

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ
ГОЛОСОВОГО ПОМІЧНИКА ДЛЯ РОБОТИ
З СЕРВІСАМИ ШТУЧНОГО ІТЕЛЕКТУ
SOFTWARE OF A VOICE ASSISTANT FOR WORKING
WITH ARTIFICIAL INTELLIGENCE SERVICES

Виконав: студент 4 курсу, групи КНТ-111
Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

БЄЛОВ К.В.

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БЄЛОВА Кирила Віталійовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення голосового помічника для роботи з сервісами штучного інтелекту. Software of a Voice Assistant for Working with Artificial Intelligence Services.

керівник проєкту (роботи) к. т. н., СТЕПАНЕНКО Олександр Олексійович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 02 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., старший викладач		

7. Дата видачі завдання “ 14 ” квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Кирило БЄЛОВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Олександр СТЕПАНЕНКО
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
94с., 31 рис., 14 табл., 3 дод., 35 джерела.

PYTHON, RTX, НЕЙРОМЕРЕЖА, AI, STT, TTS, WINDOWS, ОС,
NVIDIA, VOSK. SILERO, VOICE HELPER.

Об'єкт дослідження: процес розробки програмного забезпечення для взаємодії з сервісами штучного інтелекту та операційною системою комп'ютера.

Предмет дослідження: програмні засоби, що забезпечують зручність використання функціоналу операційної системи та взаємодію з нейронними мережами.

Мета роботи: створення програмного забезпечення, що спрощує доступ до можливостей операційної системи та нейронних мереж шляхом реалізації голосового помічника.

Матеріали, методи та технічні засоби: мова програмування Python, бібліотеки для синтезу та розпізнавання мовлення, засоби автоматизації взаємодії з операційною системою, а також базові інструменти штучного інтелекту.

Результати: розроблено голосового помічника, який дозволяє керувати операційною системою та отримувати доступ до функцій штучного інтелекту за допомогою голосових команд.

Практична цінність роботи: створена програма може використовуватись як персональний інтерфейс для полегшення взаємодії користувача з комп'ютером, що особливо корисно в контексті роботи з нейронними мережами, запуску програм та автоматизації задач.

Висновки: реалізовано сучасний інструмент у вигляді голосового помічника, який інтегрується з операційною системою та сервісами штучного інтелекту, забезпечуючи зручність, швидкість та інтуїтивність користування.

Галузь використання: інтелектуальні користувацькі інтерфейси, системи автоматизації, навчальні платформи, персональні асистенти.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 94 p., 31 figures, 14 tables, 3 appendixes, 35 sources.

PYTHON, RTX, NEURAL NETWORK, AI, STT, TTS, WINDOWS, OC, NVIDIA, VOSK. SILERO, VOICE ASSISTANT.

Object of research: the process of developing software for interaction with artificial intelligence services and the computer operating system.

Subject of research: software tools that provide ease of use of the operating system functionality and interaction with neural networks.

Purpose: to create software that simplifies access to the capabilities of the operating system and neural networks by implementing a voice assistant.

Materials, methods, and techniques: Python programming language, libraries for speech synthesis and recognition, tools for automating interaction with the operating system, and basic artificial intelligence tools.

Results: A voice assistant was developed that allows you to control the operating system and access artificial intelligence functions using voice commands.

Practical value of the work: the created program can be used as a personal interface to facilitate user interaction with a computer, which is especially useful in the context of working with neural networks, running programs, and automating tasks.

Conclusions: A modern tool in the form of a voice assistant has been implemented that integrates with the operating system and artificial intelligence services, providing convenience, speed, and intuitiveness of use.

Applications: intelligent user interfaces, automation systems, learning platforms, personal assistants.

ЗМІСТ

Перелік скорочень та умовних познач	9
Вступ	10
1. Аналіз предметної області	12
1.1 Голосові помічники в сучасному світі	12
1.2 Аналіз голосових помічників з нейромережами	14
1.2.1 Програмне забезпечення Microsoft Copilot	14
1.2.2 Програмне забезпечення Amazon Alexa	18
1.2.3 Програмне забезпечення NVIDIA ChatRTX	20
1.2.4 Програмне забезпечення Project G-Assist	24
1.3 Порівняння систем нейромереж	28
1.4 Висновки за розділом 1	29
2 Матеріали і методи	31
2.1 Вибір мови програмування	31
2.2 Вибір нейромережі для програмного забезпечення Voice Helper	33
2.3 Вибір засобу для озвучування запиту від нейромережі	36
2.4 Вибір модуля перетворення голосу на текст	39
2.5 Висновки за розділом 2	43
3 Опис програми	45
3.1 Структура програмного забезпечення Voice Helper	45
3.2 Функціонування програмного забезпечення Voice Helper	46
3.3 Розгортання програмного забезпечення голосового помічника Voice Helper	48
3.4 Проектування інтерфейсу програмного забезпечення Voice Helper	49
3.5 Функції програмного забезпечення Voice Helper	52
3.6 Висновки за розділом 3	53
4 Експлуатація, тестування та експериментальне дослідження програми	54
4.1 Призначення та умови застосування програми	54
4.2 Характеристики Voice Helper	55

4.3 Інструкція по експлуатації програми.....	55
4.3.1 Звернення до програми.....	55
4.3.2 Вхідні й вихідні дані	55
4.3.3 Повідомлення.....	56
4.4 Виконання програмного забезпечення Voice Helper	56
4.5 Тестування програмного забезпечення Voice Helper.....	56
4.6 Висновки за розділом 4	59
Висновки	60
Перелік джерел посилання	62
Додаток А Технічне завдання	66
Додаток Б Фрагмент тексту програми	70
Додаток В Слайди презентації.....	87

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AI - Artificial intelligence

USD – United States dollar

STT – Speech-to-Text

TTS – Text to Speech

ОС – Операційна система

ВСТУП

Програмне забезпечення голосового помічника для автоматизації взаємодії користувача з комп'ютером та AI-сервісами дозволяє значно спростити виконання рутинних операцій: запуск програм, управління файлами, планування завдань і обробку природної мови за допомогою нейронних мереж (наприклад, для транскрипції й синтезу мовлення) [1]. Глобальний ринок голосових асистентів у 2024 році оцінювався в близько 7,35 млрд USD, що свідчить про стрімке зростання попиту на подібні рішення [2].

Програмне забезпечення Voice Helper призначене для використання у сфері персональної автоматизації на комп'ютерах з операційною системою Windows. Воно може застосовуватись як інструмент голосового керування для підвищення зручності взаємодії з ПК, що особливо актуально для користувачів з обмеженими фізичними можливостями, стримерів, розробників або людей, які прагнуть мінімізувати використання клавіатури та миші в повсякденній роботі.

Python є провідною мовою для створення AI-додатків і наукових проєктів, завдяки простоті синтаксису, широті екосистеми бібліотек (включно зі `speech_recognition`, `pyttsx3`, `vosk`, `openai` тощо) та активній спільноті розробників, що забезпечує швидкий цикл розробки й високу читабельність коду [3].

Для розпізнавання мовлення й синтезу голосу відкритим стандартом вважається Vosk який має моделі до 100МБ, і вони не потребують інтернет для їх застосування, які масштабуються від мікро-пристроїв (Raspberry Pi) до серверних кластерів і підтримує багато мов і діалектів [4].

Для досягнення мети роботи - створення модульного голосового помічника на Python, здатного розпізнавати й синтезувати мовлення, обробляти команди та керувати ресурсами операційної системи — поставлено такі завдання:

- виконати аналіз існуючих рішень (open-source і комерційних) та оцінити їхні переваги й обмеження;
- спроектувати архітектуру із окремими модулями для STT, TTS й взаємодії з ОС;
- реалізувати розпізнавання голосу та синтез мовлення з використанням бібліотек Vosk, Silero Speech;
- інтегрувати зовнішні AI-сервіси через Python для обробки складніших запитів;
- розробити механізми керування ОС (os, subprocess) на основі інтерпретації голосових команд;
- провести функціональне і навантажувальне тестування, оцінити точність розпізнавання й затримку відповіді;
- оптимізувати продуктивність і забезпечити розширюваність для подальшого додавання нових модулів та команд.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Голосові помічники в сучасному світі

Сьогодні нейронні мережі стають надзвичайно популярними та поширеними. Їх можливості дозволяють впроваджувати їх в програми з різними задачами від рекомендацій і фінансових ринків до обробки природного мовлення. Великі моделі не мають достатньою гнучкої архітектури та енергоефективності, але для забезпечення режиму роботи на лімітованих ресурсах вони зможуть розширюватися за допомогою Eastern controllers [5].

Загалом, нейромережі стають частиною нашого життя. Вони живлять рекомендаційні системи на стрімінгових платформах, краще за пошукові алгоритми, оптимізують логістику та аналізують складні бізнес-процеси. Серед ключових трендів 2025 року також є GNN's, де граф стає базовою архітектурою в основних фреймворках Vertex.AI, що дозволяє ефективно структурувати великі аудіодані [6].

Одним з найважливіших інтерфейсів людина-комп'ютер є голосовий ввід. Завдяки технологіям Speech to Text та Text to Speech та нейронним мережам ми можемо створювати інтуїтивні системи, які сприймають природну мову та відповідають на неї так само, як говорить користувач. Основні елементи NLP двигунів, такі як розпізнавання намірів, витяг сутностей та генерація природних відповідей, все частіше базуються на довготривалих рекурентних або трансформерах з самоувагою [7].

Десктопні додатки з вбудованими нейронними мережами будуть використовуватися для голосових асистентів: від миттєвого перекладу розмовних робочих процесів до управління можливостями ПК за допомогою звукових команд. LinguaSync – десктопна програма на платформі глибокого навчання, яка може передавати емоційність мовлення та розроблена для операційної системи Windows, Linux та macOS. Додаток Voice Helper створений для перетворення мови в текст та навпаки [8].

У професійному середовищі голосові десктопні помічники демонструють вражаючу продуктивність і значно спрощують виконання рутинних завдань, таких як запуск програм, управління файлами, складання звітів і швидкий доступ до необхідної інформації. Деякі рішення, наприклад, DVAFNN, спеціально розроблені для людей похилого віку, використовують feed forward нейронні мережі для розуміння команд і адаптації інтерфейсу відповідно до когнітивних потреб користувачів [9].

Також голосові інтерфейси збільшують доступність технічних систем для осіб з обмеженими можливостями, забезпечують безконтактне керування в виробничих і медіа середовищах, сприяють підвищенню безпеки, наприклад, при керуванні транспортними засобами або в лабораторіях. Нейронні мережі на стороні клієнта знижують навантаження на хмарні сервіси і гарантують приватність даних [8].

Практична користь у роботі та житті - у бізнес-середовищі голосові помічники дозволяють автоматизувати введення даних, запуск регулярних звітів, пошук інформації в ERP/CRM-системах і обслуговування клієнтів через голосові чат-боти. У побуті вони допомагають керувати мультимедіа, планувати нагадування, контролювати «розумний будинок» та навіть готувати рецепти, прочитуючи інструкції з екрану. Голосовий ввід скорочує час на рутинні операції до кількох секунд і знижує навантаження на руки та очі користувачів.

Найближчі перспективи розвитку - це оптимізація моделей для локального застосування, підвищення швидкодії на десктоп-ПК, а також інтеграція адаптивного навчання, що дозволить системам автоматично підлаштовуватися під індивідуальні мовні особливості користувачів. Поступове поєднання edge-AI (розгортання алгоритмів і моделей штучного інтелекту (ШІ) безпосередньо на пристроях на кордоні мережі) та хмарних нейросервісів задає тон новим рішенням, які змінять спосіб взаємодії з комп'ютерами та суттєво полегшать наше повсякденне життя.

1.2 Аналіз голосових помічників з нейромережами

1.2.1 Програмне забезпечення Microsoft Copilot

Програмне забезпечення, що називається Microsoft Copilot (рис. 1.1) у Windows 11, інтегрує локальні модулі STT/TTS для розпізнавання та синтезу мовлення з хмарними AI-сервісами Azure, щоб запропонувати інтелектуальний інтерфейс, який об'єднує голосовий та текстовий ввід в єдиний досвід, не заважаючи робочому процесу користувача.

Введення голосу активується натисканням на "Натисни для розмови" (одночасне натискання клавіш Alt + Пробіл протягом щонайменше 2 секунд) [10], що дозволяє користувачу негайно ініціювати розмову з Copilot, дозволяючи відповідати в контексті того, що наразі на екрані.

Аналіз вмісту може проводитись у браузері за допомогою функції Copilot Vision, яка розпізнає пов'язані об'єкти на екрані та озвучує їх вголос, за винятком інтерактивних дій (таких як "натисніть кнопку"), які доступні лише з планом Pro.

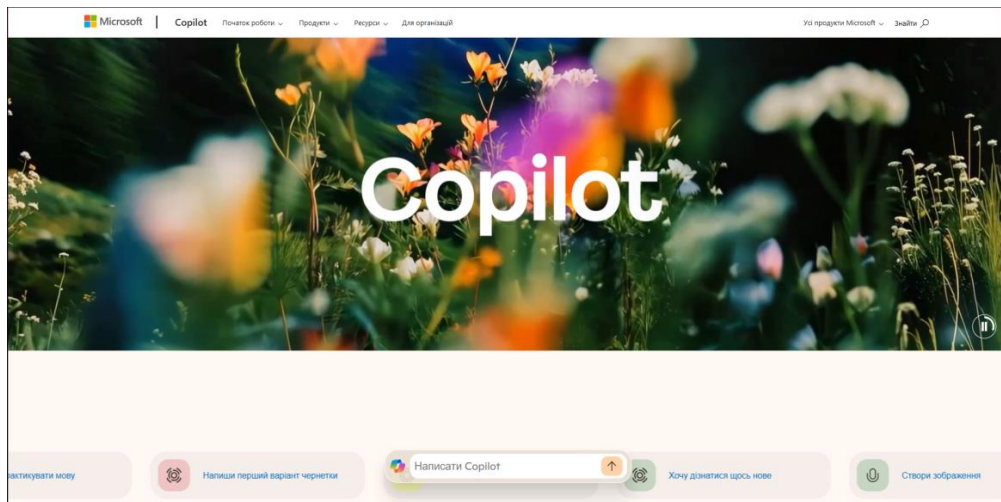


Рисунок 1.1 – Головна сторінка Microsoft Copilot [11]

Copilot Pro - це платна підписка (\$20 на місяць), яка надає пріоритетний доступ до нових моделей GPT під час високого попиту і — на відміну від

безкоштовної версії — дозволяє необмежену кількість запитів, включаючи в Word, Excel, PowerPoint, Outlook та OneNote.

Користувачі корпоративного сегмента можуть об'єднувати Microsoft 365 Copilot (\$30 за користувача/місяць) з Teams, SharePoint і Dynamics 365 для автоматизованої транскрипції нарад та звітності для бізнесу.

Microsoft Copilot дотримується конфіденційності:

- текстові запити та відповіді надсилаються в хмару;
- аудіо та знімки екрану не зберігаються;
- вживаються всі заходи для захисту запитів згідно з GDPR та налаштуваннями корпоративних DLP.

Нещодавно Microsoft також оголосила, що всі користувачі отримають:

- необмежений доступ до голосових можливостей і "глибокого мислення" на основі моделі o1 без додаткової плати;
- але затримка може збільшитися під час пікових періодів.

В огляді Copilot на PCMag[12] було виявлено, що він є гнучким для:

- досліджень і створення контенту;
- кросплатформеного досвіду;
- але відставав з невеликими затримками при складних запитах.

Справді, зараз Copilot попередньо встановлено на нові ПК, що постачаються з Windows 11 (рис. 1.2), а старіші машини можуть додати його через Microsoft Store.

Основні характеристики Microsoft Copilot:

- безшовна інтеграція з Windows та Microsoft 365 для оптимальної операційної системи та офісного досвіду;
- швидкий голосовий ввід з використанням натискання для розмови (Alt + Space). – Copilot Vision: огляд та підсвічування елементів екрану в Edge;
- про план: доступ до найновіших моделей GPT з пріоритетною підтримкою та без обмежень на запити;
- місцеві модулі STT/TTS, готові до роботи без інтернету;

- стандартні вимоги до конфіденційності (GDPR, DLP) для корпоративних клієнтів.



Рисунок 1.2 – Microsoft Copilot в Windows 11 [13]

Основні системні модулі:

- бібліотека захоплення звуку (sounddevice/pyaudio);
- двигун STT (адаптований на місці Vosk + місцевоспецифічний проти використання хмари Azure Speech);
- модуль NLP (Azure OpenAI, розуміння намірів);
- двигун TTS (pyttsx3/gTTS + Azure Neural Voices);
- copilot Vision (аналіз екранного контенту Edge) – Прошарок інтеграції (os/subprocess, Microsoft Graph, Teams API) ;
- інтерфейс користувача (бокова панель з панелі завдань, віджети з Office).

Програмне забезпечення Microsoft Copilot підходить для різноманітних користувачів, включаючи тих, хто хоче спростити повсякденні завдання, до корпоративних команд, які використовують автоматизацію для звітів,

транскрипції зустрічей та робочих процесів з Microsoft 365. Її сильні та слабкі сторони були наведені у таблиці 1.1 та таблиці 1.2.

Таблиця 1.1 – Переваги Microsoft Copilot

Переваги	Опис
1	2
Широка інтеграція	Єдиний інтерфейс у Windows 11 та Office для всіх завдань
Гібридна архітектура	Швидкі локальні модулі STT/TTS та потужні хмарні моделі GPT
Інноваційний інтерфейс	Copilot Vision і Press-to-Talk для інтуїтивного голосового керування
Розширені можливості з Pro-планом	Необмежені запити та пріоритетний доступ до останніх моделей
Відповідність корпоративним стандартам	GDPR, DLP, Microsoft Purview для захисту даних

Таблиця 1.2 – Недоліки Microsoft Copilot

Недоліки	Опис
1	2
Вартість підписки	\$20 – 30 / користувач / місяць вимагає бюджету
Інтернет-залежність	Хмарні функції недоступні офлайн
Обмежена локальна кастомізація	Фіксований набір мов і моделей у локальних модулях
Затримка при навантаженні	Під час піків відповіді можуть сповільнюватися через обмеження хмари
Copilot Vision у Pro-версії	Повний функціонал аналізу екрану — лише для підписників Pro

Цей огляд демонструє, що Microsoft Copilot поєднує зручність і швидкість локальної роботи з потужністю хмарних AI-сервісів, але потребує врахування питань підключення, вартості та апаратних ресурсів.

1.2.2 Програмне забезпечення Amazon Alexa

Amazon Alexa на Windows (рис. 1.3). Особливості Amazon Alexa для Windows - з увімкненим Amazon Alexa для Windows ви маєте доступ до знайомого голосового інтерфейсу Alexa на вашому робочому столі ПК з Windows 10, перетворюючи його на справжній центр управління розумним будинком та мультимедійний досвід [14].

Після активації «пробуджувального слова» («Alexa») або через додаток з панелі задач, окрім відтворення музики та новин, користувач може керувати гучністю, налаштуваннями екрана, календарем і списками завдань, все без використання клавіатури чи миші.

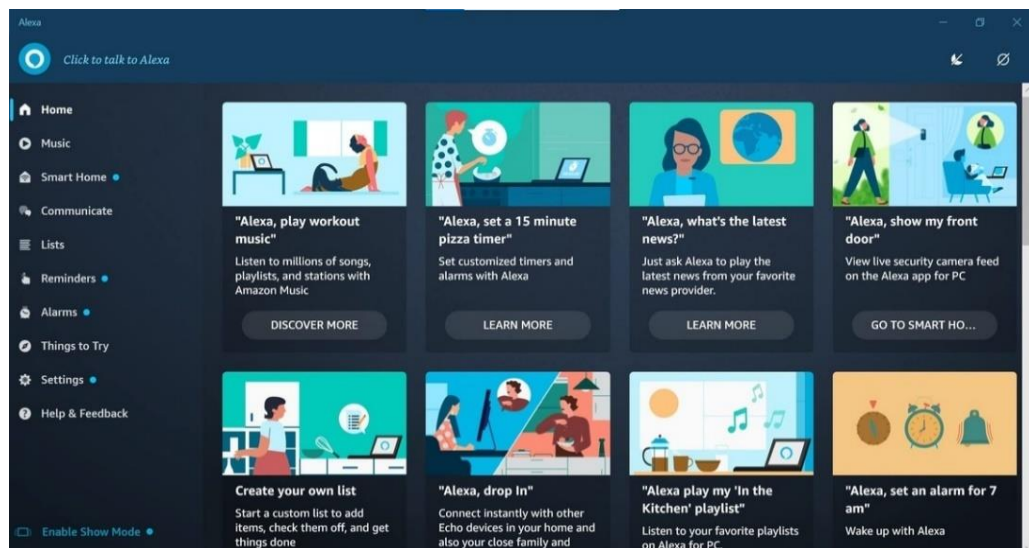


Рисунок 1.3 – Amazon Alexa в Windows [15]

Режим Show Mode перетворює ПК у версію Echo Show: візуальні зображення на повний екран на картках для таких речей, як погода, рецепти та медіа, голосові взаємодії (що досі можливі, навіть коли машина теоретично

перебуває в режимі сну; скажіть «Алекса, увімкни режим Show Mode» або натисніть відповідну кнопку в інтерфейсі користувача) [15].

Цей режим розроблений для великих екранів та високоякісних мікрофонів, коли ПК виконує роль стаціонарного розумного дисплея.

Технічна архітектура Amazon Alexa:

- локальний wake-word розпізнавач: фоновий сервіс Windows постійно слухає “Alexa” за допомогою аудіодрайверів, мінімізуючи навантаження на ресурси й забезпечуючи швидкий відгук;

- Alexa Voice Service (AVS): після виявлення «wake word» аудіо шифрується та надсилається в хмару AWS для обробки STT, NLU та генерації відповіді (TTS) з високою точністю;

- UI Show Mode: створений на базі Electron/React, додає можливості візуальної взаємодії, карток та віджетів, а також дозволяє налаштовувати час автоматичного запуску повноекранного режиму;

Сценарії використання Amazon Alexa:

- «розумний дім»: голосове вмикання ламп, термостатів, замків і камери відеоспостереження без зміни додатків або пристроїв;

- медіа та розваги: запуск музичних плейлистів, подкастів і відео на ПК, пошук контенту в стрімінгових сервісах, керування відтворенням;

Порівняння переваг та недоліків цієї програми наведені у таблиці 1.3 та таблиці 1.4

Таблиця 1.3 – Переваги Amazon Alexa:

Переваги	Опис
1	2
Постійне прослуховування	Wake word «Alexa» активує помічника миттєво без натискань кнопок
Велика екосистема Skills	Доступ до тисяч голосових навичок для дому, розваг і продуктивності
Крос-синхронізація	Спільний список справ, календар та історія запитів

Продовження таблиці 1.3

1	2
Show Mode	Повноекранний інтерфейс із візуальними картками, працює навіть у режимі очікування

Таблиця 1.4 – Недоліки Amazon Alexa:

Недоліки	Опис
Залежність від інтернету	Більшість можливостей (STT, NLU, Skills) працюють лише з підключенням до AWS
Витрати ресурсів	Фоновий сервіс постійно слухає й може споживати CPU/пам'ять
Необхідність акаунту Amazon	Для активації й налаштування потрібні облікові дані Amazon
Обмежена локальна інтеграція	Менше системних команд порівняно з вбудованими OS-помічниками

Отже, Amazon Alexa для ПК має сенс, коли ви бажаєте отримати голосового асистента та «розумний дім» на своєму робочому столі — і не зважайте на залежність додатку від інтернету та його споживання ресурсів.

1.2.3 Програмне забезпечення NVIDIA ChatRTX

NVIDIA ChatRTX - це демонстраційний додаток, який повністю працює локально на ПК з Windows (див. рис. 1.4) з GPU GeForce RTX, інтегруючи великі мовні моделі (LLM) з вашими власними даними (документами, зображеннями, нотатками) через покращене створення запитів (RAG) та апаратне прискорення TensorRT LLM.

Всі обробки виконуються на вашому локальному комп'ютері, тому ви можете використовувати голосові команди для перевірки цін десятки тисяч разів на секунду без затримок - і ми не надсилаємо ваші дані в хмару.

З додаванням Whisper від OpenAI, ChatRTX пропонує голосовий ввід, а модуль CLIP підтримує пошук контенту за зображеннями [16], [17].



Рисунок 1.4 – Інтерфейс NVIDIA ChatRTX [17]

Основні можливості та переваги NVIDIA ChatRTX:

- локальне виконання: усі модулі STT, NLP і TTS виконуються прямо на ПК без інтернету, що гарантує приватність і швидкість реакції;
- RAG з TensorRT-LLM: поєднання retrieval-augmented generation із апаратним прискоренням дозволяє швидко отримувати точні відповіді з ваших даних [17];
- голосове введення: використання Whisper для розпізнавання мовлення забезпечує комфортну hands-free взаємодію з асистентом;
- підтримка різних форматів: ChatRTX індексує файли TXT, PDF, DOC/DOCX, XML, а також зображення (JPG, PNG, GIF) та відео з YouTube, надаючи можливість шукати інформацію в будь-якому контенті [17];

- вибір моделей: користувачі можуть обирати між відкритими LLM (Mistral 7B, LLaMA 2, Gemma, ChatGLM3) залежно від вимог до продуктивності й точності.

Технічна архітектура NVIDIA ChatRTX:

- audio Capture & STT: локальний модуль захоплення аудіо (pyaudio/sounddevice) передає запис до Whisper для розпізнавання тексту;
- RAG Pipeline: індексовані дані (текст, зображення) організовуються в векторному сховищі, з якого виконується пошук перед генерацією відповіді;
- tensorRT-LLM Inference: оптимізовані моделі проганяються через TensorRT, використовуючи RTX-ядра для пришвидшення обчислень;
- TTS & Output: відповіді синтезуються локально або можуть відтворюватися за допомогою стандартних движків (pyttsx3, gTTS);
- UI: простий інтерфейс із папкою вибору даних, вікном чату та налаштуваннями моделей реалізовано на базі Qt/Electron.

Системні вимоги NVIDIA ChatRTX

- GPU: GeForce RTX 30-ї або 40-ї серії з ≥ 8 ГБ VRAM;
- ОЗП: мінімум 16 ГБ RAM;
- диски: близько 30 ГБ вільного простору для програмних компонентів і моделей;
- ОС: Windows 11 із драйвером NVIDIA 535.11 або новішим.

Також деякі переваги NVIDIA ChatRTX наведенні в таблиці 1.5.

Таблиця 1.5 – Переваги NVIDIA ChatRTX

Переваги	Опис
1	2
Повна локальність	Ніяких даних у хмарі — конфіденційність і низька затримка
Апаратне прискорення RTX	Використання TensorRT-LLM дає миттєві відповіді завдяки RTX-ядрам

Продовження таблиці 1.5

1	2
Гнучкість моделей	Підтримка Mistral, LLaMA 2, Gemma, ChatGLM3 тощо для різних задач
Багатоформатний контент	TXT, PDF, DOCX, XML, JPG, PNG, GIF, YouTube-відео — все в одній папці
Інтегроване голосове введення	Whisper забезпечує точне розпізнавання навіть за фонових шумів

Сценарії використання NVIDIA ChatRTX:

- дослідницький помічник: індексуйте наукові статті та власні нотатки, щоб швидко отримувати контекстно-релевантні відповіді й резюме;
- особистий Knowledge Base: зберіть усі робочі документи, листи, фінансові звіти - і ставте ChatRTX питання про фінанси, проєкти, дедлайни;
- студентський супровід: навчальні матеріали і лекції у папці - ChatRTX конспектує, відповідає на запитання й готує флеш-картки;
- креативний асистент: завантажте тексти, зображення та ідеї - використовуйте ChatRTX для мозкового штурму, генерації ескізів сюжетів чи дизайну;
- технічний рев'ю: індексуйте кодові репозиторії та документацію; ChatRTX допомагає знаходити помилки, пояснювати функції та пропонувати оптимізації.
- інтеграція в бізнес-аналітику: завантажте фінансові звіти, маркетингові дані та презентації - ChatRTX аналізує показники, генерує висновки та допомагає формувати стратегії прийняття рішень.

Однак при всіх можливостях та перевагах у цієї версії також є і недоліки котрі були наведенні у таблиці 1.6.

Таблиця 1.6 – Недоліки NVIDIA ChatRTX

Недоліки	Опис
1	2
Високі апаратні вимоги	RTX серії 30/40/50 із ≥ 8 ГБ VRAM, 16 ГБ RAM і 30 ГБ диска можуть бути недоступними для багатьох
Обмежена пам'ять сесії	Поки що немає повноцінного багатокрокового діалогу — лише single-shot відповіді
Великі обсяги встановлення	Залежність від моделей і NIM-компонентів може займати десятки гігабайт
Початкова стадія розробки	Демо-додаток із мінімальним UI та багатьма експериментальними функціями

NVIDIA ChatRTX демонструє, що може робити локальний чат LLM, щоб переосмислити конфіденційність та чуйність AI-додатків. Використовуючи RAG, TensorRT LLM та Whisper, він ставить користувача на чолі створення власного особистого віртуального асистента, хоча це можливо лише для останніх GPU RTX і потребує величезних інвестицій.

1.2.4 Програмне забезпечення Project G-Assist

Проект G-ASSIST — це експериментальний AI-помічник, який керується голосовими/текстовими командами і допомагає користувачам зручніше керувати налаштуванням системи та ігор, використовуючи продукти G, вбудовані в локальний ПК-продукт з GEFORCE RTX. Він включений у новий додаток NVIDIA, активується комбінацією клавіш Alt + G і використовує невеликі локальні мовні моделі (SLM) для швидкого виведення на GPU.

G-ASSIST допомагає забезпечити користувачам оптимальні налаштування для гри та системи, відслідковуючи і відображаючи показники

продуктивності, такі як кадри в секунду (FPS), температура, споживання енергії та багато іншого; він також може керувати периферійними пристроями (освітленням, профілями та інше) і тримати процес перед ПК локально для захисту приватності та низького часу відгуку.

Основні можливості Project G-Assist:

- локальний інтерфейс на RTX GPU: G-Assist використовує TensorRT-LLM для запуску SLM без залучення хмари, що забезпечує затримку відповіді в межах кількох секунд[18];

- швидке оптимування: за простим запитом «optimize my settings» асистент застосовує рекомендовані NVIDIA App пресети для гри (DLSS, Reflex, Overclock) на основі тисяч конфігурацій, протестованих NVIDIA;

- моніторинг продуктивності: голосові команди дозволяють відобразити графіки FPS, температур, споживаної потужності й затримок у реальному часі через накладку NVIDIA App (рис. 1.5);

- керування периферією: можна змінювати налаштування RGB підсвічування, профілі миші та клавіатури, а також аудіо через відповідні плагіни та API;

- простий інтерфейс: активація Alt + G відкриває компактне вікно чату без відриву від гри; також доступні текстові запити через NVIDIA App overlay.

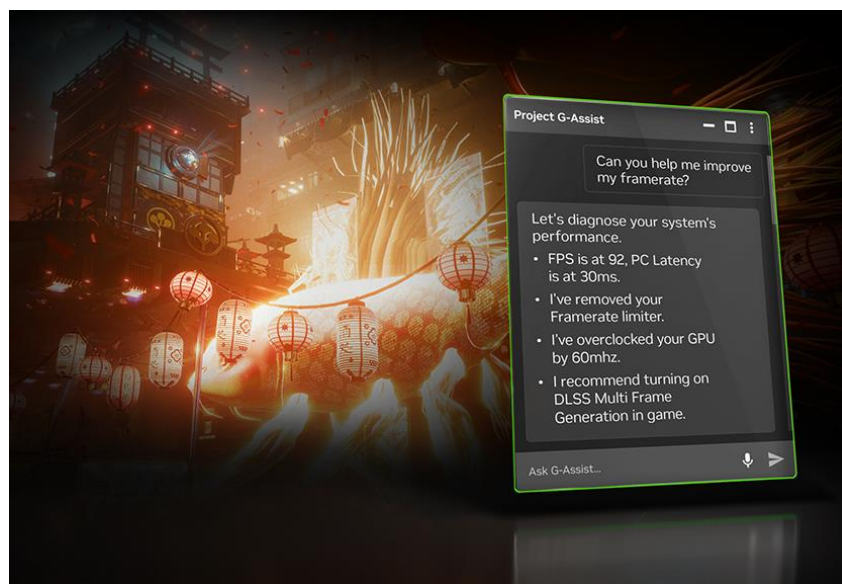


Рисунок 1.5 – Допомога в підвищенні FPS в грі Cyberpunk 2077 [18]

Технічна архітектура Project G-Assist:

- wake & Capture: Alt + G запускає прослуховування голосу або відкриває текстове поле;
- STT & TTS: Whisper-подібний модуль розпізнавання мовлення працює локально для перетворення голосу в текст і відповіді голосом через NVIDIA Neural Voices;
- RAG + TensorRT-LLM: індексовані дані системних журналів і пресетів передаються до локальної SLM, оптимізованої через TensorRT;
- UI Overlay: Electron-/Qt-подібний інтерфейс запускається у вигляді накладки, показуючи текст і графіки продуктивності;
- feedback Loop: користувачі можуть надсилати відгуки безпосередньо через кнопку «Send Feedback» в NVIDIA App[16].

Project G-Assist має потужний можливості як для геймерів так і для звичайних користувачів, хоч її встановлення потребує відеокарт доволі високої версії, що є її мінусом, які наведенні в таблиці 1.8, проте якщо користувач має відеокарту підходящу під потреби програмами можна отримати доволі непоганий помічник собі на систему, яка непогано себе показує, та не сильно грузить її, сильні сторони були наведені в таблиці 1.7

Таблиця 1.7 – Переваги Project G-Assist

Переваги	Опис
1	2
Повна локальність	Жодних даних у хмарі, всі обчислення проходять на RTX GPU, гарантує конфіденційність і низьку затримку
Швидке оптимування ігор	Одноразовий запит для застосування найкращих пресетів NVIDIA App на основі тисяч тестованих конфігурацій

Продовження таблиці 1.7

1	2
Глибокий моніторинг продуктивності	Реал-тайм графіки FPS, температур, затримок і споживання енергії в накладці без додаткових програм
Легка інтеграція периферії	Керування підсвічуванням, профілями миші/клавіатури та аудіо через плагіни в одному інтерфейсі
Простий UX	Активація Alt + G і мінімалістичний чат-інтерфейс без відриву від геймплею або роботи

Таблиця 1.8 – Недоліки Project G-Assist

Недоліки	Опис
1	2
Високі апаратні вимоги	Потребує GeForce RTX 30/40/50 серій із мінімум 12 ГБ VRAM і сучасного процесора
Обмежена універсальність моделей	Використовуються невеликі локальні SLM, які можуть давати менш точні відповіді порівняно з хмарними LLM
Тимчасова продуктивна деградація під час інференсу	Використання частини потужності GPU для AI-інференсу може призвести до короткочасного зниження FPS
Експериментальний статус	G-Assist все ще в бета тестуванні

Це реальний приклад того, як локальні AI-помічники змінюють спосіб, у який геймери та досвідчені користувачі відчувають свої ПК, одночасно забезпечуючи миттєву та конфіденційну підтримку, коли користувачі

потребують оптимізації їхніх систем, ігор та отримання допомоги щодо їхніх ПК.

1.3 Порівняння систем нейромереж

Для цього розділу розглянемо 4 з найбільших та найпопулярніших систем, які були проаналізовані в попередніх розділах:

- Microsoft Copilot;
- Amazon Alexa;
- NVIDIA ChatRTX;
- Project G-Assist.

Вони відрізняються за типом використаної архітектури, застосуваннями, рівнем конфіденційності та апаратною підтримкою.

Ось таблиця, яка розкриває основні характеристики цих систем.

Таблиця 1.9 – Порівняння нейромереж для ПК

Критерій порівняння	Голосові помічники			
	Microsoft Copilot	Amazon Alexa	NVIDIA ChatRTX	Project G-Assist
1	2	3	4	5
Синтез мовлення (TTS)	Azure Neural Voices	AVS TTS	Silero TTS локально	NVIDIA Neural Voices
Архітектура	Гібрид «edge + cloud»	Хмара (AVS)	Повністю локальна	Повністю локальна
Розпізнавання мовлення (STT)	Локальний + Azure	Wake-word «Alexa»	Whisper локально	Whisper-подібний локально
Приватність даних	Середня (текст - хмара)	Низька (аудіо - AWS)	Висока (усе локально)	Висока (усе локально)

Продовження таблиці 1.9

1	2	3	4	5
Апаратні вимоги	Сучасний ПК, інтернет	Будь-який ПК, мікрофон	RTX 30/40 ($\geq$ 8 ГБ VRAM)	RTX 30/40/50 ($\geq$ 12 ГБ
Голосовий інтерфейс	Press-to-Talk	Wake-word	Alt + G, чат-вікно	Alt + G голос/текст
Контекстна підтримка	Так	Ні	Обмежена	Так
Інтеграція з ОС	PowerShell, Graph API	Обмежена	Системні журнали	NVIDIA App overlay
Можливість кастомізації	Обмежена	Обмежена	Висока	Середня
Основні сценарії	Office, пошук, файли	«Розумний дім», медіа	Аналіз локальних даних	Оптимізація ігор/системи
Вартість	Платна	Платна	Безкоштовно	Безкоштовно

Таким чином, Microsoft Copilot та Amazon Alexa краще підходять для систем з глибокою хмарною інтеграцією та корпоративними сервісами, тоді як NVIDIA ChatRTX і Project G-Assist добре працюють для повністю локальних систем з максимальною конфіденційністю, продуктивністю для натренованих моделей, оптимізованих на сучасних графічних процесорах.

1.4 Висновки за розділом 1

Ринок голосових помічників швидко змінюється: від локальних рішень, наприклад, NVIDIA ChatRTX, Project G-Assist, до гібридних систем "edge + cloud" та хмарних сервісів. Кожен продукт вирішує свою власну проблему, націлений на свою аудиторію – деякі потребують максимальної

продуктивності та приватності, інші потребують глибокої інтеграції в екосистему та простоти використання:

- вимогливі апаратні вимоги локального помічника (Project G-Assist, ChatRTX) гарантують швидкість і приватність, але роблять його недоступним для деяких користувачів без сучасних графічних процесорів та великого обсягу пам'яті;

- гібридні рішення (Microsoft Copilot) дозволяють мати локальні можливості з живленням від батареї та хмарні функції; вони підходять як для корпоративного, так і для особистого використання, але вимагають постійного підключення до Інтернету та можуть призводити до додаткових витрат на послуги підписки;

- хмарні помічники, такі як Amazon Alexa, Google Assistant, навпаки, забезпечують спектр можливостей на різноманітних пристроях, але залежать від доступу до мережі та наявності зовнішніх API для роботи;

- спеціалізовані рішення (наприклад, голосові помічники для геймерів або професійної сцени) вражають своєю гнучкістю та автоматизацією певних процесів і пропонують глибокі варіанти налаштування, але часто не забезпечують весь спектр універсальних функцій.

Тому вибір найкращого голосового помічника залежить від особистих уподобань або від того, як кожен користувач зважає між власними серверами даних і швидкістю відгуку, зручним і багатофункціональним поєднанням навичок хмари чи компромісом між інтеграцією, вартістю та приватністю. Всі ці підходи сумісні один з одним і спонукають до більшої кількості інновацій в інтерфейсі людина-комп'ютер.

В результаті, аналізу конкурентів, виявлення їх сильних та слабих сторін був створений додаток Voice Helper. Який був спроектований з урахуванням функціоналу популярних додатків і поєднує найбільш кращі аспекти від кожного з них.

2 МЕТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

Голосовий асистент Voice Helper був розроблений з використанням мови програмування Python [19].

Python є відкритим кодом, тому розробники мають багато свободи для виправлення помилок і вдосконалення коду. Його простий і легкий для вивчення синтаксис є більш читабельним і, отже, знижує вартість обслуговування програм.

Вибір мови програмування аналізується з урахуванням вимог швидкої інтеграції модулів розпізнавання мовлення та синтезу мовлення, а також доступності методологій RAG і інтерфейсів програмування нейронних мереж (NN).

Більше інформації про переваги Python можна знайти у підручнику Міхаель Мюллера [20], де описується потенціал фреймворку для продуктів машинного навчання, а також у підручнику Марка Лутца [21], де характеристики мови синтезовані з точки зору системного програмування.

Велика та активна спільнота користувачів і розробників Python дозволяє покладатися на швидкі розв'язання будь-яких технічних проблем і безліч документації, навчальних посібників та прикладів коду, які завжди доступні під рукою. Python широко використовується в області штучного інтелекту та машинного навчання (бібліотеки TensorFlow, PyTorch), крім того, дуже добре адаптований для реалізації модулів голосового вводу та інтеграції з нейронною мережею DeepSeek [22].

Три мови, на яких я порівнюватиму Voice Helper, це Python, Java та C#, і я порівнюю їх на основі наступних аспектів, щоб зрозуміти, що дійсно відрізняє їх у розробці голосового помічника.

Java може бути використана для створення голосових асистентів головним чином на корпоративній стороні або на серверній стороні.

Завдяки інструментальним наборам, таким як Java Speech API або CMU Sphinx, можна виконувати примітивне розпізнавання мови. Однак сучасні мовні моделі та фреймворки глибокого навчання не мають достатньої підтримки. Java в основному використовується в серверному стеку, налаштуванні, в якому запити до голосового асистента обробляються окремо від машинного навчання або управляються з використанням периферійних сервісів. Java також широко використовується для розробки голосових інтерфейсів для мобільних телефонів на базі Android [23].

C# в основному використовується для розробки голосових інтерфейсів в ОС Windows. Крім того, за допомогою Microsoft Speech SDK мова може бути використана для реалізації як функціональності розпізнавання, так і синтезу. Програми використовують C# для настільного голосового інтерфейсу, який є інтегрованим у систему для його функцій (управління файлами, запуск програм, сповіщення). Але з огляду на глибоке навчання, можливості взаємодії з великими мовними моделями для C# далеко не ідеальні, і для реалізації більш просунутих AI-функцій слід використовувати сторонні інструменти або зв'язки з іншими мовами [24].

Таблиця порівняння мов та обґрунтування вибору наведено в таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для Voice Helper

Критерій порівняння	Мови програмування		
	Python	Java	C#
1	2	3	4
Широке застосування в AI/ML	+	+-	+-
Крос-платформеність	+	+	-
Наявність бібліотек для аудіо та NLP	+	-	+-
Простота синтаксису	+	+-	+-

Продовження таблиці 2.1

1	2	3	4
Швидкість розробки та прототипування	+	-	-
Підтримка спільнотою та документація	+	+-	+-
Інтеграція з ОС для автоматизації задач	+	+-	+

Отже, для реалізації голосового помічника Voice Helper обрано мову Python, яка поєднує в собі швидкість розробки, широкую підтримку AI/ML, необхідну для інтеграції нейромереж і обробки голосових команд, а також можливість простого перенесення коду між платформами.

2.2 Вибір нейромережі для програмного забезпечення Voice Helper

За останні роки нейромережі стали невід’ємною частиною нашого життя, якщо років 5 назад нам доводилось витратити багато часу на рутинні задачі, то тепер ми делегуємо це нейромережам. Завдяки науковцям та людей з освітою, котрі освітлювали та розвивали тему нейромереж ми маємо LLM моделі [25], одна з яких була інтегрована в голосовий помічник Voice Helper.

У розробці модуля обробки природної мови для Voice Helper застосовано нейронний сервіс DeepSeek [26]. Він пропонує потужний механізм для перетворення голосових запитань у відповіді, що мають семантичну правильність між ними в майже реальному часі. Також при розробці були випробувані сервіси від OpenAI GPT-4 та Google PaLM.

OpenAI GPT-4 відомий своїм генеруванням високоякісного тексту та роботою з контекстом. Модель зазвичай застосовується в чат-ботах, системах пошуку і голосових помічниках, але GPT-4 доступний лише через платний API. Реалізація на локальних пристроях також неможлива, що ускладнює контроль над користувацькими даними. Це робить його менш цікавим для систем, які вимагають конфіденційності і не хочуть залежати від публічних серверів.

Інша сильна модель LLM — це Google PaLM, модель, розроблена в рамках Google. Вона зосереджена на інтеграції з сервісами Google Cloud і має широкий спектр використання від обробки запитів до полегшення мобільного кодування. PaLM досягає високої точності, але доступна тільки через комерційний API і не може бути налаштована. До того ж, затримка перевищує 500 мс, що є неприпустимим у реальних системах голосу, таких як Voice Helper.

В свою чергу DeepSeek містить приблизно 600 мільярдів параметрів [27], що допомагає йому діяти розумніше в контексті діалогу та задовільно обробляти складні користувацькі команди з різними компонентами. Глибока архітектура та аспекти її реалізації можна знайти в вільному доступі [28], [29], які надали основні уявлення про процес навчання та теоретичні концепції щодо розгортання моделей DNN.

Його розроблено з акцентом на низьку затримку висновків: завдяки оптимізації масштабування на етапі тестування та прискореній підтримці виконання на GPU, модель видає відповіді значно швидше, ніж багато інших моделей аналогічного розміру. У голосовому асистенті це є критично важливим, адже будь-яка затримка, яку може відчувати користувач, може зменшити комфорт та ефективність взаємодії.

Нейромережа від китайської компанії має відкритий вихідний код і може бути встановлена у локальному приватному хмарному середовищі або на особистому сервері чи робочій станції. Таким чином, програма Voice Helper може у подальшому зберігати весь контроль над даними користувача та відповідати навіть найвибагливішим вимогам безпеки та вимогам GDPR. Розгортання відбувається через API, що означає, що помічник не має повної конфіденційності, проте досягає великої швидкості та не грузить систему комп'ютера.

На відміну від інших підходів, DeepSeek не вимагає жодної оплати за запити API: люди можуть використовувати сучасну модель без оплати за кількість токенів або хвилини обробки. Це значно знижує витрати на

управління хмарною інфраструктурою в порівнянні з підписними сервісами. Актуальність цього низького - для проекту Voice Helper, який, ймовірно, отримуватиме сотні таких запитів на день, це становить значні економії на витратах.

Важливий проривний фактор, це плавне інтегрування DeepSeek з голосовими фреймворками. І завдяки додаткам, сумісним із Whisper STT, та можливості підключення до будь-якого движка TTS (gTTS, Silero тощо) розробка Voice Helper отримує можливість замінити системи STT та TTS. Це дозволяє зменшити вживання коду в стилі «збірки» та робить систему більш стабільною.

Більше того, архітектура DeepSeek вбудовує нові блоки, такі як динамічно квантизовані треновані ваги та операції, оптимізовані для TensorRT на NVIDIA GPUs, що дозволяє зменшити час обробки запитів на 40-60% у порівнянні з розглянутими базовими реалізаціями трансформерів. Він обробляє запити в середньому за 1,2 секунди на запит у тесті GSM8K; "зручність" інтерфейсу мови визначається як менше 2 секунд. Для порівняння, їхні аналоги розміром LLaMA-2 70B показують час 2,8-3,5 секунди для тих самих випадків. Додаткова оптимізація виконується через масштабування пікселів під час інференсу, яке автоматично вибирає кількість обчислювальних ресурсів залежно від складності запиту.

DeepSeek є ідеальним вибором для створення систем генерації, підкріпленої пошуком: наприклад, модель може поєднувати генеративні відповіді з пошуком певних фрагментів у локальних документах/базах даних та інтегрованих носіях даних. Це може бути особливо корисним для подальшої розробки та більш тісної інтеграції DeepSeek в Voice Helper, якому потрібно не тільки відповідати загалом, але взаємодіяти з системою та програмами користувача.

Таблиця порівня нейросервісів та обґрунтування вибору наведено в таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору нейросервісу для Voice Helper

Критерій порівняння	Нейросервіси		
	DeepSeek	OpenAI GPT-4	Google PaLM
1	2	3	4
Затримка відповіді	низька (< 300 мс)	середня (≈ 600 мс)	Середня (≈ 500 мс)
Вартість запитів	безкоштовно	платно	платно
Розмір моделі	600 млрд	175 млрд	540 млрд
Спільнота та підтримка	висока	римка	висока
Можливість кастомізації	висока	обжена	середня
Інтергація з голосовими інтерфейсами	seamless або через API	через API	через API

Отже, вибір DeepSeek обґрунтований його продуктивністю та низькою затримкою, безкоштовністю API-доступу, високим рівнем конфіденційності, гнучкими опціями розгортання і широкою екосистемою підтримки, що робить його ідеальним нейросервісом для реалізації Voice Helper.

2.3 Вибір засобу для озвучування запиту від нейромережі

Синтез тексту в мовлення (TTS) або синтез мови є процесом створення мовлення з написаного тексту від початку до кінця. Він широко застосовується в різних програмах, таких як голосові помічники та навігаційні системи [30].

Щодо додавання модуля синтезу мови у Voice Helper, ми обрали Silero TTS як основну технологію. Цей метод забезпечує високоякісний вихід голосу, зменшує залежність від мережі та має добру здатність до

продуктивності в усіх сценах [31]. Також було розглянуто ряд систем для синтезу мови, а саме: Coqui, Google Cloud, Amazon Polly та gTTS.

Coqui TTS (XTTSv2) це сучасна багатомовна система з підтримкою відкритого коду для клонування голосу. Вона має хороший синтез тексту в мову і активну спільноту розробників. Однак XTTSv2 є досить громіздким і складним у локальному використанні, що викликає затримки у легких настільних додатках.

Google Cloud TTS надає хмарний API для перетворення тексту в мовлення з кількома голосами та мовами. Голос має дуже високу якість та підтримує нейронний синтез. Але це платна послуга, яка потребує постійного доступу до Інтернету, що не підходить для "незалежної" операції Voice Helper.

Amazon Polly це служба синтезу мовлення, яка використовує передові технології глибокого навчання для перетворення мови, яка звучить як людський голос. Вона добре працює з інфраструктурою AWS, однак, як і Google Cloud, рішення закрите та вимагає оплати за кожен запит, а також залежності від громадської інфраструктури.

gTTS (Google Text-to-Speech) працює з API Google Translate. Він легко інтегрується, підтримує різні мови та інші функції, має низьку якість звуку і швидко перевищує обмеження за кількістю запитів.

Silero TTS це швидка, високоякісна система тексту в мовлення, розроблена для забезпечення природного перетворення тексту в мовлення в реальному часі. Ключова особливість Silero TTS - це можливість роботи повністю офлайн: моделі (наприклад, v3_en_indic.pt для англійської та v4_ru.pt для російської) завантажуються в проект і зберігаються локально, щоб TTS міг працювати без мережевих затримок.

Важливі аспекти Silero TTS - це те, що моделі можуть бути компактні за розміром (близько 50–100 МБ кожна), що робить їх легкими для інтеграції навіть у програми з малою кількістю дискового простору, і те, що вони підтримують обчислення як на CPU, так і на GPU для прискорення висновків.

Інтегрована оптимізація для PyTorch дозволяє генерувати звук із затримкою менш ніж 200 мс, що забезпечує плавність і природність агентів розмови.

Silero TTS також підтримує функцію налаштування голосу: тон та швидкість читання, інтонацію можна змінювати за допомогою простого API, а генерація голосу є гнучкою та може бути адаптована для задоволення специфічних потреб - від системних голосових повідомлень до динамічного попереднього перегляду нарації.

Інтеграція з Python займає всього кілька рядків коду: достатньо вказати шлях до файлу моделі та використати метод синтезу, який повертає дані у форматі WAV/PCM.

Завдяки підходу на основі відкритих вихідних кодів за підтримки спільноти, Silero TTS постійно оновлюється та отримує нові моделі, щоб підтримувати більше мов та додавати покращення якості озвучки. Розробники можуть розраховувати на надійність, швидкий запуск та масштабування можливостей голосового помічника для кожного проєкту.

Таблиця порівняння модулів TTS та обґрунтування вибору наведено в таблиці 2.3.

Таблиця 2.3 – Обґрунтування вибору модуля синтезу мовлення

Критерій порівняння	Модуль TTS				
	Silero TTS	Coqui TTS (XTTSv2)	Google Cloud TTS	Amazon Polly	gTTS (Google TTS)
1	2	3	4	5	6
Розмір моделі	різні (від ~100 МБ до кілька ГБ)	різні (від ~100 МБ до кілька ГБ)	не застосовується (хмарний сервіс)	не застосовується (хмарний сервіс)	не застосовується (хмарний сервіс)

Продовження таблиці 2.3

1	2	3	4	5	6
Вартість	безкоштовно	безкоштовно	платно	платно	безкоштовно
Затримка	< 200 мс	< 200 мс	≈ 500 мс	≈ 300 – 600 мс	≈ 400 – 700 мс
Мови	5	> 16 мов із XTTSv2, +1100 моделей	40+ мов, 220+ голосів	40+ мов, 100+ голосів	40+ мов
Оффлайн-підтримка	+	+	-	-	-

Silero TTS входить до числа кращих локальних TTS-систем завдяки компактним моделям (50-100 МБ), швидкому часу відгуку (< 200 мс) та природному звучанню аудіо без підключення до Інтернету.

2.4 Вибір модуля перетворення голосу на текст

Технологія STT (Speech-to-Text), яка, по суті, є транскрибуванням, це процес перетворення усної мови в письмову. Алгоритми машинного навчання та нейронні мережі, які аналізують оцифровані звукові хвилі та перетворюють їх у текст, слугують основою цієї технології [32].

Для вибору модуля перетворення голосу в текст для Voice Helper було випробувано ряд моделей, як - Whisper, Google Speech-to-Text, DeepSpeech та Vosk. За випробуванням та тестуванням була використана модель Vosk [33]. Vosk це безкоштовний і з відкритим вихідним кодом інструментарій для розпізнавання мовлення, який працює на базі Kaldi та призначений для роботи на локальних пристроях у реальному часі, без необхідності підключення до хмари.

Whisper (OpenAI) є відкритою STT-службою для серверної частини, яка використовує невелику архітектуру трансформерів для розпізнавання мовлення на понад 90 мовах. Whisper має високу точність, але є ресурсомістким та зазвичай працює в хмарі або локально на потужних графічних процесорах (GPU). Через це час доступу може становити від 400 до 800 мс, і це критичний фактор, який відіграє велику роль у багатьох інтерактивних додатках, що вимагають миттєвого часу відгуку.

Сервіс перетворення мовлення в текст від Google має хмарний API, який розпізнає понад 120 мов та діалектів і дозволяє додаткам автоматично обирати правильну модель на основі мови тексту. Він підтримує автоматичне визначення мови та шумозаглушення. Хоча сервіс є надзвичайно надійним та добре інтегрованим з ширшою платформою Google Cloud, це виключно онлайн-API, ціноутворення базується на секунді обробленого аудіо, і є додаткові витрати на мережеві запити.

Модель STT DeepSpeech (Mozilla) архітектура моделі STT DeepSpeech є відкритою та також базується на RNN (рекурентних нейронних мережах). Важлива особливість полягає в тому, що DeepSpeech може працювати в автономному режимі на процесорі, але тоді точність розпізнавання є досить низькою порівняно з сучасними трансформерами. Повідомляється, що затримка складає 300-500 мс на середньому процесорі, і вона дуже придатна для вбудованих систем з низькою продуктивністю з деякими адаптаціями у файлах токенів та граматики для досягнення добрих результатів.

Vosk це «offline-first» API для автоматичного розпізнавання мови, яке підвищує якість розпізнавання голосу порівняно з сервісами, що працюють тільки в хмарі, з миттєвим часом відповіді та низькою затримкою. Оскільки переднавчені моделі були створені в Kaldi та оптимізовані для роботи на CPU та GPU, Vosk має таку низьку затримку менше ніж 200 мс навіть на стандартних процесорах. Це важливо для інтерактивного голосового асистента, щоб зменшити інтервали між промовою користувача та відповіддю системи.

Модель у Voice Helper має велику різноманітність мовних моделей, від англійської до української, а також підтримує десятки інших мов та діалектів. Це робить Voice Helper придатним для багатомовних середовищ і дозволяє узагальнювати застосування без необхідності перенавчання нових мереж. Розмір моделей складає приблизно 100-2000 МБ, що є сприятливим для розгортання системи на пристроях з обмеженою оперативною пам'яттю і дисковим простором. В голосовому помічнику за замовчуванням модель за розміром 90 МБ, проте рекомендується приблизно модель 1000 МБ.

Vosk має зручне API, яке добре інтегрується в проекти на Python у кількох рядках коду за допомогою пакету vosk. Це відповідає загальним цілям проектування Voice Helper, де кількість «шаблонного» коду повинна бути мінімальною, а функції голосового вводу реалізовані та тестовані швидко.

Адаптивність налаштування. Vosk приймає користувацькі граматики та словники, що значно покращує розпізнавання специфічних термінів (наприклад, назви програм, спец-символи або слова з ігор). Ви можете описати список фраз або ключових слів, які слід очікувати, використовуючи формат JSON, і модель автоматично видалить будь-які шуми, зосереджуючись на очікуваних командах, які були озвучені.

Процвітаюча спільнота та чудова документація. Vosk працює на базі Kaldi - інструментарію автоматичного розпізнавання мовлення, і є безліч прикладів для різних мов та випадків використання - від обробки аудіо до інтеграції з ботами Telegram або настільними застосунками. Це значно зменшує "потонання" в налаштуванні та розгортанні і має готові шаблони для негайного початку.

Модель у Voice Helper досягає ідеального балансу між продуктивністю/точністю та конфіденційністю: ваше мовлення розпізнається на локальному комп'ютері, завантаження на сервери не відбувається, воно відповідає навіть найвищим вимогам безпеки, водночас пропонуючи найвищий рівень користувацького досвіду з дуже низькою затримкою.

Таблиця порівняння модулів STT та обґрунтування вибору наведено в таблиці 2.4.

Таблиця 2.4 – Обґрунтування вибору модуля перетворення голосу на текст

Критерій порівняння	Модуль STT			
	Vosk	Whisper (OpenAI)	Google STT	DeepSpeech
1	2	3	4	5
Підтримка офлайн	Так	Частково	Ні	Так
Затримка	< 200 мс	≈ 400 мс	≈ 500 мс	≈ 300 мс
Локальне зберігання	Повне	Обмежене	Немає	Повне
Розмір моделі	50–3000 МБ	~1,3 ГБ	Немає	~500 МБ
Підтримка мов	> 20	> 100	> 120	~20
Простота інтеграції	Дуже висока	Висока	Середня	Висока
Залежність від Інтернету	Ні	Частково	Так	Ні
Налаштування граматики	Так	Ні	Ні	Так

Отже, вибір Vosk для модуля STT в Voice Helper обґрунтований можливістю офлайн-роботи з низькою latency, багатомовністю, простотою інтеграції, гнучким налаштуванням граматик і сильною спільнотою підтримки. Крім того, переднавчені моделі Vosk оптимізовані для ефективної роботи як на CPU, так і на GPU, що забезпечує стабільне розпізнавання навіть на пристроях із обмеженими апаратними ресурсами. Завдяки підтримці користувацьких словників та граматик, система може легко адаптуватися до специфічної термінології й уникати помилок при обробці команд, що містять цифри або незвичні назви.

2.5 Висновки за розділом 2

Голосовий помічник Voice Helper створено за допомогою мови програмування Python. Python є безкоштовним програмним забезпеченням, яке надає розробникам великий простір для маневрування та дає змогу швидко створювати прототипи та розширення програм. Він відомий своєю покращеною читабельністю та багатим набором бібліотек для обробки аудіо, автоматизації ОС і машинного навчання, які є надійною основою для реалізації модулів обробки природної мови, розпізнавання голосу та синтезу мовлення під одним дахом без особливих складнощів.

Для модуля парсингу та генерації відповідей був обраний нейронний сервіс DeepSeek через його безкоштовну ліцензію та багатомільярдний обсяг моделі (приблизно 600 мільярдів параметрів), яка забезпечує високу точність обробки складних запитів у реальному часі. DeepSeek також доступний для локальної та хмарної установки, що забезпечує максимальну конфіденційність даних та відсутність витрат на API-запити, а завдяки великій спільноті модель можна легко налаштувати для власних доменних потреб.

Для перетворення голосу в текст в офлайн-режимі був використаний Vosk, який забезпечує миттєву ідентифікацію мови (<200 мс) без потреби підключення до інтернету. Vosk також пропонує невеликі моделі для сотень мов і може використовувати кастомні граматики, що допомагає покращити точність виконання голосових команд в різних акустичних умовах.

Silero TTS в основному використовується для синтезу мовлення, забезпечуючи високу продуктивність із природною, плавною мовою та працюючи повністю офлайн з невеликими (30-50 МБ) моделями. Цей метод забезпечить безперервну роботу Voice Helper та високу якість звучання незалежно від обставин.

Вибір усіх цих інструментів був обґрунтований і використаний для створення програми, яка повинна мати легку структуру для її подальшого

розвитку. А також, щоб програма мала малі системні вимоги і нею могли користуватися навіть на старому комп'ютері.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення Voice Helper

Структура програмного забезпечення голосового помічника Voice Helper наведено на рис. 3.1

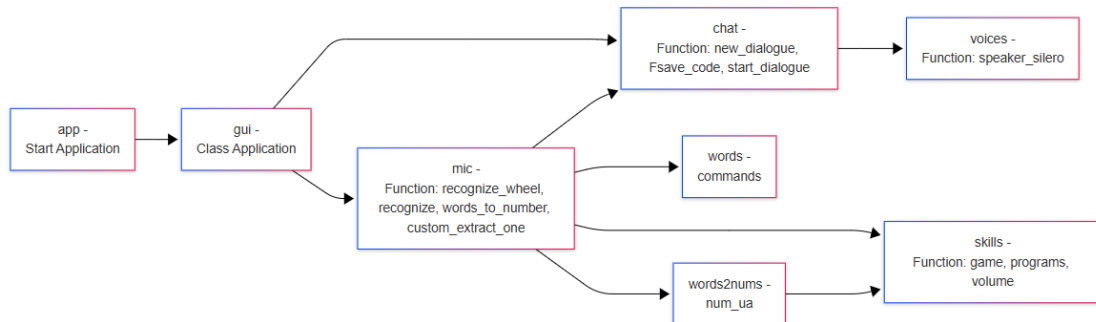


Рисунок 3.1 – Структура програмного забезпечення голосового помічника Voice Helper

Взаємодія вищезгаданих компонентів організована чітко визначеними інтерфейсами, через які отримують доступ модулі системи, і дозволяє виконувати повний цикл запитуваної обробки, подібної до запиту, направленого до завдання (від отримання запиту голосовим асистентом до виконання необхідної дії в системі).

Основний файл, `app.py`, є точкою входу додатку в код. Він запускає GUI, зчитує параметри (яку модель мережі обрати, облікові дані та інші параметри за замовчуванням) та входить в основний цикл обробки подій.

Модуль інтерфейсу `gui.py` - це візуалізація та взаємодія з користувачем. Цей файл має реалізацію кнопок взаємодії з агентом, та можливість легкого подальшого розширення.

Модуль `chat.py` містить логіку для спілкування з нейронною мережею. Його код реалізує в собі очищення діалогу, фільтрування вхідного тексту від пунктуації для коректної роботи синтезу мовлення, записувати введений код у

файл для перегляду та надсилати запит до обраної нейромережі і отримання відповіді.

Модуль `misc.py` використовується для прослуховування аудіо потоку. Він прослуховує мікрофон, якщо були задані команди де є потреба в числах, використовується функція для їх пошуку та перетворення на цифровий варіант, та передає команди для виконання в функціонал агента, або до нейромережі відповідно до заданих параметрів.

Файл команд `skills.py` містить реалізацію команд та коректне відтворення їх в системі комп'ютера. Функціонал команд може бути легко розширений у подальшій розробці.

Словник голосових команд `words.py` відповідає за відповідність між ключовими фразами користувача та ідентифікаторами функцій у `skills.py`. Він дає можливість легко розширювати список команд, не зачіпаючи тіло коду.

Модуль `voices.py` реалізує програмне втілення модулю Silero TTS для озвучування тексту від чат-боту.

Та файл `words2nums.py` має код для коректної роботи функції в `misc.py` для числових перетворень.

В результаті, поділ на окремі модулі надає Voice Helper дуже високий рівень масштабованості та зручності у обслуговуванні: легко додавати нові функціональні можливості або змінювати існуючу логіку без переписування всього коду.

3.2 Функціонування програмного забезпечення Voice Helper

Функціонування програмного забезпечення голосового помічника Voice Helper наведено на зображенні на рис. 3.2.



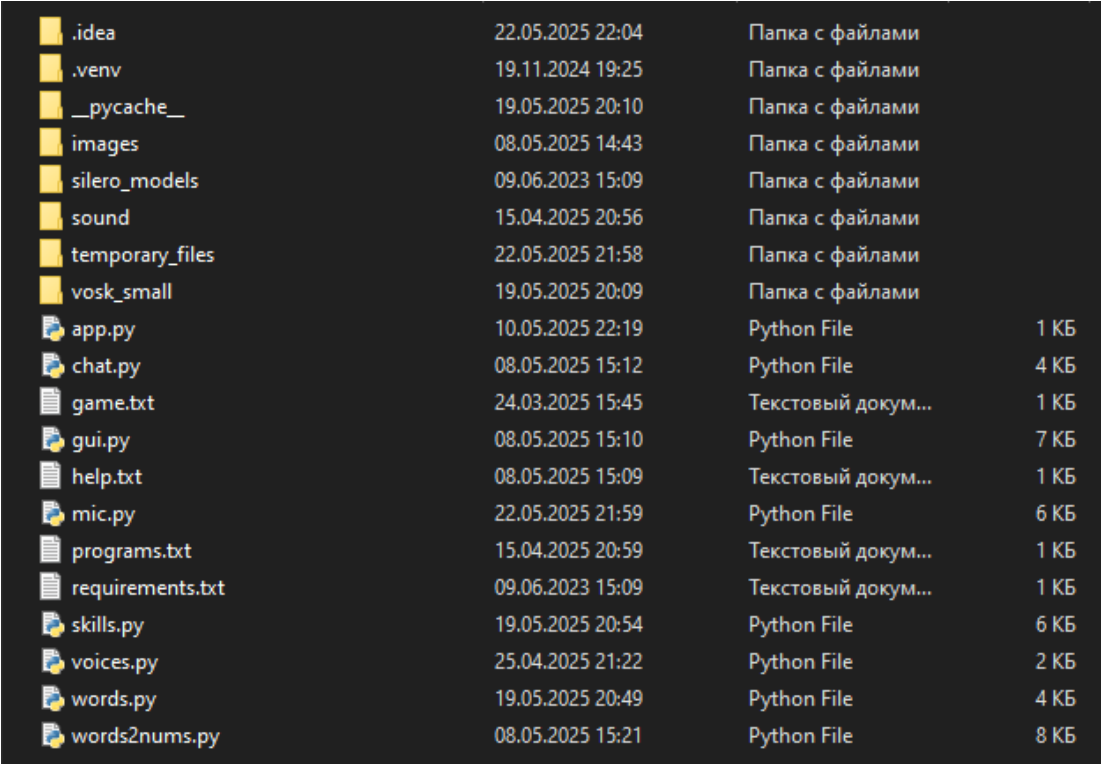
Рисунок 3.2 – Діаграма Функціонування програмного забезпечення голосового помічника Voice Helper

Користувач має можливість озвучування команди як до голосового помічника так і запиту неймережі та повторювати команди та запити за необхідності, або виходу з програми, якщо її функціонал більше не потрібен.

3.3 Розгортання програмного забезпечення голосового помічника Voice Helper

Для розгортання програмного забезпечення голосового помічника Voice Helper потрібен ПК з встановленою Windows 10/11, мовою програмування Python та підключення до інтернету для завантаження моделі Vosk і моделі Silero TTS та зв'язку з неймережею DeepSeek.

Після встановлення моделей та мови Python можна перейти до папки, де знаходяться файли програми (рис 3.3).



.idea	22.05.2025 22:04	Папка с файлами	
.venv	19.11.2024 19:25	Папка с файлами	
__pycache__	19.05.2025 20:10	Папка с файлами	
images	08.05.2025 14:43	Папка с файлами	
silero_models	09.06.2023 15:09	Папка с файлами	
sound	15.04.2025 20:56	Папка с файлами	
temporary_files	22.05.2025 21:58	Папка с файлами	
vosk_small	19.05.2025 20:09	Папка с файлами	
app.py	10.05.2025 22:19	Python File	1 КБ
chat.py	08.05.2025 15:12	Python File	4 КБ
game.txt	24.03.2025 15:45	Текстовый докум...	1 КБ
gui.py	08.05.2025 15:10	Python File	7 КБ
help.txt	08.05.2025 15:09	Текстовый докум...	1 КБ
mic.py	22.05.2025 21:59	Python File	6 КБ
programs.txt	15.04.2025 20:59	Текстовый докум...	1 КБ
requirements.txt	09.06.2023 15:09	Текстовый докум...	1 КБ
skills.py	19.05.2025 20:54	Python File	6 КБ
voices.py	25.04.2025 21:22	Python File	2 КБ
words.py	19.05.2025 20:49	Python File	4 КБ
words2nums.py	08.05.2025 15:21	Python File	8 КБ

Рисунок 3.3. – Файлова структура голосового помічника Voice Helper

Проаналізуємо папку image (рис. 3.4), де знаходяться зображення, які використовуються у програмному забезпеченні голосового помічника Voice Helper (рис. 3.4).

 exit.png	12.02.2025 15:10	Файл "PNG"	2 КБ
 gpt.png	12.02.2025 15:32	Файл "PNG"	2 КБ
 gpt_active.png	28.03.2025 17:35	Файл "PNG"	3 КБ
 help.png	08.05.2025 14:43	Файл "PNG"	2 КБ
 mic.png	09.06.2023 15:09	Файл "PNG"	30 КБ
 mic_active.png	28.03.2025 17:43	Файл "PNG"	3 КБ
 refresh.png	09.06.2023 15:09	Файл "PNG"	4 КБ
 refresh2.png	09.06.2023 15:09	Файл "PNG"	3 КБ

Рисунок 3.4. – Папка зображень для інтерфейсу

У папці звуку знаходяться mp3 файли озвучення команд (рис. 3.5), які виконує програма. Звук вмикається перед командою на 4-6 секунд, після чого програма виконує команду користувача.

 back_1.mp3	08.04.2025 16:15	KMP - MPEG Laye...	35 КБ
 back_2.mp3	08.04.2025 16:16	KMP - MPEG Laye...	37 КБ
 back_3.mp3	08.04.2025 16:16	KMP - MPEG Laye...	42 КБ
 browser_1.mp3	05.04.2025 15:32	KMP - MPEG Laye...	104 КБ
 browser_2.mp3	05.04.2025 15:38	KMP - MPEG Laye...	79 КБ
 browser_3.mp3	10.02.2025 20:30	KMP - MPEG Laye...	28 КБ
 browser_4.mp3	10.02.2025 20:32	KMP - MPEG Laye...	23 КБ
 browser_5.mp3	05.02.2025 16:46	KMP - MPEG Laye...	26 КБ
 close_game_1.mp3	05.04.2025 16:17	KMP - MPEG Laye...	87 КБ
 close_game_2.mp3	08.04.2025 16:02	KMP - MPEG Laye...	72 КБ
 close_game_3.mp3	08.04.2025 16:03	KMP - MPEG Laye...	93 КБ
 close_game_4.mp3	08.04.2025 16:04	KMP - MPEG Laye...	37 КБ
 close_programs_1.mp3	05.04.2025 16:19	KMP - MPEG Laye...	51 КБ
 close_programs_2.mp3	08.04.2025 15:55	KMP - MPEG Laye...	80 КБ

Рисунок 3.5. – Частина папки mp3 файлів

3.4 Проектування інтерфейсу програмного забезпечення Voice Helper

Проектування інтерфейсу взаємодії користувача з програмним забезпеченням виконано з урахуванням потреб при ввикористанні програми. При розробленні програмного забезпечення враховані функціональні вимоги до програми:

- включення\відключення прослуховування мікрофону для виконання команд;

- включення\відключення прослуховування мікрофону для відправки запиту до нейромережі;
- очищення чату с нейромережею ;
- допомога по командам помічника;
- вихід з програми.

Зображення інтерфейс програмного забезпечення Voice Helper наведену на рис. 3.6.



Рисунок 3.6 – Інтерфейс програмного забезпечення Voice Helper

В таблиці 3.1 описаний функціонал інтерфейсу голосового помічника Voice Helper

Таблиця 3.1 – Описання функціоналу інтерфейсу

1	2
	Кнопка мікрофону має 2 режими, в 1 вона не активована та не збирає аудіо потік с мікрофону і нікуди його не передає. 2 режим коли кнопка активована та може передавали голосові команди за рахунок постійного прослуховування мікрофону на предмет ім’я асистента, щоб потім передати команду.
	Кнопка нейромережі як і кнопка мікрофону має 2 режими де в першому випадку вона не активована та не збирає аудіо потік с мікрофону і нікуди його не передає, а в другому випадку вона може бути активною коли і активна кнопка мікрофону. Коли вона активована то увесь потік с мікрофону одразу передається нейромережі, тому після запиту її краще одразу вимикати.

Продовження таблиці 3.1

1	2
 	Кнопка нового діалогу потрібна для того, щоб чат-бот краще зрозумів новий запит, та не мав змоги орієнтуватись за попередніми запитами. Це може бути корисним коли користувач хоче отримати чисту думку про його запит. Вона має 2 режими, 1 коли діалог ще не створений, 2 коли програма виявила новий діалог та показую, що є можливість створити новий.
	Кнопка «Інформація» відкриває діалогове вікно з переліком команд, доступних для виконання помічником.
	Кнопка виходу одразу завершає сеанс роботи з програмою, це найшвидший варіант завершити роботу голосового помічника, якщо користувачу він більше не потрібен.

Кнопка «Інформація» має окремий інтерфейс при її визові, рис. 3.7.

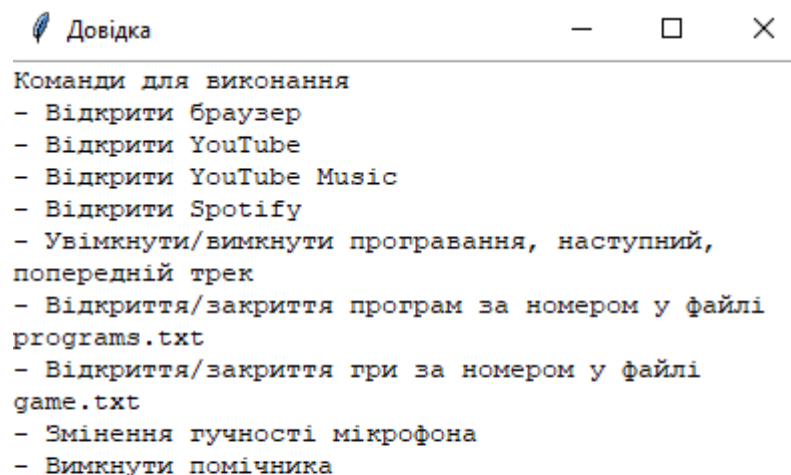


Рисунок 3.7 – Інтерфейс кнопки «Інформація»

3.5 Функції програмного забезпечення Voice Helper

Для задоволення вимог, зазначених у технічному завданні, розроблено такі функції:

- активування та деактивування програмного забезпечення було інтегровано у системні функції;
- додаток дозволяє керувати гучністю пристрою за допомогою голосу;
- у програмному рішенні реалізовано модель нейронної мережі для обробки та виконання текстових запитів користувача. Вона спроектована для можливості подавати запити до нейронної мережі за допомогою голосової команди та має функції для перегляду програмного коду, створеного нейронною мережею;
- голосові команди можуть використовуватись для відкриття музичних сервісів, таких як Spotify і YouTube, а також для відвідування певних веб-ресурсів, таких як сторінки Google та YouTube у веб-браузері;
- програма дозволяє керувати відтворенням аудіо (пауза, відтворення та пропускання попереднього або наступної композиції);
- система підтримує надсилання системних повідомлень при запуску програми або гри, наприклад, інформування користувача без затримки про виконання дії, в якій він зацікавлений.

Для забезпечення функціональності програмного забезпечення були використані численні бібліотеки, включаючи системну бібліотеку для взаємодії з операційною системою [34], та спеціальні модулі, такі як `ctypes`, `comtypes` або `pywin32`, для маніпуляції параметрами звуку. Крім того, бібліотека `Winotify` [35] була інтегрована для реалізації системи сповіщень, що забезпечує комфортне подання інформації, щодо дій в системі, котрі виконує голосовий помічник. Також повідомлення користувач може вимкнути за потреби.

3.6 Висновки за розділом 3

Розроблено програмне забезпечення голосового асистента, яке може керувати комп'ютером за допомогою голосових команд.

Запроєктована архітектура програмного забезпечення голосового асистента формується з блоків розпізнавання, генерації мови, обробки команди та взаємодії з нейронною мережею. Основні компоненти системи включають прості команди: розпізнавання голосу (на основі Vosk), мовлення (з приглушувачем tts), налаштування звуку, запуск та закриття програм, відтворення музики, формування повідомлень тощо.

Функціональність голосового асистента наступна: призупинення/відтворення треків, перемикання треків, відкриття вкладок у браузері (YouTube, Google, Spotify), запити до нейронної мережі з оглядом наданого коду.

Розроблено дизайн інтерфейсу для взаємодії користувача з програмним забезпеченням, включаючи роботу для української мови, а також розроблено систему повідомлень яка інформує про дії голосового помічника.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення та умови застосування програми

Voice Helper - це десктопне програмне забезпечення, призначене для автоматизації взаємодії користувача з операційною системою та зовнішніми сервісами за допомогою голосових команд. Основні сценарії застосування включають:

- hands-free керування ПК: запуск і завершення роботи програм та ігор, відкриття веб-ресурсів (YouTube, Google, Spotify) та вікон браузера;
- автоматизацію мультимедіа: відтворення, призупинення, перемикання треків;
- регулювання системних налаштувань: гучності, сповіщень, профілів аудіо;
- виконання довільних запитів до AI: відтворення відповіді нейромережі, генерація коду з можливістю перегляду;
- оперативне інформування: сповіщення про запуск програм/ігор тощо.

Для експлуатації програмного забезпечення веб-сайту агентства нерухомості необхідно таке апаратне та програмне забезпечення:

- ОС – Windows 10 або Windows 11 (64-біт);
- інтерпретатор Python 3.8+;
- наявність інтернет-з'єднання для первісного завантаження моделей Silero;
- встановлені залежності: vosk, silero, gtts, ctypes, comtypes, pycaw, sounddevice, playsound, winotify тощо;
- мінімальний фоновий шум (≤ 45 дБ) для забезпечення точності офлайн-розпізнавання Vosk;
- підключення до локальної мережі або інтернету під час ініціалізації та за потреби виклику онлайн-сервісів;
- вільний простір на диску – ≈ 4 ГБ.

Функціональні характеристики розробленого програмного забезпечення Voice Helper наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення вебсайту агентства нерухомості наведено у додатку Б.

4.2 Характеристики Voice Helper

Програмне забезпечення Voice Helper написано мовою програмування Python і використовує систему розпізнавання голосу VOSK та систему для синтезу мовлення Silero TTS, потребує інтернету для первісного завантаження моделей Silero та зв'язку з нейромережкою DeepSeek.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програми необхідно встановити Python 3.8+, будь який зручний інтерпретатор мови програмування Python, та ряд бібліотек: vosk, silero, gtts, ctypes, comtypes, pycaw, sounddevice, playsound, winotify тощо.

4.3.2 Вхідні й вихідні дані

Вхідними даними для Voice Helper є аудіопотік із мікрофона користувача, що містить голосові команди

Виконані дії в операційній системі (відкриття програм, керування вікнами тощо), аудіо відповіді на команди, системні сповіщення, а також повідомлення про статус виконання (успішно/з помилкою) та додаткові цифрові звіти (логи виконання команд або текстові результати, збережені в файлах чи відправлені користувачу).

4.3.3 Повідомлення

Користувач програмного забезпечення Voice Helper може отримати такі повідомлення: повідомлення про відкриття програми та її закриття, про наявність невідповідної до запуску програми, про відсутність звукових файлів, повідомлення що до виконаних команд, наприклад як повідомлення про % гучності що було встановлено голосовою командою.

4.4 Виконання програмного забезпечення Voice Helper

Після запуску програмного забезпечення Voice Helper користувачу відображається кнопки взаємодії з програмою (рис. 4.1), де є кнопка мікрофону що активується, або вимикається для прослуховування команд та голосу. Кнопка неймережі яка так само активує, або вимикає прослуховування для неймережі, її активація можлива лише при активній кнопки мікрофону. Кнопка очищення діалогу з неймережею, яка створює новий діалог. Кнопка інформації, що до команд котрі виконує помічник та кнопка виходу вимикає програму.

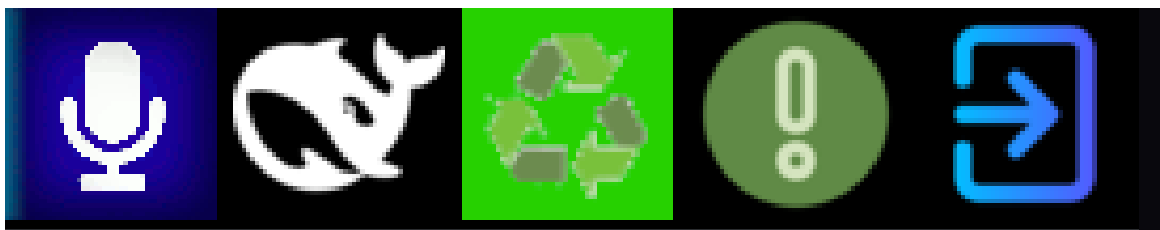


Рисунок 4.1 – Кнопки взаємодії з програмою Voice Helper

4.5 Тестування програмного забезпечення Voice Helper

Виконано тестування програмного забезпечення Voice Helper.

Перевірено такі функціональні вимоги:

- коректне відкриття та завершення роботи програм, зазначених у списку;
- відтворення всіх команд, передбачених у сценаріях користувача;
- надсилання різноманітних запитів до інтегрованої нейромережі та обробка отриманих відповідей;
- відображення згенерованого нейромережею програмного коду у відповідному вікні інтерфейсу;
- бездоганний запуск і відтворення аудіо файлів, асоційованих із конкретними голосовими командами.

Проведено тестування програмного забезпечення голосового помічника Voice Helper. Під час тестування було виявлено, що через помилки в розпізнаванні команд прослуховування аудіопотоку переривалося, і його потрібно було перезапускати вручну, що потребувало багато часу і було незручно. Цю проблему було вирішено за допомогою використання винятків try/except (рис. 4.2)

```

93         try:
94             exec(func_name + '()')
95         except Exception as e:
96             toast = Notification(title=f"Помилка виконання команди {func_name}: {e}",
97                                 app_id='Voice helper',
98                                 )
99             toast.show()
100         except Exception as e:
101             toast = Notification(title=f"Помилка у recognize: {e}",
102                                 app_id='Voice helper',
103                                 )
104             toast.show()

```

Рисунок 4.2 – Використування винятків try/except для обробки команди у файлі mic.py

Проведено тестування виведення повідомлень під час запуску програм, ігор та налаштування гучності, результат наведено на рисунку 4.3 і рисунку 4.4.

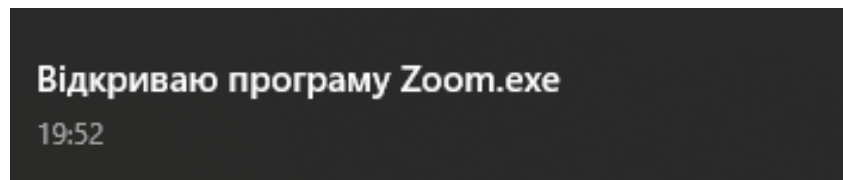


Рисунок 4.3. – Запуск Zoom

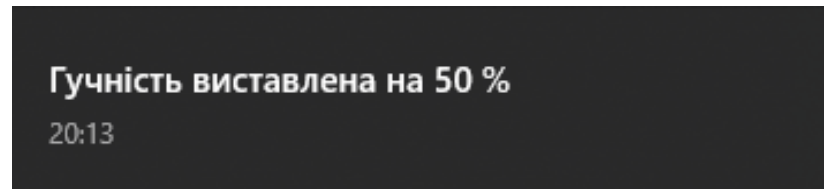


Рисунок 4.4. – Відсоток виставленої гучності

Проведено тестування нейромережі та відображення коду, який було згенеровано за запитом користувача, результат тестування на рисунку 4.5.

```
*code.py - C:\Users\Administrator\Desktop\Voice-Deep B\temporary_files\code.py (3.12.6)*
File Edit Format Run Options Window Help
python
def calculator():
    print("Ласкаво просимо в калькулятор!")

    while True:
        try:
            num1 = float(input("Введіть перше число: "))
            operation = input("Виберіть операцію (+, -, *, /): ")
            num2 = float(input("Введіть друге число: "))

            if operation == '+':
                result = num1 + num2
            elif operation == '-':
                result = num1 - num2
            elif operation == '*':
                result = num1 * num2
            elif operation == '/':
                if num2 == 0:
                    print("Помилка: Ділення на нуль!")
                    continue
                result = num1 / num2
            else:
                print("Неправильна операція. Будь ласка, виберіть із +, -, *, /.")
                continue

            print(f"Результат: {result}")

        except ValueError:
            print("Помилка: введено неправильне значення. Будь ласка, введіть число.")

        again = input("Хочете виконати ще одну операцію? (так/ні): ").lower()
        if again != 'да':
            print("Дякую за використання калькулятора!")
            break

if __name__ == "__main__":
    calculator()
```

Рисунок 4.5. – Демонстрація коду калькулятора від DeepSeek

Після проведення тестування виявлено, що програма Voice Helper функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги

та успішно виконує свою основну задачу взаємодії з користувачем за допомогою голосових запитів.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних повсякденними задачами.

4.6 Висновки за розділом 4

Описано програмне забезпечення Voice Helper. Виконано тестування розробленого програмного забезпечення Voice Helper.

Результати тестування програмного забезпечення Voice Helper показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних повсякденними задачами.

ВИСНОВКИ

У процесі виконання дослідження було детально проаналізовано розвиток настільних голосових асистентів і розглянуто сучасні технологічні фактори в цій галузі. Робота показує, що операційні системи можуть стати більш зручними для користувача, якщо користувач керуватиме системою за допомогою голосу, а додатки, що складаються з модулів ASR, TTS і NLP, стануть простішими і зручнішими. Було розглянуто декілька архітектур які є найбільш популярну на ринку серед існуючих рішень: Microsoft Copilot, Amazon Alexa, NVIDIA ChatRTX, Project G-Assist, після вивчення їхніх переваг і недоліків, і за аналогією було обрано найкращу архітектуру та модулі керування системою для голосового асистента Voice Helper.

Описані функціональні вимоги до програмного забезпечення голосового асистента. Було вирішено, що система повинна мати можливість розпізнавати голосових команд з точністю 90%, надавати локальний синтез мови, застосовувати модульні функції для підвищення функціональності та відповідати вимогам конфіденційності та безпеки.

Обрано мову програмування Python для реалізації програмного забезпечення, підтримка якої відкритим вихідним кодом, коротка крива навчання та велика екосистема бібліотек роблять її придатною для швидкого прототипування, гнучкої розробки і кросплатформенного розвитку. Для обробки запитів до нейромережі використано нейронний сервіс DeepSeek, який має відкриту ліцензію і забезпечує швидке генерування відповідей без стягнення плати за API-запити.

Модуль розпізнавання мовлення автоматично реалізується за допомогою Vosk, може працювати офлайн і обробляти потокове аудіо без інтернету, забезпечуючи негайну відповідь. Генерація мовлення здійснюється за допомогою Silero TTS - невеликих локальних моделей, що гарантують природне звучання без ефекту робота з надзвичайно низькою затримкою.

Нейромережа DeepSeek була обрана для проекту як одна з найбільш кращих варіантів представлених на ринку. Це модель з малою затримкою відповіді, яка має функцію глибокого мислення над задачею. DeepSeek має можливість відправки безкоштовних запитів через ключ API, що дає змогу на постійній основі без фінансових витрат мати доступ до сучасних технологій.

Для взаємодії з найбільш важливою функцією, якою є зв'язок з мовними моделями штучного інтелекту, а саме озвучування відповіді від моделі, було обрано Silero TTS. Цей модуль має великі переваги в зручності його використання та застосування, та в якості звучання голосу. Саме ці переваги стали основними при виборі цієї моделі TTS для голосового помічника Voice Helper.

Запропоновано архітектуру програмного забезпечення, що складається з автономних програм для захоплення аудіо, STT, обробки NLP, TTS і взаємодії з операційною системою. Враховано роботу системи: запуск і закриття додатків, контроль гучності, відтворення і налаштування звукових файлів, розподіл веб-ресурсів, запити до нейронної мережі, а також показ вибіркового вихідного коду та журналів із системними повідомленнями.

Розроблено інтерфейс користувача. Програмне забезпечення було протестоване і відповідало визначеним вимогам, а операції були стабільними під час інтерактивної роботи. Результати тестування підтвердили належну реалізацію всіх функціональних випадків і забезпечили потокове відтворення мовлення та розпізнавання команд у реальному часі.

Отже, розроблений Voice Helper забезпечує ефективну автоматизацію рутинних завдань на ПК, підвищує зручність користування системою та відкриває перспективи подальшого розвитку через інтеграцію додаткових модулів і адаптивне навчання під індивідуальні потреби користувача.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. PyPI. vosk 0.3.45 – Python bindings for Vosk API [Electronic resource]. – Access mode: <https://pypi.org/project/vosk/>.
2. NextMSC. Global Voice Assistant Market Report [Electronic resource]. – Access mode: <https://www.nextmsc.com/report/voice-assistant-market>.
3. LearnPython.com. Python in 2025: Trends and Outlook [Electronic resource]. – Access mode: <https://learnpython.com/blog/python-in-2025/>.
4. AlphaCephei Developers. Vosk: Offline Speech Recognition Toolkit [Electronic resource]. – Access mode: <https://alphacephei.com/vosk/>.
5. HowAIWorks. Exploring the Future of Neural Networks: Innovations and Applications in 2025 [Electronic resource]. – Access mode: <https://howaiworks.net/exploring-the-future-of-neural-networks-innovations-and-applications-in-2025/>.
6. IntNXT. Which Neural Network Trends Are Shaping the AI Landscape in 2025? [Electronic resource]. – Access mode: <https://intnxt.com/blog/which-neural-network-trends-are-shaping-the-ai-landscape-in-2025-intnxt/>.
7. Smart Speak Hub. A Deep Dive into Natural Language Processing in Voice Assistants [Electronic resource]. – Access mode: <https://smartspeakhub.com/a-deep-dive-into-natural-language-processing-in-voice-assistants/>.
8. IRJET. Design and Implementation of Voice Assistant for PC [Electronic resource]. – Access mode: <https://www.irjet.net/archives/V11/i10/IRJET-V11I10108.pdf>.
9. ACM Digital Library. Voice Assistants on Windows: A Comprehensive Study [Electronic resource]. – Access mode: <https://dl.acm.org/doi/fullHtml/10.1145/3675888.3676106>.
10. Microsoft. Getting Started with Copilot on Windows [Electronic resource]. – Access mode: <https://support.microsoft.com/en-us/topic/getting-started-with-copilot-on-windows-1159c61f-86c3-4755-bf83-7fbff7e0982d>.

11. Microsoft. Microsoft Copilot for Individuals [Electronic resource]. – Access mode: <https://www.microsoft.com/uk-ua/microsoft-copilot/for-individuals?form=MA13YT&OCID=MA13YT>.
12. PCMag. You Can Now Get Amazon Alexa on Your Windows 10 PC [Electronic resource]. – Access mode: <https://www.pcmag.com/news/you-can-now-get-amazon-alexa-on-your-windows-10-pc>.
13. ASUS. Microsoft Copilot and Smart Windows Apps on AI-Powered ASUS Laptops [Electronic resource]. – Access mode: <https://www.asus.com/content/microsoft-copilot-and-smart-windows-apps-on-ai-powered-asus-laptops/>.
14. Amazon. Alexa Show Mode for Windows [Electronic resource]. – Access mode: <https://developer.amazon.com/en-US/blogs/alexa/device-makers/2021/05/amazon-launches-alexa-show-mode>.
15. Amazon. Amazon Alexa on Windows [Electronic resource]. – Access mode: <https://www.amazon.co.uk/b?ie=UTF8&node=16052424031>.
16. NVIDIA. NVIDIA ChatRTX [Electronic resource]. – Access mode: <https://www.nvidia.com/en-us/ai-on-rtx/chatrtx>.
17. NVIDIA Customer Care. NVIDIA ChatRTX User Guide [Electronic resource]. – Access mode: https://nvidia.custhelp.com/app/answers/detail/a_id/5542/~nvidia-chatrtx-user-guide.
18. NVIDIA. NVIDIA App: G-Assist [Electronic resource]. – Access mode: <https://www.nvidia.com/en-us/software/nvidia-app/g-assist>.
19. Python Software Foundation. Python Official Website [Electronic resource]. – Access mode: <https://www.python.org>.
20. Leveraging Python in AI and Machine Learning: A Survey of Techniques and Educational Approaches in Software Engineering / Castro, O., Bruneau, P., Sottet, J.-S., Torregrossa, D. // Proceedings of the 2023 International Conference on Software Engineering Education, 12–14 березня 2023 р. – Т. 1. – С. 45–58.

21. Lutz, M. Learning Python: Powerful Object-Oriented Programming / M. Lutz. – 5-е вид. – Sebastopol: O'Reilly Media, 2013. – 1 648 p.
22. A Review on Python for Data Science, Machine Learning and IoT / Khalid, M., Zhang, Q., Liu, H., Wang, S. // Journal of Data Science and Internet of Things. – 2022. – Т. 5, № 2. – С. 102–118.
23. Bloch, J. Effective Java: Third Edition / Joshua Bloch. – Boston: Addison-Wesley, 2017. – 416 p.
24. Skeet, J. C# in Depth / Jon Skeet. – 4-е вид. – Shelter Island: Manning Publications, 2019. – 528 p.
25. Субботін, С. О. Нейронні мережі: теорія та практика : навч. посіб. / С. О. Субботін. – Житомир: Вид-во О. О. Євенок, 2020. – 184 с.
26. DeepSeek Inc. DeepSeek Official Website [Electronic resource]. – Access mode: <https://www.deepseek.com>.
27. Deep Learning / Goodfellow, I., Bengio, Y., Courville, A. – Cambridge, MA: MIT Press, 2016. – 775 p.
28. Upubscience. DeepSeek Large-Scale Model Overview [Electronic resource]. – Access mode: <http://www.upubscience.com/News11Detail.aspx?id=1115&proid=38>.
29. TechRxiv. DeepSeek Models: A Comprehensive Survey of Methods and Applications [Electronic resource]. – Access mode: <https://www.techrxiv.org/doi/full/10.36227/techrxiv.174198511.15158242>.
30. Taylor, P. Text-to-Speech Synthesis / Paul Taylor. – Cambridge: Cambridge University Press, 2009. – 402 p.
31. Nwakanma, C. Text-To-Speech Synthesis (TTS) / Cosmas Nwakanma // Proceedings of the 2017 International Conference on Speech Technology. – Los Alamitos: IEEE, 2017. – 42 p.
32. Rabiner, L. Fundamentals of Speech Recognition / L. Rabiner, B.-H. Juang. – Englewood Cliffs: Prentice Hall, 1993. – 560 p.

33. Soni, A. A. Improving Speech Recognition Accuracy Using Custom Language Models with the Vosk Toolkit / Aniket Abhishek Soni // arXiv preprint arXiv:2503.21025, 26 March 2025. – 10 p.

34. Python Software Foundation. Python Documentation: os Module [Electronic resource]. – Access mode: <https://docs.python.org/uk/3.13/library/os.html>.

35. PyPI. winotify 0.1.0 – Windows 10 Toast Notifications [Electronic resource]. – Access mode: <https://pypi.org/project/winotify/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для голосового управління системою.

A.1 Підстава для розробки

Розробка програмного забезпечення ґрунтується на наступних документах:

- наказ по університету про призначення керівників та затвердження тем кваліфікаційних робіт № 209 від 28 квітня 2025 р;
- тема розробки: «Програмне забезпечення голосового помічника для роботи з сервісами штучного інтелекту».

Ці документи створюють базу для роботи, підкреслюючи її важливість у навчальному процесі.

A.2 Призначення розробки

Програмний продукт призначений для для голосового управління системою комп'ютера.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення Voice Helper повинно забезпечувати виконання таких функцій:

- розпізнавання голосових команд (відкриття/закриття програм, запуск ігор, відкриття вебресурсів – YouTube, Google, Spotify);

- керування мультимедіа (відтворення, призупинення, відновлення та перемикання аудіотреків);
- регулювання рівня гучності системи за допомогою голосу;
- інтеграція з нейросервісом DeepSeek для обробки довільних запитів і озвучення згенерованих відповідей;
- відображення у графічному інтерфейсі згенерованого нейромережею програмного коду;
- відправлення системних сповіщень при запуску програм або ігор.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс Voice Helper має бути зручним і мінімалістичним. Повинна бути реалізована панель де будуть знаходитись мінімально необхідні кнопки для взаємодії з голосовим помічником.

A.3.3 Вимоги до надійності

Програмне забезпечення повинно забезпечувати стабільну роботу у режимі реального часу без втрат голосових команд

A.3.4 Умови експлуатації

Для роботи Voice Helper необхідні: персональний комп'ютер із встановленою ОС Windows 10/11, підключений мікрофон, а також інтернет-з'єднання для моделей звуку та нейромереж.

A.3.5 Вимоги до складу та параметрів технічних засобів

Для експлуатації програмного Voice Helper необхідно таке апаратне та програмне забезпечення:

- ОС Windows 10 або 11 (64-біт);
- процесор із тактовою частотою від 2 ГГц;
- оперативна пам'ять не менше 4 ГБ (рекомендовано від 8 ГБ);
- вільний простір на диску — не менше 4 ГБ для зберігання моделей Vosk і Silero;
- необов'язково: графічний прискорювач GeForce RTX (≥ 3 ГБ VRAM).

A.3.6 Вимоги до маркування і пакування

Програма Voice Helper і може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Голосовий помічник Voice Helper».

A.4 Вхідні дані до роботи

Вхідними даними Voice Helper є: аудіопотік із мікрофона та запити користувача.

Вихідними даними є: виконані системні дії (відкриття/закриття програм, регулювання гучності, відкриття вебресурсів тощо); синтезована мовна відповідь (Silero TTS або gTTS); системні сповіщення; відображення згенерованого коду.

ДОДАТОК Б
Фрагмент тексту програми

```

try:
    hallo = [f"{'sound/'}{'back'}_{i}.mp3" for i in range(1,
6)]
    ex = [file for file in hallo if os.path.exists(file)]
    audio = random.choice(ex)
    playsound(audio)
except:
    pass

SETTINGS = {
    'MODEL': 'deepseek/deepseek-chat:free',
    'OPENAI_API_KEY': 'sk-or-v1-
300f9ce777e8a2578681551103f6de9a1ea4dd63e99c9cccd92171bb60fd
2df2',
    'CHATGPT': '0',
    'MIC': '0',
    'NEW_DIALOGUE': '1',
}

for key, value in SETTINGS.items():
    os.environ[key] = value

if __name__ == '__main__':
    root = Application()
    root.mainloop()

class Application(tk.Tk):

    def __init__(self):
        tk.Tk.__init__(self)

        self.overrideRedirect(True)
        self.attributes('-topmost', True)
        self.resizable(False, False)

```

```

        self.prepare_img()
        self.create_widgets()
        self.update()

self.geometry('+{}+{}'.format(self.wininfo_screenwidth() -
self.wininfo_width(),

self.wininfo_screenheight() - 100))

        self.check_env_vars()

    def create_widgets(self):
        self.assistant_label = ttk.Label(self,
            image=self.assistant_img_frames[0], background='black')
        self.assistant_label.grid(column=0, row=0, ipadx=0,
            ipady=0, padx=0, pady=0)
        self.assistant_label.bind("<Button-1>",
            self.run_assistant)

        self.gpt_label = ttk.Label(self,
            image=self.gpt_img_frames[0], background='black')
        self.gpt_label.grid(column=1, row=0, ipadx=0,
            ipady=0, padx=0, pady=0)
        self.gpt_label.bind("<Button-1>", self.run_gpt)

        self.gpt_clear_label = ttk.Label(self,
            image=self.refresh_img, background='black')
        self.gpt_clear_label.grid(column=2, row=0, ipadx=0,
            ipady=0, padx=0, pady=0)
        self.gpt_clear_label.bind("<Button-1>",
            self.clear_gpt)

        self.help_label = ttk.Label(self,
            image=self.help_img, background='black')
        self.help_label.grid(column=3, row=0, ipadx=0,

```

```

ipady=0, padx=0, pady=0)
    self.help_label.bind("<Button-1>", self.show_help)

    self.exit_label = ttk.Label(self,
image=self.exit_img, background='black')
    self.exit_label.grid(column=4, row=0, ipadx=0,
ipady=0, padx=0, pady=0)
    self.exit_label.bind("<Button-1>", self.exit)

def read_gif_frames(self, gif):
    frames = []
    for frame in range(0, gif.n_frames):
        gif.seek(frame)
        frames.append(ImageTk.PhotoImage(gif.copy()))
    return frames

def prepare_img(self):
    self.assistant_img_frames =
[tk.PhotoImage(file="images/mic.png")]
    self.assistant_active_img =
tk.PhotoImage(file="images/mic_active.png")

    self.gpt_img_frames =
[tk.PhotoImage(file="images/gpt.png")]
    self.gpt_active_img =
tk.PhotoImage(file="images/gpt_active.png")

    self.help_img =
tk.PhotoImage(file="images/help.png")

    self.exit_img =
tk.PhotoImage(file="images/exit.png")

    self.refresh_img =
tk.PhotoImage(file="images/refresh2.png")
    self.refresh2_img =

```

```

tk.PhotoImage(file="images/refresh.png")

    def animate_mic(self, frame_index=0):

self.assistant_label.configure(image=self.assistant_active_i
mg)

    def animate_gpt(self, frame_index=0):
        self.gpt_label.configure(image=self.gpt_active_img)

    def run_assistant(self, event=None):
        if int(os.getenv('MIC')):
            os.environ.update(MIC='0')
            self.stop_mic_animation()
            self.assistant_label['image'] =
self.assistant_img_frames[0]

            if int(os.getenv('CHATGPT')):
                os.environ.update(CHATGPT='0')
                self.stop_gpt_animation()
                self.gpt_label['image'] =
self.gpt_img_frames[0]

        else:
            os.environ.update(MIC='1')
            self.animate_mic() # Устанавливаем активное
изображение для микрофона
            p1 = Thread(target=recognize_wheel, daemon=True)
            p1.start()

    def check_env_vars(self):
        dialogue_status = os.environ.get('NEW_DIALOGUE')
        if not int(dialogue_status):
            self.gpt_clear_label['image'] =
self.refresh2_img

```

```

        self.after(5000, self.check_env_vars)

def run_gpt(self, event):
    if int(os.getenv('CHATGPT')):
        os.environ.update(CHATGPT='0')
        self.stop_gpt_animation()
        self.gpt_label['image'] = self.gpt_img_frames[0]
    elif not int(os.getenv('MIC')):
        return
    else:
        os.environ.update(CHATGPT='1')
        self.animate_gpt() # Устанавливаем активное
изображение для GPT

def clear_gpt(self, event):
    new_dialogue()
    os.environ.update(NEW_DIALOGUE='1')
    self.gpt_clear_label['image'] = self.refresh_img

def show_help(self, event):
    try:
        base_dir =
os.path.dirname(os.path.abspath(sys.argv[0]))
        help_path = os.path.join(base_dir, "help.txt")
        with open(help_path, "r", encoding="utf-8") as
f:

            content = f.read()

        win = tk.Toplevel(self)
        win.title("Довідка")
        win.geometry("400x300")

        text = tk.Text(win, wrap='word')
        text.insert('1.0', content)
        text.config(state='disabled')
        text.pack(fill='both', expand=True)

```

```

        except Exception as e:
            print("Помилка", f"Не вдалося відкрити
help.txt:\n{e}")

    def exit(self, event):
        sys.exit(0)

def recognize(data):

    if len(data) < 7:
        return
    trg = words.NAMES.intersection(data.split())
    if not trg:
        if not int(os.getenv("CHATGPT")):
            return
        try:
            voices.speaker_silero(chat.start_dialogue(data))
        except:
            pass
        return
    data = data.split()
    filtered_data = [word for word in data if word not in
words.NAMES]
    data = ' '.join(filtered_data)

    match, confidence = custom_extract_one(data,
list(words.commands.keys()))
    if confidence < 20: # Порог схожести
        toast = Notification(title=f'Команда не була
розпізнана, повторіть її',
                                app_id='Voice helper',
                                )
        toast.show()

    answer = words.commands[match]
    func_name = answer.split()[0]

```

```

try:
    files = [f"{\"sound/\"}{func_name}_{i}.mp3\" for i in
range(1, 6)]
    existing_files = [file for file in files if
os.path.exists(file)]

    if not existing_files:
        raise FileNotFoundError(f"Немає файлів для
команди {func_name}")

    audio_file = random.choice(existing_files)

    playsound(audio_file)
except Exception as e:
    toast = Notification(title=f"Помилка звукового
файлу: {e}",
                        app_id='Voice helper',
                        )
    toast.show()

if func_name == "volume":
    volume(words_to_number(data))
elif func_name == "game":
    game(words_to_number(data))
elif func_name == "programs":
    programs(words_to_number(data))
elif func_name == "close_game":
    close_game(words_to_number(data))
elif func_name == "close_programs":
    close_programs(words_to_number(data))
else:
    try:
        exec(func_name + '()')
    except Exception as e:
        toast = Notification(title=f"Помилка виконання

```

```

команди {func_name}: {e}",
                                app_id='Voice helper',
                                )
        toast.show()
    except Exception as e:
        toast = Notification(title=f"Помилка у
recognize: {e}",
                                app_id='Voice helper',
                                )
        toast.show()

def custom_extract_one(data, dataset):

    best_choice = None
    best_score = 0
    query_tokens = data.split()
    num_query_tokens = len(query_tokens)

    for candidate in dataset:
        candidate_tokens = candidate.split()
        num_candidate_tokens = len(candidate_tokens)
        raw_score = fuzz.ratio(candidate, data)
        # Штрафуем за наличие лишних слов в запросе
        penalty = num_candidate_tokens / num_query_tokens
        score = raw_score * penalty

        if score > best_score:
            best_score = score
            best_choice = candidate

    return best_choice, best_score

def start_dialogue(text):
    try:
        if int(os.getenv('NEW_DIALOGUE', '0')):
            os.environ.update(NEW_DIALOGUE='0')

```

```

api_key = os.getenv('OPENAI_API_KEY')
headers = {
    "Authorization": f"Bearer {api_key}",
    "Content-Type": "application/json",
}

data = {
    "model": os.getenv('MODEL'),
    "messages": [{"role": "user", "content": text}],
    "stream": True
}

with requests.post(
    "https://openrouter.ai/api/v1/chat/completions",
    headers=headers,
    json=data,
    stream=True,
) as response:
    if response.status_code != 200:
        toast = Notification(title=f"Ошибка API:
{response.status_code}",
                                app_id='Voice helper',
                                )
        toast.show()
        return ""

    full_response = []

    for chunk in response.iter_lines():
        if chunk:
            chunk_str = chunk.decode('utf-
8').replace('data: ', '')
            try:
                chunk_json = json.loads(chunk_str)

```

```

        if "choices" in chunk_json:
            content =
chunk_json["choices"][0]["delta"].get("content", "")
            if content:
                cleaned =
process_content(content)
                print(cleaned, end='',
flush=True)

full_response.append(cleaned)
            except Exception:
                pass

        print() # Перенос строки после завершения
потока

        return check_response(''.join(full_response))
    except Exception as e:
        toast = Notification(title=f'Помилка: {str(e)}',
                             app_id='Voice helper',
                             )
        toast.show()

def read_paths(filename: str, line_number: int):
    file_op = Notification(title='Некорректный номер
строки',
                             app_id='Voice helper',
                             )

    try:
        with open(filename, 'r', encoding='utf-8') as file:
            lines = file.readlines()
            if line_number <= 0 or line_number > len(lines):
                raise file_op.show()

            # Получаем нужную строку и отсекаем по первому
пробелу
            path = lines[line_number - 1].split(' ',

```



```

        os.system(f'taskkill /f /im {file_name}')
    except Exception as e:
        toast = Notification(title=f'Виникла помилка під час
закриття гри {file_name}',
                                app_id='Voice helper',
                                )
        toast.show()

def programs(number):
    try:
        full_path, file_name = read_paths('programs.txt',
number)
        toast = Notification(title=f'Відкриваю програму
{file_name}',
                                app_id='Voice helper',
                                )
        toast.show()
        os.system(full_path)
    except Exception as e:
        toast = Notification(title=f'Виникла помилка під час
запуску програми {file_name}',
                                app_id='Voice helper',
                                )
        toast.show()

def close_programs(number):
    try:
        full_path, file_name = read_paths('programs.txt',
number)
        os.path.exists(file_name)
        toast = Notification(title=f'Закриваю програму
{file_name}',
                                app_id='Voice helper',
                                )

```

```

        toast.show()
        os.system(f'taskkill /f /im {file_name}')
    except Exception as e:
        pass
        toast = Notification(title=f'Виникла помилка під час
закриття програми {file_name}',
                                app_id='Voice helper',
                                )
        toast.show()

def volume(volume_level):
    try:
        devices = AudioUtilities.GetSpeakers()
        interface =
devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL,
None)
        volume = cast(interface,
POINTER(IAudioEndpointVolume))
        volume.SetMasterVolumeLevelScalar(volume_level /
100, None)
        toast = Notification(title=f'Гучність виставлена на
{volume_level} %',
                                app_id='Voice helper',
                                )
        toast.show()
    except:
        toast = Notification(title='Під час виклику команди
сталася помилка, будь ласка, повторіть запит',
                                app_id='Voice helper',
                                )
        toast.show()

def browser():
    webbrowser.open("https://www.google.com.ua/?hl=ru",
new=2)

```

```

def youTUBE():
    webbrowser.open("https://www.youtube.com", new=2)

def offpc():
    # Эта команда отключает ПК под управлением Windows.
    Временно приостановленна
    # os.system('shutdown \s')
    return

def offBot():
    '''Отключает бота'''
    os._exit(0)

def open_spotify():
    webbrowser.open("https://open.spotify.com/")

def open_youtube_music():
    webbrowser.open("https://music.youtube.com/")

def play_pause():
    keyboard.press_and_release('play/pause media')

def next_track():
    keyboard.send("next track")

def prev_track():
    keyboard.send('previous track')

NAMES = {'помічник'}

```

```

commands = {

    'запусти браузер': 'browser',
    'відкрий браузер': 'browser',
    'відкрий ютуб': 'youTUBE',
    'мені потрібен ютуб': 'youTUBE',
    'ютуб відкрий': 'youTUBE',
    'відкрий мені Ютуб': 'youTUBE',
    'вимкни комп'ютер': 'offpc',
    'заверши роботу': 'offBot',
    'припини слухати': 'offBot',
    'увімкни гучність': "volume",
    'встанови гучність': "volume",
    'налаштуй гучність': "volume",
    'гучність': "volume",
    'відкрий гру': 'game',
    'пограємо': 'game',
    'запусти мені гру': 'game',
    'відкрий гру номер': 'game',
    'відкрий програму': 'programs',
    'увімкни мені програму': 'programs',
    'відкрий програму номер': 'programs',
    'закрий гру номер': 'close_game',
    'вимкни гру номер': 'close_game',
    'закрий гру': 'close_game',
    'вимкни гру': 'close_game',
    'вимкни програму': 'close_programs',
    'закрий програму': 'close_programs',
    'вимкни програму номер': 'close_programs',
    'закрий програму номер': 'close_programs',
    'відкрий ютуб музику': 'open_youtube_music',
    'відкрий музику з ютуба': 'open_youtube_music',
    'відкрий ютуб мьюзик': 'open_youtube_music',
    'відкрий спотіфай': 'open_spotify',
    'музика спотіфай': 'open_spotify',
    'відкрий музику зі спотіфай': 'open_spotify',

```

```
'зупини музику': 'play_pause',  
'вимкни музику': 'play_pause',  
'постав на паузу': 'play_pause',  
'постав музику на паузу': 'play_pause',  
'наступний трек': 'next_track',  
'перемкни на наступний': 'next_track',  
'увімкни інший трек': 'next_track',  
'попередній трек': 'prev_track',  
'поверни попередній трек': 'prev_track',  
'поверни трек назад': 'prev_track',  
'поверни назад': 'prev_track',  
'увімкни музику': 'play_pause',  
'запусти музику': 'play_pause',  
'увімкни трек': 'play_pause',  
'запусти трек': 'play_pause',
```

```
}
```

ДОДАТОК В
Слайди презентації

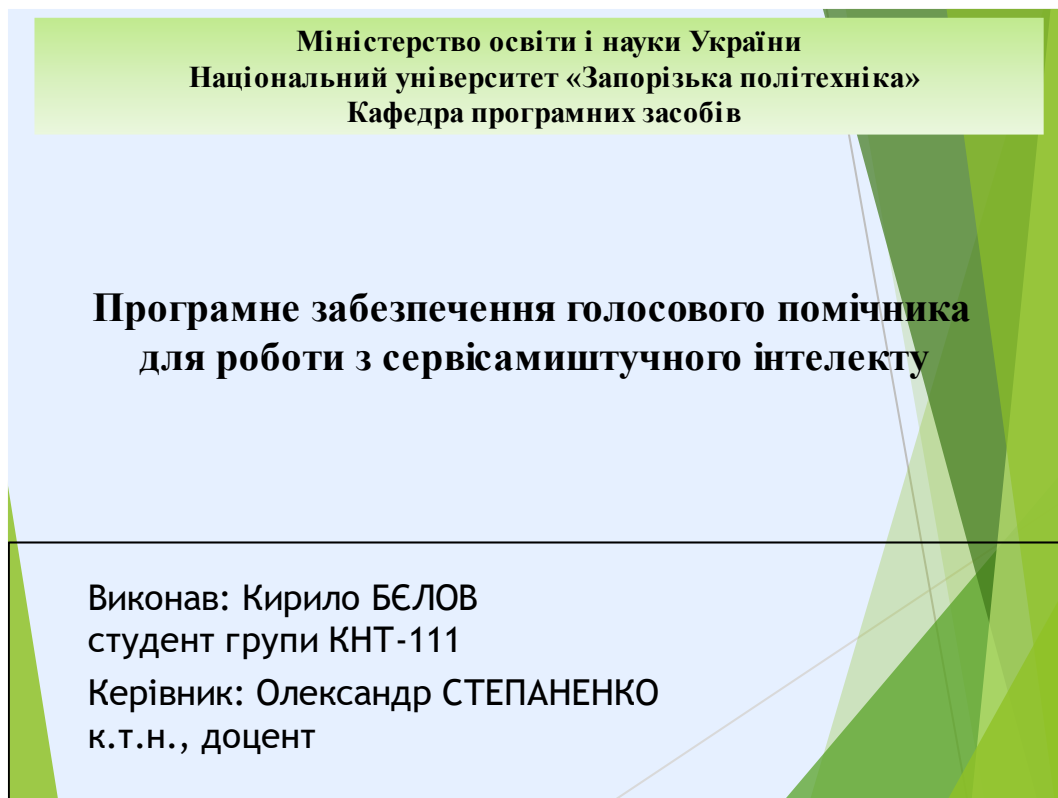


Рисунок В.1 – Слайд 1

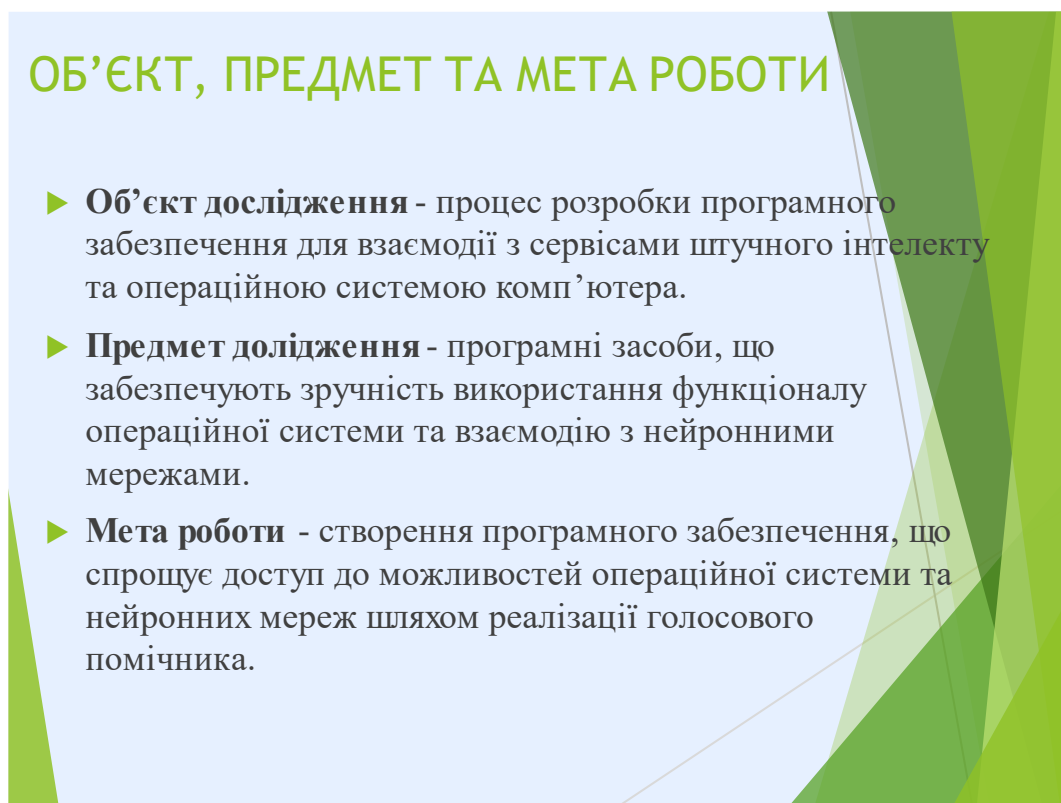


Рисунок В.2 – Слайд 2

ЗАВДАННЯ РОБОТИ

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для голосового помічника;
- здійснити проєктування програмного забезпечення для голосового помічника;
- створити програмне забезпечення для голосового помічника;
- виконати тестування розробленого програмного забезпечення для голосового помічника.

Рисунок В.3 – Слайд 3

ПОРІВНЯННЯ АНАЛОГІВ ГОЛОСОВИХ ПОМІЧНИКІВ

Критерії	Microsoft Copilot	Amazon Alexa	NVIDIA ChatRTX	Project G Assist
Синтез мовлення (TTS)	Azure Neural Voices	AVS TTS	Silero TTS локально	NVIDIA Neural Voices
Кастомізація	Обмежена	Обмежена	Висока	Середня
Основні сценарії	Office, пошук, файли	«Розумний дім», медіа	Аналіз локальних даних	Оптимізація ігор/системи
Контекстна підтримка	Так	Ні	Обмежена	Так
Інтеграція з ОС	PowerShell, Graph API	Обмежена	Системні журнали	NVIDIA App overlay
Вартість	Вбудовано / Pro-версія	Платна версія	Безкоштовно	Безкоштовно

Рисунок В.4 – Слайд 4

Вибір мови програмування

Критерій порівняння	Python	Java	C#
Широке застосування в AI/ML	+	+ -	+ -
Крос-платформеність	+	+	-
Наявність бібліотек для аудіо та NLP	+	-	+ -
Простота синтаксису	+	+ -	+ -
Швидкість розробки та прототипування	+	-	-
Підтримка спільнотою та документація	+	+ -	+ -

Рисунок В.5 – Слайд 5

ВИМОГИ ДО ПРОГРАМИ

Програмне забезпечення Voice Helper повинно забезпечувати виконання таких функцій:

- розпізнавання голосових команд (відкриття/закриття програм, запуск ігор, відкриття вебресурсів - YouTube, Google, Spotify);
- керування мультимедіа (відтворення, призупинення, відновлення та перемикання аудіотреків);
- регулювання рівня гучності системи за допомогою голосу
- інтеграція з нейросервісом DeepSeek для обробки довільних запитів і озвучення згенерованих відповідей;
- відображення у графічному інтерфейсі згенерованого нейромережею програмного коду;
- відправлення системних сповіщень при запуску програм або ігор.

Рисунок В.6 – Слайд 6

ПОРІВНЯННЯ НЕЙРОСЕРВІСІВ

Критерій порівняння	DeepSeek	OpenAI GPT-4	GooglePaLM
Розмір моделі	600 млрд	175 млрд	540 млрд
Затримка відповіді	низька (< 300 мс)	середня (≈ 600 мс)	Середня (≈ 500 мс)
Спільнота та підтримка	висока	Спільнота та підтримка	висока
Інтеграція з голосовими інтерфейсами	через API або seamless	Інтеграція з голосовими інтерфейсами	через API або seamless
Вартість запитів	безкоштовно	Вартість запитів	безкоштовно
Можливість кастомізації	висока	Можливість кастомізації	висока

Рисунок В.7 – Слайд 7

ПОРІВНЯННЯ МОДУЛЮ TTS

Модуль	Silero TTS	Coqui TTS (XTTSv2)	Google Cloud TTS	Amazon Polly	gTTS (Google TTS)
Оффлайн-підтримка	Так	Так	Ні	Ні	Ні
Вартість	безкоштовно	безкоштовно	платно	платно	безкоштовно
Мови	5	> 16 мов	40+ мов, 220+ голосів	40+ мов, 100+ голосів	40+ мов
Затримка	< 200 мс	< 200 мс	≈ 500 мс	≈ 300 – 600 мс	≈ 400 – 700 мс
Якість синтезу	Природне, плавне	Висока	Висока	Висока	Добра

Рисунок В.8 – Слайд 8

ПОРІВНЯННЯ МОДУЛЮ STT

Критерій	Vosk	Whisper (OpenAI)	Google STT	DeepSpeech
Підтримка офлайн	Так	Частково	Ні	Так
Затримка	< 200 мс	≈ 400 мс	≈ 500 мс	≈ 300 мс
Розмір моделі	50–3000 МБ	~1,3 ГБ	Немає, хмарне зберігання	~500 МБ
Підтримка мов	> 20	> 100	> 120	~20
Простота інтеграції в Python	Дуже висока	Висока	Середня	Висока
Локальне зберігання	Повне	Обмежене	Немає	Повне
Залежність від Інтернету	Ні	Частково	Так	Ні

Рисунок В.9 – Слайд 9

ЗАГАЛЬНА СХЕМА ПРОГРАМИ

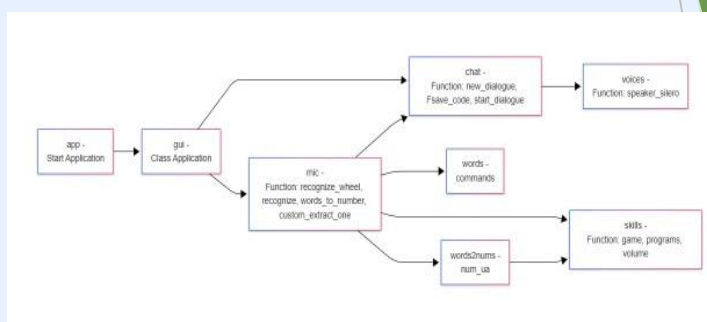


Рисунок В.10 – Слайд 10

СХЕМА ФУНКЦІОНУВАННЯ ГОЛОСОВОГО ПОМІЧНИКА

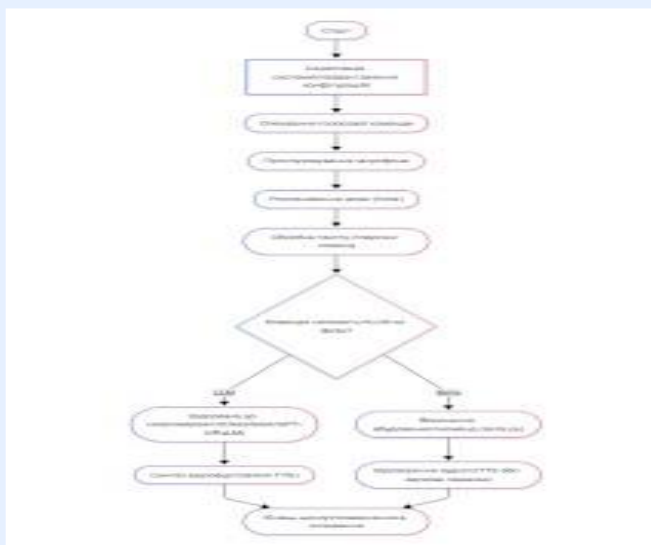


Рисунок В.11 – Слайд 11

ІНТЕРФЕЙС ЗАСТОСУНКУ



```

C:\codepy - C:\Users\Administrator\Desktop\Voice-Deep-Bitentropy\Netcode.py 0.128
File Edit Format Run Options Window Help
python
def calculator():
    print("Введіть перше число:")
    num1 = float(input())
    print("Введіть операцію (+, -, *, /): ")
    operation = input()
    num2 = float(input("Введіть друге число: "))

    while True:
        if operation == '+':
            result = num1 + num2
        elif operation == '-':
            result = num1 - num2
        elif operation == '*':
            result = num1 * num2
        elif operation == '/':
            result = num1 / num2
        elif operation == 'x':
            print("Помилка: Ділення на нуль!")
            continue
        elif operation == 'q':
            result = num1 / num2
        else:
            print("Неправильна операція. Будь ласка, виберіть із +, -, *, /.")
            continue

        print("Результат: {result}")

    while True:
        print("Введіть ще одну операцію (стан/ні): ").lower()
        again = input()
        if again == 'да':
            print("Дякуємо за використання калькулятора!")
            calculator()
        else:
            break

if __name__ == '__main__':
    calculator()
  
```

Додаток

Команди для виконання:

- Відкрити браузер
- Відкрити YouTube
- Відкрити YouTube Music
- Відкрити Spotify
- Увімкнути/вимкнути програма, доступний, попередній трек
- Відкрити/закрити програм за номером у файлі programs.txt
- Відкрити/закрити при за номером у файлі game.txt
- Знімання даних мікрофона
- Вимкнути помічника

Рисунок В.12 – Слайд 12

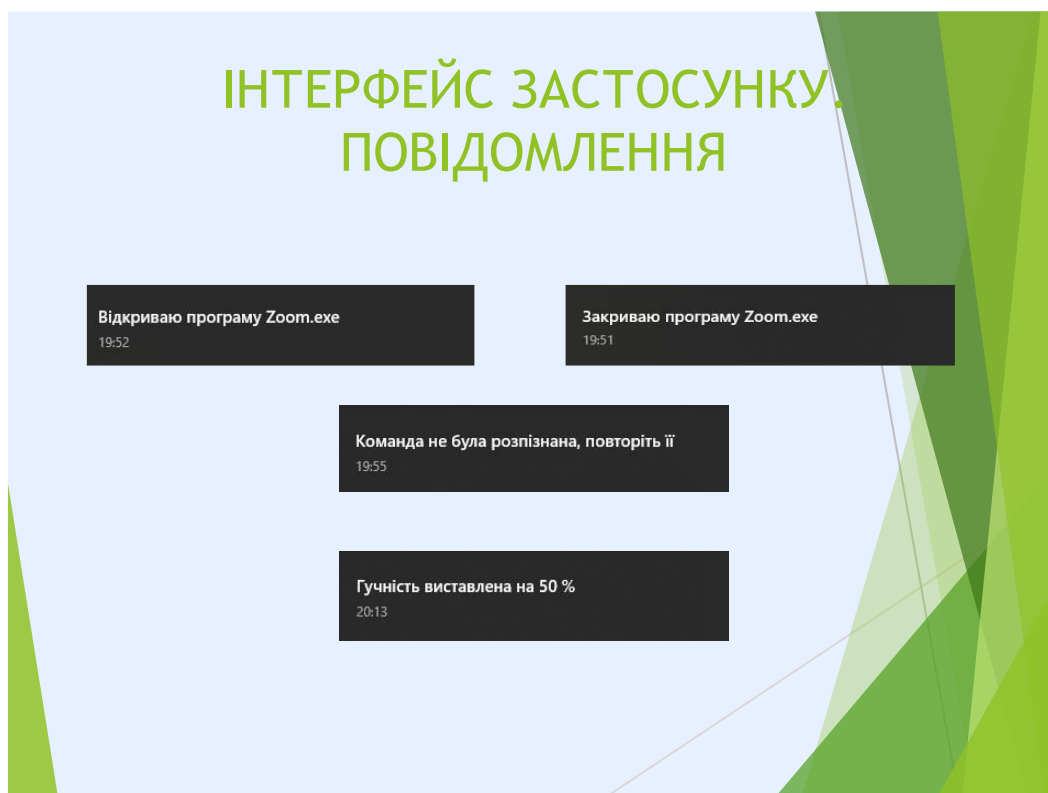


Рисунок В.13 – Слайд 13

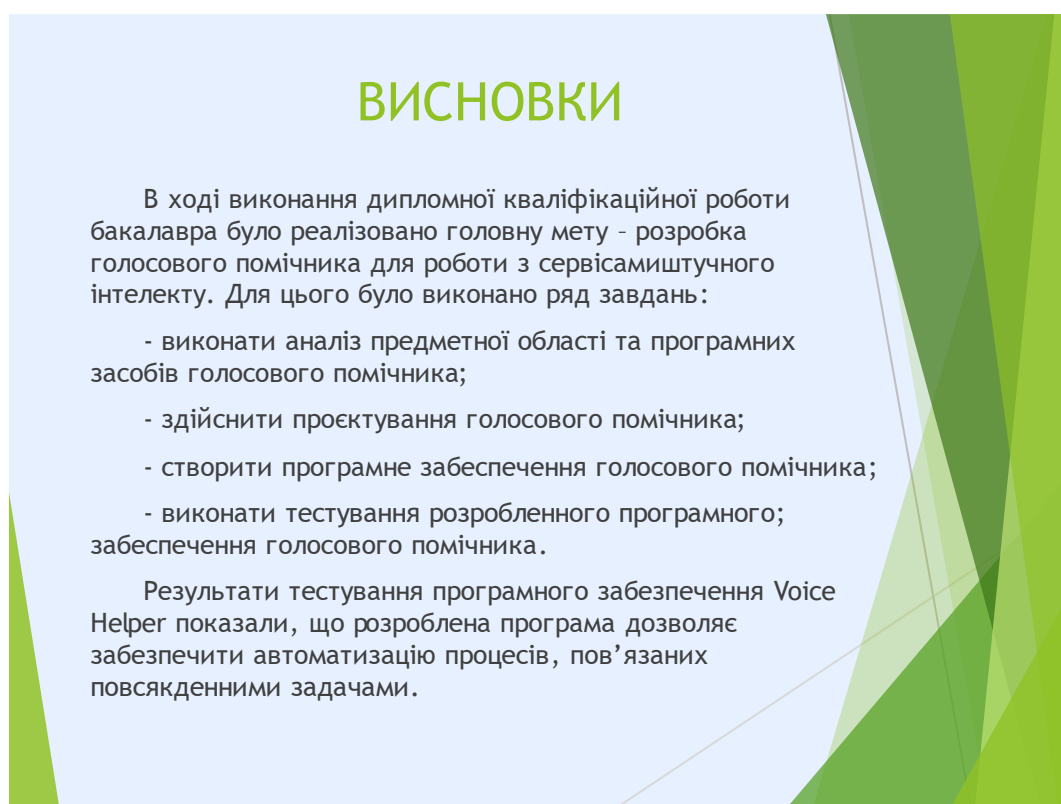


Рисунок В.14 – Слайд 14