МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

АФАНАСЬЄВА ДАР'Я СЕРГІЇВНА

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій
д-р. техн. наук., професор
_____Наталія ВЕСЕЛОВСЬКА

«____»_____ 2026 р.

РОЗРОБКА АДАПТИВНОГО МОДУЛЯ ДОСТУПНОСТІ ДО ІНТЕРНЕТ-РЕСУРСІВ ДЛЯ ОСІБ З ПОРУШЕННЯМИ ЗОРУ ТА КОГНІТИВНОГО СПРИЙНЯТТЯ

Спеціальність 122 Комп'ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
Наталія ВЕСЕЛОВСЬКА, професор кафедри
інформаційних технологій,
д. т. н., професор
_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця - 2026

АНОТАЦІЯ

Афанасьєва Д. С. Розробка адаптивного модуля доступності до інтернет-ресурсів для осіб з порушеннями зору та когнітивного сприйняття. Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні науки». Донецький національний університет імені Василя Стуса. Вінниця, 2026.

У бакалаврській роботі розглянуто проблему доступності сайтів для людей із порушеннями зору та когнітивними особливостями. Проаналізовано чинні стандарти інклюзивної побудови вебсайтів і порівняно популярні готові рішення. На основі результатів спроєктовано та реалізовано програмний модуль, який вбудовується у сайт і допомагає користувачеві самостійно адаптувати інтерфейс під свої потреби: змінювати шрифт і контраст, виділяти фокус читання, прослуховувати вміст уголос тощо. Модуль перевірено на відповідність сучасним стандартам доступності.

Ключові слова: вебдоступність, інклюзивний дизайн, порушення зору, когнітивні особливості, дислексія, адаптивний модуль.

ABSTRACT

Afanasieva D. S. Development of an adaptive accessibility module for internet resources for persons with visual impairments and cognitive perception disabilities. Speciality 122 «Computer Science», Educational programme «Computer Science». Vasyl' Stus Donetsk National University. Vinnytsia, 2026.

The bachelor's work addresses the problem of website accessibility for people with visual impairments and cognitive disabilities. Current accessibility standards are analysed, and popular ready-made solutions are compared. On this basis, a software module is designed and implemented that embeds into a website and helps the user adapt the interface to their own needs: changing the font and contrast, highlighting the reading focus, listening to the content aloud, etc. The module is verified for compliance with modern accessibility standards.

Keywords: web accessibility, inclusive design, visual impairment, cognitive disabilities, dyslexia, adaptive module.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Більша частина соціальних сфер, як-от освіта, медицина, фінансові й державні послуги, поступово переходить у цифровізацію та формує повністю цифрове середовище. Для великої кількості користувачів цей перехід відкриває нові можливості, проте для людей з порушеннями зору та особливостями когнітивного сприйняття бар'єри в інтерфейсі нерідко означають фактичну неможливість скористатися послугою. Якщо всі вони повністю перейдуть до онлайн-формату, то таким користувачам доступ буде фактично заблокований.

Незважаючи на існування міжнародних стандартів вже понад двадцять років, реальний стан вебсередовища лишається проблемним. Звіт WebAIM Million 2025 фіксує, що 95,9% з мільйона найвідвідуваніших сторінок містять автоматично виявлені помилки доступності, із середнім значенням 56,1 порушення на сторінку [1]. Дослідження Програми розвитку ООН в Україні виявило, що 97 зі 100 перевірених сайтів державних органів мають деякі бар'єри у доступності для користувачів [2].

Створення готових віджетів для доступності контенту вже мало спроби впровадження, але не завжди виправдовувало сподівання, покладені на них. Наприклад, у 2025 році Федеральна торговельна комісія США наклала на одну з компаній штраф у розмірі мільйона доларів за оманливу рекламу автоматичного приведення сайту у відповідність до стандарту WCAG [3]; інша стала відповідачем у колективному позові за подібними підставами [4]. Водночас сама ідея програмного шару, який допомагає користувачеві адаптувати вже відрендерену сторінку під свої потреби, залишається актуальною, але за умови коректної архітектури, прозорого коду та реальної відповідності стандартам.

Актуальність роботи зумовлюється: ухваленням постанови Кабінету Міністрів України № 757 від 2023 року та чинним ДСТУ EN 301 549:2022, які роблять доступність обов'язковою для державних ресурсів [5, 6]; низьким рівнем відповідності реальних сайтів стандарту WCAG 2.2 [7]; обмеженнями та репутаційними ризиками наявних готових рішень; відсутністю у відкритому

просторі модуля, що поєднував би засоби візуальної адаптації, когнітивної підтримки й озвучення контенту в одному компоненті і при цьому був доступний усім охочим без сплати коштів.

Мета дослідження – розробити адаптивний модуль доступності до інтернет-ресурсів для осіб з порушеннями зору та когнітивного сприйняття.

Завдання дослідження. Для досягнення поставленої мети необхідно:

- проаналізувати проблему доступності інтернет-ресурсів для цільових категорій користувачів та систематизувати, які основні характерні бар'єри ми спостерігаємо;
- розглянути міжнародні та національні стандарти вебдоступності й вимоги до безпеки модулів, що вбудовуються у сторонні сайти;
- виконати порівняльний аналіз існуючих рішень;
- обґрунтувати вибір технологічного стеку, спроєктувати архітектуру та інтерфейс модуля;
- реалізувати компоненти візуальної адаптації, когнітивної підтримки, та синтезу мовлення;
- забезпечити безпеку й ізоляцію модуля від хост-сайту та перевірити його відповідність WCAG 2.2.

Об'єкт дослідження – процес забезпечення доступності інтернет-ресурсів для користувачів з порушеннями зору та когнітивного сприйняття.

Предмет дослідження – методи, моделі та програмні засоби для побудови вбудовуваного модуля доступності, що відповідають сучасним стандартам.

Методи дослідження. У роботі поєднано теоретичні методи, як-от аналіз і систематизація джерел, порівняльний аналіз стандартів і програмних рішень, моделювання архітектури та більш практичні (тестування відповідності, аудит контрастності й структурної семантики, юзабіліті-перевірка з користувачами).

Практичне значення одержаних результатів. Розроблений модуль придатний для інтеграції у сайти різних типів без внесення змін до серверної частини. Закладена архітектура дозволяє в подальшому масштабувати рішення в окремий мікросервіс і в мобільні застосунки, аби можна було покрити набагато

ширший спектр інтернет-ресурсів. Модуль може бути використаний власниками публічних сайтів, освітніми платформами та органами державної влади.

Апробація результатів дослідження. Основні положення та результати роботи апробовано у трьох наукових публікаціях:

1. Афанасьєва Д. С., Січко Т. В. Аналітичне дослідження бар'єрів доступності вебресурсів для осіб з порушеннями зору та когнітивними особливостями. Вимірювальна та обчислювальна техніка в технологічних процесах, номер 2, 2026. [8].
2. Афанасьєва Д. С., Веселовська Н. Р. Алгоритмічні методи адаптації вебконтенту для людей з вадами зору: тези доповіді. VI Всеукраїнська науково-практична конференція «Комп'ютерні технології обробки даних», 2025. [9].
3. Афанасьєва Д. С., Веселовська Н. Р. Використання методів структурного аналізу вебресурсів для адаптації до когнітивних особливостей користувачів. Вісник Донецького національного університету імені Василя Стуса, 2026. [10].

Структура роботи. Робота складається зі вступу, трьох розділів, висновків, переліку використаних посилань і додатків. Перший розділ присвячений аналізу предметної області, постановці задачі та обґрунтуванню вибору технологій, другий – проєктуванню архітектури модуля, третій – програмній реалізації та тестуванню. Робота містить 58 сторінок основного тексту, 37 рисунків, 2 таблиці; перелік джерел налічує 41 найменування.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз проблеми доступності інтернет-ресурсів для осіб з порушеннями зору та когнітивного сприйняття

Поняття вебдоступності описує властивість інтернет-ресурсів, яка дає змогу використовувати їх людям з різними фізичними, сенсорними, когнітивними та мовними особливостями. У межах роботи над модулем інтерес зосереджено на особах із порушеннями зору та когнітивними особливостями сприйняття, оскільки саме для них бар'єри проявляються найчастіше й мають системний характер.

Масштаб проблеми було проаналізовано з відкритих статистичних даних. За даними Всесвітньої організації охорони здоров'я, у світі живе щонайменше 2,2 мільярда осіб з порушеннями зору різного ступеня – від короткозорості до повної сліпоти [1]. Євростат у 2024 році оцінив частку дорослого населення країн ЄС з певною формою інвалідності у 23,9 % [2]. Якщо копнути глибше, то у віковій когорті осіб понад 85 років цей показник сягає вже аж 72,3 %. Це підтверджує значний скачок кількості людей з нейрокогнітивними порушеннями, що пов'язані з процесом старіння [2]. Окрему велику групу формують користувачі з дислексією, розладом дефіциту уваги та гіперактивності (РДУГ), розладами аутистичного спектра, тривожними розладами та посттравматичним стресом – їхні потреби описано у настановах робочої групи W3C COGA [11].

У сукупності ці категорії є значною частиною аудиторії будь-якого публічного інтернет-ресурсу. Але стан реального вебсередовища лишається проблемним. Щорічний звіт WebAIM Million 2025 зафіксував автоматично виявлені порушення на 95,9 % з мільйона найвідвідуваніших сторінок, із середнім значенням 56,1 помилки на одну сторінку [1]. Звіт Програми розвитку ООН в Україні за 2023–2024 роки засвідчив, що 97 зі 100 перевірених сайтів державних органів мають значні проблеми з доступністю для більш вразливих верств населення [2]. Опитування WebAIM Screen Reader User Survey #10

показало, що користувачі програм зчитування з екрана й сьогодні стикаються з проблемами навіть на найбільш популярних ресурсах [12].

Стандарт WCAG 2.2 (Web Content Accessibility Guidelines) систематизує бар'єри відповідно до чотирьох принципів: сприйнятність (Perceivable), операбельність (Operable), зрозумілість (Understandable) та надійність (Robust) [7]. Перший принцип вимагає, щоб контент і компоненти інтерфейсу могли бути сприйняті органами чуттів або їхніми технічними замінниками. Принцип операбельності зобов'язує забезпечувати керованість всіх функцій із різних засобів введення. Зрозумілість фокусується на чіткості мови, передбачуваності навігації та підтримці користувача під час помилок. Надійність гарантує сумісність із наявними та майбутніми допоміжними технологіями.

1.1.1 Бар'єри для осіб з порушеннями зору

Категорія користувачів з порушеннями зору доволі неоднорідна, тому одні й ті самі особливості інтерфейсу можуть бути критичними для одного підкласу і нейтральними для іншого. Користувачі з повною сліпотою, наприклад, взаємодіють із сайтом переважно через програми зчитування з екрана: JAWS, NVDA, VoiceOver або TalkBack. За даними WebAIM Screen Reader User Survey #10, NVDA лідирує за поширеністю використання (65,6 %), але при цьому VoiceOver використовується у 70,6 % саме на мобільних девайсах, що робить його важливим саме для цієї ніші [12]. Для користувачів такого типу критичними є наявність текстових альтернатив для зображень (alt text), коректна семантична структура сторінки, доступність елементів керування з клавіатури, чіткий порядок читання й розрізнюваність посилань за текстом.

Для слабкозорих користувачів значно пріоритетнішими стають недостатній контраст між текстом і фоном, неможливість масштабувати текст без розриву компонування, дрібний шрифт у функціональних елементах, відсутність видимого фокуса під час навігації клавіатурою тощо. Стандарт WCAG 2.2 встановлює мінімальний коефіцієнт контрасту 4,5:1 для звичайного тексту і 3:1 для великого або жирного тексту (рівень AA) [7]. Часто саме недостатня контрастність роками лідирує серед усіх автоматично виявлених порушень.

Користувачі з порушеннями кольоросприйняття (різні форми дальтонізму), страждають від інтерфейсів, у яких інформація передається виключно через колір. Прикладом може бути, коли при заповненні форми обов'язкові поля відмічаються лише червоною рамкою без додаткової текстової вказівки над або під ними. За таких умов людина не зможе зрозуміти, де саме вона зробила помилку, а також яку саме. Згідно з критерієм 1.4.1 стандарту WCAG 2.2, колір не має бути єдиним способом передачі інформації [7].

1.1.2 Бар'єри для осіб з когнітивними особливостями

Когнітивні бар'єри традиційно отримують менше уваги, ніж зорові, хоча охоплюють надзвичайно широкий спектр людей: дислексію, РДУГ, розлади аутистичного спектра, тривожні розлади, нейрокогнітивні зміни, пов'язані зі старінням, наслідки травм. В контексті життя українського населення на поточний момент кількість людей з такими особливостями значно збільшилася за останні роки. Кожна з попередньо вказаних когнітивних особливостей висуває власні, а нерідко й суперечливі вимоги до інтерфейсу [11]. Узагальнення, запропоноване настановами W3C COGA, виокремлює кілька типових проблем.

- Надмірна складність мовлення й відсутність варіанта простої мови – критичний бар'єр для осіб з дислексією, розладами навчання та некорінних мовців.
- Непослідовні навігаційні структури та різке їх перевизначення між сторінками є джерелами дезорієнтації для користувачів з РДУГ та аутистичним спектром.
- Відволікаючий контент: автоматично відтворювані анімації, миготливі елементи, спливні сповіщення – значно порушують концентрацію уваги.
- Брак чітких підказок під час заповнення форм та недостатні повідомлення про помилки.
- Перевантаженість контенту, відсутність ясної ієрархії заголовків і прогресивного розкриття інформації значно підвищують когнітивне навантаження.

Для частини користувачів з дислексією приходить на допомогу типографіка: збільшені міжрядкові інтервали, помірна довжина рядка (60–70 символів), уникнення вирівнювання за обома краями. Також можна використовувати спеціальні шрифти. Наприклад, Martin Pysny розробив шрифт Dysfont [13], що має особливу структуру літер, завдяки чому люди з дислексією мають можливість сприймати контент набагато легше. Прикладом його успішного використання є додаток Promova. Це EdTech-додаток з вивчення мов, що має українське походження. Його команда створила спеціальний режим для дислектиків, аби більша частина аудиторії мала доступ до отримання знань без бар'єрів.

Британською асоціацією дислексії було сформовано відповідні рекомендації щодо типографічної доступності [14]. На додаток до попередньої інформації, окремі дослідження також випробовували інші спеціалізовані шрифти, зокрема OpenDyslexic [15]. Цей шрифт також доступний у режимах для дислектиків у конкурентів (мова про них буде йти у розділі 1.3). У будь-якому разі надання вибору користувачеві відповідає принципу гнучкості інклюзивного дизайну.

Окремий клас бар'єрів, які чомусь часто не враховують, становлять моторні особливості, що супроводжують когнітивні розлади: тремор, гіперкінези, обмежена точність вказівних пристроїв. Для таких користувачів важливо, щоб елементи були як великі цілі, були впроваджені достатні відступи між інтерактивними елементами, була можливість збільшити курсор та інші. Низка цих вимог увійшла в WCAG 2.2 як нові критерії успішності рівня AA, додані до версії 2.1 [7].

1.1.3 Соціальний та правовий контекст

Доступність інтернет-ресурсів є не лише етичною, а й юридичною вимогою. Європейський Союз ухвалив Директиву про доступність вебсайтів і застосунків публічного сектору ще у 2016 році, поклавши на держави-члени обов'язок узгоджувати національне законодавство з нормами WCAG [16]. На рівні стандартизації Європейський інститут стандартів електронних

телекомунікацій ETSI прийняв EN 301 549, що описує функціональні вимоги до доступності продуктів та послуг ІКТ. Саме цей документ слугує спільним технічним базисом європейських ринків. У 2025 році WCAG 2.2 додатково отримав статус міжнародного стандарту ISO/IEC 40500:2025 [7].

В Україні ключовим стало ухвалення постанови Кабінету Міністрів № 757 від 21 липня 2023 року «Деякі питання доступності інформаційно-комунікаційних систем та документів в електронній формі», яка зобов'язує органи виконавчої влади приводити свої інформаційно-комунікаційні системи у відповідність до ДСТУ EN 301 549:2022 [5, 6]. Як національний стандарт, документ дублює європейський EN 301 549:2022 і спирається на технічні критерії WCAG 2.1 [17]. На приватний сектор зобов'язання поки що формально не поширюється, однак законопроєктна робота в цьому напрямі триває, а кількість позовів щодо недоступних сайтів у країнах ЄС стабільно зростає [10].

Але варто сказати, що наразі наявна прірва між цими нормативними вимогами та фактичним станом вебресурсів. Адже в загальному контексті ці норми були прийняті не так давно, а велика кількість платформ вже існувала на ринку ще до них. Саме такий стан речей формує попит на розроблення спеціалізованих технічних засобів адаптації (як на стороні браузера користувача, так і на стороні сайту). Одночасно з цим характер бар'єрів стабільно відтворюється на ресурсах різного типу.

1.2 Огляд міжнародних та національних стандартів вебдоступності

Нормативна база вебдоступності будується на кількох рівнях. Базовий рівень утворюють настанови WCAG [7], що визначають критерії успішності для контенту й інтерфейсів. Другий рівень – специфікація Accessible Rich Internet Applications (WAI-ARIA) [18], що додає семантичні маркери для динамічних компонентів. Третій рівень – це стандарти на рівні держави або регіонального об'єднання. Четвертий – підзаконні акти, що зобов'язують конкретні організації застосовувати ці стандарти.

1.2.1 Web Content Accessibility Guidelines (WCAG)

WCAG є основним міжнародним стандартом доступності, що його підтримує консорціум World Wide Web (W3C) у межах ініціативи Web Accessibility Initiative (WAI). До діючих редакцій належать 2.0 (2008), 2.1 (2018) та 2.2 (2023). Останні дві версії повністю зворотно сумісні з 2.0, тобто відповідність WCAG 2.2 одночасно означає відповідність 2.1 і 2.0 [7, 17].

Як було вказано раніше, стандарт організовано за чотирма принципами. Кожен принцип розгалужується у керівні принципи, які, своєю чергою, конкретизуються через критерії успішності. Кожному з них присвоюється один з трьох рівнів відповідності: A (мінімальний), AA (розширений) та AAA (найвищий) [7]. У більшості правових систем мінімально прийнятним рівнем для публічних і комерційних сайтів вважається AA.

WCAG 2.2 додав дев’ять нових критеріїв порівняно з 2.1. Їх переважна частина розширює підтримку користувачів зі зниженою моторикою та когнітивними особливостями: вимоги до видимості фокуса, мінімальних розмірів інтерактивних цілей, спрощених механізмів автентифікації, послідовності інтерактивних компонентів [7]. Жоден з нових критеріїв не послаблює попередні вимоги, проте добре проєктований модуль доступності повинен враховувати їх з самого початку, оскільки саме версія 2.2 з 2025 року отримала статус міжнародного стандарту [7].

1.2.2 WAI-ARIA та керівництво з реалізації шаблонів

Специфікація WAI-ARIA, у чинній редакції 1.2, розширює семантичні можливості HTML за рахунок додаткових атрибутів. До них входять ролі (role), стани і властивості (aria-*) [18]. Основним призначенням даної специфікації є описати функціональну роль елемента інтерфейсу для допоміжних технологій, навіть коли візуальне представлення не використовує семантичних HTML-тегів. Без коректного застосування ARIA сучасні динамічні інтерфейси типу каруселей, модальних вікон, складних меню, вкладок залишаються невидимими або неоднозначними для програм зчитування з екрана. Доповненням до самої специфікації слугує ARIA Authoring Practices Guide (APG) – практичне

керівництво з реалізації типових шаблонів інтерфейсу [19]. Його використання можливе, наприклад, для модального вікна.

1.2.3 Настанови COGA

Робоча група Cognitive and Learning Disabilities Accessibility Task Force консорціуму W3C веде окремий потік документів, орієнтованих на користувачів з когнітивними особливостями. Основний документ – Making Content Usable for People with Cognitive and Learning Disabilities. Він загальнює принципи проєктування, патерни інтерфейсу та сценарії користувача [11]. На відміну від WCAG, COGA пропонує переважно якісні рекомендації: проста мова, послідовна навігація, прогресивне розкриття, контекстна допомога, передбачуваність, зменшення відволікаючих факторів та ін. У межах розробки адаптивних систем з настанов COGA безпосередньо випливають режим маски фокусу читання, режим концентрації та можливість приховати анімації відповідно до уподобання користувача.

Для візуального відображення стандартів та розуміння їх статусу для України було сформовано таблицю 1.1.

1.3 Огляд та порівняльний аналіз існуючих рішень

Ринок готових засобів вебдоступності протягом останнього десятиліття сформував окремий сегмент модулів безбар’єрності. Спільна особливість цих рішень полягає у тому, що власник сайту вбудовує в розмітку сторонній JavaScript-сценарій, який під час завантаження сторінки додає на DOM хост-сайту панель адаптації та виконує одно- або багатошарові модифікації самого DOM. Для огляду обрано чотири продукти: UserWay, accessiBe (продукт accessWidget), EqualWeb та AudioEye.

1.3.1 UserWay

UserWay (рис. 1.1) позиціонується як інструмент, що забезпечує відповідність сайту вимогам різних стандартів [20]. Також в ньому інтегровано додатково штучний інтелект. Технічно UserWay вбудовується в сайт через сторонній сценарій, формує плаваючу кнопку доступу, натискання якої відкриває

панель з налаштуваннями шрифту, контрасту, фокуса, лінз для дислексії, припинення анімацій та аналогічних функцій.

Таблиця 1.1 – Порівняльна таблиця ключових стандартів і рекомендацій вебдоступності

Документ	**Сфера дії**	**Статус для України**
WCAG 2.2	Загальна побудова контенту та інтерфейсу вебсайтів. Розроблено відповідні рівні доступності A/AA/AAA.	Базис ДСТУ EN 301 549:2022
WAI-ARIA 1.2	Визначення семантичних ролей та станів для динамічних компонентів.	Застосовується як базис для розробників
ARIA APG	Готові практичні шаблони реалізації.	Є інженерною практикою
COGA	Когнітивна доступність контенту.	Рекомендаційний
EN 301 549	ІКТ-продукти, послуги, державні закупівлі ЄС.	Прийнятий як ДСТУ
ДСТУ EN 301 549:2022	Те ж саме на території України	Чинний національний стандарт
Постанова КМУ № 757	Інформаційно-комунікаційні системи органів влади.	Обов’язковий підзаконний акт
EU Web Accessibility Directive	Сайти і застосунки публічного сектору	Орієнтир для адаптації законодавства

Сильні сторони, які можна винести з огляду: наявна локалізація багатьма мовами, простота інтеграції, є безкоштовний тестовий період (на сайті доступний також безкоштовний швидкий скан вебсайтів, проте є підозра, що він більше маркетинговий і заточений на продаж, а не на точне виявлення недосконалостей) і загалом протестувати роботу модуля можна одразу на сайті, наявність профілів доступності для різних категорій проблем користувачів. Серед мінусів було виявлено залежність від коректності семантики хост-сайту: оскільки програми зчитування з екрана опрацьовують вихідний DOM на момент завантаження сторінки, додані пізніше зміни UserWay не впливають на акустичне представлення для користувача screen reader. Також під час дослідження виявлено інформацію, що у 2024 році проти UserWay подано колективний позов із заявами про оманливу рекламу можливостей продукту [22].

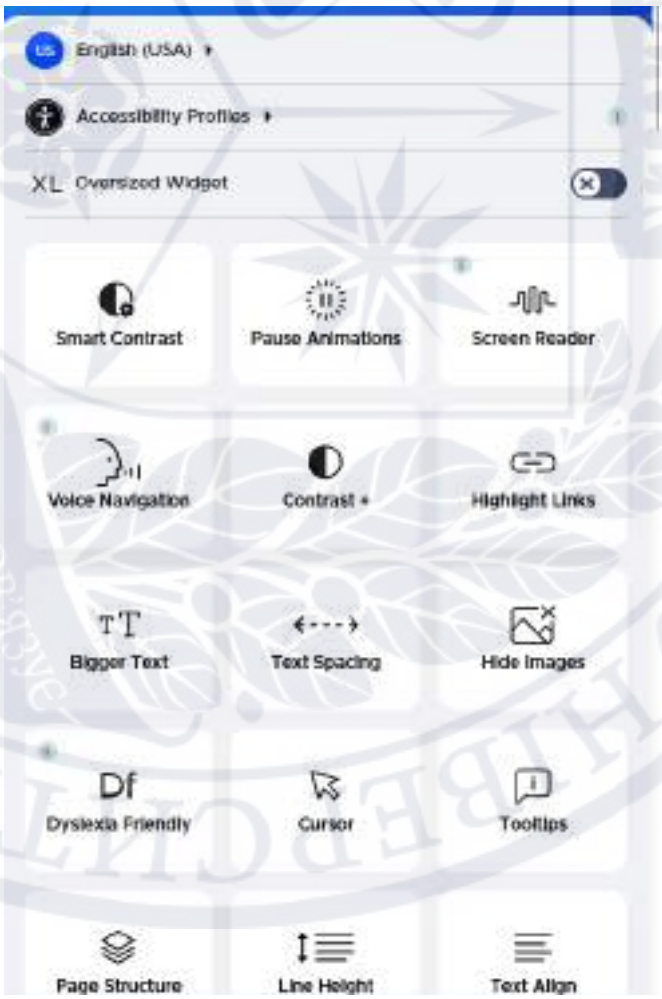

Рисунок 1.1 – Вигляд модуля доступності UserWay

1.3.2 accessiBe (accessWidget)

Продукт компанії accessiBe (рис. 1.2) тривалий час був одним із найпомітніших на ринку завдяки активній маркетинговій кампанії, що обіцяла повну автоматизовану відповідність WCAG [21]. Проте у січні 2025 року Федеральна торговельна комісія США наклала на компанію штраф у 1 млн доларів за оманливі твердження стосовно ефективності її продукту, заборонивши формулювати маркетингові обіцянки як гарантію відповідності стандарту [3]. Умови регулятивного припису забороняють компанії використовувати у рекламі заяви про автоматичне забезпечення повної відповідності і вимагають розкривати всі форми оплачених відгуків. Цей кейс став прецедентом для всієї галузі. Але, повертаючись до функціоналу, він дуже схожий на той, що був у попередньому продукті, але тут ключовим є те, що цей функціонал інтегрується саме на білдерах чи CMS-системах для створення вебсайтів.

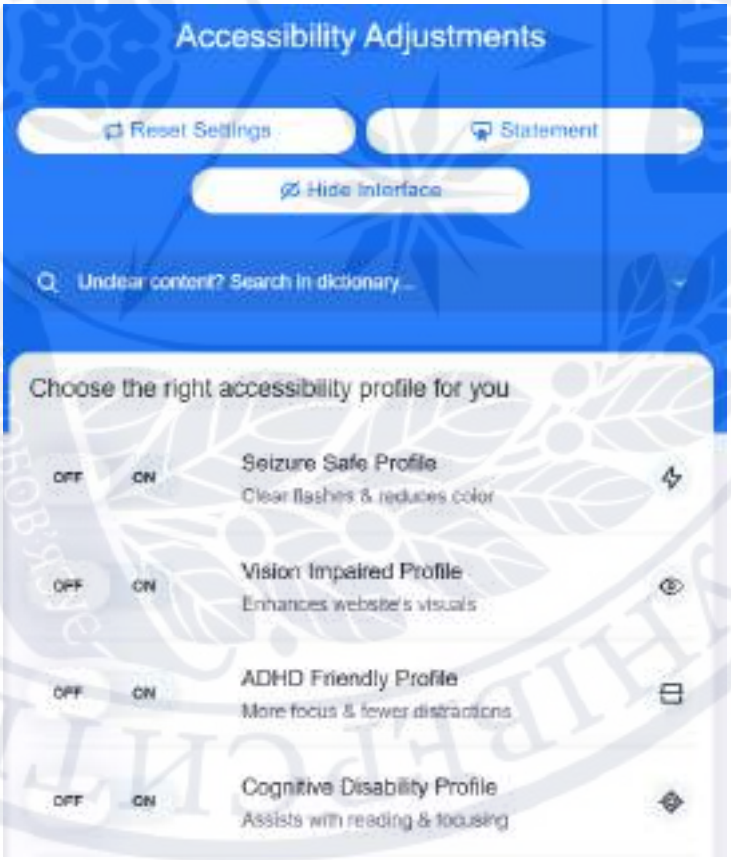

Рисунок 1.2 – Вигляд модуля доступності accessiBe

1.3.3 EqualWeb

Ще одним продуктом з цієї категорії є EqualWeb (рис. 1.3) [22]. Він пропонує гібридну модель автоматизованого overlay з ручним та ШІ аудитом і подальшим виправленням сайту експертами. Сильна сторона EqualWeb – наявність ручної експертизи, саме це відрізняє її від інших продуктів. У рішення доволі мінімалістичний інтерфейс, що легко сприймається оком. Є окрема панель для покращення навігації, адже функціонал дійсно доволі обширний. Якщо казати про обмеження, то у програм такого типу читання з екрану, які аналізують DOM до виконання, можуть виникати конфлікти; введення додаткових скриптів на основі штучного інтелекту може дещо уповільнити час завантаження. Також питання як ШІ функції сприймають люди старшого віку, що з більшою вірогідністю не знайомі з ними.

Рисунок 1.3 – Вигляд модуля доступності EqualWeb

1.3.4 AudioEye

AudioEye (рис. 1.4) пропонує комбінований підхід, у межах якого автоматизована платформа фіксує порушення, після чого частина з них виправляється спеціалістами у вихідному коді сайту, а решта – компенсується програмним шаром [23]. Сильні сторони – інтеграція з різноманітними CMS, обширна звітність, моніторинг, а з обмежень – висока вартість, залежність від хмарного бекенду провайдера. Варто зазначити, що продукт має доволі мінімалістичний, приємний інтерфейс, що має спільні риси з продуктами UserWay (рис. 1.1) та EqualWeb (рис. 1.3). Тому схожий тип панелі буде інтегровано у реалізованому модулі, адже, якщо є схожості у декількох популярних рішень, то напрошується зробити висновок, що це доволі ефективно і вже протестовано на великій кількості аудиторії. Це працює як з сайтами, коли загальна структуруа їх побудови є схожою, аби не плутати користувачів. Такий підхід має бути доволі виграшним і сприяти зменшенню відсотка неприйняття від юзерів.

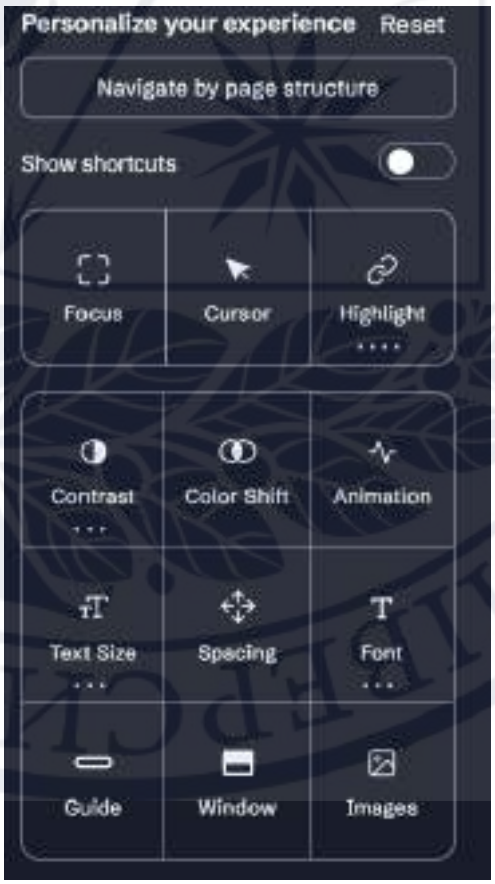

Рисунок 1.4 – Вигляд модуля доступності AudioEye

1.3.5 Порівняльний аналіз

Аби порівняти наявні на ринку продукти, було обрано наступні критерії: технологічну модель, модель ліцензування, відкритість коду, локалізацію (зокрема українську мову), сумісність з програмами зчитування з екрана, ризик ускладнення доступу для користувачів допоміжних технологій. Результат порівняння сформовано в таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика готових рішень

	UserWay	**accessiBe**	**EqualWeb**	**AudioEye**
Технологічна модель	Overlay	Overlay	Overlay + ручний аудит	Overlay + remediation
Відкритий код	Ні	Ні	Ні	Ні
Безкоштовний доступ	Так (обмежений)	Ні	Так (обмежений)	Ні
Українська локалізація	Так	Так	Так	Не було знайдено
Модифікація вихідного DOM	Так	Так	Так	Так
Конфігурованість для розробника	Низька	Низька	Середня	Середня

З порівняльної таблиці 1.2 можемо зробити висновок, що популярні рішення не задовольняють одночасно усім вимогам, але самі рішення дуже сильні самі по собі. Усі чотири продукти будуються у формі додаткового шару, тобто намагаються виправити недоступність хост-сайту з зовнішнього шару. Це підтверджує потребу в альтернативі, що залишається на рівні допоміжного інструменту користувача, не претендує на автоматичне приведення до стандарту

і працює прозоро, з відкритим кодом і чіткими обмеженнями. Також потреби людей з такими особливостями є обов’язковими до інтегрування у будь-які ресурси, тому варіант безкоштовного сервісу з повним об’ємом функціоналу є дуже важливим і затребуваним як для бізнесів, так і для самих юзерів.

1.4 Постановка задачі та формулювання вимог до модуля

Узагальнення проведеного аналізу дозволяє сформулювати наступну задачу так: спроєктувати й реалізувати програмний модуль адаптивного шару доступності, який інтегрується у сторонній вебсайт, надає користувачеві з порушеннями зору або когнітивними особливостями інструменти для самостійної адаптації інтерфейсу до своїх потреб, не порушує безпекових політик хост-сайту і не претендує на автоматичне приведення сайту у відповідність до стандарту.

Функціональні вимоги. Модуль повинен забезпечити такі функції:

- масштабування шрифту, міжрядкового інтервалу та ширини читання;
- налаштування контрастності (зокрема його збільшення, підвищення насиченості та яскравості, створення режиму темного контрасту);
- адаптацію типографіки для дислексії та інших потреб: вибір спеціалізованих гарнітур за можливості, збільшені інтервали між літерами, обмеження довжини рядка;
- збільшення курсора та підсвічування активних інтерактивних елементів;
- маску фокусу та спеціальної гайд-лінії для читання та режим концентрації, що приховує відволікаючі регіони;
- синтез мовлення для тексту, обраного користувачем;
- введення голосових команд;
- блокування автоматичних анімацій та програвання GIF;
- збереження налаштувань між сеансами та можливість їх скинути;
- можливість обрати спеціалізований режим з вже підібраними налаштуваннями для конкретних потреб;
- виклик панелі через плаваючу кнопку та клавіатурне скорочення.

Нефункціональні вимоги. Серед нефункціональних характеристик модуля визначено такі:

- відповідність WCAG 2.2 рівня AA;
- ізоляція стилів і скриптів від хост-сайту через Shadow DOM;
- сумісність із політиками та з механізмом Subresource Integrity;
- кросбраузерна підтримка останніх двох версій Chrome, Firefox, Safari, Edge;
- відкритість коду під дозвільною ліцензією для подальшого розвитку у бік окремого мікросервісу і вбудовуваного компонента для мобільних застосунків.

Окремою важливою вимогою є доступність самого модуля. Інакше ми отримаємо зворотний парадокс, коли інструмент доступу до контенту створює новий бар'єр. Тому панель має мати можливість, хоч частково, бути керованою клавіатурою, мати видимий фокус і виділятися на фоні інших елементів, мати коректну семантику, стани та повідомлення про зміни налаштувань.

Цільова аудиторія модуля складатиметься з трьох основних груп. Перша – кінцеві користувачі з порушеннями зору та когнітивними особливостями. Для них модуль є інструментом самостійної адаптації. Друга – власники публічних сайтів, освітніх платформ, які прагнуть надати своїм відвідувачам додаткові засоби адаптації. І останні – розробники, які можуть використовувати модуль як основу для подальшого розширення в окремий мікросервіс чи в мобільні застосунки.

Загрози, з якими може стикнутися будь-який зовнішній JavaScript-модуль, ризикують значно пошкодити його роботу чи призвести до зливу даних. Тому варто враховувати їх при побудові. До прикладу, атаки на ланцюг постачання. Якщо модуль доставляється з мережі поширення контенту, компрометація провайдера або підміна файлу може спричинити вбудовування шкідливого коду на всі сайти, що ним користуються. Також варто не забувати про коректну ізоляцію. Адже не виключено, що модуль може зламати верстку сайту, або, навпаки, дозволити хосту перевизначити вигляд панелі модуля. І, звісно,

порушення приватності. Якщо модуль під час роботи збиратиме дані поведінки користувача і надсилатиме їх стороннім сервісам, це само по собі здатне викликати бар’єр довіри.

1.5 Обґрунтування підходу до реалізації та вибору технологічного стеку

Після аналізу наявних рішень та чітко фіналізованої структури логічно перейти до дуже важливого питання: як саме розгорнути модуль і на якому стеку його збирати.

З огляду на наявну базу, що відома, розглядаються три можливі варіанти: вбудовуваний JavaScript-віджет, браузерне розширення і компонент для конкретного фреймворку.

Першим варіантом є вбудовуваний JavaScript-віджет. Власник сайту буде додавати у HTML єдиний тег <script>, який підвантажує файл модуля. Скрипт під час завантаження створює плаваючу кнопку і панель. Це найбільш універсальна модель, адже вона не потребує серверної частини. Стек для написання значно менший і потребує знань, що вже наявні. Також цей підхід є незалежним від сторонніх CMS-плагінів і може працювати зі статичними сайтами або односторінковими застосунками. Але варто зазначити, що модуль виконується у тому ж скриптовому контексті, що й хост. Тут може виникнути потреба в продуманій ізоляції стилів та узгодженні з Content Security Policy (CSP) адаптованого сайту.

Другим варіантом є браузерне розширення. Воно працює незалежно від конкретного сайту, тому може адаптувати будь-який ресурс, навіть той, де власник не передбачав такої можливості. Його легше поширювати серед користувачів, які не хочуть розбиратися в кодовій частині сайту, а потребують уже готового рішення тут і зараз. Але тут виникає кілька проблем. Різні браузери мають різні моделі дозволів, політики, принципи створення та інтегрування розширень. Для цього треба мати відповідний досвід. Цей варіант розглядався як першочерговий при виборі тематики дипломної роботи, але під час глибшого дослідження дана опція була менш вдалою за інші. До того ж, наприклад, аби

стати розробником та дистриб'ютором розширень в Chrome треба сплатити комісію за реєстрацію. Це також сильно вплинуло на вибір підходу до реалізації модуля.

Останнім варіантом було розглянуто компонент для конкретного фреймворку. За такого підходу модуль має постачатися як npm-пакет, який інтегрується через шаблон і управляється станом застосунку. За такого підходу можна отримати доволі глибоку інтеграцію з життєвим циклом продукту, доступ до маршрутизатора і сховища. Але окремий фреймворк передбачає жорстку прив'язку до однієї технології, що виключає інтеграцію зі статичними сайтами й платформами для створення вебсайтів на кшталт WordPress (білдери вебсайтів, як їх ще називають).

Якщо зіставити усі підходи, то оптимальною моделлю на даному етапі поставленої задачі є вбудовуваний JavaScript-віджет. Він поєднує універсальність і просту інтеграцію, дозволяє ізоляцію через Shadow DOM, не залежить від фреймворку і має очевидну перспективу масштабування у різні платформи та мобільні застосунки. Браузерне розширення варто залишити як можливе додаткове розгортання у майбутньому. А компонент конкретного фреймворку взагалі не є недоцільним з точки зору універсальності.

1.5.1 Вибір технологічного стеку

Першим аспектом вибору стеку був наявний досвід з технологіями (або максимально схожими на них). JavaScript є поширеною технологією, але мова TypeScript [24] за останні роки значно розширила свій вплив. Аргументи на користь TypeScript у контексті модуля доступності:

- статичне виявлення помилок ще до запуску;
- простіше супроводжувати у разі майбутнього розширення в окремий мікросервіс;
- краща підтримка автодоповнення і рефакторингу, можливість жорсткішої типізації.

Далі потрібно було продумати, який фреймворк обрати. Тут вибір не такий очевидний, як з TypeScript. У теоретичному розгляді можна було б використати

чисті Web Components, наприклад, через бібліотеку Lit. Проте на практиці виявилося, що React [25] дає переваги, які для модуля важливіші. Завдяки ньому можна будувати модель компонентів зі строгою типізацією, він також надає велику екосистему (зокрема для accessibility-патернів – Floating UI, Radix UI). Ця технологія є популярною, тому багато розробників потенційно зможуть взяти на себе її супровід за умови подальшого розширення. React монтується всередині Shadow DOM, тому переваги ізоляції зберігаються повністю. Додатково обрано бібліотеку Zustand [26] (над іншими технологіями типу Redux чи MobX) через мінімальний розмір, просту модель і швидкість. Вона допомагає значно полегшити керування станами. Щоб зібрати потім фронт-частину модуля, було вирішено використати Vite 5 [27]. Він зараз є доволі поширеним для нових проєктів. Час його збірки все ж іноді може бути дещо довшим, але він є простішим в інтеграції зі старішими технологіями типу Webpack чи Babel. Для додавання можливості інтеграції JavaScript-віджета у код сайту їхнім власникам було важливо продумати механізм об’єднання усіх залежностей проєкту у єдиний JS-файл. Тут Vite також виступив важливою технологією, яка дозволила зробити це.

Додавання голосового помічника можна реалізувати через Web Speech API [28]. Ця технологія доступна у всіх сучасних браузерах (Chrome, Firefox, WebKit тощо). Його обчислення виконуються локально, без зовнішніх запитів. Завдяки SpeechRecognition можна підтримати голосові команди. Це відбувається за рахунок зіставлення розпізнаного транскрипту зі списком тригерних фраз.

Для збереження налаштувань використовується Web Storage API [29]. У сховищі зберігаються лише прості дані: прапори, числові значення та вибрані параметри. Вся інформація записується у форматі JSON. Так, зберігати дані буде просто, без використання додаткових інструментів, і витягувати їх з такого формату, саме у контексті вебтехнологій, доволі зручно й швидко. Запис відбувається із затримкою 500 мс, щоб уникнути надмірної кількості звернень до сховища під час зміни налаштувань.

Висновки до розділу 1

У першому розділі здійснено системний аналіз предметної області. Розглянуто статистичні дані щодо поширеності порушень зору й когнітивних особливостей у світовій та українській аудиторіях, які підтверджують, що цільові групи становлять помітну частку користувачів вебсередовища. Систематизовано характерні бар’єри доступу за принципами стандарту WCAG 2.2.

Оглянуто чинні міжнародні та національні стандарти вебдоступності – WCAG 2.2 і 2.1, WAI-ARIA 1.2 з керівництвом APG, настанови COGA, EN 301 549, ДСТУ EN 301 549:2022, постанову Кабінету Міністрів України № 757. Виконано порівняльний аналіз чотирьох поширених готових рішень – UserWay, accessiBe, EqualWeb, AudioEye. Виявлено, що жодне з них одночасно не задовольняє вимогам відкритості коду, сумісності з допоміжними технологіями та підконтрольності для розробника.

На цій основі було обґрунтовано вибір моделі реалізації у вигляді вбудовуваного JavaScript-віджета і технологічного стеку: TypeScript як мова, React з Zustand, Vite як збирач, Shadow DOM для ізоляції, Web Speech API для синтезу й розпізнавання мовлення, Web Storage API для збереження налаштувань. Сформульовано постановку задачі та функціональні і нефункціональні вимоги до модуля, які слугують підставою для його технічного проєктування у наступному розділі.

РОЗДІЛ 2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ АДАПТИВНОГО МОДУЛЯ

У першому розділі обґрунтовано вибір моделі реалізації (вбудовуваний JavaScript-віджет) і технологічного стеку. Другий розділ присвячений власне проєктуванню модуля. А саме побудові загальної архітектури, ядру, рушіям змін, профілям користувачів, голосовому модулю, інтеграції з хост-сайтом та інтерфейсу панелі. Усі рішення спираються на сформульовані функціональні й нефункціональні вимоги.

2.1 Загальна архітектура модуля

Архітектура побудована за принципом ядра з додатковим розширенням. Ядро відповідає за життєвий цикл модуля, керування станом, події, доступ до сховища. Усі предметні зміни підключаються до ядра через спільні інтерфейси. З огляду усіх можливих архітектур такий підхід було обрано через наступні переваги:

- Чітке розмежування відповідальності, коли кожен модуль робить одну річ. Такий підхід відповідає загальноприйнятим правилам написання чистого й оптимізованого коду.
- Кожен компонент можна перевірити окремо, що значно спрощує тестування.
- Збірка є доволі гнучкою, тому непотрібні компоненти можна виключити з фінального зібраного скрипта. Це було додано для можливості в майбутньому розробникам збирати модуль під свої потреби.
- Перспектива розширення у майбутньому, адже новий тип адаптації додається підключенням нового модуля без зміни ядра.

Проаналізувавши технологічні рішення, представлені на ринку, модуль спочатку схематично було поділено на вісім великих частин. Узагальнена схема його бачення подана на рис. 2.1.

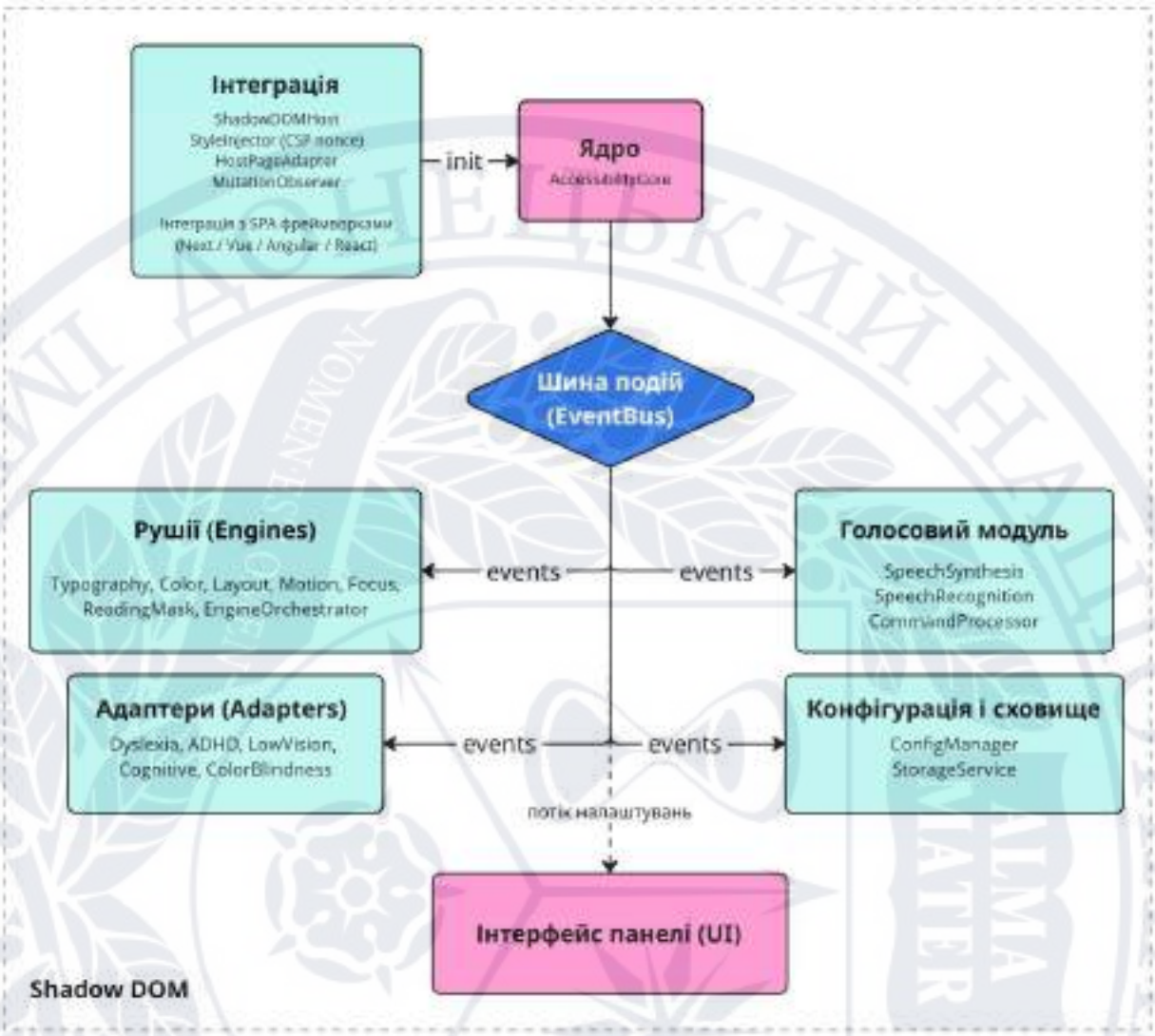

Рисунок 2.1 – Загальна архітектура адаптивного модуля доступності

Дещо детальніше про окремі частини:

- Ядро є головною частиною, що відповідає за запуск і вимкнення модуля. Він також є елементом для зберігання поточного стану, обробки подій та автоматичного збереження налаштувань у локальному сховищі.
- Рушії – окремі модулі, які змінюють вигляд і поведінку сайту, а саме: шрифти, контрастність, фокус, маску читання, параметри для моторної доступності та верстку.
- Адаптери – готові профілі налаштувань для людей з дислексією, РДУГ, порушеннями зору, дальтонізмом або когнітивними труднощами. Також вони перевіряють відповідність вимогам WCAG.

- Голосовий модуль забезпечує озвучення тексту, розпізнавання голосу та виконання голосових команд.
- Інтеграція з хост-сайтом підключає модуль до сайту через Shadow DOM, додає стилі і відстежує зміни контенту.
- Керування налаштуваннями модуля та збереження займаються конфігурація і сховище.
- І UI-частина, побудована на React-компонентах. Це фактично інтерфейс.

У кореневій папці модуля цьому розподіленню відповідає поділ на основні каталоги core/, state/, engines/, services/, presets/, accessibility/, voice/, integration/, config/, storage/, components/, providers/, hooks/ всередині src/ директорії (рис. 2.2). Окремий файл embed.ts використовується як вхідна точка для збірки, яка ініціалізує модуль за конфігурацією користувача й монтує React-додаток усередину тіньового кореня.

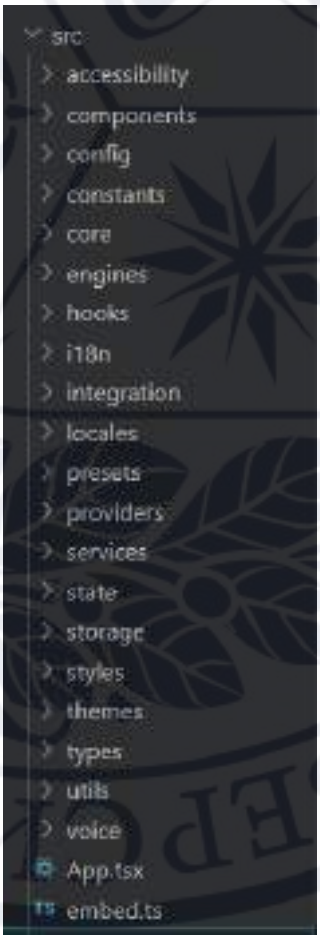

Рисунок 2.2 – Основна файлова структура модуля

Поєднання між усіма цими системами має бути реалізовано через два спільні канали. Спочатку через централізоване сховище стану, у якому зберігаються поточні налаштування користувача. А далі через події, крізь які проходять асинхронні сигнали, що повідомляють, до прикладу, про готовність модуля. Це зроблено для масштабування і відокремлення, тобто якщо ми додамо якусь нову голосову команду, то вона підключається через ті самі інтерфейси і починає реагувати на ті самі події.

Ще один важливий принцип, на якому має будуватися архітектура, – це те, що модуль повинен якомога менше залежати від оточення, у якому він працює. Він має інтегруватися у різні сайти. Тому жодна з частин модуля не має бути захардкодована чи покладатися на конкретну розмітку сайту, на наявність певних класів або CSS-правил. Усе, що змінюється на сторінці, передається через стандартні механізми, які працюють однаково у будь-якому браузері. Коли ти підключаєш його одним рядком, він починає працювати, незалежно від того, на чому зроблений сам сайт.

2.2 Ядро модуля

Основним класом ядра є AccessibilityCore. Він працює так, що, у межах однієї сторінки існує лише один екземпляр, до якого підключається вся решта компонентів. Тому виходить, що ядро має керувати єдиним станом налаштувань і стеком змін.

Після підключення модуля до сайту-клієнта він формує робочу конфігурацію, яка об’єднує параметри від власника вебу зі значеннями за замовчуванням, щоб отримати повний набір. Далі ядро ж і перевіряє локальну пам’ять браузера, щоб відстежити, чи зберігалися там попередні налаштування користувача з минулих сеансів. Якщо так, то вони завантажують їх і одразу застосовують. Такий механізм ідеальний, адже користувач, який налаштував комфортний для себе інтерфейс один раз, отримуватиме сайт у потрібному для нього вигляді при кожному наступному відвідуванні без потреби повторно щось налаштовувати. Якщо вийде інтегрувати це в модуль, то це покращить юзабіліті

для юзерів та потенційно збільшить коефіцієнт повертання до такого сайту в майбутньому.

Робота модуля передбачає зміну станів компонентів на сайті. Завдяки цій механіці будуть доступні модифікації. І враховуючи, що в планах є реалізація сховища зі збереженням налаштувань, наявні кілька моментів, які треба врахувати при архітектурі, щоб уникнути неприємностей. По-перше, debounce 500 мс. Без нього кожен рух повзунка масштабу шрифту викликав би запис у сховищі, що значно може знижувати продуктивність. По-друге, якщо людина закриє вкладку відразу після руху повзунка, то останнє налаштування користувача можна втратити. Тому планується при знищенні модуля, щоб ядро примусово зливало останні незбережені зміни синхронно. Якщо уявити ситуацію, коли локальне сховище переповнене чи наявні інші обставини неможливості запису в нього (наприклад, коли користувач увійшов у режим інкогніто), то це може викликати помилку у роботі модуля чи навіть призведе до того, що він впаде (це виникло вже на етапі тестування, тому було враховано тут). З огляду на це, прийнятим рішенням стало те, що останній виклик операції збереження треба обгорнути у блок try/catch.

Оскільки окремі модулі реагують на конкретні сигнали, було прийнято рішення додати щось схоже на буфер повідомлень, через який проходить уся службова інформація. Будь-яка частина модуля може надіслати повідомлення певного типу, не знаючи, хто його почує. Інші частини, які чекають саме на такий тип повідомлення, отримають його і відреагують. Цей підхід схожий на слабке зв’язування, коли компоненти не знають одне про одного безпосередньо, але можуть реагувати на спільні тригери. Завдяки цьому додавання нової функції не потребує перероблення наявного коду. Якщо хтось у майбутньому захоче, наприклад, збирати знеособлену статистику використання модуля – він просто напише невелику окрему частину, яка реагуватиме на потрібні сигнали і робитиме свою справу.

2.3 Рушії змін зовнішнього вигляду сторінки

Усе, що користувач бачить на сторінці після увімкнення модуля, виконується рушіями. Так, у проєкті будуть названі невеликі окремі компоненти, кожен з яких відповідає за свою категорію змін. Один змінює шрифти, інший – кольори, третій – макет сторінки і так далі. Таке розділення зроблено свідомо. Якщо колись з'явиться потреба змінити, наприклад, поведінку маски читання, достатньо буде відкрити лише один файл, не торкаючись інших. Під час планування архітектури модуля було реалізоване графічне відображення сімейства рушіїв і зв'язків між ними (рис. 2.2).

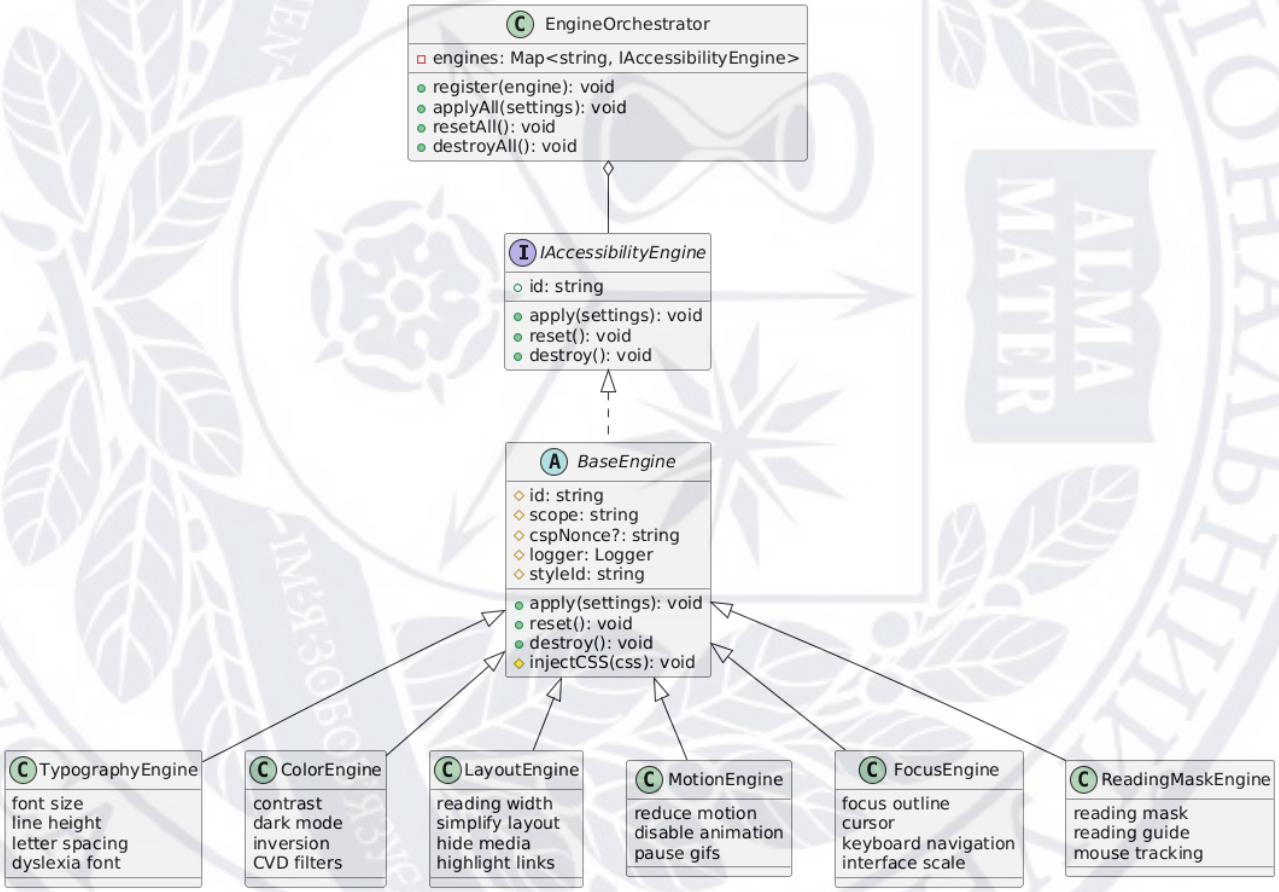

Рисунок 2.2 – Загальна структура рушіїв змін

Усі вони побудовані на спільному патерні. У них є один базовий шаблон, у якому зібрано їхні вміння за замовчуванням, по типу додавання своїх стилів до сторінки і одночасно їх скасування, деякі спільні службові функції. Конкретний

рушій бере цей шаблон і доповнює його тільки своєю специфічною логікою. Це гарантує, що всі поводяться однаково в типових ситуаціях. Якщо у майбутньому буде додано якийсь новий, то йому достатньо буде успадкувати базовий шаблон і написати один функціональний метод.

Перший рушій відповідає за все, що пов’язано зі шрифтом. Через нього користувач може збільшити або зменшити розмір тексту, змінити інтервали між рядками, літерами і словами, обрати іншу гарнітуру. Для базового модуля буде взято такий набір:

- системний шрифт;
- читабельні базові гарнітури Arial і Helvetica;
- шрифти із засічками Georgia і Times New Roman;
- спеціалізована гарнітура для дислектиків OpenDyslexic.

Згідно з поставленим технічним завданням у попередньому розділі передбачено кілька гарнітур на вибір – серед них і спеціалізована гарнітура для людей з дислексією, у якій літери мають характерну форму, що зменшує плутанину між схожими символами. Усі параметри мають застосовуватися через спеціальні CSS-змінні, які потім впливають на оформлення сторінки.

Другий рушій керує кольорами і контрастом. У ньому передбачено кілька готових тем оформлення: класичний високий контраст з чорним фоном і білим текстом, м’яка темна тема з блакитними посиланнями, світла тема з посиленим контрастом. Усі теми відповідають мінімальним вимогам стандарту WCAG 2.2 щодо контрасту – 4.5:1 для звичайного тексту і 3:1 для великого або жирного [7]. Окремо можна увімкнути темну тему незалежно від основної, інвертувати кольори, прибрати кольоровість зовсім або, навпаки, посилити насиченість. Для людей з різними формами дальтонізму (протанопія, дейтеранопія, тританопія) є спеціальні режими симуляції, що перетворюють кольори сторінки так, аби користувач побачив сайт у тому вигляді, в якому його бачать інші люди з відповідним типом сприйняття.

Третій рушій займається макетом сторінки. Він може обмежити ширину області читання до 60–70 символів у рядку – це значення відповідає

рекомендаціям Британської асоціації дислексії щодо типографічної доступності [36], спростити верстку, приховати декоративні зображення і відео, які заважають концентруватися, або підсвітити заголовки і посилання, щоб структура сторінки стала прозорішою. Найскладнішим тут є правильне визначення, де саме на сторінці знаходиться основний контент, а де – навігація, реклама, бічні панелі тощо. Сучасні сайти, особливо ті, що мають унікальний дизайн, можуть будуватися по-різному. Тому для спрощення наразі рушій спирається на стандартні підказки в розмітці і, якщо їх немає, то обирає найбільший за площею текстовий блок як основний. Цей підхід є доволі простим з точки зору реалізації на даному етапі. Але для повноцінної масштабної розробки у майбутньому варто інтегрувати більш точні механізми для ідентифікації типів контенту на вебсайтах.

Четвертий рушій вимикає автоматичні анімації, до них відносять: рухомі банери, спливні вікна, миготливі елементи та ін. Візуально такі елементи виглядають дуже естетично, привертають нашу увагу, але не для людей з вестибулярними розладами, з розладами дефіциту уваги чи з тривожними розладами. Надмірний рух на сторінці може бути справжньою перешкодою для цих категорій користувачів, бо спричинятиме дискомфорт чи відволікання. Найкращий варіант реалізації – коли рушій уміє автоматично враховувати системне налаштування користувача. Якщо людина на рівні операційної системи попросила менше руху (через медіазапит prefers-reducedmotion [30]), модуль вмикатиме відповідний режим без додаткових дій, не змушуючи користувача повторно відкривати панель і шукати потрібний перемикач.

П’ятий рушій посилює видимість фокусу. На багатьох сайтах за замовчуванням рамка, яка з’являється навколо активного елемента при навігації клавіатурою, ледь помітна або взагалі прибрана з дизайнерських міркувань. Для людей, які користуються клавіатурою замість миші, це потенційно може створити проблему, що вони втратять орієнтацію на сторінці. Цей рушій додає чітку контрастну рамку навколо активного елемента, яка завжди добре видно. Згідно з критерієм 2.4.13 стандарту WCAG 2.2 [7], така рамка має бути принаймні 3 px за шириною. За потреби можна додатково збільшити курсор миші.

Шостий рушій створює на сторінці маску фокусу читання. Це горизонтальна смуга з прозорим вікном кілька рядків заввишки, яка слідує за рухом курсора. Усе, що знаходиться за межами цієї смуги, притемнюється. Виглядає це так, ніби людина читає текст крізь спеціальну читальну лінійку. Цей засіб особливо корисний для людей з дислексією і розладами концентрації уваги. Він допомагає фокусувати погляд на потрібному рядку і знижує когнітивне навантаження від іншого контенту на сторінці.

Окремий невеликий компонент координує роботу всіх рушіїв. Коли користувач щось змінює у панелі, цей координатор отримує нові налаштування зі сховища і послідовно передає їх кожному рушієві. Якщо в одному з них раптом виникає помилка, координатор фіксує її і йде далі, не перериваючи роботу решти.

2.4 Готові режими адаптації

Можливість тонко налаштувати десятки окремих параметрів – це добре для тих, хто розуміється на доступності і знає, що саме йому потрібно. Але переважна більшість користувачів не має ані часу, ані бажання розбиратися з кожним параметром і тим паче розуміти, що саме їм потрібно. Тому в модулі передбачено окремий шар швидких готових режимів. Це попередньо підготовлені набори налаштувань, кожен з яких оптимізований для конкретної категорії користувачів. Активація режиму займає один клік і інтерфейс відразу переходить у комфортний для цієї категорії стан.

У модулі мають бути реалізовані кілька таких режимів. Один з них призначений для людей з дислексією. У ньому одразу обирається спеціалізована гарнітура шрифту, збільшуються міжлітерний (близько 0,12 em) і міжрядковий (1,9) інтервали, обмежується ширина області читання до 60–70 символів у рядку, фон стає м’яким кремовим замість різко-білого, а на сторінці з’являється маска фокусу читання. Усі ці значення спираються на настанови Британської асоціації дислексії [14] та на дослідження ефективності спеціалізованих шрифтів для адаптації текстів [15].

Інші режими орієнтовані на людей з розладом дефіциту уваги та когнітивними порушеннями. У них головний акцент має бути зроблений на зниженні відволікаючих факторів відповідно до настанов W3C щодо когнітивної доступності (COGA) [11]. Вимикаються всі автоматичні анімації, приховуються декоративні блоки, активується режим концентрації, який залишає на сторінці лише основний контент. На додачу вмикається маска фокусу читання, щоб допомогти втриматися на потрібному рядку.

Також варто додати режими для слабкозорих користувачів. Тут пріоритет – максимальна читабельність. Шрифт робиться помітно більшим (1,5–2x), контраст підвищується, додаються посилене виділення фокусу і збільшений курсор. Якщо доступний голос для обраної мови, можна одразу вмикнути озвучення сторінки, щоб людина могла прослухати текст замість читати його очима.

Далі режими для людей з різними формами кольорової сліпоти. У них має активуватися відповідний набір налаштувань , який перетворює кольори сторінки так, щоб допомогти користувачу краще розрізняти елементи, що інакше зливалися б для нього в один колір. У межах цих режимів передбачено кілька варіантів для різних типів дисфункції кольорosприйняття.

2.5 Голосовий модуль

Голосовий модуль складається з двох пов’язаних частин. Перша дозволяє користувачеві прослухати вміст сторінки замість того, щоб читати його. Друга дозволяє керувати модулем словами, просто промовляючи відповідну фразу зі заздалегідь визначеного набору. Обидві частини спираються на стандартний браузерний інтерфейс Web Speech API і не потребують підключення до жодних сторонніх сервісів. Це було вигадано для приватності на цьому етапі розробки, тобто щоб голос користувача нікуди не передавався, усе оброблялося локально на пристрої. Загальну схему роботи голосового модуля показано на рис. 2.3.

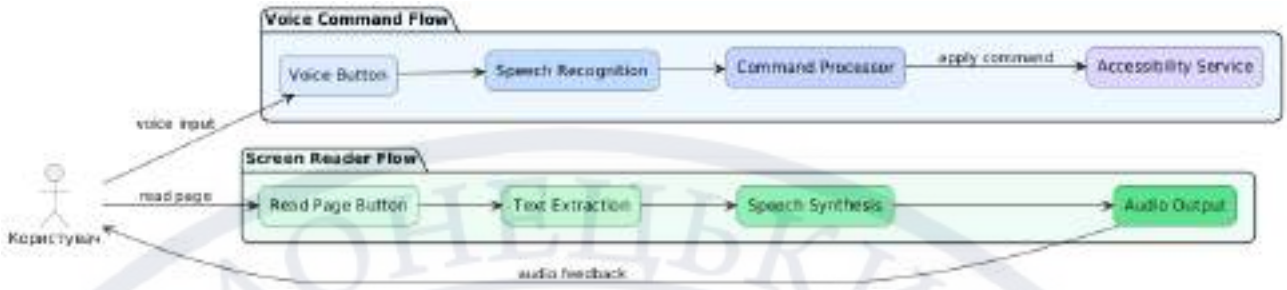

Рисунок 2.3 – Загальна схема роботи голосового модуля

Озвучення тексту працює так. Користувач натискає у панелі кнопку «Прочитати сторінку» або вибирає окремий фрагмент тексту і запускає озвучення для нього. Модуль збирає текстовий контент сторінки, передає його у браузер з обраними параметрами (мова, швидкість мовлення, тон, гучність тощо) і браузер починає говорити. Усе це відбувається на стороні користувача, без жодних запитів до зовнішніх серверів. Голос обирається з того набору, який встановлений в операційній системі: на більшості сучасних систем доступні голоси для англійської мови та багатьох інших мов. Якщо потрібного голосу немає, модуль надає повідомлення юзеру. Показує підказку про те, що для повноцінної роботи варто встановити голосовий пакет у системі. Цей підхід не є ідеальним для глобального використання, але на цьому етапі є найбільш швидким у реалізації та дієвим. До того ж, на локальному девайсі користувача з більшою вірогідністю все ж буде потрібна для нього мова з доступним типом озвучування.

Працювати з озвученням можна по-різному. Хтось вмикає його, щоб прослухати довгу статтю замість того, щоб читати. Це особливо зручно під час інших справ або для людей, які краще сприймають інформацію на слух. Хтось використовує озвучення вибраного фрагмента, коли треба перевірити, як саме звучить написане. Для людей з порушеннями зору ця функція може бути однією з основних, особливо якщо у них немає окремих програм зчитування з екрана. Модуль не намагається замінити такі програми, але він дає базовий рівень голосового доступу, який спрацьовує одразу і не потребує налаштування на рівні системи.

Дещо важча в реалізації частина – розпізнавання голосових команд. Користувач натискає у панелі відповідну кнопку, дозволяє доступ до мікрофона, і модуль починає слухати. Поки людина говорить, модуль у реальному часі показує їй у транскрипті, що саме він розчув. Коли користувач робить паузу, модуль аналізує почуту фразу і шукає в ній знайому команду. Якщо команда знайдена, модуль одразу виконує відповідну дію. Список підтримуваних команд охоплює основні сценарії: увімкнення темної теми, активація готового режиму, регулювання розміру шрифту, прочитування сторінки, відкривання і закривання панелі, перехід між розділами сторінки. Голосові команди не дублюють логіку панелі. Вони викликають той самий ланцюжок дій, що і кліки в інтерфейсі. Тому поведінка модуля однакова незалежно від того, як саме користувач взаємодіє з ним – мишею, клавіатурою, дотиком чи голосом. Це і простіше у розробці (немає двох паралельних реалізацій однієї і тієї ж функції), і безпечніше у супроводі (виправлення помилки в одному місці автоматично вирішує її в усіх способах виклику). Голосове керування особливо корисне для людей з обмеженою моторикою, яким складно або неможливо користуватися мишею чи клавіатурою. Для них модуль перетворюється на повноцінний голосовий інтерфейс. Звісно, цей шар поки що не претендує на повне розпізнавання довільних запитів, адже він працює зі заздалегідь визначеним набором команд. Але цього достатньо, щоб закрити основні сценарії взаємодії з модулем і дати користувачеві альтернативний спосіб керування.

2.6 Інтеграція з хост-сайтом і збереження налаштувань

Цей підрозділ присвячений тому, як модуль вписується у сайт, на якому працює, і як він зберігає налаштування між сеансами. Це службова частина, яку користувач безпосередньо не бачить, але без неї нічого не змогло б працювати на чужих сайтах.

Найбільша складність вбудовуваного модуля полягає в тому, що він має співіснувати з довільним хост-сайтом, не порушуючи його роботи. У хост-сайту є власні стилі, які можуть випадково вплинути на вигляд панелі модуля. Тому для

розміщення модуля використовується технологія Shadow DOM [31]. У модуля, своєю чергою, є власні стилі, які можуть випадково вплинути на верстку сайту. Якщо їх не ізолювати одне від одного, виходить хаос. Панель модуля виглядає по-різному на різних сайтах, а сайти ламаються при підключенні модуля. Тому для його розміщення використовується спеціальна технологія, яка створює всередині сторінки невелике ізольоване середовище. Усе, що знаходиться всередині нього, отримує власну область стилів: стилі сайту туди не потрапляють, стилі модуля назовні не виходять. Це також гарантуватиме, що зовні модуль виглядає однаково на будь-якому сайті, незалежно від його дизайну, і що жоден сайт не зламається від його підключення. Налаштування цього середовища робиться так, щоб клавіатурна навігація між сайтом і модулем працювала природно, і користувач не відчував різниці між тим, що належить сайту, а що – модулю.

Окрема важлива частина роботи – політика безпеки сайтів. Багато сучасних сайтів, особливо зі сфери, де працюють з великим масивом чутливих даних користувачів (банківські, фінансові, корпоративні), мають суворі обмеження на те, які скрипти і стилі можна виконувати в їхньому середовищі. Це називається політикою безпеки контенту (або з англійської мови – Content Security Policy, CSP) [32]. Зазвичай вона забороняє довільний код, який модуль міг би випадково додати до сторінки. Щоб модуль працював і на таких сайтах, передбачено механізм, у якому власник сайту може передати модулю спеціальний токен довіри. Усі стилі, які додає модуль, отримують цей токен і коректно проходять перевірку політики безпеки. Без цього механізму модуль не зміг би використовуватися на ресурсах з підвищеними вимогами до безпеки, а це – багато державних і фінансових сайтів, для яких доступність особливо важлива.

Ще одним викликом для побудови архітектури стала поведінка модуля на сучасних сайтах, які змінюють контент без перезавантаження сторінки. Це односторінкові застосунки, побудовані на популярних фреймворках по типу того ж React. У них перехід між розділами не призводить до того, що сторінка завантажується заново. Замість цього змінюється лише частина вмісту в DOM.

Якщо просто застосувати налаштування модуля один раз при першому запуску, то для нового контенту, який з’явиться пізніше, ці налаштування не подіють. Модуль має автоматично помічати такі зміни і повторно застосовувати адаптації до нових елементів. Для цього використовується стандартний браузерний механізм MutationObserver [33]. Без нього на сучасних сайтах модуль швидко перестав би працювати правильно. Нові елементи з’являлися б у звичайному вигляді, а адаптовані лишалися лише на тих частинах, які були при першому завантаженні.

Якщо у майбутньому формат налаштувань зміниться, наприклад, з’являться нові поля, а старі будуть прибрані, попередні збережені дані стануть несумісними з новим модулем. Щоб уникнути проблем у таких ситуаціях, він перед застосуванням завантажених даних перевіряє їх формат. Якщо виявляється застарілість, то модуль просто починає з налаштувань за замовчуванням замість того, щоб намагатися їх інтерпретувати і ламатися на неправильних значеннях. Користувач у цьому випадку нічого не помічає, адже він просто бачить інтерфейс у стандартному вигляді і може налаштувати його заново.

Висновки до розділу 2

У другому розділі описано архітектуру адаптивного модуля доступності. Архітектуру побудовано як набір невеликих самостійних частин, кожна з яких відповідає за свою чітко визначену задачу – ядро, рушії, готові режими, голосовий модуль, інтеграція з хост-сайтом, інтерфейс панелі. Усі частини спілкуються між собою не напряму, а через два спільні канали – центральне сховище стану і шину повідомлень. Це забезпечує слабке зв’язування компонентів, спрощує супровід проєкту і робить його перспективним з точки зору подальшого розширення.

Ядро відповідає за запуск і зупинку модуля, збереження налаштувань між сеансами і повідомлення інших частин про зміни. Рушії безпосередньо змінюють вигляд сторінки – керують шрифтами, кольорами, макетом, анімаціями, видимістю фокусу і маскою читання. Готові режими дають користувачеві

швидкий старт для типових категорій – дислексії, розладу дефіциту уваги, низького зору, дальтонізму, загальної когнітивної підтримки. Голосовий модуль додає два паралельні способи взаємодії – озвучення тексту і керування голосовими командами.

Окрема увага приділена тому, як модуль вписується у чужий сайт без шкоди для нього. Для цього використовуються ізольоване середовище всередині сторінки, підтримка політик безпеки сучасних сайтів, автоматичне реагування на динамічні зміни контенту в односторінкових застосунках, а також гнучке збереження налаштувань з кількома запасними варіантами. Інтерфейс панелі побудовано з акцентом на доступність – щоб сам інструмент адаптації не створював нових бар'єрів. Описана архітектура є основою для програмної реалізації модуля та її тестування, які детально розглядаються у наступному розділі.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ АДАПТИВНОГО МОДУЛЯ ДОСТУПНОСТІ

Ця частина буде присвячена практичній реалізації модуля. Тут буде реалізація проєктної архітектури у формі робочого коду. Показано вигляд інтерфейсу панелі для кінцевого користувача і як модуль поводиться у реальному оточенні на прикладі тестового вебсайту.

3.1 Огляд реалізації та середовища тестування

Модуль реалізовано у вигляді окремого фронтенд-проєкту мовою TypeScript із використанням бібліотеки React для побудови інтерфейсу та бібліотеки Zustand для централізованого керування станом. Для збірки застосовано Vite 5.

Загальна архітектура досягається тим, що ядро AccessibilityCore реалізоване як одиночний (singleton) клас із синхронним методом init() і повним життєвим циклом destroy(); усі рушії спілкуються через інтерфейс IAccessibilityEngine з обов’язковими методами apply(), reset() і destroy(); а EngineOrchestrator веде Map<id, engine> і послідовно делегує їм оновлення налаштувань. Поєднання забезпечує EventBus з типізованими каналами на кшталт settings:change, core:ready, core:destroy, що дозволяє додавати нові слухачі без модифікації ядра.

Реалізація AccessibilityCore використовує шар StorageService, який інкапсулює адаптери LocalStorageAdapter, IndexedDBAdapter та MemoryAdapter. Це дозволяє за допомогою єдиного інтерфейсу завантажувати та зберігати налаштування незалежно від середовища виконання браузера.

Точкою початкового входу для робочого варіанту модуля служить файл embed.ts. У ньому експортовано функцію init(), яка створює конфігурацію, монтує контейнер у DOM і запускає React-додаток усередині ізольованого тіньового кореня. Власник сайту може передати у init() параметри початкового налаштування за власного бажання чи наявних технічних потреб.

Для тестування демонстрації роботи модуля було розроблено окрему тестову сторінку, сформовану у базовому index.html файлі (рис. 3.1). На цьому

етапі усі стилі та елементи прописані саме в ньому для економії часу. Це звичайний статичний HTML-документ, який імітує типовий контентний сайт. Тестова сторінка не містить жодної заздалегідь закладеної підтримки модуля доступності, її розмітка була свідомо зроблена схожою на реальний сайт, без особливих ARIA-розширень.

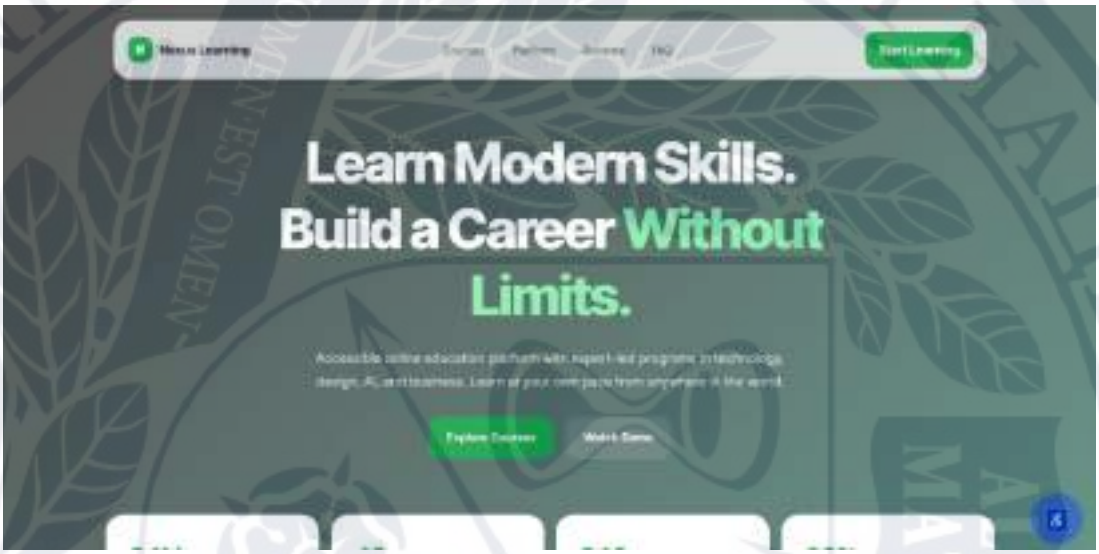

Рисунок 3.1 – Вигляд сторінки тестового вебсайту

У такому форматі вона виглядає як звичайний інформаційний ресурс з показу навчальних курсів. Було також додано інформаційні частини про доступність на сторінці, аби пов’язати її з тематикою дипломної роботи. Після додавання одного рядка з підключенням файлу модуля у нижньому правому куті екрана з’являється плаваюча кнопка (рис. 3.2) з піктограмою доступності, яка є точкою входу для всіх адаптацій.

Рисунок 3.2 – Кнопка модуля доступності

Кнопка реалізована окремим компонентом і завжди залишається на фіксованій позиції незалежно від прокручування сторінки. Її положення (нижній правий, нижній лівий, верхній правий або верхній лівий кут) задається через параметр у конфігурації підключення. Розмір кнопки становить 68 px у діаметрі, що було зроблено зі збільшенням. Схожий розмір кнопки має більшість конкурентів. Він також перевищує мінімальний розмір інтерактивної цілі 24×24 px за критерієм 2.5.8 стандарту WCAG 2.2 [7]. Натиснення кнопки (або відповідне клавіатурне скорочення) відкриває панель з усім функціоналом модуля. Колір як кнопки, так і усієї панелі було обрано за схожим патерном, як у конкурентів.

Варто зазначити, що для людей, які не користуються мишею, було додано можливість відкривати панель без кліку. У реалізації використано комбінацію Ctrl+Alt+A. Під час розробки було спочатку обрано варіант Alt+A, але виявилося, що у деяких варіаціях браузерів вона вже використовується, зокрема як альтернатива Ctrl+A для виділення всього тексту. Тому було змінено на більш довгий варіант, аби зменшити можливість такого дублювання, але при цьому і не перевантажувати кількість кнопок для натискання. Внутрішня реалізація прив’язана до фізичної клавіші Key code KeyA, а не до символу «A», аби гарантувати однакову поведінку незалежно від поточної розкладки клавіатури (українська, англійська тощо).

3.2 Вкладка ручних налаштувань доступності

Це основний інструмент, через який користувач самостійно підлаштовує сайт під свої потреби. На відміну від готових режимів (про які йдеться у підрозділі 3.3), тут користувач має повний контроль над окремими параметрами і може налаштувати інтерфейс саме так, як йому зручно, без покладання на узагальнені пресети.

3.2.1 Структура панелі і навігація між вкладками

Сама панель відкривається як модальне вікно з невеликим затемненням решти сторінки. Структурно вона складається з трьох частин: верхня шапка із

заголовком і кнопкою закриття, горизонтальна навігація вкладок одразу під шапкою, і основна область з вмістом активної вкладки.

У поточній реалізації панель містить три вкладки:

- Settings – детальні ручні налаштування,
- Modes – готові режими адаптації,
- Voice – голосовий помічник.

Перемикання між вкладками реалізовано через стандартний ARIA-шаблон tablist [6]: контейнер має role=«tablist», кожна кнопка вкладки – role=«tab» з атрибутом aria-selected, а сама область вмісту – role=«tabpanel» із зворотним посиланням aria-labelledby. Завдяки цьому програми зчитування з екрана коректно оголошуватимуть і саму вкладку, і її поточний стан.

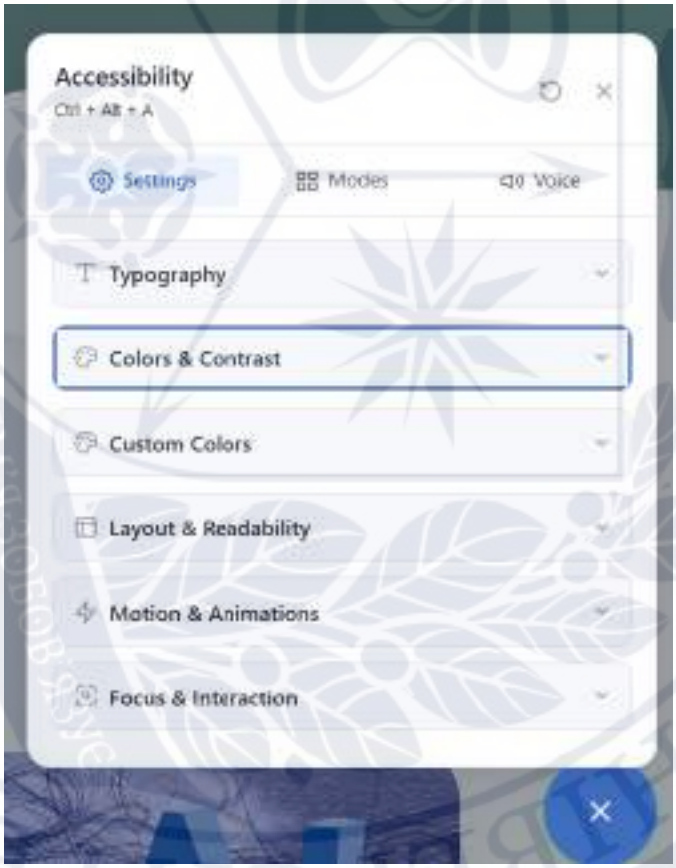

Рисунок 3.3 – Відкрита панель налаштувань модуля з активною вкладкою Settings

На рис. 3.3 показано вкладку «Settings» одразу після відкриття (але зі згорнутими усіма секціями, аби було видно на зображенні увесь функціонал). Усі блоки згорнуті за замовчуванням, окрім перших двох. Їх було обрано, адже це найбільш популярні та швидкі зміни. Користувач натискає на заголовок будь-якої секції і вона розгортається. Це має на меті знизити когнітивне навантаження від великої кількості параметрів.

Структурно вкладка «Settings» розділена на 6 згортуваних секцій: «Typography» (типографіка), «Colors» (кольори), «Custom Colors» (власні кольори), «Layout» (макет), «Motion» (анімації) і «Focus» (фокус і взаємодія). Кожна секція містить тематично згруповані контролери для налаштування кастомних параметрів. Усі вони є React-компонентами, які можна повторно використати у будь-якому місці інтерфейсу.

3.2.2 Секція типографіки

Вона дозволяє користувачеві самостійно підлаштувати оформлення тексту під свої потреби: обрати шрифт, розмір, інтервали, ширину області читання і вирівнювання.

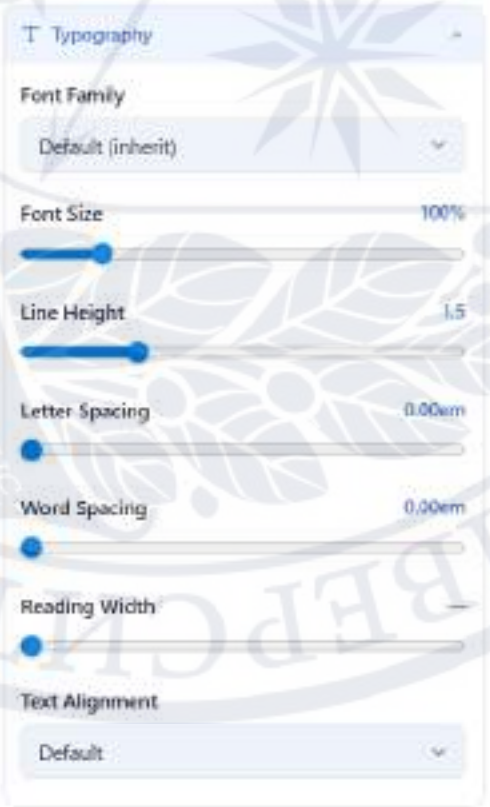

Рисунок 3.4 – Розгорнута секція типографіки

Як видно з рис. 3.4, секція містить такі елементи керування:

- Першим пунктом можна обрати шрифт.
- Далі можна міняти масштаб шрифту в діапазоні від 80 % до 200 %.
- Третім пунктом йде зміна висоти рядка від 1.0 до 3.0 з кроком 0.1.
- «Letter spacing» – редагування міжлітерного інтервалу від 0 до 0,30 em з кроком 0,01.
- Можна змінювати інтервал між словами від 0 до 1,0 em з кроком 0,05.
- Повзунок «Reading width» – обмеження ширини області читання.
- Вирівнювання тексту ліворуч, по центру чи по ширині.

Усі повзунки реалізовані окремим компонентом SliderControl. Це звичайний HTML-елемент типу range, обгорнутий у власну розмітку зі стилізованим треком і відображенням поточного значення. Випадні списки реалізовано компонентом SelectControl, перемикачі – компонентом ToggleControl, а кнопки одноразової дії (як, наприклад, «Reset») – компонентом ButtonControl.

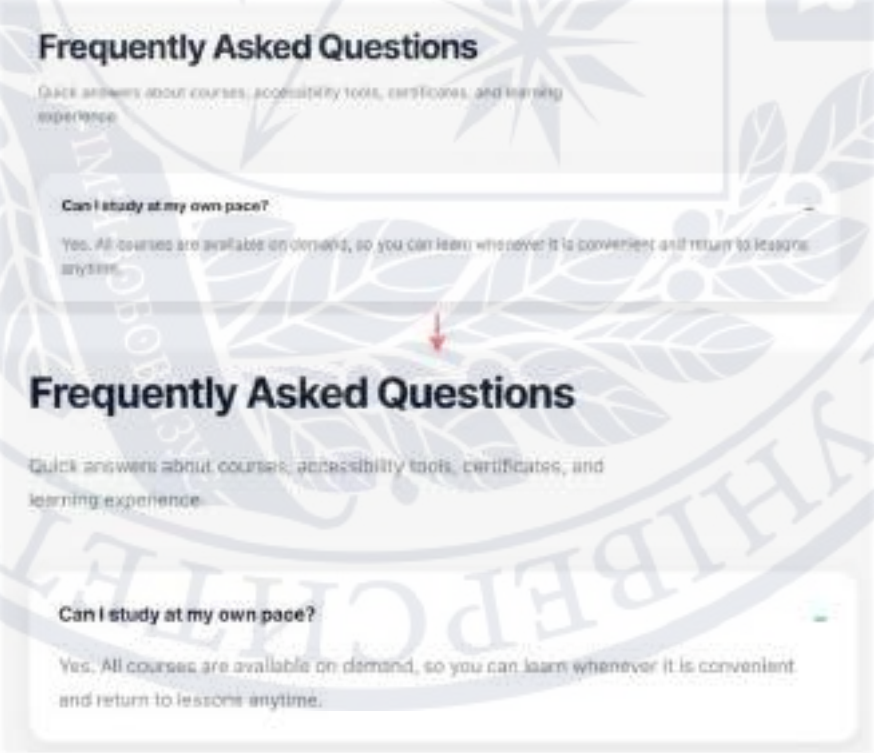

Рисунок 3.5 – Зміна розміру шрифту і висоти рядка на тестовій сторінці

На рис. 3.5 показано, як виглядає тестова сторінка після того, як користувач збільшив розмір шрифту до 130 % і висоту рядка до 1.8. На рис. 3.6 показано приклад адаптації тексту для дислектиків у одній із секцій.

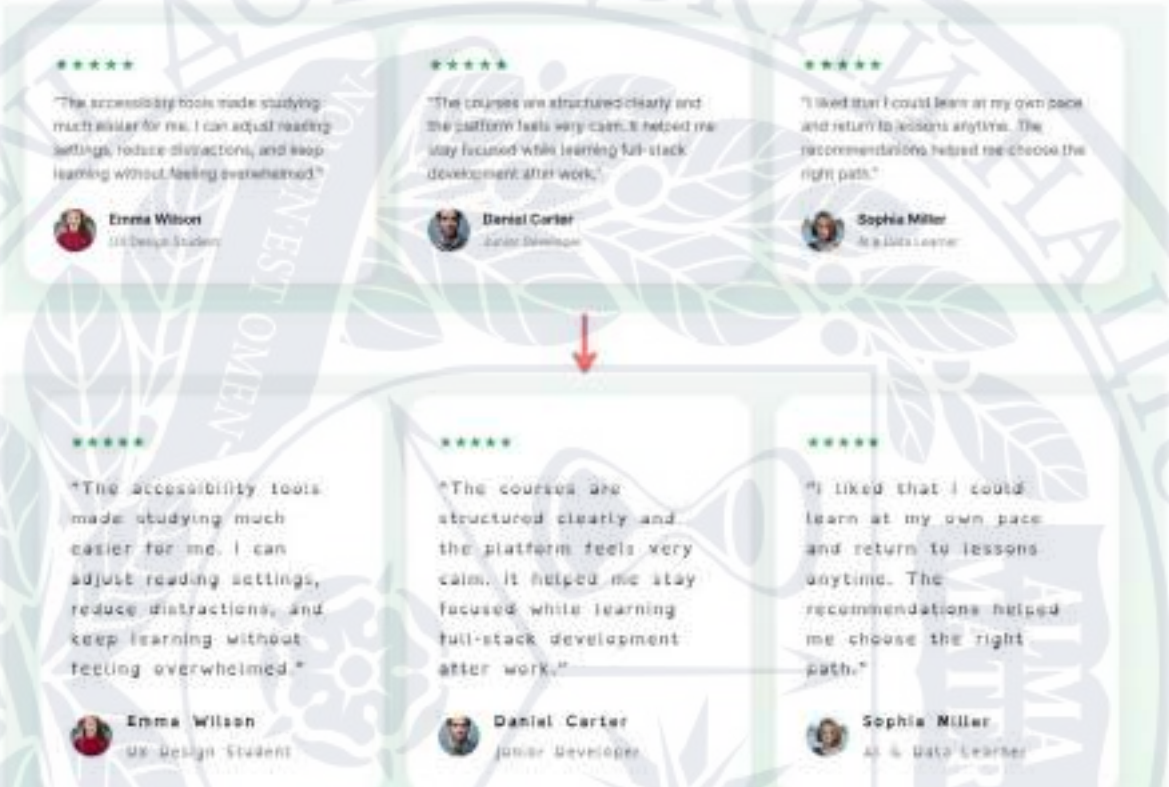

Рисунок 3.6 – Адаптація тексту для дислектиків у секції з відгуками

3.2.3 Секція кольорів і контрастності

Ця секція (рис. 3.7) дозволяє користувачеві керувати кольоровою схемою сторінки, контрастністю, насиченістю, а також активувати симуляцію різних форм порушень кольорosприйняття.

Секція містить випадний список вибору режиму симуляції дальтонізму. Передбачено сім таких режимів: повна неможливість сприймати червоний колір (протанопія), повна неможливість сприймати зелений (дейтеранопія), повна неможливість сприймати синій (тританопія), монохромне сприйняття (ахроматопсія), а також слабкі форми перших трьох (протаномалія,

дейтераномалія, тританомалія). Кожен режим реалізовано через окремий SVG-фільтр з власною матрицею перетворення кольорів.

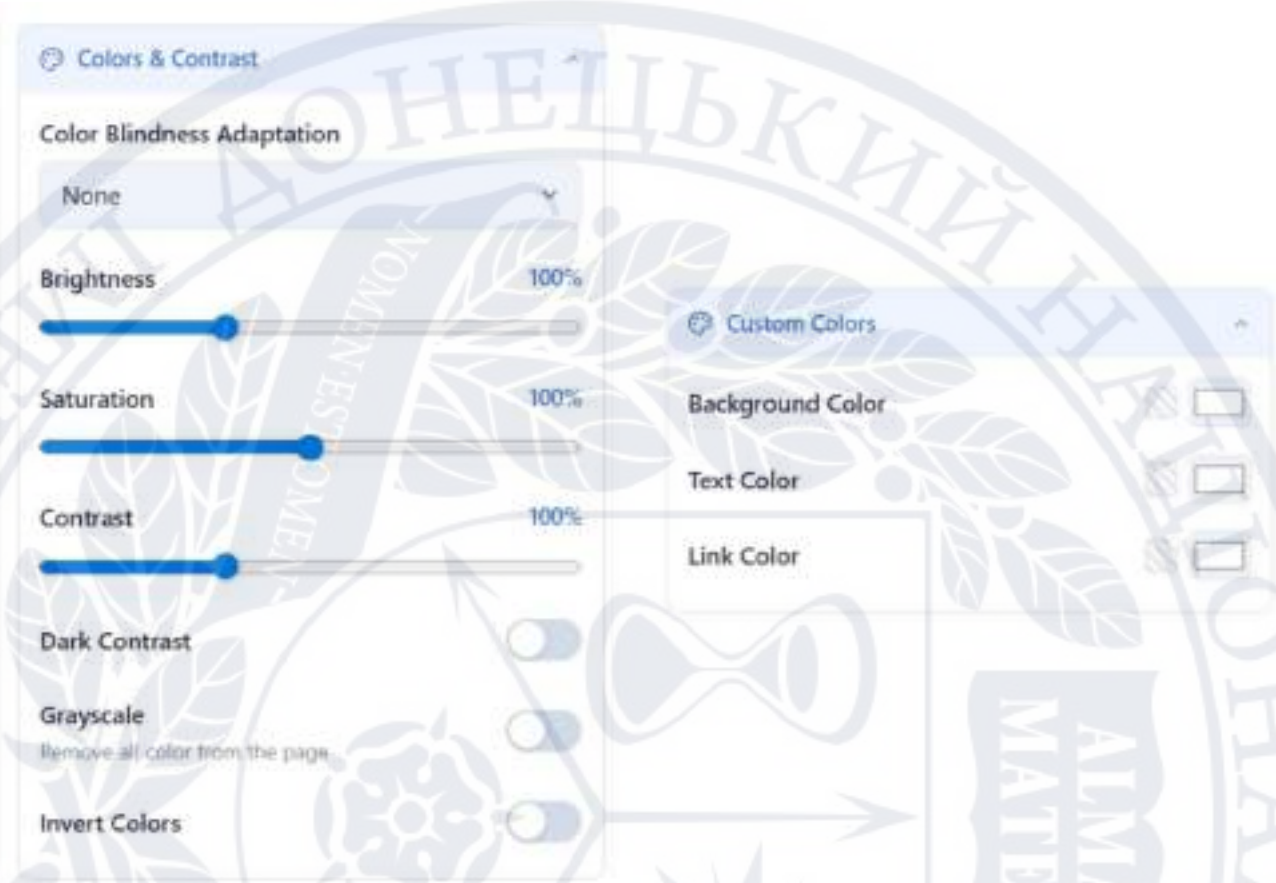

Рисунок 3.7 – Вигляд секції «Colors & Contrast» та «Custom Colors» відповідно

Поруч із селектором кольоросприйняття у секції розташовано два повзунки (які відповідають за яскравість, насиченість та контраст) і три перемикачі (для темної контрастності, теми у сірих тонах та інверсивності кольорів).

Технічно більшість з цих перетворень виконується через CSS-властивість filter, яка комбінує функції grayscale, invert, brightness, saturate і відповідне посилання на SVG-фільтр для того чи іншого типу дальтонізму. Усі CSS-правила інжектуються у тіньовий корінь модуля, але застосовуються до елемента body хост-сайту через каскад.

Окремо в межах вкладки виділено секцію «Custom Colors». У ній просунутий користувач може задати власні кольори через палітру (рис. 3.7).

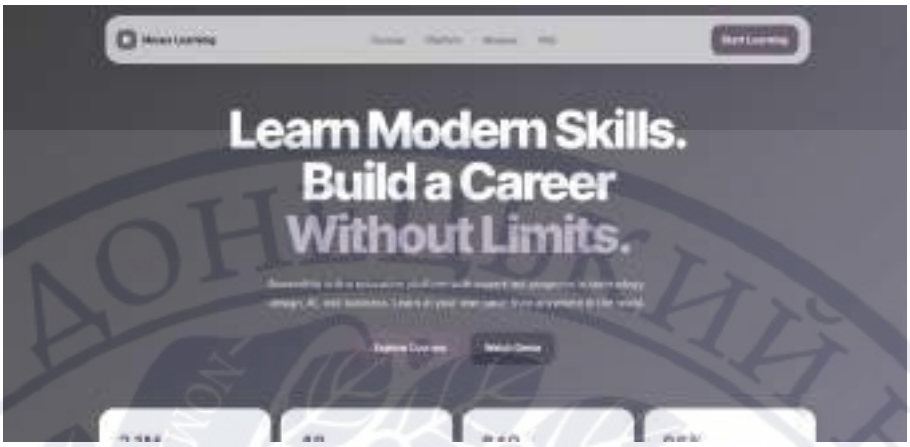

Рисунок 3.8 – Симуляція дейтеранопії на тестовій сторінці

На рис. 3.8 показано, як виглядає тестова сторінка з активованою симуляцією дейтеранопії. Кольори, у яких зелений компонент відіграє провідну роль, зливаються з іншими відтінками. Це показує, чому критерій 1.4.1 стандарту WCAG (не покладатись лише на колір як засіб передавання інформації) є дуже важливим [3]. Бо тестова сторінка побудована у більшості з акцентом на зеленому відтінку і при її дизайні не було враховано потрібної контрастності. Тому головна кнопка в початковій секції з написом «Explore Courses» зливається з фоном і майже непомітна. А другорядна кнопка навпаки – сильно виділяється. Це показує приклад того, як люди з вадами зору можуть отримувати некоректне сприйняття вебсайтів. На рис. 3.9–3.11 показано приклади з деяких інших режимів симуляції.

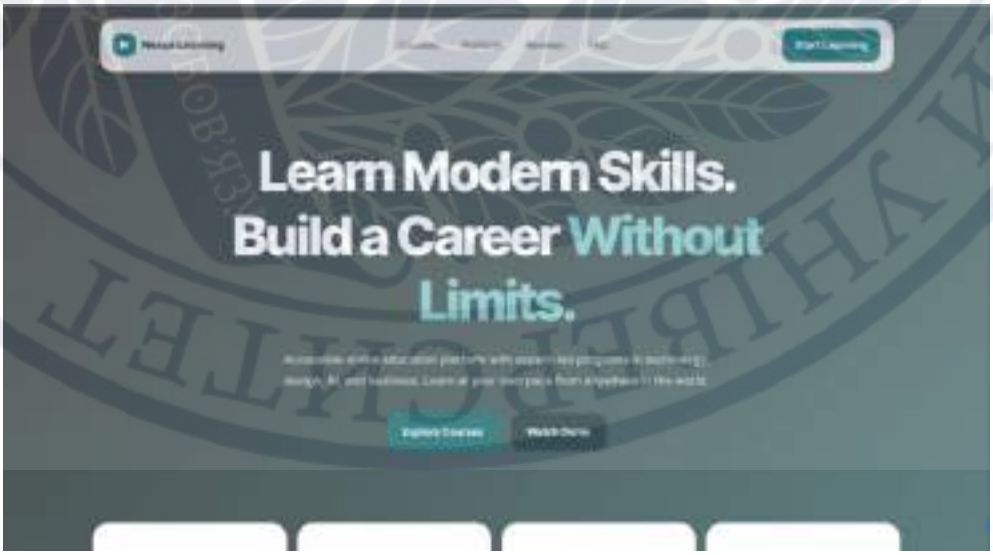

Рисунок 3.9 – Симуляція тританопії

Рисунок 3.10 – Симуляція протанопії

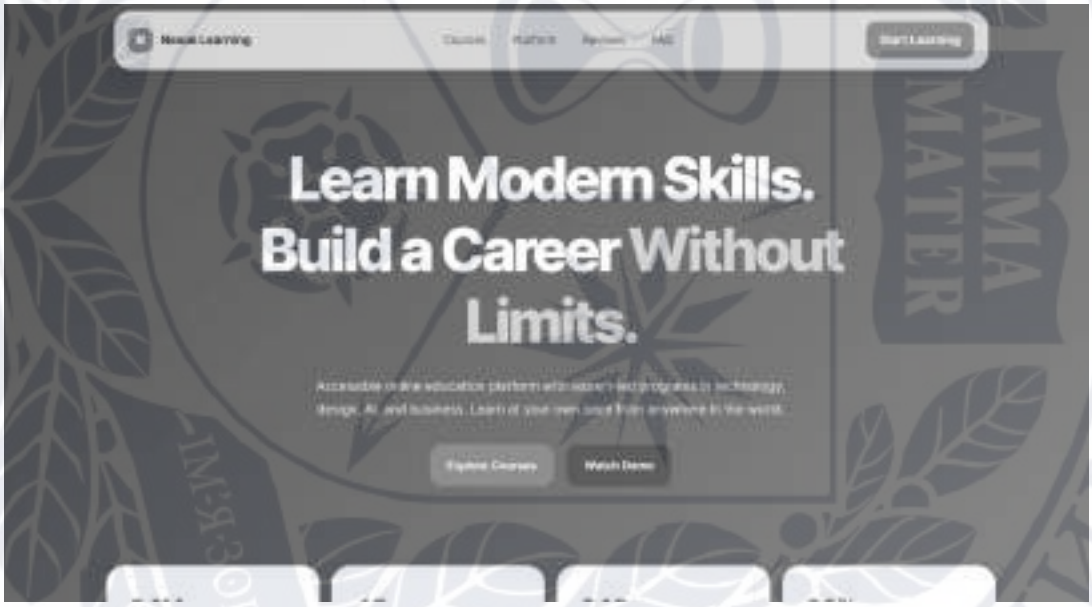

Рисунок 3.11 – Симуляція ахроматопсії

Дані режими представлені для тестування сторінки, тому що зробити уніфікований інструмент для цього неможливо. Хіба з можливим використанням штучного інтелекту як детектора і автоматизатора змін, але він все одно повинен бути провалідований людиною на кожному етапі. І схожих реалізацій саме в напрямку адаптації до різних форм дальтонізму знайдено не було.

3.2.4 Секції макета, анімацій та фокусу

Останні три секції першої вкладки згруповані за тематикою: «Layout & Readability» (макет сторінки), «Motion & Animations» (анімації та рух) і «Focus & Interaction» (видимість фокусу і допоміжні засоби для читання).

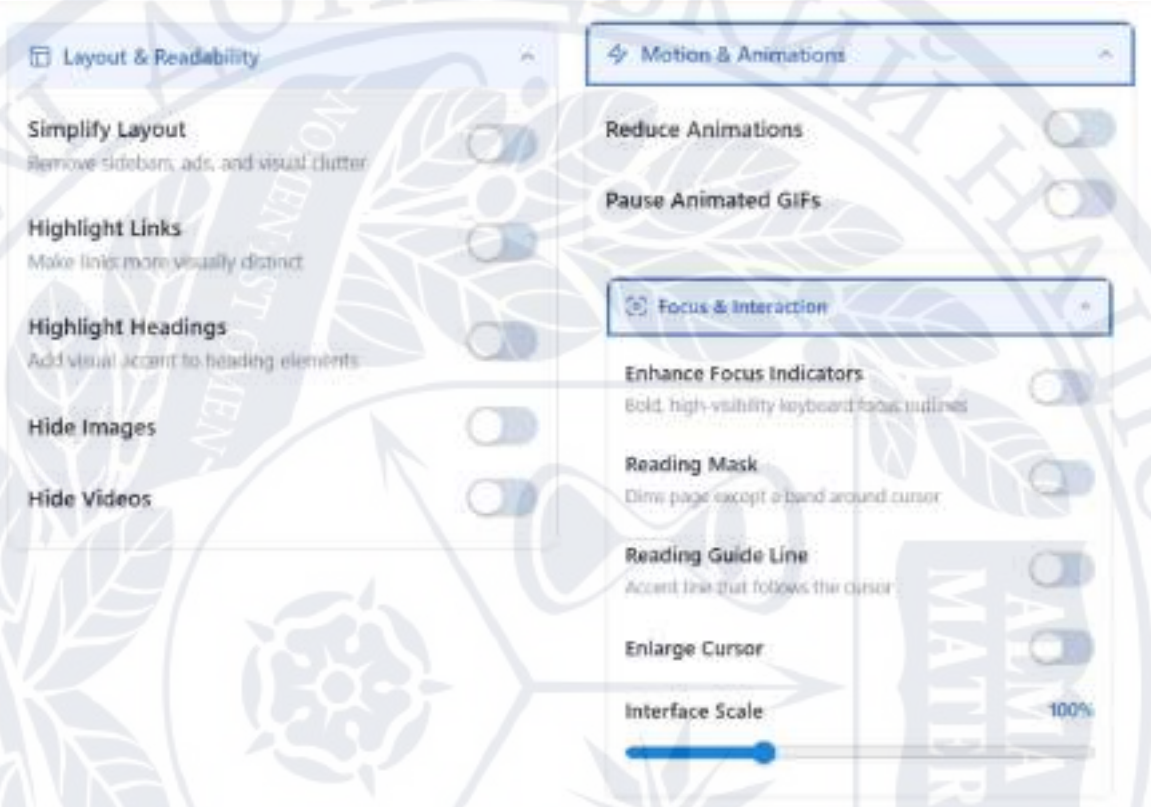

Рисунок 3.12 – Функціонал секцій «Layout & Readability», «Motion & Animations» та «Focus & Interaction»

Секція «Layout & Readability» (рис. 3.12) дозволяє користувачеві спростити верстку, приховати зображення або відео, виділити заголовки і посилання. Перший параметр приховує бічні панелі, рекламні блоки і другорядні регіони, лишаючи на сторінці лише основний контент. Це корисно для людей з порушеннями уваги – менше візуальних подразників означає простіше зосередження на головному. Приклад застосування цих налаштувань показано на рис. 3.13

Рисунок 3.13 – Прибирання зайвих елементів, виділення заголовків та посилань

Секція «Motion & Animations» (рис. 3.12) вимикає рух на сторінці. Прибираються всі CSS-анімації і переходи, глобальні анімації типу скролу чи перегортання слайдерів. «Pause GIFs» зупиняє GIF-зображення. При ініціалізації перевіряється системний медіазапит prefers-reduced-motion і автоматично пропонується користувачеві відповідні налаштування, якщо у нього на рівні операційної системи вже вибрано режим зменшеного руху. Оскільки візуально відобразити припинення анімацій у форматі зображення є неможливим, то на рис. 3.14 показано приклад того, як виглядає GIF після його відключення.

Рисунок 3.14 – Зупинка анімації GIF

Секція «Focus & Interaction» (рис. 3.12) спрямована на користувачів, які навігують клавіатурою або потребують додаткових візуальних підказок під час читання. Тут було додано такі опції:

- «Enhance focus» – посилення видимості рамки фокусу на активному елементі під час його виділення клавішою Tab (рамка щонайменше 3 пікселі з контрастом 3:1, що відповідає критерію 2.4.13 WCAG 2.2 [7]);
- «Reading mask» – горизонтальна маска з прозорим вікном кілька рядків заввишки, яка слідує за курсором (рис. 3.16);
- «Reading guide» – тонка горизонтальна лінія, що показує поточний рядок під курсором (рис. 3.16);
- «Enlarge cursor» – збільшення розміру курсора миші (рис. 3.17);
- «Interface scale» – збільшення масштабу самого інтерфейсу.

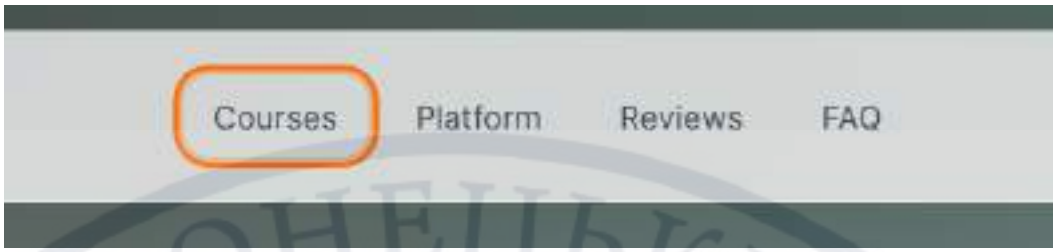

Рисунок 3.15 – Рамка фокусу під час виділення клавішою Tab

Рисунок 3.16 – Маска фокусу та лінія-гайд для читання

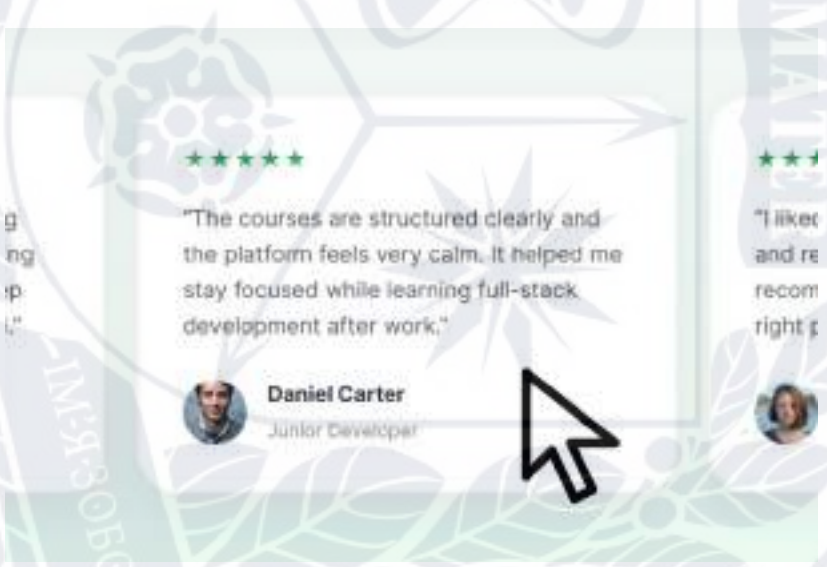

Рисунок 3.17 – Збільшений курсор миші

3.3 Спеціалізовані режими адаптації

Окремо від ручних налаштувань модуль містить готові режими адаптації. Це попередньо підготовлені набори параметрів, кожен з яких оптимізовано під конкретну категорію користувачів. Активація режиму відбувається, коли користувач відкриває вкладку «Modes» (рис. 3.18), обирає підходящу картку, клікаючи на неї, і всі параметри одразу застосовуються до сторінки.

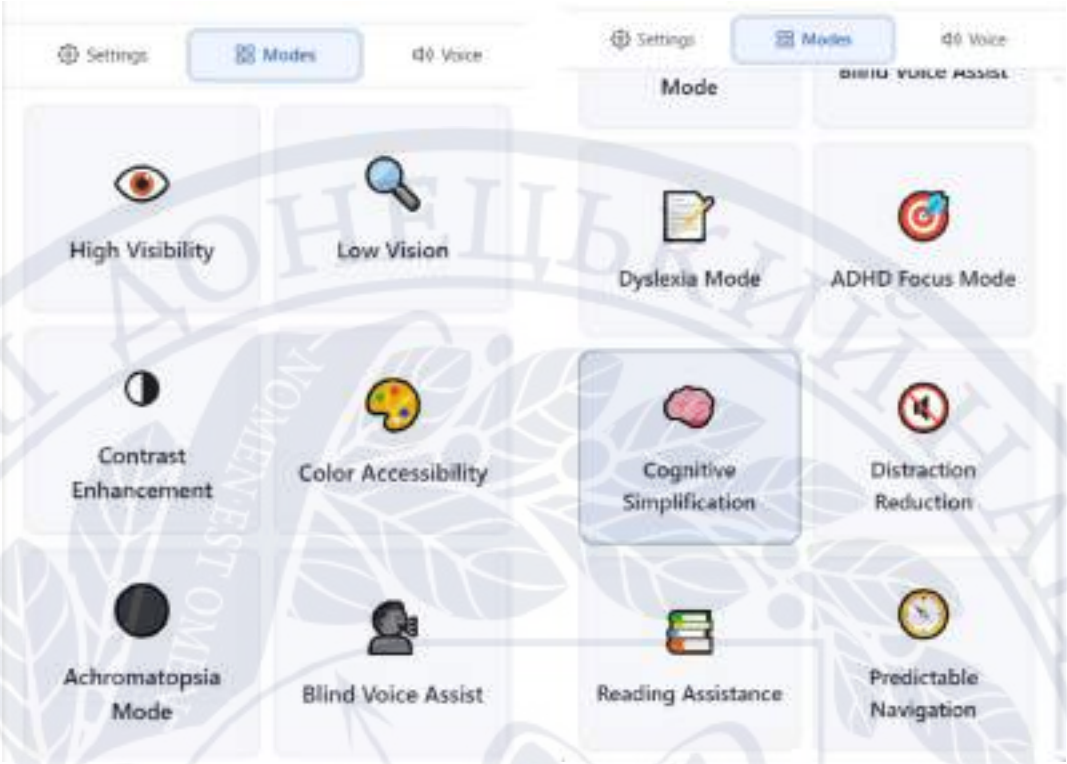

Рис. 3.18 – Вкладка «Modes» з картками готових режимів адаптації

Режими розділені на дві категорії: візуальні та когнітивні. Технічно вони описані у двох окремих файлах конфігурації (visualPresets.ts і cognitivePresets.ts), а керує ними окремий клас PresetManager.

Кожна картка містить піктограму та назву режиму. До цього був короткий опис, але було вирішено його прибрати для зменшення навантаження. Активний режим візуально виділяється рамкою з акцентним кольором. Деактивація відбувається зміною режиму або скиданням налаштувань.

3.3.1 Режим для користувачів з дислексією

Найбільш детально реалізовано режим для дислексії. Після активації він автоматично застосовує рекомендовані параметри читання: шрифт OpenDyslexic із резервними гарнітурами, збільшений розмір тексту, підвищена висота рядка, збільшені інтервали між літерами та словами, м’який кремовий фон і темно-сірий текст. Також обмежується ширина рядка, вимикаються анімації та активується reading guide для полегшення читання. Налаштування базуються на рекомендаціях Британської асоціації дислексії [14] та дослідженнях

спеціалізованих шрифтів [15]. За потреби користувач може додатково змінити параметри вручну у вкладці «Settings».

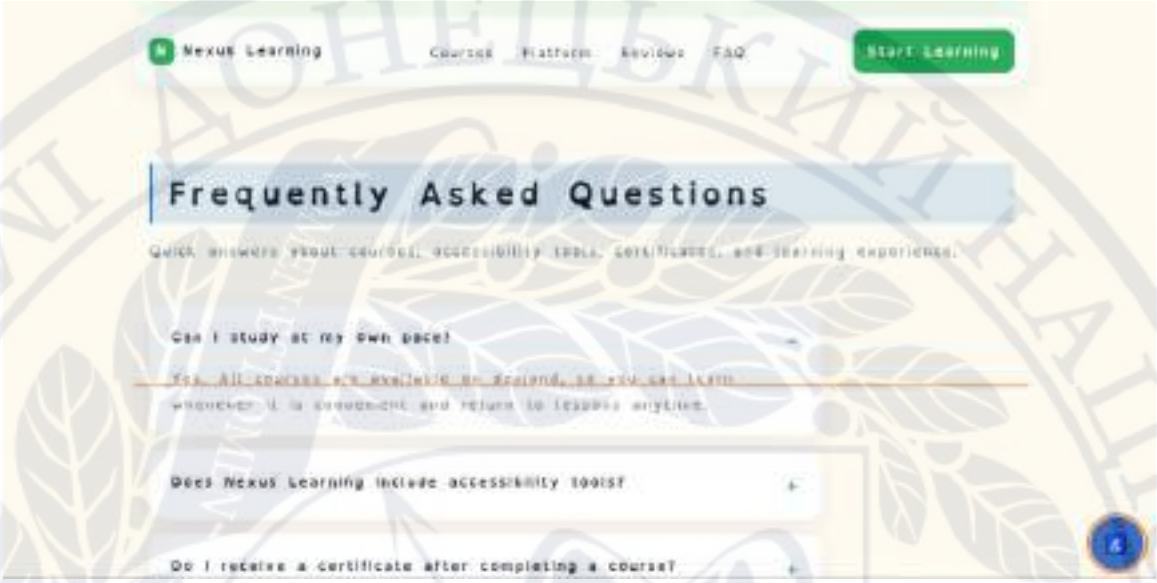

Рис. 3.19 – Тестова сторінка після активації режиму для дислексії

На рис. 3.19 показано тестову сторінку одразу після активації режиму. Гарнітура змінилась, з’явились комфортні інтервали, фон став м’якшим, активувалось підсвічування рядків.

3.3.2 Візуальні режими адаптації

Режим «High Visibility» (рис. 3.20) призначений для користувачів із серйозними порушеннями зору. У ньому збільшується розмір тексту до 110 %, підвищуються контраст, насиченість і яскравість, активуються highlight для посилань і заголовків, а інтерфейс масштабується до 110 % для зручнішої взаємодії. Також увімкнено посилений focus і клавіатурну навігацію.

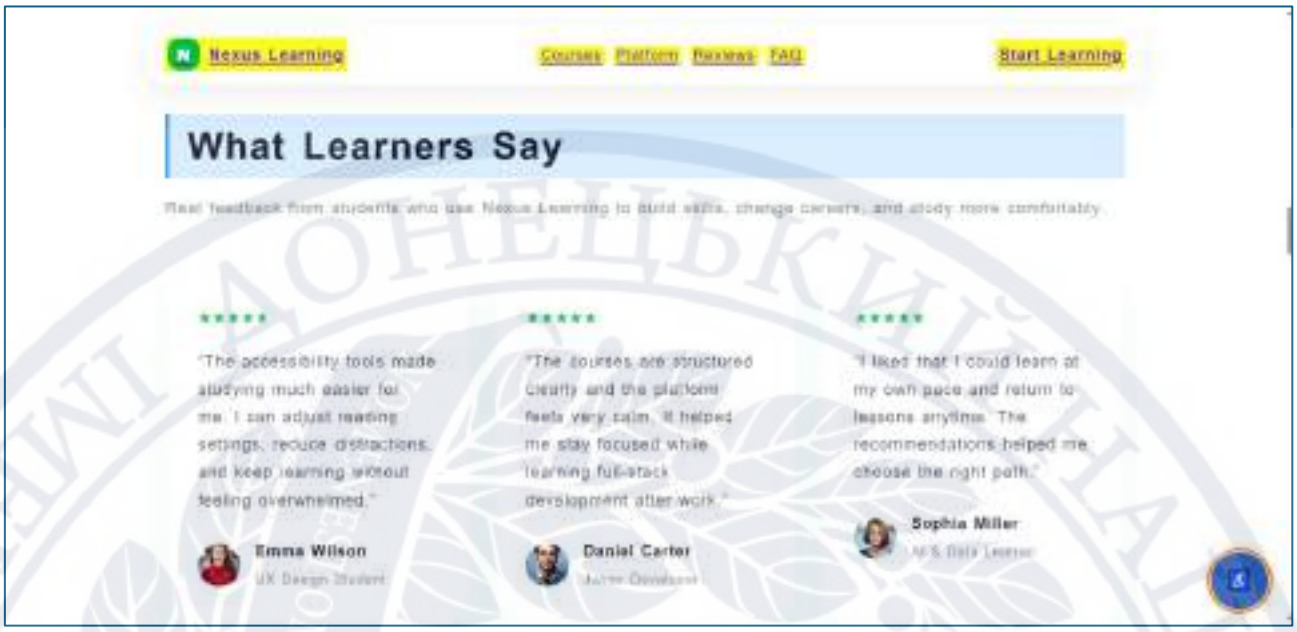

Рис. 3.20 – Тестова сторінка після активації режиму «High Visibility»

Режим «Low Vision» (рис. 3.21) адаптований для користувачів зі зниженою гостротою зору. Розмір тексту збільшується до 125 %, підвищуються міжрядкові та міжсимвольні інтервали, використовується м’який світлий фон (#fffef0), а також активуються reading guide, focus highlight і збільшений курсор.

Рис. 3.21 – Тестова сторінка після активації режиму «Low Vision»

Режим «Contrast Enhancement» (рис. 3.22) спрямований на підвищення контрасту тексту й тла без зміни шрифтів і макета. Збільшуються насиченість, яскравість, активуються користувацькі кольори (білий фон, майже чорний текст, темно-синій колір посилань для кращої впізнаваності).

Рис. 3.22 – Тестова сторінка після активації режиму «Contrast Enhancement»

Наступні режими призначені для людей з дальтонізмом (можна використати або доступність кольорів (3.23), або ахроматопсію для покращення видимості). Також додано окремо голосовий модуль для людей з відсутнім зором. Цей режим можна використовувати і при труднощах з навігацією за допомогою миші чи клавіатури.

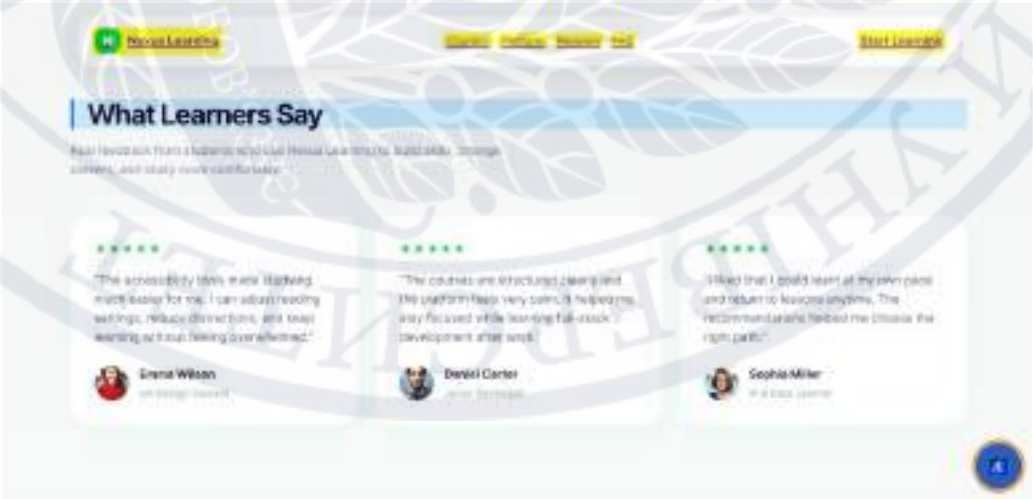

Рис. 3.23 – Тестова сторінка після активації режиму «Color Accessibility»

3.3.3 Когнітивні режими адаптації

До групи когнітивних адаптацій належать шість режимів: «Dyslexia Mode» (описаний вище), «ADHD Focus Mode», «Cognitive Simplification», «Distraction Reduction», «Reading Assistance» і «Predictable Navigation». Усі вони спираються на загальні рекомендації групи COGA консорціуму W3C [11].

Режим «ADHD Focus Mode» (рис. 3.24) спрямований на користувачів з розладом дефіциту уваги. Головний акцент – зниження кількості відволікаючих факторів. У режимі вмикається спрощення верстки, приховуються відео, вимикаються всі анімації, увімкнено маску фокусу читання. Кольори дещо приглушуються (насиченість 80 %).

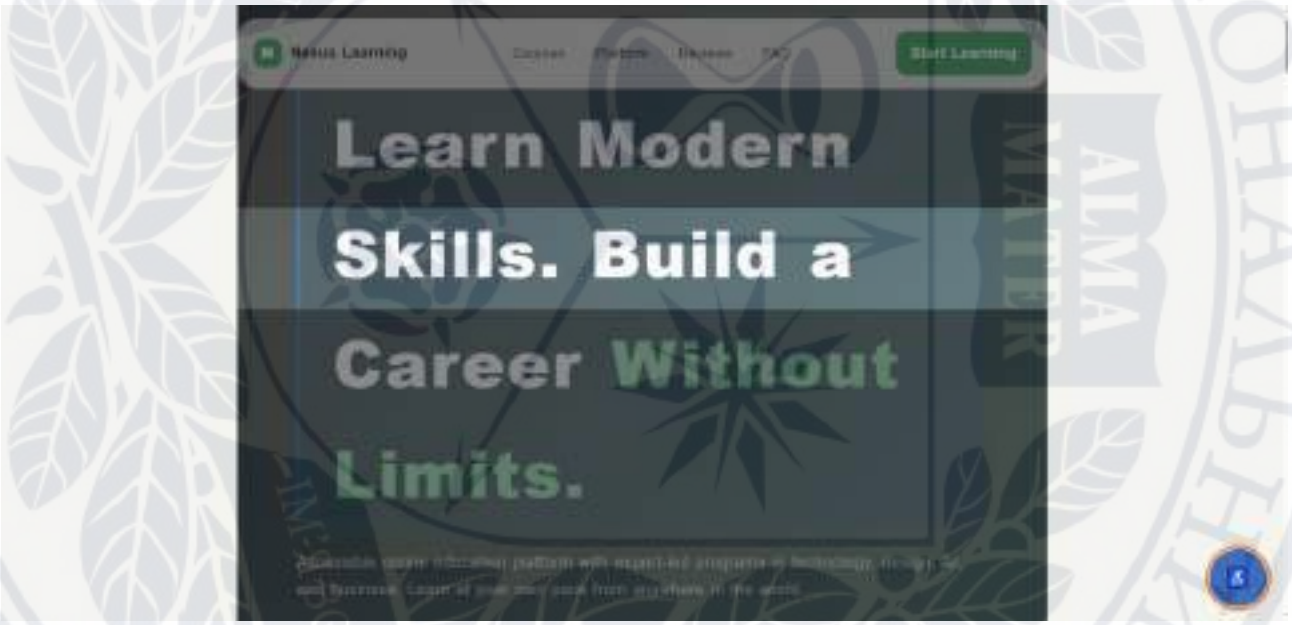

Рис. 3.24 – Тестова сторінка після активації режиму «ADHD Focus Mode»

Режим «Cognitive Simplification» (рис. 3.25) – більш узагальнений для різних когнітивних труднощів. Шрифт стає крупнішим (115 %) у читабельній гарнітурі, висота рядка збільшена (1,8), верстку спрощено, посилання й заголовки візуально виділено, анімації вимкнені.

Рис. 3.25 – Тестова сторінка після активації режиму «Cognitive Simplification»

Режим «Distraction Reduction» (рис. 3.26), окрім спрощення верстки, приховує всі зображення і відео, повністю вимикає рух і прозорість, зменшує насиченість кольорів до 60 %.

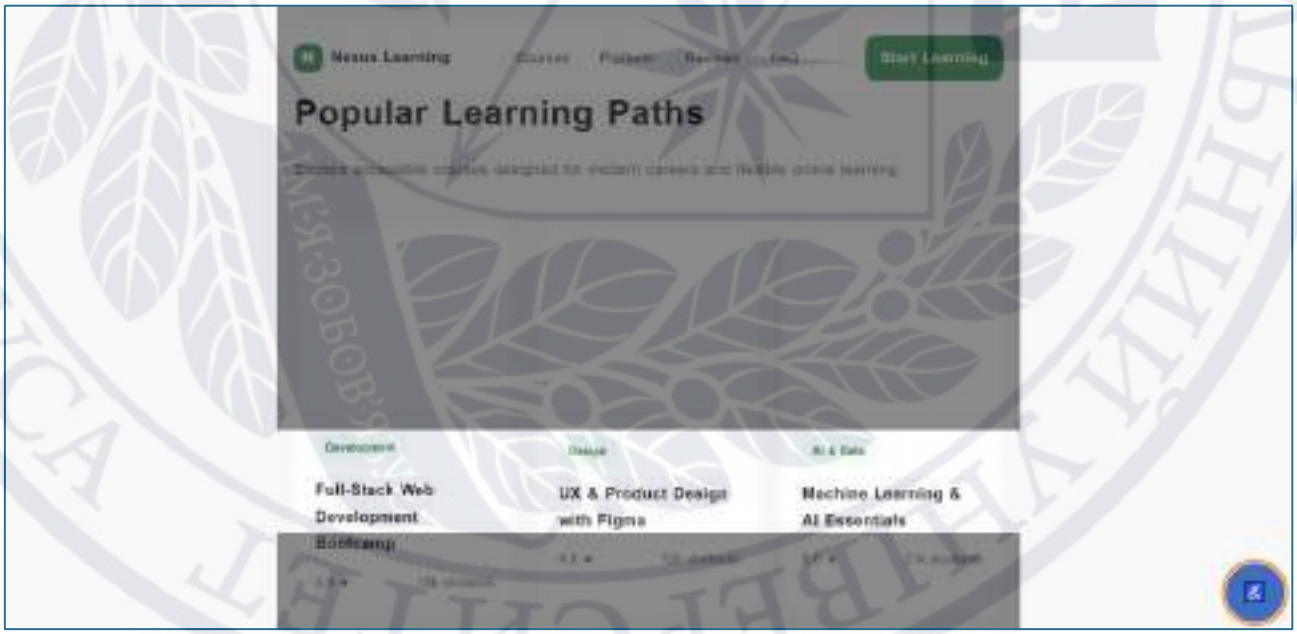

Рис. 3.26 – Тестова сторінка після активації режиму «Distraction Reduction»

Режим «Reading Assistance» (рис. 3.27) комбінує засоби допомоги при тривалому читанні: помірно збільшений шрифт, посилений фокус, увімкнено одночасно маску і guide.

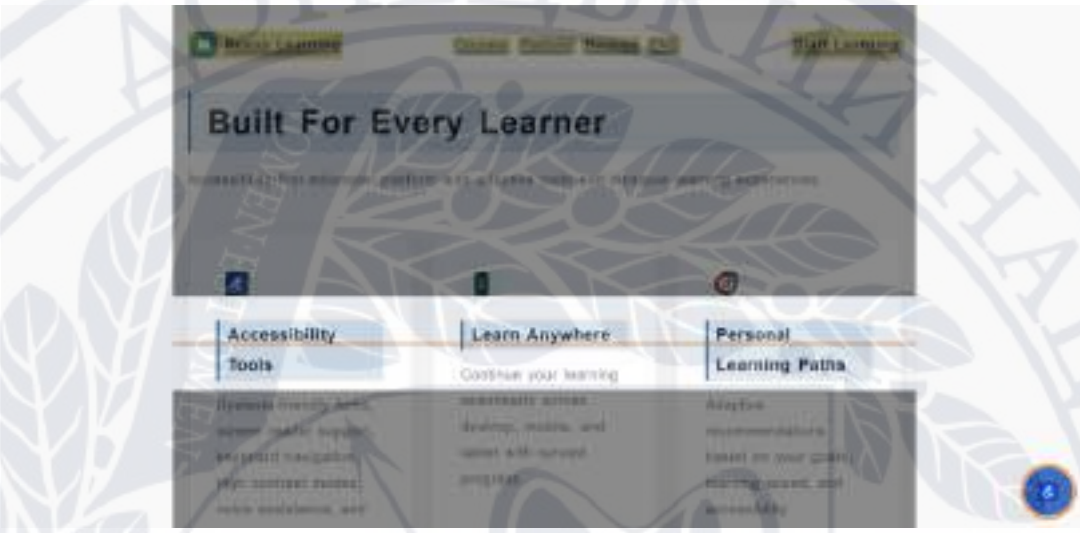

Рис. 3.27 – Тестова сторінка після активації режиму «Reading Assistance»

Режим «Predictable Navigation» (рис. 3.28) цілеспрямований на стабільність інтерфейсу. Вимикає анімації, посилює видимість фокусу, увімкнено підсвічування інтерактивних елементів для клавіатурної навігації. Призначений для користувачів, яким важлива передбачуваність – зокрема людей з розладами аутистичного спектра.

Рис. 3.27 – Тестова сторінка після активації режиму «Predictable Navigation»

Наявні режими були протестовані на коректність роботи згідно зі стандартами, відповідність схожого функціоналу на вже наявних продуктах, а також на невеликій фокус-групі з наявними особами з дислексією та РДУГ (3 людини).

3.3.4 Алгоритм застосування пресету

Активація режиму технічно зводиться до простої послідовності дій. Користувач натискає на картку, викликається відповідний метод сервісу AccessibilityService. Він звертається до PresetManager за повним об’єктом налаштувань, записує отриманий об’єкт у сховище стану, фіксує активний пресет і відправляє коротке повідомлення через aria-live регіон («Mode dyslexia activated», наприклад). Після цього вся подальша логіка така сама, як для будь-якої іншої зміни параметрів – рушії послідовно застосовують нові значення до сторінки.

Активація режиму не блокує ручні налаштування. Після того, як режим увімкнено, користувач може зайти у вкладку «Settings» і додатково підкоригувати окремі параметри, наприклад, зробити шрифт ще більшим або змінити колір фону. Пресет працює лише як стартова точка, а не як жорстко закріплений набір.

Подальший розвиток у цьому напрямку планується у двох площинах. По-перше, додавання нових пресетів. Зокрема для людей з частковою втратою зору, для літніх користувачів, для роботи у яскравому зовнішньому освітленні. По-друге, доопрацювання шару адаптерів, щоб вони давали користувачеві рекомендації щодо покращення інтерфейсу.

3.4 Голосовий модуль

Голосовий модуль (рис. 3.28) реалізований як окрема підсистема з трьох взаємодоповнюваних компонентів: служби синтезу мовлення (SpeechSynthesisService), служби розпізнавання голосу (SpeechRecognitionService) та обробника команд (CommandProcessor). Координацію між ними виконує клас VoiceEngine, який надає UI-шару єдиний

інтерфейс. Усі компоненти спираються виключно на стандартний Web Speech API, тож аудіодані не залишають пристрій користувача.

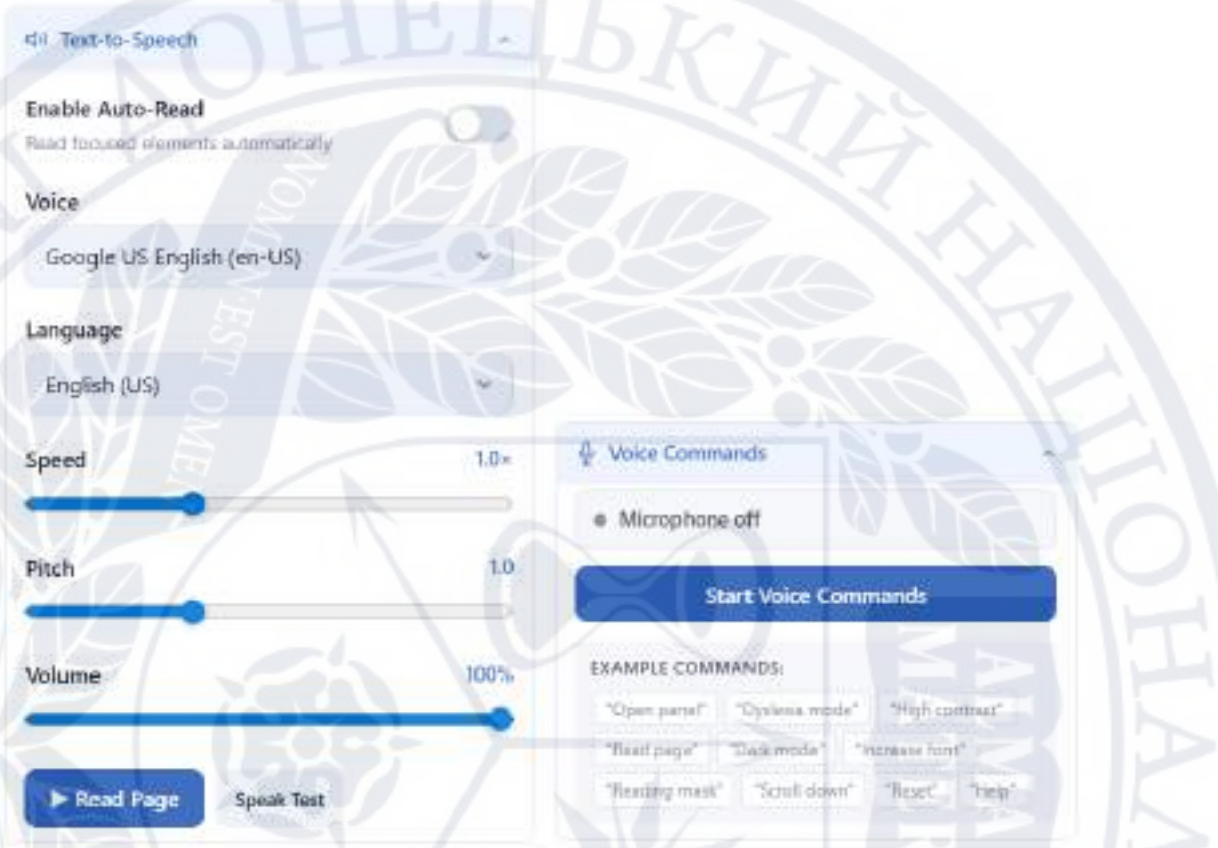

Рис. 3.28 – Голосовий модуль

Синтез мовлення обгортає інтерфейс SpeechSynthesis і виконує фільтрацію доступних голосів за мовою користувача, поділ довгих текстів на сегменти (близько 1800 символів) для обходу обмежень браузерних рушіїв, передачу параметрів rate, pitch і volume, а також безпечну зупинку та поновлення відтворення. Якщо обраної мови у системі немає, модуль автоматично повертається до резервної en-US, не перериваючи роботу.

Список голосів у списку вибору формується динамічно. VoiceProvider підписується на подію voiceschanged і фільтрує системні голоси за префіксом обраної мови (en, uk, fr, de тощо), завдяки чому користувач не бачить голосів, які не відповідають вибраній локалі. Усього вкладка «Voice» підтримує п’ятнадцять мовних варіантів, серед них українська, англійська (US/UK), німецька,

французька, іспанська, італійська, португальська, нідерландська, польська, японська, китайська, корейська, арабська.

Озвучення сторінки запускається з вкладки «Voice» (кнопка «Read page») або голосовою командою. Модуль вибирає область з основним контентом за пріоритетом, після чого обходить DOM рекурсивним TreeWalker'ом і збирає тільки видимі текстові вузли. Декоративні елементи ігноруються. Якщо користувач виділив фрагмент тексту або тримає фокус на конкретному елементі, читається саме цей вміст.

Розпізнавання голосових команд побудоване над SpeechRecognition API. Сервіс працює у режимі коротких висловлювань з автоматичним перезапуском. Поточний транскрипт виводиться у панелі у реальному часі (aria-live="polite"), а після завершення фрази CommandProcessor шукає у ній збіг із зареєстрованими тригерами. Команди розділені на дві групи – accessibilityCommands та navigationCommands. Кожна команда викликає той самий функціонал, що й кліки в інтерфейсі, тому поведінка модуля однакова незалежно від способу взаємодії.

Якщо у браузері відсутня підтримка SpeechSynthesis або SpeechRecognition (Safari старіших версій, частина мобільних оточень), відповідна секція показує повідомлення з role="alert" замість контролів.

Висновки до розділу 3

У третьому розділі описано практичну реалізацію адаптивного модуля доступності. Основну увагу приділено першій вкладці з ручними налаштуваннями – це найбільша частина модуля, через яку користувач отримує повний контроль над параметрами адаптації. Описано структуру вкладки, окремі секції (типографіка, кольори, макет, рух, фокус), реалізацію контролів, механізм оновлення стану і збереження налаштувань між сеансами. Окремо розглянуто питання локалізації інтерфейсу та доступності самої панелі – щоб інструмент адаптації не створював нових бар'єрів для своїх користувачів.

Описано також 12 спеціалізованих режимів адаптації, розділених на візуальні і когнітивні. Кожен пресет – це стартовий набір налаштувань,

оптимізований для конкретної категорії користувачів, який після активації не блокує можливість ручного доопрацювання. Також було реалізовано окремо голосовий модуль з озвучуванням та режимом голосових команд. Усі функції було протестовано на відповідність стандартам та коректність роботи та продемонстровано через зображення.

Розроблений модуль є робочою системою, яку можна підключити до довільного вебсайту одним рядком коду. На демонстраційній сторінці перевірено основні сценарії його використання – як для ручної підлаштовки інтерфейсу, так і для активації готових режимів. Реалізовані компоненти узгоджуються з вимогами стандарту WCAG 2.2 і рекомендаціями ARIA APG, що дозволяє вважати поточну версію модуля придатною для використання як учасниками тестування, так і у подальших ітераціях розробки.

ВИСНОВКИ

У межах кваліфікаційної роботи розв’язано задачу розроблення адаптивного програмного модуля доступності до інтернет-ресурсів для осіб з порушеннями зору та когнітивними особливостями сприйняття. Поєднано теоретичний аналіз предметної області, проєктування архітектури та практичну реалізацію з подальшою перевіркою відповідності модуля сучасним стандартам інклюзивного дизайну.

У першому розділі систематизовано бар’єри доступності за принципами POUR стандарту WCAG 2.2, розглянуто чинні міжнародні та національні документи (WCAG, WAI-ARIA, COGA, EN 301 549, ДСТУ EN 301 549:2022, постанова Кабінету Міністрів України № 757), проведено порівняльний аналіз чотирьох поширених готових рішень – UserWay, accessiBe, EqualWeb, AudioEye. Виявлено, що вони є дійсно гарними рішеннями, але жодне з них одночасно не задовольняє вимоги відкритості коду, передбачуваної поведінки та підконтрольності для розробника, тому обґрунтовано доцільність створення власного модуля.

У другому розділі спроєктовано архітектуру модуля за принципом ядра з підключуваними рушіями. Ядро AccessibilityCore відповідає за життєвий цикл модуля, керування станом і подіями, рушії (TypographyEngine, ColorEngine, LayoutEngine, MotionEngine, FocusEngine, ReadingMaskEngine, WidgetAdaptationEngine) виконують предметні зміни сторінки, а голосовий модуль додає альтернативний канал взаємодії. Завдяки централізованому сховищу станів на базі Zustand і шині подій компоненти зберігають слабке зв’язування.

У третьому розділі описано практичну реалізацію та проведено демонстраційне тестування модуля. Вкладка ручних налаштувань надає шість тематичних секцій (типографіка, кольори, власні кольори, макет, рух, фокус) з контрольованими параметрами; вкладка готових режимів пропонує дванадцять пресетів – шість візуальних (High Visibility, Low Vision, Contrast Enhancement,

Color Accessibility, Achromatopsia, Blind Voice Assist) та шість когнітивних (Dyslexia, ADHD Focus, Cognitive Simplification, Distraction Reduction, Reading Assistance, Predictable Navigation). Голосовий модуль забезпечує синтез мовлення з фільтрацією голосів за мовою та розпізнавання набору голосових команд через локальний Web Speech API.

Ізоляція стилів і подій досягнута завдяки Shadow DOM з режимом open і делегуванням фокуса, а внутрішній стилевий каскад керується StyleInjector з підтримкою CSP-nonce. Динамічна вставка CSS-правил, SVG-фільтри для симуляції дальтонізму та власні правила для темного контрасту реалізують вимоги до адаптивної типографіки, контрастності та підтримки користувачів з порушеннями кольоросприйняття.

Модуль безпосередньо підтримує користувачів з порушеннями зору і когнітивними особливостями. Для слабкозорих передбачено пресет Low Vision з підвищеним розміром тексту, посиленим фокусом і збільшеним курсором; для повної втрати зору – режим Blind Voice Assist з автоматичним озвученням сфокусованого вмісту. Для дислексії та РДУГ застосовуються спеціальні режими; Усіх їх можна доналаштовувати вручну.

Модуль придатний до інтеграції у різні типи сайтів одним рядком підключення скрипта без зміни серверної частини, що підтверджено демонстраційним розгортанням на тестовій сторінці. Закладена архітектура дозволяє масштабувати рішення у бік окремого мікросервісу або вбудовуваного компонента мобільних застосунків. Це робить розроблений модуль практично значущим інструментом для власників публічних сайтів, освітніх платформ тощо.

Поставлені мета і завдання роботи виконано в повному обсязі. Розроблений програмний модуль відповідає актуальним стандартам вебдоступності, поєднує засоби візуальної адаптації, когнітивної підтримки та голосової взаємодії, є відкритим для подальшого розвитку і може використовуватися як практичний інструмент покращення доступу до інтернет-ресурсів широкому колу користувачів з особливими потребами.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The WebAIM Million: An accessibility analysis of the top 1,000,000 home pages. WebAIM. 2025. URL: https://webaim.org/projects/million/ (дата звернення: 08.05.2026).

2. Population with disability – Statistics Explained. Eurostat. 2024. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Population_with_disability (дата звернення: 09.05.2026).

3. FTC Approves Final Order Requiring accessiBe to pay $1 Million for Misleading Claims. Federal Trade Commission, April 2025. URL: https://www.ftc.gov/news-events/news/press-releases/2025/04/ftc-approves-final-order-requiring-accessibe-pay-1-million (дата звернення: 02.05.2026).

4. Userway faces class action lawsuit over alleged false accessibility and ADA compliance claims. Access by Design, 2024. URL: https://accessbydesign.uk/userway-faces-class-action-lawsuit-over-alleged-false-accessibility-and-ada-compliance-claims/ (дата звернення: 03.05.2026).

5. Деякі питання доступності інформаційно-комунікаційних систем та документів в електронній формі: постанова Кабінету Міністрів України від 21.07.2023 № 757. URL: https://www.kmu.gov.ua/npas/deiaki-pytannia-dostupnosti-informatsiino-komunikatsiinykh-system-ta-dokumentiv-v-elektronnii-formi-i210723-757 (дата звернення: 07.05.2026).

6. ДСТУ EN 301 549:2022. Інформаційні технології. Вимоги щодо доступності продуктів та послуг ІКТ (EN 301 549 V3.2.1 (2021-03), IDT). Київ: ДП «УкрНДНЦ», 2022. URL: https://zakon.isu.net.ua/sites/default/files/normdocs/dstu_en_301_549_2022_informaciyni_tekhnologii._vimogi_schodo_.pdf

7. Web Content Accessibility Guidelines (WCAG) 2.2. W3C Recommendation, 5 October 2023. URL: https://www.w3.org/TR/WCAG22/ (дата звернення: 07.05.2026).

8. Афанасьєва Д. С., Січко Т. В. Аналітичне дослідження бар'єрів доступності вебресурсів для осіб з порушеннями зору та когнітивними

особливостями. Вимірювальна та обчислювальна техніка в технологічних процесах, номер 2, 2026. (довідка прийняття статті до публікації у номер 2, 2026, журналу категорії Б)

9. Афанасьєва Д. С., Веселовська Н. Р. Алгоритмічні методи адаптації вебконтенту для людей з вадами зору: тези доповіді. VI Всеукраїнська науково-практична конференція «Комп'ютерні технології обробки даних», 2025.

10. Афанасьєва Д. С., Веселовська Н. Р. Використання методів структурного аналізу вебресурсів для адаптації до когнітивних особливостей користувачів. Вісник Донецького національного університету імені Василя Стуса. Вінниця. Випуск 18. Том 1. 2026. URL: https://jvestnik-sss.donnu.edu.ua/issue/view/682

11. Making Content Usable for People with Cognitive and Learning Disabilities. W3C Working Group Note. URL: https://www.w3.org/TR/coga-usable/ (дата звернення: 07.05.2026).

12. The WebAIM Million: An accessibility analysis of the top 1,000,000 home pages. WebAIM. 2025. URL: https://webaim.org/projects/million/ (дата звернення: 10.05.2026).

13. Dysfont. Official Dysfont Website. 2026. URL: https://www.dysfont.com/ (дата звернення: 11.05.2026).

14. British Dyslexia Association. Official website. 2026. URL: https://www.bdadyslexia.org.uk/ (дата звернення: 13.05.2026).

15. Wery J. J., Diliberto J. A. The effect of a specialized dyslexia font, OpenDyslexic, on reading rate and accuracy. Annals of Dyslexia. 2017. Vol. 67. P. 114–127. DOI: https://doi.org/10.1007/s11881-016-0127-1 .

16. Web Accessibility Directive: Implementation and monitoring. European Commission. URL: https://digital-strategy.ec.europa.eu/en/policies/web-accessibility (дата звернення: 02.05.2026).

17. Web Content Accessibility Guidelines (WCAG) 2.1. W3C Recommendation. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 12.05.2026).

18. Accessible Rich Internet Applications (WAI-ARIA) 1.2. W3C Recommendation. URL: https://www.w3.org/TR/wai-aria-1.2/ (дата звернення: 08.05.2026).

19. ARIA Authoring Practices Guide (APG): Dialog (Modal) Pattern. W3C Web Accessibility Initiative. URL: https://www.w3.org/WAI/ARIA/apg/patterns/dialog-modal/ (дата звернення: 07.05.2026).

20. UserWay – Web Accessibility Solution. URL: https://userway.org/ (дата звернення: 09.05.2026).

21. accessiBe – Web accessibility solutions. URL: https://accessibe.com/ (дата звернення: 09.05.2026).

22. AudioEye – Web Accessibility Platform. URL: https://www.audioeye.com/ (дата звернення: 10.05.2026).

23. EqualWeb – Web Accessibility Solutions. URL: https://www.equalweb.com/ (дата звернення: 11.05.2026).

24. TypeScript Handbook. Microsoft. URL: https://www.typescriptlang.org/docs/handbook/intro.html (дата звернення: 15.05.2026).

25. React – A JavaScript library for building user interfaces. Meta Open Source. URL: https://react.dev/ (дата звернення: 16.05.2026).

26. Zustand – Bear necessities for state management in React. GitHub. URL: https://github.com/pmndrs/zustand (дата звернення: 16.05.2026).

27. Vite – Next Generation Frontend Tooling. URL: https://vitejs.dev/ (дата звернення: 14.05.2026).

28. Web Speech API. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API (дата звернення: 07.05.2026).

29. localStorage – Web Storage API. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage (дата звернення: 07.05.2026).

30. prefers-reduced-motion CSS media feature. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/CSS/@media/prefers-reduced-motion (дата звернення: 10.05.2026).

31. Using shadow DOM. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_components/Using_shadow_DOM (дата звернення: 06.05.2026).

32. Content Security Policy (CSP). MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy (дата звернення: 04.05.2026).

33. MutationObserver – DOM Standard. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver (дата звернення: 08.05.2026).

34. Acosta-Vargas P., Acosta T., Luján-Mora S. Challenges to assess accessibility in higher education websites: A comparative study of Latin America universities. IEEE Access. Vol. 6, pp. 36500–36508. 2018. DOI: https://doi.org/10.1109/ACCESS.2018.2848978

35. Esposito A., Dedò S., Soares Guedes L., Landoni M. Advancing Accessible Interfaces: Evaluation of Design Patterns and Recommendations for Users with Intellectual Disabilities. The 11th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion. New York, USA, 2025. P. 408–417. DOI: https://doi.org/10.1145/3696593.3696613

36. axe-core: Accessibility engine for automated Web UI testing. Deque Systems. GitHub. URL: https://github.com/dequelabs/axe-core (дата звернення: 07.05.2026).

37. Lighthouse: Automated tool for improving the quality of web pages. Google. URL: https://developer.chrome.com/docs/lighthouse/overview (дата звернення: 07.05.2026).

38. WAVE Web Accessibility Evaluation Tool. WebAIM. URL: https://wave.webaim.org/ (дата звернення: 07.05.2026).

39. Web Components. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_components (дата звернення: 07.05.2026).

40. Subresource Integrity. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/Security/Defenses/Subresource_Integrity (дата звернення: 07.05.2026).

41. Афанасьєва Д. С. Впровадження адаптивного дизайну з React: техніки для кросплатформної сумісності. Вісник студентського наукового товариства ДонНУ імені Василя Стуса. – 2024 – Т. 1, № 16 – URL: https://jvestnik-sss.donnu.edu.ua/article/view/15806

ДОДАТКИ

ДОДАТОК А

Архітектура та конфігурація (основна частина)

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';
import { resolve } from 'path';

export default defineConfig({
  plugins: [react()],
  resolve: {
    alias: {
      '@': resolve(__dirname, 'src'),
      '@core': resolve(__dirname, 'src/core'),
      '@types': resolve(__dirname, 'src/types'),
      '@constants': resolve(__dirname, 'src/constants'),
      '@config': resolve(__dirname, 'src/config'),
      '@state': resolve(__dirname, 'src/state'),
      '@storage': resolve(__dirname, 'src/storage'),
      '@engines': resolve(__dirname, 'src/engines'),
      '@accessibility': resolve(__dirname, 'src/accessibility'),
      '@presets': resolve(__dirname, 'src/presets'),
      '@themes': resolve(__dirname, 'src/themes'),
      '@voice': resolve(__dirname, 'src/voice'),
      '@services': resolve(__dirname, 'src/services'),
      '@providers': resolve(__dirname, 'src/providers'),
      '@hooks': resolve(__dirname, 'src/hooks'),
      '@components': resolve(__dirname, 'src/components'),
      '@utils': resolve(__dirname, 'src/utils'),
      '@styles': resolve(__dirname, 'src/styles'),
      '@integration': resolve(__dirname, 'src/integration'),
```

```
    },
  },
  css: {
    modules: {
      localsConvention: 'camelCase',
    },
  },
  build: {
    outDir: 'dist/embed',
    lib: {
      entry: resolve(__dirname, 'src/embed.ts'),
      name: 'AdaptiveAccessibility',
      fileName: 'adaptive-accessibility',
      formats: ['iife', 'umd'],
    },
    rollupOptions: {
      // React bundled in – no external deps for the embed build
      external: [],
      output: {
        globals: {},
        // Inline all CSS into JS so the embed is a single file
        assetFileNames: '[name][extname]',
      },
    },
    sourcemap: true,
    minify: 'terser',
  },
});
```

```
import type { ModuleConfig } from '../types/config';
import { DEFAULT_CONFIG } from './defaultConfig';

export class ConfigManager {
  private static instance: ConfigManager | null = null;
  private readonly config: Readonly<ModuleConfig>;

  private constructor(overrides: Partial<ModuleConfig> = {}) {
    this.config = Object.freeze(this.merge(DEFAULT_CONFIG, overrides));
  }

  static init(overrides?: Partial<ModuleConfig>): ConfigManager {
    if (!ConfigManager.instance) {
      ConfigManager.instance = new ConfigManager(overrides);
    }
    return ConfigManager.instance;
  }

  static getInstance(): ConfigManager {
    if (!ConfigManager.instance) {
      ConfigManager.instance = new ConfigManager();
    }
    return ConfigManager.instance;
  }

  static reset(): void {
    ConfigManager.instance = null;
  }
```

```
  get<K extends keyof ModuleConfig>(key: K): ModuleConfig[K] {
    return this.config[key];
  }
  getAll(): Readonly<ModuleConfig> {
    return this.config;
  }

  private merge(base: ModuleConfig, overrides: Partial<ModuleConfig>): ModuleConfig {
    return {
      ...base,
      ...overrides,
      widgetPosition: { ...base.widgetPosition, ...overrides.widgetPosition },
      enabledTabs: { ...base.enabledTabs, ...overrides.enabledTabs },
    };}}
```

```
import type { ModuleConfig } from '../types/config';
export const DEFAULT_CONFIG: ModuleConfig = {
  mountTarget: '#adaptive-accessibility-root',
  shadowDOM: true,
  storage: 'localStorage',
  storageKey: 'adaptive-accessibility',
  applyScope: 'html',
  widgetPosition: { vertical: 'bottom', horizontal: 'right' },
  showToggleButton: true,
  openPanelOnInit: false,
  enabledTabs: { settings: true, modes: true, voice: true },
  voiceEnabled: true,
  loadSavedPreferences: true,
```

```
  zIndex: 999999,
  debug: false,};
```

```
import { AccessibilityProvider } from './providers/AccessibilityProvider';
import { ThemeProvider } from './providers/ThemeProvider';
import { VoiceProvider } from './providers/VoiceProvider';
import { AccessibilityWidget } from './components/Widget/AccessibilityWidget';
import { ErrorBoundary } from './components/ErrorBoundary';
export function App() {
  return (
    <ErrorBoundary>
      <ThemeProvider>
        <AccessibilityProvider>
          <VoiceProvider>
            <AccessibilityWidget />
          </VoiceProvider>
        </AccessibilityProvider>
      </ThemeProvider>
    </ErrorBoundary>
  );}
```

ДОДАТОК Б

Ядро accessibility-логіки та приклад адаптера (основна частина)

```
import type { AccessibilitySettings } from '../types/accessibility';

export interface AccessibilityIssue {
  severity: 'critical' | 'major' | 'minor';
  wcagCriterion: string;
  message: string;
  recommendation: string;
}
export interface AccessibilityAssessment {
  isOptimal: boolean;
  score: number;
  issues: AccessibilityIssue[];
  suggestions: string[];
}
export interface WCAGCriterion {
  id: string;
  level: 'A' | 'AA' | 'AAA';
  name: string;
  description: string;
}
export interface IAccessibilityAdapter {
  readonly name: string;
  getOptimalSettings(): Partial<AccessibilitySettings>;
  assess(settings: AccessibilitySettings): AccessibilityAssessment;
  getWCAGCriteria(): WCAGCriterion[];
  meetsMinimumStandards(settings: AccessibilitySettings): boolean;}
```

```
export type {
  IAccessibilityAdapter,
  AccessibilityAssessment,
  AccessibilityIssue,
  WCAGCriterion,
} from './types';

export { DyslexiaAdapter, dyslexiaAdapter } from './DyslexiaAdapter';
export { ADHDAdapter, adhdAdapter } from './ADHDAdapter';
export { LowVisionAdapter, lowVisionAdapter } from './LowVisionAdapter';
export { CognitiveAdapter, cognitiveAdapter } from './CognitiveAdapter';
export { ColorBlindnessAdapter, colorBlindnessAdapter } from
'./ColorBlindnessAdapter';
```

```
import type { AccessibilitySettings } from '../types/accessibility';
import { ConfigManager } from '../config/ConfigManager';
import { StorageService } from '../storage/StorageService';
import { useAccessibilityStore } from '../state/store';
import { eventBus } from './EventBus';
import { Logger } from '../utils/logger';

export class AccessibilityCore {
  private static instance: AccessibilityCore | null = null;
  private readonly config: ConfigManager;
  private readonly storage: StorageService;
  private readonly logger: Logger;
  private unsubscribers: Array<() => void> = [];
  private saveDebounceTimer: ReturnType<typeof setTimeout> | null = null;
```

```
private constructor(config: ConfigManager) {
  this.config = config;
  this.storage = new StorageService(
    config.get('storage'),
    config.get('storageKey')
  );
  this.logger = new Logger('Core', config.get('debug'));
}

static getInstance(config?: ConfigManager): AccessibilityCore {
  if (!AccessibilityCore.instance) {
    AccessibilityCore.instance = new AccessibilityCore(
      config ?? ConfigManager.getInstance()
    );
  }
  return AccessibilityCore.instance;
}

static reset(): void {
  AccessibilityCore.instance?.destroy();
  AccessibilityCore.instance = null;
}

async init(): Promise<void> {
  this.logger.log('Initializing...');
  const store = useAccessibilityStore.getState();

  if (this.config.get('loadSavedPreferences')) {
    const saved = await this.storage.loadSettings();
```

```
      if (saved) {
        store.setSettings(saved);
        this.logger.log('Loaded saved preferences.');
      }
    }

    const initialOverride = this.config.get('initialSettings');
    if (initialOverride) {
      const merged = this.mergeSettings(store.settings, initialOverride);
      store.setSettings(merged);
    }

    if (this.config.get('openPanelOnInit')) {
      store.openPanel();
    }

    this.subscribeToStateChanges();
    store.setInitialized(true);

    eventBus.emit('core:ready', {});
    this.logger.log('Ready.');
  }

  destroy(): void {
    this.unsubscribers.forEach(fn => fn());
    this.unsubscribers = [];

    // Flush any pending save before destroying – prevents losing the last settings
change
```

```
    if (this.saveDebounceTimer) {
      clearTimeout(this.saveDebounceTimer);
      this.saveDebounceTimer = null;
      const { settings } = useAccessibilityStore.getState();
      this.storage.saveSettings(settings).catch(() => {});
    }

    eventBus.emit('core:destroy', {});
    eventBus.removeAllListeners();
    this.logger.log('Destroyed.');
  }

  async resetPreferences(): Promise<void> {
    await this.storage.clearSettings();
    useAccessibilityStore.getState().resetSettings();
    this.logger.log('Preferences reset.');
  }

  private subscribeToStateChanges(): void {
    // Persist settings whenever they change (debounced to avoid thrashing storage)
    const unsubSettings = useAccessibilityStore.subscribe(
      (state) => state.settings,
      (settings: AccessibilitySettings) => {
        this.scheduleSave(settings);
        eventBus.emit('settings:change', {
          settings,
          changed: settings,
        });
      }
```

```
  );

  this.unsubscribers.push(unsubSettings);
}

private scheduleSave(settings: AccessibilitySettings): void {
  if (this.saveDebounceTimer) clearTimeout(this.saveDebounceTimer);
  this.saveDebounceTimer = setTimeout(async () => {
    try {
      await this.storage.saveSettings(settings);
      this.logger.log('Settings persisted.');
    } catch (err) {
      this.logger.error('Failed to persist settings:', err);
    }
  }, 500);
}

private mergeSettings(
  current: AccessibilitySettings,
  partial: Partial<AccessibilitySettings>
): AccessibilitySettings {
  return {
    ...current,
    ...partial,
    typography: { ...current.typography, ...partial.typography },
    color: { ...current.color, ...partial.color },
    layout: { ...current.layout, ...partial.layout },
    motion: { ...current.motion, ...partial.motion },
    interaction: { ...current.interaction, ...partial.interaction },
```

```
    voice: { ...current.voice, ...partial.voice },
  }; }}
```

```
import type { AccessibilitySettings } from '../types/accessibility';
import type {
  IAccessibilityAdapter,
  AccessibilityAssessment,
  AccessibilityIssue,
  WCAGCriterion,
} from './types';
import { hexToRgb, relativeLuminance, contrastRatio } from '../utils/color';

export class LowVisionAdapter implements IAccessibilityAdapter {
  readonly name = 'Low Vision Support';

  static readonly WCAG_AA_CONTRAST = 4.5;
  static readonly WCAG_AAA_CONTRAST = 7.0;

  getOptimalSettings(): Partial<AccessibilitySettings> {
    return {
      typography: {
        fontFamily: 'readable',
        fontSize: 130,
        lineHeight: 1.8,
        letterSpacing: 0.04,
        wordSpacing: 0.1,
        textScale: 1,
      },
      color: {
```

```
    contrast: 100,
    saturation: 110,
    brightness: 105,
    colorBlindnessMode: 'none',
    grayscale: false,
    invertColors: false,
    darkMode: false,
    customBackground: '#fffef0',
    customText: '#000000',
    customLinks: null,
  },
  layout: {
    readingWidth: 70,
    simplifyLayout: false,
    hideImages: false,
    hideVideos: false,
    highlightLinks: true,
    highlightHeadings: true,
    textAlign: 'left',
  },
  motion: {
    reduceMotion: false,
    pauseGifs: false,
    reduceTransparency: false,
  },
  interaction: {
    enlargeCursor: true,
    enhanceFocus: true,
    keyboardNavigation: true,
```

```
      readingMask: false,
      readingGuide: true,
      focusHighlight: true,
      interfaceScale: 120,
    },
  };
}

 assess(settings: AccessibilitySettings): AccessibilityAssessment {
  const issues: AccessibilityIssue[] = [];
  const { typography, color, interaction } = settings;

  if (typography.fontSize < 110) {
    issues.push({
      severity: 'critical',
      wcagCriterion: '1.4.4',
      message: `Font size ${typography.fontSize}% is below recommended minimum for low vision`,
      recommendation: 'Set font size to at least 120–130% for low-vision users',
    });
  }

  if (color.contrast === 100 && !color.customBackground) {
    issues.push({
      severity: 'major',
      wcagCriterion: '1.4.3',
      message: 'No contrast enhancement active',
      recommendation: 'Enable dark mode or set custom high-contrast colors',
    });
```

```
    }

    if (color.customBackground && color.customText) {
      const bg = hexToRgb(color.customBackground);
      const fg = hexToRgb(color.customText);
      if (bg && fg) {
        const bgL = relativeLuminance(...bg);
        const fgL = relativeLuminance(...fg);
        const ratio = contrastRatio(bgL, fgL);
        if (ratio < LowVisionAdapter.WCAG_AA_CONTRAST) {
          issues.push({
            severity: 'critical',
            wcagCriterion: '1.4.3',
            message: `Custom color contrast ratio ${ratio.toFixed(2)}:1 is below WCAG
AA (4.5:1)`,
            recommendation: `Increase contrast between background and text colors to
at least 4.5:1`,
          });
        }
      }
    }

    if (!interaction.enlargeCursor) {
      issues.push({
        severity: 'minor',
        wcagCriterion: '1.4.4',
        message: 'Standard cursor size may be difficult to track',
        recommendation: 'Enable enlarged cursor for better visibility',
      });
```

```
  }

  if (!interaction.enhanceFocus && !interaction.focusHighlight) {
    issues.push({
      severity: 'major',
      wcagCriterion: '2.4.7',
      message: 'Focus indicator is not enhanced',
      recommendation: 'Enable enhanced focus indicators for keyboard navigation',
    });
  }

  if (typography.lineHeight < 1.5) {
    issues.push({
      severity: 'major',
      wcagCriterion: '1.4.8',
      message: 'Line height below 1.5 makes text difficult to track',
      recommendation: 'Set line height to 1.7 or higher',
    });
  }

  const total = 6;
  const passed = total - issues.length;
  const score = Math.round((passed / total) * 100);

  return {
    isOptimal: issues.length === 0,
    score,
    issues,
    suggestions: [
```

```
      `Current custom colors ${this.getContrastSuggestion(color)}`,
      'Reading guide helps track the current line across wide screens',
      'Highlighted links improve orientation on complex pages',
      'Interface scale of 120–130% enlarges all UI elements proportionally',
    ].filter(Boolean) as string[],
  };
}

getWCAGCriteria(): WCAGCriterion[] {
  return [
    {
      id: '1.4.3',
      level: 'AA',
      name: 'Contrast (Minimum)',
      description: 'Text has a contrast ratio of at least 4.5:1 (3:1 for large text)',
    },
    {
      id: '1.4.4',
      level: 'AA',
      name: 'Resize Text',
      description: 'Text can be resized without assistive technology up to 200%',
    },
    {
      id: '1.4.6',
      level: 'AAA',
      name: 'Contrast (Enhanced)',
      description: 'Text has a contrast ratio of at least 7:1',
    },
    {
```

```
      id: '1.4.8',
      level: 'AAA',
      name: 'Visual Presentation',
      description: 'Foreground/background colours can be selected by user',
    },
    {
      id: '1.4.10',
      level: 'AA',
      name: 'Reflow',
      description: 'Content can be presented without loss of information at 400% zoom',
    },
    {
      id: '2.4.7',
      level: 'AA',
      name: 'Focus Visible',
      description: 'Any keyboard operable interface has visible keyboard focus indicator',
    },
  ];
 }

 meetsMinimumStandards(settings: AccessibilitySettings): boolean {
  const { typography, color } = settings;
  const hasContrast =
   color.darkMode ||
   (color.customBackground !== null && color.customText !== null);
  return typography.fontSize >= 110 && hasContrast;
 }
```

```
  private getContrastSuggestion(color: AccessibilitySettings['color']): string {
    if (!color.customBackground || !color.customText) return '';
    const bg = hexToRgb(color.customBackground);
    const fg = hexToRgb(color.customText);
    if (!bg || !fg) return '';
    const ratio = contrastRatio(relativeLuminance(...bg), relativeLuminance(...fg));
    if (ratio >= LowVisionAdapter.WCAG_AAA_CONTRAST) return `achieve AAA
contrast (${ratio.toFixed(1)}:1)`;
    if (ratio >= LowVisionAdapter.WCAG_AA_CONTRAST) return `meet AA
contrast (${ratio.toFixed(1)}:1)`;
    return `below AA contrast (${ratio.toFixed(1)}:1) – increase contrast`;
  }
}

export const lowVisionAdapter = new LowVisionAdapter();
```

ДОДАТОК В

Інтеграція і обробка помилок (основна частина)

```
import { ToggleButton } from './ToggleButton';
import { AccessibilityPanel } from '../Panel/AccessibilityPanel';
import { useGlobalKeyboard } from '../../hooks/useKeyboard';
import { useAutoRead } from '../../hooks/useAutoRead';
import { useSelectionRead } from '../../hooks/useSelectionRead';
import { useThemeContext } from '../../providers/ThemeProvider';
import { useAccessibilityStore } from '../../state/store';
import { useVoiceContext } from '../../providers/VoiceProvider';
import { selectIsInitialized } from '../../state/selectors';
import { useEffect, useRef } from 'react';
import baseCss from '../../styles/base.css?inline';
import panelCss from '../../styles/panel.css?inline';
import headerCss from '../../styles/header.css?inline';
import tabsCss from '../../styles/tabs.css?inline';
import sectionCss from '../../styles/settingsSection.css?inline';
import toggleBtnCss from '../../styles/toggleButton.css?inline';
import voiceCss from '../../styles/voice.css?inline';
import controlsCss from '../../styles/controls.css?inline';
import modeCardCss from '../../styles/modeCard.css?inline';

const WIDGET_CSS = [
  baseCss, panelCss, headerCss, tabsCss, sectionCss,
  toggleBtnCss, voiceCss, controlsCss, modeCardCss,
].join('\n');

export function AccessibilityWidget() {
  const isInitialized = useAccessibilityStore(selectIsInitialized);
  const { theme } = useThemeContext();
  const autoRead = useAccessibilityStore((s) => s.settings.voice.autoRead);
  const { speak, stopSpeaking } = useVoiceContext();
  const widgetRef = useRef<HTMLDivElement>(null);
  useGlobalKeyboard();

  useAutoRead();
  useSelectionRead();

  useEffect(() => {
    if (!autoRead || !widgetRef.current) {
      return;
    }
```

```
    const handleFocus = (event: FocusEvent) => {
      const target = event.target as Element;

      if (!target || target === document.body || target === document.documentElement)
{
        return;
      }
      let text = target.textContent?.trim() || '';

      if (target instanceof HTMLInputElement || target instanceof
HTMLTextAreaElement) {
        text = target.placeholder || target.value || target.getAttribute('aria-label') || '';
      }

      if (text.length === 0) {
        return;
      }

      stopSpeaking();
      speak(text);
    };

    widgetRef.current.addEventListener('focus', handleFocus, true);

    return () => {
      widgetRef.current?.removeEventListener('focus', handleFocus, true);
    };
  }, [autoRead, speak, stopSpeaking]);

  if (!isInitialized) return null;

  const cssVars = Object.entries(theme)
    .filter(([, v]) => typeof v === 'string')
    .map(([k, v]) => `--aa-panel-${k.replace(/([A-Z])/g, '-$1').toLowerCase()}:
${v};`)
    .join('\n');

  return (
    <div ref={widgetRef}>
      {/* Inject theme CSS variables into shadow DOM */}
      <style>{`:host { ${cssVars} display: contents; }`}</style>
      {/* Inject all widget CSS classes */}
      <style>{WIDGET_CSS}</style>
```

```
    <ToggleButton />
    <AccessibilityPanel />
  </div> );}
```

```
import { Component, type ErrorInfo, type ReactNode } from 'react';

interface Props {
  children: ReactNode;
  fallback?: ReactNode;
}

interface State {
  hasError: boolean;
  error: Error | null;
}

export class ErrorBoundary extends Component<Props, State> {
  state: State = { hasError: false, error: null };

  static getDerivedStateFromError(error: Error): State {
    return { hasError: true, error };
  }

  componentDidCatch(error: Error, info: ErrorInfo): void {
    console.error('[AdaptiveAccessibility] Render error:', error,
info.componentStack);
  }

  private handleRetry = (): void => {
    this.setState({ hasError: false, error: null });
  };

  render(): ReactNode {
    if (this.state.hasError) {
      if (this.props.fallback) return this.props.fallback;

      return (
        <div
          role="alertdialog"
          aria-modal="false"
          aria-label="Accessibility module error"
          style={{
            position: 'fixed',
```

```
          bottom: '24px',
          right: '24px',
          padding: '14px 18px',
          background: '#fef2f2',
          border: '1.5px solid #fca5a5',
          borderRadius: '10px',
          fontSize: '13px',
          color: '#991b1b',
          maxWidth: '340px',
          zIndex: 999998,
          fontFamily: '-apple-system, BlinkMacSystemFont, "Segoe UI", sans-serif',
          lineHeight: 1.5,
          boxShadow: '0 4px 16px rgba(0,0,0,0.12)',
        }}
      >
        <strong style={{ display: 'block', marginBottom: '6px' }}>
          Accessibility module error
        </strong>
        <span style={{ fontSize: '12px', color: '#7f1d1d' }}>
          An unexpected error occurred. The panel is temporarily unavailable.
        </span>
        <button
          onClick={this.handleRetry}
          style={{
            display: 'block',
            marginTop: '10px',
            padding: '5px 12px',
            background: '#fee2e2',
            border: '1px solid #fca5a5',
            borderRadius: '6px',
            fontSize: '12px',
            color: '#991b1b',
            cursor: 'pointer',
            outline: 'none',
          }}
        >
          Retry
        </button>
      </div>
    );
  }
  return this.props.children;
}}
```

```
import { Logger } from '../utils/logger';

export class ShadowDOMHost {
  private host: HTMLElement | null = null;
  private shadowRoot: ShadowRoot | null = null;
  private readonly logger: Logger;

  constructor(debug = false) {
    this.logger = new Logger('ShadowDOMHost', debug);
  }

  mount(mountTarget: string | HTMLElement, zIndex: number): ShadowRoot {
    const container = this.resolveTarget(mountTarget);

    // Create host element
    this.host = document.createElement('div');
    this.host.id = 'adaptive-accessibility-host';
    this.host.setAttribute('data-aa', 'true');
    this.host.setAttribute('aria-hidden', 'false');
    // Prevent host-page styles from inheriting into the shadow DOM via CSS containment
    this.host.style.cssText = `
      all: initial;
      display: block;
      position: relative;
      z-index: ${zIndex};
    `;

    container.appendChild(this.host);

    // Attach shadow root – 'open' for dev inspection; in production consider 'closed'
    this.shadowRoot = this.host.attachShadow({ mode: 'open', delegatesFocus: true });

    this.logger.log('Shadow DOM host mounted.');
    return this.shadowRoot;
  }

  getShadowRoot(): ShadowRoot | null {
    return this.shadowRoot;
  }

  getHost(): HTMLElement | null {
    return this.host;
```

```
  }

  unmount(): void {
    this.host?.remove();
    this.host = null;
    this.shadowRoot = null;
    this.logger.log('Shadow DOM host unmounted.');
  }

  private resolveTarget(target: string | HTMLElement): HTMLElement {
    if (target instanceof HTMLElement) return target;

    const el = document.querySelector<HTMLElement>(target);
    if (!el) {
      // Fallback: append directly to body
      this.logger.warn(`Mount target "${target}" not found. Falling back to body.`);
      return document.body;
    }
    return el;
  }
}
```

```
import { getOrCreateStyleElement } from '../utils/css';

export class StyleInjector {
  private readonly nonce?: string;

  constructor(nonce?: string) {
    this.nonce = nonce;
  }

  injectHostPageBase(): void {
    const el = getOrCreateStyleElement('aa-host-base', this.nonce);
    el.textContent = `
      :root {
        --aa-font-size: 1em;
        --aa-line-height: 1.5;
        --aa-letter-spacing: 0em;
        --aa-word-spacing: 0em;
        --aa-filter: none;
        --aa-reading-width: none;
        --aa-focus-color: #ff6b00;
      }
      @media (prefers-reduced-motion: no-preference) {
```

```
      html { scroll-behavior: smooth; }
    }
  `;
}

injectShadowHostStyles(shadowRoot: ShadowRoot): void {
  const style = document.createElement('style');
  if (this.nonce) style.setAttribute('nonce', this.nonce);
  style.textContent = `
    *, *::before, *::after {
      box-sizing: border-box;
      margin: 0;
      padding: 0;
    }
    :host {
      all: initial;
      display: contents;
    }
    * {
      -webkit-font-smoothing: antialiased;
      -moz-osx-font-smoothing: grayscale;
    }
    ::-webkit-scrollbar { width: 5px; }
    ::-webkit-scrollbar-track { background: transparent; }
    ::-webkit-scrollbar-thumb {
      background: var(--aa-panel-border, #e5e7eb);
      border-radius: 3px;
    }
  `;
  shadowRoot.appendChild(style);
}

injectOpenDyslexicFont(customUrl?: string): void {
  if (document.getElementById('aa-dyslexic-font')) return;

  const baseUrl =
    customUrl ??
    'https://cdn.jsdelivr.net/npm/open-dyslexic@1.0.3/ttf/';

  const el = document.createElement('style');
  el.id = 'aa-dyslexic-font';
  el.setAttribute('data-aa', 'true');
  if (this.nonce) el.setAttribute('nonce', this.nonce);
```

```
    el.textContent = `
      @font-face {
        font-family: 'OpenDyslexic';
        font-display: swap;
        src: url('${baseUrl}OpenDyslexic-Regular.ttf') format('truetype');
        font-weight: 400;
        font-style: normal;
      }
      @font-face {
        font-family: 'OpenDyslexic';
        font-display: swap;
        src: url('${baseUrl}OpenDyslexic-Bold.ttf') format('truetype');
        font-weight: 700;
        font-style: normal;
      }
      @font-face {
        font-family: 'OpenDyslexic';
        font-display: swap;
        src: url('${baseUrl}OpenDyslexic-Italic.ttf') format('truetype');
        font-weight: 400;
        font-style: italic;
      }
    `;
    document.head.appendChild(el);
  }

  removeHostPageBase(): void {
    document.getElementById('aa-host-base')?.remove();
  }

  removeDyslexicFont(): void {
    document.getElementById('aa-dyslexic-font')?.remove();
  }
}
```

ДОДАТОК Г

Приклад рушія змін

```
import { BaseEngine } from './BaseEngine';
import type { AccessibilitySettings } from '../types/accessibility';
import { StyleInjector } from '../integration/StyleInjector';

const FONT_FAMILIES: Record<string, string> = {
  default: 'inherit',
  dyslexic:
    '"OpenDyslexic", "Comic Sans MS", "Trebuchet MS", sans-serif',
  readable:
    '"Arial", "Helvetica Neue", Helvetica, sans-serif',
  serif: '"Georgia", "Times New Roman", serif',
  monospace: '"Courier New", "Courier", monospace',
  system:
    'system-ui, -apple-system, BlinkMacSystemFont, "Segoe UI", sans-serif',
};

export class TypographyEngine extends BaseEngine {
  readonly id = 'typography';

  private readonly styleInjector: StyleInjector;

  constructor(
    scope: string,
    cspNonce?: string,
    debug = false
  ) {
    super(scope, cspNonce, debug);

    this.styleInjector = new StyleInjector(cspNonce);
  }

  apply(settings: AccessibilitySettings): void {
    const { typography, layout } = settings;

    const {
      fontSize,
      fontFamily,
      lineHeight,
      letterSpacing,
      wordSpacing,
      textScale,
```

```
} = typography;

if (fontFamily === 'dyslexic') {
  this.styleInjector.injectOpenDyslexicFont();
}

const fontFamilyValue =
  FONT_FAMILIES[fontFamily] ??
  FONT_FAMILIES.default;

const textAlignValue =
  layout.textAlign === 'default'
    ? ''
    : layout.textAlign;

// Final proportional scale
const scaledFontSize =
  (fontSize / 100) * textScale;

const rules: string[] = [];

rules.push(`
  ${this.scope},
  ${this.scope} * {
    -webkit-text-size-adjust: none !important;
    text-size-adjust: none !important;
  }
`);

if (
  fontSize !== 100 ||
  textScale !== 1
) {
  rules.push(`
    ${this.scope} {
      font-size: ${
        scaledFontSize * 100
      }% !important;
    }
  `);
}

if (fontFamily !== 'default') {
  rules.push(`
```

```
    ${this.scope},
    ${this.scope} * {
      font-family: ${fontFamilyValue} !important;
    }
  `);
}

if (lineHeight !== 1.5) {
  rules.push(`
    ${this.scope},
    ${this.scope} * {
      line-height: ${lineHeight} !important;
    }
  `);
}

if (letterSpacing !== 0) {
  rules.push(`
    ${this.scope},
    ${this.scope} * {
      letter-spacing: ${letterSpacing}em !important;
    }
  `);
}

if (wordSpacing !== 0) {
  rules.push(`
    ${this.scope},
    ${this.scope} * {
      word-spacing: ${wordSpacing}em !important;
    }
  `);
}

if (textAlignValue) {
  rules.push(`
    ${this.scope} p,
    ${this.scope} div,
    ${this.scope} li,
    ${this.scope} span,
    ${this.scope} article,
    ${this.scope} section {
      text-align: ${textAlignValue} !important;
    }
```

```
    `);
  }

  this.injectCSS(
    rules.join('\n')
  ); }}
```

ДОДАТОК Ґ

Голосовий модуль та сховище (основна частина)

```
import { Logger } from '../utils/logger';

const SpeechRecognitionAPI =
  window.SpeechRecognition ?? window.webkitSpeechRecognition;

export type RecognitionCallback = (
  transcript: string,
  isFinal: boolean
) => void;

export class SpeechRecognitionService {
  static readonly isSupported = typeof SpeechRecognitionAPI !== 'undefined';

  private recognition: SpeechRecognition | null = null;
  private isListening = false;
  private shouldRestart = false;
  private onTranscript: RecognitionCallback | null = null;
  private onStateChange: ((listening: boolean) => void) | null = null;
  private readonly logger: Logger;
  private language = 'en-US';

  constructor(debug = false) {
    this.logger = new Logger('SpeechRecognition', debug);
  }

  start(
    language: string,
    onTranscript: RecognitionCallback,
    onStateChange?: (listening: boolean) => void
  ): void {
    if (!SpeechRecognitionService.isSupported) {
      this.logger.warn('SpeechRecognition not supported in this browser.');
      return;
    }
    if (this.isListening) return;

    this.language = language;
    this.onTranscript = onTranscript;
    this.onStateChange = onStateChange ?? null;
    this.shouldRestart = true;
    this.startRecognition();
```

```
}

stop(): void {
  this.shouldRestart = false;
  this.recognition?.stop();
  this.isListening = false;
  this.onStateChange?.(false);
}

isActive(): boolean {
  return this.isListening;
}

private startRecognition(): void {
  this.recognition = new SpeechRecognitionAPI!();
  this.recognition.lang = this.language;
  this.recognition.continuous = false;     // single utterance – safer / simpler
  this.recognition.interimResults = true;
  this.recognition.maxAlternatives = 1;

  this.recognition.onstart = () => {
    this.isListening = true;
    this.onStateChange?.(true);
    this.logger.log('Listening started.');
  };

  this.recognition.onresult = (event: SpeechRecognitionEvent) => {
    let interimTranscript = '';
    let finalTranscript = '';

    for (let i = event.resultIndex; i < event.results.length; i++) {
      const result = event.results[i];
      if (result.isFinal) {
        finalTranscript += result[0].transcript;
      } else {
        interimTranscript += result[0].transcript;
      }
    }

    if (finalTranscript) {
      this.onTranscript?.(finalTranscript.trim(), true);
    } else if (interimTranscript) {
      this.onTranscript?.(interimTranscript.trim(), false);
    }
```

```
    };

    this.recognition.onend = () => {
      this.isListening = false;
      this.logger.log('Listening ended.');
      if (this.shouldRestart) {
        // Brief pause before restarting to avoid browser throttling
        setTimeout(() => {
          if (this.shouldRestart) this.startRecognition();
        }, 300);
      } else {
        this.onStateChange?.(false);
      }
    };

    this.recognition.onerror = (event: SpeechRecognitionErrorEvent) => {
      if (event.error === 'no-speech' || event.error === 'aborted') return;
      this.logger.error('SpeechRecognition error:', event.error);
      if (event.error === 'not-allowed') {
        this.shouldRestart = false;
        this.onStateChange?.(false);
      }
    };

    try {
      this.recognition.start();
    } catch (err) {
      this.logger.error('Failed to start SpeechRecognition:', err);
      this.shouldRestart = false;
    }}}
```

```
import { SpeechSynthesisService } from './SpeechSynthesisService';
import { SpeechRecognitionService } from './SpeechRecognitionService';
import { CommandProcessor } from './CommandProcessor';
import type { VoiceCommandContext } from '../types/voice';
import type { VoiceSettings } from '../types/accessibility';
import { useAccessibilityStore } from '../state/store';
import { presetManager } from '../presets/PresetManager';

type StateCallback = (state: {
  isSpeaking: boolean;
  isListening: boolean;
  transcript: string;
}) => void;
```

```
export class VoiceEngine {
  static readonly isTTSSupported = SpeechSynthesisService.isSupported;
  static readonly isSTTSupported = SpeechRecognitionService.isSupported;

  private readonly tts: SpeechSynthesisService;
  private readonly stt: SpeechRecognitionService;
  private readonly processor: CommandProcessor;
  private stateCallback: StateCallback | null = null;

  constructor(debug = false) {
    this.tts = new SpeechSynthesisService(debug);
    this.stt = new SpeechRecognitionService(debug);
    this.processor = new CommandProcessor(debug);
  }

  speak(
    text: string,
    voiceSettings: VoiceSettings,
    callbacks?: { onStart?: () => void; onEnd?: () => void }
  ): void {
    this.tts.speak(text, {
      rate: voiceSettings.rate,
      pitch: voiceSettings.pitch,
      volume: voiceSettings.volume,
      language: voiceSettings.language,
      voiceIndex: voiceSettings.voiceIndex,
      onStart: () => {
        this.notifyState({ isSpeaking: true, isListening: this.stt.isActive(), transcript: "
});
        callbacks?.onStart?.();
      },
      onEnd: () => {
        this.notifyState({ isSpeaking: false, isListening: this.stt.isActive(), transcript: "
});
        callbacks?.onEnd?.();
      },
    });
  }

  stopSpeaking(): void {
    this.tts.stop();
    this.notifyState({ isSpeaking: false, isListening: this.stt.isActive(), transcript: " });
  }
```

```
  readPage(voiceSettings: VoiceSettings): void {
    this.tts.readWholePage({
      rate: voiceSettings.rate,
      pitch: voiceSettings.pitch,
      volume: voiceSettings.volume,
      language: voiceSettings.language,
      voiceIndex: voiceSettings.voiceIndex,
      onStart: () =>
        this.notifyState({ isSpeaking: true, isListening: this.stt.isActive(), transcript: "
}),
      onEnd: () =>
        this.notifyState({ isSpeaking: false, isListening: this.stt.isActive(), transcript: "
}),
    });
  }

  startListening(language: string): void {
    if (!SpeechRecognitionService.isSupported) return;

    this.stt.start(
      language,
      (transcript, isFinal) => {
        this.notifyState({ isSpeaking: this.tts.isSpeaking, isListening: true, transcript });
        if (isFinal) {
          this.processor.process(transcript, this.buildContext());
        }
      },
      (isListening) => {
        this.notifyState({ isSpeaking: this.tts.isSpeaking, isListening, transcript: " });
        useAccessibilityStore.getState().setVoiceActive(isListening);
      }
    );
  }

  stopListening(): void {
    this.stt.stop();
  }

  getAvailableVoices(): SpeechSynthesisVoice[] {
    return this.tts.getVoices();
  }

  onStateChange(cb: StateCallback): void {
```

```
    this.stateCallback = cb;
  }

  destroy(): void {
    this.tts.stop();
    this.stt.stop();
    this.stateCallback = null;
  }

  private buildContext(): VoiceCommandContext {
    const store = useAccessibilityStore.getState();
    const voiceSettings = store.settings.voice;

    return {
      speak: (text) => this.speak(text, voiceSettings),
      applyMode: (modeId) => {
        const id = modeId as Parameters<typeof presetManager.applyPreset>[0];
        if (presetManager.has(id)) {
          const newSettings = presetManager.applyPreset(id);
          store.setSettings({ ...newSettings, activeMode: id });
        }
      },
      updateSetting: (path, value) => {
        // path format: "typography.fontSize" | "color.darkMode" etc.
        const [group, key] = path.split('.');
        if (!group || !key) return;
        const current = store.settings as unknown as Record<string, Record<string,
unknown>>;
        if (group in current && typeof current[group] === 'object') {
          const groupObj = { ...current[group], [key]: value };
          const updater = `update${group.charAt(0).toUpperCase() + group.slice(1)}`
as keyof typeof store;
          if (typeof store[updater] === 'function') {
            (store[updater] as (p: Record<string, unknown>) => void)(groupObj);
          }
        }
      },
      resetSettings: () => store.resetSettings(),
      openPanel: () => store.openPanel(),
      closePanel: () => store.closePanel(),
      readPage: () => this.readPage(voiceSettings),
    };
  }
```

```
  private notifyState(state: { isSpeaking: boolean; isListening: boolean; transcript: string }): void {
    this.stateCallback?.(state);
  }
}

export const voiceEngine = new VoiceEngine();
```

```
import { create } from 'zustand';
import { subscribeWithSelector } from 'zustand/middleware';
import type {
  AccessibilitySettings,
  AccessibilityModeId,
  PanelTab,
  TypographySettings,
  ColorSettings,
  LayoutSettings,
  MotionSettings,
  InteractionSettings,
  VoiceSettings,
} from '../types/accessibility';
import { DEFAULT_SETTINGS } from '../config/defaultSettings';

export interface AccessibilityStore {
  // State
  settings: AccessibilitySettings;
  isInitialized: boolean;
  isPanelVisible: boolean;
  isVoiceActive: boolean;
  isSpeaking: boolean;
  voiceTranscript: string;
  error: string | null;

  // Settings updaters
  updateTypography: (patch: Partial<TypographySettings>) => void;
  updateColor: (patch: Partial<ColorSettings>) => void;
  updateLayout: (patch: Partial<LayoutSettings>) => void;
  updateMotion: (patch: Partial<MotionSettings>) => void;
  updateInteraction: (patch: Partial<InteractionSettings>) => void;
  updateVoice: (patch: Partial<VoiceSettings>) => void;
  setSettings: (settings: AccessibilitySettings) => void;
  resetSettings: () => void;

  // Mode management
```

```
  activateMode: (modeId: AccessibilityModeId) => void;
  deactivateMode: () => void;

  // Panel control
  openPanel: () => void;
  closePanel: () => void;
  togglePanel: () => void;
  setActiveTab: (tab: PanelTab) => void;

  // Voice state
  setVoiceActive: (active: boolean) => void;
  setSpeaking: (speaking: boolean) => void;
  setVoiceTranscript: (transcript: string) => void;

  // Lifecycle
  setInitialized: (initialized: boolean) => void;
  setError: (error: string | null) => void;
}

export const useAccessibilityStore = create<AccessibilityStore>()(
  subscribeWithSelector((set) => ({
    settings: { ...DEFAULT_SETTINGS },
    isInitialized: false,
    isPanelVisible: false,
    isVoiceActive: false,
    isSpeaking: false,
    voiceTranscript: '',
    error: null,

    updateTypography: (patch) =>
      set((s) => ({
        settings: { ...s.settings, typography: { ...s.settings.typography, ...patch } },
      })),

    updateColor: (patch) =>
      set((s) => ({
        settings: { ...s.settings, color: { ...s.settings.color, ...patch } },
      })),

    updateLayout: (patch) =>
      set((s) => ({
        settings: { ...s.settings, layout: { ...s.settings.layout, ...patch } },
      })),
```

```
    updateMotion: (patch) =>
      set((s) => ({
        settings: { ...s.settings, motion: { ...s.settings.motion, ...patch } },
      })),

    updateInteraction: (patch) =>
      set((s) => ({
        settings: { ...s.settings, interaction: { ...s.settings.interaction, ...patch } },
      })),

    updateVoice: (patch) =>
      set((s) => ({
        settings: { ...s.settings, voice: { ...s.settings.voice, ...patch } },
      })),

    setSettings: (settings) => set({ settings }),

    resetSettings: () =>
      set({ settings: { ...DEFAULT_SETTINGS }, error: null }),

    activateMode: (modeId) =>
      set((s) => ({
        settings: { ...s.settings, activeMode: modeId },
      })),

    deactivateMode: () =>
      set((s) => ({
        settings: { ...s.settings, activeMode: null },
      })),
    openPanel: () => set({ isPanelVisible: true }),
    closePanel: () => set({ isPanelVisible: false }),
    togglePanel: () => set((s) => ({ isPanelVisible: !s.isPanelVisible })),

    setActiveTab: (tab) =>
      set((s) => ({ settings: { ...s.settings, activeTab: tab } })),

    setVoiceActive: (active) => set({ isVoiceActive: active }),
    setSpeaking: (speaking) => set({ isSpeaking: speaking }),
    setVoiceTranscript: (transcript) => set({ voiceTranscript: transcript }),
    setInitialized: (initialized) => set({ isInitialized: initialized }),
    setError: (error) => set({ error }),
  }))
);
```

```
import type { IStorageAdapter } from './IStorageAdapter';

export class LocalStorageAdapter implements IStorageAdapter {
  private readonly prefix: string;
  private readonly available: boolean;
  private memory = new Map<string, string>();

  constructor(prefix: string) {
    this.prefix = prefix;
    this.available = this.testAvailability();
  }

  async get<T>(key: string): Promise<T | null> {
    const raw = this.available
      ? localStorage.getItem(this.prefixed(key))
      : (this.memory.get(this.prefixed(key)) ?? null);

    if (raw === null) return null;

    try {
      return JSON.parse(raw) as T;
    } catch {
      return null;
    }
  }

  async set<T>(key: string, value: T): Promise<void> {
    const serialized = JSON.stringify(value);
    if (this.available) {
      try {
        localStorage.setItem(this.prefixed(key), serialized);
      } catch {
        // Storage quota exceeded – fall through to memory
        this.memory.set(this.prefixed(key), serialized);
      }
    } else {
      this.memory.set(this.prefixed(key), serialized);
    }
  }

  async remove(key: string): Promise<void> {
    if (this.available) localStorage.removeItem(this.prefixed(key));
    this.memory.delete(this.prefixed(key));
  }
```

```
  async clear(prefix?: string): Promise<void> {
    const target = prefix ? `${this.prefix}:${prefix}` : this.prefix;
    if (this.available) {
      const toRemove: string[] = [];
      for (let i = 0; i < localStorage.length; i++) {
        const k = localStorage.key(i);
        if (k && k.startsWith(target)) toRemove.push(k);
      }
      toRemove.forEach(k => localStorage.removeItem(k));
    }
    this.memory.forEach((_, k) => { if (k.startsWith(target)) this.memory.delete(k);
});
  }

  async has(key: string): Promise<boolean> {
    if (this.available) return localStorage.getItem(this.prefixed(key)) !== null;
    return this.memory.has(this.prefixed(key));
  }

  private prefixed(key: string): string {
    return `${this.prefix}:${key}`;
  }

  private testAvailability(): boolean {
    try {
      const probe = '__aa_probe__';
      localStorage.setItem(probe, '1');
      localStorage.removeItem(probe);
      return true;
    } catch {
      return false;
    }
  }
}
```

```
import type { IStorageAdapter } from './IStorageAdapter';
import { STORAGE_DB_NAME, STORAGE_DB_VERSION,
STORAGE_STORE_NAME } from '../constants/storage';

export class IndexedDBAdapter implements IStorageAdapter {
  private readonly prefix: string;
  private db: IDBDatabase | null = null;
  private initPromise: Promise<void> | null = null;
```

```
  constructor(prefix: string) {
    this.prefix = prefix;
  }

  private async ensureDB(): Promise<void> {
    if (this.db) return;
    if (this.initPromise) return this.initPromise;

    this.initPromise = new Promise((resolve, reject) => {
      const req = indexedDB.open(STORAGE_DB_NAME,
STORAGE_DB_VERSION);
      req.onupgradeneeded = () => {
        req.result.createObjectStore(STORAGE_STORE_NAME);
      };
      req.onsuccess = () => { this.db = req.result; resolve(); };
      req.onerror = () => reject(req.error);
    });

    return this.initPromise;
  }

  async get<T>(key: string): Promise<T | null> {
    await this.ensureDB();
    return new Promise((resolve) => {
      const tx = this.db!.transaction(STORAGE_STORE_NAME, 'readonly');
      const req = tx.objectStore(STORAGE_STORE_NAME).get(this.prefixed(key));
      req.onsuccess = () => resolve((req.result as T) ?? null);
      req.onerror = () => resolve(null);
    });
  }

  async set<T>(key: string, value: T): Promise<void> {
    await this.ensureDB();
    return new Promise((resolve, reject) => {
      const tx = this.db!.transaction(STORAGE_STORE_NAME, 'readwrite');
      const req = tx.objectStore(STORAGE_STORE_NAME).put(value,
this.prefixed(key));
      req.onsuccess = () => resolve();
      req.onerror = () => reject(req.error);
    });
  }

  async remove(key: string): Promise<void> {
```

```
    await this.ensureDB();
    return new Promise((resolve, reject) => {
      const tx = this.db!.transaction(STORAGE_STORE_NAME, 'readwrite');
      const req = tx.objectStore(STORAGE_STORE_NAME).delete(this.prefixed(key));
      req.onsuccess = () => resolve();
      req.onerror = () => reject(req.error);
    });
  }

  async clear(prefix?: string): Promise<void> {
    await this.ensureDB();
    const target = prefix ? `${this.prefix}:${prefix}` : this.prefix;
    return new Promise((resolve, reject) => {
      const tx = this.db!.transaction(STORAGE_STORE_NAME, 'readwrite');
      const store = tx.objectStore(STORAGE_STORE_NAME);
      const req = store.getAllKeys();
      req.onsuccess = () => {
        const keys = req.result as IDBValidKey[];
        keys
          .filter(k => String(k).startsWith(target))
          .forEach(k => store.delete(k));
        resolve();
      };
      req.onerror = () => reject(req.error);
    });
  }

  async has(key: string): Promise<boolean> {
    const val = await this.get(key);
    return val !== null;
  }

  private prefixed(key: string): string {
    return `${this.prefix}:${key}`;
  }
}
```

ДЕКЛАРАЦІЯ
про дотримання академічної доброчесності

Я,_____
_____
_____
_____
Повністю вказується ПІБ та статус (посада для працівників, освітня (освітньо-наукова) програма – для здобувачів вищої освіти)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____
(дата) (підпис)