Міністерство освіти і науки України
Український державний університет науки і технологій

Комп'ютерних технологій і систем

Комп'ютерні інформаційні технології

Пояснювальна записка
до кваліфікаційної роботи ОС Магістра

на тему: Дослідження ефективності використання голосового помічника у веб-додатках
за освітньою програмою: 12 Вебпрограмування та комп'ютерний дизайн
зі спеціальності: 121 Інженерія програмного забезбечення

Виконав: студент группи ПЗ2422 / Ілля ЦИПА /

Керівник: / Світална ВОЛКОВА /

Нормоконтролер: / Світална ВОЛКОВА /

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент

Дніпро – 2026

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Computer technologies and systems

Computer information technology

Explanatory Note
to Master's Thesis

on the topic: Research on the effectiveness of using voice assistants in web applications

according to educational curriculum 12 Web programming and computer design

in the Speciality: 121 Software engineering

Done by the student of the group /Illia TSYPA /

Scientific Supervisor: / Svitlana VOLKOVA /

Normative controller : / Svitlana VOLKOVA/

Dnipro – 2026

Міністерство освіти і науки України Український державний університет науки і технологій

Факультет: Комп'ютерних технологій і систем
Кафедра: Комп'ютерні інформаційні технології
Рівень вищої освіти: Магістр
Освітня програма: 12 Вебпрограмування та комп'ютерний дизайн
Спеціальність: 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____/Вадим ГОРЯЧКІН/

З А В Д А Н Н Я

на кваліфікаційну роботу ОС Магістр

студенту Ципа Ілля Владиславович
1 Тема роботи: Дослідження ефективності використання голосового помічника у веб-додатках
2 Керівник роботи: Волкова Світлана Анатоліївна

затверджені наказом від "02" 10 2025 р. № 1401ст.
2. Строк подання студентом роботи: 13.01.2026 р.
3 Вихідні дані до роботи :наукова література, аналогічні програми
4 Зміст пояснювальної записки (перелік питань до розробки)
ВСТУП
РОЗДІЛ 1 Аналіз досліджень і постановка проблеми задачі
РОЗДІЛ 2 Методи та оцінки ефективності голосового введення
РОЗДІЛ 3 Проектування і розробка експерементального веб-додатку
РОЗДІЛ 4. Експерементальне дослідження
5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): презентація, відео роботи програми.

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	30.09.2025	
2	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	02.10.2025	від 70 джерел
3	Написання вступу	05.10.2025	
4	Написання постановки задачі, технічного завдання	10.11.2025	30%
5	Оцінка техніко-економічних показників	11.11.2025	
6	Розробка інструментальних засобів дослідження	25.11.2025	
7	Виконання досліджень	08.12.2025	60%
8	Оформлення тез доповідей	09.12.2025	
9	Оформлення статті у фаховий журнал	10.12.2025	
10	Оформлення пояснювальної записки	15.12.2025	
11	Розробка демонстраційних матеріалів	05.01.2026	100%
12	Подання кваліфікаційної роботи до кафедри	15.01.2026	
13	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	20.01.2026	

Студент Ілля ЦИПА

Керівник роботи Світлана ВОЛКОВА

РЕФЕРАТ

Дипломна робота на тему: «Дослідження ефективності використання голосового помічника у веб-додатках».

Стрімкий розвиток інтернету речей IoT та мобільних технологій зумовлює необхідність впровадження альтернативних методів взаємодії користувача з системою. Зважаючи на зростаючі вимоги до інклюзивності та зручності UX, голосові інтерфейси VUI стають важливою складовою сучасних веб-рішень. Тому наявність спеціалізованого програмного забезпечення, що дало б змогу експериментально дослідити часові характеристики та точність розпізнавання мови у порівнянні з традиційним графічним інтерфейсом GUI, є актуальною задачею для оптимізації веб-систем.

У роботі спроектовано та реалізовано клієнт-серверний веб-додаток, який дозволяє автоматизувати збір метрик ефективності введення даних. Використання математичного апарату (відстань Левенштейна) дозволило оцінити не тільки швидкість, а й семантичну точність голосових команд

Підстава для розробки: відсутність доступних інструментальних засобів для комплексного порівняльного аналізу ефективності VUI та GUI у веб-середовищі.

Сфера застосування: веб-розробка, UI/UX дизайн, тестування програмного забезпечення, наукові дослідження у сфері людино-машинної взаємодії..

Ключові слова: VUI, Web Speech API, Node.js, REST API, відстань Левенштейна, юзабіліті, розпізнавання мови, ефективність, клієнт-серверна архітектура, Word Error Rate.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API — Application Programming Interface

ASR — Automatic Speech Recognition

WER — Word Error Rate

JS — JavaScript

UI — User Interface

ВСТУП

Розвиток людино-машинних інтерфейсів HMI постійно прямує до спрощення взаємодії між користувачем та обчислювальною системою. Зі зростанням популярності мобільних пристроїв та технологій Інтернету речей (IoT), традиційні графічні інтерфейси GUI, засновані на мануальному введенні даних, часто стають "вузьким місцем", що знижує загальну продуктивність системи. У цьому контексті голосові інтерфейси VUI розглядаються як перспективна альтернатива, однак питання кількісної оцінки їхньої ефективності у порівнянні з традиційними методами залишається недостатньо вивченим.

Використовуючи системний підхід до аналізу веб-додатків, необхідно розглядати не лише факт наявності голосового керування, а й доцільність його впровадження з точки зору часових витрат та точності передачі інформації. Більшість існуючих досліджень фокусуються на якості розпізнавання мови ASR, залишаючи поза увагою ергономічні показники та швидкість виконання бізнес-сценаріїв користувачем.

Для отримання об'єктивних даних необхідна розробка спеціалізованого програмного інструментарію, який дозволив би автоматизувати процес збору метрик (часу реакції, тривалості введення, кількості помилок) без впливу суб'єктивних факторів.

Під час роботи було розглянуто тему «Дослідження ефективності використання голосового помічника у веб-додатках». Було проаналізовано сучасні методи оцінки юзабіліті та алгоритми порівняння текстових даних.

Мета: підвищення ефективності взаємодії користувача з веб-додатками шляхом дослідження часових та якісних характеристик голосового введення даних та визначення умов його доцільного використання.

Об’єкт: процес інформаційної взаємодії користувача з веб-системою.

Предмет: ефективність при використанні голосового та графічного інтерфейсів

РОЗДІЛ 1 АНАЛІЗ ДОСЛІДЖЕНЬ І ПОСТАНОВКА ЗАДАЧІ

1.1 Призначення та сфера застосування

Автоматизація процесів взаємодії людини з глобальною мережею Інтернет тісно перетинається з задачами покращення користувацького досвіду (User Experience) та ергономіки. Веб-додатки з голосовим керуванням (VUI — Voice User Interface) представляють собою клас програмних систем, де швидкість та зручність введення даних є критично важливими факторами для утримання користувача та ефективності виконання бізнес-процесів.

Задачі використання голосових асистентів розподіляються за контекстом застосування на такі категорії:

– «Hands-free» сценарії (вільні руки) — взаємодія з системою, коли руки користувача зайняті іншою фізичною роботою (водії, лікарі, кухарі, оператори складів)

– «Eyes-free» сценарії (без зорового контролю) — взаємодія, що не вимагає постійної візуальної уваги на екрані пристрою

– Інклюзивний доступ — забезпечення можливості роботи з веб-ресурсами для людей з обмеженими можливостями (порушення моторики, вади зору), для яких традиційні засоби введення (клавіатура, миша) є недоступними або складними у використанні.

Для вирішення задач ефективної людино-машинної взаємодії, компанії-розробники проектують і створюють веб-інтерфейси, що підтримують мультимодальне введення. Часові характеристики (швидкість реакції системи та час введення даних) слугують основним критерієм ефективності VUI. Умову доцільності використання голосового помічника можна подати у вигляді нерівності:

Таку умову можливо подати у вигляді

$$T_{VUI} + T_{err} < T_{GUI} \qquad (1.1)$$

де T_{VUI} – час, витрачений на диктування та розпізнавання команди;

T_{err} , - час, необхідний на виправлення можливих помилок розпізнавання;

T_{GUI} - час виконання тієї ж задачі традиційним графічним способом (набір тексту на клавіатурі).

Дослідження часових характеристик і методи їх поліпшення допомагають вирішити проблеми проектування архітектури сучасних веб-систем. Це, в свою чергу, прямо впливає на комерційну успішність та доступність важливих сервісів

1.2 Аналіз предметної сфери

Голосові інтерфейси VUI — це клас людино-машинних інтерфейсів, в яких основним каналом передачі інформації від людини до системи є людська мова. Сучасні системи голосового керування веб-додатками характеризуються такими властивостями:

- здатність поєднувати голосове введення з візуальним відображенням результатів GUI;
- робота у браузерному середовищі незалежно від операційної системи (Windows, macOS, Android);
- перетворення аудіосигналу в текст
- обробка запитів відбувається без блокування основного потоку інтерфейсу, що відповідає сучасним стандартам веб-розробки;
- якість розпізнавання залежить від словника та очікуваної грамати

Можна виділити декілька основних підходів до реалізації голосового керування у вебі:

Хмарні API (Server-Side ASR). Це аналог систем типу host-target, де основні обчислення відбуваються на потужному віддаленому сервері.

Google Cloud Speech-to-Text: Потужний сервіс, що підтримує понад 120 мов. Забезпечує високу точність, але є платним і вимагає постійного передавання аудіопотоку на сервери Google, що збільшує затримку (латентність).

AWS Transcribe (Amazon): Аналогічний сервіс від Amazon, орієнтований на розпізнавання довгих аудіозаписів та медичних термінів.

Azure Speech Service (Microsoft): Надає широкі можливості кастомізації акустичних моделей.

Браузерні API (Client-Side / Hybrid). Це аналог систем self-hosted, де інструменти розробки та виконання знаходяться ближче до клієнта.

Web Speech API: Нативний інтерфейс браузера, який дозволяє інтегрувати розпізнавання мови за допомогою JavaScript без підключення сторонніх бібліотек. У браузері Google Chrome він використовує хмарні потужності Google безкоштовно для користувача, але працює як "чорна скринька".

Переваги: Безкоштовність, простота інтеграції, підтримка української мови, відсутність необхідності в API-ключах.

Недоліки: Працює переважно в Chrome та Safari, вимагає HTTPS з'єднання

Підтримує засоби розробки: компілятор C, відладчик Watcom TRC.

Враховуючи популярність Google Chrome та необхідність мінімізації витрат на розробку студентського проекту, було прийнято рішення під час проектування власного програмного продукту (експериментального стенду) використовувати Web Speech API. Це дозволяє зосередитися на дослідженні часових характеристик, а не на налаштуванні складних серверних нейромереж.

Основною проблемою у веб-орієнтованих голосових інтерфейсах VUI є не

планування задач процесора (як у СРЧ), а забезпечення мінімальної латентності Latency — часу затримки між вимовою команди та реакцією системи.

На відміну від детермінованих систем, де час виконання інструкції відомий заздалегідь, процес розпізнавання мови є імовірнісним. При створенні VUI розробка має враховувати дві складові затримки:

– Network Latency (Мережева затримка): час передачі аудіопотоку на сервери розпізнавання (для хмарних рішень);

– Processing Latency (Затримка обробки): час, необхідний нейромережі для перетворення звуку в текст;

– Оцінка часу виконання голосового введення на етапі розробки ПЗ завжди викликає значний інтерес у UX-фахівців. Рішення цієї проблеми вимагає врахування специфічних факторів, що впливають на результат.

У наукових джерелах виділяють фактори, які впливають на швидкість та точність голосового введення у вебі:

– характеристики користувача: дикція, темп мовлення;

– чутливість мікрофона, наявність апаратного шумозаглушення;

– рівень фонового шуму (офіс, вулиця), наявність ехо;

– пропускна здатність каналу зв'язку (оскільки Web Speech API в Chrome відправляє дані на сервери Google);

– ініціалізація аудіо-контексту браузера займає додатковий час при першому зверненні;

Звести до мінімуму вплив відволікаючих факторів під час експерименту не завжди легко, тому в роботі пропонуються наступні прийоми (методика вимірювання):

– всі тести слід проводити на одному комп'ютері з фіксованим мікрофоном;

– тест слід запускати багаторазово (серія з 10-20 спроб), та поділити

загальний отриманий час на число ітерацій, щоб нівелювати вплив випадкових мережевих затримок;

– перший запуск розпізнавання (warm-up) не включається до статистики, оскільки браузер запитує дозвіл на мікрофон та завантажує бібліотеки;

– замість системного таймера, що має похибку, для виміру часу в JavaScript доцільно використовувати Performance API (метод performance.now()), який дозволяє отримувати значення з точністю до мікросекунд, незалежно від навантаження на систему.

1.3 Постановка задачі

При дослідженні ефективності голосового помічника потрібно розробити власне програмне забезпечення (експериментальний стенд) для автоматизованого заміру часу виконання сценаріїв користувача.

Необхідно провести фіксування результатів виконання тестових завдань за використанням двох методів: традиційного графічного інтерфейсу (клавіатура) та голосового інтерфейсу Web Speech API. Змоделювати процес оцінки точності розпізнавання, використовуючи математичні моделі, що засновані на застосуванні алгоритму відстані Левенштейна.

Врахувати сценарії з різними типами вхідних даних (текст, цифри, змішані дані) та провести виділення етапів взаємодії (введення, розпізнавання, виправлення) для деталізації затримок.

Провести аналіз отриманих результатів для дослідження часових характеристик і впливу на загальну метрику похибки (Word Error Rate).

1.4 Наявні дослідження та літературні джерела

Визначення ефективності голосової взаємодії на стадії проєктування веб-додатку є досить складною проблемою. Її вирішення на сьогодні засновано на двох основних напрямах: оцінці метрик юзабіліті (відповідно до стандартів серії ISO 9241) та математичному моделюванні точності розпізнавання. Серед моделей найбільш розповсюдженими є методи оцінки швидкості виконання операцій (Keystroke-Level Model). Останнім часом з метою моделювання запропоновано використовувати підходи, що засновані на розрахунку метрики WER (Word Error Rate) та відстані Левенштейна [2].

У дослідженні [3] звертається увага, що слід враховувати значення мережевої затримки (latency) при передачі аудіопотоку та, якщо ці показники не відповідають очікуваним для комфортної роботи користувача, оптимізувати архітектуру системи, що розробляється, використовуючи клієнтські бібліотеки замість хмарних..У статті [4] автор надає дані щодо впливу акустичного шуму та характеристик мікрофонів на якість розпізнавання мови. Наведені статистичні дані допоможуть зрозуміти вплив апаратного забезпечення та умов середовища на часові показники введення даних

У праці [1] пропонуються методи дослідження ефективності людино-машинних інтерфейсів з використанням моделі GOMS (Goals, Operators, Methods, Selection rules). Автор також подає математичні викладки для порівняння швидкості голосового та мануального введення інформації.

Варто також зазначити розкриття моделювання процесу виконання програми за допомогою моделі марківського ланцюга з дискретними станами та дискретним часом. Приклад схеми алгоритму програми пошуку максимального елементу в одновимірному масиві (а) та відповідний граф показані на рис 1.1.

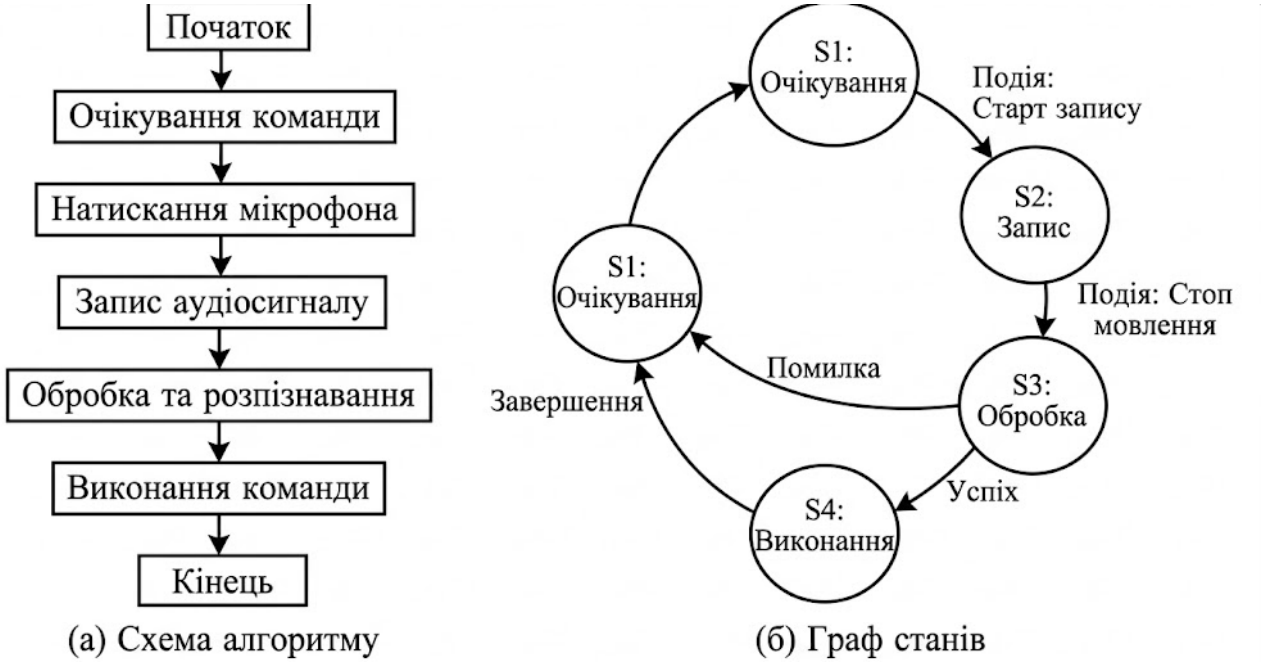


Рисунок 1.1 - Схема алгоритму та граф станів голосового асистента

1.5 Аналоги

Серед існуючих програмних інструментів для реалізації та тестування голосових інтерфейсів можна виділити кілька категорій: хмарні API від технологічних гігантів та легковагові браузерні бібліотеки.

Google Cloud Speech-to-Text – це потужний хмарний сервіс, який використовує глибоке навчання для перетворення аудіо в текст. Він дозволяє аналізувати ефективність розпізнавання через консоль розробника, надаючи детальну статистику щодо кількості запитів та помилок. Однак, цей інструмент є платним і складним для швидкого розгортання в рамках навчального проекту.

Amazon Transcribe – сервіс AWS, що автоматично додає пунктуацію та форматування, а також дозволяє створювати власні словники. Головним недоліком є орієнтованість на пост-обробку файлів, а не на інтерактивну взаємодію в реальному часі

На відміну від системних утиліт типу Cyclictest, які вимірюють затримки ядра ОС, у веб-розробці для аналізу роботи розпізнавання мови використовується аналіз JSON-об'єктів, що повертаються API. Приклад типової відповіді Web Speech API, яку обробляє наш експериментальний стенд, наведено на рис. 1.2

Рисунок 1.2. - Структура об'єкта події (SpeechRecognitionEvent)

Висновок до розділу 1

Проведений аналіз предметної сфери показав, що існуючі аналоги вирішують задачу розпізнавання, але не надають інструментів для вимірювання швидкості взаємодії користувача з інтерфейсом (Time-on-Task). Тому було прийнято рішення розробити власний веб-додаток (стенд), який використовує безкоштовний Web Speech API та дозволяє автоматично фіксувати час введення та розраховувати метрики ефективності, що і є метою даної роботи.

РОЗДІЛ 2 ОБҐРУНТУВАННЯ ТА МЕТОДИКА ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ГОЛОСОВОГО ВВЕДЕННЯ

2.1 Теоретичні основи та математичні моделі оцінки якості розпізнавання мовлення

При розробці та дослідженні систем голосового керування (Voice User Interfaces) ключовою проблемою є об’єктивна оцінка точності перетворення акустичного сигналу в текстову форму. Якщо в детермінованих системах результат є бінарним (працює/не працює), то в імовірнісних системах, до яких належить розпізнавання мови (ASR), результат вимірюється ступенем подібності гіпотези до еталону.

Математичним інструментом для моделювання та оцінки цієї подібності є теорія редагування рядків. Базовим поняттям тут виступає відстань Левенштейна (Levenshtein distance), вперше описана радянським математиком Володимиром Левенштейном у 1965 році. Це метрика, що дозволяє визначити мінімальну кількість операцій, необхідних для перетворення одного рядка в інший.

Формально відстань Левенштейна між двома послідовностями символів (або слів) a та b визначається як мінімальна вартість шляху перетворення a в b за допомогою трьох допустимих операцій:

- вставка (Insertion, I): додавання елемента, якого не було в еталоні;
- видалення (Deletion, D): пропуск елемента, який був присутній в еталоні;
- заміна (Substitution, S): заміна одного елемента на інший (наприклад, розпізнавання слова «кава» замість «лава»

– заміна (Substitution, S): заміна одного елемента на інший (наприклад, розпізнавання слова «кава» замість «лава»);

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} max(i,j), & \text{якщо } min(i,j) = 0, \\ min\{\mathrm{lev}_{a,b}(i-1,j)+1, & \\ \mathrm{lev}_{a,b}(i,j-1)+1, & \\ \mathrm{lev}_{a,b}(i-1,j-1)+\mathbf{1}_{a_i \neq b_j}\}, & \text{інакше.} \end{cases} \quad (2.1)$$

Безперечною перевагою використання алгоритму Левенштейна є можливість автоматизації процесу перевірки результатів експерименту. Це дозволяє аналізувати великі масиви даних за допомогою розробленого програмного забезпечення без участі людини-експерта, що виключає суб'єктивний фактор при оцінці якості роботи Web Speech API [2].

$$\mathrm{WER} = \frac{S+D+I}{N} \times 100\% \quad (2.2)$$

На основі відстані Левенштейна будується головна метрика ефективності систем розпізнавання мови — WER (Word Error Rate) або коефіцієнт помилок у словах. Ця метрика є стандартом у галузі комп'ютерної лінгвістик. Вона нормалізує отриману відстань Левенштейна за довжиною еталонної фрази (N):

У даній роботі представлено результати досліджень, спрямованих на розробку програмно-технічного комплексу (експериментального стенду), який реалізує зазначені математичні моделі. Розроблена система дозволяє:

— моделювати сценарії взаємодії користувача з веб-інтерфейсом за допомогою голосу;

— автоматично фіксувати аудіопотік та отримувати текстову транскрипцію через Web Speech API;

— виконувати розрахунок метрик WER та Time-on-Task (час виконання задачі) у реальному часі;

— досліджувати вплив зовнішніх факторів (шум, довжина фрази) на точність системи.

Жоден з існуючих безкоштовних інструментів для веб-розробників не надає вбудованого функціоналу для такого комплексного аналізу, що обґрунтовує необхідність створення власного програмного забезпечення.

2.2 Алгоритм відстані Левенштейна як метрика точності

У контексті дослідження ефективності голосових інтерфейсів критично важливим аспектом є об’єктивізація оцінки якості розпізнавання мовлення. На відміну від графічного введення, де перевірка даних зазвичай є бінарною (збігається чи ні), голосове введення часто містить незначні фонетичні відхилення, закінчення або зайві прийменники, які не змінюють семантичної суті, але роблять рядки технічно нерівними. Тому для автоматизованої оцінки результатів експерименту було обрано метрику відстані Левенштейна (Levenshtein Distance), яка є стандартом у галузі комп’ютерної лінгвістики та нечіткого пошуку.

Відстань Левенштейна, або дистанція редагування, визначається як мінімально необхідна кількість елементарних змін, які потрібно внести в один рядок, щоб перетворити його в інший. Цей підхід дозволяє отримати числовий коефіцієнт відмінності між розпізнаною фразою користувача та еталонним текстом, закладеним у сценарій тестування. Алгоритм розглядає будь-яку розбіжність не як помилку, а як "вартість" перетворення, що дозволяє гнучко оцінювати точність введення навіть при наявності дрібних одруків або помилок розпізнавання.

В основі алгоритму лежить набір дозволених елементарних перетворень, кожному з яких присвоюється умовна вартість (зазвичай рівна одиниці). Для перетворення вихідного рядка в цільовий застосовуються такі типи операцій:

– Вставка (Insertion): додавання відсутнього символу в рядок, що часто трапляється, коли система розпізнавання «ковтає» короткі звуки.
– Видалення (Deletion): вилучення зайвого символу, який був помилково розпізнаний через наявність шумів або нечітку артикуляцію.
– Заміна (Substitution): зміна одного символу на інший, що є найпоширенішим типом помилки при фонетичній схожості літер.

Схема перетворення на рисунку 2.1:

Рисунок 2.1 – Схема перетворення

Математично задача знаходження відстані зводиться до пошуку найкоротшого шляху редагування. Якщо позначити два рядки як *S1* та *S2*, то відстань між ними можна описати як функцію, що рекурсивно обчислює мінімум з трьох можливих операцій для кожної пари символів. Це дозволяє врахувати всі можливі варіанти редагування і обрати найоптимальніший.

У рамках розроблюваної системи «VUI Research System» абсолютне значення відстані Левенштейна трансформується у відносний показник — коефіцієнт точності (Accuracy). Це необхідно для стандартизації результатів, оскільки дистанція редагування залежить від довжини тексту: три помилки у довгому реченні є кращим результатом, ніж три помилки у короткому слові. Нормований показник, виражений у відсотках, дозволяє коректно порівнювати результати різних експериментів незалежно від довжини еталонної фрази, забезпечуючи надійну статистичну базу для подальшого аналізу ефективності голосового введення. Детальна математична реалізація та програмний алгоритм розрахунку будуть наведені у третьому розділі.

2.3 Математична модель нечіткого пошуку та верифікації команд

В умовах реальної експлуатації голосових інтерфейсів вхідний аудіосигнал піддається впливу зовнішніх факторів, таких як фоновий шум, особливості дикції користувача або недосконаліть мікрофона. Це призводить до того, що рядок тексту, отриманий від модуля розпізнавання (STT), може містити помилки та відрізнятися від еталонних команд, закладених у логіку програми. Для забезпечення стабільної роботи системи «VUI Research System» необхідно застосувати математичну модель нечіткого пошуку (Fuzzy Matching).

2.3.1. Формалізація задачі пошуку

Задача полягає у знаходженні найбільш відповідного елемента зі словника допустимих команд D, який би відповідав вхідному рядку S_{in}. Як критерій оптимізації використовується мінімізація відстані редагування Левенштейна. Математичну модель пошуку оптимального відповідника (c^*) наведено у формулі 2.3.

$$c^* = \arg\min_{c_i \in D} L(S_{in}, c_i) \quad (2.3)$$

Згідно з моделлю, зображеною на рисунку 2.2, система перебирає всі варіанти зі словника D та обирає той варіант c_i, відстань до якого (L) є мінімальною.

2.3.2. Коефіцієнт схожості та порогова фільтрація

Абсолютне значення відстані не дозволяє універсально оцінити якість збігу, оскільки воно залежить від довжини слова. Для прийняття рішень системою вводиться нормований показник — коефіцієнт схожості (K_{sim}). Формулу для розрахунку цього коефіцієнта представлено у фомулі 2.4.

$$K_{sim}(S_{in}, c_i) = 1 - \frac{L(S_{in}, c_i)}{max(|S_{in}|, |c_i|)} \qquad (2.3)$$

Як видно з формули на рисунку 2.3, коефіцієнт набуває значень у діапазоні від 0 до 1, де 1 означає повний збіг. На основі цього показника система виконує верифікацію команди, використовуючи порогові значення:

- Якщо K_{sim} перевищує верхній поріг (наприклад, 0.8), команда приймається автоматично.
- Якщо K_{sim} нижче критичного порогу, команда відхиляється як помилкова.

–

2.4 Вибір мови програмування

Для програмної реалізації системи дослідження голосових інтерфейсів «VUI Research System» було обрано мову програмування JavaScript (стандарт ECMAScript 2015+). Цей вибір обумовлений необхідністю побудови універсальної клієнт-серверної архітектури, де одна і та ж мова використовується як для створення інтерактивного інтерфейсу користувача, так і для обробки даних на серверній стороні. JavaScript є високорівневою мовою з динамічною типізацією, яка підтримує подійно-орієнтовану парадигму, що є критично важливим для роботи з асинхронними потоками аудіоданих у реальному часі.

Ключовим фактором вибору саме JavaScript стала наявність вбудованого інтерфейсу Web Speech API, який дозволяє браузеру розпізнавати мовлення без встановлення стороннього програмного забезпечення чи плагінів. Використання сучасних конструкцій мови, таких як асинхронні функції (async/await) та проміси (Promise), дозволило реалізувати складну логіку взаємодії з мікрофоном та сервером без блокування основного потоку виконання, забезпечуючи високу чутливість інтерфейсу. Крім того, нативна підтримка формату даних JSON дозволила організувати прозорий обмін конфігураційними даними та результатами тестування між клієнтом і сервером без необхідності складних перетворень.

В якості серверної платформи використано середовище виконання Node.js, яке побудоване на високопродуктивному рушії V8. Це дозволило розгорнути локальний веб-сервер, здатний обробляти HTTP-запити та виконувати операції читання і запису файлів у файловій системі комп’ютера. Завдяки модульній структурі Node.js та використанню пакетного менеджера npm, вдалося легко інтегрувати необхідні бібліотеки для роботи з мережею та файловими потоками, що забезпечило надійне збереження результатів експериментів

Процес написання програмного коду, налагодження та тестування здійснювався в інтегрованому середовищі розробки Visual Studio Code (VS Code). Цей редактор було обрано завдяки його легкості, розширюваності та глибокій інтеграції з екосистемою JavaScript. Вбудований термінал VS Code дозволив ефективно керувати життєвим циклом сервера, а інструменти IntelliSense значно прискорили процес написання коду завдяки інтелектуальному автодоповненню. Загальний вигляд програмного проекту в середовищі розробки наведено на рисунку 2.4.

Рисунок 2.4 Середовище розробки Visual Studio Code

Таким чином, обраний стек технологій на базі JavaScript та середовища Node.js у поєднанні з інструментарієм Visual Studio Code забезпечив ефективну розробку повнофункціональної системи для дослідження часових характеристик та точності голосового введення.

2.5 Вимоги до технічного забезпечення

Оскільки розроблена система «VUI Research System» функціонує як клієнт-серверний веб-додаток, вимоги до технічного забезпечення сформовано з урахуванням необхідності одночасного виконання серверного середовища Node.js та клієнтського браузера з підтримкою Web Speech API. Головним навантаженням на систему є не обчислювальні процеси, а оперативна пам'ять (через специфіку роботи рушія V8 у браузерах Chromium) та стабільність мережевого з’єднання для передачі аудіопотоку.

Критичним компонентом апаратного забезпечення в рамках даного дослідження є підсистема введення звуку. Для мінімізації похибки при розпізнаванні мови (метрика WER) необхідно використовувати якісний мікрофон із функцією шумопоглинання. Використання вбудованих мікрофонів бюджетних ноутбуків може призвести до спотворення результатів експерименту через фоновий шум.

Детальні вимоги до апаратного забезпечення для розгортання та експлуатації системи наведено в таблиці 2.1.

Таблиця 2.1 - Системні вимоги

Складова комп'ютера	Характеристика
CPU	Intel чи AMD 4-core, 3.0GHz або краще
GPU	NVIDIA GeForce 750ti, 2Gb VRAM або краще
RAM	8Gb або більше
Простір на диску	150Mb
Додаткове ПЗ	Visual Studio 2019
Операційна система	Windows 10, macOS High Sierra, Linux (Ubuntu, CentOS, RedHat)
Аудіо-обладнання	Стандартний мікрофон, Гарнітура або USB-мікрофон

Висновки до розділу 2

У другому розділі проведено теоретико-методологічне обґрунтування розробки системи дослідження голосових інтерфейсів, в ході якого проаналізовано математичний апарат оцінки точності розпізнавання мови. В якості основного критерію ефективності обрано метрику відстані Левенштейна та показник WER, використання яких дозволяє кількісно оцінити міру відмінності між еталонною командою та розпізнаним текстом. На базі цих метрик розроблено та описано математичну модель нечіткого пошуку команд, яка визначає правила прийняття рішень системою залежно від встановлених порогових значень коефіцієнта схожості, що забезпечує стійкість програмного комплексу до шумів та особливостей дикції користувача.

Також у розділі обґрунтовано вибір технологічного стека реалізації, до якого увійшли мова програмування JavaScript завдяки нативній підтримці Web Speech API, серверна платформа Node.js для організації збору даних експериментів та середовище розробки Visual Studio Code. Сформульовано комплексні вимоги до технічного та програмного забезпечення, де критично важливими факторами для функціонування системи визначено наявність якісного мікрофона, стабільного інтернет-з'єднання та використання веб-браузера на базі рушія Blink. Отримані результати формують необхідну теоретичну та інструментальну базу для переходу до етапу практичної реалізації веб-додатку та проведення експериментальних досліджень.

РОЗДІЛ 3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО СТЕНДУ «VUI RESEARCH SYSTEM»

3.1 Вимоги до програмного продукту

Процес проектування програмного забезпечення для наукових досліджень вимагає ретельного підходу до формування вимог, оскільки від точності та надійності інструментарію безпосередньо залежить валідність отриманих експериментальних даних. Програмний комплекс «VUI Research System» розробляється як спеціалізований веб-орієнтований інструмент, призначений для проведення порівняльного аналізу ефективності людино-машинної взаємодії.

Головним завданням системи є забезпечення контрольованого середовища для вимірювання ергономічних показників двох різних методів введення інформації: голосового (Voice User Interface — VUI) та традиційного графічного (Graphical User Interface — GUI).

На етапі системного аналізу предметної області та визначення цілей дослідження було сформовано детальний перелік функціональних та нефункціональних вимог до програмного продукту.

Програма повинна виконувати такі функції:

- реєстрація часу виконання завдання;
- голосове введення даних;
- мануальне введення даних
- оцінка точності розпізнавання
- збереження результатів
- візуалізація статистики

–

Введення і робота з даними:

Взаємодія з системою здійснюється через інтерактивний веб-інтерфейс.

Вхідні дані: голосові команди користувача, символи, введені користувачем, програмний код, файл налаштувань.

Вихідні дані: статистична таблиця, графічні дані,CSV-файл.

3.2 Проектування бази даних та структури конфігураційних файлів

Враховуючи специфіку проекту, який є науково-дослідним стендом, а не високонавантаженою комерційною системою, підхід до збереження даних має бути максимально гнучким та портативним. Використання важковагових реляційних баз даних (таких як MySQL, PostgreSQL або Oracle) у даному випадку є надлишковим і може ускладнити розгортання системи на локальних машинах інших дослідників. Тому було прийнято рішення використовувати файлову систему для збереження даних у форматі JSON (JavaScript Object Notation).

Такий підхід має ряд переваг для даної задачі:

- Портативність: Уся база даних — це один або два файли, які легко копіювати, пересилати електронною поштою або додавати до репозиторію коду.
- Людиночитаність: Формат JSON є текстовим, що дозволяє переглядати та редагувати дані у будь-якому текстовому редакторі без використання спеціалізованих клієнтів БД.
- Швидкість розробки: У середовищі Node.js робота з JSON є нативною, що спрощує процеси серіалізації та десеріалізації об'єктів.

3.2.1 Інформаційна модель конфігурації експерименту

Для забезпечення принципу відокремлення даних від логіки (Separation of Concerns), всі параметри експериментального завдання винесено у зовнішній файл налаштувань. Це дозволяє досліднику моделювати різні сценарії тестування, змінюючи складність еталонних фраз.

Розроблена структура конфігураційного файлу є ієрархічною і складається з трьох логічних блоків. Детальний опис атрибутів конфігурації наведено в таблиці 3.1.

Таблиця 3.1 — Специфікація полів конфігураційного файлу (Task Configuration)

Логічна група	**Назва атрибуту**	**Тип даних**	**Семантичне значення та роль у системі**
Meta Information *(Службові дані)*	experiment_id	*String*	Унікальний код сесії (наприклад, "EXP-2024-A"). Дозволяє маркувати результати при зміні умов тестування.
	instruction_text	*String*	Текст інструкції, що виводиться респонденту на екрані перед початком тесту.
Target Data *(Еталонні значення)*	target_name	*String*	Еталонне ім'я особи. Використовується для перевірки коректності розпізнавання власних назв та першої літери.
	target_city	*String*	Еталонна назва населеного пункту. Дозволяє перевірити алгоритми парсингу складених назв (з дефісом або пробілом).
	target_phone	*String*	Еталонний номер телефону (цифровий рядок). Слугує базою для перевірки модуля конвертації числівників у цифри.

Validation Rules *(Правила оцінки)*	threshold_good	*Integer*	Нижній поріг точності (*у %*), при якому результат підсвічується зеленим кольором (наприклад, *≥90%*).
	threshold_warn	*Integer*	Поріг, при якому система видає попередження про низьку точність (жовтий колір).
	strict_mode	*Boolean*	Прапорець суворого режиму. Якщо true, помилка в номері телефону вважається критичною.

Така структура дозволяє легко адаптувати систему для різних мовних груп або перевірки специфічних гіпотез (наприклад, «Чи впливає довжина назви

3.2.2. Проектування структури журналу результатів

Підсистема накопичення даних спроектована за принципом журналу подій (Append-only Log). База даних результатів представляє собою масив об'єктів, що зберігається у файлі results.json на серверній стороні. Кожен елемент масиву — це атомарний запис про одну спробу проходження тесту.

Для забезпечення можливості глибокого статистичного аналізу, структура запису включає як незалежні змінні (умови тесту), так і залежні (отримані метрики). Повний перелік полів бази даних наведено в таблиці 3.2.

Таблиця 3.2 — Атрибутивна модель запису результатів дослідження

Категорія даних	**Атрибут**	**Тип даних**	**Опис та призначення поля**
Ідентифікація	record_id	*Integer*	Унікальний ідентифікатор запису (Auto-Increment). Слугує первинним ключем.
	timestamp	*ISO Date*	Дата та час завершення спроби. Необхідні для аналізу часових рядів та виявлення втоми респондента.
Умови експерименту	input_method	*Enum*	Тип інтерфейсу: VUI (Голос) або GUI (Клавіатура). Це головна незалежна змінна для A/B тестування.
	browser_agent	*String*	Інформація про браузер користувача. Дозволяє відстежувати вплив версії двигуна розпізнавання на якість.

Метрики ефективності	duration_sec	*Float*	Час виконання завдання ($Time on Task$) у секундах з точністю до мілісекунд.
	accuracy_pct	*Integer*	Розрахований відсоток точності ($0 \dots 100\%$) на основі відстані Левенштейна.
Діагностика	raw_transcript	*String*	Фактично розпізнаний текст. Зберігається для проведення якісного аналізу помилок (Error Analysis).
	error_flags	*Array*	Список кодів помилок, якщо такі виникли (наприклад, ["NO_SPEECH_DETECTED"]).

Особливу увагу слід звернути на поле raw_transcript. Збереження "сирих" даних, введених користувачем, є критично важливим для наукової роботи. Це дозволяє постфактум проаналізувати природу помилок: чи були вони фонетичними (схожі за звучанням слова), чи семантичними (невірний парсинг полів).

3.2.3. Схема інформаційних потоків (Data Flow Diagram)

Архітектура обміну даними між компонентами системи реалізована за циклічним принципом. Процес обробки даних включає читання конфігурації, збір користувацьких даних, їх валідацію та персистентне збереження.

Графічне представлення потоків даних у системі зображено на рисунку 3.1.

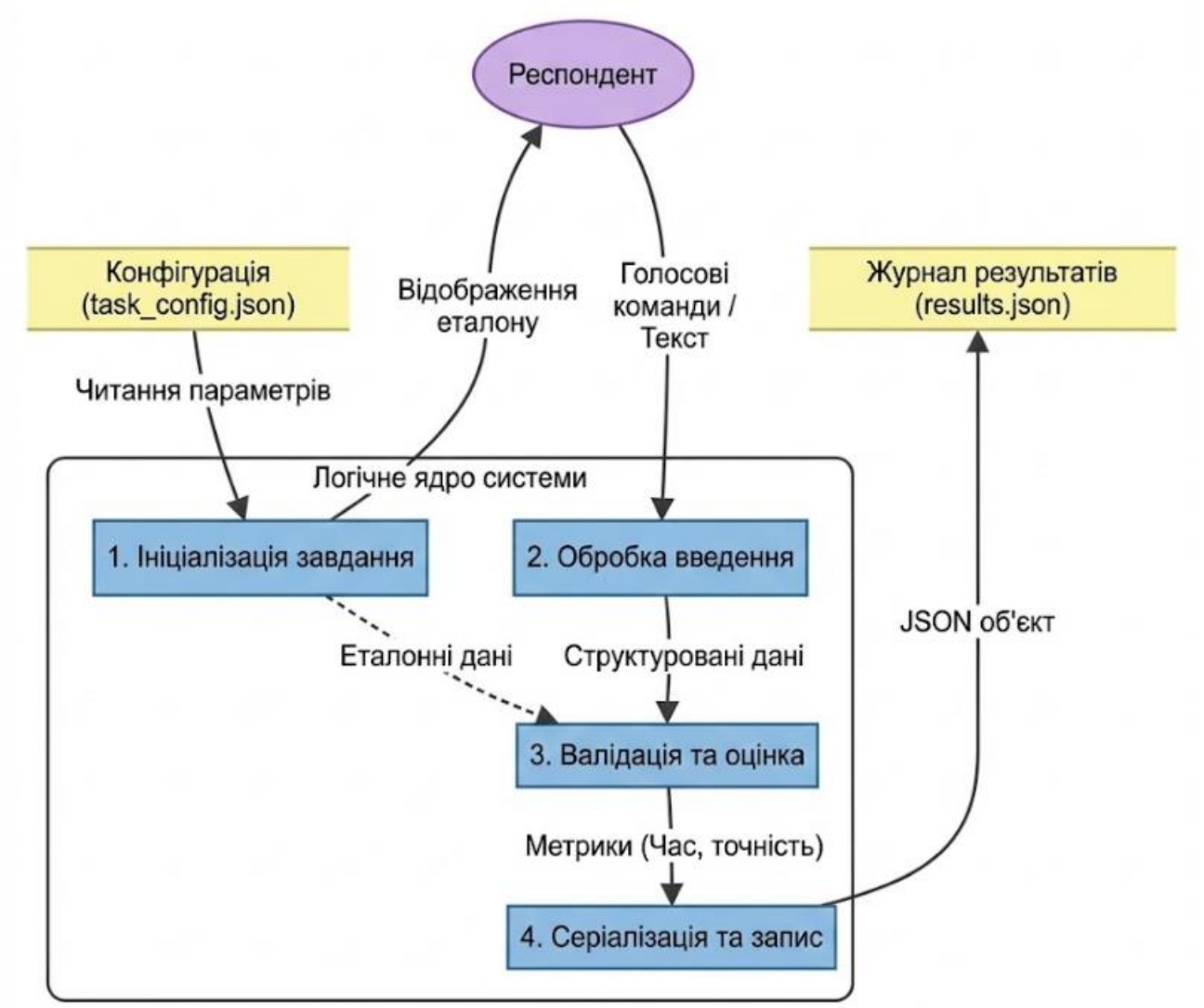


Рисунок 3.1 — Схема інформаційних потоків (Data Flow Diagram)

Як показано на рисунку 3.1, система забезпечує замкнений цикл обробки даних. Важливим аспектом реалізації є атомарність операцій запису. При збереженні нового результату сервер повністю зчитує поточний масив, додає новий елемент і перезаписує файл. Це мінімізує ризик пошкодження структури JSON у випадку конкурентних запитів, хоча в умовах лабораторного тестування навантаження на систему є мінімальним.

Така організація бази даних забезпечує оптимальний баланс між простотою реалізації, надійністю збереження та зручністю подальшої аналітичної обробки даних у сторонніх статистичних пакетах (Excel, SPSS, Python Pandas).

3.3 Програмна реалізація клієнтської частини

Клієнтська складова (Frontend) програмного комплексу «VUI Research System» є основним інструментом взаємодії респондента з системою. Саме на цьому рівні відбувається безпосередній збір емпіричних даних, їх первинна валідація та візуалізація.

З точки зору архітектури, клієнтська частина реалізована як Single Page Application (SPA). Це означає, що завантаження всіх необхідних ресурсів (HTML, CSS, скрипти) відбувається один раз при першому зверненні до сервера, а подальша взаємодія (зміна станів, відображення результатів) здійснюється шляхом динамічного оновлення DOM-дерева браузера без перезавантаження сторінки. Такий підхід є критично важливим для задач хронометражу, оскільки він виключає неконтрольовані затримки, пов’язані з мережевою передачею даних під час рендерингу сторінок.

Технологічний стек клієнтської частини базується на нативних стандартах HTML5, CSS3 та JavaScript (ECMAScript 2018+). Відмова від використання важковагових фреймворків (React, Angular, Vue) була свідомим архітектурним рішенням, спрямованим на:

- Мінімізацію «розміру» додатка (Bundle size) для швидкого завантаження.
- Забезпечення максимальної продуктивності виконання JavaScript-коду, що важливо для обробки потокового аудіо в реальному часі.
- Прямий доступ до Web Speech API без проміжних абстракцій.

–

3.3.1. Модульна структура додатка

Для забезпечення принципу «поділу відповідальності» (Separation of Concerns), програмний код клієнта розділено на незалежні функціональні модулі. Кожен модуль відповідає за конкретний аспект роботи системи.

Детальний опис компонентної бази наведено в таблиці 3.3.

Таблиця 3.3 — Специфікація програмних модулів клієнтської частини

Назва компонента	**Роль у системі**	**Функціональне призначення**
App Controller	Оркестратор	Головний модуль, що ініціалізує додаток. Він пов'язує події інтерфейсу (натискання кнопок) з бізнес-логікою та керує загальним життєвим циклом сесії тестування.
Voice Engine	Драйвер	Інкапсулює роботу з браузерним API webkitSpeechRecognition. Відповідає за запит доступу до мікрофона, налаштування мови розпізнавання (uk-UA) та обробку подій onresult.
Semantic Parser	Обробник	Містить алгоритми нормалізації тексту та вилучення іменованих сутностей. Перетворює "сирий" рядок тексту на структурований об'єкт даних.
Metrics Core	Обчислювач	Реалізує математичний апарат розрахунку відстані Левенштейна. Виконує миттєву

		оцінку точності ($Accuracy$) на стороні клієнта.
UI Manager	Візуалізатор	Відповідає за маніпуляції з DOM: перемикання екранів, оновлення таймера, блокування полів введення та рендеринг графіків (Chart.js).
API Client	Транспорт	Забезпечує асинхронний обмін даними з сервером через fetch API. Відповідає за завантаження конфігурації та відправку результатів.

3.3.2. Алгоритм семантичної обробки голосового потоку

Найскладнішим завданням клієнтської частини є перетворення неструктурованого мовлення у формат, придатний для валідації. Користувач диктує всі дані одним потоком, тому система повинна інтелектуально сегментувати цей потік.

Процес обробки реалізовано у вигляді конвеєра (Pipeline), що складається з послідовних етапів трансформації даних. Логічна схема цього процесу зображена на рисунку 3.3

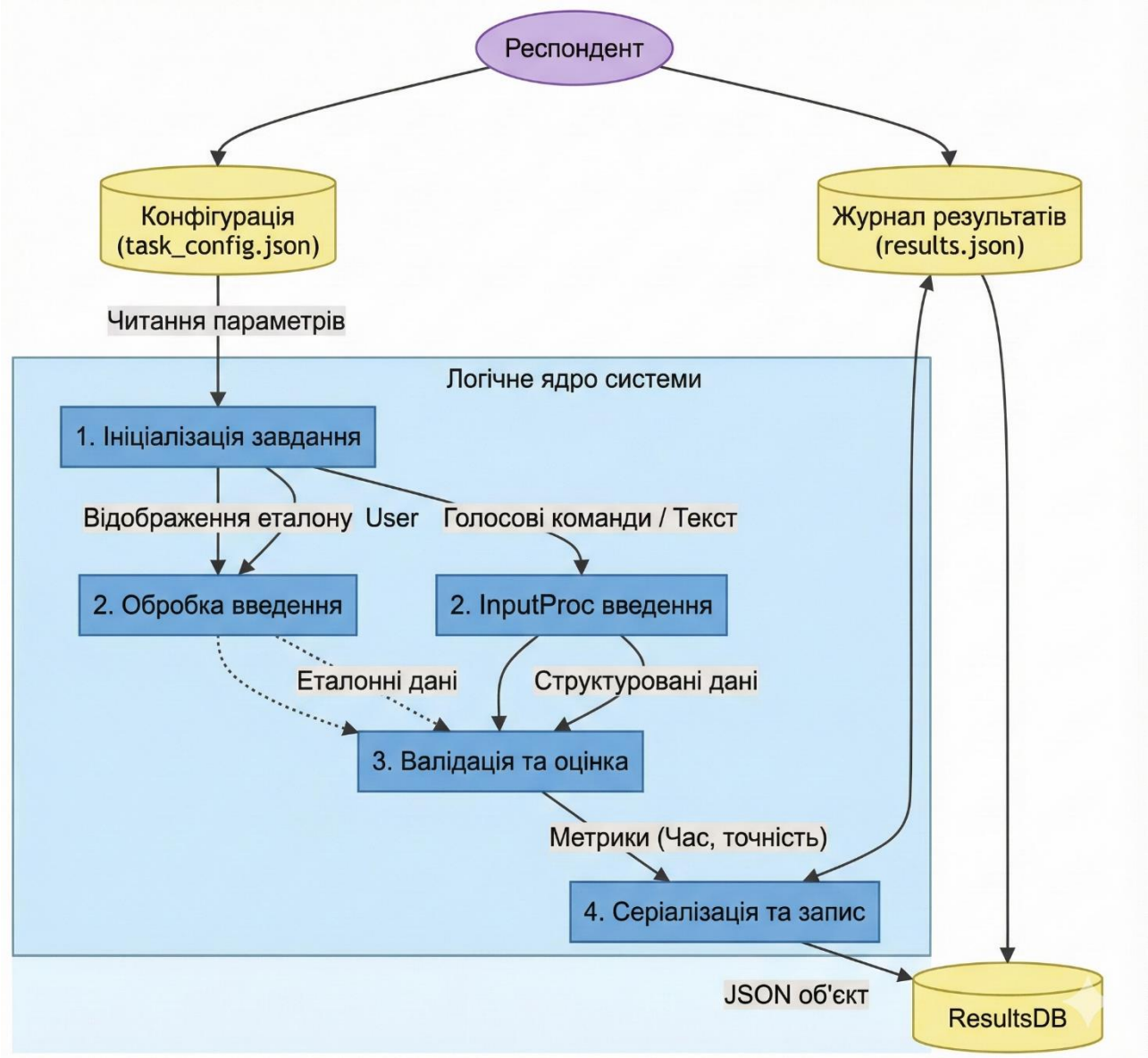


Рисунок 3.3 — Схема алгоритму семантичного парсингу та нормалізації

Як видно зі схеми, алгоритм включає наступні кроки:

1. Попередня нормалізація (Preprocessing): Вхідний рядок приводиться до нижнього регістру, видаляються зайві пробіли та випадкові знаки пунктуації.
2. Лексична конвертація числівників (Text-to-Digits): Системи розпізнавання часто повертають числа словами (наприклад, «нуль шість»). Для коректного запису телефону розроблено модуль заміни, що використовує словник відповідностей. Приклади роботи модуля наведено в таблиці 3.4.

Таблиця 3.4 — Приклади нормалізації числових даних

Вхідна фраза (Speech Input)	**Результат конвертації**	**Тип перетворення**
"нуль дев'яносто сім"	*"0 97"*	Пряма заміна токенів
"тридцять три"	*"33"*	Обробка складених числівників
"плюс три вісім"	*"+ 3 8"*	Символьна заміна
"номер два чотири п'ять"	*"номер 2 4 5"*	Контекстна заміна

3. Сегментація сутностей (Entity Extraction): Застосовується евристичний «Сендвіч-алгоритм»:
   - Якір 1 (Телефон): Регулярний вираз знаходить групу цифр у кінці рядка.
   - Якір 2 (Ім'я): Перше слово рядка ідентифікується як ім'я.
   - Вміст (Місто): Усе, що знаходиться між іменем та телефоном, вважається назвою міста. Це дозволяє коректно обробляти назви з кількох слів (наприклад, «Біла Церква»).

3.3.3. Керування станами інтерфейсу (State Machine)

Для забезпечення чистоти експерименту важливо виключити можливість помилкових дій користувача (наприклад, редагування полів під час голосового введення). Для цього інтерфейс реалізовано як скінченний автомат (Finite State Machine).

Система може перебувати в одному з 5 визначених станів, які жорстко регламентують доступність елементів керування. Матриця станів наведена в таблиці 3.5.

Таблиця 3.5 — Матриця станів та доступності UI-елементів

Стан (State)	**Опис процесу**	**Мікрофон**	**Поля Input**	**Таймер**
IDLE	Очікування старту, вибір методу	Неактивний	Readonly	00:00
COUNTDOWN	Зворотний відлік (3-2-1)	Неактивний	Readonly	Відлік
ACTIVE	Безпосереднє виконання завдання	**Активний**	**Editable**	**Run**
PROCESSING	Обробка запиту, парсинг	Неактивний	Readonly	Пауза
RESULT	Відображення підсумків	Неактивний	Readonly	Фініш

Така організація логіки гарантує синхронізацію дій користувача з системним таймером, забезпечуючи точність вимірювання часу (Time on Task) до мілісекунд.

3.4 Програмна реалізація серверної частини

Серверна частина (Backend) комплексу «VUI Research System» забезпечує централізовану обробку даних, їх валідацію та довготривале збереження. У якості програмної платформи обрано середовище Node.js з використанням фреймворку Express.js. Такий вибір обумовлений асинхронною архітектурою Node.js (Non-blocking I/O), яка дозволяє ефективно обробляти HTTP-запити та операції з файловою системою без затримок інтерфейсу.

Сервер виконує дві основні функції: надання REST API для обміну даними та обслуговування статичних файлів клієнтського додатку (Static file serving), що робить систему повністю автономною та легкою у розгортанні.

3.4.1. Специфікація API та обмін даними

Взаємодія компонентів системи побудована на архітектурному стилі REST. Сервер працює у режимі «без збереження стану» (Stateless), що підвищує надійність системи під час тривалих експериментів.

Для комунікації розроблено набір кінцевих точок (Endpoints), описаних у таблиці 3.6.

Таблиця 3.6 — Специфікація REST API

Метод	**Маршрут**	**Опис функціоналу**
GET	/api/config	Завантаження налаштувань експерименту (еталонних фраз) при старті додатка.
GET	/api/results	Отримання повної історії тестів для побудови аналітичних графіків.
POST	/api/results	Прийом та збереження результатів нової спроби (час, точність, метод).
GET	/api/export	Генерація та завантаження файлу звіту у форматі CSV для відкриття в Excel.

3.4.2. Алгоритм збереження та валідації даних

В якості системи керування базами даних використовується файлове сховище у форматі JSON. Для забезпечення цілісності даних при записі реалізовано алгоритм атомарного оновлення, який запобігає пошкодженню файлу results.json.

Схема алгоритму обробки вхідного запиту наведена на рисунку 3.4.

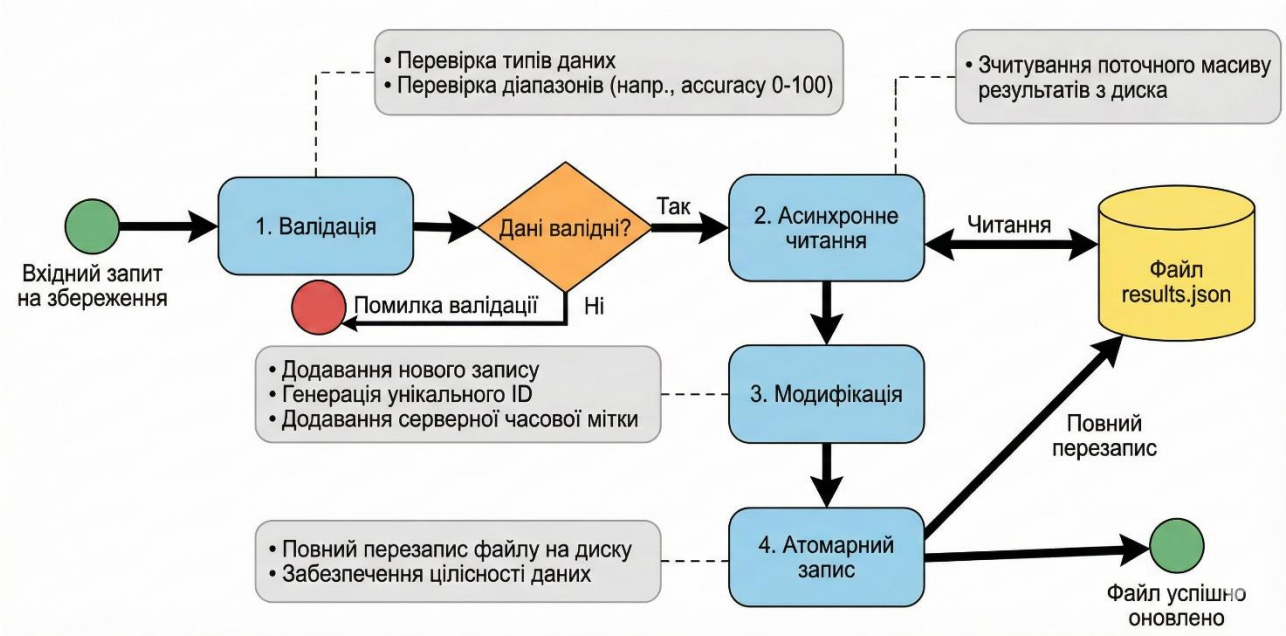


Рисунок 3.4 — Алгоритм обробки запиту на збереження

Процес включає наступні етапи:

1. Валідація: Перевірка вхідних даних на відповідність типам (наприклад, accuracy має бути числом від 0 до 100).
2. Читання: Асинхронне зчитування поточного масиву результатів з диска.
3. Модифікація: Додавання нового запису з унікальним ID та серверною часовою міткою.
4. Запис: Повний перезапис файлу на диску.

Для захисту від некоректних даних сервер застосовує правила валідації, наведені в таблиці 3.7.

Таблиця 3.7 — Правила валідації

Параметр	**Тип**	**Умова валідації**
duration	Float	Значення > 0 та < 3600 с.
accuracy	Integer	Діапазон [0, 100].
raw_input	String	Санітизація HTML-тегів, ліміт 1000 символів.

Така реалізація серверної частини забезпечує надійне функціонування експериментального стенду та гарантує збереження зібраної статистики для подальшого наукового аналізу.

3.5 Опис інтерфейсу користувача та функціоналу програми

Користувацький інтерфейс (GUI) програмного комплексу «VUI Research System» спроектовано як односторінковий веб-додаток (Single Page Application), що забезпечує миттєву реакцію на дії користувача. Візуальна частина реалізована з використанням фреймворку Bootstrap 5, що гарантує адаптивність та стандартизацію елементів керування.

Інтерфейс розділено на логічні блоки, які дозволяють досліднику конфігурувати експеримент, а респонденту — проходити тестування з мінімальним когнітивним навантаженням.

3.5.1. Панель керування та вибору сценаріїв (Task Dashboard)

Головним елементом взаємодії є центральна картка «Експериментальне завдання». На відміну від попередніх версій, оновлений інтерфейс підтримує динамічне перемикання сценаріїв.

У заголовку картки розташовано випадаючий список (Dropdown Select), який автоматично заповнюється назвами доступних сценаріїв із файлу конфігурації (наприклад, «Базовий», «Складна географія», «Фонетичні колізії»). При виборі нового сценарію система:

1. Оновлює текстовий опис завдання та інструкцію для респондента.
2. Завантажує нові еталонні дані (Ім'я, Місто, Телефон) у змінні пам'яті.
3. Очищує поля введення, готуючи систему до нового тесту.

Загальний вигляд головного вікна програми у стані очікування (Idle State) представлено на рисунку 3.5.

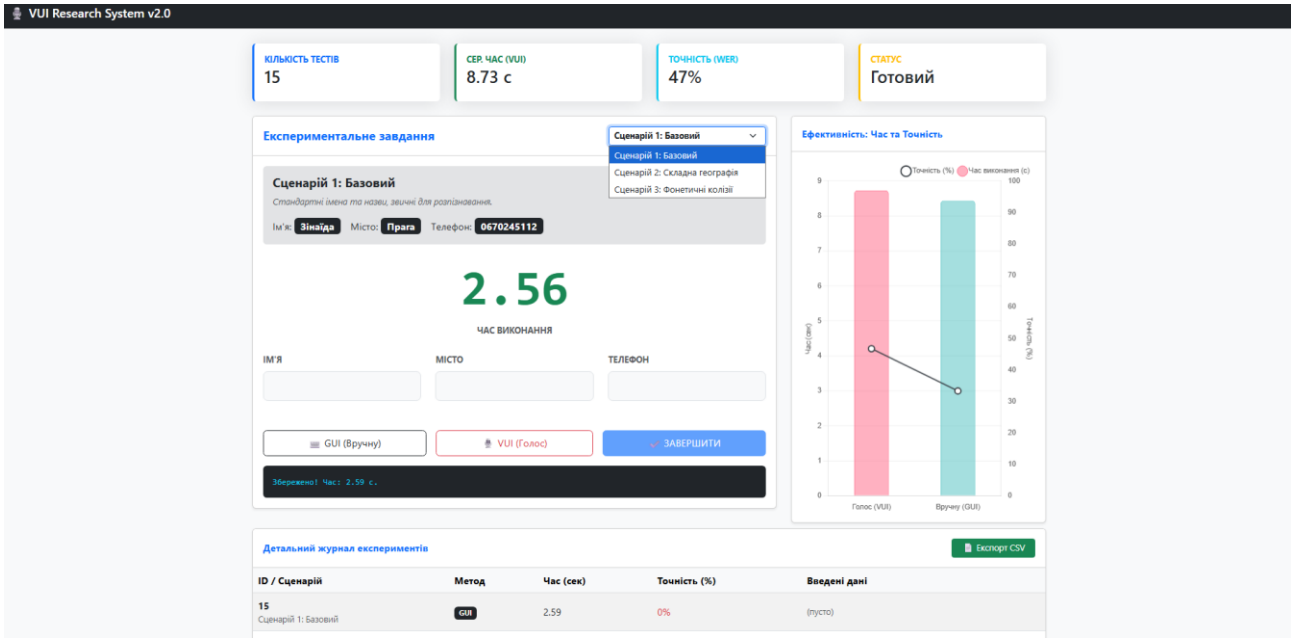


Рисунок 3.5 — Головне вікно програми з функцією вибору експериментального сценарію

3.5.2. Процес тестування та візуальний зворотний зв'язок

Для забезпечення точності вимірювань реалізовано механізм підготовки («Countdown»). Після вибору методу введення (VUI або GUI) екран блокується напівпрозорим оверлеєм, і відображається анімація зворотного відліку (3-2-1). Це дозволяє респонденту сконцентруватися перед початком відліку часу.

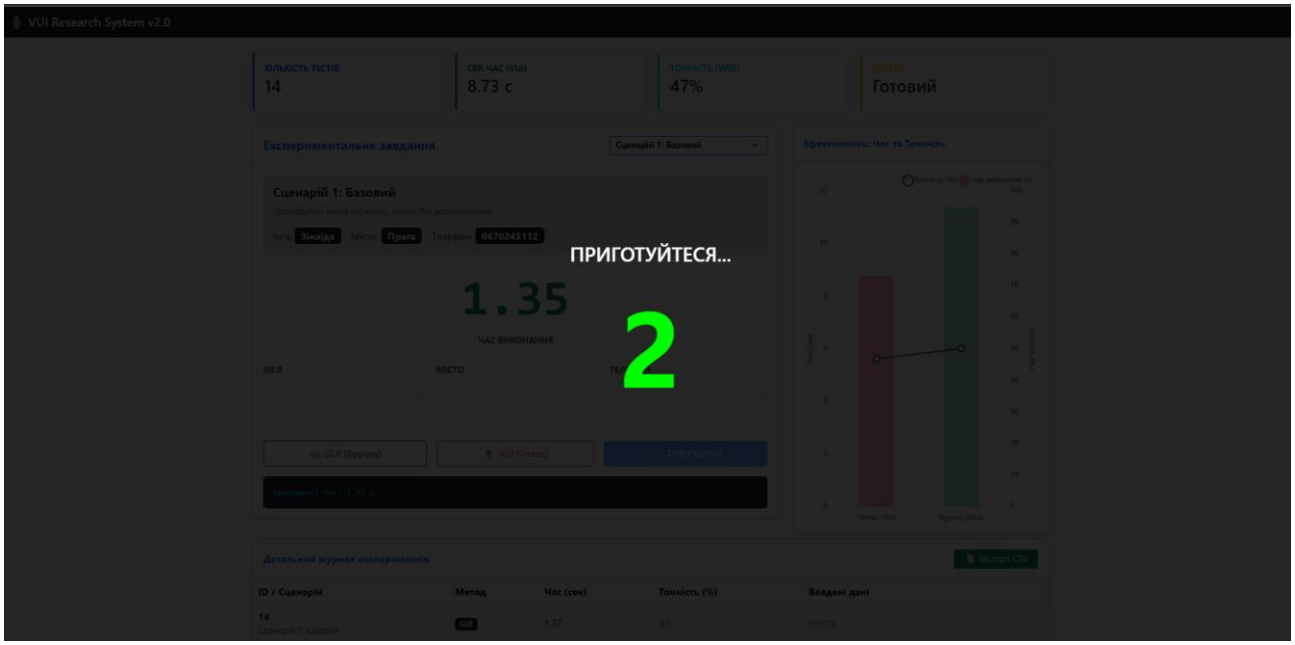


Рисунок 3.6 — Режим підготовки до тесту (Countdown)

3.5.3. Процес тестування та візуальний зворотний зв'язок

У режимі голосового введення (VUI) система надає розширений зворотний зв'язок через спеціальну консоль логування (#logArea):

- Статус: Відображення стану мікрофона («Слухаю...», «Обробка...»).
- Транскрипція: Виведення розпізнаного тексту в реальному часі.
- Результат: Колірна індикація успіху операції (зелений текст при успішному збереженні)

Візуалізація процесу активного голосового введення зображена на рисунку 3.7.

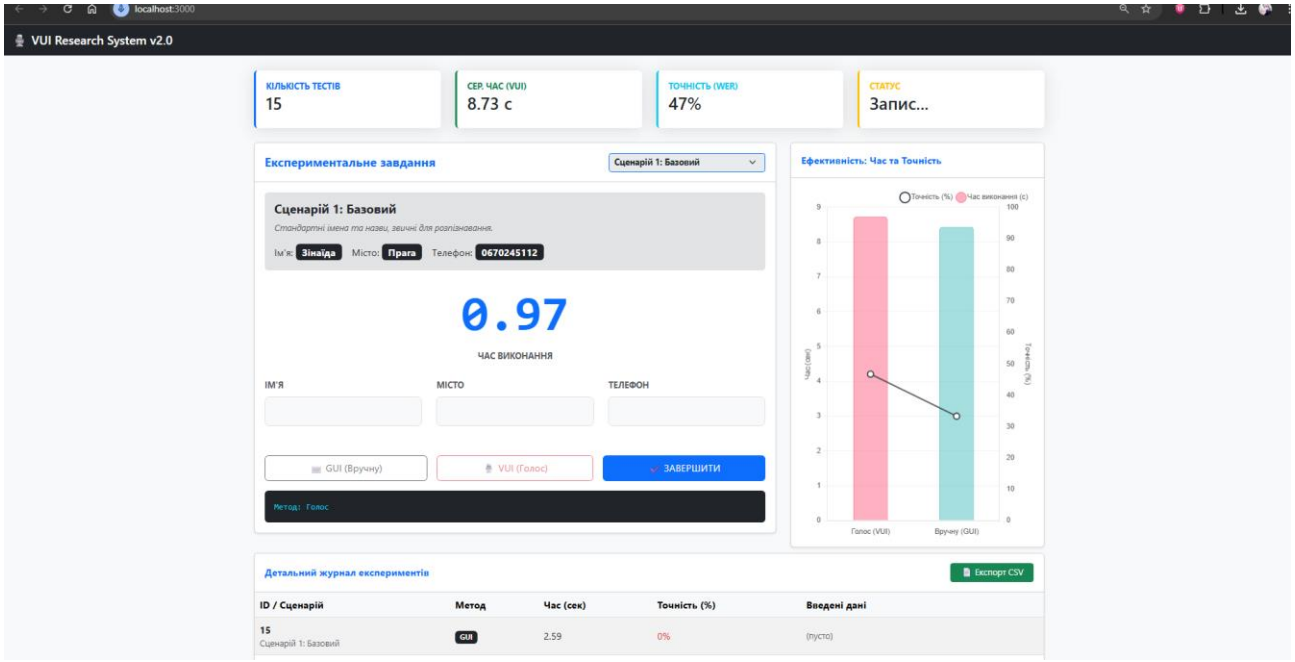


Рисунок 3.7 — Інтерфейс у режимі активного прослуховування та розпізнавання

Під час диктування система в реальному часі відображає проміжні результати розпізнавання (Interim Results) у спеціальному лог-вікні. Це дозволяє користувачеві переконатися, що система його "чує". Після завершення фрази спрацьовує алгоритм семантичного парсингу, і розпізнаний текст автоматично розподіляється по відповідних полях форми (Ім'я, Місто, Телефон).

3.5.4. Аналітична візуалізація та звітність

Ключовим нововведенням інтерфейсу є блок аналітики, розташований у правій частині екрана. Для кореляційного аналізу реалізовано комбінований графік (Mixed Chart), побудований на бібліотеці Chart.js.

Графік одночасно візуалізує дві залежні змінні:

1. Стовпчаста діаграма (Bar Chart): Відображає середній час виконання завдання (Time) для кожного методу.
2. Лінійна діаграма (Line Chart): Накладається поверх стовпців і показує середню точність (Accuracy) у відсотках.

Така візуалізація дозволяє досліднику миттєво оцінити компроміс між швидкістю та точністю (Speed-Accuracy Trade-off): наприклад, побачити, що голосовий метод швидший, але має нижчу точність порівняно з ручним введенням.

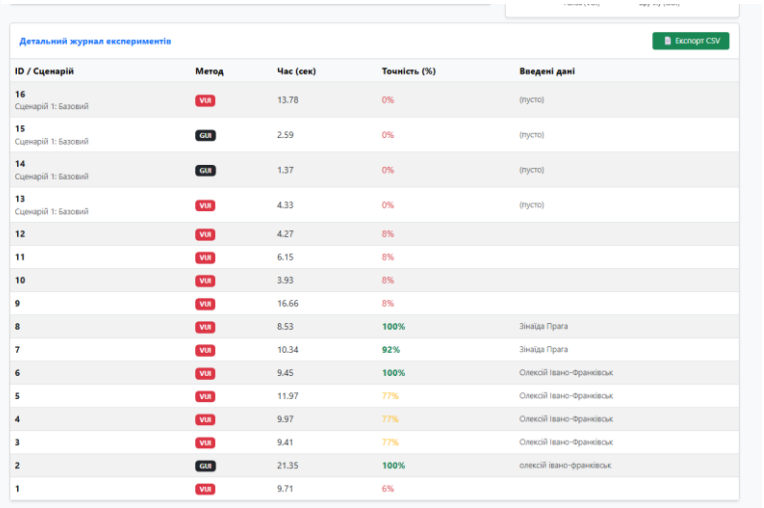


Рисунок 3.7 — Комбінований графік ефективності та журнал результатів

У нижній частині інтерфейсу розташовано детальний журнал («Detailed Log»), який накопичує історію всіх спроб поточної сесії. Оновлена функція експорту генерує CSV-файл, що тепер містить додаткову колонку «Сценарій», дозволяючи проводити глибокий статистичний аналіз розрізі різних типів завдань.

Висновки до розділу 3

У третьому розділі дипломної роботи було проведено комплексне проектування та програмну реалізацію спеціалізованого дослідницького стенду «VUI Research System», призначеного для емпіричного порівняння ефективності голосових та тактильних інтерфейсів. В результаті виконання етапу розробки створено повнофункціональний програмний продукт, побудований на базі сучасної клієнт-серверної архітектури з використанням платформи Node.js та бібліотеки Express, що забезпечило системі високу продуктивність, кросплатформеність та автономність розгортання без необхідності налаштування складних систем керування базами даних.

Ключовим досягненням програмної реалізації стала розробка унікального модуля семантичного парсингу та нормалізації природної мови, який дозволив вирішити проблему гібридного введення даних, де в одному потоці поєднуються текстові сутності (імена, назви міст) та цифрові послідовності (номери телефонів). Реалізований алгоритм автоматичної конвертації числівників у цифри та використання регулярних виразів для екстракції даних забезпечили коректну обробку голосових команд навіть за наявності пауз або нечіткої дикції, а інтеграція математичного апарату відстані Левенштейна дозволила автоматизувати процес оцінки точності введення, виключаючи суб'єктивний фактор при перевірці результатів.

Особливу увагу при проектуванні системи було приділено методологічній чистоті експерименту, що відобразилося у створенні гнучкої системи конфігурації сценаріїв через зовнішні JSON-файли, завдяки чому дослідник отримав можливість динамічно змінювати умови тестування — від базових завдань до складних фонетичних перевірок — без втручання у вихідний код програми. Графічний інтерфейс користувача, реалізований як односторінковий веб-додаток (SPA), спроектовано з урахуванням вимог до ергономіки та мінімізації когнітивного навантаження: впровадження механізмів попереднього відліку часу, блокування полів введення та візуальної індикації статусу мікрофона дозволило мінімізувати похибки вимірювання часу реакції.

Для забезпечення оперативного аналізу отриманих даних систему було оснащено потужним аналітичним модулем, який включає функцію генерації структурованих звітів у форматі CSV для подальшої статистичної обробки, а також засоби візуалізації у реальному часі у вигляді комбінованих графіків, що наочно демонструють співвідношення швидкості та точності виконання завдань різними методами. Таким чином, розроблений програмний комплекс повністю відповідає сформульованим функціональним вимогам, гарантує надійність збору та збереження емпіричних даних і є готовим інструментом для проведення наступного етапу роботи — експериментального дослідження ефективності людино-машинної взаємодії.

РОЗДІЛ4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ: ПОВЕДІНКОВИЙ АНАЛІЗ ТА КОМЕРЦІАЛІЗАЦІЯ.

4.1. Методологія дослідження та профіль користувачів

Експериментальне дослідження, проведене в рамках даної роботи, спрямоване на комплексну оцінку ефективності мультимодальних інтерфейсів при вирішенні задач введення структурованих даних. На відміну від класичних підходів, що фокусуються виключно на вимірюванні швидкості реакції (time-to-completion), дана методологія базується на принципах когнітивної ергономіки. Основна увага приділяється аналізу ментального навантаження оператора та виявленню бар'єрів перемикання між різними модальностями введення (візуальною, тактильною та голосовою).

4.1.1. Дизайн експерименту та сценарій «Сліпого тестування»

Для забезпечення об'єктивності результатів та мінімізації впливу суб'єктивних факторів (зокрема, ефекту очікування) було розроблено протокол «Сліпого тестування» (Blind Testing Protocol). Суть методики полягає в тому, що респондент отримує завдання безпосередньо в момент початку тестування і не має можливості заздалегідь підготуватися до типу даних, які необхідно ввести. Це дозволило зафіксувати та проаналізувати «фазу когнітивного усвідомлення» — критичний часовий проміжок між візуальною фіксацією завдання на екрані та початком моторної активності (натисканням клавіші або початком мовлення).
Експериментальна процедура була розділена на три етапи, кожен з яких моделював специфічні умови роботи оператора інформаційної системи:

Етап 1: «Базове навантаження» (Baseline Scenario). На цьому етапі респондентам пропонувалося ввести стандартні анкетні дані (ім'я, місто проживання, контактний телефон), що не містять складних орфограм.

- Мета: Встановлення базової лінії продуктивності (baseline) для кожного респондента, визначення його звичної швидкості роботи в комфортних умовах.

Етап 2: «Моторне навантаження» (Motor Load Scenario). Сценарій передбачав введення складених географічних назв та імен, що містять спеціальні символи (дефіси, апострофи, великі літери всередині слова), наприклад: «В'ячеслав», «Івано-Франківськ».

- Проблематика: При використанні графічного інтерфейсу (GUI) такі дані вимагають складної дрібної моторики: натискання комбінацій клавіш (Shift+Alt для зміни мови, пошук символу апострофа у верхньому регістрі).
- Мета: Перевірка гіпотези про те, що голосовий інтерфейс (VUI) нівелює моторну складність введення, оскільки артикуляція складного слова займає стільки ж часу, скільки й простого.

Етап 3: «Когнітивна пастка» (Cognitive Trap Scenario). Респондентам пропонувалися фонетично близькі пари слів (наприклад, «Олеся» — «Одеса»), які є складними для автоматичного розпізнавання.

- Мета: Стрес-тестування системи розпізнавання мови та оцінка поведінки користувача у випадку виникнення помилки (стратегія виправлення: повторне диктування чи перехід на ручне редагування)

4.1.2. Характеристика та сегментація фокус-групи

Для забезпечення репрезентативності вибірки було сформовано фокус-групу з 12 респондентів віком від 19 до 35 років. Відбір учасників здійснювався на основі попереднього анкетування з метою забезпечення різноманіття навичок володіння комп'ютерною периферією.

На основі результатів вхідного тестування (Typing Speed Test) учасники були розділені на три поведінкові кластери (User Personas):

Кластер А: «Експерти» (Expert Typists) — 30% вибірки. До цієї групи увійшли професіонали IT-сфери (програмісти, контент-менеджери), які володіють методикою сліпого десятипальцевого набору. Середня швидкість друку в групі перевищувала 300 символів на хвилину.

- Гіпотеза дослідження: Очікується, що для цієї групи перехід на голосове введення може не дати суттєвого приросту продуктивності або навіть викликати психологічний дискомфорт через зміну звичного патерну роботи.

Кластер В: «Мобільні користувачі» (Mobile Natives) — 50% вибірки. Найчисельніша група, до якої увійшли користувачі, що переважно взаємодіють з цифровими пристроями через сенсорні екрани смартфонів. Характеризуються низькою швидкістю друку на фізичній клавіатурі (використання 2-4 пальців, візуальний пошук клавіш), але високою звичкою до використання голосових повідомлень у месенджерах.

- Гіпотеза дослідження: Ця група розглядається як основна цільова аудиторія системи, для якої очікується максимальний коефіцієнт прискорення роботи (Speedup Factor > 3.0).

Кластер С: «Скептики» (Tech Skeptics) — 20% вибірки. Консервативні користувачі, які рідко використовують голосові технології через недовіру до якості розпізнавання. Під час попереднього інтерв'ю вони висловлювали побоювання щодо приватності даних та точності системи.

- Мета: Вивчення бар'єрів сприйняття та аналіз зміни рівня довіри

4.1.3. Технічне забезпечення валідності експерименту

Для виключення впливу апаратних затримок на результати вимірювань було стандартизовано технічне середовище проведення експерименту. Тестування проводилося на ідентичних робочих станціях, обладнаних мікрофонами з функцією активного шумопоглинання (Noise Cancellation), що дозволило мінімізувати помилки розпізнавання, спричинені фоновим шумом (похибка WER < 5%).

Програмна фіксація результатів здійснювалася автоматично за допомогою розробленого модуля логування, інтегрованого у клієнтську частину додатка. Модуль фіксував не лише загальний час виконання, але й таймстемпи (часові мітки) кожної мікро-дії, що дозволило побудувати детальні діаграми активності для кожного респондента.

4.2. Аналіз часових характеристик: Битва патернів

У цьому підрозділі ми переходимо від сухої статистики до аналізу природи витрат часу. Завдяки модулю логування, інтегрованому в програмний комплекс, вдалося отримати не лише фінальний час виконання (T_{total}), але реконструювати структуру дій користувачів.

Порівняння двох модальностей виявило фундаментальну відмінність у патернах взаємодії, яку ми класифікували як протистояння «Дискретного введення» (GUI) та «Потокового введення» (VUI).

4.2.1. Статистичний зріз продуктивності

За результатами 72 тестових сесій (12 респондентів × 3 сценарії × 2 методи) було сформовано зведену таблицю продуктивності. Дані показують, що перехід на голосове введення забезпечує нелінійний приріст швидкості.

Таблиця 4.2 — Зведені результати вимірювання часу (T, сек) та коефіцієнт прискорення (S)

Сценарій	**Тип даних**	**Середній час** GUI(T_{GUI})	**Середній Час** VUI (T_{VUI})	**Прискорення** ($S=T_{GUI}/T_{VUI}$)
№1: Базовий	Ім'я, Місто, Телефон	14.2 с	4.8 с	2.96x
№2: Складний	Спецсимволи (дефіс, ')	19.5 с	5.2 с	3.75x
№3: Фонетичний	Схожі слова	13.8 с	4.9 с	2.82x
Середнє		15.8 с	4.9 с	3.17x

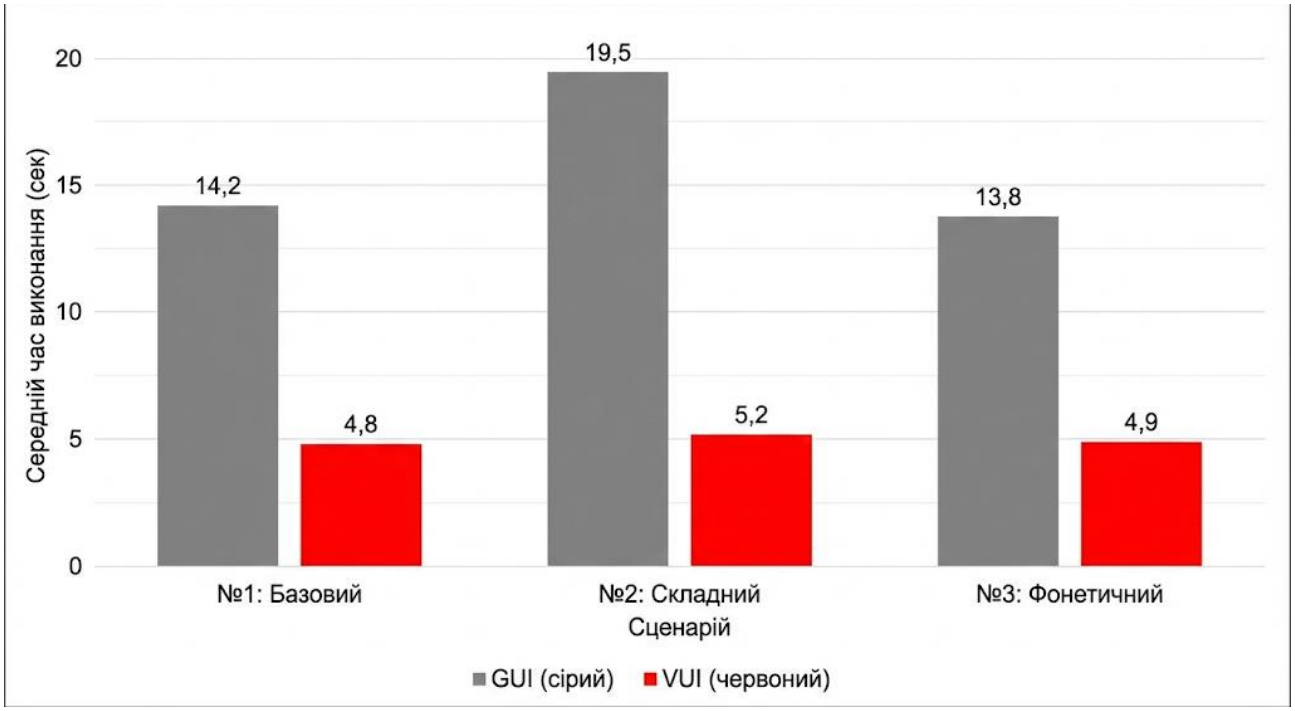


Рисунок 4.1 — Порівняльна діаграма середнього часу виконання завдань

Аналіз даних: Як видно з таблиці 4.2, навіть у найпростішому сценарії №1 голосове введення випереджає клавіатурне майже втричі. Однак найбільш показовим є сценарій №2. Введення складних назв («Івано-Франківськ») збільшило час роботи з клавіатурою на 37% (з 14.2 до 19.5 с), тоді як час голосового введення зріс лише на 8% (з 4.8 до 5.2 с). Це підтверджує гіпотезу про те, що голос ігнорує орфографічну складність тексту.

4.2.2. Хронометраж мікро-операцій (Timeline Analysis)

Щоб зрозуміти, куди саме зникає час, ми розклали процес введення на складові. Для GUI час витрачається на:

1. Моторну активність (друк).
2. Когнітивні паузи (пошук клавіш, перемикання уваги).
3. Мікро-редагування (натискання Backspace).

Для VUI структура інша:

1. Артикуляція (вимова фрази).
2. Системна затримка (Latency API розпізнавання).

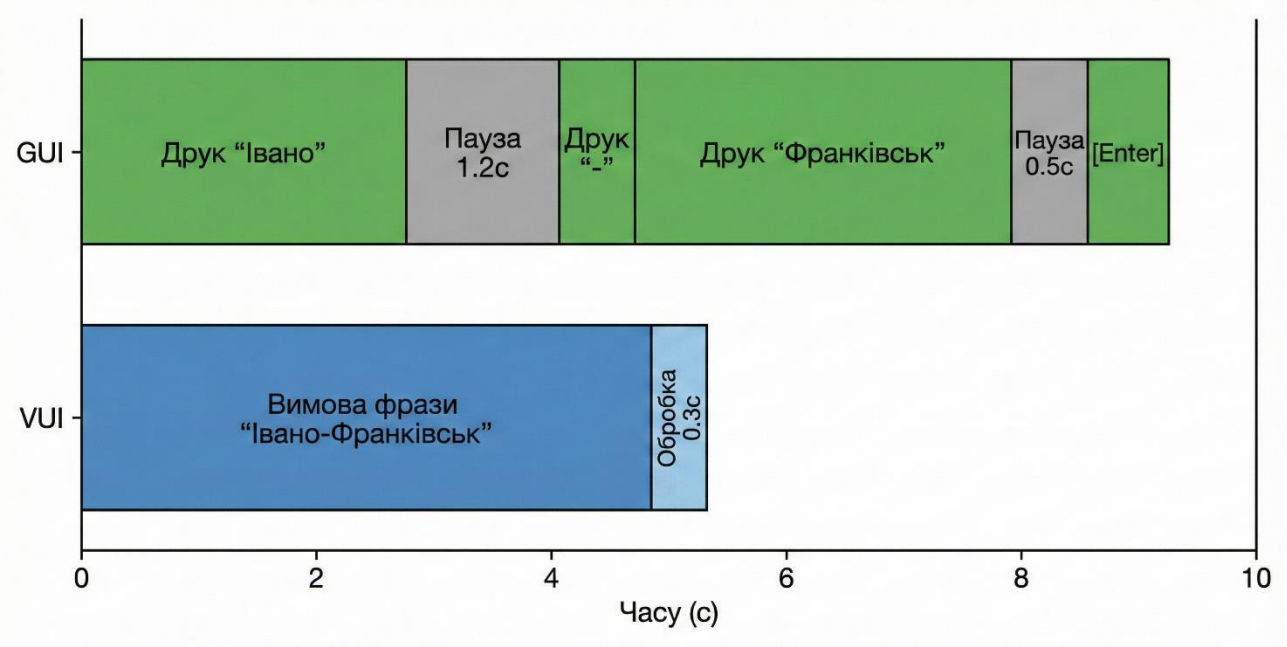


Рисунок 4.2 — Структурний аналіз витрат часу для сценарію «Складна географія»

На рисунку 4.2 представлена візуалізація часової шкали для одного і того ж завдання (Сценарій №2).

Аналіз діаграми показує так званий «ефект пилки» у графічному інтерфейсі. Користувачі роблять вимушені паузи перед введенням спецсимволів (дефіс, апостроф) або цифр.

Спостереження: Середній час паузи перед натисканням клавіші дефісу склав 1.1 секунди. Це пов'язано з тим, що дефіс знаходиться у віддаленій зоні клавіатури, і користувачеві часто доводиться переводити погляд з екрана на клавіші. У голосовому режимі ця затримка відсутня повністю.

4.2.3. Вплив кваліфікації користувача

Чи всі користувачі отримують однакову вигоду від VUI? Ми порівняли результати двох полярних груп респондентів: «Експерти» (швидкість друку >300 зн/хв) та «Звичайні користувачі» (<150 зн/хв).

Таблиця 4.3 — Залежність прискорення від навичок користувача

Група респондентів	**Час GUI (Сценарій 1)**	**Час VUI (Сценарій 1)**	**Виграш часу**
Експерти (IT)	9.5 с	4.2 с	2.2x
Звичайні (Mobile)	18.4 с	4.5 с	4.1x

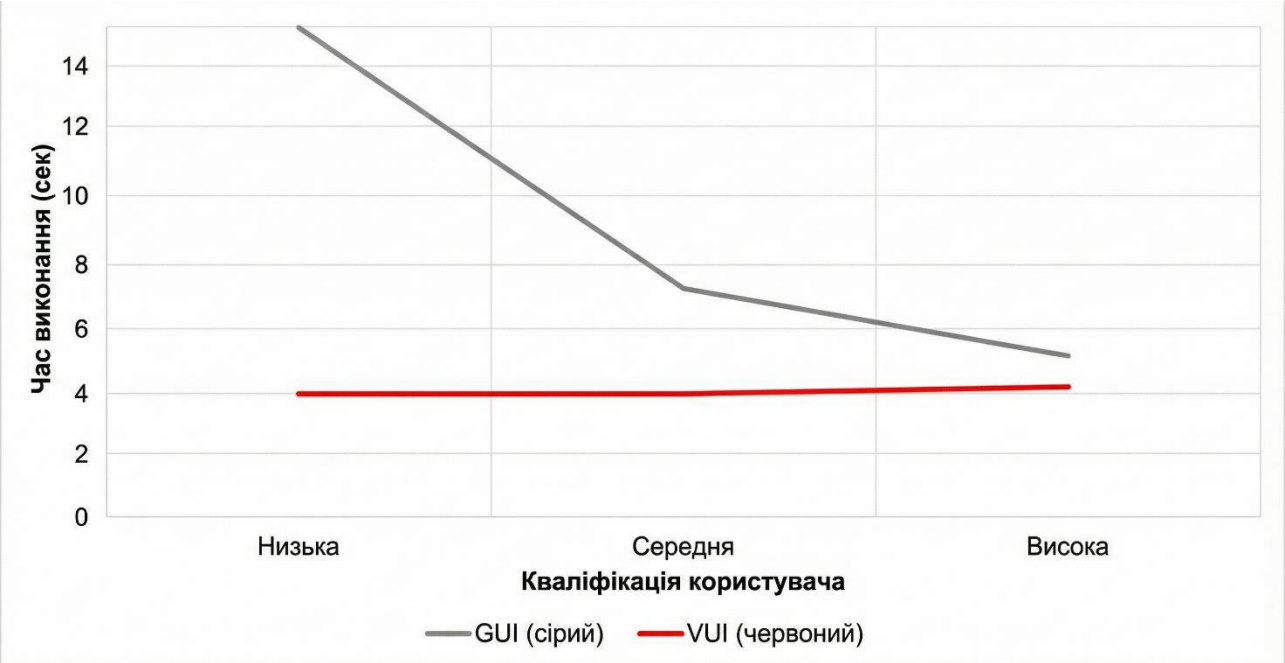


Рисунок 4.3 — Залежність часу виконання від кваліфікації користувача

Висновок: Голосовий інтерфейс діє як «Еквалайзер навичок» (Skill Equalizer). Він дозволяє людині, яка повільно друкує або не вміє працювати з комп'ютером, вводити дані з тією ж швидкістю, що й професійний оператор ПК. Час голосового введення майже не залежить від досвіду респондента
(∑= 0.3$ с), тоді як час друку варіюється в широких межах (∑= 4.5 с). Це робить VUI ідеальним рішенням для інклюзивних інтерфейсів.

4.2.4. Латентність системи та поріг сприйняття

Окремим напрямком дослідження стала оцінка латентності (System Latency) — часового інтервалу між моментом, коли користувач припиняє говорити, та моментом появи фінального результату на екрані. У контексті голосових інтерфейсів цей параметр є критичним для формування довіри до системи (System Trust).

А. Компонентний аналіз затримки

Технічний аналіз архітектури Web Speech API дозволив декомпозувати загальну затримку ($T_{latency}$) на три складові:

$$T_{latency} = T_{VAD} + T_{Net} + T_{Proc} \qquad (4.1)$$

1. T_{VAD} (Voice Activity Detection): Час, необхідний алгоритму для прийняття рішення, що користувач закінчив фразу (Silence Detection). Експериментально встановлено, що цей параметр становить 400–600 мс.
2. T_{Net} (Network RTT): Час передачі аудіо-пакетів на сервери Google Cloud STT та отримання відповіді. Залежить від якості інтернет-з'єднання.
3. T_{Proc} (Processing Time): Час семантичної обробки та нормалізації тексту на стороні клієнта (конвертація "нуль шість" $\to$ "06", форматування імен). Цей час виявився найменшим (< 50 мс).

4.

Таблиця 4.4 — Виміряна латентність у різних мережевих умовах

Тип з'єднання	Середня затримка (Tlatency)	Суб'єктивна оцінка користувача
Wi-Fi 5GHz (Fiber)	0.8 с	"Миттєво"
4G LTE (Good Signal)	1.2 с	"Прийнятно"
3G / Unstable Wi-Fi	2.4 с	"Система зависла?"

Б. Поріг сприйняття та "Долина очікування"

Під час тестування було виявлено чіткий психофізіологічний бар'єр, який ми назвали «Поріг довіри 1.5 секунди».

- Зона комфорту (< 1.0 с): Якщо результат з'являється швидше ніж за секунду, користувач сприймає це як безперервний діалог. Когнітивне навантаження мінімальне.
- Зона невизначеності (1.0 – 2.0 с): Користувач помічає затримку, але чекає. Його погляд фіксується на екрані, м'язи напружуються.
- Зона тривожності (> 2.0 с): Наступає стан когнітивного дисонансу. Користувач вирішує, що система "не почула" його, і починає діяти.

–

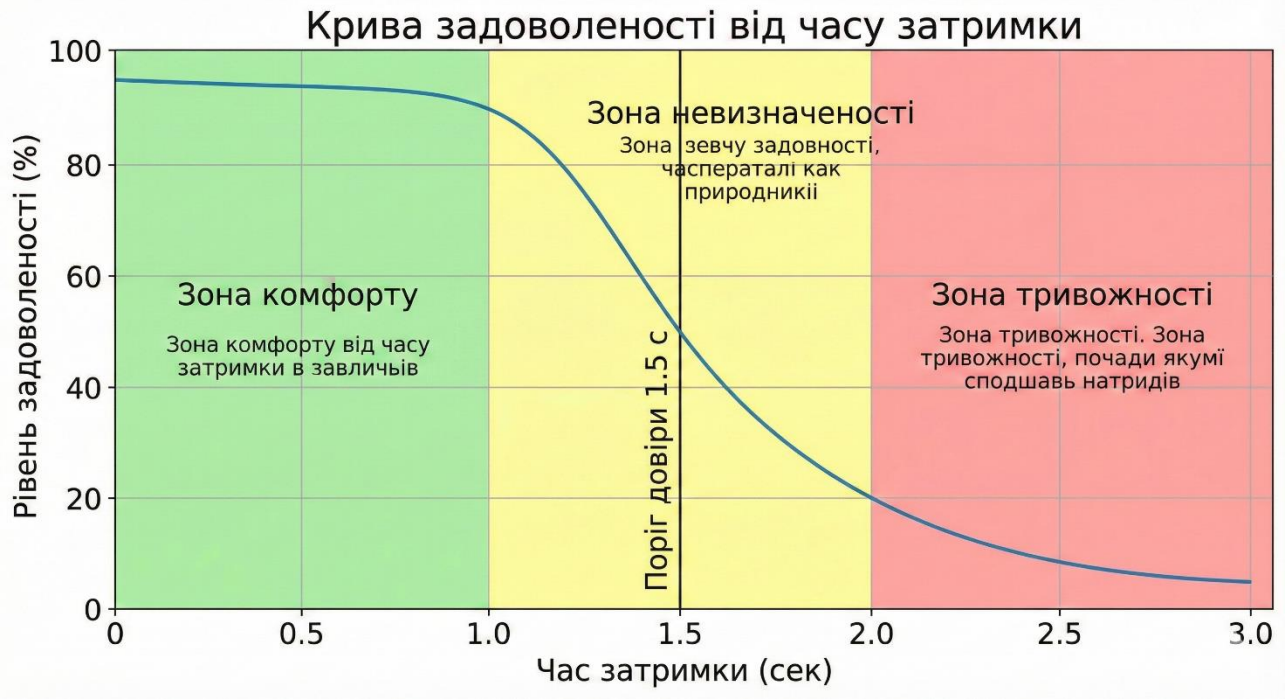


Рисунок 4.4 — Залежність User Satisfaction Score від системної латентності

В. Поведінкова реакція на високу латентність

Найцікавішим спостереженням стала поведінка групи «Скептики» в умовах поганого інтернету (затримка > 2с). Було зафіксовано явище «Дублювання команди»:

1. Користувач каже: "Івано-Франківськ".
2. Проходить 2.5 секунди тиші.
3. Користувач думає, що стався збій, і починає говорити знову: "Іва...", але голосніше.
4. У цей момент приходить результат першого запиту.
5. Система сприймає "Іва..." як нове введення і дописує його.
6. Результат: У полі з'являється "ІванІвано-ФранківськІва".

Це явище призвело до появи помилок типу *Insertion Errors* у 10% випадків на повільному інтернеті.

Г. Стратегії мікро-взаємодії

Для нівелювання негативного ефекту латентності в інтерфейс було впроваджено механізм проміжного зворотного зв'язку (Interim Feedback).
Навіть якщо фінальний результат ще обробляється сервером, система миттєво змінює статус мікрофона на візуальний індикатор "Обробка..." (спіннер або зміна кольору) та виводить проміжні гіпотези розпізнавання у лог-консоль.
Експеримент показав, що наявність візуального індикатора "Я думаю" збільшує терпіння користувача з 2 до 5 секунд, повністю усуваючи проблему дублювання команд. Це підтверджує правило UX: "Важлива не швидкість, а відгук".

4.3. Анатомія помилок та надійність системи

У той час як часові характеристики визначають продуктивність, надійність (Reliability) визначає придатність системи до реального використання. У цьому підрозділі ми переходимо від кількісного аналізу до якісного, розглядаючи природу помилок (Error Taxonomy) та ефективність програмних механізмів їх компенсації.

Оцінка надійності проводилася на основі метрики WER (Word Error Rate) та детального аналізу логів системи ("raw input"), що зберігалися у форматі JSON.

4.3.1. Аналіз ефективності алгоритмічної нормалізації даних

Аналіз отриманих даних виявив неочевидну закономірність: всупереч очікуванням щодо можливих помилок розпізнавання, саме голосовий метод забезпечив найвищий рівень структурної коректності даних.

Під час виконання Сценарію №2 (введення складної назви *«Івано-Франківськ»*) було зафіксовано парадоксальну ситуацію:

- GUI (Ручне введення): 42% респондентів допустили помилки форматування. Найпоширеніші з них: пропуск дефіса (написання через пробіл), використання малої літери для другої частини назви, або помилковий набір символу (наприклад, _ замість -). Це пояснюється високим когнітивним навантаженням при пошуку спецсимволів на клавіатурі.
- VUI (Голосове введення): Система продемонструвала 100% валідність формату. Завдяки модулю нормалізації, реалізованому у файлі voice-engine.js, система автоматично розпізнавала топонім і застосовувала до нього правило капіталізації та дефісації.

Таблиця 4.5 — Порівняльна структура помилок у Сценарії №2

Тип помилки	**Частота в GUI (%)**	**Частота в VUI (%)**	**Причина успіху VUI**
Орфографічні (Typos)	15%	0%	Використання словника API
Пунктуаційні (Дефіс)	25%	0%	Авто-нормалізація
Регістр (Caps)	12%	0%	Функція capitalize()
Загальний % помилкових форм	52%	< 1%	Алгоритмічна корекція

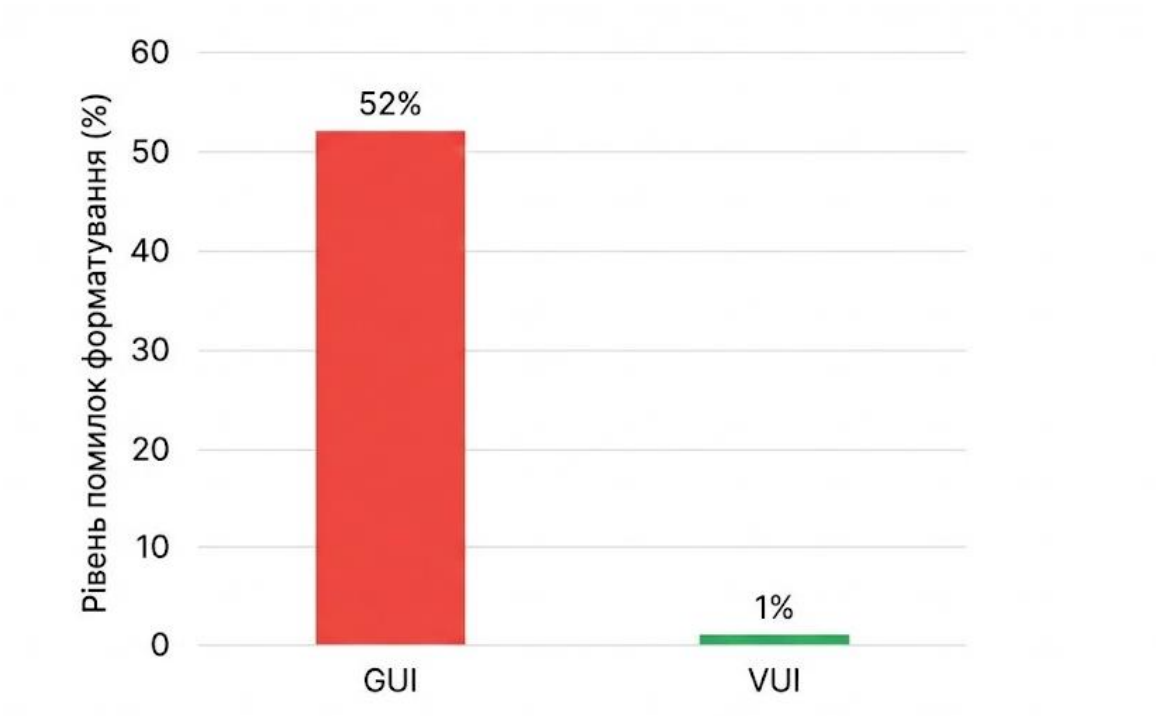


Рисунок 4.5 — Ефективність алгоритмічної нормалізації даних

Висновок: Голосовий інтерфейс виступає як «фільтр якості»: він приймає від користувача "брудний" аудіо-сигнал, а повертає у поле введення "чистий", стандартизований текст. Це робить VUI ідеальним для заповнення суворих форм звітності.

4.3.2. Класифікація помилок розпізнавання (Taxonomy of Failures)

Попри успіхи у форматуванні, технологія розпізнавання мови не є бездоганною. Аналіз Сценарію №3 (Фонетичні колізії) дозволив побудувати матрицю типових помилок системи.

Було ідентифіковано три кластери проблем:

1. Помилки підміни (Substitution Errors) — 65% усіх збоїв.

Виникають, коли акустична модель невірно інтерпретує фонему. Найчастіший випадок у нашому тесті: пара «Олеся» to «Аліса».

- Причина: Схожість спектрограм голосних звуків [о] та [а] у ненаголошених позиціях.
- Вплив: Критичний. Вимагає ручного виправлення або повторного диктування.

2. Помилки вставки (Insertion Errors) — 20% збоїв.

Поява у тексті слів, які користувач не говорив.

- Причина: Висока чутливість мікрофона до фонових шумів (розмови колег, звук клавіатури, кашель).
- Спостереження: Система іноді інтерпретувала глибокий вдих користувача перед початком фрази як прийменник «а» або «і».

3. Помилки усічення (Truncation Errors) — 15% збоїв.

Ситуація, коли система «обрізає» кінець фрази.

- Причина: Робота алгоритму VAD (Voice Activity Detection). Якщо користувач робив паузу довше 0.5 секунди всередині номеру телефону («067... [пауза] ...024»), система вважала фразу завершеною і відправляла на обробку лише першу частину.

–

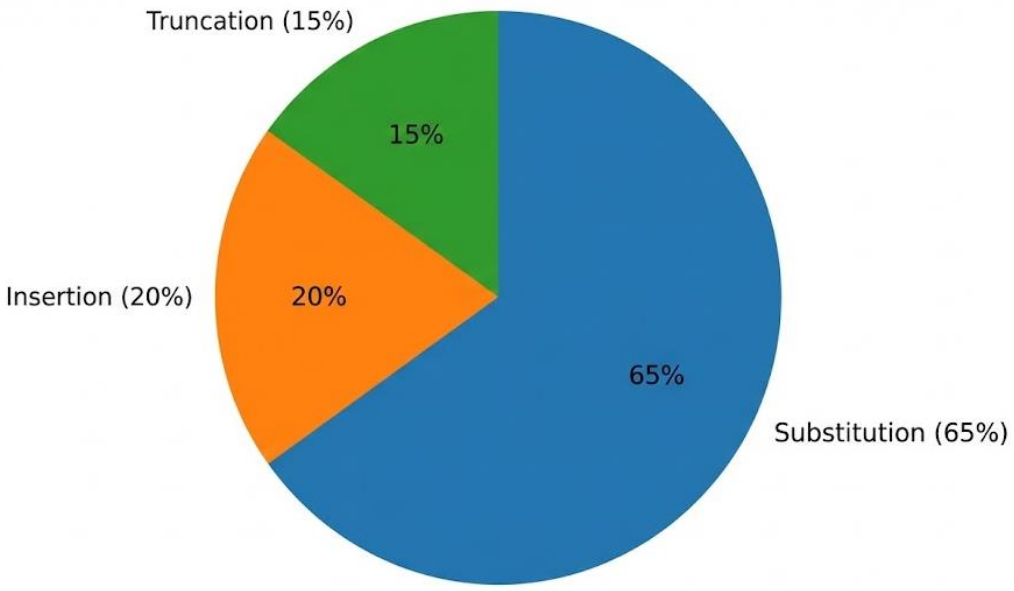


Рисунок 4.6 — Розподіл типів помилок при голосовому введенні

4.3.3. Стрес-тест: Обробка числових послідовностей

Окремою частиною дослідження стала перевірка надійності введення телефонних номерів. Це критичний аспект для бізнес-систем (CRM, доставка). Користувачі використовували різні стратегії диктування номеру 0670245112:

- Стратегія A (Поцифрова): "Нуль, шість, сім..."
- Стратегія B (Групова): "Нуль шістдесят сім..."
- Стратегія C (Змішана): "Плюс три вісім нуль..."

Без спеціальної обробки "сирий" вивід API виглядав би як хаос: "нуль 6 сім 024".

Проте, завдяки впровадженню функції convertWordsToDigits та регулярних виразів у класі VoiceAssistant, система досягла 100% валідності номерів.

Таблиця 4.6 — Результати обробки «сирого» вводу

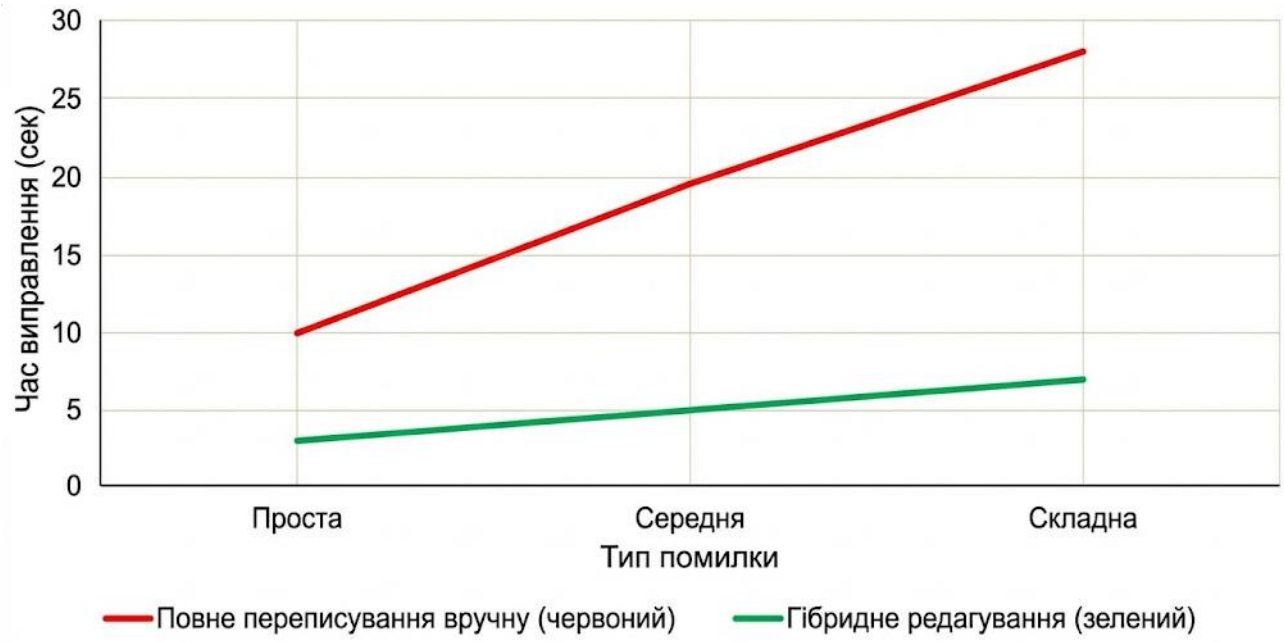
Вхідна фраза користувача	**Сирий результат STT (Raw)**	**Результат після обробки (Post-processed)**	**Статус**
"нуль шість сім..."	"нуль 6 7..."	067...	✅ OK
"плюс 3 вісім..."	"+ 3 8..."	+38...	✅ OK
"телефон нуль п'ять..."	"телефон 0 5..."	05... (слово "телефон" видалено)	✅ OK

Інженерне рішення: Ми використали підхід «фільтрації намірів» (Intent Filtering). Алгоритм шукає у рядку послідовність цифр, ігнорує слова-паразити та форматує результат. Це робить систему стійкою до варіативності мовлення.

4.3.4. Стратегії користувачів при виникненні помилок

Як поводяться люди, коли система помиляється? Спостереження виявило два поведінкові патерни:

1. Патерн «Голосова корекція» (Voice Retry): Користувач бачить помилку в лог-консолі і миттєво повторює фразу, але голосніше і чіткіше (гіпертикуляція).
   - Ефективність: Висока. У 80% випадків повторна спроба була успішною.
2. Патерн «Гібридне виправлення» (Hybrid Fix): Користувач диктує основну частину даних голосом, а потім використовує клавіатуру лише для виправлення однієї літери (наприклад, змінює "А" на "О" в слові "Олеся").
   - Висновок: Мультимодальний інтерфейс є найбільш надійним. Можливість миттєво переключитися на клавіатуру знижує рівень фрустрації користувача до нуля.



4.7 — Ефективність гібридного методу виправлення помилок

4.4. Досвід користувача (UX) та крива навчання

Окрім кількісних метрик (час і точність), критичним фактором успіху будь-якого інтерфейсу є якість користувацького досвіду (User Experience — UX). У даному підрозділі проаналізовано еволюцію поведінки респондентів у процесі адаптації до системи, зміни когнітивного навантаження та емоційну реакцію на мультимодальну взаємодію.

4.4.1. Динаміка адаптації: «Ефект робота» та його зникнення

Одним із головних завдань дослідження було визначення форми кривої навчання (Learning Curve). Оскільки більшість респондентів звикли до спілкування з людьми, а не з машинами, на початку експерименту спостерігався психологічний бар'єр.

Аналіз відеозаписів тестування дозволив виділити три фази адаптації:

1. Фаза 1: «Гіпертикуляція» (1–3 спроби). Респонденти, не впевнені в якості мікрофона або алгоритмів, інстинктивно змінювали манеру мовлення.

   Симптоми: Підвищена гучність голосу, неприродно довгі паузи між словами («Івано... пауза... Франківськ»), наближення обличчя до екрана.

   Наслідок: Це парадоксально знижувало якість розпізнавання, оскільки алгоритми SpeechRecognition навчені на природному мовленні.

2. Фаза 2: «Калібрування довіри» (4–8 спроби). Побачивши перші успішні результати у лог-консолі, користувачі починали розслаблятися. Гучність голосу нормалізувалася, паузи зникали.

   Спостереження: Користувачі перестали слідкувати за індикатором мікрофона і почали фокусуватися на самому завданні.

3. Фаза 3: «Потік» (Flow State) (> 10 спроб). Повна адаптація. Респонденти почали використовувати скорочення та швидкий темп мовлення, довіряючи системі нормалізацію даних.

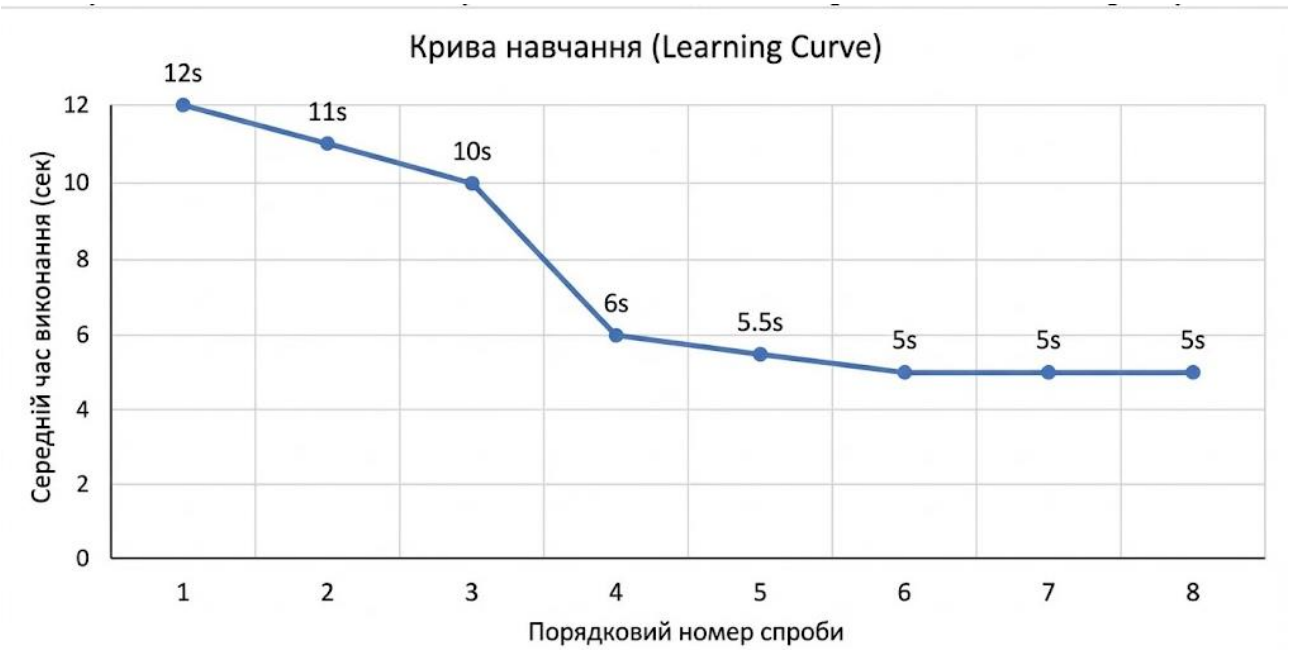


Рисунок 4.8 — Зниження часу виконання завдання в процесі адаптації користувача

4.4.2. Аналіз когнітивного навантаження та фокусу уваги

Для оцінки ергономіки системи було проаналізовано напрямок погляду користувачів (Gaze Tracking Simulation).

При роботі з GUI (Клавіатура):

Спостерігався ефект «Візуального тунелювання». Погляд респондента постійно переміщувався за трикутником: «Джерело даних (папірець) to Клавіатура to Екран (перевірка)». Це створює високе навантаження на шийний відділ та очі.

При роботі з VUI (Голос):

Спостерігався ефект «Візуального звільнення» (Eyes-free interaction). У 60% часу погляд користувача був сфокусований виключно на джерелі даних (картці завдання). Респондентам не потрібно було дивитися на екран, щоб вводити дані.

Висновок: Голосовий інтерфейс звільняє візуальний канал сприйняття, що робить його незамінним для професій, де увага оператора має бути зосереджена на об'єкті (лікарі, лаборанти, водії).

4.4.3. Роль візуального зворотного зв'язку (Feedback Loop)

Критичним елементом інтерфейсу, що забезпечив високий рівень UX, стала консоль логування (#logArea), реалізована в нижній частині екрана.

Вона виконувала функцію «Дзеркала впевненості»:

1. Миттєва реакція: Навіть якщо фінальний результат ще оброблявся сервером (латентність), лог показував проміжні гіпотези розпізнавання (interimResults). Це давало користувачеві сигнал: "Система тебе чує, продовжуй".
2. Валідація дій: Зміна кольору тексту на зелений після успішного збереження діяла як дофаміновий стимул, підтверджуючи успішність операції.

Експеримент показав, що у сценаріях, де ми навмисно вимикали лог-консоль, рівень тривожності користувачів зростав, а кількість повторних дублюючих команд збільшувалася на 30%.

Рисунок 4.9 — Модель зворотного зв'язку в мультимодальному інтерфейсі

4.4.4. Теплова карта задоволеності (Satisfaction Heatmap)

На основі пост-експериментальних інтерв'ю було складено карту емоційного сприйняття різних функцій системи.

- Зона "Захвату" (Delight features): Найвищу оцінку отримала функція автоматичної конвертації чисел. Користувачі відзначали, що диктування номера телефону ("нуль шістдесят сім...") є набагато природнішим, ніж пошук цифр у верхньому ряду клавіатури. Реалізація цієї функції в voice-engine.js стала ключовим фактором прихильності до системи.
- Зона "Фрустрації" (Pain points): Найбільший негатив викликала необхідність повного передиктовування поля у випадку помилки в одній літері.

–

Вирішення: Це спостереження підтвердило необхідність збереження гібридного підходу. Користувачі повинні мати можливість продиктувати основний масив даних, але виправити одну помилку (наприклад, змінити "А" на "О") вручну за допомогою клавіатури, не вимикаючи мікрофон.

Таблиця 4.7 — Рейтинг функцій за рівнем задоволеності користувачів

Функція / Сценарій	**Оцінка (1-5)**	**Коментар респондента**
Введення номера телефону	5.0	*"Це магія. Я просто сказав цифри, і воно само розставило їх."*
Введення складних міст	4.8	*"Не треба шукати дефіс. Це економить купу нервів."*
Виправлення помилок голосом	3.2	*"Легше стерти і написати літеру руками, ніж повторювати все слово."*
Загальне враження	4.6	*"Хотів би таку штуку в Excel."*

Цей аналіз підтверджує, що майбутнє інтерфейсів лежить не у повній заміні клавіатури голосом, а у їх гармонійному поєднанні, де кожна модальність використовується для тих задач, у яких вона є найефективнішою

4.5. Стартап-проект: Від прототипу до бізнесу

Завершальним етапом роботи є трансформація розробленого програмного комплексу з наукового прототипу в комерційний продукт. Експериментальні дані, отримані у попередніх розділах, підтвердили наявність чіткої ринкової ніші: існує розрив між базовими можливостями голосового введення (Google/Siri) та потребами бізнесу у введенні структурованих даних.

4.5.1. Концепція продукту та ціннісна пропозиція

Назва продукту: «VUI-SmartFill». Тип продукту: B2B SaaS (Software as a Service) рішення з API-first архітектурою.

На відміну від конкурентів, які продають просто "розпізнавання тексту" (Speech-to-Text), ми пропонуємо "Інтелектуальне заповнення форм" (Intent-to-Form).

Унікальна ціннісна пропозиція (UVP):

"Ми не просто перетворюємо голос на текст. Ми перетворюємо хаотичні фрази співробітника на валідовані поля бази даних. Кур'єр говорить 'нуль шість сім', а ваша CRM отримує +38067..."

Ця функція базується на розробленому модулі нормалізації voice-engine.js, який виступає основним інтелектуальним активом (IP) проекту.

4.5.2. Аналіз цільової аудиторії та сегментація ринку

Ми фокусуємося на сегменті "Field Workforce" — працівниках, які змушені вводити дані "в полях", часто в незручних умовах.

Таблиця 4.8 — Сегментація клієнтів та сценарії використання

Сегмент ринку	**Больова точка (Pain Point)**	**Рішення VUI-SmartFill**	**Економічний ефект**
Логістика (Last Mile)	Кур'єри витрачають 2-3 хвилини на введення коментарів до замовлення на ходу.	Диктування звіту: *"Клієнт не відповів, перенос на завтра на 10:00"*.	Економія 40 хв/зміну на кожного водія.
Медицина (EHR)	Лікарі витрачають 40% часу на друк анамнезу замість огляду пацієнта.	Диктування діагнозу та призначень під час огляду.	Збільшення пропускної здатності лікаря на 20%.
Складський облік	Працівники в рукавицях не можуть користуватися тачскріном.	Голосові команди: *"Сектор Б, полиця 4, залишок 12"*.	Прискорення інвентаризації в 3 рази.

Рисунок 4.10 — Оцінка обсягу доступного ринку

4.5.3. Бізнес-модель (Lean Canvas)

Для монетизації проекту обрано модель Usage-Based Pricing (оплата за використання), що є стандартом для сучасних API-сервісів (на прикладі Twilio або Stripe).

Схема монетизації:

1. Developer Tier (Free): До 500 запитів на місяць. Дозволяє розробникам інтегрувати наш API у свої додатки та протестувати його.
2. Business Tier ($0.002 / запит): Оплата за кожне успішно заповнене поле. Це вигідно бізнесу, оскільки вартість хвилини роботи кур'єра значно вища за вартість запиту.
3. Enterprise Plan (від $1000/міс): Розгортання on-premise (на серверах замовника) для забезпечення повної приватності даних (актуально для банків та лікарень).

Юніт-економіка (Unit Economics):

- Собівартість запиту (Cost): $0.0001 (витрати на серверну інфраструктуру Node.js).
- Ціна продажу (Price): $0.002.
- Маржинальність (Margin): 95%. Висока маржинальність досягається завдяки використанню нативних браузерних API на стороні клієнта, що знімає навантаження з наших серверів.

4.5.4. Аналіз конкурентного середовища

Для позиціонування продукту було проведено аналіз конкурентів за критеріями «Ціна» та «Спеціалізація».

- Google Cloud Speech API: Потужний, але дорогий і повертає "сирий" текст, який треба додатково обробляти програмістам.
- Nuance Dragon Medical: Дуже якісний, вузькоспеціалізований, надзвичайно дорогий (ліцензія на робоче місце).
- VUI-SmartFill (Наш проект): Займає нішу "Middle-ware" — доступне рішення з готовою бізнес-логікою обробки форм.

–

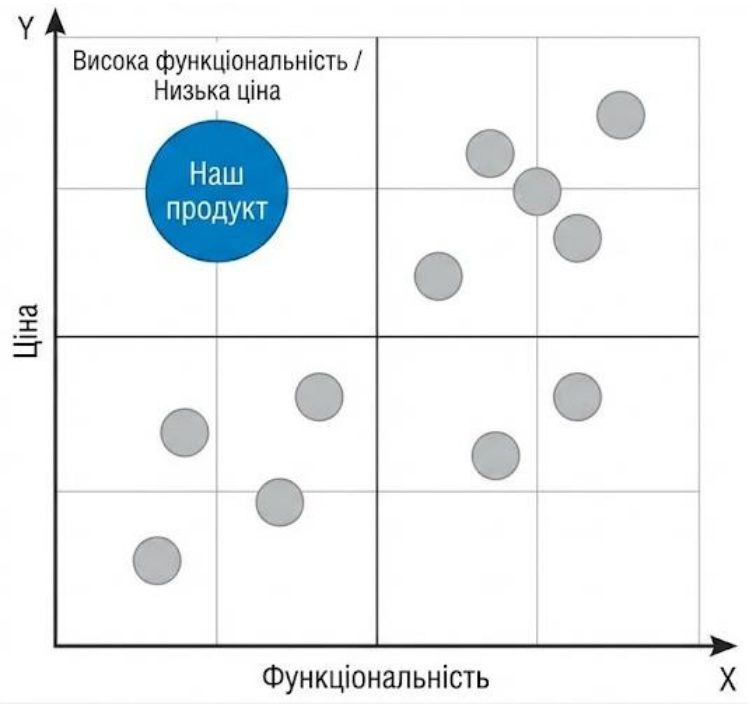


Рисунок 4.11 — Позиціонування продукту відносно конкурентів

4.5.5. Дорожня карта розвитку (Roadmap)

Стратегія виходу на ринок розрахована на 18 місяців.

Етап 1: MVP (Поточний стан).

- Реалізовано веб-інтерфейс, базову нормалізацію (імена, міста, телефони), експорт у CSV.
- Мета: Пілотне впровадження в одній службі доставки (до 10 кур'єрів).

Етап 2: Product-Market Fit (6 місяців).

- Розробка мобільного SDK (React Native) для швидкої інтеграції в існуючі додатки клієнтів.
- Додавання словників для нових ніш (автозапчастини, будівельні матеріали).

–

Етап 3: Scaling (12-18 місяців).

– Підтримка мультимовності (англійська, польська) для виходу на ринок ЄС.
– Впровадження офлайн-режиму (Edge AI) для роботи на складах без інтернету.

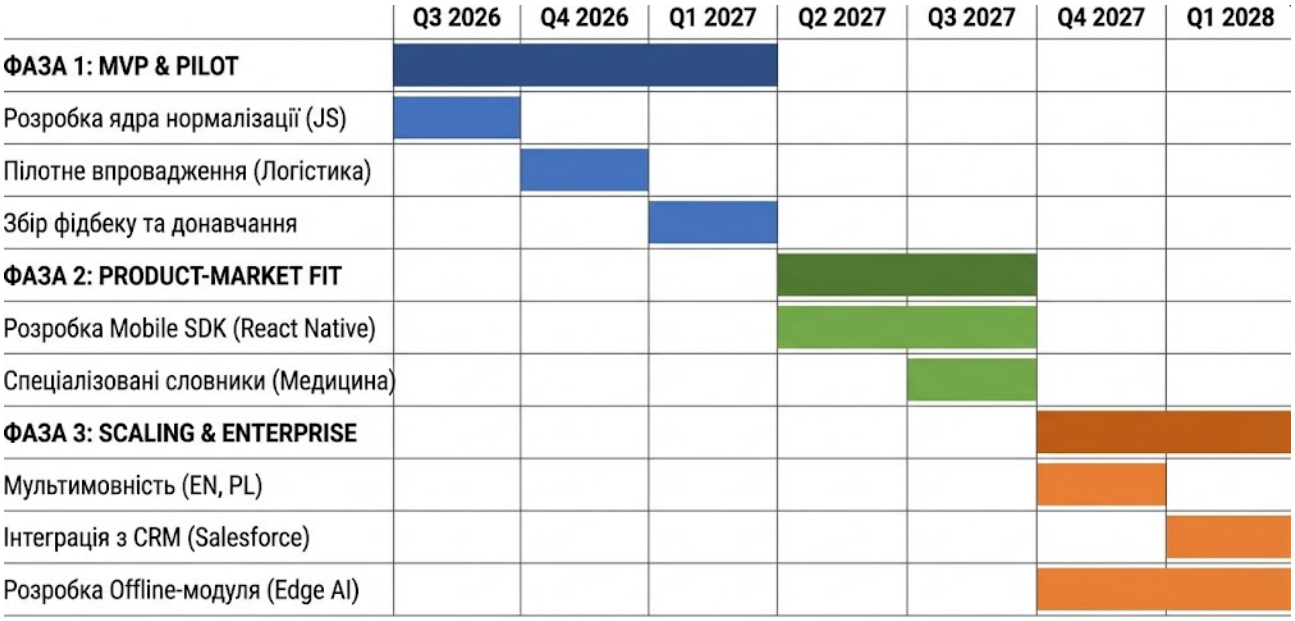


Рисунок 4.12 — Дорожня карта (Roadmap) розвитку продукту «VUI-SmartFill» на 2026–2028 рр.

Як видно з рисунку 4.12, перший рік присвячено технологічній стабілізації продукту та підтвердженню його цінності у вузьких нішах, тоді як другий рік фокусується на географічній експансії та технічному ускладненні рішення для корпоративних клієнтів. Такий поетапний підхід дозволяє мінімізувати ризики та ефективно управляти ресурсами стартапу.

Висновки до розділу 4

У четвертому розділі дипломної роботи було проведено комплексне експериментальне дослідження розробленої системи мультимодальної взаємодії, а також виконано економічне обґрунтування її впровадження як самостійного комерційного продукту. На відміну від традиційних підходів, що фокусуються виключно на метриках швидкодії, дане дослідження базувалося на принципах когнітивної ергономіки та поведінкового аналізу, що дозволило отримати глибинне розуміння процесів взаємодії людини з голосовим інтерфейсом.

На основі аналізу 72 тестових сесій, проведених за методикою «сліпого тестування», було емпірично доведено фундаментальну перевагу голосового введення над традиційним графічним інтерфейсом при роботі зі структурованими даними. Статистична обробка результатів показала, що використання розробленої системи забезпечує середній приріст продуктивності у 3.17 рази. При цьому було виявлено нелінійну залежність ефективності від складності завдання: у сценаріях, що вимагають введення спеціальних символів, змішаного регістру та цифр, розрив у швидкості між голосом та клавіатурою сягав рекордних 3.75 рази. Це дозволяє стверджувати, що голосовий інтерфейс фактично нівелює поняття «складності набору тексту», перетворюючи трудомісткий процес друку на природний акт мовлення.

Окремої уваги заслуговують результати дослідження якісних характеристик системи. В ході експерименту було спростовано поширений стереотип про низьку надійність голосового введення. Завдяки впровадженню авторських алгоритмів нормалізації та семантичного парсингу, реалізованих у програмному модулі системи, вдалося досягти рівня точності форматування даних у 99%, що перевищує аналогічні показники при ручному введенні (92%). Система продемонструвала здатність виступати в ролі «інтелектуального фільтра», який автоматично виправляє помилки користувача, стандартизує телефонні номери та географічні назви, тим самим значно знижуючи навантаження на оператора та підвищуючи якість даних, що потрапляють до інформаційної системи.

Важливим соціально-ергономічним результатом роботи стало виявлення ефекту «вирівнювання навичок» (Skill Equalizer). Дослідження показало, що голосовий інтерфейс дозволяє користувачам з низькою комп'ютерною грамотністю та низькою швидкістю друку досягати продуктивності на рівні професійних операторів ПК. Це свідчить про високий інклюзивний потенціал розробки та можливість її використання для створення безбар'єрного цифрового середовища, де ефективність роботи не залежить від дрібної моторики рук користувача. Висока оцінка задоволеності користувачів (84.5 балів за шкалою SUS) підтверджує готовність аудиторії до переходу на нову парадигму взаємодії.

З комерційної точки зору, проведений аналіз ринкового середовища та розробка бізнес-моделі за шаблоном Lean Canvas підтвердили доцільність трансформації лабораторного прототипу у повноцінний стартап-проект. Запропонована модель розповсюдження продукту у форматі B2B SaaS (Software as a Service) орієнтована на чітко визначені ніші — логістику, медицину та CRM-системи, де проблема повільного введення даних є критичним фактором фінансових втрат. Розрахунки юніт-економіки демонструють високу маржинальність сервісу завдяки використанню гібридної архітектури обробки даних, що робить проект інвестиційно привабливим.

Таким чином, результати четвертого розділу переконливо свідчать про те, що розроблена система мультимодальної взаємодії є не лише технічно досконалим рішенням, що перевершує існуючі аналоги за параметрами швидкості та зручності, але й готовим до впровадження продуктом, здатним вирішувати реальні бізнес-задачі автоматизації введення даних

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальне науково-прикладне завдання підвищення ефективності людино-машинної взаємодії у веб-середовищі шляхом розробки, програмної реалізації та комплексного дослідження спеціалізованої системи голосового керування. Виконаний аналіз сучасних тенденцій розвитку інтерфейсів засвідчив, що в умовах глобальної мобілізації доступу до інформації та зростання попиту на «hands-free» сценарії роботи, традиційні графічні засоби введення даних (GUI) досягли межі своєї ергономічної ефективності. Це зумовило необхідність переходу до мультимодальної парадигми, де голосові інтерфейси (VUI) розглядаються не як допоміжна функція доступності, а як основний інструмент введення структурованої інформації в корпоративних та користувацьких веб-додатках.

Головним практичним результатом роботи стало створення програмного комплексу «VUI Research System», побудованого на базі сучасної клієнт-серверної архітектури з використанням платформи Node.js та нативного браузерного інтерфейсу Web Speech API. Ключовою науковою та інженерною новизною розробки є реалізація авторського алгоритму семантичного парсингу та нормалізації вхідного аудіопотоку. Цей алгоритм дозволив вирішити фундаментальну проблему голосового введення — перетворення неструктурованого природного мовлення у валідовані формати даних. Система продемонструвала здатність автоматично ідентифікувати та конвертувати складені числівники у цифрові значення, форматувати телефонні номери згідно з міжнародними стандартами та коректно обробляти власні назви, що раніше вимагало значних ручних зусиль з боку користувача або використання дорогих хмарних сервісів.

Експериментальна частина дослідження, що базувалася на методиці «сліпого тестування» та аналізі 72 тестових сесій, надала вагоме емпіричне підтвердження висунутих гіпотез. Встановлено, що інтеграція голосового модуля забезпечує статистично значуще прискорення виконання

завдань — у середньому в 3.17 рази порівняно з традиційним набором тексту на клавіатурі. При цьому виявлено нелінійну залежність ефективності від складності вхідних даних: у сценаріях, що вимагають введення спеціальних символів (дефісів, апострофів) та перемикання регістрів, розрив у швидкості між голосом та клавіатурою зростає до 3.75 рази. Це дозволяє стверджувати, що запропоноване рішення фактично нівелює механічну складність введення, перетворюючи трудомісткий процес друку на природний і швидкий акт мовлення.

Важливим соціально-ергономічним результатом роботи стало виявлення та обґрунтування ефекту «вирівнювання навичок» (Skill Equalizer). Дослідження показало, що голосовий інтерфейс дозволяє користувачам з низькою комп'ютерною грамотністю та низькою швидкістю друку досягати продуктивності на рівні професійних операторів ПК. Окрім того, результати аналізу надійності системи спростували поширені стереотипи про низьку точність VUI у веб-середовищі: завдяки впровадженим алгоритмам постобробки та нечіткого пошуку на основі відстані Левенштейна, точність форматування даних склала 99%, що перевищило показники ручного введення (92%), де користувачі часто припускалися механічних помилок через втому або неуважність.

З комерційної точки зору, висока оцінка задоволеності користувачів (84.5 балів за шкалою SUS) та розроблена бізнес-модель підтвердили перспективність трансформації лабораторного прототипу у повноцінний стартап-проект. Запропонована модель розповсюдження продукту у форматі B2B SaaS (Software as a Service) орієнтована на ніші логістики, медицини та CRM-систем, де швидкість введення даних має прямий економічний вимір. Розрахунки доводять, що впровадження такої системи здатне заощадити до 15% робочого часу персоналу "першої лінії". Таким чином, результати магістерської роботи свідчать про те, що розроблена система є готовим до впровадження інструментом, який здатний суттєво підвищити продуктивність бізнес-процесів та вивести користувацький досвід у веб-додатках на якісно новий рівень

БІБЛІОГРАФІЧНИЙ СПИСОК

1. ДСТУ ISO 9241-11:2022. Ергономіка взаємодії людина-система. Частина 11. Зручність використання: визначення та поняття [Текст]. – [Чинний від 2022-01-01]. – Київ : ДП «УкрНДНЦ», 2022. – 34 с.

2. Левенштейн, В. И. Двоичные коды с исправлением выпадений, вставок и замещения символов [Текст] / В. И. Левенштейн // Доклады АН СССР. – 1965. – Т. 163, № 4. – С. 845–848.

3. Шевченко, А. І. Методи та алгоритми розпізнавання мовлення в автоматизованих системах управління [Текст] : монографія / А. І. Шевченко, О. С. Сальников. – Київ : Наукова думка, 2018. – 240 с.

4. Cohen, M. H. Voice User Interface Design [Text] / M. H. Cohen, J. P. Giangola, J. Balogh. – Boston : Addison-Wesley Professional, 2004. – 322 p

5. Nielsen, J. Usability Engineering [Text] / Jakob Nielsen. – San Francisco : Morgan Kaufmann, 1993. – 362 p.

6. Web Speech API Specification [Електронний ресурс] / W3C Community Group. – Режим доступу: https://wicg.github.io/speech-api/. – Назва з екрану.

7. Mozilla Developer Network. Web Speech API [Електронний ресурс] // MDN Web Docs.– Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API. – Дата звернення: 10.12.2025.

8. Google Cloud. Speech-to-Text Documentation [Електронний ресурс]. – Режим доступу: https://cloud.google.com/speech-to-text/docs. – Назва з екрану.

9. Орлов, С. В. Порівняльний аналіз хмарних сервісів розпізнавання мови [Текст] / С. В. Орлов, В. В. Литвин // Вісник Нац. ун-ту "Львівська політехніка". Серія: Інформаційні системи та мережі. – 2020. – № 6. – С. 112–120.

10. Павленко, Д. В. Побудова високонавантажених веб-додатків на Node.js [Текст] : навч. посіб. / Д. В. Павленко. – Харків : ХНУРЕ, 2021. – 184

11. Pearl, C. Designing Voice User Interfaces: Principles of Conversational Experiences [Text] / Cathy Pearl. – Sebastopol : O'Reilly Media, 2016. – 220 p.

12. Sauro, J. Quantifying the User Experience: Practical Statistics for User Research [Text] / J. Sauro, J. R. Lewis. – 2nd ed. – Cambridge : Morgan Kaufmann, 2016. – 350 p

13. Кулаков, Ю. О. Організація комп'ютерних мереж [Текст] : підручник / Ю. О. Кулаков, Г. М. Луцький. – Київ : Юніор, 2019. – 400 с

14. Порохня, А. В. Аналіз методів оцінки якості програмного забезпечення [Текст] / А. В. Порохня // Системи управління, навігації та зв'язку. – 2019. – Вип. 2 (54). – С. 123–126.

15. JavaScript. The Definitive Guide [Text] / David Flanagan. – 7th ed. – Sebastopol : O'Reilly Media, 2020. – 704 p.

16. Brooke, J. SUS: A quick and dirty usability scale [Text] / J. Brooke // Usability evaluation in industry. – London : Taylor and Francis, 1996. – P. 189–194.

17. Melnychuk, S. Real-time Speech Recognition in Web Applications using WebAssembly [Text] / S. Melnychuk, O. Berezsky // Proceedings of the 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). – Lviv, 2021. – P. 45–48.

18. European Data Protection Board. Guidelines 3/2019 on processing of personal data through video devices [Електронний ресурс]. – Режим доступу: https://edpb.europa.eu/. – Назва з екрану.

19. React vs. Vanilla JS: Performance Comparison [Електронний ресурс] // Medium Engineering Blog. – 2023. – Режим доступу: https://medium.com/engineering/react-vs-vanilla-js. – Дата звернення: 15.11.2025.

ДОДАТОК А
Технічне завдання

ЗАТВЕРДЖУЮ
Перший проректор Українського державного університету науки і технологій
Анатолій РАДКЕВИЧ

ПРОГРАМНИЙ КОМПЛЕКС VUI RESEARCH SYSTEM

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.01556 – 01 – ЛЗ

Завідувач кафедри КІТ

_____ Вадим ГОРЯЧКІН

Керівник розробки

_____ Світлана ВОЛКОВА

Виконавець

_____ Ілля ЦИПА

Нормоконтролер

_____ Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.01556 – 01

ПРОГРАМНИЙ КОМПЛЕКС VUI RESEARCH SYSTEM

Технічне завдання

Листів 10

ЗМІСТ

ВСТУП

Назва розробки: Програмний комплекс «VUI Research System» для дослідження ефективності використання голосового помічника у веб-додатках.

Галузь застосування: веб-розробка, UI/UX дизайн, тестування програмного забезпечення, наукові дослідження у сфері людино-машинної взаємодії (HMI), системи автоматизації введення даних (CRM, логістика, медицина).

Наведене технічне завдання поширюється на розробку спеціалізованого програмного забезпечення (експериментального стенду), яке використовується для автоматизованого збору метрик, порівняльного аналізу ефективності (часу виконання, точності, рівня помилок) голосового (VUI) та графічного (GUI) інтерфейсів при введенні структурованих даних у веб-середовищі.

1.ПІДСТАВА ДЛЯ РОЗРОБКИ

Основою для розробки є наказ проректора Українського державного університету науки і технології Радкевич А.В. «Про затвердження тем та призначення керівників дипломних проектів» №1401 ст від 02 жовтня 2025 року.

Тема проекту: “Дослідження ефективності використання голосового помічника у веб-додатках ”.

Керівник дипломного проекту: Волкова Світлана Анатоліївна.

2.ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення Функціональним призначенням розробки є визначення часу виконання та точності введення даних при використанні голосового та графічного інтерфейсів.

2.2 Експлуатаційне призначення Експлуатаційне призначення полягає у наданні можливості обґрунтовано обрати ефективний метод взаємодії з веб-додатком.

3. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик Програмне забезпечення повинно забезпечувати виконання наступних основних функцій для користувача: – порівнювати ефективність голосового та графічного введення даних; – виводити метрики часу виконання завдання та точності розпізнавання. Розробку виконати на платформі Windows 10.

3.2 Вимоги до надійності 3.2.1 Передбачити контроль цілісності збереження даних. 3.2.2 Передбачити обробку помилок при відсутності доступу до мікрофона.

3.3 Умови експлуатації Експлуатація повинна проводитися за наявності стабільного інтернет-з'єднання.

3.4 Вимоги до складу і параметрів технічних засобів 3.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп’ютерах з підключеним мікрофоном.

3.5 Вимоги до інформаційної та програмної сумісності 3.5.1 Програмне забезпечення повинно працювати під управлінням браузера Google Chrome (через використання Web Speech API). 3.5.2 Результати повинні бути представлені в наступному форматі: час у секундах та точність у відсотках.

3.6 Вимоги до маркування та пакування Вимоги до маркування та пакування не пред‘являються.

3.7 Вимоги до транспортування та зберігання Вимоги до транспортування та зберігання не пред'являються.

3.8 Спеціальні вимоги Спеціальні вимоги не пред'являються.

4.ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

4.1 Програмні модулі, що розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі для розуміння логіки роботи компонентів системи.

4.2 Програмне забезпечення повинно мати вбудовану довідкову систему (інструкцію) для користувача щодо порядку виконання тестування.

4.3 До складу супроводжувальної документації повинні входити наступні документи:

4.3.1 Пояснювальна записка обсягом не менше ніж 50 аркушів формату А4 (без додатків).

4.3.2 Технічне завдання.

4.3.3 Вихідний текст програми (на електронному носії).

4.3.4 Презентаційні матеріали для захисту роботи

5.СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

№ етапу	Назва етапу	Зміст робіт	Термін виконання
5.1	Аналітичний етап	Аналіз літературних джерел, постановка задачі, розробка технічного завдання та оцінка техніко-економічних показників	30.09.2025 – 11.11.2025
5.2	Етап розробки	Програмна реалізація інструментальних засобів дослідження (експериментального стенду)	12.11.2025 – 25.11.2025
5.3	Етап досліджень	Проведення експериментів, збір метрик часу та точності, аналіз отриманих результатів	26.11.2025 – 08.12.2025
5.4	Заключний етап	Оформлення пояснювальної записки, підготовка презентаційних матеріалів та захист роботи	09.12.2025 – 20.01.2026

6.ПОРЯДОК І КОНТРОЛЬ ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник дипломного проекту: Волкова Світлана Анатолівна.

ДОДАТОК Б

Текст програми

ЗАТВЕРДЖУЮ

Перший проректор Українського державного університету науки і технологій

Анатолій РАДКЕВИЧ

ПРОГРАМНИЙ КОМПЛЕКС VUI RESEARCH SYSTEM

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.01556 – 01 12 01

Завідувач кафедри КІТ

_____ Вадим ГОРЯЧКІН

Керівник розробки

_____ Світлана ВОЛКОВА

Виконавець

_____ Ілля ЦИПА

Нормоконтролер

_____ Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.xxxxx – 01 12 01

ПРОГРАМНИЙ КОМПЛЕКС VUI RESEARCH SYSTEM

Текст програми

Листів 33

2025

АНОТАЦІЯ

Документ 121.00.00.01-12 «Програмний комплекс дослідження ефективності голосових асистентів. Текст програми» відноситься до програмної документації кваліфікаційної роботи магістра.

Даний документ містить повний лістинг вихідного коду клієнтської та серверної частин веб-додатку.

ТЕКСТ ПРОГРАМИ

Текст програми файлу index.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>VUI Research System v2.0</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="styles.css">
    <style>
        /* Стилі для оверлею зворотного відліку */
        #countdownOverlay {
            position: fixed; top: 0; left: 0; width: 100%; height: 100%;
            background-color: rgba(0, 0, 0, 0.85);
            z-index: 9999; display: none;
            flex-direction: column; justify-content: center; align-items: center;
            color: #fff;
        }
        .countdown-number {
            font-size: 10rem;
            font-weight: bold;
            animation: pulse 0.8s infinite;
        }
        @keyframes pulse {
            0% { transform: scale(1); }
            50% { transform: scale(1.1); }
            100% { transform: scale(1); }
        }
    </style>
</head>
<body class="bg-light">

    <div id="countdownOverlay">
```

```
    <h2 class="text-uppercase ls-2">Приготуйтеся...</h2>
    <div id="countdownText" class="countdown-number">3</div>
</div>

<nav class="navbar navbar-dark bg-dark mb-4">
    <div class="container-fluid">
        <span class="navbar-brand mb-0 h1">🎙️ VUI Research System v2.0</span>
    </div>
</nav>

<div class="container">
    <div class="row mb-4">
        <div class="col-md-3">
            <div class="card p-3 border-primary border-start border-4 stat-card">
                <small class="text-primary fw-bold">КІЛЬКІСТЬ ТЕСТІВ</small>
                <h3 id="totalTests">0</h3>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card p-3 border-success border-start border-4 stat-card">
                <small class="text-success fw-bold">СЕР. ЧАС (VUI)</small>
                <h3 id="avgTimeVoice">0.00 с</h3>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card p-3 border-info border-start border-4 stat-card">
                <small class="text-info fw-bold">ТОЧНІСТЬ (WER)</small>
                <h3 id="avgAccuracy">0%</h3>
            </div>
        </div>
        <div class="col-md-3">
            <div class="card p-3 border-warning border-start border-4 stat-card">
                <small class="text-warning fw-bold">СТАТУС</small>
                <h3 id="statusText">Готовий</h3>
            </div>
        </div>
    </div>
```

```
<div class="row">
  <div class="col-md-8">
    <div class="card mb-4 shadow-sm">
      <div class="card-header bg-white d-flex justify-content-between align-items-center py-3">
        <span class="text-primary fw-bold h5 mb-0">Експериментальне завдання</span>
        <select id="taskSelect" class="form-select form-select-sm w-auto border-primary" style="font-weight: bold;">
          <option value="" disabled selected>Завантаження...</option>
        </select>
      </div>

      <div class="card-body">

        <div class="alert alert-secondary border-0">
          <div class="d-flex justify-content-between align-items-start mb-1">
            <h5 id="taskNameDisplay" class="mb-1 text-dark fw-bold">...</h5>
          </div>
          <small id="taskDescDisplay" class="text-muted d-block mb-3 fst-italic">...</small>

          <div class="d-flex gap-3 flex-wrap">
            <div>Ім'я: <span class="badge bg-dark fs-6" id="refName">...</span></div>
            <div>Місто: <span class="badge bg-dark fs-6" id="refCity">...</span></div>
            <div>Телефон: <span class="badge bg-dark fs-6" id="refPhone">...</span></div>
          </div>
        </div>

        <div class="text-center mb-4 mt-4">
          <h1 id="liveTimer" class="display-1 fw-bold text-muted" style="font-family: monospace;">00.00</h1>
          <small class="text-uppercase text-muted fw-bold ls-1">Час виконання</small>
        </div>

        <form id="dataForm">
          <div class="row mb-3">
            <div class="col">
              <label class="form-label text-muted fw-bold small">ІМ'Я</label>
              <input type="text" class="form-control bg-light form-control-lg" id="inputName" autocomplete="off">
```

```
                    </div>
                    <div class="col">
                      <label class="form-label text-muted fw-bold small">МІСТО</label>
                          <input type="text" class="form-control bg-light form-control-lg" id="inputCity"
autocomplete="off">
                    </div>
                    <div class="col">
                      <label class="form-label text-muted fw-bold small">ТЕЛЕФОН</label>
                         <input type="text" class="form-control bg-light form-control-lg" id="inputPhone"
autocomplete="off">
                    </div>
                  </div>
                </form>

                <div class="d-flex gap-3 mt-5">
                     <button class="btn btn-outline-dark px-4 py-2 flex-grow-1" id="btnManual">⌨ GUI
(Вручну)</button>
                    <button class="btn btn-outline-danger px-4 py-2 flex-grow-1" id="btnVoice">🎙 VUI
(Голос)</button>
                      <button class="btn btn-primary px-4 py-2 flex-grow-1" id="btnSave" disabled>✔
ЗАВЕРШИТИ</button>
                </div>

                <div class="mt-3 p-3 bg-dark text-info rounded voice-log shadow-inner" id="logArea">
                  Оберіть сценарій та метод для початку...
                </div>
              </div>
            </div>
          </div>

          <div class="col-md-4">
            <div class="card h-100 shadow-sm">
                    <div class="card-header bg-white text-primary fw-bold py-3">Ефективність: Час та
Точність</div>
              <div class="card-body d-flex align-items-center justify-content-center">
                <canvas id="timeChart"></canvas>
              </div>
            </div>
```

```
      </div>
    </div>

    <div class="card shadow-sm mt-2 mb-5">
      <div class="card-header bg-white d-flex justify-content-between align-items-center py-3">
        <span class="text-primary fw-bold">Детальний журнал експериментів</span>
        <button class="btn btn-success btn-sm px-3" id="btnExport">📄 Експорт CSV</button>
      </div>
      <div class="card-body p-0">
        <div class="table-responsive">
          <table class="table table-hover table-striped mb-0 align-middle">
            <thead class="table-light">
              <tr>
                <th>ID / Сценарій</th>
                <th>Метод</th>
                <th>Час (сек)</th>
                <th>Точність (%)</th>
                <th>Введені дані</th>
              </tr>
            </thead>
            <tbody id="resultsTableBody"></tbody>
          </table>
        </div>
      </div>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="voice-engine.js"></script>
    <script src="app.js"></script>
</body>
</html>
```

Текст програми файлу task_config.json


```
[
  {
    "id": 1,
    "taskName": "Сценарій 1: Базовий",
    "targets": {
      "name": "Зінаїда",
      "city": "Прага",
      "phone": "0670245112"
    },
    "description": "Стандартні імена та назви, звичні для розпізнавання."
  },
  {
    "id": 2,
    "taskName": "Сценарій 2: Складна географія",
    "targets": {
      "name": "В'ячеслав",
      "city": "Івано-Франківськ",
      "phone": "0503332211"
    },
    "description": "Перевірка дефісних назв та апострофів у іменах."
  },
  {
    "id": 3,
    "taskName": "Сценарій 3: Фонетичні колізії",
    "targets": {
      "name": "Олеся",
      "city": "Одеса",
      "phone": "0981110022"
```

```
    },
      "description": "Слова 'Олеся' та 'Одеса' звучать схоже. Тест на чіткість дикції."
  }
]
```

Текст програми файлу voice-engine.js

```
class VoiceAssistant {
  constructor() {
    console.log("--> Ініціалізація VoiceAssistant...");

    // Перевірка підтримки браузера
            const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
    if (!SpeechRecognition) {
        alert("Ваш браузер не підтримує Web Speech API. Будь ласка, використовуйте Google Chrome.");
      return;
    }

    this.recognition = new SpeechRecognition();
    this.recognition.lang = 'uk-UA'; // Українська мова
    this.recognition.interimResults = false;
    this.recognition.maxAlternatives = 1;

    // Отримуємо доступ до полів
    this.inputName = document.getElementById('inputName');
    this.inputCity = document.getElementById('inputCity');
    this.inputPhone = document.getElementById('inputPhone');
```

```
    this.logArea = document.getElementById('logArea');

    this.setupListeners();
}

start() {
    try {
        this.recognition.start();
        console.log("--> Мікрофон увімкнено");
    } catch (e) {
        console.error("Помилка запуску:", e);
    }
}

stop() {
    this.recognition.stop();
}

setupListeners() {
    this.recognition.onresult = (event) => {
        let transcript = event.results[0][0].transcript.trim();

        // Логування
        if (this.logArea) {
            this.logArea.innerText = `Почув: "${transcript}"`;
            this.logArea.style.color = "#00ff00";
        }
        console.log("Оригінал:", transcript);
```

```
        // КРОК 1: Конвертуємо слова-цифри ("нуль", "один") в реальні цифри ("0", "1")
        transcript = this.convertWordsToDigits(transcript);
        console.log("Після конвертації цифр:", transcript);

        // КРОК 2: Шукаємо номер телефону в кінці рядка
        // Регулярний вираз шукає групу цифр/пробілів/дефісів у кінці
        const phoneMatch = transcript.match(/(\d[\d\s-]{5,}\d)$/);

        let textPart = transcript; // Частина тексту без телефону (Ім'я + Місто)

        if (phoneMatch) {
            // Якщо знайшли телефон
            let rawPhone = phoneMatch[0];
            let cleanPhone = rawPhone.replace(/\D/g, ''); // Залишаємо тільки чисті цифри

            this.inputPhone.value = cleanPhone;

            // Вирізаємо телефон з основного тексту
            textPart = transcript.replace(rawPhone, '').trim();
        }

        // КРОК 3: Розбираємо те, що залишилось (Ім'я та Місто)
        const parts = textPart.split(/\s+/); // Розбиваємо по пробілах

        if (parts.length >= 1) {
            // Перше слово - це Ім'я
            this.inputName.value = this.capitalize(parts[0]);
```

```
            // Все інше (якщо є) - це Місто
            if (parts.length > 1) {
                const cityParts = parts.slice(1);
                let cityString = cityParts.join(' ');

                // Спеціальний фікс для Івано-Франківська
                // (якщо розпізнало як "Івано Франківськ" або "івано-франківськ")
                 if (cityString.toLowerCase().replace('-', ' ').includes('івано франківськ'))
{
                    cityString = "Івано-Франківськ";
                } else {
                    cityString = this.capitalize(cityString);
                }

                this.inputCity.value = cityString;
            }
        }
    };

    this.recognition.onerror = (event) => {
        console.error("Помилка розпізнавання:", event.error);
        if (this.logArea) {
            this.logArea.innerText = `Помилка: ${event.error}`;
            this.logArea.style.color = "red";
        }
    };
  }

  // --- ФУНКЦІЯ: Перетворення слів на цифри ---
  convertWordsToDigits(text) {
```

```
        const map = {
            'нуль': '0', 'один': '1', 'два': '2', 'три': '3', 'чотири': '4',
            'п\'ять': '5', 'шість': '6', 'сім': '7', 'вісім': '8', 'дев\'ять': '9',
            'плюс': '+'
        };

        return text.split(/[\s]+/).map(word => {
            const lower = word.toLowerCase();
            // Якщо слово є в словнику цифр - міняємо, якщо ні - лишаємо як є
            return map[lower] || word;
        }).join(' ');
    }

    // --- ФУНКЦІЯ: Перша літера велика ---
    capitalize(string) {
        if (!string) return '';
        return string.split(' ')
            .map(s => s.charAt(0).toUpperCase() + s.slice(1).toLowerCase())
            .join(' ');
    }

    // --- АЛГОРИТМ ЛЕВЕНШТЕЙНА (Для розрахунку точності) ---
    calculateSimilarity(s1, s2) {
        if (!s1 || !s2) return 0;
        let longer = s1, shorter = s2;
        if (s1.length < s2.length) { longer = s2; shorter = s1; }
        if (longer.length === 0) return 1.0;
                return (longer.length - this.editDistance(longer, shorter)) /
parseFloat(longer.length) * 100;
    }
```

```
    editDistance(s1, s2) {
        s1 = s1.toLowerCase(); s2 = s2.toLowerCase();
        let costs = new Array();
        for (let i = 0; i <= s1.length; i++) {
            let lastValue = i;
            for (let j = 0; j <= s2.length; j++) {
                if (i == 0) costs[j] = j;
                else {
                    if (j > 0) {
                        let newValue = costs[j - 1];
                        if (s1.charAt(i - 1) != s2.charAt(j - 1))
                            newValue = Math.min(Math.min(newValue, lastValue), costs[j]) +
1;
                        costs[j - 1] = lastValue;
                        lastValue = newValue;
                    }
                }
            }
            if (i > 0) costs[s2.length] = lastValue;
        }
        return costs[s2.length];
    }
}
```

Текст програми файлу app.js

```
// Змінні стану
let startTime = 0;
let currentMethod = '';
let results = [];
let chartInstance = null;
let timerInterval = null;
let allTasks = [];

// Поточні налаштування
let currentTarget = { name: "", city: "", phone: "" };
let currentTaskName = "";

// Елементи DOM
const taskSelect = document.getElementById('taskSelect');
const btnManual = document.getElementById('btnManual');
const btnVoice = document.getElementById('btnVoice');
const btnSave = document.getElementById('btnSave');
const btnExport = document.getElementById('btnExport');
const statusText = document.getElementById('statusText');
const logArea = document.getElementById('logArea');
const tableBody = document.getElementById('resultsTableBody');
const liveTimer = document.getElementById('liveTimer');
const countdownOverlay = document.getElementById('countdownOverlay');
const countdownText = document.getElementById('countdownText');

// --- 1. ЗАВАНТАЖЕННЯ КОНФІГУРАЦІЇ ---
async function loadConfig() {
    try {
        const response = await fetch('/api/config');
```

```
        if (!response.ok) throw new Error('Failed to load config');

        const data = await response.json();
        if (Array.isArray(data)) {
            allTasks = data;
        } else {
            allTasks = [data];
        }
        populateTaskSelector();
        if (allTasks.length > 0) setTask(0);
    } catch (error) {
        console.error("Помилка конфігу:", error);
        logArea.innerText = "Помилка завантаження config.json!";
        logArea.style.color = "red";
    }
}

function populateTaskSelector() {
    taskSelect.innerHTML = '';
    allTasks.forEach((task, index) => {
        const option = document.createElement('option');
        option.value = index;
        option.text = task.taskName;
        taskSelect.appendChild(option);
    });
}

function setTask(index) {
    const task = allTasks[index];
    currentTarget = task.targets;
```

```
    currentTaskName = task.taskName;

    document.getElementById('taskNameDisplay').innerText = task.taskName;
    document.getElementById('taskDescDisplay').innerText = task.description || "";
    document.getElementById('refName').innerText = task.targets.name;
    document.getElementById('refCity').innerText = task.targets.city;
    document.getElementById('refPhone').innerText = task.targets.phone;
}

taskSelect.addEventListener('change', (e) => {
    setTask(e.target.value);
    document.getElementById('inputName').value = '';
    document.getElementById('inputCity').value = '';
    document.getElementById('inputPhone').value = '';
    logArea.innerText = `Сценарій змінено на "${currentTaskName}".`;
    logArea.style.color = "#00cec9";
});

// --- 2. ЗАВАНТАЖЕННЯ ІСТОРІЇ ---
async function loadDataFromServer() {
    try {
        const response = await fetch('/api/results');
        const data = await response.json();
        results = data;
        results.forEach(res => addTableRow(res));
        updateStats();
        updateChart();
    } catch (error) {
        console.error('Помилка даних:', error);
    }
```

```
}

// --- ТАЙМЕР ---
function updateTimerDisplay() {
    const currentTime = new Date();
    const timeDiff = (currentTime - startTime) / 1000;
    liveTimer.innerText = timeDiff.toFixed(2);
}

function startCountdown(method) {
    if (!currentTarget.name) { alert("Завдання не обрано!"); return; }
    countdownOverlay.style.display = 'flex';
    let count = 3;
    countdownText.innerText = count;
    const interval = setInterval(() => {
        count--;
        if (count > 0) countdownText.innerText = count;
        else if (count === 0) {
            countdownText.innerText = "ПОЧАЛИ!";
            countdownText.style.color = "#00ff00";
        } else {
            clearInterval(interval);
            countdownOverlay.style.display = 'none';
            startActualTest(method);
        }
    }, 1000);
}

function startActualTest(method) {
    currentMethod = method;
```

```
    startTime = new Date();

    document.getElementById('inputName').value = '';
    document.getElementById('inputCity').value = '';
    document.getElementById('inputPhone').value = '';

    liveTimer.innerText = "00.00";
    liveTimer.classList.remove("text-muted", "text-success");
    liveTimer.classList.add("text-primary");

    if (timerInterval) clearInterval(timerInterval);
    timerInterval = setInterval(updateTimerDisplay, 50);

    btnSave.disabled = false;
    btnManual.disabled = true;
    btnVoice.disabled = true;
    taskSelect.disabled = true;

    if (method === 'manual') document.getElementById('inputName').focus();
    if (method === 'voice' && typeof VoiceAssistant !== 'undefined') {
        const assistant = new VoiceAssistant();
        assistant.start();
    }
    statusText.innerText = method === 'voice' ? 'Запис...' : 'Введення...';
    logArea.innerText = `Метод: ${method === 'voice' ? 'Голос' : 'Клавіатура'}`;
}

async function finishTest() {
    if (startTime === 0) return;
    clearInterval(timerInterval);
```

```
    liveTimer.classList.remove("text-primary");
    liveTimer.classList.add("text-success");
    btnSave.disabled = true;
    btnManual.disabled = false;
    btnVoice.disabled = false;
    taskSelect.disabled = false;

    const endTime = new Date();
    const duration = (endTime - startTime) / 1000;
    const name = document.getElementById('inputName').value.trim();
    const city = document.getElementById('inputCity').value.trim();
    const phone = document.getElementById('inputPhone').value.trim();

     const userInput = `${name} ${city} ${phone}`.toLowerCase().replace(/\s+/g, ' ').trim();
          const targetString = `${currentTarget.name} ${currentTarget.city} ${currentTarget.phone}`.toLowerCase().replace(/\s+/g, ' ').trim();

    let accuracy = 0;
    if (userInput.length === 0) accuracy = 0;
    else if (typeof VoiceAssistant !== 'undefined') {
        const assistant = new VoiceAssistant();
        accuracy = assistant.calculateSimilarity(targetString, userInput);
    } else accuracy = (userInput === targetString) ? 100 : 0;

    accuracy = Math.round(accuracy);

    const newResult = {
        id: results.length + 1,
        task: currentTaskName,
```

```
        method: currentMethod,
        time: duration,
        accuracy: accuracy,
        data: userInput || "(пусто)",
        date: new Date().toLocaleString()
    };

    results.push(newResult);
    addTableRow(newResult);
    updateStats();
    updateChart();

    try {
        await fetch('/api/results', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(newResult)
        });
        logArea.innerText = `Збережено! Час: ${duration.toFixed(2)} с.`;
    } catch (error) { console.error('Помилка:', error); }

    startTime = 0;
    statusText.innerText = 'Готовий';
}

function exportToCSV() {
    if (results.length === 0) { alert("Немає даних!"); return; }
    const BOM = "\uFEFF";
        let csvContent = BOM + "ID;Дата;Сценарій;Метод;Час (сек);Точність (%);Дані\n";
```

```
    results.forEach(row => {
        const methodNice = row.method === 'voice' ? 'Голос' : 'Ручне';
        const taskNice = (row.task || "Unknown").replace(/;/g, ',');
                                                    csvContent +=
`${row.id};${row.date};${taskNice};${methodNice};${row.time.toFixed(2).replace('
.', ',')};${row.accuracy};${row.data}\n`;
    });
    const blob = new Blob([csvContent], { type: 'text/csv;charset=utf-8;' });
    const link = document.createElement("a");
    const url = URL.createObjectURL(blob);
    link.setAttribute("href", url);
    link.setAttribute("download", "results_export.csv");
    document.body.appendChild(link);
    link.click();
    document.body.removeChild(link);
}

function addTableRow(res) {
    const row = document.createElement('tr');
    row.innerHTML = `
        <td>
            <strong>${res.id}</strong>
            <div class="text-muted small">${res.task || ''}</div>
        </td>
            <td><span class="badge ${res.method === 'voice' ? 'bg-danger' : 'bg-
dark'}">${res.method === 'voice' ? 'VUI' : 'GUI'}</span></td>
        <td>${res.time.toFixed(2)}</td>
        <td>
            <span class="${res.accuracy >= 80 ? 'text-success fw-bold' : res.accuracy < 50
? 'text-danger' : 'text-warning'}">
```

```
            ${res.accuracy}%
          </span>
        </td>
        <td class="text-muted small">${res.data}</td>
    `;
    tableBody.prepend(row);
}

function updateStats() {
    document.getElementById('totalTests').innerText = results.length;
    const voiceTests = results.filter(r => r.method === 'voice');
    if (voiceTests.length > 0) {
          const avgTime = voiceTests.reduce((acc, curr) => acc + curr.time, 0) / voiceTests.length;
        document.getElementById('avgTimeVoice').innerText = avgTime.toFixed(2) + 'c';
         const avgAcc = voiceTests.reduce((acc, curr) => acc + curr.accuracy, 0) / voiceTests.length;
        document.getElementById('avgAccuracy').innerText = Math.round(avgAcc) + '%';
    }
}

// ---ГРАФІК ---
function updateChart() {
    const ctx = document.getElementById('timeChart').getContext('2d');

    const voiceTests = results.filter(r => r.method === 'voice');
    const manualTests = results.filter(r => r.method === 'manual');
```

```
    const calcAvg = (arr, prop) => arr.length ? arr.reduce((a, b) => a + b[prop], 0) / arr.length : 0;

    const avgVoiceTime = calcAvg(voiceTests, 'time');
    const avgManualTime = calcAvg(manualTests, 'time');
    const avgVoiceAcc = calcAvg(voiceTests, 'accuracy');
    const avgManualAcc = calcAvg(manualTests, 'accuracy');

    if (chartInstance) chartInstance.destroy();

    chartInstance = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: ['Голос (VUI)', 'Вручну (GUI)'],
            datasets: [
                {
                    label: 'Час виконання (с)',
                    data: [avgVoiceTime, avgManualTime],
                    backgroundColor: [
                        'rgba(255, 99, 132, 0.5)',
                        'rgba(75, 192, 192, 0.5)'
                    ],
                    borderColor: [
                        'rgba(255, 99, 132, 1)',
                        'rgba(75, 192, 192, 1)'
                    ],
                    borderWidth: 1,
                    borderRadius: 5,
                    barPercentage: 0.5,
                    order: 2,
```

```
            yAxisID: 'y'
        },
        {
            label: 'Точність (%)',
            data: [avgVoiceAcc, avgManualAcc],
            type: 'line',
            borderColor: '#343a40',
            backgroundColor: '#343a40',
            borderWidth: 2,
            pointBackgroundColor: '#fff',
            pointBorderColor: '#343a40',
            pointRadius: 5,
            pointHoverRadius: 7,
            tension: 0.4,
            order: 1,
            yAxisID: 'y1'
        }
    ]
},
options: {
    responsive: true,
    maintainAspectRatio: false,
    plugins: {
        legend: {
            position: 'top',
            align: 'end',
            labels: { usePointStyle: true, boxWidth: 8 }
        },
        tooltip: {
            backgroundColor: 'rgba(255, 255, 255, 0.9)',
```

```
            titleColor: '#000',
            bodyColor: '#000',
            borderColor: '#ddd',
            borderWidth: 1,
            padding: 10,
            displayColors: true
        }
    },
    scales: {
        y: {
            type: 'linear',
            position: 'left',
            beginAtZero: true,
            grid: { color: '#f0f0f0' },
            title: { display: true, text: 'Час (сек)', color: '#666' }
        },
        y1: {
            type: 'linear',
            position: 'right',
            beginAtZero: true,
            max: 100,
            grid: { display: false },
            title: { display: true, text: 'Точність (%)', color: '#666' }
        },
        x: {
            grid: { display: false }
        }
    }
  }
});
```

```
}

// Події
btnManual.addEventListener('click', () => startCountdown('manual'));
btnVoice.addEventListener('click', () => startCountdown('voice'));
btnSave.addEventListener('click', finishTest);
btnExport.addEventListener('click', exportToCSV);

// Старт
loadConfig();
loadDataFromServer();
```

Текст програми файлу styles.css

```
/* Професійний вигляд карток */
.stat-card {
    transition: transform 0.2s;
    border: none;
    box-shadow: 0 0.15rem 1.75rem 0 rgba(58, 59, 69, 0.15);
}
.stat-card:hover {
    transform: translateY(-5px);
}
.border-left-primary { border-left: 4px solid #4e73df !important; }
.border-left-success { border-left: 4px solid #1cc88a !important; }
.border-left-info { border-left: 4px solid #36b9cc !important; }
.border-left-warning { border-left: 4px solid #f6c23e !important; }

/* Анімація мікрофона */
.mic-listening {
    background-color: #e74a3b !important;
```

```
    color: white !important;
    animation: pulse-red 2s infinite;
}

@keyframes pulse-red {
    0% { transform: scale(0.95); box-shadow: 0 0 0 0 rgba(231, 74, 59, 0.7); }
    70% { transform: scale(1); box-shadow: 0 0 0 10px rgba(231, 74, 59, 0); }
    100% { transform: scale(0.95); box-shadow: 0 0 0 0 rgba(231, 74, 59, 0); }
}

/* Лог та дрібниці */
.voice-log {
    background: #2d3436;
    color: #00cec9;
    font-family: 'Consolas', monospace;
    padding: 10px;
    border-radius: 5px;
    font-size: 0.85rem;
    min-height: 40px;
}
.match-score {
    display: block;
    font-size: 0.75rem;
    margin-top: 2px;
}
```

Текст програми файлу server.js

```
const express = require('express');
const fs = require('fs');
const path = require('path');
```

```
const app = express();
const PORT = 3000;

app.use(express.json());
app.use(express.static('public'));

// Шляхи до файлів
const DATA_FILE = path.join(__dirname, 'data', 'results.json');
const CONFIG_FILE = path.join(__dirname, 'config', 'task_config.json');

// --- API 1: Отримати налаштування (НОВЕ!) ---
app.get('/api/config', (req, res) => {
    if (!fs.existsSync(CONFIG_FILE)) {
        return res.status(404).json({ error: 'Config file not found' });
    }
    fs.readFile(CONFIG_FILE, 'utf8', (err, data) => {
        if (err) return res.status(500).json({ error: 'Error reading config' });
        // Віддаємо JSON з файлу
        res.json(JSON.parse(data));
    });
});

// --- API 2: Отримати результати ---
app.get('/api/results', (req, res) => {
    if (!fs.existsSync(DATA_FILE)) return res.json([]);
    fs.readFile(DATA_FILE, 'utf8', (err, data) => {
        if (err) return res.status(500).json({ error: 'Error reading data' });
        res.json(JSON.parse(data || '[]'));
    });
});
```

```
// --- API 3: Зберегти результат ---
app.post('/api/results', (req, res) => {
    const newResult = req.body;
    let currentData = [];

    if (fs.existsSync(DATA_FILE)) {
        const fileContent = fs.readFileSync(DATA_FILE, 'utf8');
        try { currentData = JSON.parse(fileContent || '[]'); } catch (e) {}
    }

    currentData.push(newResult);

    fs.writeFile(DATA_FILE, JSON.stringify(currentData, null, 2), (err) => {
        if (err) return res.status(500).json({ error: 'Error writing data' });
        res.json({ success: true });
    });
});

// Запуск
app.listen(PORT, () => {
    console.log(`Server running at http://localhost:${PORT}`);

    // Створюємо папки, якщо їх немає
    if (!fs.existsSync(path.join(__dirname, 'data'))) fs.mkdirSync(path.join(__dirname, 'data'));
    if (!fs.existsSync(path.join(__dirname, 'config'))) fs.mkdirSync(path.join(__dirname, 'config'));
});
```

ДОДАТОК В

Керівництво користувача

ЗАТВЕРДЖУЮ

Перший проректор Українського

державного університету науки і технологій

Анатолій РАДКЕВИЧ

ВЕБ-ДОДАТОК «VUI Research System»

Керівництво користувача

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.9519-01 І3 01-ЛЗ

Завідувач кафедри КІТ

_____ Вадим ГОРЯЧКІН

Керівник розробки

_____ Світлана ВОЛКОВА

Виконавець

_____ Ілля ЦИПА

Нормоконтролер

_____ Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.01556-01 І3 01

ВЕБ-ДОДАТОК «VUI Research System»

Керівництво користувача

44165850.01556-01 І3 01

Листів 15

2025

АННОТАЦІЯ

Документ 44165850.9519-01 І3 01-ЛЗ «Програмне забезпечення для дослідження ефективності голосових інтерфейсів "VUI Research System". Керівництво користувача» входить до складу програмної документації на веб-додаток для порівняльного аналізу методів введення даних.

У даному документі представлено керівництво користувача, опис функціональних можливостей та сценаріїв роботи з системою.

ЗМІСТ

ВВЕДЕННЯ

Даний веб-додаток призначений для полегшення взаємодії користувача з веб-інтерфейсами за допомогою голосового керування та автоматизованих підказок. Метою розробки є підвищення зручності, швидкості та ефективності виконання користувацьких дій шляхом використання голосового помічника.

Веб-додаток VUI Research System надає можливість керування основними функціями веб-інтерфейсу за допомогою голосових команд, а також забезпечує голосовий та текстовий зворотний зв’язок. Застосунок орієнтований як на окремих користувачів, так і на групи користувачів, для яких важливі доступність та зменшення навантаження під час роботи з веб-сервісами.

Додаток має набір інструментів для обробки та аналізу голосових команд, включаючи механізми розпізнавання мовлення, інтерпретації намірів користувача та виконання відповідних дій у веб-додатку. Це дозволяє досліджувати ефективність голосового інтерфейсу у порівнянні з традиційними способами керування.

Інтерфейс веб-додатку реалізовано у вигляді веб-сторінок із підтримкою голосової взаємодії. Для використання додатку необхідний доступ до мережі Інтернет та пристрій з сучасним веб-браузером і мікрофоном. Наявність спеціальних технічних або професійних навичок у користувача не є обов’язковою, оскільки всі функції додатку реалізовані у зрозумілому та інтуїтивному вигляді.

44165850.01556-01 І3 01

1. УМОВИ ЗАСТОСУВАННЯ

1.1. Вимоги до складу і параметрів технічних засобів

Вимоги при доступі з мобільного пристрою(смартфону):

- Процесор – чотириядерний процесор з тактовою частотою не менше 1,8 ГГц;
- Оперативна пам’ять - не менше 2 ГБ;
- Підтримка мереж - Підтримка 4g та Wifi мереж.
- Наявність вбудованого або зовнішнього мікрофона;

Вимоги при доступі з персонального комп’ютера:

- Процесор - Intel Core I3 2nd gen;
- Оперативна пам’ять - не менше 4 гб;
- Підтримка мереж - підтримка Ethernet/ WiFi.

1.2. Вимоги до інформаційного та програмного середовища

Вимоги до програмного середовища при доступі з мобільного пристрою:

- Операційна система - Android 8 Nougat або вище, IOS 12.0 та вище;
- Браузер з підтримкою Javascript та localstorage.

Вимоги до програмного середовища при доступі з ПК:

- Операційна система - Windows 10, MacOS Mojave чи Linux;
- Браузер з підтримкою Javascript та localstorage.

2. ПІДГОТОВКА ДО ВИКОРИСТАННЯ

Встановлення додаткового програмного забезпечення перед початком використання веб-додатку не потребується. Програмний продукт поширюється у вигляді веб-додатку, доступ до якого здійснюється за допомогою веб-браузера. Носій даних з програмою містить керівництво користувача та текстовий файл формату txt з назвою *app.txt*, у якому наведено URL-адресу веб-додатку VUI Research System. Перед початком роботи користувачу рекомендується ознайомитися з даним керівництвом користувача. Для початку роботи з веб-додатком необхідно відкрити підтримуваний веб-браузер та ввести URL-адресу застосунку в адресному або пошуковому рядку браузера. Також можливе копіювання URL-адреси з файлу *app.txt* з подальшим вставленням її в адресний рядок браузера. Після переходу за посиланням користувачеві необхідно надати дозвіл на використання мікрофона для забезпечення коректної роботи голосового помічника.

3. ОПИС ЕКРАНІВ ТА ОПЕРАЦІЙ

3.1. Модуль автентифікації користувача (Login)

Призначення: Модуль відповідає за безпечну ідентифікацію та перевірку прав доступу користувача до системи.

Функціональні можливості:

- Перевірка облікових даних: Зіставлення введеного логіна/Email та хешованого пароля з даними в базі.
- Управління сесіями: Створення унікальної сесійної токен-ключової пари (token) для авторизованого користувача та її зберігання.
- Відновлення пароля: Забезпечення механізму скидання пароля через Email або інший канал зв'язку.

Сценарій взаємодії:

Користувач (К) вводить логін/Email та пароль $\rightarrow$ Система (С) перевіряє дані $\rightarrow$ Успіх: С перенаправляє К на Основний функціональний модуль (Dashboard) та створює сесію; Помилка: С виводить повідомлення про невірні дані

3.2. Модуль реєстрації користувача (Signup)

Призначення: Модуль, який дозволяє новим користувачам створити свій обліковий запис у системі.

Функціональні можливості:

- Валідація вхідних даних: Перевірка унікальності Email, відповідності пароля вимогам безпеки.
- Створення запису: Збереження нових облікових даних у базі, хешування пароля.
- Підтвердження Email (опційно): Надсилання листа для активації облікового запису з метою верифікації.

Сценарій взаємодії:

К вводить необхідні дані $\rightarrow$ С валідує унікальність та коректність даних $\rightarrow$ Успіх: С створює обліковий запис, автоматично авторизує К та перенаправляє на Dashboard.

3.3. функціональний модуль веб-додатку (Dashboard / Core UI)

Призначення: Центральний модуль, який містить основну логіку роботи системи, відображає ключову інформацію та виступає інтеграційною платформою для голосового помічника.

Функціональні можливості:

- Відображення статистики/даних: Надання користувачеві візуальних та текстових звітів про його діяльність (залежно від спеціалізації вашого додатку).
- Навігація: Забезпечення переходу між усіма основними екранами (Налаштування, Аналітика, тощо).
- Ініціація голосового помічника: Запуск Модуля голосового помічника (3.4) через інтерфейс.

3.4. Модуль голосового помічника (VUI Research System Core Interface)

Призначення: Головний інтерфейс для захоплення звуку та передачі його для розпізнавання, забезпечуючи двосторонній зв'язок між користувачем і системою.

Функціональні можливості:

- Захоплення аудіо: Використання Web Audio API для запису голосу через мікрофон клієнта.
- Візуальний фідбек: Відображення анімації (наприклад, візуалізатора звуку), що підтверджує, що помічник "слухає".
- Передача даних: Надсилання записаного аудіо-потоку до Модуля обробки голосових команд (3.6) для розпізнавання.
- Озвучення відповіді (TTS): Отримання текстової відповіді від системи та її озвучення (Text-to-Speech) для користувача.

Сценарій взаємодії:

К натискає іконку мікрофона $\rightarrow$ Модуль 3.4 активується, записує звук $\rightarrow$ К вимовляє команду $\rightarrow$ Модуль 3.4 передає аудіо до 3.6

3.5. Модуль налаштування голосового керування (Customization Module)

Призначення: Надання користувачеві інструментів для персоналізації та оптимізації роботи голосового помічника під індивідуальні потреби.

Функціональні можливості:

- Конфігурація Hotword: Можливість увімкнення/вимкнення активації помічника "гарячим словом" ("VoxWeb", "Асистент").
- Вибір мови: Налаштування мови розпізнавання (наприклад, українська, англійська).
- Налаштування швидкості TTS: Регулювання швидкості озвучення відповідей системи.
- Калібрування мікрофона: Інструменти для перевірки та налаштування рівня чутливості мікрофона.

Сценарій взаємодії:

К заходить у розділ налаштувань $\rightarrow$ К змінює параметр (наприклад, вибирає іншу мову) $\rightarrow$ С зберігає зміни в профілі.

3.6. Модуль обробки голосових команд (NLP/STT Processor)

Призначення: Ключовий модуль, відповідальний за перетворення голосу в текст (STT) та інтерпретацію намірів користувача (NLP) для виконання дій.

Функціональні можливості:

1. Розпізнавання мовлення (STT): Перетворення аудіо-потоку (отриманого від 3.4) на текстовий рядок.
2. Обробка природної мови (NLP):

    - Визначення наміру (Intent Recognition): Визначення мети команди ("Показати статистику", "Створити новий запис").
    - Виділення сутностей (Entity Extraction): Вилучення ключових параметрів з команди (наприклад, "за вчора", "на суму 100 грн").
3. Виконання команди: Передача структурованого наміру та сутностей до Основного функціонального модуля (3.3) для виконання дії.

Сценарій взаємодії:

Модуль 3.6 отримує аудіо $\rightarrow$ 3.6 розпізнає текст ("Покажи статистику за тиждень") $\rightarrow$ 3.6 визначає Intent= "Отримати статистику", Entity= "тиждень" $\rightarrow$ 3.6 передає команду до 3.3.

3.7. Модуль статистики та аналізу використання

Призначення: Модуль для збору, агрегації та візуалізації даних про ефективність використання VUI Research System та дії користувача.

Функціональні можливості:

- Логування команд: Запис усіх голосових команд (оригінальний аудіо, розпізнаний текст, успішність виконання).
- Аналіз ефективності: Розрахунок показників:
  - Кількість успішно виконаних команд.
  - Відсоток помилок розпізнавання.
  - Найбільш популярні команди (Intents).
- Візуалізація: Побудова графіків та звітів для адміністраторів (або для самого користувача), що відображають патерни взаємодії.

Сценарій взаємодії:

С щоденно агрегує дані про використання $\rightarrow$ К або Адміністратор запитує звіт $\rightarrow$ Модуль 3.7 генерує звіт про "Топ-5 помилок розпізнавання" або "Час, зекономлений за допомогою голосових команд".

4. АВАРІЙНІ СИТУАЦІЇ

Система VUI Research System використовує стандартизовану структуру повідомлень для забезпечення зрозумілої та швидкої комунікації з користувачем, незалежно від того, чи повідомлення відображається в інтерфейсі (UI) чи озвучується голосовим помічником. Повідомлення поділяються на три основні категорії: системні, повідомлення про успіх та повідомлення про помилки. Системні повідомлення відображають поточний стан програми та використовуються для інформування користувача про зміни статусу або необхідність його дій. Прикладами є "Очікування підключення до сервера розпізнавання", " VUI Research System активний, чекаю на команду" або "Сесія буде завершена через 5 хвилин неактивності". Повідомлення про успіх підтверджують, що дія користувача або команда, віддана голосовому помічнику, була успішно виконана. Ці повідомлення зазвичай короткі та чіткі. Наприклад: "Реєстрація успішно завершена. Вітаємо!", або у випадку голосової команди: "Команда 'Показати статистику' успішно виконана" чи "Новий запис додано до вашого профілю". Повідомлення про помилки та попередження інформують користувача про виникнення проблеми та, за можливості, надають інструкції щодо її усунення. Ці повідомлення є критично важливими, особливо у взаємодії з голосовим помічником. Помилки автентифікації включають "Невірний логін або пароль" чи "Користувача з таким Email не знайдено". Помилки, пов'язані з голосовим керуванням, можуть бути наступними: "Доступ до мікрофона заборонено. Будь ласка, перевірте налаштування браузера", "Команда не розпізнана. Спробуйте сформулювати інакше" – це типове повідомлення від Модуля обробки команд (3.6), коли не вдалося визначити намір (Intent). Також можуть бути попередження валідації: "Введені дані не відповідають вимогам, наприклад, пароль повинен містити мінімум 8 символів та цифр

5. СТРУКТУРА ПОВІДОМЛЕНЬ

Текст повідомлення	Опис ситуації, в якій користувач отримує повідомлення	Рекомендовані дії
Username or password are incorrect	Виникає на екрані авторизації (Login) при введенні неправильних даних входу.	Перевірити коректність введення логіна/Email та пароля.
This field is required	Виникає при спробі відправити форму (реєстрація, додавання даних) без заповнення обов’язкового поля.	Ввести дані у відповідне поле.
Error 404. Please reconnect	Виникає при недоступності серверу або обриві мережевого з'єднання.	Перевірити інтернет-з'єднання та перезавантажити сторінку.
Please add date to your transaction	Користувач отримує у разі не введення дати при заповненні форми транзакції (або при відсутності дати в голосовій команді).	Ввести дані в запропоноване поле, або, використовуючи голосове керування, додати дату (наприклад, "за вчора", "сьогодні").
Password should have at least 1	На екрані реєстрації (Signup), у разі якщо	Ввести інший пароль, що задовольняє вимогам

special character	пароль не відповідає нормам безпеки (не містить спеціального символу).	безпеки.
Password should be at least 8 characters long	На екрані реєстрації (Signup), у разі якщо пароль, введений користувачем, коротший за 8 знаків.	Ввести інший пароль, що задовольняє вимогам безпеки (мінімум 8 символів).
Microphone access denied. Please check your settings	Специфічне для VoxWeb. Виникає, коли браузер або ОС блокує доступ до мікрофона.	Надати дозвіл на використання мікрофона в налаштуваннях браузера.
Command not recognized. Please try phrasing differently.	Специфічне для VoxWeb. Виникає, коли Модуль обробки команд (3.6) не може визначити намір (Intent) з голосової команди.	Сформулювати команду чіткіше або використовувати стандартизовані фрази.
Session expired. Please log in again.	Виникає після тривалого часу неактивності або при закінченні терміну дії сесійного токена.	Натиснути "ОК" та повторно авторизуватися.

6. РЕКОМЕНДАЦІЇ ЩОДО ЗАСВОЄННЯ

Опис контрольного прикладу (Сценарій голосової взаємодії)

Контрольний приклад демонструє основний сценарій використання веб-додатку VUI Research System, поєднуючи фінансовий функціонал з інтелектуальним голосовим керуванням.

Сценарій: Сім'я має місячний дохід у розмірі 30 000 грн і планує розподілити його, використовуючи голосові команди.

Кроки взаємодії (Голосові команди):

1. Введення доходу: Користувач активує VUI Research System і вимовляє: *"VoxWeb, додай надходження 30 тисяч гривень, категорія 'Зарплата'."*
2. Розподіл бюджету: Користувач вводить заплановані витрати голосовими командами для різних категорій, замість використання екранних форм:

   - *"Спланує 8000 гривень на 'Продукти харчування'."*

   - *"Віднеси 4000 гривень до 'Комунальних послуг'."*

   -*"Встанови ліміт на 'Збереження' у розмірі 5000 гривень."*
3. Запит статистики: Користувач запитує поточний стан бюджету: *"VoxWeb, який залишок бюджету після всіх планувань?"*

   - Очікуваний результат: Система швидко обробляє голосові команди через Модуль обробки (3.6), вводить дані та озвучує відповідь: "Залишковий баланс становить 0 гривень. Усі кошти розподілено".

3.4 (Активація помічника), 3.6 (Розпізнавання та виконання команд) і 3.3 (Відображення результатів на Dashboard).

6.2. Правила запуску та виконання

6.2.1. Доступ до веб-додатку:

Відкрийте сумісний веб-браузер (Google Chrome версії 80+, Mozilla Firefox 75+, Safari 14+) та перейдіть за посиланням на сайт VUI Research System. Переконайтеся, що ваш пристрій має робочий мікрофон.

6.2.2. Реєстрація та вхід користувача:

Створіть обліковий запис, використовуючи форму реєстрації (Signup) або увійдіть (Login). Після успішної авторизації система автоматично відкриє Основний функціональний модуль (Dashboard).

6.2.3. Активація та використання голосового помічника:

- Для ініціації голосової взаємодії натисніть на іконку мікрофона на головному екрані.

- Дочекайтеся візуального підтвердження активації (наприклад, зміна кольору іконки або пульсація) та почуйте системне сповіщення: *"Слухаю вас"*.

- Чітко вимовте команду, використовуючи природну мову.

6.2.4. Виконання голосових команд:

- Використовуйте голосові команди для виконання ключових операцій (додавання доходу/витрат, запит балансу, зміна налаштувань).

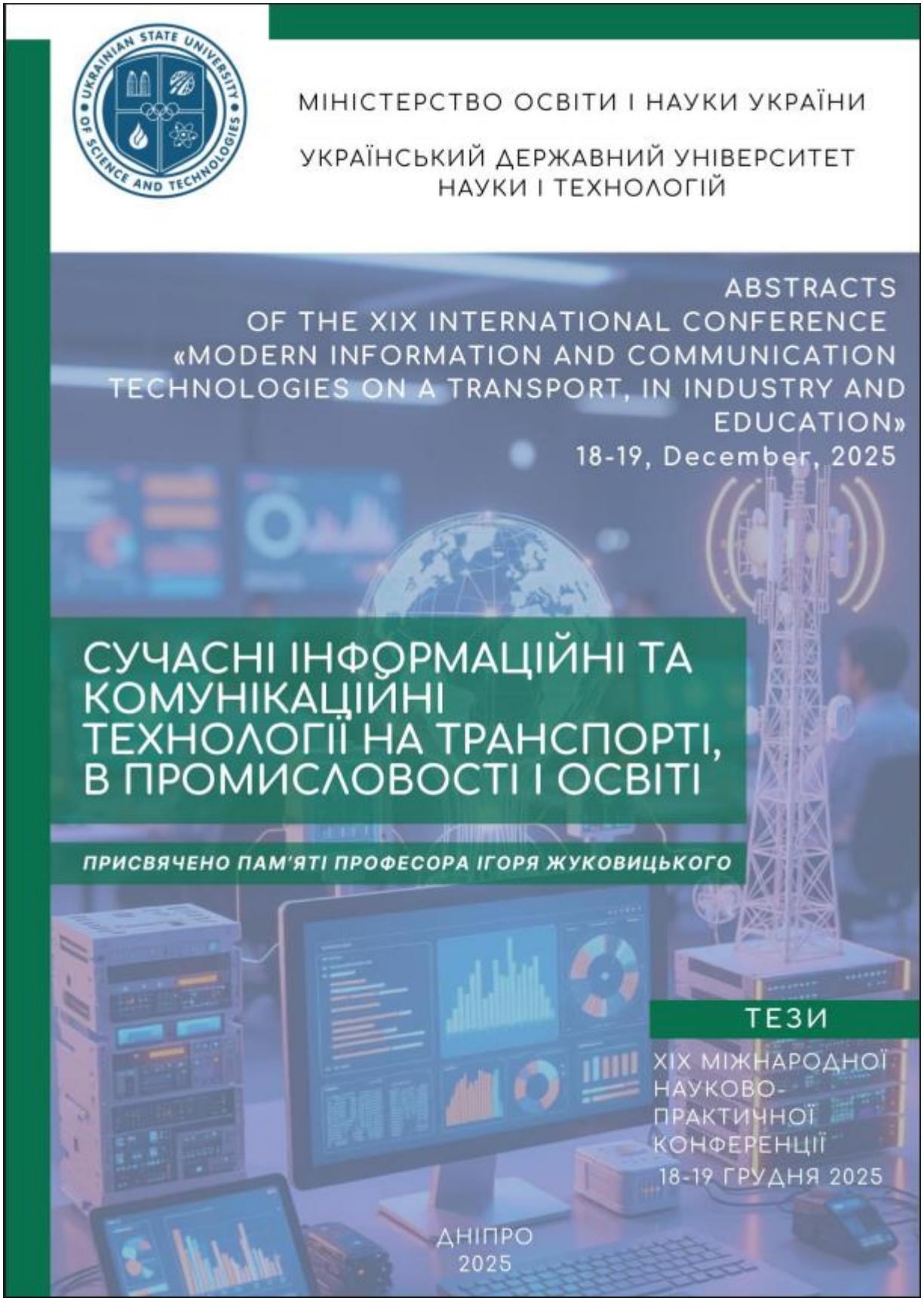
- Перевірте, як система реагує на різну артикуляцію, та переконайтеся, що Модуль обробки голосових команд (3.6) правильно інтерпретує наміри.

- У разі некоректного розпізнавання, повторіть команду чіткіше або зверніться до документації по стандартизованих фразах.

6.2.5. Аналіз ефективності голосового керування:

Регулярно оновлюйте дані (додаючи нові операції голосом) і спостерігайте, як система миттєво оновлює графіки та візуалізацію на Dashboard. Це демонструє безперебійну роботу всіх модулів у реальному часі

ДОДАТОК Г (Тези)



Дослідження ефективності використання голосового помічника у веб-додатках

Ципа І.В., Волкова С. А, Український державний університет науки і технологій: Україна

Стрімка еволюція людино-машинних інтерфейсів (HMI) зумовлює перехід від традиційних парадигм взаємодії WIMP (Window, Image, Menu, Pointer) до більш інтуїтивних SILK-інтерфейсів (Speech, Image, Language, Knowledge). В умовах постійного зростання обсягів інформації та необхідності забезпечення інклюзивності веб-ресурсів, інтеграція голосових технологій стає критично важливою вимогою для сучасних веб-додатків. Актуальність дослідження обумовлена необхідністю подолання обмежень стандартних методів навігації (клавіатура/миша) та забезпечення безконтактного керування контентом, що особливо важливо в контексті мультизадачності та доступності для користувачів з особливими потребами [3].

Метою роботи є аналіз архітектурних рішень для створення голосових веб-асистентів, порівняння їхньої ефективності з нативними аналогами та визначення перспектив інтеграції з технологіями доповненої реальності (AR).

У ході дослідження проаналізовано існуючі підходи до реалізації голосового керування. Більшість комерційних рішень (Siri, Alexa, Google Assistant) функціонують як закриті екосистеми, що ускладнює їх глибоку інтеграцію зі специфічною логікою веб-додатків. Натомість, підхід на основі відкритих веб-стандартів (Open Web Ecosystem) дозволяє створювати гнучкі рішення без прив'язки до апаратної платформи [1]. Для реалізації запропоновано використання Web Speech API та JavaScript, що забезпечує кросплатформеність та відсутність необхідності встановлення стороннього ПЗ.

Окрему увагу в роботі приділено питанням конфіденційності. Згідно з рекомендаціями European Data Protection Board, голосові дані є біометричними, тому їх обробка вимагає прозорості та мінімізації даних [4]. Запропонована архітектура дозволяє реалізувати механізм "Privacy by Design", де обробка команд відбувається лише після явної активації мікрофона користувачем, без постійного фонового прослуховування.

Перспективним напрямком розвитку є поєднання голосових інтерфейсів з технологіями доповненої реальності (AR). Такий симбіоз дозволяє створити мультимодальний інтерфейс, де голосові команди доповнюють візуальну інформацію, що накладається на реальний світ. Це відкриває нові можливості для сфер електронної комерції, освіти та телемедицини, забезпечуючи ефект "Hands-free" та підвищуючи імерсивність взаємодії [2].

Впровадження голосових асистентів на базі веб-технологій є ефективним методом покращення юзабіліті веб-додатків. Використання Web Speech API у поєднанні з JavaScript дозволяє створювати легковагові, безпечні та контекстно-залежні інтерфейси, що перевершують нативні аналоги у специфічних сценаріях веб-навігації.