Міністерство освіти і науки України

Національний університет водного господарства та природокористування

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії

Кафедра комп’ютерних технологій та економічної кібернетики

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНКЛЮЗИВНОГО ВЕБСАЙТУ З ІНФОРМАТИКИ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ДОСТУПНОСТІ ТА ПЕРСОНАЛІЗАЦІЇ НАВЧАЛЬНОГО КОНТЕНТУ

Виконала:
Здобувачка вищої освіти групи ІСТ-41
спеціальність 126
«Інформаційні системи та технології»
Яремчук Катерина Тарасівна

Науковий керівник:
Олексій Володимирович ПАРФЕНЮК

Рівне – 2026

Національний університет водного господарства та природокористування
ННІ кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Освітньо-кваліфікаційний рівень – **бакалавр**

за освітньо-професійною програмою **«Інформаційні системи і технології»**

спеціальність **126 «Інформаційні системи та технології»**

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ д. е. н., проф. П. М. Грицюк

«_____» _____ 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачу _____Яремчук Катерині Тарасівні_____
(прізвище, ім'я, по-батькові)

1. Тема роботи __Проектування та розробка інклюзивного вебсайту з інформатики з використанням технологій доступності та персоналізації навчального контенту_____

керівник роботи: __Парфенюк Олексій Володимирович_____
(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)

затверджена наказом по університету від "26 "квітня 2026 р. С № _____

2. Термін здачі здобувачем закінченої роботи "_08_"__червня__ 2026 р.

3. Вихідні дані до роботи _Вихідними даними роботи є розробка інклюзивної освітньої вебплатформи „Inclusify" на базі технологій React.js, Node.js та PostgreSQL, яка призначена для адаптації навчального процесу під потреби учнів з порушеннями зору, слуху та моторики за допомогою спеціальних інструментів доступності.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) __вступ; аналіз предметної області; вибір та обґрунтування засобів розробки програмного продукту; розробка і опис створеної інформаційної системи._____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Структурна схема архітектури вебплатформи «Inclusify».

2.Схема моделі даних (ER-діаграма) СУБД PostgreSQL.

3.Блок-схема алгоритму JWT-автентифікації та захисту даних.

4.Функціональна схема модуля налаштувань вебдоступності.

5.Графічний інтерфейс користувача (екрани платформи в інклюзивних режимах).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Усі розділи	Парфенюк О.В.		

7. Дата видачі завдання "_16_" листопада 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Постановка мети та визначення актуальності теми	16.11.25-30.11.25	
2.	Дослідження міжнародних стандартів веб-доступнрості	1.12.25 – 21.12.25	
3.	Порівняльний аналіз існуючих освітніх веб-платформ за критеріями відповідності стандартам доступності	22.12.25 – 30.12.25	
4.	Проектування, розробка, реалізація та тестування інформаційної системи	02.01.26 – 9.04.26	
5.	Підготовка тексту кваліфікаційної роботи	10.04.26 – 23.05.26	
6.	Підготовка презентації роботи	24.05.26 – 03.06.26	
7.	Відгук керівника, рецензування роботи, перевірка на плагіат	04.06.26 – 07.06.26	

Здобувач _____***К.Т. Яремчук***

(підпис) (прізвище і ініціали)

Керівник кваліфікаційної роботи

_____***О.В. Парфенюк***

(підпис) (прізвище і ініціали)

ЗАЯВА

щодо самостійності виконання випускної кваліфікаційної роботи

Я, _____(ПІБ), студент(ка) _____курсу ___ групи ___ ННІ _____ заявляю: моя випускна кваліфікаційна робота на тему

«_____

_____

___» (назва роботи), яка надається в екзаменаційну комісію із захисту бакалаврської роботи _____

_____(назва спеціальності) для захисту, виконана самостійно і не містить плагіату. Всі запозичення з друкованих та електронних джерел, у тому числі із захищених раніше випускових кваліфікаційних робіт, кандидатських і докторських дисертацій мають відповідні посилання. Я не використовував(ла) шахрайські методи заміни символів чи інші способи модифікації тексту. Я ознайомлений(а) з чинним Порядком перевірки навчальних, кваліфікаційних, навчально-методичних та наукових робіт на наявність ознак академічного плагіату в НУВГП та Положенням про академічну доброчесність в НУВГП, за яким виявлення плагіату є підставою для відмови в допуску моєї роботи до захисту та застосування відповідних санкцій (академічної відповідальності).

Дата Підпис

Метадані

ДОКУМЕНТ

Заголовок
Пояснювальна записка Яремчук.docx

Автор	Науковий керівник / Експерт	ІД документу
Яремчук Катерина Тарасівна	**Яремчук Катерина Тарасівна**	**334306093**

ОРГАНІЗАЦІЯ

Назва організації	підрозділ
National University of Water and Environmental Engineering	**National University of Water and Environmental Engineering**

ЗВІТ

Дата звіту	Дата редагування
6/13/2026	**---**

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1

КЦ

10603	83508
Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		13

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз	Колір тексту

#	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

АКТ

перевірки випускної кваліфікаційної роботи на відсоток схожості та можливих модифікацій тексту

Відповідно до даних сервісу Unicheck файл
«_____»,
автор:_____ містить: _____ % схожості,
_____ замінених символів та можливих модифікації тексту
_____. (виявлено/не виявлено/стор.)

Дата Підпис

АНОТАЦІЯ

Яремчук К. Т. Проєктування та розробка інклюзивного вебсайту з інформатики з використанням технологій доступності та персоналізації навчального контенту – кваліфікаційна робота на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 126 Інформаційні системи та технології. – Національний університет водного господарства та природокористування. – Рівне, 2026.

У кваліфікаційній роботі досліджено проблему забезпечення цифрової доступності освітніх веб-ресурсів для учнів з особливими освітніми потребами. Проаналізовано чинну нормативно-правову базу України у сфері інклюзивної освіти, вимоги міжнародного стандарту WCAG 2.1 та специфікацію WAI-ARIA. Проведено порівняльний аналіз освітніх платформ Khan Academy, Coursera та «Всеукраїнської школи онлайн» за критеріями відповідності стандартам доступності.

Спроектовано архітектуру системи із застосуванням UML-моделювання та реляційну схему бази даних з восьми таблиць, приведену до третьої нормальної форми. Розроблено веб-сайт з інформатики на технологічному стеку React.js, Node.js та PostgreSQL з вбудованим модулем персоналізації доступності, що забезпечує висококонтрастний режим відображення, масштабування шрифту в діапазоні 100–300%, підтримку шрифту OpenDyslexic, регулювання міжрядкового інтервалу та синтез мовлення на базі ElevenLabs API. Реалізовано підтримку екранних зчитувачів через WAI-ARIA розмітку, skip link, focus trap та управління фокусом при навігації односторінкового застосунку.

Тестування системи підтвердило відповідність п'ятнадцяти критеріям стандарту WCAG 2.1 рівня AA.

Робота містить 13 таблиць та 24 рисунки.

Ключові слова: інклюзивна освіта, веб-доступність, WCAG 2.1, WAI-ARIA, React.js, профіль доступності, екранний зчитувач, синтез мовлення.

ЗМІСТ

ВСТУП

Забезпечення рівного доступу до освіти для осіб з особливими освітніми потребами є одним із пріоритетів державної освітньої політики України. Закон України «Про освіту» (2017) закріплює право кожної дитини на здобуття освіти в інклюзивному середовищі, визначаючи інклюзивне навчання як систему освітніх послуг, що забезпечує реалізацію права на освіту осіб з особливими потребами з урахуванням їхніх індивідуальних можливостей та потреб [1]. Водночас практична реалізація цього права в умовах закладів загальної середньої освіти залишається обмеженою через відсутність спеціалізованих цифрових інструментів, адаптованих до потреб різних категорій учнів.

Розвиток веб-технологій та поширення дистанційних форм навчання, що набули особливої актуальності в умовах воєнного стану в Україні, зумовили потребу в розробці освітніх веб-ресурсів, які б відповідали міжнародним стандартам доступності. Настанова з доступності веб-вмісту WCAG 2.1, розроблена Консорціумом Всесвітньої павутини (W3C), визначає технічні та змістові вимоги до веб-сторінок, виконання яких забезпечує їх придатність для використання особами з порушеннями зору, слуху, моторики та когнітивних функцій [2]. Попри те, що зазначені стандарти є загальновизнаними на міжнародному рівні, більшість наявних українських освітніх платформ не відповідає навіть базовому рівню відповідності WCAG 2.1 рівня AA, що підтверджується результатами незалежних аудитів доступності [3].

Приватний ліцей «Традиція», в межах якого виконувалася переддипломна практика, є прикладом закладу, що реалізує інклюзивне навчання, проте не має у своєму розпорядженні спеціалізованого веб-ресурсу з підтримкою функцій доступності. Учні з порушеннями зору, дислексією, порушеннями моторики та іншими нозологіями змушені користуватися загальними інформаційними ресурсами без можливості їх індивідуальної адаптації. Це безпосередньо знижує якість засвоєння навчального матеріалу з інформатики, де значна частина контенту є візуальною та інтерактивною за своєю природою.

Розробка спеціалізованого веб-сайту з підтримкою функцій доступності для учнів ліцею «Традиція» передбачає використання технологічного стеку React.js, Node.js та PostgreSQL, що дозволяє реалізувати динамічну зміну параметрів інтерфейсу без перезавантаження сторінок, збереження персональних профілів доступності та повноцінну підтримку допоміжних технологій, зокрема екранних зчитувачів [4]. Застосування специфікації WAI-ARIA дозволяє розширити семантику HTML-елементів таким чином, щоб забезпечити коректне озвучення інтерфейсу програмами-дикторами [5].

Питання проектування доступних веб-інтерфейсів розглядалися в працях таких дослідників, як H. Petrie та C. Power [6], які систематизували типові бар'єри доступності на освітніх платформах, а також у роботах, присвячених застосуванню принципів універсального дизайну в цифровому середовищі [7]. В українському контексті проблематика цифрової доступності в освіті досліджувалась у роботах, що стосуються впровадження інклюзивних підходів у навчальний процес закладів загальної середньої освіти [8]. Разом з тим комплексні розробки, що поєднують проектування веб-архітектури, відповідність стандарту WCAG 2.1 та збереження індивідуальних профілів доступності в межах єдиної системи для закладу середньої освіти, у вітчизняній практиці представлені недостатньо.

Актуальність теми визначається наявною невідповідністю між законодавчо закріпленим правом учнів з особливими освітніми потребами на рівний доступ до цифрових навчальних ресурсів та фактичним станом веб-забезпечення закладів загальної середньої освіти України.

Об'єкт дослідження – процес забезпечення цифрової доступності освітніх веб-ресурсів для учнів з особливими освітніми потребами.

Предмет дослідження – методи та засоби проектування і розробки веб-сайту з інформатики для учнів з інклюзією на основі стандарту WCAG 2.1.

Мета дослідження – спроектувати та розробити веб-сайт з інформатики для учнів ліцею «Традиція», який відповідає вимогам стандарту WCAG 2.1 рівня AA та забезпечує збереження індивідуальних профілів доступності користувачів.

Для досягнення зазначеної мети було сформульовано такі *завдання*:

1. Проаналізувати концептуальні засади інклюзивної освіти та вимоги міжнародних стандартів веб-доступності WCAG 2.1 і WAI-ARIA.
2. Провести порівняльний аналіз існуючих освітніх веб-платформ за критеріями відповідності стандартам доступності.
3. Визначити функціональні вимоги до системи на основі аналізу цільової аудиторії та обґрунтувати вибір технологічного стеку розробки.
4. Спроектувати структуру бази даних для збереження профілів доступності та архітектуру системи з використанням UML-моделювання.
5. Реалізувати веб-сайт з підтримкою функцій доступності, зокрема модуль синтезу мовлення та підтримку екранних зчитувачів.
6. Провести тестування розробленої системи на відповідність стандарту WCAG 2.1.

Методи дослідження: теоретичні – аналіз наукової літератури та нормативних документів у сфері веб-доступності та інклюзивної освіти, порівняльний аналіз існуючих програмних рішень, синтез та узагальнення отриманих результатів; практичні – проектування архітектури програмної системи, розробка програмного забезпечення, автоматизоване та ручне тестування доступності.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ ПРОЄКТУВАННЯ ІНКЛЮЗИВНИХ ВЕБ-РЕСУРСІВ В ОСВІТНЬОМУ СЕРЕДОВИЩІ

1.1. Концептуальні основи інклюзивної освіти та цифрової доступності

Поняття інклюзивної освіти у вітчизняному законодавстві отримало чітке нормативне визначення з прийняттям Закону України «Про освіту» (2017), згідно з яким інклюзивне навчання розглядається як система освітніх послуг, що забезпечує реалізацію права на освіту осіб з особливими освітніми потребами в умовах закладів загальної середньої освіти [1]. Це визначення відображає принциповий концептуальний зсув у підходах до організації навчання: від моделі сегрегації, за якою діти з порушеннями розвитку здобували освіту у відокремлених спеціальних установах, до моделі інтеграції у загальноосвітнє середовище. Водночас сама по собі фізична присутність учня з особливими потребами в загальному класі не є достатньою умовою інклюзії – необхідною є системна адаптація навчального середовища, яка охоплює методики викладання, дидактичні матеріали, організацію простору та, в умовах цифровізації освітнього процесу, програмні інструменти взаємодії з навчальним контентом.

Закон України «Про повну загальну середню освіту» (2020) конкретизує ці положення, зобов'язуючи заклади середньої освіти створювати умови для навчання дітей з особливими освітніми потребами та забезпечувати їхній психолого-педагогічний супровід [8]. На практиці, однак, виконання цих вимог зосереджується переважно на організаційному та архітектурному вимірах – наявності тьюторів, пандусів, спеціального обладнання – тоді як цифровий вимір доступності залишається поза увагою. Більшість освітніх веб-ресурсів, що використовуються в українських закладах середньої освіти, проєктуються без урахування потреб учнів з порушеннями зору, моторики чи когнітивних функцій, що фактично формує цифровий бар'єр для повноцінної участі цих учнів у навчальному процесі.

Категорії учнів, для яких відсутність цифрової доступності є суттєвим бар'єром, можна розділити на три основні групи відповідно до характеру їхніх

освітніх потреб. До першої належать учні з порушеннями зору, для яких принципового значення набуває можливість зміни колірної схеми інтерфейсу на висококонтрастну, а також масштабування тексту без втрати структури сторінки. Другу групу складають учні з порушеннями дрібної моторики, які мають труднощі з використанням маніпулятора «миша» та потребують повноцінної навігації засобами клавіатури. Третя група – учні з когнітивними особливостями, зокрема з дислексією або розладами уваги, для яких перевантажений візуальний ряд, дрібний текст, анімовані елементи та шрифти із засічками суттєво ускладнюють сприйняття матеріалу. Ця класифікація відповідає підходу, прийнятому у міжнародній практиці проєктування доступних інтерфейсів, де потреби користувачів систематизуються за типами порушень: зорові, моторні, слухові та когнітивні [9].

Концептуальною основою для проєктування цифрового середовища, придатного для всіх зазначених груп, є принцип універсального дизайну (Universal Design). У педагогічному контексті цей принцип отримав розвиток у концепції Universal Design for Learning (UDL), розробленій Центром прикладних спеціальних технологій (CAST, США). Концепція UDL передбачає, що навчальне середовище має від початку проєктуватися з урахуванням різноманітності учнів, а не адаптуватися постфактум для окремих категорій [9]. Стосовно веб-середовища це означає, що доступність не є додатковою функцією, яку можна реалізувати після завершення основної розробки, – вона є невід'ємною складовою архітектурних рішень, що закладаються на етапі проєктування системи. Рисунок 1.1 демонструє три основні принципи UDL, на яких ґрунтується проєктування доступного навчального середовища.

I. Provide Multiple Means of Representation	II. Provide Multiple Means of Action and Expression	III. Provide Multiple Means of Engagement
Perception	Physical action	Recruiting interest
Language, expressions, and symbols	Expression and communication	Sustaining effort and persistence
Comprehension	Executive function	Self-regulation

Рисунок 1.1 – Три принципи Universal Design for Learning (UDL)

Перехід від загальнопедагогічного розуміння інклюзії до її технічного виміру у веб-середовищі відбувається через поняття цифрової доступності. Цифрова доступність – це властивість веб-ресурсу або програмного продукту, яка забезпечує його повноцінне використання особами з різними типами порушень за допомогою як стандартних браузерів, так і допоміжних технологій [10]. До допоміжних технологій належать екранні зчитувачі (screen readers), що озвучують вміст сторінки для незрячих користувачів, програми збільшення зображення, альтернативні пристрої введення (наприклад, джойстики або педалі для користувачів з порушеннями моторики), а також спеціалізовані шрифти на кшталт OpenDyslexic, розроблені для полегшення читання особами з дислексією. Принципово важливим є те, що доступний веб-ресурс не повинен вимагати від користувача встановлення додаткового програмного забезпечення – необхідний функціонал має бути вбудований безпосередньо в інтерфейс системи.

З технічного погляду цифрова доступність веб-ресурсу визначається двома взаємопов'язаними характеристиками: семантичною коректністю розмітки та гнучкістю параметрів відображення. Семантична коректність означає, що HTML-структура сторінки відображає змістову ієрархію документа – заголовки першого, другого і третього рівнів відповідають фактичній структурі тексту, навігаційні елементи позначені відповідними ролями, а всі нетекстові об'єкти мають текстові альтернативи. Ця умова є необхідною для коректної роботи екранних зчитувачів, які інтерпретують саме семантичну структуру документа, а не його візуальне представлення. Гнучкість параметрів відображення, у свою чергу, означає здатність інтерфейсу змінювати колірну схему, розмір шрифту та міжрядковий інтервал без втрати функціональності – і саме ця характеристика є предметом технічної реалізації в межах даної роботи.

Таким чином, проєктування веб-ресурсу для учнів з інклюзією є задачею, що перебуває на перетині педагогічної теорії та веб-інженерії. Педагогічний вимір визначає, які категорії користувачів потребують підтримки і які саме бар'єри необхідно усунути. Технічний вимір визначає, якими інструментами і в рамках яких стандартів ці бар'єри усуваються.

1.2. Міжнародні стандарти веб-доступності WCAG 2.1 та WAI-ARIA

Нормативну основу проектування доступних веб-ресурсів складає система документів, розроблених Консорціумом Всесвітньої павутини (W3C) у межах ініціативи Web Accessibility Initiative (WAI). Центральним документом цієї системи є настанова WCAG 2.1 (Web Content Accessibility Guidelines, версія 2.1), прийнята як рекомендація W3C у червні 2018 року [2]. Настанова визначає технічні та змістові вимоги до веб-контенту, виконання яких забезпечує його придатність для використання особами з різними типами порушень. В Україні відповідні вимоги закріплені у ДСТУ EN 301 549:2022 «Інформаційні технології. Вимоги щодо доступності продуктів та послуг ІКТ», який гармонізований з європейськими нормами і безпосередньо посилається на WCAG 2.1 як на технічний стандарт для веб-середовища [11].

Архітектура WCAG 2.1 побудована за ієрархічним принципом: чотири базові принципи деталізуються у тринадцяти керівних настановах, кожна з яких містить конкретні критерії успіху, що підлягають перевірці. Чотири принципи утворюють абревіатуру POUR – Perceivable (сприйнятливість), Operable (керованість), Understandable (зрозумілість), Robust (надійність) [12]. Принцип сприйнятливості вимагає, щоб уся інформація та компоненти інтерфейсу були представлені у формі, доступній для сприйняття користувачем, – зокрема, щоб нетекстові об'єкти мали текстові альтернативи, а відеоматеріали супроводжувалися субтитрами. Принцип керованості передбачає, що користувач має можливість керувати всіма функціями інтерфейсу виключно засобами клавіатури, без застосування маніпулятора «миша». Принцип зрозумілості стосується передбачуваності навігації та читабельності тексту – інтерфейс має поводитись однаково на всіх сторінках, а мова документа має бути визначена програмно. Принцип надійності вимагає, щоб контент коректно інтерпретувався різними браузерами та допоміжними технологіями, що досягається через використання валідної семантичної розмітки HTML. Рисунок 1.2 ілюструє ієрархічну структуру стандарту WCAG 2.1.

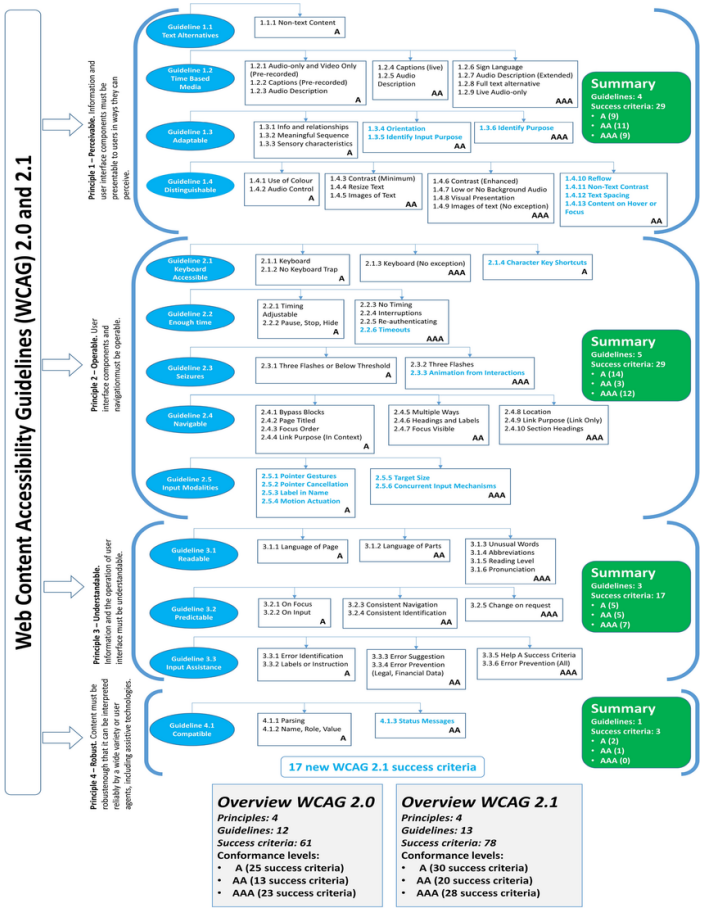

Рисунок 1.2 – Ієрархічна структура стандарту WCAG 2.1

Кожен критерій успіху WCAG 2.1 віднесений до одного з трьох рівнів відповідності: A, AA або AAA. Рівень A охоплює мінімальні вимоги, без виконання яких окремі групи користувачів взагалі не матимуть доступу до контенту. Рівень AA є обов'язковим для державних установ та освітніх закладів у більшості країн – він включає, зокрема, вимогу до коефіцієнта контрастності тексту не менше 4,5:1 відносно фону, вимогу щодо видимого індикатора фокусу клавіатури та вимогу до наявності декількох способів навігації сторінкою. Рівень AAA є найвищим і містить вимоги, реалізація яких у повному обсязі для всього сайту є технічно складною – наприклад, забезпечення сурдоперекладу відеоматеріалів або пояснення всіх абревіатур. У межах даної роботи за цільовий рівень відповідності обрано AA, що відповідає міжнародній практиці для освітніх веб-ресурсів [13].

Серед нових критеріїв, введених у версії 2.1 порівняно з попередньою WCAG 2.0, особливого значення для проєктованої системи набувають критерії, пов'язані з мобільною доступністю та когнітивними порушеннями. Зокрема,

критерій 1.4.10 (Reflow) вимагає, щоб контент не вимагав горизонтального прокручування при масштабуванні до 400%, а критерій 1.4.11 (Non-text Contrast) поширює вимоги до контрастності не лише на текст, а й на елементи інтерфейсу – кнопки, поля введення, індикатори стану. Критерій 2.5.3 (Label in Name) встановлює, що текстова мітка інтерактивного елемента має збігатися з його програмним найменуванням, що забезпечує коректну роботу технологій голосового керування [14]. Сукупність цих вимог формує конкретний перелік технічних завдань, які має вирішити реалізація модуля доступності у складі веб-сайту.

Паралельно з WCAG у межах ініціативи WAI розроблено специфікацію WAI-ARIA 1.1 (Accessible Rich Internet Applications), яка є самостійним, але взаємопов'язаним із WCAG документом [5]. Потреба у WAI-ARIA виникла у зв'язку з поширенням динамічних односторінкових застосунків (SPA), побудованих на JavaScript-фреймворках, зокрема React.js. Стандартна семантика HTML достатня для опису статичних документів, проте не охоплює динамічних станів інтерфейсу – таких як розкритий або згорнутий стан меню, активна вкладка, діалогове вікно, що відображається, або повідомлення про помилку, що з'явилося без перезавантаження сторінки. WAI-ARIA вирішує цю проблему шляхом введення системи атрибутів, що додаються до HTML-елементів і передають допоміжним технологіям інформацію про роль, стан та властивості цих елементів [5].

Специфікація WAI-ARIA визначає три категорії атрибутів. Атрибути ролей (role) вказують на функціональне призначення елемента – наприклад, role="navigation" позначає блок навігації, role="alert" вказує на область, що містить повідомлення, яке потребує негайної уваги користувача, а role="dialog" позначає модальне вікно. Атрибути стану (state) описують поточний стан динамічного елемента: aria-expanded="true" повідомляє, що розкривний список відкритий, aria-checked="true" – що прапорець позначений, aria-disabled="true" – що елемент недоступний для взаємодії. Атрибути властивостей (property) надають додаткову семантичну інформацію: aria-label задає текстову мітку для елемента, що не має видимого текстового вмісту, aria-describedby пов'язує

елемент з блоком, що містить його розширений опис, а aria-live визначає, чи має екранний зчитувач автоматично озвучувати зміни у відповідній області сторінки [5].

Застосування WAI-ARIA у React-застосунках потребує особливої уваги до управління фокусом клавіатури. При переході між «сторінками» SPA-застосунку фактичного перезавантаження документа не відбувається, тому екранний зчитувач не отримує автоматичного сигналу про зміну контексту – фокус залишається на попередньому елементі або втрачається. Для коректної роботи допоміжних технологій необхідно програмно переміщувати фокус на заголовок нової «сторінки» або на перший змістовий елемент після кожного переходу [4]. Крім того, при відкритті модальних вікон фокус має бути заблокований всередині діалогу (focus trap) до його закриття, щоб унеможливити взаємодію з фоновим контентом засобами клавіатури. Ці технічні вимоги безпосередньо визначають архітектуру модуля управління фокусом, що розглядається у розділі 3.

Взаємозв'язок між WCAG 2.1 і WAI-ARIA полягає в тому, що WCAG формулює вимоги на рівні принципів і критеріїв («усі нетекстові елементи мають текстові альтернативи», «всі функції доступні з клавіатури»), тоді як WAI-ARIA надає конкретний технічний інструментарій для виконання цих вимог в умовах динамічних веб-застосунків. Разом ці два стандарти утворюють повну нормативно-технічну базу для проектування доступного веб-інтерфейсу, що відповідає як міжнародним, так і вітчизняним вимогам у сфері цифрової доступності.

1.3. Порівняльний аналіз існуючих освітніх веб-платформ за критеріями відповідності стандартам доступності

Для обґрунтування доцільності розробки спеціалізованого веб-ресурсу з підтримкою функцій доступності необхідно проаналізувати стан наявних освітніх платформ у цьому аспекті. З цією метою було відібрано три платформи, які репрезентують різні сегменти ринку цифрової освіти: Khan Academy як

міжнародна безкоштовна навчальна платформа з широкою аудиторією, Coursera як комерційна платформа дистанційного навчання з розвиненою технічною інфраструктурою, та «Всеукраїнська школа онлайн» (ВШО) як державний освітній ресурс України. Аналіз проводився за критеріями відповідності стандарту WCAG 2.1 рівня AA з використанням автоматизованого інструменту WAVE (Web Accessibility Evaluation Tool) та ручної перевірки навігації засобами клавіатури.

Khan Academy (khanacademy.org) є однією з найпоширеніших безкоштовних освітніх платформ у світі, аудиторія якої налічує понад 150 мільйонів зареєстрованих користувачів [15]. Платформа має задокументовану програму забезпечення доступності та відображає певний рівень відповідності WCAG. Зокрема, більшість інтерактивних елементів головної сторінки та навчальних модулів доступні з клавіатури, а відеоматеріали супроводжуються автоматично згенерованими субтитрами. Семантична структура HTML-розмітки в цілому відповідає вимогам стандарту: заголовки утворюють логічну ієрархію, а навігаційні блоки позначені відповідними ARIA-landmark-ролями. Проте перевірка виявила ряд невідповідностей рівню AA: окремі інтерактивні елементи у розділі практичних завдань не мають програмно визначених текстових міток (порушення критерію 4.1.2), а деякі графічні елементи інтерфейсу не відповідають вимогам до контрастності (критерій 1.4.11). Принципово важливим є те, що платформа не надає вбудованих засобів зміни колірної схеми або розміру шрифту – користувач змушений покладатися на системні налаштування браузера, що не є достатнім для учнів з порушеннями зору. Рисунок 1.3 демонструє інтерфейс навчального модуля Khan Academy.

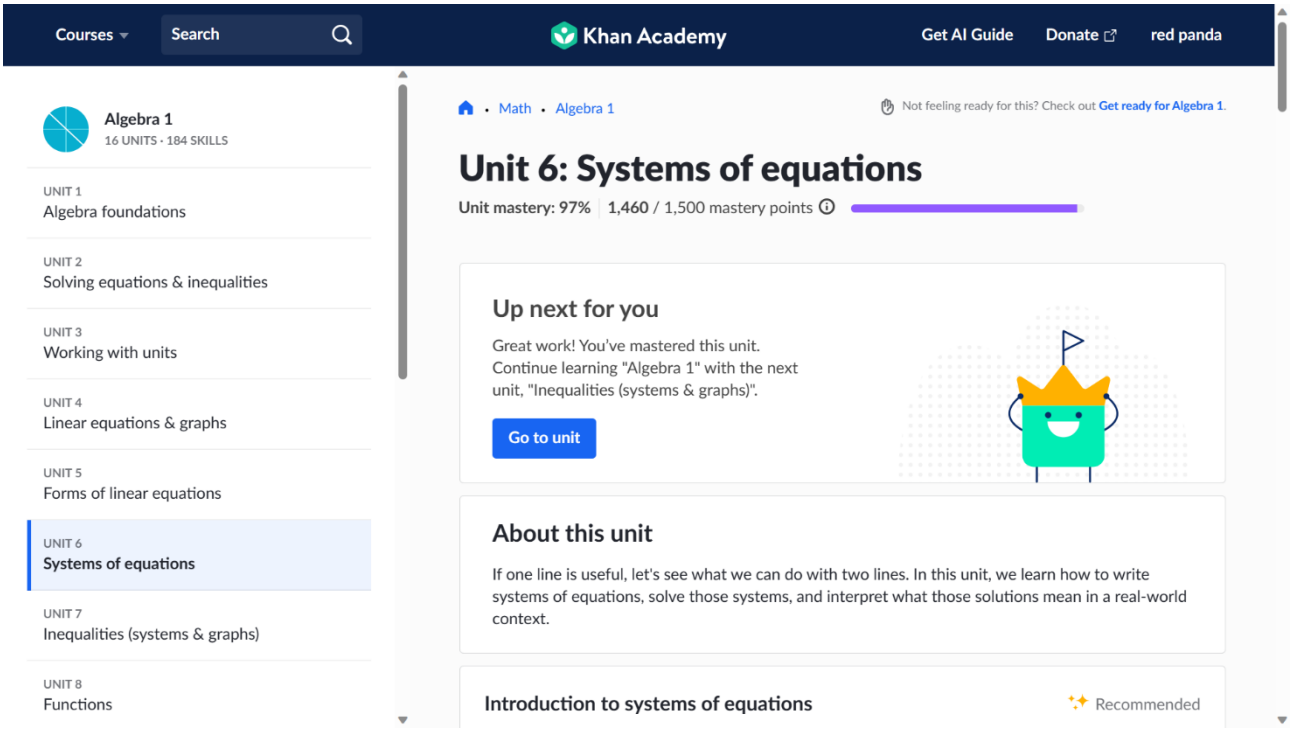

Рисунок 1.3 – Інтерфейс навчального модуля платформи Khan Academy

Coursera (coursera.org) є комерційною платформою дистанційного навчання, що надає доступ до курсів провідних університетів світу [16]. З точки зору технічної реалізації доступності платформа демонструє вищий рівень відповідності порівняно з Khan Academy: більшість елементів керування відеоплеєром повністю доступні з клавіатури, форми реєстрації та входу мають коректно пов'язані мітки, а повідомлення про помилки виводяться з атрибутом role="alert", що забезпечує їх озвучення екранними зчитувачами. Разом з тим аналіз виявив системні проблеми у розділах з тестовими завданнями: частина питань з вибором відповіді реалізована через нестандартні компоненти без належної ARIA-розмітки, що ускладнює їх використання з допоміжними технологіями. Так само, як і Khan Academy, платформа Coursera не надає вбудованого інструментарію персоналізації параметрів відображення – відсутні перемикач контрастності, регулятор розміру шрифту та підтримка спеціалізованих шрифтів для осіб з дислексією. Рисунок 1.4 відображає типовий вигляд сторінки курсу на платформі Coursera.

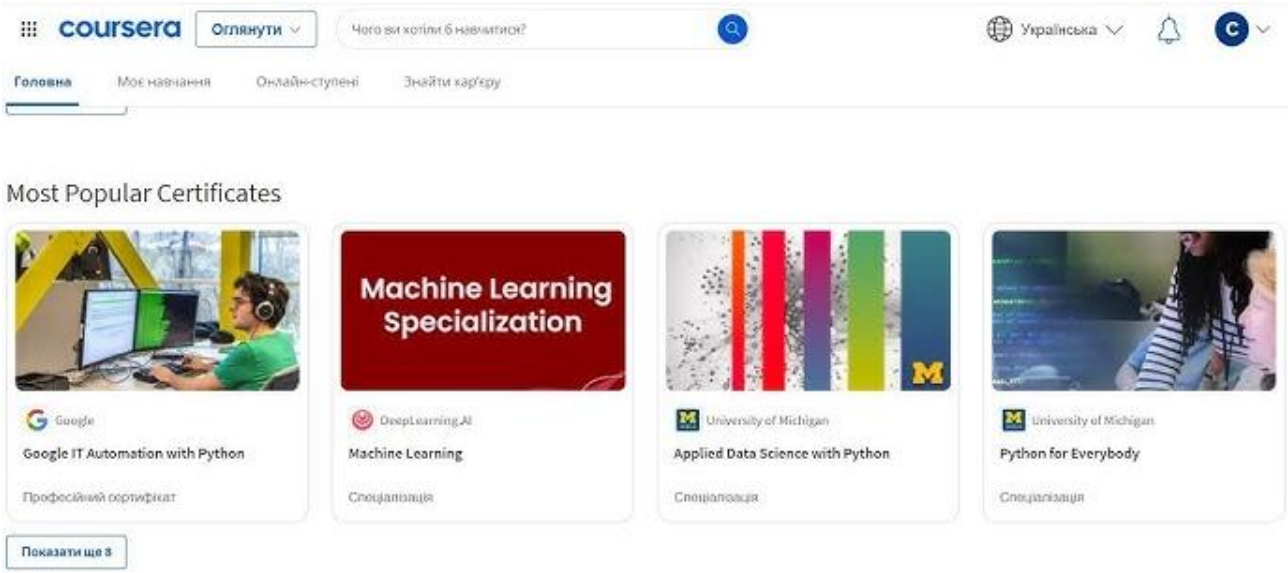

Рисунок 1.4 – Інтерфейс сторінки курсу платформи Coursera

«Всеукраїнська школа онлайн» (vso.org.ua) є державною освітньою платформою України, запущеною у 2020 році в межах програми дистанційного навчання [17]. Аналіз платформи з точки зору відповідності WCAG 2.1 виявив найбільшу кількість невідповідностей серед трьох розглянутих ресурсів. Відеоматеріали уроків не мають інтерактивних субтитрів у форматі, що підтримує пошук і масштабування тексту. Навігаційне меню не містить ARIA-атрибутів стану, внаслідок чого екранний зчитувач не отримує інформації про поточний розділ. Поля пошуку та фільтрації курсів не мають програмно визначених міток, що порушує критерій 1.3.1 стандарту WCAG 2.1. Навігація засобами лише клавіатури є частково функціональною: досягти основного контенту сторінки можливо, однак без реалізації механізму «skip links» користувач змушений послідовно обходити всі елементи шапки на кожній сторінці, що є суттєвим бар'єром для осіб з порушеннями моторики. Персоналізація параметрів відображення відсутня повністю. Рисунок 1.5 демонструє головну сторінку платформи «Всеукраїнська школа онлайн».

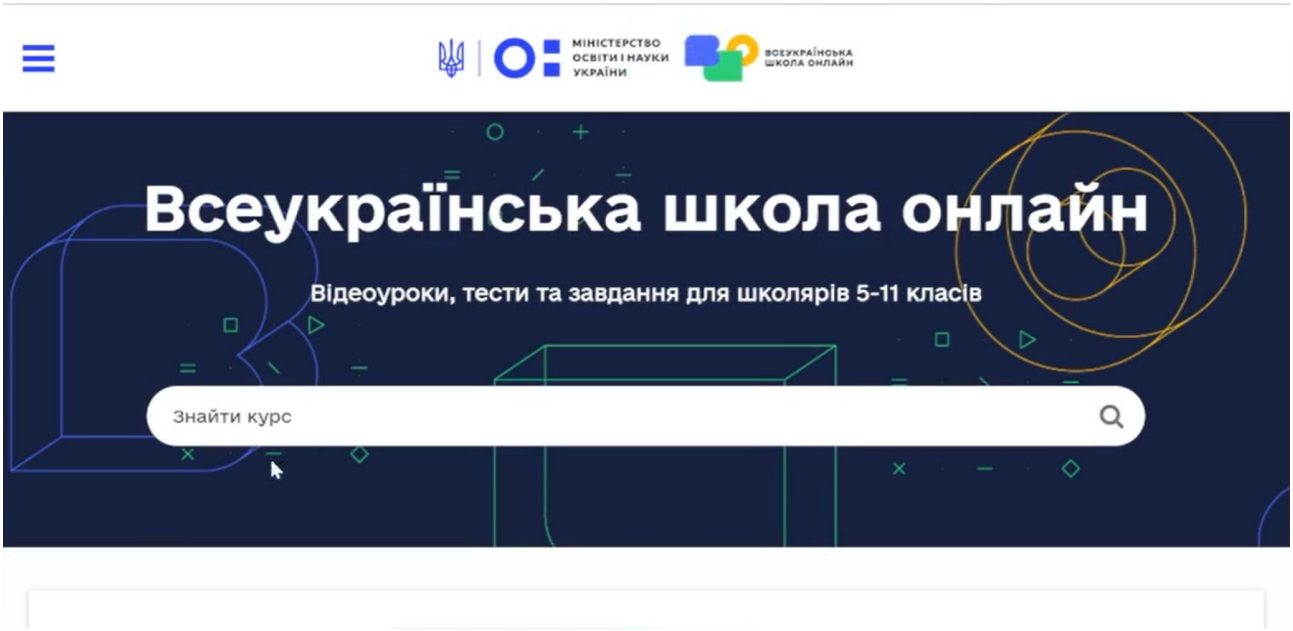

Рисунок 1.5 – Головна сторінка платформи «Всеукраїнська школа онлайн»

Результати порівняльного аналізу трьох платформ за ключовими критеріями відповідності стандарту WCAG 2.1 рівня AA зведено у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз освітніх платформ за критеріями доступності WCAG 2.1

Критерій доступності	**Khan Academy**	**Coursera**	**ВШО**	**Проектована система**
Навігація з клавіатури	Так	Так	Частково	Так
Skip links	Так	Ні	Ні	Так
Семантична розмітка (ARIA)	Частково	Частково	Ні	Так
Контрастність тексту (4,5:1)	Частково	Так	Частково	Так
Субтитри до відео	Автоматичні	Так	Ні	Так
Вбудований перемикач контрасту	Ні	Ні	Ні	Так
Масштабування шрифту	Ні	Ні	Ні	Так (100–300%)
Підтримка шрифту OpenDyslexic	Ні	Ні	Ні	Так
Збереження профілю доступності	Ні	Ні	Ні	Так (PostgreSQL)
Повідомлення про помилки (role=alert)	Частково	Так	Ні	Так

Аналіз даних таблиці 1.1 свідчить про те, що жодна з розглянутих платформ не забезпечує повного виконання вимог WCAG 2.1 рівня AA у частині персоналізації інтерфейсу. Найбільш поширеними невідповідностями є

відсутність вбудованих засобів зміни параметрів відображення та неповна реалізація ARIA-атрибутів у динамічних компонентах. Проєктована в межах даної роботи система усуває ці недоліки шляхом реалізації модуля персоналізації доступності як невід'ємної частини архітектури, а не як зовнішнього доповнення.

Висновки до розділу 1

Проведене теоретичне дослідження концептуальних засад проєктування інклюзивних веб-ресурсів в освітньому середовищі дало змогу встановити, що забезпечення цифрової доступності є комплексною задачею, яка охоплює як педагогічний, так і технічний виміри. Аналіз чинної нормативно-правової бази України, зокрема Закону «Про освіту» (2017) та Закону «Про повну загальну середню освіту» (2020), підтвердив, що право осіб з особливими освітніми потребами на рівний доступ до навчальних ресурсів закріплено на законодавчому рівні, проте механізми його реалізації у цифровому середовищі залишаються недостатньо врегульованими на практиці.

Вивчення концепції Universal Design for Learning та міжнародного стандарту WCAG 2.1 показало, що доступність веб-ресурсу визначається чотирма базовими принципами – сприйнятливістю, керованістю, зрозумілістю та надійністю, – кожен з яких формує конкретні технічні вимоги до реалізації інтерфейсу. Цільовим рівнем відповідності для освітніх веб-ресурсів є рівень АА, який охоплює вимоги до контрастності (не менше 4,5:1), повної клавіатурної навігації, видимого індикатора фокусу та коректної роботи допоміжних технологій. Специфікація WAI-ARIA є необхідним доповненням до WCAG в умовах розробки динамічних односторінкових застосунків на React.js, оскільки забезпечує передачу інформації про стан та роль елементів інтерфейсу екранним зчитувачам.

Порівняльний аналіз трьох освітніх платформ – Khan Academy, Coursera та «Всеукраїнської школи онлайн» – виявив, що жодна з них не забезпечує повного виконання вимог WCAG 2.1 рівня АА у частині персоналізації інтерфейсу. Спільною для всіх трьох платформ є відсутність вбудованих засобів зміни

колірної схеми, масштабування шрифту та збереження індивідуальних профілів доступності. Найбільш системні невідповідності зафіксовано на платформі ВШО, де відсутні механізм skip links, ARIA-атрибути навігаційних елементів та інтерактивні субтитри до відеоматеріалів.

Отримані результати підтверджують наявність практичної прогалини між законодавчо задекларованими вимогами до інклюзивності освітнього середовища та фактичним станом цифрових навчальних ресурсів. Це обґрунтовує доцільність розробки веб-сайту з вбудованим модулем персоналізації доступності, який з етапу проєктування відповідає вимогам WCAG 2.1 рівня AA та підтримує збереження індивідуальних профілів користувачів.

РОЗДІЛ 2. АНАЛІЗ ВИМОГ ТА ПРОЄКТУВАННЯ ВЕБ-САЙТУ ДЛЯ УЧНІВ З ІНКЛЮЗІЄЮ

2.1. Аналіз цільової аудиторії та формування функціональних вимог до системи

Цільову аудиторію веб-сайту складають учні закладів загальної середньої освіти, що навчаються в інклюзивних класах та мають особливі освітні потреби різного характеру. Відповідно до класифікації, прийнятої у міжнародній практиці проєктування доступних інтерфейсів, користувачів системи можна розподілити за типами порушень, що безпосередньо впливають на характер взаємодії з веб-середовищем [10]:

– учні з порушеннями зору (зниження гостроти зору, колірна сліпота) – потребують зміни колірної схеми інтерфейсу, збільшення розміру шрифту та достатнього рівня контрастності елементів;

– учні з порушеннями дрібної моторики – потребують повноцінної клавіатурної навігації без використання маніпулятора «миша», а також достатнього розміру інтерактивних елементів для сенсорного керування;

– учні з когнітивними особливостями, зокрема з дислексією або розладами уваги – потребують підтримки спеціалізованих шрифтів, збільшеного міжрядкового інтервалу та мінімізації відволікаючих анімованих елементів;

– учні з порушеннями слуху – потребують наявності субтитрів до відеоматеріалів та текстових альтернатив до аудіоконтенту.

Окрім учнів, до складу аудиторії входять також вчителі інформатики, які використовують веб-сайт як інструмент подання навчального матеріалу, та адміністратори системи, що забезпечують наповнення контентом і керування обліковими записами користувачів.

На підставі аналізу цільової аудиторії та вимог стандарту WCAG 2.1 рівня AA [2] до системи висуваються такі функціональні вимоги.

Щодо модуля доступності інтерфейсу, сайт має:

– забезпечувати перемикання між стандартною та висококонтрастною колірною схемою;

– підтримувати масштабування розміру шрифту в діапазоні від 100% до 300% без втрати структури сторінки;

– надавати можливість увімкнення шрифту OpenDyslexic для учнів з дислексією;

– забезпечувати регулювання міжрядкового інтервалу;

– зберігати обрані параметри доступності у профілі користувача між сесіями.

Щодо навігації та структури, сайт повинен:

– забезпечувати повноцінну навігацію засобами лише клавіатури на всіх сторінках;

– містити механізм «skip links» для переходу безпосередньо до основного контенту;

– відображати видимий індикатор фокусу клавіатури на всіх інтерактивних елементах;

– реалізовувати логічну семантичну структуру HTML з коректною ієрархією заголовків.

Щодо підтримки допоміжних технологій, сайт має:

– містити WAI-ARIA атрибути для всіх динамічних компонентів інтерфейсу;

– забезпечувати коректне озвучення стану інтерактивних елементів екранними зчитувачами;

– реалізовувати модуль синтезу мовлення для озвучення навчального контенту;

– програмно керувати фокусом клавіатури при переходах між розділами та відкритті модальних вікон.

Щодо навчального контенту, сайт повинен:

– надавати відеоматеріали з субтитрами;

– містити текстові альтернативи до всіх нетекстових об'єктів;

– організовувати навчальні матеріали за темами програми з інформатики з можливістю фільтрації.

Щодо системи облікових записів, сайт має:

– підтримувати реєстрацію та автентифікацію користувачів;

– зберігати індивідуальний профіль доступності кожного користувача в базі даних;

– розмежовувати права доступу між роллю учня, вчителя та адміністратора.

Нефункціональні вимоги до системи передбачають відповідність стандарту WCAG 2.1 рівня AA як обов'язкову умову приймання розробленого продукту, час відгуку інтерфейсу на дії користувача не більше 300 мс, а також коректне відображення на пристроях з різною шириною екрана – від 320 px до 2560 px.

2.2. Вибір та обґрунтування технологічного стеку розробки

Вибір технологічного стеку визначався сукупністю вимог, сформульованих у підрозділі 2.1: необхідністю динамічної зміни параметрів інтерфейсу без перезавантаження сторінки, підтримкою WAI-ARIA у компонентній моделі, збереженням профілів доступності на стороні сервера та забезпеченням реляційної цілісності даних користувачів.

Для реалізації клієнтської частини обрано бібліотеку React.js. Компонентна архітектура React дозволяє інкапсулювати логіку доступності безпосередньо в компоненті – атрибути aria-*, обробники подій клавіатури та управління фокусом визначаються один раз і відтворюються послідовно в усіх місцях використання компонента. Це унеможливлює ситуацію, коли однотипні елементи інтерфейсу на різних сторінках мають різний рівень відповідності стандарту. Застосування React також обумовлене наявністю спеціалізованих засобів для роботи з доступністю, що інтегруються безпосередньо в процес розробки [19]. Управління глобальним станом параметрів доступності реалізується через Context API, що забезпечує передачу налаштувань до всіх

компонентів дерева без необхідності передавати пропси через проміжні рівні [19].

Серверну частину реалізовано на платформі Node.js з використанням фреймворку Express.js. Node.js забезпечує уніфіковане середовище виконання JavaScript як на клієнті, так і на сервері, що спрощує організацію спільної валідації даних. REST API, реалізований засобами Express.js, обслуговує запити на збереження та завантаження профілів доступності, автентифікацію користувачів і отримання навчального контенту [20].

Для збереження даних обрано реляційну систему управління базами даних PostgreSQL. Структурований характер даних профілів доступності – фіксований набір параметрів з визначеними типами та обмеженнями – відповідає реляційній моделі зберігання. PostgreSQL забезпечує підтримку типу даних JSONB, що дозволяє зберігати розширені параметри профілю у гнучкому форматі без зміни схеми таблиці, а також транзакційну цілісність при одночасному оновленні профілю з кількох пристроїв [21].

Для автентифікації користувачів застосовується механізм JSON Web Token (JWT). Токен, що генерується після успішного входу, містить ідентифікатор користувача та його роль у системі, передається у заголовку HTTP-запиту та перевіряється на сервері без звернення до бази даних при кожному запиті [20].

Зведену характеристику технологічного стеку наведено у таблиці 2.1.

Таблиця 2.1 – Технологічний стек системи та призначення компонентів

Технологія	Рівень	Призначення у системі
React.js	Клієнт	Побудова компонентного інтерфейсу з підтримкою WAI-ARIA
Context API	Клієнт	Управління глобальним станом параметрів доступності
ElevenLabs API	Клієнт	Синтез мовлення для озвучення навчального контенту
Node.js / Express.js	Сервер	REST API для обробки запитів клієнта
JWT	Сервер	Автентифікація та авторизація користувачів
PostgreSQL	База даних	Збереження профілів доступності та навчального контенту

Наведений стек забезпечує реалізацію всіх функціональних вимог, визначених у підрозділі 2.1, у межах єдиної технологічної платформи з відкритим вихідним кодом, що виключає залежність від комерційних ліцензій та дозволяє розгортання системи на власній інфраструктурі закладу освіти.

2.3. Проєктування структури бази даних для збереження профілів доступності

Для збереження даних системи обрано реляційну модель організації даних. Цей вибір обумовлений структурованим характером предметної області: зв'язки між користувачами, профілями доступності, навчальним контентом та результатами тестування є чітко визначеними та стабільними, що відповідає природі реляційної моделі [22]. Реляційна модель також забезпечує декларативну цілісність даних засобами зовнішніх ключів та обмежень CHECK, що унеможливлює появу некоректних значень на рівні бази даних незалежно від логіки застосунку.

Спроєктована схема бази даних приведена до третьої нормальної форми (3НФ). Відповідність першій нормальній формі (1НФ) забезпечується тим, що кожен атрибут кожної таблиці містить лише атомарні значення – зокрема, параметри профілю доступності не об'єднані в один рядок, а зберігаються як окремі поля таблиці accessibility_profiles. Відповідність другій нормальній формі (2НФ) означає, що кожен неключовий атрибут функціонально залежить від усього первинного ключа, а не від його частини – ця вимога виконується автоматично, оскільки всі таблиці схеми мають простий первинний ключ типу SERIAL [22]. Відповідність третій нормальній формі (3НФ) передбачає відсутність транзитивних залежностей між неключовими атрибутами. Потенційне порушення 3НФ могло виникнути, якби, наприклад, таблиця test_results містила поля topic_title або lesson_title – тоді ці атрибути залежали б не від первинного ключа результату, а від зовнішнього ключа test_id. У спроєктованій схемі такі дані отримуються через JOIN-запити до відповідних таблиць, що усуває транзитивну залежність [22].

Таблиця users є центральною сутністю схеми, оскільки більшість інших таблиць містять посилання на ідентифікатор користувача. Структуру таблиці наведено у таблиці 2.2.

Таблиця 2.2 – Структура таблиці users

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор
email	VARCHAR(255)	UNIQUE, NOT NULL	Електронна адреса
password_hash	VARCHAR(255)	NOT NULL	Хеш паролю
role	VARCHAR(20)	CHECK (role IN ('student', 'teacher', 'admin'))	Роль у системі
created_at	TIMESTAMPTZ	DEFAULT NOW()	Дата реєстрації

Таблиця accessibility_profiles зберігає індивідуальні параметри доступності кожного користувача та пов'язана з users відношенням «один до одного» через зовнішній ключ user_id. Профіль створюється автоматично при реєстрації користувача зі значеннями за замовчуванням, що відповідають стандартному режиму відображення. Структуру таблиці наведено у таблиці 2.3.

Таблиця 2.3 – Структура таблиці accessibility_profiles

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор
user_id	INTEGER	FK – users, NOT NULL	Ідентифікатор користувача
font_size	INTEGER	DEFAULT 100	Розмір шрифту у відсотках
contrast_mode	BOOLEAN	DEFAULT FALSE	Висококонтрастний режим
font_family	VARCHAR(50)	DEFAULT 'standard'	Гарнітура шрифту
line_height	NUMERIC(3,1)	DEFAULT 1.5	Міжрядковий інтервал
tts_enabled	BOOLEAN	DEFAULT FALSE	Синтез мовлення увімкнено
updated_at	TIMESTAMPTZ	DEFAULT NOW()	Дата останнього оновлення

Навчальний контент організовано у дворівневій ієрархії: таблиця topics містить теми навчальної програми з інформатики, таблиця lessons – окремі уроки, кожен з яких належить до певної теми. Структури цих таблиць наведено у таблицях 2.4 та 2.5 відповідно.

Таблиця 2.4 – Структура таблиці topics

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор
title	VARCHAR(255)	NOT NULL	Назва теми
order_index	INTEGER	NOT NULL	Порядок теми у програмі

Таблиця 2.5 – Структура таблиці lessons

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор
topic_id	INTEGER	FK – topics, NOT NULL	Ідентифікатор теми
title	VARCHAR(255)	NOT NULL	Назва уроку
content_html	TEXT	–	HTML-вміст уроку
video_url	VARCHAR(500)	–	Посилання на відео
transcript_text	TEXT	–	Текстова стенограма відео
created_by	INTEGER	FK – users	Ідентифікатор автора
created_at	TIMESTAMPTZ	DEFAULT NOW()	Дата створення

Підсистема тестування охоплює чотири таблиці: tests, questions, answer_options та test_results. Таблиця tests підтримує два типи тестів – після уроку та після теми – що реалізується через поле type та два зовнішні ключі, один з яких є NULL залежно від типу тесту. Таблиця questions пов'язана з tests відношенням «один до багатьох» та містить поле question_type, що визначає режим вибору відповіді. Таблиця answer_options зберігає варіанти відповідей для кожного питання, при цьому поле is_correct може набувати значення TRUE для кількох рядків одного питання у випадку множинного вибору. Структури таблиць підсистеми тестування наведено у таблицях 2.6–2.8.

Таблиця 2.6 – Структура таблиці tests

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор
type	VARCHAR(10)	CHECK (type IN ('lesson','topic'))	Тип тесту
lesson_id	INTEGER	FK – lessons, NULLABLE	Ідентифікатор уроку
topic_id	INTEGER	FK – topics, NULLABLE	Ідентифікатор теми
title	VARCHAR(255)	NOT NULL	Назва тесту
created_by	INTEGER	FK – users	Ідентифікатор автора
created_at	TIMESTAMPTZ	DEFAULT NOW()	Дата створення

Таблиця 2.7 – Структура таблиці questions

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор
test_id	INTEGER	FK – tests, NOT NULL	Ідентифікатор тесту
question_text	TEXT	NOT NULL	Текст питання
question_type	VARCHAR(10)	CHECK (question_type IN ('single','multiple'))	Тип вибору відповіді
order_index	INTEGER	NOT NULL	Порядок питання у тесті

Таблиця 2.8 – Структура таблиці answer_options

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор
question_id	INTEGER	FK – questions, NOT NULL	Ідентифікатор питання
option_text	TEXT	NOT NULL	Текст варіанту відповіді
is_correct	BOOLEAN	NOT NULL, DEFAULT FALSE	Ознака правильної відповіді

Таблиця test_results фіксує факт проходження тесту користувачем та отриманий результат. Поля score та max_score мають тип NUMERIC(5,2), що дозволяє коректно відображати дробові бали при частковому зарахуванні

відповіді на питання з множинним вибором. Структуру таблиці наведено у таблиці 2.9.

Таблиця 2.9 – Структура таблиці test_results

Поле	Тип	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор
user_id	INTEGER	FK – users, NOT NULL	Ідентифікатор користувача
test_id	INTEGER	FK – tests, NOT NULL	Ідентифікатор тесту
score	NUMERIC(5,2)	NOT NULL	Отриманий бал
max_score	NUMERIC(5,2)	NOT NULL	Максимально можливий бал
passed_at	TIMESTAMPTZ	DEFAULT NOW()	Дата проходження

Взаємозв'язки між усіма таблицями схеми унаочнює ER-діаграма, наведена на рисунку 2.1.

ER-діаграма відображає дев'ять сутностей та зв'язки між ними. Таблиця users пов'язана з accessibility_profiles відношенням «один до одного» – кожен зареєстрований користувач має рівно один профіль доступності. Зв'язок users з таблицями lessons та tests через поле created_by є відношенням «один до багатьох» – один вчитель може створити довільну кількість уроків та тестів. Таблиця topics пов'язана з lessons відношенням «один до багатьох», оскільки одна тема програми охоплює кілька уроків. Аналогічне відношення існує між topics і tests та між lessons і tests – обидва зв'язки є необов'язковими з боку tests, що відповідає логіці двох типів тестів: тест після уроку заповнює lesson_id і залишає topic_id як NULL, тест після теми – навпаки. Таблиця tests пов'язана з questions відношенням «один до багатьох», questions з answer_options – також «один до багатьох», оскільки кожне питання має щонайменше два варіанти відповіді. Зв'язок users з test_results та tests з test_results є відношенням «один до багатьох» – один користувач може проходити тест кількаразово, і кожен факт проходження фіксується як окремий запис [22].

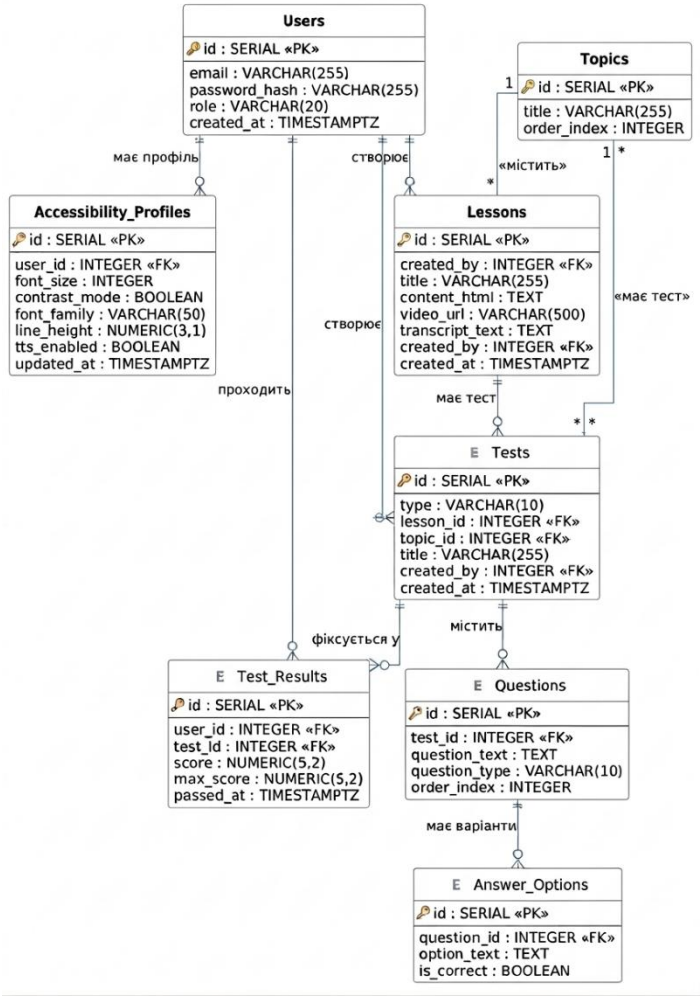

Рисунок 2.1 – ER-діаграма бази даних системи

Спроєктована схема забезпечує реалізацію всіх вимог до збереження даних, визначених у підрозділі 2.1: персоналізований профіль доступності зберігається окремо для кожного користувача, навчальний контент організовано у дворівневій ієрархії тем та уроків, а результати тестування фіксуються з точністю до дробових балів. Нормалізація до 3НФ виключає надлишковість даних та забезпечує цілісність схеми при оновленні записів [22].

2.4. Проєктування архітектури системи та UML-моделювання

Система побудована за трирівневою архітектурою, що передбачає чіткий розподіл відповідальності між рівнем представлення, рівнем бізнес-логіки та рівнем даних [24]. Рівень представлення реалізовано як односторінковий застосунок (SPA) на базі React.js, що виконується у веб-браузері користувача.

Рівень бізнес-логіки представлено сервером Node.js з фреймворком Express.js, який обробляє HTTP-запити від клієнта, виконує автентифікацію та формує відповіді. Рівень даних складає СУБД PostgreSQL, що зберігає всі постійні дані системи відповідно до схеми, спроектованої у підрозділі 2.3. Такий розподіл забезпечує незалежність рівнів – зміни у клієнтській частині не потребують модифікації серверної логіки, і навпаки [24].

Склад компонентів кожного рівня та зв'язки між ними відображено на діаграмі компонентів (рисунок 2.2).

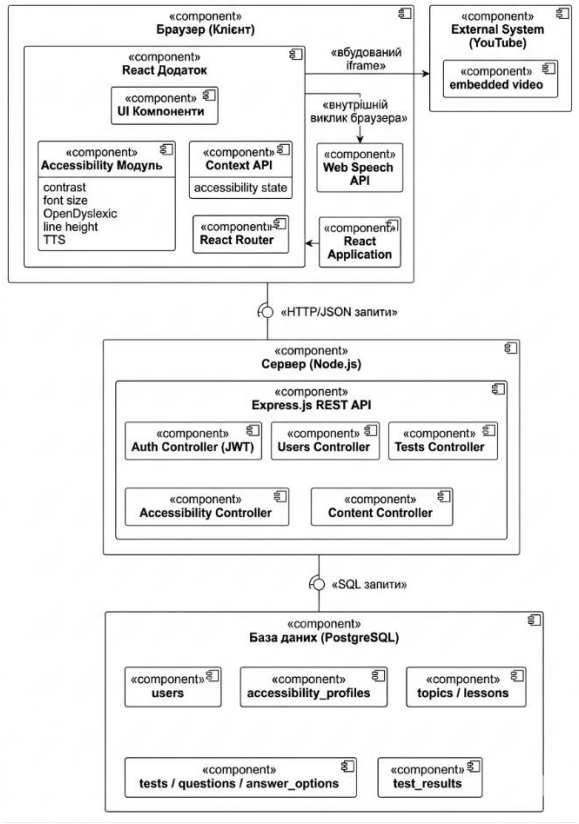

Рисунок 2.2 – Діаграма компонентів системи

Клієнтська частина містить React-застосунок з вбудованим модулем доступності, Context API для управління глобальним станом параметрів відображення, React Router для маршрутизації між розділами. Серверна частина організована у п'ять контролерів у складі Express.js REST API: Auth Controller відповідає за автентифікацію на основі JWT, Users Controller – за управління обліковими записами, Accessibility Controller – за збереження та завантаження профілів доступності, Content Controller – за операції з темами та уроками, Tests

Controller – за управління тестами та обробку результатів. Зовнішнім компонентом системи є YouTube, відео з якого вбудовується у сторінки уроків через iframe без проксіювання через власний сервер.

Сценарії взаємодії трьох категорій користувачів з функціями системи систематизовано у діаграмі варіантів використання (рисунок 2.3). Актор «Студент» має доступ до перегляду навчальних матеріалів та відео уроків, проходження тестів, перегляду власних результатів, а також до налаштування профілю доступності та увімкнення синтезу мовлення. Актор «Вчитель» має повноваження створювати та редагувати уроки і тести, а також переглядати результати тестування всіх учнів. Актор «Адміністратор» керує обліковими записами користувачів та темами навчальної програми. Усі три актори використовують спільний сценарій реєстрації та входу в систему.

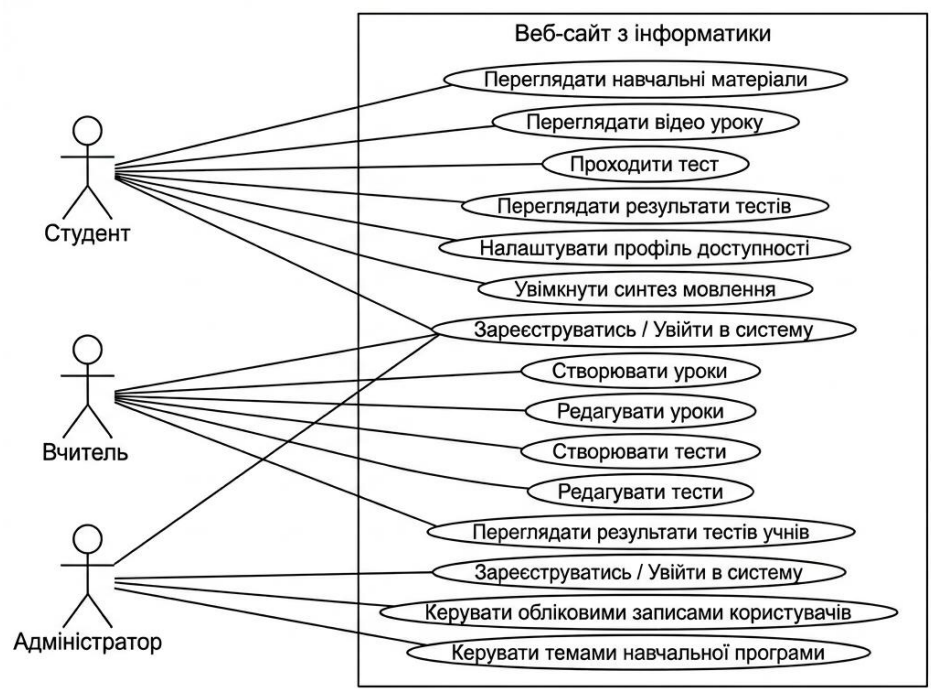

Рисунок 2.3 – Діаграма варіантів використання системи

Взаємодію компонентів при збереженні профілю доступності деталізовано у діаграмі послідовності (рисунок 2.4). Після того як користувач змінює параметр доступності у панелі налаштувань, React UI викликає dispatch з відповідною дією, Context API оновлює глобальний стан і CSS-зміни застосовуються до інтерфейсу негайно – без звернення до сервера. Збереження

параметрів у базі даних відбувається лише після явного підтвердження користувачем через кнопку «Зберегти профіль». У цей момент React UI формує PUT-запит до ендпоінту /api/accessibility/:userId з тілом, що містить усі поточні параметри профілю. Express.js API перевіряє JWT токен, після чого виконує UPDATE у таблиці accessibility_profiles та повертає клієнту підтвердження з міткою часу оновлення.

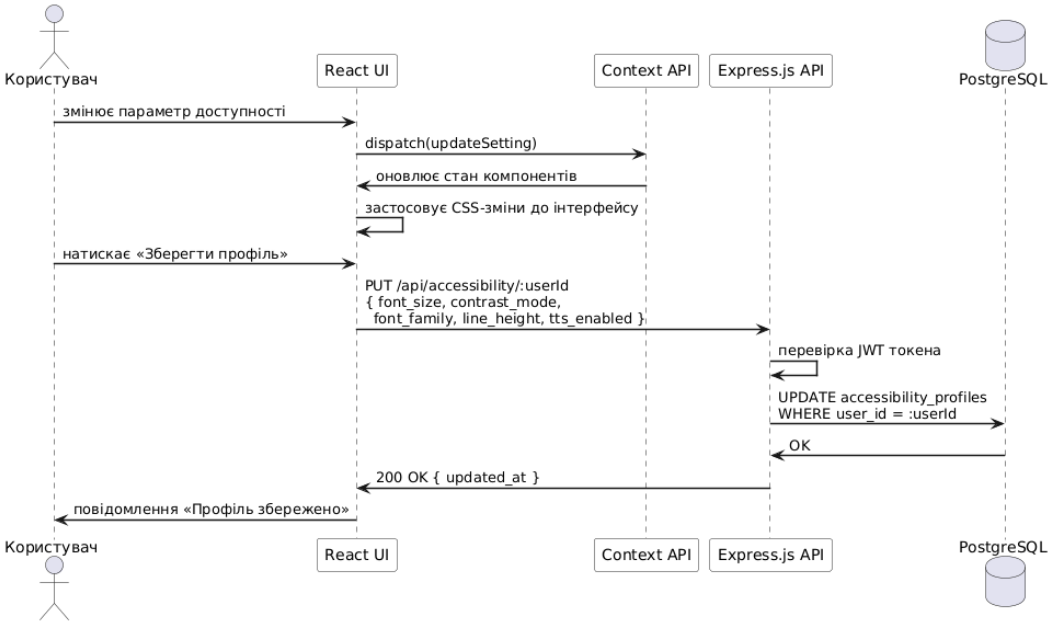

Рисунок 2.4 – Діаграма послідовності: збереження профілю доступності

Діаграма послідовності для сценарію проходження тесту наведена на рисунку 2.5. При відкритті сторінки тесту React UI надсилає GET-запит до /api/tests/:testId, у відповідь на який сервер виконує три послідовних запити до бази даних: отримання даних тесту, списку питань та варіантів відповідей. Усі дані повертаються клієнту єдиною структурою і відображаються користувачу. Після того як студент обирає відповіді та підтверджує завершення тесту, React UI надсилає POST-запит з масивом обраних відповідей. Сервер перевіряє JWT токен, виконує підрахунок балів з урахуванням типу кожного питання та записує результат у таблицю test_results. Клієнт отримує у відповіді набрані бали, максимально можливий бал та час проходження [25].

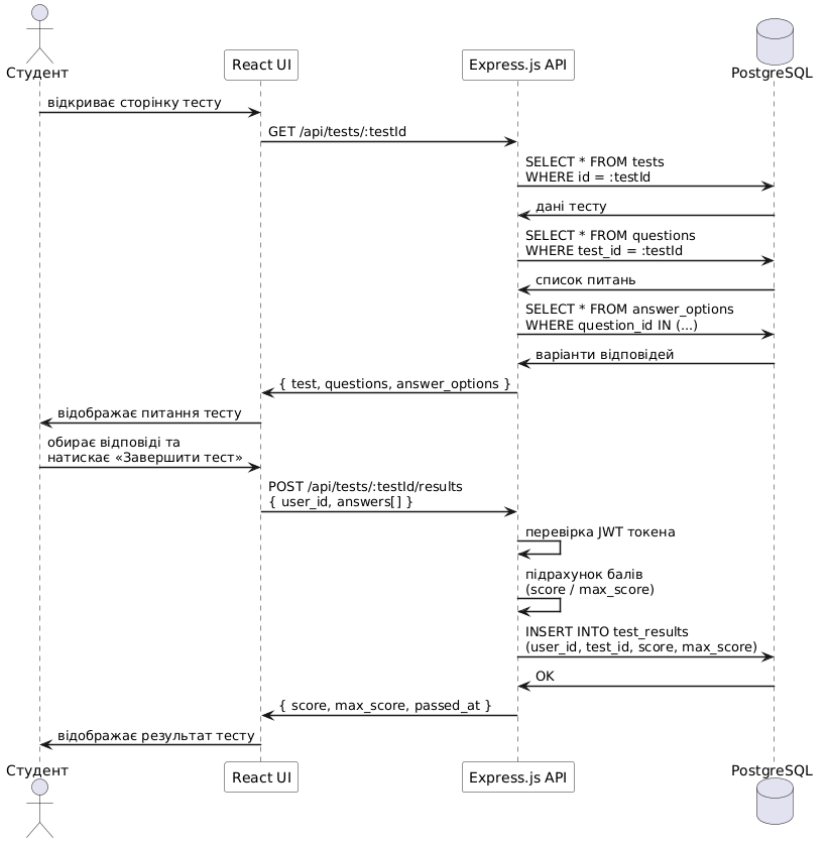

Рисунок 2.5 – Діаграма послідовності: проходження тесту

Фізичне розміщення компонентів системи відображено на діаграмі розгортання (рисунок 2.6). Система розгортається на трьох вузлах: пристрій користувача з веб-браузером, веб-сервер з Node.js та сервер бази даних з PostgreSQL. Взаємодія між пристроєм користувача та веб-сервером здійснюється за протоколом HTTPS через REST API. Зв'язок між веб-сервером та сервером бази даних реалізується через протокол TCP з використанням SQL-запитів. Вбудовані відеоматеріали YouTube завантажуються безпосередньо на пристрій користувача через HTTPS без участі власного сервера системи, що знижує навантаження на серверну інфраструктуру [24].

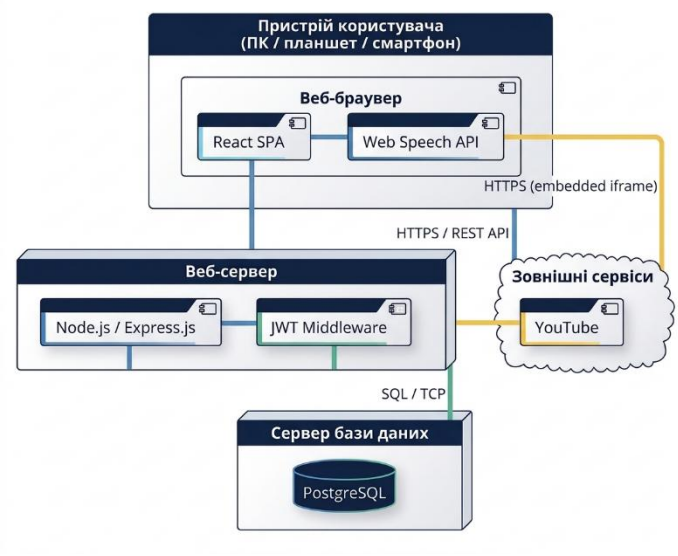

Рисунок 2.6 – Діаграма розгортання системи

Сукупність розроблених UML-діаграм забезпечує повний опис системи на трьох рівнях абстракції: функціональному – через діаграму варіантів використання, структурному – через діаграми компонентів та розгортання, і поведінковому – через діаграми послідовності [23].

Висновки до розділу 2

Проведений аналіз вимог та проєктування веб-сайту з інформатики для учнів з інклюзією дав змогу сформувати повну технічну основу для програмної реалізації системи.

Аналіз цільової аудиторії показав, що система має обслуговувати чотири категорії користувачів з особливими освітніми потребами – з порушеннями зору, моторики, слуху та когнітивними особливостями, – а також вчителів та адміністраторів. На підставі цього аналізу сформульовано функціональні вимоги до п'яти підсистем: модуля доступності інтерфейсу, навігації, підтримки допоміжних технологій, навчального контенту та системи облікових записів. Нефункціональні вимоги визначають відповідність стандарту WCAG 2.1 рівня AA як обов'язкову умову приймання системи.

Вибір технологічного стеку – React.js, Node.js/Express.js та PostgreSQL – обґрунтовано технічними характеристиками кожного компонента у контексті

вимог доступності. Компонентна архітектура React забезпечує послідовне застосування WAI-ARIA атрибутів та логіки управління фокусом у всіх елементах інтерфейсу. REST API на базі Express.js обслуговує запити на збереження та завантаження профілів доступності, а PostgreSQL забезпечує реляційну цілісність даних та підтримку типу JSONB для розширених параметрів профілю.

Спроєктована схема бази даних охоплює вісім таблиць та приведена до третьої нормальної форми, що виключає надлишковість даних. Схема підтримує збереження індивідуальних профілів доступності для кожного користувача, дворівневу ієрархію навчального контенту, два типи тестів – після уроку та після теми – з підтримкою питань одиночного та множинного вибору, а також фіксацію результатів тестування з точністю до дробових балів.

Архітектуру системи описано засобами п'яти UML-діаграм. Діаграма компонентів відображає трирівневу організацію системи та склад компонентів кожного рівня. Діаграма варіантів використання систематизує сценарії взаємодії трьох акторів з функціями системи. Дві діаграми послідовності деталізують взаємодію компонентів при збереженні профілю доступності та проходженні тесту. Діаграма розгортання визначає фізичне розміщення компонентів на трьох вузлах інфраструктури.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Реалізація інтерфейсу користувача з підтримкою вимог WCAG 2.1

Інтерфейс системи реалізовано як односторінковий застосунок на базі React.js з маршрутизацією через React Router v6. Структура маршрутів охоплює чотирнадцять сторінок, розподілених за рівнями доступу відповідно до ролі користувача. Загальний сценарій навігації користувача залежно від його ролі відображено на рисунку 3.1.

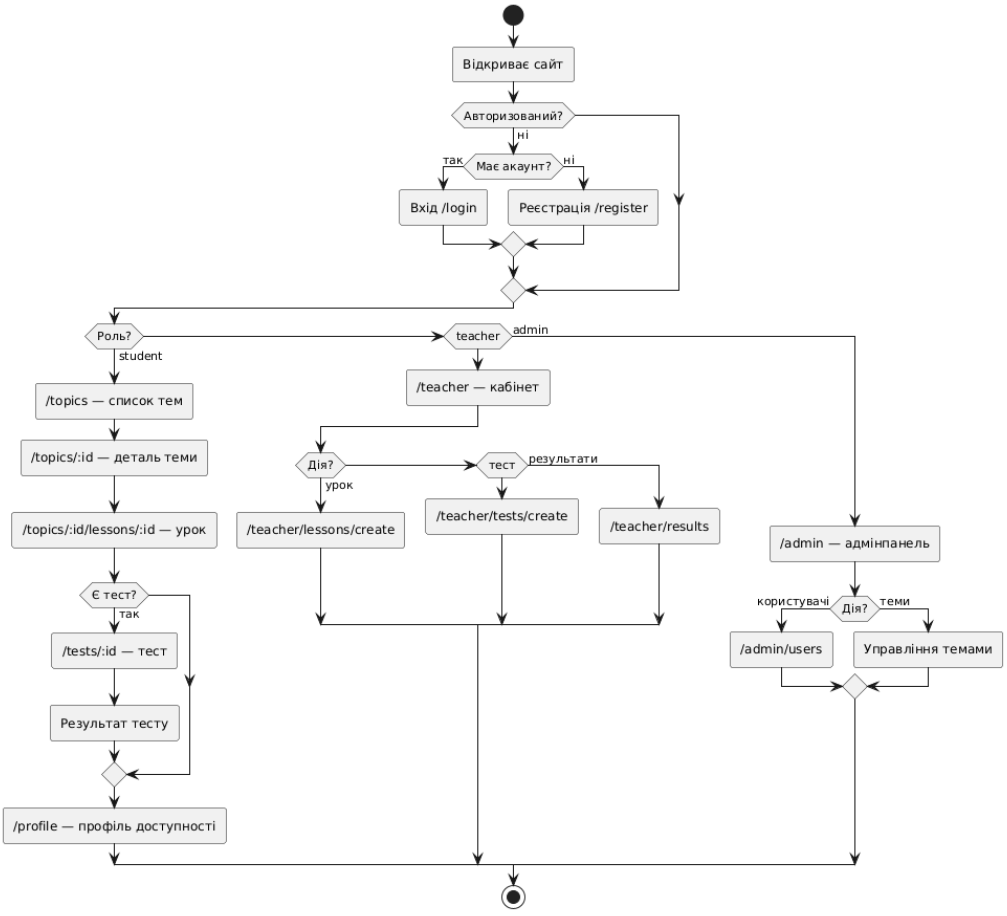

Рисунок 3.1 – User Flow діаграма системи

Захист маршрутів реалізовано через три компоненти-охоронці. PrivateRoute обмежує доступ до сторінок лише для автентифікованих користувачів – при спробі незареєстрованого користувача відкрити захищений маршрут відбувається перенаправлення на /login із збереженням початкового

URL у стані навігації для подальшого перенаправлення після успішної автентифікації. TeacherRoute додатково перевіряє наявність ролі teacher або admin, AdminRoute – виключно ролі admin. Розподіл сторінок за рівнями рольового доступу наведено у таблиці 3.1.

Таблиця 3.1 – Маршрути системи та рольовий доступ

URL	Сторінка	Гість	Student	Teacher	Admin
/	Головна	+	+	+	+
/login	Вхід	+	–	–	–
/register	Реєстрація	+	–	–	–
/topics	Список тем	–	+	+	+
/topics/:id	Деталь теми	–	+	+	+
/topics/:id/lessons/:id	Урок	–	+	+	+
/tests/:id	Тест	–	+	+	+
/profile	Профіль	–	+	+	+
/teacher	Кабінет вчителя	–	–	+	+
/teacher/lessons/create	Створити урок	–	–	+	+
/teacher/tests/create	Створити тест	–	–	+	+
/teacher/results	Результати студентів	–	–	+	+
/admin	Адмінпанель	–	–	–	+
/admin/users	Управління користувачами	–	–	–	+

Модуль доступності є центральним компонентом архітектури інтерфейсу і реалізований через AccessibilityContext – глобальний контекст React, що зберігає поточні параметри відображення та забезпечує їх застосування до всього дерева компонентів. Технічною основою модуля є CSS custom properties, визначені на псевдоелементі :root. Такий підхід дозволяє будь-якому елементу сторінки автоматично реагувати на зміну параметрів без повторного рендерингу компонентів [19]. При кожній зміні налаштувань контекст оновлює відповідні змінні на document.documentElement та синхронізує стан з localStorage:

```
    useEffect(() => {
      const root = document.documentElement;
      root.style.setProperty('--font-size-multiplier',
settings.fontSize / 100);
      root.style.setProperty('--line-height',
settings.lineHeight);
```

```
      document.body.classList.toggle('high-contrast',
settings.contrastMode);
      document.body.classList.toggle('font-opendyslexic',
        settings.fontFamily === 'opendyslexic');
      localStorage.setItem('inclusify_accessibility',
JSON.stringify(settings));
    }, [settings]);
```

Висококонтрастний режим реалізовано через додавання класу high-contrast до елемента <body>, що перевизначає кольоровий шар через CSS custom properties: колір фону змінюється на #000000, колір тексту – на #ffffff, акцентний колір – на #ffff00. Коефіцієнт контрастності тексту відносно фону у цьому режимі перевищує 21:1, що відповідає вимогам стандарту WCAG 2.1 рівня AA [2]. Масштабування шрифту підтримує діапазон від 100% до 300% з кроком 25% і реалізовано через CSS-змінну --font-size-multiplier, що застосовується до розмірів шрифтів через функцію calc(). Підтримка шрифту OpenDyslexic реалізована через директиву @font-face з підключенням накреслення з CDN та застосуванням його до всього документа через клас font-opendyslexic на елементі <body>.

Семантична розмітка сторінок відповідає вимогам стандарту щодо структурної ієрархії документа. Кожна сторінка містить елементи <header role="banner">, <main id="main-content"> та <footer role="contentinfo">. Заголовки утворюють послідовну ієрархію без пропуску рівнів: <h1> відповідає назві поточної сторінки, <h2> – секціям, <h3> – підсекціям. Питання тестів згруповано у елементи <fieldset> з відповідним <legend>, що забезпечує коректне озвучення групи варіантів відповідей екранними зчитувачами. Мова документа визначена атрибутом lang="uk" на елементі <html> [2].

Першим елементом DOM є компонент SkipLink – посилання «Перейти до основного вмісту», що веде до елемента <main id="main-content">. У стандартному стані посилання позиційно приховане за межами екрана, проте при отриманні фокусу через клавішу Tab стає видимим у верхній частині сторінки. Динамічні повідомлення системи реалізовано через компонент LiveRegion з атрибутом aria-live. Для повідомлень що не потребують негайної уваги

застосовується значення polite, для повідомлень про помилки – assertive у поєднанні з role="alert" [5].

Оскільки React Router не виконує перезавантаження сторінки при навігації між маршрутами, фокус клавіатури залишається на попередньому елементі після переходу. Для усунення цієї проблеми реалізовано хук useFocusManagement, що відстежує зміну location.pathname і програмно переміщує фокус на перший <h1> у межах <main> нової сторінки [4]. Для модальних вікон реалізовано компонент FocusTrap, що перехоплює події Tab і Shift+Tab та циклічно переміщує фокус лише між фокусовними елементами діалогу до його закриття, як того вимагає критерій 2.1.2 стандарту WCAG 2.1 [2].

Інтерактивні елементи інтерфейсу містять WAI-ARIA атрибути відповідно до їх функціонального призначення. Кнопка мобільного меню та кнопка панелі доступності містять атрибути aria-expanded і aria-controls, що передають екранному зчитувачу поточний стан розкритого або згорнутого елемента. Перемикач контрасту реалізовано як <input type="checkbox"> з атрибутом role="switch". Компонент керування розміром шрифту містить елемент з атрибутом role="progressbar" та значеннями aria-valuenow, aria-valuemin і aria-valuemax. Вибір між стандартним шрифтом та OpenDyslexic реалізований як radio-група з атрибутом role="group" та пов'язаним заголовком через aria-labelledby [5].

3.2. Реалізація модуля синтезу мовлення та підтримки екранних зчитувачів

Модуль синтезу мовлення реалізовано за дворівневою схемою резервування. Основним рівнем є ElevenLabs API з моделлю eleven_multilingual_v2, що забезпечує коректне озвучення українськомовного тексту. Резервним рівнем слугує браузерний Web Speech API, що активується автоматично у разі відсутності API-ключа або недоступності сервісу. Така архітектура гарантує працездатність функції озвучення незалежно від наявності зовнішнього сервісу [28].

Запити до ElevenLabs виконуються виключно через серверний проксі – маршрут POST /api/tts на Node.js сервері. Це рішення унеможливлює передачу API-ключа на клієнт, оскільки ключ зберігається у змінній середовища ELEVENLABS_API_KEY на сервері і жодного разу не потрапляє у відповідь браузеру. Перелік маршрутів TTS-підсистеми наведено у таблиці 3.2.

Таблиця 3.2 – Маршрути TTS-підсистеми

Метод	Маршрут	Доступ	Опис
GET	/api/tts/voices	Авторизований	Отримання списку доступних голосів з акаунту ElevenLabs
POST	/api/tts	Авторизований	Синтез мовлення з поверненням аудіопотоку у форматі MP3

Серверний проксі реалізує механізм автоматичного визначення ідентифікатора голосу через функцію resolveVoiceId. При першому зверненні функція виконує запит до /v1/voices ElevenLabs API, обирає перший доступний голос з акаунту та кешує його ідентифікатор у змінній cachedVoiceId на рівні модуля. При наступних зверненнях запит до API не виконується – використовується кешоване значення. Якщо у змінній середовища визначено ELEVENLABS_VOICE_ID, функція повертає його без звернення до API, що дозволяє зафіксувати конкретний голос. Відповідь від ElevenLabs передається клієнту у вигляді бінарного MP3-потоку через рекурсивну функцію pump, що зчитує чанки через getReader() та записує їх у відповідь без буферизації в пам'яті сервера [28]:

```
const pump = async () => {
  const { done, value } = await reader.read();
  if (done) { res.end(); return; }
  res.write(Buffer.from(value));
  return pump();
};
await pump();
```

Відповідь від ElevenLabs передається клієнту у вигляді бінарного MP3-потоку без буферизації в пам'яті сервера. Стрімінг реалізовано через рекурсивну

функцію pump, що зчитує чанки через getReader() та записує їх у відповідь безпосередньо:

```
res.set('Content-Type', 'audio/mpeg');
res.set('Cache-Control', 'no-store');

const reader = response.body.getReader();
const pump = async () => {
  const { done, value } = await reader.read();
  if (done) { res.end(); return; }
  res.write(Buffer.from(value));
  return pump();
};
await pump();
```

Перед формуванням запиту до ElevenLabs сервер валідує вхідні дані: перевіряє наявність та тип поля text, а також обмежує довжину тексту до 2500 символів. При відсутності API-ключа повертається статус 503, що є сигналом для клієнта перемкнутись на резервний Web Speech API:

```
const { text } = req.body;
if (!text || typeof text !== 'string' || !text.trim())
  return res.status(400).json({ error: 'text is required' });
if (text.length > 2500)
  return res.status(400).json({ error: 'text too long (max
2500 chars)' });

const apiKey = getApiKey();
if (!apiKey) return res.status(503).json({ error: 'TTS not
configured' });
```

На клієнті запит до серверного проксі виконується через ttsApi.synthesize. Параметр responseType: 'arraybuffer' забезпечує коректне отримання бінарних даних без спроби JSON-парсингу. Отриманий буфер перетворюється на об'єкт Blob типу audio/mpeg, після чого через URL.createObjectURL створюється тимчасова URL для відтворення:

```
synthesize: async (text, token) => {
  const response = await axios.post(
    '/api/tts',
    { text },
    {
```

```
          headers: { Authorization: `Bearer ${token}` },
          responseType: 'arraybuffer',
        }
      );
      const blob = new Blob([response.data], { type: 'audio/mpeg'
});
      return URL.createObjectURL(blob);
    },
```

Після завершення відтворення URL знищується через URL.revokeObjectURL для звільнення пам'яті браузера. Взаємодія користувача з модулем TTS реалізована через глобальний обробник події mouseup в AccessibilityContext. При увімкненому озвученні будь-який текст, виділений користувачем на сторінці, автоматично передається на синтез без необхідності натискати окремі кнопки [28].

Компонент TTSControl відображає індикатор стану ElevenLabs та надає можливість обрати резервний голос зі списку Web Speech API. Перемикач озвучення реалізовано як <input type="checkbox"> з атрибутом role="switch":

```
<label className="toggle-switch" htmlFor="tts-toggle">
  <input
    type="checkbox"
    id="tts-toggle"
    role="switch"
    checked={ttsEnabled}
    onChange={(e) => setTtsEnabled(e.target.checked)}
    aria-label="Озвучування тексту"
  />
  <span className="a11y-control-label">
    Озвучування {ttsEnabled ? '(увімкнено)' : '(вимкнено)'}
  </span>
</label>
```

Підтримка екранних зчитувачів у компонентах тестування реалізована через семантичні ролі групування варіантів відповідей. Для питань з одиночним вибором список варіантів отримує атрибут role="radiogroup", для питань з множинним вибором – role="group". Після відображення результатів тесту кожен варіант відповіді отримує атрибут aria-describedby, що пов'язує поле введення з елементом результату. При фокусуванні на варіанті відповіді екранний зчитувач озвучує і текст варіанту, і його результат як єдине повідомлення [5]:

```
      <ul
        role={question.question_type === 'multiple' ? 'group' :
'radiogroup'}
        aria-labelledby={`question-${question.id}-text`}
      >

      <input
        type={inputType}
        id={inputId}
        checked={selected}
        disabled={disabled}
        aria-describedby={showResult ? `result-${option.id}` :
undefined}
      />
      {showResult && selected && isCorrect && (
        <span id={`result-${option.id}`} aria-label="Правильна
відповідь">✓</span>
      )}
      {showResult && selected && !isCorrect && (
        <span id={`result-${option.id}`} aria-label="Неправильна
відповідь">✗</span>
      )}
      {showResult && !selected && isCorrect && (
        <span id={`result-${option.id}`} aria-label="Пропущена
правильна відповідь">✓*</span>
      )}
```

Модальні вікна системи реалізовано з повним набором атрибутів доступності. Атрибут aria-modal="true" повідомляє допоміжним технологіям що фоновий контент є неактивним. Іконка закриття має атрибут aria-hidden="true", оскільки функціональне призначення кнопки передається через aria-label [5]:

```
      <FocusTrap>
        <div
          role="dialog"
          aria-modal="true"
          aria-labelledby="modal-title"
        >
          <h2 id="modal-title">{title}</h2>
          <button aria-label="Закрити діалог" onClick={onClose}>
            <span aria-hidden="true">&times;</span>
          </button>
          <div className="modal-body">{children}</div>
        </div>
      </FocusTrap>
```

Компонент FocusTrap утримує фокус клавіатури всередині модального вікна через перехоплення подій Tab і Shift+Tab. При монтажі компонента фокус

автоматично переміщується на перший фокусовний елемент діалогу. При натисканні Tab на останньому елементі фокус циклічно повертається на перший, і навпаки для Shift+Tab [4]:

```
    const FOCUSABLE_SELECTORS = [
      'a[href]', 'button:not([disabled])',
'input:not([disabled])',
      'select:not([disabled])', 'textarea:not([disabled])',
      '[tabindex]:not([tabindex="-1"])', 'details > summary',
    ].join(', ');

    useEffect(() => {
      const getFocusable = () =>
        Array.from(trap.querySelectorAll(FOCUSABLE_SELECTORS));
      getFocusable()[0]?.focus();

      const handleKeyDown = (e) => {
        if (e.key !== 'Tab') return;
        const focusable = getFocusable();
        const first = focusable[0];
        const last  = focusable[focusable.length - 1];
        if (e.shiftKey && document.activeElement === first) {
          e.preventDefault(); last.focus();
        } else if (!e.shiftKey && document.activeElement === last)
{
          e.preventDefault(); first.focus();
        }
      };
      trap.addEventListener('keydown', handleKeyDown);
      return () => trap.removeEventListener('keydown',
handleKeyDown);
    }, []);
```

Динамічні повідомлення системи озвучуються екранним зчитувачем через компонент LiveRegion. Компонент розміщено поза межами видимої області через клас sr-only, проте залишається у DOM – це є необхідною умовою для коректної роботи aria-live регіонів, оскільки елементи приховані через display: none ігноруються допоміжними технологіями. Атрибут aria-atomic="true" забезпечує озвучення всього вмісту блоку при кожній зміні [27]:

```
    export function LiveRegion({ message, politeness = 'polite',
atomic = true }) {
      return (
        <div
          aria-live={politeness}
```

```
        aria-atomic={atomic}
        className="sr-only"
        role={politeness === 'assertive' ? 'alert' : 'status'}
      >
        {message}
      </div>
    );
  }
```

Для підтвердження збереження профілю та інших інформаційних повідомлень застосовується politeness="polite" у поєднанні з role="status" – екранний зчитувач озвучить повідомлення після завершення поточного. Для повідомлень про помилки форм застосовується politeness="assertive" у поєднанні з role="alert", що перериває поточне озвучення та негайно інформує користувача [27].

3.3. Тестування системи на відповідність стандарту WCAG 2.1

Тестування розробленої системи на відповідність стандарту WCAG 2.1 рівня AA проводилось методом ad-hoc тестування – перевірка виконувалась безпосередньо в процесі розробки без попередньо складеного формального тест-плану. Такий підхід дозволяє виявляти та усувати порушення доступності на ранніх етапах реалізації, не накопичуючи технічний борг [29]. Для автоматизованої перевірки застосовувались два інструменти: бібліотека @axe-core/react, інтегрована безпосередньо у процес розробки, та веб-розширення WAVE (Web Accessibility Evaluation Tool) для перевірки розгорнутих сторінок у браузері.

Бібліотека @axe-core/react підключалась у точці входу застосунку в режимі розробки та автоматично аналізувала DOM після кожного рендерингу компонентів, виводячи повідомлення про порушення WCAG безпосередньо у консоль браузера. Це забезпечувало негайний зворотний зв'язок при розробці нових компонентів – порушення виявлялись до моменту ручного тестування [29]. Розширення WAVE застосовувалось для візуальної перевірки готових сторінок: інструмент накладає анотації безпосередньо на сторінку, позначаючи

елементи з помилками, попередженнями та структурними особливостями, що дозволяє локалізувати проблеми у контексті реального інтерфейсу [30].

Ручне тестування охоплювало навігацію засобами лише клавіатури на всіх чотирнадцяти сторінках системи, перевірку коректності роботи skip link, видимості індикатора фокусу та поведінки focus trap у модальних вікнах. Окремо перевірялась коректність озвучення інтерфейсу – послідовність оголошень при навігації між сторінками, озвучення динамічних повідомлень через aria-live регіони та озвучення результатів тестових питань через aria-describedby.

Загальний вигляд системи у стандартному режимі відображення наведено на рисунках 3.2–3.3. Сторінка входу містить форму з коректно пов'язаними мітками для полів електронної пошти та паролю, що відповідає критерію 1.3.1 стандарту WCAG 2.1 (рисунок 3.2). Сторінка списку тем реалізована з семантичною структурою заголовків та навігаційними елементами з атрибутом aria-current="page" для позначення активного розділу (рисунок 3.3).

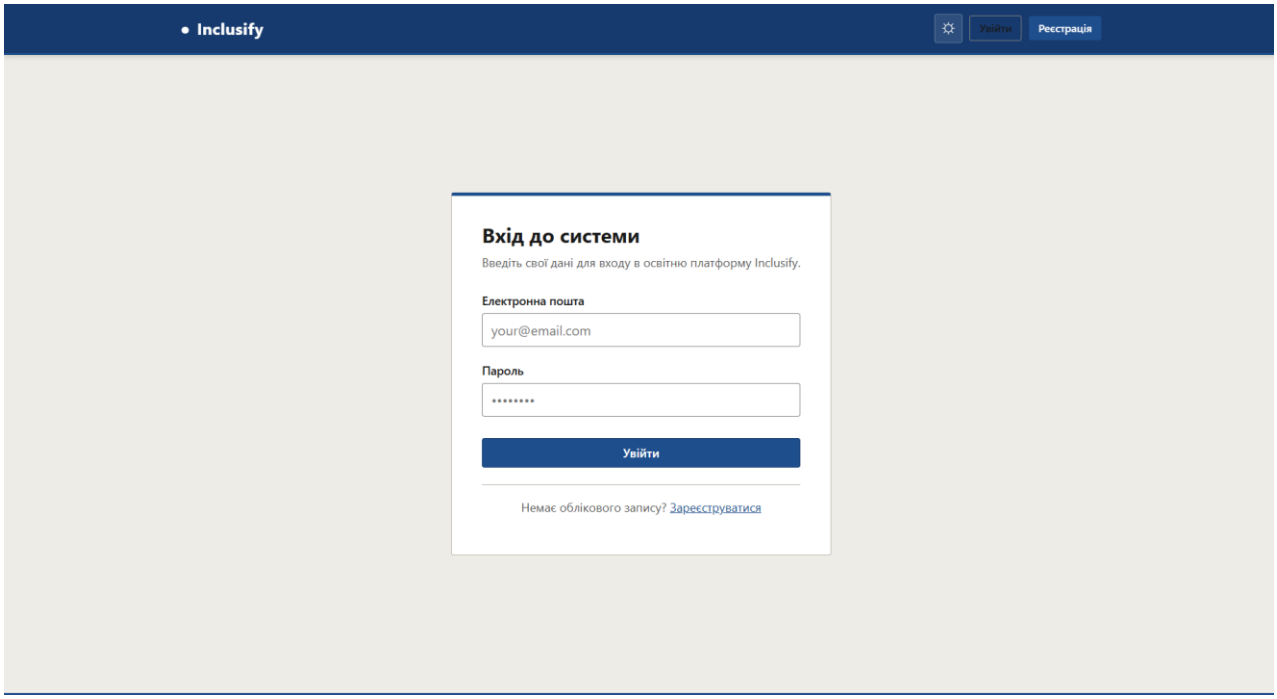

Рисунок 3.2 – Сторінка входу до системи

Рисунок 3.3 – Сторінка списку тем у стандартному режимі

Перевірка висококонтрастного режиму підтвердила відповідність критерію 1.4.3 стандарту WCAG 2.1 щодо контрастності тексту. У

висококонтрастному режимі коефіцієнт контрастності тексту відносно фону перевищує 21:1 – при чорному фоні #000000 та жовтому акцентному кольорі #ffff00 (рисунок 3.4). Сторінка деталі теми у стандартному та висококонтрастному режимах наведена на рисунках 3.5–3.6 відповідно – зміна колірної схеми не порушує структуру сторінки та читабельність контенту.

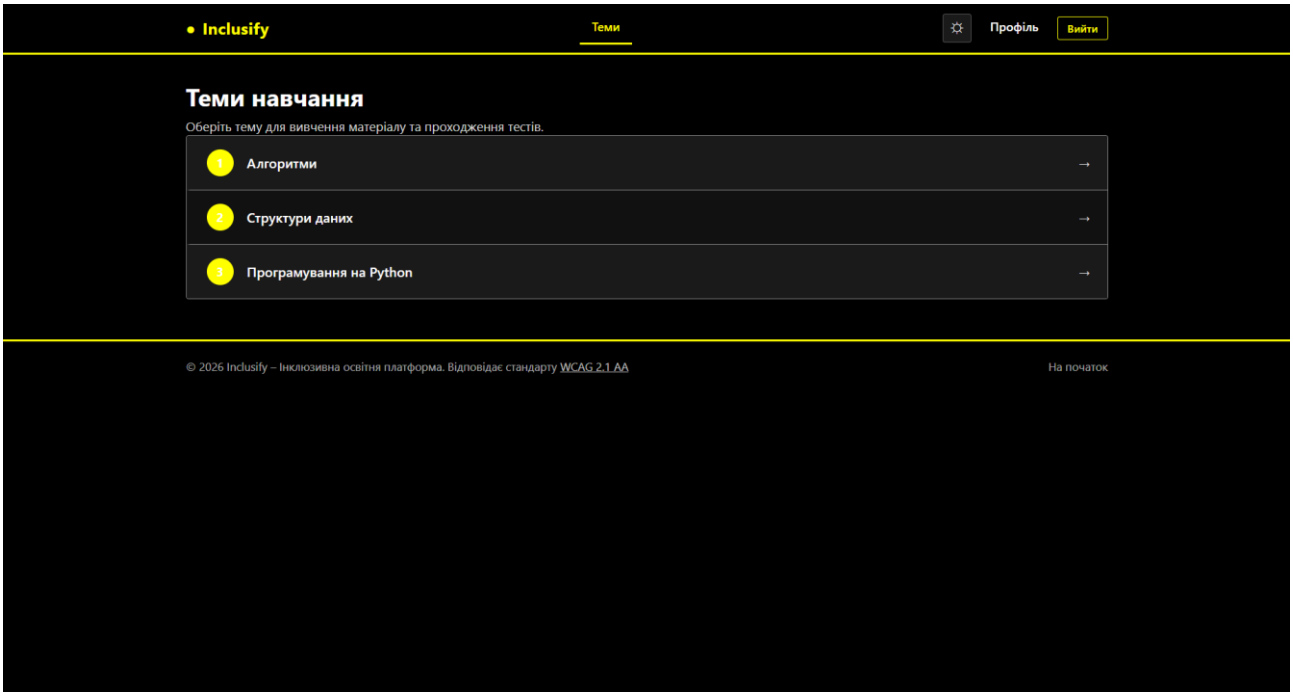

Рисунок 3.4 – Сторінка списку тем у висококонтрастному режимі

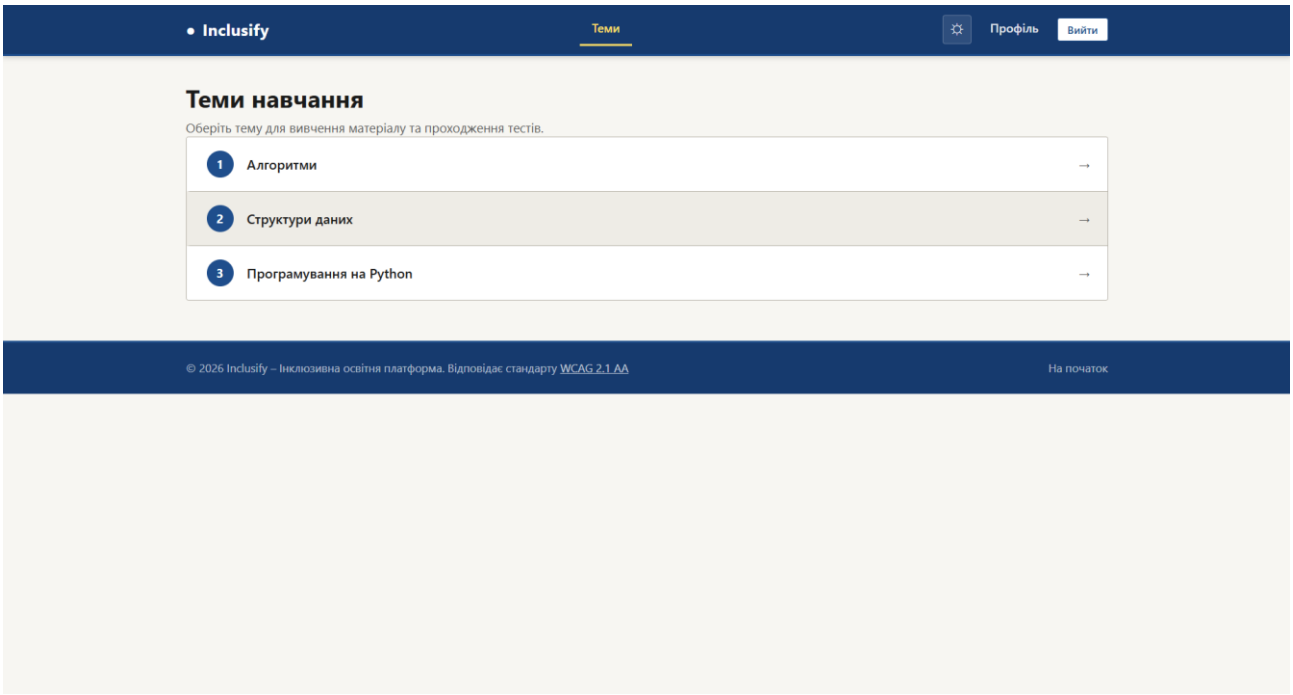

Рисунок 3.5 – Сторінка деталі теми у стандартному режимі

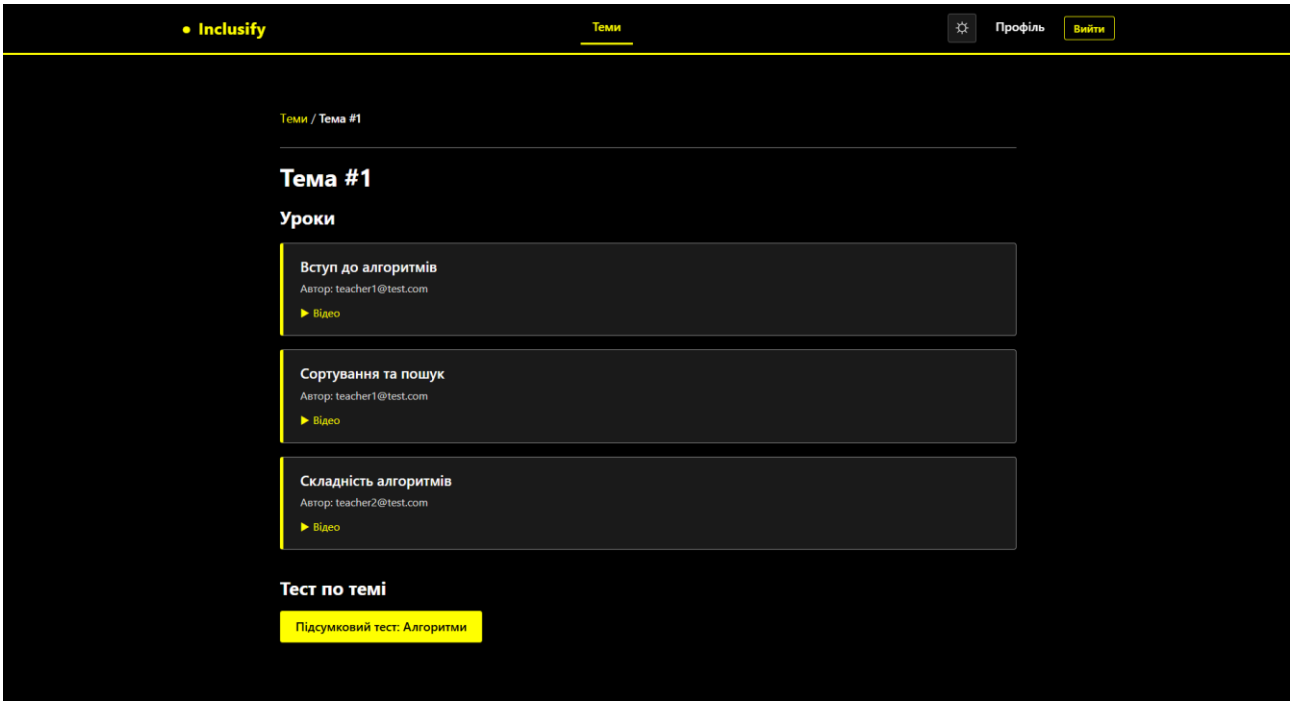

Рисунок 3.6 – Сторінка деталі теми у висококонтрастному режимі

Панель налаштувань доступності наведена на рисунку 3.7. Перевірка підтвердила коректне озвучення всіх елементів керування екранним зчитувачем: перемикачі контрасту та озвучення оголошуються як двопозиційні перемикачі з поточним станом, регулятор розміру шрифту – як елемент з поточним, мінімальним та максимальним значеннями, radio-група вибору шрифту – як група з двома варіантами. Індикатор стану ElevenLabs відображає підтвердження активності сервісу та мову озвучення.

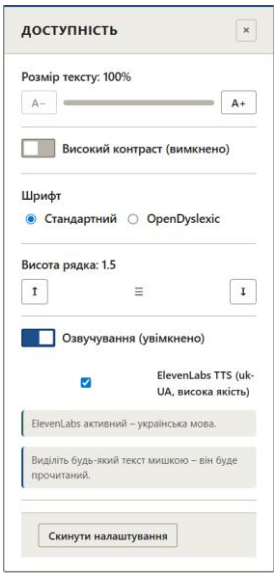

Рисунок 3.7 – Панель налаштувань доступності з увімкненим ElevenLabs TTS

Перевірка масштабування шрифту підтвердила відповідність критерію 1.4.4 – збільшення розміру тексту до 150% не порушує структуру сторінки та не викликає горизонтального прокручування (рисунок 3.8). При збільшенні міжрядкового інтервалу до значення 3 контент залишається читабельним і структурованим (рисунок 3.9), що відповідає критерію 1.4.12 стандарту WCAG 2.1 щодо адаптації текстового простору.

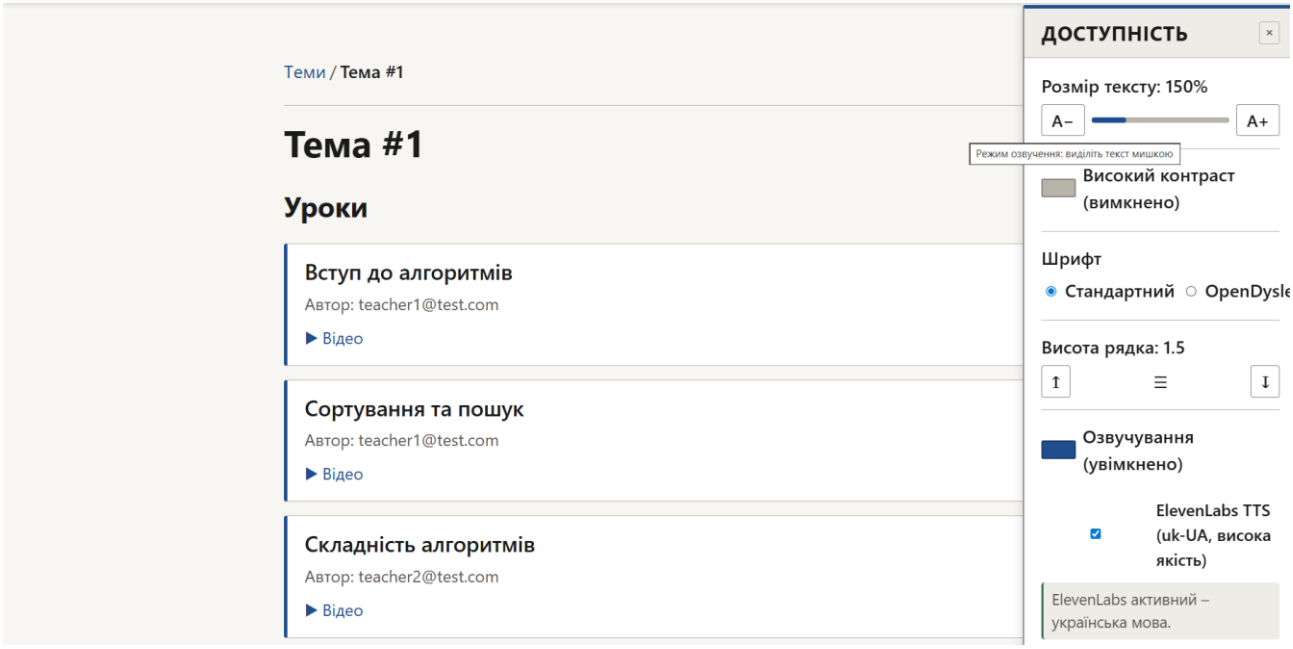

Рисунок 3.8 – Панель доступності з масштабуванням шрифту 150%

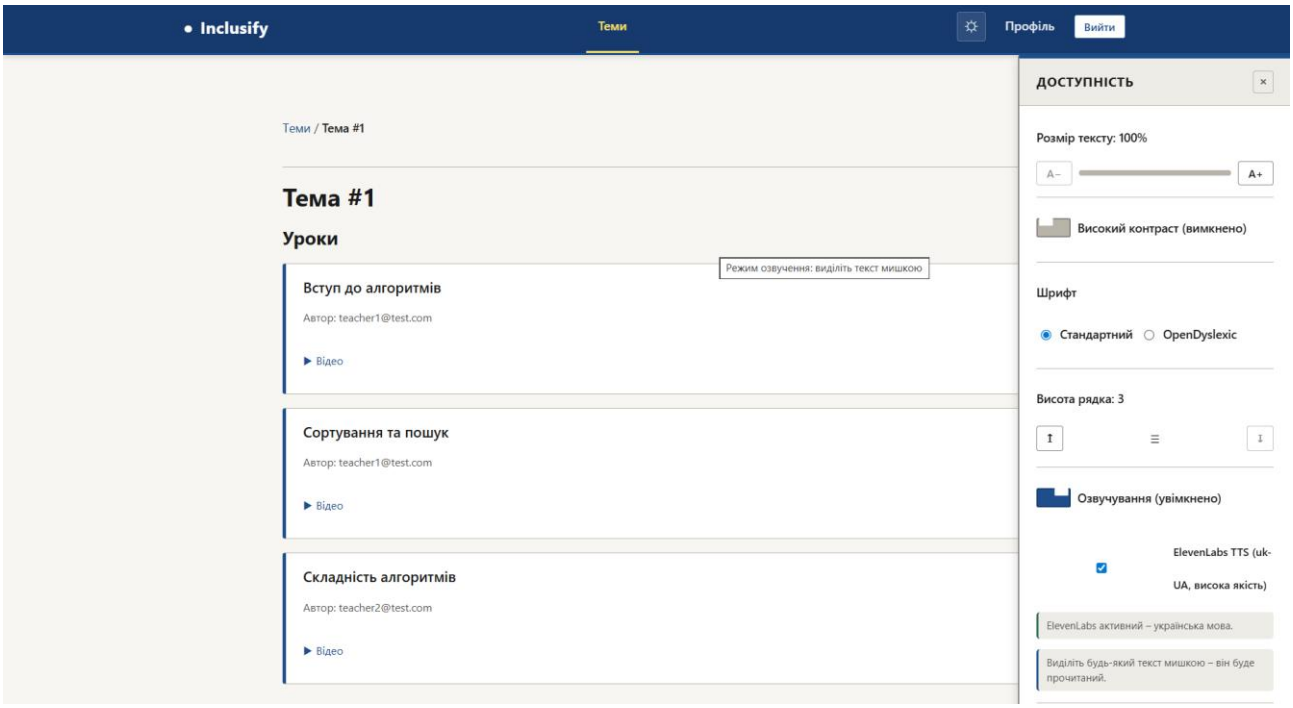

Рисунок 3.9 – Сторінка з збільшеним міжрядковим інтервалом

Сторінка уроку містить вбудоване відео YouTube з атрибутом title, текстову стенограму у розкривному блоці <details> та навчальний матеріал з семантичною розміткою (рисунок 3.10). Розкривний блок стенограми доступний з клавіатури через елемент <summary> та коректно озвучується екранним зчитувачем. Навчальний контент містить блоки коду з семантичною розміткою <pre><code>, що забезпечує коректне відображення та озвучення програмного коду (рисунок 3.11).

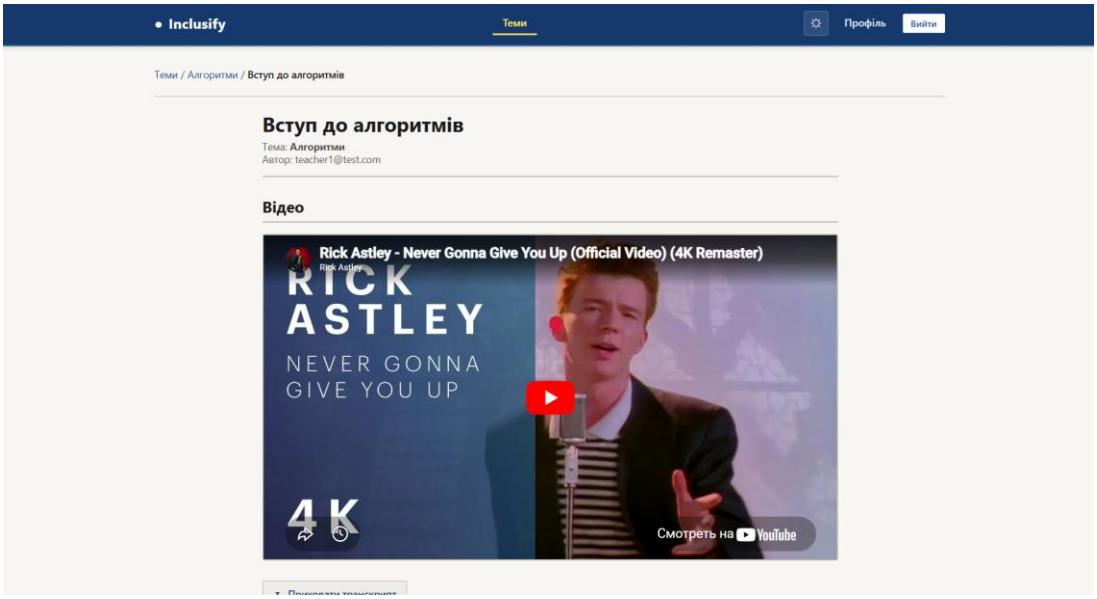

Рисунок 3.10 – Сторінка уроку з вбудованим відео YouTube

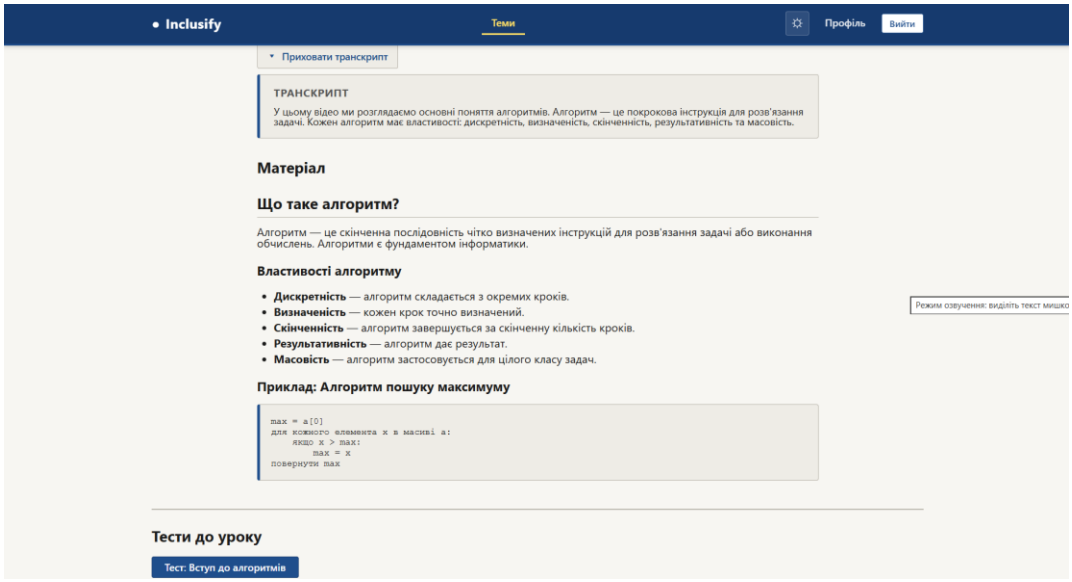

Рисунок 3.11 – Сторінка уроку з розкритою стенограмою та навчальним матеріалом

Тестування підсистеми тестування підтвердило коректну реалізацію семантичних ролей питань та варіантів відповідей. Після завершення тесту сторінка результатів відображає отриманий бал, прогрес-бар та перелік питань з позначенням правильних та неправильних відповідей (рисунок 3.12). У висококонтрастному режимі сторінка результатів зберігає повну функціональність – правильні відповіді позначені зеленим кольором, що в умовах чорного фону забезпечує достатній рівень контрастності.

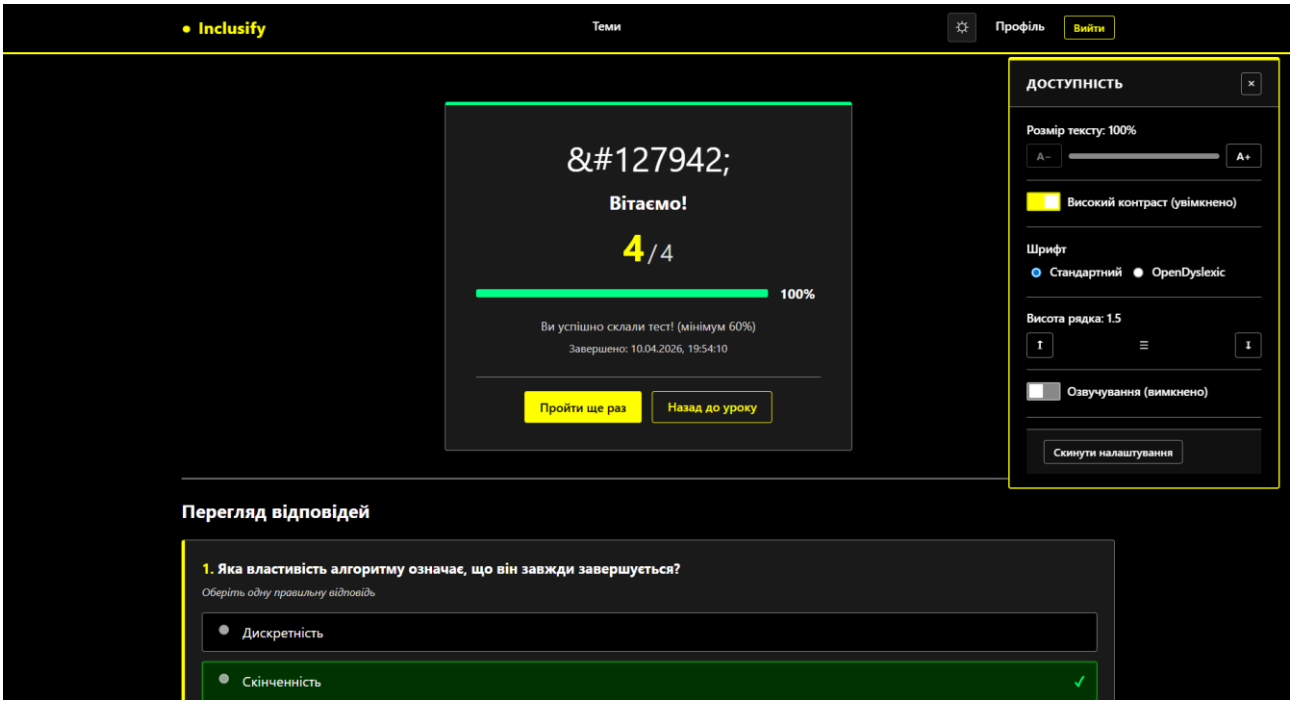

Рисунок 3.12 – Сторінка результатів тесту у висококонтрастному режимі

Кабінет вчителя надає доступ до перегляду результатів усіх студентів у табличному вигляді (рисунок 3.13). Таблиця результатів реалізована з заголовками стовпців через елементи <th scope="col">, що забезпечує коректне озвучення даних екранним зчитувачем – при навігації по клітинках таблиці зчитувач оголошує значення разом з назвою відповідного стовпця.

Зведені результати тестування системи на відповідність ключовим критеріям стандарту WCAG 2.1 рівня AA наведено у таблиці 3.3.

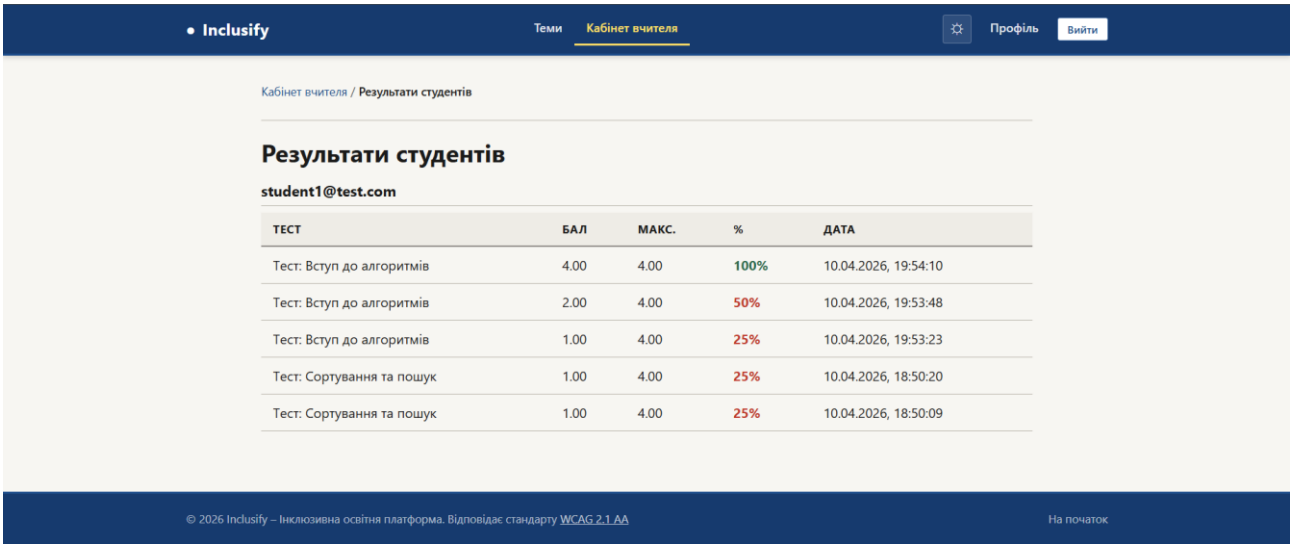

Рисунок 3.13 – Сторінка результатів студентів у кабінеті вчителя

Таблиця 3.3 – Результати тестування на відповідність WCAG 2.1 рівня AA

Критерій WCAG 2.1	Опис	Результат
1.1.1 Non-text Content	Текстові альтернативи для нетекстових об'єктів	Виконано
1.3.1 Info and Relationships	Семантична структура та ієрархія заголовків	Виконано
1.4.3 Contrast (Minimum)	Контрастність тексту ≥ 4,5:1	Виконано
1.4.4 Resize Text	Масштабування до 200% без втрати контенту	Виконано
1.4.10 Reflow	Без горизонтального прокручування при 400%	Виконано
1.4.11 Non-text Contrast	Контрастність елементів інтерфейсу ≥ 3:1	Виконано
1.4.12 Text Spacing	Адаптація при збільшенні міжрядкового інтервалу	Виконано
2.1.1 Keyboard	Повна навігація засобами клавіатури	Виконано
2.1.2 No Keyboard Trap	Відсутність пасток фокусу поза модальними вікнами	Виконано
2.4.1 Bypass Blocks	Skip link для обходу навігації	Виконано
2.4.3 Focus Order	Логічний порядок фокусу	Виконано
2.4.7 Focus Visible	Видимий індикатор фокусу	Виконано
3.1.1 Language of Page	Мова документа визначена програмно	Виконано
4.1.2 Name, Role, Value	ARIA-атрибути для всіх інтерактивних елементів	Виконано
4.1.3 Status Messages	Динамічні повідомлення через aria-live	Виконано

Результати тестування підтверджують відповідність розробленої системи вимогам стандарту WCAG 2.1 рівня AA у повному обсязі критеріїв, що були визначені як цільові у підрозділі 2.1.

Висновки до розділу 3

Програмна реалізація та тестування веб-сайту з інформатики для учнів з інклюзією підтвердили технічну здійсненність усіх вимог, сформульованих у розділі 2, та відповідність розробленої системи стандарту WCAG 2.1 рівня AA.

Реалізація інтерфейсу користувача на базі React.js забезпечила послідовне застосування вимог доступності на рівні компонентної архітектури. Структура маршрутизації охоплює чотирнадцять сторінок з розмежуванням доступу через три компоненти-охоронці відповідно до ролей student, teacher та admin. Модуль доступності реалізовано через AccessibilityContext з використанням CSS custom properties, що дозволяє застосовувати зміни параметрів відображення до всього дерева компонентів без повторного рендерингу. Реалізовано висококонтрастний режим з коефіцієнтом контрастності понад 21:1, масштабування шрифту в діапазоні 100–300%, підтримку шрифту OpenDyslexic та регулювання міжрядкового інтервалу. Вимоги семантичної розмітки виконано через коректну ієрархію заголовків, landmark-ролі, skip link, focus trap у модальних вікнах та хук useFocusManagement для управління фокусом при навігації між сторінками SPA.

Модуль синтезу мовлення реалізовано за дворівневою схемою резервування: основним рівнем є ElevenLabs API з підтримкою української мови, резервним – браузерний Web Speech API. Запити до ElevenLabs виконуються через серверний проксі на Node.js, що унеможливлює передачу API-ключа на клієнт. Стрімінг MP3 реалізовано без буферизації в пам'яті сервера. Підтримку екранних зчитувачів розширено через семантичні ролі radiogroup та group для варіантів відповідей тестових питань, атрибут aria-describedby для озвучення результатів тесту, компонент LiveRegion з диференційованим використанням aria-live="polite" та aria-live="assertive" залежно від пріоритету повідомлення.

Тестування системи методом ad-hoc з використанням інструментів @axe-core/react та WAVE підтвердило відповідність п'ятнадцяти критеріям стандарту WCAG 2.1 рівня AA. Автоматизована перевірка через @axe-core/react забезпечувала виявлення порушень безпосередньо в процесі розробки, що дозволило усувати їх до етапу ручного тестування. Ручне тестування підтвердило коректність навігації засобами клавіатури, роботи skip link, видимості індикатора фокусу та функціональності всіх режимів доступності на реальних сторінках системи.

ВИСНОВКИ

У кваліфікаційній роботі спроєктовано та розроблено веб-сайт з інформатики для учнів з інклюзією, що відповідає вимогам стандарту WCAG 2.1 рівня AA та забезпечує збереження індивідуальних профілів доступності користувачів. Усі завдання, визначені у вступі, виконано в повному обсязі.

У ході виконання першого завдання проаналізовано концептуальні засади інклюзивної освіти та вимоги міжнародних стандартів веб-доступності WCAG 2.1 і WAI-ARIA. Встановлено, що доступність веб-ресурсу визначається чотирма базовими принципами – сприйнятливістю, керованістю, зрозумілістю та надійністю, – кожен з яких формує конкретні технічні вимоги до реалізації інтерфейсу. Специфікація WAI-ARIA визначена як необхідний інструментарій для забезпечення доступності динамічних односторінкових застосунків.

У ході виконання другого завдання проведено порівняльний аналіз трьох освітніх платформ – Khan Academy, Coursera та «Всеукраїнської школи онлайн» – за критеріями відповідності стандарту WCAG 2.1 рівня AA. Встановлено, що жодна з платформ не забезпечує вбудованих засобів зміни колірної схеми, масштабування шрифту та збереження індивідуальних профілів доступності, що обґрунтовує доцільність розробки спеціалізованої системи.

У ході виконання третього завдання визначено функціональні вимоги до системи на основі аналізу чотирьох категорій цільової аудиторії – користувачів з порушеннями зору, моторики, слуху та когнітивними особливостями. Обґрунтовано вибір технологічного стеку React.js, Node.js/Express.js та PostgreSQL відповідно до вимог доступності.

У ході виконання четвертого завдання спроєктовано схему бази даних з восьми таблиць, приведену до третьої нормальної форми, та описано архітектуру системи засобами п'яти UML-діаграм – компонентів, варіантів використання, двох діаграм послідовності та діаграми розгортання.

У ході виконання п'ятого завдання реалізовано веб-сайт з повним набором функцій доступності: висококонтрастний режим з коефіцієнтом контрастності понад 21:1, масштабування шрифту в діапазоні 100–300%, підтримка шрифту

OpenDyslexic, регулювання міжрядкового інтервалу, skip link, focus trap у модальних вікнах, управління фокусом при навігації SPA та модуль синтезу мовлення на базі ElevenLabs API з резервним рівнем Web Speech API.

У ході виконання шостого завдання проведено тестування системи методом ad-hoc з використанням інструментів @axe-core/react та WAVE. Результати підтвердили відповідність п'ятнадцяти критеріям стандарту WCAG 2.1 рівня AA, зокрема критеріям контрастності, клавіатурної навігації, семантичної розмітки та коректної роботи допоміжних технологій.

Перспективами подальшого розвитку системи є розширення підтримки жестової мови через інтеграцію відеосупроводу до навчальних матеріалів, впровадження автоматичної транскрипції відеоконтенту засобами ASR-сервісів, а також розширення аналітичного модуля для відстеження прогресу учнів з формуванням індивідуальних навчальних траєкторій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про освіту» від 05.09.2017 № 2145-VIII. *Відомості Верховної Ради України.* 2017. № 38–39. Ст. 380. URL: https://zakon.rada.gov.ua/laws/show/2145-19 (дата звернення: 10.03.2025).

2. Web Content Accessibility Guidelines (WCAG) 2.1 : W3C Recommendation 05 June 2018 / ed. by A. Kirkpatrick et al. W3C, 2018. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 10.03.2025).

3. Acosta-Vargas P., Acosta T., Luján-Mora S. Challenges to access the educational websites. *Proceedings of the 2018 International Conference on eDemocracy & eGovernment (ICEDEG).* Ambato, Ecuador, 2018. P. 67–74.

4. Frisbie A. Building Accessible Websites with React. *Smashing Magazine.* 2021. URL: https://www.smashingmagazine.com/2021/07/accessible-react-guide/ (дата звернення: 11.03.2025).

5. Accessible Rich Internet Applications (WAI-ARIA) 1.1 : W3C Recommendation 14 December 2017 / ed. by J. Craig, M. Cooper. W3C, 2017. URL: https://www.w3.org/TR/wai-aria-1.1/ (дата звернення: 10.03.2025).

6. Petrie H., Power C. What do end users really want from accessible web sites? *Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility.* Dundee, Scotland, 2012. P. 1–8.

7. Treviranus J. The Three Dimensions of Inclusive Design. *Inclusive Design Research Centre.* 2020. URL: https://idrc.ocadu.ca/ideas/the-three-dimensions-of-inclusive-design/ (дата звернення: 12.03.2025).

8. Закон України «Про повну загальну середню освіту» від 16.01.2020 № 463-IX. *Відомості Верховної Ради України.* 2020. № 31. Ст. 226. URL: https://zakon.rada.gov.ua/laws/show/463-20 (дата звернення: 10.03.2025).

9. CAST. Universal Design for Learning Guidelines version 2.2. Wakefield, MA : CAST, 2018. URL: https://udlguidelines.cast.org (дата звернення: 11.03.2025).

10. Henry S. L. Introduction to Web Accessibility. W3C Web Accessibility Initiative (WAI), 2022. URL: https://www.w3.org/WAI/fundamentals/accessibility-intro/ (дата звернення: 11.03.2025).

11. Campoverde-Molina M., Luján-Mora S., Valverde L. Accessibility of university websites worldwide: a systematic literature review. *Universal Access in the Information Society.* 2021. Vol. 20. P. 1–26. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC8259087/ (дата звернення: 12.03.2025).

12. Rana N. P., Dwivedi Y. K., Williams M. D. Accessibility analysis using WCAG 2.1: evidence from Indian e-government websites. *Universal Access in the Information Society.* 2022. Vol. 22. P. 663–669. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC8743096/ (дата звернення: 12.03.2025).

13. Ismailova R., Inal Y. Web site accessibility, usability and security: a survey of government web sites in Kyrgyz Republic. *Universal Access in the Information Society.* 2021. Vol. 20. P. 293–311.

14. ДСТУ EN 301 549:2022. Інформаційні технології. Вимоги щодо доступності продуктів та послуг ІКТ (EN 301 549 V3.2.1 (2021-03), IDT). Чинний від 2022-09-01. Київ : ДП «УкрНДНЦ», 2022.

15. Khan Academy. Accessibility at Khan Academy. Khan Academy Help Center, 2023. URL: https://support.khanacademy.org/hc/en-us/articles/202483630 (дата звернення: 13.03.2025).

16. Coursera. Accessibility at Coursera. Coursera Learner Help Center, 2023. URL: https://learner.coursera.help/hc/en-us/articles/209818883 (дата звернення: 13.03.2025).

17. Всеукраїнська школа онлайн : офіційний вебсайт. Міністерство освіти і науки України, 2023. URL: https://vso.org.ua (дата звернення: 13.03.2025).

18. WebAIM. Survey of Web Accessibility Problems. WebAIM, 2021. URL: https://webaim.org/projects/screenreadersurvey9/ (дата звернення: 02.04.2025).

19. React. Accessibility – React Documentation. Meta Open Source, 2023. URL: https://react.dev/reference/react-dom/components/common#aria-attributes (дата звернення: 02.04.2025).

20. Express.js. Express 4.x API Reference. OpenJS Foundation, 2023. URL: https://expressjs.com/en/4x/api.html (дата звернення: 02.04.2025).

21. PostgreSQL Global Development Group. PostgreSQL 15 Documentation. PostgreSQL, 2023. URL: https://www.postgresql.org/docs/15/index.html (дата звернення: 02.04.2025).

22. Coronel C., Morris S. Database Systems: Design, Implementation, and Management. 13th ed. Boston : Cengage Learning, 2018. 784 p.

23. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston : Addison-Wesley, 2003. 208 p.

24. Richards M., Ford N. Fundamentals of Software Architecture. Sebastopol : O'Reilly Media, 2020. 422 p.

25. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Boston : Addison-Wesley, 2021. 624 p.

26. MDN Web Docs. ARIA: aria-live attribute. Mozilla, 2024. URL: https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Attributes/aria-live (дата звернення: 02.04.2025).

27. MDN Web Docs. CSS custom properties. Mozilla, 2024. URL: https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties (дата звернення: 02.04.2025).

28. ElevenLabs. API Reference: Text to Speech. ElevenLabs Documentation, 2024. URL: https://elevenlabs.io/docs/api-reference/text-to-speech (дата звернення: 02.04.2025).

29. Deque Systems. axe-core: Accessibility testing engine. GitHub, 2023. URL: https://github.com/dequelabs/axe-core (дата звернення: 02.04.2025).

30. WebAIM. WAVE Web Accessibility Evaluation Tools. WebAIM, 2023. URL: https://wave.webaim.org (дата звернення: 02.04.2025).

ДОДАТОК

Додаток А

Скрип створення бази даних

```
-- Users table
CREATE TABLE IF NOT EXISTS users (
  id SERIAL PRIMARY KEY,
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  role VARCHAR(10) CHECK (role IN ('student', 'teacher', 'admin')) NOT NULL DEFAULT
'student',
  created_at TIMESTAMP DEFAULT NOW()
);

-- Accessibility profiles
CREATE TABLE IF NOT EXISTS accessibility_profiles (
  id SERIAL PRIMARY KEY,
  user_id INTEGER UNIQUE REFERENCES users(id) ON DELETE CASCADE,
  font_size INTEGER DEFAULT 100,
  contrast_mode BOOLEAN DEFAULT FALSE,
  font_family VARCHAR(50) DEFAULT 'standard',
  line_height NUMERIC(3,1) DEFAULT 1.5,
  tts_enabled BOOLEAN DEFAULT FALSE,
  updated_at TIMESTAMP DEFAULT NOW()
);

-- Topics
CREATE TABLE IF NOT EXISTS topics (
  id SERIAL PRIMARY KEY,
  title VARCHAR(255) NOT NULL,
  order_index INTEGER NOT NULL DEFAULT 0
);

-- Lessons
CREATE TABLE IF NOT EXISTS lessons (
  id SERIAL PRIMARY KEY,
  topic_id INTEGER REFERENCES topics(id) ON DELETE CASCADE,
  title VARCHAR(255) NOT NULL,
  content_html TEXT,
  video_url VARCHAR(500),
  transcript_text TEXT,
  created_by INTEGER REFERENCES users(id),
  created_at TIMESTAMP DEFAULT NOW()
);

-- Tests
CREATE TABLE IF NOT EXISTS tests (
  id SERIAL PRIMARY KEY,
  type VARCHAR(10) CHECK (type IN ('lesson', 'topic')) NOT NULL,
```

```
  lesson_id INTEGER REFERENCES lessons(id) ON DELETE CASCADE,
  topic_id INTEGER REFERENCES topics(id) ON DELETE CASCADE,
  title VARCHAR(255) NOT NULL,
  created_by INTEGER REFERENCES users(id),
  created_at TIMESTAMP DEFAULT NOW()
);

-- Questions
CREATE TABLE IF NOT EXISTS questions (
  id SERIAL PRIMARY KEY,
  test_id INTEGER REFERENCES tests(id) ON DELETE CASCADE,
  question_text TEXT NOT NULL,
  question_type VARCHAR(10) CHECK (question_type IN ('single', 'multiple')) NOT
NULL,
  order_index INTEGER DEFAULT 0
);

-- Answer options
CREATE TABLE IF NOT EXISTS answer_options (
  id SERIAL PRIMARY KEY,
  question_id INTEGER REFERENCES questions(id) ON DELETE CASCADE,
  option_text TEXT NOT NULL,
  is_correct BOOLEAN DEFAULT FALSE
);

-- Test results
CREATE TABLE IF NOT EXISTS test_results (
  id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
  test_id INTEGER REFERENCES tests(id) ON DELETE CASCADE,
  score NUMERIC(5,2) NOT NULL,
  max_score NUMERIC(5,2) NOT NULL,
  passed_at TIMESTAMP DEFAULT NOW()
);
```

Додаток Б

Фрагменти програмного коду системи

Лістинг Б.1 – AccessibilityContext.jsx – контекст керування параметрами доступності

```
import React, { createContext, useState, useEffect, useCallback, useRef } from
'react';
import { ttsApi } from '../api/ttsApi';

export const AccessibilityContext = createContext(null);

const DEFAULT_SETTINGS = {
  fontSize: 100,
  contrastMode: false,
  fontFamily: 'standard',
  lineHeight: 1.5,
  ttsEnabled: false,
  ttsVoiceName: '',       // Web Speech API voice name (fallback)
  ttsUseGoogle: true,     // prefer Google Cloud TTS when available
};

function playBlobUrl(url) {
  const audio = new Audio(url);
  audio.play().catch(() => {});
  audio.addEventListener('ended', () => URL.revokeObjectURL(url), { once: true });
  return audio;
}

export function AccessibilityProvider({ children }) {
  const [settings, setSettings] = useState(() => {
    try {
      const saved = localStorage.getItem('inclusify_accessibility');
      return saved ? { ...DEFAULT_SETTINGS, ...JSON.parse(saved) } :
DEFAULT_SETTINGS;
    } catch {
      return DEFAULT_SETTINGS;
    }
  });

  const [googleTtsAvailable, setGoogleTtsAvailable] = useState(null);
  const currentAudioRef = useRef(null);

  useEffect(() => {
    const root = document.documentElement;
    root.style.setProperty('--font-size-multiplier', settings.fontSize / 100);
    root.style.setProperty('--line-height', settings.lineHeight);

    document.body.classList.toggle('high-contrast', settings.contrastMode);
```

```
    document.body.classList.toggle('font-opendyslexic', settings.fontFamily ===
'opendyslexic');

    try {
      localStorage.setItem('inclusify_accessibility', JSON.stringify(settings));
    } catch {}
  }, [settings]);

  const updateSettings = useCallback((updates) => {
    setSettings(prev => ({ ...prev, ...updates }));
  }, []);

  const resetSettings = useCallback(() => setSettings(DEFAULT_SETTINGS), []);
  const setFontSize = useCallback((size) => updateSettings({ fontSize:
Math.max(100, Math.min(300, size)) }), [updateSettings]);
  const setContrastMode = useCallback((v) => updateSettings({ contrastMode:
Boolean(v) }), [updateSettings]);
  const setFontFamily = useCallback((f) => { if (['standard',
'opendyslexic'].includes(f)) updateSettings({ fontFamily: f }); },
[updateSettings]);
  const setLineHeight = useCallback((h) => updateSettings({ lineHeight:
Math.max(1.0, Math.min(3.0, parseFloat(h))) }), [updateSettings]);
  const setTtsEnabled = useCallback((v) => {
    updateSettings({ ttsEnabled: Boolean(v) });
    // Reset ElevenLabs availability check so it retries on next speak
    if (v) setGoogleTtsAvailable(null);
  }, [updateSettings]);
  const setTtsVoice = useCallback((name) => updateSettings({ ttsVoiceName: name }),
[updateSettings]);
  const setTtsUseGoogle = useCallback((v) => updateSettings({ ttsUseGoogle:
Boolean(v) }), [updateSettings]);

  const settingsRef = useRef(settings);
  useEffect(() => { settingsRef.current = settings; }, [settings]);

  const ttsEnabledRef = useRef(settings.ttsEnabled);
  useEffect(() => { ttsEnabledRef.current = settings.ttsEnabled; },
[settings.ttsEnabled]);

  const googleAvailableRef = useRef(googleTtsAvailable);
  useEffect(() => { googleAvailableRef.current = googleTtsAvailable; },
[googleTtsAvailable]);

  const [voices, setVoices] = useState([]);
  useEffect(() => {
    if (!window.speechSynthesis) return;
    const load = () => {
      const v = window.speechSynthesis.getVoices();
      if (v.length) setVoices(v);
    };
    load();
    window.speechSynthesis.addEventListener('voiceschanged', load);
```

```
    return () => window.speechSynthesis.removeEventListener('voiceschanged', load);
  }, []);

  const voicesRef = useRef(voices);
  useEffect(() => { voicesRef.current = voices; }, [voices]);

  const findFallbackVoice = useCallback(() => {
    const all = voicesRef.current;
    if (!all.length) return null;
    const saved = settingsRef.current.ttsVoiceName;
    if (saved) {
      const found = all.find(v => v.name === saved);
      if (found) return found;
    }
    return (
      all.find(v => v.lang === 'uk-UA') ||
      all.find(v => v.lang.startsWith('uk')) ||
      all.find(v => v.lang === 'ru-RU') ||
      all.find(v => v.lang.startsWith('ru')) ||
      all.find(v => v.default) ||
      all[0] ||
      null
    );
  }, []);

  const speakWithElevenLabs = useCallback(async (text, token) => {
    try {
      const blobUrl = await ttsApi.synthesize(text, token);
      if (currentAudioRef.current) {
        currentAudioRef.current.pause();
      }
      currentAudioRef.current = playBlobUrl(blobUrl);
      setGoogleTtsAvailable(true); // reuse flag: true = ElevenLabs works
    } catch (err) {
      const status = err?.response?.status;
      if (status === 503) {
        setGoogleTtsAvailable(false); // not configured
      }
      throw err;
    }
  }, []);

  const speakWithWebSpeech = useCallback((text) => {
    if (!window.speechSynthesis) return;
    window.speechSynthesis.cancel();
    const utterance = new SpeechSynthesisUtterance(text);
    const voice = findFallbackVoice();
    if (voice) {
      utterance.voice = voice;
      utterance.lang = voice.lang;
    } else {
      utterance.lang = 'uk-UA';
```

```
    }
    utterance.rate = 1;
    utterance.pitch = 1;
    window.speechSynthesis.speak(utterance);
  }, [findFallbackVoice]);

  const speak = useCallback((text, token) => {
    if (!ttsEnabledRef.current) return;
    const cleaned = text.trim();
    if (!cleaned) return;

    const useGoogle = settingsRef.current.ttsUseGoogle;
    const googleAvailable = googleAvailableRef.current;

    if (useGoogle && googleAvailable !== false && token) {
      speakWithElevenLabs(cleaned, token).catch(() => {
        speakWithWebSpeech(cleaned);
      });
    } else {
      speakWithWebSpeech(cleaned);
    }
  }, [speakWithElevenLabs, speakWithWebSpeech]);

  const stopSpeaking = useCallback(() => {
    if (currentAudioRef.current) {
      currentAudioRef.current.pause();
      currentAudioRef.current = null;
    }
    if (window.speechSynthesis) {
      window.speechSynthesis.cancel();
    }
  }, []);

  // token is read from localStorage so the global selection handler can attempt
Google TTS without prop drilling
  useEffect(() => {
    const handleMouseUp = () => {
      if (!ttsEnabledRef.current) return;
      const selected = window.getSelection()?.toString().trim();
      if (selected) {
        const token = localStorage.getItem('inclusify_token');
        speak(selected, token);
      }
    };
    document.addEventListener('mouseup', handleMouseUp);
    return () => document.removeEventListener('mouseup', handleMouseUp);
  }, [speak]);

  useEffect(() => {
    if (settings.ttsEnabled) {
      document.body.style.cursor = 'text';
      document.body.title = 'Режим озвучення: виділіть текст мишкою';
```

```
    } else {
      document.body.style.cursor = '';
      document.body.title = '';
      stopSpeaking();
    }
  }, [settings.ttsEnabled, stopSpeaking]);

  const value = {
    settings,
    updateSettings,
    resetSettings,
    setFontSize,
    setContrastMode,
    setFontFamily,
    setLineHeight,
    setTtsEnabled,
    setTtsVoice,
    setTtsUseGoogle,
    speak,
    stopSpeaking,
    ttsVoices: voices,
    fallbackVoice: findFallbackVoice(),
    googleTtsAvailable,
  };

  return (
    <AccessibilityContext.Provider value={value}>
      {children}
    </AccessibilityContext.Provider>
  );
}
```

Лістинг Б.2 – server/routes/tts.js – серверний проксі синтезу мовлення

```
const express = require('express');
const router = express.Router();
const authMiddleware = require('../middleware/authMiddleware');

const ELEVENLABS_BASE = 'https://api.elevenlabs.io/v1';
const ELEVENLABS_TTS  = `${ELEVENLABS_BASE}/text-to-speech`;

const getApiKey = () => process.env.ELEVENLABS_API_KEY;

// GET /api/tts/voices
router.get('/voices', authMiddleware, async (req, res) => {
  const apiKey = getApiKey();
  if (!apiKey) return res.status(503).json({ error: 'TTS not configured' });

  try {
    const response = await fetch(`${ELEVENLABS_BASE}/voices`, {
      headers: { 'xi-api-key': apiKey },
```

```
    });
    const data = await response.json();
    const voices = (data.voices || []).map(v => ({
      voice_id: v.voice_id,
      name: v.name,
      labels: v.labels,
    }));
    return res.json({ voices });
  } catch (err) {
    console.error('ElevenLabs voices error:', err);
    return res.status(500).json({ error: 'Failed to fetch voices' });
  }
});

let cachedVoiceId = null;

const resolveVoiceId = async (apiKey) => {
  if (process.env.ELEVENLABS_VOICE_ID) return process.env.ELEVENLABS_VOICE_ID;
  if (cachedVoiceId) return cachedVoiceId;

  const r = await fetch(`${ELEVENLABS_BASE}/voices`, {
    headers: { 'xi-api-key': apiKey },
  });

  if (!r.ok) {
    console.error('ElevenLabs /voices failed:', r.status);
    return null;
  }

  const data = await r.json();
  console.log('ElevenLabs voices available:', (data.voices || []).map(v =>
`${v.name} (${v.voice_id})`));

  const first = data.voices?.[0]?.voice_id;
  if (first) cachedVoiceId = first;
  return first || null;
};

// POST /api/tts
router.post('/', authMiddleware, async (req, res) => {
  const apiKey = getApiKey();

  if (!apiKey) {
    return res.status(503).json({ error: 'TTS not configured' });
  }

  const { text } = req.body;

  if (!text || typeof text !== 'string' || !text.trim()) {
    return res.status(400).json({ error: 'text is required' });
  }
```

```
if (text.length > 2500) {
  return res.status(400).json({ error: 'text too long (max 2500 chars)' });
}

let voiceId;
try {
  voiceId = await resolveVoiceId(apiKey);
} catch (err) {
  console.error('resolveVoiceId error:', err);
  return res.status(503).json({ error: 'Failed to resolve voice ID' });
}

if (!voiceId) {
  console.error('No voice ID found for API key');
  return res.status(503).json({ error: 'No voices in ElevenLabs account' });
}

console.log(`TTS: using voice ${voiceId}, text length: ${text.length}`);

try {
  const response = await fetch(`${ELEVENLABS_TTS}/${voiceId}`, {
    method: 'POST',
    headers: {
      'Accept': 'audio/mpeg',
      'xi-api-key': apiKey,
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      text: text.trim(),
      model_id: 'eleven_multilingual_v2',
      voice_settings: { stability: 0.5, similarity_boost: 0.75 },
    }),
  });

  if (!response.ok) {
    const err = await response.json().catch(() => ({}));
    console.error('ElevenLabs TTS error:', response.status, err);
    return res.status(502).json({
      error: 'ElevenLabs request failed',
      details: err?.detail?.message || String(response.status),
    });
  }

  res.set('Content-Type', 'audio/mpeg');
  res.set('Cache-Control', 'no-store');

  const reader = response.body.getReader();
  const pump = async () => {
    const { done, value } = await reader.read();
    if (done) { res.end(); return; }
    res.write(Buffer.from(value));
    return pump();
```

```
    };
    await pump();

  } catch (err) {
    console.error('TTS proxy error:', err);
    if (!res.headersSent) res.status(500).json({ error: 'Internal server error' });
  }
});

module.exports = router;
```

Лістинг Б.3 – FocusTrap.jsx – утримання фокусу у модальних вікнах

```
import React, { useEffect, useRef } from 'react';

const FOCUSABLE_SELECTORS = [
  'a[href]',
  'button:not([disabled])',
  'input:not([disabled])',
  'select:not([disabled])',
  'textarea:not([disabled])',
  '[tabindex]:not([tabindex="-1"])',
  'details > summary',
].join(', ');

export function FocusTrap({ children }) {
  const trapRef = useRef(null);

  useEffect(() => {
    const trap = trapRef.current;
    if (!trap) return;

    const getFocusable = () =>
Array.from(trap.querySelectorAll(FOCUSABLE_SELECTORS));

    const focusable = getFocusable();
    if (focusable.length > 0) {
      focusable[0].focus();
    }

    const handleKeyDown = (e) => {
      if (e.key !== 'Tab') return;

      const focusable = getFocusable();
      if (focusable.length === 0) return;

      const first = focusable[0];
      const last = focusable[focusable.length - 1];

      if (e.shiftKey) {
        if (document.activeElement === first) {
```

```
          e.preventDefault();
          last.focus();
        }
      } else {
        if (document.activeElement === last) {
          e.preventDefault();
          first.focus();
        }
      }
    };

    trap.addEventListener('keydown', handleKeyDown);
    return () => trap.removeEventListener('keydown', handleKeyDown);
  }, []);

  return (
    <div ref={trapRef}>
      {children}
    </div>
  );
}
```

Лістинг Б.4 – useFocusManagement.js – керування фокусом при навігації SPA

```
import { useEffect, useRef } from 'react';
import { useLocation } from 'react-router-dom';

export function useFocusManagement() {
  const location = useLocation();
  const prevPathRef = useRef(location.pathname);

  useEffect(() => {
    if (prevPathRef.current !== location.pathname) {
      prevPathRef.current = location.pathname;
      const timer = setTimeout(() => {
        const h1 = document.querySelector('main h1');
        if (h1) {
          h1.setAttribute('tabindex', '-1');
          h1.focus();
          h1.addEventListener('blur', () => h1.removeAttribute('tabindex'), { once:
true });
        }
      }, 100);
      return () => clearTimeout(timer);
    }
  }, [location.pathname]);
}
```

Лістинг Б.5 – LiveRegion.jsx – динамічні повідомлення для екранних зчитувачів

```jsx
import React from 'react';

export function LiveRegion({ message, politeness = 'polite', atomic = true }) {
  return (
    <div
      aria-live={politeness}
      aria-atomic={atomic}
      className="sr-only"
      role={politeness === 'assertive' ? 'alert' : 'status'}
    >
      {message}
    </div>
  );
}
```