МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп’ютерних наук та кібербезпеки

ЖУКОВСЬКИЙ МАКСИМ ПЕТРОВИЧ

ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ОНЛАЙН-КУРСІВ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ LARAVEL, МОВИ PHP ТА БІБЛІОТЕКИ REACT.JS

Спеціальність: 122 Комп’ютерні науки
Освітньо-професійна програма: Комп’ютерні науки та інформаційні технології
Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»

Науковий керівник:
ЮНЧИК ВАЛЕНТИНА ЛЕОНІДІВНА,
доктор філософії, доцент кафедри
комп’ютерних наук та кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №_____
засідання кафедри комп’ютерних наук
та кібербезпеки від
_____ 2025 р.
Завідувач кафедри
(_____) Гришанович Т. О.

ЛУЦЬК - 2025

Зміст

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій спостерігається трансформація практично всіх сфер людської діяльності, зокрема й освіти. Ще донедавна здобуття знань у таких галузях, як математика чи іноземні мови, вимагало звернення до традиційних джерел, переважно у формі друкованих книг, які потрібно було шукати у бібліотеках. Згодом ці ресурси набули електронної форми, що значно підвищило доступність освітніх матеріалів для широкого кола користувачів.

Із розвитком інтернету та цифрової інфраструктури було створено нові підходи до організації навчання, серед яких особливе місце посіли онлайн-курси. Вони представляють собою структуровані системи навчального контенту, що дозволяють опанувати складні теми в стислі терміни завдяки поєднанню теоретичних матеріалів, інтерактивних завдань та відеолекцій. Це значно оптимізує навчальний процес, роблячи його адаптивним і гнучким.

Попри це, у цифровому просторі представлено значну кількість розрізнених матеріалів у вигляді відео, текстів, блогів, які хоч і сприяють навчанню, проте часто не мають чіткої структури та педагогічної логіки. Платформи на зразок YouTube, Telegra.ph чи Medium дають змогу публікувати власні освітні продукти, проте користувачеві доводиться витрачати значну кількість часу на пошук і фільтрацію релевантного контенту. Наприклад, введення у пошукову систему запиту «курси англійської мови» здебільшого повертає сотні платних результатів, тоді як безкоштовні джерела часто містять уривчасту або недостатньо якісну інформацію.

У зв’язку з цим особливої актуальності набуває розробка спеціалізованих вебзастосунків, орієнтованих на створення онлайн-курсів. Такі рішення дозволяють стандартизувати навчальний процес, забезпечити якісну структуру подачі матеріалу, оптимізувати час роботи викладача або ментора, а також надати користувачам зручний доступ до навчального контенту в будь-який час.

Метою даної роботи є розробка вебзастосунку *SkillUp*, що забезпечує можливість створення та проходження онлайн-курсів. Застосунок реалізується з

використанням сучасних вебтехнологій: Laravel для серверної частини (backend) та React.js – для клієнтської (frontend). Система має на меті забезпечити повний цикл розробки та використання курсів – від створення до проходження та перевірки знань

Для досягнення поставленої мети необхідно вирішити наступні завдання**:**

- Дослідити концепцію онлайн-курсів та принципи їх побудови.
- Проаналізувати сучасні тенденції онлайн-навчання та функціональні особливості аналогічних вебзастосунків.
- Сформувати вимоги до системи з урахуванням потреб цільової аудиторії.
- Розробити архітектуру, інтерфейс та функціональну частину веб-застосунку.
- Здійснити тестування застосунку та оцінити його ефективність у контексті організації онлайн-навчання.

Об’єкт дослідження – цифрове освітнє середовище, що забезпечує реалізацію онлайн-навчання.

Предмет дослідження – процес розробки вебзастосунку для створення онлайн-курсів із використанням сучасних вебтехнологій, зокрема фреймворку Laravel (PHP) для серверної частини та бібліотеки React.js для клієнтської частини.

Результати роботи були представленні на обговоренні на наступних конференціях:

- XIV міжнародної науково-практичної конференції «Математика. інформаційні технології. освіта»
- "Міжнародна науково-практична конференція "Проблеми комп’ютерних наук, програмного моделювання та безпеки цифрових систем""

РОЗДІЛ 1.
ТЕОРЕТИЧНІ ЗАСАДИ ТА ТЕХНОЛОГІЧНІ ОСНОВИ РОЗРОБКИ ВЕБЗАСТОСУНКІВ ДЛЯ ОНЛАЙН-НАВЧАННЯ

1.1 Поняття вебзастосунку та його характерні особливості

У контексті стрімкого розвитку інформаційно-комунікаційних технологій вебзастосунки набули ключового значення як універсальні інструменти для доступу до інформації, взаємодії з сервісами та організації комунікації. Їх популярність зумовлена високим рівнем доступності, платформної незалежності та здатністю забезпечити ефективну інтеракцію між користувачем і серверною частиною системи. Особливо актуальним є використання вебзастосунків у сфері онлайн-освіти, де вони виступають основним технологічним засобом для реалізації освітнього процесу [10].

Вебзастосунок (англ. *web application*) – це прикладне програмне забезпечення, що функціонує в середовищі веббраузера та забезпечує інтерактивну взаємодію користувача з сервером у режимі реального часу [3, 5]. На відміну від традиційних десктопних програм, які потребують локальної інсталяції, вебзастосунки є доступними безпосередньо через інтернет та не вимагають завантаження на пристрій кінцевого користувача [1].

Основу функціонування вебзастосунків становить клієнт-серверна архітектура (Рис 1.1), згідно з якою клієнт (веббраузер) ініціює запит до сервера, а останній здійснює обробку даних та формує відповідь, що повертається у вигляді HTML-сторінки або структурованого контенту [4, 5].

Сучасні вебзастосунки мають низку характеристик, що зумовлюють їхню ефективність, масштабованість та привабливість для розробників:

1. Доступність та платформна незалежність. Завдяки функціонуванню у вебсередовищі, вебзастосунки є доступними з будь-якого пристрою, незалежно від операційної системи, що забезпечує широке охоплення аудиторії та зручність використання [1].

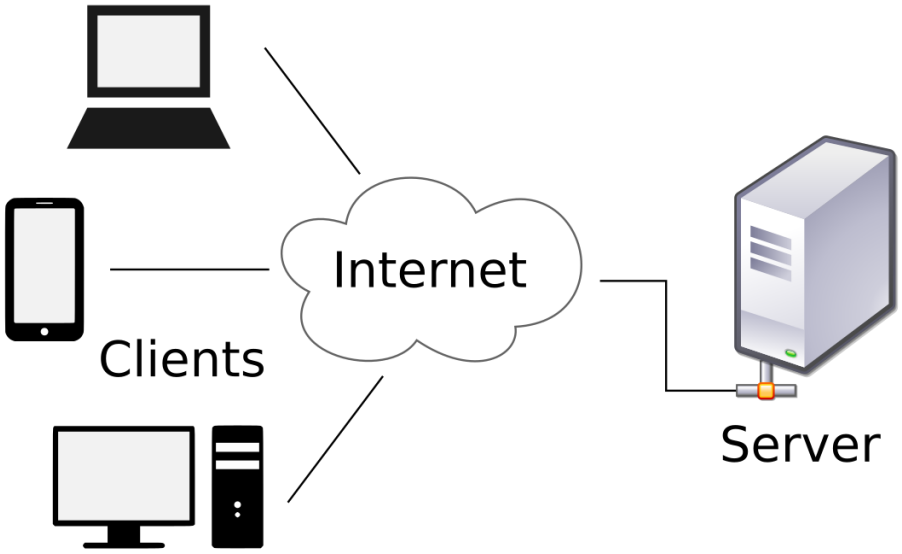


Рис 1.1. Клієнт-серверна архітектура вебзастосунку

2. Інтерактивність та динамічність. Застосування сучасних технологій фронтенд-розробки (бібліотек та фреймворків) дозволяє створювати інтерфейси, орієнтовані на користувача, з інтерактивними елементами, адаптивним дизайном і високим рівнем візуального залучення (Рис 1.2) [2, 14].

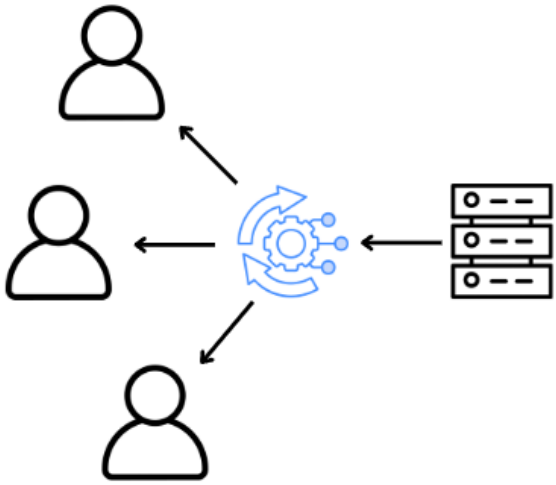


Рис 1.2. Приклад динамічного оновлення інтерфейсу у вебзастосунку

3. Масштабованість і гнучкість у розробці. Сучасні вебфреймворки забезпечують можливість поступового розширення функціоналу застосунку, полегшують інтеграцію з іншими сервісами та сприяють швидкому розгортанню нових модулів [21, 27].

4. Безпека даних. Вебзастосунки впроваджують сучасні механізми інформаційної безпеки, серед яких – шифрування, багаторівнева автентифікація, контроль доступу та збереження резервних копій. Це є критично важливим, зокрема в освітньому середовищі, де опрацьовуються персональні дані студентів і викладачів [23].

5. Централізоване оновлення. Завдяки розміщенню застосунків на віддалених серверах, оновлення програмного забезпечення здійснюється централізовано, що забезпечує синхронне оновлення для всіх користувачів без потреби ручної інсталяції оновлень.

У контексті розробки цифрових сервісів доцільним є розмежування понять вебсайт, вебзастосунок і вебсервіс, оскільки кожне з них має свою функціональну специфіку, архітектурні особливості та призначення (див. таблицю 1.1) [27].

На відміну від статичних вебсайтів, які здебільшого забезпечують лише односторонню передачу інформації (наприклад, вебсторінки з фіксованим вмістом), вебзастосунки забезпечують активну взаємодію з користувачем, реалізують бізнес-логіку, зберігають та опрацьовують дані, дозволяють здійснювати реєстрацію, автентифікацію, управління контентом тощо.

У порівнянні з вебсервісами, які орієнтовані переважно на програмну взаємодію між системами через API (наприклад, RESTful чи SOAP-сервіси), вебзастосунки зосереджені на забезпеченні інтерфейсу для людської взаємодії [30]. При цьому, у більшості випадків вебзастосунки самі використовують веб-сервіси як частину своєї функціональності для обміну даними або інтеграції з іншими системами.

Таблиця 1.1.

Порівняння понять: вебсайт, вебзастосунок, вебсервіс

Критерій	Вебсайт	Вебзастосунок	Вебсервіс
Призначення	Подання інформації	Виконання інтерактивних дій, опрацювання даних	Надання функцій іншим програмам через API

Взаємодія з користувачем	Одностороння (читання вмісту)	Двостороння (введення/виведення даних)	Немає прямої взаємодії з користувачем
Інтерфейс	Графічний інтерфейс для перегляду	Динамічний інтерфейс із можливістю керування	Інтерфейс на рівні програмного коду (JSON, XML, тощо)
Приклад	Новинний портал, блог, сайт-візитка	Онлайн-магазин, електронний щоденник, освітня платформа	API сервіс оплати, сервіс авторизації Google, REST API Moodle
Основна технологія	HTML/CSS, базовий JavaScript	Full-stack: frontend + backend, інтеграція з БД	Протоколи HTTP/HTTPS, REST, SOAP
Наявність бізнес-логіки	Мінімальна або відсутня	Реалізована на серверній стороні	Реалізована, але орієнтована на програмну взаємодію
Потреба в автентифікації	Не обов’язкова	Зазвичай необхідна	Часто обов’язкова (для доступу до API)
Оновлення вмісту	Статичне або вручну оновлюване	Динамічне, змінюється в залежності від дій користувача	Змінюється за запитами з інших систем

Таким чином, вебзастосунки виступають основою для реалізації цифрових сервісів у таких сферах, як електронна комерція, банківські послуги, управління підприємствами, а також в освіті – зокрема, для організації дистанційного навчання, тестування, управління курсами та збереження освітніх даних.

1.2. Класифікація вебзастосунків

Широкий спектр сучасних вебзастосунків обумовлює необхідність їх систематизації відповідно до різних критеріїв [6]. Залежно від цілей, функціонального призначення, архітектурних особливостей та технологічного підходу, вебзастосунки можуть мати істотні відмінності. Знання таких класифікацій є важливим як для розробників, так і для кінцевих користувачів, оскільки дає змогу обрати оптимальне рішення для реалізації конкретного

завдання, з урахуванням вимог до функціоналу, масштабованості, продуктивності та підтримуваних технологій.

Однією з ключових ознак класифікації вебзастосунків є їх функціональне призначення, що ґрунтується на їх прикладному застосуванні та завданнях, які вони вирішують [6]. Серед основних типів розрізняють:

- Освітні вебзастосунки – програмні продукти, що спрямовані на підтримку та оптимізацію освітнього процесу [10]. До них належать платформи дистанційного навчання, системи управління курсами, модулі тестування, електронні щоденники тощо. Такі системи забезпечують доступ до навчального контенту, моніторинг активності користувачів, зворотний зв'язок та автоматизовану оцінку знань.

- Комерційні вебзастосунки – системи, орієнтовані на здійснення електронної комерції: інтернет-магазини, платіжні шлюзи, платформи бронювання, CRM-системи. Їхня ключова функція полягає у підтримці транзакційної діяльності, обробці замовлень та управлінні бізнес-процесами.

- Інформаційні вебзастосунки – до цієї категорії належать портали новин, блоги, системи керування контентом (CMS), корпоративні сайти. Основне завдання таких платформ полягає в агрегуванні, редагуванні та публікації контенту.

- Соціальні та комунікаційні платформи – вебзастосунки, що забезпечують обмін повідомленнями, мультимедійним контентом, взаємодію в реальному часі. Вони широко застосовуються в освіті, бізнесі та для побудови спільнот [9].

Класифікація за архітектурними особливостями визначає принципи організації його компонентів, взаємозв’язки між модулями та спосіб обробки запитів користувача. Основні архітектурні типи:

- Монолітні вебзастосунки - Рис 1.4 – реалізуються як єдине програмне середовище, у якому всі функціональні модулі інтегровані в одне ціле. Цей підхід є доцільним для невеликих проєктів завдяки простоті розгортання. Водночас він

має обмеження у масштабованості та підтримці, особливо в умовах зростання навантаження [13].

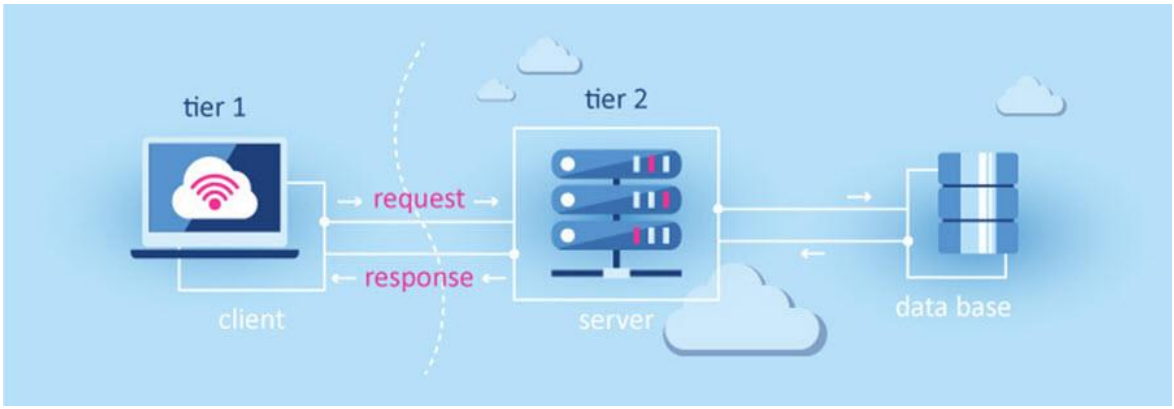


Рис 1.3 Приклад монолітного застосунка

- Мікросервісна архітектура Рис 1.4 – передбачає поділ функціональності на низку незалежних сервісів, кожен з яких виконує окрему задачу та може розгортатися автономно. Така архітектура сприяє гнучкості, масштабованості та підвищенню відмовостійкості системи, дозволяючи впроваджувати нові функції без впливу на роботу всієї платформи [5].

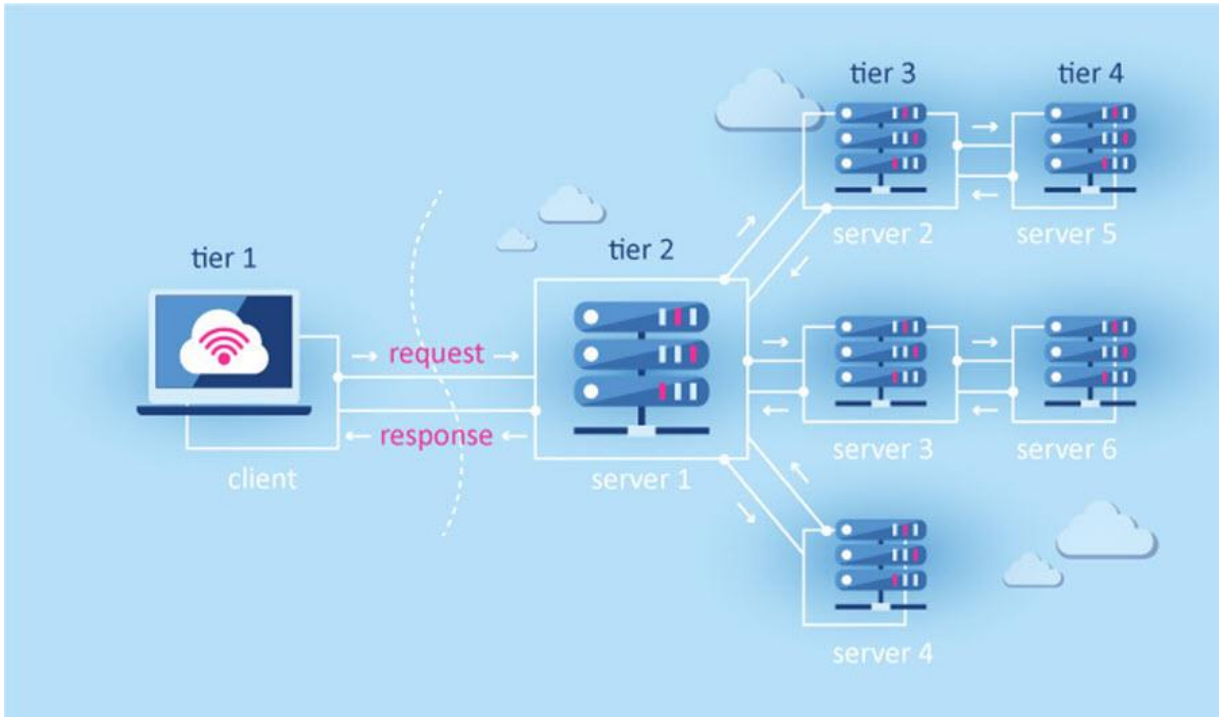


Рис 1.4 Багаторівневого застосунку

- Односторінкові застосунки (SPA) – це тип вебзастосунків, які забезпечують безперервну взаємодію з користувачем завдяки динамічному

оновленню контенту без повного перезавантаження сторінки. Такий підхід значно покращує користувацький досвід та знижує навантаження на сервер [7].

Класифікація вебзастосунків за технологічним підходом ґрунтується на методах генерації та відображення контенту, що визначає як функціональні можливості системи, так і її продуктивність [4]. Існує три основні підходи:

1. Клієнт-орієнтований рендеринг (Client-Side Rendering, CSR). У цьому підході опрацювання даних та формування інтерфейсу здійснюється на стороні клієнта, тобто безпосередньо у веббраузері користувача. До переваг належать висока інтерактивність, швидке оновлення контенту та зменшення навантаження на сервер [14]. Водночас, суттєвим недоліком є підвищене споживання ресурсів клієнтського пристрою, що може негативно впливати на продуктивність, особливо на слабших пристроях.

2. Сервер-орієнтований рендеринг (Server-Side Rendering, SSR). На відміну від клієнтського підходу, у цьому випадку HTML-сторінки формуються на сервері, після чого вже готовий контент надсилається до клієнтського браузера [4]. Такий механізм забезпечує швидше початкове завантаження сторінки, кращу індексацію пошуковими системами та стабільну продуктивність незалежно від можливостей клієнтського пристрою. Проте, він залежить від потужності серверної інфраструктури і може створювати затримки при значному навантаженні.

3. Гібридний рендеринг (Hybrid Rendering). Поєднуючи елементи CSR та SSR, гібридні підходи забезпечують оптимальний баланс між швидкістю, інтерактивністю та ефективністю [6]. Вони дозволяють динамічно адаптувати рендеринг залежно від конкретних потреб користувача або типу контенту. Такий підхід часто застосовується у сучасних вебзастосунках для досягнення найкращого користувацького досвіду.

Іншою важливою ознакою класифікації вебзастосунків є характер взаємодії з користувачем. Цей критерій визначає рівень інтерактивності та функціональні можливості інтерфейсу:

1. Статичні вебзастосунки не передбачають активної взаємодії з користувачем, або забезпечують її на мінімальному рівні. Контент таких застосунків є фіксованим та однаковим для всіх користувачів, що робить їх доцільними для невеликих інформаційних ресурсів (наприклад, сайт-візитка або рекламна сторінка). Основна перевага – простота реалізації та висока швидкість завантаження.

2. Динамічні вебзастосунки дозволяють реагувати на дії користувача в режимі реального часу, зокрема опрацювання кліків, введення даних, вибір опцій тощо. Вони використовуються у сервісах електронної комерції, освітніх платформах, соціальних мережах, де кожна дія користувача впливає на зміст сторінки [2].

3. Гібридні вебзастосунки. Найпоширеніший підхід, який об'єднує функціональність як статичних, так і динамічних систем. У таких застосунках частина контенту є постійною (наприклад, загальна інформація), тоді як інші елементи можуть змінюватися відповідно до взаємодії з користувачем (редагування даних, додавання до кошика, навігація тощо). Це забезпечує гнучкість у реалізації широкого спектру функціональних можливостей [1].

Отже, класифікація вебзастосунків за різними критеріями – функціональним призначенням, архітектурною побудовою, технологічним підходом до рендерингу та способом взаємодії з користувачем – дозволяє всебічно охарактеризувати особливості їх проєктування та використання [6, 19]. Такий підхід забезпечує раціональний вибір інструментів і технологій на етапі розробки, сприяючи підвищенню ефективності та адаптивності вебзастосунку до потреб кінцевих користувачів.

1.3 Архітектура та компоненти вебзастосунку для онлайн-курсів

Сучасний вебзастосунок для онлайн-навчання є ефективним інструментом цифрової трансформації освітнього процесу, що забезпечує централізоване управління навчальними ресурсами, забезпечує доступність контенту та сприяє

активній взаємодії між усіма учасниками освітнього середовища. Основу функціонування таких систем становить ретельно спроєктована архітектура, яка охоплює низку взаємозалежних функціональних компонентів [6].

Типовий вебзастосунок для онлайн-освіти реалізується за моделлю клієнт-серверної архітектури, де інтерфейс користувача функціонує на стороні клієнта (веббраузера), а бізнес-логіка, опрацювання запитів та збереження даних реалізуються на серверній стороні. Такий підхід дозволяє забезпечити централізоване адміністрування системи, спрощує оновлення та підтримку, а також гарантує стабільність і масштабованість навіть за умов підвищеного навантаження [27].

Типова платформа для проведення онлайн-курсів складається з низки взаємопов’язаних компонентів, кожен із яких виконує специфічні функції. Основна архітектурна модель орієнтована на тришарову клієнт-серверну організацію: клієнтський інтерфейс, серверна логіка та система збереження даних (база даних) (Рис 1.5) [6]. Такий підхід дозволяє централізовано керувати даними, масштабувати систему та інтегрувати сторонні сервіси.

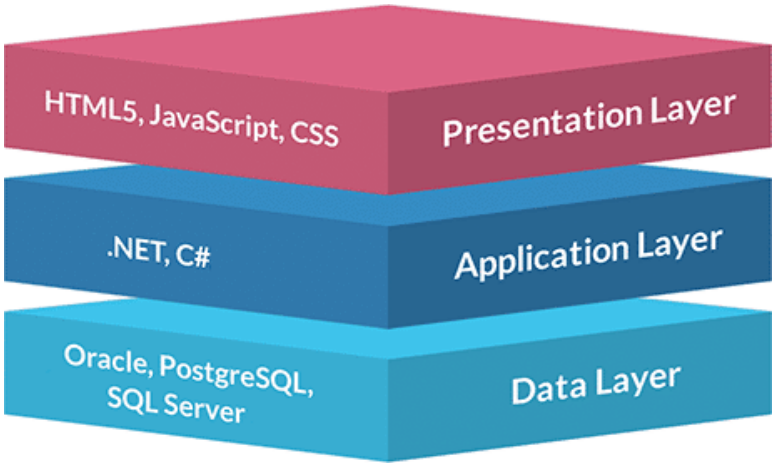


Рис 1.5 Основні компоненти застосунку

Основні компоненти вебзастосунку.

Інтерфейс користувача (User Interface, UI). Інтерфейс користувача є центральним елементом взаємодії із системою, що забезпечує інтуїтивно зрозумілу взаємодію для студентів, викладачів і адміністраторів [26]. Завдяки

адаптивному дизайну користувачі можуть ефективно працювати з вебзастосунком на різних пристроях – від мобільних телефонів до стаціонарних комп'ютерів. Основні функції UI включають:

- відображення мультимедійного навчального контенту (тексти, відео, аудіо, зображення);
- реалізацію зручної навігації по навчальних модулях та доступу до додаткових матеріалів;
- інтеграцію інтерактивних елементів: тестів, форумів, форм зворотного зв'язку тощо.

Серверна логіка (Backend/Server-side). Логіка вебзастосунку реалізується на серверній стороні і відповідає за опрацювання запитів користувача, реалізацію функціональних сценаріїв, адміністрування навчальних курсів, перевірку тестових завдань і ведення обліку успішності [21]. Вона забезпечує автоматизацію процесів навчання, що підвищує ефективність викладацької діяльності та зменшує адміністративне навантаження. Серверна логіка часто реалізується з використанням фреймворків, таких як Laravel (PHP), Django (Python) або Express (Node.js).

База даних (Database Layer). Надійне зберігання даних – один з критичних елементів будь-якого онлайн-застосунку. База даних виконує збереження структурованої інформації про користувачів, курси, результати тестування, журнал активності тощо [15, 16]. Ключовими особливостями цього компонента є:

- реалізація механізмів резервного копіювання та відновлення даних;
- застосування алгоритмів оптимізації запитів, що забезпечують високу швидкість доступу до інформації;
- дотримання стандартів безпеки та захисту персональних даних відповідно до нормативно-правових актів.

Модулі оцінювання і контролю знань. Ці модулі забезпечують створення, збереження, автоматичне оцінювання тестових завдань, формування результатів

навчання [29]. Інколи до них додають інструменти для генерації адаптивних тестів, формування сертифікатів або побудови аналітики на основі результатів. Інтеграція таких модулів сприяє об’єктивному оцінюванню та забезпеченню зворотного зв’язку між учасниками освітнього процесу.

Безпека та автентифікація. Безпекова інфраструктура вебзастосунку забезпечує цілісність, конфіденційність та доступність освітніх ресурсів [23]. Основними складовими цієї підсистеми є механізми автентифікації (перевірка особи користувача) та авторизації (контроль доступу до функціоналу системи). Для захисту даних використовується наскрізне шифрування, що мінімізує ризики несанкціонованого доступу.

Вебзастосунок функціонує як єдина система за рахунок налагодженої взаємодії між компонентами. Користувач, через браузер (UI), ініціює запити (наприклад, запит на проходження курсу або надсилання тесту), які передаються на сервер. Серверна логіка опрацьовує ці запити, звертається до бази даних за потрібною інформацією або оновлює її, а потім повертає відповідь клієнту у формі вебсторінки, повідомлення чи динамічного контенту. Така взаємодія забезпечується через RESTful API Рис 1.6, WebSocket або GraphQL, залежно від складності застосунку [30].

Використання модульного підходу в архітектурі сприяє гнучкості та масштабованості системи. Компоненти можуть оновлюватися або замінюватися незалежно один від одного, що дозволяє адаптувати платформу до нових педагогічних вимог чи технологічних інновацій [19].

Більшість сучасних вебзастосунків підтримують інтеграцію з зовнішніми системами через програмні інтерфейси (API) [22]. API дозволяють обмінюватися даними між різними сервісами (наприклад, платіжними системами, сервісами відеоконференцій, системами аналітики), розширюючи функціональність вебзастосунку та сприяючи формуванню єдиного цифрового освітнього простору.

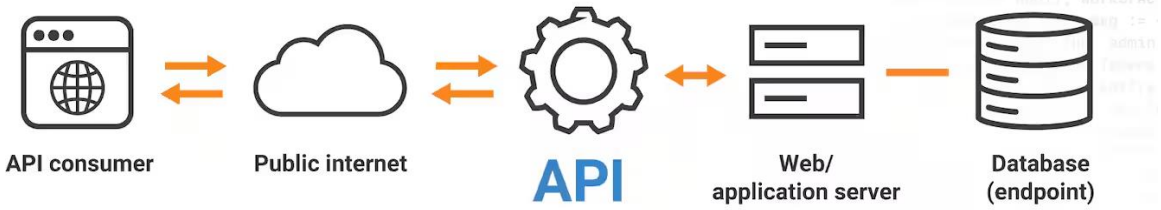


Рис 1.6 Робота API

1.4. Платформи для онлайн-навчання, їх можливості та обмеження

Цифрова трансформація суспільства, а також глобальні виклики останніх років, значною мірою вплинули на трансформацію освітньої системи. У цьому контексті онлайн-курси стали одним із ключових інструментів забезпечення безперервної освіти, що охоплює широкий спектр аудиторій – від школярів до фахівців різних галузей [11].

На сьогодні онлайн-курси вже посіли важливе місце в освітньому середовищі, адаптуючись до специфічних потреб різних груп користувачів – як у загальній середній освіті, так і у професійному та корпоративному навчанні [11]. Наприклад, для учнів старших класів створено спеціалізовані курси підготовки до національного мультипредметного тесту (НМТ), які орієнтуються на цільові знання та навички. Водночас корпоративний сектор активно використовує онлайн-курси для підвищення кваліфікації працівників, скорочуючи час на навчання з кількох років до кількох місяців.

Однак, попри очевидні переваги, онлайн-освіта має і певні обмеження. Учасники курсів, здебільшого, опановують великі обсяги інформації у стислий проміжок часу, що ускладнює її практичне засвоєння [10]. Як наслідок, виникає проблема підготовки фахівців, які мають теоретичні знання, але не володіють достатнім практичним досвідом. Це змушує роботодавців додатково інвестувати ресурси в адаптацію таких працівників до виробничого середовища.

1. Персоналізоване навчання з використанням технологій штучного інтелекту.

Інтеграція штучного інтелекту (ШІ) в освітні платформи відкриває нові можливості для реалізації принципів персоналізованого навчання [12]. ШІ може аналізувати індивідуальні особливості здобувачів освіти, зокрема темп засвоєння матеріалу, стилі навчання, рівень мотивації, і на цій основі формувати адаптивні освітні траєкторії. У майбутньому очікується впровадження повністю індивідуалізованих навчальних програм - Рис 1.7, що відповідатимуть потребам конкретного користувача, забезпечуючи максимальну ефективність засвоєння знань.

Рис 1.7 Електронне навчання в 2024-2025 роках

Хоча наразі такі системи перебувають на стадії експериментального впровадження, вже сьогодні спостерігається активне використання ШІ у вигляді інструментів для автоматичної перевірки завдань, генерації рекомендацій та підтримки студентів у режимі 24/7 [12].

2. Концепція навчання впродовж життя (Lifelong Learning).

У сучасному динамічному світі навчання впродовж життя стає необхідною умовою професійної мобільності та особистісного розвитку. Онлайн-курси у цьому контексті виступають основним засобом реалізації ідеї безперервного навчання, оскільки вони забезпечують доступ до нових знань та компетентностей незалежно від географічного положення, часу чи попереднього досвіду користувача.

Таким чином, онлайн-курси сьогодні не лише доповнюють традиційну освіту, а й формують нову парадигму освітнього процесу, що ґрунтується на гнучкості, доступності та технологічній адаптивності до потреб суспільства [11].

Сучасні вебзастосунки для онлайн-навчання стрімко розвиваються, впроваджуючи інноваційні технології, які підвищують ефективність освітнього процесу. Ключовими тенденціями є інтеграція мобільних технологій, підтримка мультимедійного контенту та розвиток інтерфейсів з фокусом на користувача [26].

Хмарні платформи забезпечують масштабованість, доступність та надійність зберігання даних, що є критичним для сучасних LMS [27]. Вони дозволяють централізовано керувати навчальними курсами, користувачами, оцінюванням та аналітикою. LMS підтримують гнучкі моделі навчання, адаптовані до різних типів аудиторії.

Використання адаптивних алгоритмів дозволяє підлаштовувати освітній контент під індивідуальні потреби здобувачів знань [12]. Освітня аналітика надає можливість оцінити прогрес студентів, виявити проблемні зони та оптимізувати навчальні стратегії. Це сприяє підвищенню якості освіти та персоналізації навчального процесу.

Впровадження елементів гейміфікації підвищує мотивацію студентів, покращує залучення та активність в процесі навчання [10]. Ігрові механіки, такі як бали, рівні, нагороди, створюють інтерактивне середовище, що стимулює регулярну участь користувачів.

Відеоматеріали виступають ефективним засобом подання складної інформації, забезпечуючи наочність та кращу засвоюваність. Чат-боти

забезпечують оперативну підтримку користувачів, автоматизують відповіді на типові запитання, сприяють організації навчального процесу та підтримці зворотного зв’язку.

Платформи підтримують різні форми комунікації: синхронні (відеоконференції, вебінари) забезпечують реальний час взаємодії учасників, тоді як асинхронні (форуми, обговорення, електронна пошта) дозволяють навчатися у зручний час, що підвищує гнучкість освітнього процесу [10].

1.5. Технологічна основа розробки вебзастосунку

Розробка сучасних вебзастосунків потребує використання цілого комплексу технологій, які забезпечують функціональність, продуктивність, масштабованість і безпеку програмних рішень [19]. Ці технології поділяються на декілька категорій відповідно до їхнього функціонального призначення: серверні фреймворки, клієнтські бібліотеки, системи керування базами даних, інтерфейси програмування застосунків (API) та засоби контролю версій.

Серверні фреймворки (англ. *backend frameworks*) є невід’ємним компонентом технологічного стеку вебзастосунків, оскільки визначають архітектурну організацію серверної частини, сприяють реалізації прикладної логіки та забезпечують ефективну опрацювання запитів від клієнтської частини [23]. Вони виступають як програмні каркаси, що надають набір структур, інструментів та шаблонів для побудови надійного, масштабованого й безпечного серверного середовища.

Одним із основних завдань серверних фреймворків є опрацювання HTTP-запитів користувача (GET, POST, PUT, DELETE тощо), які надходять до вебзастосунку, та формування відповідей, що повертаються до клієнта. У цьому контексті фреймворк керує маршрутизацією (routing), тобто визначенням того, який метод контролера слід викликати для конкретного запиту. Це дає змогу чітко структурувати програмний код і підвищити його читабельність та повторне використання.

Фреймворки також забезпечують механізми автентифікації та авторизації користувачів, що є критично важливими в умовах захисту персональних даних і реалізації рольової моделі доступу. Більшість сучасних серверних фреймворків підтримують інтеграцію з зовнішніми сервісами автентифікації (OAuth2, JWT, LDAP) та мають вбудовані засоби для керування сесіями, токенами й користувацькими ролями.

Важливою особливістю фреймворків є інтеграція з базами даних через ORM (Object-Relational Mapping) – програмний рівень, який дозволяє працювати з базами даних за допомогою об’єктно-орієнтованих конструкцій. Завдяки ORM значно спрощується процес взаємодії з даними, оскільки розробники можуть оперувати об'єктами мови програмування замість написання SQL-запитів вручну.

Крім цього, серверні фреймворки часто включають системи шаблонізації, які дозволяють динамічно формувати HTML-вміст на основі даних, що надходять із сервера. Це особливо корисно для серверної генерації сторінок у випадках, коли клієнтська частина не виконує повну візуалізацію контенту.

Усі ці можливості об’єднуються в одному цілісному середовищі, яке покращує стандартизацію, прискорює розробку, зменшує кількість помилок та дає змогу ефективно масштабувати застосунок у разі зростання навантаження.

Серед прикладів серверних фреймворків, які отримали широке поширення в галузі, варто зазначити:

- фреймворки на основі PHP (наприклад, Laravel, Symfony);
- фреймворки для Python (зокрема, Django, Flask);
- фреймворки для JavaScript (Node.js), зокрема Express.js, NestJS;
- фреймворки для Java (наприклад, Spring Boot).

Таким чином, вибір серверного фреймворку має ґрунтуватися на аналізі вимог до проєкту, мови програмування, а також підтримки масштабованості, безпеки та взаємодії з іншими технологіями у проєкті [21].

Клієнтська частина вебзастосунку (frontend) виконує ключову функцію в забезпеченні інтуїтивно зрозумілого та ефективного користувацького досвіду, відповідаючи за візуалізацію даних, реакцію на дії користувача та динамічну взаємодію з сервером. У сучасних умовах розробки вебінтерфейсів використання JavaScript-бібліотек є критично необхідним для досягнення високої продуктивності, адаптивності та інтероперабельності користувацького інтерфейсу.

Клієнтські бібліотеки – це програмні засоби, що надають структуровані інструменти для побудови інтерфейсів з використанням принципів компонентного програмування [14]. Цей підхід передбачає розбиття інтерфейсу на окремі, повторно використовувані компоненти з власним станом, логікою та шаблонами. Така модульність підвищує масштабованість застосунку, сприяє його супроводжуваності та дозволяє повторно використовувати частини інтерфейсу у різних контекстах.

Однією з ключових можливостей клієнтських бібліотек є реактивність – здатність інтерфейсу автоматично оновлюватися у відповідь на зміну даних. Це забезпечується завдяки двосторонньому зв’язку (*two-way data binding*) або односторонньому потоку даних (*unidirectional data flow*), що гарантує узгодженість стану додатку з його відображенням.

Іншою важливою особливістю є ефективна взаємодія з DOM (Document Object Model). Сучасні бібліотеки використовують такі концепції, як віртуальний DOM (virtual DOM), що мінімізує кількість змін у реальному DOM і тим самим значно підвищує продуктивність. Це особливо актуально для застосунків із великою кількістю динамічних елементів або високою частотою оновлень.

Крім того, сучасні клієнтські бібліотеки інтегруються з засобами маршрутизації, управління станом (наприклад, з використанням *state management libraries*), а також забезпечують підтримку форм, валідації, анімацій, локалізації та інших аспектів, необхідних для створення повнофункціональних користувацьких інтерфейсів.

Серед найбільш поширених клієнтських бібліотек, що широко застосовуються в розробці вебзастосунків, можна виокремити:

- React.js – бібліотеку з відкритим вихідним кодом, розроблену Meta, яка реалізує концепцію компонентів та віртуального DOM;
- Vue.js – легку бібліотеку з гнучкою архітектурою, що підтримує двостороннє зв'язування даних;
- **S**velte – компіляційно-орієнтовану бібліотеку, яка трансформує компоненти у високоефективний JavaScript-код ще до виконання в браузері [14].

Таким чином, клієнтські бібліотеки є фундаментальними інструментами для розробки адаптивних, продуктивних і масштабованих інтерфейсів користувача. Їх використання забезпечує не лише естетичну привабливість, а й функціональну ефективність сучасних вебзастосунків.

Бази даних є невіддільною складовою сучасних вебзастосунків, забезпечуючи централізоване зберігання, доступ, оновлення та керування структурованою або неструктурованою інформацією [15]. Вони дозволяють систематизувати дані у відповідності до логічної моделі, забезпечити цілісність, узгодженість та доступність інформації у багатокористувацькому середовищі.

У контексті веброзробки найбільш поширеною є класифікація баз даних на реляційні та нереляційні (NoSQL) системи, кожна з яких має свої особливості, переваги та області застосування.

Реляційні бази даних (Relational Databases, RDBMS) ґрунтуються на теоретичній основі реляційної алгебри та зберігають дані у вигляді таблиць з чітко визначеними схемами. Основною мовою запитів до таких баз даних є SQL (Structured Query Language), яка надає засоби для опису структури таблиць, вибірки даних, виконання агрегацій, об'єднання таблиць та управління транзакціями [19].

До переваг реляційних баз даних належать:

- підтримка транзакційності (ACID-властивості);

- високий ступінь цілісності даних завдяки використанню ключів, обмежень і зв’язків між таблицями;
- зручність у реалізації аналітичних запитів.

Поширеними представниками реляційних СУБД є MySQL**,** PostgreSQL**,** MariaDB, Microsoft SQL Server та Oracle Database.

Нереляційні бази даних (NoSQL) виникли як відповідь на потребу в опрацюванні великих обсягів даних, гнучких структур, високої продуктивності та масштабованості. Вони не використовують фіксовану схему таблиць, що дозволяє зберігати дані у вигляді документів, пар «ключ-значення», графів або колонкових структур. Це робить NoSQL-системи особливо зручними у випадках збереження мультимедійного контенту, JSON-даних або великих потоків неструктурованої інформації.

Серед основних переваг нереляційних баз даних:

- горизонтальна масштабованість;
- висока швидкість читання/запису для великих обсягів даних;
- гнучкість у структурі збереження даних.

До найбільш поширених NoSQL-систем належать MongoDB**,** Cassandra**,** Redis**,** CouchDB **та** Firebase Realtime Database**.**

Вибір між реляційною та нереляційною базою даних залежить від низки факторів, зокрема:

- характер даних (структуровані чи ні);
- обсяг і частота доступу до інформації;
- необхідність у транзакційності;
- масштабованість проєкту;
- швидкість розробки та підтримки.

Для навчальних або корпоративних застосунків, які потребують чітких структур, контролю доступу та складних аналітичних запитів, доцільним є використання реляційних баз даних. У той час як соціальні мережі, сервіси

стримінгу або мобільні застосунки частіше використовують NoSQL-системи через потребу у масштабованості, асинхронності та гнучкості.

Інтерфейси програмування застосунків (API, Application Programming Interfaces) виступають критично важливим компонентом у взаємодії клієнтської та серверної частин вебзастосунку. Вони забезпечують стандартизований механізм обміну даними між різними частинами програмної архітектури або навіть між окремими сервісами в межах складних розподілених систем. У контексті сучасної веброзробки найпоширенішими підходами до побудови API є REST та GraphQL, кожен із яких має свої методологічні особливості, переваги та обмеження.

REST – це архітектурний стиль, запропонований Роєм Філдінгом, який ґрунтується на використанні стандартних HTTP-методів (GET, POST, PUT, DELETE) для виконання CRUD-операцій над ресурсами. Ресурси ідентифікуються через URI (Uniform Resource Identifier), а передача даних зазвичай відбувається у форматах JSON або XML.

Основними характеристиками REST API є уніфікований інтерфейс, який передбачає взаємодію через стандартизовані HTTP-методи; підтримка принципу безстану, тобто відсутність збереження інформації про стан між окремими запитами; ієрархічна структура ресурсів, що реалізується через логічно структуровані URL-адреси; а також можливість кешування відповідей з метою зменшення навантаження на сервер.

Переваги REST включають простоту реалізації, широку підтримку в інструментах і мовах програмування, а також високий ступінь масштабованості. Проте при складних структурах даних REST може стати неефективним, оскільки для отримання повної інформації іноді доводиться виконувати кілька послідовних запитів.

GraphQL – це декларативна мова запитів і середовище виконання, розроблене компанією Facebook. Вона надає клієнту змогу формулювати запити, точно вказуючи, які саме поля або об’єкти потрібно отримати із сервера, що значно знижує обсяг надлишкових даних.

Ключовими особливостями GraphQL є гнучкість запитів, коли клієнт самостійно визначає, які саме дані йому необхідні, незалежно від структури відповіді сервера; використання єдиного кінцевого пункту для всіх запитів, що здійснюються через один уніфікований URI; сильна типізація, яка забезпечується чітко визначеною схемою з описом усіх можливих об'єктів і зв’язків між ними; а також підтримка вкладених запитів, що дає змогу отримувати дані з кількох рівнів структури за один запит.

GraphQL виявляється особливо ефективним у ситуаціях, коли клієнтські пристрої мають обмежені ресурси, зокрема у мобільних застосунках або односторінкових вебзастосунках (SPA); під час опрацювання великих обсягів зв’язаних даних, де необхідна точкова вибірка інформації; а також у розподілених мікросервісних архітектурах, де забезпечується централізований доступ до різних джерел даних через уніфікований інтерфейс.

REST і GraphQL – це два потужні інструменти для створення API у вебзастосунках [30]. REST залишається популярним через свою простоту, стабільність і сумісність з існуючими інфраструктурами. Натомість GraphQL надає вищу гнучкість і ефективність, особливо в умовах складних даних або мобільних платформ. Вибір між цими підходами повинен здійснюватися з урахуванням специфіки проєкту, обсягів даних, вимог до швидкодії та зручності для фронтенд-розробників.

Системи контролю версій (Version Control Systems, VCS) відіграють ключову роль у процесі створення та підтримки програмного забезпечення, особливо за умов колективної розробки [18]. Вони забезпечують механізми для збереження повної історії змін програмного коду, що дає змогу відстежувати внесені модифікації, визначати авторів змін і контролювати розвиток проєкту в часі.

Однією з основних функцій таких систем є синхронізація роботи між кількома учасниками команди, що дозволяє уникнути конфліктів при паралельному редагуванні одних і тих самих фрагментів коду. У разі виникнення суперечностей системи контролю версій надають інструменти для їх

ефективного вирішення. Також вони підтримують можливість відновлення попередніх станів проєкту, що є особливо важливим під час виявлення критичних помилок або тестування нових функцій.

У сучасній практиці найбільшого поширення набули розподілені системи контролю версій**,** які дозволяють кожному розробнику мати повну локальну копію репозиторію, включно з усією історією змін. Такий підхід сприяє автономності у роботі, а також забезпечує додаткову гнучкість і стійкість до збоїв у центральному сервері. Приклади таких систем включають Git, Mercurial та інші.

Завдяки використанню VCS розробники отримують змогу ефективно впроваджувати методології безперервної інтеграції та розгортання (CI/CD), що є основою сучасного життєвого циклу програмного забезпечення. Таким чином, системи контролю версій є незамінним інструментом для забезпечення структурованої, контрольованої та якісної розробки вебзастосунків [19].

1.6. Дослідження аналогів вебзастосунків для онлайн навчання

У сучасному освітньому середовищі представлено широкий спектр додатків та інструментів для створення онлайн-курсів [12], які класифікуються відповідно до специфіки та потреб користувачів. Зазвичай розробка таких платформ орієнтована на комерційну вигоду, а не на загальний розвиток, хоча існують і винятки, які, однак, часто є менш зручними для користувачів. Наприклад, платформа Telegraph пропонує мінімалістичний інтерфейс для створення блогів або невеликих інструкцій, що забезпечує простоту та швидкість публікації. У свою чергу, Udemy є масштабною платформою, орієнтованою на комерціалізацію освітніх продуктів, яка надає зручні засоби для створення і впровадження онлайн-курсів. Аналогічні сервіси, такі як Medium, підтримувані співзасновником Twitter та компанією Blogger, також дозволяють публікувати блоги та статті, пропонуючи подібні функціональні можливості, як і Telegraph. Особливу увагу заслуговує українська платформа Prometheus, заснована у 2013

році з початковим бюджетом у 2000 доларів, яка зуміла стати успішним та прибутковим бізнес-проєктом у сфері онлайн-освіти. У процесі розробки власного вебзастосунку для онлайн-навчання доцільним етапом є дослідження вже існуючих рішень у цій сфері. Аналіз аналогів дозволяє виявити найкращі практики, типові архітектурні та функціональні підходи, а також зрозуміти очікування користувачів і визначити власну конкурентну нішу. З-поміж найбільш відомих платформ, які активно використовуються в освітньому середовищі, слід розглянути Telegraph, Coursera, Moodle, Udemy та Prometheus (Таблиця 2). Кожна з них має специфічну модель взаємодії з користувачем, набір інструментів та особливості реалізації функціоналу.

Таблиця 2

Порівняльна характеристика платформ для навчання

Платформа	Основні характеристики	Переваги	Обмеження
Telegraph	Мінімалістичний інструмент для швидкої публікації контенту	Швидкість створення та публікації, простота використання	Відсутність інтерактивності, обмежений функціонал
Coursera	Масові онлайн-курси з високоякісним відео та сертифікацією	Професійний контент, адаптивне навчання, сертифікати	Закритий код, обмежена кастомізація, комерційна модель
Moodle	Відкрите LMS з широкими можливостями кастомізації	Підтримка SCORM, гнучкість, плагіни, відкритий код	Складність налаштування, базовий UX, мобільна адаптація
Udemy	Платформа для комерційних курсів з великим ринком	Простота створення курсів, система рекомендацій, бізнес-модель	Обмежена академічність, стандартизована структура
Prometheus	Безкоштовна українська MOOC платформа	Безкоштовність, партнерство з університетами, тестування	Обмежена гнучкість, технічні курси, залежність від фінансування

Telegraph є простим інструментом для публікації матеріалів без необхідності проходження процедури реєстрації. Цей мінімалістичний

вебзастосунок, розроблений командою Telegram, призначений для швидкого створення та публікації статей.

До основних переваг платформи слід віднести надзвичайно швидкий та інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам створювати і публікувати контент у лічені хвилини. Основні елементи інтерфейсу включають поля для введення заголовка (Title), автора та основного текстового блоку, що можна редагувати за власним бажанням (Рис 1.10).

Важливою характеристикою Telegraph є можливість забезпечення повної анонімності авторів, що дозволяє публікувати матеріали без розкриття особистих даних. Крім того, платформа підтримує вставку різноманітного контенту у пости, що сприяє підвищенню сприйняття тексту за рахунок його візуального різноманіття. Після натискання кнопки «публікувати» автор миттєво отримує унікальне посилання на свою статтю, яке може бути негайно поширене.

Отже, загальна структура платформи вирізняється лаконічним і зручним дизайном, що не перевантажений зайвою інформацією та сприяє ефективній роботі користувачів.

Платформа Coursera - рис 1.8 є однією з провідних глобальних систем масових відкритих онлайн-курсів (MOOC), що тісно співпрацює з провідними університетами та корпораціями світу. Особливістю Coursera є високий рівень структурованості навчальних програм, які містять якісний відеоконтент, інтегровані тестові завдання та системи сертифікації, що підтверджують набуті компетенції. Крім того, платформа підтримує адаптивне навчання, яке персоналізує освітній процес залежно від індивідуального прогресу користувача, що суттєво підвищує ефективність засвоєння матеріалу.

До переваг Coursera належить професійний освітній контент, створений провідними академічними інституціями та експертами-практиками, що забезпечує високу якість навчання. Платформа поєднує різноманітні формати навчальних матеріалів, включно з відео, текстовими ресурсами та тестуваннями, що сприяє комплексному засвоєнню знань. Додатковою перевагою є можливість

офіційного отримання сертифікатів, які визнаються у багатьох професійних і академічних колах.

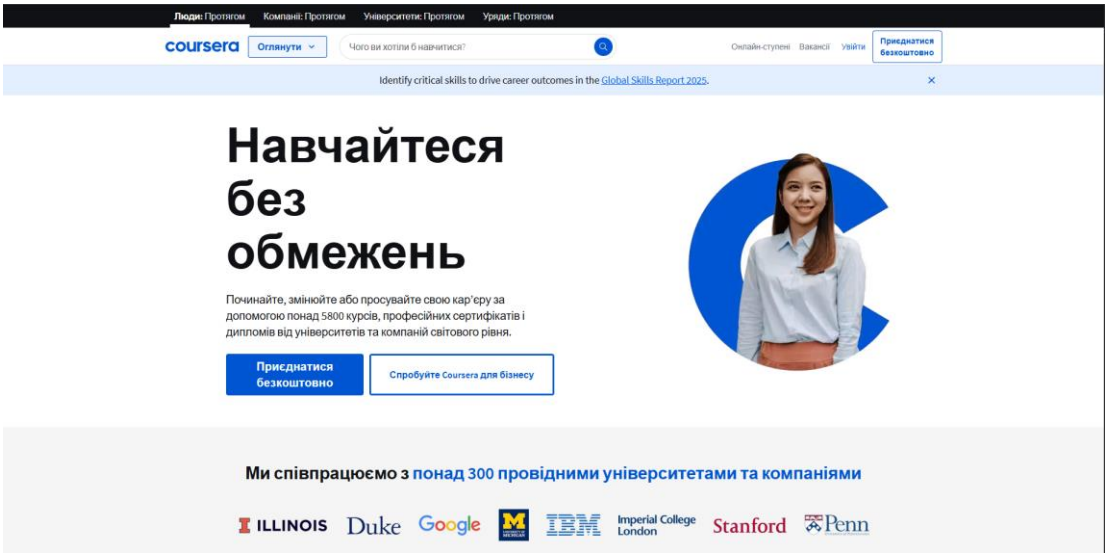


Рис 1.8 Платформа Coursera

Проте Coursera має і певні обмеження. Зокрема, код платформи є закритим, що зумовлює обмеженість у можливостях індивідуальної кастомізації та інтеграції сторонніх рішень. Також більшість курсів реалізуються за комерційною моделлю, що передбачає платний доступ до більшості навчальних матеріалів і сертифікатів. У безкоштовних версіях курсів обмежена взаємодія студентів із викладачами, що може знижувати рівень підтримки та мотивації учасників.

Отже, Coursera є потужною та професійною платформою для дистанційного навчання, орієнтованою на користувачів, які прагнуть отримати якісну освіту з офіційним підтвердженням, але вона вимагає фінансових ресурсів для повноцінного використання всього функціоналу.

Moodle - рис 1.9 є однією з найбільш розповсюджених систем управління навчанням (Learning Management System – LMS), яка відзначається відкритим вихідним кодом [10] і широкими можливостями для налаштування. Вона використовується в освітніх установах усього світу завдяки своїй функціональній гнучкості, адаптивності до різних педагогічних підходів та активній спільноті розробників. Moodle підтримує побудову повноцінних

електронних курсів із розгалуженою структурою модулів, оцінюванням, форумами, інтерактивними завданнями та системами зворотного зв’язку.

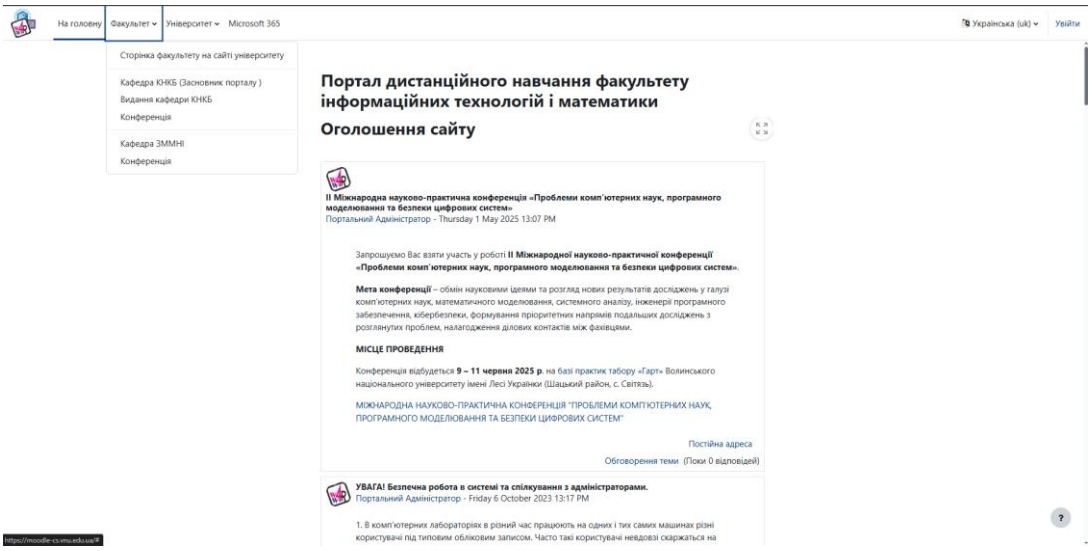


Рис 1.9 Платформа Moodle

Суттєвою перевагою платформи є її підтримка сучасних стандартів електронного навчання, таких як SCORM, LTI та H5P, що забезпечує інтеграцію з іншими системами та використання інтерактивних навчальних елементів. Відкритий код дозволяє закладам освіти повністю контролювати архітектуру платформи, адаптувати її під власні потреби, змінювати дизайн інтерфейсу, додавати функціональність за допомогою численних модулів і плагінів. Moodle активно розвивається за участі міжнародної спільноти користувачів та розробників, що сприяє його постійному оновленню та вдосконаленню.

Попри очевидні переваги, Moodle має й певні обмеження. Для повноцінного використання платформи необхідні технічні знання та досвід адміністрування, оскільки початкове налаштування та обслуговування можуть бути складними. Крім того, базова версія не завжди забезпечує оптимальну адаптацію до мобільних пристроїв, що може вплинути на зручність використання серед учасників навчального процесу. Інтерфейс Moodle, хоча й функціональний, потребує модернізації відповідно до сучасних принципів UX/UI-дизайну, що дозволило б підвищити загальну привабливість та інтуїтивність взаємодії.

Загалом, Moodle є ефективним інструментом для створення та управління освітніми середовищами, який поєднує гнучкість, масштабованість і підтримку педагогічних інновацій, проте його ефективне впровадження потребує належного технічного супроводу та стратегічного планування.

Наступним інструментом для створення та управління освітніми проєктами розглядається платформа Udemy (Рис 1.10). Udemy є глобальним маркетплейсом онлайн-курсів, заснованим у 2010 році, який посідає провідне місце серед ресурсів для здобуття нових знань та професійних компетенцій. Особливістю платформи є широкий спектр курсів, що охоплюють різні сфери – від програмування і дизайну до кулінарії та особистісного розвитку.

Платформа характеризується високою гнучкістю та доступністю, оскільки курси можна проходити з будь-якого пристрою і з будь-якого місця при наявності облікового запису. Важливою перевагою є надання сертифікатів про завершення курсів, які можуть бути використані для поповнення портфоліо та додавання до резюме. Крім того, користувачі мають можливість звертатися до експертів зі всього світу для отримання допомоги в процесі навчання або вирішення професійних питань.

Проте, слід враховувати і певні недоліки платформи. Зокрема, через велику кількість доступних курсів якість навчального контенту може значно варіюватися. Існують випадки, коли навіть дорогі курси не містять достатньо цінної інформації, тому користувачам рекомендується уважно вивчати відгуки перед придбанням навчальних матеріалів.

Отже, Udemy можна вважати ефективним інструментом для здобуття нових знань і професійної самореалізації, особливо для користувачів, які вже мають певні навички та прагнуть їх розвивати.

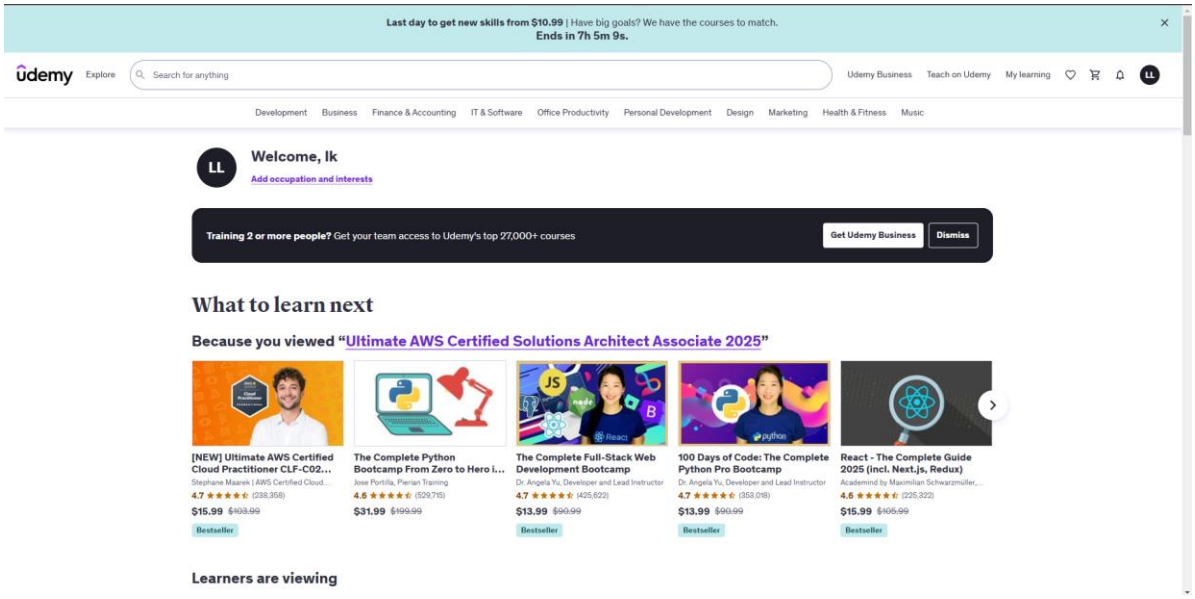


Рис 1.10 Вигляд головної сторінки Udemy

Розглянемо діяльність українського розробника Івана Примаченка, який є засновником одного з найбільших українських освітніх вебзастосунків – Prometheus. Закінчивши навчання у Київському національному університеті імені Тараса Шевченка, він отримав рекомендації щодо необхідності інвестицій у розмірі близько двох мільйонів гривень для відкриття бізнесу в освітній сфері. Проте йому вдалося створити успішний проєкт з початковим бюджетом усього дві тисячі гривень [17].

Prometheus вирізняється кількома ключовими особливостями та перевагами. Переважна більшість курсів на платформі доступна безкоштовно, що забезпечує широке охоплення якісної освіти, наданої провідними українськими викладачами. Платформа співпрацює з провідними університетами України, зокрема Київським національним університетом імені Тараса Шевченка, Київським політехнічним інститутом, Українським католицьким університетом, а також із провідними IT-компаніями та державними установами. Вибір курсів є досить широким, охоплюючи різні галузі знань – від програмування до психології, економіки та історії.

Формат навчання на платформі є зручним і включає перегляд навчальних відеоматеріалів, проходження тестів та виконання самостійних завдань для закріплення знань. Особливу увагу приділено користувацькому інтерфейсу:

дизайн сайту є інтуїтивно зрозумілим як для школярів, так і для дорослих користувачів (Рис 1.11).

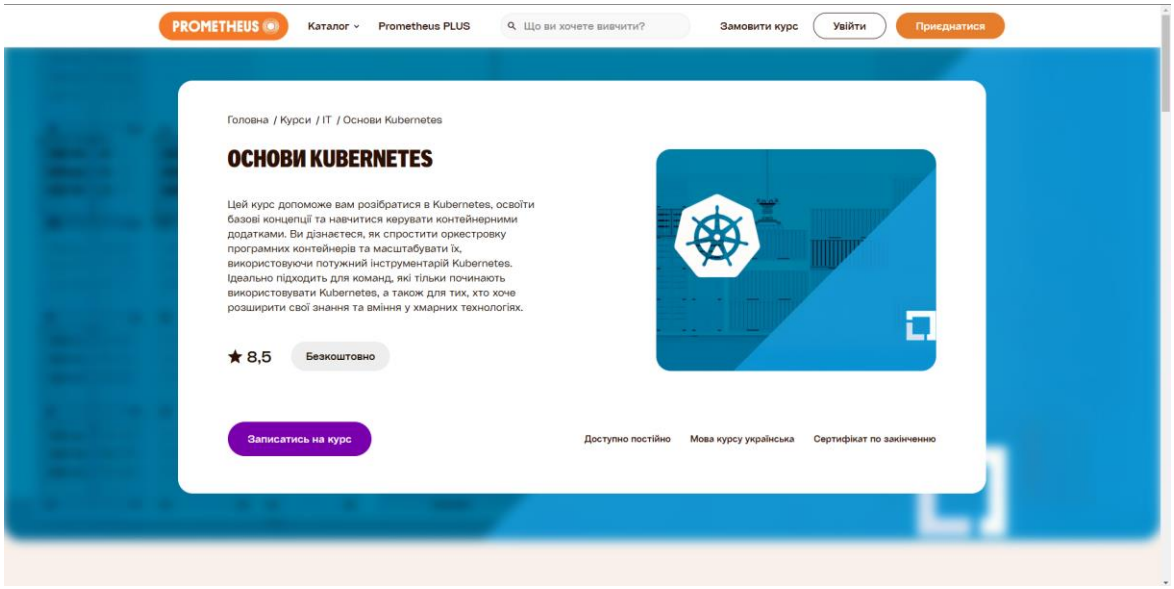


Рис 1.11 Оформлення сайту Prometheus

Серед недоліків можна відзначити обмежений вибір курсів у певних галузях. Наприклад, під час пошуку курсів з англійської мови користувач може отримати невелику кількість результатів, що не завжди відповідає очікуванням.

Отже, платформа Prometheus є якісним аналогом зарубіжних освітніх ресурсів і демонструє динамічний розвиток у сфері онлайн-освіти в Україні.

Платформа Medium є ще одним представником застосунків для публікації контенту, що вимагає реєстрації користувачів. Заснована у 2012 році співзасновником Twitter Еваном Вільясом, ця платформа має на меті створити простір, де кожен користувач може безкоштовно ділитися інформацією та особистим досвідом. Користувачі можуть переглядати та коментувати публікації безпосередньо в межах платформи, що сприяє активній взаємодії та обміну думками.

Важливою перевагою Medium є інтуїтивно зрозумілий інтерфейс із мінімалістичним дизайном, що забезпечує зручність користування (Рис 1.12). Вся необхідна функціональність розміщена на головній сторінці, що виключає потребу у тривалому пошуку інструментів для створення публікацій. Перехід до

створення нового допису здійснюється через кнопку “Write”, що веде користувача до редактора статей, виконаного в тому ж мінімалістичному стилі, із чітким розподілом полів для заповнення.

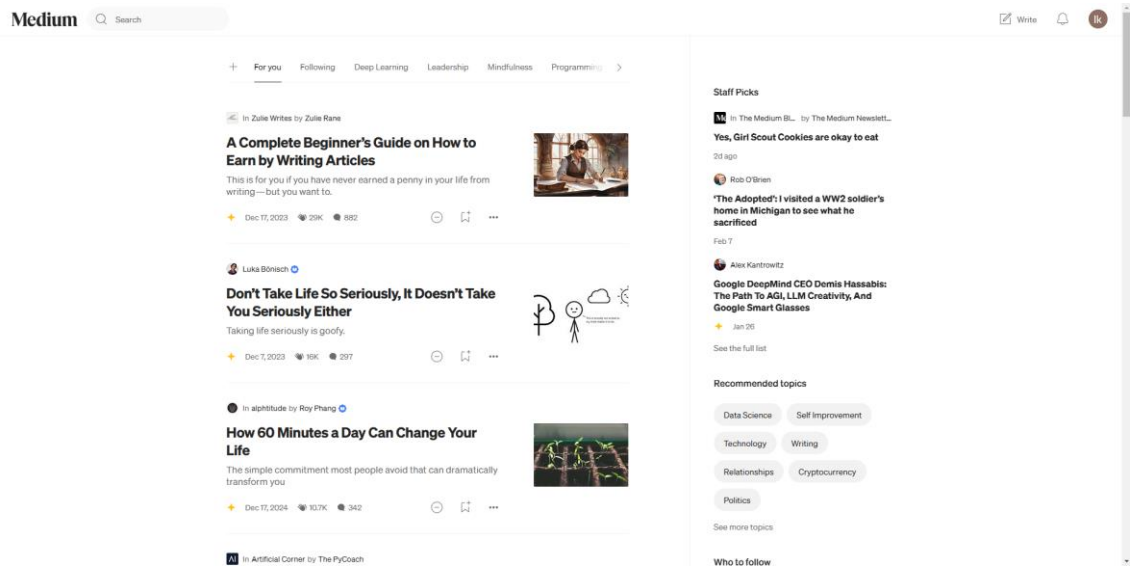


Рис 1.12 Головна сторінка Medium

Процес публікації є простим і швидким, після завершення редагування достатньо натиснути кнопку “Public”, і стаття миттєво з’являється у власній бібліотеці користувача.

Таким чином, Medium виступає як платформа, що дає змогу кожному користувачеві поділитися власними історіями та обмінятися досвідом із широкою аудиторією. Завдяки простоті та зручності інтерфейсу, цей сервіс є потужним інструментом для створення та розповсюдження публікацій.

Аналіз зазначених платформ свідчить про наявність кількох ключових тенденцій, які варто врахувати при створенні власного вебзастосунку:

- Необхідність забезпечення інтерактивності, зворотного зв’язку та адаптивного навчання (приклади: Coursera, Moodle).
- Підтримка мультимедійного контенту (відео, тести, інтерактивні блоки).
- Забезпечення гнучкості у структурі курсів, адаптації під індивідуальні потреби користувачів.

- Простота використання як для студентів, так і для викладачів.
- Важливість мобільної адаптивності та підтримки роботи в умовах обмежених ресурсів.

Власна розробка може поєднувати переваги відкритих рішень (Moodle) з інтуїтивністю Udemy або Telegraph, при цьому додати сучасні компоненти, як-от чат-боти, аналіз освітньої аналітики та можливості гейміфікації, що сприятиме залученню користувачів і підвищенню ефективності навчального процесу.

РОЗДІЛ 2.
РОЗРОБКА ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ДЛЯ ОНЛАЙН-КУРСІВ

2.1. Постановка задачі та визначення вимог до вебзастосунку онлайн-курсів

Метою даної роботи є розробка вебзастосунку "SkillUp", що надасть користувачам функціональну платформу для створення та проходження онлайн-курсів. Для успішної реалізації поставленої мети, на основі аналізу предметної області та дослідження існуючих аналогів, було сформульовано наступні ключові вимоги до програмного продукту:

- Застосунок повинен надавати можливість реєстрації нових користувачів (учнів, студентів, викладачів) за допомогою електронної пошти та пароля. Цей функціонал має на меті спрощення взаємодії та забезпечення необхідного рівня безпеки й конфіденційності даних. Автентифіковані користувачі отримують доступ до системи на основі раніше наданих облікових даних;

- Користувачі повинні мати доступ до своєї особистої інформації (ім'я, прізвище) з можливістю її редагування, а також зміни пароля. Це забезпечить комфортну та гнучку взаємодію із застосунком;

- Кожен користувач повинен мати можливість переглядати всі доступні курси та обирати необхідні. Передбачається реалізація функціоналу пошуку курсів. Ключовою функцією вебзастосунку є надання можливості користувачам (викладачам) створювати власні онлайн-курси, що включає додавання навчальних матеріалів, тестів, визначення категорії курсу та його структури;

- Для покращення навчального процесу та забезпечення зворотного зв'язку користувачі повинні мати можливість залишати коментарі до курсів.

- Вебзастосунок має бути розрахований на одночасне використання значною кількістю користувачів. Безпека даних повинна забезпечуватися шляхом SSL-шифрування

Аналіз зазначених умов призвів до висновку, що для комфортного та безпечного використання системи необхідно визначити чіткі вимоги до взаємодії користувачів із застосунком:

1. Для повноцінного використання функціоналу системи кожен користувач повинен пройти процедуру реєстрації, надавши особисту інформацію та актуальну електронну пошту.

2. Після успішної реєстрації користувач може здійснити вхід до системи, використовуючи свої облікові дані.

3. Автентифікований користувач отримує можливість обирати курси для навчання або створювати власні. Незареєстрованим користувачам може бути доступний обмежений функціонал, наприклад, лише перегляд інформації про курси.

4. При створенні курсу користувач (викладач) повинен вказати категорію, до якої належатиме курс, та визначити його основні параметри.

5. Після створення курсу користувач має можливість редагувати його зміст, назву, категорію та інші параметри.

6. Створений курс автоматично з'являється в профілі користувача-автора та стає доступним для інших користувачів системи відповідно до налаштувань.

Дотримання цих вимог та принципів спрямоване на забезпечення зручності використання, ефективності навчального процесу та створення комфортного середовища для всіх учасників освітньої платформи "SkillUp".

2.2. Вибір моделі розробки програмного засобу

Для розробки даного вебзастосунку було обрано ітераційну модель. Ця модель передбачає розбивку всього процесу розробки на серію коротких циклів або ітерацій, кожен з яких включає етапи планування, аналізу, проєктування, реалізації та тестування (рис. 2.1). Основна перевага такого підходу полягає у можливості поступового нарощування функціоналу та гнучкого реагування на зміни вимог чи виявлені проблеми.

Рис 2.1 – Ітеративна модель

Ключовою ідеєю ітераційної моделі є заміна єдиного тривалого процесу розробки послідовністю менших етапів та мініциклів. Це дозволяє не лише поступово вдосконалювати окремі компоненти системи, але й інтегрувати зміни та новий функціонал у вже розроблені частини продукту [19]. Основні етапи, що повторюються в кожній ітерації, включають:

- Аналіз вимог та планування розробки;
- Розробка архітектури системи та вибір потрібних технологій;
- Реалізація та тестування;
- Оцінка та зворотній зв’язок;
- Наступні ітерації

Обраний метод поєднує в собі необхідні кроки для створення якісного вебзастосунку. Застосування ітераційної моделі до розробки проєкту "SkillUp" дозволяє ефективно управляти ризиками, оперативно впроваджувати зміни та забезпечувати високу якість кінцевого продукту. Кожна ітерація завершується створенням працюючої версії програмного забезпечення з новим або покращеним функціоналом, що дає можливість отримувати ранній зворотний зв'язок та вчасно вносити корективи на наступних етапах розробки.

2.3. Загальний опис проекту

Для реалізації вебзастосунку онлайн-курсів "SkillUp" було обрано архітектурний підхід Clean Architecture у поєднанні з клієнт-серверною моделлю (Client-Server). Такий вибір зумовлений прагненням забезпечити гнучкість, масштабованість системи та полегшити її підтримку й модифікацію в майбутньому.

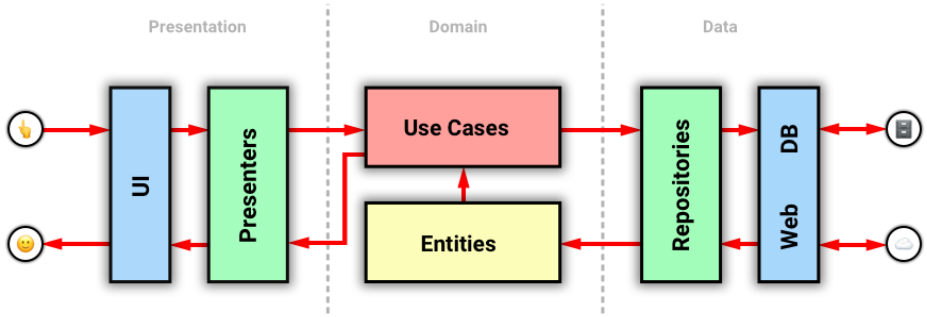


Рис 2.2 Clean Architecture будова системи

Clean Architecture, як показано на рисунку 2.2, передбачає логічний поділ системи на концентричні шари, кожен з яких має свою зону відповідальності та чітко визначені залежності. У контексті нашого проєкту це означає:

- Зовнішній шар включає конкретні технології та інструменти, такі як вебфреймворк Laravel, систему управління базами даних MySQL та бібліотеку React.js для користувацького інтерфейсу (UI).
- Шар адаптерів інтерфейсів (Interface Adapters) містить компоненти, що перетворюють дані з формату, зручного для зовнішніх інструментів, у формат, придатний для використання внутрішніми шарами, і навпаки. Тут розташовуються контролери Laravel, які обробляють HTTP-запити.
- Шар варіантів використання (Use Cases / Application Business Rules) визначає специфічну для застосунку бізнес-логіку, керуючи потоками даних та взаємодією між сутностями

- Внутрішній шар містить ключові бізнес-об'єкти (сутності) та загальні бізнес-правила, що є незалежними від конкретної реалізації застосунку.

Для роботи з даними на серверній стороні планується використання патерну "Репозиторій" (Repository Pattern). Цей патерн дозволяє створити абстрактний шар між бізнес-логікою та механізмами доступу до даних, ізолюючи таким чином деталі конкретної реалізації (наприклад, Eloquent ORM) від основного коду застосунку. Це спрощує тестування та потенційну заміну джерела даних у майбутньому.

Загалом, вибір Clean Architecture, Client-Server підходу та Repository Pattern забезпечує якість, надійність і гнучкість вебзастосунку для онлайн-курсів. Це спрощує розробку, підтримку та подальше розширення системи.

2.4. Обґрунтування вибору інструментальних засобів розробки

Для реалізації вебзастосунку, який включає клієнтську (front-end) та серверну (back-end) частини, було обрано інтегровані середовища розробки (IDE), здатні забезпечити високий рівень зручності та продуктивності в процесі програмування. Зокрема, використано інструменти компанії JetBrains: WebStorm для розробки клієнтської частини з використанням React.js та PhpStorm для створення серверної логіки на основі фреймворку Laravel і мови PHP. Застосування вказаних середовищ дозволяє ефективно структурувати процес розробки, чітко розмежовуючи роботу над окремими компонентами системи.

Середовище WebStorm є оптимальним для front-end розробки, оскільки забезпечує повноцінну підтримку JSX та TypeScript, що відповідає технологічним вимогам сучасного інтерфейсного програмування. Функціонал автоматичного завершення коду, на відміну від інших середовищ, не потребує додаткових модулів, що підвищує швидкість написання коду. Засоби інтеграції з системами контролю версій Git та інструментами CI/CD забезпечують оперативну роботу з репозиторіями, зокрема створення нових гілок та здійснення комітів. Наявність зручного відлагоджувача сприяє ефективному моніторингу та усуненню помилок під час виконання коду.

PhpStorm, у свою чергу, забезпечує високий рівень інтеграції з фреймворком Laravel, зокрема підтримку Artisan-команд, генерацію маршрутів та доступ до внутрішніх компонентів. Крім того, вбудована підтримка Composer дає змогу ефективно управляти залежностями. Інструменти для роботи з базами даних, що доступні в середовищі, значно полегшують процес моделювання та супроводу структур даних.

Таким чином, застосування WebStorm і PhpStorm у рамках реалізації даного проєкту сприяє підвищенню якості програмного коду, оптимізації процесу розробки та досягненню високої ефективності у створенні функціонального вебзастосунку.

2.4.1. Вибір PHP/Laravel для розробки серверної частини

Вибір мови програмування PHP та фреймворку Laravel для реалізації серверної частини вебзастосунку є цілком обґрунтованим і базується на низці технічних та практичних переваг. PHP належить до числа найпоширеніших мов, орієнтованих на веброзробку, що обумовлено широкою підтримкою спільноти, наявністю великої кількості бібліотек, фреймворків та ґрунтовною документаційною базою. Простота синтаксису, невисокий поріг входу, а також зручність у підтримці та супроводі коду роблять PHP ефективним інструментом для розробників різного рівня кваліфікації.

Фреймворк Laravel, що побудований на основі PHP, вважається одним з найпотужніших засобів для створення масштабованих і безпечних вебзастосунків. Його застосування забезпечує розробникам елегантний синтаксис, високу продуктивність та наявність широкого спектру функціональних можливостей уже на базовому рівні.

Основні переваги використання Laravel:

- Забезпечується захист від типових вразливостей завдяки вбудованим механізмам безпеки, які ефективно протидіють SQL-ін'єкціям, CSRF-атакам (Cross-Site Request Forgery) та XSS (Cross-Site Scripting).

- Підтримується масштабованість шляхом реалізації горизонтального масштабування, що дозволяє забезпечити стабільну роботу в умовах підвищеного навантаження та у розподілених системах.

- Реалізується гнучкість за рахунок можливості використання middleware, зовнішніх пакетів та розширень, які дозволяють адаптувати функціонал фреймворку до конкретних потреб застосунку.

- Забезпечується зручна реалізація RESTful API завдяки вбудованим інструментам, які спрощують створення та супровід прикладних інтерфейсів.

Ключовим чинником при виборі Laravel стала його розвинена екосистема, яка дозволяє прискорити процес розробки, забезпечити високу якість програмного забезпечення та підтримку сучасних стандартів вебархітектури, включаючи побудову RESTful API.

2.4.2. Використання React.js для створення клієнтської частини

React.js було обрано як основний інструмент для реалізації користувацького інтерфейсу вебзастосунку «SkillUp», оскільки він поєднує високу продуктивність, гнучкість та ефективність при створенні сучасних інтерактивних вебресурсів. Його функціональні можливості значно спрощують процес розробки та забезпечують якісний користувацький досвід.

До переваг React.js належать:

- реалізація компонентного підходу, що дає змогу створювати інтерфейс із незалежних багаторазово використовуваних елементів, кожен з яких має власну логіку та структуру;
- використання віртуального DOM, який забезпечує швидке оновлення інтерфейсу та оптимізує процес рендерингу;
- наявність засобів керування станом, таких як Redux або Context API, що дозволяють ефективно організовувати обробку даних у межах застосунку;

- висока інтерактивність, швидке завантаження інтерфейсу та покращена реакція на дії користувача.

Завдяки цим функціональним характеристикам React.js забезпечує високу інтерактивність вебзастосунку, швидке завантаження інтерфейсу та покращує загальний користувацький досвід.

2.4.3. Використання MySQL для зберігання та опрацювання даних

MySQL є однією з найпоширеніших реляційних систем управління базами даних, яка характеризується високою продуктивністю, надійністю та здатністю до масштабування. Завдяки цим властивостям вона активно використовується в галузі веброзробки, де важливими є ефективність обробки даних та гнучкість системи зберігання інформації.

До основних переваг MySQL належать:

- підтримка транзакцій відповідно до стандарту ACID, що забезпечує збереження цілісності даних навіть у разі збоїв у роботі системи;
- можливість реалізації реплікації та кластеризації, що підвищує відмовостійкість і забезпечує безперервність обслуговування;
- гнучке управління правами доступу користувачів через розширені механізми аутентифікації та авторизації;
- висока сумісність з різними мовами програмування та фреймворками, що спрощує інтеграцію у багатокомпонентні програмні системи.

Інтеграція з фреймворком Laravel за допомогою Eloquent ORM значно спрощує взаємодію з базою даних. Завдяки декларативному синтаксису ORM-моделі, розробник може ефективно реалізовувати запити, зберігаючи при цьому чисту та підтримувану архітектуру додатка.

2.5. Особливості програмної реалізації

Програмна реалізація вебзастосунку "SkillUp" для створення та проходження онлайн-курсів була виконана з використанням сучасного стеку технологій, що включає PHP-фреймворк Laravel для серверної частини та JavaScript-бібліотеку React.js для клієнтської частини. Взаємодія між ними здійснюється через розроблене RESTful API, а дані зберігаються в реляційній базі даних MySQL.

2.5.1. Серверна частина (Backend – Laravel)

Серверна логіка застосунку побудована на базі фреймворку Laravel, який забезпечує надійну структуру та інструменти для швидкої розробки.

Для управління доступом користувачів до системи реалізовано механізм автентифікації. Файл routes/api.php - Рис 2.3 визначає ендпоінт POST /api/authenticate (контролер AuthenticationController - Рис 2.3), який обробляє запити на вхід. Контролер валідує надані email та пароль, і у випадку успіху генерує токен доступу за допомогою Laravel Sanctum. Цей токен використовується для подальшої автентифікації запитів до захищених ресурсів. Також реалізовано реєстрацію нових користувачів через ендпоінт POST /api/register (контролер RegisterController - Рис 2.3), де дані користувача (ім'я, прізвище, email, пароль, роль) зберігаються в базу даних, а пароль хешується для безпеки.

Для оновлення даних профілю користувача, включаючи зміну імені, прізвища, пароля та завантаження фото, використовується ендпоінт POST /api/account (контролер AccountController - Рис 2.3), захищений middleware auth:sanctum.

Рис 2.3 Головний файл з Route на категорію

Функціонал роботи з курсами реалізований у CourseController. Користувачі можуть отримувати список всіх курсів (GET /api/courses), детальну інформацію про конкретний курс (GET /api/courses/{course}) - Рис 2.4, а також курси за певною категорією (GET /api/courses/category/{course}. Автентифіковані користувачі з відповідними правами (викладачі) можуть створювати нові курси (POST /api/courses), передаючи назву, опис, ID категорії та зображення обкладинки курсу. Оновлення існуючих курсів відбувається через ендпоінт POST /api/courses/edit/{course}. Важливою є можливість для викладачів переглядати список створених ними курсів (GET /api/user/courses).

Категорії курсів управляються через CategoryController з відповідними CRUD-операціями (GET /api/categories, POST /api/categories, GET /api/categories/{category}).

Рис 2.4 Всі маршрути на отримання курсів

Структура курсу передбачає наявність секцій (уроків). SectionController відповідає за створення (POST /api/courses/sections), отримання списку секцій для курсу (GET /api/courses/{course}/sections), отримання конкретної секції (GET

/api/courses/sections/{section}), її оновлення та видалення. Секції можуть містити текстовий контент, описи та відеоматеріали.

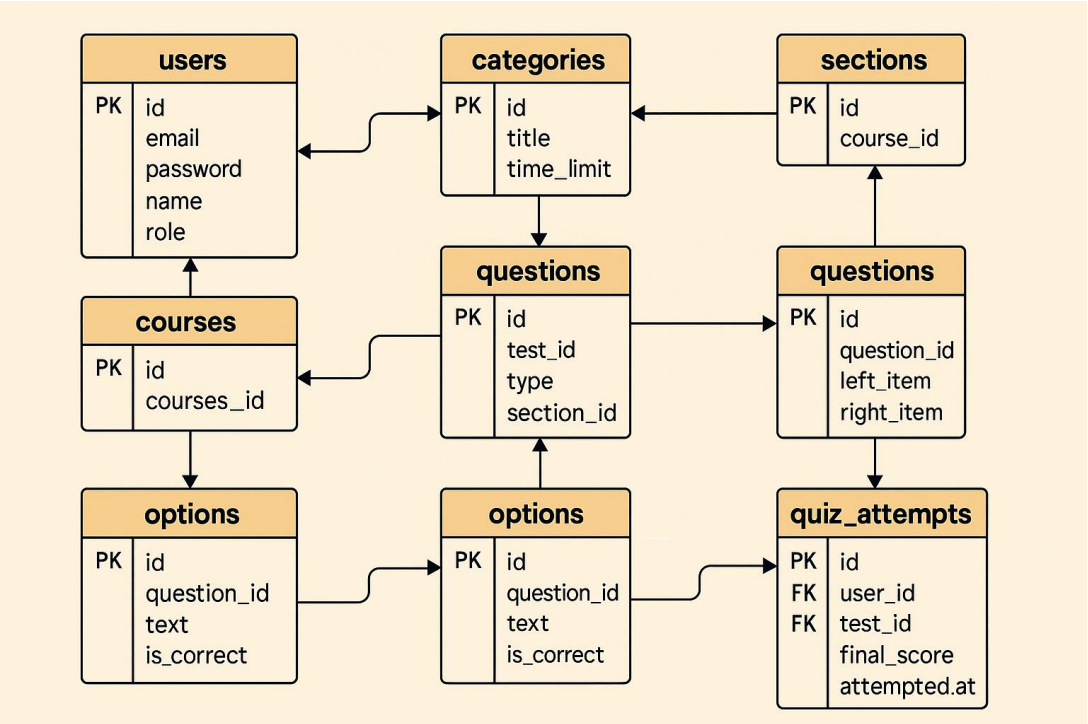
До кожної секції можуть бути прив'язані тести. TestController обробляє створення нових тестів (POST /api/courses/section/tests), їх оновлення (POST /api/courses/section/tests/{test}) та видалення. Кожен тест може містити назву, обмеження за часом (загальне або на питання) та набір питань різних типів (одиночний вибір, множинний вибір, на встановлення відповідності). Питання можуть мати текстовий опис, бали за правильну відповідь та зображення. Валідація даних при створенні та оновленні тестів реалізована за допомогою CreateTestRequest та детальної валідації в методі update контролера TestController.

Користувачі можуть проходити тести, прив'язані до секцій. Запит на отримання даних тесту для проходження надходить на ендпоінт GET /api/courses/section/tests/{test} (TestController@sectionTest) або GET /api/courses/tests/{id} (TestController@show), який повертає структуру тесту з питаннями, опціями та зображеннями. Після завершення тесту, відповіді користувача надсилаються на ендпоінт POST /api/quiz/attempts (QuizAttemptController@store). Контролер обробляє відповіді, розраховує кількість правильних відповідей, загальний бал та відсоток, і зберігає спробу в базі даних у таблиці quiz_attempts. Користувачі також можуть переглядати історію своїх спроб (GET /api/quiz/attempts) та деталі конкретної спроби (GET /api/quiz/attempts/{id}).

Для завантаження зображень (аватари користувачів, обкладинки курсів, зображення до питань тестів) використовується бібліотека Spatie MediaLibrary. Наприклад, в AccountController при оновленні профілю, якщо передано файл avatar_img, він додається до медіаколекції користувача. Аналогічно, CourseController та TestController обробляють завантаження зображень для курсів та питань.

База даних включає таблиці для зберігання інформації про користувачів (users), курси (courses), категорії (categories), секції (sections), тести (tests),

питання (questions) - Рис 2.5, опції до питань (options), пари для питань на відповідність (match_pairs) та спроби проходження тестів (quiz_attempts). Зв'язки між таблицями визначені за допомогою зовнішніх ключів (наприклад, курс має author_id та category_id; секція має course_id; тест має section_id).



2.5 ER-діаграма бази даних

2.5.2. Клієнтська частина (Frontend – React.js)

Клієнтська частина застосунку розроблена з використанням бібліотеки React.js, що дозволяє створювати динамічний та інтерактивний користувацький інтерфейс. Для управління станом, маршрутизації та взаємодії з API використовуються відповідні інструменти та підходи.

Точкою входу є файл main.tsx, де ініціалізується кореневий React-компонент App. Компонент App налаштовує маршрутизацію за допомогою react-router-dom та обгортає застосунок у ThemeProvider від Material-UI (для застосування кастомної теми theme.tsx) та AuthProvider (для управління станом автентифікації).

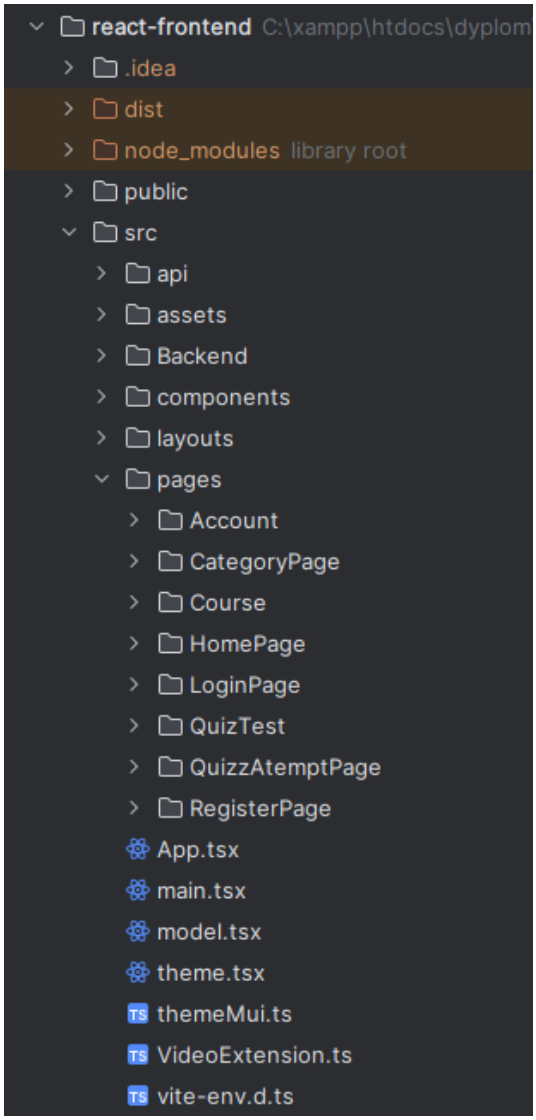


Рис 2.6 Структура проєкту

Основні сторінки застосунку представлені компонентами в директорії src/pages: HomePage - Рис 2.7, LoginPage Рис - 2.9, RegisterPage Рис - 2.10, CategoryPage Рис - 2.8, CoursePage Рис - 2.11, CourseCreatePage, CourseEditor, AccountSetting, AccountCourses, QuizTestPage та інші.

Перевикористовувані UI-елементи, такі як навігаційна панель (Navbar.tsx), картки курсів, форми, винесені в окремі компоненти в директорії src/components.

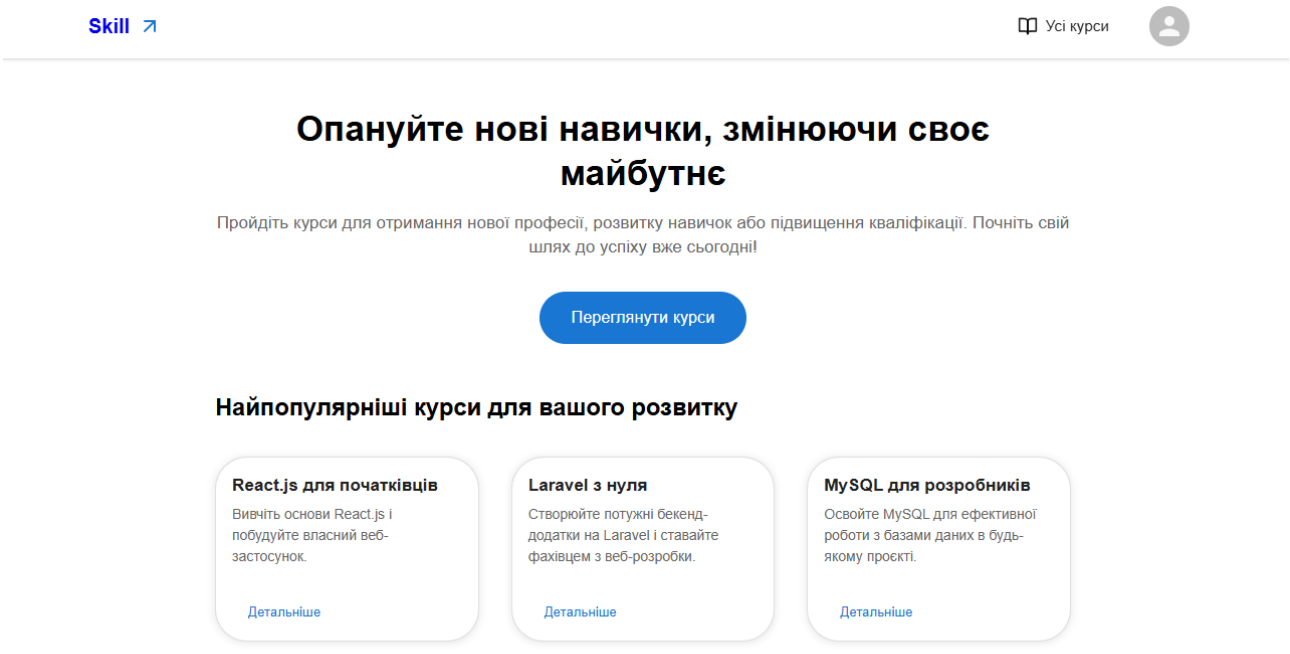


Рис 2.7 Головна сторінка застосунку (HomePage).

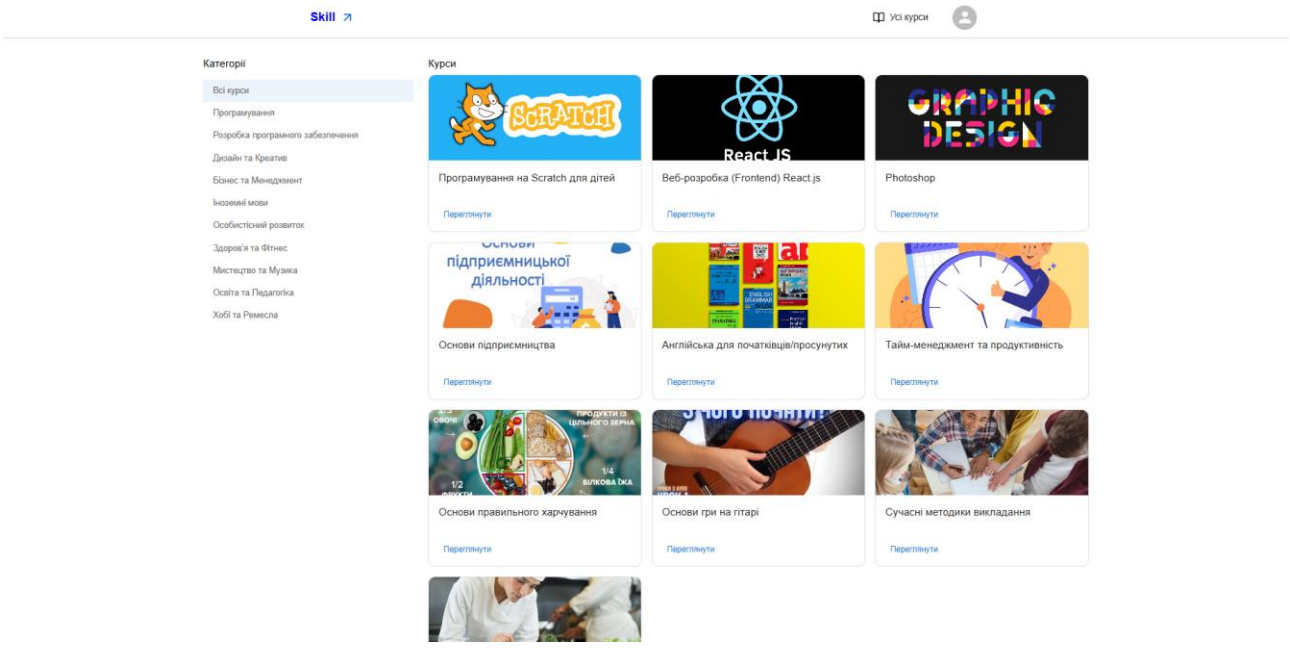


Рис 2.8 Сторінка списку курсів або сторінка категорії (CategoryPage).

Стан автентифікації користувача керується за допомогою AuthContext (файл src/Backend/Auth.tsx). AuthProvider при ініціалізації намагається відновити інформацію про користувача з localStorage. Він надає функції login (зберігає дані користувача та токен, встановлює заголовок Authorization для axios) та logout (видаляє дані та токен). Компоненти, яким потрібен доступ до

інформації про користувача або функцій login/logout, використовують useContext(AuthContext).

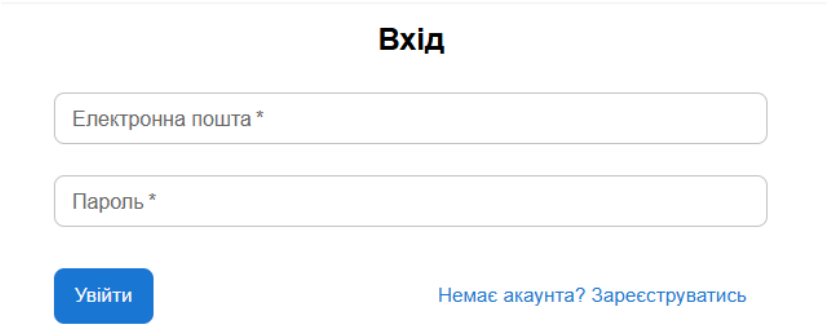


Рис 2.9. Форма логіну (LoginPage.tsx).

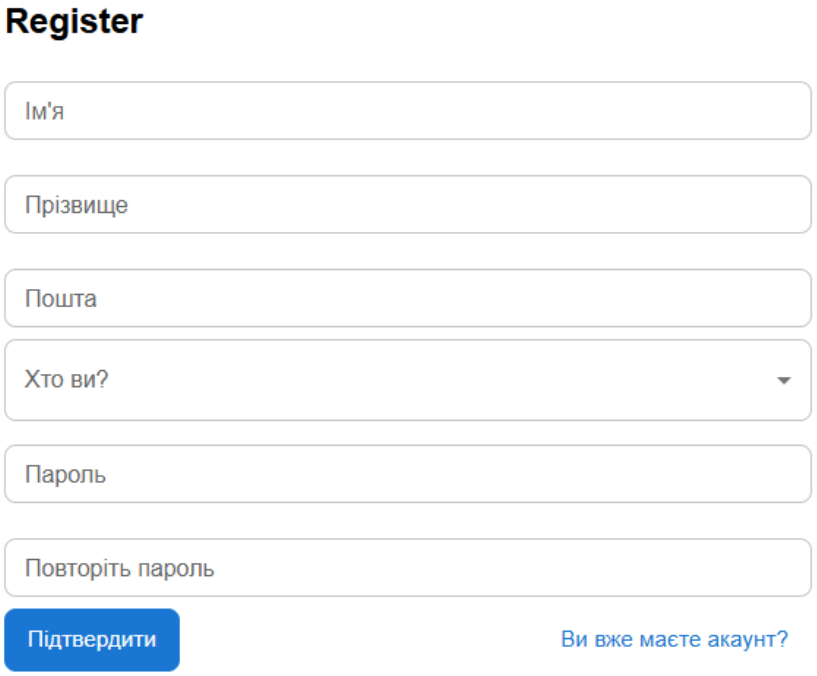


Рис 2.10. Форма реєстрації (RegisterPage.tsx).

Для взаємодії з Laravel backend використовується бібліотека axios. У файлі src/api/api.tsx створено екземпляр apiClient з базовим URL та налаштованим заголовком Authorization, який отримує токен з localStorage. Функції для виконання конкретних API-запитів (наприклад, GetCourses, CreateCourse, LoginApi, AccountSettingsSave) винесені в окремі файли (наприклад, src/api/Course.ts, src/api/Login.ts). Ці функції інкапсулюють логіку HTTP-запитів та обробку відповідей.

Для створення форм (логін, реєстрація, створення/редагування курсу, налаштування акаунту) активно використовується бібліотека Formik у поєднанні з Yup для валідації даних на стороні клієнта. Наприклад, у компоненті Login.tsx, Formik управляє станом форми, обробкою відправки та відображенням помилок валідації, визначених у validationSchema за допомогою Yup. Аналогічно побудований компонент CourseCreatePage.tsx, де Formik використовується для збору даних про новий курс, включаючи завантаження файлу зображення обкладинки.

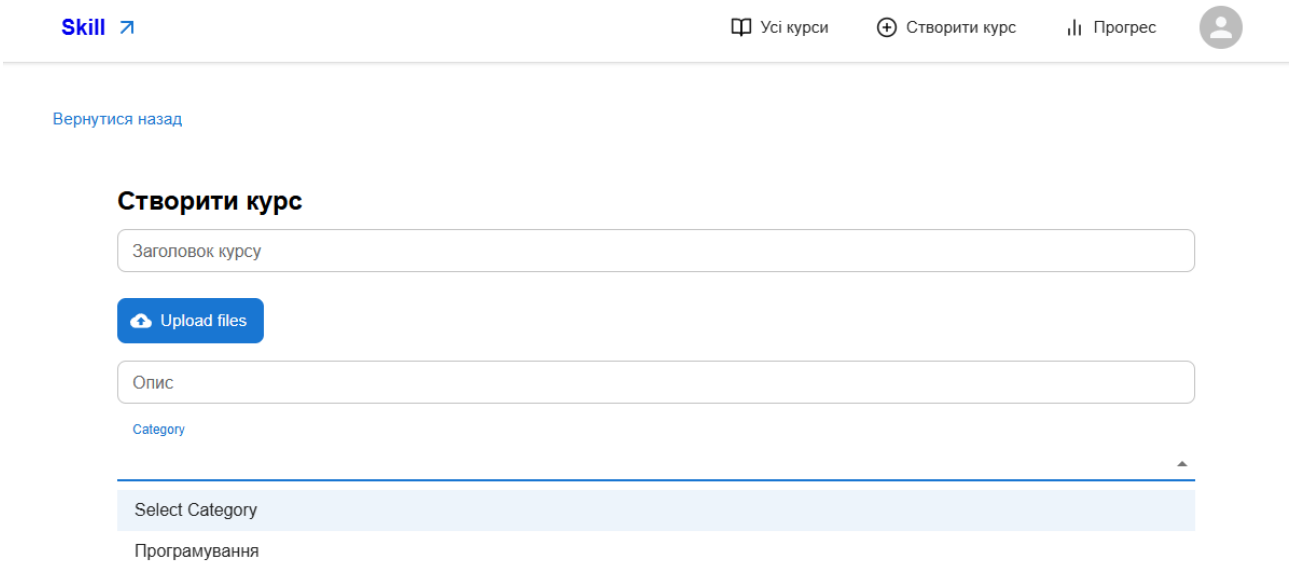


Рис 2.11 - Форма створення курсу (CourseCreatePage.tsx)

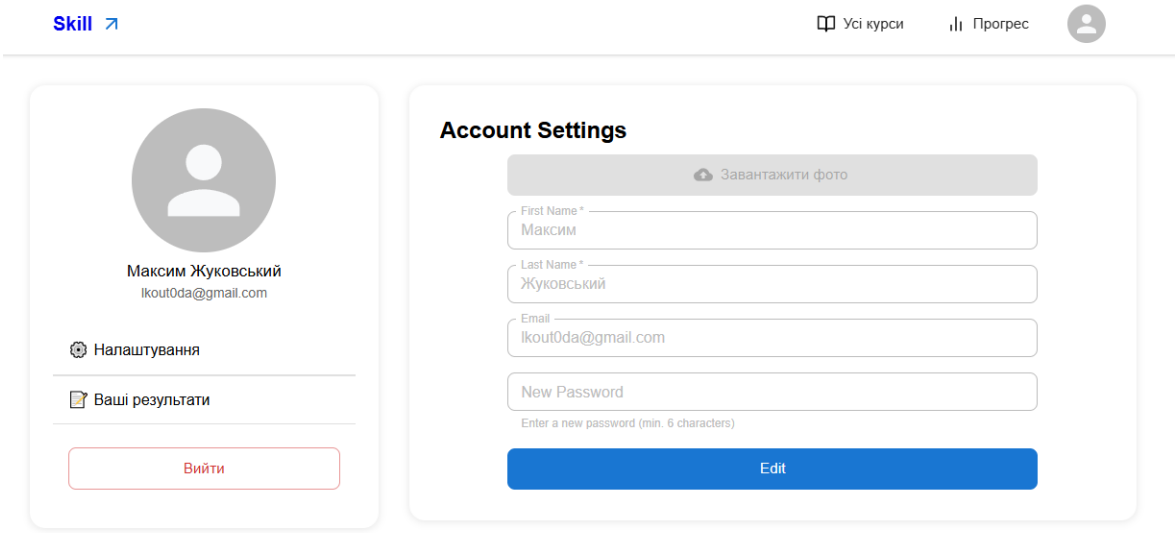
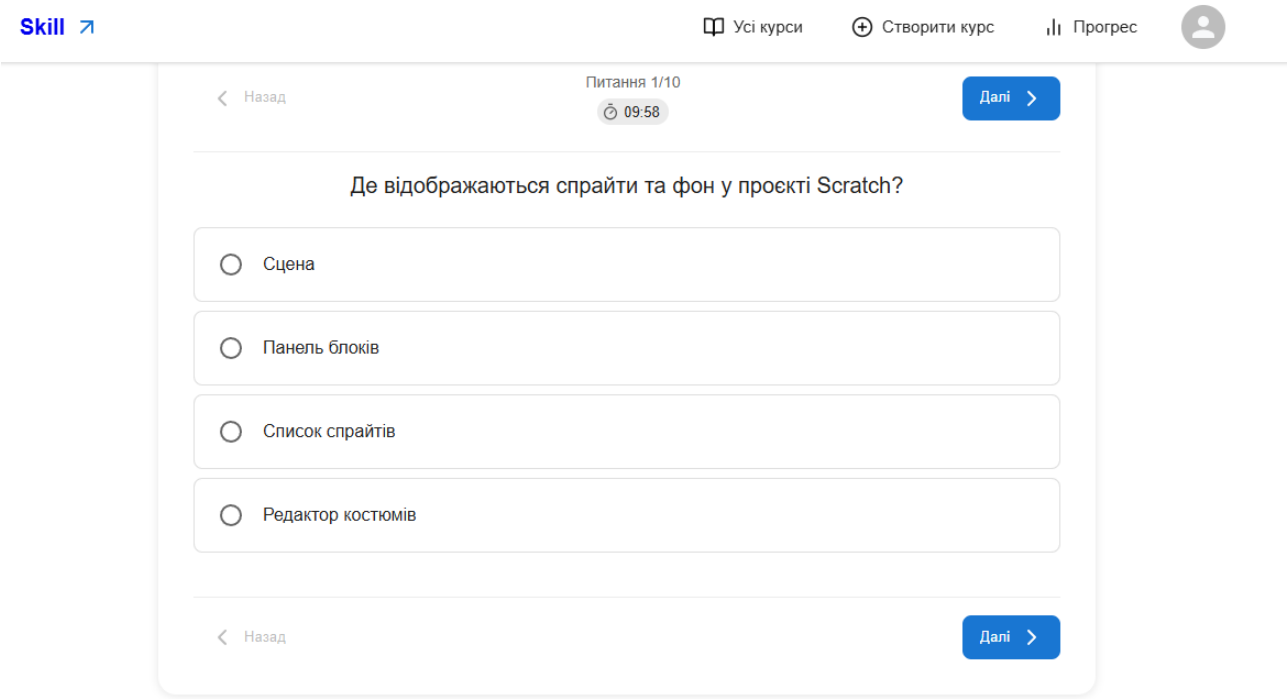


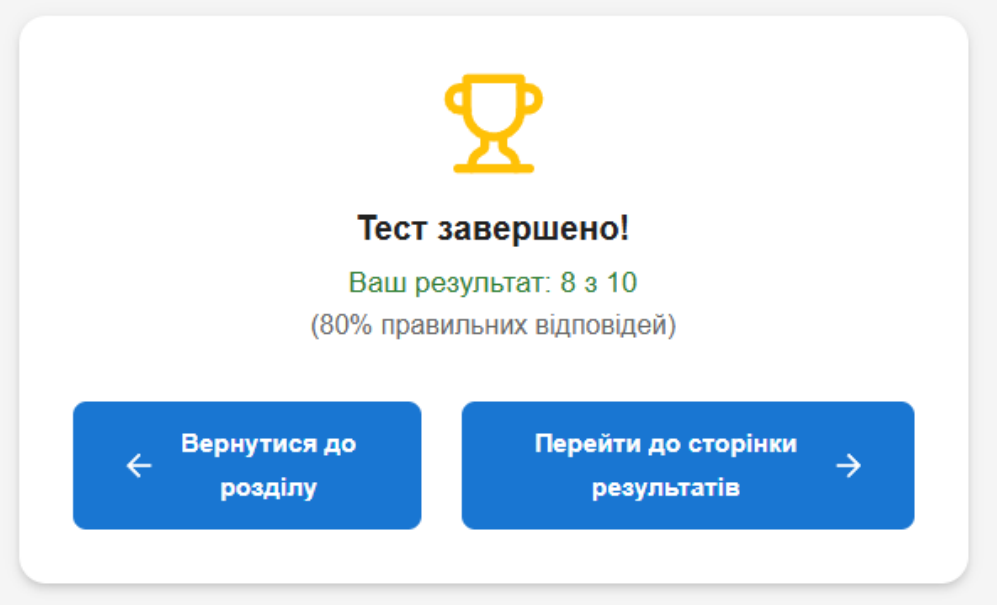
Рис 2.12. Сторінка налаштувань акаунту

Компоненти, такі як CoursePage.tsx, відповідають за відображення детальної інформації про курс, включаючи його опис, секції та можливість переходу до тестів. Для відображення списку курсів (наприклад, на головній сторінці або сторінці категорії) можуть використовуватися компоненти-картки, що показують назву курсу, зображення та коротку інформацію.

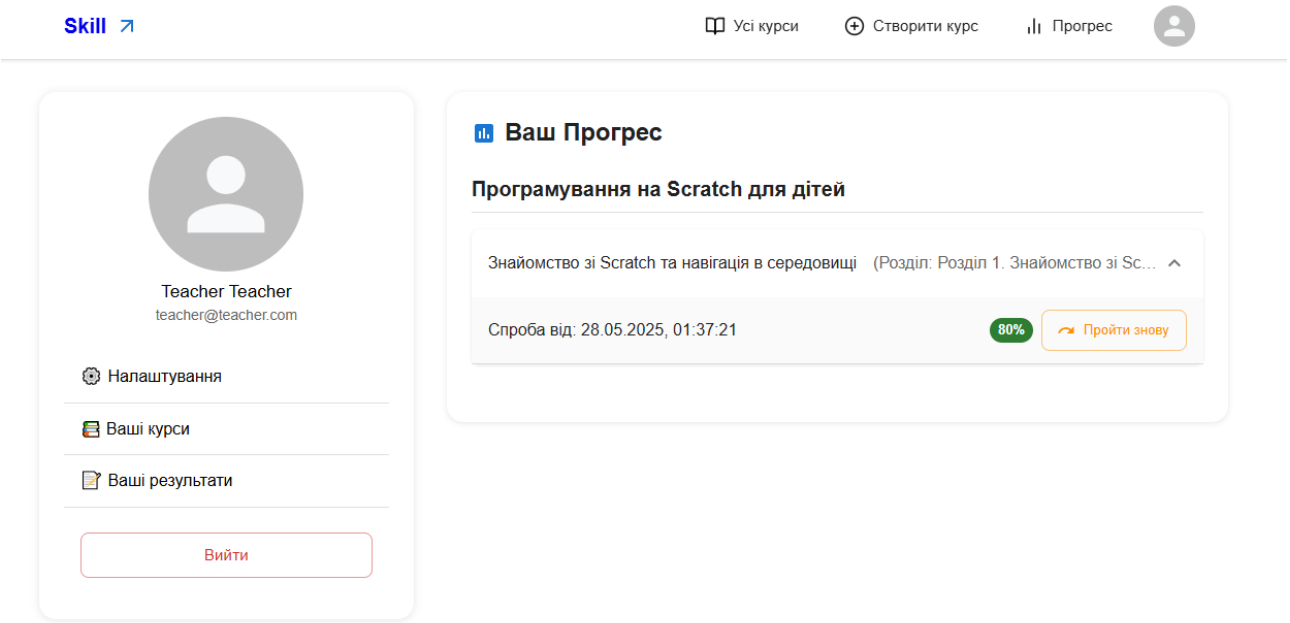
Компонент QuizTestPage.tsx є ключовим для функціоналу проходження тестів. Він отримує ID тесту з URL, завантажує дані тесту з API, управляє станом гри (завантаження, старт, гра, завершено), відображає поточне питання та варіанти відповідей. Для питань типу "на встановлення відповідності" використовується бібліотека react-beautiful-dnd для реалізації функціоналу перетягування. Компонент відслідковує відповіді користувача, управляє таймером (якщо він налаштований для тесту) та після завершення відправляє результати на сервер. Для відображення початкового екрану тесту та екрану результатів використовуються компоненти StartScreen.tsx та GameOverScreen.tsx. Після завершення курсу, користувач може отримати сертифікат про проходження курсу Рис 2.16



2.13. Інтерфейс проходження тесту (QuizTestPage.tsx)



2. 14 Екран результатів тесту (GameOverScreen.tsx).



2.15 Сторінка прогресу користувача з історією спроб (AccountProgressPage.tsx).

Для побудови користувацького інтерфейсу активно використовується бібліотека компонентів Material-UI (MUI). У файлі theme.tsx визначено кастомну тему (кольори, типографіка, стилі для компонентів), яка застосовується до всього застосунку через ThemeProvider. Компоненти MUI, такі як Button, TextField, Container, Box, AppBar, Avatar, Paper, Checkbox, Radio, List, Grid, Chip, використовуються для створення візуально привабливого та

узгодженого інтерфейсу. Наприклад, Navbar.tsx використовує AppBar, Toolbar, Button та Avatar для створення навігаційної панелі. Форми, як у CourseCreate.tsx чи AccountSetting.tsx, використовують TextField, Button, Select від MUI.

Рис 2.16 Сертифікат про проходження курсу

2.6. Організація тестування та налагодження програмного засобу

Функціональне тестування проводилось на постійній основі під час розробки кожного нового функціоналу та після інтеграції компонентів. Перевірялися ключові сценарії використання застосунку:

1. Процеси реєстрації нових користувачів та автентифікації існуючих.
2. Створення, редагування та перегляд онлайн-курсів та їх категорій.
3. Додавання та управління секціями і навчальними матеріалами в межах курсів.

Створення та проходження тестів, включаючи коректність відображення питань різних типів (одиночний вибір, множинний вибір, на встановлення відповідності) та збереження результатів.

Робота з особистим кабінетом користувача, включаючи оновлення профілю та перегляд прогресу.

Коректність завантаження та відображення медіафайлів (зображень, аватарів). Тестування користувацького інтерфейсу (UI Testing): Візуально перевірялася відповідність реалізованого інтерфейсу запланованому дизайну (якщо такий був).

Оцінювалася зручність навігації, зрозумілість елементів керування, коректність відображення тексту та графічних елементів на різних сторінках застосунку. Використання бібліотеки Material-UI значно сприяло створенню узгодженого та інтуїтивно зрозумілого інтерфейсу, що полегшило цей етап.

Перевіряється коректність взаємодії між клієнтською частиною (React.js) та серверною частиною (Laravel). Це включало перевірку відправки запитів на API ендпоінти, отримання та правильну обробку відповідей від сервера, а також коректне оновлення даних на стороні клієнта. Для швидкої перевірки API на етапі розробки могли використовуватися інструменти типу Postman.

Налагодження коду було невід'ємною частиною розробки. Для виявлення та усунення помилок використовувалися стандартні інструменти розробника, доступні в сучасних браузерах (для React.js), такі як консоль для перегляду повідомлень та помилок, інспектор елементів для аналізу DOM-структури та стилів, а також вкладка "Network" для моніторингу HTTP-запитів.

2.7. Рекомендації щодо використання та впровадження програмного засобу

Розроблений вебзастосунок "SkillUp" надає платформу для створення та проходження онлайн-курсів. Для його ефективного використання та успішного впровадження варто звернути увагу на декілька ключових аспектів.

Користувачам, які планують навчатися, рекомендується перш за все пройти реєстрацію для отримання повного доступу до функціоналу, включаючи можливість записуватися на курси та відстежувати свій прогрес. При виборі курсу варто уважно ознайомитися з його описом та структурою, а під час навчання – активно використовувати тестові завдання для самоперевірки. Авторам курсів, у свою чергу, слід зосередитися на створенні якісного, структурованого та актуального контенту, розбиваючи його на логічні секції та розробляючи ефективні тестові завдання для перевірки знань. Чітке оформлення курсу та взаємодія зі слухачами через коментарі сприятимуть його популярності та ефективності.

Щодо технічного впровадження, розгортання серверної частини на Laravel потребує відповідного серверного оточення з PHP та MySQL, а також налаштування конфігураційних файлів. Клієнтська частина на React.js компілюється у статичні файли, які розміщуються на вебсервері. Важливим є забезпечення безпеки через використання HTTPS та регулярне оновлення програмного забезпечення. Також необхідно налаштувати автоматичне резервне копіювання бази даних та файлів застосунку для запобігання втраті інформації. Базове адміністрування може включати управління обліковими записами користувачів та моніторинг системних логів для виявлення потенційних проблем.

Вебзастосунок "SkillUp" має значний потенціал для подальшого розвитку. Серед можливих напрямків удосконалення можна виокремити розширення функціоналу тестів, впровадження системи сертифікації, інтеграцію платіжних систем для платних курсів, а також додавання інтерактивних елементів, таких як

форуми чи вебінари. Покращення мобільної адаптації, розробка розширених інструментів аналітики для викладачів та впровадження більш формалізованих методів автоматизованого тестування також сприятимуть підвищенню якості та конкурентоспроможності платформи. Розгляд можливостей гейміфікації та системи сповіщень може покращити залученість користувачів.

Дотримання цих рекомендацій дозволить не лише ефективно використовувати наявні можливості "SkillUp", але й забезпечити його стабільну роботу та успішний розвиток у майбутньому.

ВИСНОВКИ

У процесі виконання дипломної роботи було успішно досягнуто поставленої мети, а саме розроблено вебзастосунок "SkillUp" для створення та проходження онлайн-курсів. Ця розробка базувалася на сучасних технологіях: PHP-фреймворку Laravel для серверної частини та JavaScript-бібліотеці React.js для клієнтської.

Робота розпочалася з аналізу актуальності онлайн-освіти та дослідження існуючих платформ, що дозволило сформулювати вимоги до функціоналу та користувацького інтерфейсу "SkillUp". Ключовими завданнями були дослідження принципів онлайн-курсів, аналіз аналогів, визначення потреб користувачів, розробка основного функціоналу (включаючи реєстрацію, створення курсів, секцій, тестів), проектування інтерфейсу та проведення тестування застосунку.

Об'єктом дослідження став процес створення вебзастосунку для онлайн-навчання, а предметом – оцінка потенційної ефективності такої платформи. Результатом роботи є функціональний вебзастосунок "SkillUp", що надає користувачам інструменти для організації навчального процесу в онлайн-форматі. Він демонструє можливість створення доступних та ефективних освітніх рішень за допомогою обраного стеку технологій.

Таким чином, всі поставлені завдання було виконано, а мета роботи – досягнута. Розроблений вебзастосунок "SkillUp" є практичним результатом дослідження та має потенціал для подальшого розвитку, зокрема через розширення функціоналу та покращення користувацького досвіду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. MDN Web Docs. Mozilla Foundation. URL: https://developer.mozilla.org/en-US/docs/Web
2. React Official Documentation. Meta Platforms, Inc. URL: https://react.dev/
3. Як створити веб-додаток: повний гайд від WEZOM. Wezom. URL: https://wezom.com.ua/ua/blog/kak-sozdat-veb-prilozhenie
4. Server-side website programming first steps. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/First_steps/Introduction
5. Клієнт-серверна архітектура. QATestLab. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/
6. Web Application Architecture: Components, Best Practices, and Types. AltexSoft. URL: https://www.altexsoft.com/blog/web-application-architecture/
7. What Is a Single-page Application? Toptal. URL: https://www.toptal.com/front-end/single-page-applications
8. Microservices at Netflix: Architectural Best Practices. F5 (Nginx). URL: https://www.f5.com/company/blog/nginx/microservices-at-netflix-architectural-best-practices
9. Прикладний рівень (Application Layer). STUD.com.ua. URL: https://stud.com.ua/94311/informatika/prikladniy_riven_application_layer#google_vignette
10. Сучасні тенденції розвитку онлайн-навчання в Україні та світі. Index Copernicus. URL: https://journals.indexcopernicus.com/api/file/viewByFileId/860062
11. Онлайн-курси в Україні: навчання для всіх. CRP.Center. URL: https://ua.crp-wroclaw.com/blog/%D0%BE%D0%BD%D0%BB%D0%B0%D0%B9%D0%BD-%D0%BA%D1%83%D1%80%D1%81%D0%B8-%D0%B2-%D1%83%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D1%96-%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F

/?srsltid=AfmBOopemqnYdlSTqPtQLQjwyrD6OiW8hhdW8GkKeqQySrmAm_Yq1bg-

12. Більше ШІ та «людських» навичок. Яким буде навчання у 2025 році? Освіторія. URL: https://osvitoria.media/experience/bilshe-shi-ta-lyudskyh-navychok-yakym-bude-navchannya-u-2025-rotsi/?utm_source=chatgpt.com
13. MVC патерн: що це таке та як він використовується у веб-розробці. IT Hillel. URL: https://blog.ithillel.ua/articles/building-dynamic-web-applications-using-mvc
14. Що таке React.js і для чого він потрібен? Dan-IT. URL: https://dan-it.com.ua/uk/blog/chto-takoe-react-js-i-dlja-chego-on-nuzhen/
15. Огляд MySQL. Ukraine.com.ua Wiki. URL: https://www.ukraine.com.ua/wiki/mysql/overview/
16. MySQL 8.4 Reference Manual: Introduction to MySQL. Oracle Corporation. URL: https://dev.mysql.com/doc/refman/8.4/en/introduction.html
17. Мені казали, що для запуску потрібен $1 млн. В мене було тільки $2 тис. Як Іван Примаченко створив платформу онлайн-освіти Prometheus. MC.today. URL: https://mc.today/uk/meni-kazali-shho-dlya-zapusku-potriben-1-mln-v-mene-bulo-tilki-2-tis-yak-ivan-primachenko-stvoriv-platformu-onlajn-osviti-prometheus/
18. Популярні моделі життєвого циклу розробки програмного забезпечення. QATestLab. URL: https://training.qatestlab.com/blog/technical-articles/popular-software-development-life-cycles/
19. Методології розробки програмного забезпечення. Evergreen. URL: https://evergreens.com.ua/ua/articles/software-development-metodologies.html
20. Ітеративна модель розробки: як гнучко створити продукт. DOU.ua. URL: https://dou.ua/lenta/articles/iterative-development-model/
21. Laravel Documentation. Taylor Otwell. URL: https://laravel.com/docs/10.x
22. API Platform Documentation. API Platform. URL: https://api-platform.com/docs/
23. OWASP Top Ten Project. OWASP Foundation. URL: https://owasp.org/www-project-top-ten/

24. Eloquent ORM (Laravel Documentation). Taylor Otwell. URL: https://laravel.com/docs/10.x/eloquent
25. Redux Toolkit Documentation. Redux Team. URL: https://redux-toolkit.js.org/
26. Material-UI (MUI) Documentation. MUI. URL: https://mui.com/material-ui/getting-started/ (дата звернення: 12.06.2025).
27. The Twelve-Factor App. Heroku. URL: https://12factor.net/
28. PHP: The Right Way. Fabien Potencier et al. URL: https://phptherightway.com/
29. Introduction to Testing (Laravel Documentation). Taylor Otwell. URL: https://laravel.com/docs/10.x/testing
30. Designing Web APIs: Building APIs That Developers Love by Brenda Jin, Saurabh Sahni, and Amir Shevat. O'Reilly Media. URL: https://www.oreilly.com/library/view/designing-web-apis/9781492026919/

ДОДАТКИ

Додаток А

Технічне завдання

1. Вступ

Технічне завдання описує веб-застосунок для створення та проходження онлайн-курсів "SkillUp". Цей проєкт являє собою веб-платформу, що дозволяє користувачам (викладачам) створювати, структурувати та публікувати навчальні курси, а іншим користувачам (студентам) – проходити ці курси, виконувати тестові завдання та відстежувати свій прогрес.

2. Підстави для розробки

Розробка проводиться на основі індивідуального завдання, поставленого науковим керівником дипломної роботи для спеціальності "Комп’ютерні науки та кібербезпека" (або вашої спеціальності).

3. Призначення розробки

Програмний продукт "SkillUp" призначений для надання інструментів для створення, управління та проходження онлайн-курсів, організації навчального процесу, тестування знань та забезпечення взаємодії між авторами курсів та студентами у зручній, інтерактивній та доступній формі.

Призначений для використання як індивідуальними користувачами, що бажають отримати нові знання або поділитися ними, так і потенційно освітніми установами чи компаніями для організації внутрішнього навчання.

4. Вимоги до програми чи програмного продукту

4.1. Вимоги до функціональних характеристик

Система повинна надавати можливість реєстрації нових користувачів (з вказанням імені, прізвища, email, пароля та ролі – студент/викладач). Забезпечувати безпечний вхід зареєстрованих користувачів за допомогою їх облікових даних (email та пароль).

Надавати можливість користувачам редагувати дані свого профілю (ім'я, прізвище, зміна пароля, завантаження/зміна фото). Користувачі (з роллю

викладача) повинні мати можливість створювати нові онлайн-курси, вказуючи назву, опис, категорію та завантажуючи зображення обкладинки.

Забезпечувати можливість редагування та видалення створених курсів. Усі користувачі повинні мати можливість переглядати список доступних курсів, фільтрувати їх за категоріями. Викладачі повинні мати можливість додавати секції (уроки) до своїх курсів, наповнюючи їх текстовим контентом, описами, завантажуючи файли (наприклад, домашні завдання) та відеоматеріали.

Забезпечувати можливість редагування та видалення секцій. До кожної секції повинна бути можливість прив'язати тестові завдання. Викладачі повинні мати можливість створювати тести з різними типами питань (одиночний вибір, множинний вибір, на встановлення відповідності), встановлювати бали за питання, додавати зображення до питань та налаштовувати часові обмеження.

Студенти повинні мати можливість проходити тести, пов'язані з секціями курсів. Система повинна коректно відображати питання, обробляти відповіді та розраховувати результати (бал, відсоток).

Результати проходження тестів повинні зберігатися в історії спроб користувача. Студенти повинні мати можливість переглядати історію своїх спроб та деталі кожної спроби.

4.2. Вимоги до надійності

1. Система повинна бути розроблена для стабільної роботи та мінімізації збоїв.
2. Забезпечити цілісність даних користувачів, курсів та результатів тестування.
3. Система повинна коректно опрацьовувати помилкові введення даних користувачем та виключні ситуації.
4. Забезпечити захист від основних веб-вразливостей (SQL-ін'єкції, XSS, CSRF).

4.3. Умови експлуатації

1. Веб-застосунок "SkillUp" повинен бути доступним користувачам через стандартні веб-браузери.
2. Для доступу до системи необхідне стабільне інтернет-з'єднання.
3. Доступність системи – 24/7, за винятком планових технічних робіт.

4.4. Вимоги до складу і параметрів технічних засобів

Будь-який персональний комп'ютер, ноутбук, планшет або смартфон з встановленим сучасним веб-браузером (Google Chrome, Mozilla Firefox, Safari, Edge останніх версій) та доступом до мережі Інтернет.

4.5. Вимоги до інформаційної і програмної сумісності

1. Клієнтська частина повинна бути сумісною з основними сучасними веб-браузерами та операційними системами (Windows, macOS, Linux, Android, iOS).
2. Серверна частина повинна взаємодіяти з клієнтською частиною через RESTful API за протоколом HTTP/HTTPS.
3. Використання стандартних форматів обміну даними (JSON).

4.6. Вимоги до транспортування і збереження

1. Дані користувачів, курсів, тестів та інша інформація повинні зберігатися в реляційній базі даних MySQL на сервері.
2. Забезпечити захист даних при передачі між клієнтом та сервером шляхом використання HTTPS.

5. Вимоги до програмної документації

1. Опис архітектури системи (Clean Architecture, клієнт-серверна модель).
2. Опис реалізованого функціоналу та його відповідності вимогам.
3. Опис структури бази даних (схеми, зв'язки).
4. Опис ключових алгоритмів та особливостей програмної реалізації (бекент та фронтент).
5. Інструкцію користувача (основні сценарії використання).
6. Результати тестування (опис проведеного мануального тестування).

7. Висновки та рекомендації щодо подальшого розвитку.

6. Техніко-економічні показники

1. Оцінка витрат на розробку (в контексті дипломної роботи – це переважно часові витрати студента).

2. Обґрунтування вибору технологій з точки зору доступності, популярності та наявності ресурсів для розробки.

3. Потенційна корисність та ефективність застосунку для організації онлайн-навчання.

7. Стадії і етапи розробки

7.1. Передбачені стадії розробки

Дослідження поняття онлайн-курсів, аналіз аналогів, формулювання вимог до системи.

Розробка архітектури (Clean Architecture), проєктування бази даних, проєктування користувацького інтерфейсу (UI/UX).

Налаштування проєкту Laravel, бази даних.

Реалізація моделей, міграцій, контролерів, маршрутів API.

Реалізація логіки автентифікації, управління курсами, секціями, тестами.

Налаштування проєкту React.

Розробка компонентів UI, сторінок, маршрутизації.

Реалізація логіки взаємодії з API, управління станом.

Інтеграція Backend та Frontend.

Тестування та налагодження: Проведення мануального тестування функціоналу, UI, інтеграції.

Підготовка документації: Написання пояснювальної записки до дипломної роботи.

7.2. Необхідні сторінки та структурні елементи на сайт

Назва елементу	Опис
HomePage (Головна	Відображення загальної інформації про платформу, можливо, список популярних курсів або категорій.

сторінка)	
LoginPage (Сторінка входу)	Форма для введення email та пароля для автентифікації.
RegisterPage (Сторінка реєстрації)	Форма для створення нового облікового запису.
CoursesPage/CategoryPage (Сторінка курсів)	Відображення списку всіх курсів або курсів певної категорії з можливістю пошуку та фільтрації.
SectionPage (Сторінка секції/уроку)	Відображення контенту конкретної секції, посилання на тест.
TestPage (Сторінка тесту)	Інтерфейс для проходження тесту: відображення питань, варіантів відповідей, таймер.
AccountDashboard (Особистий кабінет)	Загальна сторінка особистого кабінету з посиланнями на налаштування, мої курси, прогрес.
AccountSettingsPage (Налаштування акаунту)	Форма для редагування даних профілю та зміни пароля.
MyCoursesPage (Мої курси – для автора)	Список курсів, створених користувачем (викладачем), з можливістю переходу до редагування, додавання секцій.
CourseCreatePage (Сторінка створення курсу)	Форма для створення нового курсу.
CourseEditPage (Сторінка редагування курсу)	Інтерфейс для редагування існуючого курсу, управління його секціями.

SectionCreatePage (Створення секції)	Форма для додавання нової секції до курсу.
SectionEditPage (Редагування секції)	Форма для редагування існуючої секції.
TestCreatePage (Створення тесту)	Інтерфейс для створення нового тесту та додавання питань.
AccountProgressPage (Прогрес користувача)	Відображення історії проходження тестів студентом.

8. Порядок контролю і приймання

1. Контроль за виконанням кожного етапу розробки здійснюється науковим керівником.

2. Приймання роботи відбувається шляхом захисту курсової роботи, що включає:

a. Демонстрацію працездатності веб-застосунку "SkillUp".

b. Перевірку відповідності реалізованого функціоналу вимогам даного ТЗ.

c. Оцінку якості програмної реалізації та дизайну.

d. Аналіз представленої програмної документації (пояснювальної записки).

9. Додатки до технічного завдання

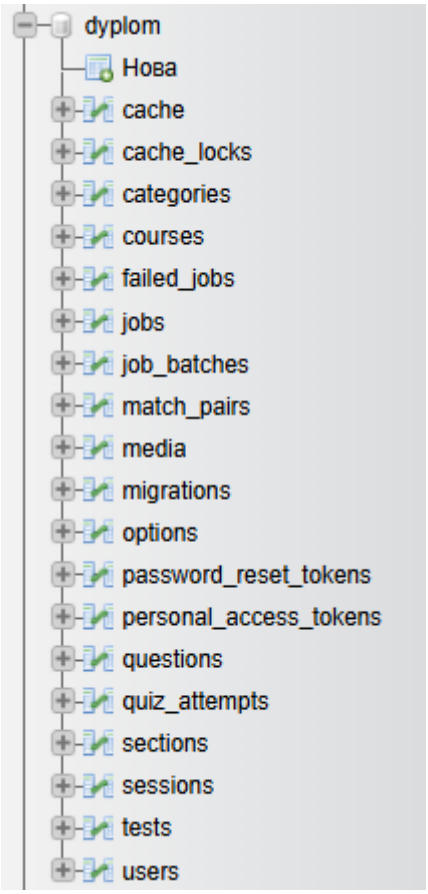


Рис.1. Усі таблиці бази даних

Додаток Б

Інструкція користувача

Для ефективного використання веб-застосунку "SkillUp" та доступу до навчальних матеріалів чи інструментів для створення курсів, необхідно спочатку зареєструватися в системі або увійти до існуючого облікового запису.

Після успішної автентифікації вам відкриються всі можливості платформи. Через навігаційну панель ви зможете легко знаходити та обирати курси для навчання, переглядати їх зміст та структуру. Для авторів курсів буде доступна функція створення нових навчальних програм, де ви зможете додавати секції, текстові матеріали, відео та файли, а також формувати тестові завдання для перевірки знань.

Під час проходження курсу ви матимете змогу вивчати матеріали секцій та проходити тести, результати яких будуть зберігатися у вашому профілі для відстеження прогресу. Автори курсів зможуть редагувати свої навчальні матеріали, додавати нові або оновлювати існуючі. В особистому кабінеті кожен користувач може керувати даними свого профілю.

АНОТАЦІЯ

Жуковський М. П. – **Проєктування та розробка вебзастосунку для онлайн-курсів з використанням фреймворку laravel, мови php та бібліотеки React.js**

Кваліфікаційна робота за спеціальністю 122 Комп'ютерні науки. – Волинський національний університет імені Лесі Українки, Луцьк. – 2025 р.

У дипломній роботі досліджено сучасні тенденції онлайн-освіти та проведено порівняльний аналіз платформ Udemy і Prometheus, а також обґрунтовано вибір технологічного стеку на основі Laravel, React.js і MySQL.

Проаналізовано вимоги до функціональності системи, включаючи автентифікацію користувачів із ролями «студент» і «викладач», особистий кабінет, модуль управління курсами зі структурою «курс → секції → тести» та різними типами питань з автоматичним підрахунком результатів.

Розроблено серверну частину на Laravel з реалізацією RESTful API, клієнтський інтерфейс на React.js із використанням компонентів Material-UI за принципами Clean Architecture, а також компонент тестування із збереженням історії проходжень. В результаті отримано функціональний прототип вебзастосунку «SkillUp», готовий до масштабування та комерційного впровадження як освітньої чи корпоративної платформи.

Ключові слова: вебзастосунок, онлайн-освіта, Laravel, React.js, MySQL, RESTful API, Clean Architecture, Material-UI