МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ.С.З.ГЖИЦЬКОГО
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему:

«РОЗРОБКА ТА ОПТИМІЗАЦІЯ УНІВЕРСАЛЬНИХ ВЕБ-ІНТЕРФЕЙСІВ З ВИКОРИСТАННЯМ БІБЛІОТЕКИ REACT»

Виконав: студент групи Іт-41
спеціальності 126 Інформаційні системи
та технології Демський А.В.
(прізвище та ініціали)

Керівник: Желєзняк А.М.
(прізвище та ініціали)

ДУБЛЯНИ 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Рівень вищої освіти перший (бакалаврський)
Спеціальність 126 Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
(підпис)
д.т.н., професор, Тригуба А. М.
(вч. звання, прізвище, ініціали)
“____” _____202__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Демського Андріяна Володимировича
(прізвище, ім’я, по батькові)

1. Тема роботи *«Розробка та оптимізація універсальних веб-інтерфейсів з використанням бібліотеки React»*
керівник роботи *к. е н., доцент., Желєзняк А.М.*
(наук.ступінь, вч. звання, прізвище, ініціали)

затверджені наказом Львівського НУП *№ 123 /к-с від 25.02.2025 р*

2. Строк подання студентом роботи 09.06.2025 р.
3. Вихідні дані: *характеристика прикладної сфери; вихідні дані та вимоги до веб-інтерфейсу, опис бібліотек мов програмування, програмна конфігурація веб-інтерфейсу; науково-технічна і довідкова література.*

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
Вступ
1. Теоретичні основи розробки веб-інтерфейсу
2. Проектування веб-інтерфейсу
3.Розробка та оптимізація продуктивності веб-інтерфейсу
4. Охорона праці та безпека в надзвичаних ситуаціях
Висновки
Бібліографічний список
5. Перелік графічного матеріалу
Графічний матеріал подається у вигляді презентації

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3	*Желєзняк А.М., к.е.н., доцент кафедри інформаційних технологій*			
4	*Городецький І.М., к.т.н., доцент кафедри інженерної механіки*			

7. Дата видачі завдання 01.11.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	*Отримання завдання. Вивчення рекомендованої літератури по темі роботи. Написання аналітичного огляду предметної області.*	*01.11.2024 – 15.12.2024*	
2	*Проектування та опис технічного завдання, функціональних вимог та технічної сторони реалізації проекту (написання проектної частини).*	*16.12.2024 – 05.02.2024*	
3	*Програмна реалізація проекту (вибір мови програмування, розробка сайту, тестування його роботи, розрахунок ефективності проекту)*	*06.02.2025 – 31.03.2025*	
4	*Розгляд питань з охорони праці та безпеки у надзвичайних ситуаціях*	*01.04.2025 – 30.04.2025*	
5	*Завершення оформлення основної частини, написання висновків та підготовка презентаційного матеріалу*	*01.05.2024 – 26.05.2024*	
6	*Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи*	*27.05.2024 – 12.06.2024*	

Студент _____ *Демський А.В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Желєзняк А.М.*
(підпис) (прізвище та ініціали)

УДК 004.738.5:004.43

Кваліфікаційна робота: 51 сторінка текстової частини, 10 таблиць, 11 рисунків, 21 джерело літератури, 6 додатків.

«Розробка та оптимізація універсальних веб-інтерфейсів з використанням бібліотеки React» Демський А.В – Кваліфікаційна робота. Кафедра інформаційних технологій. Дубляни, Львівський національний університет ветеринарної медицини та біотехнологій ім.С.З.Гжицького, 2025 р.

Розглянуті сучасні тенденції у веб-розробці та підходи у створенні веб-інтерфейсів. Проаналізовано предметну область, існуючі аналоги та визначено функціональні вимоги до веб-інтерфейсів. Здійснено проектування та розробка веб-інтерфейсу з використанням бібліотеки React.

Здійснено аналіз травматичних ситуацій при виконанні різних робіт у сфері використання комп’ютерної техніки та інформаційних технологій, розглянуто питання охорони праці та безпеки в надзвичайних ситуаціях.

ВЕБ-ІНТЕРФЕЙС, ВЕБ-ТЕХНОЛОГІЇ, REACT, ОПТИМІЗАЦІЯ

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку інтернет-ресурсів та інформаційних технологій особливе значення набуває створення ефективних, адаптивних та зручних для користувача веб-інтерфейсів. Веб-застосунки активно використовуються як в комерційній, так і соціальній сфері, обумовлюючи потребу в універсальних підходах та рішеннях, що однаково добре функціонуватимуть на різних пристроях.

Сучасні користувачі очікують високої швидкості завантаження, інтуїтивної навігації, зручного дизайну та доступності інтерфейсу незалежно від платформи або типу пристрою. Для досягнення таких результатів розробникам необхідно використовувати сучасні фреймворки та бібліотеки, які забезпечують масштабованість, гнучкість і продуктивність. Одним із найбільш популярних рішень є бібліотека React, що дозволяє будувати компонувальні інтерфейси та реалізовувати підхід односторінкових додатків (SPA).

Актуальність теми зумовлена необхідністю впровадження ефективних підходів до створення універсальних веб-інтерфейсів, які відповідають сучасним вимогам дизайну, продуктивності та кросплатформенності. React, завдяки своїй гнучкій архітектурі, широкому екосередовищу та підтримці інструментів для оптимізації продуктивності, надає розробникам широкі можливості для реалізації таких задач.

Для практичної реалізації проекту було обрано прикладну сферу, а саме розробку зручного, адаптивного, кросплатформенного веб-інтерфейсу для онлайн-курсів із першої домедичної допомоги та тактичної медицини з використанням бібліотеки React.

Сучасні виклики, пов’язані з військовими діями, надзвичайними ситуаціями та загрозами життю і здоров’ю людей, зумовлюють зростання попиту на швидке та якісне навчання з домедичної допомоги й тактичної медицини. Онлайн-курси з цієї тематики мають бути зручними, наочними,

інтуїтивними й доступними з будь-якого пристрою — як для цивільного населення, так і для військових або волонтерів.

Мета роботи полягає у розробці та оптимізації універсального веб-інтерфейсу з використанням бібліотеки React для онлайн-курсів із першої домедичної допомоги та тактичної медицини, що адаптується під різні типи пристроїв та забезпечує високу швидкість і зручність взаємодії користувача з системою.

Об’єктом дослідження є процес розробки та оптимізації інтерфейсів веб-застосунків. Предмет дослідження — технології створення універсальних інтерфейсів з використанням бібліотеки React.

Тема кваліфікаційної роботи є актуальною, оскільки вимоги до інтерфейсів сайтів та вебзастосунків постійно зростають.

В ході виконання кваліфікаційної роботи були поставлені наступні завдання:

1. Проаналізувати специфіку онлайн-курсів у сфері домедичної допомоги та визначити особливості подання навчального матеріалу.
2. Оцінити вимоги до дизайну, юзабіліті, адаптивності та мультимедійної підтримки у вебінтерфейсах освітнього типу.
3. Обґрунтувати вибір бібліотеки React як технологічної основи розробки, а також супутніх інструментів.
4. Спроєктувати структуру інтерфейсу курсу: навігацію, розділи (лекції, відео, симуляції, тести, сертифікація), особистий кабінет користувача.
5. Реалізувати адаптивний вебінтерфейс, що забезпечує зручну взаємодію користувача з платформою з різних типів пристроїв.
6. Розглянути методи оптимізації швидкодії, такі як динамічне завантаження компонентів, кешування, оптимізація рендерингу.
7. Провести тестування вебінтерфейсу для аналізу продуктивності.
8. Оцінити потенціал подальшого розширення проєкту — впровадження багатомовності, офлайн-доступу, інтеграції з мобільними додатками або чатботами.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБ-ІНТЕРФЕЙСУ

1.1. Сучасні тенденції веб-розробки

Сфера веб-розробки є однією з найдинамічніших у галузі інформаційних технологій, виходячи із значної кількості інтернет-ресурсів, розширенням сфер застосування вебзастосунків, зростанням якості та швидкості інтернету.

Із зростанням вимог до якості взаємодії користувачів із вебзастосунками розробники використовують все більший набір технологій, застосовуючи нові підходи з метою забезпечення високої швидкості, адаптивності, зручності та доступність інтерфейсів. Особливо це важливо для освітніх рішень у сфері охорони здоров’я, зокрема онлайн-курсів із першої домедичної допомоги та тактичної медицини, які мають бути максимально ефективними навіть у складних умовах використання. Загалом під вебсторінкою розуміють інформаційний ресурс, доступний в інтернет-мережі, який користувач може переглянути в веббраузері.

На сьогодні виділяють такі сучасні підходи у веброзробці, як адаптивність та мобільність, застосування односторінкового підходу в архітектурі вебзастосунку, компонентний підхід з використанням фреймворків React, Angular і Vue.js, оптимізація продуктивності вебзастосунку, реалізація принципів доступності тощо.

Однією з ключових вимог до сучасних вебзастосунків є підтримка мобільних пристроїв. За статистикою, понад 60% користувачів взаємодіють з вебресурсами саме з телефонів. У випадку створення онлайн-ресурсу для навчання із першої домедичної допомоги та тактичної медицини цей відсоток може бути вищим, оскільки доступ до інструкцій може здійснюватися в полі, на чергуванні або під час тренування. Відповідно, сучасні підходи включають адаптивну верстку (responsive design) та прогресивні вебзастосунки (PWA), які дозволяють використовувати вебзастосунок як мобільний додаток.

Прогресивні веб-додатки (PWA) схожі на звичайні рішення для розробки мобільних додатків, але вони оновлені сучасними веб-технологіями, забезпечуючи при цьому більш схожий на веб-додаток досвід роботи та взаємодії із користувачем [1]. Ключовими характеристиками прогресивних веб-додатків є адаптивність, встановлення на пристрій, незалежність у підключенні, видимість, зовнішній вигляд у формі додатку, кросплатформеність.

На рисунку 1.1. показано, що прогресивні веб-додатки PWA можуть забезпечити як видимість веб-додатку, так і потужність нативного додатку, поєднуючи сильні сторони обох типів.

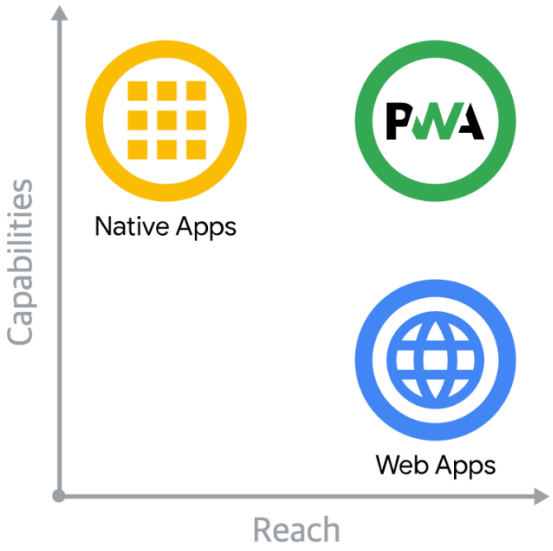


Рисунок 1.1. – Взаємозв’язок між характеристиками різного типу додатків [2]

Додатки типу PWA використовують технологію Service Workers та веб-платформу, яка не залежить від операційної системи Android або iOS і може працювати на обох платформах одночасно.

Сучасна веб-розробка активно впроваджує та застосовує принципи доступності, у відповідності до яких інтерфейс має бути зручним для користувачів із різними фізичними обмеженнями: порушенням зору, моторики або когнітивних функцій.

У сфері навчання принципам домедичної допомоги це важливо, адже потенційні користувачі вебзастосунку можуть перебувати в стресових або фізично обмежених умовах. Використання ARIA-атрибутів, контрастного дизайну, зручної навігації — це не просто тренд, а обов'язкова вимога для вебзастосунків такого типу.

ARIA (Accessible Rich Internet Applications) становить собою набір атрибутів, які додаються до класичної HTML-розмітки, щоб зробити веб-контент і вебдодатки більш доступними для людей з обмеженими можливостями.

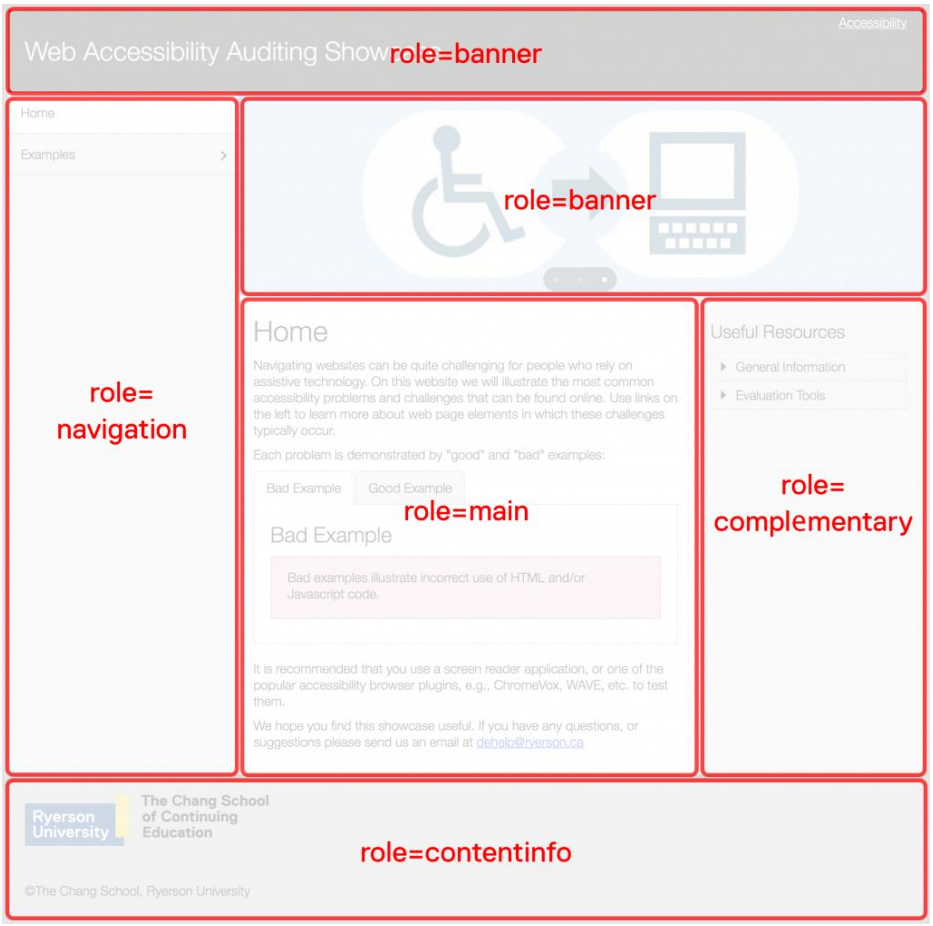


Рисунок 1.2. – Приклад застосування ролей ARIA для основних компонентів вебсторінки [8]

ARIA розширює можливості доступності, особливо для динамічного контенту, створеного за допомогою JavaScript та Ajax. Загалом світова спільнота та Україна підтримують впровадження підходів у веб-розробці у

відповідності стандартів вебдоступності Web Content Accessibility Guidelines (WCAG) 2.1 [7].

У відповідності до документації WCAG рівень вебдоступності визначається чотирма категоріями: A (мінімальний), AA (середній), AAA (розширений) і AAA+ (найвищий). Для важливих вебзастосунків, до яких належать онлайн-ресурси для онлайн-курсів із першої домедичної допомоги та тактичної медицини, доцільно дотримуватись рівня AA, щоб забезпечити базовий рівень доступності для користувачів онлайн-ресурсу.

Створення вебінтерфейсів, особливо навчальних, вимагає продуманих користувацьких сценаріїв, простих шляхів до ключової інформації, ієрархії візуального контенту, акцентів на діях. Популярними стали дизайн-системи (наприклад, Material UI, Tailwind UI), що стандартизують елементи інтерфейсу, забезпечують узгодженість і прискорюють розробку.

1.2 Основні поняття та підходи у розробці веб-інтерфейсів

У сфері інформаційних технологій під інтерфейсом програмного забезпечення часто розуміють набір засобів, правил та методів, що дозволяють взаємодіяти з системою або її частинами. У контексті сучасної веб-розробки поняття універсального інтерфейсу охоплює сукупність властивостей і рішень, що забезпечують зручну, ефективну та доступну взаємодію користувача з вебзастосунком незалежно від типу пристрою, операційної системи або роздільної здатності екрана. Такий інтерфейс має бути інтуїтивно зрозумілим, швидким, адаптивним до змін і візуально узгодженим.

Проектування інтерфейсів вебзастосунків передбачає реалізацію адаптивного або респонсивного підходів у дизайні, а також можливості реалізації кросплатформеного підходу.

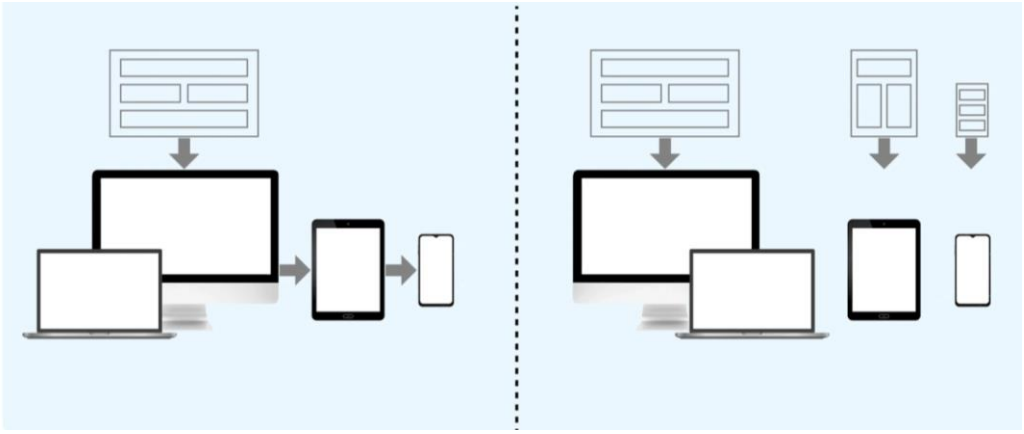
Під адаптивним інтерфейсом (adaptive design) розуміють підхід до розробки, за якого застосунок визначає тип пристрою або роздільну здатність екрану та відображає відповідну версію дизайну (наприклад, окремий шаблон

для мобільної чи десктопної версії). Цей метод дає змогу більш точно налаштувати зовнішній вигляд під специфіку пристрою, але вимагає більше ресурсів на розробку.

Таблиця 1.1. – Приклади інтерфейсу у сфері інформаційних технологій

Поняття	Зміст та трактування
Інтерфейс користувача (UI)	Спосіб, за допомогою якого користувач взаємодіє з програмним забезпеченням або пристроєм.
Інтерфейс веб-сайту	Все, що користувач бачить на веб-сторінці (меню, кнопки, форми, текст, зображення).
Інтерфейс мобільного додатка	Елементи управління, навігація, візуальний дизайн, який використовується в додатках на смартфонах
Інтерфейс програмного забезпечення	Вікна, меню, панелі інструментів, кнопки, які використовуються в комп'ютерних програмах
Програмний інтерфейс (API)	Це набір правил та специфікацій, що дозволяє різним програмним компонентам взаємодіяти між собою
Інтерфейс підключення (CI)	Це стандартний роз'єм, що використовується в сучасних телевізорах, цифрових тюнерах та комп'ютерах для підключення CAM-модулів (Conditional Access Module) з картою доступу до платного цифрового телебачення.
Універсальний інтерфейс у програмуванні	В мовах програмування, наприклад, Java, інтерфейс може бути набором абстрактних методів, які реалізуються іншими класами

Респонсивний дизайн (responsive design) натомість використовує гнучкі макети сторінок у вигляді сіток, медіа-запитів та одиниці вимірювання, що дозволяють одному інтерфейсу автоматично підлаштовуватися під різні екрани без створення окремих шаблонів. Це знижує витрати на розробку та обслуговування, проте іноді вимагає компромісів у функціональності вебзастосунків.



а) б)

Рисунок 1.3. – Приклад респонсивного (а) та адаптивного (б) підходів у дизайні веб-інтерфейсів [4]

Реалізація кросплатформеного інтерфейсу у веброзробці у означає, що користувачі можуть взаємодіяти з вебзастосунком із різних платформ (Windows, macOS, Android, iOS тощо) без зниження якості досвіду. Досягається реалізація кросплатформеного інтерфейсу шляхом використання єдиної бази коду, універсальних веб-стандартів (HTML5, CSS3, JavaScript) і бібліотек, таких як React, які дозволяють будувати динамічні та ефективні інтерфейси з повторним використанням компонентів.

Загалом можна виділити такі ключові вимоги до універсального інтерфейсу вебзастосунків, як:

- адаптивність (адаптація до різних екранів),
- доступність (врахування потреб людей з інвалідністю),
- юзабіліті (інтуїтивна зрозумілість),
- продуктивність (швидке завантаження і реакція на дії),
- узгодженість стилю і взаємодії на різних пристроях.

Прикладами універсальних вебінтерфейсів є сучасні платформи на кшталт Google Docs, GitHub, Notion — ці застосунки підтримують і десктопні, і мобільні версії з високою швидкістю і зручністю використання.

Отже, створення універсального веб-інтерфейсу — це не лише технічне завдання, а комплексний дизайн-процес, що враховує потреби широкого кола

користувачів. Застосування технологій, таких як React, сприяє побудові саме таких рішень завдяки своїй компонентній природі, повторному використанню коду та можливостям оптимізації.

1.3. Огляд і аналіз існуючих аналогів та постановка задачі

Сфера першої домедичної допомоги та тактичної медицини (ПДМД) є критично важливою складовою національної безпеки, громадського здоров'я та особистої безпеки громадян. У часи збройних конфліктів, терористичних загроз, масових аварій або стихійних лих навички домедичної допомоги можуть стати вирішальними для збереження людського життя ще до прибуття професійних медиків. Не менш важливо отримати ці навички і студентам та майбутнім фахівцям різних галузей економіки у формі самостійного навчання, неформальної освіти або ж інформаційного ресурсу під час вивчення дисциплін «Безпека життєдіяльності та охорони праці», «Фізичне виховання та основи захисту України», «Базова військова підготовка».

Перша домедична допомога передбачає надання елементарних реанімаційних або стабілізуючих заходів у перші хвилини після травми, кровотечі, зупинки серця чи дихання, шоку тощо. Вона може надаватися цивільними особами або волонтерами, які пройшли відповідне навчання.

Тактична медицина (також Tactical Combat Casualty Care, TCCC) розвинулася як відповідь на необхідність медичної допомоги в умовах бойових дій. Вона охоплює методи зупинки кровотечі, евакуацію поранених, підтримку дихання та свідомості в умовах обмеженого доступу до медичної інфраструктури. Тактична медицина застосовується не лише військовими, а й підрозділами поліції, ДСНС, добровольцями, медиками на фронті.

У мирному та воєнному середовищі підготовка населення до надання першої допомоги стала пріоритетною задачею для громадських організацій, благодійних фондів, освітніх ініціатив, закладів вищої освіти, МОЗ України та міжнародних партнерів.

Створення тематичного онлайн-ресурсу може вирішити наступні завдання:

- ✓ перегляд навчальних матеріалів, конспектів лекцій, наочних ілюстрацій та інших рекомендацій;
- ✓ проходження навчальних модулів з першої домедичної допомоги та тактичної медицини в зручному форматі;
- ✓ перегляд відеоінструкції та чеклисти;
- ✓ перевірка знань за допомогою тестів;
- ✓ отримання сертифікатів про проходження курсу.

Вебінтерфейс такої системи має бути максимально простим, адаптивним, швидким та зручним, щоби навіть у стресовій ситуації користувач міг швидко знайти потрібну інструкцію або повторити критичні дії. Зважаючи на потенційний широкий цільовий сегмент користувачів — від цивільного населення до військових, волонтерів, студентів закладів вищої освіти — система має підтримувати адаптивний дизайн, українську та англійську мови, а також бути оптимізованою під мобільні пристрої.

В ході виконання кваліфікаційної роботи були проаналізовані існуючі аналоги, а саме інтернет-ресурси, які надають навчальну інформацію про першу домедичної допомоги та тактичну медицину.

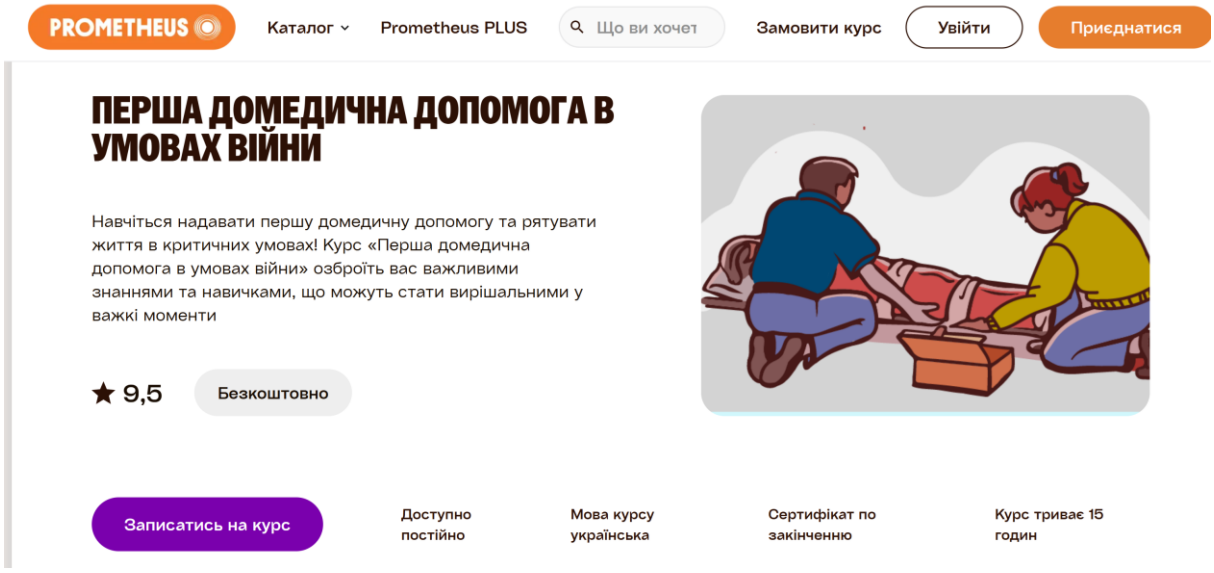


Рисунок 1.4. – Фрагмент вікна онлайн-курсу «Перша медична допомога в умовах війни» на навчальній платформі Prometheus

Одним із найбільш популярних в Україні онлайн-ресурсів для навчання основам домедичної допомоги в умовах війни є відповідний безкоштовний онлайн-курс на популярній навчальній платформі Prometheus (рис.1.4).

Онлайн-курс передбачає розміщення тематичних відео у відповідності до 10 уроків, для чого на сайті сформовані окремі блоки у формі випадаючого списку (рис.1.5).

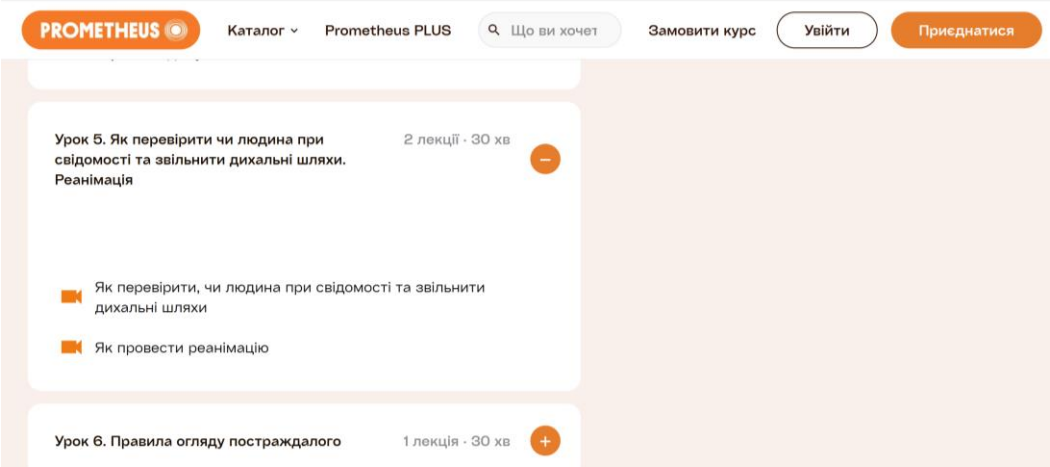


Рисунок 1.5 – Фрагмент веб-інтерфейсу вікна онлайн-курсу у розділі відео-уроків

Однією з переваг даного інтерфейсу є те, що користувач може повертатися до того чи іншого відео-уроку і при потребі його знову переглядати. Із недоліків можна відмітити, що навчальна платформа Prometheus містить різноманітні курси, які розраховані на широке коло користувачів, тому досить слабо реалізований індивідуальний підхід, вимоги по дизайну з доступності тощо.

Тематичні статті з першої домедичної допомоги часто розіщені окремими рубриками на інтернет-ресурсах провідних державних установ, компаній, органів державної влади, тематичних сайтах з безпеки життєдіяльності та охорони праці.

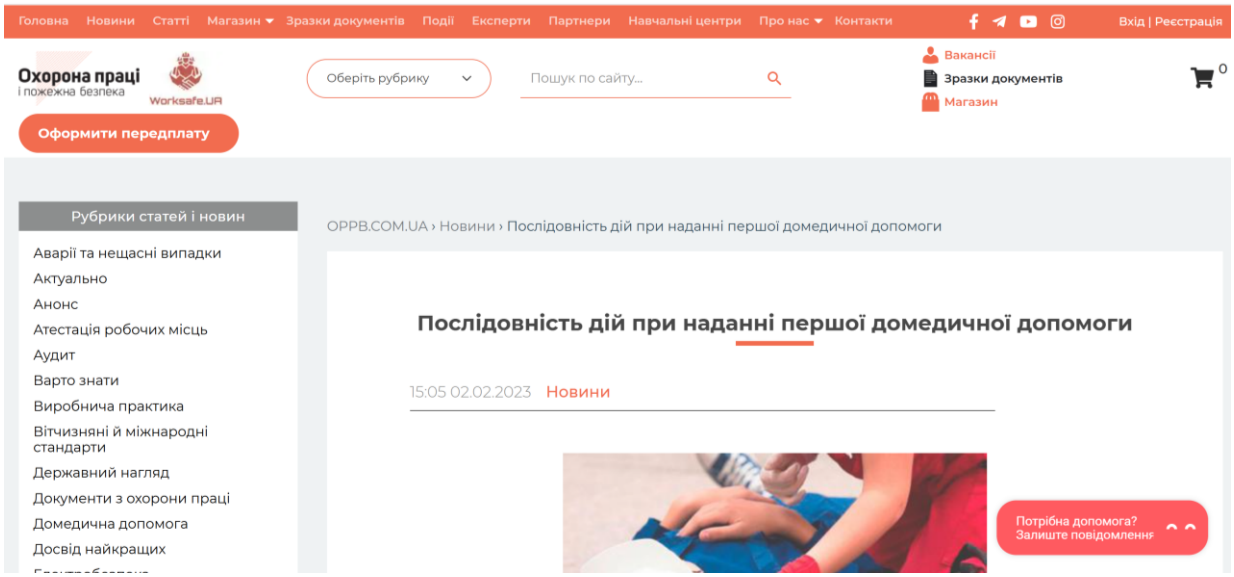


Рисунок 1.6 – Фрагмент інтерфейсу онлайн-ресурсу «Охорона праці і пожежна безпека»

Недоліком таких інтернет-ресурсів є перевантаженість їх різноманітною інформацією, відсутність можливостей для індивідуального навчання, тестування та сертифікації.

Матеріали з тактичної медицини часто розміщені на тематичних сайтах, наприклад сил територіальної оборони (рис. 1.7).

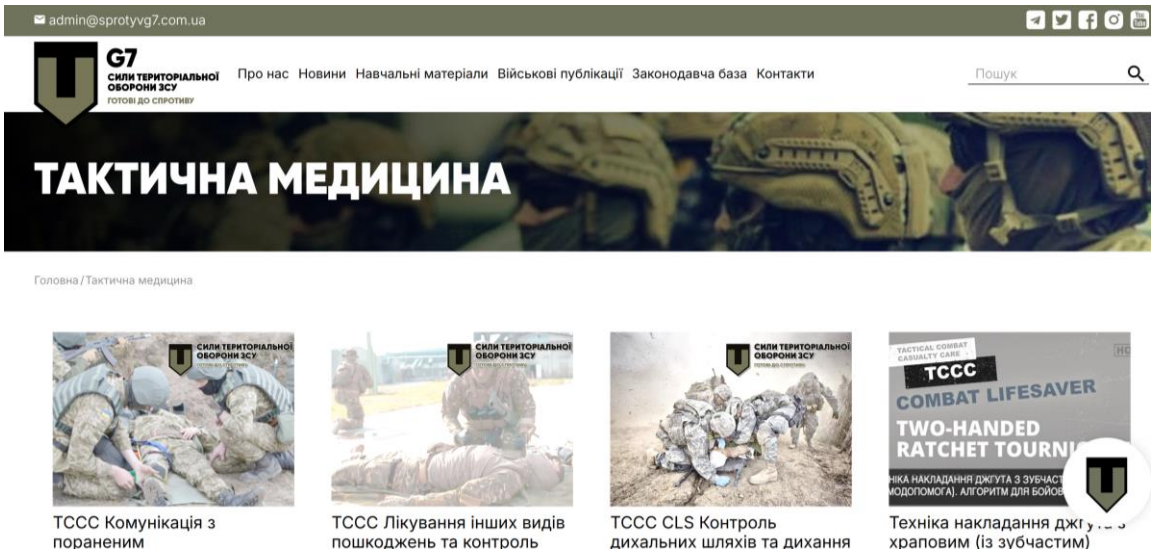


Рисунок 1.7 – Фрагмент веб-інтерфейсу онлайн-ресурсу із навчальними матеріалами з тактичної медицини (https://sprotyvg7.com.ua/lessons/taktichna-medicina)

Використання професійних та апробованих підходів і навчальних матеріалів із тактичної медицини у поєднанні з навчальним контентом з надання першої медичної допомоги може бути ефективним поєднанням для проектування та розробки тематичного вебдодатку для окремих категорій цивільного населення, зокрема студентів, які проходять навчання в закладах вищої освіти.

Процес створення вебінтерфейсу навчального застосунку для підготовки з першої домедичної допомоги та тактичної медицини передбачає комплекс дій від аналізу потреб користувачів до розгортання готового продукту. Враховуючи, що першими користувачами платформи виступатимуть студенти закладів вищої освіти, важливо забезпечити інтуїтивність, швидкодію та чітку структуру подання матеріалів.

Етапи розробки веб-інтерфейсу вебдодатку у відповідності до тематики кваліфікаційної роботи та прикладної сфери включатимуть:

1) Аналіз цільової аудиторії та формування функціональний і нефункціональних вимог проекту.

2) Проєктування структури застосунку (навігації, сторінок, компонентів).

3) Створення прототипів і дизайну інтерфейсу (UI/UX, макети).

4) Розгортання SPA-застосунку на React.

5) Реалізація основних компонентів: лекцій, відео, тестів, особистого кабінету, навігаційне меню.

6) Реалізація адаптивної верстки для мобільних і десктопних пристроїв.

7) Інтеграція з API та оптимізація роботи вебдодатку.

8) Забезпечення доступності вебдодатку (ARIA, клавіатурна навігація, контрастність).

9) Тестування (функціональне, UX, мобільне).

10) Розгортання, супровід та подальше вдосконалення.

Етапність реалізації проекту за темою кваліфікаційної роботи дозволить поступово перейти від концепції до повноцінного адаптивного онлайн-ресурсу із навчальними матеріалами, зорієнтованого на студентів ЗВО як перших користувачів.

Загалом в ході виконання першого розділу кваліфікаційної роботи було розглянуто актуальність створення універсальних вебінтерфейсів для навчальних вебзастосунків, зокрема в сфері домедичної допомоги та тактичної медицини. Проаналізовано сучасні підходи у веброзробці, такі як адаптивний дизайн, SPA-архітектуру, компонентну модель, принципи доступності (A11y). Визначено вимоги до інтерфейсу, зумовлені специфікою цільової аудиторії — студентів та користувачів, які потребують швидкого, інтуїтивного й доступного доступу до критично важливої інформації.

РОЗДІЛ 2.
ПРОЕКТУВАННЯ ВЕБ-ІНТЕРФЕЙСУ

2.1 Аналіз потреб користувачів і вимог до веб-інтерфейсу

Важливим етапом проектуванні веб-інтерфейсу додатку у відповідності до тематики кваліфікаційної роботи та етапів реалізації проекту є визначення та опис специфікацій, визначення вимог до інтернет-ресурсу із навчальним контентом, написання технічного завдання, написання відповідної документації.

Перед розробкою інтерфейсу навчального вебдодатку важливо чітко визначити, хто є цільовими користувачами системи та які функціональні й візуальні очікування вони мають. У межах цього проєкту основними користувачами є:

- ✓ Студенти закладів вищої освіти, які проходять базову підготовку з домедичної допомоги або тактичної медицини.
- ✓ Молоді спеціалісти та волонтери, які потребують швидкого, практично орієнтованого навчання.
- ✓ Заклади вищої освіти та інші навчальні заклади, що можуть використовувати платформу як засіб формальної або неформальної освіти.

Виходячи з потреб користувачів, сформульовано основні вимоги до веб-інтерфейсу (таблиця 2.1.).

Загалом під функціональними вимогами розуміють вимоги, які визначають, що саме повинен робити вебзастосунок. Вони описують конкретні функції, які має виконувати система: реєстрація користувача, перегляд лекцій, проходження тестів, отримання сертифіката тощо.

Нефункціональні вимоги — це вимоги, які описують як система повинна працювати. Вони не стосуються конкретних функцій, але визначають якість роботи інтерфейсу: зручність використання, адаптивність, доступність, швидкодію, безперервність роботи на різних пристроях, безпеку тощо.

Таблиця 2.1. – Функціональні та нефункціональні вимоги проекту

Функціональна вимога	*Нефункціональна вимога*
Простий та інтуїтивний доступ до навчального контенту (лекції, відео, практичні кейси).	Чітка структура та навігація з мінімумом відволікаючих елементів.
Можливість проходження тестування з перевіркою знань у режимі реального часу.	Висока контрастність, великі шрифти та піктограми — для використання в умовах стресу або поганої видимості.
Особистий кабінет користувача з прогресом та сертифікатом після завершення курсу.	Підтримка доступності (A11y), а саме навігації з клавіатури, ARIA-атрибутів, альтернативних описів для зображень.
Адаптивність для мобільних пристроїв та планшетів	Забезпечення інтерактивності і мультимедійності (відео, анімації, схеми дій).
Збереження результатів навчання (у локальному сховищі або через серверне API).	Стабільність роботи при слабкому інтернет-з’єднанні (використання кешу, оптимізація).
Швидкий доступ до ключових інструкцій у вигляді шпаргалок (quick-access) або мінігайдів.	

Результатом цього аналізу є чіткий перелік вимог до інтерфейсу, які ляжуть в основу його архітектури та візуального дизайну. Надалі вони враховуватимуться на всіх етапах розробки, забезпечуючи відповідність платформи реальним очікуванням та поведінці користувачів.

Використання User Stories і приймальних критеріїв на етапі проектування проекту дозволяє розбивати великий проєкт на конкретні завдання; планувати і тестувати функціонал поетапно; покращувати якість інтерфейсу, орієнтованого на користувача.

User Story (користувацька історія) — це короткий опис функціональності або потреби, сформульований з точки зору кінцевого користувача. Такий підхід широко використовується у гнучкій розробці (Agile, Scrum) для фокусування на реальних потребах користувачів. Цей формат

допомагає зрозуміти, для кого, що потрібно реалізувати і навіщо: “Як студент, я хочу переглядати навчальні відео, щоб самостійно вивчати теми курсу”.

Acceptance Criteria — це чіткий перелік умов або вимог, за яких user story вважається виконаною і прийнятною для користувача або замовника. Вони конкретизують, що саме має бути реалізовано, як має працювати функція, та слугують основою для тестування.

Таблиця 2.2. – Короткий опис User Stories та критеріїв прийому вебзастосунку для навчання з базової медичної допомоги та тактичної медицини

№п/п	User Story	Acceptance Criteria
1	Як студент, я хочу переглядати лекції з у зручному форматі, щоб навчатися у власному темпі.	- Лекції доступні у форматі відео та тексту - Користувач може перемикатися між темами курсу
2	Як студент, я хочу проходити тести після модулів, щоб перевіряти свої знання.	- Після кожної теми доступний тест - Результати тесту відображаються одразу після проходження
3	Як користувач, я хочу бачити свій навчальний прогрес у кабінеті, щоб контролювати проходження.	- Відображається % завершення курсу - Показано, які модулі пройдені, а які ще ні
4	Як мобільний користувач, я хочу мати зручний доступ до курсу з телефону.	- Інтерфейс адаптується до екранів <576px - Кнопки та текст залишаються читабельними та зручними
5	Як користувач з особливими потребами, я хочу повноцінно користуватися платформою.	- Підтримується навігація клавіатурою - Впроваджені ARIA-атрибути та alt-описи для зображень
6	Як викладач, я хочу бачити результати студентів та оновлювати контент.	- Сторінка адміністрування курсу з таблицею результатів - Можливість додавати, редагувати лекції

Після аналізу потреб користувачів і визначення функціональних вимог було здійснено макетування інтерфейсу майбутнього вебдодатку. Основна мета цього етапу — візуалізувати логіку навігації, структуру сторінок,

розташування елементів керування та забезпечити інтуїтивне сприйняття навчального контенту.

Для макетування використовувався low-fidelity та medium-fidelity підхід:

1. Low-fidelity (на початковому етапі) — прості схеми навігації, блокові розмітки без стилів.
2. Medium-fidelity — чіткі компонування без деталізації візуального стилю, з позначенням функціональних елементів.

Інструментом для створення макетів було обрано Figma, що забезпечує швидку ітерацію, спільну роботу та підтримку компонетного підходу у розробці веб-інтерфейсів.

Таблиця 2.3 – Основні компоненти макету веб-інтерфейсу проекту

№п/п	Назва макету	Призначення
1	Головна сторінка	Ознайомлення з курсом, старт навчання, загальна структура модулів
2	Сторінка лекції	Перегляд відео або текстового матеріалу з теми
3	Сторінка тестування	Проходження тестів, перевірка знань, отримання результатів
4	Особистий кабінет	Відображення прогресу користувача, завершені модулі, доступ до сертифікату
5	Навігаційне меню	Доступ до основних розділів застосунку, адаптація під мобільні пристрої

На початковому етапі реалізації проекту навігаційне меню міститиме посилання на основні розділи: «Курс», «Тести», «Профіль», «Довідка».

На основі створених макетів було визначено ключові інтерфейсні компоненти та узгоджено структуру взаємодії користувача з вебзастосунком. Ці макети стали основою для наступного етапу — обгрунтування вибору технологій для реалізації.

2.2. Огляд бібліотек для створення інтерфейсів

Основою будь-якого вебінтерфейсу є так звані базові (основні) технології веброзробки — HTML, CSS і JavaScript. Вони забезпечують базову структуру, стилізацію й інтерактивність вебсторінок.

Мова розмітки HTML (HyperText Markup Language) відповідає за структуру і зміст вебсторінки. За допомогою HTML створюються заголовки, абзаци, кнопки, таблиці, поля форм тощо.

CSS (Cascading Style Sheets) використовуються для оформлення і візуальної стилізації сторінок, а саме задання кольорів, шрифтів, відступів, анімації, адаптивності тощо.

JavaScript забезпечує інтерактивну поведінку інтерфейсу: обробку подій, динамічні зміни контенту, валідацію форм, взаємодію з API тощо.

Рисунок 2.1. - Основні обмеження базових технологій у веб-розробці

Хоча HTML, CSS і JavaScript є необхідною основою, їх використання у базовому вигляді недостатнє для реалізації повноцінного, масштабованого й зручного вебінтерфейсу для сучасного навчального вебдодатку, зокрема такого, як інтернет-ресурс з навчальними матеріалами з надання першої домедичної допомоги та тактичної медицини.

Для розробки сучасних вебінтерфейсів існує широкий спектр бібліотек і фреймворків, які спрощують створення адаптивних, інтерактивних і доступних користувацьких інтерфейсів. У цьому підпункті наведено короткий огляд найбільш популярних та доцільних для реалізації навчального вебдодатку бібліотек.

React — це JavaScript-бібліотека для побудови інтерфейсів користувача, розроблена компанією Meta. Вона використовує компонентний підхід, що дозволяє створювати багаторазові та ізольовані UI-компоненти.

React Router є бібліотекою маршрутизації для React, завдяки якій можна створювати створювати багатосторінкові SPA-інтерфейси без перезавантаження сторінки.

Використовуючи React Router можна здійснювати підтримку динамічних маршрутів, реалізувати просте перемикання між сторінками., забезпечити інтеграцію з контекстом користувача та станом застосунку. Дана бібліотека може використовуватися в проекті для реалізації навігації між розділами: лекції, тести, кабінет користувача.

Material UI — це популярна бібліотека компонентів для React, що реалізує принципи Material Design від Google. За допомогою цієї бібліотеки можна використати готові адаптивні компоненти (кнопки, форми, навігація), здійснювати підтримку темної/світлої теми в проекті та забезпечити високу доступність (A11y). MUI дозволяє швидко створити сучасний, візуально привабливий інтерфейс, з дотриманням UX-практик.

Приклад коду для кнопки із використанням Material UI:

```
<Button variant="text" startIcon={<ShoppingCartRounded />}>
  Додати елемент
```

```
</Button>
<Button variant="contained" startIcon={<ShoppingCartRounded />}>
  Додати елемент
</Button>
<Button variant="outlined" startIcon={<ShoppingCartRounded />}>
  Додати елемент
</Button>
```

В результаті виконання коду на сайті може бути проставлена кнопка у трьох різних стилях (рис.

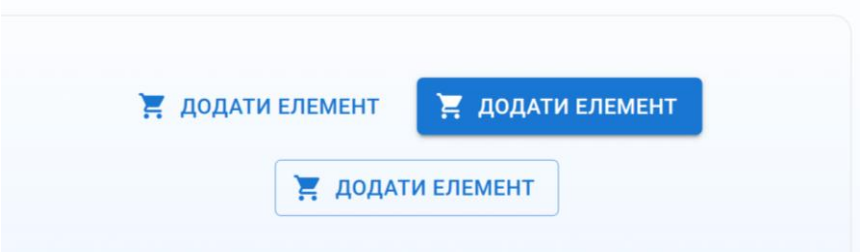


Рисунок 2.2 – Представлення елементу «Кнопка» із використанням Material UI

Tailwind CSS — це утилітарна CSS-бібліотека, яка дозволяє створювати стильові рішення без написання кастомних CSS-класів. Tailwind дає змогу розробнику швидко створювати чисті та адаптивні інтерфейси без потреби в громіздкому CSS.

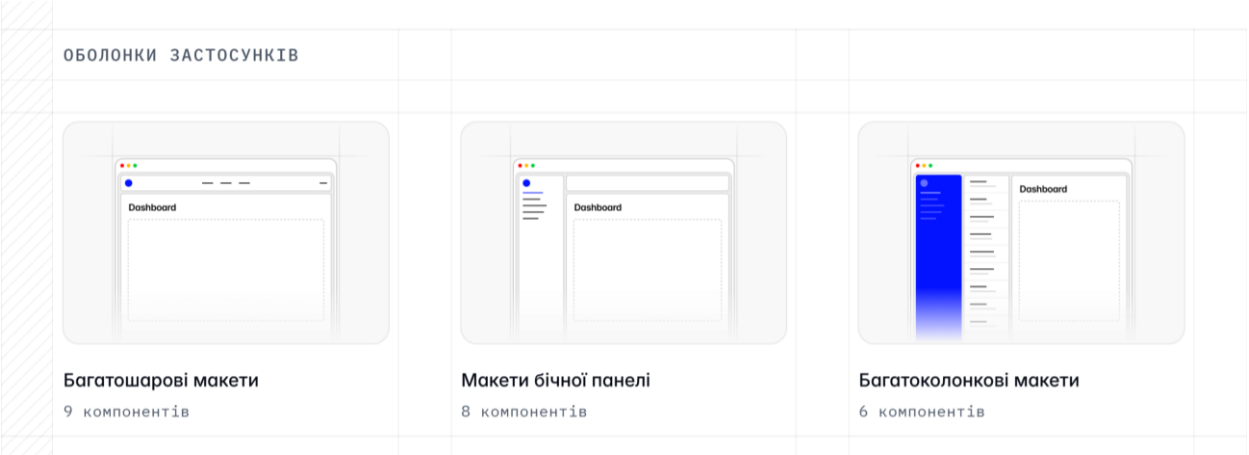


Рисунок 2.3. – Макети блоків інтерфейсу користувача із використанням Tailwind CSS [11]

Фреймворк Vue.js з'явився майже одночасно за React. Його творцем став розробник Google Еван Ю., який свого часу створював чимало прототипів інтерфейсів безпосередньо в браузері, і зіткнувся з проблемою нестачі інструментів для швидкого прототипування веб-додатків [10].

Vue.js є прогресивним JavaScript-фреймворком для створення інтерфейсів користувача, який поєднує функціональність шаблонів HTML, реактивність даних і компонентну архітектуру. Vue вирізняється низьким порогом входу, гнучкістю та лаконічним синтаксисом.

Як і React, Vue пропонує переваги віртуального DOM та компонентної структури, але відрізняється від першого реактивною прив'язкою даних та просунутою роботою з URL.

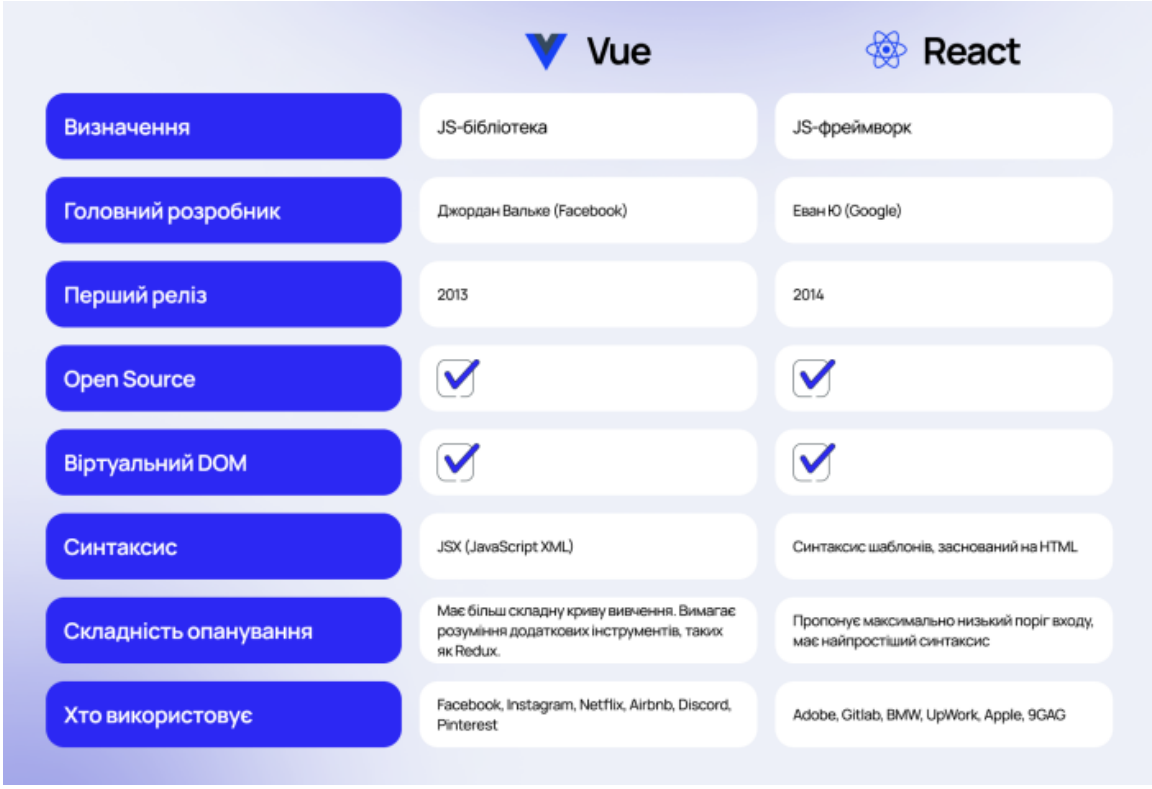


Рисунок 2.4. – Порівняльна характеристика React та Vue.js

Фреймворк Vue.js є ефективним інструментом для невеликих проєктів або сайтів із простою логікою, однак для створення масштабованого, компонентно-орієнтованого SPA-застосунку з освітніми функціями більш доцільним є використання React. Його гнучкість, модульність і активна

спільнота дозволяють реалізувати всі вимоги до функціоналу, дизайну та доступності платформи.

React дозволяє розділити інтерфейс на незалежні, повторно використовувані компоненти (наприклад: Header, LessonCard, TestForm). Це спрощує розробку, підтримку та масштабування вебзастосунку.

React-інтерфейси легко адаптуються до різних розмірів екранів, а використання атрибутів ARIA та відповідних компонентів дозволяє забезпечити доступність (A11y), що особливо важливо в освітніх платформах.

2.3 Вибір архітектрури вебдодатку

Однією з домінуючих архітектур у веб-розробці останніми роками є односторінкові вебзастосунки Single Page Application (SPA). Вебзастосунок, створений на основі SPA архітектури, завантажується динамічно без повного перезавантаження сторінки, що забезпечує швидку взаємодію з користувачем, мінімізує затримки та знижує навантаження на сервер. Це особливо важливо для інтерфейсів освітнього типу, де користувач постійно перемикається між модулями, тестами, відео та теоретичними матеріалами.

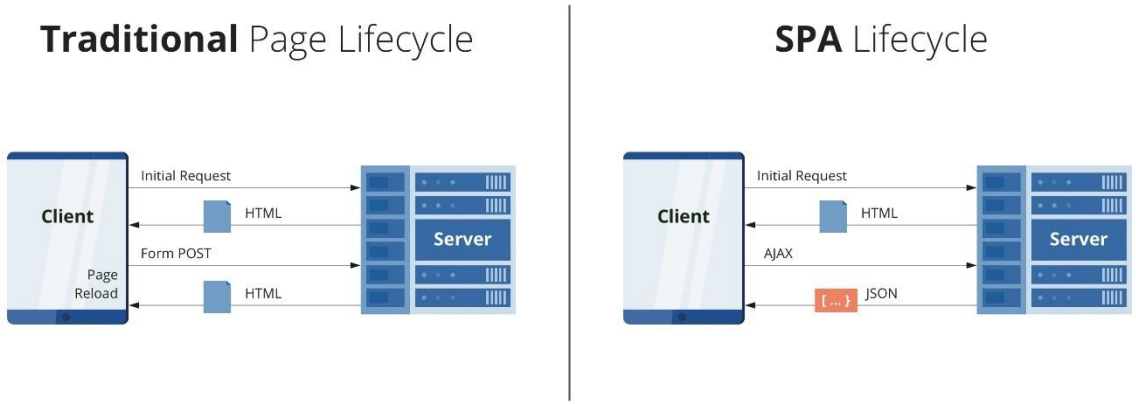


Рисунок 2.5. – Порівнянняя односторінкових (SPA) та багатосторінкових вебдодатків (MPA) [3]

Сучасні фреймворки, зокрема React, Angular і Vue.js, впроваджують компонентну архітектуру, яка дозволяє повторно використовувати частини інтерфейсу, ізолювати логіку і стиль, спростити тестування та обслуговування. У випадку вебінтерфейсу для курсів з домедичної допомоги це може допомогти легко масштабувати систему, додаючи наприклад нові блоки лекцій, окремі типи тестів, інтегруючи нові відео або створюююючи кастомні навігаційні елементи.

Під компонентною архітектурою в веб-розробці розуміють підхід, при якому додаток розбивається на незалежні, повторно використовувані частини (компоненти), які взаємодіють між собою для реалізації загальної функціональності. Цей підхід дозволяє розробникам створювати більш гнучкі, масштабовані та легші в підтримці вебдодатки [4].

До основних принципів компонентної архітектури вебдодатків відносять: модульність, ізольованість, повторне використання, зв'язок через інтерфейси та можливість легшого тестування.

У випадку застосування компонентної архітектури у веб-роробці компоненти взаємодіють між собою через чітко визначені інтерфейси, що спрощує їх інтеграцію в межах проекту та забезпечує гнучкість самої системи.

Використання 3-х рівненової архітектури вебдодатків передбачає розділення додатків на три рівні (таблиця 2.4).

Таблиця 2.4. – Компоненти трьох-рівневої архітектури

Рівень	Опис
Клієнтський або презентаційний рівень (Frontend)	Керує користувацьким інтерфейсом та взаємодією з користувачем
Рівень додатків (бізнес-логіка)	Відповідає за основну функціональність і процеси.
Рівень доступу до даних (база даних)	Керує зберіганням та пошуком даних.

Джерело: побудовано на основі даних [5]

Аналізуючи основні тенденції у сфері веб-розробки, можна також виділити підхід мікрофондендів, характерний для великих систем.

популярним стає підхід мікрофронтендів, коли інтерфейс розбито на незалежні модулі, кожен з яких може розвиватися автономно.

Це забезпечує масштабованість платформи, наприклад: окремий модуль для тестування, модуль для відео, модуль з особистим кабінетом. Хоча ця технологія складніша в реалізації, вона відповідає сучасним вимогам enterprise-рівня.

У монолітній архітектурі (SPA) веб-інтерфейс реалізується як єдиний застосунок, де всі сторінки, компоненти та логіка збираються в один фронтенд-проєкт (наприклад, на базі React). Навігація здійснюється на клієнті без перезавантаження сторінки (рисунок 2.6)

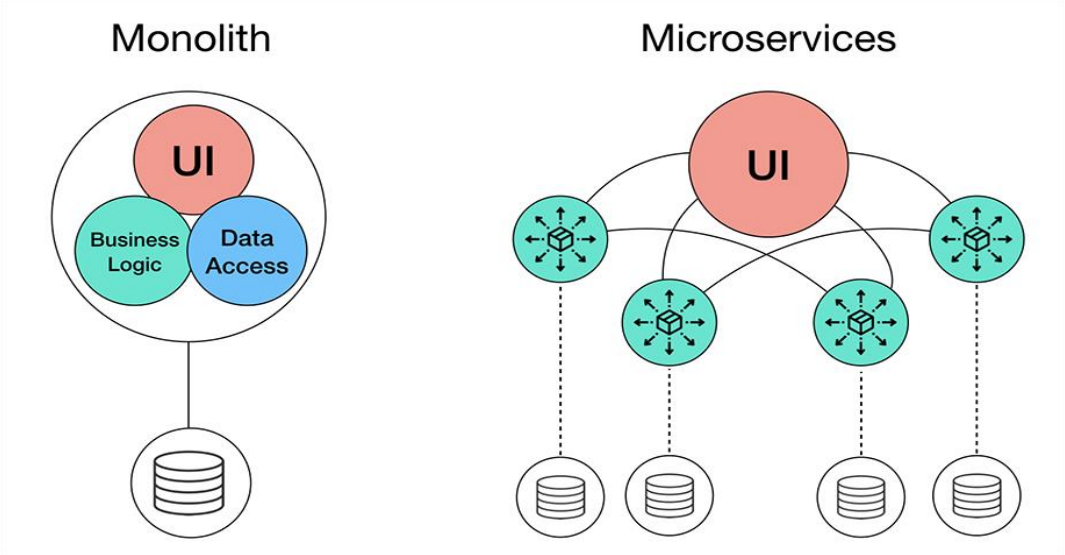


Рисунок 2.6. – Порівняння монолітної та мікросервісної архітектури у розробці вебдодатків [9]

Перевагами застосування підходу на основі монолітної архітуктури є:

- простота в реалізації і супроводі, тому цей підхід ідеально підходить для невеликих або середніх проєктів з обмеженою кількістю функцій.
- швидкий запуск MVP (мінімально життєздатного продукту).
- цілісна кодова база, що дозволяє легко контролювати залежності та компоненти.

- низький поріг входу для команди, який не потребує складної інфраструктури для реалізації проекту.

До недоліків монолітного підходу відносять те, що з часом можуть виникати складності масштабуванням або розподілом завдань між командами та менша гнучкість при інтеграції зовнішніх сервісів або модулів, створених іншими командами.

Мікрофронтенд архітектура є підхідом, за якого застосунок ділиться на незалежні модулі (міні-застосунки), кожен з яких має власну кодову базу, збирається окремо і може бути розроблений різними командами, навіть на різних технологіях.

Перевагами застосування підходу на основі мікрофронтенд архітуктури є:

- добре підходить для великих, модульних або багатофункціональних систем, де кожен модуль розвивається окремо.
- можливість перевикористання компонентів або навіть модулів в інших проєктах.
- зручність у застосуванні для декількох незалежних команд.

До недоліків застосування мікрофронтенд архітектури відносять: складність у налаштуванні інфраструктури (маршрутизацію, використання спільних бібліотек, синхронізація стану тощо); підвищення вимог до DevOps та застосування CI/CD для кожного модуля; недоцільність використання для простих застосунків.

Проаналізувавши особливості проєкту у відповідності до тематики кваліфікаційної роботи, а саме інтернет-ресурсу для навчання користувачів з домедичної допомоги й тактичної медицини, можна зробити висновок, що монолітна архітектура на основі SPA (наприклад, React + React Router) є найбільш доцільною. Це пов’язано із тим, що повноцінна реалізація проекту в перспективі може реалізуватися у формі навчальної платформи з лінійною навігацією, типовими модулями (лекції, тести, профіль користувача, сертифікати).

Практична реалізація проекту може бути здійснена невеликою командою веб-розробників, тому не виникатиме потреба у ізоляції фронтендів між різними командам. За потреби у майбутньому можна буде виділити окремі модулі (наприклад, тестування або сертифікацію) в окремі фронтенди з мінімальними змінами в архітектурі.

Таким чином, розробка сучасного вебінтерфейсу для онлайн-курсів із домедичної допомоги має базуватися на використанні адаптивного дизайну, компонентної архітектури, SPA-підходу, оптимізації продуктивності та забезпеченні доступності. Бібліотека React, як одна з провідних технологій у галузі, повністю підтримує ці тенденції та дозволяє ефективно реалізувати поставлену задачу.

2.4 Обгрунтування вибору бази даних

Для реалізації навчального вебзастосунку з надання першої домедичної допомоги та тактичної медицини використання бази даних є необхідним, оскільки:

- інформація повинна зберігатися між сесіями користувача;
- потрібен індивідуальний прогрес для кожного студента;
- слід реалізувати авторизацію та аутентифікацію користувачів;
- важливо зберігати тестові результати, сертифікати, модулі курсу та контент;
- можливе майбутнє підключення адміністративної панелі для викладача/адміністратора.

Залежно від структури застосунку, можна розглядати дві категорії БД: реляційні та нереляційні бази даних (табл.2.5).

Firebase Firestore — це хмарна NoSQL-база даних від Google, яка чудово підходить для SPA-застосунків на React завдяки: простій інтеграції з React через SDK; зберіганню даних у форматі документів/колекцій (зручно для користувачів, тем, тестів); автоматичному синхронному оновленню (live

updates); вбудованій авторизації (Google, email, password); безкоштовному тарифу для MVP або навчального проєкту.

Таблиця 2.5 – Порівняльна характеристика баз даних

Тип бази даних	Переваги	Недоліки
SQL (реляційні: PostgreSQL, MySQL)	Стабільність, складні зв'язки, транзакції	Потребує чіткої схеми таблиць
NoSQL (документні: Firebase Firestore, MongoDB)	Гнучка структура, легко масштабувати	Обмежена підтримка складних зв'язків

Firestore зберігає дані у вигляді колекцій (collections) та документів (documents). В додатку А наведено структуру бази даних проекту, що відповідає потребам вебдодатку за темою кваліфікаційної роботи. Для повноцінної роботи освітнього застосунку необхідна база даних для зберігання контенту курсу, користувацьких даних, результатів навчання. Firebase Firestore є оптимальним вибором завдяки простоті налаштування, гнучкій структурі та хорошій сумісності з React-застосунками.

В додатку Б представлено структуру бази даних у форматі JSON-моделі (Firebase Firestore-style), яка містить:

1) Ключі верхнього рівня, що відповідають колекціям у Firestore - users, courses, modules, lessons, tests.

2) Кожен документ у колекції має унікальний ID (user_123, module_1, тощо).

3) Вкладені об'єкти та масиви (наприклад, progress, questions[]) дозволяють зручно зберігати стан і пов'язані дані.

Таким чином, розробка сучасного вебінтерфейсу для онлайн-курсів із домедичної допомоги має базуватися на використанні адаптивного дизайну, компонентної архітектури, SPA-підходу, оптимізації продуктивності та забезпеченні доступності. Бібліотека React, як одна з провідних технологій у галузі, повністю підтримує ці тенденції та дозволяє ефективно реалізувати поставлену задачу.

РОЗДІЛ 3.
РОЗРОБКА ТА ОПТИМІЗАЦІЯ ПРОДУКТИВНОСТІ ВЕБ-ІНТЕРФЕЙСУ

3.1. Створення структури проекту та реалізація основних компонентів

У цьому розділі кваліфікаційної роботи розглянуто практичні аспекти реалізації вебінтерфейсу навчального застосунку з першої домедичної допомоги та тактичної медицини. Враховуючи потребу в масштабованості, повторному використанні елементів та інтерактивній взаємодії з користувачем, було обрано компонентно-орієнтований підхід із використанням бібліотеки React.

Розробка інтерфейсу охоплює такі ключові етапи:

- ✓ побудову логічної структури проєкту;
- ✓ створення користувацьких інтерфейсів (UI) із дотриманням принципів адаптивності та доступності;
- ✓ реалізацію прикладної логіки (керування станом, навігація, обробка подій);
- ✓ налаштування взаємодії з базою даних через API-запити (на прикладі Firebase Firestore).

Кожен з етапів реалізувався з урахуванням вимог до навчальних систем: зручності навігації, відстеження прогресу, швидкості взаємодії та гнучкості у поданні навчального матеріалу.

Наведена в таблиці 3.1. структура проекту із розробки вебдодатку для навчання з першої домедичної допомоги та тактичної медицини дозволить легко підтримувати й масштабувати проєкт, розділяючи відповідальність між логікою, UI та даними.

Таблиця 3.1. – Базова структура проекту

Назва компоненту	Призначення
assets	зображення, стилі
components	UI-компоненти (Button, Header, LessonCard тощо)
pages	окремі сторінки (Home, Lessons, Test, Profile)
layouts	макети (наприклад, з навігацією)
services	API-запити до Firebase або інших джерел
context	глобальний контекст (авторизація, прогрес)
routes	маршрутизація додатку (React Router)
utils	допоміжні функції
App.jsx	головний компонент

Для реалізації логіки застосунку доцільно використати React Context API для зберігання глобального стану (авторизація, поточний користувач, прогрес). Усі ключові дії (вхід, оновлення прогресу, відображення контенту) реалізовані у вигляді окремих логічних функцій. Це дасть змогу:

1) автоматично зберігати прогрес після перегляду лекції;
2) обробляти результати тестування;
3) керувати станом завантаження, помилками.

В ході виконання проекту за темою кваліфікаційної роботи було реалізовані повторно використовувані компоненти, що відповідають вимогам адаптивності та доступності:

<Header />, <Footer /> — навігаційні елементи з підтримкою ARIA;

<LessonCard /> — картки з описом теми;

<VideoPlayer /> — інтеграція відеоконтенту;

<ProgressBar /> — візуалізація прогресу;

<TestForm /> — форма для проходження тестів.

Компоненти вебдодатку реалізовано із дотриманням принципів Atomic Design, у відповідності до якого простіші компоненти (наприклад кнопки) складають складніші (форми, сторінки).

LessonCard — це окремий інтерфейсний компонент, який відповідає за відображення окремої теми або модуля навчального курсу. Його основна функція — коротко презентувати тему, щоб користувач міг переглянути її назву та опис і за потреби — перейти до повного змісту, наприклад, натиснувши або активувавши картку з клавіатури (рис.3.1).

```jsx
export default function LessonCard({ title, description, onClick }) {
  return (
    <div
      className="border rounded-2xl p-4 shadow-md hover:shadow-lg cursor-pointer transition-all"
      role="button"
      tabIndex="0"
      onClick={onClick}
      onKeyDown={(e) => e.key === "Enter" && onClick()}
      aria-label={`Переглянути тему: ${title}`}
    >
      <h3 className="text-lg font-bold">{title}</h3>
      <p className="text-sm text-gray-600">{description}</p>
    </div>
  );
}
```

Рисунок 3.1. – Фрагмент коду для реалізації компонента веб-інтерфейсу LessonCard

Компонент LessonCard, що є частиною веб-інтерфейсу проекту, реалізовано з урахуванням адаптивності та доступності:

- ✓ Використано role="button" і tabIndex="0", що робить картку доступною для людей, які користуються клавіатурою замість миші або застосовують допоміжні технології (екранні диктори).
- ✓ Застосовано обробку події onKeyDown, яка дозволяє активувати дію клавішею Enter, що є стандартом доступності.
- ✓ Використано атрибут aria-label, додаючи опис, який буде озвучено голосовим помічником (екранним диктором), і допомагає слабозорим користувачам розуміти, з якою темою вони взаємодіють.

З візуальної точки зору, компонент оформлено за допомогою класів Tailwind CSS, що забезпечує привабливий зовнішній вигляд: округлені кути, тінь, приємний ефект при наведенні миші (hover), внутрішній відступ тощо.

Використання такого компонента дозволяє створити уніфікований стиль подання навчального контенту, що сприяє покращенню користувацького досвіду (UX), а також спрощує повторне використання в різних частинах застосунку (список тем, дашборд прогресу, головна сторінка курсу).

В додатку В представлено Фрагмент коду для реалізації веб-інтерфейсу додатку з використанням React в частині створення тестових запитань до модулів.

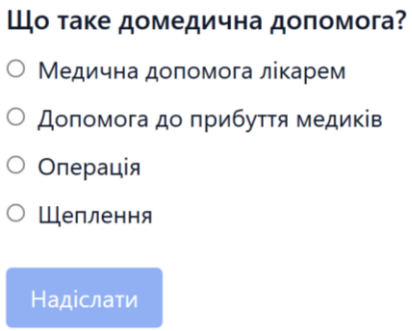


Рисунок 3.1. – Фрагмент вебдодатку із тестовим запитанням з першої домедичної допомоги

Виконуючи вимогу по забезпеченню доступності вебдодатку, що важливо для навчальних інтернет-ресурсів, які використовуються закладами вищої освіти, при проектуванні веб-інтерфейсу було реалізовано можливість розпізнавання скрінрідером запитань, варіантів, а також надання повідомлення, чи відповідь правильна чи ні після натискання кнопки.

Застосування ARIA-атрибутів в ході виконання практичої частини кваліфікаційної роботи дають змогу розробнику:

- ✓ підвищувати взаєморозуміння між UI та користувачем (особливо за допомогою скрінрідерів),
- ✓ дозволяти користувачам з вадами зору проходити тести, читати лекції та навігувати по інтерфейсу без миші,
- ✓ сприяти відповідності вимогам WCAG 2.1 (рівень A / AA).

Таблиця 3.2 – Застосування ARIA для покращення доступності тестових завдань

Елемент	Атрибут/Роль	Призначення
	role="form"	Оголошує блок як форму для скрінрідера
<h2 id="question-label">	aria-labelledby	Формально пов’язує заголовок питання з формою
<ul role="radiogroup">	role="radiogroup"	Групує радіокнопки у логічну множину для скрінрідерів
<input role="radio">	aria-checked	Повідомляє, який варіант вибрано (важливо для кастомних елементів)
	role="alert" та aria-live="assertive"	Оголошує результат негайно після відповіді (читання голосом)

В таблиці 3.3. наведено атрибути ARIA, які були застосовані до головної панелі навігації на платформі курсів.

Таблиця 3.3 – Застосування ARIA для покращення доступності навігаційного меню

Елемент	Атрибут/Роль	Призначення
<nav aria-label="Головна навігація">	aria-label	Дає скрінрідеру опис області навігації
<button aria-expanded aria-controls aria-label>	aria-expanded	Інформує, чи меню відкрите, яке воно контролює, і яка дія буде
<ul role="menu">	role="menu"	Позначає список як меню
<li role="none">	role="none"	Не маркує як елемент меню (щоб уникнути дублювання ролей)
<a role="menuitem">	role="menuitem"	Кожен пункт сприймається як пункт меню для скрінрідера

В ході виконання кваліфікаційної роботи вебдодаток з навчальними матеріалами з першої домедичної допомоги та тактичної медицини матеріалами, який може бути корисний для здобувачів закладів вищої освіти, був реалізований на етапі прототипу в частині інтерфейсу користувача.

З метою подальшої реалізації проекту на наступних кроках буде необхідно забезпечити взаємодію вебзастосунку з базою даних Firebase SDK.

Реалізувати поставлене завдання можна буде із використанням запитів, а саме:

- getCourseModules(courseId) — для отримання модулів певного курсу;
- submitTestResult(userId, testId, score) — для збереження результатів тесту;
- getUserProgress(userId) — для отримання прогресу користувача;
- updateLessonStatus(userId, lessonId) — для оновлення статусу "переглянуто".

Запити доцільно буде організувати в окремому шарі services/firebaseService.js, що дозволить централізовано змінювати джерело даних у майбутньому (наприклад, при переході з Firebase на PostgreSQL).

3.2.Тестування та оптимізація продуктивності веб-інтерфейсу

Розробка вебінтерфейсу не завершується лише створенням функціональних компонентів веб-інтерфейсу. Важливим етапом у розробці є тестування працездатності, оцінка продуктивності та оптимізація взаємодії користувача із системою. Це дозволяє забезпечити швидку роботу застосунку, коректну поведінку на різних пристроях і зручність використання інтерфейсу навіть при обмежених ресурсах.

На етапі ручного тестування веб-інтерфейсу додатку було перевірено:

- ✓ адаптивність верстки для різних розмірів екранів (мобільні, планшети, десктоп);
- ✓ коректність роботи навігації;

- ✓ правильність рендерингу компонентів у різних станах (незаповнена форма, успішне проходження тесту тощо);
- ✓ доступність через клавіатуру (Tab, Enter, Escape);
- ✓ відображення сповіщень про помилки та валідацію форм.

На наступному етапі виконання проекту для модульного тестування логіки застосунку використано бібліотеку Jest. Окремі функції, такі як обчислення прогресу та перевірка відповідей на тести, винесено в utils/, що дозволяє їх протестувати ізольовано.

Приклад простого юніт-тесту:

```
import { calculateProgress } from "../progress";

test("correctly calculates progress", () => {
  const result = calculateProgress({ total: 5, completed: 3 });
  expect(result).toBe(60);
});
```

За допомогою Google Lighthouse може бути реалізована: перевірка продуктивність рендерингу; доступності вебдодатку (A11y, ARIA, відповідність щодо контрастності); використання кешу та lazy loading.

Google Lighthouse є безкоштовний автоматизований інструмент з відкритим кодом, який використовується для аудиту продуктивності, доступності, SEO та дотримання вебстандартів у вебзастосунках. Lighthouse інтегровано в браузер Google Chrome (DevTools → Audit) або доступний як окремий CLI-інструмент.

Отримані результати свідчать, що інструменти Google Lighthouse допомагають виявити та усунути недоліки ще на етапі розробки. Особливу увагу приділяються доступності та продуктивності, адже для освітніх систем, якими можуть користуватись студенти на мобільних пристроях або користувачі з порушеннями зору, це критично важливо. Lighthouse допомагає не лише оцінити якість застосунку, але й вказує шляхи для вдосконалення.

Таблиця 3.4. - Основні категорії веб-інтерфейсу, які перевіряє Lighthouse

Категорія	Опис
Performance	Швидкість завантаження сторінки, рендеринг, час до взаємодії (TTI).
Accessibility	Перевірка доступності інтерфейсу для людей з інвалідністю (A11y).
Best Practices	Дотримання сучасних практик веброзробки, безпечність
SEO	Готовність сторінки до індексації пошуковими системами.
PWA	Аудит якості прогресивного вебдодатку (опціонально).

Продуктивність є критичним фактором для вебзастосунків, які орієнтовані на масового користувача. У випадку навчального інтерфейсу для курсів із домедичної допомоги важливо забезпечити швидкий відгук системи, низький час завантаження і стабільність навіть при нестабільному інтернет-з'єднанні чи використанні мобільних пристроїв.

Оптимізація продуктивності охоплює не лише фронтенд-частину, а й загальний підхід до архітектури, способів завантаження даних, взаємодії з API, структури компонентів та вибору технологій.

Існує ряд методів, які можна використовувати для покращення продуктивності вебдодатків, а саме: розбиття коду на частини (code splitting), застосовується ліниве завантаження компонентів (lazy loading), кешування, мемоізація даних і рендеринг за потреби.

Оскільки проект за темою кваліфікаційної роботи передбачає розміщення відео-матеріалів в межах навчальних модулів, було визначено, що для оптимізації продуктивності вебдодатку на початковому етапі буде застосовано метод оптимізації медіа-контенту.

У відповідності до методу оптимізації медіа-контенту:

- зображення зберігаються у форматі WebP, що забезпечує менший розмір без втрати якості;

- відео не вбудовані напряму, а інтегровані через YouTube, що знижує навантаження на сервер;
- усі зображення використовують width і height для уникнення зміщень контенту (CLS).

Застосовані підходи дозволили значно покращити продуктивність вебзастосунку як на десктопних, так і на мобільних пристроях. Завдяки розділенню коду, оптимізації завантаження ресурсів та обмеженню сторонніх бібліотек, інтерфейс залишається швидким і чуйним навіть за умов обмеженої пропускної здатності мережі, що особливо важливо в умовах реального використання онлайн-ресурсу студентами та викладачами.

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Аналіз травмонебезпечних ситуацій під час виконання робіт

Забезпечення безпечних умов праці є невід'ємною складовою стабільного функціонування будь-якого підприємства, незалежно від його галузі. У той час як тактичної медицини та першої домедичної допомоги численних факторів ризику є очевидною, в ІТ-сфері часто помилково вважають, що вона не становить суттєвої загрози для здоров'я персоналу. ІТ-компанії та ІТ-відділи мають власні специфічні виклики, які, хоч і не пов'язані з фізичним навантаженням, можуть серйозно впливати на фізичне самопочуття працівників та їхній психоемоційний стан.

Попри те, що діяльність ІТ-компаній переважно не пов'язана з фізичною працею, офісним персоналом або виробництвом у класичному розумінні, ризики для здоров'я та безпеки працівників все ж існують. Тривале перебування за комп'ютером, неправильна організація робочого місця, психоемоційне навантаження, а також неякісна електрифікація офісу можуть створювати загрози як короткострокового, так і накопичувального характеру.

Основні загрози полягають у порушенні принципів ергономіки, зневажанні технікою безпеки, а також у психоемоційному виснаженні. Для мінімізації ризиків необхідно впроваджувати комплекс заходів: організацію безпечного і комфортного робочого простору, регулярне навчання співробітників, моніторинг психоемоційного стану персоналу та формування культури відповідального ставлення до охорони праці.

До основних причин виникнення травмонебезпечних ситуацій на підприємствах можна віднести:

- Відсутність культури профілактики мікроперерв та фізичної активності.

- Ігнорування норм охорони праці через «офісну» специфіку діяльності.
- Роботу з дому ІТ-фахівців без належного контролю робочих умов.
- Високе навантаження, дедлайни, постійна багатозадачність ІТ-працівників.

Забезпечення безпечних умов праці є невід'ємною складовою стабільного функціонування будь-якого підприємства, незалежно від його галузі. У той час як на виробництві крафтової продукції наявність численних факторів ризику є очевидною, в ІТ-сфері часто помилково вважають, що вона не становить суттєвої загрози для здоров'я персоналу. Втім, ІТ-компанії та ІТ-відділи мають власні специфічні виклики, які, хоч і не пов'язані з фізичним навантаженням, можуть серйозно впливати на фізичне самопочуття працівників та їхній психоемоційний стан.

Аналіз травмонебезпечних ситуацій у сфері ІТ вказує на те, що хоча ці ризики є менш очевидними, вони мають накопичувальний характер і можуть істотно впливати на здоров'я персоналу. Тому важливо не тільки дотримуватись технічних норм безпеки, а й формувати корпоративну культуру здорової та безпечної праці. ІТ-компаніям доцільно регулярно проводити інструктажі, ергономічні перевірки робочих місць та стимулювати баланс між працею і відпочинком.

4.2. Структурно-функціональний аналіз дотримання охорони праці при роботі з комп'ютером

З розвитком цифрових технологій більшість професійних завдань виконується за допомогою комп'ютерної техніки. Робота за комп'ютером є характерною для працівників ІТ-компаній, офісних фахівців, студентів і багатьох інших категорій користувачів. Незважаючи на відсутність безпосередньо травмонебезпечного обладнання, тривале використання

комп’ютерів становить ризики для здоров’я людини, які мають бути враховані у системі охорони праці.

Метою структурно-функцінального аналізу дотримання охорони праці на підприємстві є виявлення основних структурних елементів організації праці за комп’ютером і визначення функціональних механізмів запобігання професійним захворюванням та іншим негативним впливам на здоров’я користувачів.

Робота з комп’ютером, яка на перший погляд здається безпечною, у разі тривалого виконання без дотримання відповідних умов може призводити до суттєвих негативних наслідків для здоров’я працівника. Одним з найпоширеніших ризиків є порушення опорно-рухового апарату, зокрема викривлення постави та хронічні болі у спині та шиї, що виникають внаслідок неправильного положення тіла під час роботи за неергономічним робочим місцем. Значне навантаження на зоровий апарат, зумовлене тривалим фокусуванням на екрані монітора без відповідного освітлення, може спричиняти втому очей, головні болі, а з часом — погіршення зору. Робота із клавіатурою та мишею у неправильному положенні рук часто стає причиною розвитку тунельного синдрому (синдрому зап’ястного каналу), який супроводжується болем, онімінням та зниженням чутливості пальців.

Окрему загрозу становить гіподинамія — нестача фізичної активності, яка є типовою для офісної праці. Вона збільшує ризик розвитку серцево-судинних захворювань, порушень обміну речовин і загального ослаблення організму. Поряд із фізичними навантаженнями, серйозною проблемою для працівників ІТ-сфери стає психоемоційне перевантаження. Постійний тиск дедлайнів, багатозадачність, велике інформаційне навантаження та тривале перебування у стресових умовах можуть призводити до професійного вигорання, зниження мотивації, розладів сну, а в окремих випадках — до депресивних станів.

Таким чином, ризики під час роботи з комп’ютером мають як фізичну, так і психологічну природу, і потребують комплексного підходу до профілактики та створення безпечного робочого середовища.

До функціональних рекомендацій для мінімізації ризиків на робочому місці можна віднести:

- ✓ Використання ергономічних меблів та налаштування робочого місця відповідно до ДСТУ та санітарних норм.
- ✓ Дотримання режиму "20-20-20": кожні 20 хвилин робити перерву на 20 секунд, дивлячись на об’єкт за 20 футів (≈6 м).
- ✓ Встановлення якісного освітлення з мінімальними відблисками на екрані.
- ✓ Використання фільтрів синього світла, темних тем, адаптивних інтерфейсів.
- ✓ Проведення регулярних інструктажів з охорони праці, особливо для нових співробітників.
- ✓ Створення умов для психологічного розвантаження: кімнати відпочинку, гнучкий графік, підтримка від HR-служби.

Робота з комп’ютером потребує не менш ретельного контролю дотримання норм охорони праці, ніж робота у фізично небезпечних умовах. Структурно-функціональний аналіз дозволяє комплексно оцінити середовище, в якому працює ІТ-спеціаліст, і впровадити заходи, що сприяють збереженню його здоров’я, працездатності та добробуту. Формування безпечного цифрового робочого простору — одна з ключових умов сталого розвитку сучасних організацій.

4.3. Обґрунтування організаційно-технічних рекомендацій з охорони праці

Для забезпечення безпечних умов праці в організаціях ІТ-сфери важливо впроваджувати не лише загальні принципи охорони праці, а й конкретні організаційно-технічні заходи, які враховують специфіку професійної діяльності за комп’ютером. Такі рекомендації повинні бути спрямовані на мінімізацію впливу шкідливих факторів, збереження фізичного та психоемоційного здоров’я працівників, підвищення їхньої продуктивності та задоволеності умовами праці.

Організаційні заходи насамперед передбачають чітке регламентування режиму праці та відпочинку. Згідно з чинними нормативами, при роботі за комп’ютером рекомендується робити перерви кожні 50–60 хвилин тривалістю 5–15 хвилин, під час яких працівники мають змогу змінити вид діяльності або виконати легку гімнастику. Важливо, щоб такі паузи не сприймались як особистий вільний час, а були передбачені внутрішнім розпорядком або регламентом роботи ІТ-компанії. Крім того, роботодавець має забезпечити інформаційне інструктування персоналу щодо ризиків комп’ютеризованої праці, методів профілактики перевтоми, порушень постави, зору, а також поширення психологічного вигорання.

До технічних рекомендацій насамперед належать вимоги до ергономічного облаштування робочого місця. Стіл і крісло мають бути регульованими за висотою, щоб працівник міг тримати спину прямо, а стопи рівно стояли на підлозі або підставці. Монітор повинен бути розташований на рівні очей, на відстані 50–70 см, із можливістю регулювання яскравості й контрастності. Клавіатура та миша повинні знаходитись на одному рівні з ліктями, не викликаючи перенапруги кистей. Не менш важливим є забезпечення достатнього рівня освітлення — як природного, так і штучного, з урахуванням відсутності відблисків на екрані.

З технічної точки зору також рекомендовано впроваджувати програмні рішення для зменшення втоми — наприклад, темні теми в інтерфейсі, фільтри синього світла, нагадування про перерви. Для віддалених працівників необхідно розробити вказівки щодо безпечного облаштування домашнього офісу та контролювати їхнє дотримання в межах політики компанії.

Окрему увагу слід приділити створенню психологічно безпечного робочого середовища: запровадженню гнучкого графіка, підтримці work-life балансу, недопущенню надмірного тиску, а також наданню можливості звертатися до HR або психолога у разі емоційного виснаження.

Усі зазначені рекомендації мають бути систематизовані в локальних нормативних документах підприємства, зокрема в інструкціях з охорони праці, техніці безпеки та внутрішніх регламентах. Їх дотримання не лише знижує ризики виникнення захворювань і нещасних випадків, а й позитивно впливає на імідж компанії як соціально відповідального роботодавця.

4.4. Безпека в надзвичайних ситуаціях

Надзвичайні ситуації (НС) — це обставини, які виникають раптово, мають потенційну загрозу життю, здоров’ю людей, майну та довкіллю, і потребують негайного реагування. До таких ситуацій належать як природні катаклізми, техногенні аварії, так і збройні конфлікти, які стали особливо актуальними для України в умовах повномасштабної війни. Тому організації, включаючи ІТ-компанії, повинні адаптувати внутрішні політики до реальних ризиків та забезпечити належну безпеку працівників в умовах воєнного стану та повітряних тривог.

У контексті ІТ-сфери, де більшість працівників знаходяться в офісах або працюють дистанційно, заходи безпеки повинні включати як організаційні інструкції, так і інфраструктурні рішення для захисту життя і здоров’я співробітників.

Найбільш поширеною загрозою в умовах війни є повітряні тривоги, під час яких існує реальна небезпека ракетного обстрілу або атаки дронами. Незалежно від характеру робочої діяльності, кожен працівник повинен мати чітке розуміння дій у разі загрози.

Рекомендації з безпеки працівників в умовах повітряної тривоги та воєнного стану:

1. Якщо компанія працює офлайн, офісне приміщення повинно бути обладнане укриттям або мати договір про користування найближчим захисним спорудженням цивільного захисту. Шлях до укриття має бути промаркований, доступним і не захараценим.

2. Працівники повинні бути ознайомлені з алгоритмом дій у разі тривоги, а також регулярно проходити навчальні евакуаційні заходи. Рекомендовано розміщення інструкцій на видимих місцях в офісі.

3. Усі співробітники мають право та обов’язок перервати робочий процес та перейти до укриття. Керівництво компанії має підтримувати відповідальне ставлення до таких ситуацій і не стимулювати «роботу попри тривогу».

4. Має бути призначена особа, відповідальна за безпеку підрозділу або офісу, яка координує евакуацію, перевіряє присутність працівників у сховищі та контролює завершення тривоги.

5. За можливості, компанії рекомендується дозволяти віддалений режим роботи у небезпечних регіонах, а також адаптувати графік залежно від кількості тривог на добу.

6. В офісі повинні бути аптечки, ліхтарики, вода, мінімальний запас їжі. Укриття має бути обладнане місцями для сидіння та зв’язком.

7. Перебування в умовах постійної загрози негативно впливає на психоемоційний стан. Рекомендовано створення внутрішніх програм підтримки: можливість консультацій з психологом, гнучке навантаження, командна підтримка.

8. Кожен працівник повинен мати доступ до внутрішніх каналів зв’язку (чатів, email, телефонів) для отримання термінових повідомлень і підтвердження своєї безпеки у випадку надзвичайних подій.

Формування безпечного середовища праці в умовах надзвичайних ситуацій — критично важливе завдання для кожного роботодавця. IT-компанії, навіть за своєю «нефізичною» природою, мають бути готовими до будь-яких сценаріїв та мати чітко визначені протоколи дій. Турбота про безпеку — це не лише обов’язок згідно з законодавством, а й показник відповідального ставлення до команди. Адаптація організаційної структури до умов воєнного часу сприяє збереженню життя, довіри колективу та стабільності в роботі компанії.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було вирішено комплексне завдання, що поєднує проєктування, реалізацію та оптимізацію сучасного веб-інтерфейсу з урахуванням принципів адаптивності, доступності та високої продуктивності. Основною метою дослідження було створення універсального користувацького інтерфейсу для вебдодатку, що забезпечує ефективну взаємодію користувача з навчальними матеріалами з домедичної допомоги та тактичної медицини.

У першому розділі було проведено теоретичний огляд понять, пов’язаних з універсальним інтерфейсом, особливостями онлайн-освіти у сфері домедичної допомоги, а також тенденціями у веброзробки. Визначено, що в умовах зростаючої потреби в якісній підготовці населення до дій у кризових ситуаціях, розробка інтуїтивно зрозумілих та доступних онлайн-курсів набуває особливої актуальності.

У другому розділі роботи було розглянуто вимоги до веб-інтерфейсу, а саме: функціональні, нефункціональні. На основі цього підготовчого етапу було проведено макетування ключових сторінок застосунку, сформовано базову інформаційну структуру та user story. Обґрунтовано доцільність вибору бібліотеки React, а також використання технологій Tailwind CSS, Firebase та ARIA-атрибутів.

У третьому розділі безпосередньо реалізовано інтерфейс застосунку, створено низку компонентів, які відповідають сучасним вимогам доступності, мобільної адаптивності і повторного використання. Під час розробки застосовано SPA-архітектуру з розподілом за маршрутом, що забезпечує плавний UX без перезавантаження сторінок.

Особливу увагу приділено питанням тестування та оптимізації продуктивності. Застосовано ручне та інструментальне тестування (Google Lighthouse, Chrome DevTools), перевірено доступність компонентів,

оптимізовано завантаження зображень, адаптацію інтерфейсу до слабких пристроїв.

У результаті виконаної роботи вдалося досягти наступних цілей:

- ✓ створити функціональний, гнучкий та зручний для користувача веб-інтерфейс;
- ✓ забезпечити адаптивність веб-інтерфейсу до різних пристроїв і розмірів екранів;
- ✓ гарантувати доступність контенту для всіх категорій користувачів;
- ✓ забезпечити оптимальну продуктивність навіть за умов обмеженого інтернет-з'єднання.

Запропоноване рішення є масштабованим і може бути розширене до повноцінної освітньої платформи, а також використане як шаблон для створення інших спеціалізованих курсів.

Таким чином, поставлені в роботі завдання були виконані повною мірою, а результати мають як практичне, так і методичне значення для подальшої розробки навчальних вебресурсів з урахуванням принципів універсального дизайну та сучасних стандартів веброзробки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Dabhi Ch. Progressive Web Apps: The Future of Mobile Development. URL: https://www.alliancetek.com/blog/post/2023/08/20/progressive-web-apps-the-future-of-mobile-development.aspx (дата звернення - 20.12.2024).
2. Progressive Web App (1) What is a PWA? URL: https://elice.io/en/newsroom/pwa_1 (дата звернення - 22.12.2024).
3. Valuy S. A Comparison of Single-Page and Multi-Page Applications. URL: https://dzone.com/articles/the-comparison-of-single-page-and-multi-page-appli (дата звернення - 25.01.2025).
4. Responsive Design: Definition, Examples, Principles, and Best Practices. URL: https://www.lambdatest.com/learning-hub/responsive-design (дата звернення - 14.12.2024).
5. Розуміння сучасних архітектур веб-додатків. URL: https://surl.li/laahar (дата звернення - 12.02.2025).
6. Михайлов В. Інклюзивний UX або як зробити продукт доступним для користувачів. URL: https://surl.lu/soaafh. (дата звернення – 14.03.2025).
7. Web Content Accessibility Guidelines (WCAG) 2.1. URL: https://www.w3.org/Translations/WCAG21-ua/ (дата звернення – 14.03.2025).
8. Орієнтири WAI-ARIA. URL: https://surl.li/bnluhn (дата звернення – 18.03.2025).
9. Зіневич О. Мікросервісний підхід у веб-розробці: micro frontends. URL: https://dou.ua/lenta/articles/micro-frontend/ (дата звернення – 24.04.2025).
10. Порівняння React та Vue за ключовими параметрами: що краще? URL: https://wezom.com.ua/ua/blog/vue-vs-react (дата звернення – 28.04.2025).
11. You can build anything with Tailwind CSS. URL: https://tailwindcss.com/showcase (дата звернення – 05.05.2025).
12. Що таке верстка сайту? URL: https://wezom.com.ua/ua/blog/verstka

13. Лісовська А., Калита, А. Контент веб-сайтів і їхня структура. *Молодий вчений*, вип. 10 (74), 2019. С. 166-170. https://doi.org/10.32839/2304-5809/2019-10-74-39

14. Веб-доступність. Що варто знати кожному Front-end розробнику і дизайнеру. URL: http://gud.org.ua/ (дата звернення – 23.05.2025).

15. Желєзняк А.М., Пташник В.В., Смолінський В.Б. Основні компоненти вебдоступності програмного забезпечення для сільського господарства. Вісник Львівського національного університету природокористування «Агроінженерні дослідження», №26 (2022). С.171-176.

16. Клим Ю.В., Тарасенко Ю.С. Тестування веб-ресурсів на інклюзивність: аудит для реінженірингу. Вісник Вінницького політехнічного інституту. 2022. №3. С.60-66.

17. Бернадська Н., Джумелія Е., Кочан О. Розробка веб-інтерфейсу інформаційно-аналітичної системи екологічного моніторингу з використанням бібліотеки react. *Information technology and society*. 2023. С. 6-12. 10.32689/maup.it.2023.1.1.

18. Липова О. М. Особливості розробки інтерфейсу веб-додатків. Сучасні електромеханічні та інформаційні системи : монографія / за заг. ред. І. В. Панасюка. -Київ : КНУТД, 2021. С. 112-117.

19. ARIA in HTML (W3C Recommendation 21 December 2023). URL: https://www.w3.org/TR/html-aria/

20. Башовий В.М., Стаценко В.В., Стаценко Д.В. Визначення швидкості роботи сучасних фреймворків для створення web-інтерфейсів. *Технології та інжиніринг*. 2022. № 4 (9). С. 9 –16.

21. Інструкція з охорони праці при роботі з персональним комп’ютером. URL: https://pro-op.com.ua/article/485-nstruktsya-z-ohoroni-prats-pri-robot-na-personalnomu-kompyuter

ДОДАТКИ

Додаток А

Структура бази даних проекту

Таблиця 1 - Інформація про зареєстрованих користувачів (**users**)

Поле	Тип	Опис
uid	string	Унікальний ідентифікатор (авто)
email	string	Електронна пошта
name	string	Ім’я користувача
role	string	"student" / "admin"
progress	map	ID тем → статус проходження
completedTests	map	ID тестів → % правильних відповідей
certificateURL	string/null	Посилання на сертифікат (опційно)

Таблиця 2 – Опис курсів (courses)

Поле	Тип	Опис
courseId	string	Ідентифікатор купсу
title	string	Назва курсу
description	string	Опис
modules	array	Масив ID модулів
imageURL	string	Посилання на зображення

Таблиця 3 - Інформація про тематичні блоки, які складаються з лекцій та тестів (модулі курсу **- modules**)

Поле	Тип	Опис
moduleId	string	Унікальний ID
courseId	string	Прив’язка до курсу
title	string	Назва модуля
description	string	Короткий опис
lessons	array	Масив ID лекцій
testId	string	ID тесту

Таблиця 4 - Інформація про навчальний матеріал (лекції **- lessons**)

Поле	Тип	Опис
lessonId	string	Унікальний ID
moduleId	string	Прив’язка до модуля
title	string	Назва лекції
content	string	HTML / markdown / JSON
videoURL	string	Посилання на відео
imageURLs	array	Зображення

Таблиця 5 - Інформація про закріплення знань після модуля (тести**- tests**)

Поле	Тип	Опис
testId	string	ID тесту
moduleId	string	Прив’язка до модуля
questions	array	Масив запитань
Підструктура сутності questions[]		
question	string	Текст питання
options	array	Варіанти відповіді
correct	number	Індекс правильної відповіді

Додаток Б

JSON-модель бази даних проекту (структура документів)


```
{
 "users": {
  "user_123": {
   "uid": "user_123",
   "email": "student@example.com",
   "name": "Іван Студент",
   "role": "student",
   "progress": {
    "module_1": "completed",
    "module_2": "in_progress"
   },
   "completedTests": {
    "test_1": 80,
    "test_2": 100
   },
   "certificateURL": null
  }
 },
 "courses": {
  "course_1": {
   "courseId": "course_1",
   "title": "Перша домедична допомога",
   "description": "Базовий курс із надання першої допомоги у надзвичайних ситуаціях.",
   "modules": ["module_1", "module_2"],
   "imageURL": "https://example.com/course-cover.jpg"
  }
 },
 "modules": {
  "module_1": {
   "moduleId": "module_1",
   "courseId": "course_1",
   "title": "Огляд домедичної допомоги",
   "description": "Теоретичні основи та типові ситуації.",
   "lessons": ["lesson_1", "lesson_2"],
   "testId": "test_1"
  }
 },
 "lessons": {
  "lesson_1": {
   "lessonId": "lesson_1",
   "moduleId": "module_1",
   "title": "Вступ до домедичної допомоги",
   "content": "<p>Що таке перша допомога...</p>",
   "videoURL": "https://example.com/lesson-video.mp4",
   "imageURLs": [
    "https://example.com/image1.jpg",
    "https://example.com/image2.jpg"
```

```
      ]
    }
  },
  "tests": {
    "test_1": {
      "testId": "test_1",
      "moduleId": "module_1",
      "questions": [
        {
          "question": "Що таке домедична допомога?",
          "options": [
            "Медична допомога лікарем",
            "Допомога, яку надає перехожий до прибуття медиків",
            "Операція",
            "Вакцинація"
          ],
          "correct": 1
        },
        {
          "question": "Який номер виклику швидкої в Україні?",
          "options": ["102", "103", "104", "112"],
          "correct": 1
        }
      ]
    }
  }
}
```

Додаток В

Фрагмент коду для реалізації веб-інтерфейсу додатку з використанням React (створення тестових запитань до модулів)

```
import { useState } from "react";
import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";

function Header() {
 return (
  <header className="bg-blue-700 text-white p-4">
   <nav aria-label="Головна навігація">
    <ul className="flex flex-wrap gap-4 justify-center">
     <li><Link to="/">Головна</Link></li>
     <li><Link to="/course">Курс</Link></li>
    </ul>
   </nav>
  </header>
 );
}

function Footer() {
 return (
  <footer className="bg-gray-100 text-center p-4 mt-8" aria-label="Футер сайту">
   <p className="text-sm text-gray-600">© 2025 Платформа домедичної допомоги</p>
  </footer>
 );
}

function LessonCard({ title, description }) {
 return (
  <div className="border rounded-2xl p-4 shadow-md hover:shadow-lg cursor-pointer transition-all">
   <h3 className="text-lg font-bold">{title}</h3>
   <p className="text-sm text-gray-600">{description}</p>
  </div>
 );
}

function VideoPlayer({ src, title }) {
 return (
  <div className="aspect-video w-full">
   <iframe
    src={src}
    title={title}
    className="w-full h-full rounded-xl"
    allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture"
    allowFullScreen
```

```
            aria-label={`Відео: ${title}`}
          ></iframe>
        </div>
      );
    }

    function ProgressBar({ progress }) {
      return (
        <div className="w-full bg-gray-300 rounded-full h-4" aria-label="Прогрес">
          <div
            className="bg-green-500 h-4 rounded-full transition-all"
            style={{ width: `${progress}%` }}
            role="progressbar"
            aria-valuemin="0"
            aria-valuemax="100"
            aria-valuenow={progress}
            aria-label={`Прогрес: ${progress}%`}
          ></div>
        </div>
      );
    }

    function TestForm({ questionData, onSubmit }) {
      const [selected, setSelected] = useState(null);

      return (
        <form onSubmit={(e) => { e.preventDefault(); onSubmit(selected); }} aria-
label="Форма тестування">
          <fieldset className="mb-4">
            <legend className="text-lg font-semibold mb-
2">{questionData.question}</legend>
            {questionData.options.map((option, idx) => (
              <label key={idx} className="block mb-2">
                <input
                  type="radio"
                  name="answer"
                  value={idx}
                  checked={selected === idx}
                  onChange={() => setSelected(idx)}
                  className="mr-2"
                  aria-checked={selected === idx}
                />
                {option}
              </label>
            ))}
          </fieldset>
          <button
            type="submit"
            disabled={selected === null}
            className="bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700
disabled:opacity-50"
          >
```

```
            Надіслати
          </button>
        </form>
      );
    }

    function HomePage() {
      return (
        <main className="p-4 max-w-xl mx-auto">
          <LessonCard title="Основи першої допомоги" description="Короткий вступ до тематики курсу" />
        </main>
      );
    }

    function CoursePage() {
      return (
        <main className="p-4 max-w-xl mx-auto space-y-6">
          <VideoPlayer src="https://www.youtube.com/embed/BdGHk8FqJ6k" title="Вступне відео" />
          <ProgressBar progress={60} />
          <TestForm
            questionData={{
              question: "Що таке домедична допомога?",
              options: [
                "Медична допомога лікарем",
                "Допомога до прибуття медиків",
                "Операція",
                "Щеплення"
              ]
            }}
            onSubmit={(res) => alert(`Обрано варіант: ${res}`)}
          />
        </main>
      );
    }

    export default function App() {
      return (
        <Router>
          <Header />
          <Routes>
            <Route path="/" element={<HomePage />} />
            <Route path="/course" element={<CoursePage />} />
          </Routes>
          <Footer />
        </Router>
      );
    }
```

Додаток Г

React-компонент із ARIA-атрибутами
(фрагмент тестового запитання)

```
import { useState } from "react";

function Question() {
  const [selected, setSelected] = useState(null);
  const [answered, setAnswered] = useState(false);
  const correctAnswer = 2;

  const answers = [
    "Оцінити ситуацію та викликати допомогу",
    "Почати СЛР без оцінки дихання",
    "Перевірити свідомість і дихання постраждалого",
    "Підняти ноги постраждалого"
  ];

  const handleSubmit = () => {
    setAnswered(true);
  };

  return (
    <div role="form" aria-labelledby="question-label">
      <h2 id="question-label">Що слід зробити перед наданням домедичної допомоги?</h2>

      <ul role="radiogroup" aria-label="Варіанти відповіді">
        {answers.map((answer, index) => (
          <li key={index}>
            <label>
              <input
                type="radio"
                name="answer"
                value={index}
                checked={selected === index}
                onChange={() => setSelected(index)}
                role="radio"
                aria-checked={selected === index}
              />
              {answer}
            </label>
          </li>
        ))}
      </ul>

      <button onClick={handleSubmit} disabled={selected === null}>
        Перевірити
      </button>

      {answered && (
```

```
        <div
          role="alert"
          aria-live="assertive"
          style={{ marginTop: "1rem", fontWeight: "bold" }}
        >
          {selected === correctAnswer
            ? "Правильна відповідь ✅"
            : "Неправильна відповідь ❌"}
        </div>
      )}
    </div>
  );
}

export default Question;
```

Додаток Д

Фрагмент коду для реалізації доступного навігаційного меню з ARIA на React

```
import { useState } from "react";

function NavMenu() {
  const [isOpen, setIsOpen] = useState(false);
  const toggleMenu = () => setIsOpen((prev) => !prev);
  return (
    <nav aria-label="Головна навігація">
      <button
        onClick={toggleMenu}
        aria-expanded={isOpen}
        aria-controls="main-menu"
        aria-label={isOpen ? "Закрити меню" : "Відкрити меню"}
      >
        ☰ Меню
      </button>
      <ul
        id="main-menu"
        role="menu"
        style={{
          display: isOpen ? "block" : "none",
          listStyle: "none",
          padding: 0,
          marginTop: "1rem",
        }}
      >
        <li role="none">
          <a role="menuitem" href="#home">Головна</a>
        </li>
        <li role="none">
          <a role="menuitem" href="#courses">Курси</a>
        </li>
        <li role="none">
          <a role="menuitem" href="#tests">Тести</a>
        </li>
        <li role="none">
          <a role="menuitem" href="#certification">Сертифікація</a>
        </li>
        <li role="none">
          <a role="menuitem" href="#about">Про проєкт</a>
        </li>
      </ul>
    </nav>
  );
}

export default NavMenu;
```

Додаток Е

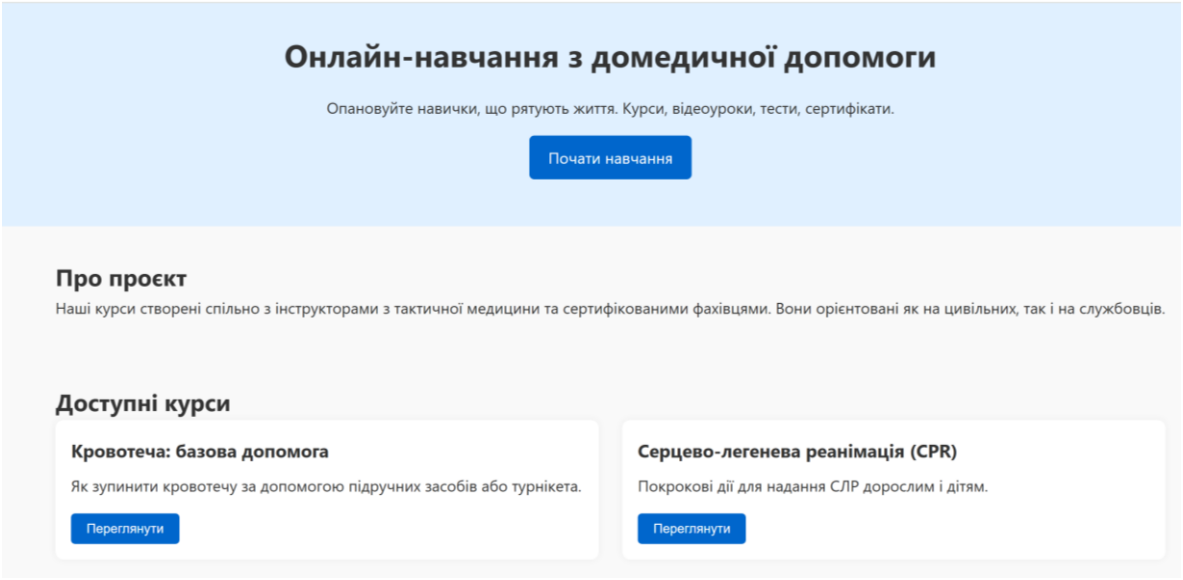


Рисунок 1 – Вигляд головного вікна прототипу вебдодатку

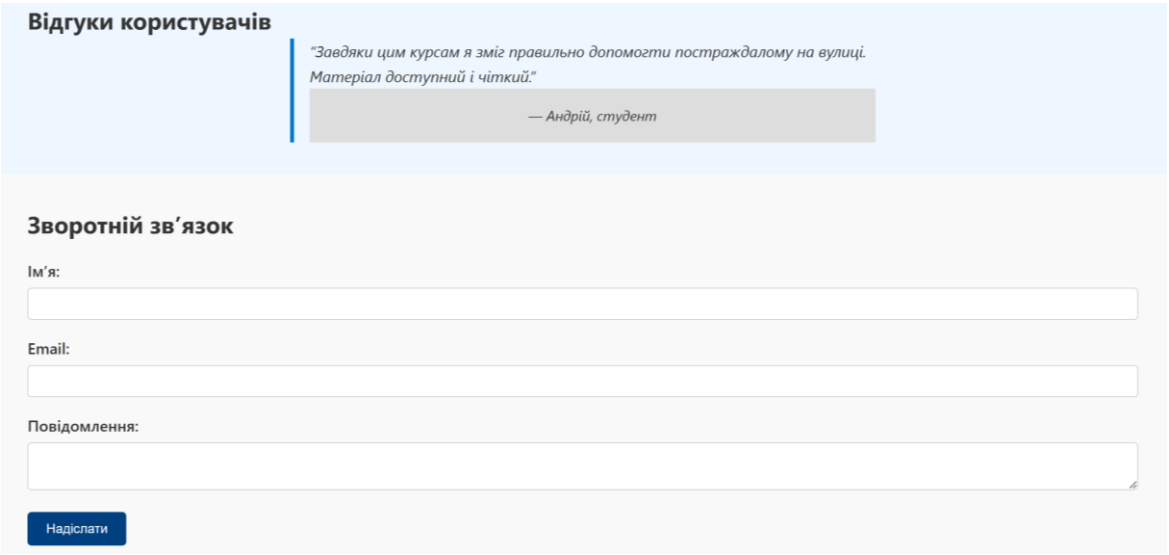


Рисунок 2 – Онлайн-форма для надання зворотнього зв'язку про навчанння

Фрагмент коду HTML для базової структури прототипу веб-додатку

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <meta name="description" content="Онлайн-курси першої домедичної допомоги та тактичної медицини з відеоуроками, тестами та сертифікацією." />
  <meta name="theme-color" content="#004080" />
  <title>Курси першої допомоги онлайн</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
  <!-- Header -->
  <header role="banner">
    <div class="container">
      <h1 class="logo">FirstAidLearn</h1>
      <nav role="navigation" aria-label="Основна навігація">
        <ul class="nav-list">
          <li><a href="#about">Про нас</a></li>
          <li><a href="#courses">Курси</a></li>
          <li><a href="#feedback">Відгуки</a></li>
          <li><a href="#contact">Контакти</a></li>
        </ul>
      </nav>
    </div>
  </header>

  <section class="hero" aria-labelledby="hero-title">
    <div class="container">
      <h2 id="hero-title">Онлайн-навчання з домедичної допомоги</h2>
      <p>Опановуйте навички, що рятують життя. Курси, відеоуроки, тести, сертифікати.</p>
      <a href="#courses" class="btn-primary">Почати навчання</a>
    </div>
  </section>

  <section id="about" class="about-section">
    <div class="container">
      <h2>Про проєкт</h2>
      <p>Наші курси створені спільно з інструкторами з тактичної медицини та сертифікованими фахівцями. Вони орієнтовані як на цивільних, так і на службовців.</p>
    </div>
  </section>
  <section id="courses" class="courses-section" aria-labelledby="courses-title">
    <div class="container">
      <h2 id="courses-title">Доступні курси</h2>
      <div class="course-grid">
        <article class="lesson-card" aria-labelledby="lesson1-title">
          <h3 id="lesson1-title">Кровотеча: базова допомога</h3>
          <p>Як зупинити кровотечу за допомогою підручних засобів або турнікета.</p>
```

```
            <button onclick="startLesson()">Переглянути</button>
          </article>
          <article class="lesson-card" aria-labelledby="lesson2-title">
            <h3 id="lesson2-title">Серцево-легенева реанімація (CPR)</h3>
            <p>Покрокові дії для надання СЛР дорослим і дітям.</p>
            <button>Переглянути</button>
          </article>
        </div>
      </div>
    </section>
    <section id="feedback" class="testimonials-section">
      <div class="container">
        <h2>Відгуки користувачів</h2>
        <blockquote>
          <p>“Завдяки цим курсам я зміг правильно допомогти постраждалому на вулиці.
Матеріал доступний і чіткий.”</p>
          <footer>— Андрій, студент</footer>
        </blockquote>
      </div>
    </section>
    <section id="contact" class="contact-section">
      <div class="container">
        <h2>Зворотній зв’язок</h2>
        <form id="contact-form" onsubmit="submitContact(event)">
          <label for="name">Ім’я:</label>
          <input type="text" id="name" required />

          <label for="email">Email:</label>
          <input type="email" id="email" required />

          <label for="message">Повідомлення:</label>
          <textarea id="message" required></textarea>

          <button type="submit">Надіслати</button>
        </form>
      </div>
    </section>

    <footer>
      <div class="container">
        <p>© 2025 FirstAidLearn. Усі права захищено.</p>
        <small>Розроблено в межах кваліфікаційної роботи, спеціальність 126 ICT</small>
      </div>
    </footer>

    <script src="script.js"></script>
  </body>
  </html>
```

Фрагмент коду JavaScript прототипу проекту
(створення форми зворотнього відгуку)

```
function startLesson() {
  alert("Переходимо до модуля з відеоуроком...");
  // Тут у повній версії буде навігація або SPA-логіка
}

function submitContact(event) {
  event.preventDefault();
  const name = document.getElementById("name").value.trim();
  const email = document.getElementById("email").value.trim();
  const message = document.getElementById("message").value.trim();

  if (!name || !email || !message) {
    alert("Будь ласка, заповніть усі поля.");
    return;
  }

  alert(`Дякуємо, ${name}! Ваше повідомлення надіслано.`);
  document.getElementById("contact-form").reset();
}
```