

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## **ПОЯСНЮВАЛЬНА ЗАПИСКА**

до кваліфікаційної роботи освітнього ступеня «бакалавр»  
за спеціальністю 121 «Інженерія програмного забезпечення»  
(освітня програма «Інженерія програмного забезпечення»)  
на тему:

**«Веб-додаток онлайн курсів»**

Виконав студент групи ІПЗ-20-3  
ШЕВЧУК Олександр Сергійович

Керівник роботи:  
КОНЦИДАЙЛО Андрій Михайлович

Рецензент:  
МОРОЗОВ Андрій Васильович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**«ЗАТВЕРДЖУЮ»**

Зав. кафедри інженерії  
програмного забезпечення

Тетяна ВАКАЛЮК

**«26» лютого 2024 р.**

**ЗАВДАННЯ**  
**на кваліфікаційну роботу**

Здобувач вищої освіти: **ШЕВЧУК Олександр Сергійович**

Керівник роботи: **КОНЦИДАЙЛО Андрій Михайлович**

Тема роботи: «Веб-додаток онлайн курсів»,

затверджена Наказом закладу вищої освіти від «23» лютого 2024 р., №74/с

Вихідні дані для роботи: **розробка веб-додатку онлайн курсів, можливості використання програмного забезпечення для підтримки онлайн-курсів в навчальних цілях.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

| Розділ | Консультант      | Завдання видав | Завдання прийняв |
|--------|------------------|----------------|------------------|
| 1      | Концидайло А. М. | 27.02.2024р.   | 27.02.2024р.     |
| 2      | Концидайло А. М. | 27.02.2024р.   | 27.02.2024р.     |
| 3      | Концидайло А. М. | 27.02.2024р.   | 27.02.2024р.     |

## РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Веб-додаток онлайн курсів» складається із вступу, трьох розділів, загальних висновків, списку бібліографічних посилань і містить 84 сторінок, 59 рисунків та 4 таблиці. Список бібліографічних посилань містить 42 найменування.

Метою роботи є розробка веб-додатку для онлайн курсів, використовуючи web-технології.

Методи дослідження: об'єктно-орієнтоване моделювання, розробка фронт-енд частини, розробка бек-енд частини, розробка та підключення бази даних.

У роботі визначено основні завдання на розробку системи, проведено аналіз вимог та функціональності додатку, також був проведений аналіз аналогів створеного програмного продукту. Було здійснено проектування бази даних та обрано набір сучасних технологій розробки, використання яких забезпечує високу продуктивність додатку.

**КЛЮЧОВІ СЛОВА:** ВЕБ-ДОДАТОК, МОДЕЛЮВАННЯ, АРХІТЕКТУРА, NEXT.JS, HTML, TAILWIND CSS, REACT.JS, MONGODB.

|           |      |                 |        |      |                          |                                            |      |         |
|-----------|------|-----------------|--------|------|--------------------------|--------------------------------------------|------|---------|
|           |      |                 |        |      | ІПЗ.КР.Б-121-24-ІПЗ      |                                            |      |         |
| Змн.      | Арк. | № докум.        | Підпис | Дата | Вебдодаток онлайн курсів | Літ.                                       | Арк. | Аркушів |
| Розроб.   |      | Шевчук О.С.     |        |      |                          |                                            | 3    |         |
| Керівник  |      | Концидайло А.М. |        |      |                          | Житомирська політехніка,<br>група ІПЗ-20-3 |      |         |
| Керівник  |      |                 |        |      |                          |                                            |      |         |
| Н. контр. |      | Тичина О.П.     |        |      |                          |                                            |      |         |
| Зав. каф. |      | Вакалюк Т.А.    |        |      |                          |                                            |      |         |

## ABSTRACT

The explanatory note to the diploma thesis "Web Application for Online Courses" consists of an introduction, three sections, general conclusions, a list of bibliographic references, and contains 84 pages, 59 figures, and 4 tables. The list of bibliographic references includes 42 titles.

The aim of the thesis is to develop a web application for online courses using web technologies.

Research methods: object-oriented modeling, front-end development, back-end development, database design, and integration.

The thesis defines the main tasks for the system development, analyzes the requirements and functionality of the application, and includes an analysis of similar existing software products. Database design was carried out, and a set of modern development technologies was selected to ensure high performance of the application.

KEY WORDS: WEB APPLICATION, MODELING, ARCHITECTURE, NEXT.JS, HTML, TAILWIND CSS, REACT.JS, MONGODB.

## ЗМІСТ

|                                                                                                 |    |
|-------------------------------------------------------------------------------------------------|----|
| РЕФЕРАТ .....                                                                                   | 3  |
| ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....                                                                  | 6  |
| ВСТУП .....                                                                                     | 7  |
| РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБДОДАТКУ<br>ОНЛАЙН КУРСІВ .....                       | 9  |
| 1.1 Постановка задачі .....                                                                     | 9  |
| 1.2 Аналіз аналогів програмного продукту .....                                                  | 10 |
| 1.3 Обґрунтування вибору інструментальних засобів та вимоги до<br>апаратного забезпечення ..... | 16 |
| Висновки до розділу 1 .....                                                                     | 21 |
| РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ОНЛАЙН КУРСІВ.....                                           | 23 |
| 2.1 Визначення варіантів використання та об'єктно-орієнтованої структури<br>.....               | 23 |
| 2.2 Опис структури даних з якими працює веб-додаток .....                                       | 30 |
| 2.3 Проектування та реалізація алгоритмів роботи системи .....                                  | 32 |
| 2.4 Реалізація функціоналу веб-додатку онлайн курсів.....                                       | 42 |
| Висновки до розділу 2 .....                                                                     | 50 |
| РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ВЕБ-ДОДАТКОМ ОНЛАЙН<br>КУРСІВ.....                        | 52 |
| 3.1 Порядок встановлення та налаштування системи.....                                           | 52 |
| 3.2 Структура інтерфейсу веб-орієнтованої системою онлайн курсів.....                           | 53 |
| 3.3 Тестування роботи веб-додатку.....                                                          | 68 |
| Висновки до третього розділу .....                                                              | 69 |
| ВИСНОВКИ.....                                                                                   | 71 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....                                                                 | 72 |
| ДОДАТКИ.....                                                                                    | 76 |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

|  |  |  |  |  |                    |      |
|--|--|--|--|--|--------------------|------|
|  |  |  |  |  | ІПЗ.КР.Б-121-24-ПЗ | Арк. |
|  |  |  |  |  |                    |      |
|  |  |  |  |  |                    |      |
|  |  |  |  |  |                    | 6    |

## ВСТУП

В сучасному світі зростає значення онлайн-курсів як засобу навчання та розвитку. З метою вдосконалення та оптимізації процесу навчання виникає потреба у створенні ефективних інформаційних та функціональних систем підтримки таких курсів. Використовуючи різні методи, підходи та інструменти, починаючи від простих та безкоштовних, до повноцінних платних рішень, кожен з яких має свої переваги, але спільною проблемою залишається збільшення обсягу інформації, що потребує уваги. Традиційні підходи до навчання не завжди відповідають потребам сучасного ринку праці та технологічного розвитку, тому важливо розробляти нові методи, які структурують матеріал ефективно та збільшують мотивацію студентів.

**Актуальність теми** зумовлена необхідністю вдосконалення процесу навчання в сучасних умовах, коли все більше людей прагнуть отримувати нові знання та навички для особистісного та професійного розвитку. Результатом дослідження є розробка веб-додатку, який реалізує новий метод навчання з використанням сучасних веб-технологій. Додаток передбачає адаптивність для мобільних пристроїв, та для інших платформ, що робить його доступним для широкого кола користувачів. Ключова особливість розробленого веб-додатку є реалізація функціональності, яка дозволяє користувачам зберігати прогрес навчання, та пошук будь яких курсів по їх вподабанню. Таким чином, розроблений веб-додаток, що реалізує новий метод навчання з використанням сучасних веб-технологій, є актуальним рішенням для вдосконалення процесу навчання в сучасних умовах.

**Метою** бакалаврської роботи є всебічне дослідження особливостей моделювання та аналізу програмних комплексів. Крім того, робота зосереджується на розробці ефективної методики навчання із застосуванням сучасних веб-технологій. Це включає аналіз існуючих методів, визначення їх сильних і слабких сторін, а також створення нових підходів для підвищення ефективності навчального процесу.

Для досягнення поставленої мети в роботі необхідно вирішити наступні завдання:

- Аналіз теоретичних засад моделювання програмного забезпечення.
- Аналіз та опис вимог користування системою.
- Методи моделювання функцій та поведінки системи.
- Проектування об'єктної структури системи.
- Фізичне моделювання програмних комплексів.
- Обґрунтування використання підходів для вирішення проблеми ефективного навчання.
- Виконати проектування web-додатку із визначеними технологіями

Грамотне впровадження цього методу дозволить багатьом охочим ознайомлюватись з новими знаннями або поглибити їх. Важливою особливістю розробленого методу навчання є його орієнтація на підготовку до працевлаштування. Це означає, що крім вивчення теоретичних основ та практичних навичок програмування, всі охочі також отримають знання та навички, необхідні для складання валідного резюме для успішного проходження співбесіди. Грамотне впровадження цього методу дозволить багатьом засвоїти основні знання для подальшої кар'єри в різних напрямках.

**Об'єкт дослідження:** розробка системи онлайн курсів.

**Предмет дослідження:** можливості використання програмного забезпечення для підтримки онлайн-курсів в навчальних цілях.

# РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБДОДАТКУ ОНЛАЙН КУРСІВ

## 1.1 Постановка задачі

Розвиток сфери інформаційних технологій та зростаючий попит на дистанційне навчання породили потребу у створенні ефективних та доступних засобів для освіти. Один із таких засобів - веб-додатки для онлайн-курсів, які дозволяють студентам вивчати різні предмети незалежно від їхнього місцезнаходження та графіку. Основною метою цього проекту є створення інформаційної та функціональної системи, яка забезпечить ефективну підтримку онлайн-курсів. Система повинна спрощувати процес навчання, забезпечувати доступ до необхідних матеріалів і інструментів для користувачів, а також покращувати взаємодію між учасниками навчального процесу.

Ця система надасть можливість проводити онлайн-курси з різних предметних областей, забезпечуючи доступ до відеолекцій, навчальних матеріалів, завдань для самостійного вивчення.

Основні функціональні вимоги для даного проекту:

1. Реєстрація користувачів і авторизація доступу до системи.
2. Перегляд доступних курсів та їхніх описів.
3. Відтворення відео-уроків та доступ до навчальних матеріалів.
4. Збір та аналіз статистики успішності користувачів.
5. Забезпечення адміністративного керування системою для адміністраторів курсів та модераторів.

Основними технічними вимогами даного проекту :

1. Мова програмування та технології: Використання сучасних мов програмування та технологій, таких як JavaScript (зокрема Next.js для серверної частини та React.js для клієнтської частини), для розробки веб-додатку.

2. Користувацький інтерфейс: Розробка зручного та інтуїтивно зрозумілого користувацького інтерфейсу з використанням сучасних методів
3. Безпека: Забезпечення високого рівня безпеки даних та конфіденційності користувачів шляхом використання шифрування даних, механізмів аутентифікації та авторизації.
4. Масштабованість: Забезпечення можливості масштабування системи для впровадження нових функцій, підтримки великої кількості користувачів та оптимізації продуктивності системи.
5. Інтеграція: Можливість інтеграції з іншими сервісами та платформами, такими як платіжні системи для оплати курсів, засоби аналізу даних для вивчення показників успішності курсів та інші.
6. Тестування: Використання автоматизованих тестів для перевірки функціональності системи та виявлення помилок на ранніх етапах розробки.

Ці основні функціональні вимоги є ключовими для успішного розроблення та впровадження інформаційної та функціональної системи підтримки онлайн-курсів, що забезпечить надійність, зручність та безпеку використання користувачами.

## 1.2 Аналіз аналогів програмного продукту

Вступ до аналізу аналогів програмного продукту є важливим етапом у процесі розробки та вдосконалення програмного забезпечення. Цей аналіз передбачає оцінку існуючих рішень на ринку з метою з'ясування їхніх функціональних можливостей та особливостей. Ретельне вивчення конкурентів допомагає виявити сильні та слабкі сторони їхніх продуктів, а також знайти простір для вдосконалення та впровадження нововведень у власну систему.

Відмінності між системами визначаються, в основному, не способом їх реалізації та логічною моделлю бази даних, а користувацьким інтерфейсом.

Набуті знання під час аналізу аналогів допоможуть нам краще розуміти динаміку ринку та вибрати стратегію подальшого розвитку нашого продукту.

- edX
- Skillshare
- Masterclass
- LinkedIn Learning

1.1. Udemy – є однією з найпопулярніших інтерактивних онлайн-платформ для навчання онлайн. Платформа була заснована у 2010 році і з тих пір здобула велику популярність серед користувачів з усього світу, вона пропонує широкий спектр курсів на різні теми, що робить його чудовим вибором для людей з різними інтересами та цілями навчання. Платформа має понад 130 000 курсів, створених понад 65000 інструкторів. Udemy користується популярністю як серед новачків, так і досвідчених фахівців, які прагнуть вдосконалити свої навички або вивчити нове (рис. 1.1)[1].

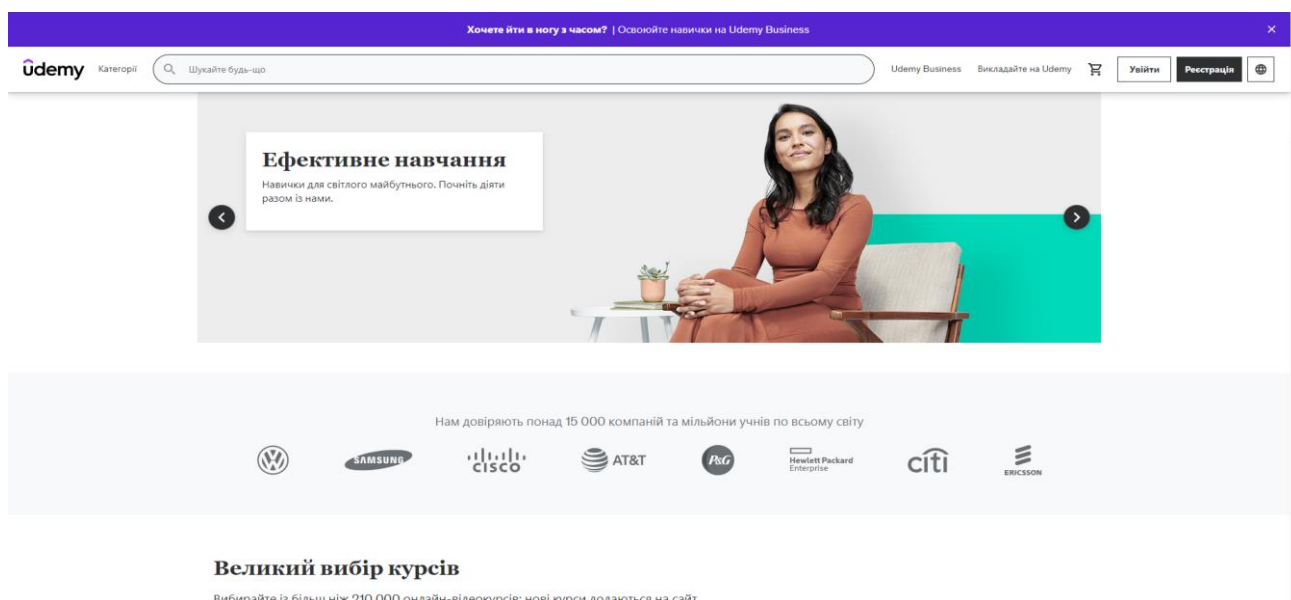


Рис. 1.1. Головна сторінка онлайн-платформи Udeemy[1]

Для доступу до повноцінного функціоналу платформи користувач повинен мати власний профіль, який створюється за декілька простих кроків.

Платформа Udemу має зручний та приємний інтерфейс де викладачі можуть додавати різні види контенту, включаючи відео-лекції, тексти,

завдання та вікторини. Це забезпечує інтерактивність навчального процесу, що сприяє кращому засвоєнню матеріалу. Також існує система досягнень виконання завдань та індикатор прогресу проходження курсу, користувачі мають змогу проходити курси у зручний для них час, що забезпечує високу гнучкість навчання для людей з насиченим графіком, залишати коментарі та задавати питання, що дозволяє отримувати зворотній зв'язок між авторами курсів та звичайними користувачами. Платформа дозволяє користувачам отримувати сертифікати після завершення їх навчання, для підтвердження їхніх знань та навичок, що є важливим мотиватором для підвищення цінності вивченого матеріалу. Присутні також, моделі монетизації, включаючи продаж окремих курсів та підписку на навчальні пакети, що дозволяє авторам заробляти на своїх знаннях. В загальному цей додаток орієнтований на широкий спектр аудиторії [1].

1.2 Coursera – онлайн платформа, яка була заснована у 2012 році професорами Стенфордського університету, Дафною Коллер та Ендрю Нгом, з метою надання доступу до різноманітних високоякісних навчальних матеріалів на різні теми, включаючи науку, технології, гуманітарні науки, інженерія, біологія, медицина та суспільствознавство, бізнес, інформатика, математика, цифровий маркетинг, комп'ютерні науки багато інших[2].

На Coursera зареєстровано понад 33 мільйони користувачів, а до 2024 року кількість доступних курсів зросла приблизно до 7000 з різних дисциплін (рис. 1.2)[2].

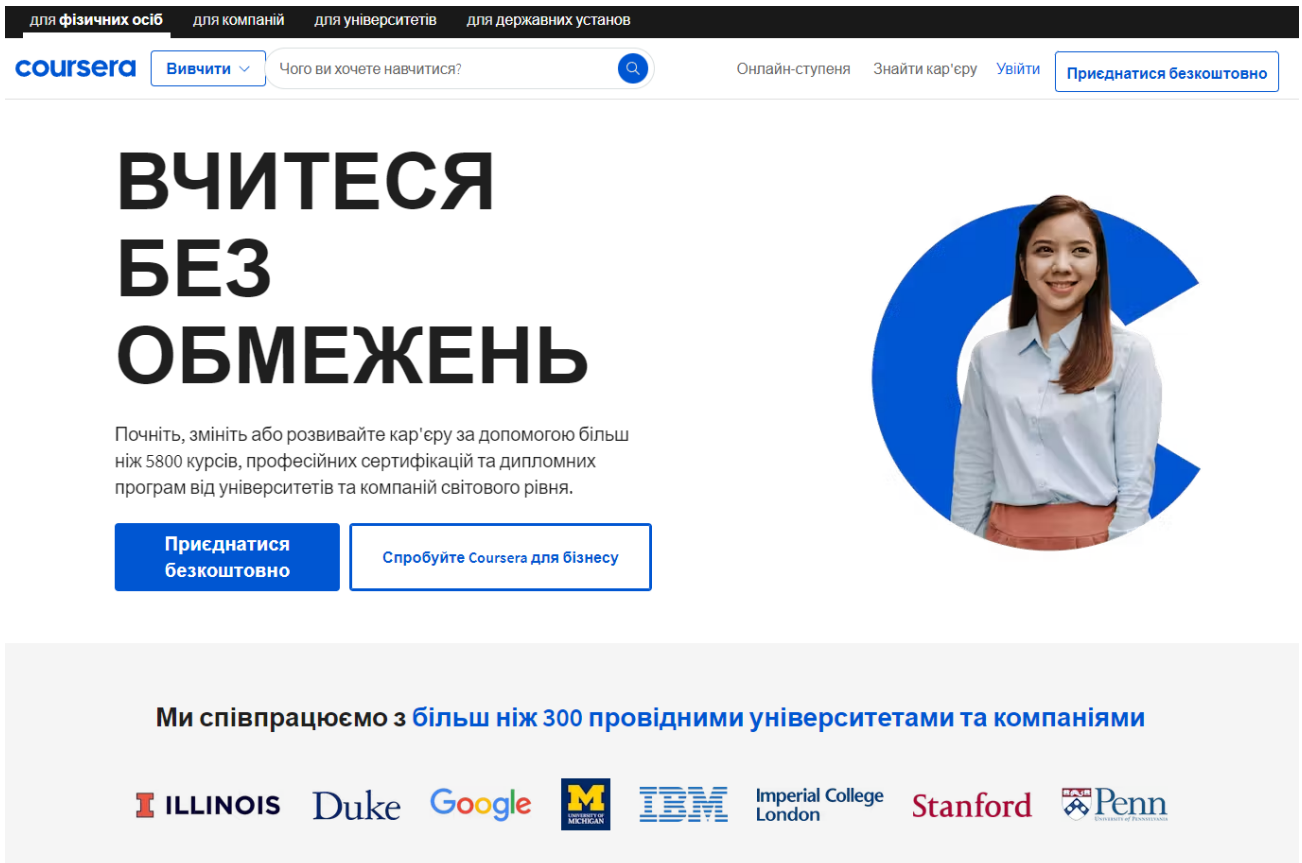


Рис. 1.2. Головна сторінка ресурсу Coursera[2]

Однією з ключових переваг Coursera є її співпраця з провідними університетами та компаніями, такими як Stanford, Yale, Google та IBM. Це забезпечує високу якість навчальних матеріалів та їх актуальність, також розширення ресурсу включають спеціалізації - набори курсів, які формують навички з певної теми, а також кваліфікації та продукти розвитку праці для організацій, бізнесу та уряду [2].

Під час початкового раунду фінансування стартапу було зібрано понад 16 мільйонів доларів за підтримки Kleiner Perkins Caufield & Byers і New Enterprise Associates. Станом на 2017 рік Coursera пропонує повний ступінь магістра [2].

1.3 edX – це платформа онлайн-навчання від провідної світової компанії цифрової освіти 2U, Inc. edX було засновано у травні 2012 року Гарвардом і Массачусетським технологічним інститутом як експеримент, щоб зробити найкращу у світі освіту доступною для всіх. Сьогодні, будучи частиною 2U, edX об'єднує понад 86 мільйонів людей у всьому світі з онлайн-навчанням, більш ніж 4600 онлайн програм та 260 контент партнерів, яке забезпечує

реальний професійний прогрес майже в усіх сферах кар'єри, від штучного інтелекту та робототехніки до сталого розвитку та охорони здоров'я. Разом з університетами та організаціями, які займають передові позиції у своїх галузях, edX пропонує тисячі професійних програм, розроблених, щоб дати кожному амбітному учневі шлях до досягнення. Курси часто включають різноманітні інтерактивні елементи, такі як відеолекції, інтерактивні завдання, тести, форуми для обговорення та вікторини. Ці елементи сприяють кращому засвоєнню матеріалу та взаємодії між студентами та викладачами (рис. 1.3) [3].

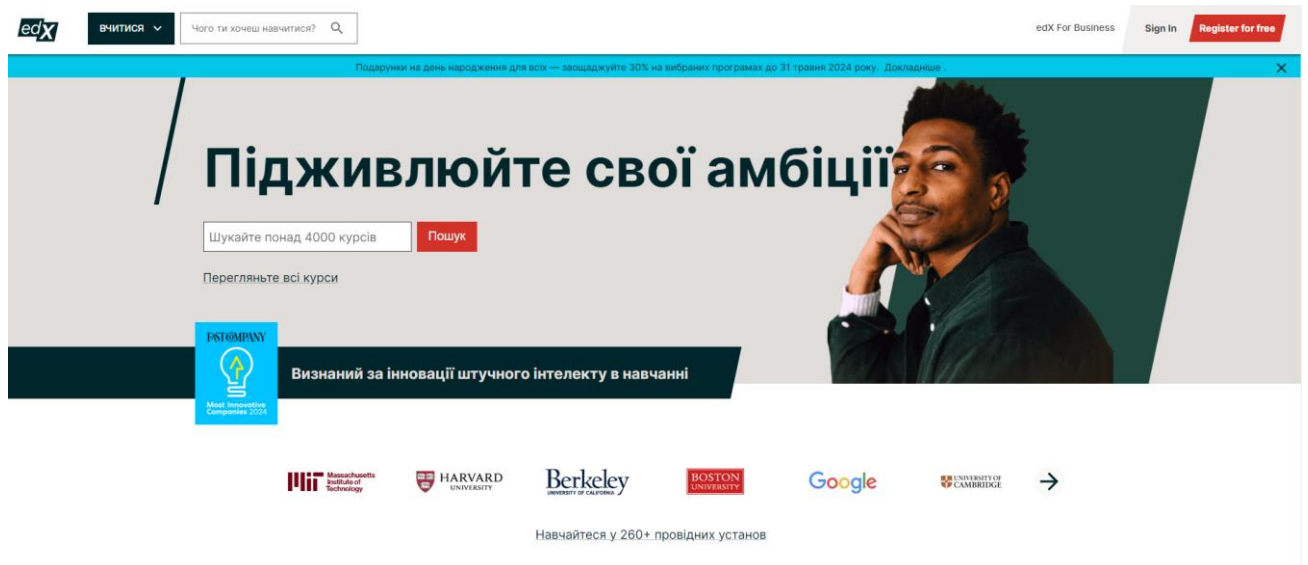


Рис. 1.3. Інтернет платформа EdX [3]

2U вірить у силу високоякісної онлайн-освіти для створення кращого майбутнього для всіх. Завдяки тому що edX доступний як додаток для iOS та Android, учні будь-де мають доступ до освітнього досвіду під керівництвом експертів, гнучких шляхів до особистих і професійних досягнень, освітню можливість на будь-якому інвестиційному рівні. У 2021 році 2U придбала edX, щоб створити глобальну платформу онлайн-навчання, яка надала мільйонам людей у всьому світі навчальний досвід, що змінить життя. А за допомогою edX Enterprise сотні провідних компаній використовують платформу, щоб впливати на всі рівні своєї організації. А досягнення штучного інтелекту в освіті, завдяки яким edX увійшов до списку «Найінноваційніших» Fast Company, є яскравими прикладами прагнення стимулювати кожну кар'єрну

|  |  |  |  |  |                   |      |
|--|--|--|--|--|-------------------|------|
|  |  |  |  |  | ИЗ.КР.Б-121-24-ИЗ | Арк. |
|  |  |  |  |  |                   | 14   |
|  |  |  |  |  |                   |      |

подорож. Також EdX-це некомерційна організація, яка працює на безкоштовній платформі OpenX з відкритим кодом OpenX[3].

1.4 LinkedIn Learning — це одна з провідних платформ для онлайн-навчання, яка стала частиною LinkedIn після придбання Lynda.com у 2015 році. Станом на 2021 LinkedIn Learning пропонує понад 16 000 курсів більш ніж на 7 мовах, які діляться на чотири категорії: бізнес, творчість, технології та сертифікати (рис. 1.4).

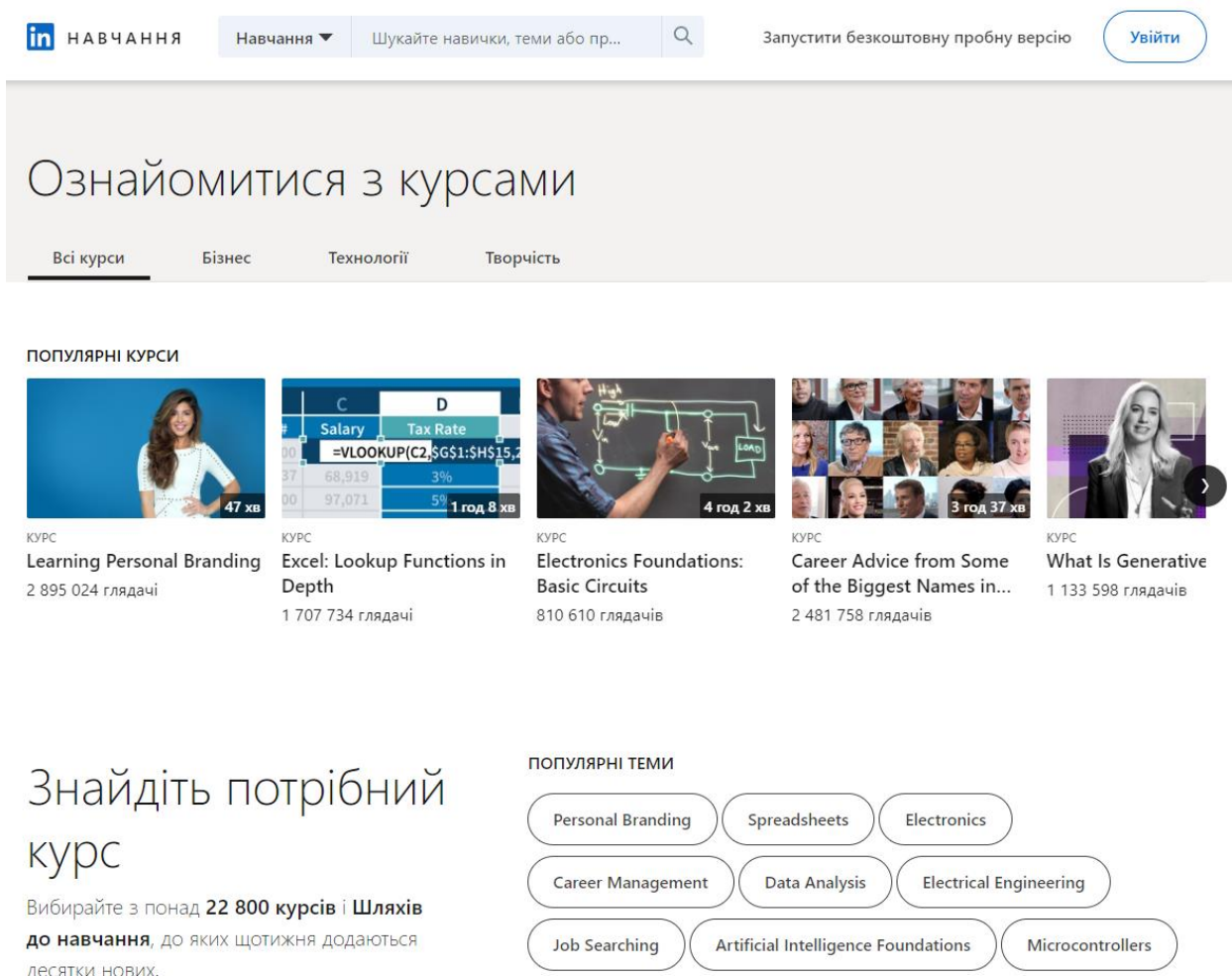


Рис. 1.4. Головна сторінка онлайн-платформи LinkedIn Learning[4]

Однією з ключових конкурентних переваг LinkedIn Learning є її тісна інтеграція з професійною соціальною мережею LinkedIn. Ця особливість дає змогу користувачам легко та зручно додавати завершені курси до своїх профілів на LinkedIn, тим самим підвищуючи видимість своїх навичок та досягнень для потенційних роботодавців та колег. Крім того, платформа LinkedIn Learning використовує дані з профілів користувачів на LinkedIn для персоналізації рекомендацій курсів. Алгоритми аналізують інформацію про

освіту, досвід роботи, навички та інтереси користувача, щоб запропонувати найбільш релевантні та корисні для нього навчальні матеріали. Це значно спрощує процес пошуку потрібних курсів та дозволяє сфокусуватись на розвитку саме тих навичок, які є актуальними для кар'єрного зростання. LinkedIn Learning пропонує як індивідуальну підписку, так і корпоративні рішення для команд та організацій. Це дозволяє компаніям забезпечувати безперервний професійний розвиток своїх співробітників. Таким чином, інтеграція з LinkedIn та персоналізація рекомендацій роблять LinkedIn Learning зручним та ефективним інструментом для професійного розвитку. Користувачі можуть не лише отримувати нові знання, а й демонструвати свої досягнення в навчанні безпосередньо на своїх профілях в провідній професійній мережі. Проте, для того щоб залишатися на передовій онлайн-освіти, платформа повинна продовжувати розширювати свою базу курсів та знижувати вартість підписки для більшої доступності[4].

Таблиця 1.1

Таблиця порівняння навчальних платформ, що досліджуються

| Характеристика                    | Udemi | Coursera | edX | LinkedIn Learning |
|-----------------------------------|-------|----------|-----|-------------------|
| Теорія                            | +     | +        | +   | +                 |
| Сертифікування                    | +     | +        | +   | +                 |
| Тестування                        | +     | +        | +   | +                 |
| Офлайн робота                     | +     | +        | +   | +                 |
| База даних                        | -     | +        | +   | +                 |
| Мобільність                       | +     | +        | +   | +                 |
| Корпоративні рішення              | -     | +        | +   | +                 |
| Інтеграція з соціальними мережами | -     | -        | -   | +                 |
| Інтерактивність                   | +     | +        | +   | +                 |

### 1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

**Next.js:** Це фреймворк для React, який забезпечує серверний рендеринг (Server-Side Rendering), статичний експорт а також підтримує статичний

генерацію (Static Site Generation) та клієнтський рендеринг (Client-Side Rendering), що полегшують розробку сучасних веб-застосунків [5].

Переваги використання Next.js:

1. Серверний рендеринг (SSR) позитивно впливає на продуктивність та SEO-оптимізацію додатку, що дозволяє зменшити час завантаження сторінок та покращити їх індексацію пошуковими системами [6].

2. Статистичний експорт сприяє швидкості завантаження сторінок та зменшує навантаження на сервер. Це покращує користувацький досвід та ефективність додатку [5].

3. Автоматичне розділення коду, адже Фреймворк автоматично розділяє код, що допомагає зменшити розмір початкового завантаження сторінки та покращити її продуктивність. Це сприяє швидкому завантаженню та оптимізації роботи додатку.

4. Підтримка динамічних маршрутів, що спрощує створення складних додатків з різноманітними структурами URL та багатою функціональністю.

5. Повна підтримка TypeScript, що забезпечує покращену типізацію та надійність коду. TypeScript допомагає уникнути помилок та полегшує підтримку великих проектів [5].

6. Фреймворк підтримує багато різних плагінів, що дозволяє розширювати його функціональність та забезпечувати більшу гнучкість розробки.

Next.js для розробки веб-додатка онлайн-курсів обумовлений його високою продуктивністю, зручністю розробки та багатими можливостями. Завдяки своїм перевагам, Next.js дозволяє створювати сучасні, швидкі та надійні веб-додатки, які задовольняють потреби як розробників, так і кінцевих користувачів.

**React** був обраний як основна бібліотека для розробки веб-додатка для онлайн-курсів завдяки його широким можливостям у створенні динамічних та інтерактивних інтерфейсів, а також захоплюючій екосистемі, що включає різноманітні сторонні бібліотеки та інструменти [7].

#### Переваги використання React:

1. Базування на компонентній структурі, яка дозволяє розробити веб-додаток на невеликі, повторно використовувані компоненти. Це спрощує розробку, збереження коду та підтримку додатка.

2. React використовує віртуальний DOM для ефективного оновлення інтерфейсу, що дозволяє зменшити кількість операцій з DOM, покращити продуктивність та швидкість рендерингу [8].

3. Односторонній потік даних (One-way data binding) дозволяє ефективно керувати станом додатка та уникати проблем з даними, що можуть виникати в більш складних архітектурах [9].

4. React має широку екосистему сторонніх бібліотек та інструментів, що дозволяє легко розширювати функціональність додатка та використовувати сучасні технології.

**Stripe** – це міжнародна технологічна компанія, що надає інфраструктуру для онлайн-платежів, тому був обраний для використання у веб-додатку для онлайн курсів через його надійність, безпеку та широкі можливості управління платежами та підписками [10].

#### Переваги використання Stripe:

1. Stripe забезпечує високий рівень безпеки платежів за допомогою шифрування даних та дотримання стандартів безпеки PCI-DSS. Це важливо для збереження конфіденційності та недопущення шахрайства з боку користувачів [10].

2. Stripe має зручну API та документацію, що спрощує процес інтеграції платіжних функцій у веб-додаток. Це дозволяє швидко та ефективно налаштувати прийом платежів.

3. Також вбудовані засоби що дозволяють управляти підписками, створювати, змінювати та скасовувати підписки користувачів. Це важливо для реалізації платних доступів до курсів та послуг.

4. Підтримується масштабованість та гнучкість у налаштуванні платіжних рішень, що дозволяє додатково адаптувати функціонал платежів під конкретні потреби додатку [11].

5. Система надає ефективну клієнтську підтримку у разі виникнення питань або проблем з платежами.

Отже, використання Stripe у веб-додатку для онлайн курсів забезпечить надійну та зручну систему прийому платежів та управління підписками, що сприятиме зручності користувачів та ефективності фінансової діяльності додатку [11].

**Mux** - це платформа для відеохостингу та управління відео контентом в Інтернеті, був обраний для використання у веб-додатку для онлайн курсів через його високу надійність, зручність та простоту інтеграції [12].

Переваги використання Mux:

1. Висока доступність та надійність відеохостингу, що забезпечує стабільну роботу та швидке завантаження відео для користувачів [13].

2. Зручний API та інструменти для інтеграції відео контенту у веб-додаток, що спрощує процес розробки та управління відеоматеріалами.

3. Mux автоматично перекодує відео для оптимального відтворення на різних пристроях та швидкостях Інтернету, що забезпечує зручність для користувачів з різними технічними характеристиками.

Також, гнучкість управління та масштабованість відеоконтентом, що дозволяє додатково адаптувати функціонал під конкретні потреби додатку та збільшувати обсяг відео матеріалів [13].

Використання Mux у веб-додатку для онлайн курсів забезпечить надійну та зручну систему управління відео контентом, що підвищить якість та ефективність навчання користувачів[13].

**Prisma** - це ORM (Object-Relational Mapping) для TypeScript та Node.js, що полегшує роботу з базами даних. Prisma забезпечує простий та інтуїтивний інтерфейс для взаємодії з базою даних, автоматично генерує SQL запити та підтримує типізацію [14].

Prisma автоматично генерує TypeScript код для взаємодії з базою даних на основі описаних моделей, що спрощує роботу з даними у вашому додатку.

Підтримка різних баз даних включаючи MongoDB, PostgreSQL, MySQL, SQLite та інші [14].

Prisma підтримує автоматичне створення та виконання міграцій для змін структури бази даних, що дозволяє безпечно оновлювати схему даних додатку [15].

Також, варто підмітити ефективність та оптимізацію що підтримує високу продуктивність додатку при роботі з базою даних [15].

**Tailwind CSS** - це утилітарний CSS фреймворк, який дозволяє створювати адаптивні та сучасні інтерфейси. Tailwind CSS надає набір готових класів, що полегшують створення стильного та функціонального дизайну [16].

Переваги використання Tailwind CSS:

1. Використання утилітарних CSS класів, що дозволяють швидко та легко застосовувати стилі без необхідності написання користувацьких CSS правил. Це сприяє швидшій розробці та швидкому створювати прототипи інтерфейсів та проводити тестування різних дизайнів.

2. Можливість повної кастомізації через конфігураційний файл, дозволяючи розробникам налаштовувати кольори, шрифти, розміри та інші параметри відповідно до вимог проекту [17].

3. Tailwind CSS використовує процес «tree-shaking», який видаляє невикористані стилі під час компіляції, значно зменшуючи розмір кінцевих CSS файлів [17].

4. Легко інтегрується з сучасними JavaScript фреймворками, такими як Vue.js, Angular та в нашому випадку React, що робить його зручним вибором для розробки сучасних веб-додатків [16].

**MongoDB** є системою управління NoSQL базами даних, розробленою компанією MongoDB Inc. Ця база даних відрізняється від реляційних систем, таких як MS SQL Server, використовуючи гнучку та швидку документорієнтовану модель зберігання даних. Однією з ключових

особливостей MongoDB є використання BSON (Binary JSON) для зберігання документів, які можуть містити різні типи даних, включаючи масиви та вкладені об'єкти. База даних підтримує горизонтальне масштабування, що дозволяє легко розширювати її для роботи з великими обсягами даних та вимогами розподілених систем [18].

#### Переваги використання MongoDB:

1. Висока продуктивність завдяки своїй здатності швидко обробляти великі обсяги даних та підтримці індексування. Це дозволяє забезпечити швидкий доступ до даних та ефективне виконання запитів.

2. MongoDB надає потужні інструменти для виконання складних запитів, агрегацій та аналітики. Завдяки цьому можна легко обробляти та аналізувати дані, що зберігаються в базі [19].

3. Широка підтримка інструментів та бібліотек для інтеграції з різними мовами програмування та фреймворками, що є зручним для використання у різних проектах [20].

#### Висновки до розділу 1

У першій частині бакалаврської роботи ми детально розглянули та обґрунтували основні аспекти розробки веб-додатків для онлайн-курсів.

Чітко визначено визначені основні цілі та завдання дослідження, а саме розробка методів навчання програмуванню з використанням веб-технологій. Важливість цього дослідження була відзначена в контексті швидкого розвитку сфери інформаційних технологій і зростаючого попиту на кваліфікованих фахівців на ринку праці.

Було проведено всебічний порівняльний аналіз існуючих рішень на ринку веб-додатків для онлайн-курсів, таких як Udemy, Coursera, edX і LinkedIn Learning. Було вивчено їх функціональність і сильні сторони, що дозволило визначити ключові аспекти для поліпшення та впровадження у власну розробку.

Обґрунтовано вибір оптимальної архітектури для створення веб-додатку. Було прийнято рішення використовувати компонентно-орієнтований підхід з урахуванням вимог до масштабованості, гнучкості та продуктивності системи. Застосування буде надійним і ефективним завдяки новітнім технологіям.

Детально розглянуто інструментальні засоби, які будуть використовуватися для розробки додатку, зокрема Next.js, React, Stripe, Mux, Prisma, Tailwind CSS та MongoDB. Було враховано потреби проекту та обрано інструменти, виходячи з їх переваг та придатності.

На основі проведеного аналізу та обґрунтування можна зробити висновок, що обрані підходи, технології та інструменти забезпечать успішну реалізацію веб-додатку для онлайн-курсів. Використовуючи веб-технології для навчання програмуванню, якість освіти буде покращена, а навички, необхідні на поточному ринку праці, будуть задоволені.

## РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ОНЛАЙН КУРСІВ

### 2.1 Визначення варіантів використання та об'єктно-орієнтованої структури

Цей розділ присвячений детальному аналізу вимог до програмного продукту. Основна мета аналізу вимог до веб-додатку онлайн курсів полягає в тому, щоб забезпечити розробку ефективного, зручного і функціонального інструменту для користувачів. Це включає виявлення та документування всіх необхідних функціональних та нефункціональних вимог, які забезпечать задоволення потреб кінцевих користувачів, а також гарантують відповідність додатку встановленим стандартам якості та продуктивності.

1. Функціональні вимоги описують те, що повинен вміти робити продукт, тобто його основні функції. До них належать, наприклад, обробка даних, генерування звітів, взаємодія з користувачем [21].

2. Нефункціональні вимоги стосуються того, як продукт повинен працювати, тобто його якості. До них належать, наприклад, продуктивність, безпека, надійність, масштабованість, зручність використання тощо [22].

Створення веб-додатку онлайн-курсів, який відповідає потребам учнів, потребує ретельного вивчення їх очікувань та побажань. Аналіз вимог – це ключовий етап цього процесу, який допомагає розібратися, які саме функції та характеристики має втілювати програмне забезпечення.

Для досягнення цієї мети були використані різні методи дослідження:

1. Інтерв'ю з учнями, студентами дозволило отримати детальну інформацію про їх потреби та «больові точки» під час проходження дистанційного навчання.

2. За допомогою онлайн-опитувань було охоплено ширше коло учнів, збираючи дані про їхні пріоритети та вподобання.

3. Аналіз відгуків та коментарів, отриманих з різних джерел, допоміг уточнити та доповнити картину потреб учнів.

Для забезпечення ефективного навчання та задоволення потреб користувачів у створенні системи онлайн-курсів, були визначені вимоги, які

сприятимуть безперебійній роботі системи та відповідатимуть всім необхідним стандартам і специфікаціям:

Функціональні вимоги до веб-додатку онлайн курсів:

#### 1. Реєстрація та авторизація:

1.1 Користувачі повинні мати можливість зареєструватися та авторизуватися в системі, використовуючи логін та пароль або інші методи (наприклад, соціальні мережі або електронна пошта). Реєстрація нових користувачів з можливістю вибору ролі (студент, викладач, адміністратор).

1.2 Система повинна мати можливість генерувати та надсилати на email користувача тимчасовий пароль для відновлення доступу до акаунту.

#### 2. Управління профілем:

2.1 Користувачі повинні мати можливість налаштувати свій профіль після реєстрації, додати пароль або електронну пошту, а також підключити Google аккаунт.

2.1 Система повинна мати можливість зберігати фото профілю користувача.

#### 3. Перегляд та пошук курсів:

3.1 Користувачі повинні мати можливість переглядати доступні курси за категоріями, тегами, ключовими словами або іншими критеріями та шукати курси за назвою, викладачем, темою тощо.

3.2 Система повинна мати можливість надавати інформацію про кожен курс, включаючи опис, програму, тривалість, вимоги, ціну.

#### 4. Запис на курси:

4.1 Користувачі повинні мати можливість записуватися на курси, які їх цікавлять та отримувати доступ до навчальних матеріалів курсу після запису або оплати.

4.2 Система повинна мати можливість обробляти платежі за курси (якщо це передбачено).

#### 5. Навчання:

5.1 Користувачі повинні мати можливість переглядати навчальні матеріали курсу, включаючи відео, тексти, презентації, тести або інші завдання.

5.2 Система повинна мати можливість відстежувати прогрес користувачів у курсі та надавати їм зворотний зв'язок.

#### 6. Спілкування та підтримка:

6.1 Користувачі повинні мати можливість спілкуватися з викладачами та іншими учнями курсу через форуми, чати або інші канали.

6.2 Система повинна мати можливість надавати користувачам доступ до бази знань, FAQ та інших ресурсів для самостійного вирішення проблем.

6.3 Користувачі повинні мати можливість звертатися до служби підтримки за допомогою електронної пошти, телефону або чату.

Важливо зазначити, що загальний перелік функціональних вимог до веб-додатку онлайн-курсів має невичерпний список. Конкретні запити можуть варіюватися залежно від цілей та аудиторії вашого проекту. Тому декілька додаткових функцій можуть містити таку інформацію:

#### 7. Додаткові функції:

7.1 Можливість створення власних курсів (для викладачів).

7.2 Визначити адаптивний дизайн для зручного перегляду на різних пристроях та розмірах екрану.

#### Нефункціональні вимоги до веб-додатку онлайн-курсів

##### 1. Продуктивність:

1.1 Веб-додаток повинен працювати швидко та без збоїв, навіть при одночасному використанні великою кількістю користувачів, завантажувати основні сторінки не довше 2 секунд при нормальному навантаженні, під час пікових навантажень (до 1000 одночасних користувачів) час завантаження сторінок не повинен перевищувати 3 секунд.

1.2 Час завантаження сторінок та навчальних матеріалів повинен бути мінімальним. Відгук на дії користувача, такі як натискання кнопок або перемикання між сторінками, повинен бути миттєвим, не більше 0,5 секунд.

Оновлення даних на сторінці без перезавантаження (використання AJAX) повинно відбуватися за 1-2 секунди.

## 2. Надійність:

2.1 Веб-додаток повинен працювати безперебійно 24/7.

2.2 Система повинна мати резервні копії даних та механізми відновлення після збоїв.

2.3 Dodatok повинен бути захищений від кібератак та інших загроз безпеці.

## 3. Зручність використання:

3.1 Інтерфейс повинен бути зрозумілим та легким у використанні навіть для некваліфікованих користувачів.

3.2 Інтерфейс повинен автоматично адаптуватися до різних розмірів екранів (десктопи, планшети, смартфони). Забезпечення зручного використання на сенсорних пристроях.

3.3 Система повинна надавати чіткі та зрозумілі інструкції та підказки.

## 4. Доступність:

4.1 Веб-додаток повинен бути доступним для користувачів з різними рівнями підготовки та навичками роботи з комп'ютером.

4.2 Dodatok повинен відповідати стандартам доступності WCAG.

## 5. Масштабованість:

5.1 Веб-додаток повинен мати можливість масштабуватися для обслуговування зростаючої кількості користувачів та курсів.

5.2 Архітектура додатку повинна дозволити легку інтеграцію нових функцій та модулів без значних змін в основному коді.

5.3 Підтримка багатомовного інтерфейсу з можливістю додавання нових мов без значних зусиль. Використання стандартів кодування (UTF-8) для коректного відображення тексту на різних мовах.

5.4 Система повинна підтримувати різні формати даних та часові пояси.

5.5 Dodatok повинен бути чутливим до культурних особливостей користувачів.

## 6. Безпека:

6.1 Усі особисті дані користувачів повинні зберігатися в зашифрованому вигляді. Дані передаватимуться через захищені протоколи (HTTPS).

6.2 Підтримка багатфакторної автентифікації для додаткової безпеки користувачів.

6.3 Використання надійних алгоритмів хешування для зберігання паролів.

6.4 Захист від поширених атак, таких як SQL-ін'єкції, XSS (міжсайтовий скриптинг) та CSRF (підробка міжсайтових запитів).

6.5 Регулярні перевірки на вразливості та оновлення системи безпеки.

## 7. Сумісність:

7.1 Підтримка основних веб-браузерів (Chrome, Firefox, Safari, Edge) останніх двох версій.

7.2 Коректна робота у браузерах на різних операційних системах (Windows, macOS, iOS, Android).

7.3 Можливість інтеграції з популярними системами управління навчанням (LMS) та іншими сторонніми сервісами через API.

## 8. Аналітика:

8.1 Веб-додаток повинен мати вбудовані інструменти аналітики для відстеження поведінки користувачів та ефективності курсів.

8.2 Система повинна збирати дані про те, як користувачі взаємодіють з додатком та навчальними матеріалами.

8.3 Додаток повинен надавати звіти та візуалізації даних, які допоможуть приймати обґрунтовані управлінські рішення щодо розвитку проекту.

Аналіз вимог користувачів дав підстави для визначення варіантів використання додатку планування онлайн курсів. На рис. 2.1. наведено діаграму використання програмної системи.



Рисунок 2.1. Діаграма варіантів використання

Таким чином, відображається підготовка до проектування системи з врахуванням всіх необхідних технічних аспектів та вимог користувача, що є важливою основою для подальших етапів розробки [23].

#### Об'єктно-орієнтована структура.

На основі визначених вимог до веб-додатку для онлайн-курсів було розроблено загальну діаграму об'єктно-орієнтованої структури системи (рис. 2.2). Ця діаграма наочно демонструє основні сутності системи, їх взаємозв'язки та взаємодію [24].

Кожна сутність на діаграмі представляє ключовий компонент або модуль веб-додатку, що містить набір атрибутів та методів, необхідних для реалізації відповідної функціональності. Наприклад, сутність "Користувач" включає атрибути, такі як ім'я, email, пароль, та методи для реєстрації, авторизації, редагування профілю тощо. Сутність "Курс" містить атрибути, що описують курс (назва, опис, категорія), та методи для створення, редагування, видалення курсів.

```

classDiagram
    class Course {
        +String id
        +String userId
        +String title
        +String description
        +String imageUrl
        +Float price
        +Boolean isPublished
        +String categoryId
        +DateTime createdAt
        +DateTime updatedAt
        +createCourse() void
        +updateCourse() void
        +deleteCourse() void
        +getCourseDetails() void
    }
    class Category {
        +String id
        +String name
        +createCategory() void
        +updateCategory() void
        +deleteCategory() void
    }
    class Attachment {
        +String id
        +String name
        +String url
        +String courseId
        +DateTime createdAt
        +DateTime updatedAt
        +createAttachment() void
        +updateAttachment() void
        +deleteAttachment() void
    }
    class Chapter {
        +String id
        +String title
        +String description
        +String videoUrl
        +Int position
        +Boolean isPublished
        +Boolean isFree
        +String courseId
        +DateTime createdAt
        +DateTime updatedAt
        +createChapter() void
        +updateChapter() void
        +deleteChapter() void
    }
    class Purchase {
        +String id
        +String userId
        +String courseId
        +DateTime createdAt
        +DateTime updatedAt
        +createPurchase() void
        +updatePurchase() void
        +deletePurchase() void
    }
    class StripeCustomer {
        +String id
        +String userId
        +String stripeCustomerId
        +DateTime createdAt
        +DateTime updatedAt
        +createStripeCustomer() void
        +updateStripeCustomer() void
        +deleteStripeCustomer() void
    }
    class User {
        +String userId
        +String email
        +String google
        +createUser() void
        +updateUser() void
        +deleteUser() void
    }
    class MuxData {
        +String id
        +String assetId
        +String playbackId
        +String chapterId
        +createMuxData() void
        +updateMuxData() void
        +deleteMuxData() void
    }
    class UserProgress {
        +String id
        +String userId
        +String chapterId
        +Boolean isCompleted
        +DateTime createdAt
        +DateTime updatedAt
        +trackProgress() void
        +updateProgress() void
        +resetProgress() void
    }

    Course "1" -- "*" Category : belongs to
    Course "1" -- "*" Attachment : contains
    Course "1" -- "1" Chapter : has
    Course "1" -- "*" Purchase : belongs to
    StripeCustomer "1" -- "1" User : belongs to
    Chapter "1" -- "1" MuxData : contains
    Chapter "1" -- "1" UserProgress : contains
    Attachment "*" -- "1" Chapter : belongs to
    Purchase "*" -- "1" Chapter : belongs to
    UserProgress "*" -- "1" Chapter : belongs to
  
```

Пояснения:

Category - клас, що представляє категорію курсів. Має методи для створення, оновлення та видалення категорії.

Attachment - клас, що представляє вкладення курсу. Має методи для створення, оновлення та видалення вкладення.

Chapter - клас, що представляє главу курсу. Має методи для створення, оновлення та видалення глави.

MuxData - клас, що представляє дані Mux для глави. Має методи для створення, оновлення та видалення даних.

UserProgress - клас, що відстежує прогрес користувача у главі курсу. Має методи для відстеження, оновлення та скидання прогресу.

Purchase - клас, що представляє покупку курсу користувачем. Має методи для створення, оновлення та видалення покупки.

StripeCustomer - клас, що представляє клієнта Stripe. Має методи для створення, оновлення та видалення даних клієнта.

Ця діаграма класів відображає структуру системи та взаємозв'язки між її компонентами, що сприяє кращому розумінню її загальної архітектури.

## 2.2 Опис структури даних з якими працює веб-додаток

У процесі створення веб-додатку для онлайн курсів було використано широкий спектр передових технологій, спрямованих на покращення продуктивності системи, зручності користування та можливості інтеграції з різноманітними зовнішніми сервісами. Однією з ключових технологій, яка використовується для взаємодії з базою даних у цьому проекті, є Prisma.

Завдяки Prisma розробники можуть швидко створювати запити до бази даних, використовуючи зручний DSL (Domain-Specific Language), що дозволяє ефективно взаємодіяти з даними та забезпечує високу якість роботи системи [27].

До особливостей PrismaClient можна віднести:

1. Клас PrismaClient генерується автоматично на основі схеми даних. Файл schema.prisma описує всі моделі, поля, типи даних та зв'язки між ними. Після опису схеми виконується команда генерації, яка створює Prisma-клієнт із типобезпечними методами для роботи з базою даних [28].

2. Prisma забезпечує типобезпеку на етапі компіляції, що дозволяє уникнути багатьох помилок, пов'язаних з неправильною роботою з базою

даних. Усі запити, що виконуються через PrismaClient, перевіряються на відповідність описаній схемі, що значно підвищує надійність та стабільність додатку [29].

3. PrismaClient надає прості та інтуїтивно зрозумілі методи для виконання CRUD-операцій (Create, Read, Update, Delete) з базою даних. Кожна модель у схемі має відповідний набір методів для роботи з її записами. Наприклад, для моделі User можна використовувати методи create, findMany, update, delete та інші [30].

Основним компонентом Prisma є клас PrismaClient, який генерується на основі схеми даних, описаної в спеціальному файлі schema.prisma. Цей клієнт дозволяє розробникам легко виконувати запити до бази даних, використовуючи зрозумілі та типобезпечні методи [31].

Файл schema.prisma є центральним місцем, де визначається структура бази даних для проекту. У ньому описуються моделі даних, їх взаємозв'язки, а також налаштування підключення до бази даних [31].

На основі схеми, описаної в schema.prisma, Prisma автоматично генерує PrismaClient - об'єкт, який надає розробникам зручний інтерфейс для взаємодії з базою даних. Цей клієнт містить типобезпечні методи для виконання різноманітних операцій, таких як [31]:

create() - створення нових записів

update() - оновлення існуючих записів

delete() - видалення записів

findMany() - отримання списку записів

findUnique() - отримання унікального запису

Наприклад, для отримання всіх записів з таблиці "User" можна використати метод findMany():

```
typescript
```

```
const allUsers = await prisma.user.findMany();
```

Деякі додаткові функції, через які було обрано саме Prisma:

1. Автоматично генерує та виконує міграції бази даних, що дозволяє легко змінювати структуру бази даних без ручного написання SQL-запитів.

2. Дозволяє визначати та використовувати реляційні зв'язки між моделями. Це означає, що можна легко виконувати запити, які включають дані з кількох пов'язаних таблиць [32].

3. Prisma підтримує складні фільтри та сортування для запитів до бази даних. Це дозволяє розробникам легко отримувати необхідні дані, застосовуючи різноманітні умови [32].

Таким чином, Prisma значно спрощує роботу з базою даних, надаючи зручний та потужний інтерфейс для виконання типових операцій. Це дозволяє розробникам зосередитись на бізнес-логіці додатку, а не витратити час на написання низькорівневого коду для взаємодії з базою даних. Використання PrismaClient у поєднанні з файлом `schema.prisma` дозволяє ефективно та безпечно взаємодіяти з базою даних у процесі розробки веб-додатку для онлайн курсів [30].

## 2.3 Проектування та реалізація алгоритмів роботи системи

Розробка та впровадження ефективних алгоритмів є ключовим аспектом створення функціональної та продуктивної системи онлайн-курсів. Ці алгоритми відповідають за виконання широкого спектру обчислювальних завдань, обробку даних та прийняття рішень в системі, забезпечуючи її стабільну та надійну роботу.

Ефективні алгоритми виконують різноманітні обчислювальні завдання, такі як:

1. Алгоритми обробки забезпечують ефективне управління великою кількістю навчальних матеріалів, даних користувачів та їх прогресу. Це включає в себе сортування, фільтрацію, індексацію та зберігання інформації у базі даних [33].

2. Адаптація контенту під індивідуальні потреби кожного студента. Використання алгоритмів машинного навчання дозволяє системі аналізувати результати навчання, визначати слабкі та сильні сторони студента і на основі цього пропонувати найбільш відповідний навчальний матеріал [34].

3. Алгоритми рекомендацій допомагають студентам знаходити курси, які найбільше відповідають їхнім інтересам та навчальним потребам. Вони аналізують історію переглядів, участі в курсах та взаємодію з матеріалами, щоб запропонувати релевантний контент [35].

4. Алгоритми управління ресурсами забезпечують ефективне використання серверних потужностей та інших технічних ресурсів системи. Вони оптимізують навантаження на сервери, забезпечують балансування трафіку та запобігають збоям системи під час пікових навантажень [36].

5. Алгоритми безпеки, які захищають конфіденційні дані користувачів від несанкціонованого доступу та кібератак. Вони включають в себе методи шифрування, автентифікації та моніторингу підозрілої активності [35].

Впровадження алгоритмів у систему онлайн-курсів відбувається за допомогою різних технологій та інструментів, таких як:

- програмних мов, таких як TypeScript.
- бібліотеки та фреймворки, такі як Prisma, допомагають розробникам створювати ефективні алгоритми та оптимізувати їх роботу.
- тестування алгоритмів дозволяє виявити та виправити помилки, що забезпечує стабільну та надійну роботу системи.

Переваги впровадження ефективних алгоритмів:

- дозволяє зменшити час виконання обчислювальних задач та збільшити швидкість реакції системи на дії користувачів.
- сприяють безперебійній роботі системи, зменшуючи ризик збоїв та забезпечуючи високу надійність.
- завдяки персоналізованим рекомендаціям та швидкому доступу до потрібної інформації, користувачі отримують більш якісне та зручне навчання.
- оптимізація використання технічних ресурсів дозволяє знизити витрати на інфраструктуру та покращити загальну ефективність системи.

Для наочного відображення потоку дій та процесів, що відбуваються в системі під час виконання алгоритмів, було розроблено кілька діаграм активності. Діаграми активності — це візуальні представлення послідовності

та паралельності операцій, які виконуються в системі. Вони допомагають розробникам та аналітикам чітко продемонструвати логіку роботи алгоритмів, їх взаємодію з іншими компонентами системи та кінцеві результати [37].

Призначення діаграм активності:

1. Забезпечують графічне відображення послідовності дій, дозволяючи легко зрозуміти логіку роботи алгоритмів. Вони демонструють, як дані проходять через різні етапи обробки та які операції виконуються на кожному етапі [37].

2. Допомагають показати взаємодію між різними компонентами системи. Це включає в себе передачу даних між модулями, виклики функцій та обмін повідомленнями [37].

3. Завдяки наочності, дозволяють розробникам та аналітикам виявити можливі проблеми у виконанні алгоритмів, такі як затримки чи вузькі місця, та оптимізувати процеси для підвищення продуктивності системи [37].

4. Слугують важливим засобом документування системи, надаючи зрозуміле представлення алгоритмічного дизайну. Це корисно для навчання нових членів команди та забезпечення наступності в розробці [37].

Діаграми активності використовують низку спеціальних елементів для візуалізації потоку дій та процесів у системі. Основними елементами діаграм активності є:

1. Початковий стан (Initial Node) - позначає початок потоку дій на діаграмі. Він зображується у вигляді закресленого кола [38].

2. Дія (Action) - представляє собою атомарну операцію, яка виконується в системі. Вона зображується у вигляді округленого прямокутника [38].

3. Рішення (Decision Node) - використовується для моделювання розгалужень у потоці дій. Воно зображується у вигляді ромба [38].

4. Злиття (Merge Node) - використовується для об'єднання кількох потоків дій в один. Воно зображується у вигляді ромба [38].

5. Розгалуження (Fork Node) - дозволяє моделювати паралельні потоки дій. Воно зображується у вигляді горизонтальної риски [38].

6. Об'єднання (Join Node) - використовується для синхронізації паралельних потоків дій. Воно зображується у вигляді горизонтальної риски [38].

7. Кінцевий стан (Final Node) - позначає завершення потоку дій на діаграмі. Він зображується у вигляді закресленого кола з маленьким колом всередині [38].

Діаграма активності для створення нового курсу (рис. 2.3.) показує послідовність дій користувача, таких як назва курсу, що в подальшому переведе на сторінку детального створення та наповнення курсу, для публікації та видимості іншим користувачам.

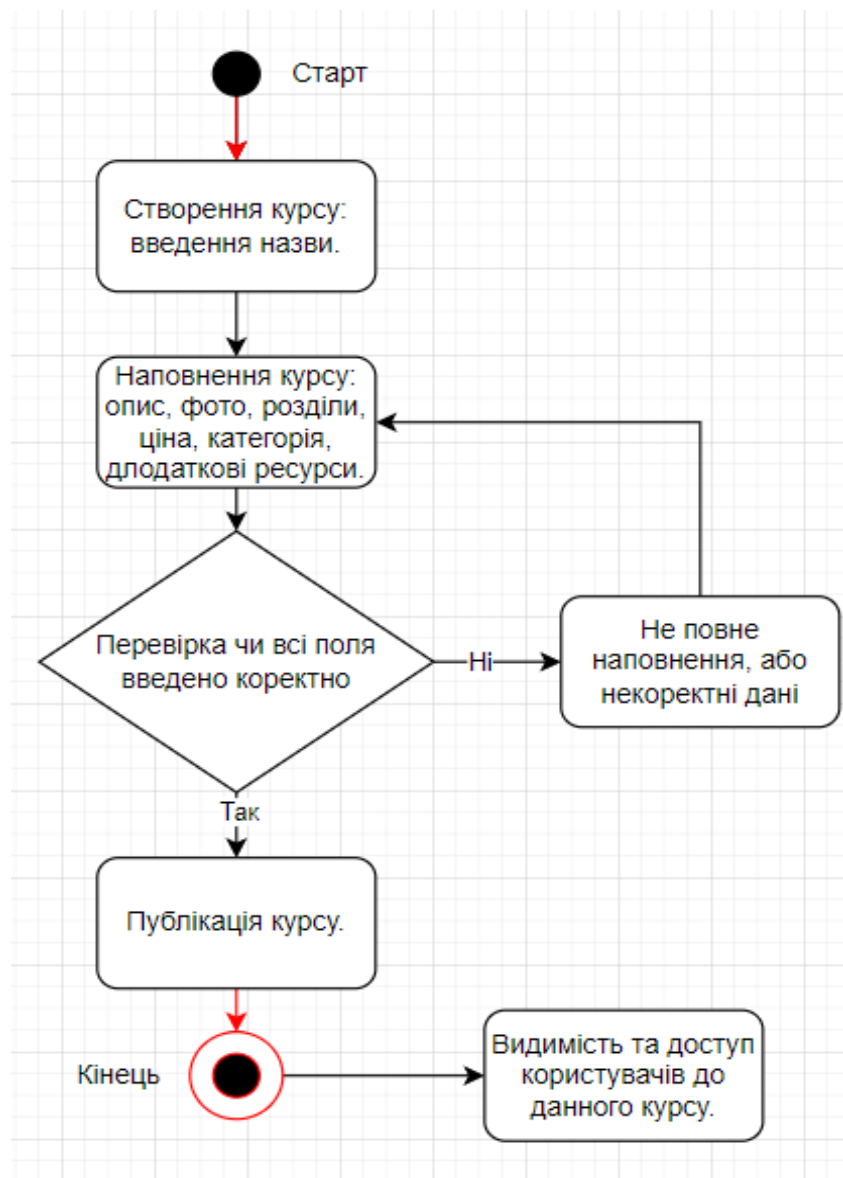


Рисунок 2.3. Діаграма активності для створення нового курсу

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

На основі діаграми активності для створення нового курсу було створено наступну таблицю специфікації(табл. 2.1):

Таблиця 2.1

Таблиця специфікації

|                                                    |            |                                                            |
|----------------------------------------------------|------------|------------------------------------------------------------|
| Ім'я варіанту використання: створення нового курсу |            | Рівень високий                                             |
| Основний актор: користувач (адміністратор)         |            |                                                            |
| Короткий опис:                                     |            | Створення нового курсу для користувачів                    |
| Передумови:                                        |            | Вхід з доступом адміністратора.                            |
| Відносини:                                         |            |                                                            |
|                                                    | Включення: | Ініціалізація                                              |
| Основний потік:                                    |            | Користувач може створити новий курс у відповідній вкладці. |
| Альтернативні/виключні потоки:                     |            | Повідомлення про помилку                                   |

Діаграма активності для наповнення даними курсу (рис 2.4.). Дана діаграма демонструє послідовність дій користувача для наповнення та редагування інформації про курс, таких як опис, головне фото, відео матеріал, ціна, розділи, додаткові матеріали.



|                                |            |                                                                                                                                       |
|--------------------------------|------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Передумови:                    |            | Вхід з доступом адміністратора, створений курс                                                                                        |
| Відносини:                     |            |                                                                                                                                       |
|                                | Включення: | Ініціалізація                                                                                                                         |
| Основний потік:                |            | Користувач повинен мати створений курс.<br>Далі йде перехід на нову сторінку відповідного курсу для внесення даних у відповідні поля. |
| Альтернативні/виключні потоки: |            | Повідомлення про помилку. неможливість публікації курсу.                                                                              |

Діаграма активностей для проходження курсу користувачами (рис 2.5.): дана діаграма може показати, як користувач проходить курс та відстеження його прогресу.

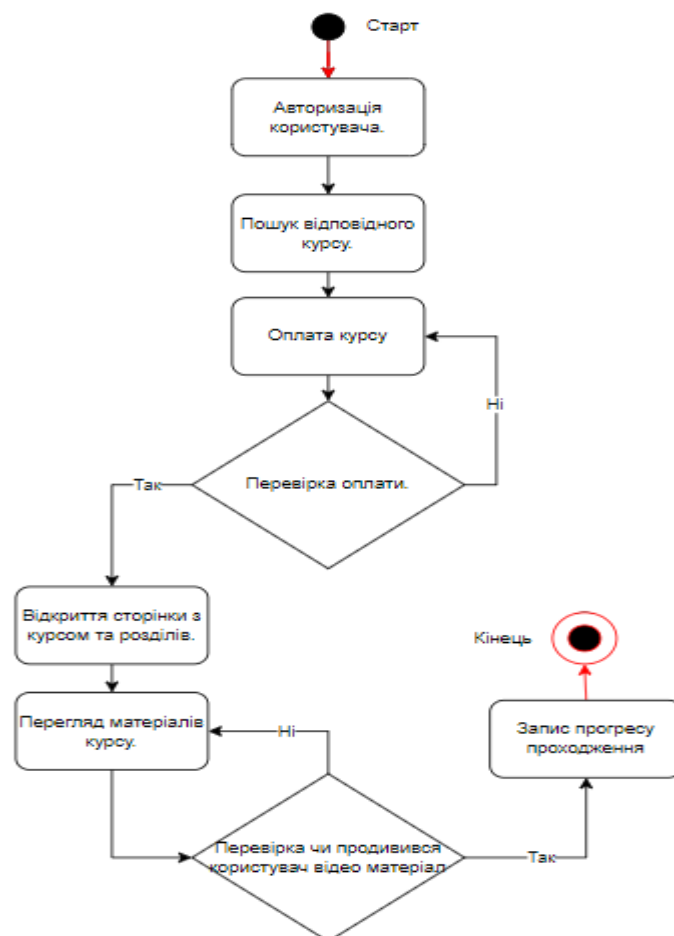


Рисунок 2.5. Діаграма активності для проходження курсу

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

На основі діаграми активності для проходження курсу було створено наступну таблицю специфікації (табл. 2.3):

Таблиця 2.3

Таблиця специфікації

|                                               |            |                                                                                                                             |
|-----------------------------------------------|------------|-----------------------------------------------------------------------------------------------------------------------------|
| Ім'я варіанту використання: проходження курсу |            | Рівень: високий                                                                                                             |
| Основний актор: користувач                    |            |                                                                                                                             |
| Короткий опис:                                |            | Процес проходження курсу та збереження інформації про прогрес.                                                              |
| Передумови:                                   |            | Авторизація користувача                                                                                                     |
| Відносини:                                    |            |                                                                                                                             |
|                                               | Включення: | Ініціалізація                                                                                                               |
| Основний потік:                               |            | Користувач повинен обрати відповідний курс для його проходження, та переглядати матеріал викладений у відповідних розділах. |
| Альтернативні/виключні потоки:                |            | Повідомлення про помилку. Неможливість перегляду відео-матеріалу.                                                           |

Діаграма послідовності - це ефективний інструмент для візуалізації взаємодії між об'єктами системи в процесі виконання певного сценарію або функціональності, оскільки вона [39]:

1. Показує, як об'єкти системи взаємодіють один з одним під час виконання певних дій або сценаріїв, що дозволяє розробникам і аналітикам побачити повний ланцюжок обміну повідомленнями та викликів методів між об'єктами [39].

2. Відображає взаємодій у часовій послідовності. Кожне повідомлення чи виклик методу показаний на вертикальній шкалі часу, що дозволяє зрозуміти, у якій послідовності відбуваються дії [39].

3. Допомогає детально проаналізувати логіку роботи системи, виявити можливі проблеми чи невідповідності у взаємодії об'єктів, а також оптимізувати процеси для покращення продуктивності та надійності [39].

4. Служить важливим засобом документування, забезпечуючи чітке та зрозуміле представлення процесів, що відбуваються у системі. Особливу користь ця функція надає для навчання нових членів команди та підтримки проекту в майбутньому [39].

Елементи діаграми послідовності:

1. Актори представляють зовнішні об'єкти або користувачів, що взаємодіють із системою. Вони знаходяться на лівому краю діаграми і ініціюють певні дії.

2. Об'єкти системи, що беруть участь у сценарії, розміщені на горизонтальній осі діаграми. Вони можуть бути як фізичними компонентами системи, так і логічними елементами.

3. Стрілки між об'єктами представляють повідомлення або виклики методів, що передаються між об'єктами. Кожне повідомлення позначено відповідною назвою та направлено від відправника до отримувача.

4. Вертикальні смуги на діаграмі, що представляють активний час об'єкта під час виконання дії або обробки повідомлення. Вони показують тривалість виконання певних процесів.

5. Діаграма може відображати створення та знищення об'єктів, що допомагає зрозуміти, коли об'єкти з'являються та зникають під час сценарію.

Використовуючи графічний редактор, можна створити наочну діаграму послідовності, яка допоможе візуалізувати потік керування та обмін даними між об'єктами системи. Це дозволяє легко зрозуміти, які об'єкти беруть участь у виконанні певної функціональності, в якій послідовності вони взаємодіють та які методи викликаються, та дозволяє виявити та усунути потенційні проблеми на ранніх етапах розробки.

Діаграма послідовності, яка описує дії користувача, що взаємодіє з веб-додатком. (рис. 2.6.)

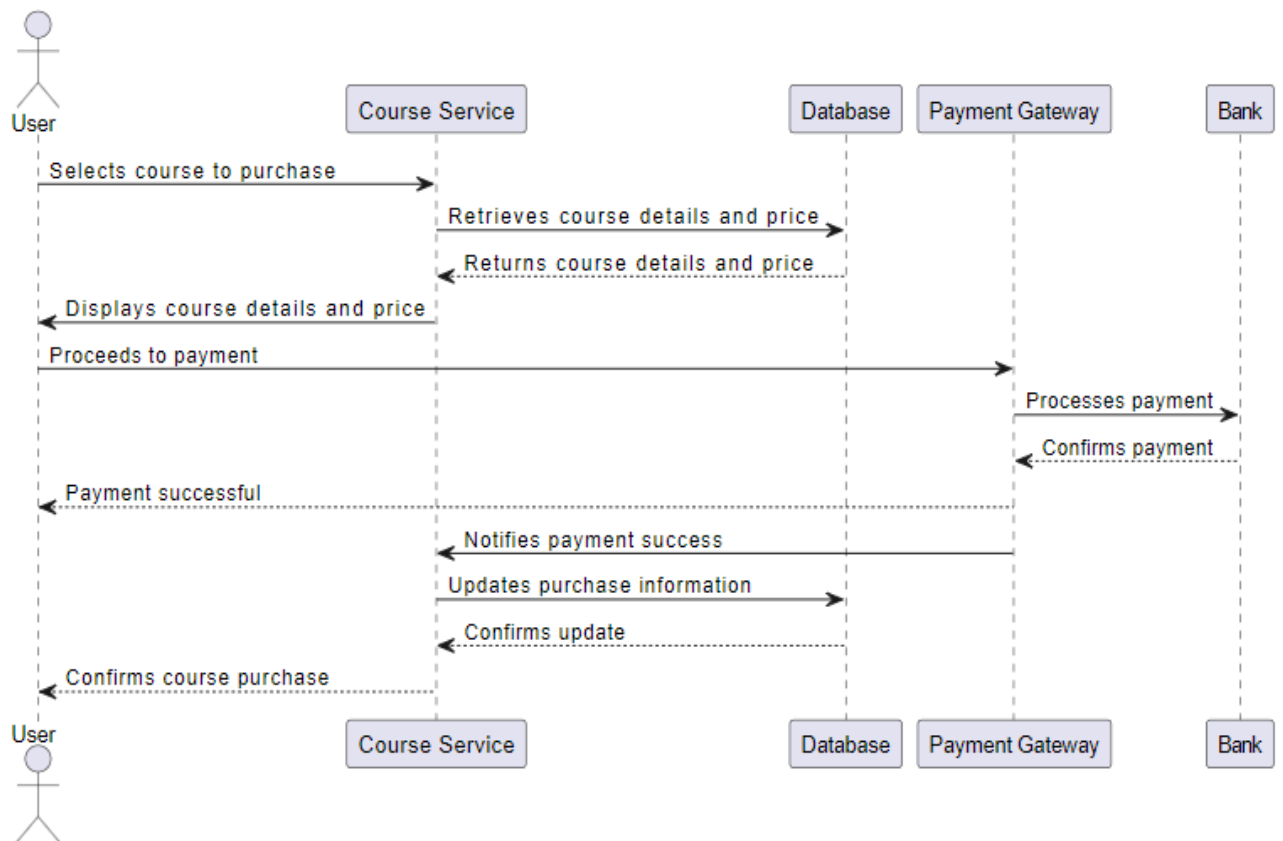


Рисунок 2.6. Діаграма послідовності взаємодії з додатком

Діаграми послідовності надають лаконічне та зрозуміле зображення динамічної взаємодії та потоку повідомлень у системі. Вони відіграють ключову роль у розумінні послідовності подій та обміну даними, що дозволяє глибше аналізувати та розуміти поведінку системи [39].

Переваги ретельного проектування та реалізації системних алгоритмів за допомогою діаграм послідовності відображаються у:

1. Надійність та безперебійність роботи системи, Завдяки детальному проектуванню алгоритмів можна забезпечити стабільне функціонування системи, мінімізуючи ризики виникнення збоїв та помилок [40].
2. Добре спроектовані алгоритми надають користувачам зручні та ефективні засоби для виконання своїх завдань, що підвищує загальну продуктивність системи [40].
3. Забезпечення безперебійного обміну повідомленнями та даними між компонентами системи сприяє кращій координації та взаємодії, що особливо важливо в складних багатокористувацьких середовищах [40].

4. Ретельне проектування дозволяє легко інтегрувати систему з різними зовнішніми сервісами та платформами, забезпечуючи розширення функціональних можливостей та покращення користувацького досвіду [40].

5. Збір та аналіз даних у реальному часі дозволяє створювати детальні звіти про роботу системи, що сприяє ухваленню обґрунтованих рішень щодо подальшого розвитку та вдосконалення [40].

Впровадження діаграм послідовності та ретельне проектування системних алгоритмів мають значний позитивний вплив на загальну якість обслуговування та продуктивність системи:

1. Завдяки чіткому розумінню взаємодій та оптимізації алгоритмів користувачі отримують більш стабільний та зручний сервіс, що підвищує їх задоволеність та лояльність.

2. Ефективні алгоритми та оптимізовані процеси сприяють більш швидкому виконанню завдань, зменшують час простоїв та забезпечують високу продуктивність системи.

Таким чином, діаграми послідовності та ретельне проектування алгоритмів відіграють критично важливу роль у розробці та експлуатації сучасних інформаційних систем, забезпечуючи їх надійність, ефективність та високу якість обслуговування користувачів.

## 2.4 Реалізація функціоналу веб-додатку онлайн курсів

Розглянемо основні етапи створення курсу у веб-додатку. Для початку користувач має здійснити вхід в акаунт системи з доступом адміністратора. Після чого йому доступні сторінки Курси та Аналітика де він може створити новий курс за допомогою відповідної функції , яка представлена на рисунку 2.7.

```
export async function POST(  
  req: Request,  
) {  
  try {  
    const {userId} = auth()  
    const {title}=await req.json()  
    if (!userId){
```

```

    return new NextResponse("Неавторизований", {status:401})
  }
  const course = await db.course.create({
    data: {
      userId,
      title,
    }
  })
  return NextResponse.json(course)
} catch (error) {
  console.log("[COURSES]", error )
  return new NextResponse("Внутрішня помилка",{status: 500})
}
}

```

Рисунок 2.7. Функція створення нового курсу

Тепер детальніше зупинимось на створенні розділів кожного курсу. Вони створюються так само як і курс але тепер прив'язані до нього. Функція створення розділу представлена на рисунку 2.9.

```

const lastChapter = await db.chapter.findFirst({
  where: {
    courseId: params.courseId,
  },
  orderBy: {
    position: "desc",
  },
});
const newPosition = lastChapter ? lastChapter.position + 1 : 1;
const chapter = await db.chapter.create({
  data: {
    title,
    courseId: params.courseId,
    position: newPosition,
  }
});

```

Рисунок 2.8. Функція створення розділу.

Спочатку відбувається знаходження останнього розділу в курсі для визначення позиції нового розділу. Якщо є попередні розділи, новий розділ отримає позицію, яка на одиницю більша за позицію останнього розділу. Інакше, новий розділ отримує позицію 1. Після чого новий розділ створюється з вказаним title, courseId та position. Цей код забезпечує створення нового розділу в курсі тільки якщо користувач автентифікований та є власником курсу.

Після створення розділів ми повинні мати змогу перейти до їх редагування, щоб додати або змінити їх наповнення. Можна додати опис, змінити назву, та обрати чи буде цей розділ безкоштовним для перегляду користувачів, які ще не придбали курс, це може бути корисно щоб показати, наприклад, демонстраційну версію того, чому буде присвячений курс, щоб користувачі більш детально ознайомились з тим, що їх очікує та обрали чи варто їм взагалі купувати його, чи ті знання та напрямки вони хотіли досягнути в процесі проходження обраного курсу. Функція чи буде розділ безкоштовним для користувачів представлена на рисунку 2.9.

```
export const ChapterAccessForm = ({
  initialData,
  courseId,
  chapterId
}: ChapterAccessFormProps) => {
  const [isEditing, setIsEditing] = useState(false);
  const toggleEdit = () => setIsEditing((current) => !current);
  const router = useRouter();
  const form = useForm<z.infer<typeof formSchema>>({
    resolver: zodResolver(formSchema),
    defaultValues: {
      isFree: !!initialData.isFree
    },
  });
  const { isSubmitting, isValid } = form.formState;

  const onSubmit = async (values: z.infer<typeof formSchema>) => {
    try {
      await
    axios.patch(`/api/courses/${courseId}/chapters/${chapterId}`, values);
      toast.success("Chapter updated");
      toggleEdit();
      router.refresh();
    } catch {
      toast.error("Something went wrong");
    }
  }
}
```

Рисунок 2.9. Функція підтвердження безкоштовного розділу

Використовуючи стан для перемикавання між режимами редагування та перегляду, хук useRouter для навігації, і хук useForm для управління формою з валідацією за допомогою бібліотеки Zod. При відправленні форми виконується асинхронний запит PATCH для оновлення даних розділу,

відображається повідомлення про успіх або помилку, режим редагування вимикається, і сторінка оновлюється.

Та найголовніше це додавання відеоматеріалу до розділу. Ця додаткова функція додавання відеоматеріалу представлена на рисунку 2.10.

```
if (values.videoUrl) {  
  const existingMuxData = await db.muxData.findFirst({  
    where: {  
      chapterId: params.chapterId,  
    }  
  });  
  if (existingMuxData) {  
    await Video.Assets.del(existingMuxData.assetId);  
    await db.muxData.delete({  
      where: {  
        id: existingMuxData.id,  
      }  
    });  
  }  
  const asset = await Video.Assets.create({  
    input: values.videoUrl,  
    playback_policy: "public",  
    test: false,  
  });  
  await db.muxData.create({  
    data: {  
      chapterId: params.chapterId,  
      assetId: asset.id,  
      playbackId: asset.playback_ids?.[0]?.id,  
    }  
  });  
}
```

Рисунок 2.10. Функція додавання відеоматеріалу

Робота полягає в декількох етапах. По-перше, перевіряється, чи передано нову URL відео в запиті. По-друге, знаходиться існуючий запис про відео в базі даних, Якщо існує, видаляється старий відео-актив з Мух, видаляється запис про старе відео з бази даних. По-третє, створюється новий відео-актив на Мух та новий запит про відео в базі даних. Ці дії забезпечують оновлення інформації про розділ та управління відео для відповідного розділу курсу.

Для того щоб відслідковувати чи всі поля мають інформацію та заповнені коректно було реалізовано лічильник, який перевіряє кількість полів наповнення для роботи подальших функцій додатку(рис2.11.).

```

const requiredFields = [
  course.title,
  course.description,
  course.imageUrl,
  course.price,
  course.categoryId,
  course.chapters.some(chapter => chapter.isPublished),
];
const totalFields = requiredFields.length;
const completedFields = requiredFields.filter(Boolean).length;
const completionText = `(${completedFields}/${totalFields})`;
const isComplete = requiredFields.every(Boolean);

```

Рисунок 2.11. Реалізація лічильника

Данна реалізація має масив необхідних полів курсу які необхідно перевірити, після чого визначається загальна кількість полів та поля які є заповнені. В результаті чого створюється текстовий рядок, який показує кількість заповнених полів від загальної кількості, у форматі (заповнені/загальна кількість).

Після того як всі поля заповнення розділу мають коректні дані, та лічильник перевірів чи кількість полів та заповнення вірне, доступна функція публікації, яка демонструє що розділ має кінцеву форму оформлення та готовий бути використаний для демонстрації користувачам під час їхнього проходження.

Також коли адміністратор вніс всі зміни в наповнення курсу, він також стає валідним для публікації, що свідчить про завершений цикл створення курсу та демонстрація його користувачам, для подальшого ознайомлення чи покупки. Функція публікації курсу представлена на рисунку 2.12.

```

const course = await db.course.findUnique({
  where: {
    id: params.courseId,
    userId,
  },
  include: {
    chapters: {
      include: {
        muxData: true,
      }
    }
  }
});

```

```

    if (!course) {
      return new NextResponse("Не знайдено", { status: 404 });
    }
    const hasPublishedChapter = course.chapters.some((chapter) =>
chapter.isPublished);

    if (!course.title || !course.description || !course.imageUrl ||
!course.categoryId || !hasPublishedChapter) {
      return new NextResponse("Відсутні обов'язкові поля", { status:
401 });
    }
    const publishedCourse = await db.course.update({
      where: {
        id: params.courseId,
        userId,
      },
      data: {
        isPublished: true,
      }
    });
  });

```

Рисунок 2.12. Функція публікації курсу

Спочатку відбувається пошук курсу за допомогою `db.course.findUnique` шукає курс за `courseId` та `userId`, включаючи пов'язані `chapters` та `muxData`. Далі відбувається перевірка обов'язкових полів і чи є хоча б одна опублікована глава (`hasPublishedChapter`). Якщо будь-яке поле відсутнє, повертається відповідь "Відсутні обов'язкові поля" з статусом 401. Після чого оновлюється курс, використовуючи `db.course.update` для оновлення статусу публікації курсу (`isPublished: true`) та повертається JSON-відповідь з оновленим курсом.

Також з додаткового функціоналу адміністратора створено функцію для відображення аналітики додатку(рис 2.13.).

```

export const DataCard = ({
  value,
  label,
  shouldFormat,
}: DataCardProps) => {
  return (
    <Card>
      <CardHeader className="flex flex-row items-center justify-between
space-y-0 pb-2">
        <CardTitle className="text-sm font-medium">
          {label}
        </CardTitle>
      </CardHeader>
      <CardContent>
        <div className="text-2xl font-bold">

```

```

        {shouldFormat ? formatPrice(value) : value}
      </div>
    </CardContent>
  </Card>
)
}

```

Рисунок 2.13. Функція аналітики курсів

Де функціональний компонент `DataCard` приймає `value`, `label` і `shouldFormat` як пропси та використовується `JSX` для побудови компонента картки. У блоці вмісту (`CardContent`) компонент відображає значення. Якщо `shouldFormat` встановлено в `true`, значення форматовуватиметься за допомогою функції `formatPrice`. В іншому випадку відображається необроблене значення.

Для відкладання оновлення значення змінної було зроблено кастомний хук React під назвою `useDebounce` (рис 2.14.). Це корисно в ситуаціях коли непотрібно, щоб дія виконувалась надто часто при швидких змінах значення, наприклад, під час введення тексту користувачем. Можна зменшити кількість викликів функцій або запитів, виконуваних після кожної зміни значення, та дочекатися стабілізації введення [41].

```
import { useEffect, useState } from "react";
export function useDebounce<T>(value: T, delay?:number): T {
  const [debouncedValue, setDebouncedValue] = useState<T>(value);

  useEffect(() => {
    const timer = setTimeout(() => {
      setDebouncedValue(value)
    }, delay || 500);
    return () => {
      clearTimeout(timer);
    }
  }, [value, delay]);
  return debouncedValue;
};
```

Рисунок 2.14. Хук “useDebounce”

У цьому прикладі, запит на сервер буде виконано тільки після того, як користувач не змінював текст протягом 500 мс. Це зменшує кількість запитів і покращує продуктивність додатку.

Не менш важливим є створення маршрутизації файлів для обробки завантаження файлів, за допомогою бібліотеки uploadthing(рис 2.15.).

```
const f = createUploadthing();
```

```

const handleAuth = () => {
  const { userId } = auth();
  if (!userId) throw new Error("Unauthorized");
  return { userId };
}
export const ourFileRouter = {
  courseImage: f({ image: { maxFileSize: "4MB", maxFileCount: 1 } })
    .middleware(() => handleAuth())
    .onUploadComplete(() => {}),
  courseAttachment: f(["text", "image", "video", "audio", "pdf"])
    .middleware(() => handleAuth())
    .onUploadComplete(() => {}),
  chapterVideo: f({ video: { maxFileCount: 1, maxFileSize: "512GB" } })
    .middleware(() => handleAuth())
    .onUploadComplete(() => {}))
} satisfies FileRouter;

```

Рисунок 2.15. Core “uploadthing”

ourFileRouter визначає різні типи файлів, які можна завантажувати, та налаштовує обробку завантаження для кожного типу:

courseImage: дозволяє завантаження зображень розміром до 4MB і кількістю до 1 файлу.

courseAttachment: дозволяє завантаження різних типів файлів, включаючи текстові файли, зображення, відео, аудіо та PDF.

chapterVideo: дозволяє завантаження відеофайлів розміром до 512GB і кількістю до 1 файлу.

Також middleware(() => handleAuth()) забезпечує, що користувач автентифікований перед завантаженням файлу. Тобто, користувачі можуть завантажувати зображення, вкладення та відео, і всі завантаження перевіряються на автентифікацію перед обробкою.

Щодо корисного функціоналу для звичайного користувача можемо виділити відстеження прогресу проходження курсу(рис 2.16.).

```

const Progress = React.forwardRef<
  React.ElementRef<typeof ProgressPrimitive.Root>,
  CombinedProgressProps
>(({ className, value, variant, ...props }, ref) => (
  <ProgressPrimitive.Root
    ref={ref}
    className={cn(
      "relative h-4 w-full overflow-hidden rounded-full bg-secondary",
      className
    )}
  >

```

```

    ))
    {...props}
  >
    <ProgressPrimitive.Indicator
      className={cn(progressVariants({ variant })))}
      style={{ transform: `translateX(-${100 - (value || 0)}%)` }}
    />
  </ProgressPrimitive.Root>
))

```

Рисунок 2.16. Прогрес користувача

Цей код визначає компонент Progress, який є функціональним компонентом, що використовує React.forwardRef для передачі посилання на DOM-елемент. Він використовує ProgressPrimitive.Root з Radix UI для базового відображення прогресу та використовує варіанти стилів з progressVariants в залежності від варіанту.

## Висновки до розділу 2

У другому розділі "Проектування веб-додатку онлайн курсів" було ретельно розглянуто та визначено ключові етапи проектування та реалізації системи, спрямованої на надання онлайн курсів. Починаючи з аналізу варіантів використання та створення об'єктно-орієнтованої структури, були виявлені та визначені основні функції та можливості системи. Серед них - управління курсами, категоріями, реєстрація та автентифікація користувачів, що є важливими складовими успішного функціонування платформи.

Детально була розглянута структура даних, що використовується в системі, включаючи моделі для курсів, категорій, додатків, розділів, прогресу користувачів та покупок. Ця структура дозволяє ефективно керувати інформацією, забезпечуючи необхідну функціональність та зручність взаємодії з системою.

Далі у розділі було проведено проектування та реалізацію алгоритмів роботи системи, включаючи розробку логіки обробки запитів користувачів, валідацію даних та забезпечення безпеки операцій. Цей етап важливий для забезпечення правильної та безперебійної роботи системи, а також захисту від можливих загроз та помилок.

На завершення розділу було проведено реалізацію функціоналу веб-додатку онлайн курсів, що включало в себе розробку UI/UX елементів, функціональність реєстрації, автентифікації, управління курсами, перегляд вмісту та здійснення покупок. Цей етап дозволив створити зручний та інтуїтивно зрозумілий інтерфейс для користувачів, що сприяє позитивному досвіду використання платформи.

У цілому, розділ 2 надає повну картину проектування та реалізації веб-додатку для онлайн курсів, враховуючи всі аспекти від визначення вимог до завершення реалізації функціоналу. Цей етап є важливим у створенні успішного та функціонального продукту, який відповідає потребам користувачів та вимогам сучасного онлайн-навчання.

## РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ВЕБ-ДОДАТКОМ ОНЛАЙН КУРСІВ

### 3.1 Порядок встановлення та налаштування системи

Встановлення та налаштування платформи для онлайн курсів вимагає комплексної роботи над усіма її компонентами та налаштуванням як окремих елементів, так і загальної системи в цілому. Це включає в себе встановлення необхідних бібліотек та інструментів, налаштування бази даних, конфігурацію сервера та інші кроки.

Для опису архітектури платформи було створено діаграму розгортання, яка відображає взаємозв'язки між усіма компонентами системи. Ця діаграма дозволяє візуалізувати та аналізувати архітектуру платформи, виявляючи можливі проблеми та оптимізуючи її роботу(рис 3.1.) [42].

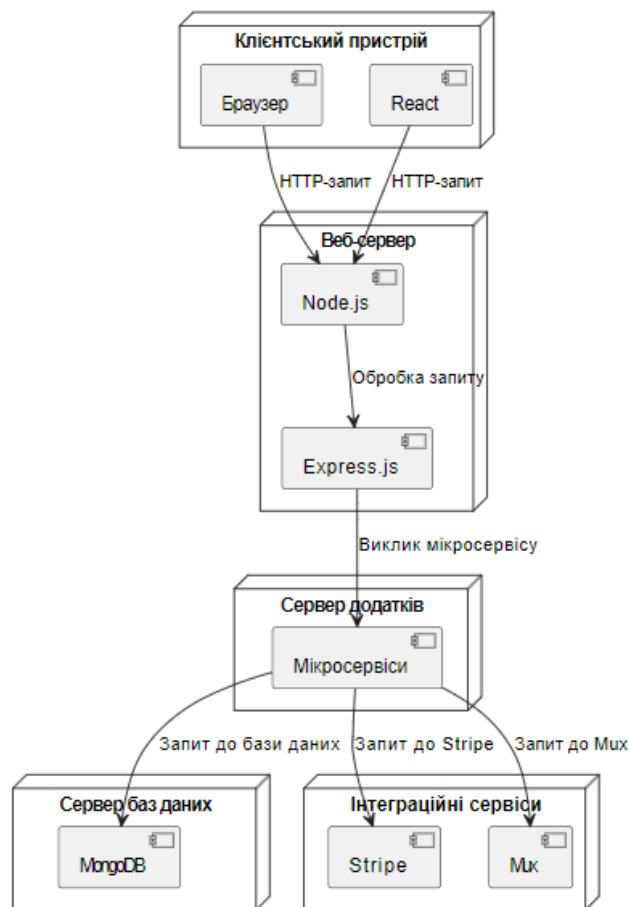


Рисунок 3.1. Діаграма розгортання

Опис діаграми:

Клієнтський пристрій: користувач взаємодіє з веб-додатком через браузер, використовуючи фронтенд, розроблений на React. Цей фронтенд забезпечує зручну та інтуїтивно зрозумілу інтерфейс для користувачів, дозволяючи їм взаємодіяти з системою.

Веб-сервер: Веб-сервер, побудований на Node.js, використовує Express.js для обробки HTTP-запитів. Це дозволяє забезпечити стабільну та швидку роботу сервера, що є важливим для успішного функціонування онлайн курсів.

Сервер додатків: Сервер додатків, що складається з мікросервісів, відповідає за бізнес-логіку системи. Ці мікросервіси забезпечують необхідні функції, такі як обробка запитів, зберігання даних та інші.

Сервер баз даних: Сервер додатків звертається до MongoDB для зберігання та отримання даних. Це дозволяє забезпечити ефективне та стабільне зберігання даних, що є важливим для успішного функціонування системи.

Інтеграційні сервіси: Веб-додаток інтегрується з сервісами Stripe для обробки платежів та Mux для роботи з відео. Ці інтеграції дозволяють забезпечити необхідні функції, такі як обробка платежів та відображення відео, що є важливим для успішного функціонування онлайн курсів.

Таким чином, веб-додаток для онлайн курсів складається з різних компонентів, які працюють разом для забезпечення стабільної та ефективної роботи. Клієнтський пристрій, веб-сервер, сервер додатків, сервер баз даних та інтеграційні сервіси всі разом забезпечують успішне функціонування системи.

### 3.2 Структура інтерфейсу веб-орієнтованої системою онлайн курсів

Переходячи до огляду, розглянемо різні аспекти інтерфейсу та робочого процесу нашого веб-додатку, щоб краще зрозуміти, як користувачі взаємодіють з системою. Було додано короткі описи кожного з скріншотів,

підкреслюючи ключові особливості та функціональні можливості програмного продукту.

Взаємодія користувачів відбувається через веб-додаток. На головній сторінці якого користувач може побачити логотип сайту, бічну панель з лівого боку веб-сторінки, яка містить зручні елементи навігації для швидкого доступу до певних функцій та розділів(рис3.2.).

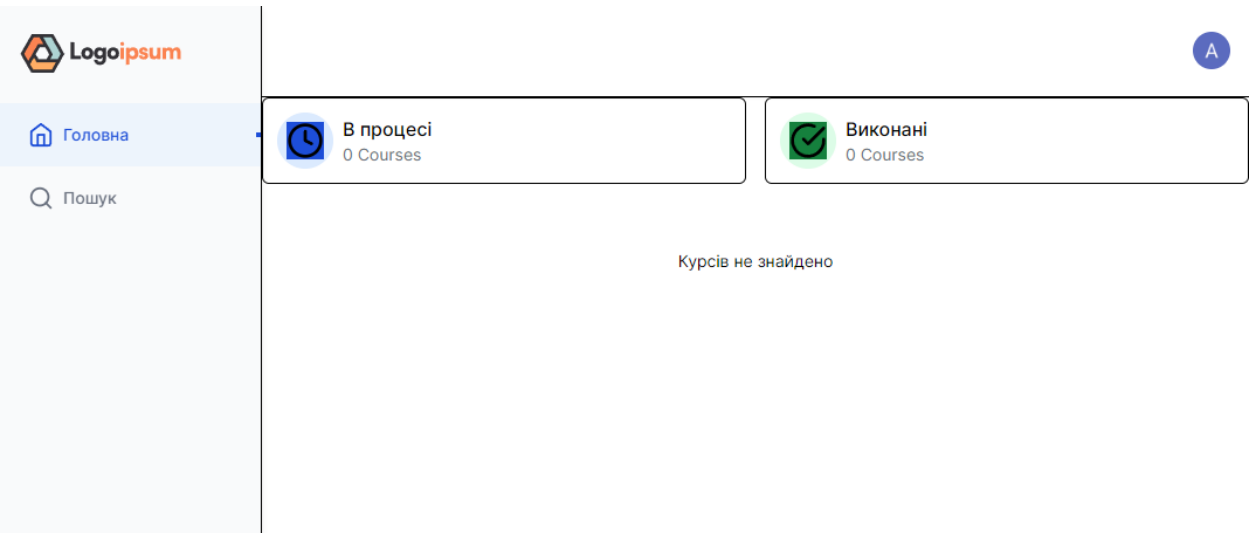


Рисунок 3.2. Головна сторінка додатку

Користувач має можливість авторизуватись в додатку за допомогою вказання пошти або через гугл-аккаунт(рис3.3.).

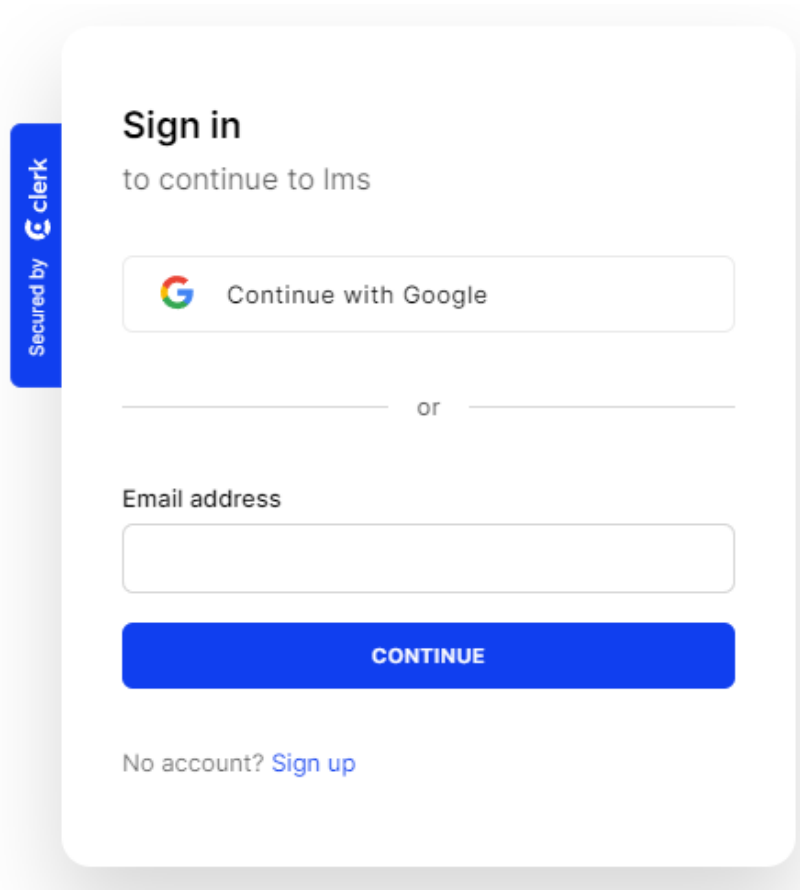


Рисунок 3.3. Авторизація в системі

Розглянемо роль звичайного користувача, який здійснив вхід в систему та не має прав адміністратора. Користувачу на сторінці пошук курсів відображаються всі опубліковані на даний момент курси які доступні для купівлі з подальшим проходженням. (рис3.4.).

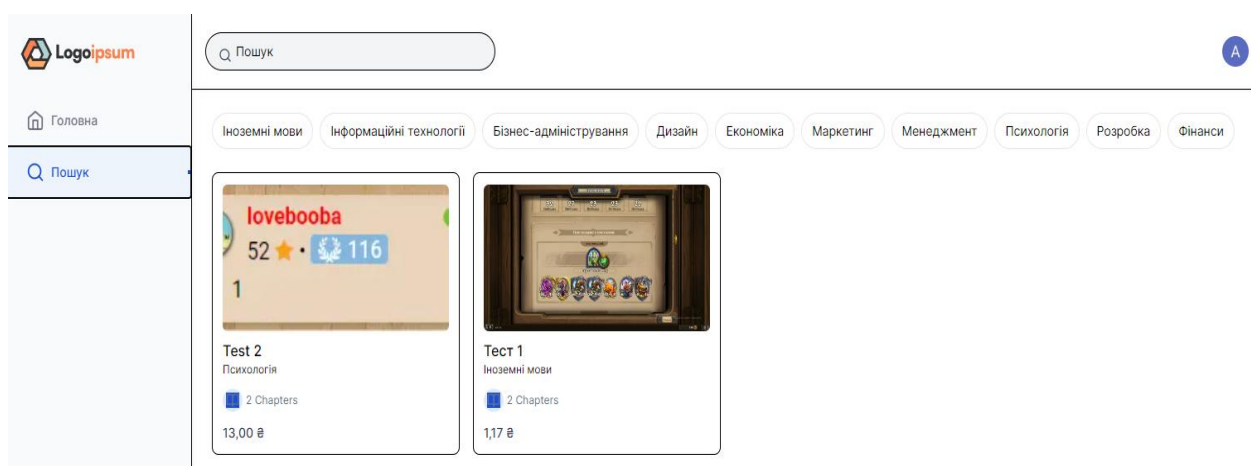


Рисунок 3.4. Сторінка пошуку курсів

На сторінці доступне пошукове меню (рис. 3.5.), що дозволяє користувачам швидко та зручно шукати необхідні курси, наприклад по категоріям або словам в назві(рис. 3.6. – 3.7.). Також можливо комбінувати

обираючи спочатку категорію, а потім коли фільтр застосувався, додатково обирати по назві.



Рисунок 3.5. Пошукове меню

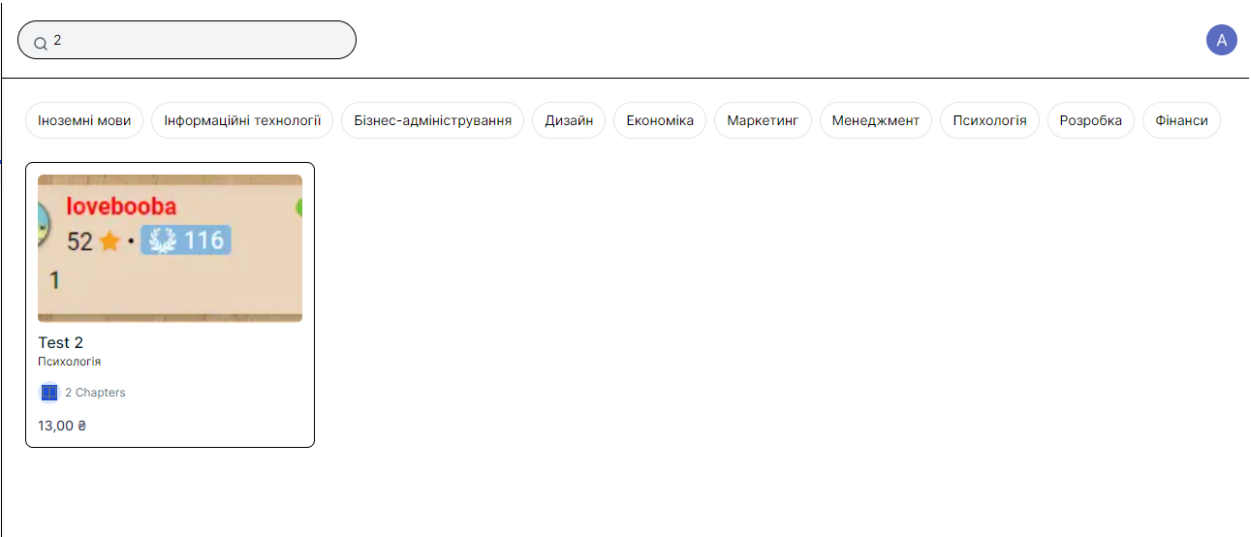


Рисунок 3.6. Пошук по назві

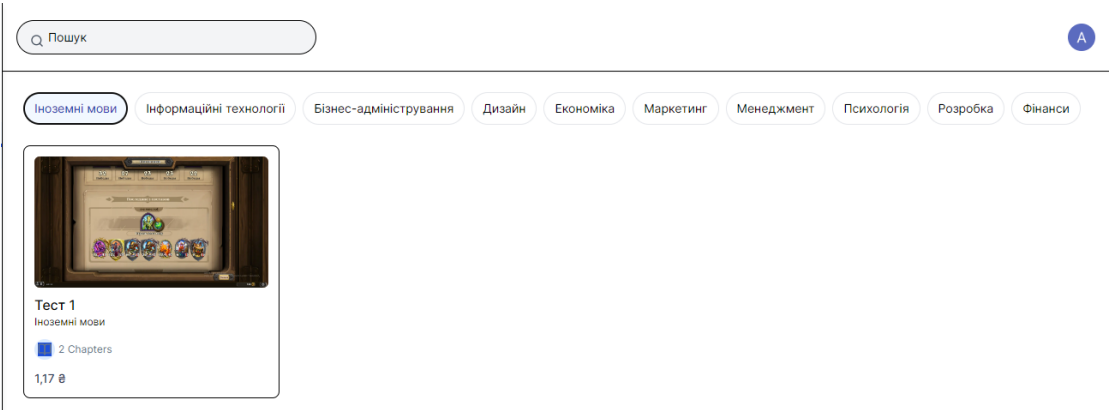


Рисунок 3.7. Пошук по категоріям

Після пошуку та обрання курсу який користувач має намір пройти, відбувається перехід на сторінку самого курсу, де відображаються розділи та їхня доступність, назва та опис. Наприклад, якщо курс платний, доступні тільки ті розділи, що є введенням, який задумувався як коротке інтерв'ю з описом або відеоматеріалом про проходження курсу, інші розділи закриті для перегляду(рис.3.8. – 3.9.).



номер кредитної картки, строк дії та код CVV. Після введення та підтвердження платіжної інформації, система обробляє транзакцію, і у випадку успішної оплати курс розблоковується, надаючи користувачеві повний доступ до всіх матеріалів курсу(рис. 3.10.). Після успішної оплати відбувається автоматичний перехід до розблокованого курсу, для подальшого його проходження.

← Lms TEST MODE

Test 3

123,00 грн.

Test3

На платформе stripe | Условия | Конфиденциальность

Оплата с link

Или оплатить картой

Эл. почта ipz203\_shos@student.ztu.edu.ua

Данные карты

1234 1234 1234 1234

VISA

MM / ГГ

Код CVV/CVC

Имя владельца карты

Имя, фамилия

Страна или регион

Украина

☐ Надежно сохранить мои данные для оформления платежа одним щелчком

Выполняйте оплату быстрее в Lms и везде, где принимают Link.

Оплатить

Рисунок 3.10. Сторінка оплати

Після здійснення оплати всі розділи курсу стають доступними користувачу(рис. 3.11.).



права доступу. Так само відбувається і перехід в звичайний режим(рис. 3.13. – рис. 3.14.)

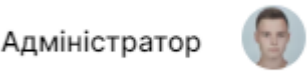


Рисунок 3.13. Перехід до режиму адміністратора

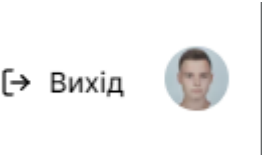


Рисунок 3.14. Перехід в звичайний режим

Якщо все виконано вірно і ми перейшли на сторінку адміністратора, можна побачити що бічна панель змінила свої навігаційні посилання на Курси та Аналітика(рис. 3.15.).

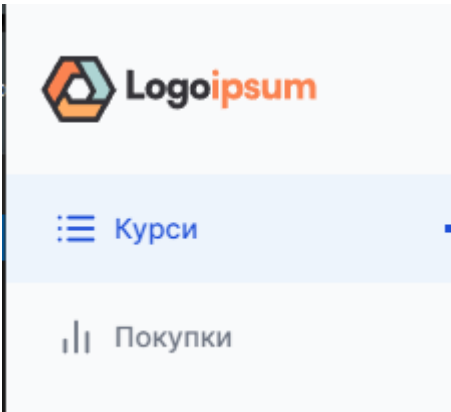


Рисунок 3.15. Навігаційна панель адміністратора

Розглянемо сторінку Курси, де можна створити новий курс, побачити загальну інформацію про кожен курс, де відображається назва, ціна, статус(чи є курс вже опублікованим та має доступ до перегляду або проходження його всіма користувачами додатку, чи ні), також присутня пагінація переходу між сторінками курсів, якщо їх велика кількість (рис. 3.16.).

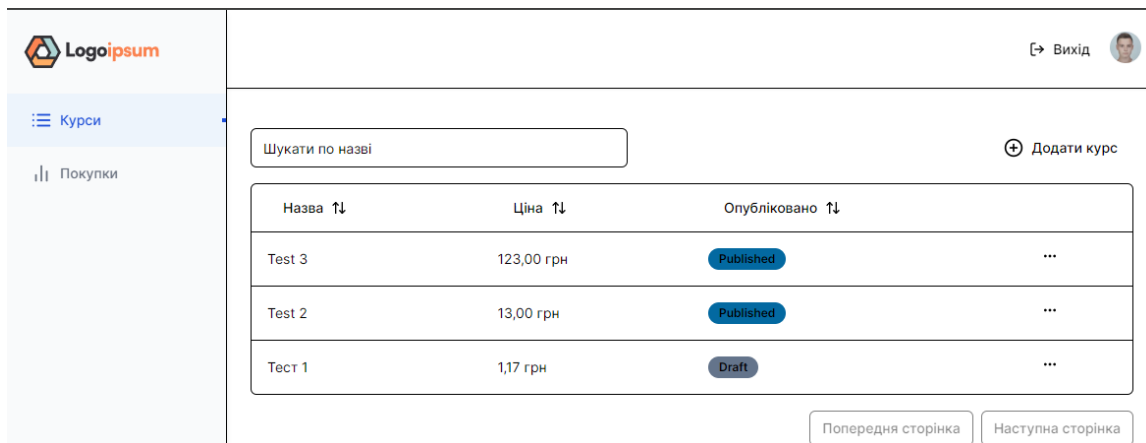


Рисунок 3.16. Сторінка Курси в режимі адміністратор

На цій же сторінці є змога користувачу здійснити фільтрування курсів по вище згаданим критеріям(рис. 3.17. – 3.18.).

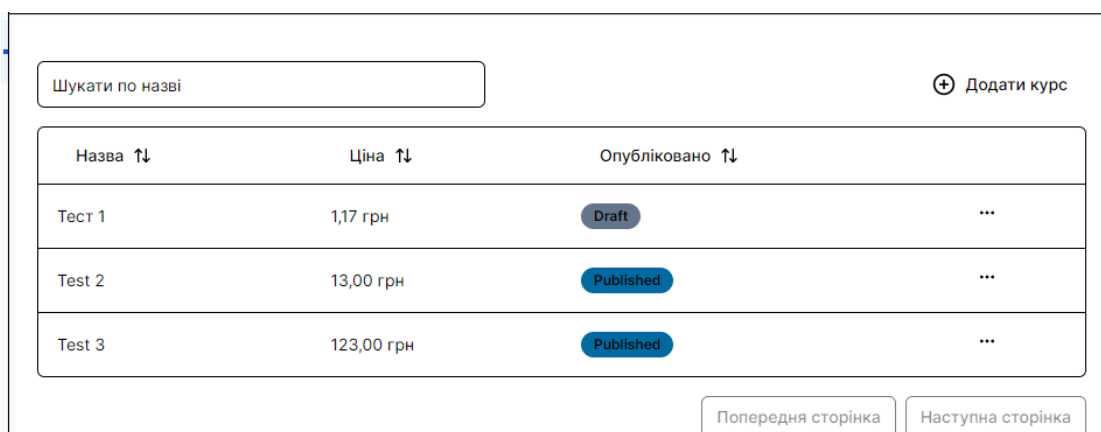


Рисунок 3.17. Фільтрування курсів по ціні

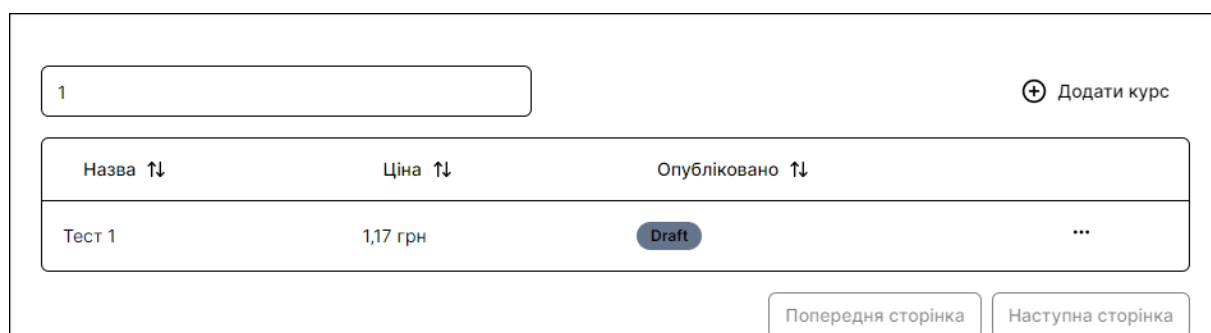


Рисунок 3.18. Пошук курсів по назві

Тепер є можливість створити новий курс. Для цього адміністратору потрібно натиснути на кнопку додати курс та у формі що відкрилась вказати лише назву, яку в подальшому буде змога змінити(рис. 3.19.).

# Назвіть курс

Як ви хочете назвати свій курс?

Назва курсу

Введіть назву

Назва повинна відповідати тематиці курсу

Відміна

Продовжити

Рисунок 3.19. Форма створення нового курсу

Одразу після створення перекидає на сторінку самого курсу де потрібно вести додаткову інформацію про курс. Користувач може змінити назву, додати опис, загальне фото, обрати категорію до якого буде відноситись курс, вказати ціну, додати розділи та додаткові ресурси для проходження. Також присутній банер що свідчить про те, що курс не опубліковано та не видно іншим користувачам (рис. 3.20.).

The screenshot shows the 'Logoipsum' interface for configuring a course. A yellow banner at the top states: 'Курс не опубліковано та не видно користувачам'. The main section is titled 'Налаштування курсу' (Course Settings) with a 'Publish' button. The settings are organized into two columns:

- Відредагуй курс** (Edit course):
  - Назва курсу** (Course name): 'Джава скрипт' (JavaScript). Action: 'Змінити назву' (Change name).
  - Опис** (Description): 'Не має опису' (No description). Action: 'Змінити опис' (Change description).
  - Логотип курсу** (Course logo): 'Додати фото' (Add photo).
  - Обрати категорію** (Select category): 'не обрано категорію' (no category selected). Action: 'Змінити категорію' (Change category).
- Розділи курсу** (Course sections):
  - Розділи курсу** (Course sections): 'Додати розділ' (Add section). Subtext: 'Перетягуйте для зміни порядку розділів' (Drag to change section order).
  - Ціна курсу** (Course price): 'Не вказано' (Not specified). Action: 'Змінити ціну' (Change price).
  - Додаткові ресурси** (Additional resources): 'Додати файл' (Add file). Subtext: 'Не має додаткових ресурсів' (No additional resources).

Рисунок 3.20. Сторінка наповнення

Розглянемо більш детально як відбуваються зміни на цій сторінці. Наприклад, для додавання фото користувач повинен натиснути на відповідну кнопку редагування, обрати або перетягнути фото для додавання та погодити свої дії, після чого відбувається завантаження та картинка відображається на сторінці(рис. 3.21. – рис. 3.25.).

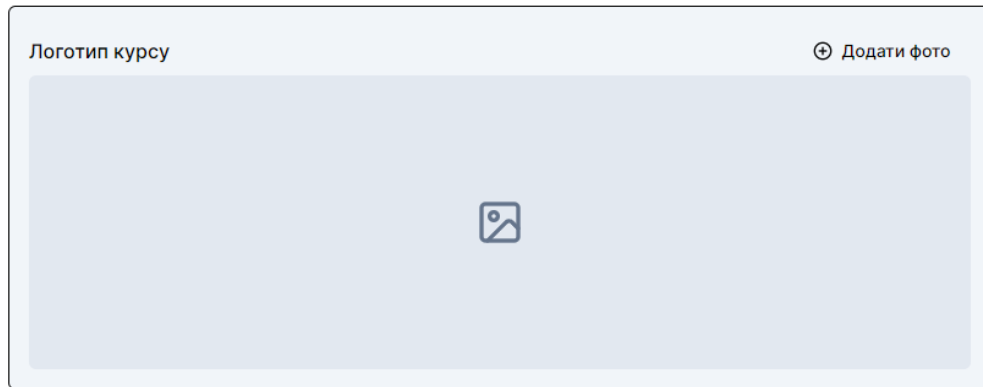


Рисунок 3.21. Завантаження фото(1-етап)

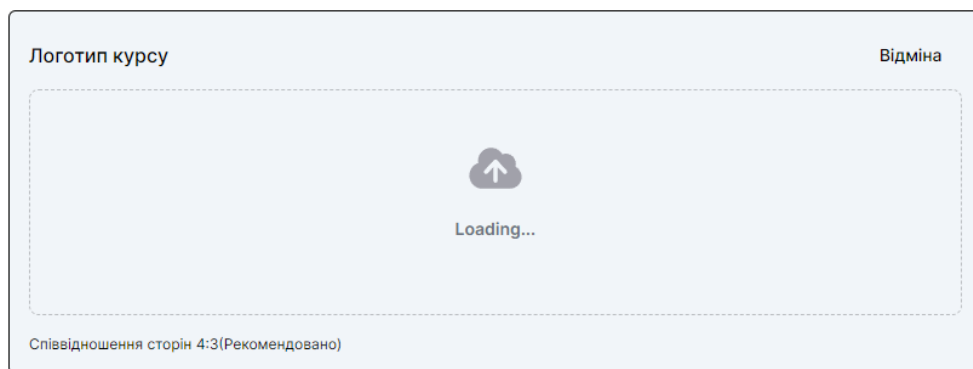


Рисунок 3.22. Завантаження фото(2-етап)

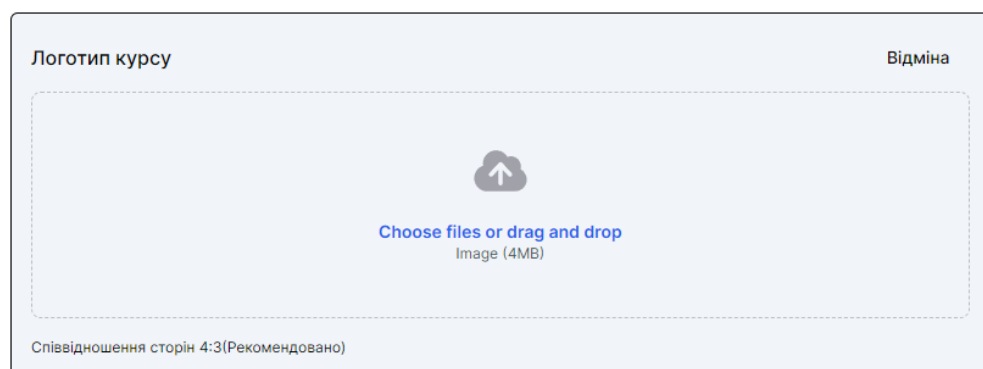


Рисунок 3.23. Завантаження фото(3-етап)

Логотип курсу

Відміна



Choose files or drag and drop

Image (4MB)

Upload 1 file

Співвідношення сторін 4:3(Рекомендовано)

Рисунок 3.24. Завантаження фото(4-етап)

Логотип курсу

Змінити фото

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

Рисунок 3.25. Завантаження фото(5-етап)

Вибір категорії до якої буде відноситись курс продемонстровано на рисунку 3.26.

Економіка

Маркетинг

Менеджмент

✓ Психологія

Розробка

Фінанси

Відміна

Обрати категорію

Психологія

Зберегти

Рисунок 3.26. Вибір категорії

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Адміністратор може вказати ціну на покупку курсу (рис. 3.27. – 3.28.)

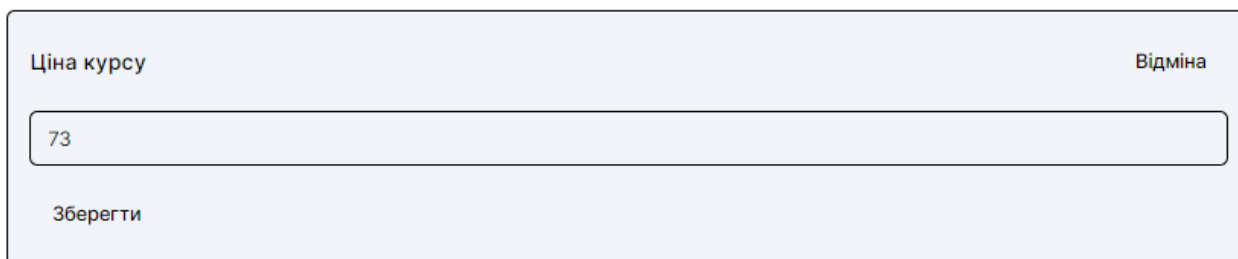


Рисунок 3.27. Додавання вартості курсу(1-етап)

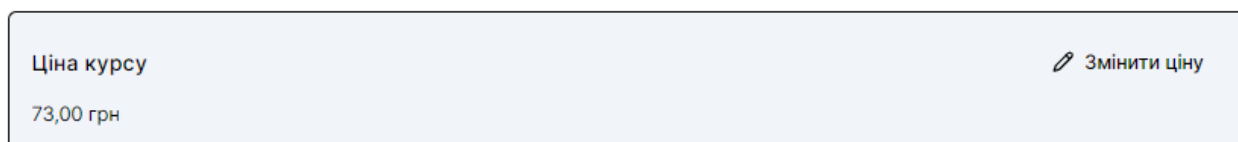


Рисунок 3.28. Додавання вартості курсу(2-етап)

Також є змога додати додаткові ресурси для проходження курсу, наприклад файл (рис. 3.29.).

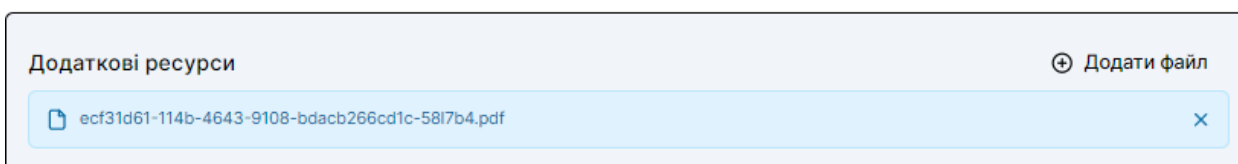


Рисунок 3.29. Відображення додаткових ресурсів

Тепер перейдемо до додавання категорій, це схоже зі створення курсу, де потрібно просто натиснути додати розділ та вказати назву(рис. 3.30. – 3.31.).

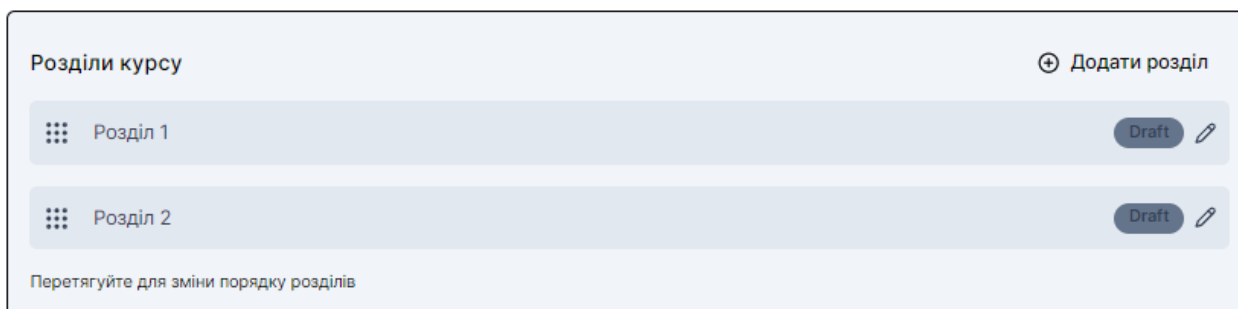


Рисунок 3.30. Поле розділи

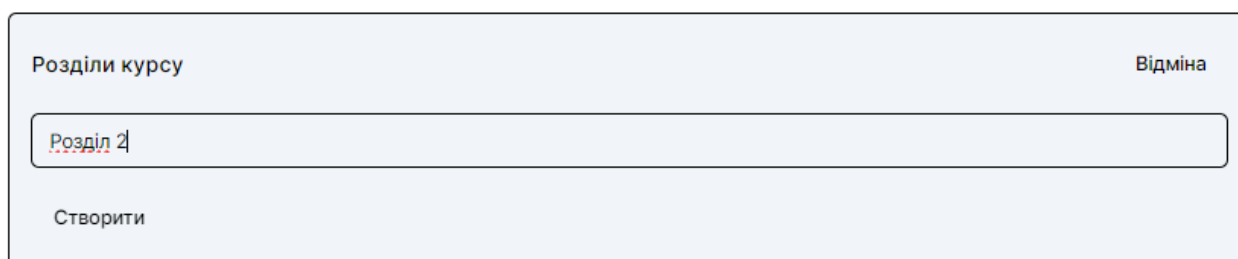


Рисунок 3.31. Додавання розділу

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

У адміністратора є можливість міняти місцями розділи звичайним перетягуванням їх курсором мишки(рис. 3.32. – 3.33.).

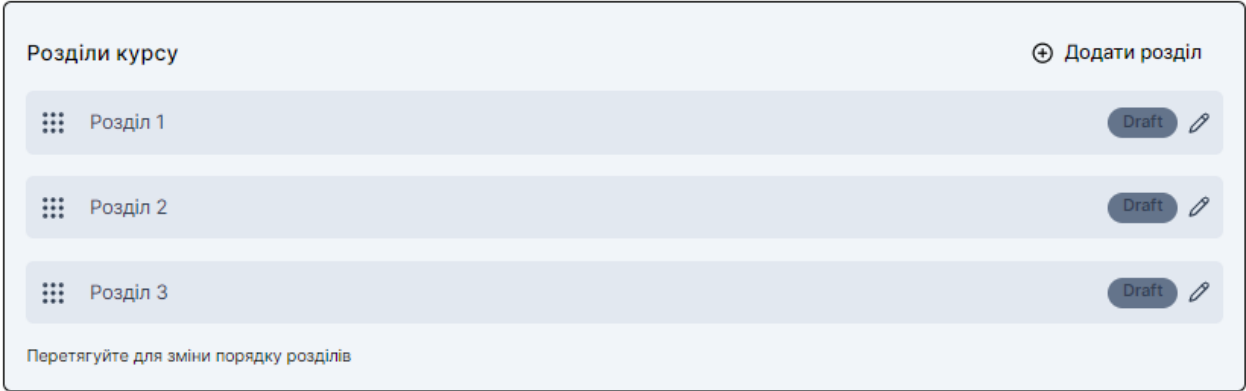


Рисунок 3.32. До зміни порядку розділів

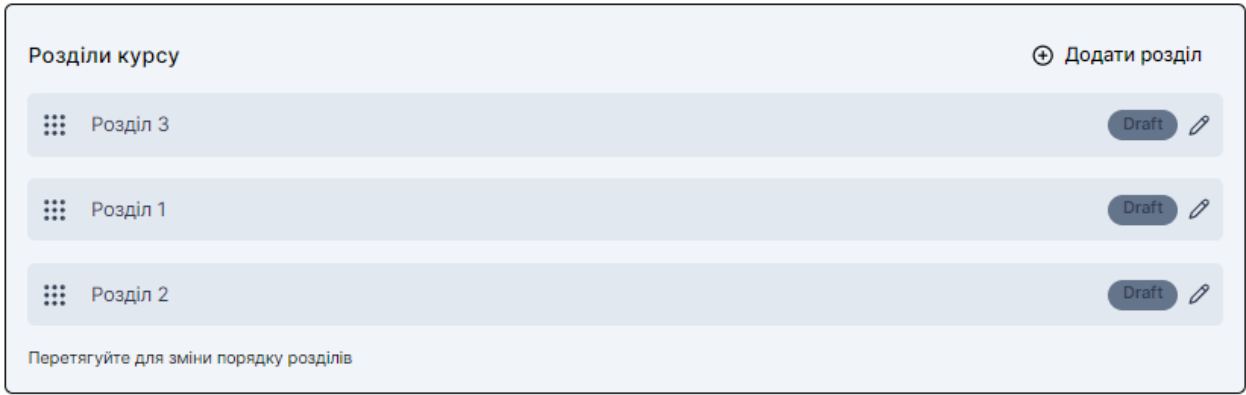


Рисунок 3.33. Після зміни порядку розділів

Після створення розділу адміністратор повинен здійснити його наповнення інформацією при переході на його редагування або видалення(рис. 3.34.).

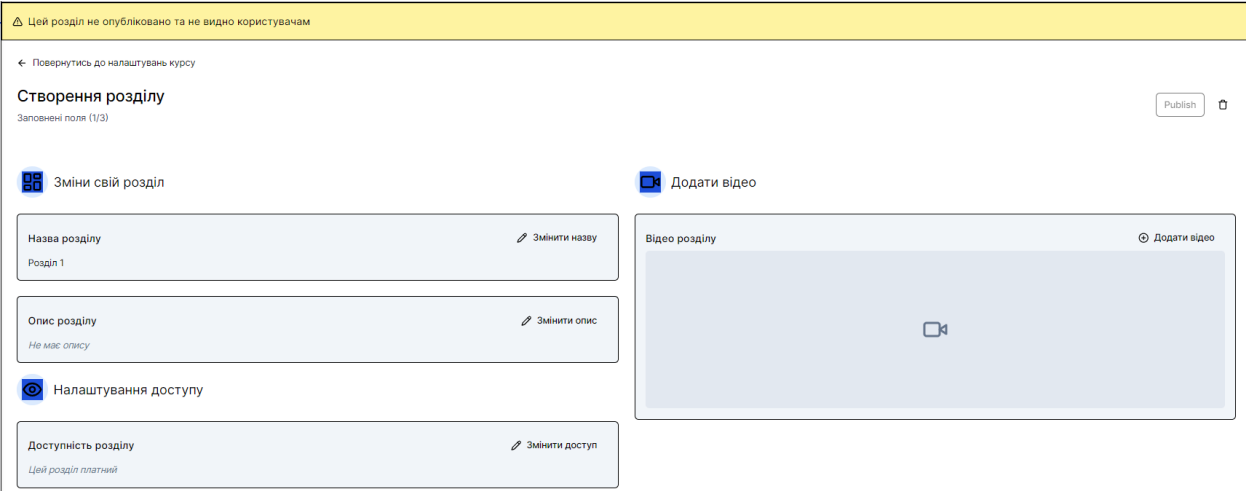


Рисунок 3.34. Сторінка розділу

Можливе оновлення назви, опису, відео, що схоже на додавання фотографії до курсу, також потрібно обрати чи цей розділ буде доступний до безкоштовної демонстрації звичайним користувачам.

Під час заповнення всіх цих полів відбувається оновлення сторінки де збільшується лічильник обов'язкових полів, який потрібен для того аби розділ став доступним для публікації(рис. 3.35. – 3.36.).

## Створення розділу

Заповнені поля (1/3)

Рисунок 3.35. Оновлення лічильника(1 – етап)

## Створення розділу

Заповнені поля (2/3)

Рисунок 3.36. Оновлення лічильника(2 – етап)

Після того як адміністратор заповнив курс всією потрібною інформацією, він має доступ до публікації цього розділу, кнопка стає активною та курс змінює свій статус який свідчить про те, що все коректно та буде відображено користувачам в процесі проходження(рис. 3.37.).



Розділи курсу

Розділи курсу

⊕ Додати розділ

|   |          |           |
|---|----------|-----------|
| ⋮ | Розділ 1 | Published |
| ⋮ | Розділ 2 | Draft     |
| ⋮ | Розділ 3 | Draft     |

Перетягуйте для зміни порядку розділів

Рисунок 3.37. Опубліковані розділи

Всі операції створення та редагування категорії схожі до операцій створення курсу.

Тепер повертаючись до головної сторінки адмін-панелі видно які курси опубліковані, а які ні(рис. 3.38.).

| Назва <sup>1</sup> | Ціна <sup>1</sup> | Опубліковано <sup>1</sup> |     |
|--------------------|-------------------|---------------------------|-----|
| Test 3             | 123,00 грн        | <span>published</span>    | ... |
| Test 2             | 13,00 грн         | <span>published</span>    | ... |
| Джава скрипт       | 73,00 грн         | <span>draft</span>        | ... |
| Тест 1             | 1,17 грн          | <span>draft</span>        | ... |

Попередня сторінка
Наступна сторінка

Рисунок 3.38. Опубліковані курси

У адміністратора є можливість переглянути статистику до всіх курсам, в них відображається загальна статистика, кількість продажів, загальна кількість отриманих коштів з продажів, це дозволяє відслідковувати динаміку(рис. 3.39.).

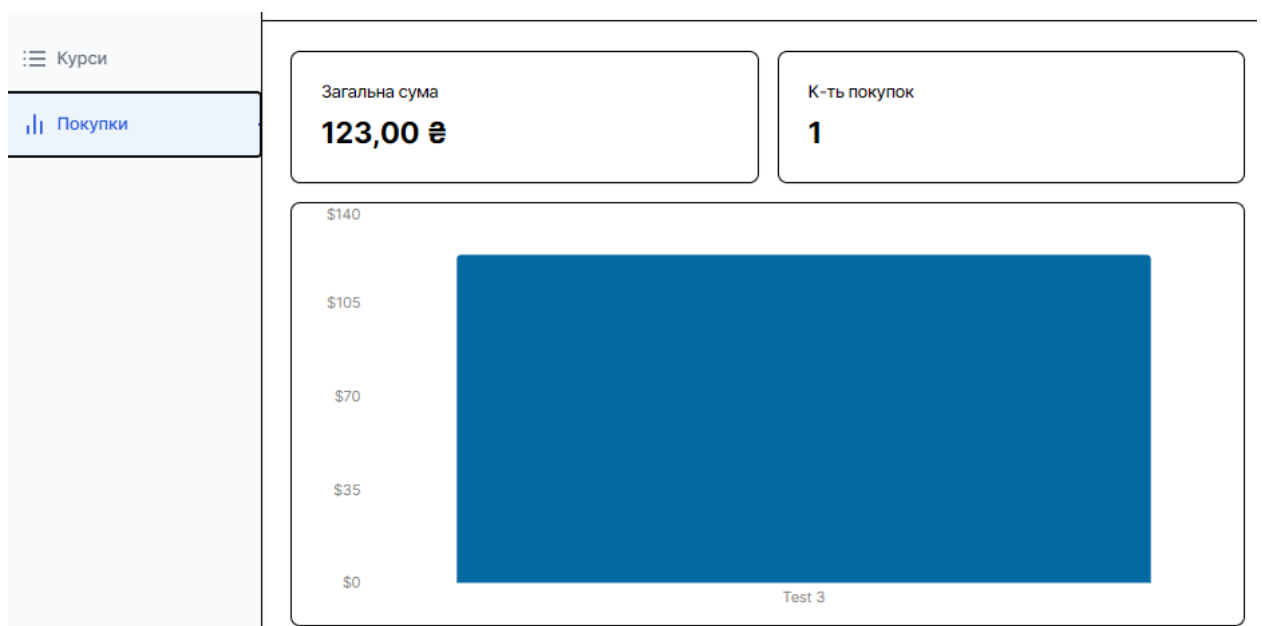


Рисунок 3.39. Статистика адміністратора

### 3.3 Тестування роботи веб-додатку

Тестування програмного продукту є важливою частиною проекту, що забезпечує коректну роботу та відповідність вимогам. Тестування веб-додатку включає перевірку низку аспектів, таких як функціональність, продуктивність, безпеку чи зручне використання. Це допомагає виявити помилки, недоліки або слабкі місця програмного продукту, що сприяє якості та задоволенню в користуванні.

Існують основні методи тестування, наприклад, функціональне тестування, що передбачає перевірку всіх функцій та можливостей додатку,

для перевірки коректності їх роботи. Тестування продуктивності включає в себе тест часу відгуку, стабільності та здатності обробляти великі обсяги даних. Тестування зручності використання здійснює перевірку навігації, доступність та загальний досвід користувача.

Розглянемо приклад функціонального тестування нашого веб-додатку для онлайн курсів. Однією з основних функцій додатку є реєстрація нового користувача, отже ми проведемо наступні кроки для тестування цієї функції.

Підготовка тестових даних: Створимо кілька наборів даних для тестування різних сценаріїв реєстрації (правильні дані, неправильні дані, вже зареєстрований користувач тощо).

Виконання тесту: Виконуємо спроби реєстрації з використанням підготовлених тестових даних.

Аналіз результатів: Перевіримо, чи система правильно обробляє кожен сценарій. Наприклад, для правильних даних користувач повинен бути успішно зареєстрований, для неправильних даних система повинна видати відповідні повідомлення про помилки.

Результати тестування показали, що система успішно обробляє всі тестові сценарії. При введенні правильних даних користувач успішно реєструється, при введенні неправильних даних система видає коректні повідомлення про помилки, а при спробі реєстрації вже зареєстрованого користувача видається відповідне повідомлення.

Отже, функціональне тестування реєстрації користувачів у веб-додатку для онлайн курсів показало, що ця функція працює належним чином і відповідає встановленим вимогам.

## Висновки до третього розділу

У цьому розділі було представлено схему розгортання платформи, що дозволило розглянути кожен структурний елемент системи та проаналізувати їх взаємодію. Були описані компоненти платформи, а його зв'язки з іншими

частинами системи були продемонстровані. Це забезпечило глибоке розуміння архітектури системи та взаємозв'язків між її різними складовими.

Крім того, було надано детальний опис інтерфейсу розробленого програмного продукту. Були розглянуті всі функціональні можливості додатку, включаючи особливості для різних типів користувачів. Для кожної ролі, будь то звичайний користувач або адміністратор, були визначені свої унікальні функції та можливості в системі. Для успішного використання системи користувачі повинні знати, які функції та можливості вони мають доступ до, та як використовувати їх. Це дозволяє користувачам ефективно використовувати систему та досягти своїх цілей.

Окрім цього, було продемонстровано, як працює статистика для адміністратора, що коректно відображаються діаграми та статистичні дані.

## ВИСНОВКИ

У процесі розробки дипломного проекту було проведено всебічний аналіз існуючих рішень на ринку онлайн-курсів, включаючи Udemy, Coursera, edX та LinkedIn Learning, що дозволило визначити ключові функціональні вимоги та сильні сторони для поліпшення власного продукту. Обрано оптимальну архітектуру з компонентно-орієнтованим підходом, що забезпечує масштабованість, гнучкість та продуктивність системи.

Для розробки додатку обрано інструментальні засоби, такі як Next.js, React, Stripe, Mux, Prisma, Tailwind CSS та MongoDB, які були вибрані з огляду на їх переваги та відповідність потребам проекту.

Розроблено основні функції системи, такі як управління курсами, категоріями, реєстрація та автентифікація користувачів, що забезпечують її успішне функціонування. Розроблено структуру даних, що включає моделі курсів, категорій, додатків, розділів, прогресу користувачів та покупок, що дозволяє ефективно керувати інформацією.

Також було здійснено проектування та реалізацію алгоритмів роботи системи, що включає обробку запитів користувачів, валідацію даних та забезпечення безпеки операцій. Розроблено функціонал веб-додатку, включаючи UI/UX елементи, функціональність реєстрації, автентифікації, управління курсами, перегляд вмісту та здійснення покупок, що сприяє зручності використання платформи.

Представлено схему розгортання платформи, описано взаємодію компонентів системи та надано детальний опис інтерфейсу для різних типів користувачів. Включено можливості системи для адміністраторів, зокрема робота зі статистикою та діаграмами.

Проведене дослідження та розробка веб-додатку для онлайн-курсів демонструють, що обрані підходи, технології та інструменти забезпечують успішну реалізацію проекту. Використання веб-технологій для навчання програмуванню сприяє покращенню якості освіти та задовольняє потреби сучасного ринку праці.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Udemу [Електронний ресурс] – Режим доступу до ресурсу: <https://about.udemy.com/>
2. Документація Coursera [Електронний ресурс] – Режим доступу до ресурсу: <https://www.coursera.org/>
3. Документація edX. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.edx.org/>
4. Документація LinkedIn [Електронний ресурс] – Режим доступу до ресурсу: <https://ua.linkedin.com/learning/>
5. Документація NestJS [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.nestjs.com/>
6. Server-side Rendering [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering>
7. Quick Start – React[Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/learn>
8. ReactDOM – React. [Електронний ресурс] – Режим доступу до ресурсу: <https://legacy.reactjs.org/docs/react-dom.html>
9. Thinking in React. [Електронний ресурс] – Режим доступу до ресурсу: <https://legacy.reactjs.org/docs/thinking-in-react.html>
10. Документація Stripe [Електронний ресурс] – Режим доступу до ресурсу: <https://www.67degrees.co.uk/blog/how-does-stripe-work-for-online-payments-the-ultimate-website-guide/>
11. Безпека Stripe. [Електронний ресурс] – Режим доступу до ресурсу: <https://stripe.com/resources/more/payment-security>
12. Документація Mux [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.mux.com/guides/video>
13. Документація Stream [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.mux.com/core/stream-video-files>

14. Чому Prisma це ORM? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.prisma.io/docs/orm/overview/introduction/why-prisma>
15. Документація Prisma [Електронний ресурс] – Режим доступу до ресурсу: <https://www.prisma.io/docs/orm/overview/prisma-in-your-stack/is-prisma-an-orm>
16. Документація Tailwind CSS [Електронний ресурс] – Режим доступу до ресурсу: <https://tailwindcss.com/>
17. Оптимізація продуктивності Tailwind CSS. [Електронний ресурс] – Режим доступу до ресурсу: <https://v2.tailwindcss.com/docs/optimizing-for-production>
18. Документація *MongoDB*. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/company/what-is-mongodb>
19. Операції агрегації в MongoDB [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/docs/manual/aggregation/>
20. Документація MongoDB Atlas[Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/docs/atlas/partner-integrations/>
21. Функціональні вимоги [Електронний ресурс] – Режим доступу до ресурсу: <https://visuresolutions.com/uk/блог/функціональні-вимоги/>
22. Нефункціональні вимоги: [Електронний ресурс] – Режим доступу до ресурсу: <https://visuresolutions.com/uk/блог/нефункціональні-вимоги/>
23. Діаграми UML [Електронний ресурс] – Режим доступу до ресурсу: <https://evergreens.com.ua/ua/articles/uml-diagrams.html>
24. Об'єктно-орієнтоване програмування [Електронний ресурс] – Режим доступу до ресурсу: <https://goit.global/ua/articles/shcho-take-ob-iektno-oriiientovane-prohramuvannia-pryntsypu-perevahy-ta-nedoliky/>
25. Архітектура програмного [Електронний ресурс] – Режим доступу до ресурсу: <https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya>

26. UML діаграми, їхні основні типи та процес розроблення[Електронний ресурс] – Режим доступу до ресурсу: <https://foxminded.ua/uml-diagramy/>
27. Prisma [Електронний ресурс] – Режим доступу до ресурсу: <https://www.prisma.io/docs/orm/prisma-schema/data-model/models>
28. Prisma Client [Електронний ресурс] – Режим доступу до ресурсу: <https://www.prisma.io/docs/orm/prisma-client/setup-and-configuration/generating-prisma-client>
29. Prisma Client | Prisma Documentation. Prisma | Simplify working and interacting with databases. URL: <https://www.prisma.io/docs/orm/prisma-client> (дата звернення: 03.06.2024).
30. CRUD [Електронний ресурс] – Режим доступу до ресурсу: <https://www.prisma.io/docs/orm/prisma-client/queries/crud>
31. Prisma Client with Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://www.prisma.io/docs/orm/more/help-and-troubleshooting/help-articles/nextjs-prisma-client-dev-practices#problem>
32. Filtering and Sorting [Електронний ресурс] – Режим доступу до ресурсу: <https://www.prisma.io/docs/orm/prisma-client/queries/filtering-and-sorting>
33. Що таке алгоритми[Електронний ресурс] – Режим доступу до ресурсу: <https://dan-it.com.ua/uk/blog/shho-take-algorytmy-kroky-pryklady-konstrukcziyi-myslennya/>
34. Машинне навчання [Електронний ресурс] – Режим доступу до ресурсу: <https://itproger.com/ua/news/sovremennye-podhody-k-mashinnomu-obucheniyu-v-veb-razrabotke>
35. Алгоритми [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/fundamentals-of-algorithms/>
36. IBM Integration [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ibm.com/docs/fi/integration-bus/10.0?topic=scratch-resource-management-overview>

37. Activity Diagram [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.mindonmap.com/uk/blog/uml-activity-diagram/>
38. Lucidchart [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.lucidchart.com/pages/uml-activity-diagram>
39. Sequence Diagrams [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.maxzosim.com/sequence-diagrams/>
40. Sequence Diagram [Електронний ресурс] – Режим доступу до ресурсу:  
<https://e5.ua/uk/blogpost-2/sequence-diagram-dosi-zhiva/>
41. use-debounce. [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.npmjs.com/package/use-debounce>
42. Діаграма розгортання [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.guru99.com/uk/deployment-diagram-uml-example.html>

# ДОДАТКИ

|  |  |  |  |  |                     |      |
|--|--|--|--|--|---------------------|------|
|  |  |  |  |  | ІІЗ.КР.Б-121-24-ІІЗ | Арк. |
|  |  |  |  |  |                     | 76   |
|  |  |  |  |  |                     |      |

```

const { PrismaClient } = require("@prisma/client");

const database = new PrismaClient();

async function main() { // список категорій
  try {
    await database.category.createMany({
      data: [
        { name: "Розробка" },
        { name: "Дизайн" },
        { name: "Іноземні мови" },
        { name: "Маркетинг" },
        { name: "Фінанси" },
        { name: "Менеджмент" },
        { name: "Психологія" },
        { name: "Економіка" },
        { name: "Бізнес-адміністрування" },
        { name: "Інформаційні технології" },
      ]
    });

    console.log("Success");
  } catch (error) {
    console.log("Error", error);
  } finally {
    await database.$disconnect();
  }
}

main();

export const getProgress = async ( // Отримання прогресу
  userId: string,
  courseId: string,
): Promise<number> => {
  try {
    const publishedChapters = await db.chapter.findMany({
      where: {
        courseId: courseId,
        isPublished: true,
      },
      select: {
        id: true,
      }
    });

    const publishedChapterIds = publishedChapters.map((chapter) => chapter.id);

    const validCompletedChapters = await db.userProgress.count({
      where: {
        userId: userId,

```

```

        chapterId: {
            in: publishedChapterIds,
        },
        isCompleted: true,
    }
});

const progressPercentage = (validCompletedChapters /
publishedChapterIds.length) * 100;

    return progressPercentage;
} catch (error) {
    console.log("[GET_PROGRESS]", error);
    return 0;
}
}

type CourseWithDetails = Course & { // Отримання таблиці курсів
    category: Category;
    chapters: Chapter[];
    progress: number | null;
};

type DashboardCoursesResult = {
    completedCourses: CourseWithDetails[];
    coursesInProgress: CourseWithDetails[];
};

export const fetchDashboardCourses = async (userId: string):
Promise<DashboardCoursesResult> => {
    try {
        const purchasedCourses = await db.purchase.findMany({
            where: { userId },
            select: {
                course: {
                    include: {
                        category: true,
                        chapters: { where: { isPublished: true } },
                    },
                },
            },
        });

        const coursesWithProgress = await Promise.all(
            purchasedCourses.map(async (purchase) => {
                const course = purchase.course as CourseWithDetails;
                course.progress = await getProgress(userId, course.id);
                return course;
            })
        );
    }
};

```

```

    const completedCourses = coursesWithProgress.filter((course) =>
course.progress === 100);
    const coursesInProgress = coursesWithProgress.filter((course) =>
(course.progress ?? 0) < 100);

    return {
      completedCourses,
      coursesInProgress,
    };
  } catch (error) {
    console.error("[GET_DASHBOARD_COURSES_ERROR]", error);
    return {
      completedCourses: [],
      coursesInProgress: [],
    };
  }
};

interface GetChapterProps { // Отримання розділів курсу
  userId: string;
  courseId: string;
  chapterId: string;
};

export const getChapter = async ({
  userId,
  courseId,
  chapterId,
}: GetChapterProps) => {
  try {
    const purchase = await db.purchase.findUnique({
      where: {
        userId_courseId: {
          userId,
          courseId,
        }
      }
    });

    const course = await db.course.findUnique({
      where: {
        isPublished: true,
        id: courseId,
      },
      select: {
        price: true,
      }
    });

    const chapter = await db.chapter.findUnique({
      where: {

```

```

        id: chapterId,
        isPublished: true,
    }
    });

    if (!chapter || !course) {
        throw new Error("Chapter or course not found");
    }

    let muxData = null;
    let attachments: Attachment[] = [];
    let nextChapter: Chapter | null = null;

    if (purchase) {
        attachments = await db.attachment.findMany({
            where: {
                courseId: courseId
            }
        });
    }

    if (chapter.isFree || purchase) {
        muxData = await db.muxData.findUnique({
            where: {
                chapterId: chapterId,
            }
        });
    }

    nextChapter = await db.chapter.findFirst({
        where: {
            courseId: courseId,
            isPublished: true,
            position: {
                gt: chapter?.position,
            }
        },
        orderBy: {
            position: "asc",
        }
    });
}

const userProgress = await db.userProgress.findFirst({
    where: {
        userId: userId,
        chapterId: chapterId,
    }
});

return {
    chapter,

```

```

        course,
        muxData,
        attachments,
        nextChapter,
        userProgress,
        purchase,
    };
} catch (error) {
    console.log("[GET_CHAPTER]", error);
    return {
        chapter: null,
        course: null,
        muxData: null,
        attachments: [],
        nextChapter: null,
        userProgress: null,
        purchase: null,
    }
}
}

export async function POST(req: Request) { // API-WEBHOOK-ROUTE
    const body = await req.text();
    const signature = headers().get("Stripe-Signature") as string;

    let event: Stripe.Event;

    try {
        event = stripe.webhooks.constructEvent(
            body,
            signature,
            process.env.STRIPE_WEBHOOK_SECRET!
        )
    } catch (error: any) {
        return new NextResponse(`Webhook Error: ${error.message}`, { status: 400 })
    }

    const session = event.data.object as Stripe.Checkout.Session;
    const userId = session?.metadata?.userId;
    const courseId = session?.metadata?.courseId;

    if (event.type === "checkout.session.completed") {
        if (!userId || !courseId) {
            return new NextResponse(`Відсутні метадан`, { status: 400 });
        }

        await db.purchase.create({
            data: {
                courseId: courseId,
                userId: userId,
            }
        })
    }
}

```

```

    });
  } else {
    return new NextResponse(`Webhook Error: Unhandled event type ${event.type}`,
    { status: 200 })
  }

  return new NextResponse(null, { status: 200 });
}

export async function PATCH( // API-COURSESID-публікація(опублікований)
  req: Request,
  { params }: { params: { courseId: string } }
) {
  try {
    const { userId } = auth();

    if (!userId) {
      return new NextResponse("Не авторизовано", { status: 401 });
    }

    const course = await db.course.findUnique({
      where: {
        id: params.courseId,
        userId,
      },
      include: {
        chapters: {
          include: {
            muxData: true,
          }
        }
      }
    });

    if (!course) {
      return new NextResponse("Не знайдено", { status: 404 });
    }

    const hasPublishedChapter = course.chapters.some((chapter) =>
      chapter.isPublished);

    if (!course.title || !course.description || !course.imageUrl ||
    !course.categoryId || !hasPublishedChapter) {
      return new NextResponse("Відсутні обов'язкові поля", { status: 401 });
    }

    const publishedCourse = await db.course.update({
      where: {
        id: params.courseId,
        userId,
      },

```

```

        data: {
            isPublished: true,
        }
    });

    return NextResponse.json(publishedCourse);
} catch (error) {
    console.log("[COURSE_ID_PUBLISH]", error);
    return new NextResponse("Критична помилка", { status: 500 });
}
}

//Сторінка пошуку курсів
const fetchData = async (userId: string, searchParams:
SearchPageProps["searchParams"]) => {
    const categories = await db.category.findMany({
        orderBy: {
            name: "asc"
        }
    });

    const courses = await getCourses({
        userId,
        ...searchParams,
    });

    return { categories, courses };
};

const SearchPage = async ({
    searchParams
}: SearchPageProps) => {
    const { userId } = auth();

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

    const { categories, courses } = await fetchData(userId, searchParams);

    return (
        <>
        <div className="px-6 pt-6 md:hidden md:mb-0 block">
            <SearchInput />
        </div>
        <div className="p-6 space-y-4">
            <Categories
                items={categories}
            />
            <CoursesList items={courses} />
        </div>
    )

```

```

    </>
  );
};

// reorder route розділу
export async function PUT(
  req: Request,
  { params }: { params: { courseId: string; } }
) {
  try {
    const { userId } = auth();

    if (!userId) {
      return new NextResponse("Не авторизований", { status: 401 });
    }

    const { list } = await req.json();

    const ownCourse = await db.course.findUnique({
      where: {
        id: params.courseId,
        userId: userId
      }
    });

    if (!ownCourse) {
      return new NextResponse("Не авторизований", { status: 401 });
    }

    for (let item of list) {
      await db.chapter.update({
        where: { id: item.id },
        data: { position: item.position }
      });
    }

    return new NextResponse("Успіх", { status: 200 });
  } catch (error) {
    console.log("[REORDER]", error);
    return new NextResponse("Помилка", { status: 500 });
  }
}

```