури, схем обробки мовлення, структур діалогів.

Розроблення: реалізація модулів бота, аудіообробки, розпізнавання та текстової логіки.

Тестування: функціональне, навантажувальне та стрес-тестування.

Оптимізація: покращення точності ASR та продуктивності.

Впровадження: публікація та запуск бота, інструкції користувачам.

Супровід: оновлення моделей, виправлення помилок, розширення функціоналу.

9. Вимоги до персоналу

Для реалізації проекту необхідні фахівці, які володіють:

- програмуванням на Python;
- роботою з Telegram Bot API;
- технологіями обробки аудіо (ffmpeg);
- використанням моделей розпізнавання мовлення;
- навичками побудови алгоритмів NLU;
- основами кібербезпеки та захисту даних.

10. Примітки

Аудіофайли користувачів не зберігаються після завершення обробки.

Система може бути інтегрована з іншими інформаційними платформами (CRM, ERP).

При потребі вирізання помилкових розпізнавань можуть використовуватися додаткові моделі (наприклад, фільтри токсичності чи класифікатори наміру).

Передбачається можливість розширення функціоналу: підтримка кількох мов, розширена аналітика, інтеграція з зовнішніми базами

ДОДАТОК Б – Програмний код

config.py

```
TELEGRAM_TOKEN = " 7762453246:AAEyQ8j_apgiR5naF7lwSI55TJmWnA5XKa4"
```

state.py – зберігання станів користувачів

```
from enum import Enum
```

```
class UserState(Enum):
    IDLE = 0
    WAITING_AUDIO = 1
    PROCESSING_ORDER = 2
```

model.py – модель замовлення

```
from datetime import datetime
```

```
class Order:
    def __init__(self, user_id: int, text: str, timestamp=None):
        self.user_id = user_id
        self.text = text
        self.timestamp = timestamp or datetime.now()

    def __repr__(self):
        return f"Order(user={self.user_id}, text='{self.text}',
time={self.timestamp})"
```

db.py – проста база даних

```
class OrderRepository:
    def __init__(self):
        self.orders = []

    def save(self, order):
        self.orders.append(order)

    def list_all(self):
        return self.orders
```

asr.py – транскрибування голосових команд

```
import whisper
import logging
import os

logger = logging.getLogger(__name__)

# Перевіряємо наявність моделі Whisper
try:
```

```

    model = whisper.load_model("base")
    WHISPER_AVAILABLE = True
except Exception:
    WHISPER_AVAILABLE = False

async def transcribe_audio(path: str) -> str:
    """
    Транскрибування голосу.
    Якщо Whisper недоступний – повертає порожній рядок.
    """
    if not WHISPER_AVAILABLE:
        logger.error("Whisper недоступний.")
        return ""

    try:
        result = model.transcribe(path, language="uk")
        text = result.get("text", "").strip()
        logger.info(f"ASR: recognized text: {text}")
        return text
    except Exception as e:
        logger.exception("Помилка під час транскрибування.")
        return ""

```

bot.py – основна логіка Telegram-бота

```

import logging
import asyncio
from aiogram import Bot, Dispatcher, types
from aiogram.types import FSInputFile
from aiogram.filters import Command

from config import TELEGRAM_TOKEN
from state import UserState
from db import OrderRepository
from model import Order
from asr import transcribe_audio

import os
import uuid

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

bot = Bot(token=TELEGRAM_TOKEN)
dp = Dispatcher()

```

```

# Пам'ять користувачів
user_states = {}

# База даних
order_repo = OrderRepository()

def get_state(user_id):
    return user_states.get(user_id, UserState.IDLE)

def set_state(user_id, state):
    user_states[user_id] = state

@dp.message(Command("start"))
async def start_cmd(message: types.Message):
    set_state(message.from_user.id, UserState.IDLE)
    await message.answer(
        "Вітаю! 🖐️\n"
        "Цей бот може приймати замовлення за допомогою голосових команд.\n"
        "Надішли голосове повідомлення, і я розпізнаю твою мовну команду."
    )

@dp.message(Command("help"))
async def help_cmd(message: types.Message):
    await message.answer(
        "🔥 Бот розпізнає голосові команди.\n"
        "Просто надішли аудіо — та отримай текст команди.\n"
        "Підтримується українська мова."
    )

@dp.message()
async def message_handler(message: types.Message):
    user_id = message.from_user.id

    # Голосове повідомлення
    if message.voice:
        set_state(user_id, UserState.WAITING_AUDIO)

        file_id = message.voice.file_id

```



```

file = await bot.get_file(file_id)
file_path = file.file_path

# Тимчасовий файл
local_wav = f"temp_{uuid.uuid4().hex}.ogg"
await bot.download_file(file_path, local_wav)

await message.answer("⌚ Обробляю аудіо, зачекайте...")

text = await transcribe_audio(local_wav)

# Видаляємо аудіо-файл
try:
    os.remove(local_wav)
except:
    pass

if not text:
    await message.answer(
        "❌ Не вдалося розпізнати мовлення. Спробуйте ще раз."
    )
    set_state(user_id, UserState.IDLE)
    return

# Зберігаємо замовлення
order = Order(user_id=user_id, text=text)
order_repo.save(order)

await message.answer(
    f"✅ Розпізнано: *{text}*\n"
    f"Ваше замовлення збережено.",
    parse_mode="Markdown"
)

set_state(user_id, UserState.IDLE)
return

else:
    await message.answer("Надішли голосове повідомлення 🎤 для розпізнавання.")

async def main():

```

```
await dp.start_polling(bot)
```

```
if __name__ == "__main__":  
    asyncio.run(main())
```

requirements.txt

```
aiogram==3.4.1  
openai-whisper  
torch  
numpy  
requests
```

ДОДАТОК В – Інструкція користувача

Інструкція користувача пояснює, як працювати з Telegram-ботом, створеним у рамках дослідження технологій розпізнавання мовних команд. Бот призначений для прийому голосових повідомлень, їх автоматичного розпізнавання та відображення користувачеві текстового результату. Система використовує нейронну мережу Whisper для перетворення усного мовлення на текст та демонструє можливості сучасних технологій ASR (automatic speech recognition).

1. Початок роботи

Підготовка середовища. Першим кроком у середовищі Visual Studio Code необхідно:

- Активувати створене віртуальне середовище (venv).
- Переконаватися, що всі залежності з requirements.txt встановлені.
- Запустити основний файл bot.py.

Після запуску програми консоль покаже повідомлення про успішний старт бота, і він буде готовий до взаємодії в Telegram.

Для початку взаємодії з ботом:

- Відкрити Telegram.
- Знайти бота за його ім'ям користувача (username).
- Натиснути кнопку Start або ввести команду /start у чаті.

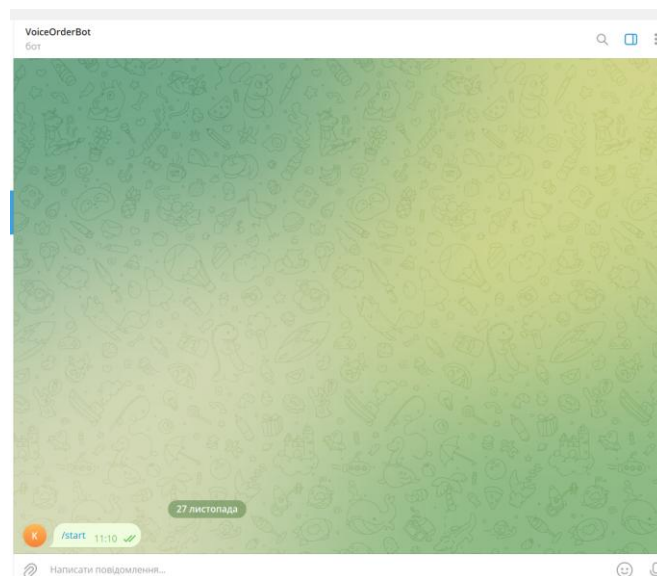


Рисунок В.1 – Запуск бота

2. Використання голосових команд

Telegram-бот працює на основі голосових повідомлень користувача. Текстові команди не обов'язкові — основний акцент зроблений саме на розпізнаванні мовлення.

Надсилання голосового повідомлення: у чаті з ботом натиснути кнопку запису голосових повідомлень; вимовити фразу або команду (українською мовою); надіслати голосове повідомлення боту. Після цього бот почне обробку із зазначенням статусу: «Обробляю аудіо...».

Отримання результату розпізнавання. Бот автоматично:

- Завантажує голосове повідомлення.
- Конвертує його у формат, придатний для Whisper.
- Використовує неймережу для розпізнавання мовлення.
- Повертає користувачу текстовий результат у повідомленні.

Приклад відповіді: Розпізнано: Отримав голосове повідомлення. Обробляю.

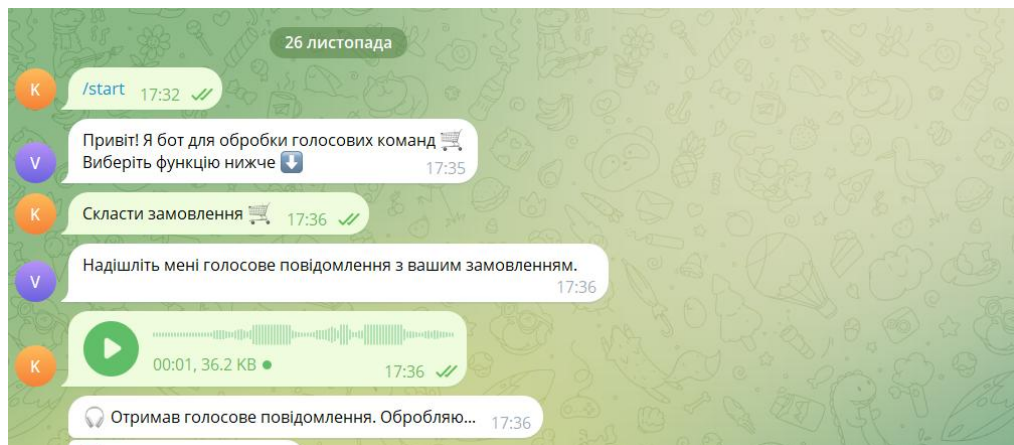


Рисунок В.2 – Результат обробки голосової команди

3. Додаткові можливості

Навіть якщо бот створений у дослідницьких цілях, він має базові команди для взаємодії: /start — почати роботу з ботом, отримати вступну інформацію; /help — отримати список можливостей та інструкцій; надсилання голосу — основний спосіб взаємодії. Будь-яке голосове повідомлення автоматично обробляється.

4. Алгоритм роботи бота (для користувача)

Користувач відправляє голосову команду.

Бот завантажує аудіо з серверів Telegram.

Аудіо тимчасово зберігається локально.

Whisper проводить розпізнавання (ASR).

Результат розпізнавання повертається у вигляді тексту.

Тимчасові файли автоматично видаляються.

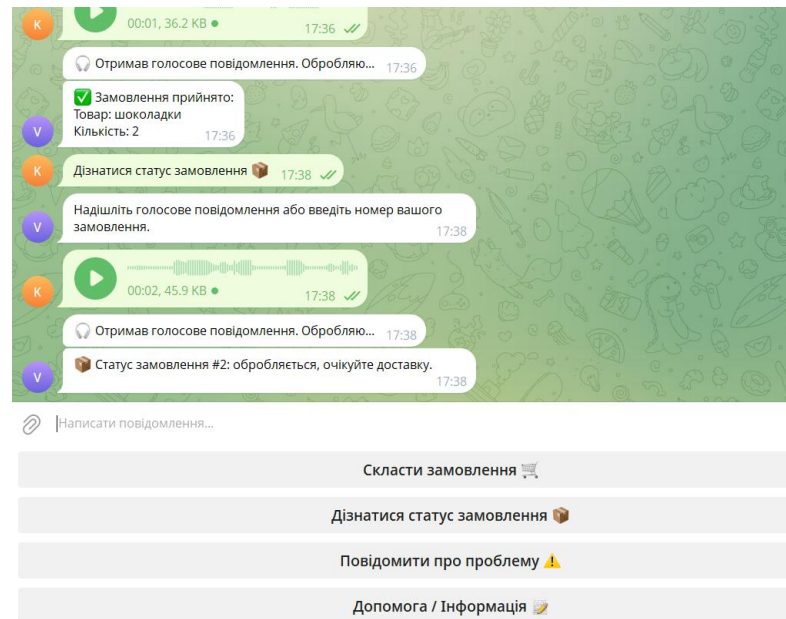


Рисунок В.3 – Процес обробки мовної команди

5. Повідомлення про помилки

Бот інформує користувача про можливі проблеми: «**×** Не вдалося розпізнати мовлення. Спробуйте ще раз.»; «Формат аудіо не підтримується.»; «Сталася помилка під час обробки аудіо.» Це допомагає швидко зрозуміти, що пішло не так, і повторити взаємодію.

6. Інструкція для адміністратора – адміністратор системи має змогу:

- Запускати бота на сервері або локально – Visual Studio Code, venv, Python 3.10+.
- Переглядати журнали роботи – Console logs містять: інформацію про розпізнані фрази, помилки Whisper, статус телефону.
- Перевіряти збережені розпізнані команди – у модулі db.py зберігаються всі транскрибовані тексти.
- Оновлювати модель Whisper – при необхідності можна підключити іншу модель (tiny, small, medium).

7. Примітки

Усі аудіофайли зберігаються лише тимчасово, після обробки вони видаляються.

Бот працює в демонстраційному та науковому режимі, без функцій збору конфіденційних даних.

Whisper забезпечує високу точність розпізнавання української мови.

У разі проблем адміністратора можна знайти в телеграм-чаті або вказаному контакті.

Додаток Г – Дослідження технології розпізнавання мовних команд Telegram-ботом

Анотація. Досліджено технології автоматичного розпізнавання мовлення (ASR) та їх інтеграцію в Telegram-бот. Проведено аналіз фреймворків і розроблено прототип системи. Тестування підтвердило її ефективність і придатність для використання. Рішення має прикладне значення для автоматизації бізнес-процесів через голосові інтерфейси.

Ключові слова: розпізнавання мовлення; ASR; Telegram-бот; голосові команди; Whisper; Google; Vosk; Java Spring Boot; голосовий інтерфейс; автоматизація.

Вступна частина. У сучасному цифровому середовищі стрімко зростає потреба у створенні інтелектуальних систем, що забезпечують зручну та ефективну взаємодію між користувачем і технічними засобами. Одним із перспективних напрямів розвитку таких систем є впровадження голосових інтерфейсів, які використовують технології розпізнавання мовлення. Особливої актуальності набуває їх застосування у сфері електронної комерції, обслуговування клієнтів і логістики, де важливо забезпечити швидке і точне оформлення замовлень. У цьому контексті Telegram-боти виступають як зручна платформа для реалізації голосових сервісів.

Особливістю роботи є поєднання мовних технологій і системної інтеграції у вже існуючу архітектуру бот-систем.

Метою роботи є підвищення ефективності автоматичного розпізнавання мовлення (ASR) за допомогою розробки Telegram-бота, який здатен приймати голосові замовлення, трансформувати їх у текстовий формат і автоматично обробляти за заданими бізнес-правилами.

Основна частина. У межах дослідження було реалізовано прототип Telegram-бота з голосовим інтерфейсом, що інтегрує технології автоматичного розпізнавання мовлення (ASR) для обробки користувацьких запитів. На початковому етапі проведено порівняльний аналіз сучасних ASR-рішень, зокрема Whisper (OpenAI), Google Speech-to-Text, Vosk та Mozilla DeepSpeech. Whisper продемонстрував найвищу точність розпізнавання для української мови,

а також здатність до локального розгортання без залежності від хмарних сервісів. У дослідженні [1] підтверджено ефективність Whisper у багатосерверному середовищі, зокрема для мов з обмеженим ресурсом, таких як українська. Моделі цієї платформи здатні працювати з різними акцентами та типами мовлення, що є важливим для практичного застосування в бізнес-середовищі [2].

Для оцінки конкурентних рішень було використано результати порівняльного дослідження [3], яке охоплює як комерційні, так і відкриті ASR-сервіси. Google Speech-to-Text, хоча й демонструє високі показники транскрипції, вимагає постійного підключення до мережі та залежність від сторонніх хмарних інструментів, що є вагомим недоліком. Whisper, навпаки, забезпечує можливість обробку офлайн, що відповідає вимогам до безпеки даних і швидкості.

Архітектура Telegram-бота побудована на основі фреймворку Java Spring Boot, який забезпечує масштабованість, надійність і підтримку REST-сервісів. Spring Boot є оптимальним інструментом для побудови високонавантажених систем, особливо в контексті автоматизації бізнес-процесів. У реалізованій системі передбачено багатоступеневу обробку голосових повідомлень: приймання аудіо через Telegram API, конвертацію у формат .wav, транскрибування за допомогою Whisper, аналіз тексту відповідно до бізнес-логіки, а також збереження результатів у базі даних PostgreSQL.

Особливу увагу приділено моделюванню діалогової взаємодії між користувачем і ботом. Запропоновано механізм контекстного керування станами, що дозволяє уточнювати запити, обробляти неструктуровані фрази та забезпечувати природну комунікацію. Для підтримки адміністративного контролю реалізовано веб-інтерфейс, який дозволяє переглядати транскрипти, змінювати статуси замовлень та аналізувати історію взаємодій. Досвід впровадження Telegram-ботів у дослідницьких середовищах підтверджує ефективність API як механізму автоматизації інформаційної взаємодії [5].

У процесі тестування було змодельовано типові сценарії взаємодії, зокрема оформлення замовлень, уточнення деталей та скасування запитів. Розроблене рішення має високу адаптивність і може бути застосоване в різних галузях,

зокрема у сфері громадського харчування, логістики, електронної комерції та реєстрації користувачів. Перспективними напрямками розвитку є впровадження багатомовності, використання моделей машинного навчання для класифікації запитів, інтеграція з платіжними сервісами та розширення функціоналу адміністративної панелі.

Висновки. На основі аналізу обрано оптимальний варіант для реалізації – Whisper, як відкритий, точний та локалізований інструмент із підтримкою української мови. Для реалізації логіки бота використано фреймворк Java Spring Boot, що дозволяє забезпечити модульність, масштабованість і підтримку REST-сервісів. За результатами тестів середня точність розпізнавання українського мовлення досягла 87%, що є прийнятним для комерційного застосування. Додатково оцінено швидкість обробки повідомлень (середній час від надсилання голосу до відповіді – 2.4 секунди).

Research of Language Command Recognition Technologies by Telegram-Bot

Abstract. Automatic speech recognition (ASR) technologies and their integration into Telegram bot are studied. Frameworks are analyzed and a prototype of the system is developed. Testing has confirmed its effectiveness and usability. The solution has applied value for business process automation via voice interfaces.

Keywords: speech recognition; ASR; Telegram bot; voice commands; Whisper; Google; Vosk; Java Spring Boot; voice interface; automation.