Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення інтерактивної платформи Learnify для онлайн-навчання з використанням Next.js, Prisma, MySQL і Stripe

Виконав: студент IV курсу, групи СН-43
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Тарас М.І. (прізвище та ініціали)

Керівник (підпис) Дмитроца Л.П. (прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Станько А.А. (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_27_» ___січня___ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Тарас Мар'яна Іванівна
(прізвище, ім'я, по батькові)

1. Тема роботи Створення інтерактивної платформи Learnify для онлайн-навчання з використанням Next.js, Prisma, MySQL і Stripe

Керівник роботи Дмитроца Леся Павлівна, кандидат технічних наук, доцент кафедри комп'ютерних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*07*» *травня* 2025 року № *4/7-444*

2. Термін подання студентом завершеної роботи 27 червня 2025р.

3. Вихідні дані до роботи Наукові публікації, Інтернет-джерела та технічна документація щодо розробки веб-застосунків з використанням фреймворку Next.js, ORM-бібліотеки Prisma, системи управління базами даних MySQL, платіжної системи Stripe, а також щодо методик розробки інтерактивних освітніх платформ

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання на розробку інтерактивної платформи для онлайн-навчання. 2. Проектування та реалізація інтерактивної платформи Learnify для онлайн-навчання. 3. Розгортання, тестування та підтримка платформи Learnify. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульна сторінка. 2. Актуальність теми дослідження. 3. Мета та завдання кваліфікаційної роботи. 4. Об'єкт і предмет дослідження. 5. Аналіз предметної області. 6. Архітектура платформи Learnify. 7. Модель бази даних та взаємозв'язки. 8. Функціональність для викладача. 9. Функціональність для студента. 10. Особливості реалізації інтерфейсу. 11. Інтеграція Stripe для оплати курсів. 12. Тестування та валідація платформи. 13. Результати розгортання та демонстрація. 14. Можливості розвитку та впровадження. 15. Висновки та практична цінність роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., кандидат технічних наук, доцент кафедри МТ		

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2025	
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	08.05.2025-15.05.2025	
3.	Виконання дослідження щодо актуального стану	16.05.2025-22.05.2025	
	цифрової освіти, аналізу конкурентів та предметної		
	області, формулювання завдання та визначення		
	функціональних і нефункціональних вимог до		
	системи		
	Розроблення архітектури вебплатформи, структури		
	бази даних, користувацького інтерфейсу (UI) та		
	серверної логіки (backend-частини)		
4.	Оформлення розділу «Аналіз предметної області та	23.05.2025-01.06.2025	
	постановка завдання на розробку інтерактивної		
	платформи для онлайн-навчання»		
5.	Оформлення розділу «Проектування та реалізація	23.05.2025-01.06.2025	
	інтерактивної платформи Learnify для онлайн-		
	навчання»		
6.	Оформлення розділу «Розгортання, тестування та	23.05.2025-01.06.2025	
	підтримка платформи Learnify»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	06.06.2025-07.06.2025	
8.	Виконання завдання до підрозділу «Основи охорони	08.06.2025-09.06.2025	
	праці»		
9.	Оформлення кваліфікаційної роботи	02.06.2025-10.06.2025	
10.	Нормоконтроль	11.06.2025-13.06.2025	
11.	Перевірка на плагіат	16.06.2025	
12.	Попередній захист кваліфікаційної роботи	18.06.2025	
13.	Захист кваліфікаційної роботи	28.06.2025	

Студент _____ Тарас М.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення інтерактивної платформи Learnify для онлайн-навчання з використанням Next.js, Prisma, MySQL і Stripe // Кваліфікаційна робота освітнього рівня «Бакалавр» // Тарас Мар'яна Іванівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-43 // Тернопіль, 2025 // С. 83, рис. – 32, додат. – 13, бібліогр. – 53 .

Ключові слова: онлайн-навчання, вебплатформа, next.js, база даних, prisma, stripe, вебінтерфейс, цифрова освіта.

Кваліфікаційна робота присвячена дослідженню процесу створення сучасної інтерактивної платформи Learnify для онлайн-навчання із застосуванням сучасних вебтехнологій. У роботі акцентується увага на проєктуванні, реалізації та тестуванні функціональних можливостей системи, що відповідає актуальним тенденціям цифровізації освіти.

У першому розділі кваліфікаційної роботи описано передумови створення освітніх платформ, проаналізовано сучасні підходи до онлайн-навчання, висвітлено основні технології, що застосовуються у проєктуванні вебзастосунків, а також розглянуто архітектурні особливості інтерактивних систем. Проаналізовано тенденції використання технологій Next.js, Prisma, MySQL та Stripe у створенні навчального програмного забезпечення.

У другому розділі кваліфікаційної роботи досліджено інструменти реалізації обраної технологічної бази, обґрунтовано вибір архітектури програмного забезпечення, подано опис структури бази даних та функціональних модулів платформи Learnify.

У третьому розділі кваліфікаційної роботи описано процес розгортання вебплатформи, проведено її тестування (кросбраузерне, швидкодії, адаптивності), проаналізовано методи збору зворотного зв'язку та технічної

підтримки, а також запропоновано інструкції з популяризації проєкту серед цільової аудиторії.

У четвертому розділі кваліфікаційної роботи досліджено питання безпеки життєдіяльності та охорони праці під час експлуатації обладнання. Зокрема, проаналізовано ергономічні проблеми, що виникають у виробничому середовищі, та визначено основні заходи техніки безпеки, спрямовані на мінімізацію професійних ризиків і забезпечення безпечних умов праці.

У висновку представлені результати усієї проведеної роботи.

ANNOTATION

Creation of the Learnify Interactive Platform for Online Learning Using Next.js, Prisma, MySQL, and Stripe // Qualification work of the educational level "Bachelor" // Taras Mariana Ivanivna // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-43 // Ternopil, 2025 // P. 83, fig. - 32, annexes. – 5, references - 53.

Keywords: online learning, web platform, next.js, database, prisma, stripe, web interface, digital education.

The qualification work is devoted to the study of the process of creating a modern interactive Learnify platform for online learning using modern web technologies. The work focuses on the design, implementation, and testing of the system's functionality, which corresponds to current trends in the digitalization of education.

The first chapter of the qualification work describes the prerequisites for creating educational platforms, analyzes modern approaches to online learning, highlights the main technologies used in the design of web applications, and considers the architectural features of interactive systems. The trends in the use of Next.js, Prisma, MySQL, and Stripe technologies in the creation of educational software are analyzed.

The second chapter of the qualification work examines the tools for implementing the selected technology base, justifies the choice of software architecture, and describes the structure of the database and functional modules of the Learnify platform.

The third section of the qualification work describes the process of deploying the web platform, conducts its testing (cross-browser, performance, adaptability),

analyzes methods of collecting feedback and technical support, and offers instructions for popularizing the project among the target audience.

The fourth section of the qualification work examines the issues of life safety and labor protection during equipment operation. In particular, it analyzes ergonomic problems that arise in the production environment and identifies the main safety measures aimed at minimizing occupational risks and ensuring safe working conditions.

The conclusion presents the results of all the work done.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – інтерфейс прикладного програмування, набір визначень і протоколів для створення та інтеграції програмного забезпечення.

CI/CD (Continuous Integration / Continuous Deployment) – безперервна інтеграція та розгортання, методологія автоматизації розробки та доставки ПЗ.

CSS (Cascading Style Sheets) – каскадні таблиці стилів, мова для опису зовнішнього вигляду HTML-документів.

DOM (Document Object Model) – об’єктна модель документа, інтерфейс для доступу та маніпуляції HTML- або XML-документом.

ER-діаграма – діаграма зв’язків сутностей (Entity-Relationship), графічне представлення структури бази даних.

HTML (HyperText Markup Language) – мова гіпертекстової розмітки для створення вебсторінок.

HTTP (HyperText Transfer Protocol) – протокол передавання гіпертексту, що використовується для передачі даних у вебмережі.

ORM (Object-Relational Mapping) – об’єктно-реляційне відображення, технологія для взаємодії програмних об’єктів з реляційною базою даних.

SEO (Search Engine Optimization) – пошукова оптимізація, процес покращення видимості сайту у пошукових системах.

SQL (Structured Query Language) – мова структурованих запитів для управління та запиту даних у СУБД.

SSG (Static Site Generation) – генерація статичних сайтів під час компіляції, один із режимів рендерингу в Next.js.

SSR (Server-Side Rendering) – рендеринг на стороні сервера, технологія попереднього завантаження сторінок на сервері.

SVG (Scalable Vector Graphics) – масштабована векторна графіка, формат для відображення векторних зображень у вебсередовищі.

UI/UX (User Interface / User Experience) – інтерфейс користувача та досвід користувача, що визначають зручність та ефективність взаємодії з цифровим продуктом.

БД – база даних, організована структура для збереження, керування та обробки даних.

СУБД – система управління базами даних, програмне забезпечення для створення, підтримки та використання БД.

ЗМІСТ

ВСТУП

Актуальність теми. Внаслідок глобалізації, цифровізації суспільства та активного впровадження інформаційних технологій у всі сфери життя, особливого значення набуває трансформація освітнього процесу. Онлайн-навчання стає не лише альтернативою традиційній формі здобуття знань, а й потужним інструментом для забезпечення гнучкості, доступності й персоналізації освіти. У сучасних умовах виникає потреба у створенні інноваційних цифрових платформ, які дозволяють ефективно реалізовувати навчальні курси, інтегрувати платіжні системи, забезпечувати зручну взаємодію між викладачами й студентами, а також гнучке адміністрування освітнього контенту. Тому розробка інтерактивних онлайн-платформ для навчання, зокрема із застосуванням таких технологій, як Next.js, Prisma, MySQL і Stripe, є актуальним напрямком сучасних досліджень в галузі веброзробки та цифрової освіти.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення доступності та зручності освітнього процесу шляхом розробки інтерактивної платформи Learnify для онлайн-навчання з використанням сучасних вебтехнологій. Платформа повинна забезпечити зручну взаємодію між користувачами, функціональну підтримку навчального контенту, безпечну авторизацію та ефективну систему монетизації курсів.

Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

- проаналізувати стан досліджень у сфері цифрової освіти та вебтехнологій, що застосовуються при створенні онлайн-платформ;
- обґрунтувати вибір архітектури платформи та інструментів реалізації, таких як Next.js, Prisma, MySQL, Tailwind CSS, Stripe і Clerk;

- спроєктувати та реалізувати функціональні модулі платформи Learnify, включаючи структуру бази даних, інтерфейс користувача й логіку курсів;
- забезпечити валідацію й тестування платформи, зокрема адаптивність, швидкодію, стабільність та кросбраузерну сумісність;
- розробити стратегії для підтримки та популяризації платформи серед цільової аудиторії.

Практичне значення одержаних результатів.

Створення інтерактивної платформи Learnify для онлайн-навчання з використанням Next.js, Prisma, MySQL і Stripe має суттєве практичне значення. Результатом виконаної роботи є повнофункціональна освітня вебплатформа, яка може бути використана для організації дистанційного навчання у закладах освіти, на курсах підвищення кваліфікації, у сфері неформальної освіти, а також для самостійного засвоєння знань. Реалізація системи із застосуванням сучасного технологічного стеку забезпечує її масштабованість, швидкодію та зручність як для розробників, так і для кінцевих користувачів. Learnify демонструє принципи побудови сучасного інтерактивного освітнього вебсервісу, який може бути адаптований, вдосконалений і кастомізований відповідно до потреб конкретної цільової аудиторії або інтегрований у вже існуючі освітні процеси для просування авторських курсів і цифрового контенту.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ ІНТЕРАКТИВНОЇ ПЛАТФОРМИ ДЛЯ ОНЛАЙН-НАВЧАННЯ

1.1 Аналіз онлайн-освітнього простору та методологічні засади дослідження

У сучасних умовах онлайн-освіта стала невід'ємною складовою освітнього процесу в Україні. За інформацією, оприлюдненою Державною службою якості освіти, протягом 2023 року близько 25% закладів освіти працювали дистанційно, а 50% – у змішаному форматі, поєднуючи очне та онлайн-навчання [1]. Це свідчить про зростаючу роль цифрових інструментів у сфері освіти та потребу в якісних вебрішеннях, що дозволяють ефективно організовувати навчальний процес.

У контексті загальнонаціональної цифрової трансформації, що охоплює малий і середній бізнес, особливого значення набуває створення цифрових рішень, здатних забезпечити доступність, ефективність і відповідність європейським стандартам. Як зазначено в дослідженні про цифровізацію українського МСБ, подібні ініціативи сприяють не лише підвищенню конкурентоспроможності, а й економічній інтеграції з країнами ЄС та G7 [2].

У цьому контексті інтерактивна платформа Learnify виступає прикладом сучасного освітнього сервісу, який відповідає потребам цифрової економіки. Вона забезпечує автоматизовану взаємодію між викладачами та студентами, підтримку гнучкого навчального процесу та прозоре управління освітнім контентом. Такі рішення не лише підвищують цифрову грамотність користувачів, а й сприяють реалізації цілей національної програми цифровізації, зокрема у сфері освіти та розвитку людського капіталу.

Паралельно спостерігається зростання глобального ринку онлайн-освіти: за даними Statista, очікується, що до 2026 року його обсяг сягне понад 300

мільярдів доларів США, порівняно з 250 млрд у 2020 році [3]. Це свідчить про стійкий попит на цифрові навчальні платформи з високим рівнем інтерфейсу, масштабованістю та можливістю монетизації контенту.

Дослідження зосереджено на вивченні функціональних можливостей, зручності використання та технічних характеристик сучасних освітніх платформ, зокрема Campster та Teachable. Аналіз їхніх сильних сторін і обмежень дозволяє сформувати концепцію вдосконаленої цифрової платформи, орієнтованої на сучасні вимоги користувачів, включаючи адаптивність, інтерактивність та доступність.

У контексті євроінтеграційних процесів цифровізація освіти в Україні набуває стратегічного значення. Відповідно до досліджень, формування ефективного освітнього середовища з використанням цифрових інструментів є важливою умовою підвищення якості навчання та розвитку цифрових компетентностей молоді [4]. Це створює сприятливі передумови для впровадження платформ на кшталт Campster, які дозволяють забезпечити гнучкий, доступний та інноваційний навчальний процес.

У межах роботи застосовано системний підхід до дослідження, що поєднує аналіз конкурентних рішень, формулювання вимог до функціоналу майбутньої системи та обґрунтування архітектурних рішень, що сприятимуть створенню ефективної інтерактивної платформи для онлайн-навчання та викладання – Learnify.

1.2 Аналітичний огляд існуючих рішень

У сучасному світі цифрова трансформація освіти зумовила стрімке зростання кількості онлайн-платформ для навчання. Така різноманітність пояснюється не лише широким попитом на дистанційне навчання, але й потребою в гнучкості, доступності та персоналізованому підході до здобуття знань. Платформи орієнтуються на різні цільові аудиторії – від школярів і студентів до дорослих, які прагнуть перекваліфікації або професійного

зростання. Кожен сервіс пропонує власне бачення освітнього процесу, впроваджуючи унікальні функції, підходи до взаємодії між викладачами та студентами, інструменти аналітики та монетизації контенту тощо.

Розглянемо дві онлайн-платформи – Campster та Teachable, з метою виявлення їхніх сильних і слабких сторін. Це дозволить визначити функціональні та структурні елементи, які варто врахувати під час створення власної платформи Learnify.

Campster є гейміфікованою інтерактивною платформою для колаборативного онлайн-навчання, заснованою у 2017 році. Наразі вона обслуговує понад 400 000 користувачів у семи країнах, включаючи Україну, Румунію та Сербію [5]. Платформа пропонує багатомовні курси, зокрема англійською, румунською, українською та сербською, що сприяє її доступності для різноманітної аудиторії.

Серед основних переваг Campster – використання механік гейміфікації для підвищення мотивації студентів, підтримка мультимовності та можливість колаборативного навчання. Однак платформа має певні недоліки, такі як обмежена аналітика прогресу, відсутність інтеграції з платіжними системами та неможливість стати викладачем і створити свої курси.

Teachable є міжнародною платформою для створення та продажу онлайн-курсів, яка налічує понад 30 000 активних шкіл і 150 000 авторів контенту. Станом на 2024 рік сукупний дохід творців на цій платформі перевищив 10 мільярдів доларів США [6].

До переваг Teachable належать можливості монетизації контенту, інтеграція з аналітичними інструментами (наприклад, Google Analytics) та широка аудиторія з усіх 195 всесвітньо визнаних країн. Разом із тим платформа має низку недоліків, зокрема відсутність детальної аналітики прогресу студентів, обмежені елементи гейміфікації та відсутність підтримки україномовного інтерфейсу.

Порівняльний аналіз платформ Campster та Teachable виявив їхні ключові слабкі сторони, такі як: недостатня аналітика навчального процесу, обмежені

можливості гейміфікації та відсутність інтеграції з платіжними сервісами. Ці аспекти будуть враховані при розробці нової платформи Learnify, яка має поєднати сильні сторони обох рішень, усунути їхні недоліки та забезпечити більш ефективний, інтерактивний і доступний формат онлайн-навчання.

1.3 Постановка задачі на розробку інтерактивної платформи Learnify

1.3.1 Область застосування та призначення розробки

Інтерактивна платформа Learnify призначена для викладачів, тренерів та експертів, які бажають створювати та продавати власні онлайн-курси, а також для студентів, які шукають якісні та доступні освітні ресурси. Основні сфери застосування платформи включають:

- Професійне навчання: підвищення кваліфікації, перекваліфікація та розвиток навичок у різних галузях.
- Академічна освіта: допомога у вивченні шкільних та університетських предметів.
- Особистісний розвиток: курси з розвитку «м'яких» навичок, хобі та інтересів.

Метою розробки є створення інтерактивної платформи Learnify для онлайн-навчання з використанням технологій Next.js, Prisma, MySQL і Stripe. Платформа повинна надавати інтуїтивно зрозумілий інтерфейс для створення курсів, управління контентом, взаємодії зі студентами та обробки платежів.

1.3.2 Вимоги до функціоналу платформи Learnify

До ключових функціональних можливостей платформи Learnify можна віднести наступні:

- Реєстрація та вхід у систему: можливість створення облікових записів для викладачів та студентів з відповідними правами доступу.

- Створення та адміністрування курсів: інструменти для додавання відео, текстових матеріалів та інших навчальних ресурсів. Автор курсу має мати можливість структурувати навчальний матеріал, розділивши курс на тематичні розділи або модулі, що полегшує сприйняття інформації та сприяє послідовному засвоєнню знань.
- Аналітика для авторів курсів: реалізація інструментів відстеження ключових показників – кількості переглядів, продажів, доходів тощо. Це дасть змогу авторам курсів аналізувати ефективність навчального контенту, оптимізувати стратегії просування та приймати зважені рішення щодо оновлення або вдосконалення своїх курсів.
- Зручна навігація та пошук: реалізація інтуїтивно зрозумілого інтерфейсу з функціональним пошуком за ключовими словами, категоріями тощо. Перед придбанням курсу користувач повинен мати змогу ознайомитися з його описом, програмою, тривалістю, рівнем складності та демо-уроком (за наявності), що сприяє усвідомленому вибору навчального контенту.
- Відстеження прогресу навчання: забезпечення можливості користувачеві бачити свій поточний прогрес у межах курсу, включаючи відсоток завершення, пройдені розділи та матеріали. Реалізація функції повернення до вже вивчених модулів для повторення вивченого матеріалу. Окрема сторінка профілю користувача, що зберігатиме інформацію про всі придбані курси, дозволяючи швидко повертатися до них у будь-який момент.
- Інтеграція платіжної системи: реалізація зручного та безпечного механізму оплати за допомогою сервісу для обробки онлайн-платежів. Це дозволить користувачам легко купувати курси, а авторам курсів – отримувати прибуток за свої цифрові навчальні продукти.
- Адаптивність інтерфейсу: забезпечення коректного та комфортного відображення платформи на різних типах пристроїв – від комп’ютерів до смартфонів і планшетів. Такий підхід гарантує, що користувач зможе навчатися зручно у будь-якому середовищі, незалежно від розміру екрана.

1.3.3 Стадії та етапи розробки платформи Learnify

Розробка інтерактивної платформи Learnify реалізовуватиметься у декілька етапів, враховуючи принципи інженерії програмного забезпечення та цілі, визначені при постановці задачі. Весь процес організовано так, щоб забезпечити логічну послідовність – від аналізу специфіки предметної галузі до впровадження завершеного рішення.

Спочатку буде проведено аналітику та проєктування, що включатимуть дослідження онлайн-освітнього середовища, аналіз наявних платформ-конкурентів, формування вимог до функціоналу та вибір відповідних технологій для реалізації. Буде визначено цільову аудиторію, сценарії використання платформи, а також основні функціональні модулі.

Далі відбуватиметься проєктування архітектури системи, що полягає у створенні загальної архітектурної моделі платформи, зокрема визначенні взаємодії між клієнтською та серверною частинами, виборі фреймворку для створення вебзастосунку, а також інструментів для взаємодії з базою даних і платіжною системою.

Під час етапу моделювання бази даних здійснюється побудова логічної схеми даних із використанням вибраної СУБД, формування ключових структур бази даних та опис взаємозв’язків між ними.

Після цього буде етап реалізації функціональності платформи, що включатиме розробку компонентів інтерфейсу та бекенду: системи реєстрації та автентифікації користувачів, створення та редагування курсів, перегляд і проходження навчальних матеріалів, обробка платежів через платіжну систему тощо.

На наступному етапі здійснюватиметься розробка користувацького інтерфейсу. Для стилізації платформи буде використано Tailwind CSS, що дозволить реалізувати сучасний, адаптивний дизайн, оптимізований для різних пристроїв. Особлива увага приділятиметься забезпеченню інтуїтивно

зрозумілої навігації, що сприятиме зручній взаємодії користувачів із платформою.

Після завершення основної розробки заплановано розгортання вебплатформи на сервері з подальшим тестуванням. Це включатиме перевірку роботи інтерфейсу, продуктивності системи, кросбраузерної сумісності, а також валідацію коректності реалізованого функціоналу.

Завершальним етапом стане підготовка до популяризації платформи Learnify серед цільової аудиторії. Буде розроблено інструкції для користувачів, а також визначено засади підтримки платформи після її запуску, зокрема моніторинг стабільності, оновлення та технічна підтримка.

Такий структурований підхід забезпечить ефективну розробку функціонального, зручного та масштабованого освітнього ресурсу, орієнтованого на потреби сучасних викладачів та учнів.

1.4 Вибір та обґрунтування використаних технологій

1.4.1 Фреймворк Next.js для створення вебзастосунків

Для реалізації інтерактивної платформи Learnify було обрано відкритий фреймворк Next.js, який є розширенням популярної бібліотеки React і забезпечує потужні можливості для створення сучасних, швидких і масштабованих вебзастосунків [5]. Next.js дозволяє ефективно поєднувати клієнтський і серверний рендеринг, що є особливо важливим для освітніх платформ, які повинні швидко завантажуватись, бути зручними у використанні та доступними для широкої аудиторії.

Серед основних переваг Next.js варто відзначити:

- Серверний рендеринг (SSR) та статичну генерацію сторінок (SSG), які покращують продуктивність, забезпечують швидкий рендеринг вмісту та позитивно впливають на SEO [7].

- Маршрутизацію на основі файлової структури, що спрощує створення та організацію сторінок вебзастосунку.
- Гнучкість у виборі підходу до відображення даних, зокрема можливість використовувати серверні функції API Route для обробки запитів без потреби в окремому бекенді [8].
- Підтримку TypeScript, що підвищує надійність коду та зменшує ймовірність помилок під час розробки.
- Активну спільноту та розвинену екосистему, що забезпечують доступ до широкого спектру інструментів, плагінів і документації.

Такий підхід було досліджено при аналізі сучасної веброзробки з використанням Next.js, React та Django REST. Порівняльна оцінка переваг Next.js над іншими популярними технологіями представлена у тезах про відповідне дослідження [9], що були опубліковані у матеріалах XIII Міжнародної студентської науково-технічної конференції «Природничі та гуманітарні науки. Актуальні питання» та подані в додатку А кваліфікаційної роботи.

Обрання Next.js також зумовлено тим, що він легко інтегрується з іншими сучасними інструментами [10], зокрема Prisma (ORM для взаємодії з базою даних) і Stripe (система обробки платежів), що є критично важливими для реалізації повного функціоналу платформи Learnify. Завдяки своїй архітектурі, Next.js забезпечує масштабованість, що дозволить у майбутньому розширювати платформу новими модулями без порушення стабільності та продуктивності системи.

1.4.2 Інструмент роботи з базами даних Prisma

Для забезпечення ефективної та зручної взаємодії з базою даних у межах платформи Learnify було обрано інструмент Prisma – сучасний інструмент ORM (Object-Relational Mapping), який надає можливість працювати з даними на основі зручних абстракцій у вигляді моделей.

Prisma надає низку переваг, які були визначальними при його виборі:

- Зрозумілий та декларативний синтаксис Prisma Schema, що дозволяє легко описувати структуру бази даних, включаючи моделі та їхні зв’язки.
- Генерація типів для TypeScript, що забезпечує повну типізацію при роботі з базою даних і зменшує кількість помилок на етапі компіляції.
- Автоматичне створення та оновлення міграцій, що спрощує управління версіями структури бази даних у процесі розробки.
- Інтеграція з багатьма СУБД, зокрема з MySQL, що використовується у даному проєкті.
- Продуктивний Prisma Client, який генерується автоматично і дозволяє виконувати запити до бази даних у зручній та безпечній формі.

Завдяки Prisma вдалося забезпечити тісну інтеграцію між backend-частиною, реалізованою на основі Next.js, та базою даних MySQL, що дозволяє ефективно реалізовувати запити на створення, оновлення, видалення та отримання інформації про користувачів, курси, покупки тощо.

Застосування Prisma рекомендовано багатьма розробниками сучасних вебзастосунків через його гнучкість, високу продуктивність, активну підтримку спільноти, що також підтверджується офіційною документацією [11].

1.4.3 Система управління базами даних MySQL

Для обробки, зберігання та адміністрування даних у рамках платформи Learnify було обрано реляційну СУБД MySQL. Це одна з найпопулярніших реляційних СУБД, яка має відкритий вихідний код і широко застосовується у сфері веброзробки завдяки своїй стабільності, ефективності та активному обміну знаннями серед користувачів і розробників [12].

MySQL забезпечує ефективну роботу з великими обсягами структурованих даних, має простий синтаксис SQL-запитів, підтримує транзакції, індексацію, механізми збережених процедур та зовнішніх ключів.

Однією з вагомих переваг є сумісність із різноманітними мовами програмування та інтеграційними засобами, зокрема Prisma ORM, що використовується у даній розробці для абстрагування від ручної роботи з SQL-запитами [13].

Вибір MySQL зумовлений також її зручністю у розгортанні та масштабуванні проєктів, що критично важливо для платформи з потенційно великою кількістю користувачів і курсів. Гнучкість, масштабованість і перевірена стабільність MySQL робить її доцільним вибором для реалізації backend-частини освітньої платформи.

1.4.4 Платіжна система Stripe

Для реалізації механізмів оплати навчальних курсів у межах інтерактивної платформи Learnify обрано платіжну систему Stripe. Stripe є однією з провідних глобальних платформ для обробки онлайн-платежів, яка вирізняється високим рівнем безпеки, простотою інтеграції та широким функціоналом для монетизації цифрового контенту.

Stripe забезпечує підтримку понад 100 варіантів платежів, серед яких – банківські картки, електронні гаманці, перекази через банки, а також регіональні методи розрахунку, що дозволяє ефективно обслуговувати користувачів з різних країн і розширити географію платформи [14]. Інтеграція Stripe дозволяє автоматизувати процес купівлі курсів студентами, а також реалізувати систему обліку доходів авторів курсів, забезпечуючи прозорість і контроль за фінансовими потоками.

Stripe також має зручну API-документацію, підтримку вебхуків, функціонал для створення кастомізованих чек-аутів, а також панель керування для адміністраторів і користувачів [15]. Ці особливості дозволяють ефективно інтегрувати Stripe у сучасні фреймворки, зокрема Next.js, що використовується у цій розробці.

Stripe повністю відповідає функціональним потребам платформи Learnify щодо обробки транзакцій, забезпечення безпеки платіжних даних і гнучкого управління оплатами.

1.5 Висновок до першого розділу

Перший розділ присвячено аналізу теоретичних і методологічних основ розробки цифрових освітніх вебзастосунків, що слугують фундаментом для створення ефективної онлайн-платформи Learnify для навчання. Основною метою розділу стало вивчення сучасного стану цифрової освіти, визначення функціональних вимог до навчальних платформ та аналіз технологій, що забезпечують їх реалізацію. На основі вивчених матеріалів було сформовано уявлення про актуальні підходи до побудови платформ змішаного навчання, включаючи необхідність інтерактивності, адаптивності та персоналізації.

Розгляд існуючих рішень дозволив виокремити як типові сильні сторони, так і обмеження існуючих освітніх платформ, що у подальшому стане основою для проектування власної структури платформи Learnify. Зокрема, зроблено акцент на важливості UX/UI-дизайну, зручного адміністрування контенту, а також підтримки авторських курсів із гнучкими налаштуваннями.

Теоретичний аналіз, проведений у цьому розділі, не лише заклав основу для технічної реалізації платформи у наступних розділах, але й підтвердив актуальність обраної тематики. Практична цінність викладеного матеріалу полягає у чіткому окресленні функціональних орієнтирів для проєктування інтерактивної платформи Learnify як конкурентоспроможного цифрового продукту, здатного задовольнити освітні потреби сучасного користувача.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОЇ ПЛАТФОРМИ LEARNIFY ДЛЯ ОНЛАЙН-НАВЧАННЯ

2.1 Архітектурно-технологічні засади реалізації платформи Learnify

2.1.1 Загальний опис архітектури системи

Цифрова платформа Learnify побудована відповідно до принципів трирівневої архітектури (three-tier architecture), яка передбачає чітке розділення системи на три незалежні рівні: презентаційний (Presentation Layer), логічний (Application Layer) та рівень даних (Data Layer) [16]. Такий підхід забезпечує масштабованість, розширюваність, зручну підтримку та ізоляцію змін у межах окремих шарів системи.

Презентаційний рівень (Presentation Layer) відповідає за взаємодію з користувачем [17]. Він реалізований за допомогою фреймворку Next.js 14, який поєднує можливості серверного рендерингу (SSR), клієнтського рендерингу (CSR) та статичного генерування сторінок (SSG). Завдяки App Router у Next.js реалізовано маршрутизацію на основі файлової структури, що дозволяє розділити доступ до контенту між типами користувачів (студенти, викладачі, адміністратори).

Інтерфейс користувача створено з використанням Tailwind CSS, що дозволяє швидко реалізовувати адаптивний та сучасний дизайн. Компоненти розташовані у папці components/ та згруповані за функціональними зонами: ui/, layout/, courses/, custom/.

Рівень бізнес-логіки (Application Layer) обробляє запити, керує логікою застосунку та забезпечує взаємодію між інтерфейсом і даними [18]. Його реалізовано у вигляді серверних компонентів Next.js та обробників API-запитів у папці /app/api. Саме тут розміщується функціональність, пов’язана з обробкою курсів, секцій, платежів, webhook-запитів тощо.

Також на цьому рівні реалізовано автентифікацію користувачів через Clerk, інтеграцію з платіжною системою Stripe, керування медіаконтентом за допомогою UploadThing, та загальну модульну логіку: керування навчанням, аналітика викладачів, персоналізовані панелі тощо.

На рівні даних (Data Layer) відбувається зберігання та обробка даних, що реалізовані через ORM Prisma, яка дозволяє типобезпечно працювати з реляційною базою даних MySQL. Структура бази даних визначається через Prisma Schema, а доступ до даних обмежується логікою серверної частини – безпосередньо з клієнтської частини запити не здійснюються. Такий підхід гарантує безпеку та контрольованість даних.

У додатку Б розміщено схему, що ілюструє трирівневу архітектуру платформи Learnify.

Ключові функціональні модулі системи охоплюють:

- реєстрацію та автентифікацію користувачів;
- створення, редагування та публікацію навчальних курсів;
- проходження курсів студентами;
- інтеграцію платіжної системи Stripe;
- обробку медіафайлів через UploadThing.

Загалом, архітектура платформи Learnify є гнучкою, чітко структурованою та модульною, що дозволяє ефективно масштабувати функціональність платформи, впроваджувати нові можливості й підтримувати стабільну роботу як для викладачів, так і для студентів.

2.1.2 Файлова структура проекту

Файлова структура інтерактивної платформи Learnify організована відповідно до принципів модульності, логічного розділення обов’язків та зручності масштабування. Вона базується на підході, закладеному у фреймворку Next.js 14, де використовується App Router та маршрутизація, що ґрунтується на структурі файлової системи.

Основні каталоги та їх призначення:

- /app/ – містить маршрути, які відображають різні області функціоналу системи. В межах цієї папки виокремлено логічні зони:
- (auth)/ – обробка автентифікації користувачів (вхід та реєстрація).
- (home)/ – основна публічна частина застосунку: пошук, категорії, навчання.
- (course)/ – сторінки курсів для студентів: перегляд, проходження, доступ до секцій.
- (instructor)/ – особистий кабінет викладача: створення, редагування курсів, аналітика.
- /api/ – реалізація REST API для обробки дій, пов'язаних з курсами, секціями, оплатами (включаючи Stripe), а також інтеграція з сервісами типу UploadThing або webhook-запитами.
- /components/ – бібліотека повторно використовуваних UI-компонентів, розділена за контекстами:
- ui/ – загальні елементи інтерфейсу користувача, такі як кнопки, поля для вводу та інші подібні компоненти.
- layout/ – компоненти розмітки (шапка, футер, навігація).
- courses/, sections/ – компоненти для роботи з курсами та секціями.
- providers/ – обгортки контекстів для стану застосунку (наприклад, авторизація).
- custom/, performance/ – розширені або специфічні блоки для окремих функцій.
- /lib/ – допоміжні утиліти, API-інтерфейси, функції роботи з даними або валідації.
- /prisma/ – файл schema.prisma, міграції бази даних та взаємодія через ORM Prisma.
- /public/ – статичні ресурси, зображення, іконки, favicon.
- /scripts/ – додаткові скрипти для налаштування або ініціалізації.

- .next/ – службова директорія, яка генерується під час збирання застосунку (не редагується вручну).

Конфігураційні файли:

- next.config.mjs, tailwind.config.ts, postcss.config.mjs – налаштування фреймворку, стилів і компіляції.
- middleware.ts – обробка глобальних проміжних обробників (наприклад, захист маршрутів).
- tsconfig.json, eslint.config.mjs – налаштування компілятора та лінтера.

У додатку В наведено діаграму структури файлової системи проєкту з використанням Next.js 14 App Router, що демонструє ієрархію директорій та ключових файлів проєкту. Така структура дає змогу легко орієнтуватись у кодовій базі, швидко додавати нові функції, а також чітко відокремлює функціональні області для кожного типу користувача.

2.1.3 Використання Next.js 14 з App Router для серверного рендерингу та маршрутизації

У вебплатформі Learnify реалізовано сучасний підхід до маршрутизації та серверного рендерингу, використовуючи функціонал App Router, доступний у версії Next.js 13+ та вдосконалений у версії 14. App Router базується на структурі директорій, що спрощує управління сторінками, логікою відображення і забезпечує підтримку серверних компонентів [19].

Замість традиційного /pages каталогу, застосунок використовує нову структуру /app, де кожна вкладена директорія відповідає певному маршруту. Це дозволяє розділяти маршрути за логікою: наприклад:

- /app/(auth)/sign-in – сторінка входу користувача.
- /app/(instructor)/instructor/create-course – інтерфейс створення курсу викладачем.

- /app/(course)/courses/[courseId]/sections/[sectionId] – перегляд окремої секції курсу.

Таким чином реалізовано динамічну маршрутизацію, коли параметри URL (наприклад, [courseId] або [sectionId]) обробляються безпосередньо на рівні структури файлів. Це спрощує логіку рендерингу контенту для різних курсів та секцій.

Для відображення вмісту окремого курсу використовується файл layout.tsx, розміщений у каталозі app/(course)/courses/[courseId]/. Він виконує такі ключові функції:

- забезпечує автентифікацію користувача через сервіс Clerk;
- реалізує отримання даних курсу з бази даних за допомогою Prisma ORM;
- виконує перевірку доступу до курсу;
- формує інтерфейс сторінки курсу, включаючи верхню панель (Topbar) і бічне меню (CourseSideBar).

У лістингу 2.1 наведено приклад реалізації компонента layout.tsx, який демонструє типову структуру сторінки курсу та використання серверних функцій Next.js.

Лістинг 2.1 – Шаблон компонента layout для сторінок курсу з авторизацією, отриманням даних і структурою інтерфейсу

```
import CourseSideBar from "@/components/layout/CourseSideBar";
import Topbar from "@/components/layout/Topbar";
import { db } from "@/lib/db";
import { auth } from "@clerk/nextjs/server";
import { redirect } from "next/navigation";

const CourseDetailsLayout = async ({
  children,
  params,
}: {
  children: React.ReactNode;
  params: { courseId: string };
}) => {
  const { userId } = auth();
```

```
  if (!userId) {
    return redirect("/sign-in");
  }

  const course = await db.course.findUnique({
    where: {
      id: params.courseId,
    },
    include: {
      sections: {
        where: {
          isPublished: true,
        },
        orderBy: {
          position: "asc",
        },
      },
    },
  });

  if (!course) {
    return redirect("/");
  }

  return (
    <div className="h-full flex flex-col">
      <Topbar />
      <div className="flex-1 flex">
        <CourseSideBar course={course} studentId={userId} />
        <div className="flex-1">{children}</div>
      </div>
    </div>
  );
};

export default CourseDetailsLayout;
```

Використання App Router дозволяє розділяти логіку для різних частин інтерфейсу (наприклад, автентифікація, курси, пошук, профіль інструктора) та підвищує масштабованість платформи. Серверний компонент layout.tsx у даному випадку виконує роль обгортки, яка надає базову структуру для всіх вкладених сторінок курсу.

Завдяки такому підходу маршрутизація платформи стала більш декларативною, масштабованою та зручною у підтримці.

2.1.4 Бібліотеки для розробки клієнтської та серверної частин проєкту

Під час розробки веб-платформи Learnify використовувались зовнішні бібліотеки – готові програмні модулі, що розширюють базову функціональність JavaScript/TypeScript і фреймворку Next.js. Завдяки бібліотекам можна суттєво пришвидшити розробку, підвищити стабільність додатку та забезпечити відповідність сучасним стандартам розробки [20].

Усі залежності зберігаються у файлі package.json, де зазначено як продуктивні залежності (dependencies), які потрібні під час роботи застосунку, так і залежності для розробки (devDependencies), що використовуються лише на етапі створення, компіляції чи стилізації проєкту.

На рисунку 2.1 відображено список залежностей платформи Learnify.

```
"dependencies": {
  "@clerk/nextjs": "^5.1.3",
  "@hello-pangea/dnd": "^16.6.0",
  "@hookform/resolvers": "^3.4.2",
  "@mux/mux-node": "^8.8.0",
  "@mux/mux-player-react": "^2.9.1",
  "@prisma/client": "^5.14.0",
  "@radix-ui/react-alert-dialog": "^1.1.5",
  "@radix-ui/react-dialog": "^1.0.5",
  "@radix-ui/react-label": "^2.0.2",
  "@radix-ui/react-popover": "^1.0.7",
  "@radix-ui/react-progress": "^1.0.3",
  "@radix-ui/react-slot": "^1.1.1",
  "@radix-ui/react-switch": "^1.1.2",
  "@tanstack/react-table": "^8.20.6",
  "@uploadthing/react": "^6.8.0",
  "axios": "^1.7.9",
  "class-variance-authority": "^0.7.0",
  "clsx": "^2.1.1",
  "cmdk": "^0.2.1",
  "lucide-react": "^0.379.0",
  "next": "14.2.3",
  "react": "^18",
  "react-dom": "^18",
  "react-hook-form": "^7.51.5",
  "react-hook-toast": "^0.0.3",
  "react-hot-toast": "^2.5.1",
  "react-quill": "^2.0.0",
  "recharts": "^2.12.7",
  "stripe": "^15.9.0",
  "tailwind-merge": "^2.3.0",
  "tailwindcss-animate": "^1.0.7",
  "uploadthing": "^6.13.3",
  "zod": "^3.23.8"
},
```


Рисунок 2.1 – Список бібліотек проекту Learnify

Наведені бібліотеки можна логічно поділити на три групи: клієнтські, серверні та бібліотеки, що забезпечують взаємодію з зовнішніми платформами.

Клієнтські бібліотеки відповідають за побудову та інтерфейс клієнтської частини платформи, взаємодію з користувачем, формами та валідацією. Зробимо детальний опис кожної із них:

- react, react-dom – основа для побудови інтерфейсу користувача.
- react-hook-form – ефективне керування станом форм із підтримкою валідації.
- @hookform/resolvers + zod – інтеграція схем валідації у форми.
- react-hot-toast – повідомлення (toast) для зворотного зв’язку з користувачем.
- react-quill – візуальний редактор контенту курсів.
- @tanstack/react-table – побудова динамічних таблиць.
- @hello-pangea/dnd – drag-and-drop функціональність (наприклад, зміна порядку розділів курсу).
- clsx, class-variance-authority, tailwind-merge – динамічна генерація CSS-класів із Tailwind.
- lucide-react – набір SVG-іконок.
- cmdk – реалізація командної панелі (Command Menu).
- tailwindcss-animate – анімації у стилі Tailwind CSS.

Серверні бібліотеки використовуються у серверних компонентах Next.js для обробки запитів, підключення до бази даних і реалізації backend-логіки. Їх менше, ніж клієнтських, але вони не є менш важливими. Розглянемо наведені бібліотеки та їх основне призначення:

- @prisma/client – доступ до бази даних через Prisma ORM.
- zod – валідація даних на сервері.
- axios – HTTP-клієнт для взаємодії з API.
- @clerk/nextjs – автентифікація та управління користувачами (інтеграція з Clerk).
- next – сам фреймворк, що реалізує SSR, маршрутизацію, API-маршрути тощо.

З великої кількості бібліотек, що забезпечують взаємодію з зовнішніми платформами використано такі:

- stripe – платіжна система для купівлі доступу до курсів.
- @mux/mux-node, @mux/mux-player-react – відеохостинг та відтворення відео-лекцій.
- uploadthing, @uploadthing/react – завантаження файлів користувачами.

Також варто згадати про залежності, що використовувались лише під час розробки платформи Learnify і не потрібні під час виконання застосунку на сервері чи у браузері. На рисунку 2.2 відображено їх перелік.

Рисунок 2.2 – Список залежностей та бібліотек, що використовувались під час розробки платформи Learnify

Розглянемо основні призначення та функції, що виконують наведені на рисунку 2.2 бібліотеки:

- @types/* – типи для бібліотек, що використовуються в TypeScript.
- eslint, eslint-config-next – перевірка коду на помилки та дотримання стилістики.
- tailwindcss, postcss – генерація стилів для інтерфейсу.
- prisma – генерація типів і міграцій для роботи з базою даних.
- typescript – типізація коду, що підвищує стабільність і підтримуваність.

Таким чином, використання бібліотек дозволило зменшити час розробки, забезпечити надійність платформи та реалізувати її ключові функції – автентифікацію, відео-лекції, інтерактивні форми та оплату – з використанням сучасних підходів і технологій.

2.1.5 Розподіл на клієнтську та серверну частини

Платформа Learnify реалізована у форматі повноцінного full-stack застосунку, використовуючи функціональні можливості фреймворку Next.js 14 з підтримкою App Router. Такий підхід дозволяє органічно поєднувати клієнтську та серверну частини в межах одного проєкту без необхідності створення окремого backend-сервера [21].

Основна серверна логіка реалізована за допомогою серверних компонентів Next.js, які обробляють запити, отримують дані з бази даних (через Prisma ORM) і виконують перевірку автентифікації та доступу користувача (з використанням Clerk). Наприклад, у файлі layout.tsx (розділ курсів) відбувається отримання інформації про курс, перевірка наявності авторизованого користувача та формування загальної структури сторінки.

З іншого боку, клієнтська частина платформи охоплює інтерактивні елементи інтерфейсу, роботу з формами, drag-and-drop функціональність, текстові редактори та обробку дій користувача. Для цього використовуються бібліотеки react-hook-form, react-quill, @hello-pangea/dnd, @uploadthing/react, react-hot-toast тощо. Компоненти, які потребують доступу до DOM або взаємодії з браузером, маркуються директивою "use client" [22].

Такий розподіл дозволяє досягти:

- ефективного використання серверних ресурсів для завантаження даних і автентифікації;
- швидкого та динамічного відображення клієнтського інтерфейсу;
- зручної підтримки та масштабування коду завдяки чіткому розмежуванню функцій між клієнтською та серверною частинами.

Таким чином, платформа Learnify повністю використовує архітектурні можливості Next.js для створення гібридного застосунку, у якому клієнтська та серверна частини взаємодіють у межах єдиної екосистеми.

2.2 Моделювання бази даних

2.2.1 Опис основних моделей

У платформі Learnify реалізовано набір моделей, які відображають ключові об’єкти предметної області. Кожна модель у Prisma визначає таблицю у базі даних і містить поля, що відповідають стовпцям, а також зв’язки з іншими моделями [23].

Основними моделями БД для платформи Learnify є: User, Course, Category, SubCategory, Level, Section, Purchase, MuxData, StripeCustomer, Resource, Progress.

Модель User містить інформацію про зареєстрованих користувачів платформи – студентів та інструкторів. Вона включає унікальний ідентифікатор (id), ім’я користувача (name), електронну адресу (email), посилання на зображення профілю (imageUrl), а також дату створення облікового запису (createdAt).

Модель Course відповідає за опис навчальних курсів. Вона містить такі поля: унікальний ідентифікатор (id), зовнішній ключ інструктора (instructorId), назву курсу (title), підзаголовок (subtitle), опис (description), зображення (imageUrl), вартість (price), ознаку опублікованості (isPublished), а також зовнішні ключі для категорії (categoryId), підкатегорії (subCategoryId) та рівня складності (levelId). Додатково зберігаються дати створення і оновлення (createdAt, updatedAt).

Модель Category дозволяє групувати курси за загальними тематиками. Вона містить ідентифікатор (id) та назву категорії (name).

Модель SubCategory деталізує поділ курсів у межах основної категорії. До неї входять: унікальний ідентифікатор (id), назва підкатегорії (name) і зовнішній ключ на модель Category (categoryId).

Модель Level визначає рівень складності курсу (наприклад, початковий, середній або просунутий). Вона складається з ідентифікатора (id) та назви рівня (name).

Модель Section представляє окремі секції курсу. Вона включає унікальний ідентифікатор (id), заголовок (title), опис (description), посилання на відео (videoUrl), позицію секції в межах курсу (position), позначки про публікацію (isPublished) і доступність безкоштовно (isFree), зовнішній ключ на курс (courseId), а також дати створення і оновлення (createdAt, updatedAt).

Модель Progress використовується для відстеження навчального прогресу користувачів. Вона містить унікальний ідентифікатор (id), зовнішні ключі на користувача (studentId) та секцію (sectionId), логічне поле для визначення завершеності (isCompleted), а також часові мітки створення та оновлення (createdAt, updatedAt).

Модель Purchase зберігає дані про придбання курсів. До її полів належать: унікальний ідентифікатор (id), зовнішні ключі на клієнта (customerId) та курс (courseId), а також дати створення та оновлення транзакції (createdAt, updatedAt).

Модель StripeCustomer використовується для зберігання даних користувачів, що здійснюють оплату через платіжну систему Stripe. Вона включає унікальний ідентифікатор (id), зовнішній ключ на користувача (customerId), ідентифікатор клієнта у Stripe (stripeCustomerId), а також дати створення та оновлення запису (createdAt, updatedAt).

Модель Resource містить додаткові матеріали до секцій, такі як документи або посилання. Вона включає ідентифікатор (id), назву ресурсу (name), посилання на файл (fileUrl), зовнішній ключ на секцію (sectionId) та дати створення й оновлення (createdAt, updatedAt).

Модель MuxData відповідає за технічну інформацію щодо відеоконтенту секцій. Вона включає ідентифікатор (id), дані для роботи з відеосервісом (assetId, playbackId) та зовнішній ключ на секцію (sectionId), який є унікальним для забезпечення зв’язку один-до-одного.

2.2.2 Зв’язки між моделями

У реляційній базі даних зв’язки (або відношення) – це логічні асоціації між таблицями (моделями), що дозволяють зв’язувати дані між собою [24]. Вони відіграють ключову роль у забезпеченні цілісності та узгодженості інформації.

Основними типами зв’язків у реляційних базах даних є:

- Один-до-одного (1:1) – кожному запису в одній таблиці відповідає не більше одного запису в іншій.
- Один-до-багатьох (1:N) – одному запису в першій таблиці може відповідати багато записів у другій.
- Багато-до-багатьох (M:N) – багато записів з однієї таблиці можуть бути пов’язані з багатьма записами іншої. Такий зв’язок зазвичай реалізується через проміжну (з'єднувальну) таблицю.

У моєму проєкті переважно використовуються зв’язки один-до-багатьох, що забезпечує ефективну організацію даних. На рисунку 2.3 представлено ER-діаграму БД платформи Learnify, яка ілюструє сутності та їхні взаємозв’язки.

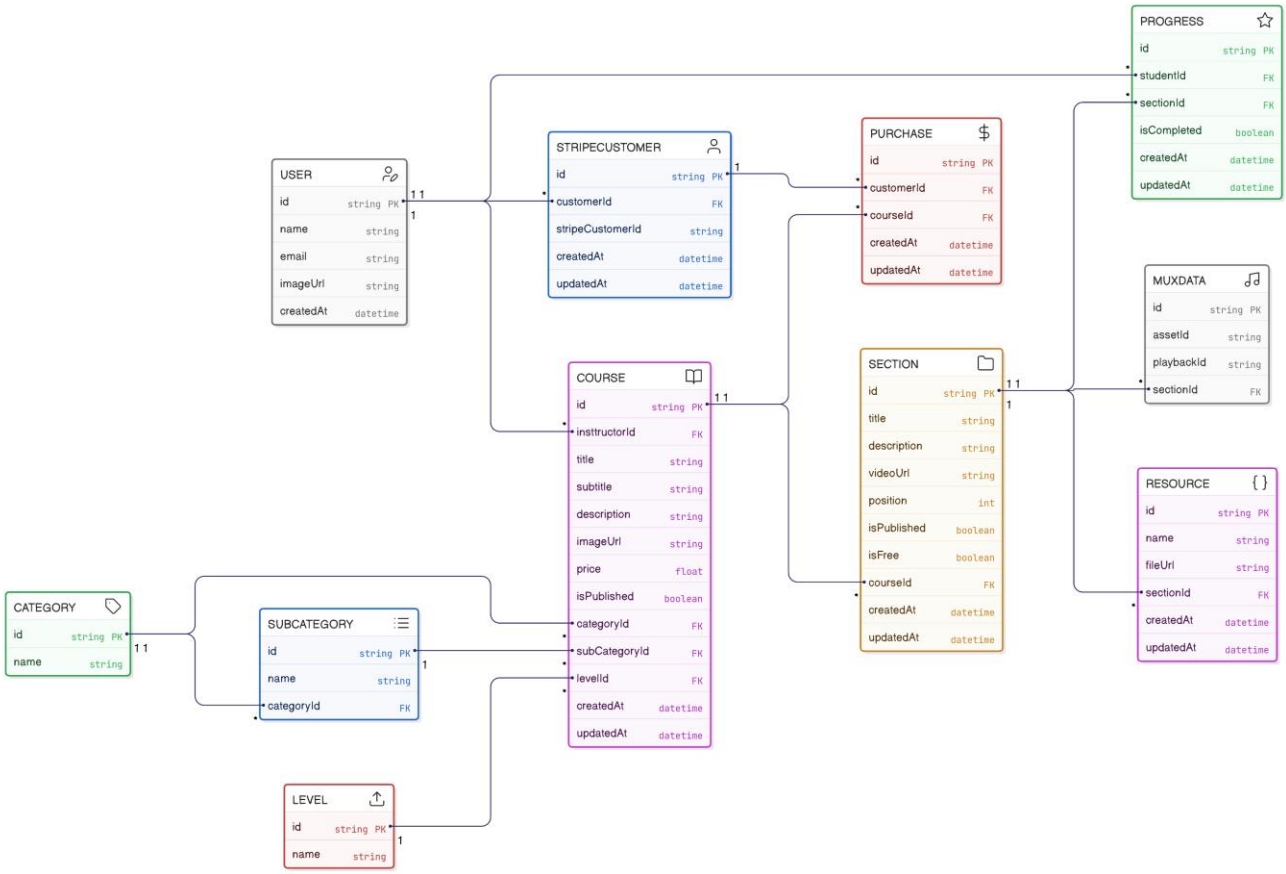


Рисунок 2.3 – ER-діаграма БД для платформи Learnify

Як видно з ER-діаграми, між моделями User та Progress встановлено зв’язок типу один-до-багатьох. Один користувач може проходити багато секцій, у той час як кожен запис про прогрес відповідає лише одному користувачу. Це реалізовано за допомогою зовнішнього ключа studentId, що посилається на id у таблиці User.

Аналогічний тип зв’язку існує між моделями User та Purchase: один користувач може мати кілька покупок, кожна з яких пов’язана з ним за допомогою поля customerId. При цьому зв’язок користувачів із покупками здійснюється опосередковано через модель StripeCustomer.

У свою чергу, StripeCustomer також пов’язана з Purchase відношенням один-до-багатьох, адже кожен Stripe-клієнт може мати декілька транзакцій. Зовнішній ключ customerId у таблиці Purchase вказує на запис у StripeCustomer.

Відношення між Course і Purchase логічно аналогічне – кожен курс може бути придбаний багатьма користувачами, але кожна покупка стосується лише одного конкретного курсу. Це забезпечується полем courseId.

Модель Course також пов’язана з Section через зв’язок один-до-багатьох, адже один курс може містити декілька секцій. У Section відповідальним за зв’язок є зовнішній ключ courseId.

Кожна секція (Section) може бути пройдена багатьма студентами, що реалізується через таблицю Progress. Тут також використовується зв’язок один-до-багатьох з ключем sectionId.

У моделі Resource реалізовано аналогічний принцип: одна секція може містити багато ресурсів, кожен з яких містить посилання на sectionId.

Особливий випадок становить зв’язок між Section та MuxData – це відношення один-до-одного. Кожна секція має лише один відеозапис, а зовнішній ключ sectionId у MuxData є унікальним (@unique), що гарантує цю відповідність.

У контексті класифікації курсів, модель Category пов’язана з SubCategory зв’язком один-до-багатьох, оскільки одна категорія може мати кілька підкатегорій. Відповідний зовнішній ключ categoryId зберігається в SubCategory.

Крім того, кожна категорія може охоплювати безліч курсів – це ще один приклад зв’язку один-до-багатьох, реалізований через поле categoryId у Course.

SubCategory, аналогічно, має зв’язок з Course, що дозволяє групувати курси за більш детальними темами.

Нарешті, модель Level, яка визначає складність курсу, також має відношення один-до-багатьох з Course: одному рівню може відповідати багато курсів, хоча поле levelId у Course є необов’язковим (nullable).

Таким чином, організація бази даних реалізована через логічно пов’язані таблиці з чітко визначеними зовнішніми ключами, що забезпечує цілісність і гнучкість при обробці даних.

2.2.3 Використання Prisma Schema для визначення структури БД

Для опису організації даних у цифровій платформі використано ORM-інструмент Prisma, що забезпечує декларативний спосіб моделювання [25].

Моделювання даних реалізовано у файлі schema.prisma, який задає типи, зв’язки, правила цілісності та індексації [26]. Це дозволяє не лише описати логіку бази даних, але й автоматично згенерувати доступ до неї у вигляді клієнтських методів Prisma Client для подальшої роботи на серверній частині застосунку.

Основною перевагою Prisma Schema є можливість опису моделей у зручному форматі, що забезпечує типізацію, перевірку коректності на етапі розробки та автоматичну генерацію SQL-міграцій.

Моделі, описані у файлі schema.prisma, відповідають сутностям, представленим в ER-діаграмі. Кожна модель визначається за допомогою ключового слова model і включає перелік полів з вказаними типами, правилами унікальності, первинними ключами та зовнішніми ключами (через ключове слово @relation).

Наприклад, у моделі Course можна спостерігати використання зовнішніх ключів (instructorId, categoryId, subCategoryId, levelId), а також визначення зв’язків один-до-багатьох та багато-до-одного. У лістингу 2.2 наведено фрагмент файлу schema.prisma, де описано модель Course.

Лістинг 2.2 – Фрагмент файлу schema.prisma із описом моделі Course

```
Course {
  id           String  @id @default(uuid())
  instructorId String
  title        String  @db.Text
  subtitle     String? @db.Text
  description  String? @db.Text
  imageUrl     String? @db.Text
  price        Float?
  isPublished  Boolean @default(false)

  categoryId String
```

```
  category    Category @relation(fields: [categoryId], references:
[id])

  subCategoryId String
  subCategory      SubCategory  @relation(fields:  [subCategoryId],
references: [id])

  levelId String?
  level   Level?  @relation(fields: [levelId], references: [id])

  sections Section[]
  purchases Purchase[]
  createdAt DateTime @default(now())
  updatedAt DateTime @default(now())

  @@index([categoryId])
  @@index([subCategoryId])
  @@index([levelId])
}
```

Такий опис дозволяє Prisma генерувати типобезпечний клієнт для взаємодії із базою, створювати міграції (prisma migrate), а також синхронізувати схему з фактичною базою даних.

Також варто зазначити, що Prisma дозволяє задавати унікальні обмеження, опціональність полів, а також обробляти складні зв’язки між таблицями (наприклад, один-до-одного, багато-до-багатьох). Загалом, Prisma забезпечує прозору, декларативну та ефективну реалізацію структури бази даних у сучасних веб-застосунках.

Повний вміст файлу schema.prisma наведено у додатку Д.

2.3 Реалізація функціональності платформи

2.3.1 Реєстрація та автентифікація користувачів

Для реалізації процесів реєстрації та автентифікації користувачів у платформі Learnify було використано готове стороннє рішення – сервіс Clerk. Такий підхід дозволяє значно спростити процес інтеграції функціоналу автентифікації, мінімізувати кількість потенційних помилок у реалізації захисту

персональних даних та сконцентруватися на розробці основної логіки платформи.

Clerk надає набір готових компонентів (наприклад, сторінки входу, реєстрації, керування профілем), які легко інтегруються в інтерфейс застосунку. Сервіс підтримує сучасні методи входу, зокрема через email/пароль, OAuth-провайдерів (Google, GitHub тощо). Крім того, він гарантує надійний захист користувацьких даних завдяки використанню таких механізмів, як двофакторна перевірка, підтвердження електронної пошти тощо [27].

На платформі Learnify під час створення облікового запису користувач вводить своє ім’я, адресу електронної пошти та пароль. Також передбачена можливість входу через Apple або Google. Інтерфейс сторінки реєстрації нового користувача показано на рисунку 2.4.

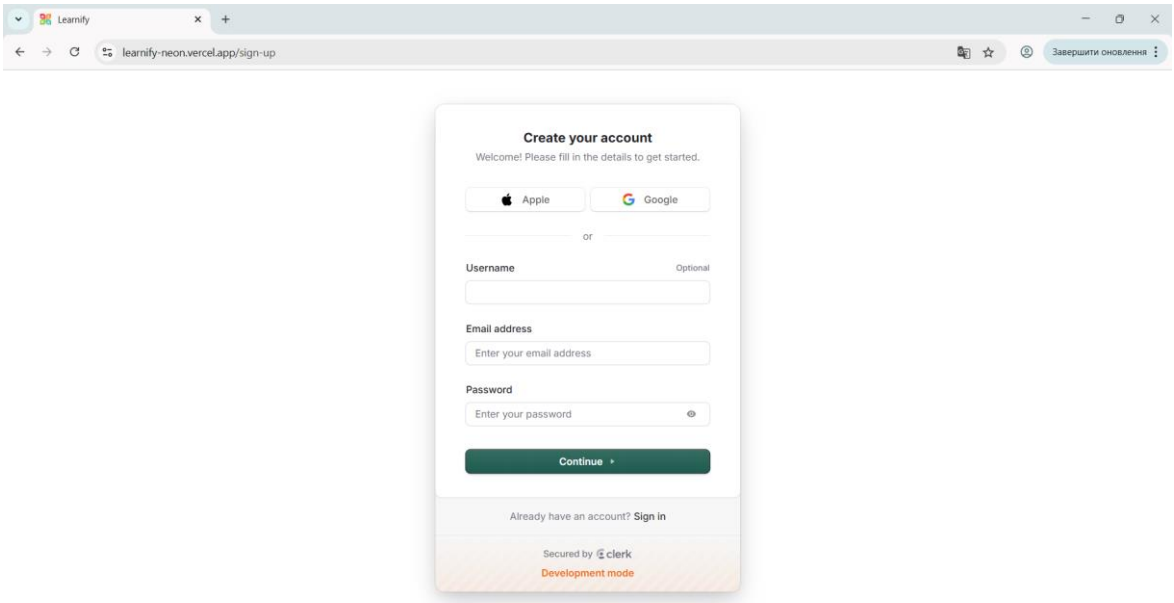


Рисунок 2.4 – Сторінка з формою реєстрації користувача

Користувачі, які вже створили обліковий запис на платформі, можуть увійти до системи через спеціальну сторінку. Для цього необхідно вказати email або логін. На рисунку 2.5 показано інтерфейс форми входу до облікового запису.

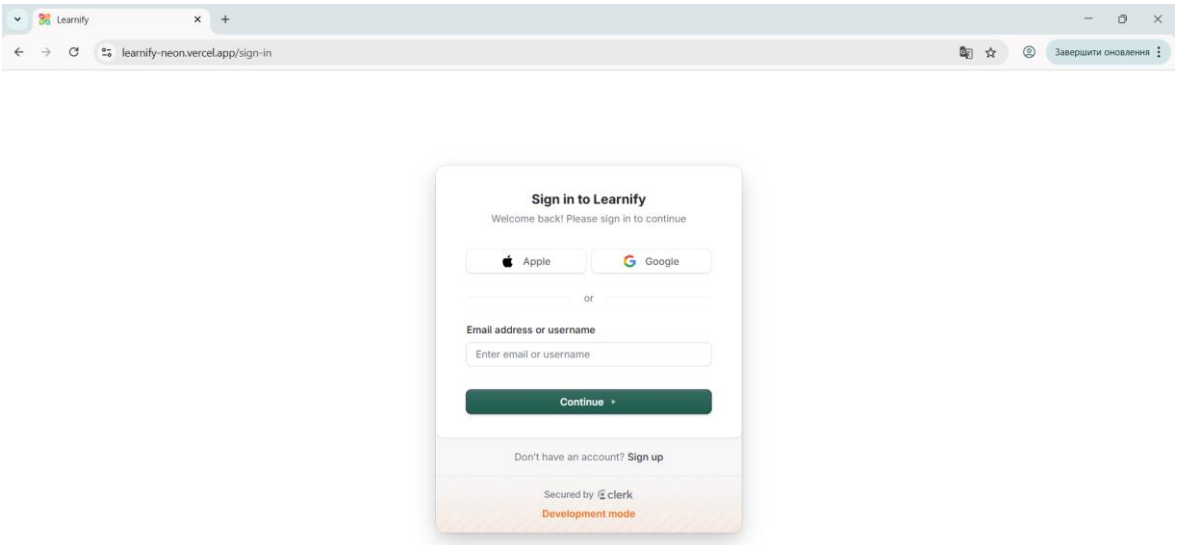


Рисунок 2.5 – Сторінка з формою входу у систему

Після введення електронної пошти чи імені користувача, система надсилає на вказану електронну адресу верифікаційний код. Користувач повинен ввести цей код у відповідне поле для підтвердження своєї особи (див. рисунок 2.6).

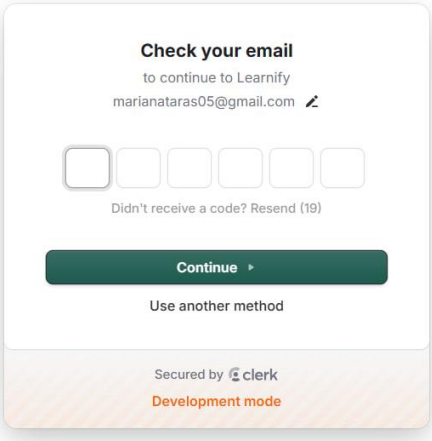


Рисунок 2.6 – Вікно введення верифікаційного коду

Такий механізм забезпечує кращу безпеку акаунта, запобігає створенню фейкових профілів та гарантує, що доступ до платформи має лише власник електронної пошти.

Питання безпечної авторизації є критично важливим для функціонування будь-якої цифрової освітньої платформи. Зокрема, на етапах автентифікації користувача важливо реалізувати валідацію на боці клієнта та сервера, контроль за терміном дії токенів і обмеження кількості спроб входу. Подібні рекомендації описані у [28], де розглянуто безпеку автентифікації у вебдодатках, зокрема в соціальних мережах, що мають аналогічну структуру облікових записів та методів підтвердження особи.

Після завершення процедури реєстрації або входу користувач автоматично потрапляє до головного інтерфейсу платформи Learnify. У верхній частині інтерфейсу, з правого боку панелі Topbar, розташовано компонент userButton – інтерактивний елемент, що відкриває доступ до профілю користувача, параметрів облікового запису та функції виходу із системи (див. рисунок 2.7) [29].

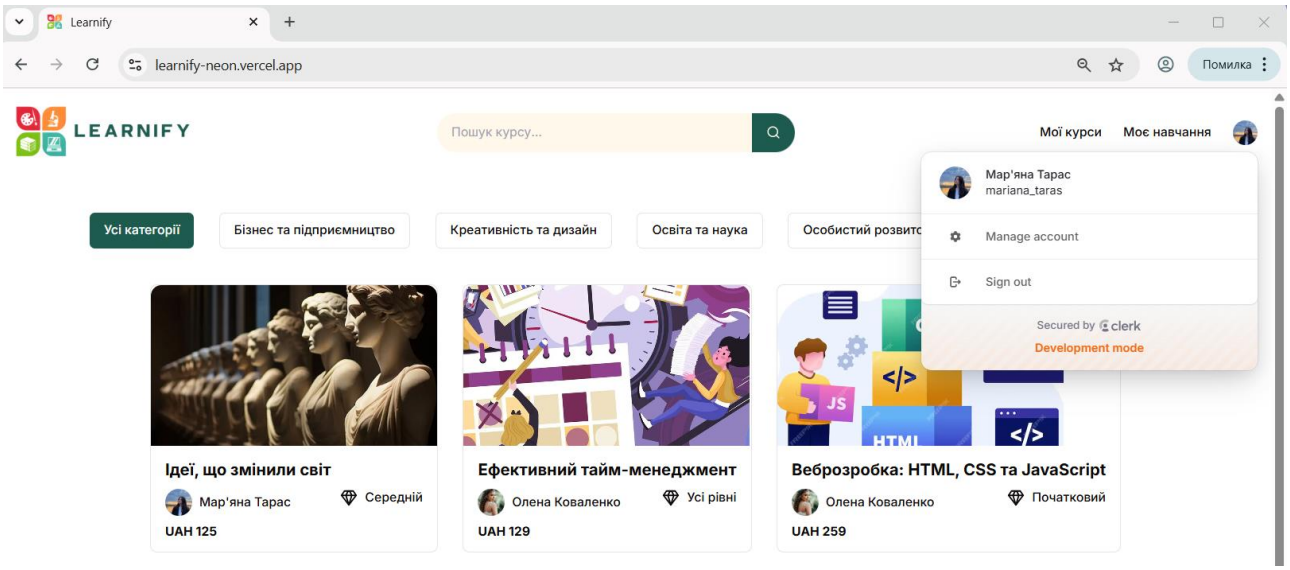


Рисунок 2.7 – Розташування та вигляд інтерактивного елементу доступу до профілю користувача

Крім того, в адміністративному акаунті Clerk доступна аналітика: можна переглядати список зареєстрованих користувачів, їхню основну інформацію, час останньої активності тощо (див. рисунок 2.8).

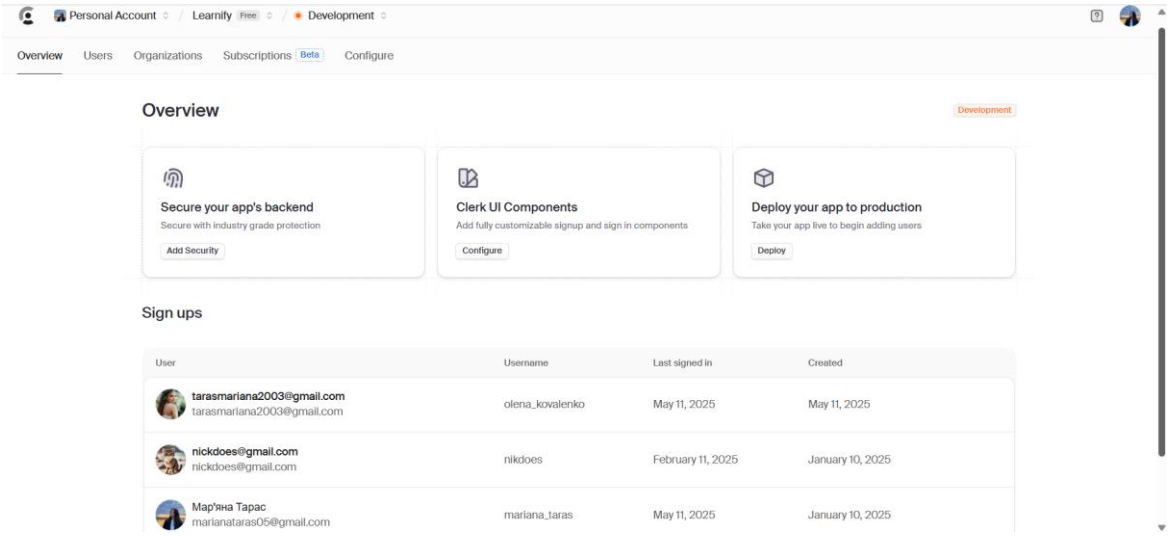


Рисунок 2.8 – Вигляд інформаційної панелі платформи Clerk

Це дає можливість контролювати користувацьку базу без додаткової розробки відповідного backend-інтерфейсу.

2.3.2 Створення та редагування навчальних курсів викладачами

Інтерактивна платформа Learnify надає інструменти для викладачів (інструкторів), які дозволяють зручно створювати нові навчальні курси, редагувати їхній вміст, додавати відеоуроки, допоміжні матеріали та визначати структуру курсу.

Після входу до системи під обліковим записом інструктора, користувач повинен обрати у верхньому горизонтальному меню вкладку «Мої курси». В результаті цього він отримує доступ до персональної панелі керування своїми курсами (див. рисунок 2.9).

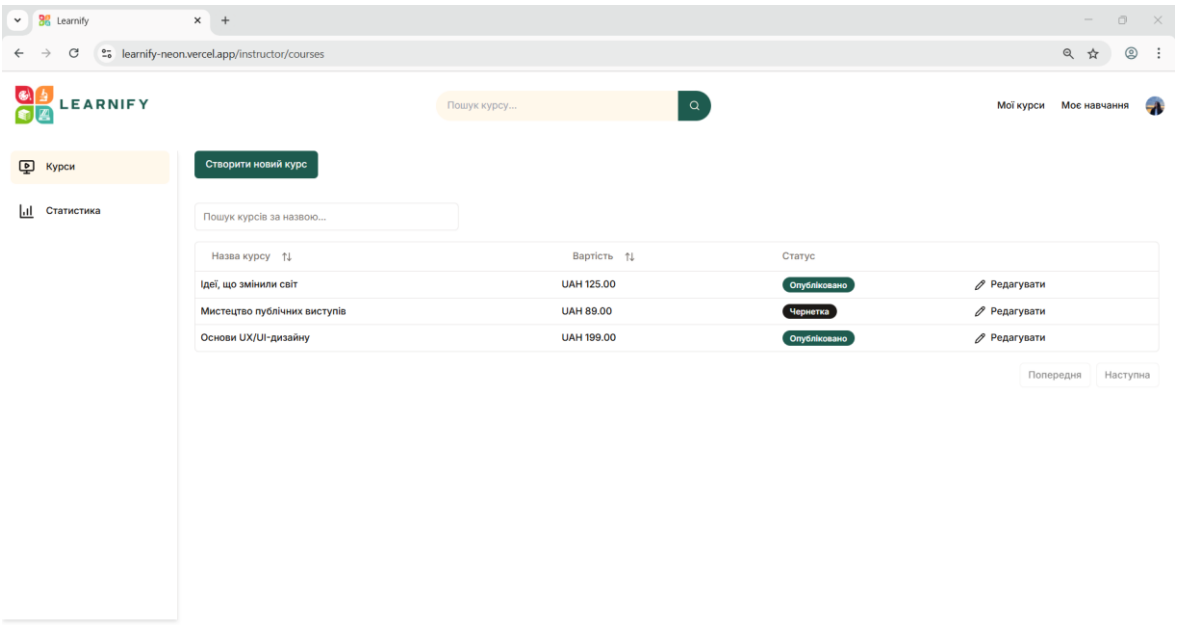


Рисунок 2.9 – Сторінка панелі керування доданими курсами

У лівій частині сторінки розташована бічна панель навігації (sidebar), яка містить основні вкладки для інструкторів – «Курси» та «Статистика». Ця панель забезпечує зручний доступ до керування навчальним контентом і перегляду аналітичних даних.

На вкладці «Курси» викладач бачить перелік усіх курсів, які він створив. Передбачена функція пошуку, що дозволяє швидко знайти потрібний курс за назвою. У таблиці з переліком курсів відображається назва, вартість, статус публікації (наприклад, опубліковано чи чернетка), а також кнопка для редагування кожного курсу.

Вкладка «Статистика» надає доступ до інформації про доходи інструктора. Тут відображається загальна сума продажів курсів, а також діаграма доходів, що дозволяє візуально оцінити динаміку прибутку за певний період. Такий формат подання аналітики значно покращує зручність і зрозумілість інтерфейсу для викладачів. Вигляд цієї сторінки показано на рисунку 2.10.

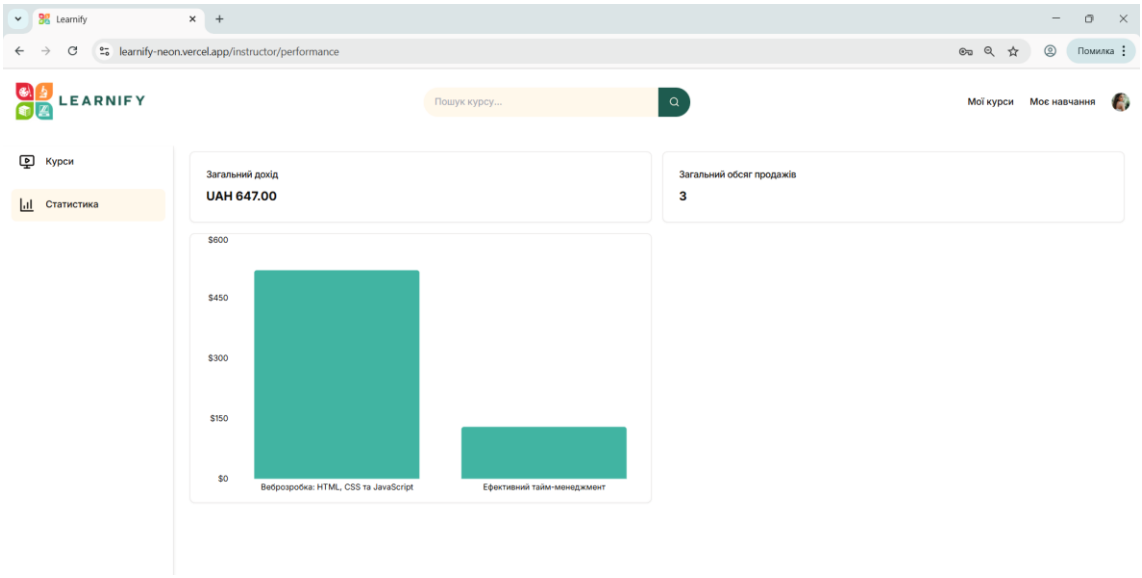


Рисунок 2.10 – Сторінка статистики викладача з аналітикою доходів

На вкладці «Курси» викладач може розпочати створення нового навчального курсу за допомогою відповідної кнопки (див. рисунок 2.9). Після її активації користувач автоматично переходить на окрему сторінку, інтерфейс якої подано на рисунку 2.11.

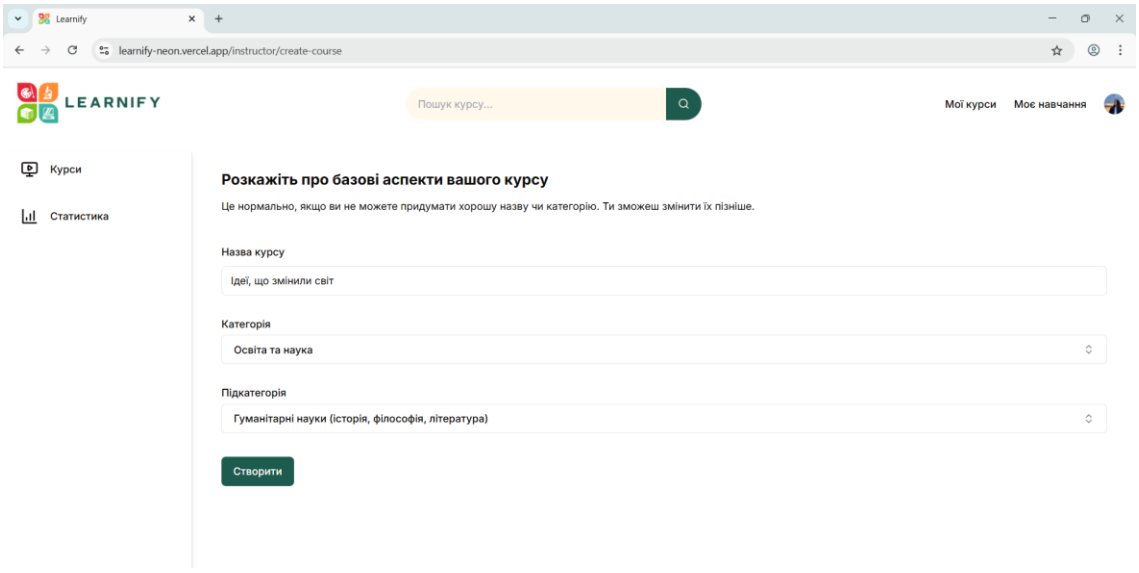


Рисунок 2.11 – Вигляд сторінки створення нового курсу

На цій сторінці слід заповнити основні характеристики курсу: його назву, категорію та підкатегорію. Щойно ці поля заповнені, стає активною кнопка «Створити», що розташована внизу сторінки.

Після створення курсу відкривається сторінка з розширеною формою редагування курсу (див. рисунок 2.12). Тут зберігаються раніше введені дані, але з'являється низка нових обов'язкових полів: підзаголовок курсу, опис, рівень складності, обкладинка курсу та вартість. У верхній частині сторінки міститься інформація про кількість незаповнених обов'язкових полів, що допомагає користувачу орієнтуватися у прогресі заповнення. Якщо є незаповнені поля, то цей елемент інтерфейсу має червоний колір, а при заповнені усіх полів змінює забарвлення на зелений.

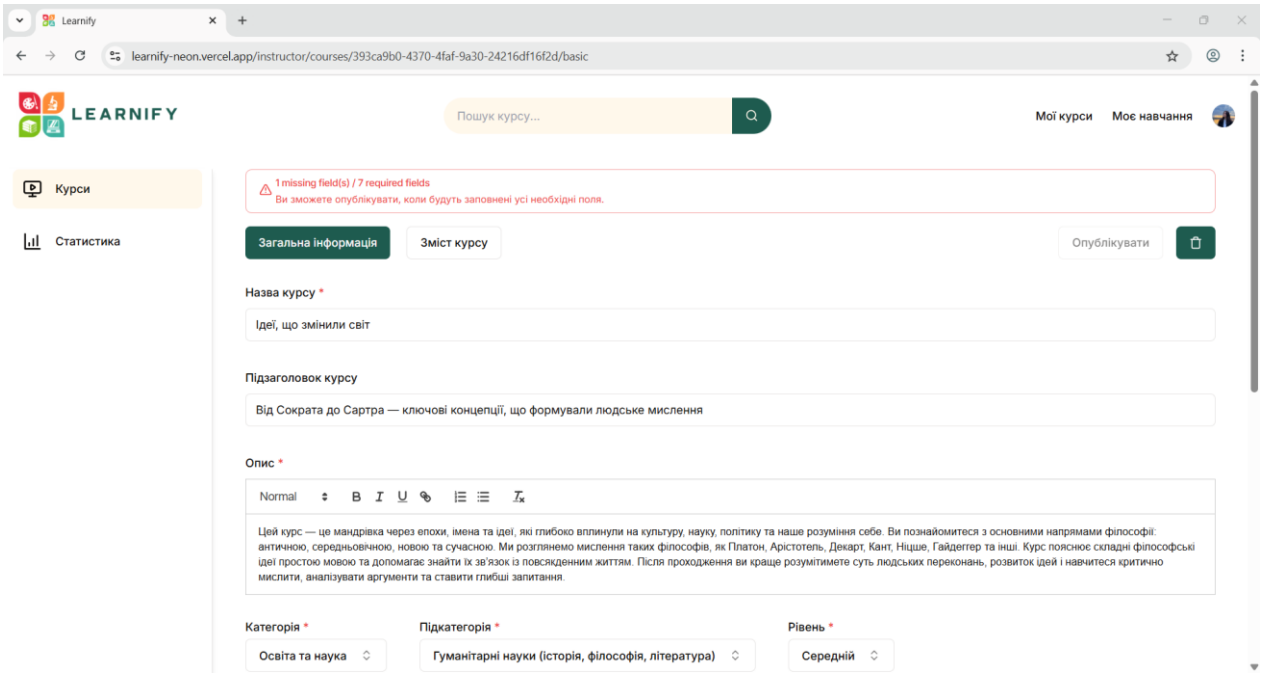


Рисунок 2.12 – Сторінка заповнення інформації про новий курс

Нижче блоку із повідомленням про кількість заповнених обов'язкових полів розміщено дві вкладки – «Загальна інформація», що активна за замовчуванням, та «Зміст курсу», яка призначена для додавання розділів курсу. У правій частині цього рядка розташовано кнопку «Опублікувати», яка є неактивною доти, доки не буде заповнено всі обов'язкові поля, а також кнопку з іконкою кошика, що дозволяє видалити курс на цьому етапі.

У нижній частині форми передбачено кнопки «Скасувати» та «Зберегти» для відповідного скасування змін або збереження введеної інформації.

Перейшовши на вкладку «Зміст курсу», користувач отримує доступ до управління структурою навчального курсу. На цій вкладці відображаються всі створені розділи, які мають вигляд окремих бежевих блоків. Ліворуч від кожної назви розміщено піктограму для перетягування, що дозволяє змінювати порядок розділів шляхом drag-and-drop [30]. Справа знаходиться значок олівця, натискання на який відкриває режим редагування відповідного розділу.

Нижче розташована форма для додавання нового розділу. Щоб створити розділ, користувачу потрібно ввести його назву у відповідне поле та натиснути кнопку «Створити» (див. рисунок 2.13).

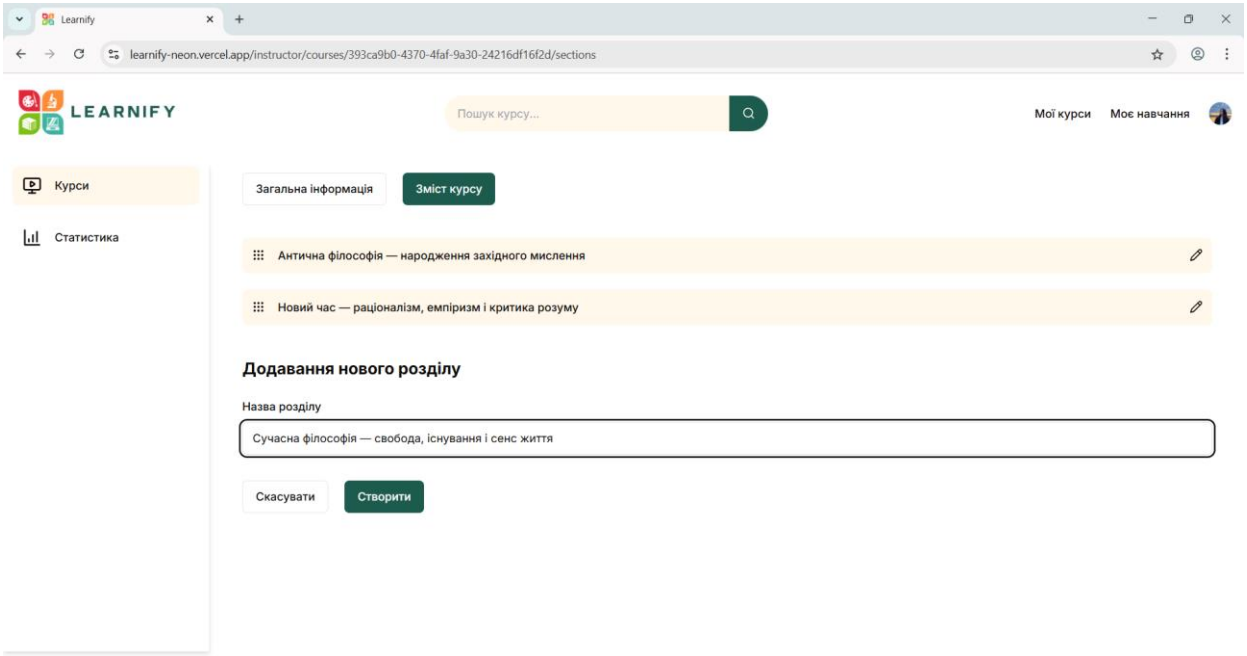


Рисунок 2.13 – Сторінка додавання нового розділу курсу

Натиснення кнопки «Створити» призводить до появи окремої сторінки з розширеною формою, що має ту ж логіку, що і форма створення курсу. Але додатково тут реалізовано додавання відеоуроків, файлових матеріалів тощо (див. рисунок 2.14).

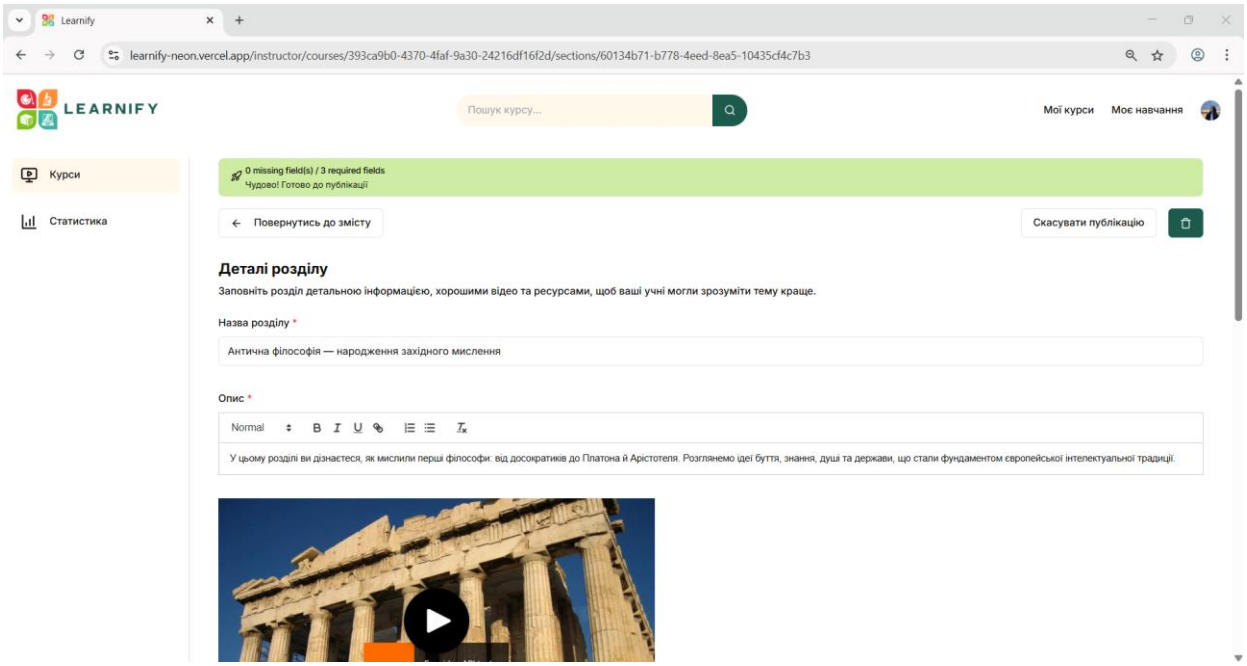


Рисунок 2.14 – Форма створення нового розділу

Після створення розділу викладач має змогу додавати до нього відеоуроки та допоміжні матеріали. Відео додається через інтеграцію з Mux – це сучасна відеоплатформа, яка дозволяє транслювати, зберігати та оптимізувати відеоконтент без потреби у складному технічному налаштуванні [31]. Система автоматично обробляє завантажене відео, забезпечуючи його сумісність з усіма типами пристроїв та якісне відтворення.

Допоміжні матеріали додаються через вбудовану систему завантаження файлів, що підтримує основні формати та дозволяє легко прикріплювати необхідні ресурси до відповідного уроку (див. рисунок 2.15).

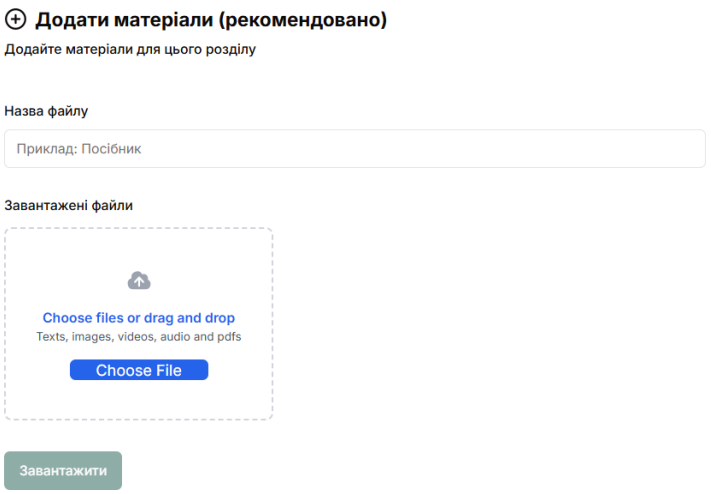


Рисунок 2.15 – Форма додавання допоміжних матеріалів

Для кожної секції також можна вказати, чи вона є безкоштовною для перегляду. Це особливо корисно, якщо автор курсу хоче надати потенційним слухачам ознайомлювальний доступ до частини навчального матеріалу – для демонстрації структури, стилю викладання або якості контенту. Вигляд цього елементу інтерфейсу показано на рисунку 2.16.

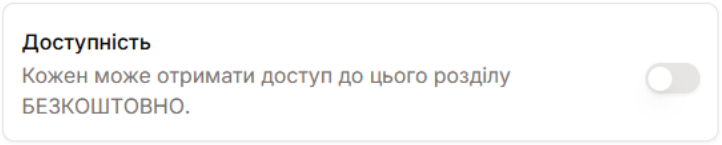


Рисунок 2.16 – Елемент керування доступністю розділу курсу з можливістю безкоштовного ознайомлення

Варто зазначити, що курс може бути опублікований лише за умови заповнення всіх обов’язкових полів, а також за наявності принаймні одного опублікованого розділу. Це є необхідною умовою для забезпечення повноти і змістовності навчального продукту перед його публікацією для користувачів. Можливість опублікувати розділ з’являється після того, як заповнено усі обов’язкові поля (див. рисунок 2.17).

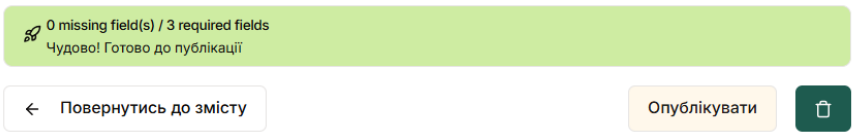


Рисунок 2.17 – Публікація розділу курсу

Редагування створеного курсу можливе в будь-який момент. Інструктор може оновити текстову інформацію, замінити відео, додати або видалити розділи.

2.3.3 Перегляд та проходження курсів студентами платформи Learnify

Після авторизації користувач може переглядати навчальні курси, доступні на платформі Learnify. Усі опубліковані курси відображаються на головній сторінці у форматі окремих блоків (див. рисунок 2.18).

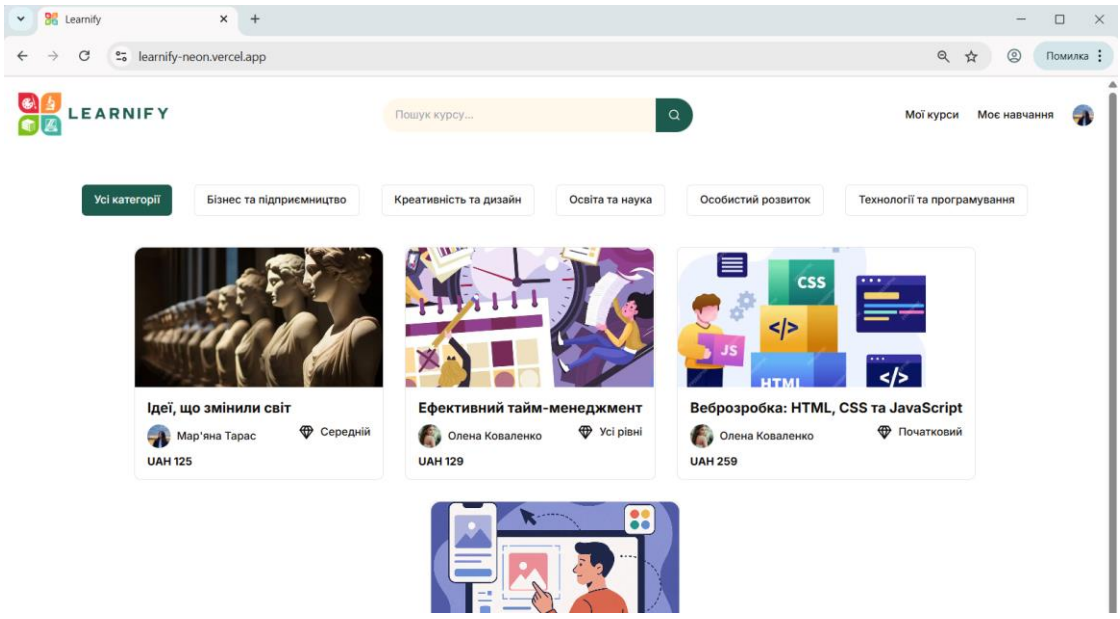


Рисунок 2.18 – Сторінка з доступними курсами платформи Learnify

Варто зауважити, що у верхньому горизонтальному меню ще є вкладка «Моє навчання», яка перенаправляє користувача на сторінку, на котрій

відображаються як уже придбані, так і безкоштовні курси, до яких користувач має доступ, або вже проходив їх раніше (див. рисунок 2.19).

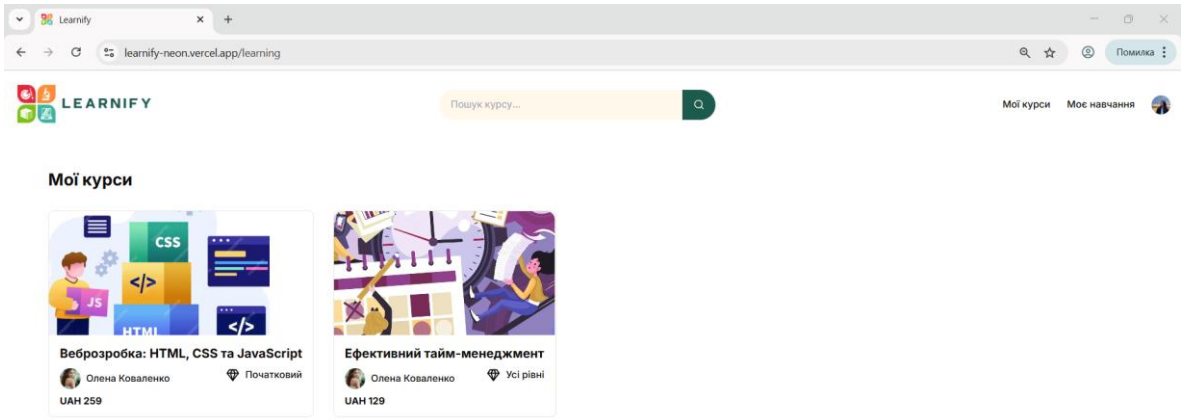


Рисунок 2.19 – Сторінка «Моє навчання», що містить інформацію про курси користувача

В обох випадках кожен курс представлений у вигляді окремого блоку, що відображає інформацію про назву курсу, його автора, рівень складності, ціну та обкладинку курсу.

Натиснувши на курс на головній сторінці платформи Learnify, користувач платформи потрапляє на детальну сторінку обраного курсу, де представлено ключову інформацію: назву, підзаголовок, повний опис курсу та інші дані. Інтерфейс сторінки організовано зручно для користувача: у лівій частині розташоване навігаційне меню (компонент CourseSideBar), яке відображає структуру курсу — список усіх наявних розділів та уроків. Це меню дозволяє легко перемикатися між розділами та переходити безпосередньо до перегляду конкретного уроку, забезпечуючи зручну навігацію по навчальному контенту. Вигляд описаної сторінки показано на рисунку 2.20.

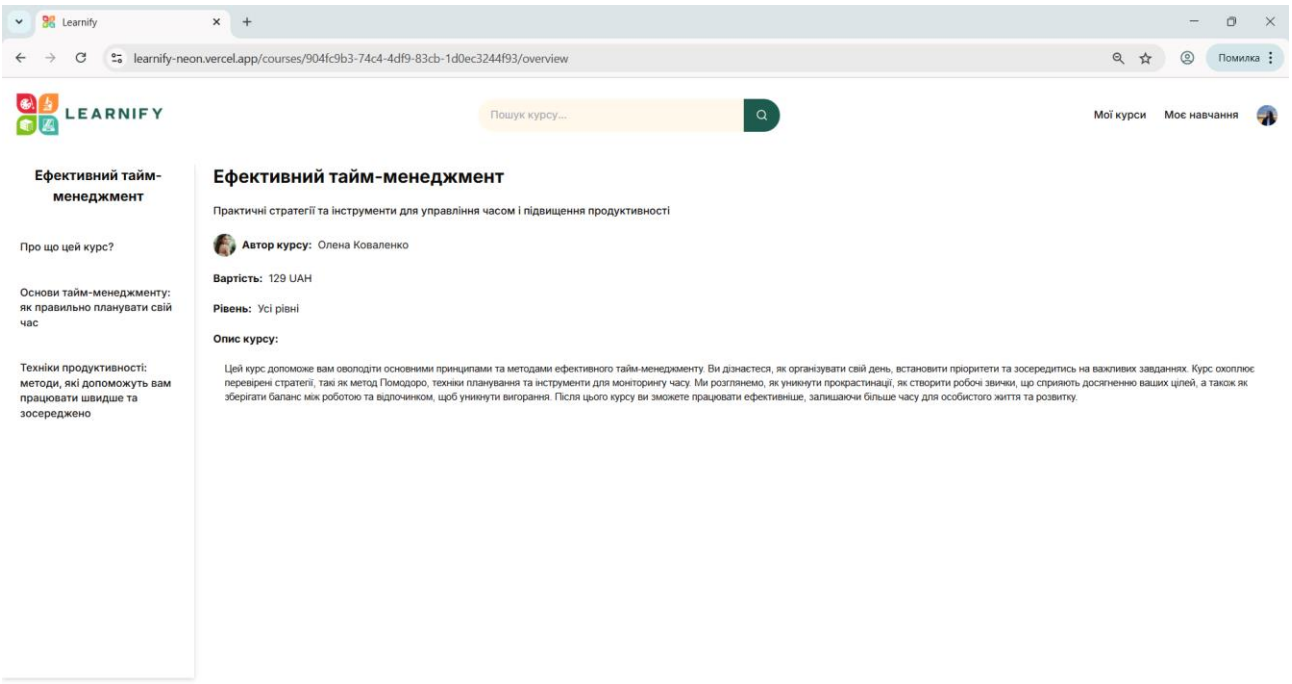


Рисунок 2.20 – Сторінка перегляду детальної інформації про курс

Під час перегляду розділів курсу студент може зіткнутися з двома сценаріями відображення контенту, залежно від того, чи позначено автором курсу цей розділ як доступний безкоштовно.

У першому випадку, якщо розділ був відзначений викладачем як безкоштовний для ознайомлення, студент отримує повний доступ до вмісту розділу. На сторінці відображаються назва, опис, а також відеоурок, який можна переглядати без обмежень.

Додатково, у правому верхньому куті інтерфейсу розміщена кнопка «Придбати цей курс», яка дозволяє зацікавленому користувачу одразу перейти до оформлення покупки повного доступу до курсу. Відображення розділу з безкоштовним доступом показано на рисунку 2.21.

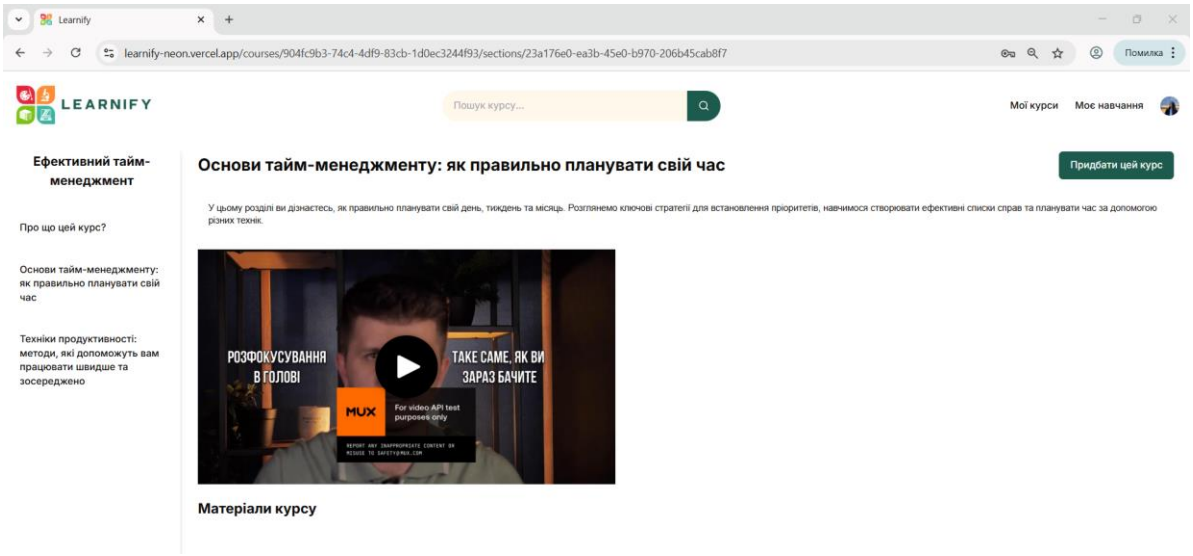


Рисунок 2.21 – Перегляд розділу із безкоштовним доступом

У другому випадку, якщо розділ не є відкритим для безкоштовного перегляду, студент все одно може переглянути назву та короткий опис розділу, однак замість відеоуроку з’являється інформаційне повідомлення: «Відео для цього розділу курсу недоступне. Придбайте курс, щоб отримати доступ!».

У цьому ж вікні також доступна кнопка «Придбати курс», що слугує прямим закликом до дії та полегшує процес придбання. Вигляд сторінки із відображенням розділу з обмеженим доступом показано на рисунку 2.22.

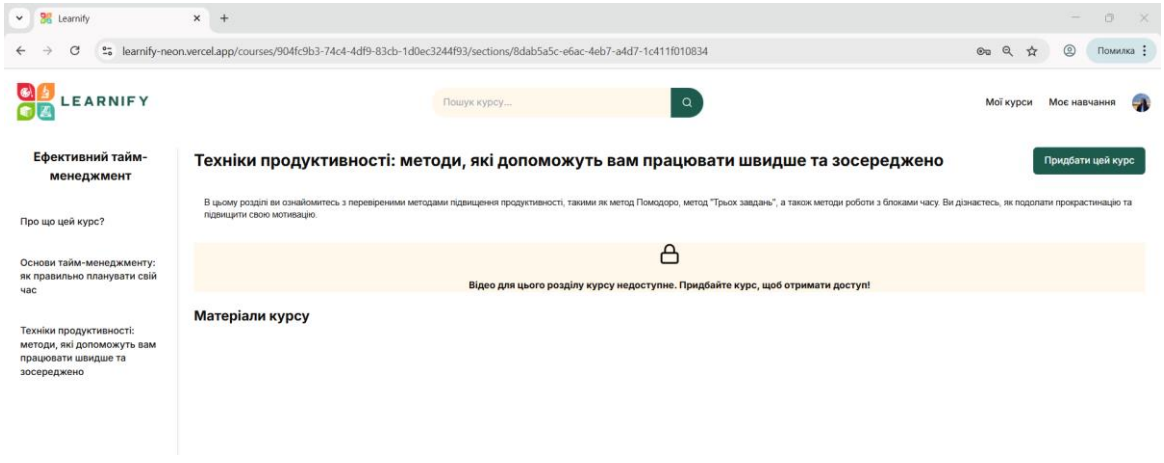


Рисунок 2.22 – Перегляд розділу із обмеженим доступом

Такий підхід дозволяє авторам курсів ефективно демонструвати частину контенту для ознайомлення.

Якщо користувач після ознайомлення з курсом та його змістом приймає рішення придбати курс і натискає кнопку «Придбати цей курс», його перенаправляє на платіжну сторінку, що реалізована через інтеграцію з платіжною системою Stripe (див. рисунок 2.23).

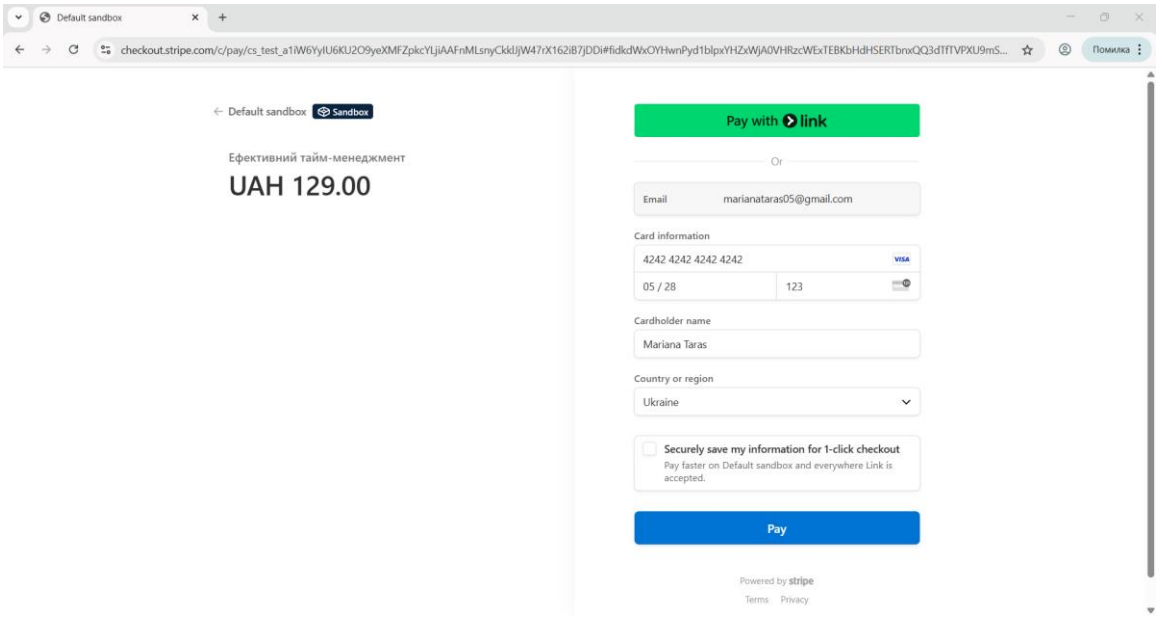


Рисунок 2.23 – Сторінка обробки платежу

На цій сторінці користувачу відображається ключова інформація про покупку:

- назва курсу;
- ціна у гривнях;
- а також форма для введення платіжних даних.

Праворуч розміщено поля для введення електронної пошти, даних платіжної картки (номер, термін дії, код CVC), імені власника картки та країни. Нижче є опція збереження платіжної інформації для пришвидшеної оплати в майбутньому. Кнопка «Pay» завершує процес придбання, після чого користувач отримує повний доступ до всіх розділів курсу, що був оплачений.

Такий підхід забезпечує простий та зрозумілий механізм оплати, використовуючи надійну інфраструктуру Stripe, яка підтримує популярні платіжні системи – Visa, Mastercard та інші [32].

Після успішної оплати курсу студент отримує повний доступ до його змісту. Інтерфейс курсу динамічно змінюється: у меню CourseSideBar, що розташоване ліворуч, з’являється індикатор прогресу проходження (див. рисунок 2.24). Він реалізований у вигляді прогрес-бару та текстового відображення відсотка виконання: [Progress bar] + «X% completed», де значення розраховується на основі кількості розділів, які студент позначив як завершені.

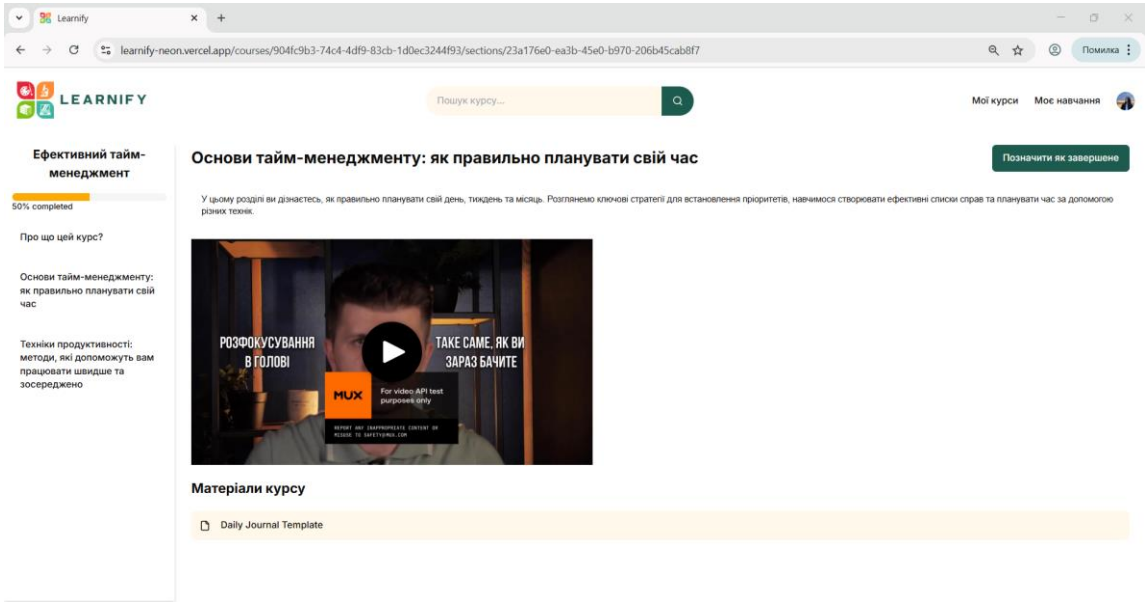


Рисунок 2.24 – Вигляд інтерфейсу перегляду курсу після оплати

Крім того, користувач отримує повний доступ до матеріалів кожного розділу, оскільки навіть якщо розділ був доступним безкоштовно, його матеріали були недоступні до перегляду для користувачів.

У верхньому правому куті інтерфейсу тепер відображається кнопка «Позначити як завершене». Вона дозволяє студенту зберігати статус проходження для кожного окремого розділу. Якщо студент вже переглянув відео та опрацював матеріали, він може натиснути цю кнопку, щоб система врахувала розділ як завершений. При цьому користувач у будь-який момент може змінити статус розділу, знову відкривши його та скасувавши позначку про завершення.

Таким чином, система проходження курсів у Learnify є інтерактивною, зручною та адаптивною, дозволяючи користувачам гнучко керувати власним навчальним прогресом.

Функціональність проходження навчальних курсів на платформі Learnify створена з урахуванням принципів зручності, адаптивності та послідовності подачі матеріалу, що сприяє ефективному засвоєнню навчального контенту.

2.4 Інтерфейс користувача та дизайн платформи Learnify

Інтерфейс користувача є ключовим чинником зручності та ефективності взаємодії із інтерактивною платформою Learnify для онлайн-навчання. У цьому пункті будуть розглянуті основні рішення, пов’язані з візуальним оформленням, стилізацією та адаптивністю інтерфейсу.

2.4.1 Обґрунтування кольорової схеми інтерфейсу користувача

Кольорова гама платформи Learnify була обрана з урахуванням психологічного впливу кольорів на користувачів, а також потреби у створенні чистого, сучасного й водночас дружнього візуального стилю. Основні кольори, використані в інтерфейсі:

- Темно-зелений (#1E5B4F) – як домінантний відтінок, що сприяє концентрації, створює відчуття довіри й стабільності.
- Світло-бежевий (#FFF8EB) – м’який фон, який зменшує зорове навантаження.
- Білий (#FFFFFF) – для збереження чистоти і простоти інтерфейсу.

У поєднанні ці кольори створюють гармонійний, спокійний інтерфейс, який не відволікає від навчального процесу та підсилює емоційне сприйняття контенту.

На рисунку 2.25 зображено основну кольорову палітру платформи, що використовується для оформлення фону, акцентів та функціональних елементів інтерфейсу.

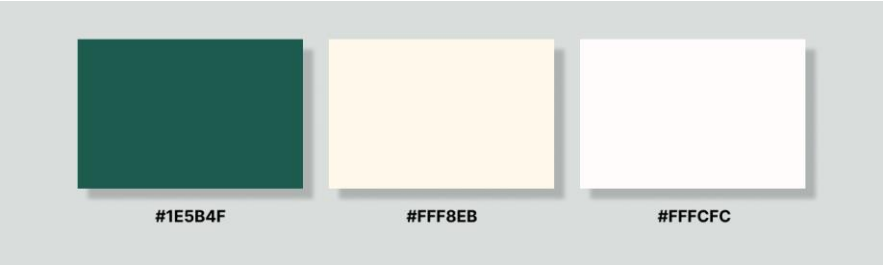


Рисунок 2.25 – Основна кольорова палітра платформи Learnify

Окремо було створено логотип платформи, який поєднує яскраві кольори: бірюзовий (#3EB2A0), зелений лайм (#7EC118), помаранчевий (#FF8E37) і червоний (#E42D3C). Вони символізують різні напрями активності, динаміку навчального процесу та індивідуальність кожного студента. Логотип показано на рисунку 2.26.

Рисунок 2.26 – Логотип інтерактивної платформи Learnify

Загальна колористика платформи Learnify спрямована на досягнення балансу між естетичною привабливістю інтерфейсу та забезпеченням комфортного середовища для навчання. Вона не перевантажує користувача яскравими чи контрастними відтінками, водночас підтримуючи візуальну виразність елементів, необхідних для орієнтації та навігації. Такий підхід дозволяє зберігати увагу студента на навчальному контенті.

2.4.2 Використання Tailwind CSS для стилізації компонентів

Для стилізації інтерфейсу платформи Learnify використовується фреймворк Tailwind CSS, який дозволяє максимально ефективно впроваджувати підхід utility-first [33]. Завдяки цьому більшість стилів задається безпосередньо в HTML-коді за допомогою класів Tailwind, що суттєво прискорює розробку і спрощує підтримку компонентів.

Крім базового стилювання, у проєкті було інтегровано кілька готових інтерфейсних компонентів з бібліотеки shadcn/ui, яка гармонійно поєднується з Tailwind CSS та забезпечує високий рівень кастомізації. Серед використаних компонентів:

- Switch – застосовується в налаштуваннях користувача для ввімкнення або вимкнення певних опцій. Компонент дозволяє створити сучасний і доступний перемикач зі зручною анімацією.
- Alert Dialog – використовується для підтвердження важливих дій, таких як видалення об'єктів або вихід із профілю. Компонент забезпечує фокусування уваги на критичних діях, блокує фонову взаємодію та відповідає принципам доступності.
- Data Table – застосовується в адміністративних панелях для зручного представлення великого обсягу даних з можливістю сортування, фільтрації та пагінації. Компонент дозволяє налаштувати структуру таблиці під потреби конкретного представлення.
- Progress – використовується для візуалізації відсотка завершення курсу користувачем. Цей елемент розміщується, зокрема, у бічному меню курсу та дає змогу легко орієнтуватися в навчальному прогресі.
- Card – базовий структурний блок, що використовується для представлення окремих курсів, секцій або іншого контенту у вигляді впорядкованих інформаційних блоків. Card-компонент забезпечує послідовний стиль і розмітку на всій платформі.

Конфігураційний файл tailwind.config.js містить визначення кастомних кольорів через CSS-змінні (hsl(var(--...))), розміри контейнерів, радіуси округлень тощо. Це дає змогу легко масштабувати стилі в майбутньому та адаптувати тему інтерфейсу під різні потреби платформи. Вміст файлу tailwind.config.js наведено у додатку Е.

2.4.3 Адаптивний дизайн для різних пристроїв та забезпечення зручної навігації для взаємодії користувача із платформою

Платформа Learnify побудована з урахуванням принципів mobile-first розробки, що дозволяє забезпечити комфортне користування нею на смартфонах, планшетах та десктопах [34].

У мобільній версії інтерфейсу передбачено використання бургер-меню замість вкладок "Мої курси" та "Моє навчання", а також автоматичне приховування рядка пошуку (див. рисунок 2.27).

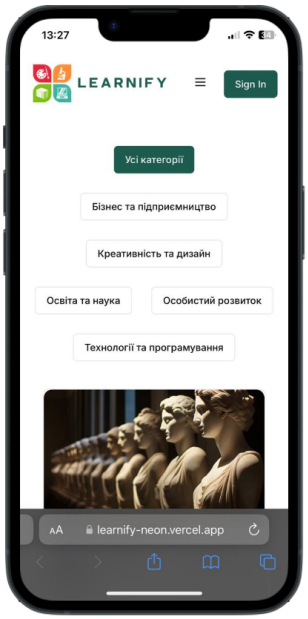


Рисунок 2.27 – Вигляд мобільної версії платформи Learnify

На сторінці курсу у мобільній версії платформи справа вгорі відображається кнопка «Розділи». Вона відкриває бічне меню з переліком

розділів курсу, що дає змогу зручно перемикатися між розділами та закриватися окремою кнопкою без втрати контексту перегляду. Вигляд описаних елементів інтерфейсу у мобільній версії платформи показано на рисунку 2.28.

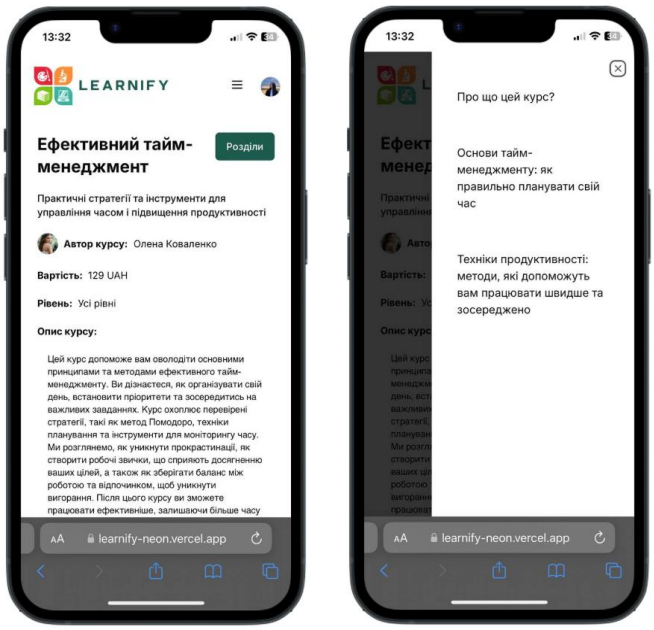


Рисунок 2.28 – Вигляд кнопки «Розділи» та бічного меню CourseSideBar

Для базового тестування адаптивності використовувались DevTools браузера та адаптивні брейкпоінти Tailwind CSS (sm, md, lg, xl, 2xl). На подальших етапах розробки платформи застосовуватимуться інструменти для більш комплексного тестування інтерфейсу на різних пристроях і операційних системах.

2.5 Висновок до другого розділу

Другий розділ присвячено розробці архітектури та користувацького інтерфейсу платформи Learnify – важливому етапу створення програмного продукту. Головною метою цього розділу було формування функціональної, логічно структурованої та естетично привабливої системи, яка відповідала б сучасним стандартам цифрових освітніх платформ. У рамках розробки цього

розділу було створено базу даних зі структурою, ретельно спроектованою для надійного зберігання й ефективної обробки інформації про курси, користувачів і транзакції.

Було обґрунтовано вибір технологічного стеку: Next.js, Prisma, MySQL, Stripe, Clerk та Tailwind CSS, а також пояснено переваги їх поєднання для побудови масштабованого та безпечного вебзастосунку. Значну увагу приділено дизайну інтерфейсу: розглянуто структуру сторінок, роль навігації, типографіку, кольорову гаму й адаптивність інтерфейсу. Інтерфейсні компоненти створено за допомогою бібліотеки ShadCN UI, що дозволило досягти візуальної гармонії та гнучкості у налаштуванні дизайну платформи.

Практична цінність виконаної роботи полягає в успішному втіленні концепції Learnify у вигляді інтерактивної платформи з багатофункціональним інтерфейсом, що відповідає вимогам сучасного освітнього середовища. Завдяки комплексному інженерному підходу до розробки структури даних і врахуванню аспектів користувацького досвіду створено міцну основу для подальшого тестування, вдосконалення та впровадження платформи в освітній процес.

РОЗДІЛ 3. РОЗГОРТАННЯ, ТЕСТУВАННЯ ТА ПІДТРИМКА ПЛАТФОРМИ LEARNIFY

3.1 Розгортання платформи Learnify в Інтернеті

Розгортання платформи Learnify відбувається на сучасному наборі технологій, що забезпечує ефективну роботу, масштабованість і зручне керування. Після завершення локальної розробки, проєкт розгорнуто у хмарному середовищі за допомогою платформи Vercel, яка забезпечує автоматичне розгортання з віддаленого репозиторію на GitHub або GitLab [35]. Основні етапи розгортання включають:

- Підключення сховища коду до Vercel (через GitHub або GitLab).
- Налаштування змінних середовища, зокрема доступу до бази даних, ключів доступу до Clerk, Stripe тощо.
- Встановлення схеми бази даних Prisma та її ініціалізація.
- Автоматичне створення збірки та розгортання на робоче середовище після кожного оновлення коду.

Завдяки застосуванню Vercel розгортання відбувається оперативно, а також полегшується підтримка безперервної інтеграції та доставки (CI/CD) [36].

3.2 Валідація й тестування платформи

3.2.1 Валідація розмітки інтерфейсу та тестування платформи

Перевірка HTML-розмітки здійснювалася за допомогою Website Validator, що дозволяє виявити потенційні помилки у структурі документа, некоректно вкладені елементи або застарілі атрибути [37]. Результати перевірки показані на рисунку 3.1.

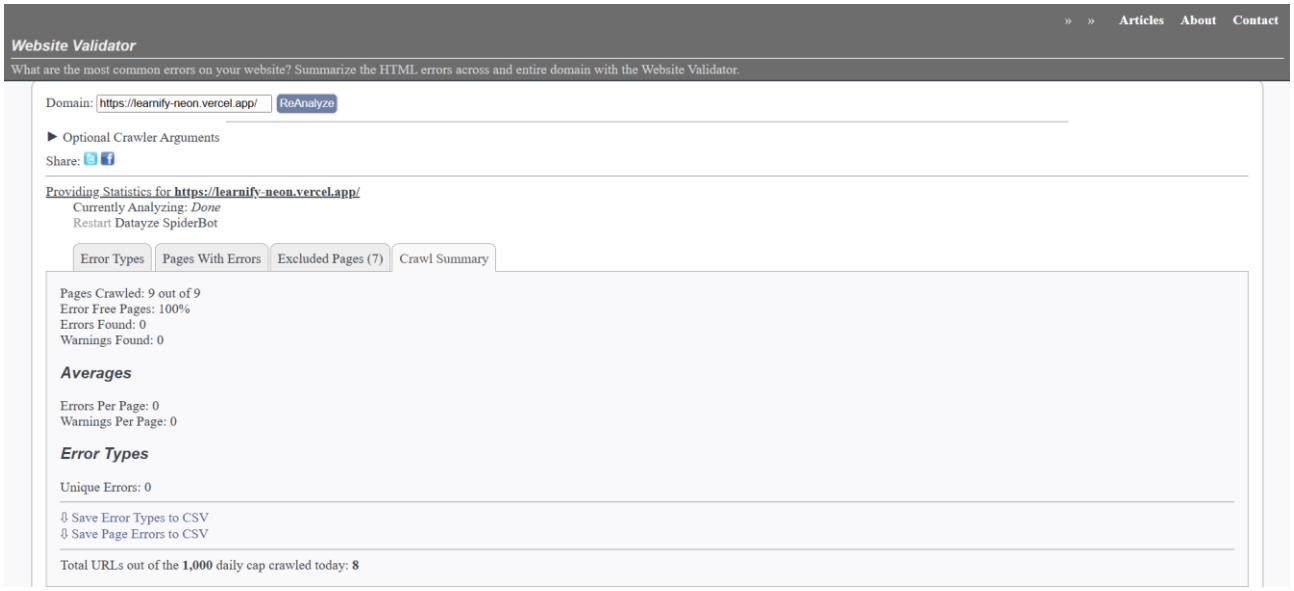


Рисунок 3.1 – Перевірка HTML-розмітки платформи Learnify

Платформа Learnify показала позитивні результати, що свідчить про дотримання сучасних веб-стандартів.

Крім того, в рамках тестування вручну перевірено основні сценарії взаємодії користувача: реєстрація, перегляд курсів, перегляд безкоштовних і платних розділів, додавання курсів, проходження розділів, зміна налаштувань профілю. Особливу увагу приділено логіці доступу до відеоматеріалів і відображення елементів інтерфейсу залежно від ролі користувача.

3.2.2 Кросбраузерне тестування платформи Learnify

З метою забезпечення стабільної та передбачуваної роботи платформи Learnify в різних середовищах було проведено кросбраузерне тестування з використанням сервісу BrowserStack. Цей інструмент дозволяє здійснити перевірку роботи інтерфейсу на реальних пристроях і в широкому спектрі браузерів без необхідності фізичного доступу до кожної конфігурації [38].

На рисунку 3.2 показано інтерфейс вебплатформи BrowserStack.

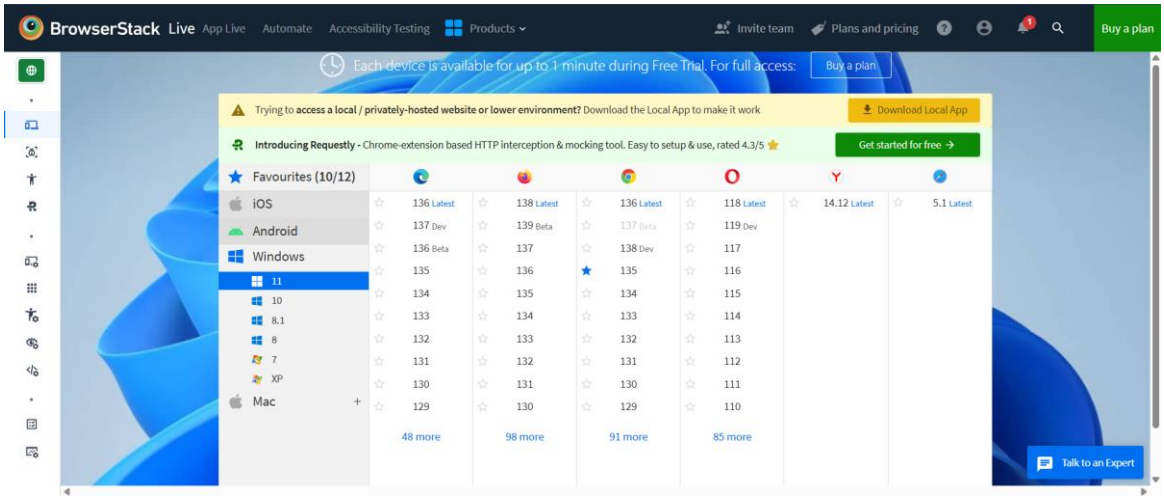


Рисунок 3.2 – Інтерфейс вебсервісу BrowserStack

Тестування проводилося у провідних браузерах, таких як Google Chrome, Mozilla Firefox, Microsoft Edge і Safari, на платформах Windows та macOS. Особливу увагу було зосереджено на:

- коректному відображенню стилів;
- працездатності клієнтських скриптів;
- узгодженості адаптивної верстки для різних розмірів екранів.

Платформа підтримує адаптивне відображення як на мобільних, так і на десктопних пристроях. У ході перевірки було протестовано відображення ключових елементів інтерфейсу: адаптивного головного меню, кнопок взаємодії, бокових панелей, динамічного контенту та спливаючих вікон. Додатково використано інструменти DevTools для ручної перевірки специфічних поведінкових сценаріїв і елементів керування.

Результати тестування засвідчили стабільну роботу платформи в більшості поширених конфігурацій, що дозволяє гарантувати однаковий досвід для користувачів незалежно від вибраного браузера або пристрою.

3.2.3 Тестування швидкості роботи платформи

Для оцінки швидкодії та загальної продуктивності платформи Learnify було використано онлайн-інструмент Google PageSpeed Insights. Цей сервіс є

одним із провідних рішень для аналізу вебсторінок, оскільки забезпечує комплексну оцінку ефективності на основі реальних користувацьких даних і емуляції різних пристроїв [39]. PageSpeed Insights використовує рушій Lighthouse і дозволяє отримати як технічні показники, так і рекомендації для поліпшення продуктивності сайту [40].

Тестування проводилося окремо для мобільної та комп'ютерної (десктопної) версій інтерфейсу, що надало можливість оцінити адаптивність та відповідність вимогам користувачів на різних пристроях.

Аналіз Google PageSpeed Insights включає такі ключові метрики:

- Ефективність (Performance) – загальна швидкість завантаження та взаємодії.
- Доступність (Accessibility) – наскільки інтерфейс зручний для людей з обмеженими можливостями.
- Оптимальні методи (Best Practices) – відповідність сучасним вебстандартам і безпеці.
- Оптимізація для пошукових систем (SEO) – як платформа підготовлена для коректної індексації.

На рисунках 3.3-3.4 показано результати тестування мобільної та десктопної версій платформи Learnify за допомогою Google PageSpeed Insights.

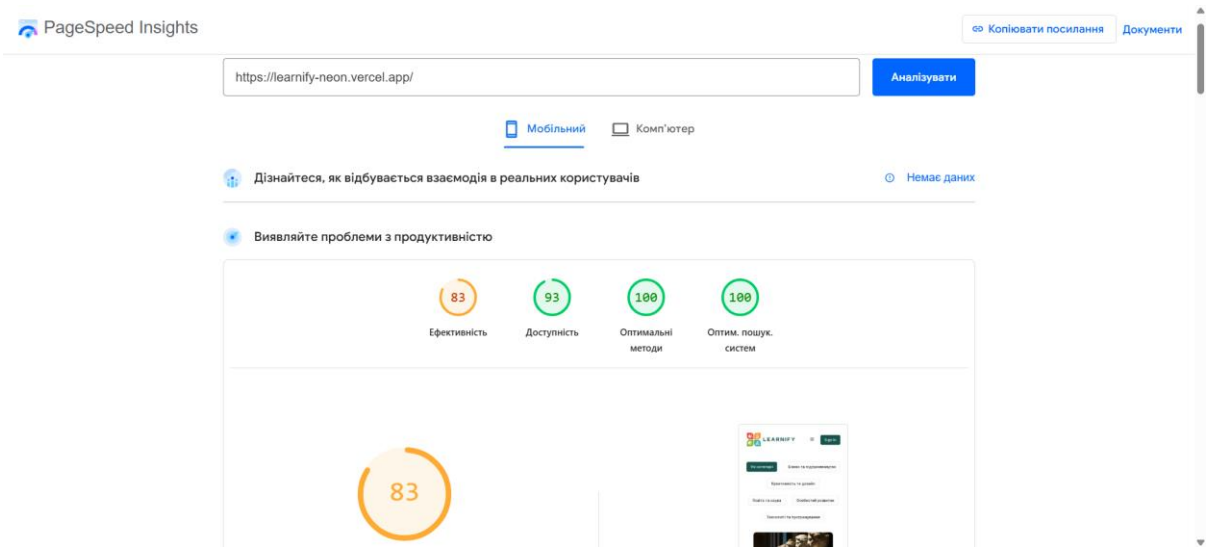


Рисунок 3.3 – Результати тестування мобільної версії платформи Learnify

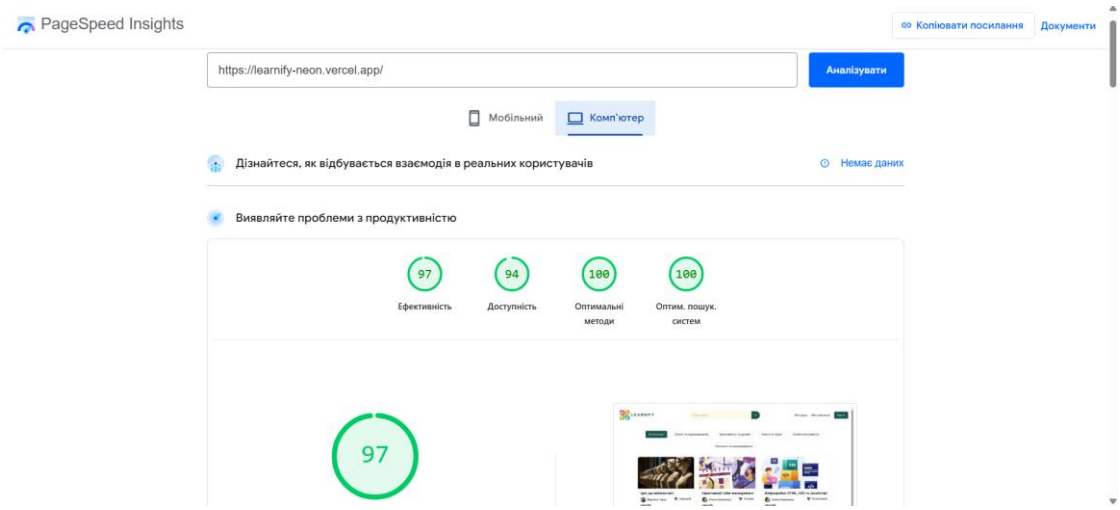


Рисунок 3.4 – Результати тестування десктопної версії платформи Learnify

Результати тестування показали високі показники як на мобільних, так і на десктопних пристроях, що свідчить про відповідність Learnify сучасним стандартам веброзробки та готовність до роботи в умовах різноманітного користувацького середовища.

3.3 Інструкція з популяризації та підтримки цифрової інтерактивної платформи Learnify

Після технічного запуску будь-якого освітнього ресурсу постає питання його подальшої підтримки, регулярного оновлення та ефективного просування серед цільової аудиторії. У випадку з Learnify важливо не лише забезпечити безперебійну технічну роботу платформи, а й сформувати сталу присутність у медіапросторі та підтримувати активну взаємодію з користувачами – студентами, викладачами та авторами освітніх курсів.

Одним із першочергових кроків для популяризації є публікація інформаційних матеріалів у соціальних мережах. Зокрема, платформа може розміщувати короткі відео, ролики у форматі Reels або TikTok, а також інфографіку, яка демонструє її функціональні переваги, простоту використання та унікальні можливості. Ці матеріали мають бути візуально привабливими й

інформативними, адаптованими під специфіку кожного каналу (Instagram, Telegram, TikTok) та орієнтованими на молодіжну аудиторію.

Важливим напрямом розвитку є партнерство з освітніми закладами. Йдеться про встановлення співпраці з університетами, коледжами, школами та приватними освітніми ініціативами. Learnify може бути запропонована як додатковий інструмент для організації дистанційного або змішаного навчання, платформою для розміщення авторських курсів або середовищем для розробки та тестування сучасних методик викладання. Таке партнерство не тільки сприяє зростанню довіри до платформи, але й забезпечує природне розширення кола її користувачів.

Регулярне оновлення контенту та функціональності – ключ до підтримки платформи в актуальному стані. Це допомагає не лише утримувати інтерес користувачів, але й своєчасно реагувати на запити цільової аудиторії, забезпечуючи конкурентні переваги.

Важливо також організувати ефективний механізм збору зворотного зв’язку від користувачів. Це можуть бути вбудовані форми оцінювання курсів, поля для коментарів, а також спеціальна форма зв’язку з адміністрацією платформи. Аналіз отриманих відгуків дозволить оперативно виявляти проблеми, оцінювати якість контенту і приймати обґрунтовані рішення щодо подальшого вдосконалення функціоналу.

З технічної точки зору необхідно постійно здійснювати моніторинг працездатності платформи. Для цього використовуються інструменти аналітики (наприклад, Google Analytics), журнали помилок та системи сповіщення про збої. Регулярна перевірка продуктивності та швидкості завантаження сторінок, а також своєчасне реагування на технічні несправності – критично важливі для забезпечення стабільного користувацького досвіду.

Окрім технічної підтримки, важливою складовою подальшого розвитку платформи Learnify є впровадження систем аналітики даних. Збір і аналіз інформації про дії користувачів — зокрема, проходження курсів, тривалість перегляду відео, успішність виконання завдань та наданий зворотний зв’язок —

дозволяє глибше зрозуміти освітні потреби аудиторії. Такий підхід відкриває широкі можливості для персоналізації навчання, вдосконалення контенту, а також оптимізації маркетингових стратегій і функціоналу платформи [41].

3.4 Висновок до третього розділу

У третьому розділі докладно описано етапи публічного розгортання платформи Learnify, проведення її валідації, тестування, а також заходи з популяризації та підтримки. Головним завданням цього етапу стало підтвердження працездатності створеного продукту в реальному середовищі, забезпечення його сумісності з сучасними технологічними вимогами та готовності до використання кінцевими користувачами.

Було описано процес розгортання вебзастосунку за допомогою сучасного інструменту Vercel, що забезпечило швидке та безпечне виведення платформи в Інтернет. Було виконано валідацію HTML-розмітки та тестування функціоналу інтерфейсу, що дозволило виявити й усунути незначні помилки на початковому етапі. За допомогою сервісу BrowserStack було здійснено кросбраузерне тестування, що підтвердило стабільну роботу платформи Learnify в найпопулярніших браузерах і на різних типах пристроїв. Також проаналізовано швидкодію платформи через Lighthouse-інструменти, що дозволило оптимізувати показники продуктивності.

Окрему увагу приділено стратегіям підтримки та популяризації Learnify. Сформульовано рекомендації щодо присутності в соціальних мережах, співпраці з освітніми закладами, збору зворотного зв’язку від користувачів і технічного моніторингу стабільності платформи. Такі кроки створюють основу для подальшого розвитку й масштабування системи як цілісного освітнього інструменту.

Результати, викладені в третьому розділі, підтверджують, що Learnify є не лише технічно реалізованою платформою, але й готовим до запуску продуктом із реальним потенціалом застосування в сучасному освітньому просторі.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Ергономічні проблеми безпеки життєдіяльності

Ергономіка як наукова дисципліна відіграє важливу роль у забезпеченні безпеки життєдіяльності, особливо в умовах виробничого середовища та офісної праці. Її основна мета полягає в адаптації умов праці до фізіологічних, психологічних та анатомічних особливостей людини з метою зменшення професійних ризиків, підвищення ефективності праці та покращення якості життя працівників [42].

Серед основних ергономічних проблем, що впливають на безпеку життєдіяльності, виокремлюють невідповідність робочого місця антропометричним характеристикам працівника, нераціональне розташування засобів управління, надмірні фізичні або зорові навантаження, несприятливі мікрокліматичні умови, неадекватне освітлення, а також високий рівень шуму або вібрацій [43].

Зокрема, неправильна організація робочого місця може призводити до розвитку захворювань опорно-рухового апарату, зниження працездатності та підвищення ризику виробничого травматизму. Тривале перебування в статичному положенні без належної підтримки спини спричиняє перевантаження м’язів та хребта, що призводить до хронічного болю [44]. Водночас, розміщення робочого обладнання поза зоною комфортної досяжності створює додаткове навантаження на м'язово-скелетну систему.

Особливої уваги заслуговує проблема зорового навантаження, характерна для офісних працівників і операторів комп’ютерної техніки. Надмірне перебування перед екранами дисплеїв спричиняє синдром зорової втоми, який проявляється сухістю очей, погіршенням зору, головним болем та загальною втомою. За дослідженнями, понад 60% користувачів комп’ютерів відчувають ці симптоми вже після 2–3 годин роботи [45].

Ще одним не менш суттєвим аспектом є мікрокліматичні умови, що впливають на терморегуляцію організму, самопочуття працівника та його продуктивність. Надмірна температура, низька вологість або протяги можуть не лише спричинити дискомфорт, а й стати фактором ризику для серцево-судинної та дихальної систем. Оптимальні параметри мікроклімату на робочому місці визначені державними санітарними нормами (ДСН 3.3.6.042-99) і повинні дотримуватись у межах нормативів [46].

Значний вплив на безпеку має психоемоційне навантаження, яке виникає внаслідок монотонної діяльності, надмірної відповідальності або конфліктних ситуацій у трудовому колективі. Такі стани спричиняють розвиток хронічного стресу, що негативно позначається як на психологічному, так і на фізичному здоров'ї працівника, знижує його уважність і, відповідно, підвищує ймовірність помилок [47].

Важливою складовою ергономічного підходу до безпеки є організація режиму праці та відпочинку. Наявність перерв, чергування типів діяльності, можливість змінювати позу або переміщуватись у межах робочого місця сприяє зменшенню втоми та запобігає професійному вигоранню [48].

Застосування ергономічних принципів під час проєктування робочих місць, програмного забезпечення, засобів захисту та інтерфейсів управління дозволяє мінімізувати шкідливі впливи на організм людини, підвищити безпеку праці та знизити ризики травматизму. Відповідно, інтеграція ергономіки у систему управління охороною праці є необхідною умовою формування безпечного та здорового виробничого середовища.

4.2 Заходи з техніки безпеки при експлуатації обладнання

Створення безпечних умов праці під час експлуатації обладнання є важливою складовою системи охорони праці в галузі інформаційних технологій. Навіть у, здавалося б, «офісному» середовищі зношеність, неправильне використання чи нехтування технічним обслуговуванням

комп’ютерного та периферійного обладнання можуть призвести до аварійних ситуацій, коротких замикань, електротравм, а також до професійних захворювань, зокрема опорно-рухового апарату або зору [49].

Одним із найбільш важливих заходів є підтримання техніки у справному стані. Комп’ютери, принтери, мережеве обладнання та інші пристрої повинні регулярно проходити технічне обслуговування, очищення від пилу, перевірку на відсутність механічних пошкоджень, справність кабелів та блоків живлення. Експлуатація несправної апаратури заборонена, оскільки вона несе пряму загрозу здоров’ю працівника та цілісності IT-інфраструктури [50].

Важливим аспектом є інструктаж з охорони праці, який мають пройти всі працівники, що працюють із технічними засобами. Вступний, первинний та періодичний інструктажі повинні охоплювати правила безпечної роботи з офісною та комп’ютерною технікою, дії в разі перегріву пристрою, виникнення диму або запаху гару, а також вимоги щодо поводження з кабелями, розетками, подовжувачами тощо [51].

Особливу увагу слід приділити розміщенню обладнання. Комп’ютери та монітори мають бути встановлені на стійких поверхнях, з урахуванням вентиляції та віддаленості від джерел тепла або вологи. Необхідно уникати перевантаження електромережі, використовуючи сертифіковані мережеві фільтри та джерела безперебійного живлення (UPS). Серверне обладнання та маршрутизатори повинні розміщуватися у відповідних шафах із дотриманням правил циркуляції повітря [52].

Також необхідно враховувати ергономічні та психофізіологічні фактори (тривале сидіння, напруження зору та повторювані рухи) можуть спричинити хронічні захворювання. Робоче місце IT-фахівця має відповідати принципам ергономіки: правильна висота столу та стільця, належне розташування монітора, клавіатури та миші, а також дотримання режиму праці та перерв для зниження навантаження на організм [53].

Важливо дотримуватись інструкцій з експлуатації, які постачаються виробниками. У них зазначено допустимі режими роботи, правила

підключення, очищення, зберігання та вимоги до умов середовища (вологість, температура, запиленість). Ігнорування таких вказівок може призвести до поломок, втрати даних або ризику ураження електричним струмом.

Реалізація заходів з техніки безпеки в IT-середовищі повинна поєднувати належне технічне обслуговування обладнання, інформованість персоналу щодо потенційних загроз та дотримання норм експлуатації. Саме це дозволяє знизити виробничі ризики та підтримувати стабільність IT-інфраструктури.

4.3 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи розглянуто основні питання безпеки життєдіяльності та охорони праці у сучасному виробничому середовищі, зокрема ергономічні проблеми та заходи з техніки безпеки при експлуатації обладнання.

Аналіз ергономічних проблем показав, що саме неврахування антропометричних, фізіологічних та психологічних особливостей людини під час організації праці є однією з головних причин зниження працездатності, виникнення професійних захворювань та підвищення ризику травматизму. Водночас правильна ергономічна організація робочого простору сприяє покращенню умов праці, зменшенню втомлюваності, підвищенню продуктивності та безпеки роботи.

Забезпечення безпеки життєдіяльності вимагає системного підходу, що поєднує ергономічні, технічні, організаційні та психологічні складові. Лише всебічне дотримання вимог охорони праці може гарантувати створення безпечного, комфортного та ефективного робочого середовища, що відповідає сучасним стандартам виробничої культури та зберігає здоров’я працівників.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи освітнього рівня «Бакалавр» досягнуто головної мети – створення цифрової інтерактивної платформи Learnify для підтримки сучасного навчального процесу. Платформа реалізована з урахуванням актуальних вимог до функціональності, дизайну, безпеки та зручності користування, що робить її корисною як для студентів, так і для викладачів та незалежних авторів освітнього контенту.

У першому розділі подано загальний огляд цифрових освітніх платформ, висвітлено їхній вплив на трансформацію навчального процесу, розглянуто приклади реалізації та визначено ключові вимоги до таких систем. Також проведено аналіз основних архітектурних підходів до створення освітніх вебплатформ і технологій, що забезпечують їх масштабованість, продуктивність та доступність. Обґрунтовано вибір сучасного стеку веброзробки для розробки власного програмного рішення.

Другий розділ присвячено безпосередній розробці платформи Learnify, де докладно розглянуто структуру бази даних, принципи роботи ключових моделей та їх взаємозв’язки. Обґрунтовано вибір технологій, таких як Next.js, Prisma, Tailwind CSS та Clerk, і описано підхід до створення зручного, інтуїтивно зрозумілого інтерфейсу. Значну увагу приділено питанню візуальної стилістики платформи, включно з колористикою, типографікою та використанням бібліотеки інтерфейсних компонентів shadcn/ui. Усі ці рішення спрямовані на досягнення сучасного вигляду системи з високим рівнем зручності для кінцевого користувача.

У третьому розділі описано етапи розгортання платформи Learnify в мережі Інтернет, а також виконано її тестування та валідацію. Було проведено перевірку коректності розмітки, тестування адаптивності, кросбраузерну перевірку із залученням сервісу BrowserStack, а також оцінено швидкодію платформи за допомогою Lighthouse. Окремо розглянуто заходи щодо

подальшої підтримки та популяризації платформи, зокрема використання соціальних мереж, партнерство із закладами освіти, оновлення функціоналу та збір зворотного зв’язку.

Отримані результати дослідження та реалізації підтверджують важливість обраної теми, досягнення визначених цілей і завдань, а також ілюструють практичну користь платформи Learnify для підтримки сучасного освітнього процесу в цифровому середовищі.

У розділі «Безпека життєдіяльності, основи охорони праці» висвітлено основні аспекти забезпечення безпечних умов праці при роботі з обладнанням, а також охарактеризовано ергономічні чинники, що впливають на ефективність і безпечність професійної діяльності. Розглянуто типові загрози, пов’язані з недотриманням ергономічних вимог, та визначено комплекс технічних і організаційних заходів з техніки безпеки, спрямованих на мінімізацію професійних ризиків. Акцент зроблено на важливості системного підходу до охорони праці, що включає поєднання технічних рішень, нормативного регулювання та підвищення рівня обізнаності працівників.

ПЕРЕЛІК ДЖЕРЕЛ

1 Як викладачі й студенти оцінили якість організації освітнього процесу у 2023 році. Державна служба якості освіти. [Електронний ресурс]. 20.02.2024. Режим доступу до ресурсу: https://sqe.gov.ua/yak-vikladachi-y-studenti-ocinili-yakist/ (дата звернення: 01.05.2025).

2 Струтинська, І. В., Дмитроца, Л. П., Сороківська, О. А., & Козбур, Г. В. (2024). Особливості цифрового розвитку малого і середнього бізнесу України, країн Європи та G7. In Трансформація бізнесу для сталого майбутнього: дослідження, цифровізація та інновації: монографія (pp. 411-427). : ФОП Паляниця В. А.

3 Гончаров А. Перспективи ринку онлайн-навчання. Bestclevers. [Електронний ресурс]. 14.02.2025. Режим доступу до ресурсу: https://www.bestcleverslms.com/piznavalne/perspektyvy-rynku-onlayn-navchannia/ (дата звернення: 01.05.2025).

4 Strutynska, I., Dmytrotsa, L., Kozbur, H., & Melnyk, L. (2021). The Digital Business Transformation Index Determining and Monitoring: Development of a National Online Platform. In ITTAP (pp. 327-334).

5 Інтернет-курси Campster. Campster. [Електронний ресурс]. Режим доступу до ресурсу: https://www.thecampster.com/ua/course/index (дата звернення: 02.05.2025).

6 Chan C. Teachable’s 2024 Rewind: Celebrating a year of growth for creators. Teach:able. [Електронний ресурс]. 18.12.2024. Режим доступу до ресурсу: https://teachable.com/blog/rewind (дата звернення: 02.05.2025).

7 Прус О. Front-end без Next.JS — як спорткар без двигуна: аналізуємо переваги та недоліки. Wezom. [Електронний ресурс]. 26.07.2024. Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/front-end-bez-nextjs-yak-sportkar-bez-dviguna-analizujemo-perevagi-ta-nedoliki (дата звернення: 05.05.2025).

8 Сальник Р. Ефективний спосіб розпочати NextJS-проект. HIGHLOAD. [Електронний ресурс]. 08.07.2024. Режим доступу до ресурсу: https://highload.tech/uk/efektivnij-sposib-rozpochati-nextjs-proekt/ (дата звернення: 05.05.2025).

9 Turingvang. Using API routes for dynamic data fetching in Next.js. Medium. 11.10.2024. [Електронний ресурс]. Режим доступу до ресурсу: https://medium.com/@turingvang/using-api-routes-for-dynamic-data-fetching-in-next-js-2ff452779ef9 (дата звернення: 06.05.2025).

10 Тарас М. Сучасна веброзробка з використанням Next.js, React та Django REST. VIII Міжнародна студентська науково-технічна конференція «Природничі та гуманітарні науки. Актуальні питання»., Тернопіль. Тернопільський національний технічний університет ім. І.Пулюя (м.Тернопіль, 24–25 квіт. 2025 р.), 2025. С. 207–210.

11 Прус О. Розбираємо фреймворк Next.JS: визначення, призначення та переваги. Wezom. [Електронний ресурс]. 20.06.2024. Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/author/aleksandr-prus (дата звернення: 08.05.2025).

12 Should you use Prisma ORM?. Prisma Documentation. [Електронний ресурс]. Режим доступу до ресурсу: https://www.prisma.io/docs/orm/overview/introduction/should-you-use-prisma (дата звернення: 10.05.2025).

13 Реляційні бази даних: усе, що необхідно про них знати. Foxminded. [Електронний ресурс]. 26.10.2023. Режим доступу до ресурсу: https://foxminded.ua/reliatsiini-bazy-danykh/ (дата звернення: 10.05.2025).

14 Prisma Documentation. Relational databases. [Електронний ресурс]. Режим доступу до ресурсу: https://www.prisma.io/docs/getting-started/setup-prisma/start-from-scratch/relational-databases-node-mysql (дата звернення: 11.05.2025).

15 Stripe: найдорожча фінтех-компанія у Кремнієвій долині. Blog.imena.ua. [Електронний ресурс]. 05.06.2024. Режим доступу до ресурсу: https://www.imena.ua/blog/stripe/ (дата звернення: 11.05.2025).

16 Цимбалюк І. Overview of Stripe’s Payments System: Benefits and Features. Rates. [Електронний ресурс]. 11.06.2024. Режим доступу до ресурсу: https://rates.fm/payment-systems/overview-of-stripes-payments-system-benefits-and-features/ (дата звернення: 12.05.2025).

17 Three-Tier Client Server Architecture in Distributed System. GeeksForGeeks. [Електронний ресурс]. 08.11.2024. Режим доступу до ресурсу: https://www.geeksforgeeks.org/three-tier-client-server-architecture-in-distributed-system/ (дата звернення: 12.05.2025).

18 Архітектура вебзастосунків 2024: Ультимативний гайд для розробників. Robot_dreams media. [Електронний ресурс]. Режим доступу до ресурсу: https://robotdreams.cc/uk/blog/567-arhitektura-vebzastosunkiv (дата звернення: 12.05.2025).

19 Three-tier architectures. IBM. [Електронний ресурс]. 29.07.2024. Режим доступу до ресурсу: https://www.ibm.com/docs/en/was/8.5.5?topic=overview-three-tier-architectures (дата звернення: 12.05.2025).

20 Bogachenkov N. Dive into Next.js App Router: Building Dynamic, Nested, and Static Pages. DEV. [Електронний ресурс]. 10.10.2024. Режим доступу до ресурсу: https://dev.to/nik-bogachenkov/dive-into-nextjs-app-router-building-dynamic-nested-and-static-pages-4e22 (дата звернення: 13.05.2025)

21 Іздебський В. Що таке фреймворк: пояснюємо простими словами. WEZOM. [Електронний ресурс]. 23.11.2022. Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/scho-take-freymvork-poyasnyujemo-prostimi-slovami (дата звернення: 13.05.2025).

22 Directives: use client. Next.js Docs. [Електронний ресурс]. Режим доступу до ресурсу: https://nextjs.org/docs/app/api-reference/directives/use-client (дата звернення: 13.05.2025).

23 Mayur B. Prisma ORM: Simplifying Database Management in Node.js. Medium. [Електронний ресурс]. 31.12.2025. Режим доступу до ресурсу: https://medium.com/@newbmayur/prisma-orm-simplifying-database-management-in-node-js-300d481e0d97 (дата звернення: 13.05.2025).

24 Srivastav A. Understanding Database Relationships with Practical Examples. Medium. [Електронний ресурс]. 17.05.2024. Режим доступу до ресурсу: https://srivastavayushmaan1347.medium.com/understanding-database-relationships-with-practical-examples-d6a8f864c9c8 (дата звернення: 13.05.2025).

25 Data modeling. Prisma Documentation. [Електронний ресурс]. Режим доступу до ресурсу: https://www.prisma.io/docs/orm/overview/introduction/data-modeling (дата звернення: 14.05.2025).

26 Bhadak M. Using Clerk Authentication Webhooks with Next.js || Sync Clerk data to your Database. DEV. [Електронний ресурс]. 13.02.2025. Режим доступу до ресурсу: https://dev.to/mihir_bhadak/using-clerk-authentication-webhooks-with-nextjs-sync-clerk-data-to-your-database-2pni (дата звернення: 14.05.2025).

27 Kemal T. What is schema in prisma. Medium. [Електронний ресурс]. 23.06.2023. Режим доступу до ресурсу: https://medium.com/@kemaltf_/what-is-schema-in-prisma-1f52ddac630e (дата звернення: 14.05.2025).

28 Романець, А., & Козбур, Г. В. (2022). Безпека соцмережі під час аутентифікації користувача. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “Тернопільського національного технічного університету імені Івана Пулюя, 45-45.

29 <UserButton /> component. Clerk Docs. [Електронний ресурс]. Режим доступу до ресурсу: https://clerk.com/docs/components/user/user-button (дата звернення: 14.05.2025).

30 Atak Y. How to implement a simple drag-and-drop using Create-React-App and react-beautiful-dnd?. DEV. [Електронний ресурс]. 05.04.2022. Режим доступу до ресурсу: https://medium.com/codex/how-to-implement-a-simple-drag-and-drop-using-create-react-app-and-react-beautiful-dnd-4e6e57a2299f (дата звернення: 14.05.2025).

31 Mux: Video streaming APIs, Data, and Players. Mux. [Електронний ресурс]. Режим доступу до ресурсу: https://www.mux.com/ (дата звернення: 14.05.2025).

32 Технічні можливості Stripe. Інтерв'ю з Володимиром (Lead PHP Developer). New Line Technologies. [Електронний ресурс]. Режим доступу до ресурсу: https://newline.tech/texnichni-mozhlyvosti-stripe-intervyu/ (дата звернення: 14.05.2025).

33 Micheletti R. What is, why and how to use Tailwind CSS. Medium. [Електронний ресурс]. 08.01.2025. Режим доступу до ресурсу: https://blog.devgenius.io/what-is-why-and-how-to-use-tailwind-css-7a687b5d0eb4 (дата звернення: 14.05.2025).

34 Fetisov M. Все, що потрібно знати про Mobile First Design: актуальність, переваги та труднощі. Wezom. [Електронний ресурс]. 08.04.2024. Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/vsyo-chto-vam-nuzhno-znat-o-mobile-first-design-uzhe-sejchas (дата звернення: 15.05.2025).

35 Patil R. How to Deploy a Website or Web App on Vercel. Medium. [Електронний ресурс]. 03.10.2024. Режим доступу до ресурсу: https://medium.com/@rashmipatil24/deploy-a-website-or-web-app-on-vercel-f5a6a2a4f0ab (дата звернення: 15.05.2025).

36 Cruz J. J. Building a CI/CD Pipeline with Vercel and GitHub Actions. Medium. [Електронний ресурс]. 11.04.2022. Режим доступу до ресурсу: https://medium.com/@jjzcru/building-a-ci-cd-pipeline-with-vercel-and-github-actions-f80d3a4a7de3 (дата звернення: 15.05.2025).

37 Website Validator. Data·yze. [Електронний ресурс]. Режим доступу до ресурсу: https://datayze.com/site-validator (дата звернення: 15.05.2025).

38 Johnson E. Pros, Cons, Benefits & Features of BrowserStack – [2025] Overview. Test Automation Tools. [Електронний ресурс]. 08.05.2025. Режим доступу до ресурсу: https://testautomationtools.dev/pros-cons-benefits-features-of-browserstack-overview/ (дата звернення: 16.05.2025).

39 Дмитроца, Л. П., & Старицький, О. Т. (2024). Оптимізація продуктивності та SEO при масштабуванні проєктів на основі React і Next. js. Збірник тез доповідей XIII Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій “, 465-466.

40 Raykova L. Understanding Your Google PageSpeed Insights Report (2024 Guide). NitroPack. [Електронний ресурс]. 28.11.2025. Режим доступу до ресурсу: https://nitropack.io/blog/post/google-pagespeed-insights-beginner-guide (дата звернення: 16.05.2025).

41 Duda, O., Kochan, V., Kunanets, N., Matsiuk, O., Pasichnyk, V., Sachenko, A., & Pytlenko, T. (2019, September). Data processing in IoT for smart city systems. In 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) (Vol. 1, pp. 96-99). IEEE.

42 Державні санітарні правила і норми «Гігієнічні вимоги до умов праці при роботі з відеодисплейними терміналами» – ДСанПіН 3.3.2.007–98.

43 Левченко І.Л. Ергономічна оцінка умов праці оператора: навчальний посібник. – Львів: Видавництво ЛНУ, 2020. – 156 с.

44 Ткаченко О.П. Основи охорони праці: підручник. – Харків: ХНАДУ, 2021. – 312 с.

45 ДСТУ 8604:2015. Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги. – [Чинний від 01.07.2017]. – К.: ДП «УкрНДНЦ», 2017. – 10 с.

46 ДСН 3.3.6.042-99. Державні санітарні норми мікроклімату виробничих приміщень.

47 Сидоренко В.В. Психофізіологічні основи безпеки праці. – К.: Кондор, 2017. – 192 с.

48 Єрмоленко Л.А. Охорона праці: навч. посіб. – Дніпро: ДНУ, 2019. – 240 с.

49 Закон України «Про охорону праці» від 14.10.1992 № 2694-XII (із змінами).

50 Яворовський О. П., Шевцова В. М., Зенкіна В. І. Безпека життєдіяльності, основи охорони праці : навч. посіб. / за заг. ред. О. П. Яворовського. – К. : Медицина, 2015. – 288 с.

51 Мацюк В.І., Баранов В.В. Охорона праці: навч. посіб. – К.: Центр учбової літератури, 2020. – 348 с.

52 Гандзюк М.П., Желібо Є.П., Халімовський М.О. Основи охорони праці: Підручник. / За ред. М. П. Гандзюка. - К.: Каравела, 2008. - 384 с.

53 Пістун І.П. Безпека життєдіяльності (психофізіологічні аспекти). Практичні заняття. – Львів: Афіша, 2000. 240 с.

ДОДАТКИ

Додаток А

Тези VIII Міжнародної студентської науково - технічної конференції "ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VIII МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

24-25 квітня 2025 р.

(збірник тез конференції)

Тернопіль 2025

ББК 72+34 (Укр)
М34

Матеріали VIII Міжнародної студентської науково - технічної конференції / Тернопіль: Тернопільський національний технічний університет ім. І.Пулюя (м. Тернопіль, 24-25 квітня 2025 р.), 2025.- 391 с.

УДК 004.8
Тарас М. - ст. гр. СН-43
Тернопільський національний технічний університет імені Івана Пулюя

СУЧАСНА ВЕБРОЗРОБКА З ВИКОРИСТАННЯМ NEXT.JS, REACT ТА DJANGO REST

Науковий керівник: к.т.н, доцент Дмитроца Л.П.

Taras M.
Ternopil Ivan Puluj National Technical University

MODERN WEB DEVELOPMENT USING NEXT.JS, REACT AND DJANGO REST

Supervisor: PhD, Associate Professor Dmytrotsa L.P.

Ключові слова: технології, веброзробка, фреймворк
Keywords: technologies, web-development, framework

У світі цифрових технологій вебзастосунки стали не просто засобом комунікації чи презентації інформації – вони перетворилися на фундамент бізнесу, освіти, медицини та навіть особистої взаємодії. Зі зростанням попиту на швидкі, зручні та масштабовані рішення постає питання щодо вибору інструментів для створення сучасного вебсайту чи сервісу?

Розробники мають у своєму розпорядженні десятки фреймворків, бібліотек і платформ. Однак серед цього розмаїття окремо вирізняються три популярні технології: React, Next.js та Django REST Framework. Усі вони зарекомендували себе як потужні інструменти, але кожна має свою роль, підхід і сферу найкращого застосування.

Порівняння цих технологій ґрунтується не лише на технічних характеристиках, а й на практичних аспектах застосування в реальних проєктах. Зокрема, береться до уваги основні аспекти роботи, їх переваги та обмеження, а також взаємодія між собою для створення швидких, масштабованих та ефективних рішень.

React – це JavaScript-бібліотека, створена компанією Meta у 2013 році для побудови користувацьких інтерфейсів. Із моменту свого запуску React швидко набув популярності завдяки своїй простоті, гнучкості та активній спільноті [1]. У сучасній веброзробці він став майже стандартом для створення динамічних фронтенд-рішень.

Головна ідея React – компонентність. Усі частини інтерфейсу – кнопки, форми, списки – реалізуються як незалежні, багаторазово використовувані компоненти. Це значно спрощує розробку, масштабування та підтримку великих застосунків. Наприклад, якщо в застосунку є форма реєстрації, її можна створити як окремий компонент і надалі використовувати в кількох місцях без дублювання коду.

Однією з ключових інновацій React стала концепція віртуального DOM – внутрішнього уявлення сторінки, яке дозволяє мінімізувати кількість змін у реальному DOM браузера. Це покращує продуктивність застосунку, особливо на мобільних пристроях або при повільному інтернет-з'єднанні.

React орієнтований на побудову SPA – односторінкових застосунків, які працюють без повного перезавантаження сторінки. Користувач взаємодіє з додатком швидко, немовби перебуває в мобільному застосунку, тоді як нові дані підвантажуються динамічно у фоновому режимі.

Next.js – це фреймворк на базі React, створений компанією Vercel, який значно розширює можливості розробника у створенні сучасних вебзастосунків. Якщо React – це конструктор інтерфейсів, то Next.js – це повноцінний інструмент для розробки повнофункціональних застосунків, який охоплює як фронтенд, так і частину серверної логіки [2].

Однією з головних переваг Next.js є підтримка різних типів рендерингу [3]:

- SSR (Server-Side Rendering) – сторінки генеруються на сервері при кожному запиті, що покращує SEO та перше завантаження.
- SSG (Static Site Generation) – сторінки попередньо рендеряться під час збірки, що дає неймовірну швидкість на фронтенді.
- ISR (Incremental Static Regeneration) — гібридна модель, яка дозволяє оновлювати вже згенеровані сторінки частково, без повного перезбирання сайту.

Завдяки цим можливостям Next.js особливо добре підходить для сайтів з публічним контентом, блогів, маркетплейсів та платформ, де важливі як продуктивність, так і пошукова оптимізація.

На відміну від React, де роутинг налаштовується вручну, Next.js використовує автоматичну маршрутизацію, що базується на структурі файлів у папці pages. Це спрощує логіку навігації й економить час при розробці.

Next.js дозволяє створювати серверні API-ендпоінти прямо в межах проєкту, без необхідності окремого бекенду. Це зручно для невеликих сервісів, адмін-панелей або прототипів. Однак для масштабних рішень все ж доцільно використовувати повноцінний бекенд (наприклад, Django REST).

Next.js фактично є містком між фронтендом і сервером. Він дозволяє створювати гібридні застосунки, де деякі частини генеруються на сервері, інші – в браузері, а логіка бекенду може частково жити всередині самого проєкту. Але що робити, коли потрібен потужний, незалежний бекенд, який обробляє великі обсяги даних, має розвинену авторизацію, взаємодіє з базою даних і надає API? Тут у гру вступає Django REST Framework.

Django REST Framework (DRF) – це розширення до одного з найпопулярніших Python-фреймворків Django, яке дозволяє швидко й ефективно створювати RESTful API [4]. Якщо React і Next.js відповідають за інтерфейс та логіку на стороні користувача, то Django REST забезпечує серверну частину застосунку – взаємодію з базами даних, авторизацію, бізнес-логіку та надання даних у зручному форматі (наприклад, JSON).

Одна з головних переваг DRF – це використання зрілої, перевіреної часом архітектури Django. У ньому чітко визначено моделі, маршрутизацію, контролери (view) і шаблони, що дозволяє швидко стартувати з проєктом та дотримуватись зрозумілої структури.

DRF доповнює Django спеціалізованими класами для створення API: Serializer, APIView, ViewSet, Router та іншими. Це дозволяє гнучко керувати даними, перетворюючи об'єкти моделей у формат, придатний для фронтенду (наприклад, React/Next.js).

Django REST Framework має вбудовану підтримку:

- Аутентифікації через токени, сесії, OAuth2.
- Права доступу до ресурсів (permissions).
- Обмеження запитів (rate limiting).
- Документації API через Swagger або ReDoc.

Це робить його придатним для розробки великомасштабних та безпечних проєктів, які обробляють тисячі запитів щодня – від інтернет-магазинів до освітніх платформ і медичних систем.

Відмінності та переваги кожної з розглянутих технологій подані в таблиці 1.

Таблиця 1 – Порівняння ключових характеристик React, Next.js та Django REST

Критерій	React	Next.js	Django REST Framework
Тип технології	JavaScript-бібліотека	React-фреймворк (full-stack frontend)	Python-фреймворк для REST API
Роль у проєкті	Побудова інтерфейсу (UI)	Побудова повного фронтенду з SSR/SSG	Створення серверного API
Мова програмування	JavaScript / TypeScript	JavaScript / TypeScript	Python
Рендеринг	Клієнтський (CSR)	Клієнтський + серверний (CSR/SSR/SSG/ISR)	Немає інтерфейсу, лише API
SEO-оптимізація	Слабка	Висока завдяки SSR/SSG	Не застосовується
Маршрутизація (роутинг)	Вручну через React Router	Автоматично за структурою файлів	Через URLConf (ручна конфігурація)
Підтримка бекенду	Ні (потребує стороннього API)	Частково (через API-роути)	Так (повноцінний бекенд)
Підключення до БД	Ні	Частково (через ORM або API)	Так (ORM Django, SQL/NoSQL)
Простота старту	Висока	Середня	Середня/висока (для Python-розробників)
Крива навчання	Полога	Помірна	Помірна до крутої
Готовність до масштабування	Потребує налаштувань	Висока (оптимізації "з коробки")	Висока (чітка архітектура та безпека)
Ідеальний випадок застосування	SPA, UI-бібліотеки	SEO-дружні сайти, повні фронтенд-рішення	Побудова надійного API для мобільних та вебзастосунків

Кожна з технологій виконує свою роль у сучасному вебстеку:

- React – ідеальний для створення інтерфейсів і швидкої розробки компонентів.
- Next.js – потужний фреймворк для створення повноцінних фронтенд-застосунків із серверним рендерингом.
- Django REST – чудовий вибір для бекенду з акцентом на безпеку, структуру і масштабованість.

У сучасній веброзробці ефективність часто визначається не лише якістю окремих інструментів, а й тим, наскільки вдало вони поєднуються між собою. Саме тому комбінація React та Next.js на фронтенді та Django REST Framework на бекенді стала популярною архітектурною моделлю для створення продуктивних, масштабованих та зручних у підтримці вебзастосунків.

Одна з головних переваг такої зв'язки – можливість розділити застосунок на два незалежні шари:

- Фронтенд (React та Next.js) – відповідає за інтерфейс, взаємодію з користувачем, маршрутизацію, відображення даних.

- Бекенд (Django REST) – забезпечує логіку бізнес-процесів, обробку запитів, збереження даних, автентифікацію.

Це дозволяє різним командам працювати паралельно, використовувати переваги кожної технології, а також масштабувати або оновлювати окремі частини системи незалежно.

Обмін даними між фронтендом і бекендом відбувається через REST API, побудований за допомогою Django REST Framework. Фронтенд надсилає запити до серверу (наприклад, отримати список курсів або надіслати замовлення), а бекенд відповідає даними у форматі JSON.

Цей підхід має кілька ключових переваг:

- Універсальність – API може використовуватись не лише вебклієнтом, а й мобільними застосунками.
- Безпека – чіткий контроль доступу та автентифікації.
- Гнучкість – легко змінювати інтерфейс без змін у логіці сервера.

Приклад типового стеку:

- Frontend: Next.js + React, з можливим використанням Tailwind CSS або Material UI для стилізації.
- Backend: Django + Django REST Framework + PostgreSQL або MySQL.
- Інтеграція: Axios або Fetch API для HTTP-запитів; JWT або сесії для автентифікації; Stripe або PayPal для платіжних систем (при потребі).

Таке поєднання технологій дозволяє розробляти інтерактивні, швидкі, масштабовані вебзастосунки, де фронтенд забезпечує зручний користувацький досвід, а бекенд гарантує надійну обробку даних. Успішна інтеграція React/Next.js з Django REST – це вже не просто тренд, а ефективна архітектурна практика, яка відповідає вимогам сучасного ринку.

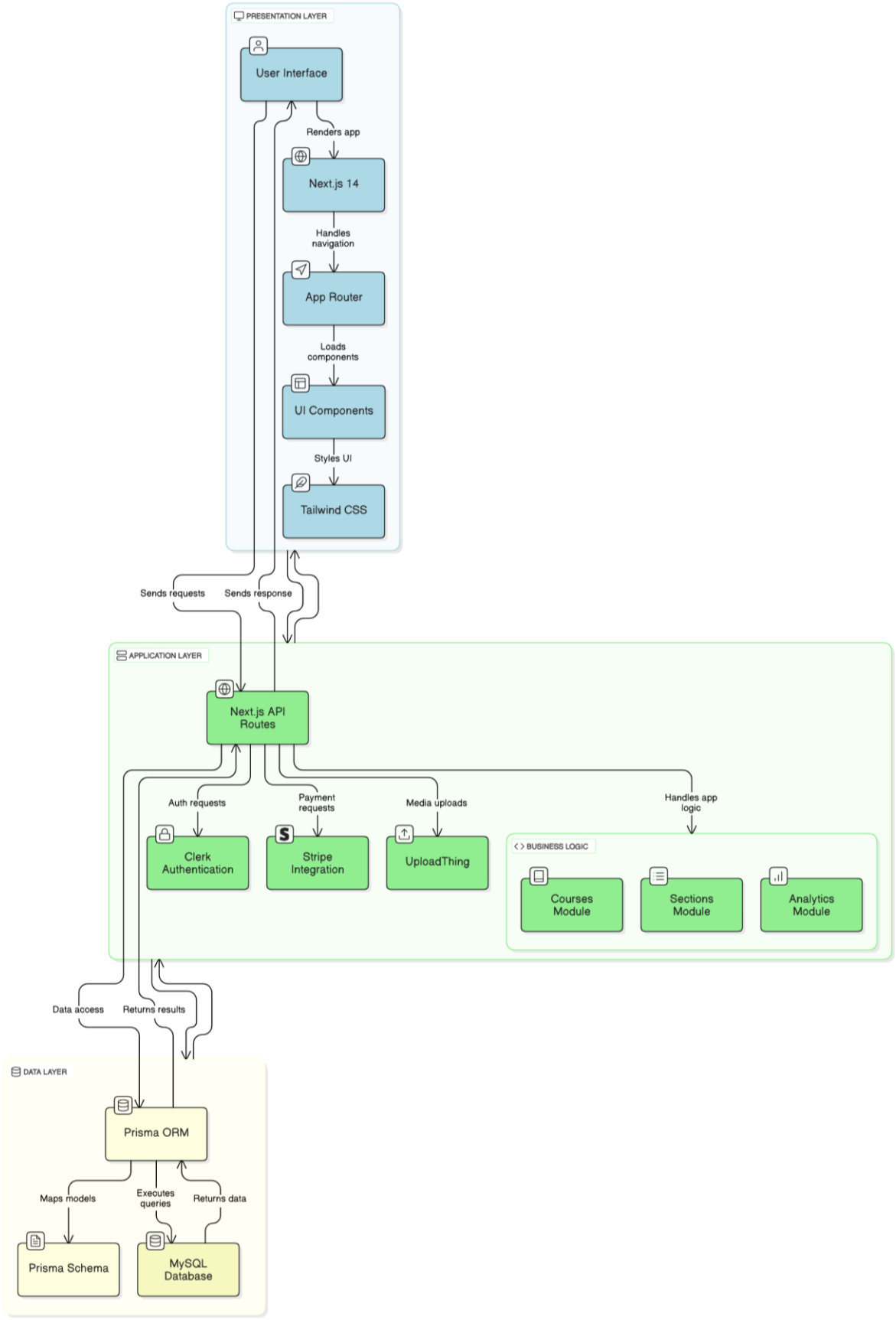
Висновок. Аналіз сучасних вебтехнологій React, Next.js та Django REST Framework показав, що їх поєднання створює потужний технологічний стек для розробки масштабованих, продуктивних та зручних у використанні вебзастосунків. React забезпечує гнучкість інтерфейсів, Next.js додає можливості серверного рендерингу та оптимізації, а Django REST виконує роль надійного бекенду з API-доступом до даних. Така інтеграція дозволяє ефективно реалізовувати як прості, так і комплексні проєкти, відповідаючи вимогам сучасного цифрового середовища.

Література

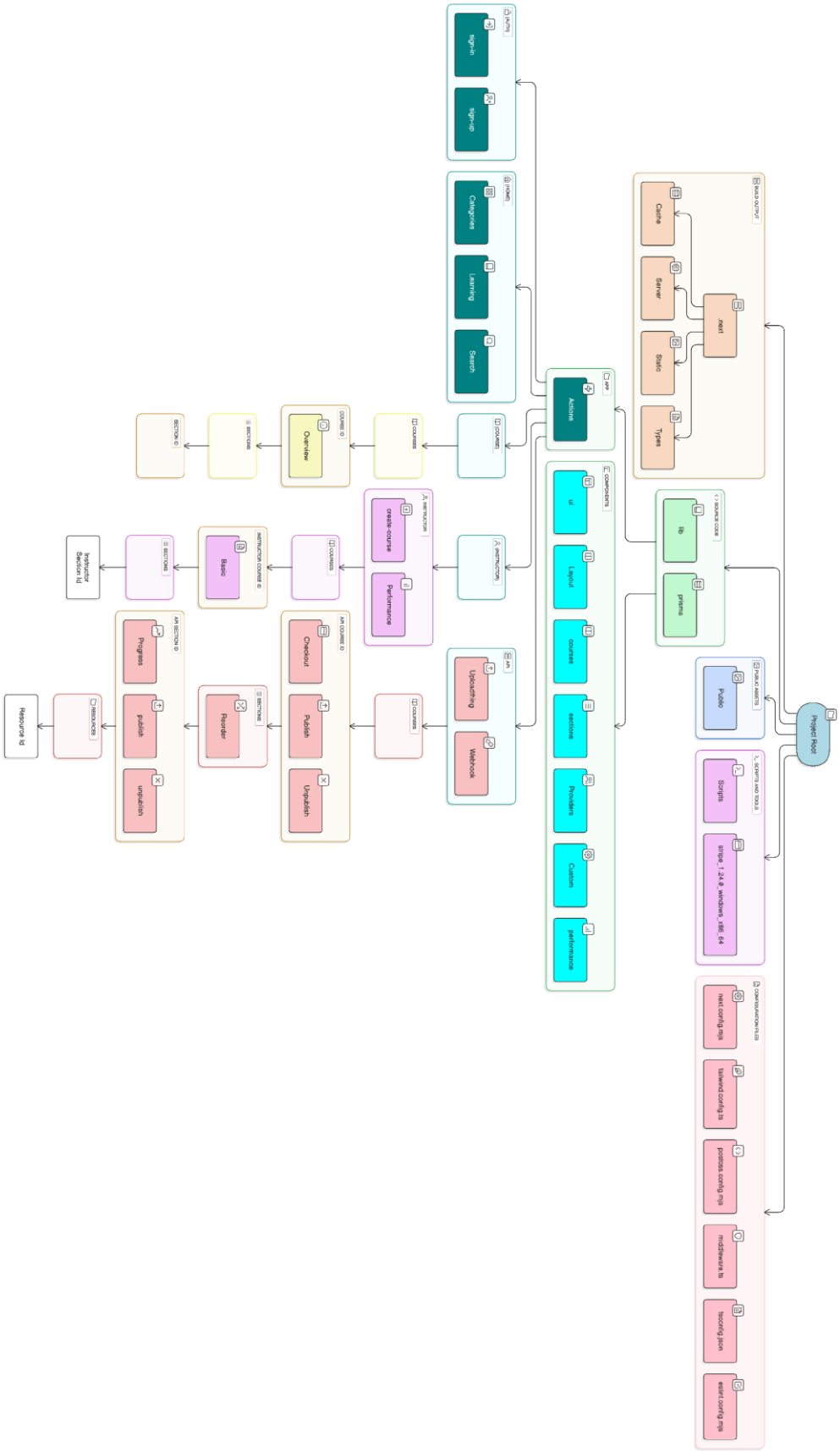
1. 9 Reasons Why React is Still Popular in 2024. KOMODO. URL: https://www.komododigital.co.uk/insights/9-reasons-why-react-is-still-popular-in-2021/ (дата звернення: 04.04.2025).
2. Bhati A. Next JS vs React: Which Framework Should You Choose For Web Development?. TRooInbound. 10.06.2024. URL: https://www.trooinbound.com/blog/next-js-vs-react/ (дата звернення: 08.04.2025).
3. Aditya Kumar Tiwari. Understanding CSR, SSR, SSG, and ISR: A Next.js Perspective. Medium. 10.07.2023. URL: https://medium.com/design-bootcamp/understanding-csr-ssr-ssg-and-isr-a-next-js-perspective-fcaf36686de6 (дата звернення: 10.04.2025).
4. Junaidul Islam. The Perfect Pair: Why Next.js and Django REST Framework are Ideal for Full-Stack Web Development. DEV. 04.09.2024. URL: https://dev.to/junaaid96/the-perfect-pair-why-nextjs-and-django-rest-framework-are-ideal-for-full-stack-web-development-4a02 (дата звернення: 10.04.2025).

Додаток Б

Структурна схема трирівневої архітектури платформи Learnify



Додаток В


Project Root
.next
Cache
Server
Static
Types
lib
prisma
Public
Scripts
stripe_1.24.8_windows_x86_64
next.config.mjs
tailwind.config.ts
postcss.config.mjs
middleware.ts
tsconfig.json
eslint.config.mjs
app
Actions
components
ui
Layout
courses
sections
Providers
Custom
performance
(auth)
sign-in
sign-up
(main)
Categories
Learning
Search
(dashboard)
courses
instructor
create-course
Performance
courses
Basic
sections
Instructor Section Id
courses
Overview
sections
Section Id
api
Uploadthing
Webhook
courses
Checkout
Publish
Unpublish
sections
Reorder
courses
Progress
publish
unpublish
resources
Resource Id

Вміст файлу schema.prisma

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider     = "mysql"
  url          = env("DATABASE_URL")
  relationMode = "prisma"
}

model Course {
  id           String  @id @default(uuid())
  instructorId String
  title        String  @db.Text
  subtitle     String? @db.Text
  description  String? @db.Text
  imageUrl     String? @db.Text
  price        Float?
  isPublished  Boolean @default(false)

  categoryId String
  category   Category @relation(fields: [categoryId], references:
[id])

  subCategoryId String
  subCategory   SubCategory @relation(fields: [subCategoryId],
references: [id])

  levelId String?
  level   Level?  @relation(fields: [levelId], references: [id])

  sections Section[]

  purchases Purchase[]

  createdAt DateTime @default(now())
  updatedAt DateTime @default(now())

  @@index([categoryId])
  @@index([subCategoryId])
  @@index([levelId])
}

model Category {
  id            String        @id @default(uuid())
  name          String        @unique
  subCategories SubCategory[]
  courses       Course[]
}
model SubCategory {
  id   String @id @default(uuid())
  name String

  categoryId String
```

```
  category   Category @relation(fields: [categoryId], references:
[id])

  courses Course[]

  @@index([categoryId])
}

model Level {
  id      String   @id @default(uuid())
  name    String   @unique
  courses Course[]
}

model Section {
  id          String  @id @default(uuid())
  title       String
  description String? @db.Text
  videoUrl    String? @db.Text
  position    Int
  isPublished Boolean @default(false)
  isFree      Boolean @default(false)

  courseId String
  course   Course @relation(fields: [courseId], references: [id],
onDelete: Cascade)

  muxData MuxData?

  resources Resource[]
  progress Progress[]

  createdAt DateTime @default(now())
  updatedAt DateTime @default(now())

  @@index([courseId])
}

model MuxData {
  id         String  @id @default(uuid())
  assetId    String
  playbackId String?

  sectionId String  @unique
  section   Section @relation(fields: [sectionId], references:
[id], onDelete: Cascade)
  @@index([sectionId])
}
model Resource {
  id      String @id @default(uuid())
  name    String
  fileUrl String

  sectionId String
  section   Section @relation(fields: [sectionId], references:
[id], onDelete: Cascade)
```

```
  createdAt DateTime @default(now())
  updatedAt DateTime @default(now())

  @@index([sectionId])
}

model Progress {
  id        String @id @default(uuid())
  studentId String

  sectionId String
  section   Section @relation(fields: [sectionId], references:
[id], onDelete: Cascade)
  isCompleted Boolean @default(false)

  createdAt DateTime @default(now())
  updatedAt DateTime @default(now())

  @@index([sectionId])
  @@unique([studentId, sectionId])
}

model Purchase {
  id         String @id @default(uuid())
  customerId String

  courseId String
  course   Course @relation(fields: [courseId], references: [id],
onDelete: Cascade)

  createdAt DateTime @default(now())
  updatedAt DateTime @default(now())

  @@index([courseId])
  @@unique([customerId, courseId])
}

model StripeCustomer {
  id               String @id @default(uuid())
  customerId       String @unique
  stripeCustomerId String @unique
  createdAt DateTime @default(now())
  updatedAt DateTime @default(now())
}
```

Вміст файлу tailwind.config.js

```
import type { Config } from "tailwindcss";

export default {
    darkMode: ["class"],
    content: [
    "./pages/**/*.{js,ts,jsx,tsx,mdx}",
    "./components/**/*.{js,ts,jsx,tsx,mdx}",
    "./app/**/*.{js,ts,jsx,tsx,mdx}",
    './src/**/*.{ts,tsx}',
  ],
  theme: {
    container: {
        center: true,
        padding: '2rem',
        screens: {
            '2xl': '1400px'
        }
    },
    extend: {
        colors: {
            background: 'hsl(var(--background))',
            foreground: 'hsl(var(--foreground))',
            card: {
                DEFAULT: 'hsl(var(--card))',
                foreground: 'hsl(var(--card-foreground))'
            },
            popover: {
                DEFAULT: 'hsl(var(--popover))',
                foreground: 'hsl(var(--popover-foreground))'
            },
            primary: {
                DEFAULT: 'hsl(var(--primary))',
                foreground: 'hsl(var(--primary-foreground))'
            },
            secondary: {
                DEFAULT: 'hsl(var(--secondary))',
                foreground: 'hsl(var(--secondary-foreground))'
            },
            muted: {
                DEFAULT: 'hsl(var(--muted))',
                foreground: 'hsl(var(--muted-foreground))'
            },
            accent: {
                DEFAULT: 'hsl(var(--accent))',
                foreground: 'hsl(var(--accent-foreground))'
            },
            destructive: {
                DEFAULT: 'hsl(var(--destructive))',
                foreground: 'hsl(var(--destructive-foreground))'
            },
            border: 'hsl(var(--border))',
            input: 'hsl(var(--input))',
            ring: 'hsl(var(--ring))',
            chart: {
```

```
                '1': 'hsl(var(--chart-1))',
                '2': 'hsl(var(--chart-2))',
                '3': 'hsl(var(--chart-3))',
                '4': 'hsl(var(--chart-4))',
                '5': 'hsl(var(--chart-5))'
            }
        },
        borderRadius: {
            lg: 'var(--radius)',
            md: 'calc(var(--radius) - 2px)',
            sm: 'calc(var(--radius) - 4px)'
        }
    }
  },
  plugins: [require("tailwindcss-animate")],
} satisfies Config;
```