МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»_____,

освітньо-професійної програми «Інформаційні технології проектування»_____

на тему: Віртуальний голосовий помічник на основі штучного інтелекту

Здобувача групи ___ІТ.м-22__ __Радченко Олег Сергійович_____
(шифр групи) (прізвище, ім'я, по батькові)

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

_____ ___Олег РАДЧЕНКО_____
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник _____к.т.н., доц. Яна ЧИБІРЯК_____ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІТ

_____
«_____» _____ 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу магістра студентові

Радченко Олег Сергійович
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи Віртуальний голосовий помічник на основі штучного інтелекту

затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI

2 Термін здачі студентом кваліфікаційної роботи «___» ___грудня___ 2023 р.

3 Вхідні дані до кваліфікаційної роботи голосова команда користувача

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі та методи дослідження, проектування інформаційної системи, практична розробка рішення

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) актуальність, постановка задачі, аналіз аналогічних голосових помічників, таблиця порівняння аналогів, функціональні вимоги до голосового помічника, контекстна діаграма в нотації IDEF0, діаграма декомпозиції першого рівня, декомпозиція процесу «Виконання Команди», діаграма сценаріїв використання, високорівневий дизайн, практична реалізація, демонстрація роботи голосового помічника, тестування за методом білого ящика, висновок

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
	Підготовка специфікацій	20.09.2023	
	Розробка голосового помічника	20.11.2023	
	Тестування	23.11.2023	
	Впровадження в дію	01.12.2023	

Магістрант _____ Олег РАДЧЕНКО

Керівник роботи _____ к.т.н., доц. Яна ЧИБІРЯК

АНОТАЦІЯ

Тема кваліфікаційної роботи магістра «Віртуальний голосовий помічник на основі штучного інтелекту».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 57 найменувань, 3 додатків. Загальний обсяг роботи – 104 сторінок, у тому числі 48 сторінок основного тексту, 5 сторінок списку використаних джерел, 41 сторінка додатків.

Кваліфікаційну роботу магістра присвячено розробці Віртуального голосового помічника на основі штучного інтелекту. В роботі проведено аналіз предметної області, визначено потребу в розробці голосового помічника, обрано технології для реалізації голосового помічника, поставлено мету розробки, визначено перелік задач та описано методи дослідження. У роботі виконано проектування системи на різних рівнях та описано базові модулі голосового помічника. Здійснено програмну реалізацію спроектованого голосового помічника на основі штучного інтелекту з додаванням води до його функціоналу.

Результатом проведеної роботи є повноцінно розроблений голосовий помічник на основі штучного інтелекту та включає в себе додаткові можливості. Розроблений голосовий помічник володіє розширеним функціоналом, що включає в себе не лише базові завдання, але й додаткові функції для поліпшення користувальницького досвіду. Практичне значення роботи виявляється у вдосконаленні взаємодії користувача з комп'ютером. Голосовий помічник на основі штучного інтелекту, не лише виконує рутинні завдання, але й покликаний спрощувати роботу з технічними засобами шляхом надання додаткових можливостей та функціоналу. Цей голосовий помічник прагне полегшити щоденне використання комп'ютера, забезпечуючи користувача швидким та ефективним виконанням завдань, а також забезпечує часткову автоматизацію процесів, що сприяє більш зручному взаємодії з технічним обладнанням.

Ключові слова: голосовий, асистент, віртуальний, штучний інтелект, помічник, python.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI – Штучний інтелект
IoT – Інтернет речей
HLD – Дизайн високого рівня
Git – (GitHub) Система контролю версій
HTML – Мова гіпертекстової розмітки
CSS – Каскадна таблиця стилів
JS – (JavaScript) Мультипарадигмова мова програмування
JSON – Текстовий формат обміну даними
IT – Інформаційні технології
IC – інформаційна система(ми)

ВСТУП

Актуальність. У сучасному світі інформаційних технологій (ІТ), впровадження нових технологічних рішень вирішує актуальні завдання оптимізації та поліпшення повсякденних робочих процесів. Віртуальні голосові помічники, що базуються на штучному інтелекті, стають цілком важливою складовою для забезпечення ефективності та зручності користування інформаційними ресурсами. З урахуванням стрімкого розвитку цієї технології, виникає необхідність у вивченні та впровадженні віртуальних голосових помічників для оптимізації взаємодії користувачів із цифровим середовищем.

Тема. Віртуальний голосовий помічник на основі штучного інтелекту.

Мета. Створення віртуального голосового помічника на основі штучного інтелекту за рахунок належної організації даного процесу.

Для досягнення мети, необхідно виконати такі задачі:

- виконати аналіз предметної області,
- провести аналіз програм-аналогів,
- обрати технологію та програмні засоби реалізації
- виконати моделювання функціональних можливостей,
- виконати програмну реалізацію голосового помічника
- провести тестування.

Об’єкт дослідження. Використання технологій штучного інтелекту під час створення віртуальних голосових помічників.

Предмет дослідження. Процес удосконалення функціональних можливостей віртуального голосового помічника.

Практична новизна. Використання віртуального голосового помічника спрощує рутинні завдання та підвищує продуктивність за рахунок інтелектуальної обробки голосових команд. Це призводить до збільшення ефективності комунікації та сприяє оптимізації робочих процесів. Введення такого інструмента може революціонізувати спосіб взаємодії з технологічним середовищем, забезпечуючи

швидкий та зручний доступ до інформації, а також сприяючи зростанню продуктивності працівників в різних галузях діяльності.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень та публікацій

З року появи AI, розробники зрозуміли, що використання AI в свої проектах може допомогти як привернути увагу до своїх проектів, так і підвищити якість цього продукту.

На сьогоднішній день важко уявити хоча б місяць без новин про штучний інтелект, що говорить про розповсюдження їх по інфополю користувачів як прогресивний і дуже амбіціозний напрям розвитку IT сфери. Мільйони користувачів вже оцінили його могутність використовуючи ChatGPT [1] або менш розповсюджений Bard від Google [2], хоча вони і мають схожий функціонал, але на мою думку, Bard стане значно потужнішим інструментом, так як Google буде інтегрувати його в свої продукти, як приклад, Google Assistant, це описано в [3].

У світі стрімкого розвитку технологій штучного інтелекту віртуальні голосові помічники стають не лише інструментом, але й невід'ємною частиною нашого щоденного життя. Їхні можливості постійно розширюються, надаючи користувачам нові способи взаємодії з технікою та оточуючим світом.

Цей експоненційний ріст викликає великий інтерес до дослідження можливостей оптимізації голосових інтерфейсів. Дослідники зосереджуються на покращенні розпізнавання мови, зменшенні часу реакції та створенні більш інтуїтивних та ефективних взаємодій. [4-7]

Однією з ключових тенденцій є інтеграція віртуальних голосових помічників у віртуальну та розширену реальність. Це відкриває перед нами нові горизонти використання - від ігор [8] і навчання до ефективного керування побутовою технікою.

Забезпечення безпеки та конфіденційності користувачів - ще один важливий аспект досліджень. Розробка передових методів шифрування та захисту даних стає критичною для підтримання довіри до цих інноваційних технологій.

Окрім технічних питань, вивчення впливу віртуальних голосових помічників на психологію користувачів стає актуальним. Важливо розуміти, як ці технології взаємодіють з емоціями та психічним станом людини.

Іншим напрямком досліджень є розробка глобальних систем мовного взаєморозуміння. Це вирішує завдання створення універсальних голосових інтерфейсів, які можуть працювати з різними мовами та діалектами.

Важливим аспектом стає також вивчення етичних питань використання віртуальних помічників [9].

Останні дослідження також присвячені взаємодії віртуальних голосових помічників з іншими передовими технологіями, такими як IoT [10] та блокчейн. Це може відкривати нові можливості для створення інтегрованих та ефективних систем.

Усі ці аспекти досліджень формують динамічний ландшафт в галузі віртуальних голосових помічників, де інновації та вдосконалення визначають майбутнє цієї галузі.

1.2 Аналіз інформаційних систем

На фоні стрімкого розвитку штучного інтелекту, голосові помічники стають невід'ємною частиною нашого цифрового повсякдення. Різноманіття голосових асистентів від різних технологічних гігантів пропонує користувачам широкий спектр можливостей та функціональності.

Amazon Alexa, один з найпопулярніших голосових асистентів, перевершує звичайні очікування користувачів. Цей інтелектуальний асистент від Amazon вбудований в широкий спектр пристроїв, що робить його невід'ємною частиною домашнього та побутового середовища.

Завдяки високому рівню інтеграції, Amazon Alexa дозволяє користувачам не лише задавати питання та виконувати команди голосом, але й керувати різними "розумними" пристроями, що зображено на рис 1.1. Це може бути відтворення музики на колонках, регулювання освітлення, контроль температури, а навіть запуск автомобіля - усе це можливо завдяки Alexa [11].

Присутність Amazon Alexa в автомобілях також стає неоціненною. Від зміни маршруту до відтворення улюбленої музики, голосовий асистент допомагає зробити подорож більш комфортною та безпечною. Таким чином, Amazon Alexa виходить за рамки простої відповіді на питання та стає важливим партнером у щоденних аспектах нашого життя.

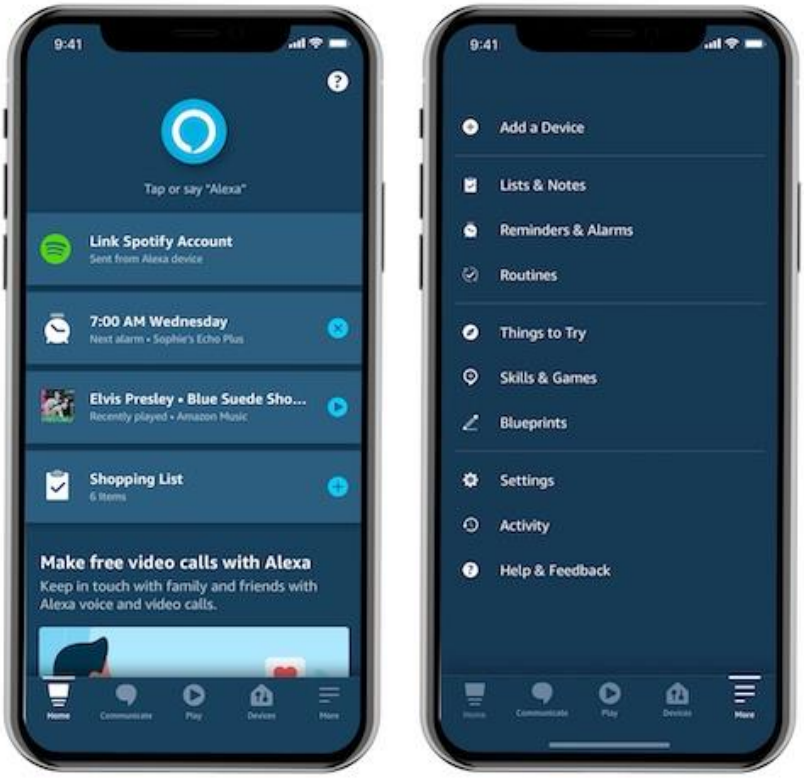

Рисунок 1.1 – Інтерфейс голосового помічника Alexa. *Джерело:* [12]

Google Assistant вирізняється своєю унікальною спроможністю аналізу та інтеграції, роблячи його важливим інструментом для різних аспектів нашого цифрового життя (зображення). Акцент на аналітичних можливостях дозволяє цьому голосовому асистенту виходити за межі звичайних завдань та надавати користувачам високоякісні відповіді на різноманітні запитання.

Інтеграція Google Assistant з іншими сервісами Google робить його важливим інструментом для розваг та підвищення продуктивності. Завдяки цьому асистент може легко доступатися до календаря, заміток, списків завдань, а також інших персональних даних користувача, що зображено на рисунку 1.2. Використання

розумного аналізу дозволяє Google Assistant зрозуміти контекст запитань, що робить взаємодію з ним більш натуральною та ефективною, це зображено на рисунку 1.3.

Окрім того, Google Assistant може слугувати джерелом розваг. Від відтворення музики до запуску відомих графічних ігор, він може забезпечити різноманіття відпочинку для користувачів. Такий широкий функціонал робить Google Assistant не тільки помічником у щоденних справах, але й невід'ємною частиною цифрового стилю життя [13].

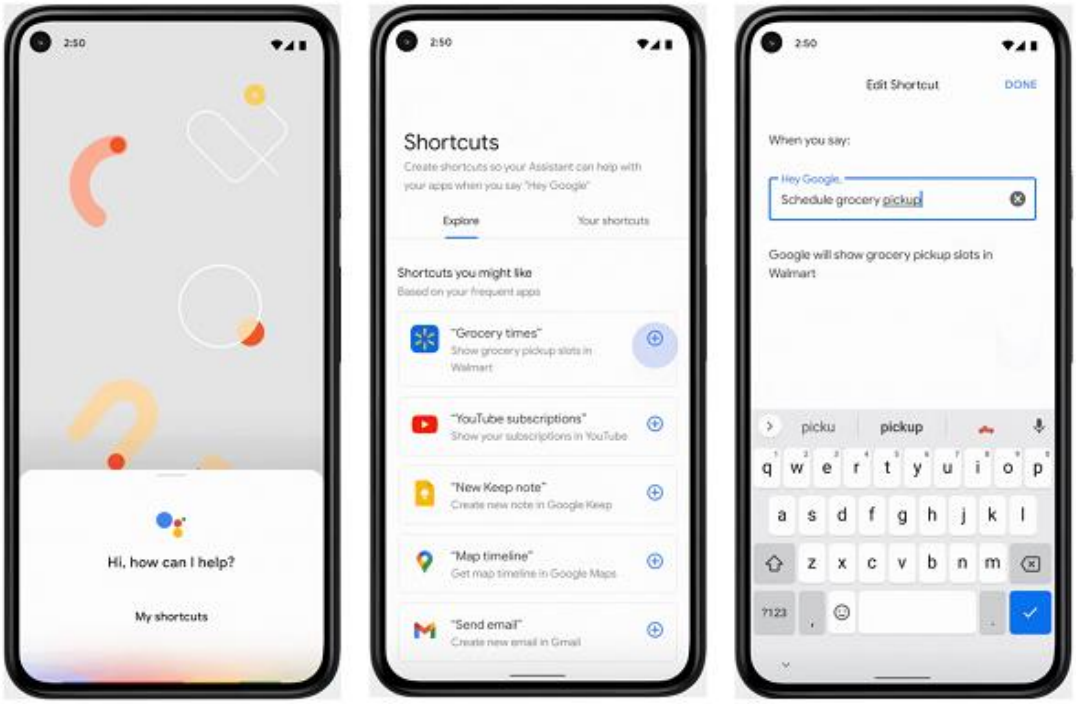

Рисунок 1.2 – Інтерфейс Google Assistant. *Джерело:* [14]

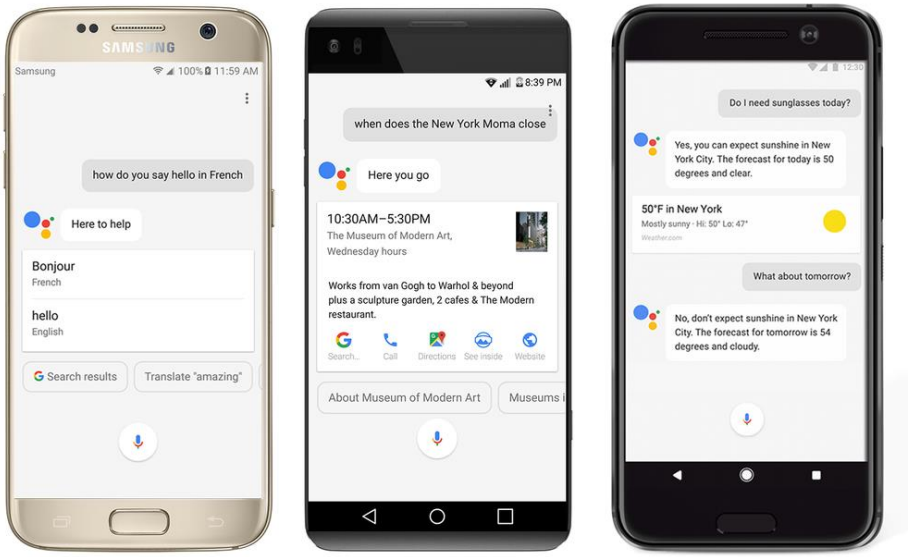

Рисунок 1.3 – Взаємодія користувача з голосовим асистентом. *Джерело:* [15]

Apple Siri визначає новий стандарт голосових асистентів, поєднуючи у собі високу інтеграцію та бездоганну зручність в користуванні всередині екосистеми бренду (зображення). Цей інтелектуальний асистент не просто виконує функції, але стає невід'ємною частиною життя користувачів Apple.

Специфічні функції Siri, такі як відправлення повідомлень, встановлення нагадувань, і контроль за побутовою технікою, встановлюють новий стандарт зручності в щоденному використанні гаджетів Apple. Інтеграція Siri з пристроями бренду, такими як iPhone, iPad, Apple Watch та Mac, робить його не просто функціональним асистентом, але і ключовим елементом гармонійної взаємодії між пристроями [16].

Легкість взаємодії з різними пристроями та застосунками робить Siri надійним компаньйоном у повсякденному житті. Чи йдете ви на роботу, займаєтеся спортом чи відпочиваєте вдома - Siri завжди готовий виконати ваші команди. Цей голосовий асистент стає символом зручності, інтеграції та відмінності в світі технологій, які визначають бренд Apple.

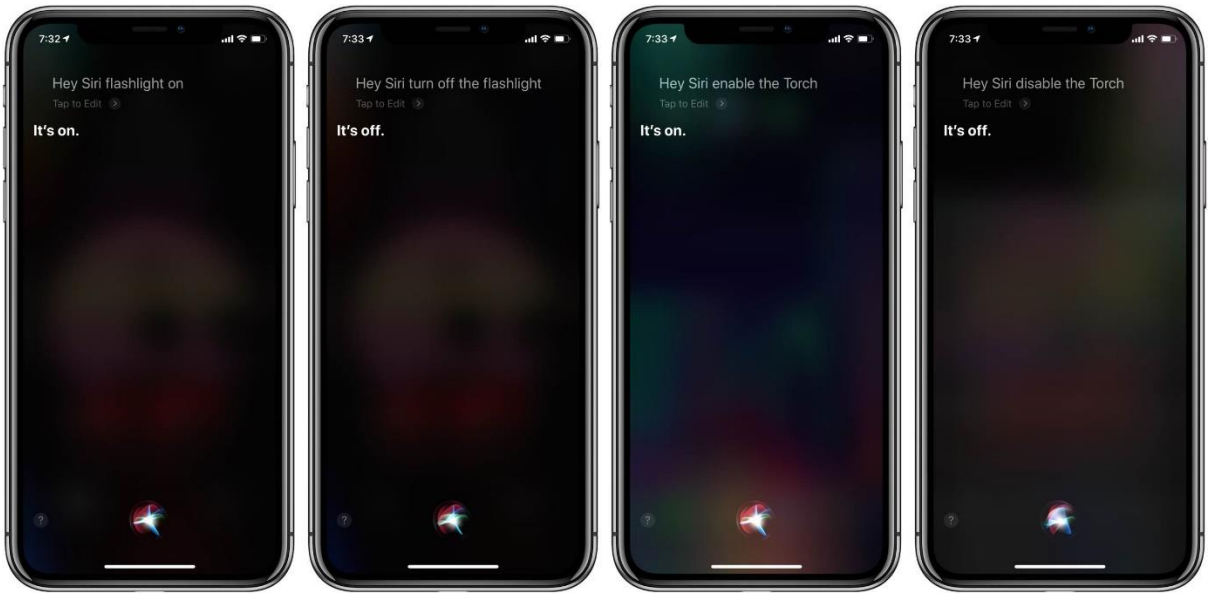

Рисунок 1.4 – Голосовий помічник Siri. *Джерело:* [17]

Microsoft Cortana визначається своєю винятковою інтеграцією з продуктами компанії, ставши важливим інструментом для ефективної роботи та організації завдань, на рисунку 1.5 зображено приклад використання. Цей голосовий асистент

розроблений так, щоб бути невід'ємною частиною повсякденного професійного життя.

Однією з ключових особливостей Cortana є його висока інтеграція з продуктами Microsoft, такими як Office та Windows. Асистент може виконувати широкий спектр завдань, починаючи від відправлення електронної пошти і завершуючи встановленням зустрічей. Завдяки цій інтеграції, Cortana стає не просто функціональним інструментом, але і невід'ємним компаньйоном для користувачів, які ведуть багатозадачний стиль життя [18].

Для професіоналів Cortana стає відмінним помічником у щоденній роботі, надаючи можливість швидко виконувати завдання та керувати робочими процесами лише за допомогою голосу. Цей асистент забезпечує не лише функціональність, але й зручність, допомагаючи кожному користувачеві зосередитися на своїх професійних завданнях.

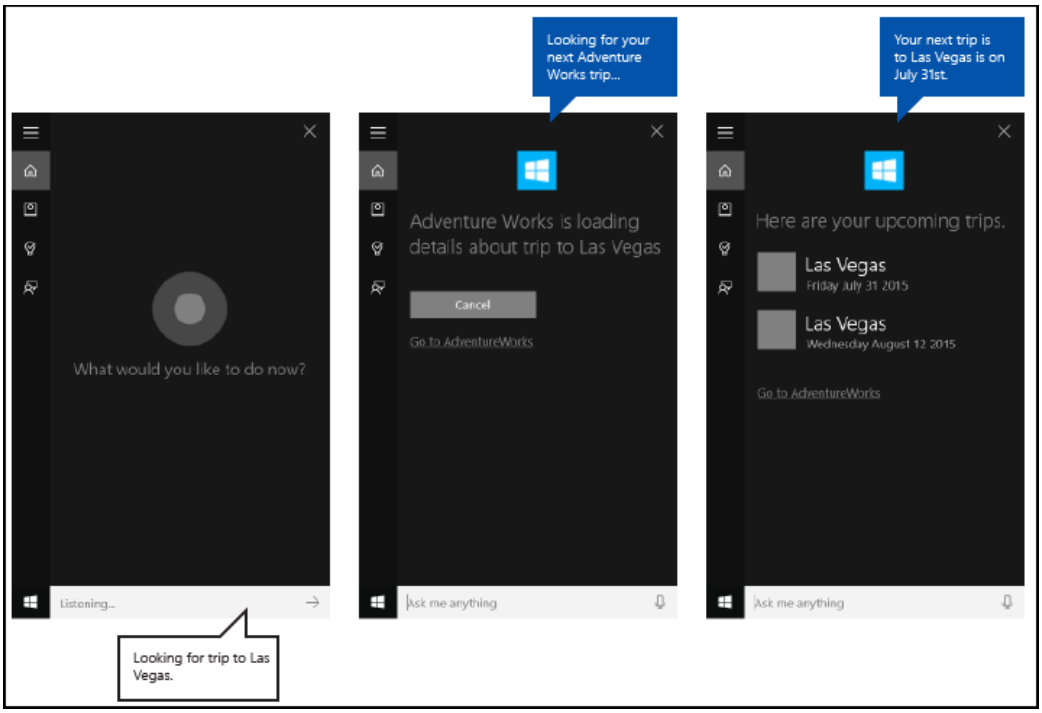

Рисунок 1.5 – Голосовий асистент Cortana, приклад використання.
Джерело: [19]

Samsung Bixby вирізняється своєю унікальною спеціалізацією на використанні камери та функціях розпізнавання зображень, що надає користувачам широкий спектр новаторських можливостей. Цей голосовий асистент від Samsung стає

важливим інструментом для тих, хто активно використовує камеру свого пристрою та цінує розширені функції розпізнавання.

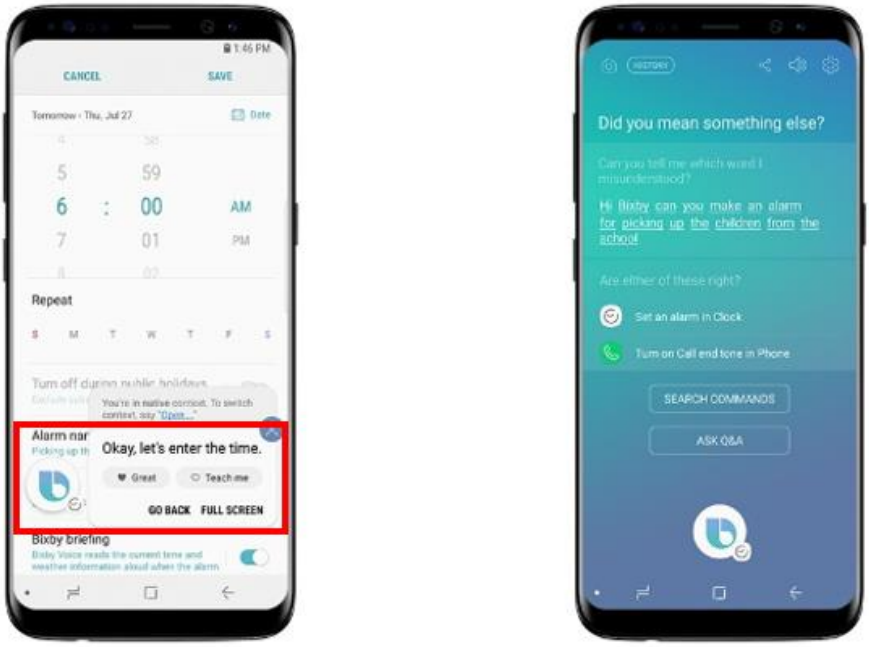

Рисунок 1.6 – Інтерфейс голосового помічника Samsung Bixby, та приклад використання. *Джерело:* [20]

Однією з ключових особливостей Bixby є можливість користувачів здійснювати пошук за зображеннями, визначати об'єкти та використовувати ряд інших додаткових функцій, що робить його незамінним помічником для фотографів та тих, хто бажає отримати більше інформації про світ навколо. Завдяки інноваційним функціям, Bixby перетворює використання камери на власний цифровий досвід.

Не лише голосові команди, але й можливості розпізнавання зображень роблять Bixby не просто асистентом, але і креативним інструментом для тих, хто прагне більше від свого пристрою. Використання камери стає новим етапом взаємодії з технологією, де Bixby є надійним супутником для виявлення і використання нових можливостей [21].

Результатом порівняльного аналізу п’яти вищезазначених голосових асистентів є таблиця 1.1.

Таблиця 1.1 – Порівняльна таблиця аналогів голосових асистентів

Джерело: побудовано автором

Характеристика Голосових помічників	**Amazon Alexa**	**Google Assistant**	**Apple Siri**	**Microsoft Cortana**	**Samsung Bixby**	**MyVoiceAssistant**
Сучасний дизайн	+	+	+	-	+	+
Зручний інтерфейс та навігація	+	+	+	+	+	+
Підтримка декількох мов	+	+	+	+	+	+
Виконання команд	+	+	+	+	+	+
Кросплатформеність	+	-	+	+	-	+
Генерація голос	+	+	+	-	-	+
Розпізнавання мови	+	+	+	+	+	+
Перекладання тексту	-	+	+	-	-	+
Зручне управління	+	+	+	-	-	+
Використання разом з IoT	+	+	+	-	-	-

Дані з таблиці 1.1 виділяють всі плюси та мінуси, з яких можна виділити певні функціональні рішення, які можна використати у власній розробці. Отже, голосовий асистент повинен:

- розпізнавання людського голосу,
- генерувати мовну відповідь (відповідати голосом),
- перекладати текст\голос на іншу мову,
- виконувати команди користувача,
- виконувати запити до нейромережі,
- відкривати файли\теки в системи,
- виконувати запуск різноманітних програм у ОС,
- мати можливість до налаштувань
- бути кросплатформним

2 ПОСТАВНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою кваліфікаційної роботи магістра є створення віртуального голосового асистента на основі штучного інтелекту. Розроблений голосовий помічник мусить мати попит серед звичайних користувачів як у рутинних справах: зменшувати час пошуку інформації, перекладати; так і допомагати користувачам у їх робочих процесах. Впровадження голосового помічника має скоротити час на пошук інформації, перекладу тексту, та, загалом, спрощувати користування персональним комп'ютером, що може підійти для людей з обмеженнями.

Розроблений голосовий помічник має задовольняти наступні функціональні вимоги:

- розпізнавання людського голосу,
- генерувати мовну відповідь (відповідати голосом),
- перекладати текст\голос на іншу мову,
- виконувати команди користувача,
- виконувати запити до нейромережі,
- відкривати файли\теки в системи,
- виконувати запуск різноманітних програм у ОС,
- мати зручний інтефрейс
- мати можливість до налаштувань
- бути кросплатформним

Голосовий асистент, що розробляється, повинен володіти інтерфейсом, який є не лише легким у використанні, але і зрозумілим при налаштуванні. Адже відчуття комфорту у користуванні представляє собою один з ключових факторів у визначенні пріоритетів при виборі програмного продукту.

Для досягнення мети даного проекту необхідно виконати наступні вимоги:

- провести аналіз аналогів голосових асистентів: визначити їх функціональні можливості. Необхідно визначити найоптимальніший функціонал для голосового асистента, які команди голосовий асистент матиме змогу виконувати, на основі проведеного аналізу аналогічних голлосових помічників.
- визначити програмні засоби реалізації голосового помічника: обрати найбільш підходящу мову для реалізації модулів розпізнавання команд та генерації людського голосу, модуля перекладу тексту, модуля налаштувань та формат збереження цих налаштувань. Обрати тип голосових моделей для модулів розпізнавання та генерації. Визначитись з інтеграцією чат-ботів.
- виконати моделювання функціональних можливостей голосового помічника за допомогою різноманітних діаграм для спрощення розуміння роботи голосового асистента,
- реалізувати голосовий помічник обраними програмними засобами та протестувати його роботу згідно вимог.

При визначенні цілей проекту було проведено ретельне планування виконання робіт, подробиці якого описано в Додатку А під час форматування мети проекту.

2.2 Методи дослідження

Після постановки мети проекту та визначення функціональних вимог до розроблювального голосового асистента, форматуванні обов'язкових для виконання задач проекту, необхідно визначитись з методами даного дослідження.

Загалом існують чотири методи проведення досліджень:

- метод моделювання [22]
- системно-функціональний метод
- теоретичний метод
- емпіричний метод

Теоретичний підхід передбачає докладний аналіз особливостей та можливостей обраної сфери. Проект "Віртуальний голосовий помічник на основі штучного інтелекту" використовує теоретичний метод для аналізу бізнес-процесів, потреб і тенденцій у галузі штучного інтелекту. Також при виборі інструментів для розробки буде використовуватись теоретичний підхід. Це було обґрунтовано необхідністю виділення переваг та недоліків різних технологій, їх особливостей застосування залежно від розміру проекту, тривалості його реалізації, методів збереження даних та інших важливих аспектів.

Використання емпіричного методу передбачає використання спостережень, вимірювань та експериментів для отримання пізнання у конкретній предметній області. Цей підхід був використаний на етапі аналізу процедур зміни налаштувань, мов генерації та розпізнавання людської мови в проекті "Віртуальний голосовий помічник на основі штучного інтелекту".

Метод моделювання включає в себе розробку схем і діаграм, які відображають процес розробки програмного продукту і його подальше використання. У відношенні до проекту "Віртуальний голосовий помічник на основі штучного інтелекту", цей метод був використаний на етапі створення діаграм IDEF0 та UseCase, що дозволило більш детально розібратися в процесі роботи та взаємодії з програмним продуктом.

Системно-функціональний метод описує систему, розглядаючи її компоненти, залежності та взаємодію в рамках єдиного цілого. Він був використаний при розробці програмного модулю проекту. А саме здійснення варіантів використання, зображених на діаграмі UseCase [23].

Що до програмної реалізації модуля голосового помічника, було обрано мову програмування Python [24], так як вона має широкий спектр бібліотек для обробки голосу та природної мови, таких як SpeechRecognition [25] та Natural Language Toolkit. Важливим аспектом є також забезпечення безпеки та конфіденційності даних користувачів. Усі дані голосового введення обробляються локально на пристрої користувача, і лише анонімізована інформація може передаватися для виконання певних завдань, забезпечуючи при цьому високий рівень захисту особистих даних.

Для модулю налаштувань – HTML [26], який буде використовуватися для створення шаблонів веб сторінок, також слід сказати про каскадні таблиці стилів CSS [27], що дозволить зробити веб-сторінки більш приємними для очей користувачів. Що до бекенду, він буде написаний з використанням сторонньої бібліотеки Flask[28], що дозволить оперувати даними через Python. Для організації збереження налаштувань голосового помічника, будуть використовуватись файли з розширенням JSON[29].

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Після вивчення предметної області, визначення актуальності завдання, постановки мети та задач на розробку, а також завершення процесу планування робіт для втілення проекту у життя, було вирішено перейти до наступного етапу – проектування голосового помічника. Було розроблено діаграми нотацій IDEF0, детально описано послідовність процесів роботи голосового асистента.

3.1 Діаграми нотації IDEF0

Для моделювання функцій роботи голосового асистента використовуються різні методи об'єктно орієнтованого та структурованого підходу. Однією з методологій є IDEF0. Вона реалізується за допомогою графічного відображення структури системи та процесів діяльності компанії у вигляді взаємопов'язаних функцій. Використання IDEF0 дозволяє розглядати функції ізольовано від об'єктів, які взаємодіють з цими функціями. Діаграма включає інформацію про вхідні, керуючі, вихідні та механізми [30].

Для контекстної діаграми процесу використання голосового помічника було визначено наступні дані:

1. Вхідні дані: JSON файл з налаштуваннями серверу, команда від користувача
2.
3. Управління: Інструкція користувача
4. Вихідні данні: згенерована відповідь голосового помічника.
5. Механізми: операційна система, Web-сервер та голосовий помічник

На рисунку 3.1 зображено контекстну діаграму IDEF0 для процесу використання віртуального голосового помічника

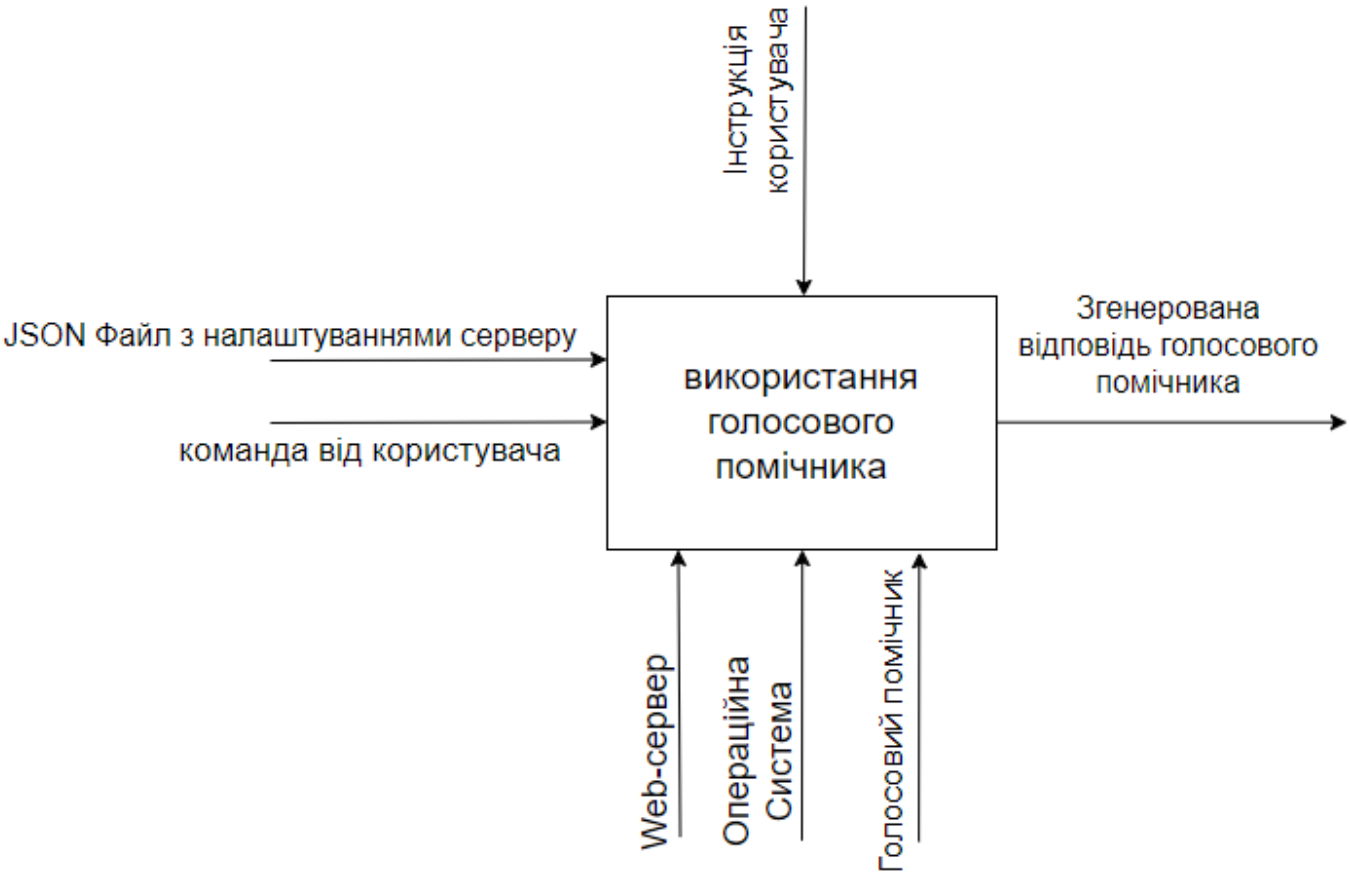

Рисунок 3.1 – Контекстна діаграма IDEF0. *Джерело: побудовано автором*

Для деталізації процесів інформаційної системи було описано декомпозицію першого рівня для процесу використання голосового помічника (рис. 3.2)

Дані для діаграми наступні:

1. Вхідні дані: JSON файл з налаштуваннями серверу, команда від користувача
2. Управління: Інструкція користувача
3. Вихідні данні: згенерована відповідь голосового помічника.
4. Механізми: операційна система, Web-сервер та голосовий помічник

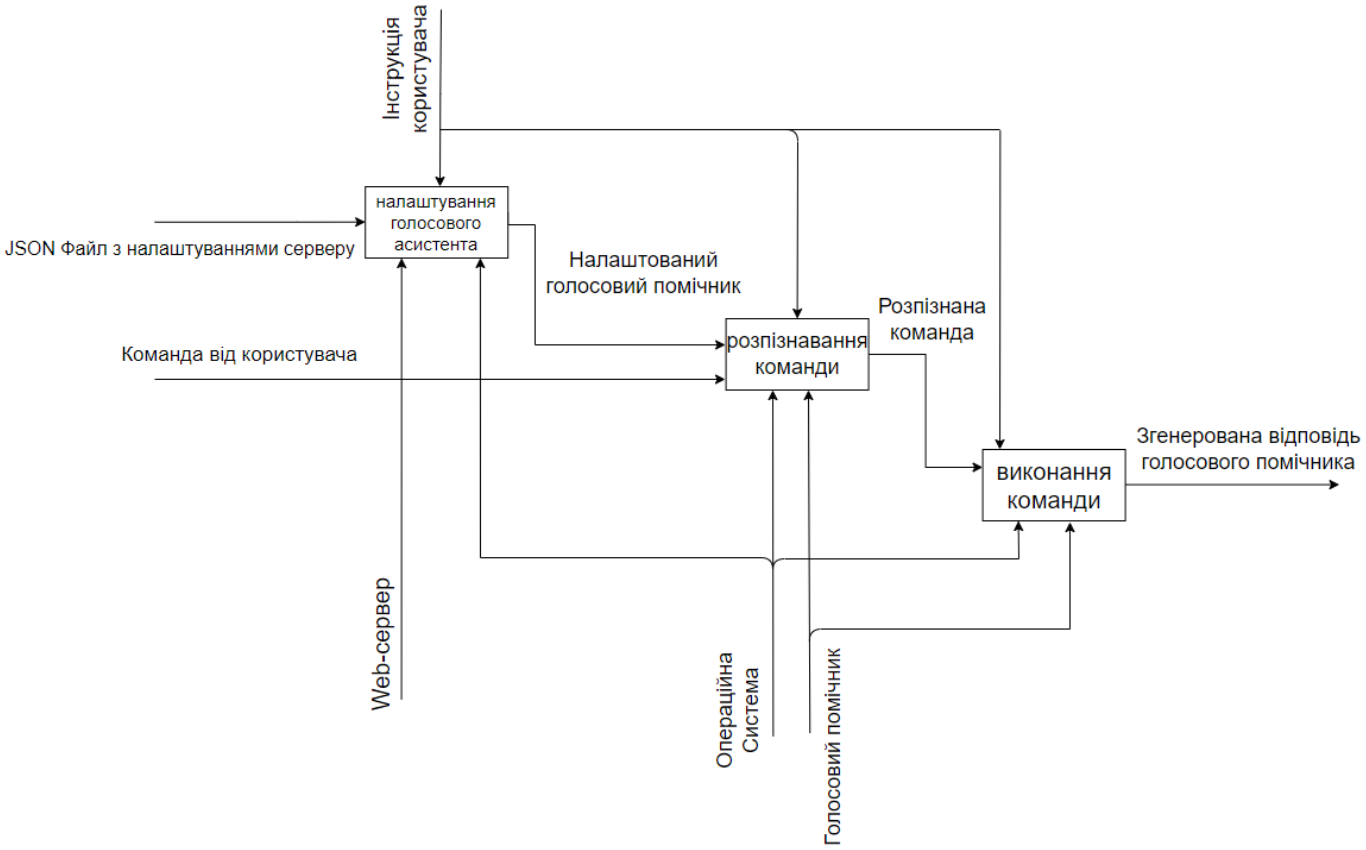

Рисунок 3.2 – Декомпозиція першого рівня процесу «Використання голосового помічника». *Джерело: побудовано автором*

Для деталізації процесу виконання програми було описано декомпозицію першого рівня (рис. 3.3)

Дані для діаграми наступні:

1. Вхідні дані: JSON файл з налаштуваннями серверу, команда від користувача
2. Управління: Інструкція користувача
3. Вихідні данні: згенерована відповідь голосового помічника.
4. Механізми: операційна система, Web-сервер та голосовий помічник

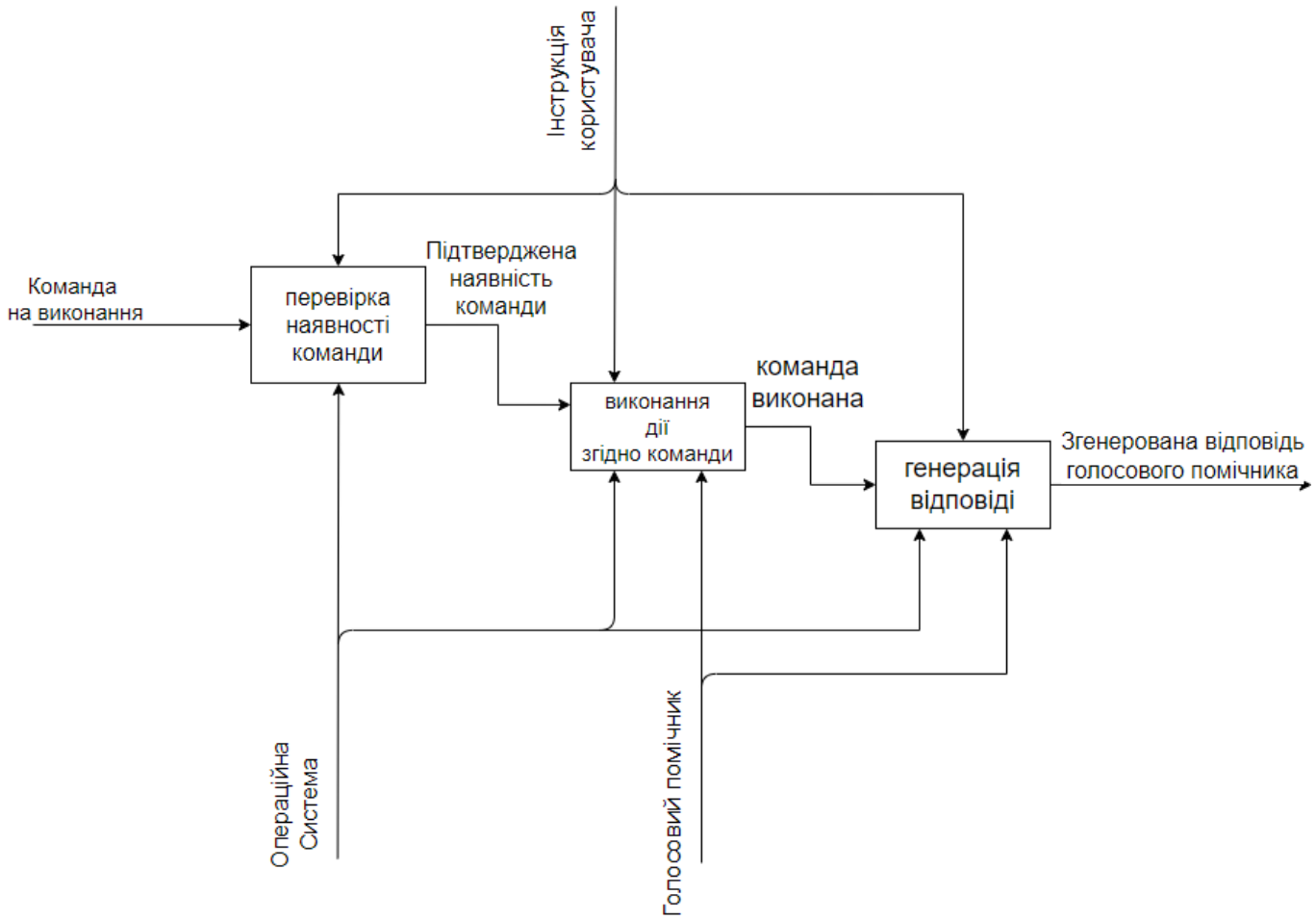

Рисунок 3.3 – Декомпозиція першого рівня для процесу «Виконання команди».

Джерело: побудовано автором

3.2 Use Case діаграми

Після завершення етапу моделювання важливо створити діаграму варіантів використання (Use Case diagram). Ця діаграма є ключовим інструментом у сфері розробки програмного забезпечення, особливо в рамках об’єктно-орієнтованого аналізу та проектування. Діаграма варіантів використання дозволяє візуалізувати функціональні вимоги системи та взаємозв'язки між різними користувачами (акторами) та самою системою [31]. Основними складовими діаграми варіантів використання є актори, відносини, системні межі та самі варіанти використання. Діаграми Use Case поліпшують розуміння функціоналу системи, допомагають визначити вимоги користувачів та сприяють ефективній комунікації між розробниками та зацікавленими сторонами.

Для віртуального голосового помічника на основі штучного інтелекту можна виділити такі use case-и:

– Дії зі списком команд
– Дії з ідентифікатором голосового асистента
– Змін мови голосового асистента
– Розпізнавання команди користувача
– Генерація команд для голосового асистента

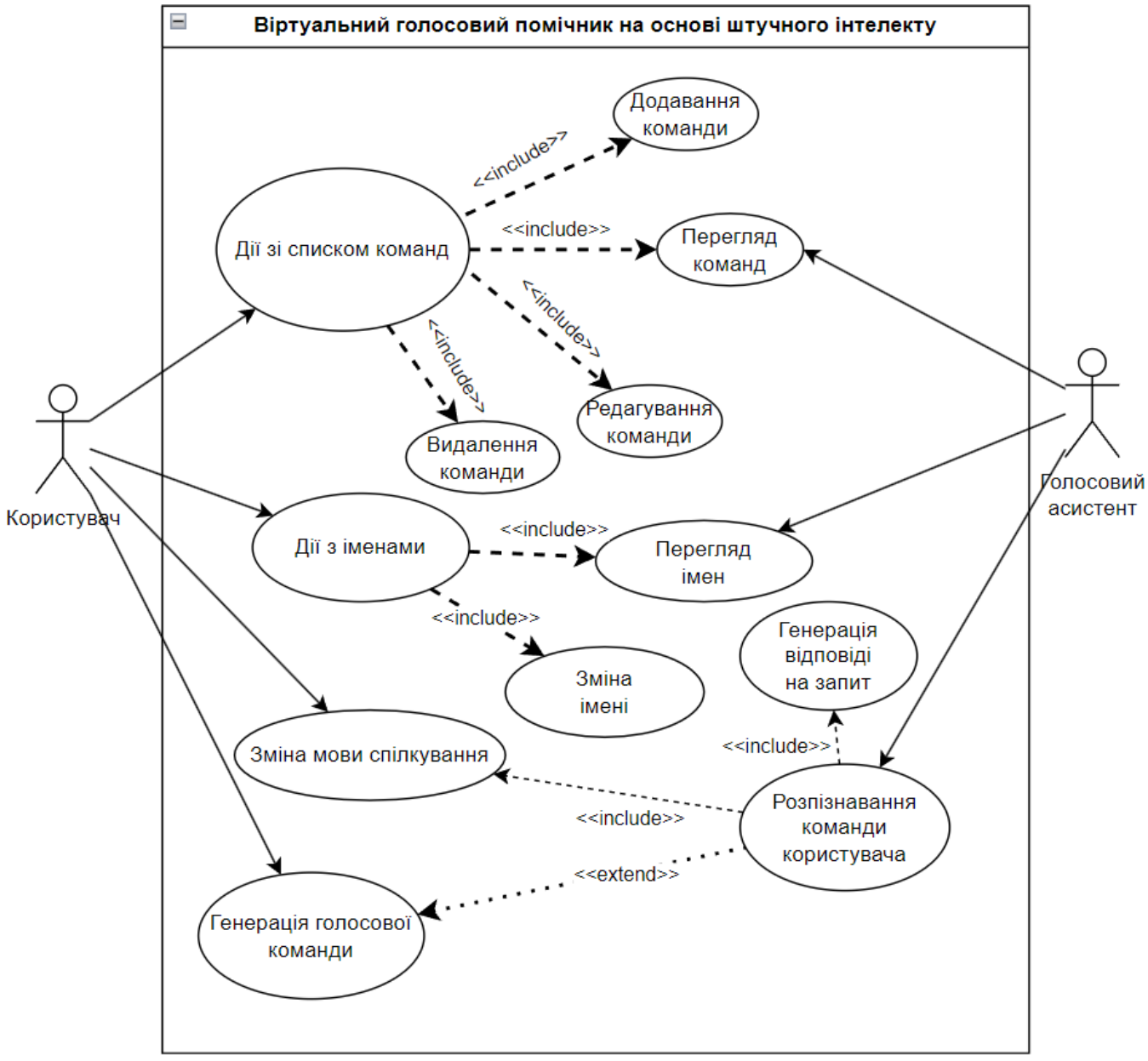

Рисунок 3.4 – Use Case діаграма. *Джерело: побудовано автором*

4 ПРАКТИЧНА РОЗРОБКА РІШЕННЯ

4.1 Архітектура голосового асистента

Процес створення голосового помічника на базі штучного інтелекту розпочинається з визначення архітектури програмного додатку за допомогою діаграми високого рівня проекту (HLD) [32]. Ця діаграма призначена для визначення компонентів програмного застосунку.

Архітектура голосового помічника на базі штучного інтелекту включає наступні елементи:

- Модуль розпізнавання голосової команди – є основною системою комунікації з користувачем
- Модуль командного процесора – основний модуль, який виконує команди
- Модуль генерації голосу – є одним зі способів отримання інформації вед голосового помічника
- Модуль налаштувань – являє собою модуль, який надає налаштування для всіх модулів, які потребують конфігурації\
- Модуль перекладання тексту – є інструментом перекладу тексту різних мов
- Модуль зв’язку з ChatGPT – виконує запити до нейромережі

Рисунок 4.1 – Діаграма високого рівня. *Джерело: побудовано автором*

4.2 Розробка модулю налаштувань

Першим етапом написання модулю налаштувань – є завантаження редакторів та встановлення необхідних бібліотек. Для написання програмної реалізації всіх модулів включно з модулем налаштувань було використано редактор PyCharm Community [33]. Так як редактор PyCharm є безкоштовним, і повної підтримки розробки сайтів в ньому немає, на відміну від PyCharm Professional, для написання HTML та CSS коду було використано VSCode [34], а щоб забезпечити контроль розробки та версій – Git[35].

Модуль налаштувань можна поділити на чотири логічні розділі, а саме:

Частина 1 модулю налаштувань: це клас `SettingsManager`, який описує логіку збереження даних. Цей клас використовую бібліотеку JSON [36] для збереження даних в однойменному форматі. Програмний код цього класу представлено нижче:

```
import default_config_assistant as default_config
import json


class SettingsManager:
    # init
    def __init__(self, filename:str='settings.json'):
        self.filename = filename
        self.settings = default_config.DEFAULT__SETTINGS


    # loading settings
    def load_settings(self):
        try:
            with open(self.filename, 'r') as file:
                try:
                    self.settings = json.load(file)
                except json.JSONDecodeError:
                    self.settings = json.load(file)
        except FileNotFoundError:
            self.settings = default_config.DEFAULT__SETTINGS


    # saving settings
    def save_settings(self):
        with open(self.filename, 'w') as file:
            json.dump(self.settings, file, indent=4)


    # getting settings
```

```
    def get_setting(self, key:str, default=None):
        return self.settings.get(key, default)


    # setting settings
    def set_setting(self, key:str, value):
        self.settings[key] = value


    # deleting settings from key
    def delete_setting_from_key(self, key:str, key_to_remove:str):
        if key in self.settings and key_to_remove in
self.settings[key]:
            del self.settings[key][key_to_remove]
            self.save_settings()
            self.load_settings()


    # deleting value by key
    def delete_value_from_key(self, key:str, value:str):
        if key in self.settings and value in self.settings[key]:
            self.settings[key].remove(value)
            self.save_settings()
            self.load_settings()


# creating settings_manager object
settings_manager = SettingsManager()
```

Клас `SettingsManager` містить наступні методи:

- метод `__init__` – простими словами, це конструктор,
- `loading_settings` – метод завантаження налаштувань з файлу налаштувань, якщо такого файлу немає – використовуються налаштування за замовчуванням,
- `save_settings` – метод збереження налаштувань у файл.
- `get_setting` – метод який надає налаштування по ключу, необхідно для перегляду налаштувань по ключу,
- `set_setting` – метод для збереження налаштувань по ключу,
- `delete_setting_from_key` – видаляє налаштування з ключу,
- `delete_value_from_key` – видаляє значення по ключу.

Частина 2 та 3 модулю налаштувань – це кінцеві точки Web-серверу Flask та обробники для цих кінцевих точок, знаходяться в одному файлі server.py, та використовують наступні бібліотеки:

- `from SettingsManager import SettingsManager, settings_manager` – клас для дійй з налаштуваннями та сам екземпляр цього класу.
- `from flask import Flask, send_file, render_template, url_for, request` – бібліотека веб серверу та методи необхідні для опису кінцевих точок.
- `from multiprocessing import Process` – бібліотека на базі якої реалізовано окремий процес для модулю налаштувань [37].
- `from default_config_assistant import last_request_time` – значення останнього запиту до серверу, це необхідно для вимкнення серверу після неактивного проміжку часу
- `import json` – конвертація з та в JSON формат
- `import webbrowser` – бібліотека за допомогою якої автоматично відкривається браузер після запуску серверу [38].
- `import re` – бібліотека яка підтримує регулярні вирази [39].
- `import os` – бібліотека яка підтримує різні інтерфейси операційної системи [40].
- `import time` – доступ до часу та його конвертації [41].
- `import sys` – бібліотека яка підтримує специфічні для системи параметри та функції [42]
- `import signal` – встановлює обробники для асинхронних подій [43].
- `import logging` – підтримує можливість входу в систему для Python [44].

Повний програмний код реалізації модуля Web-серверу налаштувань приведено в розділі Б.1 Додатку Б.

Остання, четверта, частина цього модулю присвячена візуальному web-сторінок та їх оформленню. Шаблони сторінок модулю налаштувань представлено на рисунках 4.2-4.11

```
{% if message_type %}
<div class="message {{message_type}}">
    {{ message_text }}
</div>
{% endif %}

<div class="main">
    <form class="content" method="POST">
        <h1>Voice Assistant</h1>
        <div class="horizontal-flex">
            <label>Ім'я голосового помічника</label>
            <a href="{{ url_for('voice_assistant_names') }}"><div class="link">Редагувати</div></a>
        </div>
        <div class="horizontal-flex">
            <label>Команди голосового помічника</label>
            <a href="{{ url_for('commands_list') }}"><div class="link">Редагувати</div></a>
        </div>
        <div class="horizontal-flex">
            <label>Розпізнавання голосу:</label>
            <select name="sttLang" id="sttLang">
                {% for key, value in assistant_stt.items() %}
                <option value="{{ key }}"{% if current_settings["ASSISTANT_STT"] == key %}selected{% endif %}>{{ value.label }}</option>
                {% endfor %}
            </select>
        </div>
        <div class="horizontal-flex">
            <label>Генерація голосу:</label>
            <select name="ttsLang" id="ttsLang">
                {% for key, value in assistant_tts.items() %}
                <option value="{{ key }}"{% if current_settings["ASSISTANT_TTS"] == key %}selected{% endif %}>{{ value.label }}</option>
                {% endfor %}
            </select>
        </div>
        <div class="horizontal-flex">
            <label>Переклад на:</label>
            <select name="transLang" id="transLang">
                {% for key, value in assistant_tra.items() %}
                <option value="{{ key }}"{% if current_settings["ASSISTANT_TRA"] == key %}selected{% endif %}>{{ value.label }}</option>
                {% endfor %}
            </select>
        </div>
        <div class="under">
            <a href="{{ url_for('other_settings') }}"><div class="link">Інше</div></a>
            <a href="{{ url_for('add_languages') }}"><div class="link">Додати мови</div></a>
            <button type="submit" value="save">Зберегти</button>
        </div>
    </form>
</div>
```

Рисунок 4.2 – Шаблон домашньої сторінки. *Джерело: побудовано автором*

```
{% if message_type %}
<div class="message {{message_type}}">
    {{ message_text }}
</div>
{% endif %}

<div class="main">
    <form class="content" method="POST">
        <div class="underlogotxtEnable">
            <h1>Voice Assistant</h1>
            <span class="under-logotext">Інші налаштування</span>
        </div>
        <div class="checkboxes">
            <input type="checkbox" name="SpeakTheAnswer" id="SpeakTheAnswer" value="True"
            {% if current_settings["SPEAK_THE_ANSWER"] == "True" %}checked{% endif %}>
            <label for="SpeakTheAnswer">Озвучувати відповідь голосового помічника</label>
        </div>
        <div class="checkboxes">
            <input type="checkbox" name="IsQuickAnswer" id="IsQuickAnswer" value="True"
            {% if current_settings["IS_QUICK_ANSWER"] == "True" %}checked{% endif %}>
            <label for="IsQuickAnswer">Використовувати короткі відповіді для ChatGPT</label>
        </div>
        <div class="under">
            <button type="submit" value="save">Зберегти</button>
            <a href="{{ url_for('serve_html') }}"><div class="link">Назад</div></a>
        </div>
    </form>
</div>
```

Рисунок 4.3 – Шаблон сторінки «Інші налаштування». *Джерело: побудовано автором*

```
{% if message_type %}
<div class="message {{message_type}}">
    {{ message_text }}
</div>
{% endif %}
<div class="main">
    <form class="content" method="POST">
        <div class="underlogotxtEnable">
            <h1>Voice Assistant</h1>
            <span class="under-logotext">Імена голосого помічника</span>
        </div>
        <div class="va-names">
        {% for value in assistant_alias %}
            <div class="va-name">
                <span>{{ value }}</span>
                <button type="submit" name="delete_name" value="{{ value }}">
                    <i class="fa-solid fa-xmark"></i>
                </button>
            </div>
        {% endfor %}
        </div>
        <div class="horizontal-flex margin-top">
            <label>Нове ім'я: </label>
            <div class="in-horizontal-flex">
                <input type="text" name="new_name" placeholder="Введіть нове ім'я" maxlength="10" value="{{ new_name }}">
                <button type="submit" name="add_name" value="add_name">Додати ім'я</button>
            </div>
        </div>
        <div class="under">
            <a href="{{ url_for('serve_html') }}"><div class="link">Назад</div></a>
        </div>
    </form>
</div>
```

Рисунок 4.4 – Шаблон сторінки «Імена голосового помічника». *Джерело: побудовано автором*

```
<div class="main">
    <form class="content" method="POST">
        <div class="underlogotxtEnable">
            <h1>Voice Assistant</h1>
            <span class="under-logotext">Список команд</span>
        </div>
        <table>
            <tr>
                <th class="command_name">Назва команди та редагування:</th>
                <th class="command_delete">Видалення:</th>
            </tr>
            {% for command, details in assistant_cmd_list.items() %}
              <tr>
                <td class="command_name">
                  <button type="submit" name="command_edit" value="{{ command }}">
                    <span>{{ command }}</span>
                    <i class="fa-solid fa-pen-to-square"></i>
                  </button>
                </td>
                <td class="command_delete">
                    {% if details.can_delete != 'False' %}
                        <button type="submit" name="command_delete" value="{{ command }}">
                            <i class="fa-regular fa-trash-can"></i>
                    {% else %}
                        <button>
                            <i class="fa-solid fa-shield-halved tooltip">
                                <span class="tooltiptext">Це базова команда, ви не можете її видалити</span>
                            </i>
                    {% endif %}
                  </button>
                </td>
              </tr>
            {% endfor %}
        </table>
        <div class="under">
            <a href="{{ url_for('add_command') }}"><div class="link">Додати команду</div></a>
            <a href="{{ url_for('serve_html') }}"><div class="link">Назад</div></a>
        </div>
    </form>
</div>
```

Рисунок 4.5 – Шаблон сторінки «Список команд голосового помічника». *Джерело: побудовано автором*

```
<form class="content" method="POST">
    <div class="underlogotxtEnable">
        <h1>Voice Assistant</h1>
        <span class="under-logotext">Редагування команди {{ command_edit }}</span>
    </div>
    <div class="horizontal-flex margin-top">
        <label>Назва команди (англійскою)</label>
        <div class="in-horizontal-flex">
            <input type="text" name="key" placeholder="Назва команди" maxlength="10" minlength="2" value="{{ command_edit }}" readonly>
            <i class="fa-solid fa-circle-question tooltip">
                <span class="tooltiptext">Введіть назву команди англійською. {% if assistant_cmd_list[command_edit].can_delete == false %}<br><br> Назву базової команди не можна змінювати {% endif %}</span>
            </i>
        </div>
    </div>
    <div class="horizontal-flex margin-top with-textarea">
        <label>Ключові слова</label>
        <div class="in-horizontal-flex">
            <textarea cols="33" name="word_list" placeholder="Ключові слова" maxlength="100" minlength="4">{{ word_list_str }}</textarea>
            <i class="fa-solid fa-circle-question tooltip">
                <span class="tooltiptext">Введіть ключові слова. Ключові слова слугують основним джерелом інформації для боту, що буде розпізнаватися. <br><br>Ключові слова розділяються комами</span>
            </i>
        </div>
    </div>
    {% if assistant_cmd_list[command_edit].isCustom == "True" %}
    <div class="horizontal-flex">
        <label>Тип команди:</label>
        <select name="commandType" id="commandType">
            <option value="None"{% if commandType == 'None' %} selected {% endif %}>None</option>
            <option value="explorer"{% if commandType == 'explorer' %} selected {% endif %}>Відкрити теку в системі</option>
            <option value="execute"{% if commandType == 'execute' %} selected {% endif %}>Запустити програму</option>
            <option value="openWebPage"{% if commandType == 'openWebPage' %} selected {% endif %}>Відкрити веб сторінку</option>
        </select>
        <i class="fa-solid fa-circle-question tooltip">
            <span class="tooltiptext">Оберіть тип кастомної команди:<br> Відкрити теку в системі - відкриє теку;<br> Запустити програму - запустить команду;<br> Відкрити веб сторінку - Відкриє веб сторінку</span>
        </i>
    </div>
    <div class="horizontal-flex margin-top with-textarea">
        <label>Команда на виконання</label>
        <div class="in-horizontal-flex">
            <textarea cols="33" name="customCommand" placeholder="Шлях до теки / Шлях до програми / Посилання на веб сторінку" maxlength="100" minlength="4">{{ customCommand }}</textarea>
            <i class="fa-solid fa-circle-question tooltip">
                <span class="tooltiptext">Введіть шлях до теки <br> або шлях до виконавчого файлу програми <br> або посилання по веб сторінку </span>
            </i>
        </div>
    </div>
    {% endif %}
    <div class="under">
        <button type="submit" name="edit" value="pass">Редагувати</button>
        <a href="{{ url_for('commands_list') }}"><div class="link">Назад</div></a>
    </div>
```

Рисунок 4.6 – Шаблон сторінки «Редагувати команду». *Джерело: побудовано автором*

```
<div class="main">
    <form class="content" method="POST">
        <div class="underlogotxtEnable">
            <h1>Voice Assistant</h1>
            <span class="under-logotext">Додавання мов</span>
        </div>
        <div class="horizontal-flex margin-top">
            <label>Мова розпізнавання голосу</label>
            <a href="{{ url_for('add_stt_language') }}"><div class="link">Додати</div></a>
        </div>
        <div class="horizontal-flex">
            <label>Мова генерації голосу</label>
            <a href="{{ url_for('add_tss_language') }}"><div class="link">Додати</div></a>
        </div>
        <div class="horizontal-flex no-margin">
            <label>Мова перекладу</label>
            <a href="{{ url_for('add_translate_language') }}"><div class="link">Додати</div></a>
        </div>
        <div class="under">
            <a href="{{ url_for('serve_html') }}"><div class="link">Назад</div></a>
        </div>
    </form>
</div>
```

Рисунок 4.7 – Шаблон сторінки «Додавання мов». *Джерело: побудовано автором*

```
<div class="main">
    <form class="content" method="POST">
        <div class="underlogotxtEnable">
            <h1>Voice Assistant</h1>
            <span class="under-logotext">Додати мову розпізнавання голосу</span>
        </div>
        <div class="horizontal-flex margin-top">
            <label>Лейбл мови</label>
            <div class="in-horizontal-flex">
                <input type="text" name="label" placeholder="Лейбл мови" maxlength="10" minlength="2" value="{{ label }}">
                <i class="fa-solid fa-circle-question tooltip">
                    <span class="tooltiptext">Назовіть мову на ваш розсуд</span>
                </i>
            </div>
        </div>
        <div class="horizontal-flex margin-top">
            <label>2х символьний код мови:</label>
            <div class="in-horizontal-flex">
                <input type="text" name="key" placeholder="2х символьний код мови" maxlength="10" minlength="2" value="{{ key }}">
                <i class="fa-solid fa-circle-question tooltip">
                    <span class="tooltiptext">Двох символьний код мови за стандартом <a href="https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes" target="_blank">ISO-639-1</a></span>
                </i>
            </div>
        </div>
        <div class="horizontal-flex margin-top">
            <label>Назва папки моделі:</label>
            <div class="in-horizontal-flex">
                <input type="text" name="model" placeholder="Назва папки моделі"  maxlength="30" minlength="5" value="{{ model }}">
                <i class="fa-solid fa-circle-question tooltip">
                    <span class="tooltiptext">Введіть назву папки моделі позпізнавання голосу, яка лежить в корні програми в папці "models_stt"
                        <br><br>
                        Завантажте модель з <a href="https://alphacephei.com/vosk/models" target="_blank">сайту</a>, розархівуйте її до "models_stt" теки в корні програми
                    </span>
                </i>
            </div>
        </div>
        <div class="under">
            <button type="submit" value="save">Додати</button>
            <a href="{{ url_for('add_languages') }}"><div class="link">Назад</div></a>
        </div>
    </form>
</div>
```

Рисунок 4.8 – Шаблон сторінки «Додати мову розпізнавання голосу». *Джерело: побудовано автором*

```
<div class="main">
    <form class="content" method="POST">
        <div class="underlogotxtEnable">
            <h1>Voice Assistant</h1>
            <span class="under-logotext">Додати мову перекладу тексту</span>
        </div>
        <div class="horizontal-flex margin-top">
            <label>Лейбл мови</label>
            <div class="in-horizontal-flex">
                <input type="text" name="label" placeholder="Лейбл мови" maxlength="10" minlength="2" value="{{ label }}">
                <i class="fa-solid fa-circle-question tooltip">
                    <span class="tooltiptext">Назовіть мову на ваш розсуд</span>
                </i>
            </div>
        </div>
        <div class="horizontal-flex margin-top">
            <label>2х символьний код мови:</label>
            <div class="in-horizontal-flex">
                <input type="text" name="key" placeholder="2х символьний код мови" maxlength="2" minlength="2" value="{{ key }}">
                <i class="fa-solid fa-circle-question tooltip">
                    <span class="tooltiptext">Двох символьний код мови за стандартом <a href="https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes" target="_blank">ISO-639-1</a></span>
                </i>
            </div>
        </div>
        <div class="under">
            <button type="submit" value="save">Додати</button>
            <a href="{{ url_for('add_languages') }}"><div class="link">Назад</div></a>
        </div>
    </form>
</div>
```

Рисунок 4.9 – Шаблон сторінки «Додати мову розпізнавання тексту». *Джерело: побудовано автором*

```
<div class="main">
    <form class="content" method="POST">
        <div class="underlogotxtEnable">
            <h1>Voice Assistant</h1>
            <span class="under-logotext">Додати мову генерації голосу</span>
        </div>
        <div class="horizontal-flex margin-top">
            <label>Лейбл мови</label>
            <div class="in-horizontal-flex">
                <input type="text" name="label" placeholder="Лейбл мови" maxlength="10" minlength="2" value="{{ label }}">
                <i class="fa-solid fa-circle-question tooltip">
                    <span class="tooltiptext">Назовіть мову на ваш розсуд</span>
                </i>
            </div>
        </div>
        <div class="horizontal-flex margin-top">
            <label>2х символьний код мови:</label>
            <div class="in-horizontal-flex">
                <input type="text" name="key" placeholder="2х символьний код мови" maxlength="2" minlength="2" value="{{ key }}">
                <i class="fa-solid fa-circle-question tooltip">
                    <span class="tooltiptext">
                        Двох символьний код мови за стандартом <a href="https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes" target="_blank">ISO-639-1</a>
                    </span>
                </i>
            </div>
        </div>
        <div class="horizontal-flex margin-top">
            <label>ID моделі:</label>
            <div class="in-horizontal-flex">
                <input type="text" name="id_model" placeholder="ID моделі" maxlength="10" minlength="3" value="{{ id_model }}">
                <i class="fa-solid fa-circle-question tooltip">
                    <span class="tooltiptext">Введіть ID моделі згідно документації <a href="https://github.com/snakers4/silero-models" target="_blank">Silero</a></span>
                </i>
            </div>
        </div>
        <div class="horizontal-flex margin-top">
            <label>Спікер:</label>
            <div class="in-horizontal-flex">
                <input type="text" name="speaker" placeholder="Спікер" maxlength="10" minlength="4" value="{{ speaker }}">
                <i class="fa-solid fa-circle-question tooltip">
                    <span class="tooltiptext">Введіть Спікера згідно документації <a href="https://github.com/snakers4/silero-models" target="_blank">Silero</a></span>
                </i>
            </div>
        </div>
        <div class="under">
            <button type="submit" value="save">Додати</button>
            <a href="{{ url_for('add_languages') }}"><div class="link">Назад</div></a>
        </div>
    </form>
</div>
```

Рисунок 4.10 – Шаблон сторінки «Додати мову генерації голосу». *Джерело: побудовано автором*

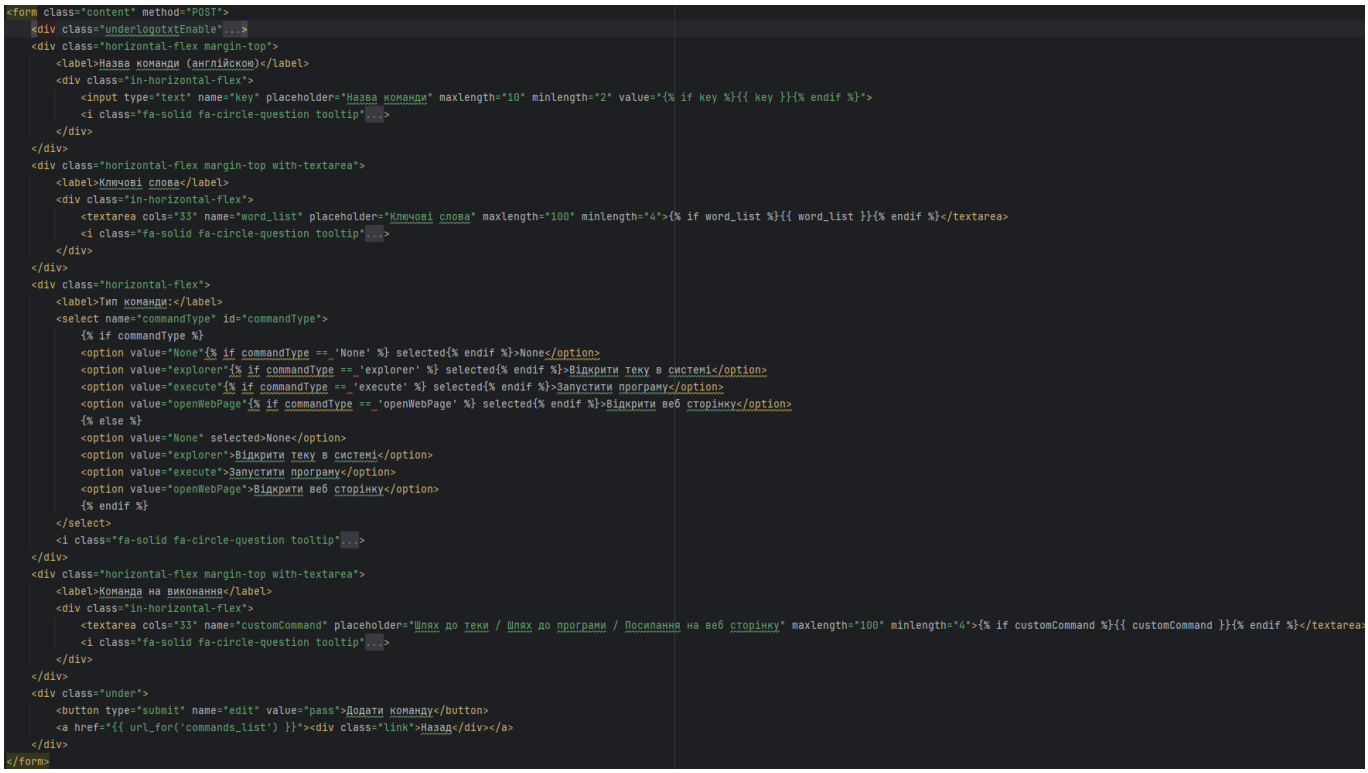

Рисунок 4.11 – Шаблон сторінки «Додати команду». *Джерело: побудовано автором*

4.3 Розробка модулю розпізнавання команд

При розробці модулю розпізнавання голосових команд користувача використовувались наступні бібліотеки: `vosk` [45], `sys`, `sounddevice` [46], `queue` [47], `json`.

Було описано клас `SpeachToText`, який має наступні методи, ініціюючий та `get_model`. Також було описано функцію `listen`, що викликає вбудований метод бібліотеки `vosk` на розпізнавання людської мови для подальшої оброки. Бібліотека `vosk` використовую вже нейронну навчену модель розпізнавання людської мови. Нижче наведено програмний код класу `SpeachToText`. Повний модулю приведено в розділі Б.2 Додатку Б.

```
class SpeachToText:
    # init
    def __init__(self, model:str):
        self.model = vosk.Model('models_stt/' + model)
        self.sample_rate = 16000
        self.block_size = 8000
        self.device = 1
        self.d_type = 'int16'
        self.channels = 1


    # getting the model
```

```
        def get_model(self):
            return {'model': self.model,
                    'sample_rate': self.sample_rate,
                    'block_size': self.block_size,
                    'device': self.device,
                    'd_type': self.d_type,
                    'channels': self.channels}

        # setting the model
        def set_model(self):
            pass
```

Розпізнавання команди використовує бібліотеки `fuzzywuzzy` [48]. Ця бібліотека використовується в функції: `recognize_cmd`. Блок коду, який відповідає за розпізнавання команди приведено нижче:

```
    # recognizing function
    def recognize_cmd(cmd: str, assistant_cmd_list):
        rc = {'cmd': '', 'percent': 0, 'key_world': ''}

        for word_list_key in assistant_cmd_list:
            word_list = assistant_cmd_list[word_list_key]["word_list"]
            max_per, key_world = max((fuzz.partial_ratio(cmd, x), x)
for x in word_list)
            if max_per > rc['percent']:
                rc['cmd'] = word_list_key
                rc['percent'] = max_per
                rc['key_world'] = key_world

        return rc

    # voice recognition callback function
    def stt_respond(voice: str, ...):
        if voice.startswith(assistant_alias):
            cmd = recognize_cmd(filter_cmd(voice, ...),
assistant_cmd_list)
            if cmd['cmd'] not in assistant_cmd_list.keys():
                show_notification("Голосовий помічник", "Я Вас не
зрозумів")
```

```
            else:
                execute_cmd(cmd['cmd'], cmd['key_world'], voice, ...)


    # filtering the CMD, removing epy assistant alias from raw_voice
    def filter_cmd(raw_voice: str, assistant_alias):
        cmd = raw_voice

        for x in assistant_alias:
            cmd = cmd.replace(x, "").strip()

        return cmd
```

4.4 Розробка модулю генерації голосу

Для того, щоб надавати відповіді користувачеві було реалізовано клас VoiceModel, за допомогою якого людська мова створюється та відтворюється користувачеві. Цей клас використовує наступні бібліотеки: `json`, `torch` [49], `sounddevice`, `time`, `silero` [50], `numpy` [51], `wave` [52], `sys`, `os`. Бібліотека `silero` використовує вже навчену нейронну модель генерації голосу. В класі реалізовані методи: `__init__`, `create_model`, `create_audio` та `play_audio`. Нижче наведено код модулю `VoiceModel`.

```
    class VoiceModel:
        repo_or_dir = 'snakers4/silero-models'
        model = 'silero_tts'


        # init
        def __init__(self, language:str, model_id:str,
sample_rate:str, speaker:str, put_accent:bool =True, put_yo:bool =True):
            self.language = language
            self.model_id = model_id
            self.sample_rate = sample_rate
```

```
            self.speaker = speaker
            self.put_accent = put_accent
            self.put_yo = put_yo
            self.device = torch.device('cpu')
            self.audio = None
            self.model = None
            VoiceModel.create_model(self)



        # create model method
        def create_model(self):
            sys.stderr = open(os.devnull, "w")
            self.model,                                          _                                    =
torch.hub.load(repo_or_dir=VoiceModel.repo_or_dir,
                                           model=VoiceModel.model,
                                           language=self.language,
                                           speaker=self.model_id)
            self.model.to(self.device)



        # create audio method
        def create_audio(self, text:str):
            try:
                self.audio = self.model.apply_tts(text=text,
                                                  speaker=self.speaker,

sample_rate=self.sample_rate,

put_accent=self.put_accent,
                                                  put_yo=self.put_yo)
            except ValueError:
                show_notification("Голосовий помічник","Я вас не
зрозумів")
```

```
        # play audio method
        def play_audio(self, text:str):
            try:
                VoiceModel.create_audio(self, text)
                sd.play(self.audio, self.sample_rate)
                time.sleep(len(self.audio) / self.sample_rate)
                sd.stop()
            except TypeError:
                show_notification("Голосовий помічник","Я вас не
зрозумів")
```

4.5 Розробка модулю перекладу

Для того, щоб голосовий помічник мав змогу перекладати текст з однієї мови на іншу, було описано модуль перекладу, який перекладає розпізнаний голос користувача представлений текстом на іншу мову, яка виставлена в налаштуваннях голосового асистента. Цей модуль було побудовано на базі готової бібліотеки `translator` [53]. Було описано клас `TranslatorModel` і реалізовано наступні методи `__init__`, `change_langs`, `translate_text`. Нижче наведено код модуля перекладу:

```
    from translate import Translator


    class TranslatorModel:
        # init
        def __init__(self, from_lang: str, to_lang: str):
            self.to_lang = to_lang
            self.from_lang = from_lang


        # changing the language
        def change_langs(self, to_lang: str, from_lang: str = None) ->
None:
            self.to_lang = to_lang
            if from_lang is not None:
                self.from_lang = from_lang
```

```
        # translating function
        def translate_text(self, text_to_translate: str) -> str:
            translator     =     Translator(from_lang=self.from_lang,
to_lang=self.to_lang)
            return translator.translate(text_to_translate)
```

4.6 Розробка модулю інтеграції Chat-GPT

Щоб інтегрувати та використовувати Chat-GPT у голосовий асистент було описано клас `TextNeuralNetwork`, який використовує бібліотеки `freeGPT` [54] та `asyncio` [55] та раніше описаний модуль налаштувань `SettingsManager`. В класі `TextNeuralNetwork` реалізуються методи: `__init__` та `create_prompt`. Під час використання методу `create_prompt` генерується асинхронний запит до Chat-GPT, після отримання відповіді, вона викладається користувачеві через командний процесор. Нижче наведено код модуля `TextNeuralNetwork`:

```
from freeGPT import AsyncClient
from SettingsManager import SettingsManager, settings_manager
import asyncio


class TextNeuralNetwork:
    def __init__(self):
        settings_manager.load_settings()
        current_settings =
settings_manager.get_setting('CURRENT_SETTINGS', {})
        self.model = "gpt3"
        self.is_quick_answer = current_settings['IS_QUICK_ANSWER']


    # chatGPT request
    async def create_prompt(self, prompt: str):
        try:
            if self.is_quick_answer == 'True':
                prompt += ' (Дай найкоротшу відповідь на тій мові, яка
було до дужок)'
```

```
                resp = await AsyncClient.create_completion(self.model,
prompt)
                return resp
            else:
                resp = await AsyncClient.create_completion(self.model,
prompt)
                return resp

        except Exception as e:
            return f"Error {e}"
```

4.7 Розробка модулю командного процесора

Командний процесор є основним модулем, що виконує команди які надсилаються модулем розпізнавання команди. Нижче наведено лістинг коду командного процесора. Повний лістинг програмного коду наведено в розділі Б.3 додатку Б:

```
    cur_tra_to_lang = current_settings['ASSISTANT_TRA']
    cur_speach_lang = current_settings['ASSISTANT_TTS']
    cur_speaker_lang = current_settings['ASSISTANT_STT']
    is_quick_answer = current_settings['IS_QUICK_ANSWER']
    speak_the_answer = current_settings['SPEAK_THE_ANSWER']



    if cmd == 'help':
    # Add browser opening to documentation
        webbrowser.open('…')



    # command time - makes notifications with time \ or says so
    elif cmd == 'time':
        current_time = datetime.now()
        p = inflect.engine()
```

```
        hour_text = p.number_to_words(current_time.hour)
        minute_text = p.number_to_words(current_time.minute)
        time_text = f"'{hour_text} {p.plural('hour', current_time.hour)} and {minute_text} {p.plural('minute', current_time.minute)} at the moment"

        translate = TranslatorModel('en', cur_speach_lang)
        translated_text = translate.translate_text(time_text)

        if speak_the_answer == "False":
            show_notification("Голосовий помічник",translated_text)
        else:
            speech_the_text(translated_text, tts_model)


    # command open browser - opens the browser
    elif cmd == 'open browser':
        webbrowser.open('https://www.google.com.ua/?hl=uk')


    # The joke command - makes a notification with a joke \ or tells a joke
    elif cmd == 'joke':
        jokes = []
        joke = random.choice(jokes)
        translated_text = ''
        if speak_the_answer == "False":
            if cur_speach_lang != 'ua':
                translate = TranslatorModel('uk', cur_speach_lang)
                translated_text = translate.translate_text(joke)
            else:
                translated_text = joke
            show_notification("Голосовий помічник",translated_text)
```

```
        else:
            if cur_speach_lang != 'ua':
                translate = TranslatorModel('uk', cur_speach_lang)
                translated_text = translate.translate_text(joke)
            else:
                translated_text = joke


            speech_the_text(translated_text, tts_model)



    # command write - writes to the active window, everything that the
user says
    elif cmd == 'write':
        write_process          =          Process(target=listen_write,
args=[assistant_stt[current_settings['ASSISTANT_STT']]['model']])
        write_process.start()


        answer = "Диктуйте"
        if speak_the_answer == "False":
            if cur_speach_lang != 'ua':
                translate = TranslatorModel('uk', cur_speach_lang)
                translated_text = translate.translate_text(answer)
            else:
                translated_text = answer
            time.sleep(5)
            show_notification("Голосовий помічник",translated_text)
        else:
            if cur_speach_lang != 'ua':
                translate = TranslatorModel('uk', cur_speach_lang)
                translated_text = translate.translate_text(answer)
            else:
                translated_text = answer
```

```
            time.sleep(6)
            speech_the_text(translated_text, tts_model)



    # command find - opens a browser window with a user request
    elif cmd == 'find':
        va_request = make_req_from_string(voice, key_world)
        url_request = "https://www.google.com/search?q=" +
'+'.join(va_request.split())
        webbrowser.open(url_request)
        show_notification("Голосовий помічник","Відкрито браузер з
запитом '"+va_request+"'")



    # the say command - either says what the user said \ or notifies
that the voice is not turned on
    elif cmd == 'say':
        if speak_the_answer == "False":
            show_notification("Голосовий помічник","Увімкніть
озвучування відповідей голосового помічника")


        else:
            speech_the_text(make_req_from_string(voice, key_world),
tts_model)



    # makes a request to the GPT chat
    elif cmd == 'gpt':
        va_request = make_req_from_string(voice, key_world)
        show_notification("Голосовий помічник","Очікуйте, звертаюсь до
ChatGPT, це може зайняти декілька хвилин")
        result = asyncio.run(gpt_req(va_request))
        result_without_numbers = convert_numbers_in_text(result)
```

```
        if len(result_without_numbers) < 500:
            if speak_the_answer == "False":
                if len(result_without_numbers) < 100:
                    show_notification("Голосовий помічник", result)
                else:
                    paint_gpt_answer(result)
            else:
                paint_gpt_answer(result)
                translate = TranslatorModel('en', cur_speach_lang)
                translated_text                                          =
translate.translate_text(result_without_numbers)
                speech_the_text(translated_text, tts_model)


        else:
            paint_gpt_answer(result)



    # translate command
    elif cmd == 'translate':
        va_request = make_req_from_string(voice, key_world)
        translate = TranslatorModel(cur_speach_lang, cur_tra_to_lang)
        translated_text = translate.translate_text(va_request)
        result = va_request +f'\u000a\u000a'+translated_text
        paint_gpt_answer(result)



    # custom command
    else:
        if assistant_cmd_list[cmd]['commandType'] == 'explorer':

execute_custom_command_exe(assistant_cmd_list[cmd]['customCommand'],
'path')
        elif assistant_cmd_list[cmd]['commandType'] == 'execute':
```

```
execute_custom_command_exe(assistant_cmd_list[cmd]['customCommand'],
'file')
        elif assistant_cmd_list[cmd]['commandType'] == 'openWebPage':

webbrowser.open(f'{assistant_cmd_list[cmd]["customCommand"]}')
        else:
            show_notification("Голосовий помічник", "Я Вас не зрозумів")
```

4.8 Розробка модулю керування голосовим асистентом

За для зручності керування голосовим асистентом було розроблено модуль керування, для нього було описано клас `TrayMenu`, який використовує наступні бібліотеки `pystray` [56], `PIL` [57], `multiprocessing`, `time` , та наступні методи `__init__`, `start_stopVA`, `changeState`, `openSettings`, `on_quit`. Код модулю керування відображено нижче:

```
class TrayMenu:
    # init object
    def __init__(self):
        self.VAstate = "Stop"
        self.image = Image.open("src/images/icon.png")
        self.menu = Menu(MenuItem('Settings', self.openSettings),
                         MenuItem('Start \ Stop', self.start_stopVA),
                         MenuItem('Quit', self.on_quit))
        self.icon = Icon("name", self.image, menu=self.menu)
        self.server_process = None
        self.server_stopper = None
        self.command_process = None
        show_notification("Голосовий помічник","Увімкніть
прослуховування голосу")

    # start\stop action handler
    def start_stopVA(self):
        self.changeState()
        if self.VAstate == "Start":
            self.command_process =
Process(target=run_command_processor)
            self.command_process.start()
        else:
            self.command_process.terminate()
            self.command_process.join()
            show_notification("Голосовий помічник","Ваш голос більше
не слухається")
```

```
    #change state method
    def changeState(self):
        if self.VAstate == "Start":
            self.VAstate = "Stop"
        else:
            self.VAstate = "Start"

    # open setting action handler
    def openSettings(self):
        if self.server_process == None and self.server_stopper ==
None:
            self.server_process = Process(target=run_server)
            self.server_process.start()
            server_pid = self.server_process.pid
            self.server_stopper = Process(target=stop_server, kwargs =
{"server_process":server_pid})
            self.server_stopper.start()
            return 1

        if self.server_process.is_alive() and
self.server_stopper.is_alive():
            open_browser()
            return 1
        else:
            self.server_process = None
            self.server_stopper = None
            self.openSettings()

    # quit action handler
    def on_quit(self, icon):
        if self.server_process != None:
            if self.server_process.is_alive():
                self.server_process.terminate()
                self.server_process.join()
        if self.server_stopper != None:
            if self.server_stopper.is_alive():
                self.server_stopper.terminate()
                self.server_stopper.join()
        if self.command_process != None:
            if self.command_process.is_alive():
                self.command_process.terminate()
                self.command_process.join()

        icon.stop()


    # run icon
    def run(self):
        self.icon.run()
```

4.9 Демонстрація роботи голосового помічника

Перше, що необхідно зробити користувачу – запустити голосового помічника, як він буде готовий до роботи, то голосовий помічник має сповістити користувача про це. Сповіщення відображено на рисунку 4.12.

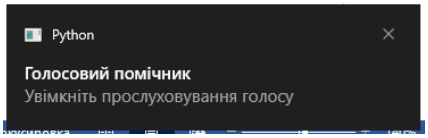

Рисунок 4.12 – Сповіщення про готовність до роботи. *Джерело: побудовано автором.*

Після запуску голосового помічника з'являється іконка в системному треї, після натискання на неї відображається меню керування голосовим помічником, що відображено на рисунку 4.13.

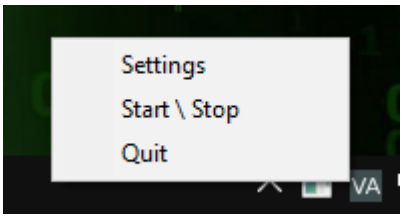

Рисунок 4.13 – Керування голосовим помічником. *Джерело: побудовано автором.*

Для запуску налаштувань необхідно натиснути на кнопку «Settings» (рис. 4.14), а для запуску самого голосового асистента та виконання команд, треба натиснути «Start \ Stop». Після натискання на кнопку «Start \ Stop» голосовий помічник сповістить про його готовність, це відображено на рисунку 4.15.

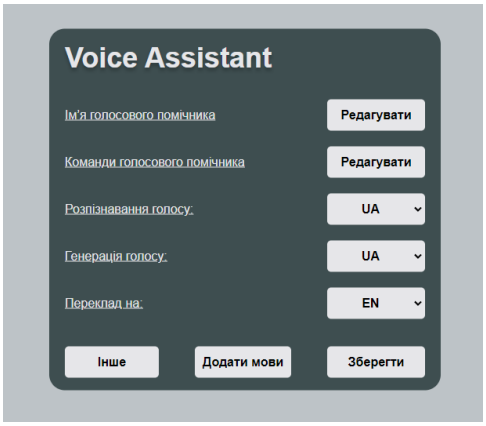

Рисунок 4.14 – Стартова сторінка модулю налаштувань. *Джерело: побудовано автором.*

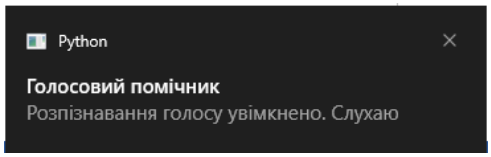

Рисунок 4.15 – Сповіщення про початок роботи та розпізнавання команд. *Джерело: побудовано автором.*

Деякі команди мають графічне відображення результатів, як приклад «gpt», ця команда виконує запит до нейромережі та відображає користувачеві результати у текстовому форматі. Результат виконання цієї команд изображено на рисунку 4.16.

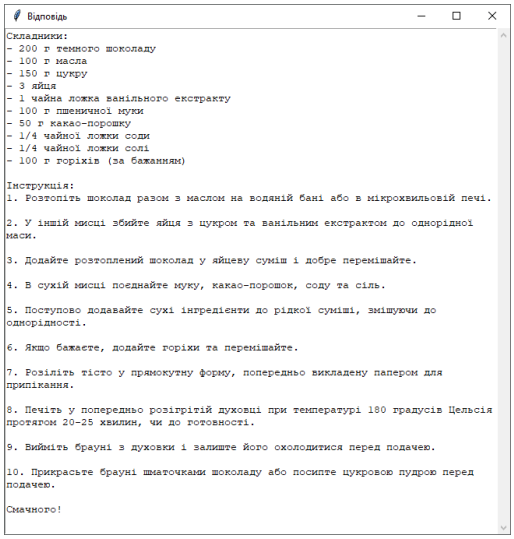

Рисунок 4.16 – Результат виконання команди «gpt». *Джерело: побудовано автором.*

Детальна робота голосового помічника представлена на відео-демонстрації. Посилання на відео-демонстрацію наведено в пункті В.2 додатка В.

4.10 Тестування

Таблиця 4.1 – Тестування голосового помічника. *Джерело: побудовано автором*

#	Summary	Precondition	Steps	Expected result	0\1
1	Запуск додатку	-	1. Запустити додаток голосового помічника	Відображається повідомлення про готовність голосового помічника	1

Продовження таблиці 4.1

2	Успішне відкриття налаштувань	Додаток голосового помічника увімкнено	1. Натиснути правою клавішею миші на іконку в системному треї 2. Натиснути на кнопку «Settings»	Відкривається домашня сторінка налаштувань в браузері	1
3	Успішне увімкнення розпізнавання команд	Додаток голосового помічника увімкнено.	1. Натиснути правою клавішею миші на іконку в системному треї 2. Натиснути на кнопку «Start \ Stop»	Відображається повідомлення про активне прослуховування голосу	1
4	Успішне виконання команди «help»	Розпізнавання команди увімкнено.	1. Натиснути правою клавішею миші на іконку в системному треї 2. Натиснути на кнопку «Start \ Stop» 3. Сказати «Ассистент допомога»	Команда «help» розпізнана. Відкрито сторінку допомоги в браузері.	1

Продовження таблиці 4.1

5	Успішне виконання команди «time»	Розпізнавання команди увімкнено.	1. Натиснути правою клавішею миші на іконку в системному треї 2. Натиснути на кнопку «Start\ Stop» 3. Сказати «Ассистент час»	Звукова інформація точного часу програється користувачеві	1
6	Успішне виконання команди «browser»	Розпізнавання команди увімкнено.	1. Натиснути правою клавішею миші на іконку в системному треї 2. Натиснути на кнопку «Start\ Stop» 3. Сказати «Ассистент браузер»	Браузер за замовчуванням відкривається	1

Продовження таблиці 4.1

7	Успішне виконання команди «joke»	Розпізнавання команди увімкнено.	1. Натиснути правою клавішею миші на іконку в системному треї 2. Натиснути на кнопку «Start \ Stop» 3. Сказати «Ассистент жарт»	Жарт програється користувачеві	1
8	Успішне виконання команди «write»	Розпізнавання команди увімкнено.	1. Натиснути правою клавішею миші на іконку в системному треї 2. Натиснути на кнопку «Start \ Stop» 3. Сказати «Ассистент диктую»	Відкривається віконце з кнопкою «Стоп», можливість до написання тексту голосом користувача активується	1
9	Успішне виконання команди «find»	Розпізнавання команди увімкнено.	1. Натиснути на іконку в системному треї 2. Натиснути на кнопку «Start \ Stop» 3. Сказати «Ассистент знайди …»	Відкривається браузер з запитом користувача у гугл в браузері за замовчуванням	1

Продовження таблиці 4.1

10	Успішне виконання команди «say»	Розпізнавання команди увімкнено.	1. Натиснути на іконку в системному треї 2. Натиснути на кнопку «Start \ Stop» 3. Сказати «Ассистент скажи …»	Програється звуком те, що сказав користувач у запиті	1
11	Успішне виконання команди «gpt»	Розпізнавання команди увімкнено.	1. Натиснути на іконку в системному треї 2. Натиснути на кнопку «Start \ Stop» 3. Сказати «Ассистент нейромережа …»	Відкривається віконце з результатом запиту до нейромережі ChatGPT	1
12	Успішне виконання команди «translate»	Розпізнавання команди увімкнено.	1. Натиснути на іконку в системному треї 2. Натиснути на кнопку «Start \ Stop» 3. Сказати «Ассистент переклади»	Перекладений текст відображається користувачеві у віконці.	1

ВИСНОВОК

Під час написання магістерської роботи було попередньо спроектовано віртуального помічника на основі штучного інтелекту. Проведено аналіз існуючих інформаційних технологій та обрано оптимальний функціонал:

- розпізнавання людського голосу,
- генерувати мовну відповідь (відповідати голосом),
- перекладати текст\голос на іншу мову,
- виконувати команди користувача,
- виконувати запити до нейромережі,
- відкривати файли\теки в системи,
- виконувати запуск різноманітних програм у ОС,
- мати зручний інтефейс
- мати можливість до налаштувань
- бути кросплатформним

Створюваний голосовий помічник може бути використано як у побуті для виконання рутинних задач, так і у робочому середовищі для спрощення робочих процесів .

В процесі дослідження предметної області було виявлено необхідність розробки легкого в користуванні та налаштування, анонімного голосового асистента з використанням ШІ, оскільки віртуальні голосові помічники, що базуються на штучному інтелекті, стають цілком важливою складовою для забезпечення ефективності та зручності користування інформаційними ресурсами, а якщо брати до уваги, що особисті дані користувача не відправляються на сервіси обробки інформації, то розробка такого асистента більш ніж доцільна.

Наступним кроком були встановлені цілі дипломного проекту, визначення функціоналу, вимоги і сформульоване завдання для реалізації, з погляду на це було виконано планування робочих процесів та визначення термінів виконання проекту.

У процесі розробки інформаційної технології було проведено детальне проектування робочих процесів та сценаріїв використання. Для цього були ство-рені

IDEF0 діаграми, які дозволили декомпозувати загальні процеси на більш деталізовані. Use Case діаграми проілюстрували різноманітні сценарії використання системи. Такий підхід дозволив точно визначити функціонал інформаційної технології та розбити її на менші компоненти для зручності в реалізації.

Результатом проведеної роботи є повноцінно розроблений голосовий помічник на основі штучного інтелекту та включає в себе додаткові можливості. Розроблений голосовий помічник володіє розширеним функціоналом, що включає в себе не лише базові завдання, але й додаткові функції для поліпшення користувальницького досвіду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 ChatGPT: Jack of all trades, master of none [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S156625352300177X

2 What is Google Bard? Everything you need to know about ChatGPT rival [Електронний ресурс] – Режим доступу до ресурсу: https://www.tomsguide.com/news/google-bard-ai

3 Google тестує власний чат-бот на базі штучного інтелекту Bard. [Електронний ресурс] – Режим доступу до ресурсу: https://forbes.ua/innovations/gonka-shtuchnikh-intelektiv-google-vikotiv-analog-chatgpt-bard-chim-vin-vidriznyaetsya-vid-svogo-vbivtsi-07022023-11572

4 Штучний інтелект способи використання в сфері телекомунікацій [Електронний ресурс] – Режим доступу до ресурсу: https://archive.logos-science.com/index.php/conference-proceedings/article/view/909

5 Основні тренди запровадження сучасних цифрових технологій у сфері маркетингу [Електронний ресурс] – Режим доступу до ресурсу: https://financial.lnu.edu.ua/wp-content/uploads/2020/06/2020_VIKL._ZBIRNIK.pdf#page=144

6 Дослідження способів застосування штучного інтелекту в ігровій індуст-рії [Електронний ресурс] – Режим доступу до ресурсу: http://195.230.140.114/jspui/bitstream/123456789/13503/1/%D0%97%D0%B1%D1%96%D1%80%D0%BD%D0%B8%D0%BA_%D1%82%D0%B5%D0%B7_2023%281%29.pdf#page=123

7 Сучасні ризик-орієнтовані веб-продукти [Електронний ресурс] – Режим доступу до ресурсу: http://biblio.umsf.dp.ua/jspui/bitstream/123456789/4608/1/%D0%9A%D0%BE%D0%BD%D1%84%2005_11_21%20%D0%A2%D0%BE%D0%BC%202%20.pdf#page=505\

8 Способи використання нейронних мереж та машинного навчання в комп'ютерних іграх [Електронний ресурс] – Режим доступу до ресурсу: http://journals.khnu.km.ua/vestnik/wp-content/uploads/2021/11/vknu-ts-2021-n2-295.pdf#page=97

9 Дослідження ролі етики ШІ у формуванні відповідального розвитку штучного інтелекту [Електронний ресурс] – Режим доступу до ресурсу: https://ts2.space/uk/%D1%8F%D0%BA-%D0%B5%D1%82%D0%B8%D0%BA%D0%B0-ai-%D1%81%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B2%D1%83%D1%94-%D0%B2%D1%96%D0%B4%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9-%D1%80%D0%BE/#gsc.tab=0

10 The internet of things, big data, and intelligence in smart city [Електронний ресурс] – Режим доступу до ресурсу: https://elartu.tntu.edu.ua/bitstream/lib/30405/2/IMST_2019_Zabihailo_O-The_internet_of_things_Big_45.pdf

11 What is Amazon Alexa, and what can it do? [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitaltrends.com/home/what-is-amazons-alexa-and-what-can-it-do/

12 Amazon launches brand new Alexa app [Електронний ресурс] – Режим доступу до ресурсу: https://www.chargedretail.co.uk/2020/07/28/amazon-launches-brand-new-alexa-app/

13 Google Assistant is built to keep your information private, safe, and secure. [Електронний ресурс] – Режим доступу до ресурсу: https://safety.google/intl/en_us/assistant/

14 Google Assistant Unveils New 'Capabilities' and 'Widgets' for Developers at Google I/O [Електронний ресурс] – Режим доступу до ресурсу: https://voicebot.ai/2021/05/19/google-assistant-unveils-new-capabilities-and-widgets-for-developers-at-google-i-o/

15 Google's Have Now Virtual Assistant To Deliver More Personalized Results [Електронний ресурс] – Режим доступу до ресурсу: https://www.webvisitors.co.in/googles-virtual-assistant-personalized-result/

16 What is Siri and how does Siri work? [Електронний ресурс] – Режим доступу до ресурсу: https://www.pocket-lint.com/apps/news/apple/112346-what-is-siri-apple-s-personal-voice-assistant-explained/

17 iOS 12: Siri agora pode ligar sua lanterna [Електронний ресурс] – Режим доступу до ресурсу: https://www.iera.pt/ios-12-siri-can-now-turn-on-your-flashlight

18 What is Microsoft Cortana? [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/searchenterprisedesktop/definition/Cortana

19 Взаємодія з фоновою програмою в Кортані [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/ru-ru/windows/apps/design/input/cortana-interact-with-a-background-app

20 Bixby 101: Get to Know the Ins and Outs of Samsung's Intelligent Interface [Електронний ресурс] – Режим доступу до ресурсу: https://news.samsung.com/za/bixby-101-get-to-know-the-ins-and-outs-of-samsungs-intelligent-interface

21 Get to Know the Ins and Outs of Samsung's Intelligent Interface [Елект-ронний ресурс] – Режим доступу до ресурсу: https://news.samsung.com/global/bixby-101-get-to-know-the-ins-and-outs-of-samsungs-intelligent-interface

22 Методи і техніка досліджень [Електронний ресурс] – Режим доступу до ресурсу: https://elib.tsatu.edu.ua/dep/mtf/ophv_10/page3.html

23 Мова UML. Діаграма використання [Електронний ресурс] – Режим доступу до ресурсу: http://p4ilka.blogspot.com/2018/12/uml.html.

24 What Is Python Used For? A Beginner's Guide [Електронний ресурс] – Режим доступу до ресурсу: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

25 Library for performing speech recognition [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/SpeechRecognition/

26 HTML Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/html/html_intro.asp#:~:text=HTML%20stands%20for%20Hyper%20Text,how%20to%20display%20the%20content

27 CSS: Cascading Style Sheets [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/CSS

28 How To Make a Web Application Using Flask in Python 3 [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3

29 What is JSON? [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/whatis/whatis_json.asp

30 Створення схем IDEF0 [Електронний ресурс] – Режим доступу до ресурсу: https://support.microsoft.com/uk-ua/office/%D1%81%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F-%D1%81%D1%85%D0%B5%D0%BC-idef0-ea7a9289-96e0-4df8-bb26-a62ea86417fc

31 UML Use Case Diagram Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/pages/uml-use-case-diagram

32 What is a High Level Design (HLD)? [Електронний ресурс] – Режим доступу до ресурсу: https://ipwithease.com/what-is-a-high-level-design-hld/.

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

34 Code editing. Redefined. [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/

35 Python 3.10.9 documentation / JSON encoder and decoder [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/json.html

36 What is Git: Features, Command and Workflow in Git [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/git-tutorial/what-is-git

37 Python 3.10.9 documentation / Process-based parallelism [Електронний ресурс] – Режим доступу до ресурсу:

https://docs.python.org/3/library/multiprocessing.html?highlight=process#module-multiprocessing

38 Python 3.10.9 documentation / Convenient web-browser controller [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/webbrowser.html?highlight=web#module-webbrowser

39 Python 3.10.9 documentation / Regular expression operations [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/re.html?highlight=re#module-re

40 Python 3.10.9 documentation / Miscellaneous operating system interfaces [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/os.html?highlight=os#module-os

41 Python 3.10.9 documentation / Time access and conversions [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/time.html?highlight=time#module-time

42 Python 3.10.9 documentation / System-specific parameters and functions [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/sys.html?highlight=sys#module-sys

43 Python 3.10.9 documentation / Set handlers for asynchronous events [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/signal.html?highlight=sig#module-signal

44 Python 3.10.9 documentation / Logging facility for Python [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/logging.html?highlight=loggin#module-logging

45 Python 3.10.9 documentation / Offline open source speech recognition API based on Kaldi and Vosk [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/vosk/

46 Python 3.10.9 documentation / Play and Record Sound with Python [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/sounddevice/

47 Python 3.10.9 documentation / A synchronized queue class [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/queue.html?highlight=queue#module-queue

48 Python 3.10.9 documentation / Fuzzy string matching in python [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/fuzzywuzzy/

49 Python 3.10.9 documentation / Tensors and Dynamic neural networks in Python with strong GPU acceleration [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/torch/

50 Python 3.10.9 documentation / Silero Models: pre-trained enterprise-grade STT / TTS models and benchmarks. [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/silero/

51 Python 3.10.9 documentation / Fundamental package for array computing in Python [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/numpy/

52 Python 3.10.9 documentation / Whole Architecture Verification [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/Wave/

53 Python 3.10.9 documentation / Translate text using google translate [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/translator/

54 Python 3.10.9 documentation / freeGPT provides free access to text and image generation models. [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/freeGPT/

55 Python 3.10.9 documentation / Asynchronous I/O [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/asyncio.html?highlight=asyncio

56 Python 3.10.9 documentation / Provides systray integration [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/pystray/

57 Python 3.10.9 documentation / Python Imaging Library (Fork) [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/Pillow/

ДОДАТОК А

А.1. Планування робіт.

У сучасному світі голосові помічники стрімко розвиваються і становляться невід'ємною частиною життя і багато людей дедалі частіше починають використовувати їх частіше, бо прагнуть автоматизувати рутинні процеси, а не витрачати час на рутину.

Віртуальний голосовий помічник є простим інструментом для швидкого і безпечної оптимізації часу користувача, та відіграє важливу роль в автоматизації процесу користування комп'ютером

А.2. Деталізація мети методом SMART.

Чітке визначення цілей на етапі концептуального проектування відіграє важливу роль у забезпеченні ефективного та високоякісного виконання проекту. Використання методу SMART для розгортання деталей цілей проекту дозволяє структурувати та чітко визначити необхідні параметри, сприяючи більш ефективному плануванню та виконанню завдань. Результати цього процесу представлені у таблиці А.1.

Таблиця А.1 – Деталізація мети проекту методом SMART

Specific(Конкретна)	Розробити віртуального голосового помічника на основі штучного інтелекту
Measurable(Вимірювана)	Результат – створений віртуальний голосовий помічник на основі штучного інтелекту
Achievable(Досяжна)	Досягнення мети проекту неможливе без знань Python, , html, css, Flask, зовнішніх бібліотек torch, vosk, Process, silero, та навичок створення документації.
Relevant(Реалістична)	Імплементуваний віртуальний голосовий помічник доволить швидко та безпечно оптимізувати час користувача, та відіграє важливу роль в автоматизації процесу користування комп'ютером

Продовження таблиці А.1 – Деталізація мети проекту методом SMART

Time-framed(Обмежена у часі)	Термін досягнення мети проекту визначено з замовником і дорівнює 3 місяці.

А.3. Планування змісту робіт.

Work Breakdown Structure (WBS) є інструментом розподілу проекту на управляючі компоненти, сприяючи керівникам проектів і командам у визначенні та структуруванні обсягу робіт. Основна мета розробки WBS полягає у створенні чіткої, деталізованої та ієрархічної структури. WBS організує роботу у формі ієрархії, де верхній рівень представляє кінцевий продукт, а кожен наступний, менший рівень включає більш деталізоване визначення робочих завдань. Процес декомпозиції завдань триває досягнення оптимального розміру для ефективного управління та контролю, але при цьому забезпечує практичне значення. На рисунку А.1 наведено приклад WBS для проекту віртуального голосового асистента на основі штучного інтелекту.

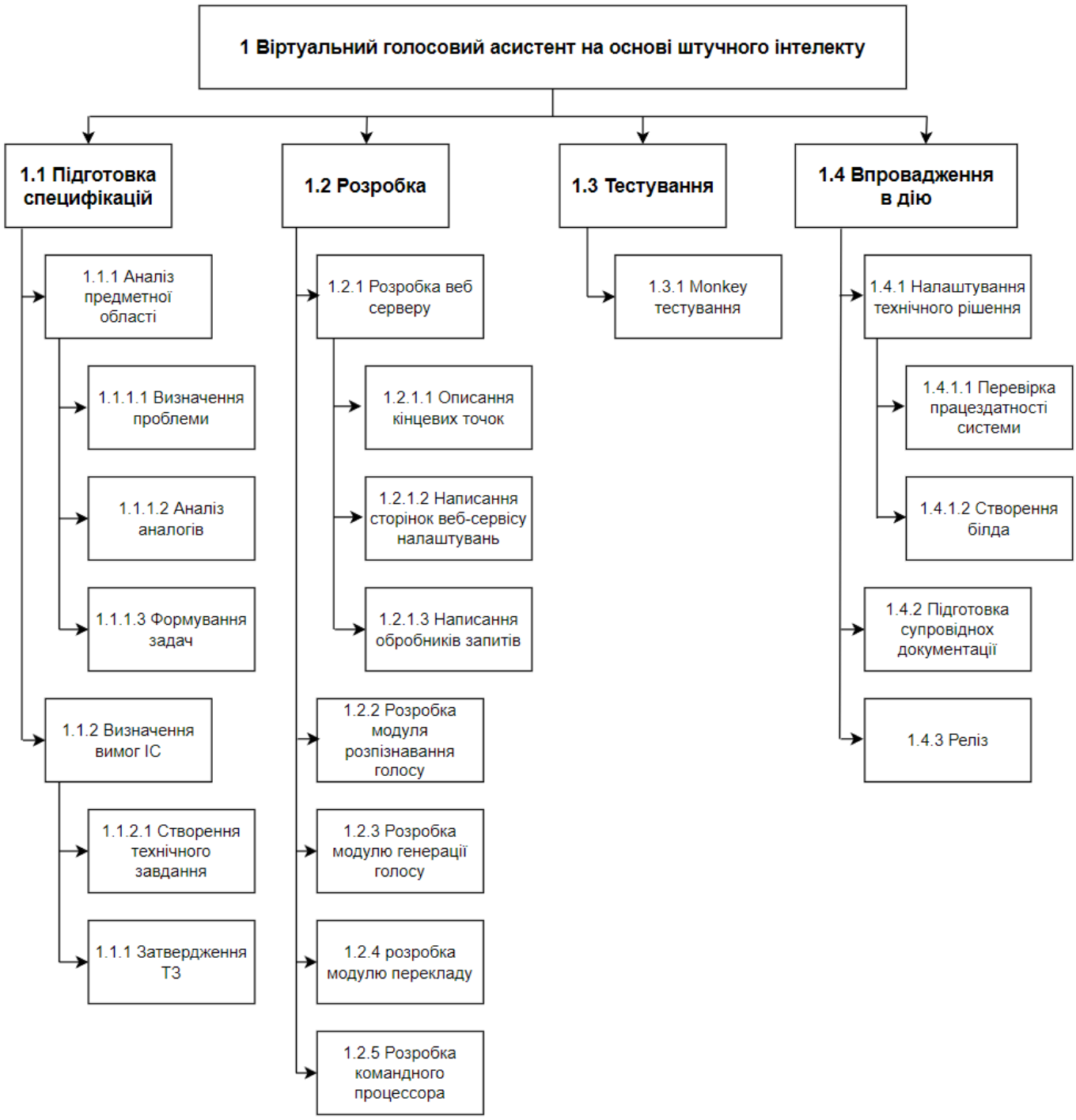

Рисунок А.1 – WBS-структура робіт проекту

А.4. Планування структури виконавців.

Створення організаційної структури виконавців (OBS) представляє наступний етап після декомпозиції процесів за допомогою WBS. OBS, що означає Organization Breakdown Structure, є системою, яка використовується для відображення ієрархії команди виконавців проекту та розкриває, як організована команда та як розподіляються ресурси відповідно до цілей проекту. У рисунку А.2 зображено

організаційну структуру планування робіт проекту, а дані про учасників проекту представлені в таблиці А.2.

Таблиця А.2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Радченко О.С	Розробка WEB-серверу налаштувань, сторінок, обробників запитів, кінцевих точок. Розробка модуля розпізнавання голосу, модуля модулю генерації голосу, розробка модуля перекладу, розробка командного процесору.
Тестувальник	Радченко О.С	Проведення тестування програмного продукту
Проектувальник	Радченко О.С	Проектування структури програмного продукту
Керівник проекту	Чибіряк Я.І.	Створення завдання на розробку проекту, рецензія імплементованого програмного продукту

Продовження таблиці А.2.

Менеджер	Радченко О.С.	Контроль над дотриманням дедлайнів, розподілення ресурсів, задач на розробку

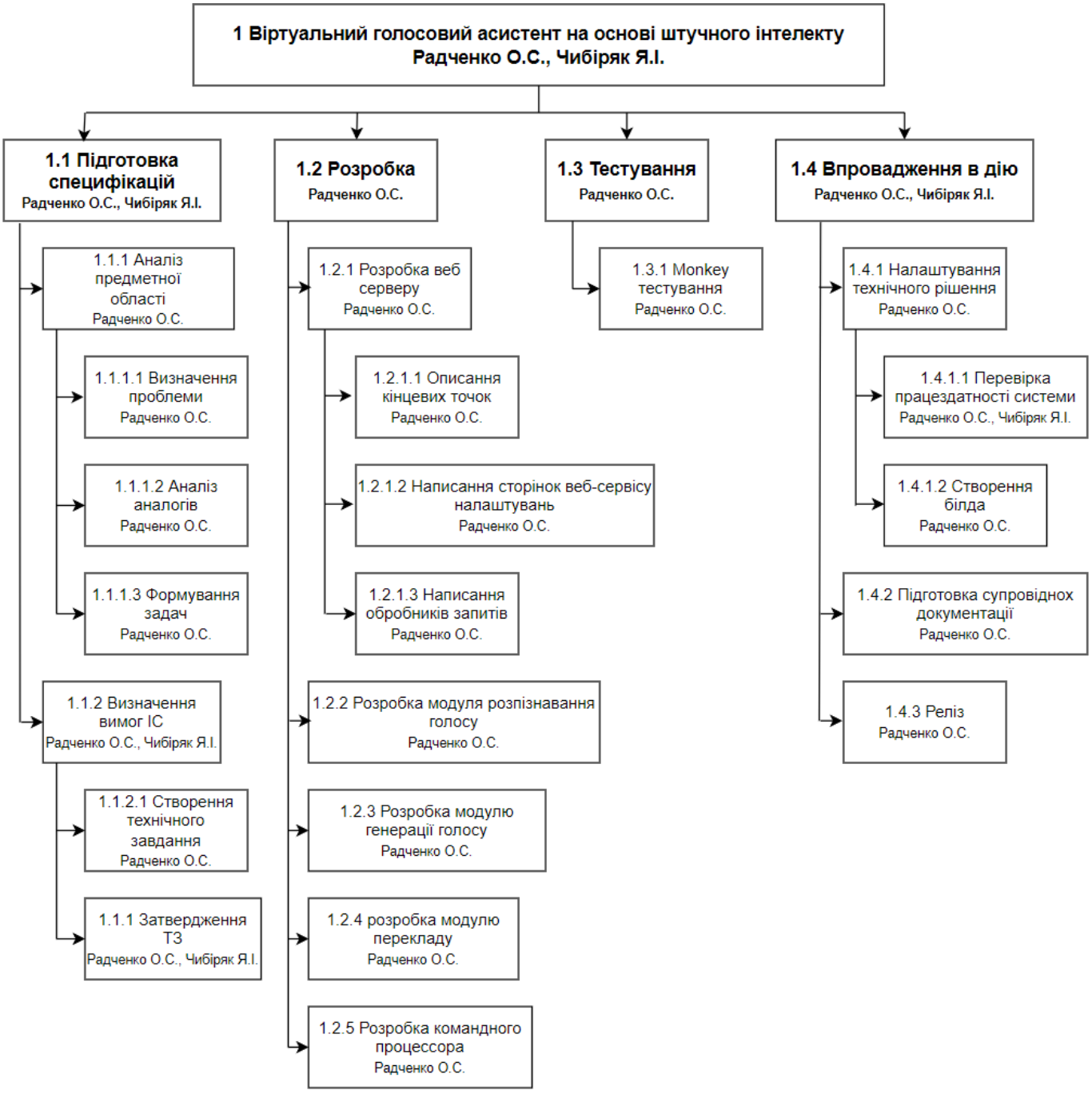

Рисунок А.2 – OBS-структура робіт проекту

А.5. Діаграма Ганта.

Розробка календарного плану робіт є ключовою складовою управління проектом. Цей план представляє собою діаграму з розподілом часових рамок для кожного завдання, що сприяє точному визначенню загальної тривалості проекту з урахуванням доступних ресурсів. Календарний графік проекту представлено на рисунку А.3.

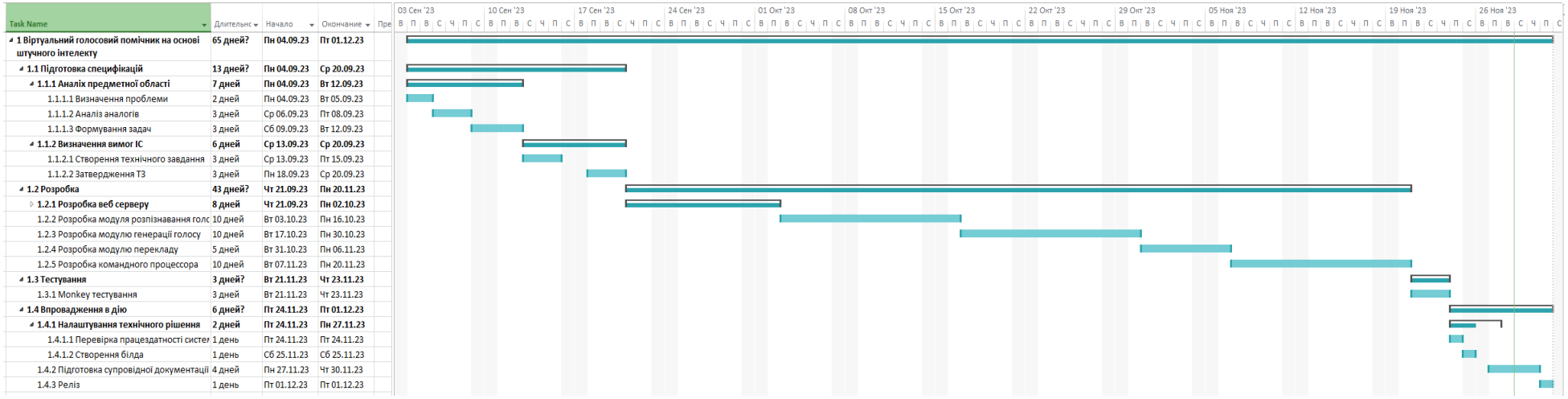

Рисунок А.3 – Календарний графік проекту

А.6. Управління ризиками проекту.

На етапі планування проектних робіт особлива увага приділяється аналізу та управлінню ризиками. Цей процес включає кілька кроків, таких як ідентифікація ризиків, оцінка їх впливу та ймовірності, розробка стратегій для зменшення або усунення їх впливу, а також моніторинг та контроль за використанням цих стратегій. У таблиці А.3 перераховано ризики даного проекту, а їх оцінки подані у таблиці А.4. Таблиця А.5 вказує шкалу ризиків в залежності від їх типу, розміру впливу та ймовірності.

Таблиця А.3 – Ризики проекту.

№ ризику	Назва (опис) ризику
1	Відсутність кваліфікованих знань
2	Технічні збої у обладнанні
3	Проблеми з інтернетом
4	Проблема з електропостачанням
5	Відсутність достатнього фінансування проекту
6	Захворювання одного з членів команди
7	Недостатня якість вихідного коду
8	Поява альтернативного продукту
9	Недостатній контроль якості
10	Зміна вимог замовника

Таблиця А.4 – Результати визначення ймовірності, впливу та рангу ризику проекту.

№ ризику	Назва (опис) ризику	Ймовірність (0,1-0,9)	Вплив (0,05-0,8)	Ранг
1	Відсутність кваліфікованих знань	0,2	0,4	0,08
2	Технічні збої у обладнанні	0,1	0,8	0,08
3	Проблеми з інтернетом	0,4	0,05	0,02
4	Проблема з електропостачанням	0,1	0,8	0,08
5	Відсутність достатнього фінансування проекту	0,1	0,1	0,01
6	Захворювання одного з членів команди	0,1	0,1	0,01
7	Недостатня якість вихідного коду	0,75	0,3	0,225
8	Поява альтернативного продукту	0,7	0,1	0,07
9	Недостатній контроль якості	0,34	0,2	0,068
10	Зміна вимог замовника	0,2	0,8	0,16

Таблиця А.5 – Шкала оцінювання ризика типом, ймовірністю та величиною впливу.

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Для зменшення негативного впливу потенційних загроз на проект є критично важливим розробити докладний план реагування на ризики. Цей план повинен включати аналіз та оцінку ефективності запропонованих стратегій зниження ризиків, з урахуванням можливих наслідків для проекту. Оцінка ґрунтується на критеріях, визначених у таблиці А.3. Внаслідок створення плану реагування була сформована матриця, яка відображає ймовірність та вплив різних ризиків, описаних у таблиці А.6. У цій матриці застосовано різні кольори для класифікації ризиків: зелений позначає прийнятні, жовтий - ті, що можна виправдати, а червоний - недопустимі ризики.

Таблиця А.6 – Матриця ймовірності та впливу

Ймовірність виникнення ризику	Вплив ризику				
	0,05	0,1	0,2	0,4	0,8
0,9	0,045	0,09	0,18	0,3	0,72
0,7	0,035	0,07 R8	0,14	0,28	0,56
0,5	0,025 R3	0,05	0,1	0,2	0,4
0,3	0,015	0,03	0,06 R8	0,12 R1, R7	0,24
0,1	0,005 R5, R6	0,01	0,02	0,04	0,08 R2, R4, R10

В таблиці А.4 описано класифікацію ризиків за їх рівнем, відповідно до їх індексних значень. Детальний опис ризиків та стратегій їх управління описано у таблиці А.7.

Таблиця А.7 – Шкала оцінювання за рівнем ризику.

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	$0{,}005 \leq R \leq 0{,}05$	3, 5, 6
2	Виправдані	$0{,}05 \leq R \leq 0{,}14$	1, 2, 4, 7, 8, 9, 10
3	Недопустимі	$0{,}14 \leq R \leq 0{,}72$	

Вирішення проблеми ризиків та стратегії їх регулювання відображено на табличці А.8

Таблиця А.8 – Ризики та стратегії реагування

Id	Статус	Опис	Ймовірність	Вплив	Ранг	План 1	Стратегія	План 2
RS-1	Open	Відсутність кваліфікованих знань	Low	Mid	0,08	Вивчити необхідні технології	Зменшення	Підвищення кваліфікації
RS-2	Open	Технічний збій у обладнанні	Low	High	0,08	Обслуговування обладнання	Зменшення	Використання іншого обладнання
RS-3	Open	Проблеми з інтернетом	Mid	Low	0,02	Декілька точок підключення	Зменшення	Скористатися послугою Коворкінгу
RS-4	Open	Проблема з електропостачанням	Low	High	0,08	Використовувати накопичувачі електроенергії	Зменшення	Скористатися послугою Коворкінгу
RS-5	Open	Відсутність достатнього фінансування проекту	Low	Low	0,01	Підписання контракту з замовником щодо своєчасного та повноцінного фінансування послуг	Прийняття	Мати резервне фінансування

Продовження таблиці А.8

RS-6	Open	Захворювання одного з членів команди	Low	Low	0,01	Продовжити працювати	Прийняття	Зробити перерву в 1-2 дні для одужання
RS-7	Open	Недостатня якість вихідного коду	High	Mid	0,225	Хмарне зберігання даних	Зменшення	Локальні сховища
RS-8	New	Поява альтернативного продукту	High	Low	0,07	Розробка нового функціоналу	Прийняття	
RS-9	Open	Недостатній контроль якості	Mid	Low	0,068	Найм тестувальника	Зменшення	Впровадження системи управління якістю
RS-10	Open	Зміна вимог замовника	Low	High	0,16	Підписання ТЗ	Прийняття	

ДОДАТОК Б

Б.1. Лістинг коду модуля Web-серверу

```
from SettingsManager import SettingsManager, settings_manager
from flask import Flask, send_file, render_template, url_for, request
from multiprocessing import Process
from default_config_assistant import last_request_time
import json
import webbrowser
import re
import os
import time
import sys
import signal
import logging



# configs of VA assistant
=========================================================================
=======================
settings_manager.load_settings()
current_settings = settings_manager.get_setting('CURRENT_SETTINGS',
{})
assistant_cmd_list =
settings_manager.get_setting('ASSISTANT_CMD_LIST', {})
assistant_alias = settings_manager.get_setting('ASSISTANT_ALIAS', {})
assistant_stt = settings_manager.get_setting('ASSISTANT_STT', {})
assistant_tts = settings_manager.get_setting('ASSISTANT_TTS', {})
assistant_tra = settings_manager.get_setting('ASSISTANT_TRA', {})



# config for local web server
=========================================================================
====================
web_config = ''
with open('web_config.json', 'r') as file:
    web_config = json.load(file)


last_request_time = time.time()
```

```
# web Server
app = Flask(__name__, template_folder='web')
log = logging.getLogger('werkzeug')
log.setLevel(logging.ERROR)

# functions
================================================================================
====================================

# run server
def run_server():
    open_browser()
    app.run(host=web_config['host'], port=web_config['port'],
threaded=True, processes=1, use_reloader=False, debug=False)


# changing the last_request_time
def change_last_request_time():
    current_settings['LAST_REQUEST'] = time.time()
    settings_manager.set_setting('CURRENT_SETTINGS', current_settings)
    settings_manager.save_settings()

# getting the last_request_time
def get_last_request_time():
    settings_manager.load_settings()
    return settings_manager.get_setting('CURRENT_SETTINGS',
{})['LAST_REQUEST']

# stopping web server
def stop_server(**kwargs):
    server_inactive_live = web_config['server_inactive_live']
    server_last_answer = web_config['server_last_answer']
    server_process = kwargs.get("server_process")

    if server_inactive_live <= server_last_answer:
        server_inactive_live = 300
        server_last_answer = 100

    change_last_request_time()
    while True:
        try:
            elapsed_time = time.time() - get_last_request_time()
            if elapsed_time > server_inactive_live:
                os.kill(server_process, signal.SIGTERM)
                print("Server Stopped")
                break
```

```
            time.sleep(server_last_answer)
        except(KeyboardInterrupt):
            sys.exit(0)


# check string on latin symbols only
def is_latin_only(input_string:str):
    return bool(re.match("^[a-zA-Z]+$", input_string))


# check string on the correct link path
def if_link(input_string:str):
    url_pattern =
re.compile(r'(https?://|ftp://|file://|http://localhost:|http://\d{1,3
}\.\d{1,3}\.\d{1,3}\.\d{1,3}(:\d+)?/)(?:www\.)?[a-zA-Z0-9-]+(?:\.[a-
zA-Z]{2,})+(?::\d+)?(?:/[^\s]*)?')
    return bool(url_pattern.match(input_string))


# check string on the folder path
def if_folder_path(input_string:str):
    return bool(input_string and input_string[1:3] == ":\\" and not
input_string.endswith('\\') and '.' not in input_string)


# check string on the executable file path
def if_executable(input_string:str):
    return bool(
        input_string and
        input_string[1:3] == ":\\" and
        not input_string.endswith('\\') and
        '.' in input_string and
        input_string.lower().endswith(('.exe', '.bat', '.cmd'))
    )


# open browser with home page
def open_browser(destination:str = ''):

webbrowser.open('http://'+str(web_config['host'])+':'+str(web_config['
port'])+'/'+destination)


# endpoints
================================================================
====================================
```

```
# home page
@app.route('/', methods=['GET', 'POST'])
def serve_html():
    change_last_request_time()

    # handler
    if request.method == 'POST':
        current_settings['ASSISTANT_TTS'] =
request.form.get('ttsLang')
        current_settings['ASSISTANT_STT'] =
request.form.get('sttLang')
        current_settings['ASSISTANT_TRA'] =
request.form.get('transLang')

        message_type = 'info'
        message_text = "Збережено"
        return render_template('index.html',
assistant_stt=assistant_stt, assistant_tts=assistant_tts,
            assistant_tra=assistant_tra,
current_settings=current_settings,
            message_type=message_type, message_text=message_text)


    return render_template('index.html', assistant_stt=assistant_stt,
assistant_tts=assistant_tts,
        assistant_tra=assistant_tra,
current_settings=current_settings)


# other settings page
@app.route('/other_settings', methods=['GET', 'POST'])
def other_settings():
    change_last_request_time()

    # handler
    if request.method == 'POST':
        if request.form.get('SpeakTheAnswer') == None:
            current_settings['SPEAK_THE_ANSWER'] = "False"
        else:
            current_settings['SPEAK_THE_ANSWER'] = "True"
        if request.form.get('IsQuickAnswer') == None:
            current_settings['IS_QUICK_ANSWER'] = "False"
        else:
            current_settings['IS_QUICK_ANSWER'] = "True"
```

```
        settings_manager.set_setting('CURRENT_SETTINGS',
current_settings)
        settings_manager.save_settings()

        message_type = 'info'
        message_text = "Збережено"
        return render_template('otherSettings.html',
current_settings=current_settings,
            message_type=message_type, message_text=message_text)

    return render_template('otherSettings.html',
current_settings=current_settings)


# add languages page
@app.route('/add_languages')
def add_languages():
    change_last_request_time()
    return render_template('addLanguages.html')


# add speech to text language page
@app.route('/add_stt_language', methods=['GET', 'POST'])
def add_stt_language():
    change_last_request_time()

    # handler
    if request.method == 'POST':
        if request.form.get('label') == '' or request.form.get('key')
== '' or request.form.get('model') == '':
            message_type = 'error'
            message_text = "Заповніть всі поля"
            return render_template('addSTTLanguage.html',
                message_type=message_type, message_text=message_text,
label=request.form.get('label'),
                key=request.form.get('key'),
model=request.form.get('model'))

        if request.form.get('key') not in default_config.ISO_639_1:
            message_type = 'error'
            message_text = "Двох символьний код мови невірний,
ознайомтесь з стандартом"
            return render_template('addSTTLanguage.html',
                message_type=message_type, message_text=message_text,
label=request.form.get('label'),
```

```
                key=request.form.get('key'),
model=request.form.get('model'))

        if os.path.exists(os.path.join(os.getcwd(),
"models_stt//"+request.form.get('model'))) == False:
            message_type = 'error'
            message_text = "Модель з ім'ям
'"+request.form.get('model')+"' відсутня в теці 'models_stt' в корні
проекту"
            return render_template('addSTTLanguage.html',
                message_type=message_type, message_text=message_text,
label=request.form.get('label'),
                key=request.form.get('key'),
model=request.form.get('model'))

        key = request.form.get('key')
        model = request.form.get('model')
        label = request.form.get('label')

        assistant_stt[key] = {
            "model": model,
            "label": label
        }


        settings_manager.set_setting('ASSISTANT_STT', assistant_stt)
        settings_manager.save_settings()

        message_type = 'info'
        message_text = "Додано мову розпізнавання тексту"
        return render_template('addSTTLanguage.html',
            message_type=message_type, message_text=message_text)

    return render_template('addSTTLanguage.html')


# add text to speach language page
@app.route('/add_tts_language', methods=['GET', 'POST'])
def add_tss_language():
    change_last_request_time()

    # handler
    if request.method == 'POST':
        if request.form.get('label') == '' or request.form.get('key')
== '' or request.form.get('id_model') == '' or
request.form.get('speaker') == '':
```

```
            message_type = 'error'
            message_text = "Заповніть всі поля"
            return render_template('addTTSLanguage.html',
                message_type=message_type, message_text=message_text,
                label=request.form.get('label'),
key=request.form.get('key'),
                id_model=request.form.get('id_model'),
speaker=request.form.get('speaker'))

        if request.form.get('key') not in default_config.ISO_639_1:
            message_type = 'error'
            message_text = "Двох символьний код мови невірний,
ознайомтесь з стандартом"
            return render_template('addTTSLanguage.html',
                message_type=message_type, message_text=message_text,
                label=request.form.get('label'),
key=request.form.get('key'),
                id_model=request.form.get('id_model'),
speaker=request.form.get('speaker'))


        key = request.form.get('key')
        label = request.form.get('label')
        model_id = request.form.get('id_model')
        sample_rate = 48000
        speaker = request.form.get('speaker')

        assistant_tts[key] = {
            "label": label,
            "language": key,
            "model_id": model_id,
            "sample_rate": sample_rate,
            "speaker": speaker
        }

        settings_manager.set_setting('ASSISTANT_TTS', assistant_tts)
        settings_manager.save_settings()

        message_type = 'info'
        message_text = "Додано мову генерації голосу"
        return render_template('addSTTLanguage.html',
            message_type=message_type, message_text=message_text)

    return render_template('addTTSLanguage.html')
```

```
# add translate language page
@app.route('/add_translate_language', methods=['GET', 'POST'])
def add_translate_language():
    change_last_request_time()

    # handler
    if request.method == 'POST':
        if request.form.get('label') == '' or request.form.get('key')
== '':
            message_type = 'error'
            message_text = "Заповніть всі поля"
            return render_template('addTranslateLanguage.html',
                message_type=message_type, message_text=message_text,
                label=request.form.get('label'),
key=request.form.get('key'))

        if request.form.get('key') not in default_config.ISO_639_1:
            message_type = 'error'
            message_text = "Двох символьний код мови невірний,
ознайомтесь з стандартом"
            return render_template('addTranslateLanguage.html',
                message_type=message_type, message_text=message_text,
                label=request.form.get('label'),
key=request.form.get('key'))

        key = request.form.get('key')
        label = request.form.get('label')

        assistant_tra[key] = {
            "label": label,
            "lang": key
        }

        settings_manager.set_setting('ASSISTANT_TRA', assistant_tra)
        settings_manager.save_settings()

        message_type = 'info'
        message_text = "Додано мову перекладу тексту"
        return render_template('addSTTLanguage.html',
            message_type=message_type, message_text=message_text)

    return render_template('addTranslateLanguage.html')


# commands list page
@app.route('/commands_list', methods=['GET', 'POST'])
```

```
def commands_list():
    change_last_request_time()

    # handler
    if request.method == 'POST':

        # edit command render
        if request.form.get('command_edit') != None:
            word_list = [word.strip() for word in
assistant_cmd_list[request.form.get('command_edit')]['word_list']]
            word_list_str = ', '.join(word_list)

            return render_template('commandEdit.html',
command_edit=request.form.get('command_edit'),
                assistant_cmd_list=assistant_cmd_list,
word_list_str=word_list_str,

commandType=assistant_cmd_list[request.form.get('command_edit')]['comm
andType'],

customCommand=assistant_cmd_list[request.form.get('command_edit')]['cu
stomCommand'])


        # edit command handler
        if request.form.get('edit') == 'pass':
            if request.form.get('key') == '' or
request.form.get('word_list') == '' or
request.form.get('customCommand') == '':
                message_type = 'error'
                message_text = 'Будь-ласка, заповніть всі поля!'
                word_list = [word.strip() for word in
assistant_cmd_list[request.form.get('key')]['word_list']]
                word_list_str = ', '.join(word_list)


                return render_template('commandEdit.html',
message_type=message_type, command_edit=request.form.get('key'),
                    message_text=message_text,
assistant_cmd_list=assistant_cmd_list, word_list_str=word_list_str,
                    customCommand=request.form.get('customCommand'),
commandType=request.form.get('commandType'))


            if is_latin_only(request.form.get('key')) == False:
                message_type = 'error'
```

```
                message_text = 'Будь-ласка, введіть назву команди
англійською без пробілів!'
                word_list = [word.strip() for word in
assistant_cmd_list[request.form.get('key')]['word_list']]
                word_list_str = ', '.join(word_list)

                return render_template('commandEdit.html',
message_type=message_type, command_edit=request.form.get('key'),
                    message_text=message_text,
assistant_cmd_list=assistant_cmd_list, word_list_str=word_list_str,
                    customCommand=request.form.get('customCommand'),
commandType=request.form.get('commandType'))

            if assistant_cmd_list[request.form.get('key')]['isCustom']
== 'True':
                if request.form.get('commandType') == 'None':
                    message_type = 'error'
                    message_text = 'Будь-ласка, виберіть тип команди'
                    word_list = [word.strip() for word in
assistant_cmd_list[request.form.get('key')]['word_list']]
                    word_list_str = ', '.join(word_list)

                    return render_template('commandEdit.html',
message_type=message_type, command_edit=request.form.get('key'),
                        message_text=message_text,
assistant_cmd_list=assistant_cmd_list, word_list_str=word_list_str,

customCommand=request.form.get('customCommand'),
commandType=request.form.get('commandType'))


                if if_link(request.form.get('customCommand')) == False
and request.form.get('commandType') == 'openWebPage':
                    message_type = 'error'
                    message_text = "Це не посилання"
                    word_list = [word.strip() for word in
assistant_cmd_list[request.form.get('key')]['word_list']]
                    word_list_str = ', '.join(word_list)

                    return render_template('commandEdit.html',
message_type=message_type, command_edit=request.form.get('key'),
                        message_text=message_text,
assistant_cmd_list=assistant_cmd_list, word_list_str=word_list_str,

customCommand=request.form.get('customCommand'),
commandType=request.form.get('commandType'))
```

```
                else:
                    if
if_folder_path(request.form.get('customCommand')) == False and
request.form.get('commandType') == 'explorer':
                        message_type = 'error'
                        message_text = "Не шлях до теки"
                        word_list = [word.strip() for word in
assistant_cmd_list[request.form.get('key')]['word_list']]
                        word_list_str = ', '.join(word_list)

                        return render_template('commandEdit.html',
message_type=message_type, command_edit=request.form.get('key'),
                            message_text=message_text,
assistant_cmd_list=assistant_cmd_list, word_list_str=word_list_str,

customCommand=request.form.get('customCommand'),
commandType=request.form.get('commandType'))
                    else:
                        if
if_executable(request.form.get('customCommand')) == False and
request.form.get('commandType') == 'execute':
                            message_type = 'error'
                            message_text = "Не шлях до виконавчого
файлу"
                            word_list = [word.strip() for word in
assistant_cmd_list[request.form.get('key')]['word_list']]
                            word_list_str = ', '.join(word_list)

                            return render_template('commandEdit.html',
message_type=message_type, command_edit=request.form.get('key'),
                                message_text=message_text,
assistant_cmd_list=assistant_cmd_list, word_list_str=word_list_str,

customCommand=request.form.get('customCommand'),
commandType=request.form.get('commandType'))


            key = request.form.get('key')
            words_string = request.form.get('word_list')
            word_list = [word.strip() for word in
words_string.split(',')]


            if assistant_cmd_list[key]['can_delete'] == "False" and
assistant_cmd_list[key]['isCustom'] == "False":
```

```
                assistant_cmd_list[key] = {
                    "word_list": word_list,
                    "can_delete": "False",
                    "isCustom": "False",
                    "commandType": "None",
                    "customCommand": ""
                }
            else:
                commandType = request.form.get('commandType')
                customCommand = request.form.get('customCommand')
                assistant_cmd_list[key] = {
                    "word_list": word_list,
                    "can_delete": "True",
                    "isCustom": "True",
                    "commandType": commandType,
                    "customCommand": customCommand
                }

            settings_manager.set_setting('ASSISTANT_CMD_LIST',
assistant_cmd_list)
            settings_manager.save_settings()

            message_type = 'info'
            message_text = "Данні збережено"
            return render_template('commandEdit.html',
message_type=message_type, command_edit=request.form.get('key'),
                                   message_text=message_text,
assistant_cmd_list=assistant_cmd_list, word_list_str=words_string,

customCommand=assistant_cmd_list[key]['customCommand'],

commandType=assistant_cmd_list[key]['commandType'])


        # delete command handler
        if request.form.get('command_delete') != None:

settings_manager.delete_setting_from_key('ASSISTANT_CMD_LIST',
request.form.get('command_delete'))

            message_type = 'info'
            message_text = "Команду видалено"
            return render_template('commandsList.html',
assistant_cmd_list=assistant_cmd_list,
                message_type=message_type, message_text=message_text)
```

```
    return render_template('commandsList.html',
assistant_cmd_list=assistant_cmd_list)


# edit command page
@app.route('/command_edit', methods=['GET', 'POST'])
def command_edit():
    change_last_request_time()

    return render_template('commandsList.html',
assistant_cmd_list=assistant_cmd_list)


# add command page
@app.route('/add_command', methods=['GET', 'POST'])
def add_command():
    global last_request_time
    last_request_time = time.time()

    # handler
    if request.method == 'POST':
        if request.form.get('key') == '' or
request.form.get('word_list') == '' or
request.form.get('customCommand') == '':
            message_type = 'error'
            message_text = 'Будь-ласка, заповніть всі поля!'
            return render_template('addCommand.html',
message_type=message_type,
                message_text=message_text,
key=request.form.get('key'), word_list=request.form.get('word_list'),
                commandType=request.form.get('commandType'),
customCommand=request.form.get('customCommand'))


        if is_latin_only(request.form.get('key')) == False:
            message_type = 'error'
            message_text = 'Будь-ласка, введіть назву команди
англійською без пробілів!'
            return render_template('addCommand.html',
message_type=message_type,
                message_text=message_text,
key=request.form.get('key'), word_list=request.form.get('word_list'),
                commandType=request.form.get('commandType'),
customCommand=request.form.get('customCommand'))
```

```
        if request.form.get('commandType') == 'None':
            message_type = 'error'
            message_text = 'Будь-ласка, виберіть тип команди'
            return render_template('addCommand.html',
message_type=message_type,
                message_text=message_text,
key=request.form.get('key'), word_list=request.form.get('word_list'),
                commandType=request.form.get('commandType'),
customCommand=request.form.get('customCommand'))


        if if_link(request.form.get('customCommand')) == False and
request.form.get('commandType') == 'openWebPage':
            message_type = 'error'
            message_text = "Це не посилання"
            return render_template('addCommand.html',
message_type=message_type,
                message_text=message_text,
key=request.form.get('key'), word_list=request.form.get('word_list'),
                commandType=request.form.get('commandType'),
customCommand=request.form.get('customCommand'))
        else:
            if if_folder_path(request.form.get('customCommand')) ==
False and request.form.get('commandType') == 'explorer':
                message_type = 'error'
                message_text = "Не шлях до теки"
                return render_template('addCommand.html',
message_type=message_type,
                    message_text=message_text,
key=request.form.get('key'), word_list=request.form.get('word_list'),
                    commandType=request.form.get('commandType'),
customCommand=request.form.get('customCommand'))
            else:
                if if_executable(request.form.get('customCommand')) ==
False and request.form.get('commandType') == 'execute':
                    message_type = 'error'
                    message_text = "Не шлях до виконавчого файлу"
                    return render_template('addCommand.html',
message_type=message_type,
                        message_text=message_text,
key=request.form.get('key'), word_list=request.form.get('word_list'),
                        commandType=request.form.get('commandType'),
customCommand=request.form.get('customCommand'))
```

```
        key = request.form.get('key')
        word_list = [word.strip() for word in
request.form.get('word_list').split(',')]
        commandType = request.form.get('commandType')
        customCommand = request.form.get('customCommand')
        assistant_cmd_list[key] = {
            "word_list": word_list,
            "can_delete": "True",
            "isCustom": "True",
            "commandType": commandType,
            "customCommand": customCommand
        }

        settings_manager.set_setting('ASSISTANT_CMD_LIST',
assistant_cmd_list)
        settings_manager.save_settings()

        message_type = 'info'
        message_text = "Команду додано!"
        return render_template('addCommand.html',
message_type=message_type, message_text=message_text)

    return render_template('addCommand.html')


# VA names page
@app.route('/voice_assistant_names', methods=['GET', 'POST'])
def voice_assistant_names():
    change_last_request_time()

    # handler
    if request.method == 'POST':

        # deleting name
        if request.form.get('delete_name') != None:
            if request.form.get('delete_name') not in assistant_alias:
                message_type = 'error'
                message_text = 'Помилка при імені видаленні. <br>
Спробуйте ще раз!'
                return render_template('VANames.html',
message_type=message_type,
                       message_text=message_text,
assistant_alias=assistant_alias,
new_name=request.form.get('new_name'))
```

```
            settings_manager.delete_value_from_key('ASSISTANT_ALIAS',
request.form.get('delete_name'))
            message_type = 'info'
            message_text = "Ім'я голосого асистента видалено"
            return render_template('VANames.html',
message_type=message_type,
                message_text=message_text,
assistant_alias=assistant_alias,
new_name=request.form.get('new_name'))


        # adding new name
        if request.form.get('add_name') == 'add_name':
            if request.form.get('new_name') == '':
                message_type = 'error'
                message_text = "Введіть ім'я!"
                return render_template('VANames.html',
message_type=message_type,
                    message_text=message_text,
assistant_alias=assistant_alias,
new_name=request.form.get('new_name'))

            if request.form.get('new_name') not in assistant_alias:
                assistant_alias.append(request.form.get('new_name'))
                settings_manager.set_setting('ASSISTANT_ALIAS',
assistant_alias)
                settings_manager.save_settings()
                message_type = 'info'
                message_text = "Ім'я успішно додане"
                return render_template('VANames.html',
message_type=message_type,
                    message_text=message_text,
assistant_alias=assistant_alias)

            message_type = 'error'
            message_text = "Таке і'мя вже присутнє! <br> Придумайте
нове."
            return render_template('VANames.html',
message_type=message_type,
                message_text=message_text,
assistant_alias=assistant_alias,
new_name=request.form.get('new_name'))

    return render_template('VANames.html',
assistant_alias=assistant_alias)
```

```
# load styles to page
@app.route('/styles.css')
def serve_css():
    change_last_request_time()
    return send_file('web/styles.css', mimetype='text/css')
```

Б.2. Лістинг коду модуля розпізнавання людської мови

```
from notificationModule import show_notification
import vosk
import sys
import sounddevice as sd
import queue
import json

if getattr(sys, 'frozen', False):
    if sys.stderr is None:
        sys.stderr = sys.__stderr__

if getattr(sys, 'frozen', False) and hasattr(sys, '_MEIPASS'):
    sys.stderr = sys.__stderr__

# the queue
q = queue.Queue()

# This callback function when recording audio
def query_callback(indata, frames, time, status):
    if status:
        print(status, file=sys.stderr)
    q.put(bytes(indata))

# function data flow handler from the "vosk" model
def listen(callback, stt_model, assistant_alias, assistant_cmd_list,
assistant_tts, assistant_stt, assistant_tra, current_settings,
tts_model):
    model = stt_model['model']
    sample_rate = stt_model['sample_rate']
    block_size = stt_model['block_size']
    device = stt_model['device']
    d_type = stt_model['d_type']
    channels = stt_model['channels']

    show_notification("Голосовий помічник","Розпізнавання голосу
увімкнено. Слухаю")
```

```
    try:
        with sd.RawInputStream(samplerate=sample_rate,
blocksize=block_size, device=device, dtype=d_type, channels=channels,
                               callback=query_callback):
            rec = vosk.KaldiRecognizer(model, sample_rate)
            while True:
                data = q.get()
                if rec.AcceptWaveform(data):
                    callback(json.loads(rec.Result())["text"],
assistant_alias, assistant_cmd_list, assistant_tts, assistant_stt,
assistant_tra, current_settings, tts_model)
    except(KeyboardInterrupt):
        sys.exit(0)

class SpeachToText:
    # init
    def __init__(self, model:str):
        self.model = vosk.Model('models_stt/' + model)
        self.sample_rate = 16000
        self.block_size = 8000
        self.device = 1
        self.d_type = 'int16'
        self.channels = 1


    # getting the model
    def get_model(self):
        return {'model': self.model,
                'sample_rate': self.sample_rate,
                'block_size': self.block_size,
                'device': self.device,
                'd_type': self.d_type,
                'channels': self.channels}

    # setting the model
    def set_model(self):
        pass
```

Б.3. Лістинг коду командного процессора

```
from ttsModule import VoiceModel
from translatorModule import TranslatorModel
from openAIModule import TextNeuralNetwork as gpt
from sttModule import SpeachToText as speachToText, listen
from SettingsManager import SettingsManager, settings_manager
from fuzzywuzzy import fuzz
```

```
from notificationModule import show_notification
from datetime import datetime
from writeCommand import listen_write
from multiprocessing import Process
from tkinter import Scrollbar, Text, Frame
import random
import tkinter as tk
import time
import asyncio
import subprocess
import platform
import webbrowser
import inflect


settings_manager.load_settings()

# Asynchronous function for request in chatGPT
async def gpt_req(text_to_gpt: str):
    freeGPT = gpt()
    return await freeGPT.create_prompt(text_to_gpt)


# The function converts numbers to a string
def convert_numbers_in_text(text: str):
    p = inflect.engine()
    words = text.split()
    for i in range(len(words)):
        try:
            num = int(words[i])
            words[i] = p.number_to_words(num)
        except ValueError:
            pass
    return ' '.join(words)


# The function creates a tkinder window to display the string
def paint_answer(answer_text: str):
    root = tk.Tk()
    root.title("Відповідь")

    frame = Frame(root)
    frame.pack(fill='both', expand=True)

    text_widget = Text(frame, wrap='word', width=60, height=15)
    text_widget.insert('1.0', answer_text)
```

```
    scroll = Scrollbar(frame, command=text_widget.yview)
    text_widget.config(yscrollcommand=scroll.set)

    text_widget.grid(row=0, column=0, sticky='nsew')
    scroll.grid(row=0, column=1, sticky='ns')

    frame.grid_rowconfigure(0, weight=1)
    frame.grid_columnconfigure(0, weight=1)

    root.geometry("500x500")
    root.mainloop()


# Function creates a process to display a string via tkinter
def paint_gpt_answer(text: str):
    window_process = Process(target=paint_answer, args=[text])
    window_process.start()


# function of opening a folder or executable file of a custom command
def execute_custom_command_exe(ff_path: str, type: str):
    try:
        if type == 'file':
            result = subprocess.run([ff_path], check=True, text=True,
capture_output=True)
        if type == 'path':
            system_type = platform.system()
            if system_type == "Windows":
                result = subprocess.run(['explorer', ff_path],
check=True)
                show_notification("Голосовий помічник","Теку
выдкрито")
            elif system_type == "Linux":
                result = subprocess.run(['xdg-open', ff_path],
check=True)
                show_notification("Голосовий помічник","Теку
выдкрито")

    except subprocess.CalledProcessError as e:
        #print("Error:", e)
        pass


# the function translates all spoken text into a user request without
a command and an alias assistant
```

```
def make_req_from_string(original_string: str, key_world: str):
    key_world_count = len(key_world.split())
    words = original_string.split()
    remaining_words = words[key_world_count+1:]
    result_string = ' '.join(remaining_words)
    return result_string


# making the textToSpeach model
def make_tss_model(lang: str, tts: str):
    language = tts[lang]['language']
    model_id = tts[lang]['model_id']
    sample_rate = tts[lang]['sample_rate']
    speaker = tts[lang]['speaker']
    return VoiceModel(language, model_id, sample_rate, speaker)


# The function translates text into voice
def speech_the_text(text: str, tts_model: VoiceModel):
    tts_model.play_audio(text)
    return 0


# execute command function
def execute_cmd(cmd: str, key_world: str, voice: str, assistant_alias,
assistant_cmd_list, assistant_tts, assistant_stt, assistant_tra,
current_settings, tts_model: VoiceModel):
    cur_tra_to_lang = current_settings['ASSISTANT_TRA']
    cur_speach_lang = current_settings['ASSISTANT_TTS']
    cur_speaker_lang = current_settings['ASSISTANT_STT']
    is_quick_answer = current_settings['IS_QUICK_ANSWER']
    speak_the_answer = current_settings['SPEAK_THE_ANSWER']


    if cmd == 'help':
    # Add browser opening to documentation
        webbrowser.open('https://github.com/OlegRad4encko/Diploma')


    # command time - makes notifications with time \ or says so
    elif cmd == 'time':
        current_time = datetime.now()
        p = inflect.engine()
        hour_text = p.number_to_words(current_time.hour)
        minute_text = p.number_to_words(current_time.minute)
```

```
        time_text = f"'{hour_text} {p.plural('hour',
current_time.hour)} and {minute_text} {p.plural('minute',
current_time.minute)} at the moment"

        translate = TranslatorModel('en', cur_speach_lang)
        translated_text = translate.translate_text(time_text)

        if speak_the_answer == "False":
            show_notification("Голосовий помічник",translated_text)
        else:
            speech_the_text(translated_text, tts_model)


    # command open browser - opens the browser
    elif cmd == 'open browser':
        webbrowser.open('https://www.google.com.ua/?hl=uk')


    # The joke command - makes a notification with a joke \ or tells a
joke
    elif cmd == 'joke':
        jokes = ['Є люди, які несуть в собі щастя. Коли ці люди поруч,
все ніби стає яскравим і барвистим. Але моя дружина їх називає
алкашами!',
            'З одного боку, в гості без пляшки не підеш. А з іншого
якщо в тебе є пляшка, та на холєру в гості пертись?',
            'Кохана, давай миритися. Вибач мене, я був не правий.
Стоп! Не їж! Я приготую щось інше!',
            'Кажуть, що у геніїв в будинку має бути безлад. Дивлюсь на
свою дитину і гордість розпирає! Енштейна виховую!.',
            'Христос родився! Маю три вакцини, сертифікат вакцинації,
негативний тест, оброблений антисептиком! Колядувати можна?',
            'Пішла, куда послав. Веду себе, як ти назвав. І чому я
тебе раніше не слухала?!']

        joke = random.choice(jokes)
        translated_text = ''
        if speak_the_answer == "False":
            if cur_speach_lang != 'ua':
                translate = TranslatorModel('uk', cur_speach_lang)
                translated_text = translate.translate_text(joke)
            else:
                translated_text = joke
            show_notification("Голосовий помічник",translated_text)
        else:
            if cur_speach_lang != 'ua':
```

```
                translate = TranslatorModel('uk', cur_speach_lang)
                translated_text = translate.translate_text(joke)
            else:
                translated_text = joke

            speech_the_text(translated_text, tts_model)


    # command write - writes to the active window, everything that the
user says
    elif cmd == 'write':
        write_process = Process(target=listen_write,
args=[assistant_stt[current_settings['ASSISTANT_STT']]['model']])
        write_process.start()

        answer = "Диктуйте"
        if speak_the_answer == "False":
            if cur_speach_lang != 'ua':
                translate = TranslatorModel('uk', cur_speach_lang)
                translated_text = translate.translate_text(answer)
            else:
                translated_text = answer
            time.sleep(5)
            show_notification("Голосовий помічник",translated_text)
        else:
            if cur_speach_lang != 'ua':
                translate = TranslatorModel('uk', cur_speach_lang)
                translated_text = translate.translate_text(answer)
            else:
                translated_text = answer
            time.sleep(6)
            speech_the_text(translated_text, tts_model)


    # command find - opens a browser window with a user request
    elif cmd == 'find':
        va_request = make_req_from_string(voice, key_world)
        url_request = "https://www.google.com/search?q=" +
'+'.join(va_request.split())
        webbrowser.open(url_request)
        show_notification("Голосовий помічник","Відкрито браузер з
запитом '"+va_request+"'")


    # the say command - either says what the user said \ or notifies
that the voice is not turned on
```

```
    elif cmd == 'say':
        if speak_the_answer == "False":
            show_notification("Голосовий помічник","Увімкніть
озвучування відповідей голосового помічника")

        else:
            speech_the_text(make_req_from_string(voice, key_world),
tts_model)


    # makes a request to the GPT chat
    elif cmd == 'gpt':
        va_request = make_req_from_string(voice, key_world)
        show_notification("Голосовий помічник","Очікуйте, звертаюсь до
ChatGPT, це може зайняти декілька хвилин")
        result = asyncio.run(gpt_req(va_request))
        result_without_numbers = convert_numbers_in_text(result)

        if len(result_without_numbers) < 500:
            if speak_the_answer == "False":
                if len(result_without_numbers) < 100:
                    show_notification("Голосовий помічник", result)
                else:
                    paint_gpt_answer(result)
            else:
                paint_gpt_answer(result)
                translate = TranslatorModel('en', cur_speach_lang)
                translated_text =
translate.translate_text(result_without_numbers)
                speech_the_text(translated_text, tts_model)

        else:
            paint_gpt_answer(result)


    # translate command
    elif cmd == 'translate':
        va_request = make_req_from_string(voice, key_world)
        translate = TranslatorModel(cur_speach_lang, cur_tra_to_lang)
        translated_text = translate.translate_text(va_request)
        result = va_request +f'\u000a\u000a'+translated_text
        paint_gpt_answer(result)


    # custom command
    else:
```

```
        if assistant_cmd_list[cmd]['commandType'] == 'explorer':

execute_custom_command_exe(assistant_cmd_list[cmd]['customCommand'],
'path')
        elif assistant_cmd_list[cmd]['commandType'] == 'execute':

execute_custom_command_exe(assistant_cmd_list[cmd]['customCommand'],
'file')
        elif assistant_cmd_list[cmd]['commandType'] == 'openWebPage':

webbrowser.open(f'{assistant_cmd_list[cmd]["customCommand"]}')
        else:
            show_notification("Голосовий помічник", "Я Вас не
зрозумів")



## recognizing function v1
# def recognize_cmd(cmd: str, assistant_cmd_list):
#     rc = {'cmd': '', 'percent': 0, 'key_world': ''}
#     max_per = 0
#     key_world = ''
#     for word_list_key in assistant_cmd_list:
#         word_list = assistant_cmd_list[word_list_key]["word_list"]
#         for x in word_list:
#             vrt = fuzz.ratio(cmd, x)
#             if max_per < vrt:
#                 max_per = vrt
#                 key_world = x
#             if vrt > rc['percent']:
#                 rc['cmd'] = word_list_key
#                 rc['percent'] = vrt
#                 rc['key_world'] = key_world
#     return rc



## recognizing function v2
def recognize_cmd(cmd: str, assistant_cmd_list):
    rc = {'cmd': '', 'percent': 0, 'key_world': ''}

    for word_list_key in assistant_cmd_list:
        word_list = assistant_cmd_list[word_list_key]["word_list"]
        max_per, key_world = max((fuzz.partial_ratio(cmd, x), x) for x
in word_list)
        if max_per > rc['percent']:
```

```
            rc['cmd'] = word_list_key
            rc['percent'] = max_per
            rc['key_world'] = key_world

    return rc


# voice recognition callback function
def stt_respond(voice: str, assistant_alias, assistant_cmd_list,
assistant_tts, assistant_stt, assistant_tra, current_settings,
tts_model):
    if voice.startswith(assistant_alias):
        cmd = recognize_cmd(filter_cmd(voice, assistant_alias),
assistant_cmd_list)
        if cmd['cmd'] not in assistant_cmd_list.keys():
            show_notification("Голосовий помічник", "Я Вас не
зрозумів")
        else:
            execute_cmd(cmd['cmd'], cmd['key_world'], voice,
assistant_alias, assistant_cmd_list, assistant_tts, assistant_stt,
assistant_tra, current_settings, tts_model)


# filtering the CMD, removing epy assistant alias from raw_voice
def filter_cmd(raw_voice: str, assistant_alias):
    cmd = raw_voice

    for x in assistant_alias:
        cmd = cmd.replace(x, "").strip()

    return cmd


# update settings for voice assistant
def update_settings():
    settings_manager.load_settings()
    current_settings =
settings_manager.get_setting('CURRENT_SETTINGS', {})
    assistant_cmd_list =
settings_manager.get_setting('ASSISTANT_CMD_LIST', {})
    assistant_alias = settings_manager.get_setting('ASSISTANT_ALIAS',
{})
    assistant_stt = settings_manager.get_setting('ASSISTANT_STT', {})
    assistant_tts = settings_manager.get_setting('ASSISTANT_TTS', {})
    assistant_tra = settings_manager.get_setting('ASSISTANT_TRA', {})
```

```
    current_settings =
settings_manager.get_setting('CURRENT_SETTINGS', {})
    return {
        "current_settings": current_settings,
        "assistant_cmd_list": assistant_cmd_list,
        "assistant_alias": assistant_alias,
        "assistant_stt": assistant_stt,
        "assistant_tts": assistant_tts,
        "assistant_tra": assistant_tra,
        "current_settings": current_settings
    }


# Run command processor
def run_command_processor():
    settings = update_settings()
    current_settings = settings['current_settings']
    assistant_cmd_list = settings['assistant_cmd_list']
    assistant_alias_list = settings['assistant_alias']
    assistant_stt = settings['assistant_stt']
    assistant_tts = settings['assistant_tts']
    assistant_tra = settings['assistant_tra']
    current_settings = settings['current_settings']

    cur_speach_lang = current_settings['ASSISTANT_TTS']

    tts_model = make_tss_model(cur_speach_lang, assistant_tts)
    assistant_alias = tuple(assistant_alias_list)

    speachVA =
speachToText(assistant_stt[current_settings['ASSISTANT_STT']]['model']
)
    listen(stt_respond, speachVA.get_model(), assistant_alias,
assistant_cmd_list, assistant_tts, assistant_stt, assistant_tra,
current_settings, tts_model)
```

ДОДАТОК В

В.1 Посилання на віддалений репозиторій GitHub

Рисунок В.1 – посилання на віддалений репозиторій GitHub

В.2 Посилання на відео-демонстрацію голосового помічника

Рисунок В.2 – посилання на відео-демонстрацію роботи голосового асистента