МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ГІНЗУЛА Володимир Ярославович

Програмний модуль голосового помічника з використанням технологій машинного навчання / Voice Assistant Software Module Using Machine Learning Technologies

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-43
Гінзула В.Я.
_____

Науковий керівник
к.т.н., доцент Д.І. Загородня
_____

Кваліфікаційну роботу допущено до
захисту
«___»_____ 20___ р.
В.о. завідувача кафедри
_____ Н. М.Васильків

Тернопіль – 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ГІНЗУЛА Володимир Ярославович
(прізвище, ім'я, по батькові)

1. Програмний модуль голосового помічника з використанням технологій машинного навчання / Voice Assistant Software Module Using Machine Learning Technologies
керівник роботи Д. І. Загородня к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 20 грудня 2024 р. № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:

- здійснити огляд існуючих віртуальних асистентів та їх функціональних можливостей;
- проаналізувати сучасні підходи до побудови голосових помічників, визначити їх функціональні можливості, переваги та обмеження;
- розробити архітектурну, алгоритмічне та інформаціне забезпечення системи та створити загальну структуру модуля голосового помічника;
- реалізувати програмне забезпечення, створити інтерфкейс користувача і провести тестування програми.

5. Перелік графічного матеріалу в роботі:

- архітектура віртуального голосового помічника;
- діаграма послідовності взаємодії голосового помічника з користувачем;
- результати тестування програми.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2025 р.	

Студент_____ В. Я. Гінзула_____
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ Д. І. Загородня
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль голосового помічника з використанням технологій машинного навчання» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом у 72 сторінки і містить 24 ілюстрацій, 2 таблиці, 2 додатки та 25 використаних джерел.

Метою роботи є дослідження сучасних методів побудови голосових помічників та створення програмного засобу, який забезпечує взаємодію користувача з комп'ютером через голосові команди з використанням технологій розпізнавання, обробки та синтезу мовлення.

Методами розроблення обрано метод аналізу (для вивчення існуючих рішень у сфері голосових асистентів), метод синтезу (для поєднання окремих компонентів системи), методи обробки природної мови (для аналізу голосових команд і генерації відповідей), метод моделювання (для побудови логіки обробки діалогів), а також експериментальні методи тестування (для перевірки точності розпізнавання мовлення та функціональності системи).

Внаслідок виконання роботи обґрунтовано доцільність використання обробки природної мови у створенні голосових інтерфейсів, розроблено архітектуру та реалізовано персонального голосового помічника, який здатен розпізнавати голосові запити, інтерпретувати їх, знаходити відповідь та озвучувати її користувачеві.

Ключові слова: ГОЛОСОВИЙ ПОМІЧНИК, NLP, РОЗПІЗНАВАННЯ МОВИ, СИНТЕЗ МОВЛЕННЯ, МАШИННЕ НАВЧАННЯ, API, КЕРУВАННЯ КОМАНДАМИ, PYTHON, ШТУЧНИЙ ІНТЕЛЕКТ.

ANNOTATION

Qualification work on the topic “Voice Assistant Software Module Using Machine Learning Technologies” for Bachelor’s degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written in 72 pages and contains 24 illustrations, 2 tables, 2 appendices and 25 references.

The aim of the work is to study modern methods of building voice assistants and create a software tool that provides user-computer interaction through voice commands using speech recognition, processing and synthesis technologies.

The development methods chosen are the analysis method (for studying existing solutions in the field of voice assistants), the synthesis method (for combining individual system components), natural language processing methods (for analysing voice commands and generating responses), the modelling method (for building dialogue processing logic), and experimental testing methods (for checking the accuracy of speech recognition and system functionality).

As a result of the work, the expediency of using natural language processing in the creation of voice interfaces has been substantiated, the architecture has been developed and a personal voice assistant has been implemented that is able to recognise voice queries, interpret them, find the answer and voice it to the user.

Keywords: VOICE ASSISTANT, NLP, SPEECH RECOGNITION, SPEECH SYNTHESIS, MACHINE LEARNING, API, COMMAND CONTROL, PYTHON, ARTIFICIAL INTELLIGENCE.

ЗМІСТ

ВСТУП

В останні роки віртуальні асистенти зі штучним інтелектом набирають дедалі більшої популярності, дедалі міцніше входять у наше щоденне життя. Завдяки вбудованим алгоритмам машинного навчання та обробки природної мови, ці технології можуть трансформувати буденні дії на більш зручні та ефективні, надаючи підтримку та полегшуючи виконання завдань.

Голосові помічники інтегровані в різні технологічні пристрої, включаючи смартфони, ноутбуки і настільні комп'ютери, ігрові консолі, телевізори, розумні колонки (аудіосистеми) та автомобілі. Вони здатні використовувати голосову модальність як централізований метод комунікації, що робить графічний інтерфейс користувача непотрібним або менш важливим.

Люди використовують технологію голосових асистентів в різних аспектах свого життя, наприклад для виконання простих завдань, таких як отримання прогнозу погоди, пошук потрібної інформації в інтернеті, керування електронною поштою, та багато інших корисних речей. Це виводить взаємодію людей з обчислювальними системами на новий рівень користування.

Незважаючи на те, що технологія голосових помічників була протягом певного часу, її широке поширення відбулося лише нещодавно. Першим сучасним та відомим голосовим помічником є Siri від Apple, який був представлений у 2011 році. Незабаром з'явилися Google Assistant, Alexa від Amazon та Cortana від Microsoft. Зараз на ринку існує велика кількість голосових помічників, і всі вони мають різноманітний функціонал та можливості [1].

Метою даної кваліфікаційної роботи є розробка програмного модуля голосового помічника, який використовує технології машинного навчання для забезпечення ефективного розпізнавання, обробки та генерації мовленнєвих команд у реальному часі. Для досягнення цієї мети необхідно виконати наступні завдання:

– здійснити огляд існуючих віртуальних асистентів та їх функціональних можливостей;

– проаналізувати сучасні підходи до побудови голосових помічників, визначити їх функціональні можливості, переваги та обмеження;

– розробити архітектурну, алгоритмічне та інформаціне забезпечення системи та створити загальну структуру модуля голосового помічника;

– реалізувати програмне забезпечення, створити інтерфкейс користувача і провести тестування програми.

Об'єктом дослідження є процес автоматизованої взаємодії людини з комп’ютерною системою за допомогою голосових команд.

Предметом дослідження є методи та засоби розробки програмного модуля голосового помічника з використанням технологій машинного навчання та обробки природної мови.

У процесі дослідження було використано низку методів, що забезпечують комплексний підхід до розробки програмного модуля голосового помічника з використанням технологій машинного навчання. Зокрема, застосовано методи аналізу та синтезу наукової літератури для вивчення сучасного стану досліджень у галузі розпізнавання мовлення, машинного навчання та обробки природної мови. Математичне моделювання використовувалося для формалізації процесів аналізу голосових команд, а також для побудови ефективних моделей обробки мовлення. Обробка природної мови (NLP) дозволила реалізувати розуміння контексту запитів користувача та формування відповідей. У роботі також використовувалися методи проєктування програмного забезпечення для створення архітектури та реалізації функціонального модуля.

Практичне значення запропонованого в кваліфікаційній роботі програмного модуля полягає в тому, що він може бути інтегрований у вже існуючі програмні продукти, підвищуючи їх доступність і зручність використання. Результати дослідження можуть бути використані для створення адаптивних голосових інтерфейсів у різних галузях: освітніх платформах, мобільних застосунках, системах "розумного дому", навігаційних системах, сервісах для людей з обмеженими можливостями тощо.

Структура і розмір дослідження. Кваліфікаційна робота складається із вступу, трьох розділів, висновків та списку використаних джерел. Загальний обсяг

роботи складає 72 сторінки і містить 24 рисунки, дві таблиці, два додатки та 25 використаних джерел.

Апробацію дослідження проведено на студентській науково-практичній конференції, м. Тернопіль, 27-29 травня 2025 р. “Інтелектуальні інформаційні технології в прикладних дослідженнях (IITAR-2025)”, де була представлена доповідь на тему “ Голосовий помічник на основі машинного навчання ” (додаток А).

1 АНАЛІЗ ГОЛОСОВИХ ПОМІЧНИКІВ

1.1 Опис предметної області

У 21 столітті взаємодія людей все більше замінюється автоматизацією. Продуктивність - це є головним рушієм цих змін є значний зсув у технології, а не просто прогрес. Сьогодні суспільство навчає машини виконувати завдання автономно або мислити як люди за допомогою технологій наприклад машинне навчання та нейронні мережі. У наш час це дозволяють віртуальні помічники спілкуватися з машинами за допомогою голосових команд.

Віртуальні помічники – це програмне забезпечення, призначене для полегшення щоденних завдань, таких як показ звітів про погоду, надання щоденних новин і пошук в Інтернеті. Ці помічники можуть приймати голосові команди, активовані за допомогою слова виклику або пробудження, за яким слідує команда користувача. Приклади популярних віртуальних помічників включають Siri від Apple, Alexa від Amazon і Cortana від Microsoft. Розвиток і успіх цих програмних застосунків надихнули на проект гололосового помічника [3].

Голосові помічники - це програми на цифрових пристроях які слухають і реагують на словесні команди. Наприклад, користувач може запитати: «Яка погода?» в голосовий помічник на дасть звіт про погоду на цей день і місце. Голосові помічники – це системи штучного інтелекту (ШІ), призначені для полегшення взаємодії користувача з цифровими пристроями.

Мобільні технології стали відомими своїм користувацьким досвідом, що дозволяє легко отримувати доступ до програм і послуг з будь-якою геолокацією. Різні відомі та широко використовувані мобільні операційні системи включають Android, Apple, Windows Blackberry. Ці операційні системи пропонують безліч програм та послуг. Наприклад, контактні програми зберігають контактні дані користувача та полегшують дзвінки або SMS. Подібні програми доступні по всьому світу через Apple Store і Play Store. Ці функції призвели до реалізації різноманітних датчиків і функцій у мобільних пристроях [4].

Голосовий помічник — це програма, яка включає такі технології, як розпізнавання голосу, обробка мови/мовлення, штучний інтелект, а також синтез

голосу для точного й ефективного виконання завдань за запитом користувача. Є різні типи голосових помічників, наприклад голосові помічники загального призначення, які допомагають у складанні списку товарів у кошику та здійснення платежів. Наступне десятиліття стало свідком того, як технологічні гіганти Android, Apple, Windows Blackberry удосконалили та налаштували своїх голосових помічників до такого рівня, що вони досягли неймовірного рівня розуміння та чутливості. Голосові помічники тепер є стандартом у цифрових пристроях сучасного світу. Вони схожі на електронні гаджети, від керування розкладами до налаштування будильників [2].

Голосові інтерфейси, які можуть активно керувати рішеннями споживачів на основі штучного інтелекту можуть природним чином спілкуватися з користувачами, контекстно розробляти запити та динамічно збільшувати свої знання, навчаючись на помилках на основі штучного інтелекту використовують передові технології штучного інтелекту, такі як автоматичне розпізнавання мови та розуміння природної мови, щоб вести діалог з людиною за допомогою людської мови. Ці розумні пристрої працюють як особистий консьєрж або покупець, виконуючи різноманітні запити користувачів у режимі реального часу. Далі обговорюється концептуалізація подвійної медіа ролі перед окресленням очікуваних наслідків для споживчих брендів. Як помічники, особливості цієї технології вимагають нових теорій, які ще не повністю розроблені. Під час взаємодії зі споживачами вони можуть обробляти і зберігати значну кількість вхідних даних і таким чином, поступово розуміти своїх споживачів і контекст. Зокрема, помічники забезпечують адаптивні відповіді шляхом обробки контекстних підказок.

Зазвичай набір тексту та пошук або виконання повсякденних завдань перетворюється на метушню. Можна звернутися за допомогою до голосових помічників. Вони дозволяють користувачам виконувати завдання за допомогою мовної команди, а також отримувати інформацію за допомогою голосового синтезу.

Щоб знайти якусь інформацію на веб-сайті часом потрібно багато часу або якщо порахувати кількість кліків, які потрібно зробити, щоб знайти потрібну річ.

Голосові помічники не викликають таких труднощів. Можна задати питання, і ви отримаєте відповідь. І голосовий пошук є гарячою темою для обговорення. Голосова видимість, безсумнівно, буде проблемою. Це пов’язано з відсутністю візуального інтерфейсу голосових помічників. Користувачі не можуть бачити або взаємодіяти з голосовим інтерфейсом, якщо він не пов’язаний із додатком Alexa або Google Assistant. У результаті пошукова поведінка кардинально зміниться. Рекламні агентства стають все більш популярними, оскільки популярність голосового пошуку зростає. Голосові помічники також продовжуватимуть пропонувати більш індивідуальний досвід, оскільки вони краще розрізнятимуть голоси. Очікується, що кількість людей, які користуються голосовими помічниками, зростатиме.

1.2 Огляд і аналіз існуючих рішень

Огляд і аналіз наявних рішень допомагає оцінити наявні технології та їхні характеристики, щоб вибрати найбільш відповідне рішення для конкретного випадку використання. Для аналізу наявних рішень потрібно визначити критерії оцінювання голосових помічників. До них належать такі параметри, як функціональність, точність розпізнавання мови, швидкість, підтримка різних мов, інтеграція з іншими сервісами, простота використання, безпека та конфіденційність даних.

Віртуальний або персональний асистент - це універсальна програма, призначена для виконання низки завдань за запитом користувача. В її основі лежить модель обробки природної мови (Natural Language Processing, NLP), що дозволяє розпізнавати голосові команди. Деякі програми також мають графічний інтерфейс для обробки зображень. Найчастіше цифрові асистенти - це хмарні сервіси. Це дозволяє вбудовувати їх у багато пристроїв - як стаціонарних, так і мобільних. Щоб користуватися таким асистентом, потрібен лише відповідний пристрій (наприклад, смартфон) і доступ до інтернету [6].

Голосові помічники стали невід'ємною частиною сучасного цифрового життя, допомагаючи користувачам виконувати різноманітні завдання – від пошуку

інформації до управління розумним будинком. Найпопулярнішими серед них є Siri від Apple, Alexa від Amazon і Google Assistant від Google. Кожен із них має свої унікальні особливості, переваги та недоліки [5].

Голосовий помічник Siri - це віртуальний помічник, розроблений компанією Apple у 2011 році для пристроїв на платформах iOS, MacOS, iPadOS, watchOS, tvOS та audioOS. Управління здійснюється за допомогою голосових запитів і жестів, а також звичайних натискань кнопок. Помічник заснований на технології обробки природної мови, яка дозволяє системі обробляти і відповідати на запити, надавати рекомендації і контролювати роботу інтегрованих цифрових пристроїв.

Apple Siri відповідає на запитання людини за допомогою вбудованого мовного генератора і виводить всю необхідну інформацію на екран пристрою, на якому вона встановлена. Цей віртуальний помічник також може використовуватися для диктування електронних листів, SMS різного типу, обробки вхідних листів і повідомлень. Зображення логотипу Siri представлено на рисунку 1.1.

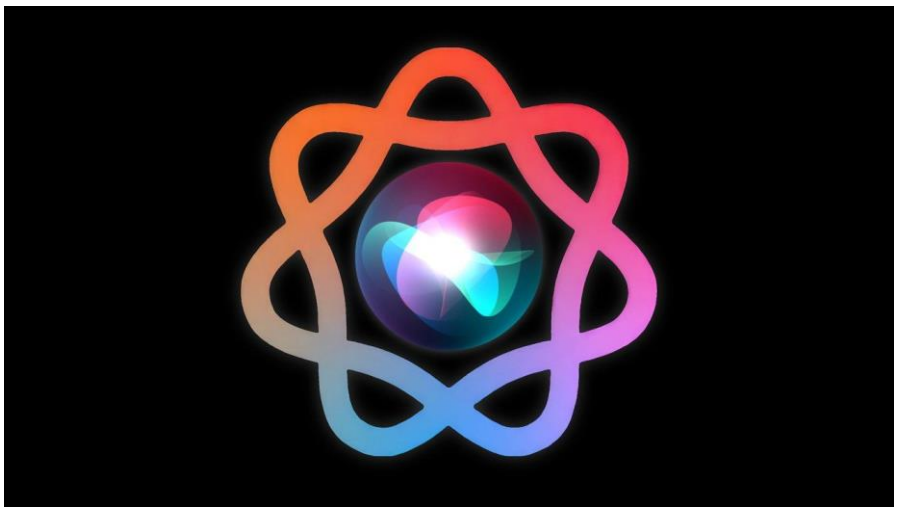

Рисунок 1.1 – Зображення логотипу «Apple Siri»

Функціональність Siri складається з трьох основних компонентів:

– розмовний інтерфейс;
– контекстна обізнаність;
– делегування завдань з урахуванням контексту.

Перша функція допомагає додатку розпізнавати людський голос і розуміти, що від нього вимагається. Оскільки це одна з найважливіших якостей для такого

сервісу, розробники приділили їй особливу увагу. Безсумнівною перевагою Siri є її здатність розпізнавати голос і відповідати на 20 різних мовах. Щоб активувати цього помічника, достатньо вимовити універсальну фразу для пробудження «Hey Siri» біля будь-якого пристрою Apple, включаючи пристрої розумного будинку.

Ще однією важливою особливістю цього помічника є система контекстного розуміння, яка робить інтелектуальні додатки ще більш розумними. Завдяки цій системі Siri легко адаптується до звичок, мовних особливостей та часто вживаних слів і фраз користувача. Контекстне розуміння допомагає Siri розуміти користувача з першого погляду і надавати персоналізовані відповіді.

Делегування завдань дає змогу віртуальному помічнику отримувати доступ до сторонніх додатків на підтримуваних пристроях Apple і керувати ними за запитом користувача. Siri відрізняється від інших помічників тим, що вона централізована, отримує голосові команди і відтворює голос. Немає жодного пристрою-концентратора. Смарт-колонки HomePod і HomePod Mini найбільш схожі одна на одну, але це не їхня основна функція.

Широкий набір голосових команд відкриває доступ до таких функцій Siri, як:

– управління викликами та повідомленнями (віртуальний помічник телефонуватиме або надсилатиме повідомлення вашим контактам);

– пошук актуальної інформації в Інтернеті (наприклад, прогнозу погоди, курсів валют), включаючи загальновідомі факти (наприклад, населення країни). Планування подій, встановлення нагадувань і таймерів;

– керування функціями пристрою (наприклад, зйомка фотографій, увімкнення/вимкнення Wi-Fi);

– розширений пошук в Інтернеті (наприклад, зображень, повідомлень у соціальних мережах);

– прокладання оптимальних маршрутів і відображення поточних дорожніх умов;

– переклад слів і фраз з однієї мови на іншу. Керування додатками на пристроях Apple;

– пошук інформації про розважальні заходи (наприклад, про сьогоднішні сеанси в кінотеатрах);

– управління транзакціями Apple Pay; управління рахунком Apple Pay.

Голосовий помічник Alexa – віртуальний асистент компанії Amazon, запущений у 2014 році; порівнюючи Alexa і Siri, варто зазначити, що продукт Amazon вже кілька років лідирує на ринку США, займаючи понад 60% його частки. Amazon Echo є центральним пристроєм для цього ШІ-асистента. Крім того, цей хмарний додаток підтримує безліч інших сторонніх пристроїв. Його можна переглядати на різних пристроях, включаючи смартфони, планшети та смарт-телевізори. Якщо Alexa не встановлена на вашому пристрої за замовчуванням, ви можете завантажити її з App Store або Google Play. Зображення голосового помічника Alexa представлено на рисунку 1.2.

Рисунок 1.2 - Зображення логотипу «Alexa»

У більшості випадків сервіс активується за допомогою пробуджувальної фрази «Alexa, open wake up», проте деякі пристрої не підтримують цю функцію, тому для запуску необхідно натискати кнопку. Він доступний вісьмома мовами: англійською, іспанською, французькою, німецькою, італійською, португальською, хінді та японською. Однією з головних переваг цього персонального помічника є його універсальність.

Ще одна важлива перевага цього віртуального помічника - велика кількість навичок. На сьогодні їх кількість перевищує 25 000 і продовжує зростати. Amazon також надає стороннім розробникам можливість створювати нові скіни для розумного помічника. Завдяки цим навичкам користувачі можуть гнучко керувати

різними функціями своїх пристроїв за допомогою голосових команд, за створення яких відповідає модуль Alexa Skills Kit.

Серед найпопулярніших функцій цього віртуального помічника:

– прослуховування музики, онлайн-радіо, подкасти, аудіокниги;

– пошук новин, спортивних подій та рецептів;

– перевірка прогнозу погоди;

– керування пристроями розумного будинку;

– керування календарями та списками справ, дзвінками та повідомленнями;

– встановлення будильників та таймерів;

– замовлення доставки їжі;

– покупки на Amazon.

Ще одна корисна функція Alexa - управління підприємством. Вона називається Alexa for Business і дає змогу керувати робочими місцями та додатками за допомогою розпізнавання голосу. Варто згадати і модуль Alexa for Hospitality. Готелі, ресторани та інші підприємства сфери HoReCa можуть надавати голосовий доступ своїм клієнтам і керувати своїми послугами.

Ще одним голосовий помічником є Google Assistant. Порівняння Google Assistant, Alexa і Siri за роком запуску показує, що Google запустила свого персонального асистента пізніше за конкурентів, у травні 2016 року. Водночас це дало компанії можливість розглянути плюси і мінуси вже наявних рішень. Наразі Google Assistant доступний на смартфонах і планшетах під управлінням Android, на розумній колонці Google Home для управління «розумним будинком», у месенджері Google Alo і на смарт-годиннику Android Wear. Крім того, цей голосовий помічник доступний для широкого спектра пристроїв, включно зі смарт-дисплеями, розумними телевізорами та автомобільними системами [7].

Google Assistant заснований на сервісі персонального пошуку Google Now. Як і інші віртуальні помічники, він використовує технології когнітивних обчислень, машинного навчання і розпізнавання голосу. Асистент доступний на більшості пристроїв під управлінням Android 5.0 і вище. Зображення логотипу «Google Assistant» представлено на рисунку 1.3.

Крім того, його можуть встановити і користувачі Apple. Щоб запустити застосунок на старих смартфонах, достатньо натиснути й утримувати кнопку «Додому», а на останніх моделях - просто провести пальцем по діагоналі в нижньому кутку екрана. Асистент також підтримує голосову активацію: достатньо вимовити фразу пробудження «OK Google».

Рисунок 1.3 - Зображення логотипу «Google Assistant»

Google Assistant підтримує команди більш ніж 40 мовами, що робить його корисним для користувачів по всьому світу. Завдяки цьому голосовому помічнику можна:

– керувати пристроями «розумного будинку»;
– отримувати доступ до онлайн-календаря та керувати ним;
– шукати інформацію в Інтернеті: планувати маршрути, обирати ресторани, перевіряти новини та прогнози погоди;
– відтворювати музику і відео та керувати треками;
– відтворення контенту на Chromecast та інших сумісних пристроях;
– встановлюйте будильники, таймери та нагадування;
– організовувати зустрічі та заходи й надсилати повідомлення;
– запуск і керування додатками на пристрої;
– проговорювати текстові повідомлення;

– перекладати текст підтримуваними мовами та відтворюйте його голосом [8].

Підбиваючи підсумки порівняння віртуальних голосових помічників, можна помітити, що кожен з них має свої особливості, сильні та слабкі сторони. Представлені тут помічники не дарма вважаються першокласними. Вони розробляються найбільшими технологічними компаніями, тому тут немає явних лідерів або аутсайдерів. Обираючи найкращий сервіс, насамперед потрібно орієнтуйтуватися на пристрій, який ви збираєтеся використовувати, і цілі, для яких ви збираєтеся його застосовувати. Нижче представлена порівняльна таблиця 1.1., яка допоможе розібратися, який голосовий асистент найкраще підійде.

Таблиця 1.1 – Порівняння голосових помічників

Характеристика	Siri	Alexa	Google Assistant
1	2	3	4
Розробник	Apple	Amazon	Google
Пристрої	iPhone, iPad, Mac, Apple Watch, Apple TV, HomePod	Amazon Echo, Fire TV, сторонні пристрої	Android-смартфони, Google Nest, сторонні пристрої
Голосова активація	"Hey Siri"	"Alexa"	"Hey Google" / "OK Google"
Інтеграція	Глибока інтеграція з iOS, macOS та Apple HomeKit	Підтримка екосистеми Amazon, сумісність із багатьма розумними пристроями	Інтеграція з Android, Google Home, Google-сервісами
Розпізнавання голосів	Так, персоналізоване розпізнавання голосу	Так, підтримка розпізнавання декількох користувачів	Так, точне розпізнавання голосів користувачів
Розумний дім	Підтримує HomeKit	Великий вибір пристроїв з підтримкою Alexa	Широка сумісність із пристроями Google Home і Nest
Якість відповідей	Добре для простих питань, слабше у складних запитах	Відмінно працює для покупок, контролю пристроїв	Найкращий у пошукових запитах та складних питаннях
Додатки та розширення	Скорочений набір команд, обмежена підтримка сторонніх додатків	Велика бібліотека "Skills" для розширення функціоналу	Доступні "Actions", широка підтримка сервісів

Продовження таблиці 1.1

1	2	3	4
Мультимовність	Обмежена підтримка кількох мов	Обмежена підтримка одночасного використання мов	Найкраща підтримка кількох мов одночасно
Особливості	Безпека та конфіденційність, глибока інтеграція з Apple	Найкраще працює з Amazon-сервісами, великі можливості кастомізації	Найкраща відповідь на загальні запити, природніше спілкування

Підсумуючи, Google Assistant – це найуніверсальніший і найфункціональніший з віртуальних персональних помічників. Однак якщо віддаєте перевагу пристроям Google або операційній системі Android, вам слід вибрати Google Assistant. Для справжніх шанувальників продукції Apple вибір очевидний. Крім того, він забезпечує підвищену безпеку та захист даних. Amazon Alexa має важливу перевагу. На сьогоднішній день це найкраще рішення для інтеграції з системами «розумного будинку» [9].

1.3 Постановка задачі дослідження

Після аналізу наявних голосових асистентів потрібно поставити чітку задачу для реалізації даного задання, та позначити чіткий інструментапрій для написання програмоного коду. Реалізація програмоного коду буде за допомогою мови програмування Python. Після аналізу наявних голосових помічників було зрозуміло, що основна їх частина:

– направлена на роботу на смартфоні (Android, iOS), мають обмежену кількість функцій при роботі на стаціонарних пристроях або працює лише на пристроях від компанії-виробника технології голосового помічника;

– не може використовуватись на території України по різним причинам (не орієнтована на країну, накладені обмеження проти продукту, потребує додаткових операцій, котрі не завжди успішні, для роботи з застосунком або інше);

– проблеми з розпізнаванням голосових запитів та надання недостовірних даних по ньому;

– мають внутрішню рекламу яка не завжди є регіональною.

Основну кількість вказаних недоліків можна легко усунути, що вже використовують компанії. Таким чином, переглядаючи ринок голосових помічників, можна зробити висновок, що їх варіація з кожним разом стає все більше. Однак вони не мають такої популярності через мале впізнавання бренду або через велику кількість помилок при обробці запитів. Крім того, однією з проблем більшості таких "стартапів" є те, що вони продають своїх голосових асистентів або пропонують підписку на обмежений період за високу ціну, хоча, як вже було зазначено, продукт може бути недоробленим і містити багато помилок.

Також потрібно врахувати що голосовий помічник повинен активуватись із кодового слова для того щоб можна було з ним працювати . Якщо помічник буде зчитувати кожне слово, алгоритм почне працювати не коректно і надавати недостовірні відповіді на непотрібні запити. З цієї причини асистент повинен починати обробку запису тільки після того, як почує кодове слово від користувача.

Саме тому голосові помічники мають бути затребувані широким загалом як для вирішення повсякденних завдань, таких як скорочення часу на пошук і переклад інформації, так і для допомоги користувачам у їхніх робочих процесах. Впровадження голосових помічників має скоротити час на пошук інформації та переклад тексту і спростити використання персональних комп'ютерів загалом. Щоб вирішити зазначені проблеми, які зустрічаються при роботі з голосовими помічниками на сьогодні потрібно перейти до комплексного підходу вирішення цих проблем. Тому, були виокремлені наступні завдання:

– здійснити огляд існуючих віртуальних асистентів та їх функціональних можливостей;

– проаналізувати сучасні підходи до побудови голосових помічників, визначити їх функціональні можливості, переваги та обмеження;

– розробити архітектурну, алгоритмічне та інформаціне забезпечення системи та створити загальну структуру модуля голосового помічника;

– реалізувати програмне забезпечення, створити інтерфкейс користувача і провести тестування програми;

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Загальна структура програмного модуля голосового помічника

Мета кваліфікаційної роботи це створення віртального голосового помічника на основі машинного навчання. Даний голосовий помічник має вирішувати наступні вимоги:

- розпізнавання людської мови;
- створювати голосову відповідь (давати відповідь голосом);
- займатись перекладом мови та тексту;
- виконувати поставленні команди;
- відкривати програми та файли;
- мати зручний інтерфейс;

Для розробки голосового помічника та забезпечення двосторонньої взаємодії між користувачем і системою потрібно:

- проаналізувати предметну область голосових помічників;
- розглянути та проаналізувати наявні аналогічні продукти;
- визначити основні функціональні вимоги до голосового помічника;
- створити основні сценарії взаємодії користувача з системою;
- створити зрозумілий, інтуїтивно зрозумілий і унікальний дизайн інтерфейсу.

Розроблюваний голосовий помічник повинен мати популярність поміж звичайних користувачів щоб зменшити час у рутинних справах: пошук інформації, переклад тексту та допомагати користувачам у їх робочих процесах. Впроваджуючи голосового помічника у роботу має скоротитись час на переклад тексту, на пошук інформації та спростити користування персональним комп’ютером, що допоможе людям з обмеженими можливостями.

Користувачі є головним ініціатором взаємодії з голосовим помічником і вони відіграють важливу роль у поданні команд та вимог. Людина яка використовує голосовий помічник з використанням машинного навчання маючи на меті виконання деяких рутинних задач.

Роль користувача полягає у надсиланні голосових команд або запитів голосовому помічнику. На основі розпізнавання та опрацювання мови голосовий помічник приймає ці команди та дає відповідь, виконує дію або надає необхідну інформацію.

Користувачі мають різні потреби та цілі при використанні голосового помічника. Наприклад, можуть використовувати для пошуку інформації, виконувати завдання, для контролю певних пристроїв або систем, запускати програми, розпоряджатися покупками, отримувати оновлення або нагадування, а також здійснювати інші дії, що покращують їхню продуктивність та зручність.

Для успішної взаємодії з голосовим помічником необхідно:

- чітко та зрозуміло формулювати команди;
- правильно вимовляти слова та чітко артикулювати;
- використовувати відповідні ключові слова, фрази або шаблони запитівж;

Щоб отримати найкращі результати, користувачам слід:

- навчитися оптимальним способам використання голосового помічника;
- ознайомитися з його функціями та можливостями.

Отже, щоб ефективно використовувати голосовий помічник, потрібно практикуватися в чіткій вимові та вивчити, як правильно ставити запитання.

Аудіоприймачі, також відомі як мікрофони, є важливим компонентом системи голосового помічника. Він використовується для збору голосових сигналів і перетворення їх в електричні сигнали, які можуть бути оброблені голосовим помічником AI.

Роль аудіоприймача полягає в зборі мовних даних, що складаються з голосових команд і запитів користувача. Він виявляє звукові хвилі в навколишньому середовищі та передає їх голосовому помічнику у вигляді електричних сигналів для подальшого опрацювання. Звукоприймачі відіграють важливу роль у забезпеченні якості та точності розпізнавання мови. Приймачі мають бути чутливими до звуків різних частот і напрямків. Добре налаштований

приймач забезпечує чітке і точне приймання голосових команд, незалежно від шумів і перешкод у навколишньому середовищі.

Голосовий процесор є серцем системи голосового помічника, відповідаючи за обробку та аналіз звукових сигналів, що надходять від мікрофона. Він використовує складні алгоритми та моделі машинного навчання, щоб розпізнати та зрозуміти голосові команди чи запити користувача. Фактично, він перетворює звукові хвилі на текст, який потім може бути оброблений системою для виконання відповідних дій.

Розпізнавач мови Wasp - важливий компонент системи голосового помічника. Він відповідає за аналіз та інтерпретацію вимовлених користувачем слів і фраз і перетворення їх на текстовий формат, який може бути опрацьований системою. Задача розпізнавача мови у голосових помічників включає багато аспектів.

Система застосовує акустичні моделі, натреновані на великих масивах аудіоданих. Ці моделі аналізують звукові хвилі, отримані з голосу користувача, вивчаючи такі характеристики, як частота, гучність і тривалість. Це дозволяє системі зіставити вимовляні звуки з відповідними мовними одиницями.

Лінгвістичне моделювання надає системі змогу розбирати синтаксис і семантику тексту, використовуючи для цього специфічні мовні моделі.Він застосовує граматичні правила та словники для ідентифікації мовних структур, розпізнавання словосполучень і визначення значення сказаного.

Також система проводить аналіз контекстуальних відносин між словами та фразами, застосовуючи контекстуальні моделі для більш точного визначення цілей користувача.Воно використовує історію взаємодії з користувачем, контекстну інформацію і природну мову для визначення сенсу і намірів вимовлених команд.

Система розпізнавання мовлення впроваджує засоби виправлення помилок та подальшої обробки результатів з метою збільшення точності транскрибування, адже акустичне і лінгвістичне моделювання мають притаманні їм недоліки.. Однак, вона враховує можливість неправильного розуміння або помилкового розпізнавання. Для покращення точності система використовує різні методи, такі як алгоритми корекції помилок, аналіз контексту та моделі ймовірності [10].

Ці методи допомагають системі краще розуміти, що саме було сказано, і надавати більш точні результати.

Штучний інтелект надзвичайно важливий для ефективної роботи голосової помічників, забезпечуючи точну та зручну взаємодію між користувачами та системами. Він поєднує в собі різноманітні технології, які працюють разом, щоб покращити якість розпізнавання мови, розуміти наміри користувача та дати правильну відповідь.

У акустичному моделюванні звукові характеристики знань відповідають звуку між звуком та мовою одиниць. Моделювання мови дозволяє визначити мовну структуру, розпізнавати фрази та розуміти зміст фраз. Сприйняття знань, пов'язаних з контекстом, допомагає розглянути історію взаємодії з користувачами, покращує розуміння намірів та демонструє виконання команди.

Голосовий помічник складається з декількох ключових компонентів, які забезпечують ефективну роботу. Мовний процесор відповідає за обробку голосових команд та виконання відповідних дій. Він отримує текстові дані від Speech Eicherner і перетворює їх у внутрішні інструкції для виконання таких завдань, як бази даних, логічна обробка та вірний розподіл.

Штучний інтелект відіграє важливу роль у визнанні мови та розуміння намірів користувачів. За допомогою алгоритмів та нейронних мереж для машинного навчання вона аналізує синтаксис, семантику та контекст команди. Тобто їх можна трактувати більш точно. Крім того, AI забезпечує генерацію відповідей шаблонами, структурованими даними або генерацією природної мови. Це робить взаємодію більш комфортними та природними. Завдяки своїй здатності вчитися та адаптуватися, система вдосконалює визнання, розуміння та точність реагування з часом, надаючи користувачам персоналізований досвід

На рисунку 2.1 показано діаграма послідовності взаємодії голосового помічника з користувачем. Для прикладу розглянемо процес обробки запиту щодо погоди .

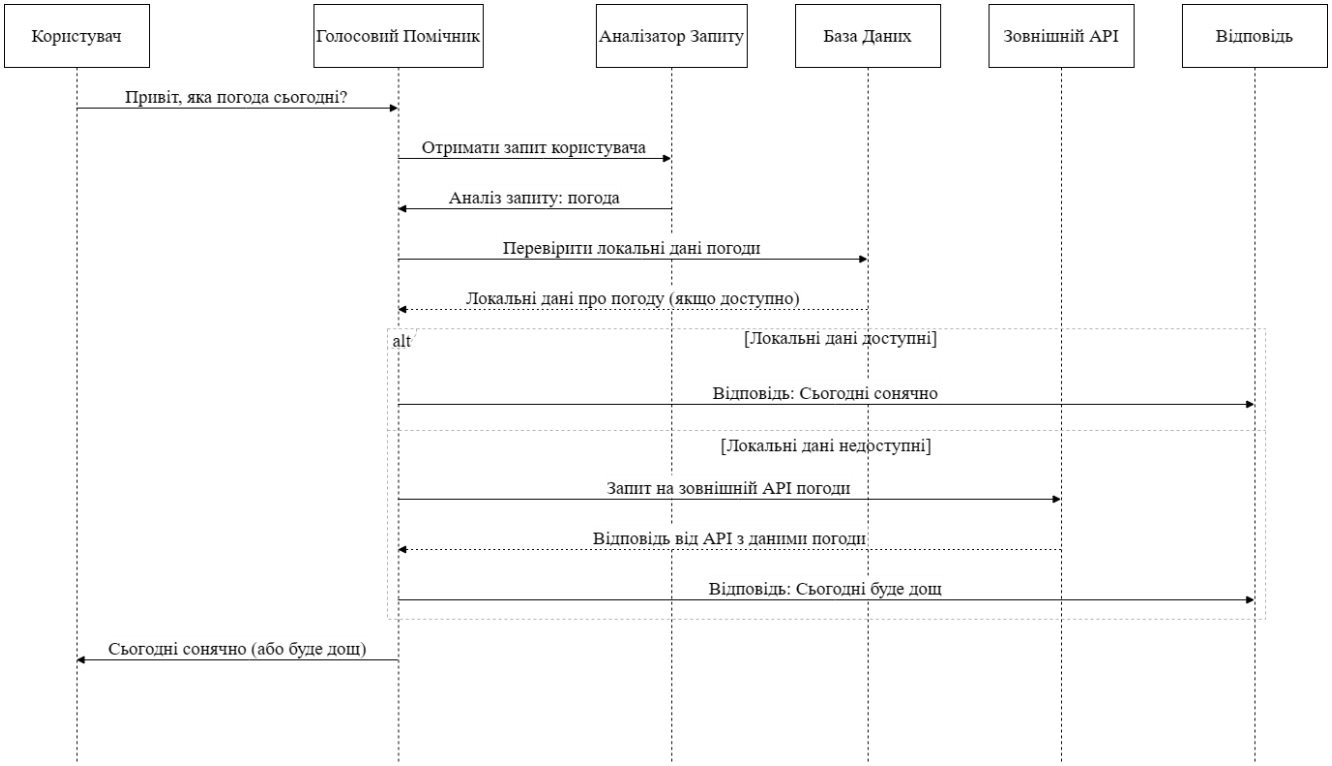

Рисунок 2.1 – Діаграма послідовності взаємодії голосового помічника з користувачем

База даних є важливим елементом голосового помічника, оскільки зберігає необхідну інформацію, таку як спеціальні профілі, налаштування, контакти, взаємодія тощо. Забезпечує швидкий доступ до даних, які можна використовувати для виконання ваших вимог відразу. База даних також допомагає персоналізувати взаємодії шляхом аналізу налаштувань користувачів для надання індивідуальних рекомендацій. Інтеграція в інші системи, такі як календарі, електронні листи та музичні послуги, значно розширить функціональність вашого помічника. Крім того, ви можете використовувати базу даних для збору статистики та використання системи. Система може проаналізувати ефективність та підвищити роботу.

Усі ці компоненти працюють разом для створення адаптивних та корисних інструментів для взаємодії з користувачами. Штучний інтелект дозволяє голосовим помічникам "розуміти" та дізнаватися більше, що дозволяє їм краще задовольнити потреби користувачів та надавати персоналізовані відповіді.

2.2 Алгоритмічне забезпечення

Голосовий асистент є збірником великого спектру технологій штучного інтелекту для належного функціонування, але потрібно виділити дві, це ключові компоненти його архітектури: виявлення та мова інтеграція

Перетворення звуку на текст, або STT (speech-to-text), є ключовим процесом, що дозволяє голосовим помічникам розуміти людську мову. Система "слухає" голос і "розпізнає" слова, перетворюючи їх на текстовий запит. Зворотний процес, TTS (text-to-speech), перетворює текст на звук, імітуючи природну мову. Обидві технології, STT і TTS, є основою для ефективної взаємодії з голосовими помічниками, забезпечуючи двосторонній зв'язок між користувачем і системою.

Розпізнавання мовлення яке називають також має назву автоматичним розпізнавання мовлення (ASR), комп'ютерним розпізнаванням мовлення або перетворення мови в текст (STT), є формою штучного інтелекту та відноситься до здатності комп'ютера або машини перетворювати слова та перетворювати їх у текст.

Розпізнавання мовлення та розпізнавання голосу – це дві різні, хоча й пов'язані технології. Розпізнавання мовлення зосереджується на перетворенні усного мовлення на письмовий текст. Воно аналізує звукові сигнали, враховуючи такі характеристики, як висота, темп та акцент, щоб зрозуміти та записати слова.

На відміну від цього, розпізнавання голосу – це біометрична технологія, яка використовується для ідентифікації особи за її унікальними голосовими характеристиками. Воно аналізує такі особливості голосу, як тон, ритм та висота, для підтвердження особистості користувача. Ця технологія часто застосовується в системах безпеки та автентифікації.

Людська мова є надзвичайно складною для аналізу через її різноманітність та мінливість. Тому розробка технологій, здатних точно розпізнавати та розуміти мовлення, завжди була непростим завданням. Сучасні системи розпізнавання мовлення використовують складні алгоритми та моделі машинного навчання, щоб враховувати ці складнощі та забезпечувати високу точність.

Блок-схема демонструє функціонування голосового асистента або системи розпізнавання мови, що опрацьовує запити користувача. Все починається з голосової команди, яку вимовляє користувач, і яка стає вхідним сигналом системи. Далі відбувається розпізнавання мови, перетворюючи голос на текстову інформацію. Після цього система здійснює обробку звуку, а також аналізує запит для визначення його суті. На наступному кроці визначається валідність запиту. У випадку позитивної відповіді відбувається його опрацювання: пошук даних, формування відповіді, трансляція тексту в мовлення та вивід результату. Якщо запит є недійсним, система реалізовує обробку помилки та пропонує користувачеві повторно сформулювати запит. Таким чином, схема відображає весь цикл взаємодії користувача з голосовим асистентом, від голосової команди до отримання відповіді або пропозиції щодо уточнення.

Більшість методів сьогоднішнього розпізнавання голосів оцінюються точністю або помилкою (WER). WER - є кількістю помилок, поділеними на загальні слова Щоб обчислити це , ви повинні узагальнити заміни, вставки та делеції, які трапляються в наборі визнаних слів, і поділитися цим числом із загальною кількістю слів, які спочатку були сказані. Існує також багато факторів, які можуть впливати на точність, такі як акцент, крок, вимова, гучність та фоновий шум. Завдяки , цьому є багато алгоритмів, винайдених для досягнення найкращих результатів для розпізнавання мови, таких як ПММ (HIDDEN MARKOV MODEL), заснований на більшості сучасних систем розпізнавання мови. Деякі моделі здатні адаптуватися до змін параметрів, тоді як інші можуть працювати лише з обмеженими їх значеннями

Деякі моделі здатні адаптуватися до змін параметрів, тоді як інші можуть працювати лише з обмеженими їх значеннями. Важливо враховувати, що більшість моделей було навчено на наборах даних із частотою дискретизації 16 Гц та одним каналом (моноаудіо).

Щоб зменшити можливі невідповідності, можна застосовувати різні методи обробки звуку залежно від мови програмування. Приклад роботи голсового помічника для отрмання відповдповіді на запит представлено на схемі, яка зображена на рисунку 2.2 [11].

У Python виконання таких операцій, як читання, перетворення та запис аудіо, можливе за допомогою бібліотек Librosa, PyDub, scipy.io.wavfile, playsound та інших. Також Python має широкий вибір фреймворків і бібліотек для реалізації розпізнавання мовлення та створення голосових асистентів. Серед них можна виділити Microsoft Azure Speech Services, Google Cloud Speech-to-Text API, pocketsphinx, Mozilla DeepSpeech, Nvidia Nemo, Kaldi, SpeechRecognition та багато інших.

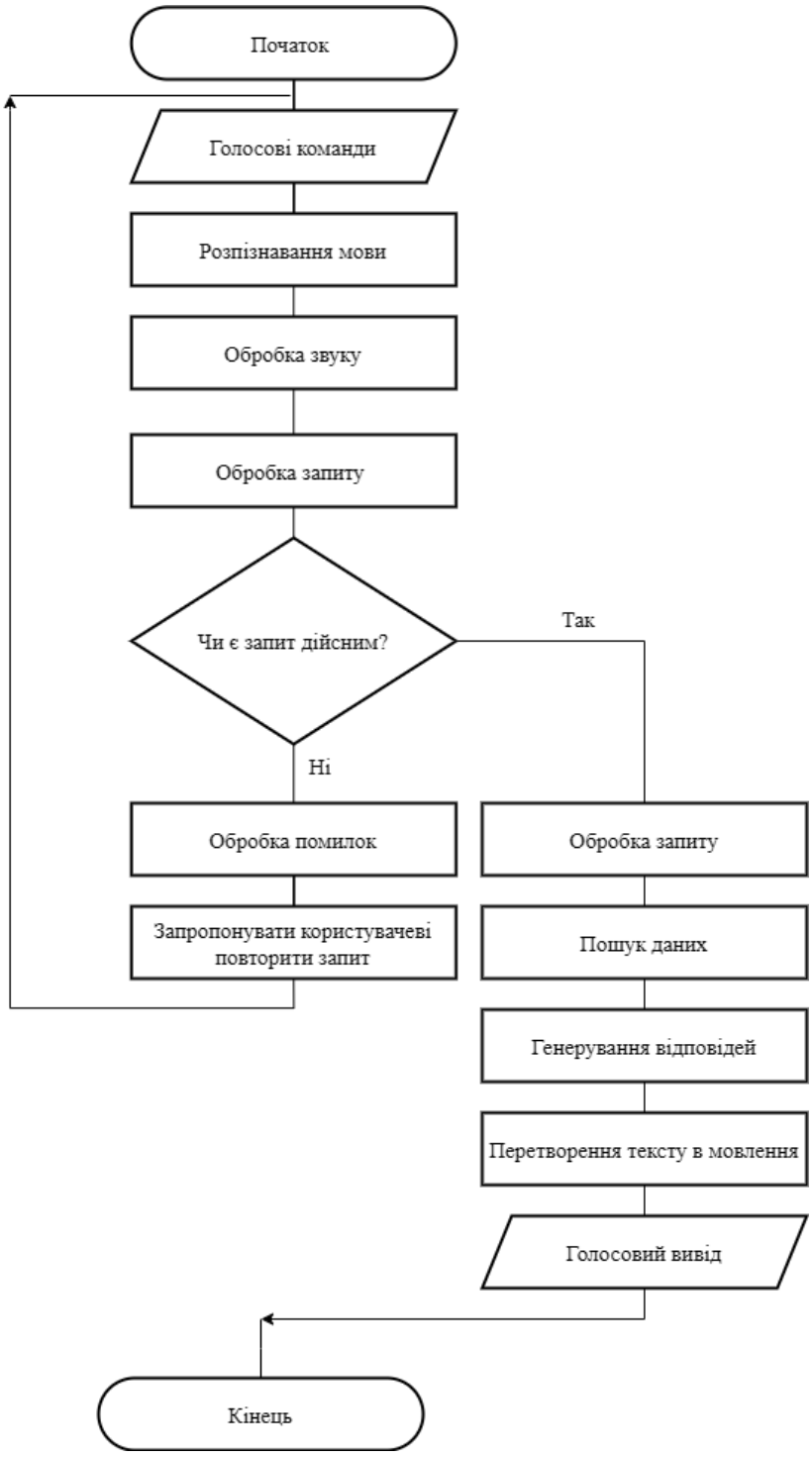

Рисунок 2.2 - Схема роботи голосового помічника

2.3 Інформаційне зебезпечення

Інформаційне забезпечення голосового помічника – це комплекс структурованих даних, знань, мовних моделей та довідкових ресурсів, що гарантують правильну та ефективну роботу системи. Воно включає як вхідну, так і вихідну інформацію, а також базу знань, з якою система взаємодіє з користувачем.

Робота Гглосового помічника починається з фіксації голосу через мікрофон, після чого отриманий аудіосигнал передається до модуля розпізнавання мовлення. Цей модуль, використовуючи мовні моделі та алгоритми глибинного навчання, перетворює мовлення в текстовий формат. Далі текст обробляється голосовим процесором, який аналізує зміст, виділяє ключові слова та фрази, а також визначає наміри користувача. Для цього часто застосовується система обробки природної мови (NLP), що дозволяє враховувати як синтаксис, так і контекст команди. Наступним етапом є виконання відповідних дій: пошук інформації, встановлення нагадувань, управління пристроями або надання відповіді. При цьому активно використовується база даних, яка є центральним джерелом збереження та доступу до потрібної інформації.

Вона зберігає профілі користувачів, історію команд, персональні налаштування, календарі, списки контактів, а також інші структуровані дані. База даних виконує не лише функцію сховища, а й забезпечує швидкий доступ до даних, їхню обробку та персоналізацію запитів. Важливу роль відіграє також інтеграція з зовнішніми джерелами, такими як інтернет-сервіси, хмарні платформи або інші застосунки. Завдяки цьому помічник може отримувати додаткову інформацію, надсилати повідомлення, здійснювати пошук, додавати календарні події чи нагадування. Всі ці компоненти працюють злагоджено, створюючи зручну, адаптивну та ефективну систему, що дозволяє користувачу взаємодіяти з технікою природним способом – через голос.

Словники та мовні моделі є ключовими компонентами при розробці голосових помічників, оскільки вони забезпечують машині здатність розуміти та відтворювати людську мову в різних її аспектах. Словники — це структуровані сховища даних, які містять лексеми (слова), доповнені інформацією про їх

фонетичні, морфологічні, синтаксичні та семантичні властивості. Завдяки словникам система може "знати", як вимовляється слово, до якої граматичної категорії воно належить (наприклад, іменник, дієслово), які має форми та як поєднується з іншими словами. Також вони містять синонімічні ряди, що дає змогу розуміти варіативність мовлення та гнучкіше обробляти команди користувача.

Мовні моделі працюють не з окремими словами, а з їхніми комбінаціями та контекстами. Вони створюються на основі аналізу великих текстових масивів і показують ймовірність появи певного слова чи фрази після інших слів. Приміром, мовна модель може "знати", що після фрази "встанови будильник на" найімовірніше буде "сім годин ранку", а не якесь інше слово. Такі знання сприяють як кращому розпізнаванню мовлення (зокрема, в умовах шуму або неповного вхідного сигналу), так і точнішому формулюванню відповідей. Це реалізується завдяки поєднанню статистичних методів, алгоритмів глибокого навчання, рекурентних нейронних мереж (RNN), трансформерів, а також сучасних моделей, таких як GPT, BERT чи Whisper.

Словники та мовні моделі активно використовуються і на етапі генерації природної мови, коли голосовий помічник формулює відповідь на запит користувача. Наприклад, замість формального "Результатів не знайдено" система може відповісти: "На жаль, я не знайшов потрібної інформації, чи не спробуєте щось інше?", що робить взаємодію більш людяною. У багатьох випадках словники містять спеціальні терміни (наприклад, з медицини або техніки), адаптуючи систему до конкретного користувача чи завдання.

Отже, словники надають основні знання про мову, а мовні моделі – механізми її використання в режимі реального часу, спираючись на ймовірнісні та контекстуальні зв'язки. Разом вони формують основу, без якої голосовий помічник не зміг би адекватно сприймати, інтерпретувати та генерувати мовлення, перетворюючи голосові інтерфейси з просто технологічної інновації на повноцінний інструмент для щоденної взаємодії.

Набір даних – це зібрання прикладів.

Чимало наборів зберігають дані у вигляді таблиць (сіток), наприклад, як значення, розділені комами (CSV), або безпосередньо з електронних таблиць чи

таблиць баз даних. Таблиці – це інтуїтивно зрозумілий формат вводу для моделей машинного навчання. Уявіть, що кожен рядок таблиці – це приклад, а кожен стовпець – можлива ознака або мітка. Водночас набори даних можна отримати з інших форматів, зокрема з файлів журналів і буферів протоколів.

Яким би не був використаний формат, ефективність моделі машинного навчання повністю залежить від якості даних, на яких її тренують. У цьому розділі будуть розглянуті ключові характеристики даних.

У наборі може бути багато типів даних, зокрема, але не виключно:

- числові дані, описані в іншому модулі;
- категорійні дані, описані в іншому модулі;
- людська мова – від окремих слів і речень до цілих текстових документів;
- мультимедійні дані (наприклад, зображення, відео й аудіофайли);
- вихідні дані інших систем машинного навчання;
- вектори ембедингу.

За загальним правилом, прикладів, на яких слід навчати модель, має бути принаймні в десять (або двадцять) разів більше, ніж навчальних параметрів. Однак хороші моделі зазвичай навчаються на *значно* більшій кількості прикладів.

Моделі, навчені на великих наборах даних із незначною кількістю ознак, зазвичай перевершують ті, які тренувалися на невеликих наборах даних, маючи багато ознак. Традиційно Google дуже успішно навчає прості моделі на великих наборах даних.

Набори даних для різних програм машинного навчання можуть вимагати абсолютно різної кількості прикладів для створення корисної моделі. Для деяких відносно простих задач, імовірно, буде достатньо кількох десятків прикладів, а для інших – і трильйона може бути замало.

Невеликий набір даних може дати хороші результати, якщо використовується для адаптації наявної моделі, яку вже навчали на великій кількості даних за тією самою схемою.

В основному потрібна висока якость даних, але це настільки розмите визначення, що його можна трактувати по-різному. З якісним набором даних модель видає бажані результати. Набір даних низької якості їй це ускладнює.

Високоякісний набір даних, як правило, також є надійним . Саме тому використовується в роботі WolframAlpha. WolframAlpha це міцний обчислювальний апарат знань, розроблений компанією Wolfram Research. Він радикально відрізняється від пошукових систем, таких як Google, оскільки не здійснює пошук сторінок в Інтернеті, а використовує власну базу знань і алгоритми для обробки запитів, надаючи конкретні відповіді чи обчислення.

Вона не є типовим пошуковиком, на кшталт Google, бо не вишукує відомості в мережі, а працює зі своєю базою знань, проводячи обчислення для отримання чіткої відповіді на запит. WolframAlpha може розуміти розмовну мову, трансформувати запит у формальний математичний чи логічний вигляд, знаходити потрібні дані з перевірених джерел та надавати результат у вигляді тексту, графіків або таблиць. В основі лежить масштабна структурована база даних - Wolfram Knowledgebase - де зібрані відомості з різних галузей: від математики та фізики до географії, економіки та медицини. Усе це обробляється з використанням мови програмування Wolfram Language, що також застосовується в системі Mathematica. До того ж, WolframAlpha володіє потужним API, котрий дозволяє розробникам інтегрувати її обчислювальні функції у власні програми.

База знань WolframAlpha слкадається з того що не використовується інтернет-пошуково індексацію але опраєтсья вона на певні структуровані набори, а саме :

- перевірені дані (наукові бази, урядові джерела, академічні публікації);
- структуровані набори даних (таблиці, графіки, формули);
- символьні обчислення (математика, логіка, формальні системи);
- алгоритми обробки природної мови (NLP для розуміння запитів).

У використанні надзвичайно потужна обчислювальна система знань, що використовує структурувану базу даних, яка поєднує кураторські джерела (Wolfram Knowledgebase) з перевіреними зовнішніми ресурсами. Внутрішня база містить формалізовану інформацію з різних сфер: математики (формули, рівняння), фізики (фундаментальні константи), хімії (молекулярні структури), історії

(хронології), географії (координати, демографія) та фінансів (біржові показники, ВВП). Для підтримки актуальності, система інтегрує дані з зовнішніх джерел: статистику ООН та ВООЗ, астрономічні дані NASA, економічні звіти Світового банку, наукові публікації з arXiv та PubMed, а також офіційну статистику урядів різних країн. Вся інформація зберігається у вигляді взаємопов'язаних сутностей (наприклад, `Entity ["Country", "Ukraine"]`), що дає можливість WolframAlpha автоматично оновлювати дані, встановлювати зв'язки між ними (наприклад, між кліматом та економікою) та видавати точні, обчислювані результати - від графіків до розв'язків рівнянь у символьній формі. На відміну від ChatGPT, що генерує тексти, WolframAlpha працює як «машина знань», перетворюючи запити на структуровані результати, базуючись на достовірних даних. Приклад того як відрізняється WolframAlpha від ChatGPT наведений в таблитці 2.1 [10].

Таблиця 2.1 – Порівняння WolframAlpha та ChatGPT

Критерій	WolframAlpha	ChatGPT
Тип системи	Бази знань і обчислення	Генеративний ШІ
Відповіді	Точні на основі даних	Творчі, та можуть бути не точні
Математика	Здатність розявзувати рівняння та інтграли	Не точні обрахунки , помилки в складних обрахунках
Джерела	Структуровані набори даних	Тренувальні набори даних
Візуалізація	Графіки, таблиці, 3D-моделі	Текст, генеровані моделі

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Реалізація програмоного забезпечення

Структурна схема програмного забезпечення є ключовим елементом для аналізу компонентів голосового помічника та їх взаємодії. Вона слугує основою для створення чіткого уявлення про архітектурні особливості системи, що дає змогу розробникам і інженерам детально зрозуміти процес узгодженого функціонування окремих модулів у рамках інтегрованого голосового інтерфейсу. У випадку голосового помічника на основі штучного інтелекту архітектура системи формується з декількох взаємопов’язаних модулів: модуль розпізнавання мовлення виконує перетворення звукового сигналу в текстові дані; модуль обробки природної мови забезпечує семантичний аналіз запиту користувача; логічне ядро відповідає за прийняття рішень щодо виконання необхідної дії; додаткові компоненти для доступу до баз даних, зовнішніх API або сервісів, таких як WolframAlpha чи погодні інтерфейси, сприяють розширенню функціональності системи; а модуль синтезу мовлення формує аудіовідповідь, яка передається користувачеві.

Використання такої схемної організації є фундаментом для створення зручної, інтелектуально розвиненої системи, спрямованої на оптимізацію взаємодії між людиною та комп’ютером.

Ключовим технологіями у WolframAlpha є :wolfram language, mathematica, семантичний аналіз (nlp), wolfram knowledgebase, хмарні обчислення, машинне навчання, візуалізація.

Wolfram Language унікальна мова програмування, призначена для здійснення символьних обчислень. Володіє здатністю оперувати математичними виразами, як цілісними об'єктами (наприклад, без проблем розв'язує інтеграли або диференціальні рівняння в автоматичному режимі) [12].

Mathematica платформа для складних обчислень і візуалізації. Використовується для: побудови 3d-графіків, розв'язання оптимізаційних задач машинного навчання.

Семантичний аналіз проходить аналіз запиту у природній мові наприклад «Скільки мешканців у Києві?» і перетворює їх у формальні обчислювальні структури «`CityData ["Kyiv", "Population"])`».

Wolfram Knowledgebase власна база структурованих даних, що включає у себе фізичні константи, формули, закони тощо. Велику кількість задокументованих історичних подій. Величезну кількість фінансових та економічних показників. Усі хімічні сполуки назви їх елементи та формули.

Хмарні обчислення та запити опрацьовуються на високопродуктивних серверах, забезпечуючи можливість оперативно інтегрувати інформацію з API, скажімо, динаміку обмінних курсів. Використовуються паралельні обчислення для ефективної роботи зі складними завданнями.

Машинне навчання застосовується для класифікації типів запитів, покращення розпізнавання природної мови, птимізації алгоритмів (наприклад, вибору методу інтегрування)

Для візуалізації проводить автоматичні генерацію інтерактивних графіків, географічних карт, молекулярні моделей даних на основі реальних перевіриних даних.

Розпізнавання мовлення, галузь на перетині лінгвістики, інформатики та електротехніки, спрямована на розробку систем, здатних розпізнавати та перетворювати розмовну мову в текст. Python, відомий своєю простотою та надійними бібліотеками, пропонує кілька модулів для ефективного вирішення завдань розпізнавання мовлення. Дослідимо суть розпізнавання мовлення в Python, включаючи огляд його ключових бібліотек, способів їх реалізації та їхнього практичного застосування.

Важливим етапом у процесі розробки будь-якого програмного забезпечення є вибір мови програмування. У випадку створення індивідуального голосового помічника було прийнято рішення використовувати мову Python.

Python — це високорівнева мова програмування загального призначення, яка відзначається читабельністю коду завдяки використанню значних відступів. Python має динамічну типізацію та автоматичне збирання сміття. Вона підтримує різноманітні парадигми програмування, включаючи структурне (зокрема

процедурне), об'єктно-орієнтоване та функціональне програмування. Часто цю мову називають "батарейкою в комплекті" через її обширну стандартну бібліотеку.

Серед основних її переваг можна назвати такі:

– чистий синтаксис (для виділення блоків слід використовувати відступи);

– переносність програм (що властиве більшості інтерпретованих мов);

– стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);

– можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);

– стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке називається IDLE і яке написане мовою Python;

– зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);

– відкритий код (можливість редагувати його іншими користувачами).

Мова програмування Python займає важливе місце у галузях штучного інтелекту, машинного навчання та глибинного навчання. Така популярність пояснюється її потужністю, гнучкістю та широким набором вбудованих і сторонніх бібліотек та інструментів. Завдяки цьому Python забезпечує зручність і ефективність виконання досліджень, аналізу даних і розробки моделей у зазначених сферах.

Для написання програмного коду голосового асистента було обрано середовище PyCharm. Це одна з найпотужніших і найпопулярніших інтегрованих середовищ розробки (IDE) для мови програмування Python, створена компанією JetBrains з акцентом на ефективну роботу з Python-кодом. PyCharm пропонує безліч корисних функцій: інтелектуальне автозаповнення, автоматичний пошук помилок, зручне налагодження, інтеграцію з віртуальними середовищами, підтримку систем контролю версій і бібліотек. Усе це дозволяє зробити процес розробки голосового асистента більш зручним, організованим і швидким. Крім того, PyCharm забезпечує легке підключення зовнішніх бібліотек для розпізнавання мовлення, синтезу звуку

та роботи з API, що робить його оптимальним варіантом для створення інтелектуальних голосових систем [12].

Передусім, PyCharm вирізняється своєю легкістю та швидкодією, пропонує зручний інтерфейс, який сприяє максимальній продуктивності. Він допомагає розробникам зосередитися на процесі написання коду, мінімізуючи вплив непотрібних відволікаючих елементів [13].

PyCharm забезпечує вбудовану підтримку для різноманітних мов програмування та платформ, таких як HTML, CSS, JavaScript, SQL, YAML, Markdown, Django, Flask, FastAPI багато інших. Це робить його дуже універсальним та зручним інструментом для розробки різного роду програмного забезпечення.

По-третє, PyCharm може значно розширити свої функціональні можливості завдяки численним плагінам. Багато з них є безкоштовними і доступними через вбудований менеджер плагінів. Серед них можна знайти інструменти для роботи з додатковими мовами програмування, інтеграції з фреймворками, базами даних, системами керування проєктами тощо. Це дозволяє швидко і зручно налаштувати середовище розробки відповідно до індивідуальних потреб користувача. PyCharm здатний задовольнити потреби як початківців, так і досвідчених розробників у Python-програмуванні.

Ключові бібліотеки Python для розпізнавання мовлення:

- SpeechRecognition, яка є однією з найпопулярніших бібліотек Python для розпізнавання мовлення. В процес водить забезпечення підтримки кількох рушіїв та API, таких як Google Web Speech API, Microsoft Bing Voice Recognition та IBM Speech to Text. Відома своєю простотою використання та гнучкістю, що робить її чудовою відправною точкою як для початківців, так і для досвідчених розробників [14].
- PyAudio який, необхідний для введення та виведення аудіо в Python, забезпечує зв'язки Python для PortAudio, кросплатформної бібліотеки введення/виведення аудіо. Він часто використовується разом із SpeechRecognition для захоплення вхідного сигналу з мікрофона для розпізнавання мовлення в режимі реального часу.

- DeepSpeech — це система розпізнавання голосу з відкритим кодом на основі глибокого навчання, яка використовує моделі, навчені в рамках дослідницького проекту Baidu Deep Speech. Вона підходить для розробників, які прагнуть реалізувати складніші функції розпізнавання мовлення за допомогою можливостей глибокого навчання.

Реалізація розпізнавання мовлення за допомогою Python. Базова реалізація з використанням бібліотеки SpeechRecognition включає кілька кроків:

– захоплення аудіо з мікрофона за допомогою PyAudio;

– обробка аудіо що полягає перетворенні аудіосигналу на дані, з якими може працювати бібліотека розпізнавання мови;

– розпізнавання для якого проводиться виклик методу recognize_google() (або іншого доступного методу розпізнавання) у бібліотеці SpeechRecognition для перетворення аудіоданих у текст.

Приклад коду наведений в наступному лістингу:

```
import speech_recognition as sr
with sr.Microphone() as source:
    print("Talk")
    audio_text = r.listen(source)
    print("Time over, thanks"
    try:
        print("Text: "+r.recognize_google(audio_text))
    except:
         print("Sorry, I did not get that")
```

Вивід:

```
Привіт, це Джон
```

Весь код програмного модуля наведений в додатку Б.

Практичне застосування для розпізнавання мовлення має широкий спектр використання:

– голосові помічники (для створення персональних помічників, таких як Siri або Alexa);

– інструменти доступності (допомога людям з інвалідністю у взаємодії з технологіями);

– домашня автоматизація (увімкнення голосового керування пристроями розумного дому);

– послуги транскрипції (автоматична транскрипція зустрічей, лекцій та інтерв'ю).

Під час інтеграції систем розпізнавання мови, програмісти можуть наштовхнутися на перешкоди, такі як сторонні шуми, вимова та місцеві говірки. Надзвичайно важливо брати до уваги ці аспекти та проводити випробування застосунку в різноманітних середовищах. До того ж, слід приділяти увагу питанням збереження приватності та моралі, зокрема, коли йдеться про обробку чутливих аудіофайлів.

Розпізнавання мовлення, здійснене засобами Python, у парі з WolframAlpha, є неймовірно продуктивним тандемом при розробці інтелектуальних голосових інтерфейсів. Бібліотеки, на кшталт `SpeechRecognition`, `PyAudio` і `DeepSpeech`, спрощують завдання для розробників, дозволяючи впроваджувати програми, здатні розпізнавати природну мову користувача без особливих зусиль. Сформований голосовий запит передається до WolframAlpha, яка, вдаючись до поєднання обчислювальних алгоритмів, елементів штучного інтелекту та структурованої бази знань, не просто проводить пошук відповідей. Вона ретельно аналізує суть, обчислює результат і візуалізує його у приємному для сприйняття вигляді: текстовому, числовому чи графічному. Завдяки цьому поєднанню, стає можливим створення складних розмовних систем, здатних як розпізнавати голос користувача, так і надавати точну, аргументовану інформацію. Як наслідок, досягаємо нового рівня взаємодії людини з машиною — інтелектуального, адаптивного та природного.

3.2 Інтерфейс користувача

Користувацький інтерфейс — це комунікація людини і технології. Він повинен бути зрозумілим, інтуїтивним і зручним, щоб дозволити користувачеві виконувати свої завдання без зайвих ускладнень. Основна задача розробника — створити дизайн, який відповідає потребам користувачів, враховуючи їхні звички та очікування. Ефективний інтерфейс не лише вирішує технічні проблеми, але й робить роботу з програмою приємною та продуктивною. Важливо, щоб кожен елемент мав логічне обґрунтування і сприяв досягненню цілей програми. Простота, зрозумілість і привабливість — це три основи, на яких базується якісний користувацький досвід. Інтерфейс має бути не тільки функціональним, але й адаптивним, щоб задовольняти вимоги різних користувачів в різних умовах використання.

Зручний інтерфейс — це той, яким користуються інтуїтивно, без зайвих роздумів. Він дозволяє виконувати завдання легко і природно, так ніби взаємодієш із розумним помічником. Кожен елемент повинен займати своє правильне місце, а функції мають бути доступними саме тоді, коли вони необхідні. Інтерфейс слід сприймати як супутника: він веде користувача в процесі роботи, не нав’язуючи власний сценарій взаємодії. У найкращому випадку робота з програмою стає такою ж невимушеною, як саме дихання — без напруги чи складностей.

Ефективність інтерфейсу вимірюється тим, наскільки швидко і безпомилково людина досягає своєї мети під час його використання. Навіть найскладніші системи можуть бути зручними і зрозумілими, якщо їхній дизайн продуманий до деталей. Тут важливо брати до уваги не лише технічні аспекти, а й психологію сприйняття: кольори, розмір елементів та логіку їх розташування, а також послідовність дій. Ці фактори суттєво впливають на те, наскільки позитивним буде досвід користувача. Інтерфейс повинен спілкуватися мовою самого користувача, а не висловлюватися термінами розробників. Це означає, що тексти підписів, підказок і повідомлень повинні бути максимально зрозумілими і не викликати додаткових запитань. Користувач не має ламати голову над тим, що значить якась іконка чи куди

спрямовує певна кнопка. Ідеальний інтерфейс – це той, де все зрозуміло з першого погляду, навіть якщо це перший досвід роботи з програмою такого типу.

У випадку голосового помічника інтерфейс мінімальний, так як головна задача полекшити роботу користувача та копютера. В інтерфейсі є основні фукнкції – це голосовий ввід та ввід текстовий ввід а також закритя програми , основна робота програми це виклик фукнкцій за ключовими словами.

Основні функції програми це голосовий ввід але також є кнопки і ось як це представлено в коді . Даний код зображений на рисунку 3.1 [15] .

```
def create_widgets(self):
    # Текстове поле для виводу
    self.output_text = scrolledtext.ScrolledText(self.root, width=70, height=20, wrap=tk.WORD)
    self.output_text.pack(pady=10)
    self.output_text.insert(tk.END, "Anatoliy 1.5: Hello Я ваш голосовий помічник.\n")

    # Поле для текстового введення
    self.input_frame = tk.Frame(self.root)
    self.input_frame.pack(pady=5)
    self.input_label = tk.Label(self.input_frame, text="Ваш запит:")
    self.input_label.pack(side=tk.LEFT)
    self.input_entry = tk.Entry(self.input_frame, width=50)
    self.input_entry.pack(side=tk.LEFT, padx=5)
    self.input_entry.bind("<Return>", self.process_text_command)

    # Кнопки
    self.button_frame = tk.Frame(self.root)
    self.button_frame.pack(pady=5)
    self.voice_button = tk.Button(self.button_frame, text="Голосовий ввід", command=self.start_voice_input)
    self.voice_button.pack(side=tk.LEFT, padx=5)
    self.text_button = tk.Button(self.button_frame, text="Текстове відправлення", command=self.process_text_command)
    self.text_button.pack(side=tk.LEFT, padx=5)
    self.clear_button = tk.Button(self.button_frame, text="Очистити", command=self.clear_output)
    self.clear_button.pack(side=tk.LEFT, padx=5)
```

Рисунок 3.1 – Код інтерфейсу програми

На рисунку 3.2 зображено інтерфейс програми "Анатолій 1.5 - Голосовий помічник" виконано у вигляді текстового поля, яке демонструє діалог між користувачем і асистентом. У верхній частині інтерфейсу розміщено привітальні повідомлення від помічника, що містять англійське "Hello" та українське вітання з коротким представленням. Нижче текстового виведення знаходиться поле для введення запитів, де користувач може обрати між голосовим введенням (кнопка "Голосовий ввід") або текстовим (кнопка "Текстове відправлення"). Додатково передбачена кнопка "Очистити", яка видаляє історію діалогу. Дизайн інтерфейсу витримано в мінімалістичному стилі з пріоритетом на функціональність, основний

акцент зроблено на текстовому полі, яке забезпечує зручну взаємодію та створює ілюзію природного спілкування між користувачем та голосовим помічником [16].

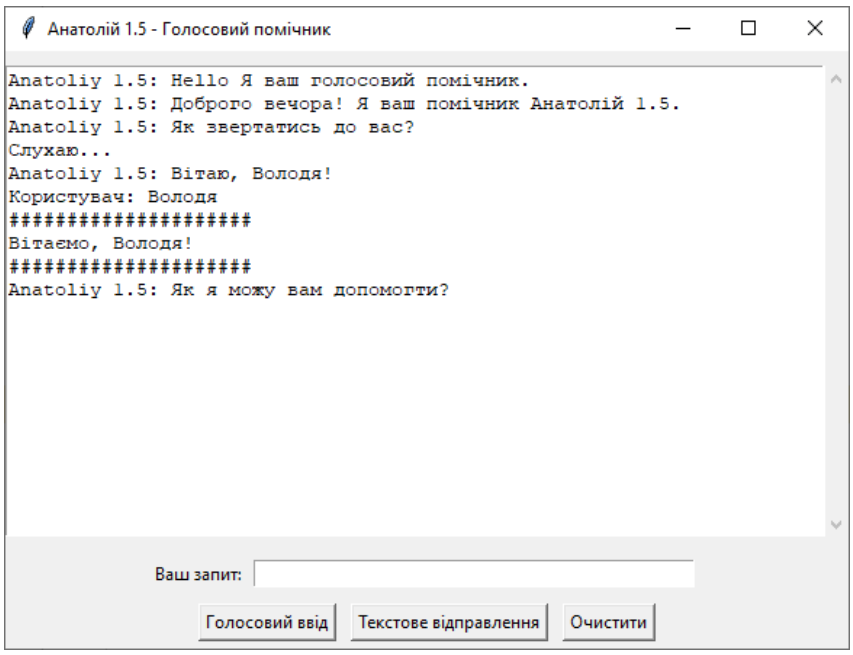

Рисунок 3.2 – Інтерфейс для роботи з голосовим асистентом

Голосовий інтерфейс запитує як звуть того хто буде працювати з в своєрідному діалозі і далі він готовий отримуват команди і виконувати їх, реалізація зображена на рисунку 3.3.

```
def get_username_voice(self):
    try:
        r = sr.Recognizer()
        with sr.Microphone() as source:
            self.output_text.insert(tk.END, "Слухаю...\n")
            r.pause_threshold = 1
            audio = r.listen(source)

        uname = r.recognize_google(audio, language='uk-UA')
        self.process_username(uname)
    except Exception as e:
        self.speak("Не вдалося розпізнати ім'я")
        # Пропонуємо ввести текстовий варіант
        self.output_text.insert(tk.END, "Будь ласка, введіть ваше ім'я в текстовому полі та натисніть Enter\n")
```

Рисунок 3.3 – Реалізація «знайомства» з користувачем

Розберемо осоновні складові цього коду який реалізує функцію голосового введення імені для голосового помічника, використовуючи бібліотеку `speech_recognition`. Завдяки цій бібліотеці аудіо, отримане через мікрофон,

перетворюється в текст. Робота починається з ініціалізації розпізнавача мови та виведення індикатора "Слухаю...", який сигналізує про початок запису. Для плавності роботи параметр `pause_threshold` встановлений на 1 секунду, що дозволяє системі коректно визначати завершення фрази. Після запису аудіо система намагається розпізнати сказане користувачем, орієнтуючись на українську мову ('uk-UA'). Це дає змогу точно обробляти імена українською мовою. У разі успішної обробки розпізнане ім'я передається у функцію `process_username` для подальшого використання. Якщо ж виникають труднощі, наприклад, проблеми з мікрофоном або нечітка вимова, програма повідомляє про це як голосовим супровідом, так і текстом. У таких випадках користувач може ввести ім'я вручну через текстове поле. Такий підхід створює зручний і гнучкий користувацький досвід: голосовий ввід залишається пріоритетним, але надається резервний текстовий варіант для забезпечення стабільного функціонування програми.

Розглянемо функціонування трьох основних кнопок. Після того як користувач представиться, голосовий помічник очікує на запит, який потрібно виконати, запитуючи: «Як я можу вам допомогти?» Після цього у користувача з’являється вибір: скористатися голосовим введенням або текстовим, залежно від його потреб. Реалізація голового вводу зображена на рисунку 3.4 [17].

```
def take_command(self):
    r = sr.Recognizer()
    with sr.Microphone() as source:
        self.output_text.insert(tk.END, "Слухаю...\n")
        self.output_text.see(tk.END)
        r.pause_threshold = 1
        audio = r.listen(source)

    try:
        self.output_text.insert(tk.END, "Розпізнавання...\n")
        self.output_text.see(tk.END)
        query = r.recognize_google(audio, language='uk-UA')
        self.output_text.insert(tk.END, f"Ви сказали: {query}\n")
        self.output_text.see(tk.END)
        self.process_query(query.lower())
    except Exception as e:
        self.speak("Не вдалося розпізнати ваш голос")
        self.output_text.insert(tk.END, f"Помилка: {str(e)}\n")
        self.output_text.see(tk.END)
```

Рисунок 3.4 – Реалізація голосового вводу команди

Код, який реалізує функціонал кнопки для введення тексту зображени на рисунку 3.5 [18].

```
def process_text_command(self, event=None):
    query = self.input_entry.get()
    if query:
        self.output_text.insert(tk.END, f"Ви: {query}\n")
        self.output_text.see(tk.END)
        self.input_entry.delete(0, tk.END)
        self.process_query(query.lower())
```

Рисунок 3.5 – Реалізація текстового вводу команди

Однією із основних команд для розширення можливостей голосового помічника застосовується бібліотека WolframAlpha. Вона забезпечує широкі функціональні можливості, включаючи обробку запитів, виконання обчислень і отримання достовірної інформації в реальному часі. Завдяки інтеграції з WolframAlpha значно підвищується інтелектуальний рівень помічника, а також розширюється набір підтримуваних команд, що дозволяє користувачам отримувати швидкий та точний доступ до необхідних знань. Особливості цієї команд що запити потрібно їй надисали тільки англійською і починатись вони повинні на «What is …», «Who is…»і результат видає також англійською мовою, код який нам дає можливість користуватись програмаю зображено на рисунку 3.6.

```
def answer_question(self, query):
    """Answer questions in English only using WolframAlpha"""
    try:
        # Verify the query contains only English characters
        if not all(ord(char) < 128 for char in query):
            self.speak("Please ask in English")
            self.output_text.insert(tk.END, "Error: Only English questions supported\n")
            return

        client = wolframalpha.Client("                ")
        res = client.query(query)
        result = next(res.results).text

        self.speak(result)
        self.output_text.insert(tk.END, f"Answer: {result}\n")

    except StopIteration:
        self.speak("No results found")
        self.output_text.insert(tk.END, "No results found\n")
    except Exception as e:
        self.speak("Search error")
        self.output_text.insert(tk.END, f"Error: {str(e)}\n")
```

Рисунок 3.6 – Реалізація використання бібліотеки «WolframAlpha»

Єдина особливість це те, що запит потрібно давати англійською мовою і результат також буде видаватись англійсою мовою. Для наочності процесу роботи програми наведемо приклад, у якому сформулюємо кілька запитів, що розпочинаються словами «What is…» «Who is…». Потім проаналізуємо отримані, що демонструють основні етапи та функціональність програми в дії. У ході цього прикладу зможемо детально розглянути, як система обробляє запити, формує відповіді та презентує інформацію користувачеві у зрозумілому вигляді на рисунку 3.7.

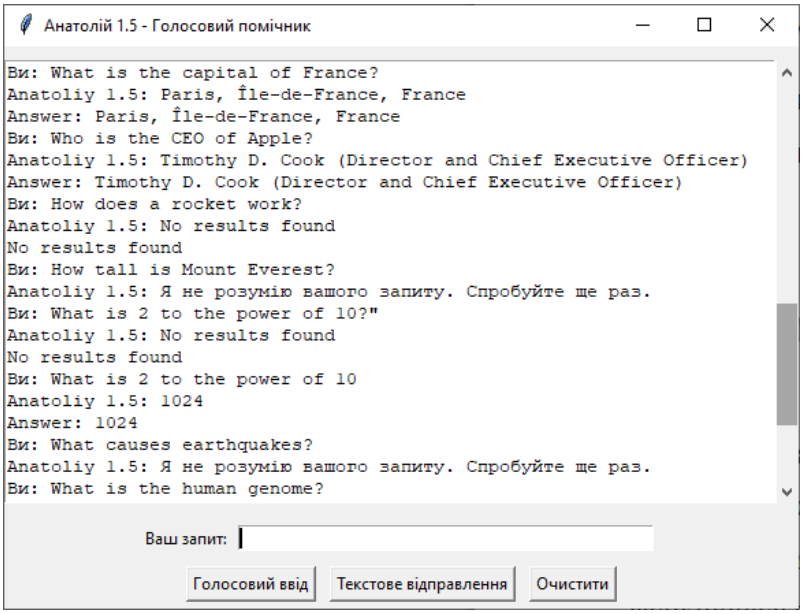

Рисунок 3.7 – Робота бібліотеки «WolframAlpha»

Видно, що бібліотека працює і виводить відповіді на питання, а якщо ввести не вірне питання, то видає, так би мовити, помилку. Ця бібліотека є дуже потужною та дає змогу отримати відповіді на різні наукові, інформаційні, математичні та інші види питань [19] .

3.3 Тестування програмного забепечення

Тестування програмного забезпечення є важливим етапом у процесі створення будь-якої системи, оскільки воно допомагає виявити та виправити помилки ще до випуску виробу. Для створеної системи було обрано три основні види тестування: тестування адаптивності інтерфейсу користувача та

функціональне тестування. Ці підходи забезпечують всебічну перевірку системи, що дає змогу підтвердити її надійність, функціональність та зручність використання. Нижче наведено детальний опис процедур тестування.

Проведення тестування екрану користувача дає змогу перевірти чи зручно користуватись програмою , в випадку цієї програми її використання призначене для користувача на комюптері а розміри екрану програми не є велики там як основний фукнкціонал це голосові команди. Отже розміри сторінки є сталі і ширина сторінки 70 рядків тексту по ширині і 20 рядків по висоті, Це дає змогу зручно налаштовувати розміри відповідно до очікуваного обсягу тексту, а не конкретних пікселів (рисунок 3.8).

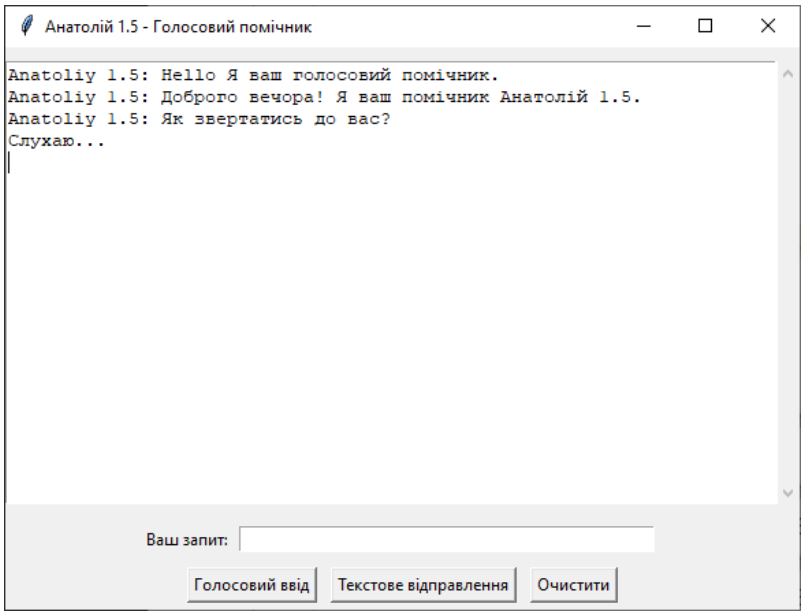

Рисунок 3.8 - Інтерфейс програми

Ці параметри є сталими і якщо користувач хоче змінити розміри вікна то робоча область залишиться в тих же розмірах, а вікно збільшиться (рисунок 3.9).

Як видно на рисунку 3.9 якщо збільшувати вікто, то робочий простір не буде міняти форми, тоді зробимо перевірку якщо зменшити вікно до розмірів які менші за вказані в самий налаштування коду рисунок 3.10.

Як видно, що при зменшені вікна до розмірів які менші за вказані і тоді втрачаємо кнопки але текст переноситься так щоб вміщався у рядок [20].

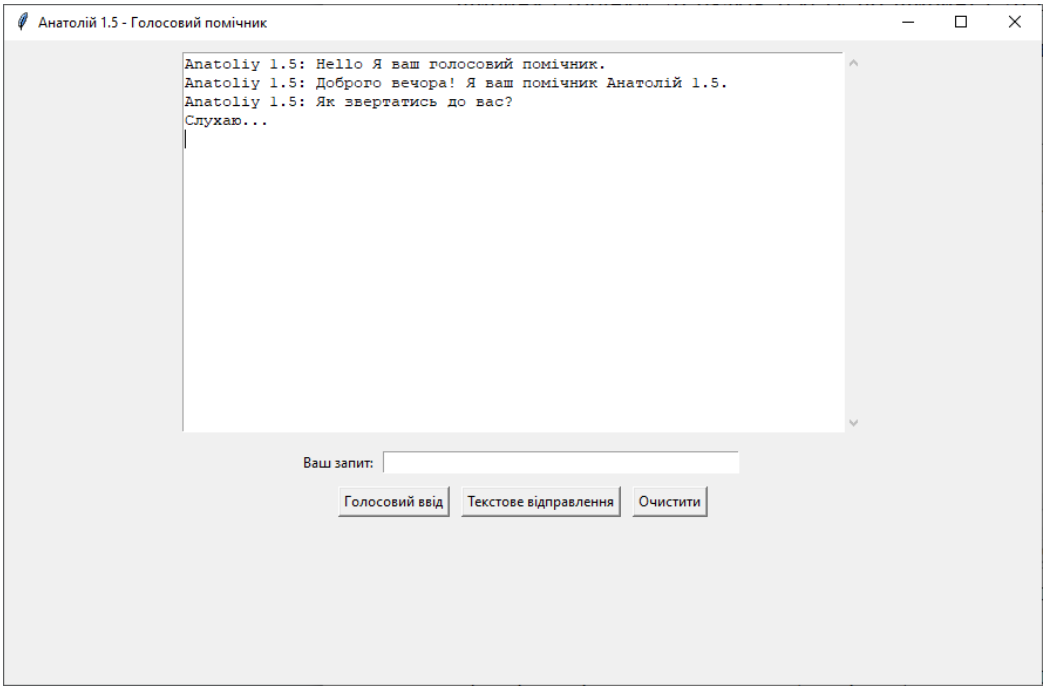

Рисунок 3.9 - Зміна розміру вікна

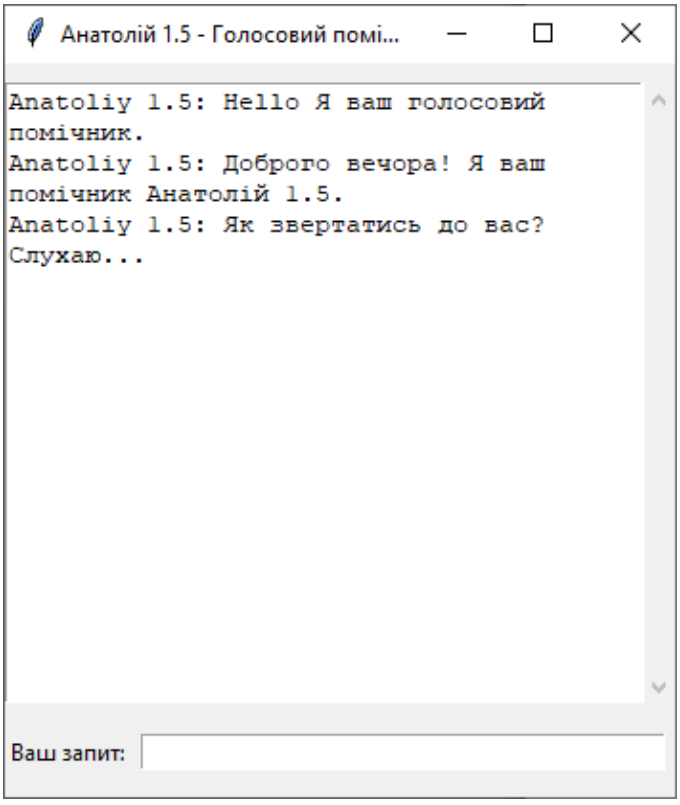

Рисунок 3.10 – Нестандартні розміри робочої програми

Функціональне тестування є важливим етапом у процесі перевірки програмного забезпечення, під час якого система оцінюється на предмет відповідності функціональним вимогам і специфікаціям. Це тестування покликане забезпечити, що застосунок адекватно виконує завдання, визначені вимогами або

специфікаціями. Особливу увагу приділяють аналізу результатів обробки, наслідуючи реальні сценарії використання системи, при цьому не враховується структура системи .

Отже розпочнемо роботу запустивши програму та скажемо йому своє імя і побачимо що програма готова до отримання задач і видачі результатів рисунок 3.11.

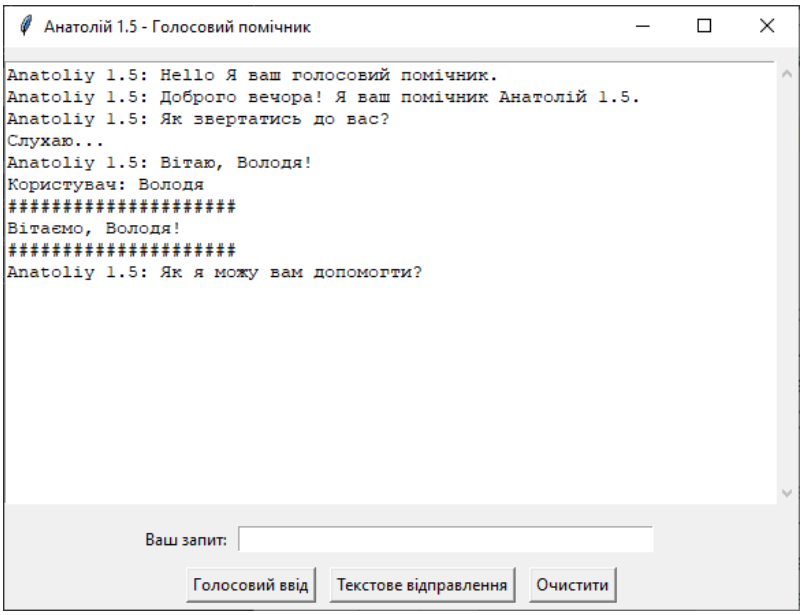

Рисунок 3.11 - Початок роботи з голосовим помічником

Отже, подаємо запит на відкриття таких програм, як «Word», «PowerPoint» та «Excel». У результаті цих дій зазначені програми мають успішно запуститися і відобразитися у головному меню для подальшого використання рисунок 3.12.

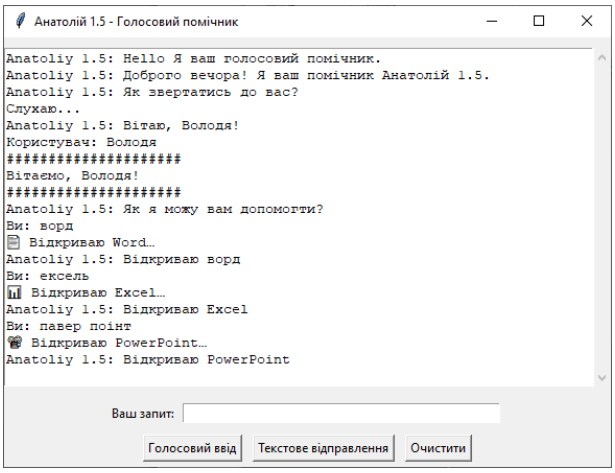

Рисунок 3.12- Результат запитів

Усі програми було відкрито, ще особливість що відбувається голосове озвучення процесу і відкритвається програма (рисунок 3.13).

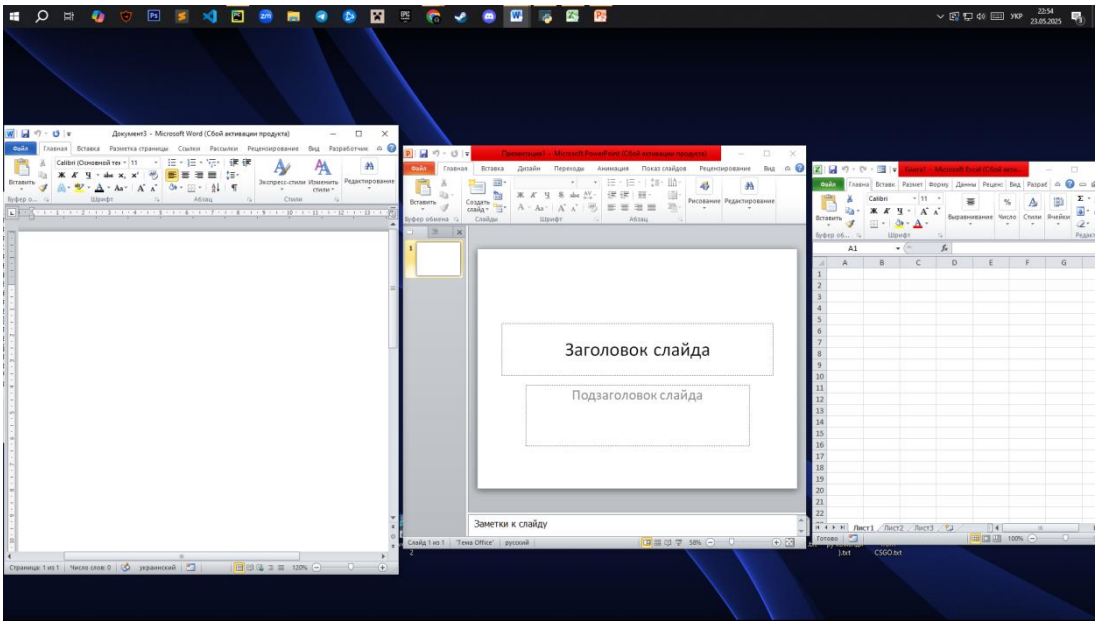

Рисунок 3.13 – Результат роботи запитів

Наступний запит буде на отримання новин, пишемо «новини» і з сайту який вказаний у логіці прогрми за допомогою бібліотеки зчитає новити на надішле на інтерфейс (рисунок 3.14).

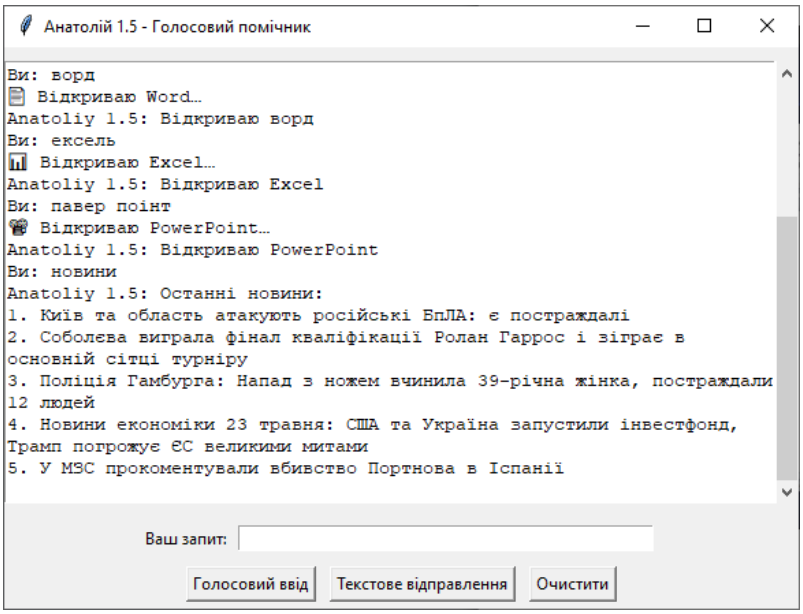

Рисунок 3.14 – Виведення останіх новин з сайту

Далі виконаємо запит до вікіпедії у випадку програми можна запросити, аби відкрило сайт вікіпедії або дати запит на те що цікавить, проте через особливості бібліотеки слід давати запит англійською мовою і результат теж буде отримано англійською мовою рисунок 3.15.

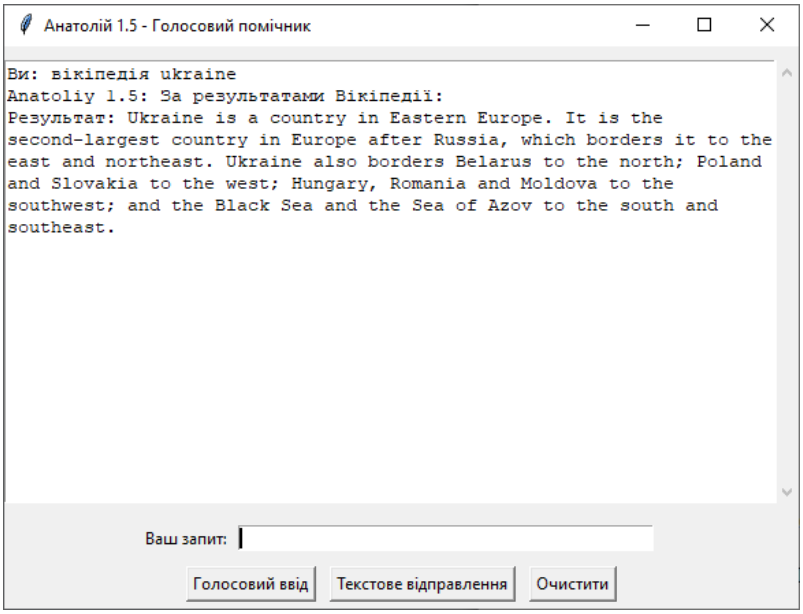

Рисунок 3.15 – Отриманий результат за запитом «вікіпедія Ukraine»

Зробимо запит для того щоб взяти погоду, дамо запит «погода» і відкриє сторінку в браузері, де буде відображатися погода за теперішнім містом рисунок 3.16.

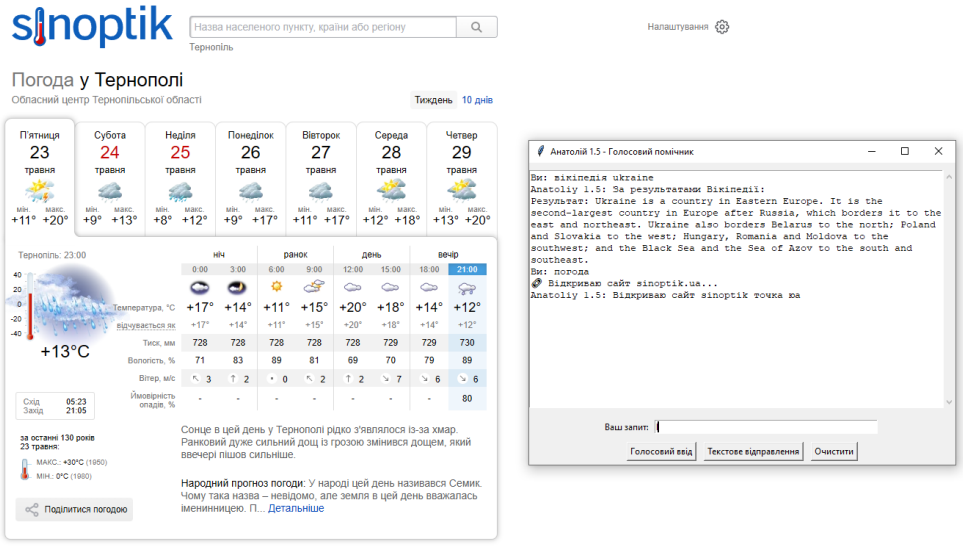

Рисунок 3.16 – Результат роботи запиту «погода»

Тепер перевіримо роботу бібліотеки «WolframAlpha» дамо декілька запитів, математичний – «What is the factorial of 5?», науковий – «What is DNA», географічний – «What is the capital ukraine» та інформаційний – «Who is the CEO of Apple?» рисунок 3.17 [21].

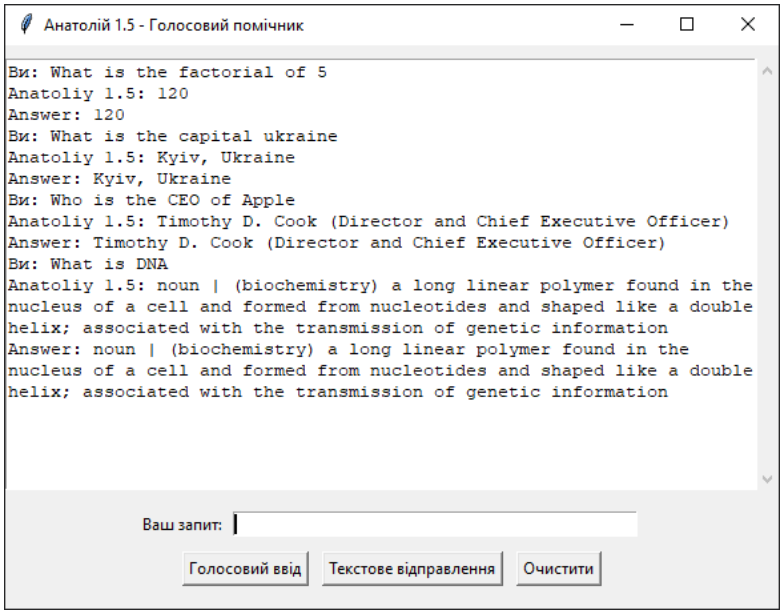

Рисунок 3.17 – Отримані запити на поставлені питання

Тут є ще кілька цікавих функцій: можна дізнатися жарт проте цей жарт англійською, а також отримати інформацію про авторів та тих, хто працював над удосконаленням програми рисунок 3.18.

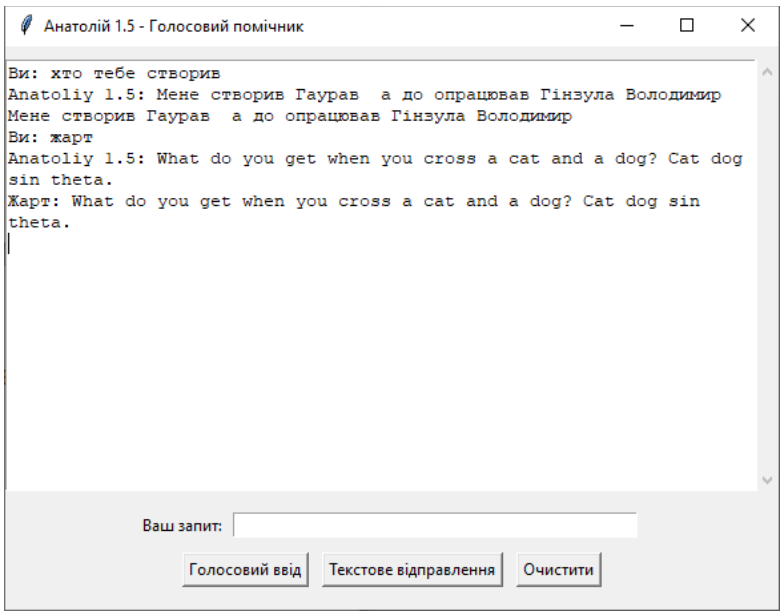

Рисунок 3.18 – Реалізації цікавих функцій

У підсумку цього розділу було проведено розробку програмного забезпечення, яка включала створення інтерфейсу користувача для забезпечення максимальної зручності взаємодії з системою, а також комплексне тестування програми, що підтвердило її належну функціональність та відповідність визначеним вимогам. Завдяки виконаній роботі вдалося розробити повноцінний і функціональний продукт [22].

ВИСНОВКИ

Метою даної кваліфікаційної роботи було розробити голосовий помічник для полегшення роботи користувачів з операційною системою, вирішення дрібних задач, які можна виконати без людини.

Під час виконання кваліфікаційної роботи було вирішено наступні завдання:

1. Здійснено огляд існуючих віртуальних асистентів, таких як Siri, Alexa та Google Assistant та проаналізовані їх функціональні можливості.
2. Проаналізовані сучасні підходи до побудови голосових помічників, визначено їх функціональні можливості, переваги та обмеження.
3. Описано архітектурну, алгоритмічне та інформаціне забезпечення системи та створено узагальнену структуру модуля голосового помічника.
4. Реалізовано програмне забезпечення з допомогою бібліотек DeepSpeech, бібліотеки SpeechRecognition, PyAudio для обробки голосу та WolframAlpha для отримання точних наукових відповідей
5. Створено інтерфейс користувача і проведено функціональне, тестування та тестування сценаріїв програми.

Представлений програмний модуль голосового помічника гарантує комфортну й дієву взаємодію користувача з комп'ютером через голосові команди. Вона використовує технології розпізнавання та синтезу мовлення, плюс обробку природної мови для аналізу запитів користувача й створення відповідей. Система показує стійку роботу, точність розпізнавання, надійність у функціонуванні та швидкий час відповіді, що дає змогу поліпшити зручність користування комп'ютерними системами та розширити доступність послуг за допомогою голосового інтерфейсу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Voice Assistant Use Cases: Business Implementations of VUIs in 2023. Електронний ресурс] - URL: https://masterofcode.com/blog/voice-assistant-use-cases-businessimplementations-of-vuis-in-2021

2. Voice Assistant Application for Avoiding Sedentarism in Elderly People Based on IoT Technologies, 2021. помічників [Електронний ресурс] - URL: https://doi.org/10.3390/electronics10080980

3. Squeezeformer-CTC XS (uk-UA). [Електронний ресурс] - URL: https://huggingface.co/theodotus/stt_uk_squeezeformer_ctc_xs

4. Siri, Alexa, Google Assistant: огляд віртуальних помічників [Електронний ресурс] - URL: https://apix- drive.com/ua/blog/reviews/siri-alexa-google-assistant-ogljad

5. Capgemini. How organizations and consumers are embracing voice and chat assistants 2019. [Електронний ресурс] URL: https://www.capgemini.com/wpcon tent /uploads/2019/09/Report-%E2%80%93-Conversational-Interfaces_WebFinal.pdf

6. ChatGPT: Jack of all trades, master of none [Електронний ресурс] – URL: https://www.sciencedirect.com/science/article/pii/S156625352300177X

7. What is Amazon Alexa, and what can it do? [Електронний ресурс] – URL: https://www.digitaltrends.com/home/what-is-amazons-alexa-andwhat-can-it-do/

8. Google Assistant is built to keep your information private, safe, and secure. [Електронний ресурс] – URL: https://safety.google/intl/en_us/assistant/

9. Набори даних: характеристики даних [Електронний ресурс] – URL: https://developers.google.com/machine-learning/crash-course/overfitting/data-characteristics?hl=uk

10. Діаграма послідовності (Sequence Diagrams) [Електроний ресурс] – URL: maxzosim.com/sequence-diagrams/

11. Draw.io — безкоштовний онлайн-інструмент для створення діаграм [Електроний ресурс] – URL: https://www.drawio.com/

12. Про Wolfram|Alpha Зробити знання світу обчислюваними [Електронний ресурс] - URL: https://www.wolframalpha.com/about

13. PyCharm Features [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/pycharm/features/

14. Модуль розпізнавання мовлення Python [Електронний ресурс] - URL: https://www.geeksforgeeks.org/python-speech-recognition-module/

15. Tkinter — Python interface to Tcl/Tk¶ [Електроний ресурс] – URL: https://docs.python.org/uk/3.13/library/tkinter.html

16. Python Tkinter is a standard GUI (Graphical User Interface) [Електроний ресурс] – URL: https://www.geeksforgeeks.org/python/python-gui-tkinter/

17. Text to Speech (TTS) library for Python [Електроний ресурс] – URL: https://pypi.org/project/pyttsx3/

18. Python(мовапрограмування) [Електронний ресурс] - URL: https://en.wikipedia.org/wiki/Python_(programming_language)

19. Репозиторій даних Wolfram — це публічний ресурс, який містить зростаючу колекцію обчислювальних наборів даних [Електронний ресурс] - URL : https://datarepository.wolframcloud.com/?source=footer

20. Library for performing speech recognition, with support for several engines [Елктронний ресурс] – URL: https://pypi.org/project/SpeechRecognition/

21. Тестування інтерфейсу користувача: детальний посібник [Електронний ресурс] - URL: https://www.browserstack.com/guide/ui-testing-guide

22. Функціональне тестування - Тестування програмного забезпечення [Електронний ресурс] - URL; https://www.geeksforgeeks.org/software-testing-functional-testing/

23. Гінзула В.Я. Голосовий помічник на основі машинного навчання. *Інтелектуальні інформаційні технології в прикладних дослідженнях (IITAR-2025)*: збірник тез студентської науково-практичної конференції, м. Тернопіль, 27-29 травня 2025 року. Тернопіль: ЗУНУ, 2025. С.142-143.

24. В. М. Островерхов, Л. І. Біловус, К. З. Возьний, О. О. Луцишин, Г. Л. Монастирський, С. А. Надвиничний, С. В. Питель,С. К. Шандрук., Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання

кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

25. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

Апробація отриманих результатів

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

Студентської науково-практичної конференції

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

(IITAR-2025)

27-29 травня 2025 року

Тернопіль

2025

Збірник тез доповідей студентської науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR-2025). (Тернопіль, 27-29 травня 2025 року). Тернопіль: ЗУНУ, 2025. 372 с.

До збірника увійшли тези доповідей учасників студентської науково-практичної конференції "Інтелектуальні інформаційні технології в прикладних дослідженнях" (IITAR – 2025), що відбувалась у рамках AI Week на базі кафедри інформаційно-обчислювальних систем і управління Західноукраїнського національного університету. Мета конференції — об'єднати науковців, освітян, представників IT-бізнесу та здобувачів освіти для обміну досвідом, презентації сучасних досліджень і впровадження інноваційних рішень у сфері комп'ютерних наук, штучного інтелекту та суміжних галузей.

За зміст наукових праць та достовірність наведених фактологічних і статистичних матеріалів відповідальність несуть автори публікацій та їхні наукові керівники. У збірнику зберігається стилістика та орфографія авторів матеріалів.

Склад організаційного комітету конференції

Гінзула Володимир
студент групи КН-43
volodaginzula4@gmail.com
Загородня Діана
к.т.н., доцент
dza@wunu.edu.ua
Західноукраїнський національний університет
Тернопіль, Україна

ГОЛОСОВИЙ ПОМІЧНИК НА ОСНОВІ МАШИННОГО НАВЧАННЯ

Останніми роками віртуальні помічники, що використовують штучний інтелект, стають дедалі популярнішими та все глибше проникають у наше повсякденне існування. Завдяки інтегрованим алгоритмам машинного навчання та обробки природної мови, ці технології здатні перетворювати звичайні справи на більш комфортні та продуктивні, забезпечуючи допомогу та спрощуючи виконання поставлених задач. Люди застосовують технологію голосових помічників у різних сферах життя, скажімо, для вирішення простих завдань, як-от перегляд прогнозу погоди, пошук необхідної інформації в інтернеті, робота з електронною поштою та інші.

Голосові помічники вже впроваджено у різноманітні гаджети. Це стосується смартфонів, лептопів, настільних комп'ютерів, ігрових приставок, телевізорів, інтелектуальних аудіосистем та автомобілів. Вони можуть використовувати голос як ключовий спосіб взаємодії, фактично знецінюючи або зменшуючи потребу у традиційному графічному інтерфейсі.

Голосовий помічник являє собою програмне забезпечення, що забезпечує розпізнавання мовлення користувача, інтерпретацію його запитів та відповідне формування реакції або виконання дій на основі інтегрованих алгоритмів. Такі системи належать до класу інтерактивних інтелектуальних технологій і знаходять широке застосування у побуті, освітньому процесі та виробничих процесах. Завдяки стрімкому розвитку технологій обробки природної мови (Natural Language Processing, NLP), автоматичного розпізнавання мовлення (Automatic Speech Recognition, ASR) та синтезу мовлення (Text-to-Speech, TTS), ефективність та зручність використання голосових інтерфейсів значно зросли [1].

Сучасні рішення в цій сфері використовують моделі машинного навчання, які тренуються на великих масивах даних. Це дозволяє їм адаптуватися до специфіки вимови, семантичного контексту запитів та інтонаційних особливостей користувачів. Серед ключових характеристик голосових помічників слід виділити їхню здатність інтегруватись із різноманітними програмами та сервісами, такими як календарі, поштові системи, погодні сервіси, органайзери файлів та інші додатки [2]. Наприклад, після команди "відкрий PowerPoint", система може виконати запит шляхом застосування відповідної

системної функції. Основна архітектура типового голосового помічника включає такі компоненти:

1. модуль розпізнавання мовлення;
2. обробник тексту (зокрема з використанням NLP);
3. модуль прийняття рішень або інтерпретації команд;
4. синтезатор мовлення;
5. інтерфейс доступу до зовнішніх ресурсів (API, файли, системні функції тощо).

Актуальним напрямом у сучасних технологіях є локалізація голосових асистентів для української мови, що передбачає не лише ефективне розпізнавання звуків, а й адаптацію мовної системи на рівні синтаксису, морфології, а також інтеграцію спеціалізованої термінології. У цьому контексті використання бібліотек із відкритим кодом, таких як SpeechRecognition, gTTS, pyttsx3 та transformers для задач обробки природної мови (NLP), сприяє створенню гнучких і налаштованих рішень, здатних відповідати вимогам конкретних застосувань.

Науковий і практичний аспект досліджуваної теми полягає у розвитку голосових інтерфейсів, які покращують природність взаємодії між людиною і комп'ютером. Це має особливе значення в умовах сучасної автоматизації процесів, забезпеченні інклюзивності та зростаючій потребі у безконтактному управлінні пристроями, що стає дедалі важливішим у різних сферах діяльності [4].

Голосові помічники є результатом поєднання передових технологій машинного навчання, штучного інтелекту та обробки природної мови. Вони відкривають нові можливості для автоматизації завдань, покращення взаємодії між людиною та комп'ютером, а також розвитку інтелектуальних систем. Зі збільшенням доступності технологій кожен розробник може створити власного голосового асистента, що відповідає його потребам, використовуючи наявні інструменти та методи програмування.

Список використаних джерел

1. Jurafsky D., Martin J.H. Speech and Language Processing. 3rd Edition. Draft. Stanford University, 2023.
2. Zhang Y., Zhao Y., Lei Y. Voice Assistant Development Based on Speech Recognition and Natural Language Understanding. Journal of Intelligent Systems, 2021.
3. Коваленко П.І. Технології штучного інтелекту у взаємодії людини з комп'ютером. Київ: Ліра-К, 2020.
4. Алгоритми та методи обробки природної мови / за ред. С. І. Литвиненка. Львів: ЛНУ, 2022.

Додаток Б

Код програми

Importss.py – використані бібліотеки

```
import wolframalpha
import wikipedia
import speech_recognition as sr
from ecapture import ecapture as ec
import subprocess
import wolframalpha
import pyttsx3
import tkinter
import json
import random
import operator
import datetime
import wikipedia
import webbrowser
import os
import winshell
import pyjokes
import feedparser
import smtplib
import ctypes
import time
import requests
import shutil
from clint.textui import progress
from ecapture import ecapture as ec
from bs4 import BeautifulSoup
import win32com.client as wincl
from urllib.request import urlopen
from simpleeval import simple_eval
import xml.etree.ElementTree as ET
import random
import ctypes
from transliterate import translit
```

maine_VA.py – Основник код програми

```
import tkinter as tk
from tkinter import scrolledtext
import threading
import queue
from Importss import *

class VoiceAssistantApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Анатолій 1.5 - Голосовий помічник")
        self.engine = pyttsx3.init()
        voices = self.engine.getProperty('voices')
        self.engine.setProperty('voice', voices [2].id)
        self.q = queue.Queue()
```

```
        self.voice_thread = threading.Thread(target=self.speak_thread,
daemon=True)
        self.voice_thread.start()
        # Налаштування голосового двигуна
        self.assname = "Анатолій 1.5"

        # Інтерфейс
        self.create_widgets()
        self.wish_me()

    def create_widgets(self):
        # Текстове поле для виводу
        self.output_text = scrolledtext.ScrolledText(self.root, width=70,
height=20, wrap=tk.WORD)
        self.output_text.pack(pady=10)
        self.output_text.insert(tk.END, "Anatoliy 1.5: Hello Я ваш голосовий
помічник.\n")

        # Поле для текстового введення
        self.input_frame = tk.Frame(self.root)
        self.input_frame.pack(pady=5)
        self.input_label = tk.Label(self.input_frame, text="Ваш запит:")
        self.input_label.pack(side=tk.LEFT)
        self.input_entry = tk.Entry(self.input_frame, width=50)
        self.input_entry.pack(side=tk.LEFT, padx=5)
        self.input_entry.bind("<Return>", self.process_text_command)

        # Кнопки
        self.button_frame = tk.Frame(self.root)
        self.button_frame.pack(pady=5)
        self.voice_button = tk.Button(self.button_frame, text="Голосовий ввід",
command=self.start_voice_input)
        self.voice_button.pack(side=tk.LEFT, padx=5)
        self.text_button = tk.Button(self.button_frame, text="Текстове
відправлення", command=self.process_text_command)
        self.text_button.pack(side=tk.LEFT, padx=5)
        self.clear_button = tk.Button(self.button_frame, text="Очистити",
command=self.clear_output)
        self.clear_button.pack(side=tk.LEFT, padx=5)

    def speak_thread(self):
        while True:
            text = self.q.get()
            if text is None:
                break
            self.engine.say(text)
            self.engine.runAndWait()

    def speak(self, audio):
        self.output_text.insert(tk.END, f"Anatoliy 1.5: {audio}\n")
        self.output_text.see(tk.END)
        self.q.put(audio)

    def wish_me(self):
        hour = datetime.datetime.now().hour
        if 0 <= hour < 12:
            greeting = "Good Morning!"
        elif 12 <= hour < 18:
```

```
            greeting = "Доброго дня!"
        else:
            greeting = "Доброго вечора!"

        self.speak(f"{greeting} Я ваш помічник {self.assname}.")
        self.username()  # Додаємо виклик функції після привітання

    def username(self):
        self.speak("Як звертатись до вас?")

        # Запускаємо голосовий ввід у окремому потоці
        threading.Thread(target=self.get_username_voice).start()

    def get_username_voice(self):
        try:
            r = sr.Recognizer()
            with sr.Microphone() as source:
                self.output_text.insert(tk.END, "Слухаю...\n")
                r.pause_threshold = 1
                audio = r.listen(source)

            uname = r.recognize_google(audio, language='uk-UA')
            self.process_username(uname)
        except Exception as e:
            self.speak("Не вдалося розпізнати ім'я")
            # Пропонуємо ввести текстовий варіант
            self.output_text.insert(tk.END, "Будь ласка, введіть ваше ім'я в
текстовому полі та натисніть Enter\n")

    def process_username(self, uname):
        self.user_name = uname  # Зберігаємо ім'я
        self.speak(f"Вітаю, {uname}!")
        self.output_text.insert(tk.END, f"Користувач: {uname}\n")
        self.output_text.insert(tk.END, "######################\n")
        self.output_text.insert(tk.END, f"Вітаємо, {uname}!\n")
        self.output_text.insert(tk.END, "######################\n")
        self.speak("Як я можу вам допомогти?")

    def start_voice_input(self):
        threading.Thread(target=self.take_command).start()

    def take_infinity_command(self):
        with True:
            r = sr.Recognizer()
            with sr.Microphone() as source:
                self.output_text.insert(tk.END, "Слухаю...\n")
                self.output_text.see(tk.END)
                r.pause_threshold = 1
                audio = r.listen(source)

            try:
                self.output_text.insert(tk.END, "Розпізнавання...\n")
                self.output_text.see(tk.END)
                query = r.recognize_google(audio, language='en-in')
                self.output_text.insert(tk.END, f"Ви сказали: {query}\n")
                self.output_text.see(tk.END)
                return query
            except Exception as e:
```

```
                self.speak("Не вдалося розпізнати ваш голос")
                self.output_text.insert(tk.END, f"Помилка: {str(e)}\n")
                self.output_text.see(tk.END)

    def take_command(self):
        r = sr.Recognizer()
        with sr.Microphone() as source:
            self.output_text.insert(tk.END, "Слухаю...\n")
            self.output_text.see(tk.END)
            r.pause_threshold = 1
            audio = r.listen(source)

        try:
            self.output_text.insert(tk.END, "Розпізнавання...\n")
            self.output_text.see(tk.END)
            query = r.recognize_google(audio, language='uk-UA')
            self.output_text.insert(tk.END, f"Ви сказали: {query}\n")
            self.output_text.see(tk.END)
            self.process_query(query.lower())
        except Exception as e:
            self.speak("Не вдалося розпізнати ваш голос")
            self.output_text.insert(tk.END, f"Помилка: {str(e)}\n")
            self.output_text.see(tk.END)

    def process_text_command(self, event=None):
        query = self.input_entry.get()
        if query:
            self.output_text.insert(tk.END, f"Ви: {query}\n")
            self.output_text.see(tk.END)
            self.input_entry.delete(0, tk.END)
            self.process_query(query.lower())

    def process_query(self, query):
        """Функція обробки команд"""
        if 'вікіпедія' in query:
            self.search_wikipedia(query)
        elif 'відкрий youtube' in query:
            self.open_website("youtube.com")
        elif 'відкрий google' in query:
            self.open_website("google.com")
        elif 'пошук' in query:
            self.search_web(query.replace("пошук", ""))
        elif 'час' in query:
            self.show_time()
        elif 'новини' in query:
            self.get_news()

        elif 'вихід' in query or 'закрий' in query:
            self.root.destroy()

        elif any(word in query.lower() for word in  ['stackoverflow', 'stack
overflow', 'стек оверфлоу']):
            self.open_stackoverflow()

        elif any(cmd in query.lower() for cmd in  ['play music', 'play song',
'музика', 'включи музику']):
            self.play_music()
```

```
        elif any(word in query.lower() for word in  ['time', 'час', 'котра
година']):
            self.show_time()

        elif any(word in query.lower() for word in  ['how are you', 'як справи',
'як себе почуваєш']):
            self.handle_mood(query)

        elif any(word in query.lower() for word in  ['fine', 'good', 'чудово',
'добре']):
            self.handle_mood(query)

        elif any(word in query.lower() for word in  ['exit', 'quit', 'вихід',
'закрий']):
            self.exit_program()

        elif any(phrase in query.lower() for phrase in  ["who made you", "who
created you", "хто тебе створив",
                                                        "хто твій автор"]):
            self.show_creator()

        elif any(word in query.lower() for word in  ['joke', 'жарт', 'розвесели',
'розкажи жарт']):
            self.tell_joke()

        elif any(word in query.lower() for word in  ['calculate', 'калькулятор',
'порахувати']):
            self.open_calculator()

        elif any(phrase in query.lower() for phrase in
                ['power point', 'powerpoint', 'презентація', 'відкрий powerpoint',
'павер поінт']):
            self.open_powerpoint()

        elif any(phrase in query.lower() for phrase in  ['word', 'Word', 'ворд',
'відкрий ворд']):
            self.open_word()

        elif any(phrase in query.lower() for phrase in  ['excel', 'Excel',
'ексель', 'відкрий ексель']):
            self.open_excel()

        elif any(phrase in query.lower() for phrase in  ["who are you", "хто ти",
"що ти"]):
            self.introduce()

        elif any(phrase in query.lower() for phrase in  ['lock', 'блок', 'заблокуй
екран', 'lock window']):
            self.lock_computer()

        elif any(phrase in query.lower() for phrase in  ['shutdown', 'вимкни',
'вимкни комп"ютер', 'shutdown system']):
            self.shutdown_system()

        elif any(word in query.lower() for word in  ['погода', 'weather', 'яка
погода']):
            self.get_weather()
```

```
        elif any(phrase in query.lower() for phrase in  [
            'good morning', 'good afternoon', 'good evening',
            'добрий ранок', 'добрий день', 'добрий вечір']):
            self.respond_to_greeting(query)

        elif any(phrase in query.lower() for phrase in  ['what is', 'who is']):
            self.answer_question(query)
        else:
            self.speak("Я не розумію вашого запиту. Спробуйте ще раз.")


    def answer_question(self, query):
        """Answer questions in English only using WolframAlpha"""
        try:
            # Verify the query contains only English characters
            if not all(ord(char) < 128 for char in query):
                self.speak("Please ask in English")
                self.output_text.insert(tk.END, "Error: Only English questions
supported\n")
                return

            client = wolframalpha.Client("[redacted]")
            res = client.query(query)
            result = next(res.results).text

            self.speak(result)
            self.output_text.insert(tk.END, f"Answer: {result}\n")

        except StopIteration:
            self.speak("No results found")
            self.output_text.insert(tk.END, "No results found\n")
        except Exception as e:
            self.speak("Search error")
            self.output_text.insert(tk.END, f"Error: {str(e)}\n")



    def get_weather(self):
        """Відкриває сайт sinoptik.ua з невеликою затримкою"""
        self.output_text.insert(tk.END, "🕶 Відкриваю сайт sinoptik.ua...\n")
        self.output_text.see(tk.END)
        self.speak("Відкриваю сайт sinoptik точка юа")

        time.sleep(0.5)
        # Відкриття сайту
        url = "https://sinoptik.ua/pohoda"
        webbrowser.open(url)



    def shutdown_system(self):
        """Вимкнення комп'ютера"""
        try:
            self.speak("Зачекайте, система вимикається")
            self.output_text.insert(tk.END, "Ініціюється вимкнення системи...\n")
            self.output_text.see(tk.END)

            # Додаємо підтвердження
```

```
            confirm = tk.messagebox.askyesno(
                "Підтвердження",
                "Ви впевнені, що хочете вимкнути комп'ютер?",
                parent=self.root
            )

            if confirm:
                # Безпечне вимкнення через 1 хвилину
                subprocess.call( ["shutdown", "/s", "/t", "60"])
                self.output_text.insert(tk.END, "Комп'ютер вимкнеться через 1 
хвилину\n")
                self.speak("Комп'ютер вимкнеться через 1 хвилину. Збережіть усі 
дані.")
            else:
                self.speak("Вимкнення скасовано")
                self.output_text.insert(tk.END, "Вимкнення скасовано 
користувачем\n")

        except Exception as e:
            self.speak("Помилка при вимкненні")
            self.output_text.insert(tk.END, f"Помилка: {str(e)}\n")

    def lock_computer(self):
        """Заблокувати комп'ютер"""
        try:
            self.speak("Блокування комп'ютера")
            self.output_text.insert(tk.END, "Комп'ютер буде заблоковано...\n")
            self.root.after(1000, lambda: ctypes.windll.user32.LockWorkStation())
        except Exception as e:
            self.speak("Не вдалося заблокувати комп'ютер")
            self.output_text.insert(tk.END, f"Помилка: {str(e)}\n")

    def open_word(self):
        """Відкриття Microsoft Word """
        msg = "📄 Відкриваю Word…"
        self.output_text.insert(tk.END, msg + "\n")
        self.output_text.see(tk.END)

        # Якщо є метод озвучування, не завадить повідомити голосом
        try:
            self.speak("Відкриваю ворд")
        except Exception:
            pass
        self.root.after(500, lambda: os.system('start WINWORD'))

        # os.system('start WINWORD')

    def open_powerpoint(self):
        msg = "📽 Відкриваю PowerPoint…"
        self.output_text.insert(tk.END, msg + "\n")
        self.output_text.see(tk.END)

        # Якщо є метод озвучування, не завадить повідомити голосом
        try:
            self.speak("Відкриваю PowerPoint")
        except Exception:
            pass
        self.root.after(500, lambda: os.system('start POWERPNT'))
```

```
    #os.system('start POWERPNT')
def open_excel(self):

    msg = "📊 Відкриваю Excel…"
    self.output_text.insert(tk.END, msg + "\n")
    self.output_text.see(tk.END)
    try:
        self.speak("Відкриваю Excel")
    except Exception:
        pass
    self.root.after(500, lambda: os.system(' start EXCEL'))

def open_calculator(self):
    """Відкриває системний калькулятор"""
    try:
        # Виведення повідомлення в інтерфейс
        msg = "🧮 Відкриваю калькулятор..."
        self.output_text.insert(tk.END, msg + "\n")
        self.output_text.see(tk.END)

        # Голосове підтвердження
        self.speak("Відкриваю калькулятор")

        # Відкладений запуск калькулятора (через 500 мс)
        self.root.after(500, lambda: os.system('calc'))

    except Exception as e:
        error_msg = f"Не вдалося відкрити калькулятор: {str(e)}"
        self.output_text.insert(tk.END, error_msg + "\n")
        self.speak("Помилка при відкритті калькулятора")


def tell_joke(self):
    """Розповісти жарт"""
    try:
        joke = pyjokes.get_joke(language='en')
        self.speak(joke)
        self.output_text.insert(tk.END, f"Жарт: {joke}\n")
        self.output_text.see(tk.END)
    except Exception as e:
        self.speak("Не вдалося знайти жарт")
        self.output_text.insert(tk.END, f"Помилка: {str(e)}\n")

def show_creator(self):
    """Відповідь про творця помічника """
    creator = "Гаурав "  # Можете змінити на своє ім'я
    response = f"Мене створив {creator} а до опрацював Гінзула Володимир"
    self.speak(response)
    self.output_text.insert(tk.END, f"{response}\n")
    self.output_text.see(tk.END)

def change_name(self):
    """Змінити ім'я помічника"""
    self.speak("Як ви хочете мене називати?")
    # Для голосового вводу
    threading.Thread(target=self._process_name_change).start()
```

```
    def _process_name_change(self):
        try:
            r = sr.Recognizer()
            with sr.Microphone() as source:
                audio = r.listen(source)
            new_name = r.recognize_google(audio, language='en')
            self.assname = new_name
            self.speak(f"Дякую! Тепер я {new_name}")
            self.output_text.insert(tk.END, f"Нове ім'я помічника: {new_name}\n")
        except Exception as e:
            self.speak("Не зрозуміла нове ім'я")
            self.output_text.insert(tk.END, "Введіть нове ім'я в текстове поле\n")

    def handle_mood(self, query):
        """функція перевірк настрою"""
        if 'how are you' in query.lower ():
            self.speak("Я в порядку, дякую!")
            self.speak("А як ваші справи?")
        elif any(word in query.lower() for word in  ['fine', 'good', 'чудово']):
            self.speak("Радий чути, що у вас все добре!")

    def show_time(self):
        """Функція виведення часу"""
        try:
            current_time = datetime.datetime.now().strftime("%H:%M:%S")
            self.speak(f"Зараз {current_time}")
            self.output_text.insert(tk.END, f"Поточний час: {current_time}\n")
            self.output_text.see(tk.END)
        except Exception as e:
            self.speak("Не вдалося отримати час")
            self.output_text.insert(tk.END, f"Помилка: {str(e)}\n")

    def play_music(self):
        """Функція вмикання музики"""
        try:
            music_dir = "C:\\Users\\WellDone\\Desktop\\Документи\\В дорогу"
            songs =  [f for f in os.listdir(music_dir) if f.endswith(('.mp3',
'.wav', '.ogg'))]

            if not songs:
                self.speak("У папці немає музичних файлів")
                return

            random_song = random.choice(songs)
            self.speak(f"Граю {os.path.splitext(random_song) [0]}")
            os.startfile(os.path.join(music_dir, random_song))

        except Exception as e:
            self.speak("Помилка відтворення музики")
            self.output_text.insert(tk.END, f"Помилка: {str(e)}\n")

    def open_stackoverflow(self):
        """Відкрити стак оверфлов"""
        self.speak("Відкриваю Stack Overflow. Щасливого кодування!")
        webbrowser.open("https://stackoverflow.com")

    def search_wikipedia(self, query):
        """Пошук у вікіпедії"""
```

```
        try:
            query = query.replace("вікіпедія", "")
            results = wikipedia.summary(query, sentences=3)
            self.speak("За результатами Вікіпедії:")
            self.output_text.insert(tk.END, f"Результат: {results}\n")
            self.output_text.see(tk.END)
        except Exception as e:
            self.speak("Не вдалося знайти інформацію у Вікіпедії")
            self.output_text.insert(tk.END, f"Помилка: {str(e)}\n")
            self.output_text.see(tk.END)

    def open_website(self, url):
        webbrowser.open(url)
        self.speak(f"Відкриваю {url}")

    def search_web(self, query):
        webbrowser.open(f"https://www.google.com/search?q={query}")
        self.speak(f"Шукаю {query} в Google")

    def get_news(self):
        """Отримання новин"""
        try:
            url = "https://www.pravda.com.ua/rss/"
            response = requests.get(url)
            root = ET.fromstring(response.content)

            items = root.findall('.//item')
            self.speak("Останні новини:")
            for i, item in enumerate(items [:5], start=1):
                title = item.find('title').text
                self.output_text.insert(tk.END, f"{i}. {title}\n")
                self.output_text.see(tk.END)
        except Exception as e:
            self.speak("Не вдалося отримати новини")
            self.output_text.insert(tk.END, f"Помилка: {str(e)}\n")
            self.output_text.see(tk.END)

    def exit_program(self):
        """Завершення роботи програми"""
        self.speak("Дякую за ваш час. До побачення!")
        self.output_text.insert(tk.END, "Програма завершує роботу...\n")
        self.root.after(2000, self.root.destroy)

    def introduce(self):
        """Представлення помічника """
        response = f"Я ваш віртуальний помічник {self.assname}. Мене створив
Гаурав І Володя "
        self.speak(response)
        self.output_text.insert(tk.END, f"{response}\n")
        self.output_text.see(tk.END)

    def respond_to_greeting(self, query):
        """Обробка привітань """
        greeting_map = {
            'good morning': ('Доброго ранку', 'Як ваші справи?'),
            'good afternoon': ('Доброго дня', 'Як ваші справи?'),
            'good evening': ('Доброго вечора', 'Як ваші справи?'),
            'добрий ранок': ('Доброго ранку', 'Як ваші справи?'),
```

```
            'добрий день': ('Доброго дня', 'Як ваші справи?'),
            'добрий вечір': ('Доброго вечора', 'Як ваші справи?')
        }

        # Знаходимо відповідне привітання
        for phrase, response in greeting_map.items():
            if phrase in query.lower():
                self.speak(f"{response [0]}, {self.user_name if hasattr(self,
'user_name') else ''}")
                self.speak(response [1])
                self.output_text.insert(tk.END, f"{response [0]}! {response
[1]}\n")
                self.output_text.see(tk.END)
                return

        # Якщо привітання не розпізнано
        self.speak("Вітаю!")
        self.output_text.insert(tk.END, "Вітаю!\n")

    def clear_output(self):
        self.output_text.delete(1.0, tk.END)



if __name__ == "__main__":
    root = tk.Tk()
    app = VoiceAssistantApp(root)
    root.mainloop()
```