НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою ««Інформаційне забезпечення робототехнічних систем»»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Діалоговий агент для робототехнічної системи»

Виконала:
студентка IV курсу, групи ІК-12
Макарчук Ольга Вячеславівна _____

Керівник:
Старший викладач кафедри ІСТ
Коваль Олександр Сергійович _____

Рецензент:
с.н.с., к. т. н., доцент
Писаренко Юлія Валеріївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студентці
Макарчук Ользі Вячеславівні

1. Тема проєкту «Діалоговий агент для робототехнічної системи», керівник проєкту Коваль Олександр Сергійович, старший викладач кафедри ІСТ, затверджені наказом по університету від «23» травня 2025 р. № 1705-с

2. Термін подання студентом проєкту: 9 червня 2025 року

3. Вихідні дані до проєкту: для реалізації мобільного застосунку було обрано платформу .NET MAUI. Серверна частина розроблена з використанням технологій ASP.NET Core та Python. Для зберігання даних використовуються Azure Cosmos DB та Azure Blob Storage.

4. Зміст пояснювальної записки: провести аналіз предметної області. Здіснити огляд наявного методичного та технологічного забезпечення та обрати технології, що відповідають потребам додатку. Описати реалізацію додатку. Оформити інструкцію користувача мобільного застосунку.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) Діаграма прецедентів (А3), Діаграма дільності (А3), Алгоритм розрахунку вартост запитів (А3), Схема архітектури мобільного застосунку (А3)

6. Дата видачі завдання: 01 березня 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вибір теми дипломного проєкту	01.03.2025	
2	Аналіз предметної області	05.03.2025	
3	Аналіз наявних рішень та проблем	10.03.2025	
4	Ознайомлення з технологіями	20.03.2025	
5	Проектування дизайну	01.04.2025	
6	Проектування ПЗ	15.04.2025	
7	Проектування архітектури	25.04.2025	
8	Реалізація застосунку	10.05.2025	
9	Оформлення роботи	01.06.2025	

Студентка Ольга МАКАРЧУК

Керівник Олександр КОВАЛЬ

АНОТАЦІЯ

Діалоговий агент для робототехнічної системи.

Проєкт містить 64 с. тексту, 32 рисунків, посилання на 25 літературних джерел, 3 формули, додаток та 4 конструкторських документів.

Ключові слова: діалоговий агент, інтелектуальна система, NLP, роботехнічний компонент, взаємодія голосом, персоналізація.

Об’єктом дослідження є процес голосової взаємодії між користувачем та інтелектуальною системою.

Предметом дослідження є методи та інструменти реалізації діалогового агента з підтримкою розпізнавання, персоналізації та ідентифікації голосу користувача у багатокористувацькому середовищі.

Метою дипломного проєкту є автоматизація голосової взаємодії користувача з системою шляхом створення інтелектуального діалогового агента, що підтримує гнучку персоналізацію та налаштування.

На основі проведених досліджень, було розроблено діалогового агента, якого було представлено, як мобільний додаток. В системі було реалізовано персоналізацію, ідентифікацію користувачів, а також оптимальний набір інструментів для голосової комунікації, збору та аналізу даних, а також простий і зручний інтерфейс, який забезпечує інтуїтивну взаємодію з користувачем.

Отриманий діалоговий агент, може бути корисним у різних сферах, зокрема в обслуговуванні клієнтів, освітніх платформах, медичній сфері, а також у системах розумного будинку чи мобільних асистентах.

SUMMARY

Conversational agent for a robotic system.

The project contains 64 pages. text, 32 figures, links to 25 literary sources, 3 mathematics formulas, annexe and 4 design documents.

Keywords: dialogue agent, intelligent system, NLP, robotic component, voice interaction, personalization.

The object of research is the process of voice interaction between the user and the intelligent system.

The subject of the research is the methods and tools for implementing a dialogue agent with support for voice recognition, personalization, and user voice identification in a multi-user environment.

The purpose of the diploma project is to automate voice interaction between the user and the system by creating an intelligent dialogue agent that supports flexible personalization and customization.

Based on the conducted research, a dialogue agent was developed and presented as a mobile application. The system implemented personalization, user identification, as well as an optimal set of tools for voice communication, data collection and analysis, and a simple and user-friendly interface that ensures intuitive interaction with the user.

The developed dialogue agent can be useful in various fields, including customer service, educational platforms, healthcare, as well as in smart home systems and mobile assistants.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІК12.170БАК.006 ПЗ	Діалоговий агент для	64		
6			робототехнічної системи.			
7			Пояснювальна записка			
8						
9	А3	ІК12.170БАК.006 Д1	Діалоговий агент для	1		
10			робототехнічної системи.			
11			Діаграма прецедентів			
12						
13	А3	ІК12.170БАК.006 Д2	Діалоговий агент для	1		
14			робототехнічної системи.			
15			Діаграма діяльності			
16						
17	А3	ІК12.170БАК.006 Д3	Діалоговий агент для	1		
18			робототехнічної системи.			
19			Блок-схема алгоритму			
20			розрахунку вартості запитів			
21						
22	А3	ІК12.170БАК.006 Д4	Діалоговий агент для	1		
23			робототехнічної системи.			
24			Структурна схема бази даних			
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІК12.170БАК.006 ТП		
Розроб.		Макарчук О. В.			Діалоговий агент для робототехнічної системи. Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Коваль О.С.				Т	
Затв.						КПІ ім. Ігоря Сікорського Група ІК-12	

Пояснювальна записка
до дипломного проєкту
на тему: «Діалоговий агент для робототехнічної системи»

Київ – 2025 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ШІ – штучний інтелект

API (Application Programming Interface) – інтерфейс програмування застосунків

ASR (Automatic Speech Recognition) – технологія автоматичного розпізнавання мовлення

CLR (Common Language Runtime) – середовище виконання загальної мови

CQRS (Command and Query Responsibility Segregation) – розділення відповідальності на команди та запити

Google OAuth (Open Authorization) – відкрита авторизація Google

JSON (JavaScript Object Notation) – формат обміну даними у вигляді об’єктів JavaScript

JWT (JSON Web Token) – JSON-веб-токен

NLP (Natural Language Processing) – обробка природної мови

REST API (Representational State Transfer API) – передача стану представлення API

SDK (Software Development Kit) – набір засобів розробки програмного забезпечення

TTS (Text-to-Speech) – технологія перетворення тексту в мовлення

ВСТУП

У контексті сучасного розвитку робототехнічних систем, дедалі актуальнішим стає питання забезпечення природної та інтуїтивної взаємодії між людиною і технічними засобами. Одним із перспективних напрямів у цій сфері є застосування голосової комунікації, як основного каналу управління та обміну інформацією. Такий підхід дає змогу створювати більш гнучкі, персоналізовані та доступні інтерфейси, здатні адаптуватися до індивідуальних потреб користувачів.

Актуальність обраної теми зумовлена необхідністю створення інтелектуальної голосової системи, яка не лише імітує діалог, а й забезпечує персоналізовану взаємодію з користувачем на основі індивідуальних налаштувань та ідентифікації голосу. З огляду на стрімкий розвиток голосових технологій, як-от розпізнавання мовлення, клонування голосу та нейронні мовні моделі, підкреслюється необхідність створення комплексних рішень. Ці рішення мають поєднувати зазначені інструменти в єдиний, функціональний застосунок з підтримкою багатокористувацького середовища. Такі системи відкривають широкі перспективи для впровадження в бізнесі, освіті, медицині та повсякденному житті.

У межах дипломного проєкту було розроблено інтелектуального діалогового агента у форматі мобільного додатку з подальшою перспективою інтеграції з робототехнічними платформами. Система функціонує в багатокористувацькому режимі, забезпечує розпізнавання мовлення, дає змогу формувати персоналізовані профілі з індивідуальними параметрами та зберігає історію взаємодій. Після активації, додаток переходить у режим безперервного прослуховування, реагуючи на ключові фрази. Система визначає момент завершення мовлення, ідентифікує користувача на основі голосу та реалізує відповідні команди. Для конфіденційного спілкування передбачено особисті чати.

Метою дипломного проєкту є автоматизація голосової взаємодії користувача з системою шляхом створення інтелектуального діалогового агента, що підтримує гнучку персоналізацію та налаштування.

Для досягнення поставленої мети, були визначені такі завдання:

– провести аналіз сучасних технологій голосової взаємодії та діалогових агентів;

– визначити основні функціональні модулі автоматизованої системи;

– забезпечити розпізнавання голосу користувача;

– створити розпізнавання голосу, перевірку прав доступу та збереження індивідуальних налаштувань.

– реалізувати функцію клонування голосу;

– створити повний автоматизований голосовий цикл;

– забезпечити точне розпізнавання мовлення та природне озвучення відповідей, включаючи налаштування голосу.

– розробити ітелектуальну обробку запитів у багатокористувацькому середовищі;

– додати додаткові функції для гнучкості системи, такі як, підтримка аналітики, керування платіжними даними та автоплатижем, додавання калькулятора для розрахунку вартості запитів тощо;

– спроєктовати архітектуру багатокористувацької системи з підтримкою персональних голосових профілів;

– розробити зручний мобільний додаток, що виконує функції діалогового агента.

Об’єктом дослідження є процес голосової взаємодії між користувачем та інтелектуальною системою.

Предметом дослідження є методи та інструменти реалізації діалогового агента з підтримкою розпізнавання, персоналізації та ідентифікації голосу користувача у багатокористувацькому середовищі.

Дипломний проєкт містить наступні розділи: вступ, аналіз предметної області та постановка задачі, проектування архітектури додатку, програмна реалізація додатку, робота користувача із застосунком, висновки, список джерел із 25 найменувань, 1 додаток. Графічна частина включає 4 креслеників формату А3. Загальний обсяг 64 сторінок.

Для розробки системи було обрано клієнт-серверну та мікросервісну архітектуру. Android-додаток є клієнтом, створений за допомогою .Net MAUI (Multi-platform App UI), який взаємодіє з двома серверними частинами, які є незалежними, кожен з них виконує своє поставлене завдання.

Перший сервер створений за допомогою ASP .NET Core і відповідає за доступ до сторонніх сервісів. Другий сервер було реалізовано на Python та відповідає за аудіоопераії, включаючи ідентифікацію, автоматизацію всього голосового циклу, транскрибування, генерацію векторного представлення (embedding) голосу, діаризацію, виявлення кінцевої фрази. Для виконання цих задач використовувалися різні моделі.

Отриманий діалоговий агент, може бути корисним у різних сферах, зокрема в обслуговуванні клієнтів, освітніх платформах, медичній сфері, а також у системах розумного будинку чи мобільних асистентах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальний огляд проблем у галузі голосових технологій

Голосові технології стали одними з найбільш значущих інновацій у сфері взаємодії людини з комп'ютерними системами. Вони дозволяють користувачам здійснювати різноманітні дії, просто розмовляючи, що розширює горизонти зручності та інклюзивності цифрових систем. Від голосових помічників у телефонах до автоматизованих систем для людей з обмеженими можливостями, голосові інтерфейси дедалі глибше входять у наше щоденне життя. Попри значний прогрес у цій області, існують певні виклики, що потребують вирішення для забезпечення високої продуктивності та безпеки таких систем.

Одним з ключових викликів, постає точність та ефективність розпізнавання мовлення. Системи повинні бездоганно розпізнавати слова й фрази, ігноруючи акцент, тембр голосу, фоновий шум та інші змінні обставини. Помилки під час розпізнавання можуть призвести до непорозумінь або ж невиконання команд, що негативно позначається на користувацький досвід.

Ще однією значущою проблемою є ідентифікація користувачів. Оскільки голос має неповторну природу для кожної особи, його застосування для аутентифікації та доступу до систем набуває вагомого значення в безпеці. Технології, що стосуються ідентифікації за голосом, наразі не досягли ідеального рівня, що породжує потенційні загрози для забезпечення конфіденційності користувачів.

Проблема багатокористувацької підтримки також є однією з важливих в цій галузі. Значна частина голосових систем розроблені для особистого користування, проте сьогодення потребує більш адаптивних рішень, що забезпечують взаємодію між кількома користувачами. Це критично важливо для сімейних, робочих чи навчальних обставин, де одна система мусить надавати послуги різним особам, враховуючи їхні власні індивідуальні налаштування.

Отже, незважаючи на те, що голосові технології вже міцно закріпилися як фундамент багатьох інноваційних рішень, їхній подальший розвиток та

впровадження потребують вирішення низки серйозних викликів, починаючи від точності розпізнавання й закінчуючи питаннями безпеки. Технології, засновані на штучному інтелекті, демонструють значний потенціал для подолання цих перешкод, проте їх успішне застосування залежить від подальших досліджень і вдосконалення існуючих методів та алгоритмів.

1.2 Діалогові агенти в робототехнічних системах

Діалогові агенти, також відомі як розмовні або голосові помічники, є програмними комплексами, котрі здатні підтримувати спілкування з користувачем, використовуючи природну мову. У робототехнічному середовищі ці агенти виступають у ролі посередника між людиною та роботом, пропонуючи зручний метод управління, конфігурації та обміну даними. Завдяки застосуванню мовних моделей, систем розпізнавання і відтворення мовлення, а також механізмів розуміння контексту, такі агенти прагнуть максимально наблизитися до природного способу спілкування [8].

Впровадження розмовних агентів у роботизовані платформи значно полегшує спілкування з машинами, особливо там, де потрібне керування без рук, як-от у виробництві, медицині, повсякденному житті чи навчанні. Це відкриває свіжі перспективи, включно з голосовим управлінням роботами, що дозволяє юзерам управляти ними голосовими наказами, що є особливо комфортним, коли фізичний дотик утруднений або не потрібен. Завдяки контекстній взаємодії агент розшифровує запити, зважаючи на історію розмови, місцеперебування або стан користувача, гарантуючи більш гнучку реакцію робота. Персоналізація стає можливою через розпізнавання та ідентифікацію голосу, що дає змогу роботу підлаштовувати свої дії під конкретного юзера, приміром, коригувати режим роботи або відповідати згідно з індивідуальними налаштуваннями. Більше того, інтеграція з сенсорними даними дає змогу розмовному агенту використовувати інформацію з датчиків, як-от температура, заряд батареї чи розташування у просторі, для формування більш інформативних та чітких відповідей.

У системах робототехніки, діалогові агенти нерідко виступають у ролі віртуальних помічників. Вони допомагають користувачеві не тільки керувати роботом, але й отримувати інформацію про його стан, діагностувати несправності, а також навчатися користуванню новими функціями. Ці агенти можуть бути втілені як вбудованим програмним забезпеченням, що функціонує безпосередньо на робототехнічному пристрої, так і у формі хмарних сервісів, що надають доступ до потужніших мовних моделей, аналізу даних та централізованого управління.

Отже, діалогові агенти становлять собою ключовим елементом сучасних робототехнічних систем, гарантуючи не тільки технічну працездатність, але й збільшуючи ступінь довіри, зручності та доступності згаданих систем для широкого кола користувачів. Їх подальший розвиток безпосередньо впливатиме на темпи інтеграції робототехніки у різноманітні сфери життя – від повсякденного побуту до життєво важливих галузей.

1.3 Роль штучного інтелекту в досліджуваній проблемі

Штучний інтелект (ШІ) стає ключовим інструментом у подоланні основних викликів, з якими стикаються голосові технології. Застосування алгоритмів машинного навчання, зокрема глибоких нейронних мереж, відкриває нові можливості в таких областях як розпізнавання мовлення, ідентифікація користувачів, синтез голосу та адаптація до контексту.

У галузі розпізнавання мовлення, штучний інтелект робить кроки до радикального покращення точності. Це стає можливим через навчання моделей на масивах даних, які охоплюють широкий спектр мов, акцентів, голосових інтонацій та умов запису. Моделі, на кшталт трансформерів, зокрема, Whisper від OpenAI, виявляють здатність пристосовуватися до складних акустичних середовищ. Вони також вирізняються стійкістю до шумів, що значно поліпшує якість взаємодії користувача з системою.

У синтезі мовлення (TTS), сучасні моделі ШІ вже забезпечують майже людську якість голосу. Вони можуть відтворювати емоційний стан, переходи в

інтонації та акценти, що робить комунікацію з голосовими інтерфейсами більш природною та ефективною. Особливо перспективним є напрям персоналізованого синтезу мовлення, який дозволяє створювати індивідуальні голоси, адаптовані під потреби конкретних користувачів або брендів.

Ще одним ключовим моментом є пристосування до обставин. Впровадження контекстно-чутливих моделей (на кшталт BERT або архітектур, схожих на GPT) дає змогу розробляти системи, котрі здатні «розуміти» задуми користувача й реагувати відповідно до ситуації.

Отже, роль ШІ у розвитку голосових технологій не просто допоміжна, а визначальна. Він забезпечує інтелектуалізацію усіх ключових складових цих систем: від розпізнавання мовлення до його синтезу, від безпеки до персоналізації. Проте, для досягнення високого рівня надійності та здатності адаптуватися, необхідне подальше вдосконалення моделей та покращення якості тренувальних даних.

1.4 Аналіз існуючих рішень

1.4.1 Siri

Siri — голосовий діалоговий агент, розроблений Apple, тісно інтегрований в екосистему їхніх пристроїв, зокрема iPhone, iPad, Mac, Apple Watch та HomePod [16]. Siri працює на основі розпізнавання мовлення та дає змогу користувачам керувати функціями голосом: встановлювати будильники, надсилати повідомлення, шукати інформацію в інтернеті, відтворювати музику, користуватися навігацією, а також здійснювати просте управління "розумним домом".

Не зважаючи на те, що Siri переважно використовується на мобільних платформах, вона інтегрована і в стаціонарне обладнання, наприклад, HomePod. Це вказує на потенціал її включення у фізичні апаратні платформи.Це відкриває можливості її застосування в контексті робототехніки, але слід підкреслити, що Siri не пропонує безпосередньої інтеграції з робототехнічними пристроями інших

виробників. До того ж, її API (Application Programming Interface — інтерфейс програмування застосунків) обмежені межами екосистеми Apple.

Siri має кілька суттєвих недоліків, що викликають певні обмеження для користувачів. Наприклад, персоналізація суттєво обмежена, оскільки користувачі мають можливість вибрати лише між чоловічим або жіночим голосом та акцентами для певних мов. Це серйозно звужує потенціал налаштування та кастомізації. Також відсутня функція перегляду історії попередніх запитів, що може бути вкрай незручно.

Крім того, інтелект Siri залишається на досить базовому рівні. Вона здатна розуміти та відповідати лише на обмежену кількість запитів. Складніші діалоги, що передбачають кілька кроків, практично неможливі без необхідності постійної активації фразою, щоразу.

Рисунок 1.1 – Пристрої, що підтримують Siri [21]

1.4.2 Alexa

Amazon Alexa — голосовий агент, спроектований для управління голосовими командами на різноманітних пристроях: смарт-колонки, телевізори, розумні екрани та інтегровані IoT-пристрої. Alexa демонструє перспективи інтеграції з роботизованими системами, особливо завдяки підтримці Alexa Voice Services (AVS), що дозволяє вбудовувати Alexa у сторонні прилади, зокрема роботів [18].

Ключова перевага Alexa полягає у підтримці режиму для кількох користувачів і наявності Follow-Up Mode, завдяки чому можливо вести розмови без

потреби знову викликати помічника стартовою фразою. Alexa пропонує добре розвинений набір функцій для автоматизації, таких як Alexa Skills Kit. Їх можна пристосувати для створення голосової взаємодії з різноманітними фізичними приладами, включно з роботами-асистентами, роботами-доставниками та іншими.

Втім, Alexa має й певні вади. Користувачам доступний обмежений вибір голосових варіантів, що здебільшого зосереджені на акцентах чи стилістичних особливостях, водночас як повне налаштування голосу, включаючи його тональність чи тембр, поки що недоступне. До того ж, дозволено обирати тільки з кількох заздалегідь визначених стартових фраз, як-от “Alexa”, “Amazon”, “Echo” чи “Computer”. Варто ще зауважити, що розглянутий агент не забезпечує повністю ізольований приватний чат для кожного користувача — історія запитів доступна всім, хто має доступ до спільного облікового запису.

Голосовий помічник Alexa слугує зразком вдалої реалізації діалогового агента, спроможного до природної мовної комунікації з користувачем.. Система перманентно вдосконалюється за рахунок застосування машинного навчання та оновлення моделей, що дозволяє пристосовуватись до індивідуальних потреб користувачів та підтримувати багатокористувацьке середовище.

Рисунок 1.2 – Пристрої, що підтримують Alexa [22, 23]

1.4.3 Google Assistant

Google Assistant — один з найпотужніших діалогових агентів, що володіє передовим штучним інтелектом, який демонструє здатність до глибокого розуміння контексту та підтримки природнього спілкування [17]. Незважаючи на

те, що його часто пов'язують з мобільними девайсами та сервісами Google, наприклад Gmail, Calendar, YouTube, його можна інтегрувати у стороннє обладнання, включаючи роботів і вбудовані системи, через Google Assistant SDK.

Це робить Google Assistant надзвичайно корисним у сфері робототехніки, де необхідна взаємодія з урахуванням контексту, персоналізації та голосового керування. Функція "Continued Conversation" дозволяє підтримувати безперервний діалог, уникаючи повторення фрази активації. Це значно покращує інтерактивну взаємодію з автономними роботами.

Асистент також вміє працювати з голосовими профілями. Це означає, що він може розпізнавати кожного користувача за голосом. Завдяки цьому він надає відповіді, враховуючи особисті налаштування: наприклад, відображає інформацію з календаря, показує сповіщення або прокладає маршрути, які відповідають саме вам. Користувачі мають змогу обирати з-поміж різноманітних варіантів голосу. Вони адаптовані під різні мови та акценти, щоб було зручно всім. Повної зміни голосу, як-от створення власного унікального голосу, на жаль, поки що немає. Але навіть ці налаштування дають певний базовий рівень персоналізації.

Стосовно активації, Google Assistant оперує сталою фразою для запуску ("Hey Google"), що не передбачає кастомізації.

Незважаючи на низку позитивних рис, помічник Google також має свої недоліки. Серед них – обмежена кількість голосових варіантів, а можливості персоналізації голосу для кожного користувача досить скромні. Окрім того, існують питання щодо безпеки приватних даних, оскільки опрацювання запитів відбувається на хмарних серверах Google, що вимагає довіри до політики компанії стосовно захисту особистої інформації.

Завдяки підтримці контекстних розмов, часткової персоналізації профілів та інтеграції з безліччю сервісів, Google Assistant залишається одним з найгнучкіших голосових помічників на ринку. Його здатності у сфері природного спілкування з користувачами роблять його корисним як для щоденного використання, так і для складних сценаріїв автоматизації.

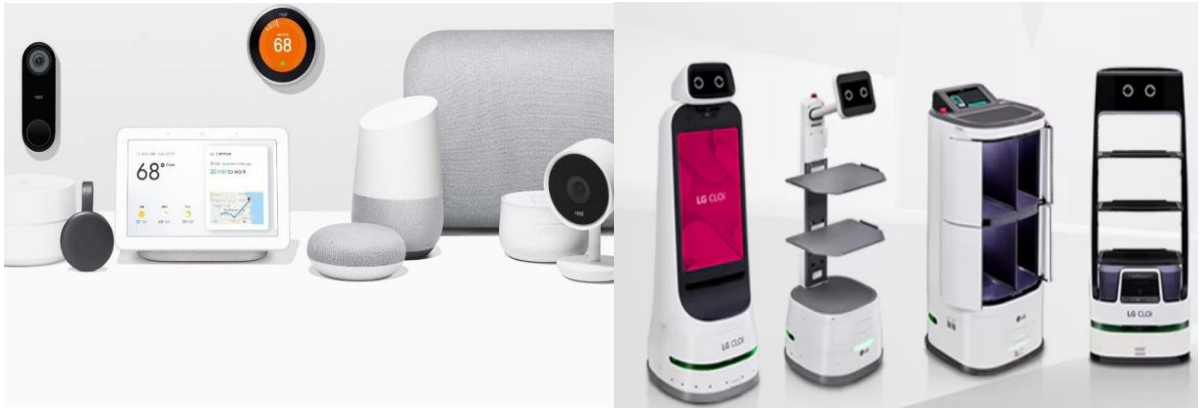

Рисунок 1.3 – Пристрої, що підтримують Google Assistant [24, 25]

1.5 Постановка задачі

У сьогоденні, коли цифрові та робототехнічні технології стрімко розвиваються, виникає дедалі більша потреба у розробці інтелектуальних систем, що базуються на голосовому спілкуванні. Ці системи повинні вміти природно, контекстно та ефективно взаємодіяти з користувачем. Особливого значення набуває інтеграція діалогових агентів у платформи робототехніки, адже зменшення фізичного контакту людини з інтерфейсом дозволяє збільшити автономність системи та покращити зручність її використання.

Завдання полягає у розробці багатогранної програмної системи, що інтегруватиме різноманітні складові голосових технологій. Це передбачає розпізнавання людської мови, ідентифікацію користувачів за їхнім голосом, персоналізований синтез мовлення, а також функцію клонування голосу. Розроблена система має бути спроектована для роботи в багатокористувацькому режимі, з підтримкою кількох окремих голосових профілів, прив'язаних до одного облікового запису.

Одним з ключових завдань, що потребує розв'язання в межах поставленої цілі, є гарантування стабільної автоматизації процесу взаємодії — від розпізнавання ключової фрази до завершення голосової бесіди. До того ж, система має бути спроможною:

– самостійно визначати межі мовлення користувача;

– виконувати ідентифікацію та перевірку прав доступу на основі характеристик голосу;\

– зберігати історію взаємодій та індивідуальних налаштувань кожного користувача;

– забезпечувати конфіденційність даних і безпечний доступ до персоналізованих голосових чатів;

– реалізовувати розпізнавання мовлення (Speech-to-Text) для розуміння голосових команд у реальному часі;

– виконувати синтез мовлення (Text-to-Speech) з можливістю клонування голосу або вибору серед базових голосових варіантів;

– забезпечувати гнучке налаштування голосового профілю;

– обробляти запити користувачів із використанням технологій обробки природної мови (NLP і формування логічних відповідей;

– підтримувати багатокористувацьке середовище з веденням особистих діалогів і засобами захисту доступу;

– забезпечувати зручний перегляд історії голосових запитів та відповідей у рамках інтерфейсу користувача.

Висновки до розділу 1

У розділі було здійснено ґрунтовний аналіз предметної області голосових технологій та окреслено актуальні проблеми, пов’язані з точністю розпізнавання мовлення, безпечністю обробки даних, персоналізацією синтезу мовлення та обмеженою підтримкою багатокористувацьких сценаріїв. Розглянуто роль штучного інтелекту, зокрема технологій машинного навчання та обробки природної мови, як основи для створення інтелектуальних систем голосової взаємодії, що здатні підвищувати ефективність взаємодії та забезпечувати індивідуалізований досвід.

Siri, Alexa та Google Assistant – найпопулярніші голосові помічники, котрі можуть похвалитися своїми перевагами та різняться у питанні інтеграції в

роботизовані системи. Siri має вагому перевагу завдяки тісній інтеграції з екосистемою Apple та орієнтації на конфіденційність. Проте, через обмежений доступ до платформи, її нелегко впроваджувати в сторонні роботизовані пристрої, а можливості персоналізації та обробки складних розмов поки що знаходяться на початковому рівні. Alexa вигідно вирізняється підтримкою багатокористувацького режиму та функцією Follow-Up Mode, а також відкритим AVS-SDK для вбудовування у різне обладнання, починаючи від розумних колонок і закінчуючи автономними роботами. Але Alexa програє в питанні гнучкості налаштування голосу та активаційних фраз. Google Assistant пропонує найзручніший досвід користувача завдяки функції Continued Conversation, широким можливостям персоналізації та щільній інтеграції з сервісами Google, а також легко інтегрується в роботизоване обладнання завдяки Google Assistant SDK (Software Development Kit — набір засобів розробки програмного забезпечення). Водночас модель обробки даних у хмарі може викликати занепокоєння щодо конфіденційності даних. Таким чином, вибір платформи залежить не лише від потреб користувача та його цифрового середовища, але й від специфіки апаратної інтеграції, відкритості SDK та особливостей роботизованого середовища.

Це дослідження починається з формування основної мети: створити розумного діалогового агента для застосування в робототехнічних системах. Цей агент має інтегрувати в собі розпізнавання мови, ідентифікацію особистості, синтез мовлення, а також дозволяти гнучко налаштовувати профілі, змінювати голосові дані та забезпечувати безпечну автоматизовану взаємодію з користувачами. На даному етапі, розробка ведеться у вигляді мобільного додатку, який служить прототипом і буде інтегрований в майбутньому з апаратними засобами робототехнічної платформи.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ДОДАТКУ

2.1 Вибір програмних технологій для розробки

2.1.1 C# як обрана мова програмування

Використання C# та ASP.NET Core має високий рівень надійності завдяки суворій типізації і, зокрема, чіткому та оптимальному синтаксису, що значно спрощує читання коду, налагодження та подальший рефакторинг [19].

Сучасне CLR (Common Language Runtime — загальне середовище виконання мов) із підтримкою JIT-компіляції (Just-In-Time) гарантує відмінну продуктивність і кероване розподілення пам’яті, а вбудовані механізми безпеки захищають застосунок від поширених уразливостей.

Також C# підтримує об’єктно-орієнтоване програмування, що сприяє модульності, повторному використанню коду та зручному масштабуванню проєкту.

Завдяки багатій екосистемі пакетів і бібліотек, яка задовольняє всі потреби та економить час — такі як Entity Framework для комфортної роботи з БД (базою даних) чи Azure SDK для інтеграції з хмарними сервісами — дозволяє швидко масштабувати проєкт і реалізовувати зберігання та обробку даних будь-якого обсягу [5, 6].

Через уніфікований стек технологій можна без проблем поєднувати серверну логіку, веб-інтерфейс на ASP.NET Core та клієнтські додатки на .NET MAUI (Multi-platform App UI - Багатоплатформенний інтерфейс користувача), що покривають десктопні (Windows, macOS) і мобільні (iOS, Android) платформи з мінімальними витратами на підтримку коду [12, 15].

Узагальнюючи, можна зробити узагальнення, що стек C#/.NET — це гнучке, надійне та масштабоване рішення для створення високонавантажених бізнес-орієнтованих сервісів з довгостроковою підтримкою та швидким виходом на ринок. Відкритий вихідний код .NET гарантує безперервний розвиток платформи, що робить її актуальною як для невеликих стартапів, так і для великих корпоративних систем.

2.1.2 Python як обрана мова програмування

Вибір Python, як мову програмування, зумовлено через її переваги у поєднанні з машинним навчанням, обробкою сигналів і математичними обчисленнями. Її простий, легкий для читання синтаксис і динамічна типізація дозволяють швидше писати код, тестувати ідеї та спрощувати подальше розширення функціоналу [20].

Python має одну з найрозвиненіших екосистем для роботи зі штучним інтелектом. Завдяки бібліотекам, таким як PyTorch, TensorFlow, SpeechBrain, NVIDIA NeMo, librosa та torchaudio, можлива ефективна реалізація модуля для обробки аудіо і верифікації голосу [3].

Для розробки API використовується FastAPI — сучасний високопродуктивний фреймворк, що підтримує REST API (Representational State Transfer Application Programming Interface) та WebSocket-з'єднання. Це дозволяє краще організувати взаємодію між клієнтом і модулем обробки аудіо.

Активна спільнота Python, велика кількість відкритого коду, регулярні оновлення та наявність ґрунтовної документації забезпечують стабільність, безперервну підтримку та швидкий розвиток проєкту. У комбінації з іншими технологіями, Python є потужним інструментом для створення інтелектуальних програмних модулів.

2.1.3 Blob Storage як сховище даних

Для зберігання великих обсягів неструктурованих даних (зокрема аудіофайлів, світлин) обрана служба Azure Blob Storage. Даний сервіс є оптимальним рішенням завдяки своїй орієнтації на масштабування, високу доступність та низькі витрати при роботі з великими файлами [6].

Blob Storage автоматично масштабується відповідно до обсягів збережених даних та навантаження на систему. За допомогою глобальній інфраструктурі Azure,

файли легкодоступні з мінімальною затримкою, що є критично важливим для обробки запитів у режимі реального часу.

Сервіс підтримує кілька рівнів доступу (Hot, Cool, Archive), які дозволяють оптимізувати вартість зберігання залежно від частоти використання даних. Наприклад, активні записи голосових сесій можна розміщувати в Hot-рівні для миттєвого доступу, а старі й рідко використовувані аудіофайли архівувати на Archive-рівні.

Офіційні SDK для C# (.NET Azure.Storage.Blobs) забезпечують зручні методи для роботи з контейнерами, файлами й потоками даних. Вбудовані асинхронні API дозволяють ефективно обробляти аудіо в реальному часі, не блокуючи виконання інших сервісів.

Використання Azure Blob Storage у якості основного сховища неструктурованих даних гарантує необхідну продуктивність, надійність і безпеку та дозволяє гнучко масштабувати систему згідно з ростом кількості користувачів і обсягів голосових записів.

2.1.4 База даних Cosmos DB

Для зберігання структурованих даних було обрано базу даних Azure Cosmos DB. Цей сервіс пропонує гнучку роботу з документно-орієнтованими моделями даних у глобально розподіленому середовищі з низькою затримкою [6].

Cosmos DB підтримує зберігання даних у форматі JSON(JavaScript Object Notation – запис об'єктів JavaScript), що є зручним для зберігання метаданих. Крім того, гнучка схема (підхід до зберігання даних, при якому немає суворо визначеної структури для кожного документа в базі даних) робить простим коригування структури даних при змінах у системі, без необхідності складних оновлень бази даних.

Оскільки система дозволяє реплікацію даних між кількома регіонами, вона може продовжувати надавати високу доступність з мінімальною затримкою, звідки

б не надсилалися запити. Це важливо для підтримки швидкого доступу до даних у реальному часі.

Cosmos DB також має можливості масштабування. З цією функцією, можна створювати бази даних, які відповідають поточним потребам, тим самим зменшуючи витрати, коли активність знижується.

Для зручності роботи з базою даних доступні API, а також інтеграція з Entity Framework Core, що спрощує використання бази даних за допомогою звичних методів і запитів. Ці особливості роблять Azure Cosmos DB надійним і ефективним рішенням для зберігання даних, забезпечуючи необхідну швидкість, доступність та безпеку.

2.1.5 Інтегроване середовище розробки

Visual Studio та Visual Studio Code – є необхідними інструментами, які використовуються в проєкті, завдяки зручності та ефективності в роботі. Ці дві сучасні платформи взаємодоповнюють одна одну: Visual Studio Code — це легкий та швидкий редактор коду з великими можливостями для розширення, а Visual Studio — потужне інтегроване середовище розробки (IDE), яке надає повний набір функцій для розробки, налагодження та тестування, особливо для платформ ASP.NET Core і C#. Такий підхід дозволяє працювати з різними мовами програмування та інструментами, забезпечуючи гнучкість і продуктивність у процесі розробки.

Visual Studio Code – це кросплатформне середовище розробки, яке вирізняється легкістю та швидкістю запуску та можливістю розширення. Воно чудово підходить для розробки повноцінних застосунків, зокрема на Python, JavaScript, TypeScript, Go, C++ та інших мовах. Завдяки багатому функціоналу розширень, підтримці Docker, вбудованому терміналу, інтеграції з Git та системами контролю версій, а також функціям автоматичного форматування й перевірки коду, VS Code сприяє високій продуктивності. Гнучкість цієї платформи дозволяє її адаптувати під конкретні вимоги кожного проєкту.

Visual Studio – це потужне інтегроване середовище розробки (IDE), що оптимально підходить для розробки програмного забезпечення на базі .NET, C#, ASP.NET, Blazor та інших пов'язаних технологіях. Воно пропонує широкі можливості для проєктування архітектури, створення API, модульного тестування, управління залежностями, профілювання та розгортання. Інтеграція з хмарними сервісами, зокрема Azure, дозволяє автоматизувати процеси DevOps, а ведення історії збірок полегшує контроль стабільних версій продукту.

Об'єднавши Visual Studio Code і Visual Studio, можна ефективно розв'язувати широкий спектр задач – від написання окремих скриптів до розробки великомасштабних систем. Такий підхід забезпечує гнучкість, високу продуктивність і швидке реагування на зміни вимог протягом усього життєвого циклу програмного забезпечення.

2.2 Визначення функціональних модулів системи

Система складається з ряду функціональних модулів, кожен з яких відповідає за конкретний набір функцій і втілює певну частину бізнес-логіки. Цей поділ забезпечує високу модульність, що полегшує супровід, розширення функціоналу та повторне використання коду.

Такий підхід сприяє реалізації принципів SOLID, особливо принципу єдиної відповідальності, що підвищує якість архітектури системи.

Кожен модуль має чітко визначені інтерфейси взаємодії з іншими частинами системи, що сприяє підтримці принципів розподіленої архітектури та зменшує взаємозалежності між компонентами. Це також дозволяє незалежно розробляти, перевіряти та оновлювати окремі моділі, не ризикуючи порушити загальну стабільність системи.

Отже, така модульна структура сприяє підвищенню загальної надійності системи, оскільки ізоляція функціональних компонентів зменшує ризик поширення помилок між модулями. Крім того, скорочується час розробки завдяки паралельній роботі над окремими модулями та спрощує тестування. У довгостроковій

перспективі це полегшує підтримку, масштабування й адаптацію системи до нових вимог.

2.2.1 Модуль активації голосового вводу

Модуль голосової активації реалізує безперервне пасивне прослуховування аудіосигналу, маючи на меті розпізнавання фрази-ініціатора. Ця фраза виступає сигналом для переходу системи у режим готовності до сприйняття запитів.

Головна мета - підтримувати постійну "увагу" інтерфейсу, не створюючи надмірного навантаження на апаратні ресурси пристрою. Доки користувач не вимовить попередньо визначену тригерну фразу, система функціонує у режимі очікування, утримуючись від активного обробляння мовлення. Завдяки цьому, вдається зменшити ймовірність хибних спрацьовувань та ефективно заощаджувати енергію.

Для втілення модуля, використовується клас SpeechRecognizer, включений до стандартної реалізації Android, який відповідає за розпізнавання голосових команд. Після ініціалізації цього класу, формується спеціальний запит з налаштуваннями мови та мінімальної тривалості прослуховування. За допомогою обробника подій для розпізнавання мови система може реагувати на часткові результати (наприклад, коли ще не завершено введення) та помилки. Це дозволяє оперативно виявляти моменти вимовляння ключової фрази користувачем, що дає можливість негайно активувати відповідну реакцію.

У момент виявлення стартової фрази, відбувається перехід до наступного модуля: активується запис аудіо для подальшого перетворення мовлення на текст. У ситуаціях відмови сервісу або виникнення помилок, система автоматично перезапускає прослуховування з використанням попередньо заданих налаштувань, що забезпечує безперебійну роботу. Застосований підхід гарантує високу стабільність та продуктивність, додаючи можливість безперешкодної адаптації під інші мобільні операційні системи з аналогічними інтерфейсами розпізнавання мовлення.

Підсумовуючи, модуль активації голосового введення є ключовим компонентом для створення інтуїтивно зрозумілого голосового інтерфейсу, що реагує лише тоді, коли це дійсно необхідно. Це значно поліпшує взаємодію з користувачем та гарантує надійність в умовах реального застосування.

2.2.2 Модуль транскрибування та ідентифікації

Модуль транскрибування та ідентифікації виконує завдання перетворення аудіосигналу у текстовий формат та перевірки, чи ідентифікований голос належить конкретному користувачеві. В його роботі застосовано комбінацію методів діаризації, розпізнавання мовлення та ідентифікації особи за унікальними голосовими характеристиками.

Важливу роль виконує модель діаризації SortformerEncLabelModel, розроблена NVIDIA NeMo. Її завдання - розпізнавати, які сегменти аудіозапису належать кожному з мовців. Це дає змогу ізолювати тільки ті частини, де звучить голос особи, що сформулювала запит [11]. Такий підхід підвищує точність та гарантує безпеку під час транскрибування, бо виключає потрапляння в обробку голосів сторонніх людей.

Фундаментом для транскрипції слугує модель Whisper від OpenAI, що належить до передових систем автоматичного розпізнавання мовлення (ASR – Automatic Speech Recognition). Whisper демонструє високу точність розпізнавання мовлення навіть за несприятливих обставин: шуму, акцентів чи розмовної манери. У межах цього модуля модель функціонує в хмарному режимі через API OpenAI, що передбачає надсилання згенерованих аудіосегментів і отримання текстової розшифровки [13].

Для визначення особи користувача за голосом використовується модель ECAPA-TDNN, втілена в бібліотеці SpeechBrain. Ця модель створює векторне представлення (embedding) голосу сталого розміру, яке потім зіставляється з еталонними embedding-ами відомих користувачів, використовуючи косинусну відстань.

Косинусна відстань - це міра, що визначає, наскільки схожі два вектори за їхнім напрямком у багатовимірному просторі. Ця величина обчислюється на основі кута між двома векторами в багатовимірному просторі. Спочатку знаходиться косинус кута між векторами, що визначається як відношення скалярного добутку векторів до добутку їх довжин (формула 2.1).

Потім косинусна відстань визначається як одиниця мінус косинусна схожість (формула 2.2).

Якщо схожість перевищує встановлений рівень (тобто відстань є меншою за визначене значення), система визначає, що це голос відповідного користувача.

$$a(\mathrm{v}_1, \mathrm{v}_2) = \frac{\mathrm{v}_1 \cdot \mathrm{v}_2}{||\mathrm{v}_1|| \cdot ||\mathrm{v}_2||} = \frac{\sum_{i=1}^{n} \mathrm{v}_{1i}\, \mathrm{v}_{2i}}{\sqrt{\sum_{i=1}^{n} \mathrm{v}_{1i}^2 \cdot \sum_{i=1}^{n} \mathrm{v}_{2i}^2}}, \quad (2.1)$$

$$b = 1 - a, \quad (2.2)$$

де: a – косинусна схожість, b – косинусна відстань, v1 – перший вектор, v2 – другий вектор

Крім того, у даному модулі було використано модель Vosk (vosk-model-uk-v3-lgraph) для виявлення кінцевої фрази. Вона використовує лінійні графи, щоб моделювати мовні зв'язки, що дає змогу точно та оперативно визначати кінцеву точку фрази, одночасно скорочуючи затримки в реальному часі, Це сприяє швидкій реакції системи, уникаючи зайвих пауз, що критично важливо для діяльності в реальному часі.

На рисунку 2.1 показано всі нейромережі, які були використані в цьому модулі, зокрема моделі Whisper від OpenAI для транскрипції, ECAPA-TDNN дозволяє отримувати вектори ознак із голосових записів користувачів для ідентифікації користувача, SortformerEncLabelModel для діаризації(розбиття аудіо на мовців) та Vosk для виявлення кінцевої фрази. Крім того, на рисунку 2.2, зображено візуально шари модуля транскрибування та ідентифікації.

```python
client = OpenAI(api_key=OPENAI_API_KEY)
transcription_response = client.audio.transcriptions.create(
    model="whisper-1",
    file=audio_file,
    language="uk"
)

model = Model("D:/Diploma/vosk-model-uk-v3-lgraph")
def create_vosk_recognizer():
    return KaldiRecognizer(model, RATE)

classifier = EncoderClassifier.from_hparams(
    source="speechbrain/spkrec-ecapa-voxceleb",
    run_opts={"device": device}
).to(device)

diar_model = SortformerEncLabelModel.from_pretrained("nvidia/diar_sortformer_4spk-v1")
diar_model.eval()
```

Рисунок 2.1 – Використані моделі ШІ

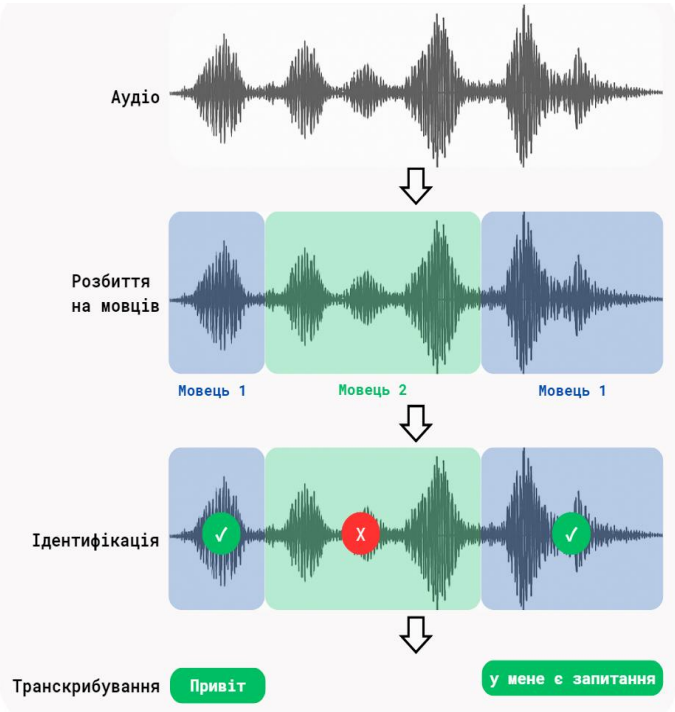

Рисунок 2.2 – Шари модуля транскрибування та ідентифікації

Підсумовуючи, можна стверджувати, що модуль транскрипції та ідентифікації є прикладом високотехнологічного рішення, яке поєднує в собі кілька рівнів аналізу аудіосигналу: від перетворення мовлення в текст до визначення мовця та розрізнення голосів у багатомовних середовищах. Його впровадження, надає великі перспективи для розробки безпечних, гнучких та комфортних голосових сервісів у різних сферах — від віртуальних асистентів до систем голосової перевірки ідентичності у різних організаціях.

2.2.3 Модуль обробки запитів

Модуль обробки запиту відповідає за отримання вже транскрибованого та первинно розпізнаного тексту від користувача, а також реалізацію подальшої логіки визначення наміру та створення відповіді.

Цей модуль аналізує контекст діалогу, беручи до уваги попередні повідомлення, поточні параметри сесії (наприклад, чи то приватне, чи загальне спілкування), а також можливі системні команди. Наступним кроком, є генерація відповіді за допомогою OpenAI.

Головним аспектом цього модуля є подальша трансформація відповіді в мовленнєвий потік через API ElevenLabs: згенерований OpenAI текст додається до TTS запиту, який, використовуючи визначений голос, повертає масив байтів з аудіо [9]. Після цього, функція програвання аудіо відтворює відповідь безпосередньо для користувача.

Також присутній функціонал підрахунку витрат за кожен запит і формування аналітики витрат.

Отже, модуль обробки запиту інтегрує інтелектуальний аналіз та генерацію природної мови з використанням OpenAI, а саме GPT-4o, підрахунок та збереження аналітики витрат і перетворення тексту в мовлення за допомогою ElevenLabs, а саме Eleven Flash v2.5, гарантуючи безперебійний перехід від голосового введення до голосового виведення в межах застосунку.

2.2.4 Модуль управління користувачами та клонування голосу

Модуль управління користувачами та клонування голосу охоплює всі функції, необхідні для створення, конфігурування та оновлення акаунтів підкористувачів системи, а також для роботи з голосовою інформацією – від запису та генерування унікального голосового відбитку до вибору базового або клонуваного голосу. При створенні нового користувача, система проводить

валідацію обов'язкових полів. Крім того, за власним бажанням, користувач має можливість встановити пароль для доступу до особистої історії виконаних запитів.

Якщо користувач обирає стандартний набір голосів, відбувається завантаження каталогу доступних голосових моделей із зовнішнього API, фільтрація за описом, віковою групою, гендером та сценарієм використання, після чого можна прослухати обраний голос безпосередньо в інтерфейсі. Водночас модуль надає можливість створити унікальний голосовий відбиток: записати фразу через мікрофон або завантажити готовий аудіофайл, після чого сервіс генерує вектор ознак голосу для подальшого використання в системі.

Якщо користувач вибирає стандартний пакет голосових опцій, система ініціює завантаження каталогу наявних голосових моделей. Відбувається сортування інформації за описом голосу, віковою категорією, статтю та метою застосування. Після цього, користувач має змогу відтворити зразок вибраного голосу прямо у вікні програми.

У випадку потреби клонування голосу, задіюється окремий підпроцес, який взаємодіє з API ElevenLabs. Після того, як користувач записує аудіо на 30 секунд або завантажує аудіофайл і підтверджує, на віддаленому сервері формується модель голосу, щойно після цього генерується унікальний ідентифікатор клону. Якщо користувач має намір оновити свій клон, попередній запис ліквідується автоматично, перед тим як буде створено новий.

Одночасно модуль дозволяє створити власний, неповторний голосовий відбиток: можна записати аудіо на 10 секунд через мікрофон. Після цього сервіс обробляє дані, створюючи вектор характеристик голосу, який надалі використовується системою.

Отже, цей модуль дозволяє адаптувати параметри для кожного окремого підкористувача. Він підтримує, як стандартні голосові моделі, так і можливість повного клонування голосу на основі аудіозапису. Крім того, він пропонує усі необхідні інструменти для надійного зберігання та керування особистою інформацією користувачів.

2.2.5 Модуль управління даними

У рамках розробки було створено спеціалізований модуль управління даними на основі API, реалізованого за допомогою платформи ASP.NET Core. Цей API слугує сполучною ланкою між клієнтською стороною системи та сервісами зберігання даних, забезпечуючи безперебійний обмін інформацією [15]. Для зберігання неструктурованих даних, зокрема зображень і документів, використовується хмарне сховище Azure Blob Storage. Натомість для зберігання структурованої інформації, такої як метадані, налаштування користувача та інші параметри, застосовується розподілена база даних Cosmos DB, що гарантує високу продуктивність, можливість масштабування та постійну доступність.

API функціонує через стандартизований RESTful-інтерфейс, використовуючи принципи архітектури REST для обміну даними між складовими системи. Це спрощує взаємодію з різними модулями та сервісами за допомогою стандартних HTTP-методів (GET, POST, PUT та DELETE), дозволяючи отримувати, додавати, змінювати або видаляти інформацію [10]. Крім того, в модулі реалізовано механізми аутентифікації та авторизації, спрямовані на захист чутливих даних. Разом ці компоненти утворюють стабільну архітектуру для централізованого та безпечного керування даними в рамках програмного забезпечення.

2.2.6 Модуль інтерфейсу користувача

Модуль користувацького інтерфейсу відповідає за візуальну та інтерактивну складову взаємодії з системою. Його задача – зробити так, щоб доступ до основних функцій був простим та інтуїтивним. Головний акцент зроблено на комфорті, адаптивності до різних типів пристроїв та мінімізації кроків для виконання основних задач.

Інтерфейс виконано в сучасному дизайні з зрозумілою структурою, що допомагає швидко переходити між головними розділами: головна, особисті історії,

аналітика, налаштування користувача та платіжна інформація. Користувачі мають змогу переглядати й прослуховувати записи, активувати автовідтворення, переглядати витрати, додавати або змінювати платіжні дані та керувати налаштуваннями підкористувачів та обліковим записом.

Зручна навігація, функція автоматичного відтворення, взаємодія з сервісами TTS та ASR, а також доступ до розширеної статистики роблять інтерфейс ключовим інструментом керування усіма функціями системи. Завдяки цьому користувачі мають в своєму розпорядженні повноцінний, візуально приємний та функціональний інструмент для ефективної роботи з голосовим помічником.

2.3 Основні вимоги щодо архітектури системи

Для забезпечення високої продуктивності, зручності та стабільної роботи з голосовими запитами, архітектура системи розроблена з урахуванням на оптимізації для платформи Android. Для досягнення цих цілей було реалізовано модульну структуру, яка забезпечить ефективну обробку голосових команд, інтеграцію з сучасними API для розпізнавання мовлення та синтезу голосу, а також непомітний перехід між ключовими компонентами системи [1]. Загальна схема архітектури застосунку наведена на рисунку 2.3.

Основні модулі, серед яких голосова активація, транскрибування, обробка запитів та управління користувачами, взаємодіятимуть через чітко окреслені API, що гарантують високу швидкість роботи та мінімальне споживання ресурсів пристрою.

Завдяки модульності існує можливість легко масштабувати систему, пристосовувати її до нових потреб та функцій без суттєвої зміни платформи, що забезпечує стабільність та ефективність, навіть якщо можливості апаратного забезпечення обмежені.

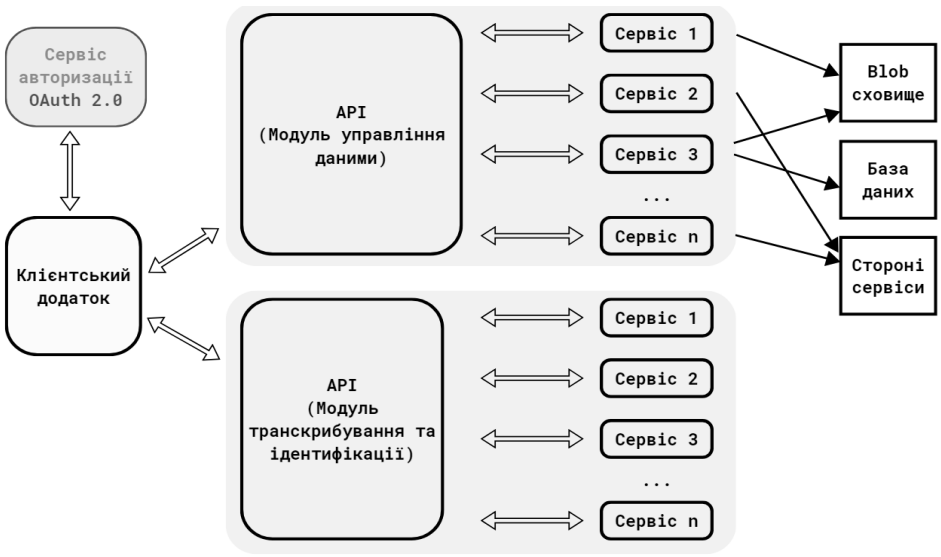

Рисунок 2.3 – Схема архітектури застосунку

Висновки до розділу 2

В процесі архітектурного планування застосунку, для забезпечення цілісності реалізації серверної та клієнтської частин, було обрано платформу на базі ASP.NET Core, доповнену гнучкими рішеннями на Python для швидкого прототипування й інтеграції модуля, що використовує моделі ШІ. C#, з Застосування його суворою типізацією, асинхронним підходом та кросплатформністю, що забезпечується .NET MAUI, гарантує високу продуктивність, надійність і зручність обслуговування, тоді як Python-модуль забезпечує швидкий розвиток та масштабування обробки аудіо.

Для зберігання файлів використовується Azure Blob Storage, а в якості бази даних обрано Azure Cosmos DB, що підтримує документно-орієнтовану модель.

Модульна структура системи, де кожен функціональний блок має свій чіткий обов'язок (від простого прослуховування й реагування на ключові фрази до розшифровки аудіо, перевірки особистості користувача, обробки його запитів та представлення інформації візуально), робить можливим просте додавання нових або заміну старих компонентів, не впливаючи на роботу всієї системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Опис функціоналу системи

Як було зазначено у попередньому розділі, розроблюваний в рамках цього дослідження додаток має втілювати в собі повний спектр функцій голосового помічника, з можливістю налаштування персональних профілів. Окрім цього, зазначений мобільний застосунок голосового асистента наразі орієнтований виключно на роботу українською мовою.

У цій системі реалізовано повний цикл функціонування голосового помічника, що включає можливість створення й керування кількома профілями користувачів. Спочатку відбувається реєстрація основного акаунта: користувач має змогу зареєструватися, використовуючи електронну пошту та пароль, або ж здійснити авторизацію через Google OAuth (Open Authorization – Відкрита авторизація). Після успішної реєстрації система генерує JWT (JSON Web Token – JSON-веб-токен), котрий зберігається у кеші мобільного застосунку – за умови наявності дійсного токена, повторна авторизація відбувається автоматично.

Головний користувач має змогу додавати підкористувачів. Кожен з них отримує свій голосовий профіль, де налаштовуються унікальні стартові, фінальні фрази або тайм-аут очікування. Для голосової ідентифікації проводиться запис голосу користувача, який в подальшому перетворюється у векторне представлення голосу.

Для озвучення відповідей пропонується вибір: базовий голос чи клонування голосу. Клонування може відбуватися на основі завантаженого аудіофайлу або записаного через мікрофон зразка.

Усі голосові взаємодії структуровані в два формати чату: загальний та приватний. Загальний чат доступний для всіх користувачів платформи, він виступає платформою для групового обміну голосовими повідомленнями з інтелектуальним помічником. Особиста ж розмова містить повідомлення виключно конкретного користувача та персонального асистента.

Крім голосового функціоналу, застосунок включає розширені можливості управління системою. У розділі “Користувачі” відображається список підкористувачів, є функція додавання, зміни та видалення учасників. У розділі “Платіжна інформація” можна перевірити баланс, історію транзакцій, активувати автопоповнення та використати калькулятор вартості запиту, який розраховує ціну за внутрішньою формулою, беручи до уваги тривалість аудіо, кількість токенів і символів. Розділ аналітики містить детальну статистику витрат за часовими проміжками та профілями, а у розділі “Історія” зберігаються усі особисті розмови, доступ до яких користувач може захистити паролем.

Додатково передбачено можливості управління аудіозаписами, а саме: старт, зупинка, автоматичне відтворення, а також перехід між окремими повідомленнями.

У системі голосового помічника впроваджено внутрішній алгоритм обробки голосових звернень, що гарантує комфортну та безпечну взаємодію з користувачем. Фундаментом комунікації слугують тригерні фрази, часові обмеження (таймаути) та механізми перевірки ідентичності голосу. Взаємодія починається зі стартової фрази, що активує систему та ініціює голосову сесію. Далі помічник звертається до користувача з фразою, наприклад: «Чим я можу вам допомогти, Ольга?» — і система переходить у режим очікування голосового запиту.

Після цього користувач має можливість поставити питання або сформулювати запит. Завершення мовлення фіксується або за допомогою власної кінцевої фрази, або після завершення таймера очікування (який встановлюється індивідуально для кожного голосового профілю). Як тільки система виявляє завершення звернення, аудіо передається в обробку: діаризація, ідентифікація користувача, транскрипція, обробка та генерація відповіді через ШІ-модель. Далі відповідь озвучується вибраним голосом і повертається користувачу у вигляді аудіо.

Після активації стартовою фразою, користувач може сформулювати питання або висловити свій запит. Кінець розмови визначається або особистою фразою для завершення, або після спрацювання таймера очікування (який налаштовується

окремо для кожного голосового профілю). Як тільки система фіксує завершення звернення, аудіо направляється на обробку: діаризація, ідентифікація користувача, транскрипція, обробка та формування відповіді за допомогою ШІ-моделі. Далі сформована відповідь відтворюється обраним голосом.

У випадку, коли після початку сесії (вимови початкової фрази) відсутня мовленнєва активність або користувача не було ідентифіковано, сесія припиняється після встановленого періоду. У цьому разі система видає повідомлення: “Ваш голос не розпізнано” або “Вас не було ідентифіковано”. Це слугує захистом системи від ненавмисного чи несанкціонованого використання.

Коли користувач надсилає серію запитів один за одним, після кожної відповіді система надає йому до 10 секунд на формування нового питання. Якщо протягом цього часу новий запит не надійде, система переходить у режим очікування, готовий до нової стартової фрази, завершуючи поточну сесію. Це дозволяє асистенту підтримувати оптимальний баланс між активною взаємодією та економією ресурсів, при цьому забезпечуючи персоналізований і безпечний діалог.

Коли користувач, озвучуючи початкову фразу, додає фразу “Особиста розмова” (наприклад, "надай мені відповідь, особиста розмова"), автоматично активується режим приватної бесіди. Візуально це виглядає так: звичний загальний чат зникає, поступаючись місцем порожньому вікну персональної сесії. Це вікно згодом заповнюється виключно повідомленнями між користувачем та віртуальним помічником.

Щойно приватний чат завершиться (внаслідок закінчення розмови або ж спрацювання таймауту, скажімо, після 10 секунд бездіяльності), вікно особистого листування зникає, а система відновлює відображення загального чату, доступного для всіх користувачів чи учасників простору.

Для кращого розуміння роботи системи, було створено діаграму прецедентів, яка демонструє основні взаємодії між користувачами та системою, відображаючи ключові функції та сценарії її роботи. Її наведено на кресленику ІК12.170БАК.006 Д1.

Крім того було розроблено діаграму діяльності, що ілюструє послідовність дій і процесів, що виконуються в системі для досягнення певної мети. Дану діаграму наведено на кресленику ІК12.170БАК.006 Д2.

Отже, впроваджена система голосового помічника об'єднує в собі гнучкість, захищеність та індивідуальний підхід, надаючи комфортну взаємодію для кількох користувачів одночасно. Завдяки механізмам розпізнавання мовлення, ідентифікації профілів, підтримці приватних та загальних сеансів, а також адаптивним таймерам і тригерним фразам, система забезпечує чіткий та контрольований процес спілкування.

3.2 Опис архітектури системи

Система побудована на трьох ключових елементах, котрі взаємодіють один з одним, керуючись принципами модульності та розподіленої архітектури:

– клієнтський застосунок – реалізований за допомогою .NET MAUI. Він представляє собою зручний інтерфейс та слугує стартовою точкою для взаємодії з системою. Здійснює відправлення запитів до API, щоб отримати доступ до усіх функцій системи.

– API-сервіс побудовано на базі платформи ASP.NET Core, його реалізовано з використанням архітектури Onion для забезпечення чіткого розмежування шарів та CQRS (Command and Query Responsibility Segregation - розділення відповідальності команд та запитів) для відокремлення операцій запитів від операцій команд. Він відповідає за обробку бізнес-логіки, взаємодію з базами даних та файловими сховищами, а також за виконання команд та обробку запитів.

– сервіс для обробки аудіо – це незалежний мікросервіс, що базується на Python. Він здійснює діаризацію аудіо, який надходить, транскрибує мовлення та ідентифікує особу користувача.

Система втілена запринципами мікросервісної архітектури, де кожен із трьох основних компонентів розгортається, масштабується та оновлюється окремо, а їх взаємодія відбувається через стандартизовані REST-інтерфейси [1, 7].

Детальний розбір кожної складової системи наведено у наступних підрозділах, а їхню взаємодію та використані технології можна розглянути на рисунку 3.1.

Мікросервісна архітектура - це підхід до створення програмного забезпечення, де застосунок розбивається на сукупність маленьких, незалежних служб. Ці служби спілкуються між собою через чіткі інтерфейси. Для цього найчастіше використовують протоколи HTTP, REST API, тощо.

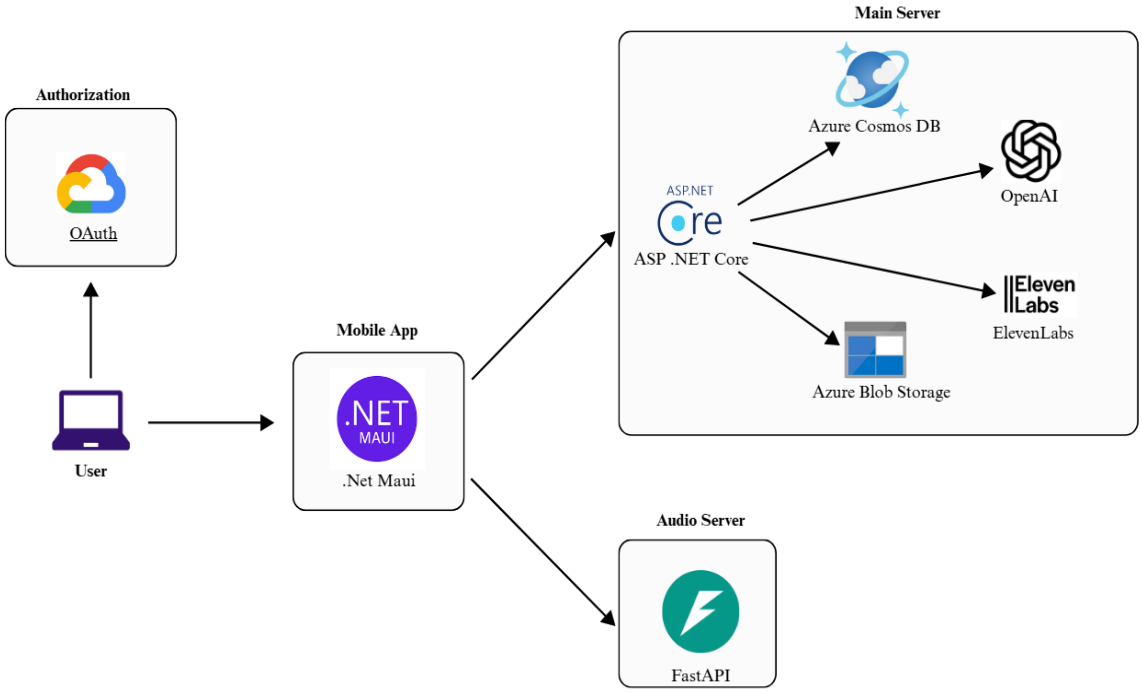

Рисунок 3.1 – Архітектура мобільного застосунку

3.3 Огляд програмної реалізації додатку-клієнта

У клієнтському додатку використано архітектурний шаблон MVVM (Model–View–ViewModel), що набув широкого розповсюдження в сучасній розробці інтерфейсних програм. Головна його задача - розподіл обов'язків між відображенням інформації (View), логікою взаємодії (ViewModel) та джерелом даних (Model). Такий спосіб дозволяє створити структурований, зрозумілий і легкий в обслуговуванні код [4].

Використання MVVM сприяє підвищенню зрозумілості програми, полегшує процес розробки, тестування та майбутнього розширення функціоналу. Завдяки чіткій організації складових, зменшується ступінь взаємозалежності між графічним інтерфейсом та бізнес-логікою, що є особливо актуальним для великих або складних проєктів. Це забезпечує гнучкість архітектури, спрощує внесення змін, повторне використання коду та знижує ймовірність помилок при модифікації окремих частин системи.

Основні складові MVVM:

– Model – представляє класи, що визначають бізнес-інформацію та основні об'єкти конкретної предметної області. Ці класи не залежать від користувацького інтерфейсу чи алгоритмів взаємодії.

– View – це вигляд інтерфейсу визначається файлами XAML. Ці файли містять лише розмітку, але не логіку. З ViewModel пов'язуються через властивість BindingContext, яка встановлює прив'язки між елементами інтерфейсу та властивостями і командами ViewModel.

– ViewModel – головна частина логіки взаємодії. ViewModel містить властивості, команди (ICommand) і обробники подій, які реагують на зміни у Model та оновлюють інтерфейс за допомогою механізму двостороннього зв'язування (data binding). Також реалізує інтерфейс INotifyPropertyChanged для відстеження змін стану.

На додаток до базової структури MVVM, у застосунку реалізовано два додаткові шари, що покращують модульність та розширюваність проєкту:

– Contracts шар — цей шар містить моделі, які встановлюють правила взаємодії між різними компонентами програми, наприклад для зв'язку між ViewModel та сервісами.

– Application шар — тут втілюється прикладна логіка, яка регулює взаємодію між ViewModel, Model та зовнішніми сервісами.

Отже, архітектура MVVM в .NET MAUI з додаванням шарів Contracts та Application створює ще більш зрозуміліший поділ обов'язків. Це полегшує процес розробки, покращує зчитування коду і дає можливість легко розширювати

функціонал застосунку. Загальна схема взаємодії між Model, View і ViewModel показана на рисунку 3.2.

Додатково, для покращення користувацького досвіду, реалізовано механізми кешування даних, що дозволяють зменшити кількість звернень до сервера та пришвидшити завантаження інтерфейсу.

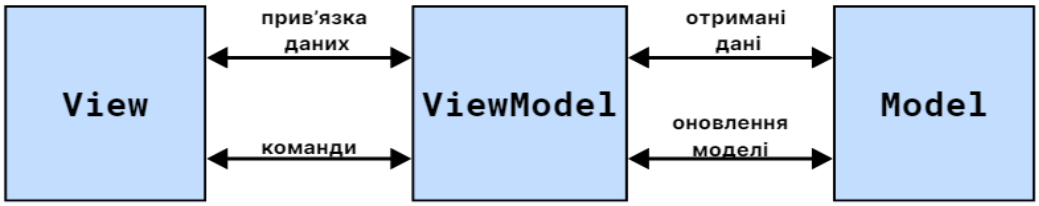

Рисунок 3.2 – Шаблон проєктування MVVM

3.4 Серверні рішення

Система складається з двох незалежних серверних рішень. Кожне з них має чітко окреслену функцію. Ці рішення можуть розгортатися, масштабуватися та оновлюватися автономно.

Першим із сервісів виступає мікросервіс для обробки аудіо, розроблений на Python з використанням FastAPI. Він приймає HTTP-запити, що містять необхідні дані, виконує діаризацію та валідацію метаданих, після чого передає дані для транскрибації мовлення і здійснює ідентифікацію користувача на основі голосового відбитку. FastAPI гарантує високу продуктивність та асинхронність, а автоматично згенерована документація OpenAPI полегшує інтеграцію.

З огляду на теоретичні засади та моделі ШІ, викладені в підрозділі 2.2.2, в цьому розділі проаналізуємо ключові етапи опрацювання голосового запиту користувача. Система може функціонувати у різних режимах залежно від отриманих вхідних даних, таких як: тривалість очікування, заключна фраза та індикатор початку діалогу. Алгоритм обробки передбачає наступну послідовність дій:

a) На першому етапі визначається, чи було передано час очікування, кінцеву фразу, а також чи є звернення першим у поточній розмові. Якщо це перше звернення, користувач має почати говорити одразу, без затримки. Якщо ж звернення не є першим, користувачу може надаватися обмежений проміжок часу (наприклад, 10 секунд) для формування запиту, без необхідності повторно вимовляти стартову фразу.

b) Обробка згідно з часом очікування. Якщо задано час очікування, то кожна секунда отриманого по WebSocket аудіо порівнюється з голосовим відбитком користувача. Якщо протягом вказаного часу не було виявлено присутності голосу користувача, прослуховування вважається завершеним.

c) Якщо було обрано кінцеву фразу замість часу очікування, то система очікує її появи в потоці мовлення для завершення прослуховування.

d) Отримане аудіо передається на діаризацію, в результаті чого воно розбивається на сегменти, що відповідають окремим користувачам.

e) Сегменти, отримані на попередньому етапі, порівнюються з голосовим відбитком користувача для виявлення належних йому фрагментів мовлення.

f) Всі ідентифіковані сегменти об’єднуються в єдиний аудіофайл. Якщо таких сегментів не виявлено, обробка припиняється і користувачу повідомляється про завершення сесії.

g) На завершальному етапі відібране аудіо передається для транскрибування з подальшою обробкою тексту.

Endpoint (кінцева точка зв’язку) — це конкретні URL-адреси на сервері, через які клієнт, такий як веб-додаток, мобільна програма або інша служба, має змогу надсилати запити для отримання або передачі даних. Кожен маршрут відповідає за виконання певної функції та обробку конкретного виду інформації.

Застосунок включає такі ключові endpoints маршрути:

1. /ws — обробляє з'єднання WebSocket для прийому аудіо в режимі реального часу. Цей шлях застосовується для відправлення аудіо від клієнта до сервера в процесі голосового спілкування.

2. /embedding — відповідає за обробку векторних представлень голосу (voice embeddings). Через цей endpoint відбувається перетворення голосових даних користувача на векторний формат для подальшої ідентифікації.

3. /tokenizer — опрацьовує запити стосовно токенізації тексту, а саме надає можливість дізнатися про кількість токенів, використаних для обробки запиту в OpenAI.

Друге рішення — це API-сервіс, розроблений на базі .NET Core, із застосуванням архітектурного підходу Onion та шаблону проектування CQRS.

На рівні Presentation (Controllers) відбувається обробка REST-запитів, які перетворюються у команди чи запити відповідного типу, далі передаючись для подальшого опрацювання шаром Application.

Application шар містить всю бізнес-логіку, реалізовану через окремі обробники, відповідальні за обробку команд та запитів.

Domain шар включає в себе сутності, об'єкти зі значеннями та бізнес-правила, які визначають функціональність системи.

Взаємодія з інфраструктурою, такою як база даних (через Entity Framework Core) та файлове сховище, реалізована в шарі Infrastructure.

Застосування CQRS забезпечує чітке розділення операцій читання та запису, що веде до покращення продуктивності, можливості масштабування та полегшення підтримки системи [2]. Для маршрутизації запитів використовується бібліотека MediatR, яка зменшує зв'язок між компонентами та спрощує впровадження шаблону.

Вся взаємодія між шарами є чітко розмежованою, що відповідає принципам архітектури Onion, забезпечуючи слабке з'єднання та зручність під час тестування. [14] На рисунку 3.3 представлено загальну структуру архітектури рішення. Такий підхід до побудови сприяє чіткому розподілу відповідальностей і дозволяє легко масштабувати окремі компоненти.

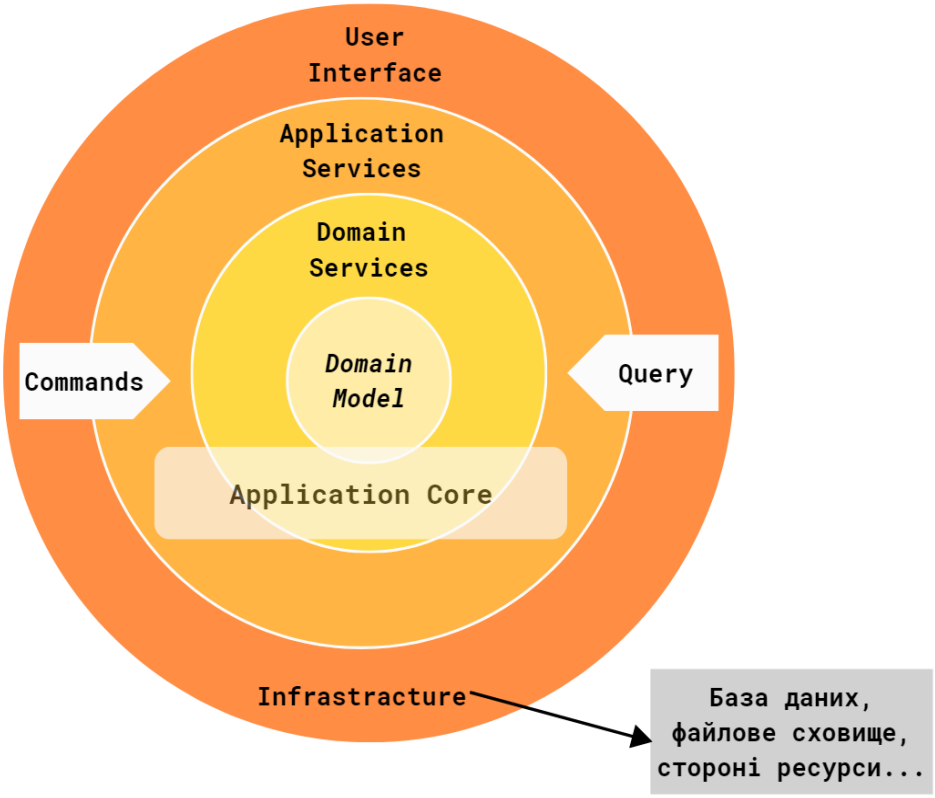

Рисунок 3.3 – Onion архітектура з CQRS шаблоном

API-сервіс містить значну кількість різноманітних endpoints, через які відбувається взаємодія з користувачами та іншими сервісами. Згідно з рисунком 3.4, кожна кінцева точка виконує конкретну функцію. Така організація дозволяє розмежувати обробку різних типів запитів, тим самим збільшуючи гнучкість та масштабованість API-сервісу. Кожна з цих точок є ключовою для клієнтських застосунків, забезпечуючи ефективну роботу всієї системи в цілому.

Кожен endpoint має чітко визначений HTTP-метод і маршрут, що забезпечує зручність і передбачуваність взаємодії з сервісом. Документація API у форматі OpenAPI/Swagger дозволяє розробникам швидко ознайомитися зі структурами запитів та відповідей, спрощуючи інтеграцію клієнтських застосунків. Версіонування endpoint-ів гарантує зворотну сумісність під час оновлення

функціоналу, що підвищує стабільність і надійність сервісу.

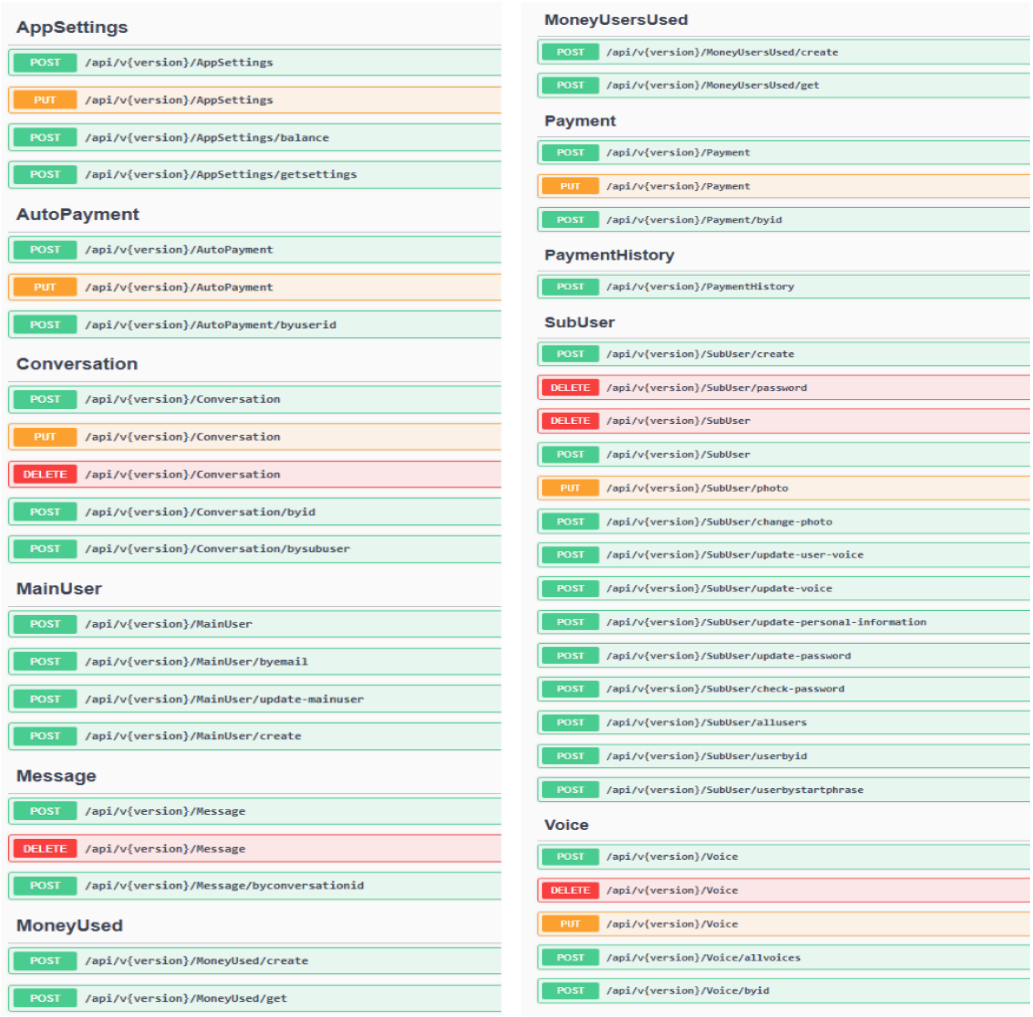

Рисунок 3.4 – Endpoints API-сервісу

3.5 База даних

Згідно розділу 2.1.4, зберігання даних реалізовано в Azure Cosmos DB, яка є високоефективною, що масштабується та є глобально розподіленою NoSQL базою даних. Обрання цього рішення було вмотивовано потребою у зберіганні гнучких, частково структурованих даних у форматі JSON. Це надає зручний спосіб для управління метаданими, що стосуються користувачів системи, результатів їхньої взаємодії, а також параметрів функціонування підсистем та іншого.

На кресленику ІК12.170БАК.006 Д4 зображено структурну схему бази даних, яка демонструє логічну організацію інформації всередині системи. Зокрема, тут видно структуру таблиць, котрі використовуються для зберігання даних

користувачів та метаданих у форматі JSON. До того ж, на кресленнику позначено тип кожного поля, що дає змогу чітко встановити формат і обмеження для зберігання даних.

У цій конструкції застосовано зв'язки «багато до одного» та «один до одного», що сприяє логічному поєднанню взаємозалежних даних. Це надає змогу ефективно управляти взаємозв'язками між різними елементами системи, підтримуючи цілісність даних та простоту доступу до них.

3.6 Аворизаційне рішення

У системі передбачено два способи авторизації користувачів, що дає змогу гнучко та зручно потрапляти в додаток.

Перший варіант авторизації використовує електронну пошту та пароль. Під час реєстрації користувач створює аккаунт, вказуючи коректну електронну адресу і унікальний пароль. Цей пароль зберігається у базі даних у зашифрованому вигляді задля забезпечення безпеки. Після успішної реєстрації чи входу, формується JWT-токен (JSON Web Token), котрий надсилається клієнту. Цей токен застосовується для автентифікації у наступних запитах до захищених ресурсів.

Другий варіант авторизації – вхід через Google-акаунт, використовуючи протокол OAuth. Інтерфейс системи має кнопку швидкого входу, натискання на яку переадресовує користувача на сторінку аутентифікації Google. Після перевірки електронної адреси користувач автоматично повертається до застосунку. Якщо користувач заходить вперше (реєстрація), йому буде запропоновано завершити створення облікового запису, встановивши власний пароль. У випадку повторного входу система автоматично перенаправляє на головну сторінку. Як і у першому варіанті, після успішної авторизації клієнт отримує JWT-токен, котрий використовується для доступу до захищених ресурсів.

Система пропонує два варіанти авторизації: вхід з використанням електронної пошти та паролю, а також швидку автентифікацію через Google-акаунт, що працює за протоколом OAuth. Обидва методи гарантують надійний

захист особистої інформації користувача та передбачають генерацію JWT-токена для подальшої взаємодії з ресурсами, що потребують авторизації. Ця гнучкість дає змогу користувачам вибирати найбільш відповідний для них спосіб входу.

Висновки до розділу 3

У третьому розділі ми розглянули практичне втілення програмної системи, зокрема архітектурні рішення, логіку взаємодії складових та використані технології в процесі розробки. Детально описано створення системи на основі мікросервісної архітектури, що дозволяє гарантувати гнучкість, масштабованість та незалежність під час розгортання окремих частин проєкту.

Клієнтська програма втілена на платформі .NET MAUI, згідно з шаблоном MVVM. Це забезпечує розмежування обов'язків між шарами: інтерфейсу, логіки та даних, роблячи структуру зрозумілою.

Серверна сторона поділена на дві окремі частини. Аудіо-сервіс розроблено на Python, з використанням FastAPI, для обробки голосових даних. API-сервіс створено на .NET Core з архітектурою Onion та патерном CQRS. Він відповідає за бізнес-логіку, доступ до даних, обробку запитів та команд. Такий підхід сприяє підтримці коду та полегшує масштабування системи, підвищуючи її продуктивність.

Розроблений мобільний додаток, що діє як голосовий помічник, на поточний момент працює тільки з українською мовою. Втім, у планах передбачено розширення можливостей програми, аби додати підтримку й інших мов. Це дозволить значно збільшити коло користувачів.

Було розглянуто структурну схему БД і реалізацію авторизаційного механізму, який включає два варіанти — класичний вхід через email і пароль, а також авторизацію через Google OAuth.

Загалом, впроваджена система відповідає актуальним стандартам щодо продуктивності, захищеності та здатності до розширення, гарантуючи стабільну платформу для обробки запитів користувачів та взаємодії з голосовими сервісами.

4 РОБОТА КОРИСТУВАЧА ІЗ ЗАСТОСУНКОМ

4.1 Інструкція користувача

Щоб почати використання мобільної програми, потрібно увійти у систему. Відразу після запуску користувач опиняється на стартовій сторінці застосунку, яка є сторінкою авторизації (рисунок 4.1).

У випадку, коли користувач вже авторизувався, і токен доступу ще зберігається в кеші програми, цей етап буде пропущено автоматично, а користувач буде переадресований на головний екран програми.

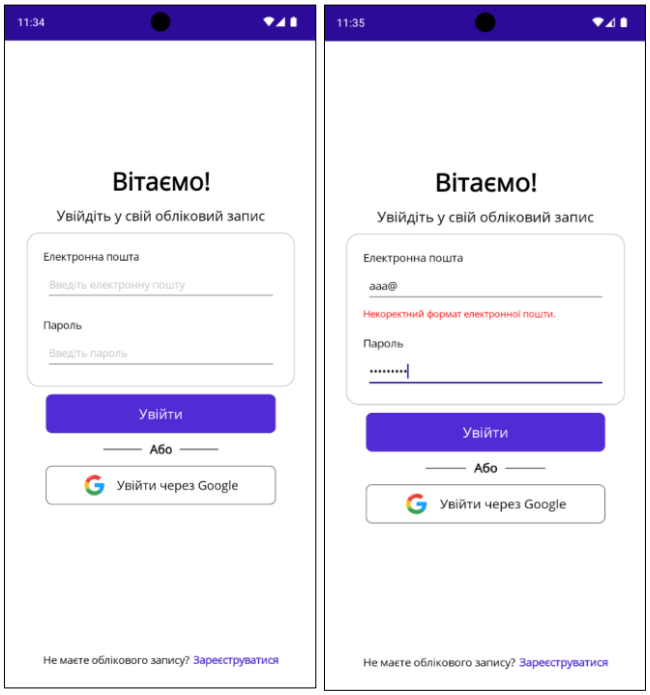

Рисунок 4.1 – Сторінка авторизації

Якщо ж користувач ще не зареєстрований, то він має можливість зареєструватися (рисунок 4.2). Якщо було обрано реєстрацію за допомогою google акаунту, то користувач має завершити реєстрацію, створивши пароль до особистого профілю (рисунок 4.3).

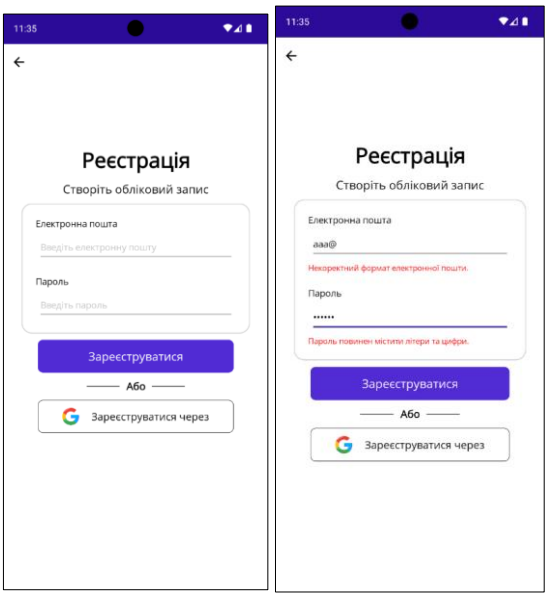

Рисунок 4.2 – Сторінка реєстрації

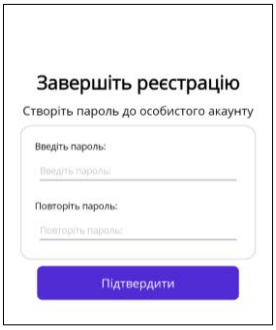

Рисунок 4.3 – Сторінка завершення реєстрації

Після успішної авторизації (або реєстрації) користувач потрапляє на головний екран (рисунок 4.4). Тут розміщено загальний чат, у якому відображаються повідомлення від усіх учасників та відповіді діалогового агента їм. Користувач може прослуховувати будь-яке, скористатися панеллю керування аудіо та увімкнути режим автовідтворення.

Для початку або зупинки спілкування з діалоговим асистентом потрібно скористатися кнопкою Старт/Стоп, тоді він перейде у режим пасивного прослуховування при якому очікується активаційна фраза від користувача. Система сама визначає, коли діалог з даним користувачем завершується, тому не потрібно постійно натискати на кнопку.

Детальна інформація про алгоритм спілкування наданий в підрозділі 3.1.

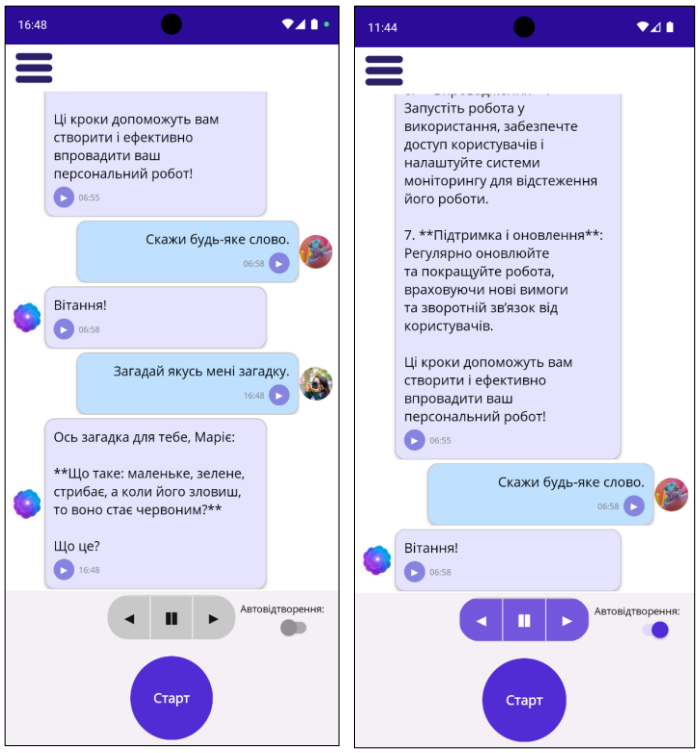

Рисунок 4.4 – Сторінка загального чату

Щоб розпочати особисту розмову, користувач каже свою стартову фразу з доповненням “особиста розмова”. Наприклад, у користувача є фраза “Дай мені відповідь”, тому для переходу в приватний чат, йому потрібно сказати “Дай мені відповідь. Особиста розмова”

Після активації особистої розмови, на інтерфейсі застосунку зникає загальний чат, адже створився новий, готовий до подальшого обміну повідомленнями.

Після натискання на кнопку у верхній частині інтерфейсу, яка відкриває меню (рисунок 4.5), користувачеві стає доступна навігація всередині програми. В цьому розділі можна побачити актуальний баланс користувача, а також зручні посилання для швидкого переходу до ключових розділів: Головна сторінка, Користувачі, Налаштування, Аналітика та Історія.

Інтерфейс меню було створено, керуючись принципами зручності та зрозумілості. Це гарантує миттєвий доступ до найважливіших функцій, не вимагаючи складного переходу між розділами.

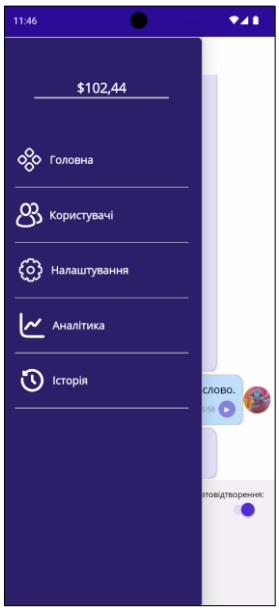

Рисунок 4.5 – Сторінка меню

Перейшовши до розділу “Головна” відобразиться загальний чат, який було зображено на рисунку 4.4.

У розділі “Користувачі” відображається перелік всіх підкористувачів (голосових профілів) (рисунок 4.5). Тут присутня кнопка для переходу на сторінку додавання нового учасника. Крім того, можна оновити дані існуючого користувача, натиснувши на одного зі списку.

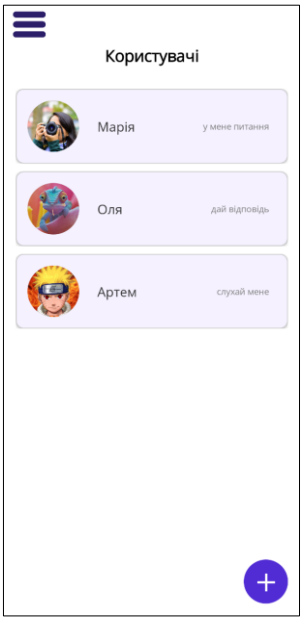

Рисунок 4.8 – Сторінка користувачів

На сторінці створення користувача, інтерфейс, якої поданий на рисунку 4.9, потрібно вказати основні дані.

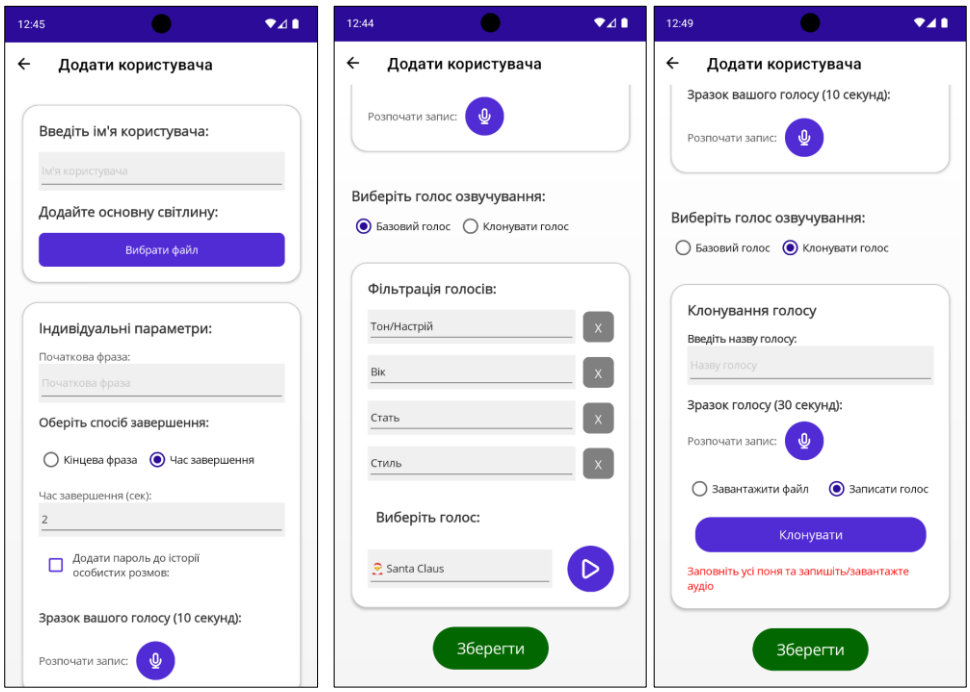

Рисунок 4.9 – Сторінка додавання нового користувача

По-перше, потрібно вказати ім’я та завантажити фотографію для даного голосового профілю, для кращої навігації.

По-друге, потрібно задати стартову фразу, за допомогою якої можна буде активувати спілкування з діалоговим агентом. Крім того, необхідно вказати яким способом буде визначення завершення прийому запиту від користувача. Перший варіант, це кінцева фраза, а другий варіант це час очікування, тобто час при якому голос користувача не було розпізнано.

Крім того, за бажанням — можна встановити пароль для доступу до особистої історії спілкування.

Також користувач обирає голос озвучення, він може бути або базовим, або клонованим. Якщо користувач обрав перший варіант, то для зручності доступна фільтрація за тоном/настроєм, віком, статтю та стилем, що дозволяє користувачеві робити вибір швидше і без потреби переслухувати всі голоси, що гарно впливає на користувацький досвід (рисунок 4.10). Якщо ж було обрано другий, то потрібно

передати системі аудіозапис голосу для подальшого клонування, бажано щоб він був без зайвих шумів та голосів, адже через ці звуки, результат може бути не таким, який очікувався. Передати запис можна двома способами, або записати зразок голосу через мікрофон в застосунку, або завантажити вже готовий аудіофайл.

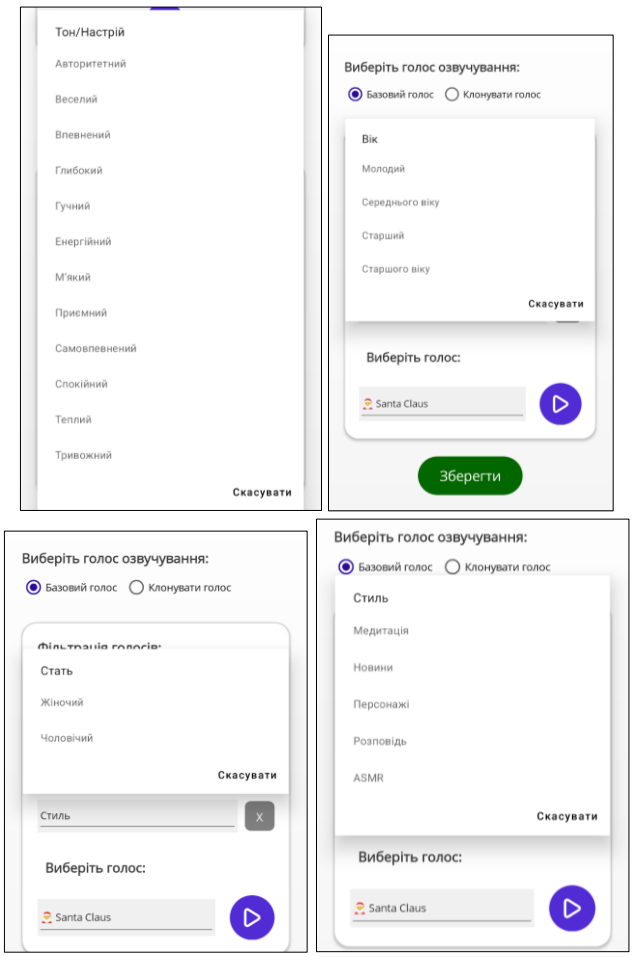

Рисунок 4.10 – Фільтри базових голосів

Зрештою, останній параметр, що необхідний при створені підкористувача, це зразок особистого голосу, натиснувши на кнопку запису, потрібно протягом десяти секунд говорити. Цей зразок необхідний для подальшої ідентифікації користувача та перевірки його доступу.

Усі поля мають валідацію: без заповнення обов’язкових пунктів або при некоректному форматі вводу, форма не дозволить завершити створення нового користувача (рисунок 4.11).

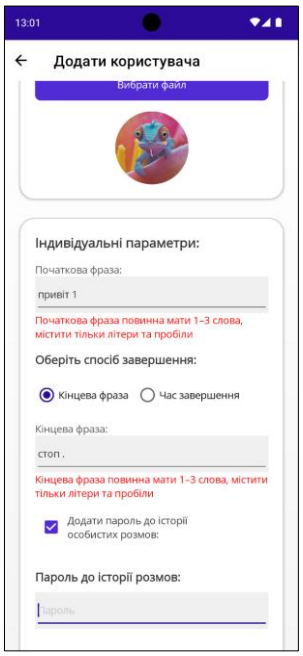

Рисунок 4.11 – Валідація при додаванні нового підкористувача

Перейшовши на сторінку оновлення профілю підкористувача, з’являється можливість оновити будь-який параметр, який був встановлений при створені користувача (рисунок 4.12).

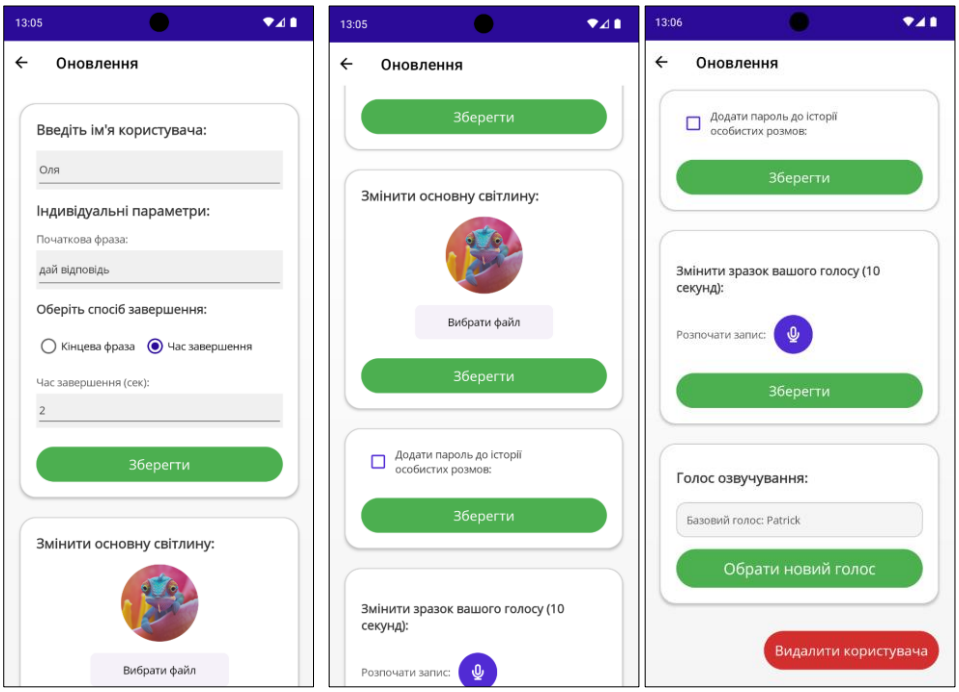

Рисунок 4.12 – Сторінка оновлення користувача

Крім того тут доступна функція видалення голосового профілю через відповідну кнопку. Інтерфейс налаштувань розроблено так, щоб усі зміни було легко виконувати, а користувачі могли швидко орієнтуватися у доступному функціоналі.

Якщо користувач раніше встановив пароль для особистих розмов, але хоче його видалити, достатньо зняти позначку та зберегти зміни. Після цього з'явиться модальне вікно, у якому потрібно підтвердити свою особу, ввівши чинний пароль, який користувач хоче вимкнути (рисунок 4.13).

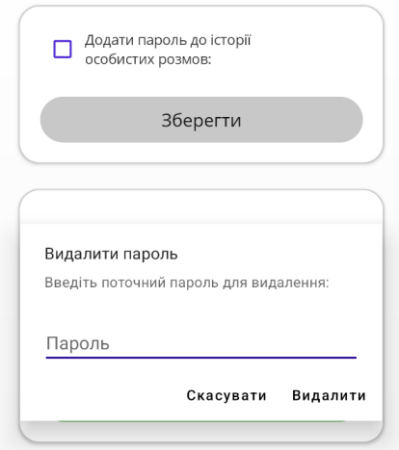

Рисунок 4.13 – Модальна сторінка для підтвердження особи

Після оновлення профілю, користувач отримає повідомлення про підтвердження змін або помилку.

У розділі «Налаштування» (рисунок 4.14) відображається інформація про обліковий запис користувача, а також надаються основні інструменти для керування обліковими даними. Користувач має доступ до таких опцій: зміна теми оформлення інтерфейсу, оновлення пароля, вихід з облікового запису, перегляд поточного балансу коштів, а також перехід на сторінку «Платіжна інформація», де можна переглядати збережені платіжні реквізити або редагувати їх за потреби.

У випадку спроби виходу з облікового запису система обов’язково запитує підтвердження цієї дії. Такий механізм захищає від випадкового або ненавмисного завершення сеансу, що може призвести до втрати прогресу в роботі або

переривання важливих процесів, зокрема під час здійснення платежів чи активної взаємодії із системою.

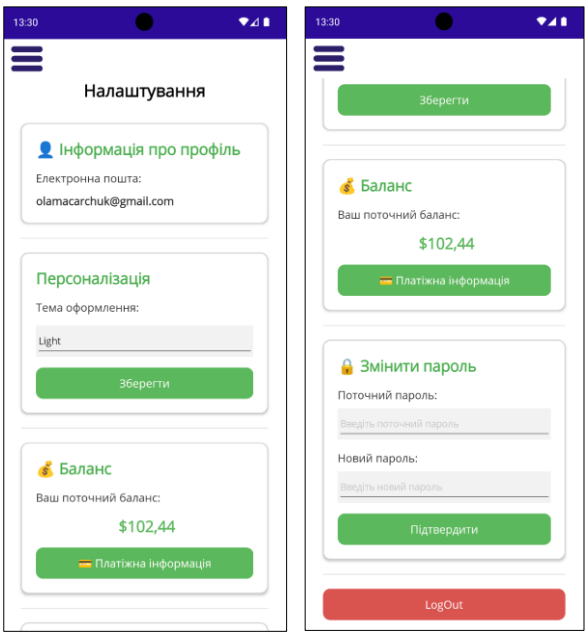

Рисунок 4.14 – Сторінка оновлення облікового запису

На сторінці «Платіжна інформація» (рисунок 4.15) користувач має змогу керувати своїми платіжними методами та контролювати стан балансу облікового запису. Зокрема, передбачено можливість додавання нової платіжної картки, редагування вже збереженої або повне її видалення. Усі операції супроводжуються перевіркою даних на валідність і підтвердженням дій, що забезпечує безпеку та запобігає помилкам.

Крім цього, доступна функція автоматичного поповнення балансу. Користувач може задати мінімальний поріг залишку коштів, за досягнення якого система автоматично ініціює поповнення на вказану суму з прив’язаної картки. Це дозволяє уникнути ситуацій, коли через недостатній баланс призупиняється обробка запитів або надання послуг.

Також передбачена можливість ручного поповнення балансу. Користувач може в будь-який момент ввести бажану суму та здійснити оплату, обравши одну з доступних платіжних карток.

З метою зручності та прозорості користувач має змогу переглядати історію змін платіжних методів та транзакцій, що підвищує довіру до системи та дозволяє легко контролювати фінансові операції.

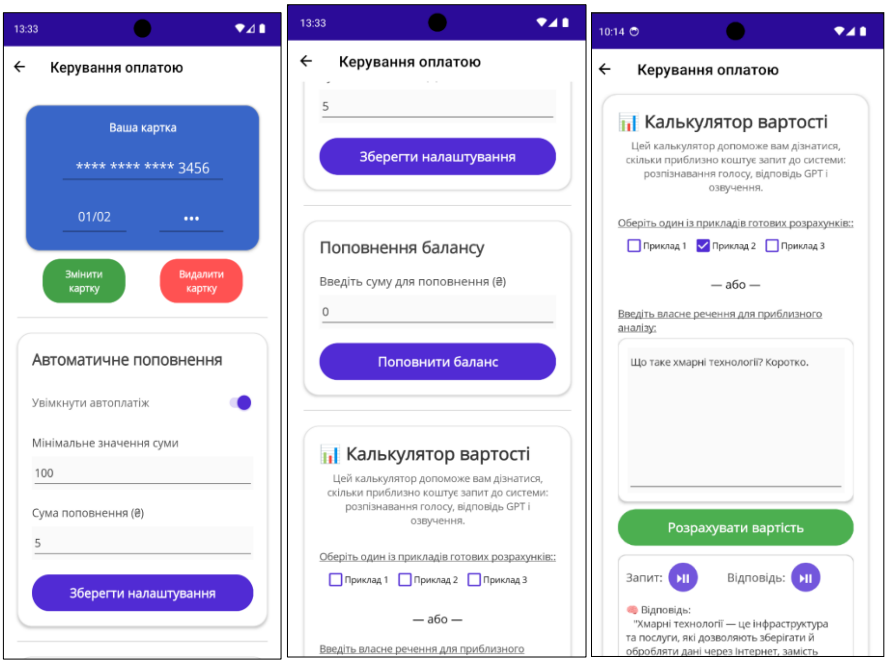

Рисунок 4.15 – Сторінка платіжної інформації

Крім того, на сторінці «Платіжна інформація» доступний калькулятор вартості, який дозволяє попередньо розрахувати вартість запиту та побачити детальну інформацію про складові оплати.

Користувач може вручну ввести власний запит у спеціальне поле калькулятора, щоб дізнатися приблизну вартість його обробки та відповіді (рисунок 4.16). Або можна переглянути один із трьох прикладів реальних запитів з уже розрахованою точною вартістю — це дає змогу краще зрозуміти, як формується ціна, та заздалегідь оцінити потенційні витрати перед використанням системи (рисунок 4.17). Кожен приклад містить не лише текстову інформацію, але й аудіо-звернення користувача та відповідь системи у форматі аудіо. Це дозволяє отримати повне уявлення про якість сервісу, швидкість обробки запиту та структуру результату, який надається у відповідь.

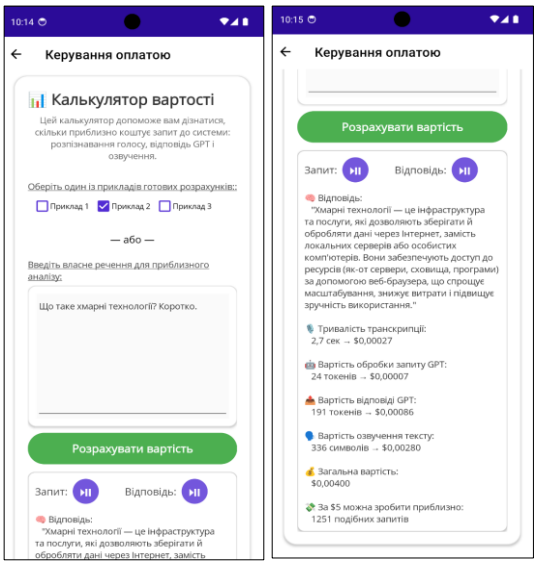

Рисунок 4.16 – Приклад реального запиту для розрахунку вартості

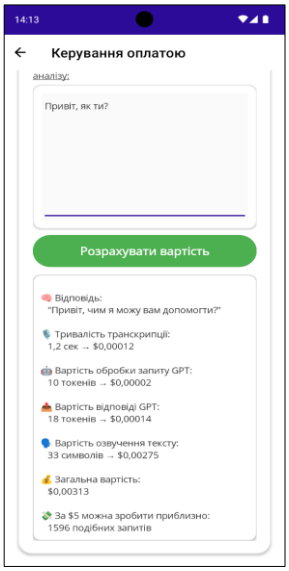

Рисунок 4.17 – Приклад власного запиту для розрахунку вартості

Для визначення вартості обробки запиту був розроблений окремий алгоритм. Діаграма алгоритму наведена на кресленику ІК12.170БАК.006 Д3.

Залежно від типу запиту (реального або приблизного) використовується відповідний метод визначення тривалості аудіо. На наступному етапі визначається кількість токенів, що містяться у запиті до OpenAI, а також у його відповіді. Далі виконується підрахунок кількості токенів, використаних для транскрибаї аудіо. Після цього обчислюється кількість символів у згенерованій текстовій відповіді.

Оскільки система має інтеграцію зі сторонніми сервісами, такими як OpenAI та ElevenLabs, у процесі обробки враховуються такі показники вартості:

– надсилання аудіо на транскрипцію – $6 \cdot 10^{-3}$ $/хв,

– отримання транскрибованого аудіо – 10^{-5} $/токен,

– надсилання запиту до OpenAI – $2 \cdot 10^{-6}$ $/токен,

– отримання відповіді від OpenAI – $8 \cdot 10^{-6}$ $/токен,

– генерація голосового озвучення відповіді – $83{,}3 \cdot 10^{-6}$ $/символ;

Таким чином, на основі наведеного алгоритму, поданого на кресленику, можна сформулювати загальну формулу (4.1) для розрахунку вартості обробки запиту в системі голосової взаємодії. Ця формула враховує всі основні параметри запиту.

Крім того, було додано коефіцієнт, яким буде відбувати регулювати підвищення ціни з часом.

$$V(\text{запит}) = (2a + 8b + 6d \cdot 10^3 + 10e + 83.3c) \cdot f 10^{-6}, \qquad (4.1)$$

де:

– a – кількість токенів у запиті,

– b – кількість токенів у відповіді,

– c – кількість символів у відповіді,

– d – тривалість аудіо запиту,

– e – кількість токенів у транскрибованому аудіо,

– f – коефіцієнт коригування вартості;

У розділі “Аналітика” відображається інформація про використання коштів користувачами (рисунок 4.18). Інтерфейс поділений на три вкладки:

– перша вкладка містить діаграму та список усіх користувачів із зазначенням суми витрачених коштів кожним із них.

– друга вкладка відображає загальну статистику витрат з можливістю фільтрації даних за поточний рік, місяць або тиждень. у третій

– у третій вкладці представлено історію оплат — перелік усіх транзакцій із зазначенням дати, суми, платіжної карти та опису операції (поповнення чи автопоповнення).

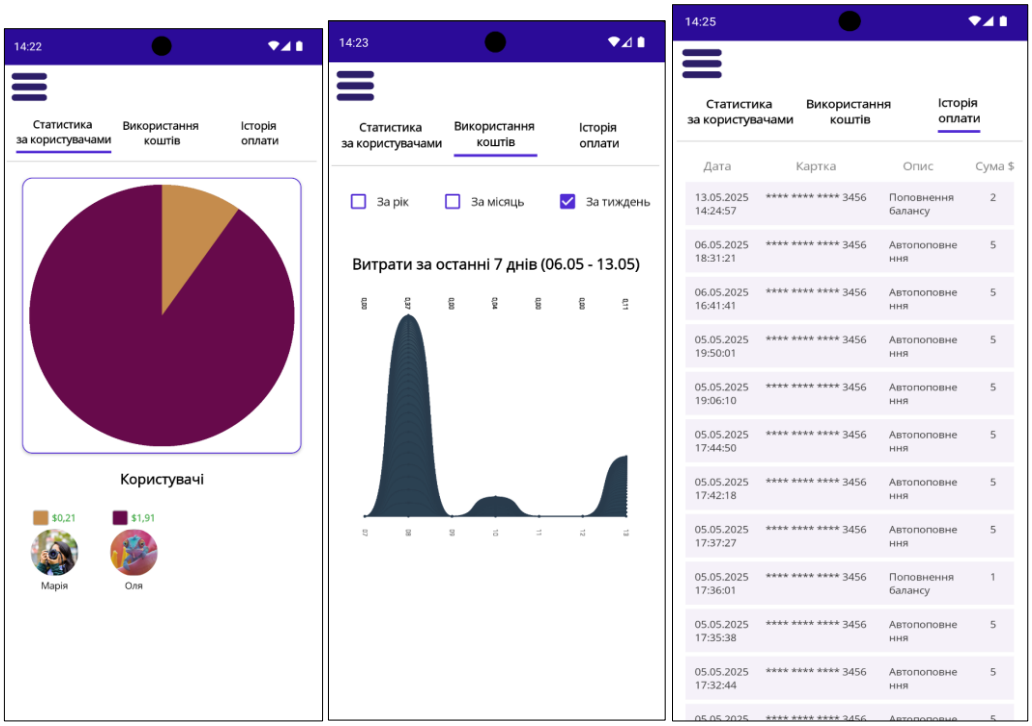

Рисунок 4.18 – Розділ аналітики

Щоб перейти до розділу “Історія”, спочатку потрібно обрати користувача, по якому здійснюється перехід і, за необхідності, потрібно ввести пароль (рисунок 4.19). У даному розділі містяться усі особисті чати конкретного користувача (рисунок 4.20).

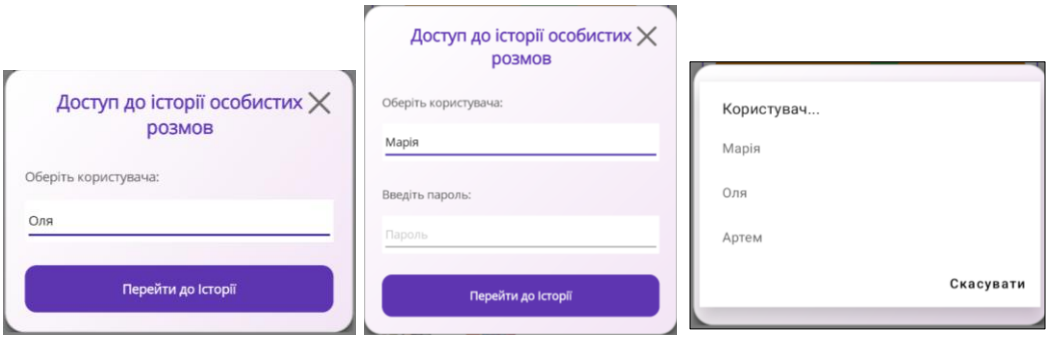

Рисунок 4.19 – Модальне вікно для доступу до особистих розмов

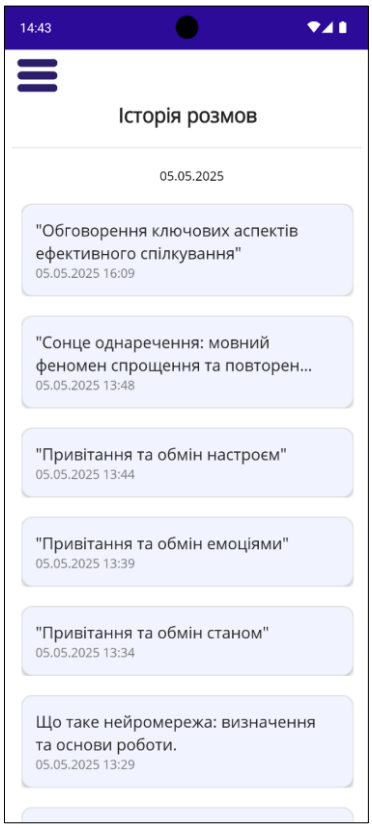

Рисунок 4.20 – Сторінка історії особистих розмов

Клікнувши на одну з розмов на сторінці історії особистих розмов, користувач переходить до перегляду повідомлень у межах цієї розмови (рисунок 4.21). Інтерфейс подібний до загального чату: кожне повідомлення можна прослухати, є можливість переходити до наступного або попереднього повідомлення, ставити відтворення на паузу, а також скористатися функцією автовідтворення.

Також у застосунку реалізовано кнопку для видалення особистої розмови — при її натисканні необхідно підтвердити видалення.

Окрім цього, доступна опція для продовження розмови. Це дозволяє користувачеві повернутися до попереднього спілкування та продовжити його з урахуванням контексту минулої взаємодії. Така можливість є важливою, адже вона забезпечує більш природний, безперервний діалог і знижує потребу в повторному введенні вже згаданої інформації. Це значно підвищує зручність використання системи та економить час користувача.

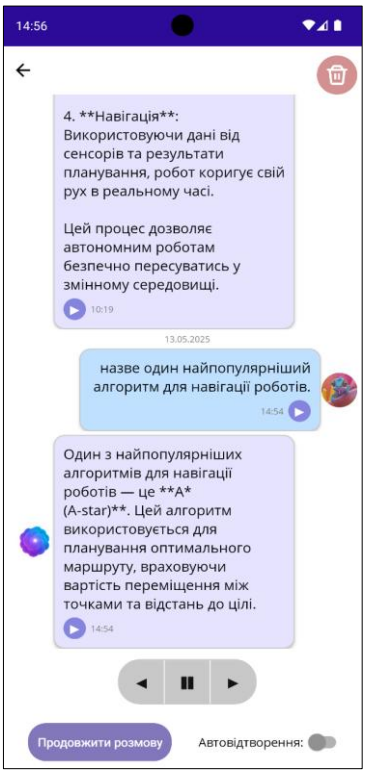

Рисунок 4.21 – Сторінка повідомлень особистої розмови

Після вибору функції для продовження особистої розмови, розмова переноситься на головний екран застосунку (рисунок 4.22). Тут присутній весь доступний функціонал, що і для загального чату, але з доповненням – можна скасувати продовження особистої розмови

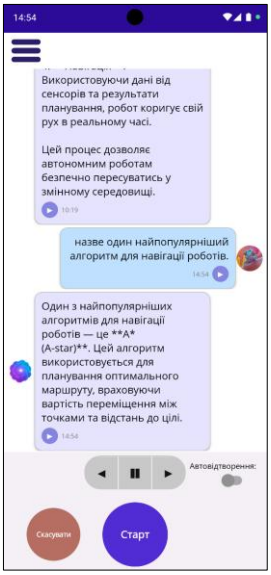

Рисунок 4.22 – Продовження особистої розмови

Висновки до розділу 4

У четвертому розділі було детально розглянуто інтерфейс мобільного застосунку автоматизованої системи взаємодії голосом, з використанням створеного діалогового агента для робототехнічної системи. Описано всі ключові етапи взаємодії користувача із застосунком. Навігація по додатку здійснюється через зручне бокове меню, яке забезпечує швидкий доступ до основних розділів: “Головна”, “Користувачі”, “Налаштування”, “Аналітика”, “Історія”. Кожен із цих розділів виконує окрему функцію: перегляд чату, управління профілями, зміну налаштувань, поповнення балансу, перегляд статистики та доступ до історій розмов.

Таким чином, інтерфейс мобільного застосунку поєднує зручність, функціональність та інтуїтивну навігацію, що забезпечує ефективну взаємодію користувача з системою. Чітка структура меню, широкий спектр можливостей та орієнтація на користувацький досвід дозволяють застосунку повноцінно виконувати роль інструменту для голосової комунікації й управління персональними даними в межах автоматизованої системи.

ВИСНОВКИ

У рамках дипломного проєкту, було досягнуто успішної реалізації інтелектуального діалогового агента у вигляді мобільного додатку. Цей застосунок є багатокористувацькою системою, тобто, у межах одного особистого профілю, може бути декілька користувачів (голосових профілів). Він підтримує функцію голосової ідентифікації та персоналізації, що повністю відповідає початковій меті – автоматизація голосової взаємодії між користувачем і системою.

Всі визначені завдання, починаючи від аналізу сучасних голосових рішень і закінчуючи проектуванням архітектури та розробкою функціонального мобільного додатку, були виконані. Створений агент забезпечує повний автоматизований голосовий цикл, взаємодію із серверами, збереження історії спілкування, обробку команд користувача в реальному часі, забезпечення конфіденційного спілкування в особистих чатах та створення аналітики.

Обрана клієнт-серверна і мікросервісна архітектура виявилася ефективною, адже вона дозволила розподілити навантаження між окремими компонентами системи (мобільний застосунок, сервер доступу до сервісів, сервер аудіообробки), що спростило масштабування та подальший розвиток функціоналу.

Android-додаток є клієнтом, що був створений за допомогою .Net MAUI. та патерну MVVM. Сервер доступу до сервісів був реалізований за допомогою ASP .NET Core та Onion-архітектури з шаблоном CQRS. Другий сервер, що відповідає за аудіооперації, було реалізовано на Python, який використовує різні моделі ШІ.

Використання хмарних сервісів, зокрема Azure Blob Storage і Cosmos DB, гарантує стабільний і безпечний доступ до даних.

У цьому проєкті було успішно впроваджено передові технології штучного інтелекту, серед яких: моделі розпізнавання, діаризації, транскрибування, синтезу мовлення та генератори векторів голосового представлення (embedding). Завдяки такому набору складових, система продемонструвала відмінну адаптивність, легко пристосовуючись до різних користувачів.

Створений діалоговий агент має значний потенціал практичного використання у багатьох сферах, адже все більше робототехнічних систем було впроваджено у щоденне життя людей, а саме діалогові агенти допоможуть єфективно надавати можливість природного спілкування.

Під час реалізції системи, було виявлено кілька недоліків, зокрема, це залежність якості розпізнавання від навколишнього шуму, а у випадках великої кількості користувачів зі схожими голосами, система може хибно ідентифікувати користувача. Крім того є труднощі з адаптацією до мовних діалектів або акцентів.

Результати проєкту демонструють потенціал для розширення: інтегрування з фізичними роботизованими системами, включення підтримки різноманітних регіональних мов, розробка рекомендаційної системи, що базується на голосових даних користувачів, а також введення більш гнучких платіжних механізмів та аналітичних інструментів.

Підсумовуючи, створений інтелектуальний діалоговий агент не тільки відповідає сучасним потребам голосових систем, але й формує фундамент для подальших досліджень та покращення персоналізованої взаємодії людини з технологіями. Здобуті результати демонструють актуальність та практичну корисність обраної тематики, а також значний потенціал її інтеграції у велику кількість сучасних застосувань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Skantze G. Error Handling in Spoken Dialogue Systems, 2007: Chapter 2, Spoken dialogue systems. URL: https://www.speech.kth.se /~gabriel/thesis /chapter2.pdf (дата звернення: 5.04.2025).

2. Siri for developers. URL: https://developer.apple.com/siri/ (дата звернення: 10.04.2025).

3. Understanding 'Hey Siri' — how to use it, commands, and more. URL: https://istore.mu/blogs/news/understanding-hey-siri-how-to-use-it-commands-and-more (дата звернення: 15.05.2025).

4. Alexa developer documentation. URL: https://developer.amazon.com/en-US/docs/alexa/documentation-home.html (дата звернення: 10.04.2025).

5. Amazon Astro. URL: https://au.pcmag.com/smart-home/98422/amazon-astro (дата звернення: 15.05.2025).

6. Amazon Echo & Echo Show model. URL: https://www.aftvnews.com /nearly-ever-amazon-echo-echo-show-model-is-on-sale-most-are-within-10-of-all-time-low-price/ (дата звернення: 15.05.2025).

7. Google Assistant for developers. URL: https://developers.google.com /assistant/ (дата звернення: 10.04.2025).

8. New Google Assistant Feature Speeds Up Voice Assistant. URL: https://voicebot.ai/2022/02/17/new-google-assistant-feature-speeds-up-voice-assistant/ (дата звернення: 15.05.2025).

9. LG Robot. URL: https://www.lg.com/global/business/robot (дата звернення: 15.05.2025).

10. C# language documentation. URL: https://learn.microsoft.com/uk-ua/dotnet/csharp/ (дата звернення: 10.03.2025).

11. Jon P. Smith. Entity Framework Core in Action, Second Edition. 2021. С. 125–158.

12. Azure for .NET developers. URL: https://learn.microsoft.com/en-us/dotnet/azure/ (дата звернення: 15.04.2025).

13. .NET Multi-platform App UI Documentation. URL: https://learn.microsoft.com/en-us/dotnet/maui/?view=net-maui-9.0 (дата звернення: 10.04.2025).

14. ASP.NET Core Documentation. URL: https://learn.microsoft.com/en-us/aspnet/core/ (дата звернення: 01.04.2025).

15. The Python Tutorial. URL: https://docs.python.org/3/tutorial/index.html. (дата звернення: 10.03.2025).

16. François Chollet. Deep Learning with Python. 2017.

17. Speaker Diarization Documentation. URL: https://docs.nvidia.com/nemo-framework/user-guide/24.09/nemotoolkit/asr/speaker_diarization/intro.html (дата звернення: 01.04.2025).

18. OpenApi Documentation. URL: https://learn.openapis.org/ (дата звернення: 05.04.2025).

19. ElevenLabs Documentation. URL: https://elevenlabs.io/docs/overview (дата звернення: 10.04.2025).

20. .NET Api Documentation. URL: https://learn.microsoft.com/en-us/dotnet/api/ (дата звернення: 01.04.2025).

21. Martin R. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. 2017.

22. Richardson Ch. Microservices Patterns. API Gateway pattern. URL: https://freecontent.manning.com/the-api-gateway-pattern/ (дата звернення: 15.04.2023).

23. Pieter Nijs. The MVVM Pattern in .NET MAUI. 2023.

24. CQRS pattern. URL: https://learn.microsoft.com/en-us/dotnet/azure/ (дата звернення: 10.04.2025).

25. Jeffrey Palermo. The Onion Architecture: Part 1. URL: https://jeffreypalermo.com/blog/the-onion-architecture-part-1/ (дата звернення: 01.04.2025).