ПРИКЛАДНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК,
БЕЗПЕКИ ТА МАТЕМАТИКИ, 2026, № 6

УДК 004.75:004.4

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ НАВЧАННЯМ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ NEXT.JS

Гришанович Т.О. (ORCID: 0000-0002-3595-6964),
Бондарчук В.О.
Волинський національний університет імені Лесі Українки

DESIGN AND IMPLEMENTATION OF A LEARNING MANAGEMENT SYSTEM USING THE NEXT.JS FRAMEWORK

Hryshanovych T.O., Bondarchuk V.O.
Lesya Ukrainka Volyn National University

Abstract. Extended English abstract up to 500 words.The article is devoted to the design and implementation of a Learning Management System (LMS) intended for use in a foreign language school, including the delivery of online classes. The relevance of the study is обусловлена by the growing demand for distance education and the increasing requirements for the quality and performance of web-based platforms. Modern approaches to client-side web development are analyzed, and a comparative review of the Angular, React, and Vue frameworks is conducted. The choice of Next.js as the foundation of the LMS is justified by its support for hybrid rendering, performance optimization, and improved SEO. The system architecture, user roles, core functional modules, and testing methods are examined. The obtained results confirm the effectiveness of the selected technological solutions for developing scalable educational platforms.
Keywords: online school, web interface, SPA, PWA, Next.js, JavaScript frameworks.

Вступ. Стрімкий розвиток цифрових технологій зумовлює активне поширення дистанційних форм навчання. Різноманітні онлайн-школи, навчальні сервіси та інтерактивні освітні платформи відкривають можливості для здобуття знань незалежно від географічного розташування чи часових обмежень. Такий формат навчання є особливо зручним для учнів, студентів, а також фахівців, зацікавлених у безперервному професійному розвитку.

Водночас результативність використання подібних рішень значною мірою визначається якістю реалізації інтерфейсу користувача. Ефективно спроєктований інтерфейс відіграє ключову роль у забезпеченні комфортної взаємодії між користувачем і системою. Він повинен бути логічно зрозумілим, візуально привабливим, коректно працювати на різних типах пристроїв та відповідати актуальним принципам UI/UX-дизайну. Особливу увагу слід приділяти потребам різних категорій користувачів — від школярів і студентів до викладачів та дорослих слухачів, рівень цифрових навичок яких може суттєво відрізнятися. Не менш важливим є врахування вимог доступності для осіб з особливими освітніми потребами з метою забезпечення інклюзивного навчального середовища.

Актуальність досліджуваної проблематики зумовлена зростаючим попитом на сучасні освітні платформи, що здатні гармонійно інтегруватися в навчальний процес без порушення його логіки та структури. Постійне збільшення кількості користувачів онлайн-освіти актуалізує завдання створення інтерфейсних рішень, які покращують користувацький досвід, стимулюють навчальну мотивацію та підвищують ефективність засвоєння навчального матеріалу. Раціональна організація контенту, інтуїтивна навігація та використання інтерактивних компонентів суттєво впливають на рівень залученості учасників освітнього процесу.

Метою цієї роботи є проєктування та реалізація вебінтерфейсу онлайн-школи, який забезпечить зручну та результативну взаємодію користувачів із системою, сприятиме

підвищенню якості дистанційного навчання та автоматизації ключових організаційних процесів.

Завданням дослідження є створення інформаційної системи, яка дозволяє централізовано адмініструвати навчальні групи, формувати розклад занять з автоматичним додаванням подій до календаря викладачів та студентів, інтегрувати проведення онлайн-уроків за допомогою сервісу Zoom, створювати та перевіряти домашні завдання, відображати результати навчання в електронних таблицях успішності, а також забезпечити зручну комунікацію між учнями, викладачами та адміністраторами через систему повідомлень і чатів.

Методологія дослідження. Методологічну основу дослідження становить комплексний підхід, що поєднує теоретичні та практичні методи аналізу, проєктування й експериментальної перевірки програмних рішень у сфері веброзробки освітніх платформ. Дослідження спрямоване на обґрунтування архітектурних і технологічних рішень для розробки онлайн-платформи школи з урахуванням вимог до продуктивності, масштабованості та зручності використання.

На першому етапі застосовано метод аналізу наукових джерел, стандартів і сучасних публікацій, що стосуються клієнтських вебтехнологій, архітектур SPA та PWA, а також концепцій роботи з DOM. Це дозволило сформувати теоретичну базу дослідження та визначити актуальні проблеми, пов'язані з продуктивністю інтерфейсів і організацією освітнього процесу в онлайн-середовищі.

Другий етап включав порівняльний аналіз JavaScript-фреймворків і бібліотек (React, Angular, Vue, Next.js) за критеріями продуктивності, підтримки серверного рендерингу, SEO-оптимізації, інтеграції сторонніх сервісів і придатності для реалізації навчальних курсів програмування. За результатами аналізу висунуто гіпотезу про те, що використання Next.js у поєднанні з компонентно-орієнтованою архітектурою забезпечує підвищення ефективності вебінтерфейсу та покращення користувацького досвіду в онлайн-навчанні.

На третьому етапі застосовано метод проєктування програмних систем, який передбачав розробку архітектури вебзастосунку, структури сторінок, моделі ролей і прав доступу, а також інтеграцію ключових функціональних модулів (автентифікація, календар, чати, навчальні матеріали, оцінювання). Практична реалізація платформи слугувала експериментальною базою для перевірки висунутої гіпотези.

Для оцінки якості розробленого інтерфейсу використано методи програмного тестування, зокрема автоматизоване (unit-, integration- та E2E-тестування) і ручне тестування за участю QA-фахівця. Це дозволило перевірити коректність функціонування системи, її стабільність, адаптивність і зручність використання на різних пристроях та в різних браузерах.

Узагальнення отриманих результатів здійснювалося за допомогою методів аналізу та синтезу, що дало змогу сформулювати висновки щодо доцільності обраних технологічних рішень і їхньої ефективності.

Результати дослідження та їхнє обговорення. Із початком еволюції вебтехнологій однією з ключових цілей розробників стало наближення вебзастосунків за рівнем візуального подання та користувацького досвіду до нативних настільних програм. У межах цього прагнення було запропоновано низку технологічних рішень, орієнтованих на підвищення інтерактивності та природності взаємодії з користувачем, серед яких слід виокремити IFrames [5], Java-аплети [1], Adobe Flash [4] та Microsoft Silverlight [6]. Незважаючи на відмінності в архітектурі та принципах реалізації, усі зазначені підходи мали спільне завдання: розширити функціональні можливості вебзастосунків до рівня традиційного програмного забезпечення, зберігаючи при цьому переваги браузерної кросплатформності.

На початку 2000-х років у сфері веброзробки сформувався новий концептуальний підхід, пов'язаний із впровадженням технології AJAX [7]. Перші реалізації ґрунтувалися на використанні компонента ActiveX у браузері Microsoft Internet Explorer, який забезпечував асинхронний обмін даними між клієнтом і сервером. Згодом цей механізм було уніфіковано у вигляді стандартизованого програмного інтерфейсу XMLHttpRequest (XHR), що отримав підтримку з боку основних браузерів. Поєднання XHR із мовою JavaScript, а також технологіями HTML і CSS дало змогу створювати більш динамічні та продуктивні вебсторінки. Запровадження AJAX (Asynchronous JavaScript and XML) дозволило здійснювати серверні запити без повного перезавантаження сторінки, оперативно оновлювати вміст шляхом маніпуляції об'єктною моделлю документа (DOM) та змінювати візуальне оформлення за допомогою CSS, що суттєво покращило користувацький досвід.

Подальший розвиток зазначених принципів сприяв формуванню концепції односторінкових застосунків (Single Page Application, SPA), у межах якої ідеї AJAX було поширено на всю архітектуру вебзастосунку. SPA являє собою вебзастосунок, у якому керування маршрутизацією здійснюється на клієнтській стороні засобами JavaScript, а не серверною логікою [9]. Такий підхід усуває потребу в повному перезавантаженні сторінок під час навігації, а також мінімізує повторне завантаження спільних елементів інтерфейсу, зокрема заголовків, бічних панелей і футерів сторінок.

Після початкового завантаження застосунку всі необхідні компоненти для формування та відображення інтерфейсних представлень стають доступними в браузері користувача. У разі потреби нові представлення генеруються локально та динамічно інтегруються в DOM за допомогою JavaScript без необхідності оновлення сторінки. За таких умов відповідальність за поєднання даних і шаблонів інтерфейсу переноситься із серверної частини на клієнтську, що відповідає сучасним тенденціям вебархітектури. Використання SPA супроводжується застосуванням усталених шаблонів проєктування та методів розробки, які сприяють підвищенню продуктивності створення програмного забезпечення, спрощенню супроводу коду та скороченню термінів розробки.

Водночас проєктування односторінкових застосунків немає єдиного універсального підходу. Різноманіття сучасних концепцій, а також наявність великої кількості конкуруючих бібліотек і фреймворків, зумовлюють складність вибору оптимальної технологічної платформи для реалізації конкретного SPA-проєкту.

Прогресивні вебзастосунки (Progressive Web Applications, PWA) розглядаються як сучасний етап еволюції вебтехнологій, що поєднує переваги вебсайтів і нативних мобільних застосунків [6]. Використання можливостей PWA забезпечує користувацький досвід, наближений до взаємодії з нативним програмним забезпеченням, без необхідності встановлення застосунку з офіційних плеймаркетів. Зокрема, такі вебрішення підтримують автономний режим роботи після початкового завантаження, дозволяючи переглядати контент без активного підключення до мережі Інтернет, а також реалізують інтерактивні сервіси за рахунок механізмів кешування, push-сповіщень і сервіс-воркерів. Сукупність зазначених можливостей, реалізованих на основі стандартів HTML5, значною мірою зменшує відмінності між веб- і нативними застосунками, особливо в мобільному середовищі, забезпечуючи низькі затримки передавання даних та спрощені процедури публікації і доступу користувачів.

Слід зазначити, що PWA не є окремою мовою програмування або фреймворком, а являє собою набір усталених практик і технологічних підходів, які використовуються для формування високоякісного користувацького досвіду. Основними складовими прогресивних вебзастосунків є Service Worker, архітектурна модель App Shell [13], маніфест вебзастосунку та механізм push-повідомлень, спільне використання яких забезпечує характерні властивості PWA. Такі застосунки можуть бути встановлені на

головний екран пристрою безпосередньо з браузера, що надає можливість запуску у повноекранному режимі та скорочує час доступу до функціоналу. Завдяки ефективному використанню кешування PWA демонструють високу швидкодію навіть за умов нестабільного або обмеженого мережевого з'єднання, підтримуючи роботу в офлайн-режимі. Додатковою перевагою є можливість надсилання релевантних push-сповіщень, що позитивно впливає на рівень залученості користувачів.

Сфера застосування прогресивних вебзастосунків активно розширюється, зокрема в електронній комерції, корпоративних інформаційних системах, новинних ресурсах та інших галузях. Однією з ключових переваг PWA є їхня доступність у більшості сучасних браузерів, що забезпечує охоплення широкої аудиторії користувачів незалежно від платформи. Механізми автоматичного оновлення гарантують, що користувачі отримують актуальний функціонал і вміст одразу після підключення до мережі Інтернет, при цьому основна оболонка та закешовані дані завантажуються з локального сховища пристрою.

Окрему роль у підвищенні повторної взаємодії користувачів із застосунком відіграють push-повідомлення, реалізовані за допомогою інтерфейсів Notifications API та Push API. Вони дозволяють серверній частині ініціювати надсилання повідомлень, які відображаються на пристрої користувача у форматі, ідентичному до сповіщень нативних застосунків, що сприяє зростанню показників повернення користувачів.

З точки зору доступності та адаптивності, прогресивні вебзастосунки зберігають універсальну структуру вебсайтів, яка коректно відображається на мобільних, планшетних і настільних пристроях. Адаптація до різних розмірів екранів досягається завдяки використанню адаптивного дизайну та медіа-запитів CSS3. Наявність маніфесту вебзастосунку, стандартизованого W3C, дає змогу пошуковим системам ідентифікувати PWA як окремий застосунок, що підвищує їхню видимість у результатах пошуку порівняно з нативними програмами, доступними виключно через плеймаркети. Використання сервіс-воркерів дозволяє ефективно обробляти втрату мережевого з'єднання, розглядаючи її не як критичну помилку, а як передбачуваний сценарій, що може бути оброблений без суттєвого погіршення продуктивності.

Обов'язковою умовою функціонування прогресивних вебзастосунків є використання захищеного HTTPS-з'єднання та SSL-сертифікатів, що забезпечує захист від атак типу «людина посередині», несанкціонованого доступу до облікових даних і підміни вмісту.

Як продукт вебсередовища, PWA характеризуються нижчою вартістю розробки та підтримки порівняно з нативними застосунками, а також зменшенням витрат на залучення користувачів, оскільки завжди надають актуальну версію функціоналу та контенту. Водночас дана технологія має певні обмеження. Серед основних недоліків слід виокремити проблеми кросбраузерної сумісності, що проявляються в обмеженій підтримці окремих API та доступу до апаратних ресурсів пристроїв у деяких браузерах. Крім того, на сьогодні відсутня повноцінна реалізація механізмів міжзастосункової автентифікації, що також обмежує можливості інтеграції PWA в складні програмні екосистеми.

Об'єктна модель документа (Document Object Model, DOM) є базовим поняттям клієнтської веброзробки [8]. Водночас безпосередня робота з DOM характеризується підвищеною складністю та низькою зручністю, а також може спричиняти суттєві проблеми з продуктивністю. У відповідь на ці обмеження в сучасних вебфреймворках, зокрема Angular, React.js та Vue.js, були сформовані дві ключові концепції оптимізації роботи з DOM — Shadow DOM та Virtual DOM [8].

DOM являє собою прикладний програмний інтерфейс для HTML- та XML-документів, який визначає їх логічну ієрархічну структуру та забезпечує можливість

доступу до елементів і керування ними. За допомогою мови JavaScript розробник може отримувати доступ до елементів об'єктної моделі документа та змінювати їхній стан і властивості. Основною метою впровадження DOM було створення універсального стандартизованого інтерфейсу програмування, придатного для використання різними мовами програмування та в різних програмних середовищах. Модифікація DOM охоплює операції додавання, видалення або зміни елементів вебсторінки, а також призначення їм певної поведінки.

Кожен веббраузер надає глобальний об'єкт window, який містить набір властивостей і методів для взаємодії з середовищем виконання. Однією з його ключових властивостей є об'єкт document, через який реалізується доступ до елементів DOM та їх програмне керування.

Shadow DOM застосовується для побудови компонентно-орієнтованих вебзастосунків і не відображає повну об'єктну модель документа. Його доцільно розглядати як окреме піддерево або локальну об'єктну модель, пов'язану з конкретним елементом. Таке піддерево називається тіньовим деревом (shadow tree), а елемент, до якого воно прикріплюється, — тіньовим хостом (shadow host). На відміну від класичного DOM, елементи Shadow DOM мають власну ізольовану область видимості, що відокремлена від глобальної структури документа.

Ключовою перевагою Shadow DOM є ізоляція як структури, так і стилів компонента. Усі елементи та CSS-правила, визначені всередині тіньового дерева, не впливають на глобальний DOM і не зазнають впливу ззовні. Це значно спрощує процес стилізації, зменшуючи потребу в складних іменах класів і запобігаючи конфліктам стилів. Такий підхід дозволяє розглядати вебзастосунок як сукупність незалежних компонентів, а не як єдиний монолітний об'єкт. Крім того, Shadow DOM може позитивно впливати на продуктивність, оскільки браузер чітко визначає локальні області, які потребують оновлення.

Virtual DOM є альтернативною концепцією оптимізації, що використовується у фреймворках React.js та Vue.js. Вона передбачає зберігання в пам'яті віртуальної копії DOM-дерева. Усі зміни спочатку застосовуються до цієї копії, після чого виконується порівняння з попереднім станом для виявлення відмінностей. На основі цього аналізу браузер оновлює лише ті елементи реального DOM, які зазнали змін, уникаючи повного повторного рендерингу сторінки. Такий механізм істотно підвищує продуктивність інтерфейсних бібліотек, особливо у випадку великої кількості динамічних оновлень. Крім того, Virtual DOM дозволяє групувати декілька змін і застосовувати їх одночасно, що додатково зменшує накладні витрати на перерендеринг.

Таким чином, і Shadow DOM, і Virtual DOM спрямовані на розв'язання проблем продуктивності під час роботи з об'єктною моделлю документа, однак реалізують цю мету різними способами. Virtual DOM створює повну віртуальну копію DOM-дерева, тоді як Shadow DOM формує ізольовані локальні піддерева, пов'язані з окремими елементами, що мають власну область дії.

Після аналізу архітектурної організації вебзастосунку для онлайн-школи доцільно перейти до розгляду програмних засобів і технологій, що можуть бути використані для відповідного функціоналу.

У галузі програмної інженерії відсутнє універсальне та формалізоване визначення критеріїв, за якими програмний продукт однозначно можна віднести до фреймворків, що часто призводить до їх ототожнення з бібліотеками. Водночас між цими підходами існують принципові відмінності, пов'язані з роллю та способом використання програмного коду в процесі розробки.

Бібліотека, як правило, являє собою сукупність попередньо реалізованих компонентів, зокрема класів, функцій, процедур, сценаріїв і конфігурацій, які можуть

бути інтегровані в уже наявний програмний проєкт. Її основним призначенням є скорочення витрат часу на реалізацію типових задач, оскільки значна частина базових алгоритмів і функціональних можливостей вже була опрацьована іншими розробниками. Використання бібліотек дозволяє зосередити увагу на прикладній логіці застосунку, однак залучення сторонніх рішень може супроводжуватися певними ризиками, зокрема щодо безпеки, сумісності або підтримки.

Фреймворки, на відміну від бібліотек, надають комплексний набір інструментів і визначають загальну архітектуру програмного забезпечення. Вони беруть на себе керування основними аспектами розробки, пропонуючи готові механізми для маршрутизації запитів, управління станом застосунку, взаємодії та композиції компонентів тощо. Окрім цього, фреймворки сприяють оптимізації процесу розробки, впроваджуючи рекомендовані підходи та найкращі практики щодо організації структури застосунку і створення шаблонних рішень, що підвищує узгодженість та підтримуваність коду [12].

У таблиці нижче наведено результати порівняння найпоширеніших фреймовркв для веброзробки.

Таблиця 1.

Порівняльна характеристика сучасних JavaScript-фреймворків і платформ

Критерій порівняння	**Angular**	**React**	**Vue**	**Next.js**
Тип технології	Повнофункціональ-ний фреймворк	Бібліотека / фреймворк для побудови UI	Прогресивний фреймворк	Фреймворк на базі React
Архітектурний підхід	Строга, формалізована архітектура	Гнучка, компонентна	Компроміс між структурованістю та гнучкістю	Гібридна архітектура з підтримкою різних стратегій рендерингу
Вбудовані інструменти	Повний набір (маршрутизація, DI, сервіси тощо)	Обмежені, часто потребує сторонніх бібліотек	Основні інструменти + плагіни	Розширений набір: SSR, SSG, оптимізація ресурсів
Керування станом	Інтегровані механізми	Незмінний стан через setState() , Redux за замовчуванням	Vuex як вбудоване рішення	Використовує підходи React (Redux, Context API тощо)
Шаблони представлення	HTML-шаблони з розширеннями	JSX	Декларативні шаблони з підтримкою JS	JSX (успадковано від React)
Складність опанування	Висока через велику кількість концепцій	Середня, потребує розуміння JSX і станів	Низька, вважається найпростішою для вивчення	Поступове освоєння від базових до складних концепцій
Масштабованість	Висока, орієнтована на великі команди	Висока, залежить від архітектурних рішень	Достатня для більшості проєктів	Висока, орієнтована на складні та корпоративні системи
Продуктивність	Висока, але з більшими накладними витратами	Висока завдяки Virtual DOM	Висока, оптимізована робота з DOM	Дуже висока завдяки SSR, SSG та гібридному рендерингу
SEO-оптимізація	Обмежена без додаткових рішень	Обмежена в класичній моделі	Обмежена	Висока завдяки серверному та статичному рендерингу
Типові сфери застосування	Корпоративні та великі проєкти	Інтерактивні вебзастосунки	Середні та невеликі проєкти	E-commerce, корпоративні сайти, SPA, блоги, портфоліо
Підтримка та екосистема	Підтримка Google	Активно використовується Facebook та спільнотою	Активна спільнота, плагіни (Quasar, Vuetify)	Підтримка Vercel, використання провідними компаніями

На основі проведеного аналізу для розробки було обрано технологію Next.js. Нижче описано її основні переваги.

Платформа Node.js, зокрема у поєднанні з фреймворком Next.js, забезпечує високу продуктивність вебзастосунків завдяки ефективній роботі з ресурсами та оптимізованому виконанню серверних операцій. Однією з ключових переваг є автоматичний розподіл коду та завантаження лише необхідних компонентів, що зменшує час відгуку системи та прискорює відображення сторінок.

Використання серверного рендерингу (SSR) у середовищі Node.js сприяє покращенню пошукової оптимізації, оскільки клієнту та пошуковим системам надається попередньо згенерований HTML-контент. Це позитивно впливає на індексацію вебресурсів і знижує показники відмов.

Додатковою перевагою є зручна інтеграція з зовнішніми сервісами та API, що забезпечує гнучке розширення функціональності без суттєвих втрат продуктивності. Вбудована підтримка TypeScript підвищує надійність програмного коду, зменшує кількість помилок і спрощує супровід великих проєктів, що є важливим фактором у командній та корпоративній розробці.

Розроблення програмного засобу для школи вивчення іноземних мов передбачає створення інтегрованої вебплатформи, здатної забезпечити повний цикл освітнього процесу в онлайн-середовищі. Ключовою вимогою є підтримка взаємодії між основними групами користувачів — адміністраторами, викладачами та студентами — із забезпеченням ефективної комунікації, управління навчальними групами, планування занять і контролю результатів навчання.

Функціональність адміністративного рівня охоплює керування обліковими записами користувачів, організацію навчальних груп, формування розкладу, розподіл викладачів за курсами, а також доступ до узагальненої аналітики щодо перебігу навчального процесу та успішності здобувачів освіти. Для викладачів передбачається персоналізоване робоче середовище з доступом до розкладу, навчальних матеріалів, результатів оцінювання та інструментів проведення інтерактивних занять і перевірки завдань.

Студентський модуль забезпечує доступ до розкладу, навчальних ресурсів, онлайн-занять і засобів комунікації з викладачами, а також до інформації про відвідуваність і навчальні досягнення. Контроль знань реалізується через автоматизоване формування підсумкових іспитів із використанням різних типів завдань, результати яких зберігаються в системі та доступні для аналізу.

Важливою характеристикою платформи є адаптивність і зручність інтерфейсу, що забезпечує коректну роботу на різних типах пристроїв. Загалом система інтегрує основні компоненти сучасного онлайн-навчання, зокрема реєстрацію, оплату, планування занять, проведення уроків, оцінювання, зворотний зв'язок і аналітику, що сприяє безперервності освітнього процесу та підвищенню ефективності навчання (рис 1).

У процесі розробки програмного забезпечення реалізовано механізм аутентифікації користувачів із підтримкою входу через обліковий запис Google (рис. 2). Такий підхід забезпечує захищену взаємодію із зовнішнім сервісом автентифікації, спрощує процедуру доступу до платформи та зменшує ризики, пов'язані з обробкою й зберіганням облікових даних. Використання сторонньої автентифікації відповідає сучасним вимогам до безпеки вебзастосунків і підвищує зручність користування системою.

Крім того, у системі впроваджено рольову модель керування доступом, яка забезпечує розмежування прав користувачів відповідно до їхніх функціональних обов'язків (рис. 3, 4). Адміністративний модуль дозволяє створювати додаткові ролі з

чітко визначеними повноваженнями, що підвищує гнучкість управління платформою та рівень інформаційної безпеки.

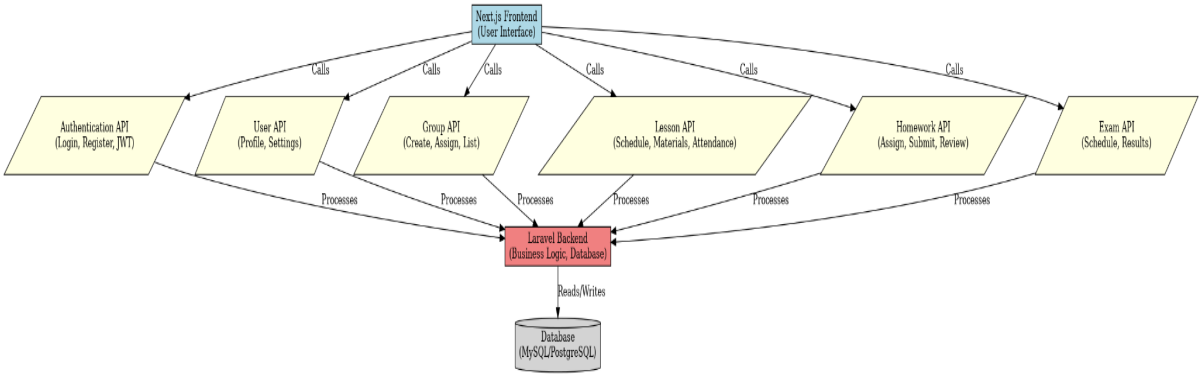

Рис. 1. UML-діаграма структури проєкту

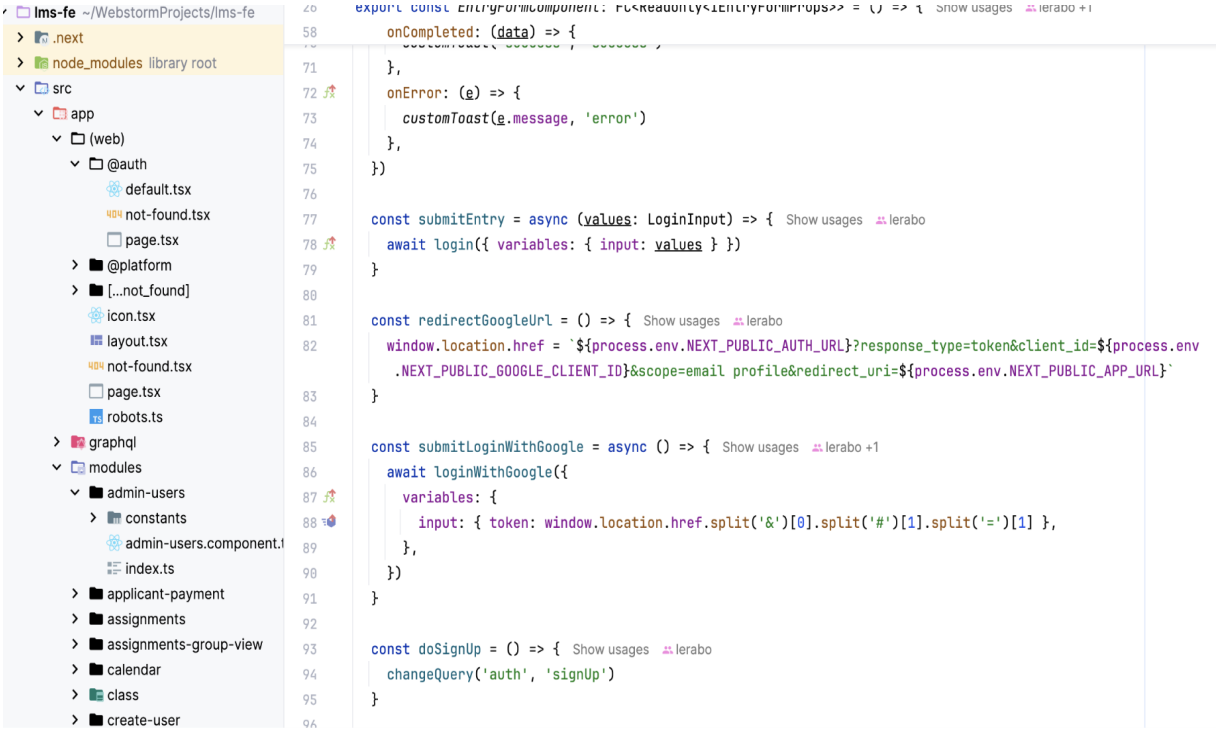

Рис. 2. Реалізація авторизації через Google

Для організації навчального процесу реалізовано інтерактивний календар подій, функціонально та візуально орієнтований на популярні календарні сервіси (рис. 5). Використання бібліотеки react-big-calendar забезпечило зручне відображення розкладу занять, підтримку динамічних подій і ефективну навігацію, що сприяє кращій структуризації освітньої діяльності.

З метою забезпечення оперативної взаємодії між учасниками освітнього процесу впроваджено систему індивідуальних і групових чатів. Для реалізації обміну повідомленнями в режимі реального часу застосовано технологію WebSocket, яка дозволяє здійснювати безперервну комунікацію без перезавантаження сторінок, підвищуючи ефективність і зручність спілкування (рис. 7).

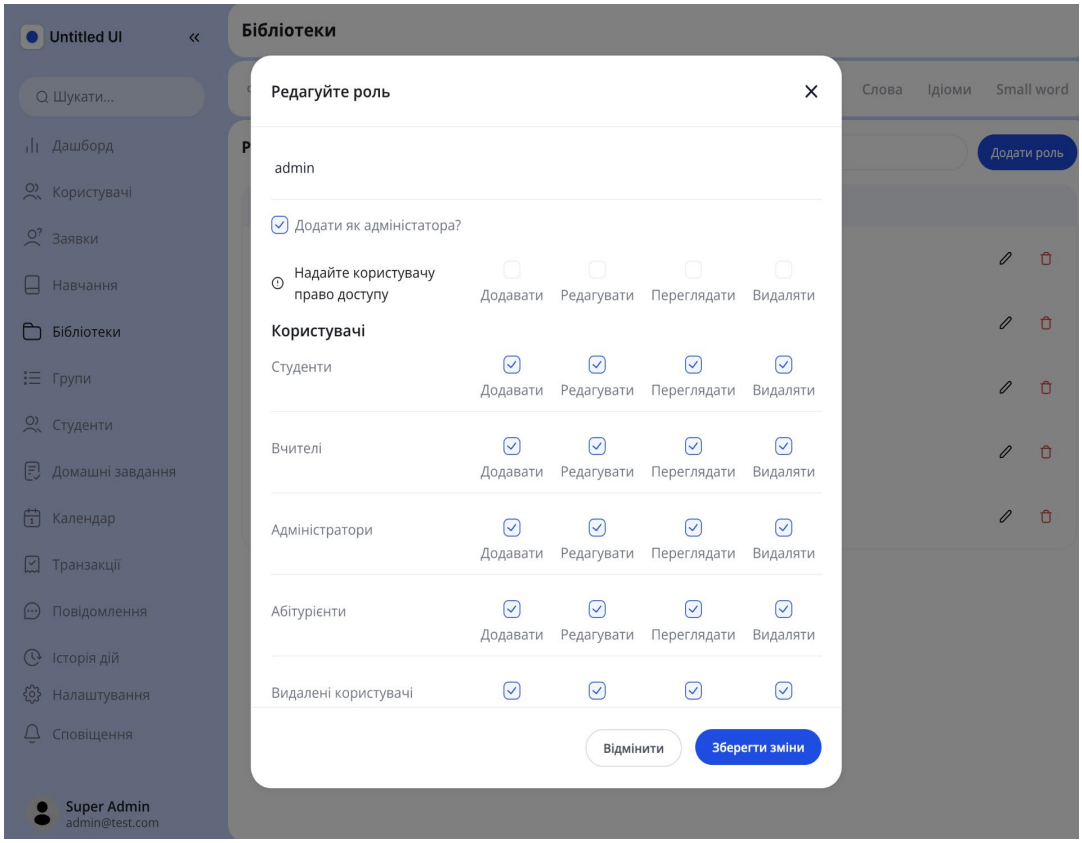

Рис. 3. Модальне вікно регулювання прав

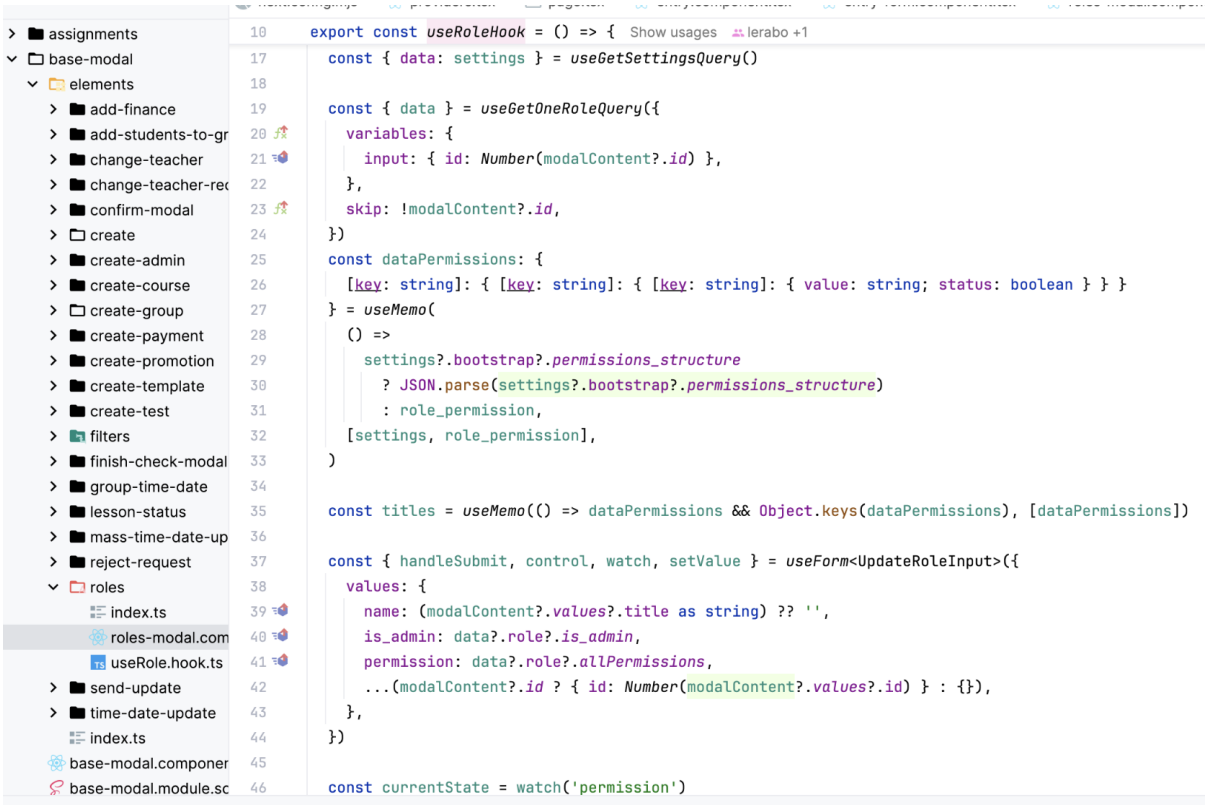

Рис. 4. Регулювання прав різних ролей

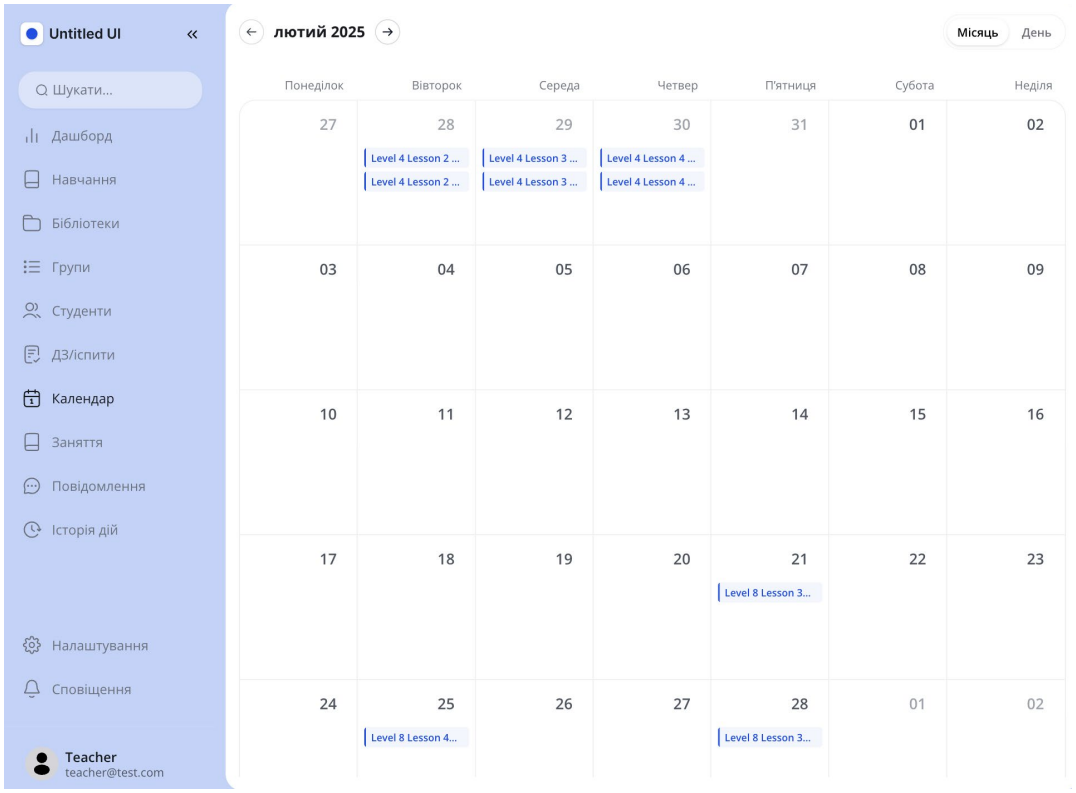

Рис. 5. Вигляд календаря

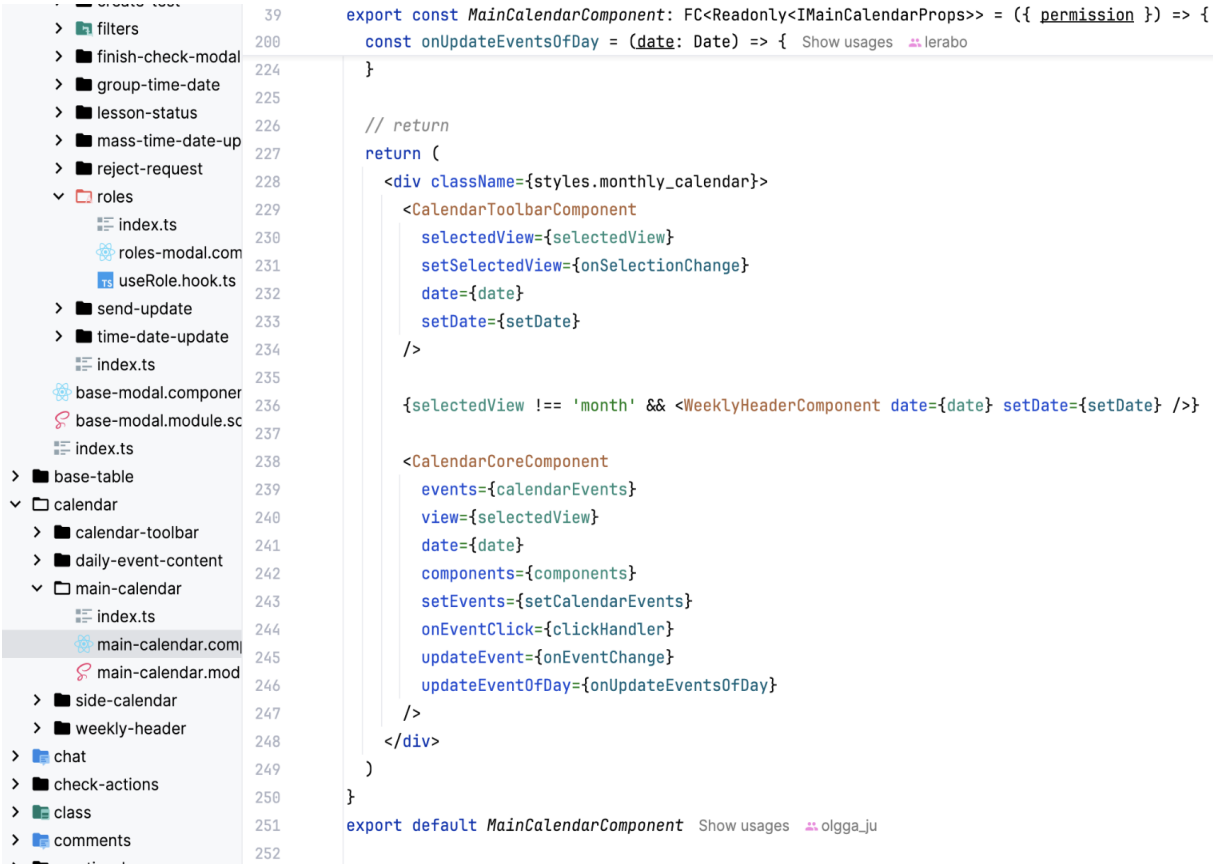

Рис. 6. Підключення бібліотеки react-big-calendar

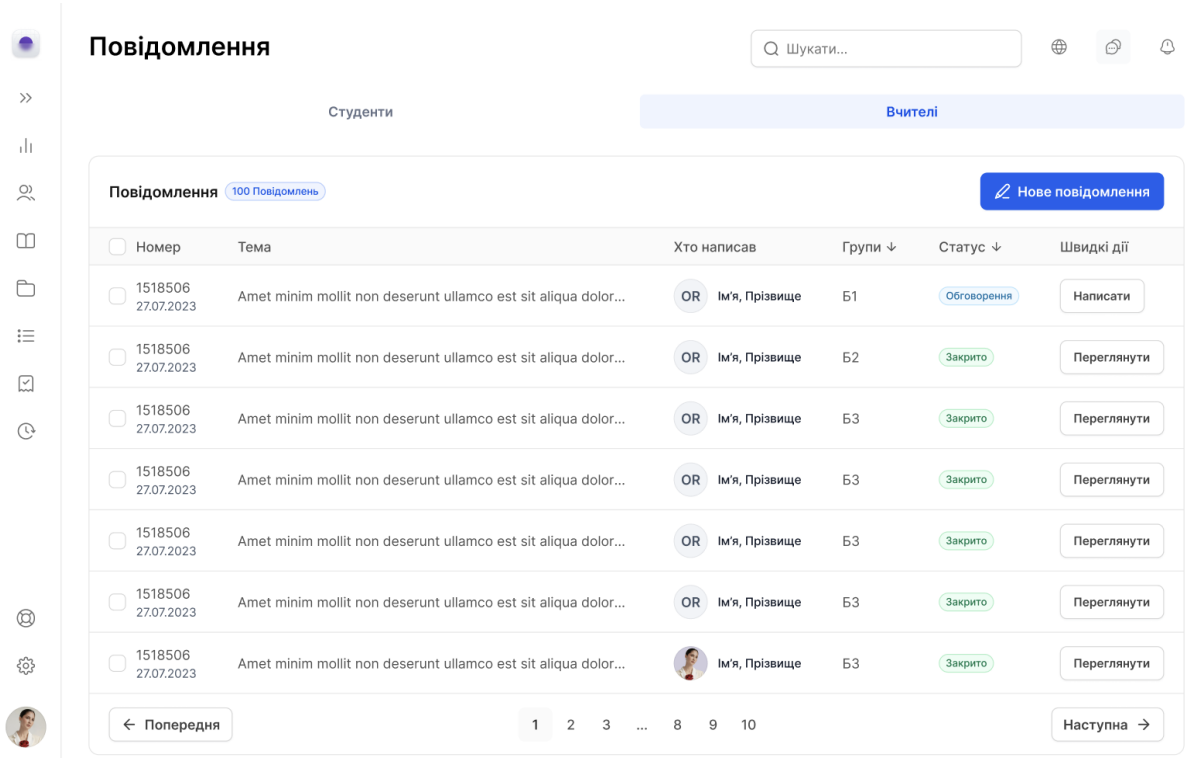

Рис. 7. Mockup загального чату

Розробка користувацького інтерфейсу здійснювалася з використанням бібліотеки NextUI, що забезпечило адаптивність, сучасний дизайн і високу продуктивність UI-компонентів. Кастомізація елементів інтерфейсу відповідно до проєктних макетів сприяла створенню цілісного та інтуїтивно зрозумілого користувацького середовища [2] (рис. 8).

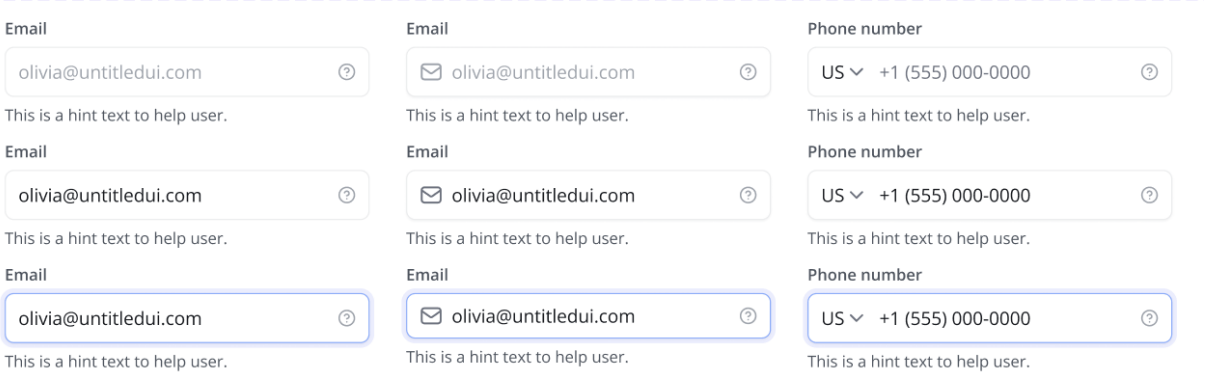

Рис. 8. Частина ui-kit

Таким чином, реалізована платформа поєднує функціональність, зручність і масштабованість, що дозволяє ефективно використовувати її для організації дистанційного навчання та забезпечує потенціал для подальшого розвитку з урахуванням зростаючих вимог користувачів.

У процесі розробки інтерфейсу онлайн-платформи було застосовано поєднання автоматизованого тестування та ручної перевірки QA-фахівцем для забезпечення стабільності, безпеки та зручності користування. Автоматизоване тестування включало

кілька ключових напрямів: юніт-тести за допомогою Jest та React Testing Library перевіряли функціональність окремих компонентів інтерфейсу, таких як кнопки, форми та інші інтерактивні елементи. Інтеграційні тести з використанням Cypress забезпечували коректну взаємодію між модулями, зокрема між формою входу та API авторизації. Для оцінки повного сценарію користувача — від реєстрації до проходження тестів і отримання результатів — застосовувалося E2E-тестування за допомогою Playwright та Cypress.

Ручне тестування QA-спеціалістом дозволило оцінити функціональність та зручність інтерфейсу з точки зору кінцевого користувача. Було перевірено реєстрацію, авторизацію, завантаження курсів, проходження тестів та інші основні функції. Окрема увага приділялася юзабіліті: оцінювалася логічність розташування елементів, інтуїтивність навігації та доступність платформи для користувачів різного віку. Крім того, проводилося кросбраузерне та кросплатформене тестування для перевірки коректного відображення сайту в Google Chrome, Mozilla Firefox, Safari та Microsoft Edge, а також на пристроях із Windows, macOS, iOS та Android. Для оцінки продуктивності системи під високим навантаженням використовували JMeter.

Результати тестування документувалися у Jira, після чого верифікувалися виправлені помилки. Поєднання автоматизованих і ручних методів дозволило забезпечити високу якість інтерфейсу, стабільність роботи платформи, безпеку обробки даних і зручність користування системою.

Висновки. У статті розглянуто сучасні підходи до проєктування та реалізації вебінтерфейсу онлайн-школи з урахуванням вимог до зручності користування, продуктивності та масштабованості. Проаналізовано еволюцію вебтехнологій від класичних динамічних сторінок до односторінкових і прогресивних вебзастосунків, а також обґрунтовано доцільність використання компонентно-орієнтованої архітектури. Досліджено особливості роботи з DOM, зокрема застосування Virtual DOM і Shadow DOM для підвищення ефективності інтерфейсів. На основі порівняльного аналізу сучасних JavaScript-фреймворків обрано Next.js як оптимальну платформу для розробки навчального вебзастосунку. Реалізована система забезпечує повний цикл онлайн-навчання, підтримує рольову модель доступу, інтеграцію зовнішніх сервісів і відповідає сучасним вимогам до якості освітніх програмних продуктів, що робить її придатною для використання в курсах програмування та цифрової освіти.

Бібліографія

1. Гавриляк М.С. Програмування мережних послуг. Чернівці: Чернівец. нац. ун-тет, 2022, с. 179
2. Ключові інструменти та технології Front-end, що використовуються для розробки веб-сайтів. *Webbook Studio*. URL: https://webbookstudio.com/ua/articles/key-tools-and-technologies-used-in-front-end-website-development (дата звернення: 21.11.2025).
3. Розкриття потужності React та Next.js для створення високопродуктивних веб-додатків - javascript.org.ua - JS Communities. *javascript.org.ua - JS Communities*. URL: https://javascript.org.ua/rozkrittya-potuzhnosti-react-ta-next-js-dlya-stvorennya-visokoproduktivnih-veb-dodatkiv/ (дата звернення: 10.10.2025).
4. Самборський В. С. Flash-анімація та сфери її використання. *Актуальні питання сучасної інформатики*. 2016. Т. 2. с. 52-56.
5. Що таке iFrame і чому його небажано використовувати для пошукової оптимізації – Serpstat Blog. *Serpstat – Marketing Tool for SEO and Content Tasks*. URL: https://serpstat.com/uk/blog/sho-take-iframe/ (дата звернення: 21.01.2026).
6. Bazilinskyy P. Microsoft Silverlight. Overview of the new way of building RIA. Conference: Science Days 2011. Modern Information Technologies, VII International science-practical conference/ Prague, Czech Republic. DOI: 10.13140/RG.2.1.2983.9844
7. Korzachenko O., Cherniavskyi K. Applications: revolutionary changes in web development. *Modeling and Information Systems in Economics*. 2020. No. 99. P. 92–101. URL: https://doi.org/10.33111/mise.99.8 (дата звернення: 14.12.2025).

8. Oladele D. What is the DOM? The Document Object Model Explained in Plain English. *freeCodeCamp.org*. URL: https://www.freecodecamp.org/news/what-is-the-dom-explained-in-plain-english/ (дата звернення: 21.06.2025).
9. SPA або MPA: що вибрати для бізнесу в 2025 році?. *IT-компанія з розробки цифрових продуктів в Україні – Brander*. URL: https://brander.ua/blog/spa-abo-mpa-shcho-vybraty-dlya-biznesu-v-2025-rotsi (дата звернення: 22.06.2025).
10. SSR vs. SSG in Next.js. *Strapi - Open source Node.js Headless CMS.* URL: https://strapi.io/blog/ssr-vs-ssg-in-nextjs-differences-advantages-and-use-cases (дата звернення: 10.07.2025).
11. Stfalcon E. Топ-10 компаній з веб-розробки 2025 | Stfalcon. *Logistics & Transportation Software Development Agency | Stfalcon*. URL: https://stfalcon.com/uk/blog/post/web-development-trends-part-2 (дата звернення: 16.09.2025).
12. What is App Shell Model? Definition. *Decodo (Formerly Smartproxy) | Official Website & Login*. URL: https://decodo.com/glossary/app-shell-model (дата звернення: 21.01.2026).

References

1. Havryliak M. S. *Programming of Network Services*. Chernivtsi: Chernivtsi National University, 2022, p. 179.
2. Key Tools and Technologies of Front-End Used for Website Development. *Webbook Studio*. URL: https://webbookstudio.com/ua/articles/key-tools-and-technologies-used-in-front-end-website-development (date of access: 21.11.2025).
3. Unleashing the Power of React and Next.js for Building High-Performance Web Applications. *javascript.org.ua – JS Communities*. URL: https://javascript.org.ua/rozkrittya-potuzhnosti-react-ta-next-js-dlya-stvorennya-visokoproduktivnih-veb-dodatkiv/ (date of access: 10.10.2025).
4. Samborskyi V. S. Flash Animation and Its Areas of Application. *Current Issues of Modern Informatics*. 2016. Vol. 2. pp. 52–56.
5. What Is an iFrame and Why It Is Not Recommended for Search Engine Optimization. *Serpstat Blog*. Serpstat – Marketing Tool for SEO and Content Tasks. URL: https://serpstat.com/uk/blog/sho-take-iframe/ (date of access: 21.01.2026).
6. Bazilinskyy P. Microsoft Silverlight: Overview of the New Way of Building RIA. In: *Science Days 2011. Modern Information Technologies*, VII International Scientific and Practical Conference, Prague, Czech Republic. DOI: 10.13140/RG.2.1.2983.9844.
7. Korzachenko O., Cherniavskyi K. Applications: Revolutionary Changes in Web Development. *Modeling and Information Systems in Economics*. 2020. No. 99. pp. 92–101. URL: https://doi.org/10.33111/mise.99.8 (date of access: 14.12.2025).
8. Oladele D. What Is the DOM? The Document Object Model Explained in Plain English. *freeCodeCamp.org*. URL: https://www.freecodecamp.org/news/what-is-the-dom-explained-in-plain-english/ (date of access: 21.06.2025).
9. SPA or MPA: What Should a Business Choose in 2025? *Brander – Digital Product Development Company in Ukraine*. URL: https://brander.ua/blog/spa-abo-mpa-shcho-vybraty-dlya-biznesu-v-2025-rotsi (date of access: 22.06.2025).
10. SSR vs. SSG in Next.js. *Strapi – Open Source Node.js Headless CMS*. URL: https://strapi.io/blog/ssr-vs-ssg-in-nextjs-differences-advantages-and-use-cases (date of access: 10.07.2025).
11. Stfalcon E. Top 10 Web Development Companies in 2025. *Stfalcon – Logistics & Transportation Software Development Agency*. URL: https://stfalcon.com/uk/blog/post/web-development-trends-part-2 (date of access: 16.09.2025).
12. What is App Shell Model? Definition. *Decodo (Formerly Smartproxy) | Official Website & Login*. URL: https://decodo.com/glossary/app-shell-model (date of access: 21.01.2026).