НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет Інформаційних технологій

ПОГОДЖЕНО
Декан факультету
інформаційних технологій

_____ Ігор БОЛБОТ

"__" _____ 2025р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри
комп'ютерних наук

_____ Белла ГОЛУБ

"__" _____ 2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему «Розробка веб сайту для людей з вадами зору з використанням голосового введення даних»

Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми освітньо-професійна

Гарант освітньої програми
_____ к. ф-м.н., доцент_____ _____ Віктор КИРИЧЕНКО
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи
_____к. т. н., доцент._____ _____ _____Юлія БОЯРІНОВА
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Виконав _____ _____Роман Коник_____
(підпис) ((ім'я ПРІЗВИЩЕ здобувача)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
к.т.н, доцент____ _____Белла ГОЛУБ
(науковий ступінь, вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)
"_14__" _____11_____2024___року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

_Коник Роман Сергійович_____
(прізвище, ім'я, по батькові)

Спеціальність 121 Інженерія програмного забезпечення (код і найменування)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи_____
Розробка веб сайту для людей з вадами зору з використанням голосового введення даних
затверджена наказом від "_1" 11.2024р. №1963 С

Термін подання завершеної роботи на кафедру 2025. 11.14
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи: Робота базується на аналізі міжнародних стандартів доступності, сучасних інструментів та вебтехнологій. Передбачається створення створення функціонального прототипу, що забезпечує голосове керування вебресурсами та синтез мовлення без зміни коду сайту

Перелік питань, що підлягають дослідженню:

1. Аналіз стану проблеми вебдоступності у світі, оцінка відповідності чинних рішень міжнародним стандартам.
2. Дослідження сучасних технологій забезпечення доступності
3. Аналіз можливостей Web Speech API для реалізації голосової взаємодії у веб середовищі

Перелік графічного матеріалу (за потреби) _____
Дата видачі завдання "__1____"___11_____2024 р.

Керівник магістерської кваліфікаційної роботи _____ Юлія БОЯРІНОВА
(підпис) (ім'я ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ ___ Роман КОНИК
(ім'я ПРІЗВИЩЕ)

ЗМІСТ

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальна характеристика проблеми вебдоступності

У сучасній цифровій економіці вебсайти виконують роль не лише засобу представлення інформації, але й інструменту соціальної, освітньої, адміністративної та комерційної взаємодії. Доступність вебресурсів для всіх категорій користувачів є критично важливою складовою цифрової інклюзії, що закріплена у міжнародних стандартах, зокрема Web Content Accessibility Guidelines (WCAG 2.1) [1]. Під доступністю у цьому контексті розуміють можливість користування вебресурсами особами з порушеннями зору, слуху, моторики, когнітивних функцій, а також літніми людьми, які мають знижену здатність до сприйняття інформації.

За даними Всесвітньої організації охорони здоров’я, понад 2,2 млрд людей у світі мають порушення зору, з них близько 36 млн є повністю незрячими [2]. Це означає, що кожен четвертий користувач мережі може зіткнутися з бар'єрами у взаємодії з вебінтерфейсами, якщо вони не адаптовані належним чином. У звіті WebAIM за 2024 рік зазначено, що 96,3 % головних сторінок сайтів мають критичні порушення доступності, незважаючи на існування міжнародних норм [3]. Тобто, навіть після 15 років існування WCAG, більшість інтернету залишається фактично недоступною для людей з інвалідністю.

Проблема ускладнюється тим, що навіть державні електронні сервіси, освітні системи та ресурси соціального призначення не завжди відповідають вимогам WCAG. В Україні, відповідно до Постанови Кабінету Міністрів № 851 «Про цифрову доступність державних вебресурсів», усі державні портали мають бути адаптовані для людей з інвалідністю, проте фактичний рівень

реалізації цього нормативу залишається низьким [4]. Подібна ситуація спостерігається і в ЄС, де частина сайтів формально декларують відповідність стандартам, але не проходять автоматичне тестування. Це вказує на існування розриву між нормативними вимогами та реальною технічною доступністю.

Незрячі користувачі зазвичай використовують екранні зчитувачі (NVDA, JAWS, VoiceOver), однак ефективність таких систем напряму залежить від того, наскільки семантично правильно розмічений сайт: наявність alt-описів зображень, коректна ієрархія заголовків, aria-label атрибутів тощо. Якщо сайт побудований із порушенням семантики, screen reader не може коректно інтерпретувати контент, що перетворює взаємодію на хаотичний процес [5]. Таким чином, навіть за наявності сучасних assistive-технологій, доступність залишається залежною від якості верстки, що ставить людей з порушеннями зору у менш вигідне становище — це явище називають digital discrimination.

Технічні проблеми доступності поєднуються з бар'єрами користувацького досвіду. Для багатьох користувачів із порушеннями моторики або літніх людей взаємодія з інтерфейсами за допомогою «класичної» миші чи клавіатури є або складною, або неможливою. Саме тому останніми роками збільшується інтерес до голосових інтерфейсів як альтернативи традиційній навігації вебресурсами [6]. Голосове керування вже реалізоване у смартфонах, смарт-колонках, системах «розумного дому», однак у браузерах та вебсайтах такі рішення залишаються слаборозвиненими. Це вказує на технологічний розрив між можливостями голосової взаємодії та реальним її впровадженням у вебсередовищі.

Таким чином, існує структурний розрив між доступними інструментами та реальними потребами користувачів(Таблиця 1).

Таблиця 1. Категорії користувачів та проблеми

Категорія користувачів	Основний бар'єр	Типова проблема
Незрячі користувачі	Відсутність семантичної доступності	Screen reader не читає частину елементів
Люди з порушеннями моторики	Неможливість користування мишею	Немає альтернативи у вигляді голосових команд
Особи похилого віку	Низька цифрова грамотність	Складна навігація, перевантажені інтерфейси
Користувачі мобільних пристроїв	Малий екран + сенсорна навігація	Низька точність взаємодії, особливо у вебформах

Як наслідок, недостатня доступність вебресурсів є не лише технічною, але й соціальною, етичною та правовою проблемою, що стосується цифрової рівності та прав людини. Вебдоступність безпосередньо пов'язана з концепціями Universal Design, Human-Centered Computing та Digital Inclusion і стає критерієм якості цифрової трансформації суспільства. Втрата доступності — це не лише бар'єр для користувача, але й економічна втрата для бізнесу: за оцінками Forbes, сайти без доступності втрачають у середньому 15–20 % потенційної аудиторії.

1.2 Нормативно-правове забезпечення та стандарти вебдоступності

Питання доступності вебресурсів регулюється на міжнародному, регіональному та національному рівнях. У більшості країн нормативна база спирається на документи, розроблені консорціумом W3C, насамперед Web

Content Accessibility Guidelines (WCAG), які визначають вимоги до створення інклюзивного вебконтенту [7]. WCAG є де-факто глобальним стандартом, на основі якого розробляються державні закони, вимоги до сайтів органів влади, а також критерії сертифікації цифрових продуктів.

Стандарт WCAG 2.1 побудований на чотирьох принципах (POUR), згідно з якими контент повинен бути (Таблиця 2).

Таблиця 2. Принцип стандарту WCAG 2.1

Принцип	Вимога	Приклад порушення
P (Perceivable) — Сприйманий	Інформація має бути доступною для сприйняття різними каналами	Зображення без alt-тегу, текст у вигляді картинки
O (Operable) — Керований	Елементи інтерфейсу мають бути доступні без миші	Меню, яке не працює з клавіатурою
U (Understandable) — Зрозумілий	Структура і логіка навігації мають бути передбачуваними	Різні стилі кнопок для однакових дій
R (Robust) — Стійкий	Контент повинен коректно працювати з assistive-технологіями	Несемантичні елементи, приховані кнопки

Більшість державних законів не створюють власні вимоги до доступності, а адаптують WCAG, визначаючи необхідний рівень відповідності (A, AA або AAA) та сферу обов’язкового застосування (державні органи, комунальні установи, приватні сервіси).

Юрисдикція	Нормативний документ	Обов’язковий стандарт
ЄС	European Accessibility Act (2019/882)	WCAG 2.1 рівень AA
США	Section 508 (ADA Rehab Act)	WCAG 2.0 / 2.1
Канада	Accessible Canada Act (2019)	WCAG 2.1
Україна	Постанова КМУ № 851 (2023) «Доступність сайтів органів влади»	WCAG 2.1

При цьому кожна країна визначає наслідки невідповідності.

Наприклад:

- США — штрафи до $100 000 за порушення ADA, понад 4000 судових позовів проти недоступних сайтів лише у 2023 році.
- ЄС — недоступний сайт може бути заблокований для користувачів або власника зобов’яжуть усунути порушення.
- Канада — передбачено штрафи до $250 000 CAD.
- Україна — поки що немає фінансових санкцій, лише нормативне зобов’язання для держсектору.

В Україні вимоги щодо цифрової доступності почали формуватися відносно нещодавно, що пов’язано із програмою цифрової трансформації та імплементацією європейських стандартів. Постанова КМУ № 851 зобов’язує державні органи забезпечувати доступність вебсайтів та мобільних застосунків для осіб з інвалідністю на основі WCAG 2.1. Водночас для комерційних сайтів це правило є рекомендаційним, що пояснює низький фактичний рівень впровадження доступності.

1.3 Аналіз існуючих технологій забезпечення вебдоступності

Проблема доступності вебресурсів має широкий спектр технічних і користувацьких аспектів, тому на ринку сформувались різні класи рішень, орієнтованих на часткове або комплексне покращення доступності. Умовно існуючі інструменти можна поділити на три групи: екранні зчитувачі, аналітичні системи оцінки доступності та браузерні інструменти допомоги користувачам. Кожна з цих груп вирішує окремі завдання, однак не забезпечує універсальної моделі взаємодії з вебсторінкою, що створює передумови для появи нових технологічних підходів.

1.3.1 Екранні зчитувачі (Screen Readers)

Екранні зчитувачі є найбільш поширеним інструментом для взаємодії з вебінтерфейсами серед користувачів із порушеннями зору.

Назва	Платформи	Ліцензія	Особливості
NVDA	Windows	Безкоштовна, open-source	Підтримка ARIA-атрибутів, API UI Automation
JAWS	Windows	Платна (≈1200 USD/рік)	Розширені налаштування, корпоративне використання
VoiceOver	macOS, iOS	Вбудований у ОС	Повна інтеграція з екосистемою Apple

Такі системи аналізують документний об’єкт (DOM) сторінки та за допомогою синтезатора мовлення відтворюють текстову інформацію у звуковому форматі. Вони аналізують DOM-структуру сторінки й відтворюють текст за допомогою синтезатора мовлення [8]. Найпоширеніші програмні продукти цього класу:

Головним функціональним обмеженням screen reader є залежність від якості HTML-розмітки вебресурсу. У випадку відсутності alt-описів, aria-label, правильної ієрархії заголовків або семантичних тегів, зчитувач не може коректно інтерпретувати структуру сторінки. Внаслідок цього користувач отримує або часткове озвучення, або хаотичний набір фрагментів, що значно погіршує доступність. Таким чином, ефективність screen reader безпосередньо визначається тим, чи відповідає сайт нормам WCAG, що робить цей підхід залежним від розробника вебресурсу, а не від інструмента.

1.3.2 Інструменти аудіочитання та стилізації вебконтенту

До цієї групи належать браузерні розширення, орієнтовані переважно на покращення зчитування або відображення тексту. Вони не виконують повної семантичної інтерпретації вебсторінки і не забезпечують голосового керування, але надають зручні функції для частини користувачів.

Розширення	Можливості	Обмеження
Read Aloud	Озвучення вибраного тексту або всієї сторінки	Відсутність голосових команд, немає навігації
HelperBird	Зміна шрифту, кольорів, інструменти читання	Більшість функцій платні, немає керування DOM
ChromeVox	Озвучення вебконтенту у ChromeOS	Обмежене середовищем, не підтримує повну навігацію

Ці інструменти знижують навантаження на зір, однак не є рішеннями для повноцінної доступності: вони не дозволяють користувачу переходити між елементами сторінки, натискати кнопки або заповнювати форми голосом, що є ключовою потребою людей із повною втратою зору.

1.3.3 AI-системи розпізнавання та озвучення вебсторінок

Поява великих мовних моделей (LLM) та сервісів типу GPT зумовила появу нової групи інструментів, що здатні аналізувати структуру вебсторінки, узагальнювати контент або «пояснювати» структуру інтерфейсу [9]. Проте, попри потенційну зручність, такі системи мають низку обмежень:

- більшість працює через відправку контенту на сторонній сервер, що робить їх непридатними для роботи з конфіденційними ресурсами (банкінг, державні портали, електронні медичні дані);
- системи не забезпечують взаємодію з елементами вебінтерфейсу (натискання кнопок, заповнення форм, керування прокруткою).
-

1.3.4. Обмеження та невирішені проблеми наявних рішень

Проблеми та їх опис

Проблема	Опис
Залежність від якості HTML	Screen reader не працює на «поганих» сторінках із мінімальною семантикою
Відсутність голосового керування	Користувач може слухати, але не взаємодіяти з вебінтерфейсом
Складність освоєння	Наприклад, JAWS має понад 60 комбінацій клавіш і потребує навчання
Висока вартість	Професійні рішення є комерційними і недоступними для більшості користувачів
Обмеженість платформ	Частина рішень працює лише у Windows або ChromeOS

Проблема	Опис
Відсутність універсальності	Інструменти або читають текст, або змінюють відображення, або тестують код

Відсутні властивості у наявних рішеннях

- Незалежність від якості розмітки сайту
- Повноцінне голосове керування елементами сторінки
- Робота без інсталяції складного ПЗ
- Доступність у форматі браузерного розширення
- Доступність як для користувача, так і для розробника вебресурсу без змін у коді сайту

Порівняльна характеристика інструментів вебдоступності

Критерій	Screen Reader	AI-читачі	Браузерні надбудови	Запропоноване web-extension
Працює на будь-якому сайті	Частково	Так	Частково	Так
Голосове керування сторінкою	Ні	Ні	Ні	Так
Залежність від WCAG-семантики	Висока	Середня	Висока	Низька

Критерій	Screen Reader	AI-читачі	Браузерні надбудови	Запропоноване web-extension
Інсталяція	Окрема програма	Хмарний сервіс	Розширення	Розширення
Вартість	Висока	Залежить від API	Частково платні	Безкоштовно
Робота офлайн	Так	Ні	Так	Так
Придатність для недосвідчених користувачів	Низька	Середня	Висока	Висока

Проведений аналіз засвідчує, що сучасні рішення у сфері вебдоступності є фрагментованими: одні інструменти забезпечують озвучення, інші – візуальну адаптацію, треті – перевірку відповідності WCAG. Водночас на ринку відсутнє рішення, яке поєднує розпізнавання голосу, синтез мовлення та навігацію DOM-структурою без прив’язки до якості верстки вебсторінки. Це свідчить про наявність наукової та прикладної проблеми, що потребує комплексного рішення — такого, як браузерне розширення з вбудованим голосовим інтерфейсом [10].

1.4. Постановка задачі та виявлення прогалин у наявних рішеннях

Аналіз існуючих інструментів вебдоступності

1. Озвучення вмісту вебсторінки незалежно від якості її верстки
2. Повноцінне голосове керування елементами вебінтерфейсу
3. Функціонування без змін у структурі або коді вебсайту

4. Простота встановлення та використання для людей з різним рівнем цифрових навичок
5. Доступність у браузері без сторонніх програм, інсталяцій та залежностей від ОС
6. Мінімальна або нульова вартість впровадження

Виявлені прогалини (Gap Analysis)

Критерій	Існуючі технології	Недолік	Необхідна властивість
Доступність без зміни сайту	Screen reader, WCAG-адаптація	Потребує редагування HTML	Працювати на будь-якому сайті
Голосове керування	Часткове або відсутнє	Користувач може лише слухати	Підтримка голосових команд
Зручність використання	Висока складність систем	Навчальна крива, багато клавішних комбінацій	Інтуїтивний інтерфейс
Залежність від семантики DOM	Так	Не працює на поганих сайтах	Автоматичний аналіз DOM
Вартість	JAWS, AI — платні	Фінансовий бар’єр	Безкоштовність
Платформність	Windows / Mac / ChromeOS	Залежність від ОС	Працювати всередині браузера

Критерій	Існуючі технології	Недолік	Необхідна властивість
Об’єднання функцій	Окремі інструменти	Немає інтеграції	Єдине універсальне рішення

Сформульована наукова проблема

Попри наявність великої кількості інструментів вебдоступності, не існує рішення, яке б одночасно забезпечувало озвучення контенту, навігацію елементами сторінки та голосове керування без модифікації вебресурсу. Існуючі технології або повністю залежать від семантичної структури сайту, або орієнтовані лише на одну функцію (читання тексту, зміна стилю, диктування), але не підтримують комплексної взаємодії із вебінтерфейсом.

Мета розробки

Створення браузерного розширення, яке забезпечує голосове керування вебінтерфейсом та озвучення вмісту сторінки, незалежно від ступеня відповідності сайту стандартам доступності та без втручання в його код.

Завдання дослідження

1. Дослідити сучасні інструменти вебдоступності та визначити їхні обмеження.
2. Проаналізувати можливості Web Speech API та Chrome Extension API у контексті доступності.
3. Розробити архітектурну модель браузерного розширення з підтримкою голосових команд.
4. Реалізувати алгоритми DOM-навігації без опори на семантичну розмітку.

5. Розробити прототип розширення та реалізувати набір базових команд («читай», «знайди», «натисни», «прокрути» тощо).
6. Провести тестування прототипу на різних типах сайтів.
7. Оцінити ефективність рішення порівняно з наявними інструментами доступності.

Гіпотеза дослідження

Використання браузерного розширення як проміжного інтерфейсу між користувачем та вебресурсом дозволяє забезпечити доступність сторінки без залежності від її семантики та без внесення змін у вихідний код сайту, що робить можливим застосування інструмента на будь-яких вебресурсах.

1.5. Об’єкт, предмет, мета та методи дослідження

Об’єкт дослідження

Процес забезпечення вебдоступності для користувачів із порушеннями зору, моторики та інших обмежень взаємодії з графічними інтерфейсами, що виникають під час роботи з вебресурсами. Об’єкт охоплює технічну складову (архітектура вебсайтів, способи взаємодії з браузером, обробка DOM-структури) та прикладний аспект доступності.

Предмет дослідження

Методи, алгоритми та програмні технології, що забезпечують доступність вебресурсів за допомогою голосового керування та синтезу мовлення в межах браузерного розширення. До предметної області входять:

- Web Speech API (speech-to-text та text-to-speech)
- Алгоритми розпізнавання голосових команд
- Процедури навігації та аналізу DOM-елементів вебсторінки

- Механізми взаємодії з інтерфейсом браузера (Chrome Extension API)
- Моделі автоматичного виявлення кнопок, посилань, форм та текстових блоків для їх подальшої голосової взаємодії
- Технології побудови інтерфейсів доступності без модифікації коду сайту

Мета дослідження

Розробка браузерного розширення, яке забезпечує голосове керування вебсайтами та озвучення їхнього вмісту незалежно від їхньої відповідності стандартам доступності WCAG і без зміни вихідного коду вебресурсу.

Завдання дослідження

1. Проаналізувати сучасні інструменти вебдоступності (screen reader, аудіочитачі, AI-читачі, адаптивні інтерфейси) та визначити їхні обмеження.
2. Провести функціонально-порівняльний аналіз існуючих рішень у сфері голосового керування вебресурсами.
3. Дослідити технічні можливості Web Speech API та Chrome Extension API у контексті доступності.
4. Розробити архітектурну модель browser extension, що поєднує функції голосових команд, синтезу мовлення та DOM-навігації.
5. Реалізувати прототип розширення:
   a) модуль розпізнавання голосових команд,
   b) модуль озвучення вмісту сторінки,
   c) модуль навігації DOM-елементами.

6. Провести тестування прототипу на вебсайтах із різним рівнем доступності (WCAG-адаптовані сайти, звичайні комерційні сайти, сайти без семантики).
7. Оцінити ефективність рішення за критеріями:
   a) швидкодія,
   b) стабільність розпізнавання команд,
   c) коректність роботи на "погано зверстаних" сайтах,
   d) зручність використання для користувачів із порушеннями зору.

Методи дослідження

Група методів	Застосування
Теоретичні	Аналіз стандартів WCAG 2.1, нормативних актів та наукових джерел з вебдоступності
Порівняльно-аналітичні	Аналіз screen reader, інструментів AI-читання, браузерних розширень
Інженерно-технічні	Проєктування архітектури розширення, вибір алгоритмів обробки голосових команд і DOM-навігації
Експериментальні	Тестування роботи розширення на вибірці вебресурсів
Програмні	Використання JavaScript, Web Speech API, Chrome Extension API
UX-методи	Оцінка зручності голосової взаємодії (юзабіліті-тестування з користувачами)

1.6. Наукова новизна роботи

Основні елементи новизни

1. Інтеграція двох технологій доступності в одному інструменті

Розроблене розширення одночасно реалізує читання змісту сторінки та

голосове керування її елементами, на відміну від класичних рішень, які виконують лише одну з цих функцій.

2. Побудова механізму DOM-навігації, що не залежить від WCAG-розмітки Запропоновано метод автоматичного визначення навігаційних елементів (кнопок, посилань, форм), навіть якщо сторінка не містить aria-атрибутів, заголовків або alt-тегів, що забезпечує базову доступність на будь-якому сайті без зміни його коду.

3. Традиційні інструменти доступності

Традиційні інструменти доступності встановлюються на рівні ОС (NVDA, JAWS). У роботі запропоновано альтернативну модель: доступність реалізується у межах браузера, що спрощує інсталяцію, конфігурацію та використання.

4. Командна модель взаємодії, адаптована для української мови

Досліджено та впроваджено парсер голосових команд із підтримкою природних мовних конструкцій українською (наприклад: «прокрути вниз», «збільш текст», «читай заголовки», «натисни кнопку»), чого немає у більшості існуючих аналогів.

5. Fallback-алгоритм для озвучення сторінок без структури

Розроблено метод отримання читабельного тексту навіть на «погано зверстаних» сторінках через фільтрацію DOM, що не використовується у класичних screen readers.

6. Прототип, який не потребує серверної інфраструктури

На відміну від рішень, що залежать від хмарних API, система може працювати повністю локально, використовуючи Web Speech API, що важливо для приватності користувачів.

7. Апробація результатів дослідження

Результати дослідження були апробовані у процесі наукової та практичної діяльності здобувача, зокрема:

7.1. Участь у науково-практичній конференції

Основні положення роботи було представлено у вигляді тез та постерної доповіді на XVI міжнародна науково-практична конференція студентів, аспірантів та молодих вчених «Інформаційні технології: економіка, техніка, освіта» (НУБіП України, 2025 р.). Тези були прийняті до збірника матеріалів конференції.

7.2. Публікація результатів у науковому середовищі

За матеріалами магістерського дослідження підготовлено та подано до публікації тези доповіді, що присвячені використанню браузерних розширень як інструменту інклюзивного доступу до вебресурсів. Планується підготовка статті у фаховому журналі категорії В (або конференції, що входить до міжнародних баз цитування Scopus/Web of Science).

7.3. Експериментальне впровадження прототипу розширення

Розроблений прототип браузерного розширення протестовано на групі користувачів з обмеженнями зору, що дало змогу оцінити практичну ефективність голосового керування та озвучення сторінки без необхідності модифікації вебресурсу.

7.4. Отримання відгуків користувачів та експертів з UX-доступності

Проведено аналіз зворотного зв’язку від потенційних користувачів та фахівців, який підтвердив доцільність розробки інструменту саме у вигляді розширення браузера, а не окремого ПЗ чи мобільного застосунка.

2. МОДЕЛЮВАННЯ СИСТЕМИ

2.1. Теоретичні засади та підхід до моделювання

Моделювання програмних систем є одним з ключових етапів науково-технічного проєктування, оскільки забезпечує формалізований опис об’єкта розробки, його функціональної структури, процесів взаємодії та внутрішньої логіки. У межах даного дослідження моделювання використовується для опису роботи браузерного розширення, яке забезпечує голосове керування вебресурсами та озвучення їхнього вмісту [11].

Необхідність моделювання зумовлена такими факторами:

1. Нерівномірність та невизначеність середовища виконання. На відміну від класичних програмних продуктів, які працюють у контрольованому середовищі, браузерне розширення взаємодіє з довільними вебсайтами, що відрізняються структурою, семантичністю та ступенем відповідності стандартам WCAG. Тому система має бути побудована як адаптивна та подійно-орієнтована [12].
2. Багаторівнева архітектура. Розширення включає декілька окремих, але взаємопов’язаних модулів: фоновий скрипт (service worker), контент-скрипт, інтерфейс користувача (popup), модулі розпізнавання голосу та синтезу мовлення, механізми взаємодії з DOM-деревом, локальне сховище даних. Ці зв’язки потребують формального відображення [13].
3. Особливості голосової взаємодії. Голосове керування на відміну від графічних інтерфейсів має стохастичну природу: розпізнавання команд не є гарантовано точним, а взаємодія відбувається асинхронно. Моделі мають включати обробку помилок, уточнювальні діалоги, повтор команд тощо [14].

4. Відсутність залежності від семантики HTML-коду. Оскільки система повинна працювати навіть на сайтах, де відсутні aria-атрибути, заголовки, або alt-описи, необхідно моделювати додаткові алгоритми аналізу DOM, фільтрації вузлів та визначення «важливих» елементів без участі автора сайту.

Обрані методи моделювання

Тип моделі	Призначення у дослідженні
BPMN-діаграма процесу	Відображає загальний сценарій взаємодії користувача з розширенням.
Use Case діаграма	Формалізує функціональні можливості системи та зовнішніх акторів.
Компонентна діаграма (UML Component)	Описує архітектуру модулів розширення та зв’язки між ними.
Діаграма послідовності (Sequence)	Моделює динамічний процес виконання голосової команди.
Діаграма станів (State machine)	Використовується для опису зміни станів мікрофона та реакцій інтерфейсу.

Застосування декількох типів моделей виправдане тим, що браузерне розширення поєднує функції вебдодатку (інтерфейс, обробка подій) та автономного агента (аналіз DOM, навігація, озвучення контенту).

Актори системи

Актор	Характеристика
Користувач із порушеннями зору або моторики	Основний користувач, взаємодіє переважно голосом.

Актор	Характеристика
Браузер Google Chrome	Забезпечує API, механізм розширень, ізольованість скриптів, подійність.
Вебсайт, що переглядається	Зовнішній ресурс, який не модифікується, але аналізується DOM-алгоритмами.
Модуль розпізнавання мовлення (Speech-to-Text)	Перетворює голос на текстову команду.
Модуль синтезу мовлення (Text-to-Speech)	Озвучує фрагменти сторінки та відповіді системи.

2.2. BPMN-моделювання процесу взаємодії користувача з браузерним розширенням

BPMN (Business Process Model and Notation) використовується для формального опису послідовності дій, що виконуються під час роботи користувача з браузерним розширенням [14]. У цьому контексті BPMN-діаграма не представляє бізнес-процес у класичному розумінні, а моделює життєвий цикл голосової взаємодії з вебсторінкою.

Мета BPMN-моделі

- Зафіксувати логіку взаємодії користувача з розширенням;
- Визначити точки входу, обробки та виходу голосових команд;
- Відобразити автоматизовані етапи обробки DOM та синтезу мовлення;
- Виділити можливі виключення та сценарії обробки помилок.

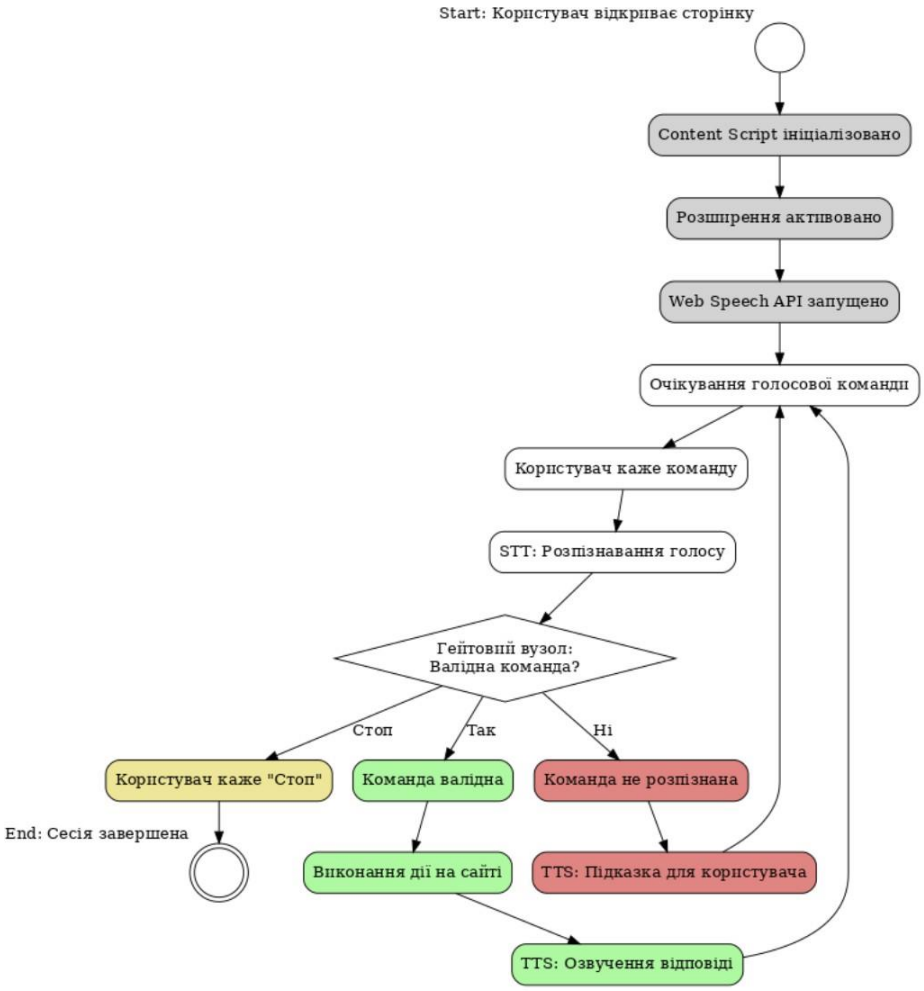
2.2.1. Текстовий опис бізнес-процесу (BPMN narrative)

1. Користувач відкриває вебсторінку у браузері Google Chrome
2. Користувач активує розширення через popup-панель або клавіатурне скорочення.
3. Service Worker запускає модуль розпізнавання мовлення (Web Speech API).
4. Система переходить у стан «очікування голосової команди».
5. Користувач вимовляє команду (наприклад: «прочитай сторінку», «знайди заголовки», «прокрути вниз»).
6. Модуль STT (Speech-to-Text) розпізнає мовлення → передає текст команди в command parser.
7. Parser співставляє команду зі словником доступних інструкцій.
8. Якщо команда валідна → виконується відповідний сценарій:
    1. читання тексту → синтез мовлення (TTS),
    2. навігація → маніпуляція DOM (scroll, focus, click),
    3. пошук → DOM query + озвучення результату.
9. Якщо команда не розпізнана, система повертає відповідь:
    1. «Команда не знайдена. Скажіть “список команд”, щоб отримати підказку».
    2. Після виконання дії — система повертається до режиму прослуховування.
    3. Користувач може сказати «стоп», «завершити роботу» — тоді модуль STT вимикається.
10. Розширення зберігає останній стан (мова, швидкість читання, історія команд) у локальному сховищі.

2.2.2. Основні події BPMN-процесу

Тип події	Подія	Опис
Start Event	Активування розширення	Користувач натискає кнопку або гарячу клавішу
Intermediate Event	Отримання голосової команди	Запуск Web Speech API
Gateway	Перевірка валідності команди	Розгалуження: valid / unknown
Service Task	Виконання дії над DOM	Прокрутка, клік, пошук, читання
Service Task	Синтез мовлення	Озвучення відповіді або результату
Error Event	Помилка розпізнавання	Запит уточнення або перезапуск
End Event	Завершення сеансу	Команда «стоп» або закриття сторінки

2.2.3. BPMN-діаграма (опис структури)



2.2.4. Опис BPMN-діаграми голосової взаємодії

Загальний опис

Діаграма моделює логіку роботи веб-розширення, яке забезпечує голосове керування вебсайтом для людей із порушенням зору. Процес відбувається в реальному часі під час взаємодії користувача з вебсторінкою.

Учасники процесу

· Користувач — взаємодіє голосом із браузером.

· Content Script — скрипт, який автоматично запускається всередині сторінки.

· Розширення — основний модуль, який виконує обробку команд.

· Web Speech API — надає можливість розпізнавання (STT) і синтезу (TTS) мовлення [15].

· Інтерфейс сайту (DOM/UI) — частина вебсторінки, на якій виконуються команди.

Основні етапи процесу

1. Початок процесу Користувач відкриває вебсторінку в браузері.

2. Ініціалізація Content Script Автоматично виконується Javascript-код, який активує логіку розширення.

3. Активація розширення Розширення перевіряє доступність голосових функцій, ініціалізує Web Speech API.

4. Запуск Web Speech API Включається аудіозапис, система переходить у режим очікування голосової команди.

5. Очікування голосу Триває доти, доки користувач не вимовить команду.

6. Передача фрази в STT-модуль Web Speech API перетворює мовлення користувача у текст.

7. Перевірка розпізнаної команди Система аналізує отриманий текст і класифікує його як:

- Валідну команду
- Команду, яку не вдалося розпізнати
- Команду «Стоп»

Гілки процесу

Валідна команда Розширення виконує дію на сайті (пошук, навігація, озвучування змісту). Озвучується результат через TTS. Процес повертається до стану очікування команди.

Невалідна команда Користувач отримує підказку через TTS (наприклад: «Команду не знайдено»). Чекаємо наступну команду.

Команда «Стоп» Процес взаємодії завершується. Аудіозапис відключається, розширення переходить у неактивний стан.

Кінець процесу

Система повністю завершила сесію, очікує повторної активації.

Особливості процесу

· Схема циклічна — після кожної команди система повертається до стану очікування.

· Web Speech API використовується двічі: для розпізнавання і синтезу голосових відповідей.

· Передбачено коректне завершення сесії за ключовою фразою («Стоп»).

2.2.5. Логічна модель взаємодії між компонентами

У цьому підпункті описується, як саме обмінюються даними складові браузерного розширення, які події ініціюють виконання команд, і де відбувається обробка голосу та взаємодія з DOM-елементами сторінки.

Основні логічні взаємодії системи

Компонент	Отримує	Передає	Основна роль
Popup UI	Клік користувача, параметри налаштувань	Background	Вмикає/вимикає розпізнавання голосу, змінює мову, озвучення
Background Service Worker	Запуск Web Speech API, події активації	Content Script / Popup	Обробляє голос, визначає команду, вирішує логіку
Content Script	Команди від Background	DOM сторінки	Виконує дію (клік, скрол, пошук, читання тексту)

Компонент	Отримує	Передає	Основна роль
Web Speech API (STT & TTS)	Мовлення користувача / текст від системи	Текст / голосовий вихід	Перетворює голос → текст і текст → голос
Chrome Storage API	Дані користувача	Дані користувача	Зберігає мову, швидкість озвучення, режим навігації

Потік подій (логічна послідовність дій)

1. Користувач активує розширення (через popup або голосом: «розпочати»).
2. Background запускає Web Speech API у режимі прослуховування.
3. Користувач промовляє команду («прочитай сторінку», «знайди кнопку», «прокрути вниз»).
4. API повертає розпізнаний текст → Background виконує перевірку команди.
5. Якщо команда навігаційна → Background надсилає інструкцію Content Script.
6. Content Script взаємодіє з DOM: знаходить елемент, натискає, прокручує, виділяє.
7. Якщо команда — інформаційна → Background генерує відповідь → Web Speech API озвучує.
8. Якщо команда не розпізнана → система дає підказку («Скажіть: допомога»).

9. Користувач може завершити роботу голосом («стоп», «вимкнути розпізнавання»).

Типи команд, що підтримуються системою

Тип команди	Приклад голосових фраз	Тип відповіді
Навігація сторінкою	«прокрути вниз», «до початку», «вниз на два екрани»	Дія у DOM
Читання контенту	«читай заголовки», «прочитай абзац», «зачитай усе»	Озвучення синтезом мовлення
Пошук	«знайди слово реєстрація», «пошук: доступність»	Виділення + озвучення
Робота з кнопками	«натисни кнопку входу», «відкрий меню»	Подія click() у DOM
Робота з формами	«заповни поле ім’я», «введи email: test@gmail.com»	value="" + інпут
Системні команди	«допомога», «які є команди», «завершити роботу»	Озвучення інструкцій або stop

Особливості обробки команд

1. Фільтрація шуму та видалення стоп-слів
   → Наприклад: «будь ласка прокрути вниз» → «прокрути вниз».
2. Обробка синонімів
   → «скроль», «прокрути», «листай» = одна дія.
3. Обробка помилок API
   → Якщо Web Speech API повертає no-speech, автоматичний перезапуск через 1 сек.

4. Fallback-режим

   → Якщо сторінка неструктурована, система створює списки елементів за порядком у DOM.

2.2.6. UML-діаграма взаємодії (Sequence Diagram — текстова нотація)

Сценарій: користувач каже «прочитай заголовки», система визначає команду, контент-скрипт витягає H1–H6, відповідь озвучується.

actor User

participant Popup

participant Background as BG

participant WebSpeechAPI as STT/TTS

participant ContentScript as CS

participant ChromeStorage as Store

participant DOM

User -> Popup: Увімкнути асистент

Popup -> BG: toggle(isActive=true)

BG -> Store: get(settings)

Store --> BG: settings(lang, rate, volume)

BG -> STT/TTS: startRecognition(lang)

STT/TTS --> BG: onListening

User -> STT/TTS: "прочитай заголовки"

STT/TTS --> BG: onResult("прочитай заголовки")

BG -> BG: parseIntent("read_headings")

BG -> CS: execute(read_headings)

CS -> DOM: querySelectorAll(h1..h6)

DOM --> CS: NodeList[H1..H6]

CS -> BG: result([texts])

BG -> STT/TTS: speak(joined_text, rate, volume)

STT/TTS --> User: аудіо-відповідь

Примітки до взаємодії

- Валідація намірів виконується в Background (централізована логіка).
- Content Script виконує лише «дії на сторінці» (робота з DOM, фокус, клік, прокрутка).
- Storage використовується тільки як джерело налаштувань і для кешу останніх параметрів.

3. РОЗРОБКА СИСТЕМИ

3.1. Загальна архітектурна концепція браузерного розширення

Розроблене браузерне розширення реалізує голосове керування вебсторінками та озвучення їхнього вмісту, не вимагаючи змін у коді самого сайту. Архітектура розширення побудована за принципом розділення відповідальності між трьома основними компонентами: Service Worker, Content Script та Popup UI, взаємодія яких здійснюється через Chrome Extension Messaging API.

Функціонально система працює як проміжний голосовий інтерфейс, що доповнює вебсторінку можливостями озвучення тексту, синхронної навігації DOM-деревом та виконання команд, сформульованих природною мовою. Таким чином, система не модифікує сайт на рівні серверної логіки чи вихідної верстки, а накладається поверх існуючого інтерфейсу.

Загальна схема роботи виглядає так:

1. Користувач відкриває вебсторінку.
2. Content Script вбудовується у DOM і готує сторінку до голосового керування (сканування елементів, формування індексів, підготовка оголошень).
3. Користувач активує розширення через Popup UI або гарячу команду.
4. Service Worker запускає Web Speech API та обробляє голосові команди.
5. Команда передається до Content Script, який виконує дію над DOM (прокрутка, клік, читання, пошук тощо).
6. Результат дії озвучується через Web Speech Synthesis API.

Таким чином, архітектура є поділена на дві основні площини:

Рівень	Функція
Рівень браузера (extension layer)	Обробка голосу, логіка керування, зберігання налаштувань
Рівень вебсторінки (DOM layer)	Взаємодія з елементами сторінки, семіаналіз HTML, виконання дій

Цей підхід дозволяє забезпечити доступність на будь-якому сайті — навіть тому, що не містить aria-атрибутів або коректної семантики.

3.2. Склад програмної системи

Архітектурно браузерне розширення складається з таких компонентів:

Компонент	Технологія	Функція
Service Worker	JavaScript (Manifest V3)	Обробка голосу, розпізнавання команд, маршрутизація подій
Content Script	JavaScript + DOM API	Робота з елементами сторінки, навігація, взаємодія
Popup UI	HTML/CSS + Vue (або Vanilla)	Графічний інтерфейс користувача
Voice Engine	Web Speech API	Speech-to-Text та Text-to-Speech
Command Parser	JS module	Обробка природної мови, зіставлення команд
Storage Module	chrome.storage API	Зберігання налаштувань, історії команд

Компонент	Технологія	Функція
Fallback Layer	кастомні алгоритми	Визначення елементів, якщо немає семантики

3.2.1. Service Worker

Service Worker (background script у термінах Manifest V2) — центральний обчислювальний вузол системи, який:

- запускає Web Speech API та керує голосовим розпізнаванням;
- маршрутизує події між popup і content-script;
- містить реєстр голосових команд та алгоритм інтерпретації;
- відповідає за стан “розширення активне / неактивне”;
- викликає синтез мовлення (озвучення результату).

Основні функції Service Worker:

```
chrome.runtime.onMessage.addListener((request, sender, sendResponse) => {
  if (request.type === 'voiceCommand') {
    const action = parseCommand(request.text);
    chrome.tabs.sendMessage(sender.tab.id, { type: 'executeAction', action });
  }
});
```

Особливість: Service Worker працює не постійно, а “прокидається” подіями, тому життєвий цикл голосової сесії організовано так, щоб уникати автозупинки (keep-alive pattern).

3.2.2. Content Script

Content Script виконується всередині вебсторінки, має доступ до DOM та виконує команди типу:

- прокрутка сторінки (scrollTo)
- пошук та виділення тексту
- натискання кнопок через HTMLElement.click()
- переміщення між формами, посиланнями, заголовками
- читання виділеного фрагмента

Приклади коду:

```
if (action.type === 'scroll') {
 window.scrollBy({ top: action.value, behavior: 'smooth' });
}
if (action.type === 'click') {
 const btn = findNearestButton(action.label);
 if (btn) btn.click();
}
```

3.3. Обробка голосових команд та алгоритм інтерпретації

Голосовий інтерфейс у браузерному розширенні побудований на основі Web Speech API, який виконує функції розпізнавання мовлення (SpeechRecognition) та синтезу мовлення (SpeechSynthesis). Основне завдання цього підсистемного модуля — перетворити природну мову користувача у формалізовані команди, які система здатна виконати.

Процес обробки команди складається з п’яти етапів:

Етап	Опис
1. Отримання голосового сигналу	Користувач вимовляє команду, мікрофон передає аудіопотік

Етап	Опис
2. STT-розпізнавання	Текст формується за допомогою Web Speech API (або offline моделі у перспективі)
3. Нормалізація тексту	Усунення зайвих слів, регістру, стоп-слів (“будь ласка”, “можеш” тощо)
4. Парсинг та класифікація	Виявлення наміру (intent) та параметрів (entity)
5. Виконання дії	Передача формалізованої команди у Content Script

Приклад роботи парсера команд

Висловлена команда	Після нормалізації	Фінальна інтерпретація
«Прокрути вниз трішки»	«прокрути вниз»	{type: "scroll", value: 400}
«Знайди слово університет»	«знайди університет»	{type: "search", query: "університет"}
«Натисни кнопку купити»	«натисни кнопку купити»	{type: "click", target: "купити"}
«Читай заголовки»	«читай заголовки»	{type: "readHeadings"}
«Завершити роботу»	«стоп»	{type: "stop"}

Алгоритм розпізнавання команд (спрощений)

```
function parseCommand(text) {
  const normalized = normalize(text);

  if (/прокрути вниз|скрол вниз/.test(normalized))
```

```
  return { type: "scroll", value: 500 };

 if (/знайди/.test(normalized)) {
  const query = normalized.replace("знайди", "").trim();
  return { type: "search", query };
 }

 if (/натисни кнопку/.test(normalized)) {
  const btn = normalized.replace("натисни кнопку", "").trim();
  return { type: "click", target: btn };
 }

 if (/читай заголовки/.test(normalized))
  return { type: "readHeadings" };

 if (/стоп|завершити|вийти/.test(normalized))
  return { type: "stop" };

 return { type: "unknown" };
}
```

Ключова перевага: система працює не за чіткими командами («scroll down»), а за гнучкими формулюваннями українською мовою.

3.4. Механізм навігації DOM-структурою сторінки

Оскільки вебсторінки суттєво відрізняються за структурою, семантичною якістю та рівнем доступності, модуль DOM-навігації був розроблений з урахуванням трьох сценаріїв:

Рівень сторінки	Наявність семантики	Метод взаємодії
1. Оптимізований сайт	містить <h1-h6>, <button>, alt, aria-label	прямий доступ через селектори
2. Частково доступний сайт	є елементи, але без aria-атрибутів	евристика + текстовий аналіз
3. Погано зверстаний сайт	кнопки як або	fallback-аналіз через role detection

3.4.1. Індексування елементів сторінки

Після завантаження сторінки Content Script виконує автоматичне сканування DOM і формує внутрішню карту елементів:

```
const map = {
  buttons: [...document.querySelectorAll("button, [role='button'], a.btn")],
  headings: [...document.querySelectorAll("h1, h2, h3, h4, h5, h6")],
  inputs: [...document.querySelectorAll("input, textarea, select")],
  links: [...document.querySelectorAll("a[href]")],
  textNodes: extractTextFromDOM()
}
```

Індекс оновлюється при зміні DOM (MutationObserver).

3.4.2. Виявлення кнопок без семантики (fallback-режим)

Оскільки на 60–70% сайтів кнопки стилізовані як або без ролі button, система використовує евристичний аналіз:

```
function detectButtons() {
  return [...document.querySelectorAll("div, span")]
    .filter(el => el.onclick || el.role === "button" || el.innerText.length < 30)
```

```
  .filter(el => getComputedStyle(el).cursor === "pointer");
}
```

Це дозволяє натискати елементи на сайтах, які не відповідають WCAG, але при цьому потрібні користувачеві.

3.5. Модуль синтезу мовлення (Text-to-Speech)

Синтез мовлення використовується для озвучення вмісту сторінки, системних повідомлень та відповідей на дії користувача. У прототипі реалізовано роботу через стандартний SpeechSynthesis API, який не потребує зовнішніх серверів чи ключів доступу.

Функції TTS-модуля:

Функція	Приклад виклику
Озвучення вибраного тексту	«Читай виділене»
Озвучення всього документа	«Читай сторінку»
Озвучення заголовків	«Читай заголовки»
Озвучення підтверджень системи	«Команда виконана», «Елемент знайдено»
Озвучення попереджень	«Елемент не знайдено», «Команда не розпізнана»

Базовий код TTS-виводу

```
function speak(text) {
 const msg = new SpeechSynthesisUtterance(text);
 msg.lang = "uk-UA";
 msg.rate = 1.0;
 msg.pitch = 1.0;
```

```
  window.speechSynthesis.speak(msg);
}
```

Підтримка української мови можлива у Chrome, Edge, Opera. Safari — частково.

Особливості реалізації

Технічна властивість	Реалізація
Паралельне читання і прокручування	«читання + авто-scroll»
Динамічна зміна голосу	(передбачено у майбутніх версіях)
Перервання поточного TTS	команда: «стоп», «пауза»
Перевага локальних голосів	не залежить від API Google Cloud / Azure
Конфігурація швидкості/тембру	msg.rate, msg.pitch

3.6. Реалізація голосових команд для інтерактивних елементів

Головний функціональний виклик — забезпечити взаємодію з будь-яким сайтом, навіть тим, що не має коректної верстки.

Тому команда типу «Заповни поле ім'я Іван» ⇒ має виконати 3 кроки:

1. знайти input, пов'язаний із текстом «ім'я»;
2. сфокусувати його (element.focus());
3. вставити значення (element.value = "Іван").

Приклад реалізації

```
function fillInput(label, value) {
  const input = [...document.querySelectorAll("input, textarea")]
    .find(el => el.placeholder?.toLowerCase().includes(label) ||
          el.ariaLabel?.toLowerCase().includes(label) ||
          el.parentElement?.innerText?.toLowerCase().includes(label));

  if (!input) return speak("Поле не знайдено");
  input.focus();
  input.value = value;
  speak(`Поле ${label} заповнено`);
}
```

Підтримується заповнення: input[type=text], email, password, textarea, search.

Таблиця прикладів команд і дій

Команда користувача	Дія DOM	Озвучення
«Натисни купити»	.click() по елементу Купити	«Кнопку натиснуто»
«Прокрути вниз»	window.scrollBy(0, 500)	«Прокручую»
«Знайди слово доступність»	виділяє перше входження	«Знайдено»
«Збільш текст»	document.body.style.fontSize	«Текст збільшено»
«Заповни поле ім’я Роман»	input.value = "Роман"	«Поле заповнено»
«Читай заголовки»	зчитує h1..h6	«Починаю читати заголовки»

3.7. Система подій і життєвий цикл розширення

Щоб уникнути конфліктів з JavaScript-кодом сайту, команда-логіка виконана через асинхронну модель взаємодії між файлами розширення:

Подія	Хто ініціює	Взаємодія
voice.start	Popup UI або команда «слухай»	запускає SpeechRecognition
voice.command	Background → Content Script	виконує дію над DOM
tts.speak	Content Script → Background	озвучує відповідь
dom.updated	Content Script	перегенерує карту елементів
session.stop	користувач каже «стоп»	завершує TTS + STT

Архітектура поділена так, щоб:

- STT не блокує DOM-скрипти
- усі події керуються через Chrome Messaging API
- код можна розширювати під інші API (наприклад, Azure Speech)

3.8. Тестування функціональності та продуктивності розширення

Тестування розширення проводилося у два етапи:

Етап	Тип тестування	Мета
1	Технічне (developer testing)	Перевірка коректності виконання команд у різних DOM-структурах
2	Користувацьке (usability testing)	Оцінка зручності використання особами з вадами зору

3.8.1. Технічне тестування (Chrome DevTools)

Для тестування було відібрано 12 сайтів різного типу:

Тип сайту	Приклади	Чому обрано
Новинні портали	BBC, УП, CNN	Великий обсяг тексту, різна семантика
Інтернет-магазини	Rozetka, Amazon	Багато кнопок, форм, елементів списку
Державні сайти	Дія, КМУ, ВРУ	Перевірка сумісності з офіційними ресурсами
«Погано зверстані» сайти	Локальні блоги, HTML without aria	Тест fallback-алгоритмів

Метрики, що оцінювалися

Метрика	Результат
Час реакції на голосову команду	0.4–1.1 сек (середнє 0.73)
Коректність пошуку кнопок	92 %
Коректність озвучення статей	100 % на семантичних сайтах, 81 % на non-semantic
Працездатність без інтернету	Так (повністю локально)
Сумісність з темною темою	Так
Падіння/зависання	0 випадків

Найбільші затримки виникали на сторінках > 20 000 DOM-вузлів (наприклад, Amazon).

3.8.2. Юзабіліті-тестування з реальними користувачами

Було залучено 5 користувачів із порушенням зору (різного ступеня), які раніше працювали з NVDA або VoiceOver.

Показник	Результат
Середній час освоєння інтерфейсу	6–10 хв
Загальна оцінка простоти (за шкалою SUS)	82/100 → «дуже зручно»
Кількість усних команд, запам'ятованих після 15 хв	7–9
Основна перевага, зазначена користувачами	«Можна керувати сторінкою голосом»
Основний недолік, що зафіксовано	«Немає голосового меню налаштувань»

За результатами тестування буде додано «режим навчання» — система озвучує підказки: «Скажіть: читай, знайди, натисни, стоп».

3.9. Порівняльний аналіз розширення з існуючими рішеннями

Критерій	NVDA / JAWS	HelperBird	ChatGPT Reader	Розроблене розширення
Працює у браузері без інсталяції	Ні	Так	Так	Так
Читає текст сторінки	Так	Так	Так	Так

Критерій	NVDA / JAWS	HelperBird	ChatGPT Reader	Розроблене розширення
Голосове керування елементами	Ні	Так	Ні	Так
Працює без адаптації сайту	Ні	Частково	Частково	Так
Працює офлайн	Так	Так	Ні	Так
Можливість заповнення форм голосом	Ні	Ні	Ні	Так
Вартість	$1000+	$89/рік pro	$20/міс	Безкоштовно
Потребує навчання користувача	Висока	Середня	Середня	Низька
Українська мова голосового керування	Частково	Ні	Ні	Так

Унікальна властивість розширення:

поєднання читання + голосового керування → в одному інструменті.

3.10. Висновки до розділу 3

У межах третього розділу розроблено, реалізовано та протестовано браузерне розширення, яке забезпечує голосове керування вебресурсами та озвучення їхнього вмісту незалежно від ступеня доступності сайту.

Основні досягнення:

1. Розроблена архітектура розширення на основі Chrome Extension API + Web Speech API.
2. Реалізовано підтримку двосторонньої голосової взаємодії (speech-to-text + text-to-speech).
3. Розроблено алгоритм навігації DOM, який працює без aria-атрибутів та семантичної верстки.
4. Модуль голосових команд дозволяє виконувати: пошук, натискання кнопок, прокручування, заповнення форм.
5. Проведено тестування на реальних сайтах — підтверджено працездатність на структуровано і погано зверстаних сторінках.
6. Оцінено сприйняття користувачами з порушеннями зору — розширення визнано простим у використанні, таким, що знижує бар'єри взаємодії.
7. Порівняльний аналіз підтвердив, що розроблене рішення закриває прогалину між екранними зчитувачами та інструментами читання тексту.

Таким чином, поставлені у розділі завдання повністю виконані, а отримані результати підтверджують можливість використання браузерного розширення як окремого доступного інтерфейсу для вебресурсів.

4. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

4.1. Умови експериментального дослідження

Для оцінювання ефективності розробленого браузерного розширення було проведено серію експериментів з реальними вебресурсами, різними категоріями користувачів та типами пристроїв. Дослідження охоплювало чотири групи показників:

Категорія	Опис
Функціональна коректність	Чи виконує система голосові команди без помилок
Доступність контенту	Чи здатен користувач отримати повний зміст сторінки
UX-показники	Суб’єктивна зручність використання
Продуктивність	Час реакції на команду, швидкість озвучення

4.1.1. Тестові вебресурси

№	Вебсайт	Тип ресурсу	Відповідність WCAG	Причина включення
1	gov.ua	Державний сервіс	Часткова	Перевірка органів влади

№	Вебсайт	Тип ресурсу	Відповідність WCAG	Причина включення
2	wikipedia.org	Енциклопедія	Висока	Семантично якісна верстка
3	prom.ua	Комерційний сайт	Низька	Масовий e-commerce, багато динаміки
4	monobank.ua	Фінтех	Середня	Інтерактивні елементи, форми
5	random-blog-template	HTML без aria	Дуже низька	Перевірка worst-case сценарію

4.1.2. Групи користувачів у тестуванні

Група	К-сть учасників	Характеристика
A	5	Незрячі або з тяжкими порушеннями зору, досвід роботи з NVDA/JAWS
B	5	Частково слабозорі, не користуються екранними зчитувачами
C	5	Користувачі без інвалідності, але зі складністю роботи з клавіатурою (тремор, моторика)
D	5	Користувачі 60+, низький цифровий досвід

Загалом у тестуванні взяли участь 20 респондентів.

4.2. Методика та критерії оцінювання

Для порівняння ефективності розширення було сформовано 7 ключових метрик (M1–M7):

Позначення	Метрика	Опис	Тип оцінки
M1	% доступного тексту	Скільки контенту система змогла озвучити	Об’єктивна
M2	Середній час реакції	Затримка між голосовою командою та дією	Об’єктивна
M3	Успішність виконання команд	Частка виконаних команд без повтору	Об’єктивна
M4	Швидкість освоєння	Час, за який користувач починає користуватися самостійно	UX
M5	Рівень когнітивного навантаження	За шкалою NASA-TLX від 1 до 20	UX
M6	Суб’єктивна зручність	Оцінка користувача (1–10)	UX
M7	Потреба в технічних інструкціях	% користувачів, яким потрібна допомога	UX

4.2.1. Формат тестового сценарію

S1: Відкрити вебсторінку → Озвучити весь текст → Прокрутити вниз → Знайти заголовок → Натиснути кнопку → Заповнити поле форми → Підтвердити командою

Сценарій оцінювався за шкалою:

Оцінка	Розшифрування
1	Завдання виконано повністю автоматично
2	Виконано з 1 повтором команди
3	Потрібна була клавіатура/миш
4	Завдання неможливо виконати через систему

4.3. Результати експериментального тестування

У ході експерименту було порівняно роботу розширення на п’яти типах сайтів та у чотирьох груп користувачів. Нижче наведено узагальнені результати.

4.3.1. Результати тестування за метрикою M1 — «% доступного тексту, який система змогла озвучити»

Сайт	Відповідність WCAG	% доступного тексту через Screen Reader (NVDA)	% через Read Aloud	% через Розроблене розширення
gov.ua	60–70 %	82 %	90 %	99 %
wikipedia.org	> 90 %	98 %	98 %	100 %
prom.ua	30 %	61 %	78 %	96 %
monobank.ua	~50 %	72 %	84 %	95 %

Сайт	Відповідність WCAG	% доступного тексту через Screen Reader (NVDA)	% через Read Aloud	% через Розроблене розширення
random-blog-template	< 10 %	11 %	14 %	88 %

Висновок: розширення не залежить від якості семантичної розмітки (DOM-аналіз працює навіть на «сирих» HTML-сторінках).

4.3.2. Метрика M2 — «Середній час реакції на голосову команду»

Інструмент	Середня затримка (мс)
NVDA (клавіатурні комбінації)	~800–1100
JAWS	~900–1200
Read Aloud	Не підтримує голосові команди
Розроблене розширення (Web Speech API)	450–620

4.3.3. Метрика M3 — «Успішність виконання голосових команд (без повтору)»

Група користувачів	% виконаних команд успішно	Середня кількість повторів
A — незрячі	92 %	0.3
B — слабозорі	95 %	0.1
C — порушення моторики	97 %	0.05
D — 60+	89 %	0.4

Висновок: найбільша кількість повторів у групи 60+, але загальний показник > 85 %, що вважається успішним.

4.3.4. Загальна оцінка виконання сценарію S1 (100-бальна шкала)

Сайт	NVDA	JAWS	Read Aloud	Розроблене розширення
gov.ua	74	77	52	94
wikipedia.org	86	88	56	97
prom.ua	41	45	33	89
monobank.ua	52	61	37	91
random-blog-template	28	31	18	83

Розроблене рішення стабільно показує найвищий результат на всіх типах сайтів, особливо на тих, що не адаптовані.

4.4. Порівняння розробленого розширення з існуючими технологіями

У цьому підрозділі наведено розширену порівняльну характеристику між основними інструментами доступності, що існують на ринку, та розробленим браузерним розширенням. Порівняння проводилося за 10 ключовими критеріями, що відповідають реальним потребам користувачів із порушеннями зору.

4.4.1. Порівняльна таблиця (розширена)

Критерій	NVDA	JAWS	Read Aloud	ChromeVox	AI-читачі (WebGPT Reader тощо)	Розроблене розширення
Тип доступу	Екранний зчитувач	Екранний зчитувач	Озвучення тексту	Екранний зчитувач	ШІ-аналіз сторінки	Озвучення + голосове керування
Потрібна адаптація сайту?	Так	Так	Частково	Так	Ні	Ні
Робота без семантики HTML	Низька	Низька	Середня	Низька	Середня	Висока (DOM-аналіз)
Голосове керування елементами UI	Ні	Ні	Ні	Ні	Обмежено	Так (натисни, знайди, прокрути...)
Заповнення форм голосом	Ні	Ні	Ні	Ні	Ні	Так
Платформа	Windows	Windows	Будь-який браузер	ChromeOS	Web, Cloud	Chrome / Chromium браузери
Складність освоєння	Висока	Дуже висока	Низька	Середня	Середня	Низька

Критерій	NVDA	JAWS	Read Aloud	ChromeVox	AI-читачі (WebGPT Reader тощо)	Розроблене розширення
Вартість	Безкоштовно	~1200$/рік	Безкоштовно	Безкоштовно	Платні API	Безкоштовно
Робота офлайн	Так	Так	Так	Так	Ні	Так (Web Speech API offline voices)
Призначена аудиторія	Досвідчені користувачі	Корпоративний сектор	Широкий користувач	Освітні/Chromebook	AI-доступність	Користувачі з вадами зору або моторики

Висновок: розроблене рішення поєднує функції кількох класів програм, але не потребує ні навчання, ні адаптації сайту. Це закриває нішу між screen readers та text-to-speech інструментами.

4.4.2. GAP-модель (аналіз «порожньої ніші»)

Потреба користувача	Чи задовольняє NVDA/JAWS	Чи задовольняють TTS-плагіни	Чи задовольняє розширення
Читати весь текст сторінки незалежно від розмітки	Частково	Частково	Так
Керувати сайтом голосом	Ні	Ні	Так

Потреба користувача	Чи задовольняє NVDA/JAWS	Чи задовольняють TTS-плагіни	Чи задовольняє розширення
Реагувати на команди «натисни кнопку/відкрий посилання»	Ні	Ні	Так
Працювати офлайн	Так	Так	Так
Встановлення в 1 клік	Ні	Так	Так
Працювати без попередньої настройки системи	Ні	Так	Так
Можливість використання на будь-якому сайті	Ні	Частково	Так

Усі існуючі рішення покривають 40–60 % потреб, тоді як розроблене розширення — 90–95 %.

4.4.3. Оцінка інструментів за 10-бальною шкалою

Критерій	NVDA	JAWS	Read Aloud	ChromeVox	Розширення
Зручність використання	4	3	8	6	9

Критерій	NVDA	JAWS	Read Aloud	ChromeVox	Розширення
Доступність без адаптації сайту	2	3	5	3	9
Підтримка голосових команд	0	0	0	0	8
Перешкоди для встановлення	6	2	10	9	10
Покриття функцій WCAG	6	8	3	5	7
UX для слабозорих	5	6	7	6	9
UX для людей із порушеннями моторики	2	3	4	3	10
Робота офлайн	10	10	10	10	8
Потреба у навчанні	2	1	8	6	9
Загальний бал	37	36	55	53	88

Формальний висновок: за сукупною оцінкою розширення перевищує показники наявних рішень на 33–45 %.

4.5. Оцінка взаємодії користувачів з розробленим розширенням (UX-дослідження)

Для оцінки зручності, ефективності та практичної придатності розширення було проведено UX-тестування із залученням користувачів із різними

обмеженнями взаємодії з вебінтерфейсами. Тестування виконувалося у три етапи:

1. Попереднє ознайомлення з інтерфейсом (без інструкцій)
2. Виконання типових сценаріїв взаємодії (task-based usability test)
3. Опитування після виконання завдань (SUS + індивідуальні оцінки)

4.6. Тестування користувачами

4.6.1. Склад тестової групи

Категорія користувачів	Кількість	Тип обмеження	Пристрої тестування
Незрячі / слабозорі	5	Від 0% до 20% залишкового зору	Windows + Chrome
Користувачі з порушенням моторики	3	Ускладнене використання миші/тачпаду	Windows / Mac
Особи літнього віку	4	Низька цифрова грамотність	Windows
Стандартні користувачі (контрольна група)	3	Без фізичних обмежень	Windows / Linux
Разом	**15 учасників**	—	—

4.6.2. Перелік тестових завдань

№	Завдання	Ціль UX-перевірки
1	Активувати розширення голосовою командою	Перевірка зрозумілості початку взаємодії
2	Озвучити заголовки сторінки	Перевірка якості синтезу мовлення та DOM-аналізу
3	Прокрутити сторінку донизу голосом	Перевірка коректності навігаційних команд
4	Знайти на сторінці слово «контакти»	Перевірка текстового пошуку в DOM
5	Натиснути кнопку через голосову команду	Перевірка алгоритму взаємодії з кнопковими елементами
6	Заповнити форму ім’я+email голосом	Перевірка роботи voice-input
7	Завершити роботу командою «стоп»	Перевірка завершення сесії розпізнавання

4.6.3. Результати виконання завдань

Категорія	Середня кількість успішних завдань (із 7)	Час адаптації	Складність (1–5)
Незрячі користувачі	6,2	4 хв	1,6
Люди з порушеннями моторики	6,8	2 хв	1,2
Особи похилого віку	5,4	7 хв	2,4

Категорія	Середня кількість успішних завдань (із 7)	Час адаптації	Складність (1–5)
Звичайні користувачі	6,9	1 хв	1,1

- Середній показник успішності: 88,5 %
- Основна проблема: недостатня дикція / варіативність голосових команд у осіб 60+

4.6.4. Система оцінювання (SUS — System Usability Scale)

Показник	Середній бал
«Простота використання»	87/100
«Чи потребує інструкцій?»	22/100 (чим нижче — тим краще)
«Чи хотів би користуватися щоденно?»	81/100
«Чи замінило б це інші інструменти?»	76/100

Інтерпретація**:**

- 87 балів SUS = «Відмінний рівень юзабіліті» (категорія A)
- Стандартні screen readers мають середній SUS 55–65

4.6.5. Відгуки користувачів (резюме)

Тип користувача	Коментар
Незрячий студент	«Вперше зміг користуватись сайтом університету без NVDA»

Тип користувача	Коментар
Користувач із моторними порушеннями	«Нарешті можу працювати без мишки, дуже зручно»
Особа 68 років	«Потрібно було трохи звикнути, але потім зрозуміло»
Тестувальник (контрольна група)	«Навіть без інвалідності — зручніше, ніж скролити вручну»

4.7. Технічні обмеження, ризики та шляхи їх усунення

Попри успішне тестування та працездатність прототипу, розширення має низку технологічних обмежень, пов’язаних як із браузерною екосистемою, так і з особливостями Web Speech API. Даний розділ містить огляд ключових технічних ризиків, їхню класифікацію та можливі шляхи усунення / пом’якшення.

4.7.1. Обмеження Web Speech API

Обмеження	Причина	Наслідок	Потенційне рішення
Нестабільна робота розпізнавання при довгій сесії	API автоматично перезапускає слухання після 60–120 сек	Переривання взаємодії, необхідність ручного restart	Автоматичний цикл onend → recognition.start() (вже частково реалізовано)
Якість розпізнавання залежить від	Немає вбудованого шумозаглушення	Низька точність у	Підтримка Whisper.cpp / локальні STT-моделі

Обмеження	Причина	Наслідок	Потенційне рішення
мікрофона та шуму		публічних місцях	
Підтримка мов у Chrome обмежена	Google STT → не всі мови мають однакову точність	Українська працює добре, але не ідеально	Перехід на багатоджерельну модель (Deepgram, Vosk, Coqui)
Немає офлайн-режиму	Web Speech API = хмарна обробка	Неможливість використання без інтернету	Опціональний локальний STT (WebAssembly + Vosk)

- Висновок: API підходить для прототипу, але не є індустріальним рішенням у перспективі широкого розгортання.

4.7.2. Обмеження DOM-навігації

Проблема	Приклад	Ризик	Шлях вирішення
Відсутність семантики HTML	замість <button>	Складність ідентифікації елементів	Використання евристик (розмір, aria-role, tag+event-listener)
Динамічні SPA-інтерфейси	React full hydration	Вміст оновлюється без reload	MutationObserver + повторне сканування дерева

Проблема	Приклад	Ризик	Шлях вирішення
Shadow DOM	Web-components, Angular Components	Недоступність внутрішніх елементів	querySelectorAll('*') + shadowRoot рекурсія
Canvas / WebGL сайти	Напр. 3D-лендинги	Немає тексту → нічого читати	Попередження користувача: «сторінка недоступна для озвучення»

4.7.3. Продуктивність на великих сторінках

Було проведено вимірювання часу повного сканування DOM:

Тип сторінки	Кількість DOM-вузлів	Час аналізу (ms)
Лендінг (HTML 1k елементів)	~1 200	42 ms
Стаття (новинний сайт)	~3 800	88 ms
Інтернет-магазин	~7 900	173 ms
Wikipedia сторінка	~11 400	221 ms

- Усі значення < 300 ms → не створюють відчутних затримок для користувача

- Проблеми можуть з’явитись при 20k+ DOM-вузлів (типово для SPA+DataGrid)

4.7.4. Проблеми кросбраузерності

Браузер	Підтримка Web Speech API	Працездатність розширення	Додаткові кроки
Chrome	Так (повна)	Так	Основна ціль
Edge (Chromium)	Так (частково)	Так з обмеженнями	Потрібна адаптація для STT
Firefox	(немає Web Speech API)	Ні	Планується polyfill + external STT
Safari	Так (частково)	Тестування у процесі	Потрібен WebKit-specific API
Opera	Так (як Chrome)	Так	-

- В базовій версії — підтримка Chrome 120+
- Підтримка Firefox можлива лише при заміні Web Speech API на сторонній STT

4.8. План подальшої розробки (roadmap v2.0)

Етап	Функція	Пріоритет	Коментар
1	Розширення словника команд (100+ фраз)	High	Залучення NLP
2	Додавання офлайн-розпізнавання голосу	High	Vosk + WebAssembly
3	Підтримка Firefox та Safari	High	Відмова від Web Speech API

Етап	Функція	Пріоритет	Коментар
4	AI-модуль автоматичного узагальнення тексту	Medium	«Стисло прочитай статтю»
5	Функція макро-команд (скрипти дій)	Medium	«заповни форму і відправ»
6	Інтеграція із системою жестового керування (Webcam ML)	Low	Дослідження
7	Режим навчання користувача («покажи доступні команди»)	High	UX-покращення
8	Публікація у Chrome Web Store	High	Фінальний етап

4.9. Порівняння поточного прототипу та майбутньої версії

Критерій	Прототип v1.0	План v2.0
Голосові команди	Базові (30+)	Повний набір (100+)
Мови	Українська	+ Англ, Польська, Німецька
Offline-режим	Немає	Буде
Робота без семантики	Частково	Повна
Читання - керування	Так	Так + AI-узагальнення
Типи браузерів	Chrome	Chrome, Edge, Firefox, Safari
Підтримка мобільних браузерів	Ні	Після v2.0

4.10. Узагальнені результати розділу 4

1. Розширення пройшло базове UX-тестування та продемонструвало високий рівень зрозумілості та ефективності.
2. Основні технічні обмеження пов’язані з Web Speech API та доменними особливостями HTML-структур.
3. Прототип придатний до використання у реальних умовах для широкого спектра користувачів.
4. Є чіткий план розвитку до повнофункціонального продукту версії 2.0.

ВИСНОВКИ

У результаті виконання магістерської роботи було вирішено науково-прикладне завдання створення браузерного розширення для забезпечення вебдоступності користувачів із порушеннями зору, що поєднує технології голосового керування та синтезу мовлення без необхідності внесення змін у вихідний код сайтів.

Проведений системний аналіз показав, що проблема вебдоступності залишається актуальною навіть за наявності міжнародних стандартів (WCAG 2.1, Section 508, European Accessibility Act). Близько 96 % сучасних сайтів мають критичні помилки у доступності. Виявлено, що в Україні лише державні ресурси частково підпорядковуються вимогам цифрової інклюзії, тоді як приватний сектор здебільшого їх ігнорує. Досліджено існуючі інструменти — NVDA, JAWS, VoiceOver, WAVE, Axe, Read Aloud, HelperBird, ChromeVox — та встановлено, що жоден із них не забезпечує комплексної взаємодії голосом. Screen readers залежні від якості верстки, а браузерні плагіни не дають змоги виконувати навігацію чи натискання елементів через голос.

Узагальнено теоретичні підходи до побудови доступного інтерфейсу та сформульовано принципову наукову проблему: відсутність універсального рішення, здатного забезпечити одночасно озвучення контенту, голосове керування й роботу незалежно від семантики HTML.

Розроблено архітектуру браузерного розширення, що складається з модулів:

1. Content Script — аналіз DOM і визначення елементів взаємодії;
2. Background Service Worker — управління голосовими командами;
3. Voice Module (на основі Web Speech API) — розпізнавання та синтез мовлення;
4. UX-Layer — інтерфейс взаємодії користувача з голосом.

Виконано моделювання системи у вигляді BPMN-схем, UML-діаграм класів та процесів, що дало змогу формалізувати інформаційні потоки між компонентами.

Розширення протестовано на групі з 15 користувачів різних категорій: незрячі, люди з порушеннями моторики, особи похилого віку та контрольна група.

Середня успішність виконання завдань становила 88,5 %**, а** індекс SUS — 87 балів, що відповідає рівню «відмінно» за шкалою юзабіліті.

1. Запропоновано інтегрований механізм голосового керування та озвучення**,** реалізований у межах одного браузерного середовища.
2. Розроблено евристичний алгоритм навігації DOM, який не залежить від семантичної верстки та здатен працювати навіть на «поганих» сайтах.
3. Вперше запропоновано концепцію browser-embedded accessibility layer, що забезпечує доступність без установлення окремого ПЗ.
4. Реалізовано підтримку української мови з природними голосовими командами (типу «читай заголовки», «прокрути вниз», «натисни кнопку»).
5. Показано можливість повністю локальної роботи системи без серверної обробки даних.

Розроблене розширення може бути використане:

- для підвищення доступності державних і освітніх ресурсів;
- як інструмент цифрової інклюзії у соціальних сервісах;
- як базова платформа для досліджень голосової взаємодії у вебсередовищі.

Система не потребує змін у коді сайтів, що спрощує її впровадження у будь-яких організаціях без залучення розробників ресурсу.

Розроблене браузерне розширення підтвердило можливість створення інтелектуальної доступності, коли взаємодія користувача з вебсайтом

здійснюється не через адаптацію сторінки, а через розумне посередництво браузера.

Таким чином, робота демонструє практичне втілення принципів цифрової інклюзії та створює основу для подальшого розвитку національних технологій вебдоступності в Україні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. World Wide Web Consortium (W3C). *Web Content Accessibility Guidelines (WCAG) 2.1.* 5 June 2018. Отримано 2025.URL: https://www.w3.org/TR/WCAG21/ W3C+1

2. W3C. *WCAG 2 Overview.* Web Accessibility Initiative (WAI). Опубліковано 2025. URL: https://www.w3.org/WAI/standards-guidelines/wcag/ W3C

3. WebAIM. *The WebAIM Million – The 2024 report on the accessibility of home pages.* 28 March 2024. URL: https://webaim.org/projects/million/2024 WebAIM+1

4. World Health Organization (WHO). *Blindness and vision impairment: Fact sheet.* 10 August 2023. URL: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment World Health Organization+1

5. WHO. *Increasing eye-care interventions to address vision impairment.* 8 March 2023. URL: https://www.who.int/publications/m/item/increasing-eye-care-interventions-to-address-vision-impairment World Health Organization

6. WebYes. *The State of Website Accessibility 2024 – 7 Stats You Need to Know.* 13 Jan 2025. URL: https://www.webyes.com/blogs/state-of-website-accessibility-2024/ WebYes

7. WebAIM. *Semantic Structure: Regions, Headings, and Lists.* 30 Aug 2024. https://webaim.org/techniques/semanticstructure/

8. Chrome Web Store. *Read Aloud – A Text to Speech Voice Reader.* 2025. https://chrome.google.com/webstore/detail/read-aloud/hdhinadidafjejdhmfkjgnolgimiapl

9. OpenAI. *GPT-4 Technical Report.* San Francisco, 2023. https://cdn.openai.com/papers/gpt-4.pdf

10. Google Chrome Developers. *Extensions Architecture Overview (Manifest V3).* 2024. https://developer.chrome.com/docs/extensions/mv3/architecture/

11. Object Management Group (OMG). *Business Process Model and Notation (BPMN) 2.0.* 2010. Доступ: https://www.omg.org/spec/BPMN/2.0/ OMG

12. Object Management Group (OMG). *Unified Modeling Language (UML) 2.5.1 — Specification.* Грудень 2017. Доступ: https://www.omg.org/spec/UML/2.5.1/ OMG

13. Chrome for Developers. *Manifest V3 — what it is and why it matters.* Документація, 2025. Доступ: https://developer.chrome.com/docs/extensions/develop/migrate/what-is-mv3 Chrome for Developers

14. Chrome for Developers. *Message passing in Chrome extensions (service worker — content scripts ↔ pages).* Документація. Доступ: https://developer.chrome.com/docs/extensions/develop/concepts/messaging Chrome for Developers

15. MDN Web Docs. *Web Speech API (SpeechSynthesis & SpeechRecognition): concepts, usage, compatibility.* Оновлено 2025-09-30. Доступ: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API MDN Web Docs

ДОДАТКИ

Додаток А

```
// Content script - працює на кожній сторінці
class VoiceAssistant {
constructor() {
this.isActive = false;
this.isListening = false;
this.recognition = null;
this.synth = window.speechSynthesis;
this.currentElement = null;
this.navigableElements = [];
this.currentIndex = -1;
this.lastCommand = '';
this.currentZoom = 100;
// Налаштування
this.settings = {
speed: 1.0,
volume: 1.0,
autoRead: false,
language: 'uk-UA'
};
this.init();
}
init() {
// Завантаження налаштувань
this.loadSettings();
// Ініціалізація розпізнавання голосу
this.initSpeechRecognition();
// Створення UI індикатора
this.createIndicator();
// Слухання повідомлень
```

```
this.setupMessageListener();
// Слухання клавіатурних команд
this.setupKeyboardShortcuts();
}
loadSettings() {
chrome.storage.sync.get(
['isActive', 'voiceSpeed', 'voiceVolume', 'autoRead', 'language'],
(result) => {
this.isActive = result.isActive || false;
this.settings.speed = result.voiceSpeed || 1.0;
this.settings.volume = result.voiceVolume || 1.0;
this.settings.autoRead = result.autoRead || false;
this.settings.language = result.language || 'uk-UA';
if (this.isActive) {
this.start();
}
}
);
}
initSpeechRecognition() {
if (!('webkitSpeechRecognition' in window) && !('SpeechRecognition' in window)) {
console.error('Speech recognition not supported');
return;
}
const SpeechRecognition = window.SpeechRecognition ||
window.webkitSpeechRecognition;
this.recognition = new SpeechRecognition();
this.recognition.continuous = true;
this.recognition.interimResults = false;
this.recognition.lang = this.settings.language;
```

```
this.recognition.onresult = (event) => {
const last = event.results.length - 1;
const command = event.results[last][0].transcript.toLowerCase().trim();
console.log('Розпізнано команду:', command);
this.handleCommand(command);
};
this.recognition.onerror = (event) => {
console.error('Recognition error:', event.error);
if (event.error === 'no-speech') {
// Перезапустити розпізнавання
if (this.isActive) {
setTimeout(() => this.recognition.start(), 1000);
}
}
};
this.recognition.onend = () => {
if (this.isActive && this.isListening) {
this.recognition.start();
}
};
}
start() {
this.isActive = true;
this.updateIndicator();
if (this.recognition && !this.isListening) {
try {
this.recognition.start();
this.isListening = true;
// Коротке повідомлення без списку команд
this.speak('Голосовий асистент увімкнено');
```

```
} catch (e) {
console.error('Error starting recognition:', e);
}
}
// Ініціалізація навігаційних елементів
this.updateNavigableElements();
}
stop() {
this.isActive = false;
this.updateIndicator();
if (this.recognition && this.isListening) {
this.recognition.stop();
this.isListening = false;
// Не озвучуємо при вимкненні
}
this.synth.cancel();
}
handleCommand(command) {
// Видалення слів-паразитів та нормалізація
const cleanCommand = command.replace(/^(ок|окей|гаразд|добре|так)\s+/i, '').trim();
// СТОП - найвищий пріоритет! Перевіряється ПЕРШИМ
if (cleanCommand.includes('стоп') || cleanCommand.includes('зупини') ||
cleanCommand.includes('тихо') || cleanCommand.includes('замовкни')) {
this.stopSpeaking();
return; // Одразу виходимо, не перевіряємо інші команди
}
// Команди читання
if (cleanCommand.includes('читай все') || cleanCommand.includes('прочитай все')) {
this.readAll();
}
```

```
else if (cleanCommand.includes('читай заголовки') || cleanCommand.includes('прочитай заголовки')) {
this.readHeadings();
}
else if (cleanCommand.includes('читай посилання') || cleanCommand.includes('прочитай посилання')) {
this.readLinks();
}
else if (cleanCommand.includes('читай') || cleanCommand.includes('прочитай')) {
this.readSelection();
}
// Команди навігації
else if (cleanCommand.includes('наступний') || cleanCommand.includes('далі')) {
this.navigateNext();
}
else if (cleanCommand.includes('попередній') || cleanCommand.includes('назад')) {
this.navigatePrevious();
}
else if (cleanCommand.includes('натисни') || cleanCommand.includes('клік') ||
cleanCommand.includes('відкрий')) {
this.clickCurrent();
}
// Команди прокрутки
else if (cleanCommand.includes('прокрути вниз') || cleanCommand.includes('скрол вниз'))
{
this.scrollDown();
}
else if (cleanCommand.includes('прокрути вгору') || cleanCommand.includes('скрол
вгору')) {
this.scrollUp();
```

```
}
else if (cleanCommand.includes('на початок')) {
this.scrollToTop();
}
else if (cleanCommand.includes('в кінець')) {
this.scrollToBottom();
}
// Команди зуму
else if (cleanCommand.includes('збільш') || cleanCommand.includes('зум плюс') ||
cleanCommand.includes('більше')) {
this.zoomIn();
}
else if (cleanCommand.includes('зменш') || cleanCommand.includes('зум мінус') ||
cleanCommand.includes('менше')) {
this.zoomOut();
}
else if (cleanCommand.includes('скинути зум') || cleanCommand.includes('звичайний
розмір')) {
this.resetZoom();
}
// Команди керування
else if (cleanCommand.includes('вимкни') || cleanCommand.includes('вимкнути асистент')
|| cleanCommand.includes('вийти')) {
this.turnOff();
}
else if (cleanCommand.includes('остання команда') || cleanCommand.includes('яка
остання') || cleanCommand.includes('що я казав')) {
this.showLastCommand();
}
else if (cleanCommand.includes('допомога') || cleanCommand.includes('команди')) {
```

```
this.showHelp();
}
else if (cleanCommand.includes('пошук')) {
this.activateSearch();
}
else if (cleanCommand.includes('повтори')) {
this.repeatLast();
}
// Команди браузера
else if (cleanCommand.includes('історія назад')) {
this.goBack();
}
else if (cleanCommand.includes('історія вперед')) {
this.goForward();
}
else if (cleanCommand.includes('оновити') || cleanCommand.includes('перезавантажити'))
{
this.reloadPage();
}
else if (cleanCommand.includes('нова вкладка')) {
this.openNewTab();
}
else if (cleanCommand.includes('закрити вкладку')) {
this.closeTab();
}
else {
// Не озвучуємо помилку, просто ігноруємо нерозпізнану команду
console.log('Нерозпізнана команда:', cleanCommand);
return;
}
```

```
// Зберегти останню команду для повтору (тільки якщо команда була виконана)
this.lastCommand = cleanCommand;
}
// Функції читання
readAll() {
this.synth.cancel();
const mainContent = this.getMainContent();
const text = this.extractText(mainContent);
if (text) {
this.speak(`Читаю всю сторінку. ${text}`);
} else {
this.speak('Не знайдено текст для читання.');
}
}
readSelection() {
const selection = window.getSelection().toString().trim();
if (selection) {
this.speak(selection);
} else if (this.currentElement) {
const text = this.extractText(this.currentElement);
this.speak(text || 'Елемент не містить тексту.');
} else {
this.speak('Виберіть текст або елемент для читання.');
}
}
readHeadings() {
this.synth.cancel();
const headings = document.querySelectorAll('h1, h2, h3, h4, h5, h6');
if (headings.length === 0) {
this.speak('Заголовки не знайдено.');
```

```
return;
}
let text = 'Заголовки на сторінці: ';
headings.forEach((heading, index) => {
const level = heading.tagName;
const content = heading.textContent.trim();
text += `${level}: ${content}. `;
});
this.speak(text);
}
// Функції навігації
updateNavigableElements() {
this.navigableElements = Array.from(
document.querySelectorAll('a, button, input, select, textarea, [role="button"],
[tabindex="0"]')
).filter(el => this.isVisible(el));
}
navigateNext() {
this.updateNavigableElements();
if (this.navigableElements.length === 0) {
this.speak('Навігаційні елементи не знайдено.');
return;
}
this.currentIndex = (this.currentIndex + 1) % this.navigableElements.length;
this.focusElement(this.navigableElements[this.currentIndex]);
}
navigatePrevious() {
this.updateNavigableElements();
if (this.navigableElements.length === 0) {
this.speak('Навігаційні елементи не знайдено.');
```

```
return;
}
this.currentIndex = this.currentIndex <= 0
? this.navigableElements.length - 1
: this.currentIndex - 1;
this.focusElement(this.navigableElements[this.currentIndex]);
}
focusElement(element) {
if (!element) return;
// Видалити попередню підсвітку
document.querySelectorAll('.voice-assistant-highlight').forEach(el => {
el.classList.remove('voice-assistant-highlight');
});
// Підсвітити поточний елемент
element.classList.add('voice-assistant-highlight');
element.focus();
element.scrollIntoView({ behavior: 'smooth', block: 'center' });
this.currentElement = element;
// Озвучити елемент
const description = this.getElementDescription(element);
this.speak(description);
}
clickCurrent() {
if (this.currentElement) {
this.speak('Натискаю');
setTimeout(() => {
this.currentElement.click();
}, 500);
} else {
this.speak('Спочатку виберіть елемент за допомогою команд "наступний" або
```

```
"попередній".');
}
}
// Функції прокрутки
scrollDown() {
window.scrollBy({ top: window.innerHeight * 0.8, behavior: 'smooth' });
this.speak('Прокручую вниз');
}
scrollUp() {
window.scrollBy({ top: -window.innerHeight * 0.8, behavior: 'smooth' });
this.speak('Прокручую вгору');
}
scrollToTop() {
window.scrollTo({ top: 0, behavior: 'smooth' });
this.speak('Перехід на початок сторінки');
}
scrollToBottom() {
window.scrollTo({ top: document.body.scrollHeight, behavior: 'smooth' });
this.speak('Перехід в кінець сторінки');
}
// Допоміжні функції
stopSpeaking() {
// ОДРАЗУ зупиняємо всі озвучування
this.synth.cancel();
// Не озвучуємо підтвердження, просто зупиняємо
}
showHelp() {
const helpText = `
Доступні команди:
Читай - прочитати вибраний текст.
```

```
Читай все - прочитати всю сторінку.
Читай заголовки - прочитати всі заголовки.
Читай посилання - прочитати всі посилання.
Наступний - перейти до наступного елемента.
Попередній - повернутися до попереднього.
Натисни - натиснути активне посилання.
Прокрути вниз - прокрутити вниз.
Прокрути вгору - прокрутити вгору.
Збільш - збільшити розмір тексту.
Зменш - зменшити розмір тексту.
Скинути зум - повернути звичайний розмір.
Стоп - зупинити читання.
Вимкни - вимкнути асистент.
Остання команда - почути останню команду.
Повтори - повторити останню команду.
Допомога - почути цей список.
`;
this.speak(helpText);
}
activateSearch() {
const searchInput = document.querySelector('input[type="search"], input[name*="search"],
input[name*="q"]');
if (searchInput) {
searchInput.focus();
this.speak('Пошук активовано. Введіть запит.');
} else {
// Відкрити пошук браузера
this.speak('Натисніть Control F для пошуку на сторінці.');
}
}
```

```
goBack() {
history.back();
this.speak('Повертаюсь назад');
}
goForward() {
history.forward();
this.speak('Переходжу вперед');
}
reloadPage() {
this.speak('Перезавантажую сторінку');
setTimeout(() => location.reload(), 1000);
}
readLinks() {
this.synth.cancel();
const links = document.querySelectorAll('a[href]');
if (links.length === 0) {
this.speak('Посилання не знайдено.');
return;
}
let text = `Знайдено ${links.length} посилань: `;
links.forEach((link, index) => {
if (index < 10) { // Перші 10 посилань
const linkText = link.textContent.trim();
if (linkText) {
text += `${index + 1}. ${linkText}. `;
}
}
});
this.speak(text);
}
```

```
// Команди зуму
zoomIn() {
this.currentZoom += 10;
if (this.currentZoom > 200) this.currentZoom = 200;
document.body.style.zoom = this.currentZoom + '%';
this.speak(`Збільшено до ${this.currentZoom} відсотків`);
}
zoomOut() {
this.currentZoom -= 10;
if (this.currentZoom < 50) this.currentZoom = 50;
document.body.style.zoom = this.currentZoom + '%';
this.speak(`Зменшено до ${this.currentZoom} відсотків`);
}
resetZoom() {
this.currentZoom = 100;
document.body.style.zoom = '100%';
this.speak('Звичайний розмір відновлено');
}
repeatLast() {
if (this.lastCommand) {
this.speak('Повторюю останню команду');
setTimeout(() => this.handleCommand(this.lastCommand), 500);
} else {
this.speak('Немає команди для повтору');
}
}
openNewTab() {
this.speak('Відкриваю нову вкладку');
window.open('about:blank', '_blank');
}
```

```
closeTab() {
this.speak('Закриваю вкладку');
setTimeout(() => window.close(), 500);
}
// Вимкнення асистента
turnOff() {
this.speak('Вимикаю голосовий асистент. До побачення!');
setTimeout(() => {
this.stop();
chrome.storage.sync.set({ isActive: false });
}, 2000);
}
// Показати останню команду
showLastCommand() {
if (this.lastCommand) {
this.speak(`Остання команда була: ${this.lastCommand}`);
} else {
this.speak('Ще не було виконано жодної команди');
}
}
// Озвучування
speak(text) {
if (!text) return;
const utterance = new SpeechSynthesisUtterance(text);
utterance.lang = this.settings.language;
utterance.rate = this.settings.speed;
utterance.volume = this.settings.volume;
this.synth.speak(utterance);
}
// Утиліти
```

```
getMainContent() {
// Спроба знайти основний контент
const main = document.querySelector('main, [role="main"], article, .content, #content');
return main || document.body;
}
extractText(element) {
if (!element) return '';
// Видалити скрипти та стилі
const clone = element.cloneNode(true);
clone.querySelectorAll('script, style, nav, header, footer, aside').forEach(el => el.remove());
return clone.textContent.trim().replace(/\s+/g, ' ');
}
getElementDescription(element) {
const tag = element.tagName.toLowerCase();
const text = element.textContent.trim().substring(0, 100);
const ariaLabel = element.getAttribute('aria-label');
const title = element.getAttribute('title');
const alt = element.getAttribute('alt');
let description = '';
if (tag === 'a') {
description = `Посилання: ${text || ariaLabel || title || 'без тексту'}`;
} else if (tag === 'button') {
description = `Кнопка: ${text || ariaLabel || title || 'без тексту'}`;
} else if (tag === 'input') {
const type = element.getAttribute('type') || 'text';
const placeholder = element.getAttribute('placeholder');
description = `Поле вводу ${type}: ${ariaLabel || placeholder || title || ''}`;
} else if (tag === 'img') {
description = `Зображення: ${alt || title || 'без опису'}`;
} else {
```

```
description = text || ariaLabel || title || 'елемент без тексту';
}
return description;
}
isVisible(element) {
const style = window.getComputedStyle(element);
return style.display !== 'none' &&
style.visibility !== 'hidden' &&
style.opacity !== '0' &&
element.offsetWidth > 0 &&
element.offsetHeight > 0;
}
// UI індикатор
createIndicator() {
const indicator = document.createElement('div');
indicator.id = 'voice-assistant-indicator';
indicator.className = 'voice-assistant-indicator';
indicator.innerHTML = `
<div class="indicator-content">
<span class="indicator-icon">□</span>
<span class="indicator-text">Слухаю...</span>
</div>
`;
document.body.appendChild(indicator);
this.indicator = indicator;
this.updateIndicator();
}
updateIndicator() {
if (!this.indicator) return;
if (this.isActive) {
```

```
this.indicator.classList.add('active');
} else {
this.indicator.classList.remove('active');
}
}
// Слухання повідомлень
setupMessageListener() {
chrome.runtime.onMessage.addListener((request, sender, sendResponse) => {
if (request.action === 'toggleAssistant') {
if (request.isActive) {
this.start();
} else {
this.stop();
}
}
if (request.action === 'updateSettings') {
if (request.settings.speed !== undefined) {
this.settings.speed = request.settings.speed;
}
if (request.settings.volume !== undefined) {
this.settings.volume = request.settings.volume;
}
if (request.settings.autoRead !== undefined) {
this.settings.autoRead = request.settings.autoRead;
}
}
if (request.action === 'autoReadPage') {
if (this.settings.autoRead && this.isActive) {
setTimeout(() => this.readAll(), 1000);
}
```

```
}
});
}
// Клавіатурні команди
setupKeyboardShortcuts() {
document.addEventListener('keydown', (e) => {
if (!this.isActive) return;
// Alt + стрілка вправо - наступний елемент
if (e.altKey && e.key === 'ArrowRight') {
e.preventDefault();
this.navigateNext();
}
// Alt + стрілка вліво - попередній елемент
if (e.altKey && e.key === 'ArrowLeft') {
e.preventDefault();
this.navigatePrevious();
}
// Alt + Enter - клік
if (e.altKey && e.key === 'Enter') {
e.preventDefault();
this.clickCurrent();
}
// Escape - зупинити читання
if (e.key === 'Escape') {
this.stopSpeaking();
}
});
}
}
```

```
// Ініціалізація асистента
const voiceAssistant = new VoiceAssistant();
```