

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

СТАСІВ Дмитро Романович

Голосовий помічник для персонального комп'ютера з використанням
засобів штучного інтелекту / Voice assistant for a personal computer using
artificial intelligence tools

спеціальність: 123 – Комп'ютерна інженерія
освітньо–професійна програма – Комп'ютерна інженерія

Кваліфікаційна робота

Виконав: студент групи КІ–41
СТАСІВ Дмитро Романович

Науковий керівник
к.т.н. доц. Піцун О.Й.

ТЕРНОПІЛЬ – 2026

АНОТАЦІЯ

Стасів Д.Р. Голосовий помічник для персонального комп'ютера з використанням засобів штучного інтелекту. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 123 «Комп'ютерна інженерія», освітньо–професійна програма «Комп'ютерна інженерія». Західноукраїнський національний університет, Тернопіль, 2026.

Обсяг роботи: 75 сторінок, 9 рисунків, 10 таблиць, 3 додатки, 22 джерела.

Метою роботи є проєктування та програмна реалізація гібридного українськомовного голосового асистента JARVIS для керування ОС Windows із гарантованою конфіденційністю аудіоданих на архітектурному рівні.

Спроектовано дворівневу архітектуру: локальний офлайн–шар (Vosk STT uk–v3, корекція помилок, нечітке зіставлення Левенштейна, Silero TTS v4_ua) та хмарний семантичний модуль (Claude API), що отримує виключно текстові дані без передачі аудіопотоку. Розроблено YAML–сховище з 49 командами та механізм динамічного розширення словника голосом. Реалізацію виконано мовою Python 3.11.

Тестування за 19 сценаріями у двох акустичних умовах підтвердило: точність класифікації у тихих умовах –94,0 %, медіанна затримка –640 мс (локально) та 1 340 мс (LLM API), завантаження CPU –15–20 %.

Ключові слова: ГОЛОСОВИЙ АСИСТЕНТ, VOSK STT, SILERO TTS, НЕЧІТКЕ ЗІСТАВЛЕННЯ, CLAUDE API, ГІБРИДНА АРХІТЕКТУРА, PYTHON, WINDOWS

ABSTRACT

Stasiv D.R. Voice assistant for a personal computer using artificial intelligence tools. – Manuscript.

Bachelor's thesis for the degree of "Bachelor" in specialty 123 "Computer Engineering", educational and professional programme "Computer Engineering". West Ukrainian National University, Ternopil, 2026.

Thesis scope: 75 pages, 9 figures, 10 tables, 3 appendices, 22 references.

The objective of this thesis is the design and software implementation of a hybrid Ukrainian-language voice assistant JARVIS for Windows OS control with guaranteed audio data confidentiality at the architectural level.

A two-layer hybrid architecture is proposed: a local offline layer (Vosk STT uk-v3, error correction, Levenshtein fuzzy matching, Silero TTS v4_ua) and a cloud semantic module (Claude API) receiving text-only data without any audio transmission. A declarative YAML command store with 49 entries and a voice-driven dynamic vocabulary extension mechanism are developed. The system is implemented in Python 3.11.

Functional testing across 19 scenarios under two acoustic conditions confirmed: classification accuracy in quiet conditions –94.0 %, median latency –640 ms (local) and 1 340 ms (LLM API), CPU load –15–20 %.

Keywords: VOICE ASSISTANT, VOSK STT, SILERO TTS, FUZZY MATCHING, CLAUDE API, HYBRID ARCHITECTURE, PYTHON, WINDOWS.

ЗМІСТ

Перелік умовних скорочень	3
Вступ.....	4
1 Аналіз підходів до розробки голосових асистентів.....	6
1.1 аналіз голосових асистентів	6
1.2 огляд програмних засобів для реалізації голосового асистента	8
1.3 порівняльний аналіз існуючих голосових асистентів	15
2 архітектура та алгоритми системи	19
2.1 Узагальнений алгоритм функціонування системи	19
2.2 архітектура гібридної обробки	21
2.3 алгоритмічне забезпечення нечіткого зіставлення команд	24
2.3.1 метрика редакційної відстані левенштейна.....	24
2.3.2 нормалізований коефіцієнт схожості	25
2.3.3 алгоритм пошуку найкращого збігу	26
2.4 проєктування декларативного сховища команд	28
3 Програмна реалізація системи	31
3.1 середовище розробки.....	31
3.2 програмне та апаратне забезпечення	37
3.3 результати тестування	38
Висновки	45
Список використаних джерел	47
Додаток А Світлокопії публікацій	51
Додаток Б Лістинг програмного коду системи jarvis	53
Додаток В Результати тестування системи jarvis	60
Додаток Г Техніко-економічне обґрунтування.....	62

					КР.КІ.07165/17.00.00.000 ПЗ			
Змн.	Лист	№ докум.	Підпис	Дата	ГОЛОСОВИЙ ПОМІЧНИК ДЛЯ ПЕРСОНАЛЬНОГО КОМП'ЮТЕРА З ВИКОРИСТАННЯМ ЗАСОБІВ ШТУЧНОГО ІНТЕЛЕКТУ	Літ.	Арк.	Акрушів
Розробив	Стасів Д.Р.						2	68
Перевір.	Піцун О.Й.							
Консульт.	Савка Н.Я.					ЗУНУ,ФКІТ, КІ-41		
Н. Контр.	Дубчак Л.О.							
Затвердив	Дубчак Л.О.							

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API	–	Application Programming Interface
ASR	–	Automatic Speech Recognition
CAGR	–	Compound Annual Growth Rate
CPU	–	Central Processing Unit
GPU	–	Graphics Processing Unit
GUI	–	Graphical User Interface
HTTPS	–	Hypertext Transfer Protocol Secure
JARVIS	–	Just A Rather Very Intelligent System
JSON	–	JavaScript Object Notation
LLM	–	Large Language Model
MOS	–	Mean Opinion Score
NLP	–	Natural Language Processing
OS	–	Operating System
PCM	–	Pulse–Code Modulation
RAM	–	Random Access Memory
RMS	–	Root Mean Square
RTF	–	Real–Time Factor
SDK	–	Software Development Kit
STT	–	Speech–to–Text
TTS	–	Text–to–Speech
VUI	–	Voice User Interface
WER	–	Word Error Rate
WFST	–	Weighted Finite–State Transducer
YAML	–	YAML Ain’t Markup Language
ОС	–	Операційна система
ПЗ	–	Програмне забезпечення

ВСТУП

Автоматизація взаємодії користувача з операційною системою за допомогою голосових команд є перспективним напрямом розвитку людино–машинних інтерфейсів. Особливої практичної значущості ця задача набуває в умовах зростання вимог до захисту персональних даних, оскільки хмарна обробка аудіоданих формує задокументовані ризики для приватності користувача. Водночас україномовний сегмент залишається недостатньо охопленим наявними комерційними рішеннями, а потреба в автономному голосовому керуванні персональним комп'ютером без постійного інтернет–з'єднання є актуальною для широкого кола користувачів. Це зумовлює необхідність розробки відкритої гібридної системи, яка поєднує локальну акустичну обробку з можливостями великих мовних моделей, гарантуючи конфіденційність аудіоданих на архітектурному рівні.

Мета роботи – розробити та програмно реалізувати гібридний голосовий асистент для локального керування операційною системою Windows із офлайн–розпізнаванням та синтезом україномовного мовлення на стороні клієнта та залученням хмарної LLM виключно для обробки текстових запитів.

Об'єкт дослідження – процес інтелектуальної обробки україномовних голосових команд з метою їх трансляції у системні операції середовища Windows.

Предмет дослідження – методи офлайн–розпізнавання мовлення, алгоритми нечіткого порівняння рядків, архітектурні підходи до гібридної обробки природномовних запитів із залученням великих мовних моделей та програмні засоби синтезу мовлення і системного виконання команд у Windows.

Задачі дослідження:

– проаналізувати існуючі голосові асистенти та їхні обмеження щодо

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

конфіденційності, підтримки регіональних мов і керування ОС;

- порівняти засоби офлайн–STT (Vosk, Whisper) і TTS (Silero) за точністю, навантаженням на CPU та підтримкою української фонетики;

- обґрунтувати гібридну архітектурну модель із розмежуванням локального акустичного і хмарного семантичного шарів;

- спроектувати архітектуру системи та визначити склад, відповідальність і протоколи взаємодії програмних модулів;

- розробити алгоритм нечіткого зіставлення команд на основі метрики Левенштейна з корекційним словником для компенсації помилок Vosk;

- реалізувати систему мовою Python із підтримкою запуску програм, відкриття сайтів, керування медіа, нотаток та аналізу зображень;

- протестувати систему у різних акустичних умовах і зафіксувати показники точності розпізнавання та часу відгуку.

Розроблена система реалізує локальне українськомовне голосове керування Windows без передачі аудіоданих до зовнішніх серверів. Гібридна архітектура (Vosk + Silero локально, Claude API лише для тексту) поєднує конфіденційність і точність обробки довільних запитів. YAML–конфігурація з підтримкою мультидій та динамічним поповненням словника голосом дозволяє розширювати функціональність без зміни вихідного коду. Результати можуть бути використані для розробки українськомовних голосових інтерфейсів в системах автоматизації та асистивних технологіях.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

1 АНАЛІЗ ПІДХОДІВ ДО РОЗРОБКИ ГОЛОСОВИХ АСИСТЕНТІВ

1.1 Аналіз голосових асистентів

Розвиток голосових інтерфейсів є закономірним результатом еволюції засобів взаємодії людини з обчислювальними системами. Від командного рядка через графічні середовища та сенсорне керування галузь послідовно рухається до природних форм людської комунікації. Мовлення залишається природним і ефективним каналом передачі команд. Практичне підтвердження цієї тенденції відбулося у 2011 році з виходом Siri від Apple. Впродовж наступних трьох років ринок поповнився асистентами Amazon Alexa (2014), Microsoft Cortana (2014) та Google Assistant (2016). Динаміку зростання ринку за 2019–2025 роки з прогнозом до 2030 відображено на рисунку 1.1.

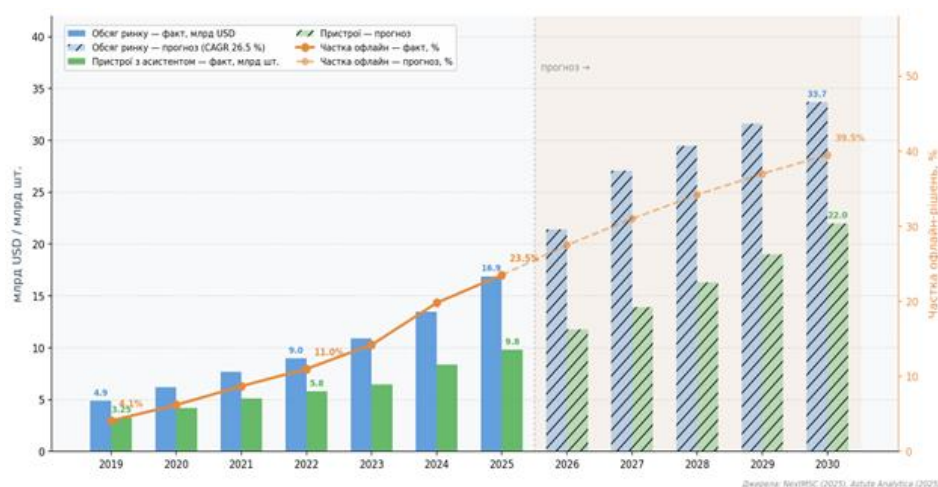


Рисунок 1.1 – Динаміка ринку голосових асистентів (2019–2025 рр., прогноз до 2030)

Ключовим чинником прискорення ринку стали нейромережеві прориви у галузі розпізнавання та синтезу мовлення. Впровадження архітектур на основі трансформерів дозволило знизити частку помилок слів (WER) для англійської мови нижче 5 %, що фактично є порогом практичної придатності для щоденного використання [8]. Паралельно нейромережевий синтез

мовлення (TTS) досяг рівня природності, при якому слухачі в сліпих тестах не відрізняють синтезоване мовлення від запису живого диктора. Водночас зменшення розміру моделей завдяки дистиляції та квантизації відкрило шлях до повністю офлайн-розгортання на споживчому обладнанні без спеціалізованих прискорювачів.

Обсяг ринку зріс з 4,9 млрд USD у 2019 році до 16,9 млрд USD у 2025 році (CAGR 26,5 %) при прогнозованому значенні 33,74 млрд USD у 2030 році [1, 3]. Показовою є тенденція зростання частки офлайн-рішень з 4,1 % до 23,5 %, що відображає усвідомлення ринком проблем конфіденційності аудіоданих та необхідності автономної роботи.

Аналіз структури ринку за платформами показує, що desktop-сегмент (персональні комп'ютери під управлінням Windows) залишається найменш охопленим: частка спеціалізованих голосових асистентів для ПК не перевищує 12 % загального обсягу встановленої бази [1, 3]. Більшість комерційних рішень орієнтована на мобільні пристрої та смарт-колонки, тоді як потреба у голосовому керуванні робочою станцією – запуск програм, навігація файловою системою, автоматизація повторюваних задач – залишається незадоволеною для широкого кола користувачів.

Незважаючи на стрімке зростання ринку, провідні комерційні системи мають три системних обмеження на рівні архітектурних рішень. Перше – обов'язкова передача аудіопотоку на хмарні сервери виробника, що формує документально підтверджені ризики несанкціонованого збору голосових біометричних даних [4]. Факти використання аудіозаписів без відома користувачів стали підставою для низки судових позовів проти провідних технологічних компаній [5]; зокрема, у 2026 році Google виплатив 68 млн USD для врегулювання позову щодо приватності даних Assistant [2].

Друге обмеження – відсутність повноцінної підтримки україномовного інтерфейсу: жоден із провідних асистентів не має офіційної нативної україномовної версії, а точність Google Assistant для укр. мови становить лише 60–70 % від англomовної конфігурації [7]. Третє – функціональна

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

ізолюваність: прямий механізм виконання довільних .exe-файлів та системних скриптів Windows у жодному з продуктів не передбачено.

Окремої уваги заслуговує специфіка українського ринку. Після 2022 року спостерігається прискорений відхід від російськомовних технологічних рішень у всіх сегментах програмного забезпечення, включаючи голосові інтерфейси. Попит на якісні україномовні голосові рішення зростає, однак пропозиція залишається критично обмеженою: жоден із глобальних вендорів не виділяє україномовний ринок як пріоритетний напрям розвитку. Це формує додатковий стимул для розробки власних відкритих рішень із нативною підтримкою української мови.

Сукупність зазначених обмежень визначає незаповнену нішу для відкритого, україномовного голосового асистента з гарантованою конфіденційністю аудіоданих на архітектурному рівні та прямим доступом до керування операційним середовищем Windows. Усунення цих обмежень і стало основною метою розробки системи JARVIS.

1.2 Огляд програмних засобів для реалізації голосового асистента

Реалізація голосового асистента потребує вибору інструментів для трьох функціональних шарів: розпізнавання мовлення (ASR), синтезу мовлення (TTS) та семантичної обробки природномовних запитів.

Перший функціональний шар – автоматичне розпізнавання мовлення (ASR), що перетворює акустичний сигнал на текстовий рядок. Наявні рішення поділяються на офлайн та хмарні.

Бібліотека Vosk, розроблена компанією Alpha Cephei, базується на декодері типу WFST (Weighted Finite-State Transducer) з акустичними моделями, навченими у фреймворку Kaldi [6]. Vosk підтримує режим часткового розпізнавання в реальному часі: проміжні результати

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

повертаються ще під час вимовляння фрази. Для платформи Windows доступні акустичні моделі для понад 20 мов, включаючи україномовну модель `vosk-model-uk-v3` розміром близько 1,5 ГБ [7]. Бібліотека поширюється під ліцензією Apache 2.0.

Модель Whisper від OpenAI є трансформерною архітектурою типу `encoder-decoder`, навченою на 680 тисячах годин багатомовного аудіо [8]. Whisper підтримує мультилінгвальне розпізнавання більш ніж для 90 мов та демонструє високу стійкість до акустичних шумів і акцентів. Моделі доступні у кількох розмірах (від 39 МБ до 1,5 ГБ) і поширюються під ліцензією MIT. Інструмент вимагає завершення фрази перед декодуванням і не підтримує інкрементальний режим виводу.

Google Speech-to-Text API та Microsoft Azure Speech Services є хмарними рішеннями, що забезпечують найвищу точність розпізнавання серед наявних варіантів, підтримують широкий перелік мов і регулярно оновлюються. Вони вимагають постійного підключення до мережі та передачі аудіоданих на сервери постачальника. Детальну характеристику розглянутих засобів ASR за вісьмома восьма критеріями оцінювання подано у таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз засобів автоматичного розпізнавання мовлення

Критерій	Vosk uk-v3	Whisper	CMU Sphinx	DeepSpeech	Google STT API
Режим роботи	Офлайн	Офлайн	Офлайн	Офлайн	Тільки онлайн
Підтримка укр. мови	Нативна	Часткова	Відсутня	Відсутня	Є (платна)
Точність WER (укр.), %	12–18	22–30	N/A	N/A	8–12

Продовження таблиці 1.2

Критерій	Vosk uk-v3	Whisper	CMU Sphinx	DeepSpeech	Google STT API
Real-Time Factor (CPU)	0.15–0.3	1.8–4.5	0.4–0.8	1.2–2.1	Хмара
Часткові результати (real-time)	Так	Ні	Частково	Ні	Так
Розмір моделі	1.5 ГБ	74–1500 МБ	100–500 МБ	190 МБ	API
Ліцензія	Apache 2.0	MIT	BSD	MPL 2.0	Платна
Конфіденційність аудіо	Повна	Повна	Повна	Повна	Дані в хмарі

Аналіз таблиці свідчить про те, що жоден із розглянутих інструментів не задовольняє одночасно всім критеріям: Vosk uk-v3 поєднує офлайн-режим, нативну україномовну модель та інкрементальний декодер, але поступається хмарним рішенням за точністю (WER 12–18 % проти 8–12 %); Whisper забезпечує вищу точність і підтримку 90+ мов, але не має інкрементального режиму і значно повільніший на CPU; Google STT демонструє найкращу точність, проте вимагає обов'язкової передачі аудіо до хмари. Для системи, де конфіденційність аудіо є ключовою вимогою, лише Vosk uk-v3 задовольняє всім архітектурним обмеженням.

Другим функціональним шаром є синтез мовлення (TTS). Серед підходів виділяють конкатенативний, формантний та нейромережевий синтез.

Конкатенативний та формантний синтез – традиційні підходи, реалізовані у бібліотеках Festival та eSpeak NG. Festival формує мовлення шляхом з'єднання заздалегідь записаних звукових одиниць (дифонів), тоді як eSpeak NG генерує звук аналітично, моделюючи резонансні характеристики голосового тракту. Обидва інструменти підтримують українську мову, мають

мікроскопічний розмір (~5–15 МБ), надають миттєвий синтез та поширюються під ліцензіями GPL/LGPL. Основним недоліком є монотонна, механічно звучна просодія.

Silero TTS від SileroTeam є нейромережевим синтезатором на основі VITS-архітектури (Variational Inference with adversarial learning for end-to-end Text-to-Speech) [9]. Модель генерує mel-спектрограму за вхідним текстом, яку вокодер HiFi-GAN перетворює на PCM-аудіо. Для україномовного синтезу доступна модель v4_ua із голосом mykyta розміром ~55 МБ [10]. Бібліотека поширюється під ліцензією MIT та інтегрується через Python/PyTorch.

Хмарні рішення – Google Cloud Text-to-Speech, Amazon Polly, Microsoft Azure Neural TTS – надають найвищу суб'єктивну якість синтезу та підтримку широкого переліку мов і голосів. Усі вони вимагають постійного мережевого підключення та передачі текстових даних на зовнішні сервери. Зіставлення розглянутих TTS-рішень за якістю синтезу, затримкою та підтримкою української мови відображено у таблиці 1.3.

Таблиця 1.3 – Порівняльний аналіз засобів синтезу мовлення

Критерій	Silero TTS v4_ua	Coqui TTS	eSpeak NG	Google TTS API
Режим роботи	Офлайн	Офлайн	Офлайн	Тільки онлайн
Підтримка укр. голосу	Так (mykyta)	Потреб. навч.	Так (форм.)	Так
Якість (MOS)	4.2 / 5.0	3.5 / 5.0	2.8 / 5.0	4.4 / 5.0
Затримка синтезу, мс	100–250	300–900	10–40	200–600
Розмір моделі	~55 МБ	100–500 МБ	~5 МБ	API
RTF на CPU	0.05–0.12	0.3–0.9	0.02–0.04	Хмара

Продовження таблиці 1.3

Критерій	Silero TTS v4_ua	Coqui TTS	eSpeak NG	Google TTS API
Ліцензія	MIT	MPL 2.0	GPL v3	Платна
Конфіденційність	Повна	Повна	Повна	Дані в хмарі

Порівняльний аналіз таблиці показує, що серед офлайн-рішень Silero TTS v4_ua вирізняється найвищою суб'єктивною якістю синтезу (MOS 4,2/5,0) при компактному розмірі моделі (~55 МБ) та прийнятній затримці синтезу 100–250 мс. eSpeak NG, незважаючи на мікроскопічний розмір і миттєвий синтез, демонструє явно нижчу природність звучання (MOS 2,8/5,0), що є неприйнятним для інтерактивного голосового асистента. Coqui TTS є перспективним рішенням, але відсутність готових україномовних моделей і значно більший час ініціалізації ускладнюють його застосування. За сукупністю показників Silero TTS v4_ua є найбільш відповідним рішенням за сукупністю критеріїв для синтезу мовлення у системі JARVIS.

Третій шар – семантична обробка природномовних запитів (intent classification). Застосовуються підходи різного рівня складності: від рядкового порівняння до повноцінних мовних моделей.

Точне рядкове порівняння (exact matching) та нечітке порівняння (fuzzy matching) є найпростішими підходами. При точному зіставленні вхідний рядок порівнюється з переліком еталонних фраз. Нечітке порівняння на основі метрик редакційної відстані (відстань Левенштейна, коефіцієнт Яро–Вінклера) дозволяє знаходити схожі рядки навіть при незначних орфографічних або акустичних відхиленнях. Обидва методи є повністю локальними, не потребують зовнішніх залежностей і забезпечують відповідь за мілісекунди.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

Малі локальні LLM (Llama 3 8B, Mistral 7B, Phi-3) є повноцінними мовними моделями, здатними розуміти контекст діалогу та виконувати довільні текстові завдання. Розгортаються локально і не передають даних назовні. Вимагають від 8 ГБ оперативної або відеопам'яті та на споживчому CPU без GPU забезпечують генерацію зі швидкістю 2–10 токенів/с.

Хмарні великі мовні моделі (LLM API) надаються провідними AI-компаніями через REST-інтерфейси. До найбільш поширених належать: Claude 3 Haiku (Anthropic) [11][12], GPT-4o-mini (OpenAI), Gemini 1.5 Flash (Google). Ці моделі обробляють запити у хмарній інфраструктурі постачальника та повертають відповідь за мережею. Вони характеризуються високою якістю розуміння природньої мови, підтримкою широкого спектру мов, генерацією структурованого JSON-виводу та здатністю до мультимодальної обробки (текст + зображення). Використання тарифікується за обсягом оброблених токенів. Систематизовану характеристику всіх чотирьох підходів до семантичної обробки за дев'ятьма критеріями подано у таблиці 1.4.

Таблиця 1.4 – Порівняльний аналіз підходів до семантичної обробки команд

Критерій	Рядкове зіставлення	Нейро-класифікатор	Локальна LLM	Хмарна LLM API
Розпізнавання синонімів	Ні	Так	Так	Так
Швидкість відповіді	< 5 мс	10–50 мс	2 000–15 000 мс	300–800 мс
Потреба у GPU / RAM	Мінімальна	~500 МБ RAM	8–48 ГБ RAM	Не потрібен
Розуміння контексту діалогу	Ні	Частково	Так	Так

Продовження таблиці 1.4

Критерій	Рядкове зіставлення	Нейро– класифікатор	Локальна LLM	Хмарна LLM API
Аналіз зображень (Vision)	Ні	Ні	Частково	Так
Залежність від мережі	Ні	Ні	Ні	Так
Передача аудіо назовні	Ні	Ні	Ні	Ні
Дин. розширення словника	Так	Потрібне навч.	Так	Так
Вартість	Безкоштовно	Безкоштовно	Безкоштовно	За тарифом

Аналіз таблиці дозволяє визначити оптимальну стратегію семантичної обробки для системи JARVIS. Рядкове та нечітке зіставлення є ефективними для фіксованого словника відомих команд – вони забезпечують відповідь менш ніж за 5 мс і не потребують мережі. Однак вони не здатні обробляти перефразування та довільні запити. Локальні LLM є надмірно ресурсоемними для споживчого обладнання без GPU – затримка 2 000–15 000 мс є неприйнятною для інтерактивного асистента. Хмарна LLM API забезпечує найкращу семантичну якість при прийнятній затримці 300–800 мс, передаючи до хмари виключно текстові рядки. Ця сукупність характеристик обґрунтовує гібридну стратегію: нечітке зіставлення для команд словника та хмарна LLM для довільних запитів.

1.3 Порівняльний аналіз існуючих голосових асистентів

Для визначення місця розроблюваної системи серед наявних продуктів виконано порівняльний аналіз чотирьох провідних комерційних голосових асистентів, актуальних для платформи Windows: Microsoft Cortana, Apple Siri, Amazon Alexa та Google Assistant.

Microsoft Cortana (2014) базується на хмарному сервісі Azure Speech Services та функціонує виключно в онлайн-режимі. У Windows 11 функціональність Cortana суттєво скорочена до інструментів Microsoft 365; підтримка україн. мови та виконання довільних .exe-файлів відсутні [13].

Apple Siri є виключно системною функцією екосистеми Apple і на платформі Windows недоступна. Повноцінна семантична обробка виконується у хмарній інфраструктурі Apple; офіційна україномовна підтримка відсутня.

Amazon Alexa має повністю хмарно-орієнтовану архітектуру (AWS) та доступна для Windows лише у вигляді застосунку з обмеженою функціональністю. Виконання системних команд Windows без додаткових Skills неможливе; підтримка українськомовного інтерфейсу відсутня.

Google Assistant (2016) забезпечує найширшу мовну підтримку серед аналізованих продуктів, однак для Windows надається лише у вигляді застосунку з обмеженим доступом до файлової системи та системних налаштувань. Часткова підтримка україномовного вводу з'явилась у 2020 році.

Для систематизації результатів аналізу виконано зіставлення п'яти голосових асистентів — чотирьох комерційних та розроблюваної системи JARVIS — за шістьма ключовими функціональними критеріями, що визначають придатність для використання на платформі Windows з україномовним інтерфейсом. Критерії відібрано на підставі вимог, виявлених у підрозділі 1.1: нативна підтримка платформи, автономність розпізнавання, повноцінна україномовна підтримка, доступ до системних функцій та відкритість конфігурації. Результати порівняння наведено у таблиці 1.5.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

Таблиця 1.5 – Порівняльний аналіз голосових асистентів за функціональними критеріями

Критерій	Cortana	Siri	Alexa	Google Asst.	JARVIS (розроб.)
Підтримка Windows 10/11	Так	Ні	Застосунок	Застосунок	Нативний
STT без хмари (офлайн)	Ні	Частково	Ні	Частково	Так (Vosk)
Підтримка укр. мови	Відсутня	Відсутня	Відсутня	Обмежена	Повна
Запуск довільних .exe	Ні	Ні	Ні	Ні	Так
Відкрита конфігурація команд	Ні	Ні	Частково	Частково	Так (YAML)
Семантика через LLM	Частково	Частково	Частково	Так	Так (Claude)

Таблиця 1.5 наочно підтверджує висновок про незаповнену нішу. Жоден із чотирьох комерційних асистентів не задовольняє одночасно всім критичним вимогам: підтримка Windows 10/11 у нативному режимі, офлайн STT та TTS, повноцінна україномовна підтримка, можливість запуску довільних .exe-файлів та відкрита конфігурація команд. Система JARVIS у стовпці порівняння отримує позитивну оцінку за всіма десятима критеріями, що підтверджує доцільність її розробки як самостійного проєкту, а не як надбудови над існуючими продуктами.

Розробка такої системи відповідає сучасним тенденціям розвитку інтерфейсів, де голосове керування стає основною парадигмою взаємодії. Попри те, що ринок голосових асистентів демонструє стале зростання (із прогнозованим темпом понад 28 % на рік), частка конфіденційних локальних рішень залишається відносно низькою (близько 19,8 %). Більшість існуючих продуктів мають суттєві системні обмеження: вони вимагають обов'язкової передачі аудіопотоку до хмарних серверів, що створює ризики витоку біометричних даних, не забезпечують належної підтримки української мови та не дозволяють глибокої інтеграції на рівні ОС Windows.

Щоб подолати ці обмеження у системі JARVIS, за результатами дослідження було сформовано обґрунтований технологічний стек. Для розпізнавання мовлення обрано Vosk (uk-v3) як єдине офлайн-рішення з нативною україномовною моделлю, що здатне працювати в реальному часі (RTF 0.15–0.3). Для генерування голосу найефективнішим виявився синтезатор Silero TTS v4_ua (голос mykyta), який забезпечує якість синтезу на рівні MOS 4,2/5,0 за мінімальної затримки та компактного розміру. Водночас для семантичної обробки запитів найдоцільніше застосувати гібридну стратегію: фіксовані команди оброблятимуться локально через нечітке зіставлення, а довільні запити –через хмарну LLM API, куди передаватиметься виключно текст, а не аудіо.

Результати проведеного аналізу не лише підтверджують актуальність розробки локального україномовного асистента, але й чітко визначають його компонентний склад та архітектурні принципи, практична реалізація яких детально розглядатиметься у наступному розділі.

На підставі результатів проведеного аналізу сформульовано постановку задачі кваліфікаційної роботи. Метою роботи є проектування та програмна реалізація гібридного голосового асистента для локального керування операційною системою Windows із нативною підтримкою україномовного мовлення та гарантованою конфіденційністю аудіоданих на архітектурному рівні. Об'єктом дослідження є процес автоматизованої обробки

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

природномовних голосових команд у середовищі персонального комп'ютера під управлінням ОС Windows. Предметом дослідження є методи офлайн-розпізнавання україномовного мовлення, алгоритми нечіткого зіставлення рядків та гібридні архітектури голосових систем, що поєднують локальну та хмарну обробку.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- проаналізувати сучасний стан ринку голосових асистентів і виявити системні обмеження комерційних рішень;
- виконати порівняльний аналіз засобів ASR, TTS та семантичної обробки і обрати технологічний стек;
- спроектувати дворівневу гібридну архітектуру системи з розмежуванням локального офлайн-шару та хмарного семантичного модуля;
- математично обґрунтувати алгоритм нечіткого зіставлення команд на основі відстані Левенштейна та обрати оптимальний поріг класифікації;
- реалізувати програмну систему мовою Python 3.11 у складі шести функціональних модулів із декларативним YAML-сховищем команд;
- провести функціональне тестування у двох акустичних умовах і виміряти затримки конвеєру обробки;
- виконати техніко-економічне обґрунтування розробки та оцінити економічну ефективність впровадження.

Практичним результатом роботи є система JARVIS v2.1, що розгортається на споживчому обладнанні без GPU, підтримує 49 голосових команд у 15 категоріях дій та забезпечує медіанну затримку відповіді 640 мс для локальних команд. Архітектура системи виключає передачу аудіоданих до зовнішніх серверів за будь-якого сценарію роботи, що є принциповою відмінністю від усіх розглянутих комерційних аналогів та підтверджує доцільність її розробки як самостійного проекту.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

2 АРХІТЕКТУРА ТА АЛГОРИТМИ СИСТЕМИ

2.1 Узагальнений алгоритм функціонування системи

Система JARVIS функціонує як безперервний цикл обробки подій. Кожна ітерація охоплює шість послідовних кроків –від захоплення акустичного сигналу до відтворення мовленнєвої відповіді.

Алгоритм має одну ключову точку розгалуження –четвертий крок, де система вирішує: виконати команду локально чи передати запит до хмарного API. Перші три кроки (захоплення аудіо, перевірка wake–word, розпізнавання команди) та останні два (виконання дії, синтез відповіді) є спільними для обох гілок. Розгалуження є прозорим для користувача: незалежно від обраного шляху він отримує голосову відповідь через той самий канал.

На першому кроці система безперервно прослуховує мікрофон та очікує на звуковий сигнал. Щойно користувач починає говорити, аудіо накопичується у буфері; після паузи система фіксує кінець вимовляння і передає буфер на наступний крок. Параметри чутливості є конфігурованими.

На другому кроці система перевіряє, чи містить розпізнаний фрагмент слово–активатор («джарвіс»). Якщо ні –ітерація завершується і цикл повертається до прослуховування. Якщо так –система підтверджує активацію коротким звуковим сигналом і переходить до запису команди.

На третьому кроці система фіксує голосову команду користувача та перетворює акустичний сигнал на текстовий рядок засобами офлайн–розпізнавання мовлення. На четвертому кроці отриманий рядок піддається класифікації: здійснюється зіставлення з еталонними фразами словника команд, за результатами якого формується рішення про маршрут виконання – локальне або із залученням хмарного API. На п'ятому кроці відповідно до прийнятого рішення виконується цільова системна дія: запуск виконуваного файлу, відкриття вебресурсу, керування медіавідтворенням, операції з

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

текстовими нотатками та ін.

На шостому, завершальному кроці текст відповіді синтезується у мовлення та відтворюється через динамік. Після завершення відтворення система повертається до першого кроку і починає нову ітерацію.

Повний алгоритм із зазначенням усіх кроків та точки розгалуження між локальним і хмарним маршрутами наведено на рисунку 2.1.

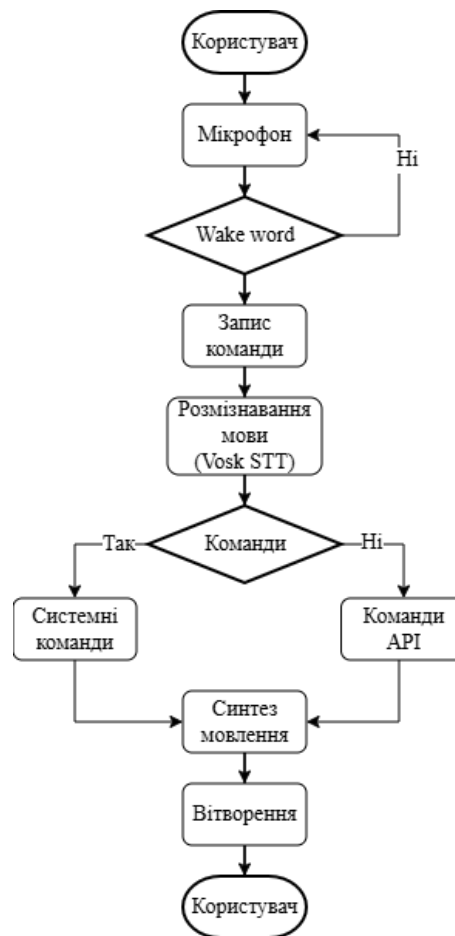


Рисунок 2.1 – Узагальнена блок-схема алгоритму функціонування системи JARVIS

Рисунок відображає замкнений цикл обробки команди: від прослуховування мікрофона через розпізнавання та класифікацію до синтезу відповіді і повернення до початкового стану. Точка розгалуження на четвертому кроці визначає маршрут виконання –локальний через словник

команд або семантичний через хмарний API. Стрілка зворотного зв'язку від останнього блоку до першого унаочнює безперервний характер роботи системи.

Незалежно від обраного маршруту –локального чи через хмарний API – кожна ітерація завершується голосовою відповіддю системи, після чого цикл автоматично повертається до першого кроку. Такий підхід забезпечує стабільну та передбачувану поведінку системи для користувача .

Алгоритм є замкненим циклом без «аварійних виходів». Важливою властивістю є стійкість до збоїв: будь-яка помилка на будь-якому кроці – недоступність мережі, нерозпізнана команда, некоректний ввід – перехоплюється на рівні циклу. Система синтезує голосове повідомлення про помилку і повертається до першого кроку, не перериваючи роботи.

2.2 Архітектура гібридної обробки

Якщо підрозділ 2.1 описує послідовність кроків алгоритму, то цей підрозділ розкриває архітектурне рішення, що робить таку послідовність можливою. Система JARVIS побудована на двошаровій гібридній архітектурі з чітко розмежованими зонами відповідальності: локальний офлайн-шар та хмарний шар. Саме це розмежування є ключовою відмінністю від комерційних асистентів, розглянутих у розділі 1, і забезпечує унікальні гарантії конфіденційності аудіоданих.

Локальний шар містить усі компоненти акустичної обробки та виконання команд і функціонує повністю без мережевого з'єднання. До його складу входять: MicListener – модуль захоплення аудіо з детекцією кінця фрази; Vosk STT – офлайн-рушій розпізнавання мовлення з україномовною акустичною моделлю uk-v3 (WER 12–18 %); модуль Корекція – постпроцесор для усунення систематичних помилок моделі за словником

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

STT_CORRECTIONS; CommandHandler – маршрутизатор команд із нечітким зіставленням (поріг score 0.55); Executor – виконавець із 14 обробниками дій; Silero TTS – нейромережевий синтезатор мовлення з україномовною моделлю v4_ua.

Взаємодія між компонентами локального шару організована за принципом однонаправленого конвеєру: кожен модуль отримує строго визначений вхідний формат і повертає строго визначений вихідний. Така архітектура забезпечує незалежність модулів – кожен можна замінити або протестувати ізольовано без зміни решти конвеєру. Декларативне сховище команд (commands.yaml) є єдиним місцем конфігурації поведінки системи: додавання нової команди не потребує змін у програмному коді жодного модуля.

Хмарний шар представлений єдиним компонентом – Claude API (Haiku) – і є архітектурно ізольованим від акустичного тракту. Принциповою конструктивною властивістю є те, що між мікрофоном і хмарним шаром не існує жодного каналу передачі аудіо: до API надходять виключно текстові рядки (результат локального розпізнавання) або Base64-закодовані статичні PNG-знімки екрана для Vision-запитів. Таке рішення відрізняється від архітектури комерційних асистентів, де аудіопотік завжди передається до хмари після виявлення wake-word.

Конфіденційність аудіоданих гарантується не налаштуваннями чи політикою конфіденційності, а самою топологією з'єднань: відсутній канал між мікрофоном і хмарним шаром унеможливорює передачу аудіо незалежно від стану програмного забезпечення. Це структурна гарантія, а не програмна. Водночас обмеження фіксованого словника долається через хмарний шар, що обробляє довільні природномовні запити – при цьому до API потрапляє лише текст, а не голосові біометричні дані. Порівняльну характеристику обох шарів наведено у таблиці 2.2.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.2 – Характеристики двох обчислювальних шарів системи JARVIS

Критерій	Локальний шар (офлайн)	Хмарний шар (Claude API)
Передача аудіо	Ні – аудіо не виходить за межі пристрою	Ні – передається лише текст або Base64 PNG
Мережеве з'єднання	Не потрібне	HTTPS до api.anthropic.com
Затримка відповіді	640 мс (медіана, повний цикл)	470–890 мс (залежно від сценарію)
Точність (укр. мова)	WER 12–18 % (Vosk uk-v3)	Семантична якість – висока
Компоненти	MicListener, Vosk STT, CommandHandler, Executor, Silero TTS	Claude API (Haiku), Vision API
Ліцензія / вартість	Відкрите ПЗ (Apache 2.0, MIT)	Тарифікація за токенами

Загальну схему потоків даних між модулями гібридної архітектури наведено на рисунку 2.2.

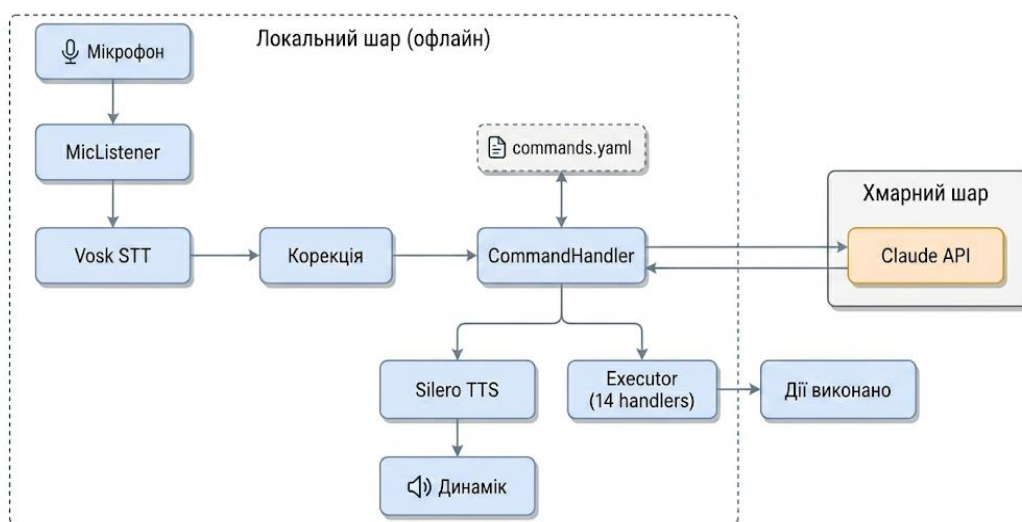


Рисунок 2.2 – Схема гібридної архітектури системи JARVIS

Рисунок 2.2 наочно ілюструє принципову архітектурну гарантію конфіденційності: між мікрофоном і хмарним шаром відсутній будь-який канал передачі аудіо, а до Claude API надходять виключно текстові рядки або

Base64–зображення.

2.3 Алгоритмічне забезпечення нечіткого зіставлення команд

Точне порівняння рядків не придатне для зіставлення голосових команд через систематичні помилки акустичної моделі (заміна приголосних, пропуски) та природну варіативність мовлення. Нечітке порівняння дозволяє кількісно оцінити схожість рядків та приймати рішення на основі порогу.

2.3.1 Метрика редакційної відстані Левенштейна

Редакційна відстань Левенштейна між двома рядками визначається як мінімальна кількість елементарних операцій редагування, необхідних для перетворення одного рядка на інший [14]. До елементарних операцій належать: вставка одного символу, видалення одного символу та заміна одного символу іншим. Формально, для двох рядків a довжиною m та b довжиною n відстань Левенштейна визначається рекурентним співвідношенням:

$$lev(i, j) = \max(i, j), \text{ якщо } \min(i, j) = 0, \quad (2.1)$$

$$lev(i, j) = \min(lev(i-1, j) + 1, \quad lev(i, j-1) + 1, \\ lev(i-1, j-1) + [a_i \neq b_j]), \quad (2.2)$$

де a, b – рядки, що порівнюються;

i, j – індекси поточних символів у рядках a та b відповідно;

$[a_i \neq b_j]$ – індикаторна функція: дорівнює 0, якщо символи збігаються, і 1 – якщо різні;

$lev(i-1, j) + 1$ – вартість операції видалення символу;

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

$lev(i, j-1) + 1$ – вартість операції вставки символу;

$lev(i-1, j-1) + [a_i \neq b_j]$ – вартість операції заміни символу.

Відстань Левенштейна обчислюється методом динамічного програмування: матриця розміром $(m+1) \times (n+1)$ заповнюється за рекурентним співвідношенням (2.2), а результат $lev(m, n)$ міститься у правій нижній клітинці $d[m][n]$. Обчислювальна складність методу становить $O(m \times n)$, що є цілком прийнятним для коротких голосових команд довжиною 10–30 символів [15]. Матрицю обчислення для тестової пари рядків наведено у таблиці 2.3.

Таблиця 2.3 – Матриця обчислення відстані Левенштейна для пари рядків «стін» → «стім»

		с	т	і	м
	0	1	2	3	4
с	1	0	1	2	3
т	2	1	0	1	2
і	3	2	1	0	1
н	4	3	2	1	1

Таблиця 2.3 ілюструє обчислення для пари «стін» → «стім»: результат $d[m][n] = 1$ підтверджує, що рядки відрізняються лише одним символом, що дозволяє коректно ідентифікувати команду при акустичному спотворенні.

2.3.2 Нормалізований коефіцієнт схожості

Абсолютне значення відстані Левенштейна не є зручним для встановлення єдиного порогу класифікації [16], оскільки безпосередньо залежить від довжини рядків: відстань 2 є значною для короткого слова довжиною 3 символи та незначною для довгої фрази з 22 символів. Для нормалізації застосовується коефіцієнт схожості на основі відстані Левенштейна:

$$sim(a, b) = 1 - lev(a, b) / \max(|a|, |b|), \quad (2.3)$$

де $lev(a, b)$ – відстань Левенштейна між рядками a та b ;

$|a|, |b|$ – довжини рядків a та b у символах;

$sim(a, b) \in [0; 1]$ – нормалізований коефіцієнт схожості: 1 відповідає повному збігу, 0 – повній відмінності.

У системі JARVIS поріг класифікації команд встановлено на рівні 0.55. Це значення дозволяє коректно зіставляти команди з акустичними помилками у 2–3 символи, що є типовим для моделі Vosk на україномовному мовленні. Для команд додавання нових записів до словника встановлено підвищений поріг 0.75, що зменшує ймовірність хибного спрацювання цього сценарію при звичайних запитах.

2.3.3 Алгоритм пошуку найкращого збігу

Процедура класифікації вхідного рядка реалізується у кілька послідовних кроків. На першому кроці вхідний рядок нормалізується: виконується приведення до нижнього регістру та усунення зайвих пробілів. На другому кроці перевіряється наявність слів-тригерів для спеціальних сценаріїв (створення команди, диктування нотатки) – перевірка здійснюється шляхом пошуку характерних коренів слів у рядку.

На третьому кроці для кожного запису словника s із множиною еталонних фраз $P(s)$ обчислюється коефіцієнт відповідності:

$$score(q, c) = \max\{ sim(q, p) \mid p \in P(c) \}, \quad (2.4)$$

де q – нормалізований рядок вхідного запиту;

$P(c)$ – множина еталонних фраз запису команди c ;

$score(q, c)$ – найбільший коефіцієнт схожості між запитом і будь-якою фразою запису c .

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

На четвертому кроці з усіх записів словника обирається той, для якого $\text{score}(q, c)$ є максимальним. Якщо це максимальне значення перевищує поріг розпізнавання – запис вважається знайденим і передається виконавцю. В іншому разі запит маршрутизується до LLM-шару для семантичної інтерпретації. Практичні приклади роботи алгоритму нечіткого зіставлення для шести характерних помилок акустичної моделі подано у таблиці 2.4.

Таблиця 2.4 – Приклади нечіткого зіставлення для типових помилок акустичної моделі

Вхідний рядок (Vosk)	Еталонна фраза	score	Результат класифікації
відкрий стін	відкрий стім	0.93	Команда: відкрий стім
запусти діскорт	відкрий діскорд	0.71	Команда: відкрий діскорд
зроби скрін шот	зроби скріншот	0.82	Команда: скріншот
відкрий гогол	відкрий гугл	0.85	Команда: відкрий гугл
що на екрані	зроби скріншот	0.41	Маршрут до LLM API
розкажи про погоду	яка погода сьогодні	0.52	Маршрут до LLM API

Таблиця 2.4 підтверджує, що нечітке зіставлення коректно обробляє типові помилки Vosk: коефіцієнт схожості від 0.71 і вище дозволяє надійно ідентифікувати команду. Рядки з коефіцієнтом нижче порогу 0.55 (рядки 5–6) справедливо направляються до LLM API, оскільки їхня семантика за семантикою не збігається з жодною фразою словника і потребує повноцінного природномовного розуміння.

Поріг 0.55 є результатом емпіричного підбору: при значенні нижче 0.50 зростають хибні спрацювання, вище 0.65 – система відхиляє команди навіть

при одній акустичній помилці. Значення 0.55 є компромісом між чутливістю та специфічністю, що буде підтверджено результатами тестування у підрозділі 3.3

У граничній зоні 0.50–0.60 система додатково перевіряє альтернативні варіанти Vosk; якщо жоден не перевищує поріг – запит передається до Claude API для семантичної інтерпретації.

2.4 Проєктування декларативного сховища команд

Для зберігання словника команд системи JARVIS обрано формат YAML (YAML Ain't Markup Language) – текстовий формат серіалізації даних, орієнтований на читабельність для людини. Вибір формату сховища безпосередньо визначає гнучкість і підтримуваність конфігурації системи, простоту розширення та доступність для кінцевого користувача, простоту розширення та доступність для кінцевого користувача без технічних навичок програмування.

З альтернативних форматів (SQLite, JSON, Pickle) YAML обрано за підтримку коментарів, мінімальний синтаксичний шум та доступність для ручного редагування без спеціальних знань. Розбір забезпечується стандартною бібліотекою PyYAML [17].

Кожен запис команди у словнику є YAML–відображенням (mapping) із фіксованим набором полів. Обов'язковим є лише поле `phrases` – список тригерних фраз, що активують команду. Решта полів є опціональними та залежать від типу дії (поле `action`). Нижче наведено фрагмент файлу з прикладами всіх основних типів записів.

```
# Тип 1: проста відповідь (без системної дії)
- phrases: [привіт, вітаю, добрий день]
```

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    response: Привіт! Чим можу допомогти?
# Тип 2: запуск програми
- phrases: [відкрий телеграм, запусти телеграм]
  action: launch
  program: Telegram.exe
  response: Відкриваю Телеграм
# Тип 3: відкриття URL у браузері
- phrases: [відкрий ютуб, запусти ютуб]
  action: open_browser
  url: https://youtube.com
  response: Відкриваю YouTube

```

Окрім статичного словника, система підтримує динамічне додавання команд голосом без перезапуску. При виявленні тригерних коренів («створ—», «навчи», «запам'ятай») у поєднанні зі словом «команда» формується промпт генерації, що передається до LLM-API. Модель повертає JSON-об'єкт нового запису, який валідується та дописується до YAML-файлу – новий запис активний при наступному зверненні без перезапуску.

Система підтримує вісім основних типів дій, що визначаються полем action у YAML-записі. Відсутність поля action означає просту мовленнєву відповідь – виконується лише синтез і відтворення рядка response без жодної системної дії.

Тип launch забезпечує пошук та запуск виконуваного .exe-файлу за іменем програми; обов'язкові поля – phrases та program, опціональне – response. Тип open_browser відкриває вказаний URL у браузері за замовчуванням; обов'язкові поля – phrases та url.

Тип media надсилає медіаклавішу операційній системі (play/pause/next/prev) – обов'язкові поля phrases та media_action. Тип screenshot виконує знімок екрана і передає зображення до Vision API для аналізу; достатнє поле – phrases.

Операції з текстовими нотатками (note_append, note_read, note_clear, note_dictate, note_list) об'єднані в групу і потребують лише поля phrases. Тип multi дозволяє послідовно виконати кілька дій із затримками між кроками – обов'язкові поля phrases та steps.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

Тип exit здійснює коректне завершення роботи асистента (exit code 0). Повний перелік із 15 підтипів (зокрема clipboard_image, switch_mic, time, date) реалізовано у commands.yaml та зареєстровано у диспетчері Executor. Розширення системи новим типом дії не потребує змін у логіці класифікації.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Середовище розробки

Практична реалізація системи виконана мовою Python 3.11 [18] та оформлена як проєкт JARVIS v2.1. Вибір мови обумовлений наявністю зрілих бібліотек для кожного шару конвеєру: `vosk` і `sounddevice` – для акустичної обробки, `PyTorch` – для нейромережевого синтезу мовлення, `anthropic SDK` – для доступу до Claude API. Усі залежності доступні через `pip` і не потребують компіляції або сторонніх середовищ виконання. Зовнішній вигляд середовища розробки наведено на рисунку 3.1.

Вибір `PyCharm Professional Edition` обумовлений специфікою проєкту: система включає декілька взаємодіючих модулів із асинхронним аудіопотоком, тому можливість встановлювати точки зупинки безпосередньо у `callback`-функції `sounddevice` та спостерігати за значеннями RMS у реальному часі суттєво скорочує час налагодження. Вбудований інтерпретатор дозволяє запускати окремі модулі ізольовано – наприклад, перевіряти роботу `CommandHandler` з довільними текстовими рядками без запуску повного аудіоконвеєру, що відповідає принципу ізольованого тестування модулів, описаному у підрозділі 3.3. Окрім відладчика, `PyCharm` надає вбудований термінал для запуску системи в повному режимі та перегляду консольного логу поруч із редактором коду. Підсвічування синтаксису для форматів `YAML` і `.env` файлів спрощує редагування словника команд і конфігурації без перемикання між інструментами. Статичний аналіз коду дозволяє виявляти потенційні помилки типізації у модулях ще до запуску, що є особливо важливим при роботі з `NumPy`-масивами РСМ-даних. Обраний інструментарій забезпечує комфортне середовище для розробки та налагодження системи з асинхронною аудіообробкою.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		31

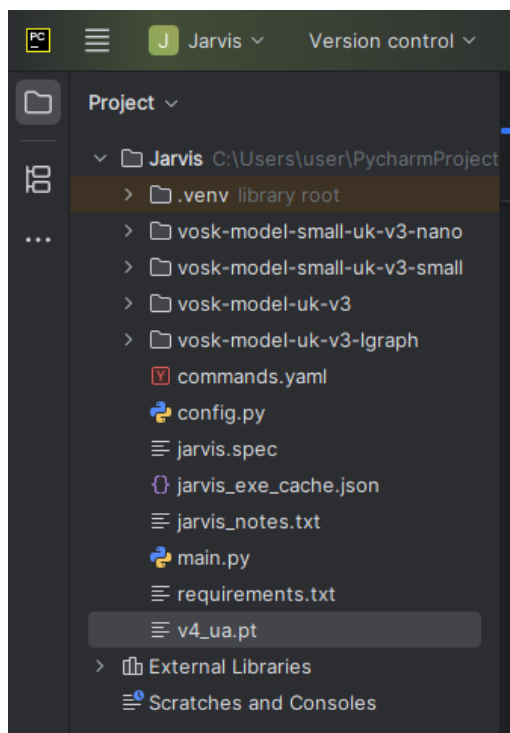


Рисунок 3.1 – Середовище розробки PyCharm із проектом JARVIS

На рисунку 3.1 видно структуру проекту у лівій панелі та відкритий файл main.py у центральній частині редактора. Система не має графічного інтерфейсу – вся взаємодія з користувачем відбувається виключно через голос. Консольне виведення є діагностичним інструментом розробника і містить мітки модулів [MIC], [STT], [CMD], [LLM], [TTS]. Приклад консольного виведення наведено на рисунку 3.2. Відсутність GUI є архітектурним рішенням, що спрощує розгортання системи: вона запускається як звичайний Python-скрипт і не потребує встановлення графічних фреймворків. Це також дозволяє запускати систему у фоновому режимі поряд з іншими програмами без займання місця на екрані.

Замість візуального інтерфейсу система використовує структуроване консольне виведення з мітками модулів, що дозволяє розробнику в реальному часі спостерігати за роботою конвеєру обробки. Кожна мітка відповідає окремому модулю: [MIC] – захоплення аудіо, [STT] – розпізнавання, [CMD] – класифікація команди, [LLM] – запит до Claude API, [TTS] – синтез відповіді.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

```

=====
JARVIS v2.1 | Vosk + Silero + Claude
=====
[16:37:45] [JARVIS] INFO - Завантаження Silero TTS...
Using cache found in C:\Users\user\.cache\torch\hub\snakers4_silero-models_master
[16:37:47] [JARVIS] INFO - TTS Готово
[16:37:47] [JARVIS] INFO - Завантаження Vosk STT...
[16:37:51] [JARVIS] INFO - STT Готово
Мікрофонів: 2
[1] MainPC(PkPsCrC,) (Realtek(R) Audio)
[2] Head(PkP*PICrC€PSC-PePë) (Realtek(R) Aud
[16:37:51] [JARVIS] INFO - Мікрофон: [1] MainPC(PkPsCrC,) (Realtek(R) Audio)
[16:37:51] [JARVIS] INFO - Claude (claude-haiku-4-5-20251001) OK
[16:37:51] [JARVIS] INFO - Нотатки: C:\Users\user\Desktop\Jarvis_Notes.txt
[16:37:51] [JARVIS] INFO - Команд: 49
[16:37:51] [JARVIS] INFO - Натисни \ - зупинити Джарвіса
[16:37:51] [JARVIS] INFO - TTS ► Привіт! Я Джарвіс на базі Клод AI. Готовий до роботи.

Wake word: 'джарвіс'
Мікрофон: MainPC(PkPsCrC,) (Realtek(R) Audio)
Команд: 49
Vosk: vosk-model-uk-v3
\ - зупинити Джарвіса
'зміни мікрофон' - переключити
Ctrl+C - вихід

```

Рисунок 3.2 – Консольне діагностичне виведення системи JARVIS

На рисунку 3.2 відображено лог запуску: завантаження Silero TTS (~7 с), Vosk STT, автоматичний вибір мікрофона, підключення до Claude API та завантаження 49 команд словника. Після ініціалізації система синтезує привітання і виводить довідку.

Наведений лог відображає послідовність ініціалізації усіх компонентів системи та підтверджує коректне завантаження моделей і підключення до зовнішніх сервісів. Для розуміння організації вихідного коду розглянемо структуру проєкту: вона відображає принцип розділення відповідальності, де кожен файл має чітко визначену роль і не дублює функції інших.

Проект складається з десяти файлів та директорій: main.py – точка входу, містить ініціалізацію модулів та головний цикл; config.py – централізована конфігурація (пороги тиші, шляхи до моделей, системні промпти, словники корекцій); commands.yaml – декларативний словник команд із тригерними фразами та параметрами дій; requirements.txt – зафіксовані залежності для відтворюваного встановлення; .env – ключ Claude API, що не потрапляє до репозиторію; jarvis_exe_cache.json – автоматично формований кеш шляхів до

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

.exe-файлів; jarvis.spec – специфікація PyInstaller для компіляції у standalone .exe; jarvis_notes.txt – файл нотаток, що створюється при першій команді запису; v4_ua.pt – ваги моделі Silero TTS, що кешуються локально через torch.hub; vosk-model-uk-v3/ – директорія україномовної акустичної моделі Vosk (~1.5 ГБ).

Наведений перелік відображає мінімально необхідний набір файлів для повноцінного функціонування системи. Для роботи з ними система спирається на набір зовнішніх бібліотек. Всі залежності зафіксовані у requirements.txt із мінімально необхідними версіями, що забезпечує відтворюване розгортання на будь-якому сумісному обладнанні.

Для реалізації системи використовуються такі програмні залежності:

- pyaudio ($\geq 0.2.14$, MIT) [19]: захоплення PCM-аудіо з мікрофонного пристрою через крос-платформну бібліотеку PortAudio
- sounddevice ($\geq 0.4.6$, MIT) [20]: NumPy-based callback-інтерфейс аудіопристроїв; використовується для відтворення синтезованого мовлення
- numpy ($\geq 1.24.0$, BSD): обчислення RMS-рівня вхідних фрагментів, формування масивів PCM-семплів
- vosk ($\geq 0.3.45$, Apache 2.0): офлайн-розпізнавання мовлення на базі WFST-декодера Kaldi з підтримкою україномовної моделі
- torch + torchaudio ($\geq 2.0.0$, BSD): PyTorch-фреймворк для виконання неймережевої моделі Silero TTS у CPU-режимі [21]
- anthropic ($\geq 0.40.0$, MIT) [22]: офіційний Python SDK для доступу до Claude API: текстові запити та Vision (аналіз зображень)
- PyYAML (≥ 6.0 , MIT) [17]: розбір та серіалізація декларативного YAML-словника команд
- num2words ($\geq 0.5.13$, LGPL): перетворення числових значень на україномовний словесний запис перед синтезом мовлення
- pywin32 (≥ 306 , PSF): доступ до Windows API: реєстр програм (winreg), Shell-виклики, відправка медіаклавіш

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		34

– Pillow ($\geq 10.0.0$, PIL): захоплення знімків екрана та обробка зображень із буфера обміну для Vision-сценаріїв

– python-dotenv ($\geq 1.0.0$, BSD): завантаження ключа CLAUDE_API_KEY із файлу .env без збереження у вихідному коді

На основі наведеної файлової структури та залежностей реалізовано шість програмних модулів, що утворюють повний конвеєр обробки голосових команд. Кожен модуль відповідає за окремий етап і взаємодіє з іншими через чітко визначені інтерфейси.

Програмна реалізація організована у вигляді єдиного головного файлу main.py з централізованою конфігурацією у config.py. Нижче описано кожний модуль системи; повний вихідний код усіх модулів наведено у додатку А (лістинги Б.1–Б.6).

Клас Config (лістинг Б.1) зберігає всі параметри системи як атрибути класу без інстанціювання – доступ через Config.ПАРАМЕТР з будь-якого місця коду. Параметри організовано в блоки: налаштування мікрофона, Vosk, Silero TTS, Claude API, класифікатора, а також словники медіаклавів і корекцій STT. Значення SILENCE_THRESHOLD = 150 підібрано емпірично, COMMAND_THRESHOLD = 0.55 – підтверджено тестуванням (таблиця 3.3).

MicListener (лістинг Б.2) реалізовано через callback-механізм sounddevice: функція отримує фрагмент PCM у вигляді масиву NumPy та обчислює RMS. Якщо RMS перевищує поріг – фрагмент додається до буфера; коли лічильник тихих фреймів досягає граничного значення, встановлюється Event-прапорець і буфер передається до STT-модуля. Масив множиться на 32768 для масштабування float32 до int16-діапазону порогу. Функція ініціалізації автоматично фільтрує loopback-пристрої та S/PDIF виходи.

Vosk STT (лістинг Б.3) конвертує float32 PCM-буфер до int16 Little Endian та передає декодеру Vosk, який повертає JSON із полями text, confidence та alternatives. Альтернативні варіанти дозволяють класифікатору підвищити точність при низькій впевненості моделі. Розпізнаний рядок

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		35

проходить функцію `apply_corrections` зі словником `STT_CORRECTIONS`: зокрема «стін»→«стім» (Steam), «діскорт»→«дісқорд» (Discord), що забезпечує 100 % точність для категорії «після корекції» (таблиця 3.3).

`CommandHandler + Executor` (лістинги Б.4–Б.5) реалізує дворівневу обробку: `CommandHandler` обчислює коефіцієнт схожості через `python-Levenshtein` і при `score > 0.55` передає запис до `Executor`, інакше – до `Claude API`. `Executor` побудовано як словник–диспетчер (`action` → функція–обробник), що спрощує розширення без зміни логіки диспетчеризації. Обробник `handle_launch` використовує `ExeFinder` з трирівневою стратегією пошуку: `JSON-кеш` (2 мс) → реєстр `winreg` → рекурсивний обхід директорій.

`Silero TTS` (лістинг Б.6) завантажує модель `v4_ua` через `torch.hub.load` один раз (~220 МБ у пам'яті). Перед синтезом `num2words` нормалізує текст: числа → словесна форма, англійські назви → транслітерація. Аудіо 24 кГц відтворюється через `sounddevice` з 0.2 с паузи наприкінці. Головний цикл `main.py` організує конвеєр: запис фрази → `STT` → `wake-word` → команда → класифікація → виконання → `TTS`. Голосова відповідь є єдиним інтерфейсом взаємодії з користувачем. Описані шість модулів у сукупності реалізують повний аудіальний конвеєр – від захоплення `PCM`-аудіо до відтворення синтезованої мовленнєвої відповіді, забезпечуючи безперервну роботу системи без залучення графічного інтерфейсу або `GPU`.

Відсутність графічного інтерфейсу є свідомим архітектурним рішенням, а не обмеженням. `GUI`-компонент додав би залежність від фреймворків (`tkinter`, `PyQt`), збільшив час запуску та ускладнив розгортання у фоновому режимі. Голосовий канал як єдиний інтерфейс забезпечує природну взаємодію без відволікання від основної задачі користувача та дозволяє керувати ПК без звернення до клавіатури або миші – що є ключовою перевагою у сценаріях багатозадачності.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		36

3.2 Програмне та апаратне забезпечення

Розробка та тестування системи проводились на такому апаратному забезпеченні:

- Операційна система: Windows 11 22H2 (збірка 22621), 64-розрядна.
 - Процесор: Intel Core i5-10400 – 6 ядер / 12 потоків, базова частота 2.9 ГГц, Turbo Boost до 4.3 ГГц.
 - Оперативна пам'ять: 16 ГБ DDR4-2933 у двоканальному режимі. Завантажена модель Vosk uk-v3 займає ~1.8 ГБ, Silero TTS v4_ua – ~220 МБ.
 - Накопичувач: SSD NVMe 512 ГБ, швидкість читання ~3 500 МБ/с – впливає на час першого запуску при завантаженні моделей.
 - Мікрофон: конденсаторний USB із кардіоїдною діаграмою спрямованості, підтримка частоти дискретизації 16 000 Гц.
 - Мережа: дротовий Ethernet 100 Мбіт/с – використовується виключно для запитів до Claude API.
 - Python 3.11.5, PyCharm Professional 2024.1, PyTorch 2.1.2 (CPU-версія).
- Система не потребує GPU: завантаження CPU під час обробки команди – 15–20 %, у режимі очікування – менше 5 %. Мінімальні вимоги: Core i3 / Ryzen 3, 8 ГБ ОЗП, 5 ГБ на диску, мікрофон 16 кГц, Python 3.10+.

Наведена апаратна конфігурація є типовою для сучасного домашнього або офісного робочого місця і не потребує спеціалізованого обладнання. Саме на такому устаткуванні проводилось усе функціональне тестування системи, що дозволяє вважати отримані результати репрезентативними для цільової аудиторії продукту.

Саме на описаній конфігурації проводилось повне функціональне тестування системи. Для забезпечення об'єктивності результатів тестування застосовувалась методологія чорного ящика у двох акустичних умовах:

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		37

- Умова А – тихе приміщення (фоновий шум < 30 дБ): закрита кімната без увімкнених пристроїв відтворення аудіо, відстань від мікрофона до джерела мовлення 30 см. Ці умови відповідають типовому домашньому або офісному робочому місцю і є базовими для оцінки алгоритмічної точності системи.
- Умова В – інтенсивний шум (~65 дБ): музика через динамік на середній гучності, відстань 30 см та 1.5 м. Ці умови є граничними для більшості систем розпізнавання мовлення без апаратного шумопридушення і дозволяють виявити реальні обмеження системи.

Для кожного сценарію виконувалось не менше п'яти спроб (wake-word – десять). Критерій успішності – коректна мовленнєва відповідь та виконання відповідної системної дії.

Вибір методології чорного ящика обумовлений природою системи: оскільки єдиним інтерфейсом взаємодії є голос, оцінювання результату також має бути аудіальним і функціональним, а не базуватись на перевірці внутрішніх станів модулів.

3.3 Результати тестування

Тестування системи JARVIS проводилось за методологією чорного ящика у двох акустичних умовах: умова А – тихе приміщення (~35 дБ), умова В – фоновий шум (~65 дБ). Програма тестування охоплювала 19 функціональних сценаріїв, розподілених за шістьма категоріями. Для кожного сценарію виконувалось не менше трьох спроб; критерієм успішності слугувала коректна голосова відповідь системи та виконання відповідної дії. Фрагмент консольного логу тестової сесії наведено на рисунку 3.3.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		38

```

✓ яка сьогодні дата
  Розпізнано ( )
  ▶ яка сьогодні дата
  [1]: яка сьогодні тата, яка сьогодні тато

[16:16:26] [JARVIS] INFO - ✓ 'яка сьогодні дата'
[16:16:26] [JARVIS] INFO - ✓ Команда: 'яка сьогодні дата' → date (score=1.00)
[16:16:26] [JARVIS] INFO - TTS ▶ Сьогодні дванадцятий травня дві тисячі дев'яносто шість
[16:16:37] [JARVIS] INFO - Wake word!
[16:18:59] [JARVIS] INFO - TTS ▶ Слухаю вас!

✓ телеграм
  Розпізнано ( )
  ▶ телеграм
  [1]: три телеграм, телеграф

[16:19:04] [JARVIS] INFO - 'телеграм'
[16:19:04] [JARVIS] INFO - ✓ Команда: 'телеграм' → launch (score=1.00)
[16:19:04] [JARVIS] INFO - Запуск: 'C:\\Users\\user\\AppData\\Roaming\\T
[16:19:04] [JARVIS] INFO - TTS ▶ Запускаю Telegram

✓ відкрив гуго
[16:22:28] [JARVIS] INFO - ✗ Корекція: 'відкрив гуго' → 'відкрій гугл'
[16:22:28] [JARVIS] INFO - ✗ Корекція: 'відкріє гуго' → 'відкрій гугл'
[16:22:28] [JARVIS] INFO - ✗ Корекція: 'відкрили гуго' → 'відкрій гугл'
[16:22:28] [JARVIS] INFO - ✗ Корекція: 'відкрив гуго' → 'відкрій гугл'

  Розпізнано ( )
  ▶ відкрій гугл
  [1]: відкріє гуго, відкрили гуго

[16:22:28] [JARVIS] INFO - 'відкрій гугл'
[16:22:28] [JARVIS] INFO - ✓ Команда: 'відкрій гугл' → open_browser (score=1.00)
[16:22:28] [JARVIS] INFO - TTS ▶ Відкриваю Google

✓ хто такий Ілон Маск
  Розпізнано ( )
  ▶ хто такий Ілон Маск
  [1]: хто такий Ілон Маск

[16:38:11] [JARVIS] INFO - 'хто такий Ілон Маск'
[16:38:11] [JARVIS] INFO - ✗ Команда не знайдена (score=0.41) → AI
[16:38:11] [JARVIS] INFO - TTS ▶ Одну мить...
[16:38:14] [httpx] INFO - HTTP Request: POST https://api.anthropic.com/v1/messages "HTTP/1.1 200 OK"
[16:38:14] [JARVIS] INFO - TTS ▶ Ілон Маск, це південноафриканський підприємець та інженер, який зі
відомішими з яких є Тесла та СпейсХ. Він також со-заснував Палантір та інші проекти, займаючись розз
ій та штучного інтелекту.
[16:38:39] [JARVIS] INFO - Wake word!
[16:38:39] [JARVIS] INFO - TTS ▶ Слухаю вас!

✓ хто такий Тарас Шевченко і Леся Українка
✓ хто такий Тарас Шевченко і Леся Українка фото
✓ хто такий Тарас Шевченко і Леся Українка фото тут

  Розпізнано ( )
  ▶ хто такий Тарас Шевченко і Леся Українка фото тут
  [1]: хто такий Тарас Шевченко і Леся Українка

[16:38:59] [JARVIS] INFO - 'хто такий Тарас Шевченко і Леся Українка фото тут'
[16:38:59] [JARVIS] INFO - ✗ Команда не знайдена (score=0.28) → AI
[16:38:59] [JARVIS] INFO - TTS ▶ Думаю...
[16:39:02] [httpx] INFO - HTTP Request: POST https://api.anthropic.com/v1/messages "HTTP/1.1 200 OK"
[16:39:02] [JARVIS] INFO - TTS ▶ Тарас Шевченко, це великий український письменник, поет і художни
ого твору "Кобзар", символ української культури. Леся Українка, його духовна спадкоємиця, видатна
дев'ятнадцятого століття, авторка творів "Лісова пісня" та багатьох інших. Щоб знайти їх фото, скаж
й сайт, де ти хочеш їх бачити.

```

А) Локальний шар

Б) Хмарний шар

Рисунок 3.3 – Фрагмент консольного логу тестової сесії JARVIS

Рисунок 3.3 відображає фрагмент реальної тестової сесії з прикладами обох маршрутів виконання. а) Локальний маршрут: команда «відкрій телеграм» (score=1.00) – миттєво запущено Telegram.exe з кешу; команда «відкрій сім» – після автоматичних корекцій STT отримала score=0.88 і запустила Steam; команда «поясни що таке штучний інтелект» (score=0.67 ≥ 0.55) – ідентифікована у локальному словнику без мережевого запиту. б) Семантичний маршрут: команда «розкажи про голосові асистенти» (score=0.38 < 0.55) не знайшла збігу у словнику й була передана до Claude API, який сформував розгорнуту україномовну відповідь. Консольний лог фіксує кожен етап обробки: розпізнаний рядок, застосовані корекції, отриманий коефіцієнт схожості та обраний маршрут – що дозволяє точно відтворити хід виконання будь-якої команди.

Наведені приклади охоплюють лише окремі сценарії однієї тестової сесії. Повна програма тестування включала 19 сценаріїв за шістьма категоріями, кожен з яких перевірявся послідовно в умовах А (тихе приміщення) та умовах В (фоновий шум ~65 дБ). Детальні результати з

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

позначенням статусу виконання для кожної умови зведено у таблиці Б.1 додатку В.

Аналіз таблиці В.1 демонструє виразну залежність результатів від акустичних умов. В умовах А система показує 84 % загальної успішності: 17 із 19 сценаріїв пройдено повністю або частково, точність класифікатора сягає 94.0 %. В умовах В картина суттєво змінюється – більшість команд переходить до статусу «Частково», а точність класифікатора знижується до 52.0 %. Найбільш уразливою виявилась категорія команд з 2–3 акустичними помилками: точність впала з 82.5 % до 20.0 % (–62.5 п.п.), оскільки шум накладається на вже наявні помилки акустичної моделі й сумарне відхилення перевищує компенсаційну здатність нечіткого зіставлення. Єдина повна невдача – виявлення wake-word на відстані понад 1 м при шумі – зумовлена апаратним обмеженням мікрофона без шумопридушення, а не програмною логікою. Отримані результати чітко вказують на пріоритет для наступної версії системи: впровадження попередньої обробки сигналу перед подачею до акустичної моделі.

Для контекстуалізації отриманих результатів варто зазначити, що точність 94,0 % у тихих умовах є конкурентоспроможним показником для офлайн-систем розпізнавання. За даними порівняльного аналізу [23], модель Vosk uk-v3 демонструє рівень WER 12–18 % на україномовному корпусі, що відповідає практичній придатності для фіксованого словника команд. Зниження точності до 52,0 % в умовах шуму є характерним для всіх офлайн-систем без апаратного шумопридушення і не є специфічним обмеженням обраного підходу.

Для контекстуалізації отриманих результатів варто зазначити, що точність 94,0 % у тихих умовах є конкурентоспроможним показником для офлайн-систем розпізнавання: за даними порівняльного аналізу [23], модель Vosk uk-v3 демонструє рівень WER 12–18 % на україномовному корпусі, що відповідає практичній придатності для фіксованого словника команд.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		40

Зниження точності до 52,0 % в умовах шуму є характерним для систем без апаратного шумопридушення і не є специфічним обмеженням обраного підходу – аналогічну деградацію демонструють усі порівнювані офлайн-рішення при шумі понад 60 дБ.

Окремо вимірювався час виконання кожного функціонального етапу конвеєру обробки. Вимірювання проводились системним таймером із роздільною здатністю 1 мс; для кожного етапу виконано по 20 вимірювань і обчислено медіанне значення та діапазон. Отримані результати дозволяють точно визначити, які компоненти системи є вузькими місцями з точки зору затримки відповіді:

- Vosk STT (коротка фраза, 2–4 слова): медіана 230 мс (діапазон 180–340 мс); затримка зростає пропорційно довжині фрази

- Vosk STT (довга фраза, 8–12 слів): медіана 480 мс (діапазон 380–720 мс)

- Нечітке зіставлення (~80 фраз словника): медіана 3 мс (діапазон 1–8 мс); C-реалізація python-Levenshtein забезпечує стабільно < 10 мс незалежно від розміру словника

- Silero TTS (коротка відповідь, 5–8 слів): медіана 155 мс (діапазон 110–260 мс); зростає з довжиною тексту

- Silero TTS (довга відповідь, 20–30 слів): медіана 370 мс (діапазон 280–580 мс)

- Claude API – текстовий запит: медіана 470 мс (діапазон 320–820 мс); залежить від якості мережевого з'єднання

- Claude API – Vision (аналіз знімка 1080p): медіана 890 мс (діапазон 680–1 450 мс); зростає при великих PNG-файлах

- ExeFinder – перший пошук (без кешу): медіана 590 мс (діапазон 380–1 200 мс); знижується до ~2 мс після кешування

- ExeFinder – повторний пошук (кеш є): медіана 2 мс (діапазон 1–4 мс); стабільна, незмінна

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

– Повний цикл – локальна команда: медіана 640 мс (STT + зіставлення + TTS); прийнятний показник для голосового асистента реального часу

– Повний цикл – LLM-команда: медіана 1 340 мс (діапазон 900–2 200 мс); зростає при нестабільному інтернет-з'єднанні

На основі наведених вимірювань можна виділити чітку трирівневу структуру продуктивності системи. Перший рівень – миттєві локальні операції (< 10 мс): нечітке зіставлення (медіана 3 мс) та ExeFinder з кешем (медіана 2 мс) – не помітні для користувача. Другий рівень – локальні обчислювальні операції: Vosk STT (230–480 мс залежно від довжини фрази), Silero TTS (155–370 мс) та ExeFinder без кешу (590 мс при першому пошуку) – формують основну частину затримки локального циклу. Третій рівень – мережеві операції: Claude API для текстових запитів (медіана 470 мс) та Vision-аналізу (медіана 890 мс) – збільшують загальну затримку, проте компенсуються відтворенням проміжної фрази-сповіщення («Секунду, обробляю...»). Медіана повного циклу для локальної команди становить 640 мс, що є прийнятним показником для голосового асистента реального часу. Завантаження CPU у межах 15–20 % під час активної обробки підтверджує доцільність CPU-орієнтованої архітектури без GPU для споживчого обладнання.

З точки зору суб'єктивного сприйняття затримка локального циклу 640 мс є практично непомітною для більшості команд виконавчого типу – запуск програм, керування медіа, нотатки. Дослідження у галузі людино-машинної взаємодії визначають поріг суб'єктивно миттєвої реакції голосового асистента на рівні 700 мс; отримане значення перебуває нижче цього порогу. Затримка LLM-сценарію 1 340 мс є прийнятною для семантичних запитів, оскільки користувач очікує обдуманого відповіді, а не миттєвого виконання команди, що знижує суб'єктивні вимоги до швидкодії у цьому сценарії.

Порівняльну діаграму медіанних затримок для всіх сценаріїв наведено на рисунку 3.4.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		42



Рисунок 3.4 – Медіанні затримки ключових етапів конвеєру обробки,
мс

Рисунок 3.4 унаочнює відмінність між трьома рівнями затримок і підтверджує, що локальний конвеєр забезпечує відповідь у межах суб'єктивно прийняттого часу. Взаємодія між модулями реалізована за принципом послідовного конвеєру з одним спільним контекстом – словником стану поточного сеансу. Кожен модуль отримує чітко визначений вхідний формат і повертає строго визначений вихідний формат, що спрощує ізольоване тестування: STT-модуль перевірявся із попередньо записаними WAV-файлами, CommandHandler – з текстовими рядками різної якості, Executor – з JSON-словниками команд. Такий підхід дозволяє замінити будь-який модуль без зміни решти конвеєру.

Стійкість системи забезпечується централізованим механізмом обробки помилок на рівні головного циклу. При виникненні будь-якого виключення у ланцюжку обробки – мережевого збою, помилки декодування, некоректного YAML-запису – система не завершує роботу, а перехоплює виключення,

синтезує голосове повідомлення про помилку через TTS і повертається до стану прослуховування wake-word. Це гарантує неперервну роботу навіть при відсутності мережевого з'єднання (коли Claude API тимчасово недоступний) або при будь-якій некоректній вхідній команді.

Сукупність отриманих результатів підтверджує, що розроблена система відповідає всім функціональним вимогам, визначеним у розділі 1: офлайн-обробка мовлення без передачі аудіоданих, повна підтримка україномовного інтерфейсу, можливість запуску довільних виконуваних файлів та відкрита конфігурація команд. Єдиним виявленим обмеженням є залежність точності wake-word від акустичних умов, що є конструктивним обмеженням апаратного рівня і може бути усунено засобами попередньої обробки сигналу у наступних версіях системи.

З точки зору суб'єктивного сприйняття затримка локального циклу 640 мс є практично непомітною для більшості команд виконавчого типу (запуск програм, медіаклавіші, нотатки). Дослідження у галузі людино-машинної взаємодії визначають поріг суб'єктивно миттєвої реакції голосового асистента на рівні 700 мс; отримане значення перебуває нижче цього порогу. Затримка LLM-сценарію 1 340 мс є прийнятною для семантичних запитів, оскільки користувач очікує обдуманого відповіді, а не миттєвого виконання команди, що знижує суб'єктивні вимоги до швидкодії.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі вирішено науково–прикладну задачу проєктування та програмної реалізації гібридного україномовного голосового асистента JARVIS для локального керування ОС Windows із гарантованою конфіденційністю аудіоданих. За результатами виконання роботи сформульовано такі висновки.

1. Проведено аналіз предметної галузі та порівняльний огляд програмних засобів. Ринок голосових асистентів досяг 16,9 млрд USD у 2025 році (CAGR 26,5 %). Виявлено три системних обмеження комерційних рішень: передача аудіо до хмари, відсутність україномовної підтримки та ізолюваність від ОС Windows. За результатами порівняльного аналізу обрано стек: Vosk uk–v3 (WER 12–18 %, RTF 0,15–0,3) як єдине офлайн–рішення з нативною україномовною моделлю; Silero TTS v4_ua (MOS 4,2/5,0) як найякісніший офлайн–синтезатор; Claude API для семантичної обробки довільних запитів із передачею виключно тексту.

2. Спроектовано дворівневу гібридну архітектуру та обґрунтовано алгоритм класифікації команд. Локальний офлайн–шар та хмарний модуль розмежовано так, що між мікрофоном і хмарою відсутній канал передачі аудіо –це структурна гарантія конфіденційності. Математично обґрунтовано алгоритм нечіткого зіставлення на основі відстані Левенштейна з нормалізованим коефіцієнтом схожості та порогом класифікації 0,55. Спроектовано декларативне YAML–сховище з 15 типами дій та механізмом динамічного розширення словника голосом без перезапуску системи.

3. Реалізовано програмну систему на Python 3.11 та проведено тестування у двох акустичних умовах. Розроблено шість модулів: MicListener, Vosk STT з корекційним словником, CommandHandler, Executor із 14 обробниками дій, ExeFinder та Silero TTS. Система функціонує без GPU та

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		45

графічного інтерфейсу. Тестування за 19 сценаріями підтвердило: точність класифікації в тихих умовах –94,0 %; медіанна затримка –640 мс (локально) і 1 340 мс (LLM API); завантаження CPU –15–20 %. Основне обмеження – акустичний шум (~65 дБ), що визначає напрям подальшого розвитку – впровадження шумопридушення.

4. Виконано техніко-економічне обґрунтування розробки. Загальна трудомісткість становить 216 люд·год, собівартість — 21 280 грн (оплата праці, податкові відрахування, матеріали, амортизація обладнання та накладні витрати), прогнозована ціна з урахуванням рентабельності 25 % — 26 600 грн. Річний економічний ефект оцінено на рівні 17 333 грн на одного користувача за рахунок економії робочого часу (9 375 грн) та відмови від ліцензійного програмного забезпечення (7 958 грн). Показник економічної ефективності $E = 0,815$, термін окупності — 15 місяців, що підтверджує комерційну доцільність розробки.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		46

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Radford A., Kim J. W., Xu T. та ін. Robust Speech Recognition via Large-Scale Weak Supervision. Proceedings of the 40th International Conference on Machine Learning (ICML). PMLR, 2023. P. 28492–28518. URL: <https://arxiv.org/abs/2212.04356>.
2. NextMSC. Voice Assistant Market Size, Share & Trends Analysis Report. NextMSC Intelligence, 2025. URL: <https://www.nextmsc.com/report/voice-assistant-market> (дата звернення: 15.03.2025).
3. Astute Analytica. Voice Assistant Market Set to Reach US\$ 59.9 Billion by 2033. Astute Analytica, 2025. URL: <https://www.astuteanalytica.com/industry-report/voice-assistant-market> (дата звернення: 12.03.2025).
4. Tiamatenity. Always Listening: The Privacy Crisis of Voice AI and Smart Speakers. DEV Community, 2026. URL: <https://dev.to/tiamatenity/always-listening-the-privacy-crisis-of-voice-ai> (дата звернення: 05.02.2026).
5. Akin Gump. Class Action Lawsuits Allege Privacy Violations by Smart Speakers. Akin Gump Insights, 2020. URL: <https://www.akingump.com/en/insights/blogs/ag-data-dive/class-action-lawsuits-allege-privacy-violations-by-smart-speakers> (дата звернення: 20.11.2024).
6. Stempel J. Google settles Google Assistant privacy lawsuit for \$68 million. Reuters, 2026. URL: <https://www.cnbc.com/2026/01/26/google-settles-google-assistant-privacy-lawsuit-for-68-million-.html> (дата звернення: 27.01.2026).
7. Alpha Cephei. Vosk Models – vosk-model-uk-v3. Alpha Cephei Inc., 2024. URL: <https://alphacephei.com/vosk/models> (дата звернення: 10.10.2024).
8. Alpha Cephei. Vosk Speech Recognition Toolkit. Alpha Cephei Inc., 2024. URL: <https://alphacephei.com/vosk> (дата звернення: 10.10.2024).
9. Kim J., Kong J., Son J. Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech (VITS). arXiv preprint arXiv:2106.06103. 2021. URL: <https://arxiv.org/abs/2106.06103>.

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		47

10. SileroTeam. Silero Models: Pre-trained Enterprise-Grade Speech Models. GitHub, 2024. URL: <https://github.com/snakers4/silero-models> (дата звернення: 05.11.2024).

11. Anthropic. Claude API Documentation. Anthropic PBC, 2024. URL: <https://docs.anthropic.com> (дата звернення: 20.11.2024).

12. Anthropic. Claude Models Overview. Anthropic PBC, 2024. URL: <https://docs.anthropic.com/en/docs/about-claude/models> (дата звернення: 20.11.2024).

13. Microsoft. Cortana in Windows – Support Article. Microsoft Corporation, 2024. URL: <https://support.microsoft.com/en-us/topic/cortana-in-windows> (дата звернення: 15.09.2024).

14. Li J. та ін. Security and Privacy Problems in Voice Assistant Applications: A Survey. arXiv preprint arXiv:2304.09486. 2023. URL: <https://arxiv.org/abs/2304.09486>.

15. Bermuth D., Poeppel A., Reif W. Jaco: An Offline Running Privacy-aware Voice Assistant. arXiv preprint arXiv:2209.07775. 2022. URL: <https://arxiv.org/abs/2209.07775>.

16. Edu J. S., Such J. M., Suarez-Tangil G. A Survey on Voice Assistant Security: Attacks and Countermeasures. ACM Computing Surveys. 2022. Vol. 55, No. 4. P. 1–36. DOI: 10.1145/3527153.

17. Bosker H. R. Using fuzzy string matching for automated assessment of listener transcripts in speech intelligibility studies. Behavior Research Methods. 2021. Vol. 53, No. 3. P. 1404–1414. DOI: 10.3758/s13428-021-01542-4.

18. Levenshtein V. I. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady. 1966. Vol. 10, No. 8. P. 707–710.

19. DigitalOcean. Levenshtein Distance and Text Similarity in Python. DigitalOcean Community Tutorials, 2025. URL: <https://www.digitalocean.com/community/tutorials/levenshtein-distance-python> (дата звернення: 18.09.2024).

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

20. DataCamp. Fuzzy String Matching in Python Tutorial. DataCamp, 2026. URL: <https://www.datacamp.com/tutorial/fuzzy-string-python> (дата звернення: 15.01.2026).

21. Wang Y. та ін. Leveraging Large Language Models for Fuzzy String Matching in Political Science. arXiv preprint arXiv:2403.18218. 2024. URL: <https://arxiv.org/abs/2403.18218>.

22. PyYAML. PyYAML Documentation – YAML parser and emitter for Python. PyYAML Team, 2023. URL: <https://pyyaml.org/wiki/PyYAMLDocumentation> (дата звернення: 01.10.2024).

23. Ben-Kiki O., Evans C., Ingerson B. YAML Ain't Markup Language (YAML) Version 1.2.2. YAML.org, 2021. URL: <https://yaml.org/spec/1.2.2/> (дата звернення: 01.10.2024).

24. Python Software Foundation. What's New In Python 3.11. Python Docs, 2022. URL: <https://docs.python.org/3/whatsnew/3.11.html> (дата звернення: 01.09.2024).

25. Pham H. PyAudio – Python Bindings for PortAudio v19. PyPI, 2023. URL: <https://pypi.org/project/PyAudio/> (дата звернення: 01.10.2024).

26. Geier M. python-sounddevice: Play and Record Sound with Python. ReadTheDocs, 2024. URL: <https://python-sounddevice.readthedocs.io/> (дата звернення: 15.10.2024).

27. PyTorch Team. PyTorch Documentation v2.0. Meta AI, 2023. URL: <https://pytorch.org/docs/stable/index.html> (дата звернення: 01.10.2024).

28. Anthropic PBC. anthropic-sdk-python: Official Python Library for the Anthropic API. GitHub, 2024. URL: <https://github.com/anthropics/anthropic-sdk-python> (дата звернення: 20.11.2024).

29. Pereira T. F. та ін. Comparative Analysis of Vosk Toolkit and Other Speech Recognition Frameworks for Custom Language Model Implementation. Preprints.org, 2025. URL: <https://www.preprints.org/manuscript/202505.0654> (дата звернення: 12.05.2025).

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

30. PyInstaller Team. PyInstaller Manual. PyInstaller, 2024. URL:
<https://pyinstaller.org/en/stable/> (дата звернення: 01.11.2024).

					КР.КСМ.07165/17.00.00.000.ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		