Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій і технічного захисту інформації
(повна назва)

Кафедра Радіотехнологій інформаційно–комунікаційних систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти **перший (бакалаврський)**

РОЗРОБКА ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ ТА ПОШУКУ ОНЛАЙН КУРСІВ
(тема)

Виконала:
студентка **4** курсу, групи **ІТІР–20–1**
Гринишина С.О.
(прізвище, ініціали)

Спеціальність **126 Інформаційні системи та технології**
(код і повна назва спеціальності)

Тип програми освітньо–професійна

Освітня програма **Інформаційні технології інтернету речей**
(повна назва освітньої програми)

Керівник **ст. викл. Штих І.А.**
(посада, прізвище, ініціали)

Допускається до захисту

В.о.зав. кафедри _____ **Зарудний О.А.**
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій і технічного захисту інформації

Кафедра Радіотехнологій інформаційно–комунікаційних систем

Рівень вищої освіти перший (бакалаврський)

Спеціальність 126 Інформаційні системи та технології
(код і повна назва)

Тип програми освітньо–професійна

Освітня програма інформаційні системи інтернету речей
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові **ГРИНИШИНІЙ СОФІЇ ОЛЕГІВНІ**
(прізвище, ім'я, по батькові)

1. Тема роботи **Розробка платформи для управління та пошуку онлайн курсів**

затверджена наказом університету від **27 травня 2024 р. № 500 Ст**

2. Термін подання студентом роботи до екзаменаційної комісії **10 червня 2024 р.**

3. Вихідні дані до роботи

3.1 Провести аналіз існуючих платформ для пошуку та управління курсами, таких як Coursera, Udemy, Khan Academy тощо

3.2 Визначити основні вимоги до розроблювальної платформи.

3.3 Проаналізувати особливості, переваги та недоліки фреймворку Angular, використаного .для розробки програми

4. Перелік питань, що потрібно опрацювати в роботі

Вступ

1 Аналіз існуючих сервісів навчання в онлайні: огляд та перспективи

2 Вимоги до розроблюваної платформи

3 Вибір інструментів для розробки

4 Розробка front-end частини платформи

5 Розробка back-end частини платформи

Висновки. Перелік джерел посилання. Дадатки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)
Комп'ютерна презентація – слайди у форматі Power Point

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	ст. викл. Штих І.А.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Ознайомлення із завданням. Уточнення ТЗ.	06.05.2024	виконано
2	Підбір літератури за темою роботи	07.05.– 08.05.2024	виконано
3	Аналіз існуючих платформ для онлайн-навчання	08.05.2024	виконано
4	Визначення вимог до платформи	09.05.2024	виконано
5	Вибір і аналіз інструментів для розробки платформи	10.05.2024	виконано
6	Розробка front-end частини	11.05. – 25.05.2024	виконано
7	Розробка back-end частини	25.05. – 08.06.2024	виконано
8	Оформлення презентаційного матеріалу	08.06. –10.06.2024	виконано

Дата видачі завдання 6 травня 2024 р.

Студентка _____ С.О. Гринишина
(підпис)

Керівник роботи _____ ст. викл. Штих І.А.
(підпис)

РЕФЕРАТ

Кваліфікаційна робота бакалавра складається з пояснювальної записки, що містить 102 сторінки тексту, 74 рисунка, 8 джерел посилання і 3 додатка.

КУРСИ, ПЛАТФОРМА, ANGULAR, ФРЕЙМВОРК, ТЕСТУВАННЯ, ОНЛАЙН-НАВЧАННЯ

Об’єкт розробки – створення інтегрованої платформи для управління та пошуку онлайн курсів

Метод дослідження – аналіз існуючих платформ для навчання, вивчення потреб користувачів, розробку та впровадження функціональності платформи, а також тестування та вдосконалення програмного продукту.

Платформи для онлайн навчання стають все більш популярними у світі сучасних технологій. Завдяки їм користувачі можуть здобувати нові знання у будь-який час і в будь-якому місці за допомогою інтернету. Тому розробка ефективної та зручної платформи для управління онлайн курсами стає актуальною задачею.

У роботі проведено огляд існуючих платформ для навчання, визначено основні вимоги до нової платформи та розроблено концепцію її реалізації. Надана можливість користувачам знаходити, вибирати та управляти онлайн курсами шляхом інтеграції різноманітних функцій, таких як пошук, редагування профілю, оцінювання курсів тощо.

Результатом цього дослідження є створення платформи, яка не лише дозволяє знайти та обрати курси відповідно до індивідуальних потреб користувачів, але й сприяє їхньому зручному управлінню.

ABSTRACT

The qualification work of the bachelor consists of an explanatory note containing 102 pages of text, 74 figures, 8 literary sources and 3 appendices.

COURSES, PLATFORM, ANGULAR, FRAMEWORK, TESTING, ONLINE LEARNING

The object of development - creation of an integrated platform for managing and searching for online courses.

Research method – to analyze existing learning platforms, study user needs, develop and implement platform functionality, and test and improve the software product.

Online learning platforms are becoming increasingly popular in the world of modern technology. They allow users to acquire new knowledge anytime and anywhere via the Internet. Therefore, developing an effective and convenient platform for managing online courses is becoming an urgent task.

This paper reviews existing learning platforms, identifies the main requirements for a new platform, and develops a concept for its implementation. The paper enables users to find, select and manage online courses by integrating various functions such as search, profile editing, course evaluation, etc.

The result of this research is the creation of a platform that not only allows users to find and select courses according to their individual needs, but also facilitates their convenient management.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AES (Advanced Encryption Standard) – симетричний алгоритм блочного шифрування;

CSS (Cascading Style Sheets) – це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду;

FAQ (Frequently Asked Questions) – часто поставлені, поширені питання; питання, які часто ставляться;

HTML (Hypertext markup language) – мова розмітки гіпертексту;

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертекстових документів.

MOOC (Massive Open Online Courses) – інтернет-курс з великомасштабною інтерактивною участю та відкритим доступом через інтернет.

SASS (Syntactically Awesome StyleSheets) – скриптова метамова, яка інтерпретується в каскадні таблиці стилів (CSS).

Wi–Fi (Wireless Fidelity) – технологія бездротової локальної мережі;

БД – база даних;

ДСТУ – державний стандарт України;

ПК – персональний комп’ютер;

ВСТУП

У сучасному світі, де технології стрімко розвиваються, освіта не може залишатися осторонь цих процесів. Онлайн-курси набувають все більшої популярності, оскільки вони пропонують гнучкість, доступність та можливість навчатися у зручному темпі та місці. Однак, для ефективного управління та підтримки онлайн-курсів необхідні зручні інструменти, які дозволять легко редагувати, додавати та знаходити потрібний контент.

Саме з цією метою було розроблено платформу для редагування, додавання та пошуку онлайн-курсів на базі фреймворку Angular. Ця платформа є зручним та потужним інструментом, який дозволяє створювати, редагувати та керувати онлайн-курсами різноманітної тематики. Вона забезпечує зручний інтерфейс для викладачів та адміністраторів, що сприяє ефективному процесу створення та підтримки якісного навчального контенту.

Крім того, платформа забезпечує ефективний пошук онлайн-курсів за ключовими словами, темами або іншими критеріями, що полегшує навігацію та знаходження потрібних ресурсів. Це є особливо корисним для студентів та слухачів, які можуть швидко знайти необхідний курс та розпочати навчання.

Використання фреймворку Angular для розробки платформи забезпечує високу продуктивність, масштабованість та надійність системи. Angular є потужним інструментом для створення веб-застосунків, який дозволяє розробникам писати чистий, структурований код та легко підтримувати проект в майбутньому.

Ця дипломна робота детально описує процес розробки платформи для редагування, додавання та пошуку онлайн-курсів, а також її основні функціональні можливості та архітектуру. Крім того, в роботі розглядаються переваги та виклики, пов'язані з використанням Angular, а також можливі напрямки вдосконалення та розширення платформи в майбутньому.

1 АНАЛІЗ ІСНУЮЧИХ СЕРВІСІВ НАВЧАННЯ В ОНЛАЙНІ: ОГЛЯД ТА ПЕРСПЕКТИВИ

1.1 Історія розвитку онлайн-навчання

При розгляді історії дистанційної освіти під певним кутом можна помітити, що успіхи, досягнуті у її розвитку, належать кількох "поколінням". Першим таким "поколінням" були рукописи та друковані матеріали. Протягом багатьох століть інформацію передавали за допомогою рукописів, але поява книгодруку дозволила випускати доступні підручники. З середини XIX століття навчальні матеріали почали постачати за допомогою залізничної системи та швидкої та економічної поштової служби. З виникненням радіо в 20-і роки XX століття з'явилися Радіокурси, які часом доповнювалися друкованими матеріалами або аудиторними заняттями. А вже в 50-х роках розвивалися телевізійні курси.

Початок "другого покоління" дистанційної освіти пов'язаний із заснуванням університету у Великобританії у 1969 році. Саме тоді вперше в цій галузі використовувався комплексний підхід до навчання. Було розроблено велику кількість якісних посібників, спеціально призначених для дистанційного навчання. Взаємодія університету із студентами відбувалася як через друкований матеріал, так і за допомогою радіо- і телепередач. Двосторонній контакт здійснювався через очні консультації, короткострокові курси і листування. Хоча ця модель була дорогою на початковому етапі, зі створенням всіх необхідних матеріалів та програм для навчання нових студентів витрати зменшилися [1].

"Третє покоління" дистанційної освіти базується на активному використанні комунікаційних та інформаційних технологій. З появою цих технологій з'явилися нові можливості для двостороннього зв'язку як у синхронному (відео- або аудіографічні конференції), так і в асинхронному режимі (електронна пошта, Інтернет, телеконференції). Ці методи навчання можуть бути використані як у доповненні до курсів перших двох поколінь, так і

самостійно, проте в будь-якому випадку значно спрощують взаємодію між викладачем та студентом.

Історія розвитку дистанційного навчання свідчить про його абсолютну гнучкість та здатність адаптуватися до змінних потреб суспільства. З появою нових технологій змінюються й способи організації дистанційного навчання. Ця система продовжує швидко розвиватися, і все більше навчальних закладів, підприємств і державних організацій впроваджують технології дистанційного навчання у свій навчальний процес.

В цілому, розвиток світового ринку дистанційного навчання продовжує йти досить активно, чому сприяє з одного боку підвищення попиту на освітні послуги, а з іншого боку - розвиток власне інформаційних технологій, зростання інтернет-аудиторії [1].

Одним за аспектів такого підходу до навчання є онлайн-платформи. У світі існує велика кількість варіантів таких платформ для навчання, які пропонують різноманітні курси з різних галузей знань, починаючи від програмування та мов до мистецтва та бізнесу. Платформи, такі як Coursera, Udemy, edX, Khan Academy та інші, стають улюбленими серед студентів та професіоналів, які бажають покращити свої навички або отримати нові знання. Їхні унікальні функції, такі як великий вибір курсів, можливість вивчати матеріали у власному темпі, взаємодія з викладачами та спільнотою студентів, а також сертифікація після завершення курсу, роблять їх дуже привабливими для навчання на віддаленій основі. Такі платформи перетворили спосіб, яким люди отримують освіту, забезпечуючи доступні та якісні навчальні ресурси для всіх бажаючих.

1.2 Основні компоненти системи онлайн-навчання

Основні компоненти системи онлайн-навчання створюють той каркас, на якому базується весь процес навчання та взаємодії між користувачами платформи. Ці компоненти охоплюють широкий спектр функцій, від керування

контентом та навчальними матеріалами до інтерактивного спілкування та оцінювання успішності студентів.

Серед компонентів, які входять в систему онлайн навчання є контент, який становить основу знань та інформації, яку студенти вивчають. Цей розділ включає різноманітні матеріали, які можуть бути представлені у формі відео, текстових матеріалів, аудіофайлів, інтерактивних завдань та багато іншого. Контент розробляється викладачами або спеціалістами у відповідній галузі та завантажується на платформу, де студенти мають доступ до нього. Важливо, щоб контент був якісним, доступним та цікавим для аудиторії, щоб забезпечити ефективне навчання та залучення студентів до вивчення матеріалу. Якісний контент повинен бути структурованим, легким у сприйнятті та доступним для різних типів учнів. Крім того, він може бути персоналізованим, щоб враховувати індивідуальні потреби та рівень знань кожного студента;

Платформа є основою всього навчального процесу, забезпечуючи доступ до навчального контенту та інструментів спілкування. Вона може бути веб-сайтом, мобільним додатком або комплексною системою, що об'єднує різноманітні функції та можливості. Платформа для онлайн-навчання також включає в себе різноманітні інструменти для підтримки навчального процесу. Серед них можуть бути системи відеоконференцій для віртуальних занять в реальному часі, форуми для обговорення матеріалів та співпраці між учасниками курсу, системи оцінювання та відстеження прогресу студентів, а також інтерактивні інструменти для взаємодії з контентом, такі як відеоуроки з можливістю питань та відповідей, вправи та тести. Крім того, платформа може включати в себе інструменти адміністрування для викладачів та адміністраторів курсу, які дозволяють керувати розкладом занять, виставляти оцінки, стежити за активністю студентів та аналізувати їхній прогрес. Ці інструменти спрощують організацію та управління навчальним процесом, забезпечуючи викладачам зручний та ефективний спосіб взаємодії зі своїми студентами.

Загалом, платформа для онлайн-навчання виступає важливим інструментом для навчання у сучасному світі, забезпечуючи доступ до якісного

навчального матеріалу та сприяючи взаємодії між учасниками навчального процесу. Її розвиток та функціональні можливості продовжують зростати, що робить онлайн-навчання ще більш доступним та ефективним для широкого кола людей.

Керування користувачами: відіграє критичну роль у забезпеченні безпеки та ефективного функціонування платформи для онлайн-навчання. Одним із основних завдань цього компонента є реєстрація нових користувачів та збереження їхніх особистих даних у безпечній формі. Для цього зазвичай використовуються різноманітні методи аутентифікації, такі як електронна пошта, соціальні мережі або створення унікального логіну та пароля.

Після реєстрації користувачів необхідно автентифікувати, тобто перевірити їхню ідентичність перед наданням доступу до ресурсів платформи. Це забезпечується за допомогою різних методів, включаючи введення логіну та пароля, використання двофакторної аутентифікації, а також біометричні методи, такі як сканування відбитків пальців або розпізнавання обличчя.

Після успішної автентифікації користувачі отримують доступ до певних ресурсів платформи, проте цей доступ може бути обмежений в залежності від їхніх ролей та прав. Керування правами доступу дозволяє адміністраторам платформи регулювати, які ресурси можуть бути доступні для кожного користувача, а також встановлювати правила та обмеження щодо редагування, перегляду чи видалення певної інформації.

Усі ці аспекти обробки даних користувачів важливі для забезпечення безпеки, конфіденційності та цілісності даних на платформі для онлайн-навчання. Адже належна увага до керування користувачами гарантує, що тільки авторизовані та вповноважені особи матимуть доступ до навчальних ресурсів, а також можуть взаємодіяти з ними відповідно до своїх ролей та функціональних можливостей.

Інтерактивність та взаємодія є важливим аспектом у забезпеченні ефективного навчального процесу онлайн навчання. Цей компонент створює можливості для активного спілкування, обміну думками та досвідом між

учасниками навчального процесу, що сприяє кращому засвоєнню матеріалу та розвитку критичного мислення.

Одним із ключових елементів цього компонента є можливість студентів та викладачів спілкуватися між собою, обговорювати навчальний матеріал, ставити запитання та отримувати відповіді у режимі реального часу. Це може здійснюватися через форуми, чати, відеоконференції та інші засоби комунікації, які надають можливість вирішувати питання та розглядати концепції разом з іншими учасниками навчання. Крім того, компонент інтерактивності та взаємодії також включає в себе інструменти для проведення тестів, виконання завдань та інших форм активного навчання. Студенти можуть виконувати тести для перевірки своїх знань, здійснювати практичні завдання для закріплення матеріалу та отримувати зворотний зв'язок від викладачів.

Забезпечення можливостей для інтерактивності та взаємодії на платформі для онлайн-навчання стимулює активність студентів, сприяє поглибленому розумінню матеріалу та розвитку навичок самостійності та співпраці. Такий підхід до навчання дозволяє створити стимулююче та підтримуюче середовище для досягнення навчальних цілей.

Оцінювання та звітність: є важливою складовою платформи для онлайн-навчання, яка дозволяє ефективно вимірювати навчальні результати студентів та надавати відповідну звітність для викладачів і адміністраторів. Цей компонент включає в себе кілька ключових функцій:

– створення тестів та завдань: викладачі можуть створювати різноманітні тести та завдання для оцінки знань студентів. Це можуть бути тестові завдання, практичні вправи, кейс-стаді, проекти тощо;

– оцінювання робіт студентів: платформа надає можливість викладачам оцінювати та коментувати роботу студентів онлайн. Вони можуть виставляти оцінки за завдання, враховуючи різні критерії;

– видача сертифікатів: після успішного завершення курсу студентам може бути виданий сертифікат або диплом, який підтверджує їхні навички та досягнення.

– звітність: платформа автоматично формує звіти про успішність студентів, які можуть бути використані для аналізу прогресу, виявлення слабких місць та прийняття подальших навчальних рішень;

– підтримка: технічна підтримка забезпечує студентам та викладачам допомогу з технічними питаннями, такими як вхід в систему, навігація по платформі, використання функцій веб-сайту чи мобільного додатку та вирішення технічних питань. Технічні спеціалісти або адміністратори можуть надавати консультації та вирішувати проблеми користувачів.

Педагогічна підтримка включає консультації та підтримку для викладачів з педагогічних питань. Вона може включати поради щодо дизайну курсів, методів навчання, організації уроків та інші аспекти викладання.

Онлайн-консультації надають можливість планувати та проводити консультації між викладачами та студентами в реальному часі. Це дозволяє вирішувати навчальні питання, отримувати зворотний зв'язок та спілкуватися з учасниками курсу.

Довідковий матеріал, доступний на платформі, може включати інструкції, відео-уроки, FAQ та інші ресурси, які допомагають користувачам максимально використовувати можливості системи та досягати успіху у навчанні.

Забезпечення належної підтримки є важливим аспектом успішного функціонування платформи для онлайн-навчання, оскільки воно допомагає користувачам максимально використовувати її можливості і досягати успіху у навчанні.

1.3 Ринок онлайн-навчання: основні учасники

Ринок онлайн-навчання є одним з найбільш динамічних та швидкозростаючих секторів у сучасній освітній сфері. З кожним роком все більше людей обирають цей спосіб отримання знань через доступні та зручні онлайн-платформи. Разом із зростанням популярності цього формату навчання

з'являється також розмаїття учасників ринку, які пропонують широкий спектр курсів та навчальних програм.

Існує безліч різноманітних платформ з курсами та учбовими програмами, але можна виділити декілька найголовніших та найвідоміших програм.

1.3.1 Платформа Udemy: загальні відомості та огляд

Платформа Udemy, заснована в травні 2010 року Ереном Балі, Гаганом Біяні та Октаем Чагларом, представляє собою освітню технологічну компанію, що надає онлайн-платформу для навчання та викладання. За даними на червень 2023 року, на платформі зареєстровано 64 мільйони учнів, доступно понад 210 000 курсів та понад 75 000 інструкторів, які викладають курси майже у 75 мовах, з загальною кількістю понад 870 мільйонів записаних курсів. Udemy Business має понад 14 900 клієнтів, з яких більше 50% належать до списку Fortune 100. Клієнти мають доступ до понад 24 тисяч курсів, з яких понад 10,5 тисячі пропонуються англійською мовою, а близько 13,5 тисяч іншими мовами.

Студенти переважно відвідують курси для поліпшення своїх професійних навичок. Деякі курси надають кредити для технічної сертифікації. Udemy привертає корпоративних тренерів, які мають намір створювати курси для навчання співробітників своїх компаній.

Платформа має зручний та зрозумілий інтерфейс, що сприяє швидкому пошуку необхідних матеріалів по назві або категоріях. На рисунках 1.1 та 1.2 відповідно зображений інтерфейс пошуку курсів.

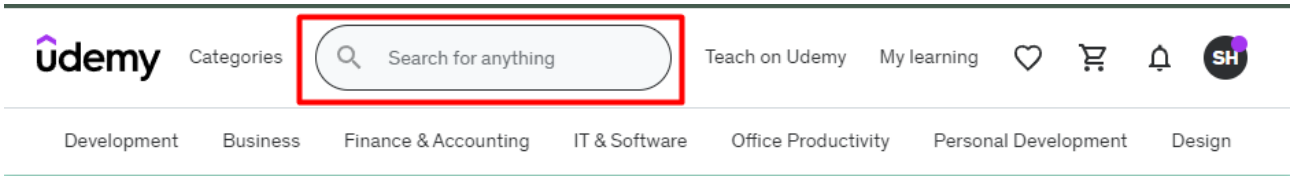


Рисунок 1.1 – Пошук курсу по назві

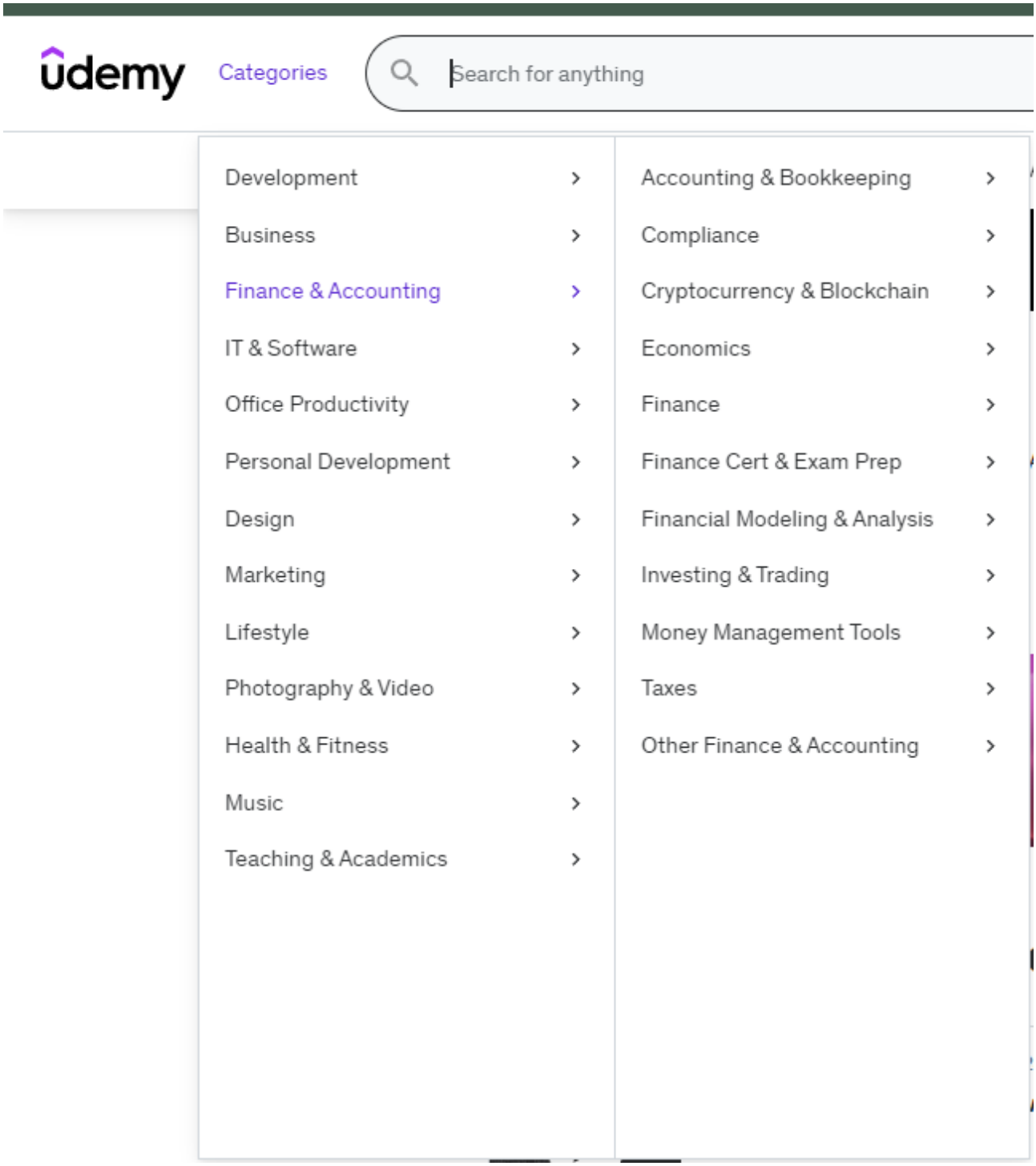


Рисунок 1.2 – Пошук курсів по категорії

При введені в пошуковий рядок запиту на сторінці з’являється список запропонованих курсів. Також є різноманітні варіанти фільтрації результату відповідно до потреб користувача. На рисунку 1.3 зображений приклад списку курсів знайдених за запитом «Angular».

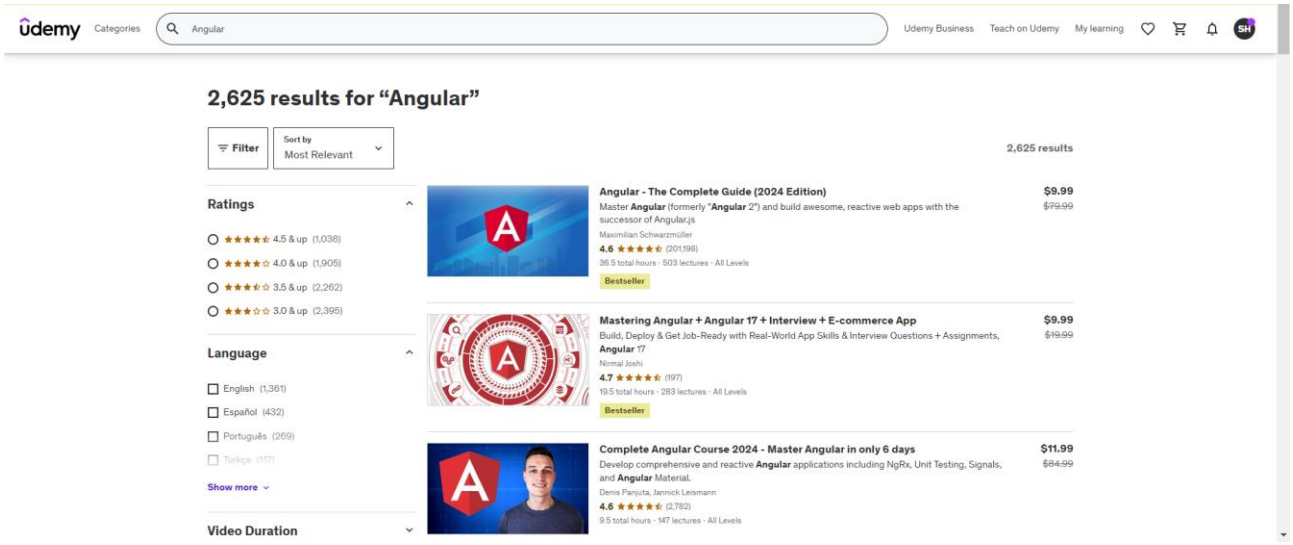


Рисунок 1.3 – Пошук курсів по категорії

При виборі конкретного крусу відкривається більш детальна інформація: дата осаннього оновлення, мова, список доступних субтитрів, зміст, опис, перелік необхідних навичок, яких буде достатньо користувачеві, аби пройти матеріал, а також відгуки інших людей. Користувач може безкоштовно переглянути декілька перших уроків, також є можливість подарувати комусь курс. На рисунках 1.4, 1.5, 1.6 зображено приклад детального опису курсу «Angular - The Complete Guide (2024 Edition)».

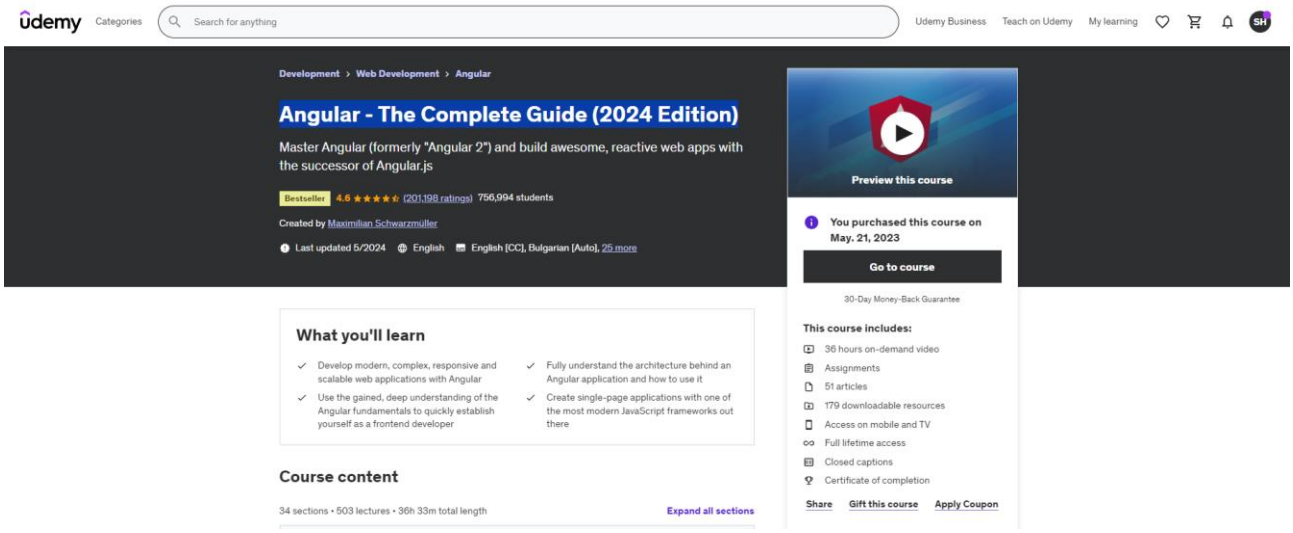


Рисунок 1.4 – Опису курсу «Angular - The Complete Guide (2024 Edition)».

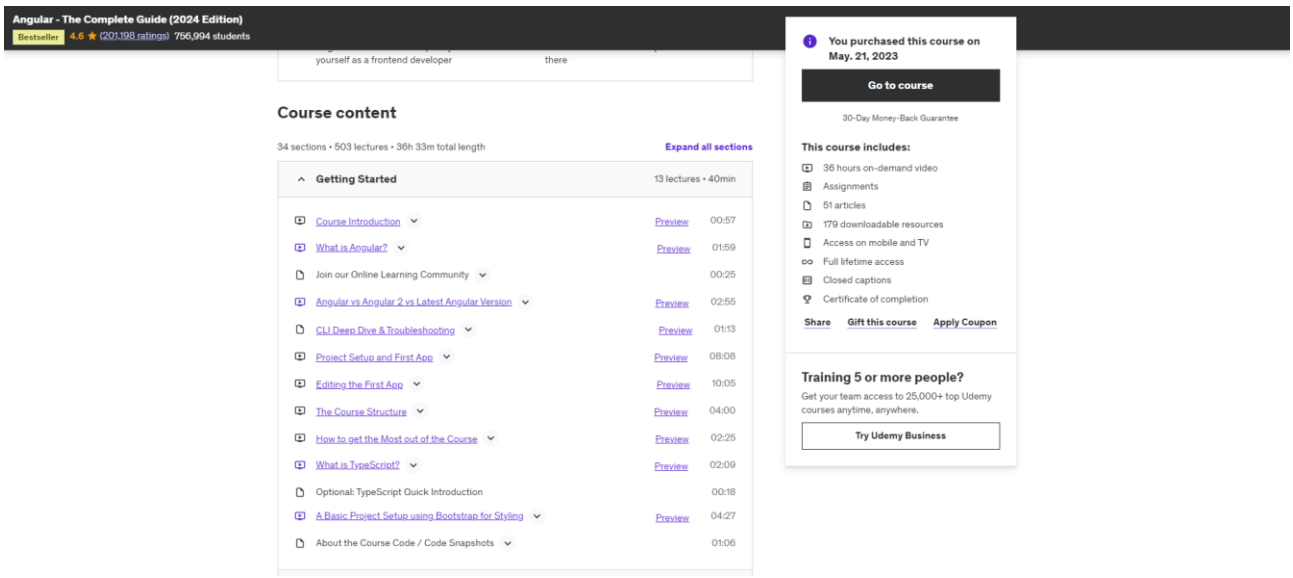


Рисунок 1.5 – Опису курсу «Angular - The Complete Guide (2024 Edition)»

Якщо ви обираєте курс, який ще не придбали, з’являється віконце, яке дозволяє додати курс в кошик або оплатити одразу. Також є можливість додати його в обрані. На рис. 1.7 зображено приклад інтерфейсу сторінки покупки курсу.

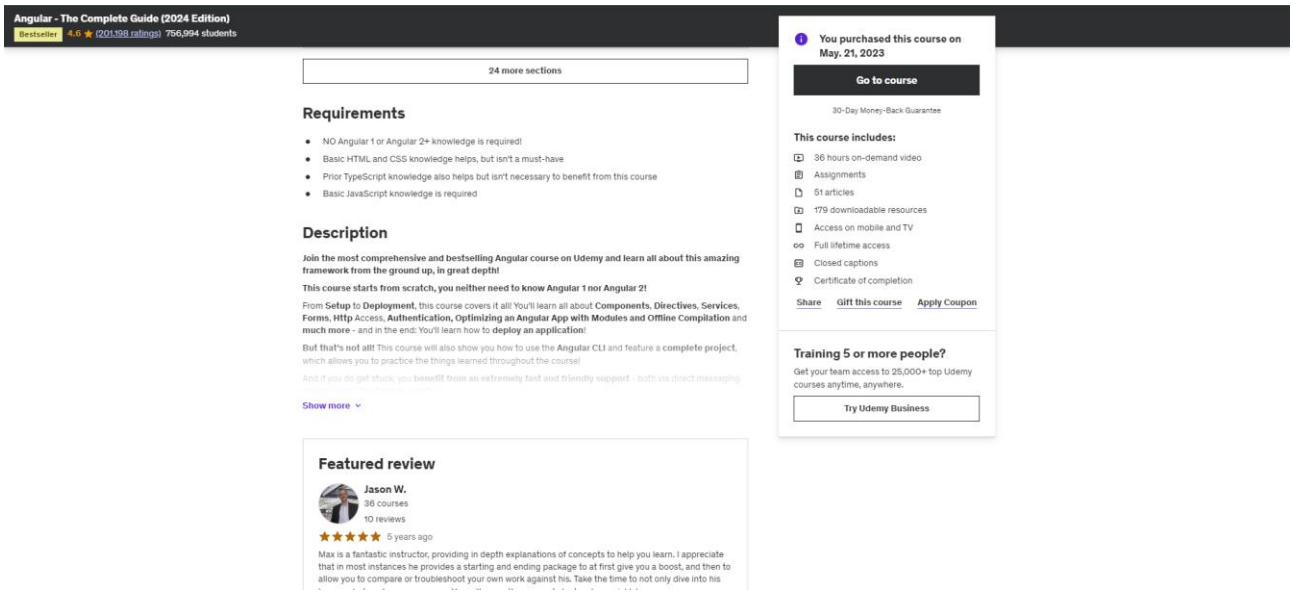


Рисунок 1.6 – Опису курсу «Angular - The Complete Guide (2024 Edition)»

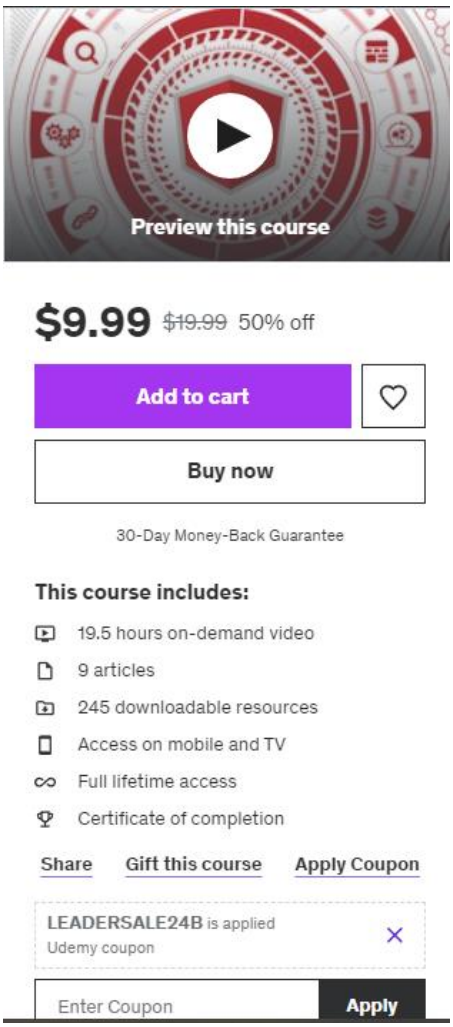


Рисунок 1.7 – Операції з курсом

В розділі «Кошик» користувач може оплатити додані курси та перейти до навчання. Також можна застосувати промокод на знижку. На рисунку 1.8 зображено приклад корзини.

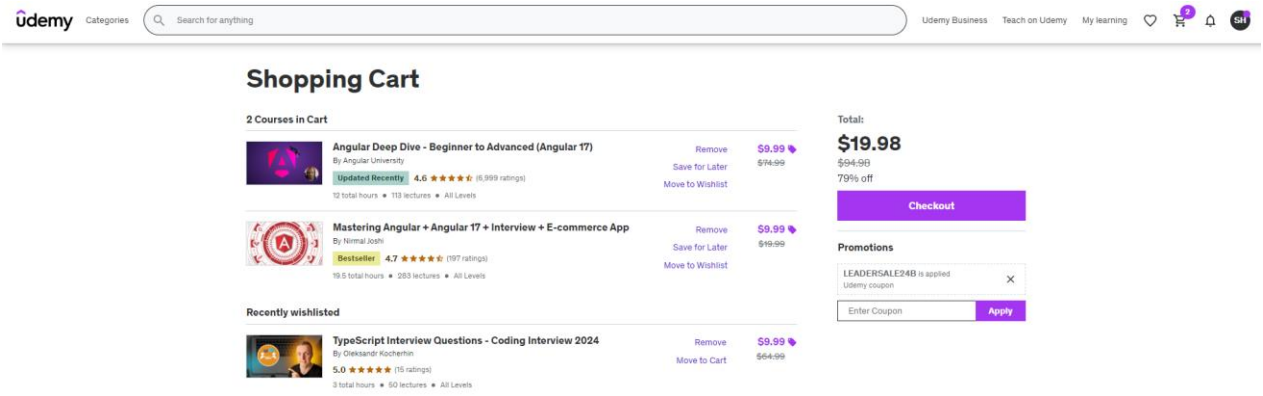


Рисунок 1.8 – Корзина

На сторінці акаунту користувача є можливість переглянути та змінити особисті дані, налаштувати безпеку акаунту, підписки, методи оплати, приватність, сповіщення та ін. На рисунку 1.9 зображено інтерфейс профілю користувача.

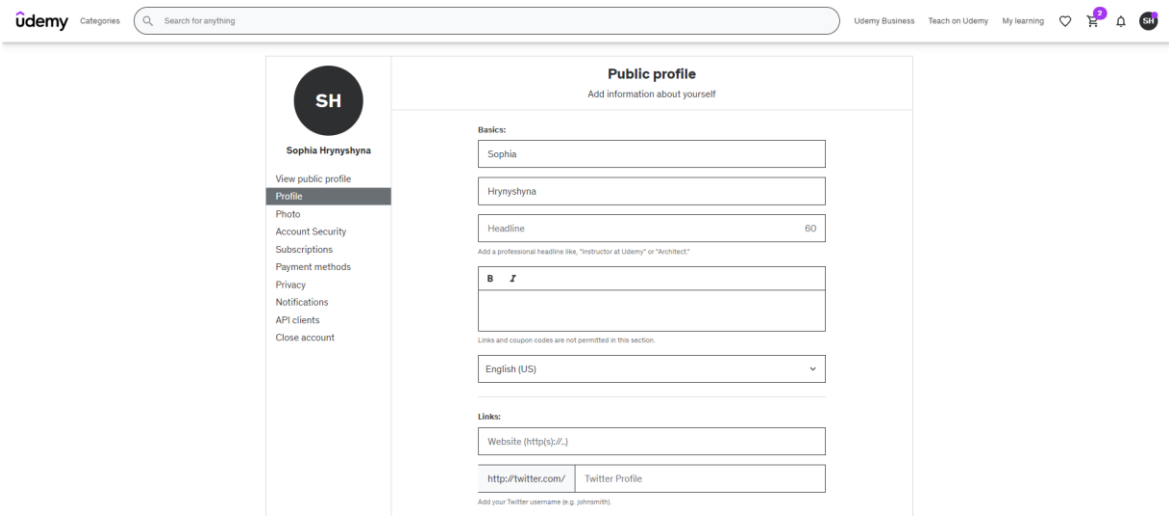


Рисунок 1.9 – Інтерфейс профілю користувача

Загалом, інтерфейс сайту є досить зрозумілим, проте, на мою думку, дизайн можна було б удосконалити, зробивши його більш лаконічним та кольоровим. З логотипу можна зрозуміти, що основними кольорами є сірий та фіолетовий. Гарним рішенням, було б розбавити фіолетовими акцентами сторінки сайту. Також в профілі користувача багато налаштувань розділені на різні секції, чого, на моє переконання, у цьому випадку можна було б уникнути, оскільки в кожній з цих секцій кількість налаштувань досить мала, тому деякі з них могли б бути об’єднані.

1.3.2 Платформа Coursera: загальні відомості та огляд

Coursera - це міжнародна освітня платформа, що пропонує онлайн-курси, відомі як масивні відкриті онлайн-курси або MOOC - Massive Open Online Courses від найкращих університетів світу. Заснована у 2012 році двома професорами комп’ютерних наук зі Стенфордського університету Ендрю Ин та

Дафною Колле. Coursera на сьогодні є однією з провідних компаній у сфері МООС.

Тривалість курсів на платформі Coursera варіюється від шести до десяти тижнів. Навчання включає в себе відеолекції, консультації, форумні обговорення, завдання та тести. Після успішного завершення курсу учасники отримують сертифікати. Навчання доступне як на персональних комп'ютерах, так і на мобільних пристроях [2].

Концепція Coursera значно відрізняється від Udemy. Якщо в першому варіанті користувач міг проходити курси, тривалість яких вимірюється в годинах, а вартість не перевищує 200$ і людина сама обирає, коли їй проходити матеріал, робить це самостійно і в кінці отримує сертифікат, то деякі курси, пропоновані платформою Coursera - це повноцінне навчання в різних університетах, яке може тривати декілька років, і цінник на навчання може доходити десятків тисяч доларів. Проте серед списку курсів є багато таких, що також можна пройти в своєму темпі, отримати бажані навички та сертифікат. Приклад головної сторінки платформи зображено на рисунку 1.10.

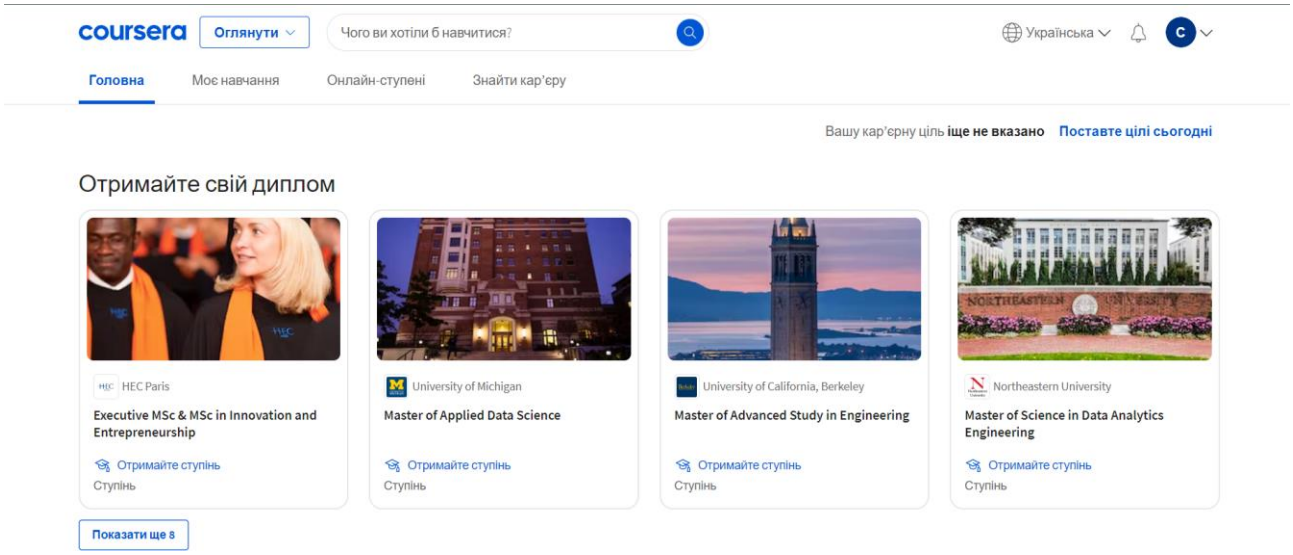


Рисунок 1.10 – Головна сторінка Coursera

При виборі курсу так само можна прочитати детальну інформацію: вартість, тривалість, опис і тд. На рисунках 1.10, 1.11, 1.12 можна побачити приклади сторінки курсу.

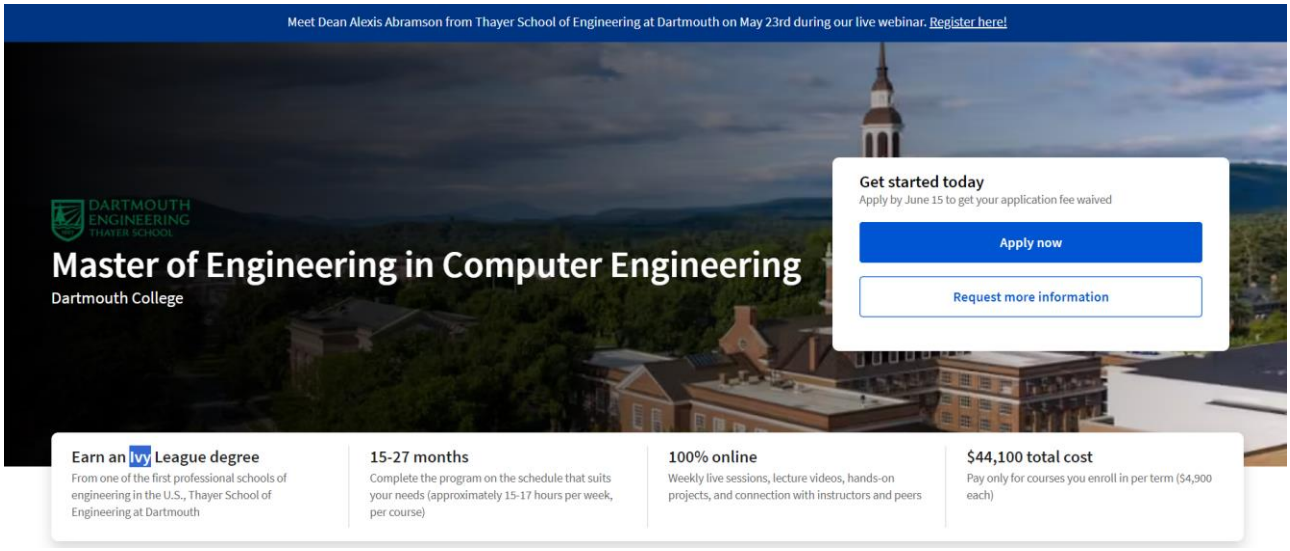


Рисунок 1.11 – Головна сторінка Coursera

Рисунок 1.12 – Головна сторінка Coursera

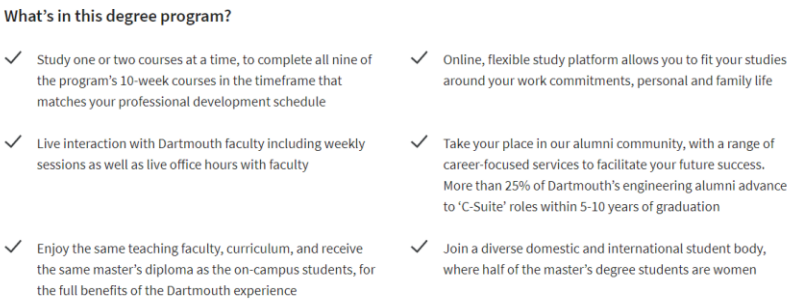


Рисунок 1.13 – Головна сторінка Coursera

В особистому профілі користувач може редагувати особисту інформацію, вказувати свої проекти, досвід роботи та освіту. На рисуну 1.14 зображено скріншот акаунту користувача.

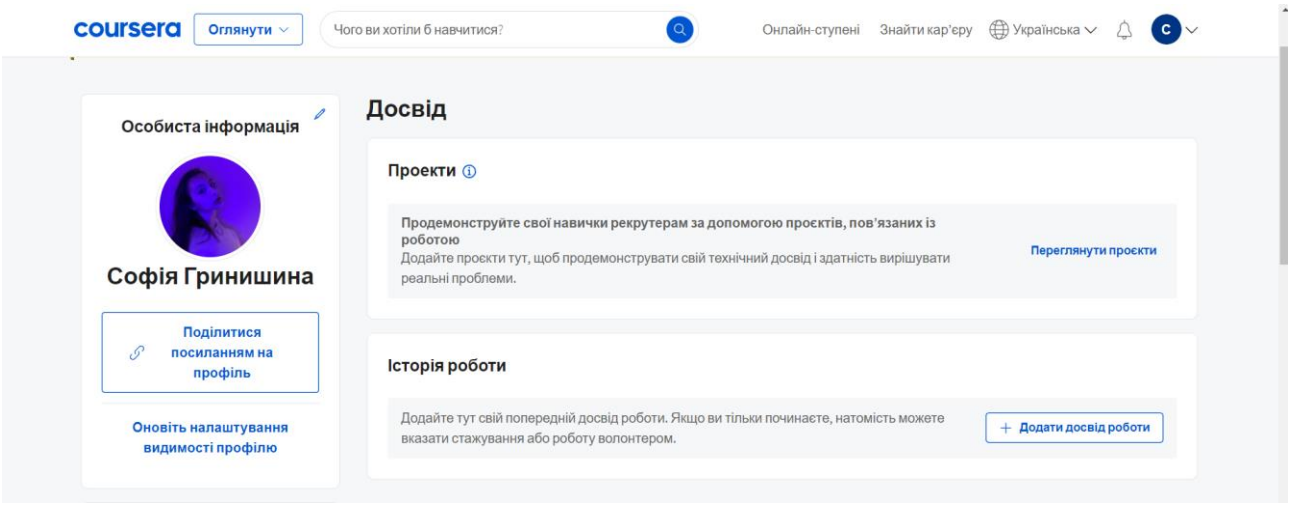


Рисунок 1.14 – Акаунт користувача Coursera

Інтерфейс платформи Coursera, на мою думку, є лаконічним, інтуїтивно зрозумілим та візуально привабливим. Проте, мені кинулось в очі те, що на головній сторінці текст кнопки виходить за її рамки, що, як на мене неприпустимо для такого відомого проєкту. Скріншот сторінки з кнопкою зображено на рисунку 1.15.

Рисунок 1.15 – Сторінка зі “зламаною” кнопкою

Також важливим фактом є те, що міністерство освіти і науки України (МОН) уклало важливе партнерство з провідною платформою онлайн-освіти Coursera, що стало значним кроком у забезпеченні українських студентів якісними освітніми ресурсами в умовах воєнного часу. Починаючи з березня 2022 року, ця співпраця дозволила українським закладам вищої освіти та їхнім

студентам отримати безкоштовний доступ до платформних ресурсів Coursera for Campus до 31 липня 2024 року.

У рамках цієї ініціативи, 250 українських закладів вищої освіти отримали необмежений доступ до понад 8 700 курсів, що охоплюють широкий спектр дисциплін і спеціальностей. Завдяки цьому понад 56 000 українських слухачів змогли пройти близько 180 000 курсів та здобути сертифікати, що підтверджують їхні знання та навички.

Coursera виконала значний обсяг роботи з перекладу понад 4 000 курсів українською мовою та запровадила нові функції на основі штучного інтелекту, що забезпечують більш персоналізоване та інтерактивне навчання. Зокрема, найпопулярніші курси, такі як «Наука про благополуччя» від Єльського університету, «ШІ для кожного» від DeepLearning.AI та «Що таке наука про дані?» від IBM, тепер доступні українською мовою.

Генеральний директор Coursera Джефф Магіонкалда також підкреслив важливість цього партнерства, висловивши готовність продовжувати підтримувати українських студентів у їхньому навчальному шляху, використовуючи можливості штучного інтелекту, перекладаючи курси українською мовою та надаючи необмежений доступ до платформи.

Популярність курсів серед українських студентів свідчить про їхнє прагнення до здобуття знань у цифрових та технічних галузях. Серед найпопулярніших курсів варто відзначити «Англійська для науки, технологій, інженерії та математики» від Університету Пенсільванії, «Програмування для всіх (Початок роботи з Python)» від Університету Мічигану, «Основи управління проєктами» від Google, «Нейронні мережі та глибоке навчання» від DeepLearning.AI та інші курси від провідних університетів та компаній.

Це партнерство є яскравим прикладом того, як міжнародна співпраця може сприяти розвитку освіти в умовах кризи, забезпечуючи студентам доступ до високоякісних навчальних матеріалів та підтримуючи їхні освітні прагнення.

Аналіз двох провідних освітніх платформ, Udemy та Coursera, показав їх значний внесок у розвиток онлайн-освіти та підвищення доступності знань по

всьому світу. Платформи пропонують широкий спектр курсів, що охоплюють різноманітні галузі та дисципліни, забезпечуючи користувачів високоякісним навчальним контентом.

Обидві платформи мають свої унікальні переваги та недоліки, але разом вони формують значний освітній ресурс, що задовольняє потреби різних аудиторій. Udemy та Coursera продовжують розширювати свої можливості, впроваджуючи інноваційні підходи та нові формати навчання, що сприяє зростанню популярності онлайн-освіти у всьому світі.

Udemy вирізняється своїм інтуїтивно зрозумілим інтерфейсом, значною кількістю курсів та великим різноманіттям мов викладання. Платформа надає можливість кожному, хто має відповідні знання та навички, стати інструктором і створювати власні курси. Це сприяє великій кількості унікального контенту та підходів до навчання. Однак дизайн сайту можна зробити більш лаконічним і естетично привабливим за допомогою оптимізації кольорової гами та об'єднання налаштувань профілю.

Coursera, у свою чергу, орієнтована на академічну співпрацю з провідними університетами світу, забезпечуючи користувачів курсами, створеними найкращими викладачами та професорами. Платформа відрізняється структурованістю та тривалістю курсів, що включають різноманітні навчальні елементи. Особливо варто відзначити ініціативу Coursera під час карантину, яка надала безкоштовний доступ до своїх програм для безробітних, що є важливим кроком у підтримці освіти в кризові часи.

2 ВИМОГИ ДО РОЗРОБЛЮВАНОЇ ПЛАТФОРМИ

2.1 Функціональні вимоги

2.1.1 Автентифікація користувача

Для забезпечення безпеки та особистої ідентифікації кожного користувача, система має включати модуль автентифікації. Цей модуль дозволяє користувачам вхід на платформу за допомогою унікальних облікових записів та паролів. Аутентифікація забезпечує захист особистих даних та забезпечує доступ до індивідуальних налаштувань та інших персоналізованих функцій платформи. Важливим аспектом цього модуля є його надійність та захищеність від несанкціонованого доступу, що забезпечується шляхом використання сучасних методів шифрування та перевірки ідентичності користувачів.

Доступ до редагування, створення та видалення курсів має бути тільки у авторизованих користувачів. Перегляд існуючих варіантів дозволений без входу в акаунт, але при спробі будь яких махінації користувача буде перекинуто на сторінку входу.

2.1.2 Список курсів

Модуль "Список курсів" є одним із ключових компонентів платформи для управління онлайн-навчанням. Він забезпечує користувачів зручним та ефективним способом перегляду доступних курсів, надаючи можливість швидко знаходити і вибирати ті курси, які найбільше відповідають їхнім потребам та інтересам. Цей модуль реалізує кілька важливих функцій, що забезпечують інтуїтивність та легкість у використанні.

Модуль надає користувачам можливість перегляду курсів у зручному форматі. Курси будуть представлені у вигляді списку, що дозволяє швидко

отримати основну інформацію про курс, таку як назва, короткий опис, тривалість та іншу інформацію.

Інтегрований механізм пошуку дозволяє користувачам знаходити курси за ключовими словами що значно спрощує процес навігації по великій кількості доступних курсів і допомагає швидко знаходити потрібний контент.

Макет інтерфейсу сторінки з переліком курсів зображено на рисунку 2.1.

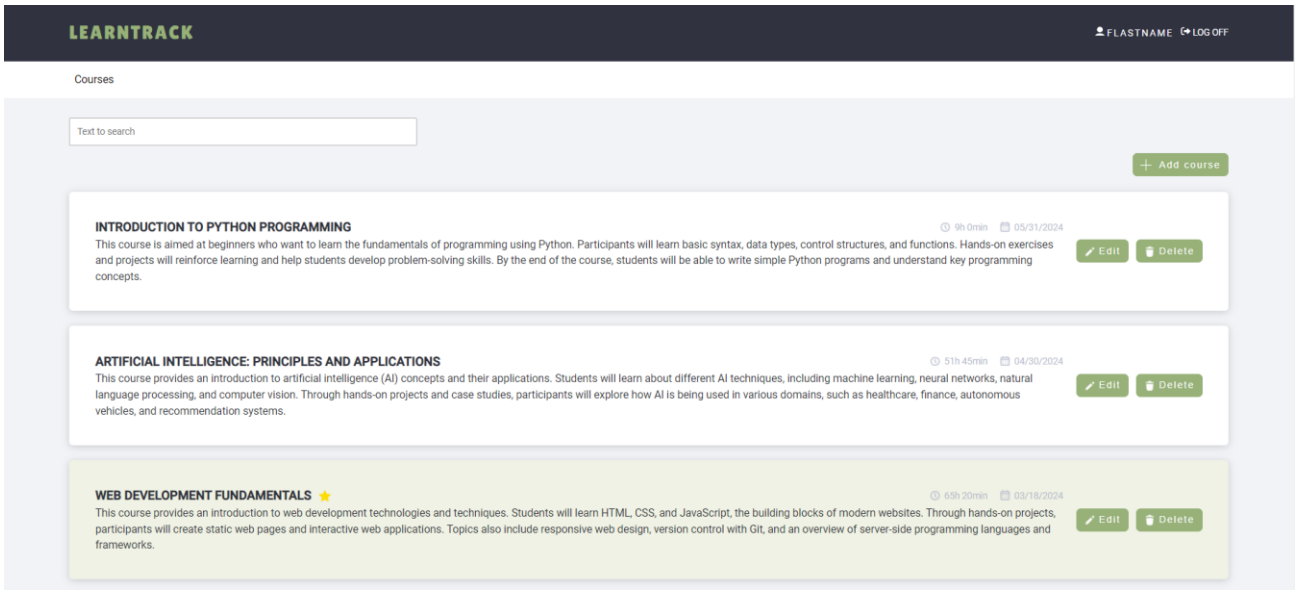


Рисунок 2.1 – Макет сторінки зі списком курсів

2.1.3 Створення курсів

Модуль створення курсів надає можливість викладачам та авторам контенту створювати нові навчальні програми та курси для користувачів платформи. Крім того, він дозволяє налаштовувати параметри курсів, такі як назва, опис, список авторів та дату виходу курсу.

При створенні нового курсу користувачеві буде надана форма для заповнення, яка матиме такі поля: заголовок, опис, дата виходу курсу, тривалість та перелік авторів. Тривалість курсу буде вказуватись у хвилинах, оброблятиметься програмою і виводитиметься в форматі «X год. XX хв».

Також форма матиме валідацію: всі поля мають бути обов’язково заповненими, та бути правильного формату, а саме: ім’я та опис не може

складатися з цифр, тривалість не може містити в собі літери. При некоректному заповненні форма не може бути збережена. Макети інтерфейсу пустої, правильно заповненої та неправильно заповненої сторінки створення курсу, зображено на рисунках 2.2, 2.3, 2.4 відповідно.

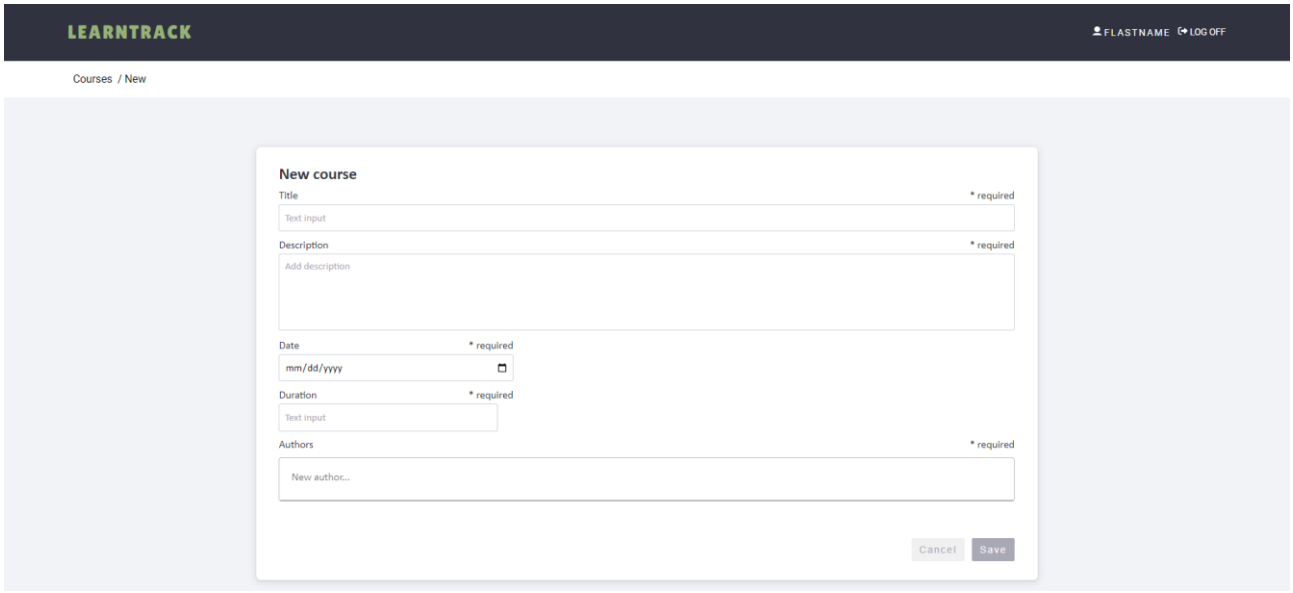


Рисунок 2.2 – Макет пустої сторінки додавання курсу

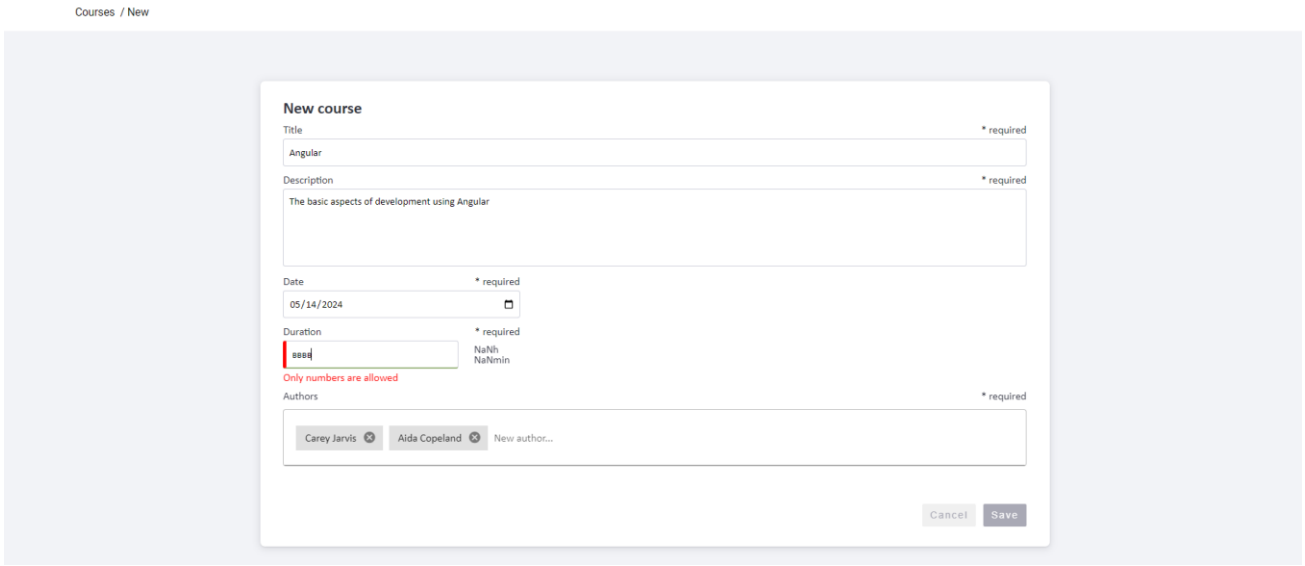


Рисунок 2.3 – Макет неправильно заповненої сторінки додавання курсу

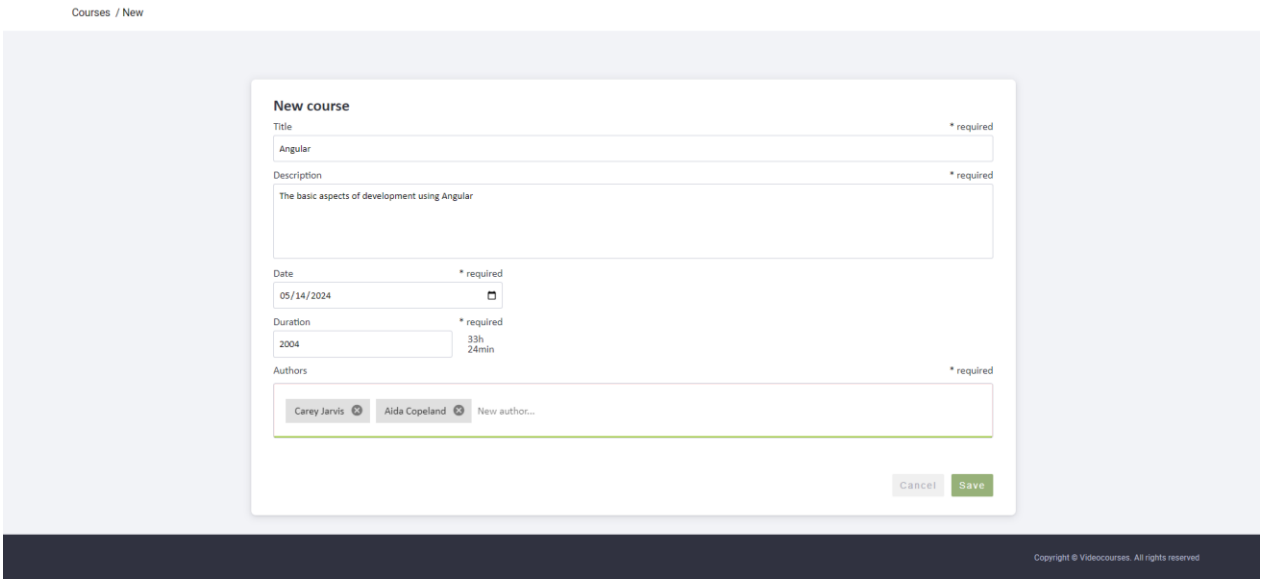


Рисунок 2.4 – Макет правильно заповненої сторінки додавання курсу

2.1.4 Пошук

Механізм пошуку курсів є важливою складовою платформи для управління онлайн-навчанням, який надає користувачам зручний та ефективний спосіб знаходити курси, що відповідають їхнім потребам та інтересам. Цей модуль дозволяє користувачам шукати курси за назвою та описом.

При розробці враховується зручність і простота його використання для користувачів. Інтерфейс пошуку розробляється з урахуванням сучасних стандартів дизайну та управління інформацією, щоб забезпечити зручний та інтуїтивно зрозумілий спосіб взаємодії з платформою.

2.1.5 Профіль користувача

Профіль користувача містить особисту інформацію користувача, таку як фото, ім'я та прізвище. Фото може бути завантажене користувачем або імпортоване з соціальних мереж. Ці дані допомагають ідентифікувати користувача та роблять інтерфейс більш персоналізованим. Користувачі мають можливість редагувати свою особисту інформацію безпосередньо у своєму профілі. Це включає зміну фотографії, оновлення імені та прізвища, а також

додавання інших контактних даних або інформації, яку вони бажають відображати на платформі. Однією з важливих функцій профілю користувача є доступ до збережених курсів. Це дозволяє користувачам швидко знайти та перейти до курсів, які вони раніше зберегли для подальшого перегляду або навчання. Макет сторінки профілю користувача зображений на рисунку 2.5.

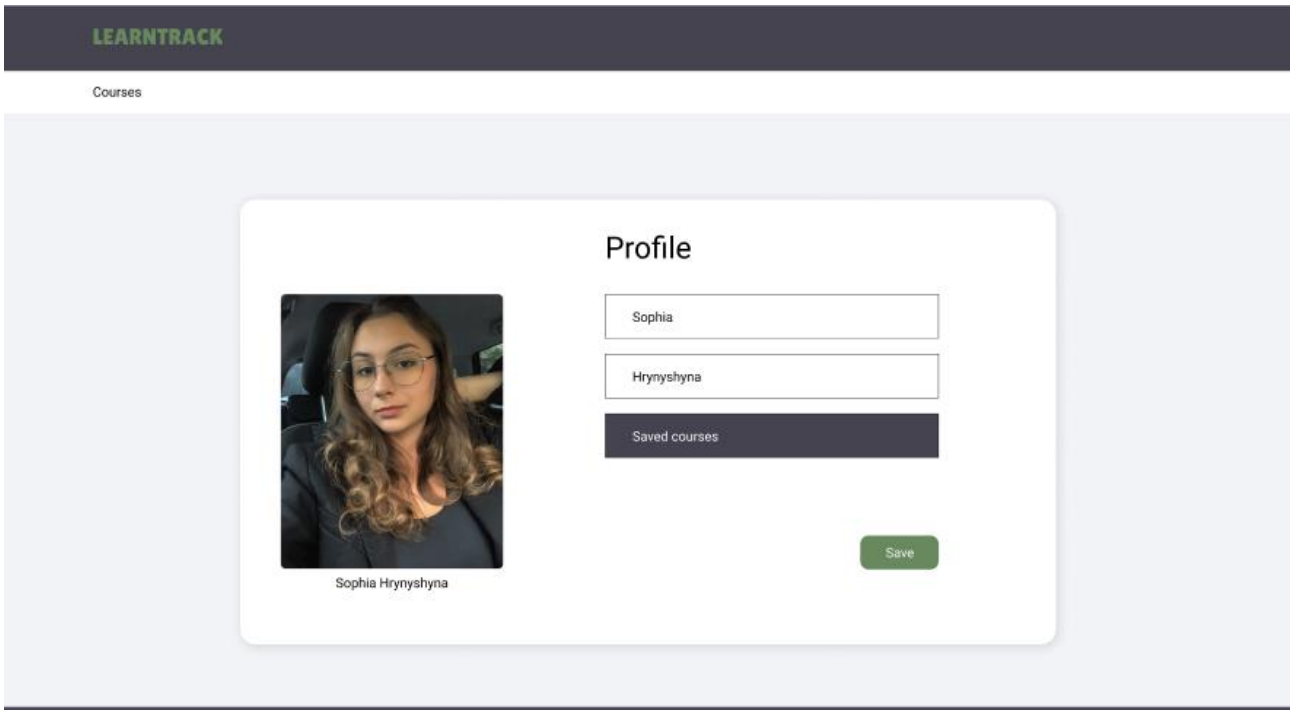


Рисунок 2.5 – Макет сторінки профілю користувача

2.1.6 Додаткові елементи

На головному екрані платформи буде виведено зручний та інтуїтивно зрозумілий список з першими п'ятьма курсами, що дозволить користувачам швидко ознайомитись з найпопулярнішими або найактуальнішими пропозиціями. З метою забезпечення комфортної навігації та збереження оптимальної продуктивності сайту, реалізована буде функція «Завантажити більше», яка дозволить користувачам переглядати додаткові п'ять курсів без перезавантаження сторінки.

Крім того, в верхній частині екрану буде відображатися шлях, який вказуватиме користувачам їхню поточну локацію на сайті. Ця функція буде

працювати як посилання, що дозволить користувачам з легкістю переходити на попередні сторінки або вибирати конкретні розділи сайту за допомогою зручного інтерфейсу. Схему функцій, доступних користувачам можна переглянути на рисунку 2.6.

Рисунок 2.6 – Схема функцій, доступних користувачам

Такий підхід до організації інтерфейсу платформи дозволить забезпечити користувачам максимальну зручність та ефективність взаємодії з сайтом, сприяючи збільшенню їхньої задоволеності від використання платформи для пошуку та вибору курсів.

2.2 Технічні вимоги

2.2.1 Масштабованість

Масштабованість є критичною для платформи з управління онлайн курсами, оскільки вона повинна бути здатна ефективно обробляти великий обсяг користувачів та навчального контенту. Для цього використовуються різні підходи, включаючи горизонтальне та вертикальне масштабування.

Використання технологій мікросервісної архітектури може допомогти розбити функціональність платформи на окремі компоненти, які можна масштабувати незалежно один від одного. Це дозволяє гнучко реагувати на зміну навантаження та оптимізувати використання ресурсів.

Додатково, розподілене обчислення дозволяє розподілити завдання обробки даних між різними серверами, що сприяє паралельній обробці та підвищує продуктивність системи при великому обсязі роботи. Такий підхід дозволяє забезпечити швидку та безперебійну роботу платформи навіть при значному навантаженні.

2.2.2 Безпека даних

Для забезпечення безпеки даних, система має використовувати комплексний підхід до захисту інформації. Основними компонентами цього підходу є шифрування даних, механізми автентифікації та авторизації, а також система моніторингу доступу та виявлення порушень безпеки.

Шифрування даних в спокої є важливим кроком у захисті конфіденційності інформації. Застосування сучасних методів шифрування, таких як AES (Advanced Encryption Standard), дозволяє захистити дані від несанкціонованого доступу.

Механізми автентифікації та авторизації гарантують, що лише вповноважені користувачі мають доступ до конфіденційної інформації. Це може

бути досягнуто за допомогою паролів, двофакторної аутентифікації, а також ідентифікації за допомогою біометричних даних.

Система моніторингу доступу та виявлення порушень безпеки дозволяє відстежувати дії користувачів і вчасно реагувати на потенційні загрози. Це включає в себе аналіз логів, виявлення незвичайних патернів поведінки та автоматизовані системи сповіщення про можливі порушення.

Застосування стандартів безпеки, таких як SSL/TLS, для захищеної передачі даних через мережу, допомагає забезпечити надійний захист конфіденційності та цілісності інформації під час її транспортування.

2.2.3 Сумісність з різними пристроями

Для забезпечення сумісності з різними пристроями, платформа повинна мати адаптивний дизайн, що означає, що інтерфейс коректно відображається на різних пристроях та екранних розмірах. Використання сучасних технологій, таких як HTML5 та SCSS дозволяє створювати адаптивний інтерфейс, який змінює свої параметри відповідно до розмірів екрану.

Застосування адаптивного дизайну дозволяє забезпечити однаково комфортний та зручний доступ до платформи для користувачів з будь-яких пристроїв, що робить її більш доступною та привабливою для широкого кола аудиторії.

3 ВИБІР ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ

3.1 Фреймворк Angular для розробки front-end частини.

Angular – це веб-фреймворк з відкритим кодом, який підтримується спеціальною командою в Google, надає широкий набір інструментів, API та бібліотек для спрощення та оптимізації робочого процесу розробки. Angular дає надійну платформу, на якій можна створювати швидкі, надійні програми, які масштабуються як з розміром команди, так і з розміром кодової бази [3].

3.1.1 Компоненти Angular

Angular дозволяє розробляти додатки, базуючись на компонентному підході. Це означає, що додаток складається з окремих, незалежних компонентів, кожен з яких відповідає за певну частину функціоналу. Компоненти представляють окремі частини інтерфейсу користувача та мають свою логіку та стиль. Компоненти можуть містити шаблони, стилі та логіку поведінки, що дозволяє розробникам організовувати код у логічній структурі [4]. Такий підхід значно полегшує розробку, тестування та підтримку коду, а також сприяє повторному використанню компонентів у різних частинах додатку.

3.1.2 Двустороннє зв’язування даних та модульність

Фреймворк забезпечує двостороннє зв'язування даних, що дозволяє автоматично синхронізувати дані між моделлю та відображенням. Це робить розробку більш ефективною, оскільки розробники можуть зосередитися на логіці додатку, а не на ручному оновленні відображення.

Підтримує модульність, що дозволяє організувати код у вигляді окремих модулів. Це полегшує управління залежностями та сприяє масштабованості додатку. Наприклад, модулі для роботи з курсами, користувачами, контентом

можуть бути розроблені окремо і підключені до основного додатку при необхідності. Такий підхід дозволяє легко розширювати функціонал платформи, додаючи нові модулі без необхідності внесення змін до вже існуючого коду.

Також Angular використовує декларативний підхід до опису шаблонів, що дозволяє розробникам чітко визначати, як виглядатиме інтерфейс додатка. Це забезпечує більш зрозумілий та підтримуваний код. Шаблони можуть включати директиви Angular, які дозволяють динамічно змінювати вигляд додатка залежно від стану моделі. Схему базової архітектури проєкту зображено на рисунку 3.1.

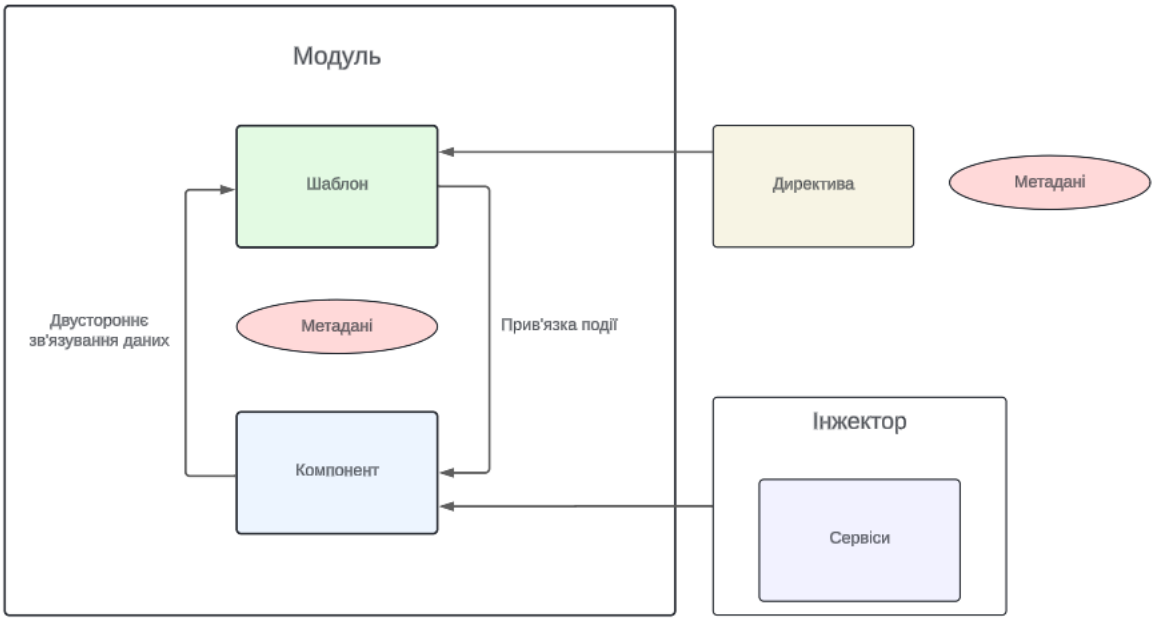


Рисунок 3.1 – Стандартна структура проєкту на Angular

3.1.3 Сервіси

Angular пропонує потужні інструменти для побудови масштабованих додатків. Він використовує концепцію модульної архітектури, що дозволяє розробникам легко розширювати та підтримувати додаток. Крім того, Angular має вбудовану оптимізацію продуктивності, що дозволяє створювати швидкі та ефективні додатки навіть при великій кількості даних та складності логіки [5].

Інструмент надає багатий набір функціональності для розробки веб-додатків. Це включає в себе маршрутизацію, форми, валідацію даних, анімацію, тестування та багато іншого. Завдяки цьому розробники можуть швидко та зручно реалізувати різноманітні функціональність у своїх додатках без необхідності писати власний код з нуля [6].

3.1.4 Інші переваги Angular

Angular побудований на базі TypeScript, що є надбудовою над JavaScript і забезпечує статичну типізацію. Це сприяє покращенню якості коду, полегшує налагодження і підвищує безпеку додатку. Статична типізація дозволяє виявляти помилки на етапі розробки, що значно знижує кількість помилок у готовому продукті. Також фреймворк використовує потужну систему шаблонів, яка дозволяє легко створювати складні та динамічні інтерфейси користувача. Шаблони Angular дозволяють використовувати директиви, які розширюють HTML новими можливостями і дозволяють створювати гнучкі та адаптивні інтерфейси. Вбудована система роутингу дозволяє створювати додатки з багатьма сторінками та динамічними переходами між ними без встановлення додаткових розширень а у разі необхідності Angular легко розширюється за допомогою сторонніх бібліотек і модулів. Це дозволяє додавати нові функціональні можливості до додатку без значних зусиль. Існує велика кількість готових рішень, що дозволяє скоротити час розробки і зосередитися на специфічних вимогах платформи.

Інструмент підтримує різні типи тестів, такі як юніт-тести та інтеграційні, що значно спрощує процес тестування додатку та забезпечує його стабільність та надійність. Також він має потужні інструменти для взаємодії з серверною частиною додатку, такі як HTTPClient для виконання HTTP-запитів. Це дозволяє легко інтегрувати платформу з сервером, обробляти дані курсів, користувачів, автентифікацію та інші серверні операції.

3.2 Фреймворк Jasmine для тестування.

Jasmine є незалежним фреймворком для тестування JavaScript-коду, що працює як у браузерах, так і в середовищі Node.js [7]. Його універсальність дозволяє розробникам використовувати Jasmine для різноманітних проектів, незалежно від середовища виконання. Jasmine підтримує написання тестів як для синхронного, так і для асинхронного коду, що робить його дуже гнучким і зручним для широкого спектра задач. Завдяки цьому фреймворку можна ефективно автоматизувати тестування функціональності програмного продукту, знижуючи ризики помилок і підвищуючи якість кінцевого продукту.

Jasmine має декілька ключових концепцій, які забезпечують структурованість і зрозумілість тестів:

– описи (describe): ця концепція дозволяє групувати тести в логічні блоки, що допомагає структурувати тести і робить їх більш організованими. Наприклад, один опис може містити всі тести для конкретного модуля або функції;

– специфікації (it): специфікації описують конкретні тести, які мають бути виконані. Кожна специфікація являє собою окремий тестовий випадок, що перевіряє певний аспект функціональності;

– очікування (expect): ця концепція використовується для визначення очікуваних результатів тесту. Вона дозволяє перевірити, чи відповідають фактичні результати виконання коду очікуваним, що є основою для визначення успішності або невдачі тесту.

Jasmine має кілька суттєвих переваг, які роблять його привабливим вибором для розробки платформи з курсами: простий синтаксис, який значно полегшує процес написання тестів, підтримка асинхронного коду дозволяє без проблем тестувати асинхронний код, що є критично важливим для сучасних веб-додатків, які часто залежать від асинхронних операцій, таких як запити до серверу або таймери. Jasmine також має велику і активну спільноту розробників, яка надає підтримку та ресурси. Це включає документацію, навчальні матеріали,

приклади коду та допомогу в вирішенні проблем, що значно спрощує процес навчання і використання фреймворку.

3.3 Платформа Node.js для розробки back-end частини.

Node.js — це кросплатформне середовище виконання JavaScript з відкритим вихідним кодом, що стало популярним інструментом для різноманітних проектів. Основою Node.js є рушій V8 JavaScript, який використовується в браузері Google Chrome, що дозволяє Node.js працювати продуктивно поза браузером.

Особливістю Node.js є те, що додатки працюють в одному процесі без створення нових потоків для кожного запиту. Node.js забезпечує набір асинхронних примітивів введення-виведення у своїй стандартній бібліотеці, що запобігає блокуванню коду JavaScript. Як правило, бібліотеки в Node.js пишуться з використанням неблокуючих парадигм, що робить блокуючу поведінку винятком, а не нормою.

Під час виконання операцій введення-виведення, таких як читання з мережі, доступ до бази даних або файлової системи, Node.js не блокує потік, очікуючи на відповідь. Натомість він відновлює операції, коли відповідь повертається. Це дозволяє Node.js обробляти тисячі одночасних з'єднань з одним сервером, уникаючи складнощів управління паралелізмом потоків, що часто є джерелом помилок.

Node.js має унікальну перевагу, оскільки мільйони фронтенд-розробників, які пишуть JavaScript для браузера, можуть використовувати ті ж навички для написання серверного коду. Це знімає потребу у вивченні нової мови для розробки серверної частини додатків.

Крім того, Node.js дозволяє використовувати нові стандарти ECMAScript без затримок, пов'язаних з оновленням браузерів користувачів. Розробники можуть самостійно обирати версію ECMAScript, змінюючи версію Node.js або

увімкнувши певні експериментальні функції за допомогою прапорців при запуску Node.js.

Node.js є відмінним вибором для розробки back-end частини платформи управління онлайн-курсами завдяки своїй високій продуктивності, масштабованості, підтримці реального часу, широкому вибору модулів, а також активній спільноті розробників. Використання Node.js забезпечить стабільну та ефективну роботу нашої платформи, дозволяючи обслуговувати велику кількість користувачів та забезпечуючи високу якість сервісу.

3.4 Препроцесор SCSS для стилізації

CSS-препроцесори – це надбудови над CSS, спеціальні скрипти, які розширюють можливості CSS та спрощують процес створення стилів, що потім вбудовуються в CSS файли. Препроцессори надають можливість використовувати змінні, умовні оператори, цикли та інші зручні функції, які відсутні в звичайному CSS. Це робить написання та підтримку стилів більш ефективними та організованими [8].

SCSS є одним з найпопулярніших препроцесорів CSS, розробленим в 2007 році в рамках проекту Sass (Syntactically Awesome StyleSheets) Хамптона Кетліна (Hampton Catlin) [8].

Основні переваги, на які я спиралась під час вибору препроцесора:

– використання змінних: SCSS дозволяє визначати змінні, що значно спрощує управління кольоровою схемою, шрифтами та іншими параметрами дизайну. Змінивши значення змінної в одному місці, можна автоматично оновити всі стилі, які використовують цю змінну;

– вкладеність: SCSS підтримує вкладеність стилів, що дозволяє структурувати CSS-код так само, як HTML. Це підвищує читабельність та організованість коду, полегшуючи його підтримку;

– міксини та розширення: SCSS дозволяє створювати міксини (функції стилів) та використовувати розширення (extend), що сприяє повторному використанню коду. Це зменшує дублювання коду та покращує його структуру.

– умовні оператори та цикли: використання умовних операторів та циклів в SCSS дозволяє динамічно генерувати стилі, що значно підвищує гнучкість та функціональність;

– імпорт файлів: SCSS підтримує розбиття стилів на окремі файли та їх імпорт, що сприяє кращій організації коду. Це особливо корисно у великих проєктах, де стилі можуть бути розділені на модулі;

– сумісність з CSS: SCSS є розширенням CSS, тому будь-який існуючий CSS-код є коректним SCSS-кодом. Це дозволяє легко інтегрувати SCSS в існуючі проєкти без необхідності переписувати всі стилі з нуля.

4 РОЗРОБКА FRONT-END ЧАСТИНИ ПЛАТФОРМИ

4.1 Концепція назви

Вибір назви "LearnTrack" для платформи навчання була ретельно обрана з урахуванням кількох важливих факторів, що відображають місію та цілі платформи, а також її функціональні можливості.

По-перше, слово "Learn" вказує на основний фокус платформи – навчання. Це слово є зрозумілим та одразу асоціюється з освітніми процесами, що привертає увагу потенційних користувачів, які шукають ресурси для свого професійного або особистісного розвитку.

По-друге, слово "Track" підкреслює можливість відстеження прогресу користувачів у навчанні. Це слово додає відчуття структурованості та організованості, що є важливим для користувачів, які бажають мати чітке уявлення про свої досягнення та наступні кроки у процесі навчання. Функції відстеження можуть включати моніторинг завершених курсів, отриманих сертифікатів, а також прогресу в поточних навчальних програмах.

Поєднання цих двох слів створює назву, яка не лише легко запам'ятовується, але й чітко відображає основні функції та переваги платформи. "LearnTrack" асоціюється з процесом навчання, який знаходиться під постійним контролем і моніторингом, забезпечуючи користувачів можливістю відстежувати свої успіхи та коригувати свій навчальний шлях у разі необхідності.

Крім того, назва "LearnTrack" є короткою, лаконічною та зручною для вимови, що робить її легкою для запам'ятовування та пошуку в інтернеті. Це важливий аспект з точки зору маркетингу та брендингу, оскільки прості та зрозумілі назви мають більше шансів залишитися в пам'яті користувачів і привернути їхню увагу.

4.2 Дизайн платформи

При створенні дизайну освітньої платформи LearnTrack використано поєднання темних та світлих відтінків, що створює сучасний і професійний вигляд інтерфейсу. Основний фон платформи має світло-сірий колір, який є нейтральним і не відволікає користувачів від контенту. Верхня панель навігації виконана в темно-синьому кольорі, що додає контрасту та допомагає виділити назву платформи "LEARNTRACK" і опції входу/виходу.

Назви курсів та кнопки на платформі виділені насиченим зеленим кольором, що асоціюється з навчанням, зростанням та розвитком. Цей колір також використовується для кнопок "Edit" та "Delete", роблячи їх легко помітними і доступними для користувачів. Текстова інформація про курси виконана у чорному кольорі, що забезпечує хорошу читабельність на світлому фоні.

Заголовки курсів виділені жирним шрифтом, що дозволяє користувачам швидко ідентифікувати назви курсів. Додатково, важлива інформація, така як тривалість курсу та дати, представлена сірим кольором меншого розміру, щоб не перевантажувати основний текст, але при цьому залишатися доступною для огляду.

Курс, позначений як рекомендований, виділений світло-жовтим фоном, що привертає увагу користувачів і дозволяє легко знайти важливий або популярний курс серед інших. Загальний підбір кольорів створює приємний для очей інтерфейс, полегшує навігацію та забезпечує гармонійний вигляд платформи.

При наведенні миші на кнопки вони змінюють колір, це явище називається "ефект наведення" (hover effect) і має кілька ключових функцій, які покращують взаємодію користувача з платформою:

– зворотний зв'язок для користувача: коли кнопка змінює колір при наведенні, це дає користувачеві миттєвий візуальний зворотний зв'язок, що кнопка є інтерактивним елементом. Це підтверджує, що користувач може клікнути на цю кнопку для виконання певної дії.

– підвищення естетичної привабливості: ефект наведення додає динаміки до статичного дизайну сторінки, роблячи його більш привабливим і сучасним. Це може сприяти кращому враженню користувачів від роботи з платформою.

– візуальна індикація фокусу: для користувачів, які користуються клавіатурою для навігації, ефект наведення також може слугувати візуальною підказкою про те, на який елемент зараз спрямована увага. Це особливо корисно для покращення доступності сайту.

Загалом, зміна кольору кнопок при наведенні миші підвищує зручність та інтуїтивність користування платформою, покращує візуальну привабливість і сприяє більш ефективній взаємодії з інтерфейсом. Приклади зміни кнопок при наведенні на них курсору можна побачити на рисунках 4.1, 4.2 та 4.3, 4.4.

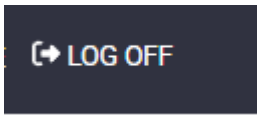


Рисунок 4.1 – Кнопка “Log off” без наведення курсору миші

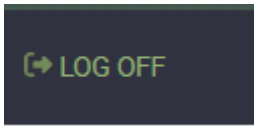


Рисунок 4.2 – Кнопка “Log off” при наведенні курсору миші

Рисунок 4.3 – Кнопка “Delete” без наведення курсору миші

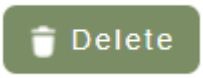


Рисунок 4.4 – Кнопка “Delete” при наведенні курсору миші

4.3 Структура front-end частини

Проєкт Angular має чітку та організовану структуру, що сприяє підтримці коду, його читабельності та масштабованості. Основні компоненти та папки структури проєкту включають “.angular”, “node_modules”, “src”, “assets”, та конфігураційні файли.

Папка “.angular” містить конфігураційні файли Angular, які використовуються для налаштування середовища розробки та збірки. Папка “node_modules” зберігає всі залежності та пакети, встановлені через npm. Це включає бібліотеки, необхідні для роботи Angular та інших додаткових пакетів.

Головна папка проєкту, “src”, містить весь код додатку і включає підпапки та файли, необхідні для роботи додатку. Вона включає папку “app”, яка містить різні модулі та компоненти. Під-папка “account” містить компоненти та модулі, пов'язані з обліковими записами користувачів. “add-edit-course” відповідає за компоненти, які дозволяють додавати та редагувати курси. Папка core містить основні сервіси та інфраструктуру додатку, наприклад, сервіси для роботи з API. Директорія “courses” включає компоненти та модулі для управління курсами, їх відображення та взаємодії з ними. Тека “login” містить компоненти, необхідні для авторизації користувачів. Папка “shared” зберігає спільні модулі, компоненти та сервіси, які використовуються в різних частинах додатку. Каталог store включає в себе управління станом додатку за допомогою бібліотек, таких як NgRx. “styles” містить глобальні стилі додатку.

Крім того, у папці “app” є важливі файли, такі як “app-routing.module.ts”, який є файлом маршрутизації додатку і визначає шляхи та компоненти, що відображаються за цими шляхами. Файл “app.component.html” є головним шаблоном додатку, а “app.component.scss” містить головні стилі для основного компонента. Файл “app.component.spec.ts” містить тести для основного компонента, а “app.component.ts” є основним компонентом додатку. Файл “app.module.ts” є головним модулем додатку, який імпортує інші модулі та налаштовує додаток.

Папка “assets” призначена для зберігання статичних файлів, таких як зображення, шрифти та інші ресурси. Файл “index.html” є головним HTML файлом додатку, який завантажується при старті. “main.ts” є основним файлом входу для запуску додатку, а “styles.scss” містить глобальні стилі для всього додатку.

Конфігураційні файли включають “.editorconfig”, який містить налаштування редактора коду, “.eslintrc.json” для налаштувань ESLint, інструменту для перевірки коду на помилки та стиль, та “.gitignore”, що визначає, які файли та папки ігнорувати системі контролю версій Git. Файл “angular.json” є головним конфігураційним файлом Angular CLI, що містить налаштування проєкту. Файл “package-lock.json” автоматично зберігає точні версії встановлених npm пакетів для відтворюваності, а “package.json” містить метадані про проєкт та залежності, необхідні для його роботи.

Ця структура проєкту дозволяє зручно організовувати код, розподіляючи різні частини додатку по окремих папках та файлах, що значно полегшує його підтримку та масштабування.

4.3.1 Розробка головної сторінки

Основні компоненти та функціонал головної сторінки веб-додатку LearnTrack включають в себе панель пошуку, кнопку додавання курсу, список курсів у вигляді карток, повідомлення про відсутність курсів і кнопку для завантаження додаткових курсів. Розглянемо детально структуру та логіку цієї сторінки.

Основний контейнер сторінки визначено за допомогою класу content, який об'єднує всі внутрішні елементи. Всередині нього розташовані секції для пошуку, додавання курсів, відображення курсів та повідомлень про їх відсутність. У секції section розміщена панель пошуку та кнопка додавання курсу. Компонент app-search-bar відповідає за пошук курсів, використовуючи двосторонню прив'язку даних. Подія (search) викликає метод

onSearchCourse($event), який обробляє введений пошуковий запит. Кнопка "Add course" дозволяє користувачам додавати нові курси. При натисканні на кнопку викликається метод onAddCourse(), який відкриває форму для додавання нового курсу.

Секція “courses” містить список курсів, представлених у вигляді карток. Для цього використовується компонент app-course-card, який відображає інформацію про кожен курс. Директива *ngFor проходить по всіх курсах у масиві courses та відображає кожен курс за допомогою компонента app-course-card. Метод trackByFn використовується для оптимізації відображення списку курсів.

Секція “emptyCourses” відображає повідомлення "NO DATA. FEEL FREE TO ADD NEW COURSE", якщо список курсів порожній. Це визначається за допомогою методу isCoursesEmpty(), який повертає true, якщо в масиві курсів немає жодного елемента.

Секція “loadMore” містить кнопку "LOAD MORE", яка дозволяє завантажувати додаткові курси. По замовчуванню на сторінці відображаються лише 5 карток. Це зроблено з метою оптимізації сайту, аби скоротити час завантаження контенту. При натисканні на кнопку викликається метод onLoadMore(), який завантажує додаткові курси.

Класи “content”, “section”, “courses”, “emptyCourses”, та “loadMore” використовуються для організації та стилізації елементів на сторінці. Вони забезпечують належне розташування та вигляд елементів, роблячи сторінку зручною та привабливою для користувачів.

Html-шаблон компоненту “courses” зображено на рисунку 4.5.

```
<div class="content">
  <div class="section">
    <app-search-bar (search)="onSearchCourse($event)"></app-search-bar>
    <button type="submit" (click)="onAddCourse()" class="btn">
      Add course
    </button>
  </div>
  <div class="courses">
    <app-course-card
      *ngFor="let course of courses ; trackBy: trackByFn"
      [course]="course"
      (loadCoursesEvent)="onLoadCourses()"
    ></app-course-card>
  </div>
  <div class="emptyCourses" *ngIf="isCoursesEmpty()">
    NO DATA. FEEL FREE TO ADD NEW COURSE
  </div>
  <div class="loadMore" *ngIf="!isCoursesEmpty()">
      <button class="loadMore" (click)="onLoadMore()">LOAD MORE</button>
  </div>
</div>
```

Рисунок 4.5 – Html-шаблон компоненту “courses”

Typescript файл використовує ряд Angular бібліотек та служб для забезпечення необхідного функціоналу, таких як Component, OnInit, ChangeDetectorRef, EventEmitter, Router, ActivatedRoute, NavigationEnd, Store, і специфічні для додатку модулі та служби, такі як CourseService, LoadingService, CoursesActions і selectCourses. Компонент визначається за допомогою декоратора @Component, який містить метадані, включаючи селектор, шаблон і стилі компонента, а також провайдери служб. Ознайомитись з імпортом бібліотек та ініціалізацією компонента можна на рисунку 4.6.

```
1  import {
2    Component,
3    OnInit,
4    ChangeDetectorRef,
5    Output,
6    EventEmitter,
7  } from '@angular/core';
8  import { Router, ActivatedRoute, NavigationEnd } from '@angular/router';
9  import { Course } from '../shared/models/course.model';
10 import { CourseService } from '../shared/services/courseService/course-service.service';
11 import { LoadingService } from '../shared/services/loading-service/loading-service.service';
12 import { Store, select } from '@ngrx/store';
13 import * as CoursesActions from '../store/courses/courses.actions';
14 import { selectCourses } from '../store/courses/courses.selectors';
15
   You, 5 months ago | 1 author (You)
16 @Component({
17   selector: 'app-courses',
18   templateUrl: './courses.component.html',
19   styleUrls: ['./courses.component.scss'],
20   providers: [CourseService],
21 })
```

Рисунок 4.6 – Імпорт бібліотек та ініціалізація компонента

Клас CoursesPageComponent імплементує інтерфейс OnInit і використовує властивості courses, filteredCourses, isLoading, courses_start і courses_count для управління станом і даними. Конструктор компонента ініціалізує необхідні служби, такі як CourseService, Router, ActivatedRoute, ChangeDetectorRef, LoadingService і Store. Під час ініціалізації компонента в методі ngOnInit виконується диспетчеризація дії getCourses для завантаження початкового списку курсів. Підписка на зміни в стані стору забезпечує автоматичне оновлення списку курсів. Крім того, компонент підписується на події маршрутизатора, щоб відстежувати зміни URL і оновлювати відображення компонентів при навігації. На рисунку 4.7 зображено метод “onInit”, створення необхідних змінних та опис конструктору.

```
22  export class CoursesPageComponent implements OnInit {
23    @Output() loadCoursesEvent = new EventEmitter<void>();
24    courses!: Course[];
25    filteredCourses: Course[] = [];
26    isLoading = false;
27    courses_start = 4;
28    courses_count = 5;        You, 5 months ago • Finished project
29    constructor(
30      private readonly courseService: CourseService,
31      private readonly router: Router,
32      private readonly route: ActivatedRoute,
33      private readonly changeDetectorRef: ChangeDetectorRef,
34      private readonly loadingService: LoadingService,
35      private readonly store: Store
36    ) {}
37
38    ngOnInit(): void {
39      this.store.dispatch(CoursesActions.getCourses({ start: 0, count: 5 }));
40      this.store.pipe(select(selectCourses)).subscribe((courses) => {
41        this.courses = courses;
42      });
43      this.router.events.subscribe((event) => {
44        if (event instanceof NavigationEnd) {
45          const currentUrl = event.urlAfterRedirects;
46          if (currentUrl === '/courses') {
47            this.changeDetectorRef.markForCheck();
48          }
49        }
50      });
51    }
52
```

Рисунок 4.7 – Метод “onInit”, створення змінних та опис конструктору

Функція onSearchCourse виконує диспетчеризацію дії для пошуку курсів за введеним користувачем значенням. Після виконання дії, компоненти отримують відфільтровані курси і оновлюють список курсів. Функція onAddCourse перенаправляє користувача на сторінку додавання нового курсу, використовуючи Router. З кодом функцій можна ознайомитись на рисунку 4.8.

```
onSearchCourse(searchName: string) {
  this.store.dispatch(
    CoursesActions.getCourses({
      start: 0,
      count: undefined,
      searchValue: searchName,
    })
  );
  this.store.pipe(select(selectCourses)).subscribe((filteredCourses) => {
    this.courses = filteredCourses;
  });
}
onAddCourse(): void {
  this.router.navigate(['new'], { relativeTo: this.route });
}
```

Рисунок 4.8 – Код функцій “onSearchCourse” та “onAddCourse”

Функція onLoadCourses відповідає за завантаження списку курсів з сервера. Вона змінює стан isLoading на true перед початком запиту і повертає його в false після завершення, а також оновлює список курсів і змінює стан завантаження за допомогою LoadingService. Код функції наведено на рисунку 4.9.

```
onLoadCourses(): void {
  this.isLoading = true;
  this.courseService.getCoursesList(0, 5).subscribe((courses) => {
    this.isLoading = false;
    this.courses = courses;
    this.loadingService.toggleLoading(false);
  });
}
```

Рисунок 4.9 – Код функцій “onLoadCourses”

Функція onLoadMore дозволяє користувачам завантажувати додаткові курси, диспетчеризуючи дію getCourses з новими параметрами start і count, що дозволяє поступово збільшувати кількість відображуваних курсів. На рисунку 4.10 зображено код цього методу.

```
onLoadMore(): void {
  this.store.dispatch(
    CoursesActions.getCourses({
      start: this.courses_start,
      count: this.courses_count,
    })
  );
  this.store.pipe(select(selectCourses)).subscribe((courses) => {
    this.courses = this.courses.concat(courses);
  });
  this.courses_start += this.courses_count;
}
```

Рисунок 4.10 – Код функцій “onLoadCourses”

Таким чином, CoursesPageComponent забезпечує основний функціонал для управління та відображення списку курсів, використовуючи Angular Router для навігації, NgRx Store для управління станом і різні служби для завантаження даних і відображення стану завантаження. Така структура дозволяє легко масштабувати та підтримувати компонент, забезпечуючи інтерактивність та ефективність роботи з даними.

4.3.2 Розробка картки курсу

Компонент картки курсу відображається у вигляді HTML <section>, який використовує Angular директиви для динамічного відображення даних. Основні елементи цього компонента включають назву курсу, його тривалість, дату, опис, а також кнопки для редагування та видалення курсу. Компонент використовує директиву “[ngClass]”, яка динамічно додає або видаляє класи CSS до елементу. Наприклад, якщо курс має високий рейтинг (isTopRated), до класу додається

“isTopRated”. Директива “*ngIf“ умовно відображає елемент в залежності від значення виразу, що дозволяє, наприклад, відображати зірку лише тоді, коли курс має високий рейтинг. Назва курсу виводиться у верхньому регістрі за допомогою пайпу uppercase, а зірка, яка позначає, що курс є високо оціненим, відображається тільки якщо курс має властивість isTopRated. Тривалість курсу відображається за допомогою кастомного пайпу durationPipe, який перетворює значення тривалості в зручний для читання формат, а дата курсу відображається у форматі "MM/dd/yyyy" за допомогою вбудованого пайпу date. Опис курсу виводиться у <p> елементі, а кнопки "Edit" та "Delete" викликають відповідні методи (onEdit та onDeleteCourse) при натисканні, що дозволяє редагувати або видаляти курс. На рисунку 4.11 зображено код HTML розмітки компоненту

```
<section [ngClass]="{ isTopRated: course.isTopRated }" class="course__card border">
    <div class="course__firstCol">
        <div class="course__row">
            <h1 class="course__title">
                {{ course.name | uppercase }}
                <img class="course__star" *ngIf="course.isTopRated" src="../../../../assets/pics/general/star.png"
                    alt="" />
            </h1>
            <div class="course__details">
                <p class="course__duration">{{ course.length | durationPipe }}</p>
                <p class="course__date">{{ course.date | date : "MM/dd/yyyy" }}</p>
            </div>
        </div>
        <p class="course__description">
            {{ course.description }}
        </p>
    </div>
    <div class="course__secondCol">
        <button class="btn btn__edit" (click)="onEdit()">Edit</button>
        <button class="btn btn__delete" (click)="onDeleteCourse()">Delete</button>
    </div>
</section>      You, 5 months ago • Finished project
```

Рисунок 4.11 – Код html розмітки компоненту “CourseCard”

Для реалізації функціоналу цього елементу компонент імпортує кілька модулів та залежностей, зокрема “Component”, “Input”, “OnInit”, “Output”, “EventEmitter” з пакету @angular/core для створення та ініціалізації компонента, модель “Course”, яка описана в окремому файлі, для типізації даних курсу, “CourseService” для роботи з даними курсу, а також “Router”, “ActivatedRoute” з бібілотеки “router” для маршрутизації і “Store” з “ngrx/store” для управління

станом додатка. Компонент має властивість “course, яка отримує дані курсу від батьківського компонента, та подію “loadCoursesEvent”, яка виконується при зміні даних курсу. Конструктор компонента отримує залежності, включаючи сервіси “courseService” для роботи з курсами, “router” для навігації, “route” для отримання даних про маршрут, а також “store” для управління станом за допомогою вбудованого в фреймворк Angualr “NgRx Store”. На рисунку 4.12 зображено імпорти для компонента.

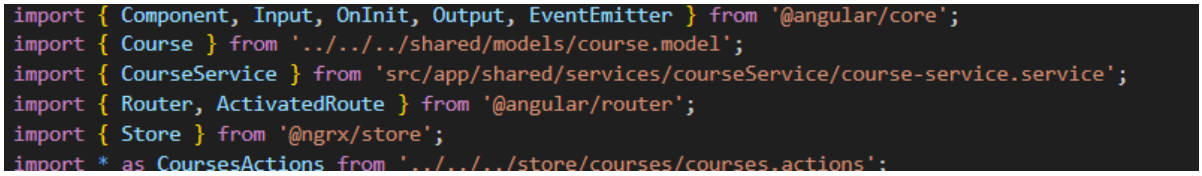


Рисунок 4.12 – Код імпортів для компонента “CourseCard”

Метод ngOnInit викликається при ініціалізації компонента і підписується на подію “wasSomethingChange” сервісу “courseService”. Якщо відбуваються зміни, викликається подія “loadCoursesEvent” для перезавантаження курсів. Метод “onDeleteCourse” викликається при натисканні на кнопку видалення курсу, показує повідомлення з питанням, чи дійсно ви хочете видалити курс, і, у разі підтвердження, дає команду системі видалити цей курс, використовуючи його унікальний номер. На рисунку 4.13 зображено код функції.

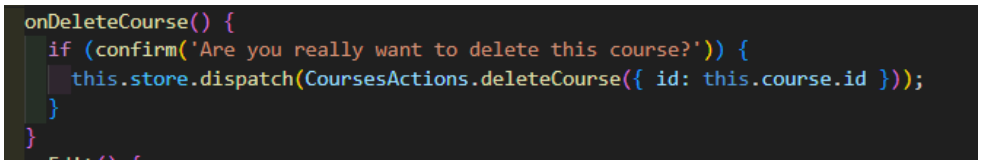


Рисунок 4.13 – Код функції “onDeleteCourse”

Метод onEdit викликається при натисканні на кнопку редагування курсу і використовує маршрутизатор для навігації до сторінки редагування курсу, зберігаючи поточні параметри запиту. З кодом методу можна ознайомитись на рисунку 4.14.

```
onEdit() {
  this.router.navigate([this.course.id], {
    relativeTo: this.route,
    queryParamsHandling: 'preserve',
  });
```

Рисунок 4.14 – Код функції “onEdit”

Стилізація компоненту відбувається у файлі “course-card.component.scss” за допомогою перпроцесора SCSS.

Сама картка має білий фон, заокруглені краї, тінь для створення глибини, а також стилі для внутрішніх відступів та розташування елементів. Кнопки всередині картки мають основний зелений колір, а також використовують міксин для додаткових стилів. Код стилізації наведено на рисунку 4.15.

```
.course__card {
    padding: 2vw;
    margin-bottom: 20px;
    background-color: white;
    border-radius: 5px;
    box-shadow: rgba(0, 0, 0, 0.171) 0px 2px 10px;
    display: flex;
    line-height: 1.5;
    justify-content: space-between;
    color: $darkGray;
    .btn {
        background-color: rgb(48, 182, 221);
        @include button($greenMain)
    }
```

Рисунок 4.15 – Код стилізації картки

Перший стовпець картки містить інформацію про курс і має додаткові відступи. Внутрішній рядок “.course__row” організовує заголовок курсу, деталі про тривалість і дату курсу, а також відображає зірочку для рекомендованих курсів. Для кожної деталі курсу використовується іконка перед текстом, яка додається за допомогою міксинів. Приклад коду зображено на рисунку 4.16.

Рисунок 4.16 – Код стилізації першого стовпця

Другий стовпець містить кнопки для редагування та видалення курсу, кожна з яких також містить відповідну іконку. Картки рекомендованих курсів отримують інший фон для виділення. Додаткові класи “.border-fresh” і “.border-upcoming” змінюють кольори рамок для різних станів курсів. Вся ця стилізація забезпечує чітке та привабливе відображення картки курсу, роблячи її зручною для взаємодії та візуально приємною для користувачів. З кодом можна ознайомитись на рисунку 4.17.

Рисунок 4.17 – Код стилізації другого ствопця

Результатом є картка курсу, яка містить в собі заголовок, опис, тривалість та дату створення курсу, дві кнопки зеленого кольору для редагування та

видалення відповідно. Елемент має заокруглені поля, чорний колір основного тексту, та сірий для додаткового. На рисунку 4.18 зображена розроблена картка курсу “Introduction To Python Programming”, тривалістю 9 годин, створеного 31 травня 2024 року. На рисунку 4.19 зображена картка курсу, який є рекомендованим, має назву “Web Development Fundamentals”, триває 65 год. 20 хв. та створений 30 квітня 2024.

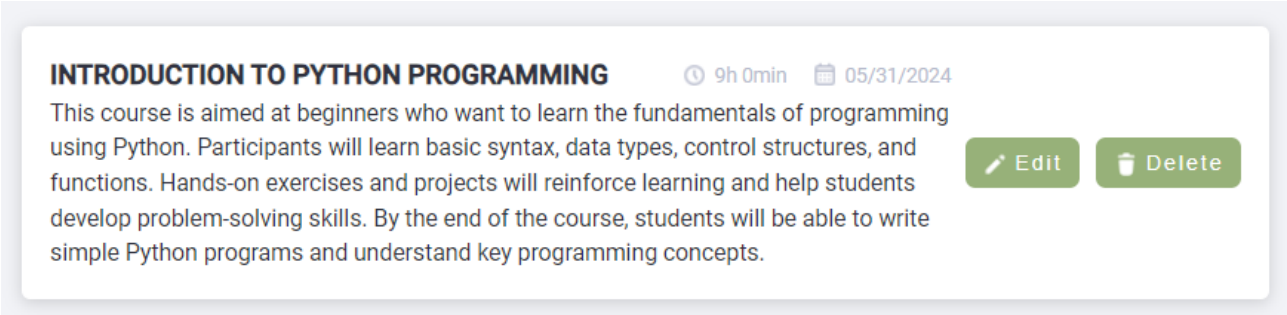


Рисунок 4.18 – Картка курсу “Introduction To Python Programming”

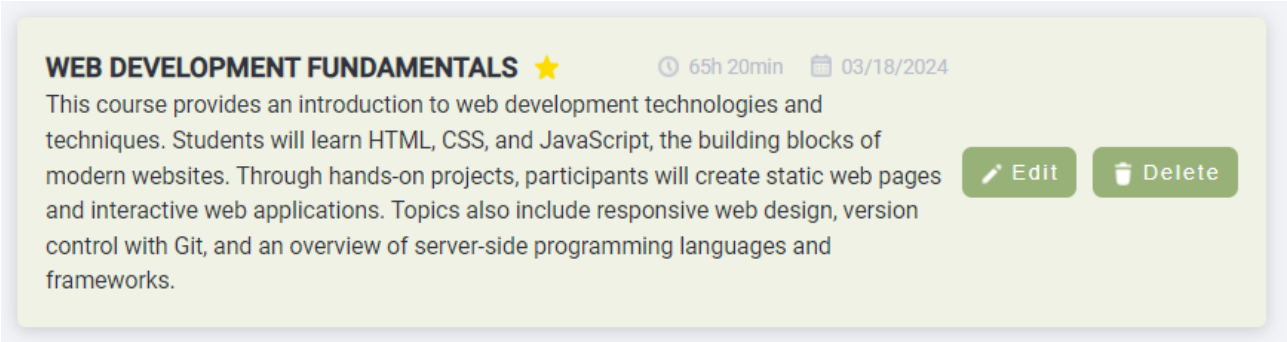


Рисунок 4.19 – Картка курсу “Web Development Fundamentals”

У рамках тестування компонента “CourseCardComponent” було використано набір інструментів “Angular Testing”, які включають модулі та сервіси для створення і проведення тестів. Ці тести перевіряють функціональність та коректність відображення інформації про курс, а також обробку події видалення курсу. Для налаштування тестового середовища було використано імпорти необхідних модулів, таких як “HttpClientTestingModule” та “RouterTestingModule”, а також декларування компонентів

“CourseCardComponent”, “BorderChangeDirective” і “DurationPipe”. З кодом можна ознайомитись на рисунку 4.20.

```
You, 5 months ago | 1 author (You)
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { HttpClientTestingModule } from '@angular/common/http/testing';
import { RouterTestingModule } from '@angular/router/testing';

import { DurationPipe } from 'src/app/shared/pipes/durationpipe/duration.pipe';
import { CourseCardComponent } from './course-card.component';
import { BorderChangeDirective } from 'src/app/shared/directives/border-change/border-change.directive';
import { CourseService } from 'src/app/shared/services/courseService/course-service.service';
```

Рисунок 4.20 – Код імпортів необхідних модулів

Спочатку створюється тестове середовище за допомогою методу “TestBed.configureTestingModule”, де включаються необхідні модулі, декларуються компоненти та надаються сервіси. Потім створюється тестовий компонент за допомогою методу “TestBed.createComponent”, і задається тестовий курс з попередньо визначеними властивостями, такими як “id”, “name”, “description”, “isTopRated”, “date”, “authors” та “length”. Після цього викликається метод “fixture.detectChanges()”, щоб застосувати зміни в шаблоні компонента. Код створення тестового компоненту зображено на рисунку 4.21.

```
describe('CourseCardComponent', () => {
    let component: CourseCardComponent;
    let fixture: ComponentFixture<CourseCardComponent>;
    beforeEach(() => {
        TestBed.configureTestingModule({
            declarations: [CourseCardComponent, BorderChangeDirective, DurationPipe],
            providers: [CourseService],
            imports: [HttpClientTestingModule, RouterTestingModule],
        });
        fixture = TestBed.createComponent(CourseCardComponent);
        component = fixture.debugElement.children[0].componentInstance;
        component.course = {
            id: 8693,
            name: 'duis mollit reprehenderit ad',
            description:
                'Est minim ea aute sunt laborum minim eu excepteur. Culpa sint exercitation mollit enim ad culpa aliquip laborum cillum.
            isTopRated: false,
            date: '2017-09-28T04:39:24+00:00',
            authors: [
                {
                    id: 1370,
                    name: 'Polly',
                },
            ],
            length: 157,
        };
        fixture.detectChanges();
    });
```

Рисунок 4.21 – Код створення тестового компоненту

Перший тест перевіряє, чи створюється компонент успішно. За допомогою методу “expect(component).toBeTruthy()” перевіряється, чи існує компонент після його створення. Код функції зображено на рисунку 4.22.

```
it('should create', () => {
    expect(component).toBeTruthy();
});
```

Рисунок 4.21 – Код методу перевірки створення компоненту

Другий тест перевіряє, чи правильно відображається інформація про курс. Використовуючи метод “fixture.nativeElement”, отримуються HTML-елементи, що містять інформацію про курс, такі як назва, тривалість, дата та опис. Потім, за допомогою методу “expect”, перевіряється, чи відповідає текстовий вміст цих елементів значенням властивостей тестового курсу. З кодом можна ознайомитись на рисунку 4.22.

```
it('should display course information corectly', () => {
    const element = fixture.nativeElement;
    const courseTitle = element.querySelector('.course__title');
    const courseDuration = element.querySelector('.course__duration');
    const courseDate = element.querySelector('.course__date');
    const courseDescription = element.querySelector('.course__description');
    expect(courseTitle.textContent.toLowerCase()).toContain(
        component.course.name
    );
    expect(courseDuration.textContent).toContain('2h 37min');
    expect(courseDate.textContent).toContain('09/28/2017');
    expect(courseDescription.textContent).toContain(
        component.course.description
    );
});
```

Рисунок 4.22 – Код методу перевірки відображення інформації про курс

Третій тест перевіряє обробку події видалення курсу. За допомогою методу “spyOn(window, 'confirm').and.returnValue(true)” імітується підтвердження видалення курсу. Потім викликається метод “component.onDeleteCourse()”, який

обробляє подію видалення. Використовуючи метод “expect(window.confirm).toHaveBeenCalledWith('Are you really want to delete this course?')”, перевіряється, чи було викликано вікно підтвердження з відповідним повідомленням. На рисунку 4.23 є можливість ознайомитись з кодом методу.

```
it('should handle course deletion', () => {
    spyOn(window, 'confirm').and.returnValue(true);
    fixture.detectChanges();
    component.onDeleteCourse();
    expect(window.confirm).toHaveBeenCalledWith(
        'Are you really want to delete this course?'
    );
});
```

Рисунок 4.23 – Код методу видалення курсу

Таким чином, цей тестовий код перевіряє, чи компонент CourseCardComponent створюється коректно, чи правильно відображає інформацію про курс та чи правильно обробляє подію видалення курсу. Таке тестування допомагає забезпечити стабільність і надійність компонента в різних сценаріях використання, що є важливою складовою розробки якісного програмного забезпечення.

4.3.3 Розробка сторінки створення та редагування курсу

Компонент створений для управління процесом створення та редагування курсів у системі управління навчальними програмами. Він забезпечує зручний інтерфейс для введення та редагування даних про курси. Компонент складається з кількох основних частин: загального контейнера, секції форми, полів для введення даних і кнопок управління.

Загальний контейнер представлений у вигляді елемента з класом “content”. Цей контейнер використовується для центрування вмісту на сторінці

та забезпечення загального стилю компоненту. Всі інші елементи компонента розміщені всередині нього.

Секція форми представлена елементом <section> з класом “add-edit-course”. Вона містить основну форму для створення або редагування курсу. Заголовок форми, що вказує на те, що це новий курс або редагування існуючого, відображається за допомогою елемента <h2> з класом “add-edit-course__title”.

Форма містить кілька полів для введення даних, кожне з яких має відповідну валідацію. Поле заголовка включає текстове поле для введення назви курсу. Воно має валідацію на обов'язковість заповнення та максимальну довжину введеного тексту. Якщо поле не заповнене або перевищує максимальну довжину, відображається відповідне повідомлення про помилку. Поле опису курсу представлено у вигляді текстової області для введення розширеного опису курсу. Це поле також має валідацію на обов'язковість заповнення та максимальну довжину. Відповідні повідомлення про помилки відображаються, якщо поле не заповнене або перевищує максимальну довжину. Приклад коду цієї частини форми зображено на рисунку 4.24.

```
<div class="content">
    <section class="add-edit-course">
        <form [formGroup]="courseForm" (ngSubmit)="onSubmit()" action="" >
            <h2 class="add-edit-course__title">New course</h2>
            <div class="title__input-container">
                <div class="title__input-text input-text">
                    <label for="title__input">Title</label>
                    <span class="required-indicator">* required</span>
                </div>
                <input [ngClass]="{'error-input': (title?.invalid && title?.touched) || title?.hasError('maxlength')}" formControlName="title"
                type="text" id="title__input" placeholder="Text input" name="title" required />      You, 10 months ago • Uncommitted changes
                <div *ngIf="title?.hasError('required') && title?.touched " class="error">This field is required</div>
                <div *ngIf="title?.hasError('maxlength')" class="error">The maximum length of the title is 50 characters </div>

            </div>
            <div class="description__textarea-container">
                <div class="description__textarea-text input-text">
                    <label for="description__input">Description</label>
                    <span class="required-indicator">* required</span>
                </div>
                <textarea [ngClass]="{'error-input': (description?.invalid && description?.touched) || description?.hasError('maxlength')}"
                formControlName="description" type="text" id="description__input" placeholder="Add description" name="description" required ></textarea>
                <div *ngIf="description?.hasError('required') && description?.touched " class="error">This field is required</div>
                <div *ngIf="description?.hasError('maxlength')" class="error">The maximum length of the description is 500 characters </div>
            </div>
```

Рисунок 4.24 – Перша частина форми створення та редагування курсу

Крім того, форма включає спеціалізовані компоненти для вибору дати, тривалості курсу та авторів курсу. Компонент вибору дати представлений елементом <app-date>, який підключений до відповідного формового контролю

за допомогою директиви “formControlName”. Компонент тривалості курсу представлений елементом <app-duration>, який також підключений до відповідного формового контролю. Компонент вибору авторів курсу представлений елементом <app-authors> і підключений аналогічним чином.

Компонент тривалості курсу складається з кількох основних частин: контейнера для введення тривалості, поля вводу, текстових повідомлень для валідації та відображення відформатованої тривалості. Загальний контейнер для введення тривалості курсу представлений елементом з класом “duration__input-container”. Всередині нього розташовані всі необхідні елементи для введення та валідації даних. Поле вводу тривалості курсу має форму текстового поля, підключене до формового контролю customControl за допомогою директиви [formControl]. Також використовується директива [ngClass], яка додає клас “error-input”, якщо введені дані є некоректними (недійсні або не були введені). HTML-код компоненту можна переглянути на рисунку 4.25.

```
<div class="duration__input-container">
    <div class="duration__input-text">
        <label for="duration__input">Duration</label>
        <span class="required-indicator">* required</span>
    </div>
    <div class="duration__durationValue">
        <input
            type="text" id="title__input"
            placeholder="Text input"
            [ngClass]="{'error-input': customControl.invalid && customControl.touched}"
            [formControl]="customControl"
            required
        />
        <div class="duration_formatted">{{ (customControl.value || 0) | durationPipe }}</div>
    </div>
    <div *ngIf="customControl.hasError('required') && customControl.touched" class="error">This field is required</div>
    <div *ngIf="customControl.hasError('invalidFormat')"  class="error">Only numbers are allowed</div>

</div>      You, 5 months ago • Finished project
```

Рисунок 4.25 – HTML-код компоненту тривалості курсу

Компонент забезпечує валідацію введених даних:

– валідація обов'язковості заповнення (Validators.required).

– валідація формату введених даних (дозволяються тільки числові значення), реалізована за допомогою користувацького валідатора “formatValidator”.

Якщо введені дані є некоректними, відображаються відповідні повідомлення про помилки:

– повідомлення “This field is required”, якщо поле не заповнене.

– повідомлення “Only numbers are allowed”, якщо введені дані мають недійсний формат.

Введене значення тривалості форматовано по шаблону XX год. XX хв. та відображається у елементі з класом duration_formatted за допомогою функції “durationPipe”.

Компонент реалізує інтерфейси ControlValueAccessor та Validator для забезпечення інтеграції з Angular формами. Методи writeValue, registerOnChange, registerOnTouched використовуються для інтеграції з модельним підходом форм, а метод validate забезпечує валідацію компоненту. Приклад коду зображено на рисунку 4.26.

Компонент дати курсу дозволяє користувачам вводити дату для курсу у форматі день/місяць/рік. Віджет введення дати представлений елементом <input> з типом “date”, що дозволяє користувачам вибирати дату з вбудованого календаря. Поряд з ним є мітка "Date" та позначка обов'язковості поля "* required". Вони допомагають користувачам зрозуміти, що введення дати є обов'язковим. Якщо поле для введення дати залишено порожнім та воно було "торкнуте" (змінено користувачем), то виводиться повідомлення "This field is required", щоб попередити про необхідність заповнення поля. Компонент також перевіряє введену дату на коректність формату - день/місяць/рік. Якщо формат не відповідає вказаному шаблону, відображається повідомлення про помилку "invalidFormat". Також в такому випадку буде передана інформація батьківському компоненту “AddEditCourseComponent” про те, що поле не

заповнене або заповнене некоректно і користувач не зможе зберегти курс, кнопка буде недійсною.

```
export class DurationComponent implements ControlValueAccessor, Validator {
  constructor() {
    this.customControl.valueChanges.subscribe((value) => {
      this.onChange(value);
      this.onTouched();
    });
  }
  private onChange: any = () => {};
  private onTouched: any = () => {};
  customControl = new FormControl(0, [
    Validators.required,
    this.formatValidator.bind(this),
  ]);
  validate(): ValidationErrors | null {
    return this.customControl.errors;
  }

  writeValue(value: any): void {
    this.customControl.setValue(value);
  }
  registerOnChange(fn: any): void {
    this.onChange = fn;
  }
  registerOnTouched(fn: any): void {
    this.onTouched = fn;
  }

  private formatValidator(
    control: AbstractControl
  ): { [key: string]: any } | null {
    const value = control.value;
    if (value) {
      const isValid = /^\d+$/.test(value);
      if (!isValid) {
        return { invalidFormat: true };
      }
    }
    return null;
  }
  isNumber(value: any): boolean {
    return typeof value === 'number';
  }
}
```

Рисунок 4.26 – Typescript-код компоненту тривалості курсу

Компонент для вибору авторів курсу використовується для введення та вибору авторів для певного курсу в системі управління навчальними програмами. Цей компонент є частиною інтерфейсу користувача, створеного з використанням бібліотеки Angular Material.

У шаблоні компонента ми маємо поле введення з можливістю автодоповнення, де користувач може вводити імена авторів. Кожен обраний автор відображається у вигляді чіпа, який містить ім'я автора та кнопку для видалення. Користувач може вибирати авторів, вводячи їх імена у поле введення або обираючи їх із списку автодоповнення.

У класі компонента міститься логіка взаємодії з даними та обробки подій. Компонент інтегрується зі стором за допомогою ngrx для отримання списку авторів. Використовуються функціонал Angular для реалізації інтерфейсу ControlValueAccessor та Validator для інтеграції компонента з Angular-формами та проведення валідації даних.

Для створення зручного інтерфейсу вибору авторів використовуються компоненти Angular Material, такі як mat-form-field, mat-chip, mat-autocomplete. Компонент також використовує LiveAnnouncer для оголошення подій з метою поліпшення доступності. На рисунку 4.27 можна ознайомитись з прикладом інтерфейсу вибору авторів.

Компонент проводить валідацію даних, перевіряючи, чи вибрано хоча б одного автора. При відсутності обраних авторів кнопка збереження курсу стає недоступною.

В компоненті створення та редагування курсу Кнопки управління розташовані в нижній частині форми і дозволяють користувачу зберегти або скасувати зміни. Кнопка "Save" має тип submit і буде активована лише тоді, коли всі поля форми заповнені коректно. Кнопка "Cancel" дозволяє користувачу скасувати поточні зміни і повернутися до списку курсів.

У компоненті використовуються Angular форми, що керуються моделлю (Reactive Forms). При створенні форми використовується сервіс FormBuilder для зручного налаштування форми та її валідації. Компонент також інтегрований з NgRx Store для управління станом додатку. Зокрема, при ініціалізації компонента здійснюється завантаження списку авторів з Store та, у разі редагування курсу, завантажуються дані про курс для їх попереднього заповнення у форму.

Для валідації полів форми використовуються стандартні валідатори Angular, такі як Validators.required для перевірки обов'язковості заповнення полів та Validators.maxLength для перевірки максимальної довжини введеного тексту. Динамічна зміна класів полів форми здійснюється за допомогою директиви [ngClass], яка дозволяє додавати класи до елементів залежно від стану валідації полів форми.

Таким чином, компонент для створення та редагування курсу забезпечує повний цикл управління даними курсу, включаючи їх валідацію, інтерактивний вибір авторів, дати та тривалості, а також інтеграцію з централізованим сховищем даних за допомогою NgRx Store. Це дозволяє створити зручний та надійний інтерфейс для користувачів системи управління навчальними програмами. З результатом розробки, а саме прикладами інтерфейсу редагування та створення курсу можна ознайомитись на рисунках 4.28 та 4.29 відповідно.

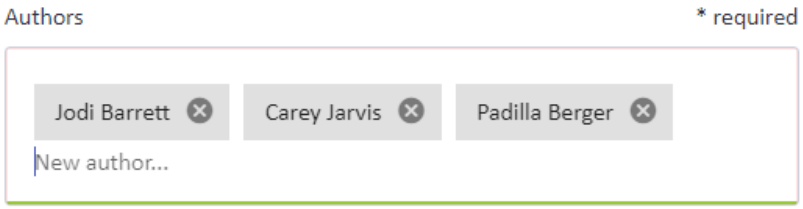


Рисунок 4.27 – Інтерфейсу вибору авторів

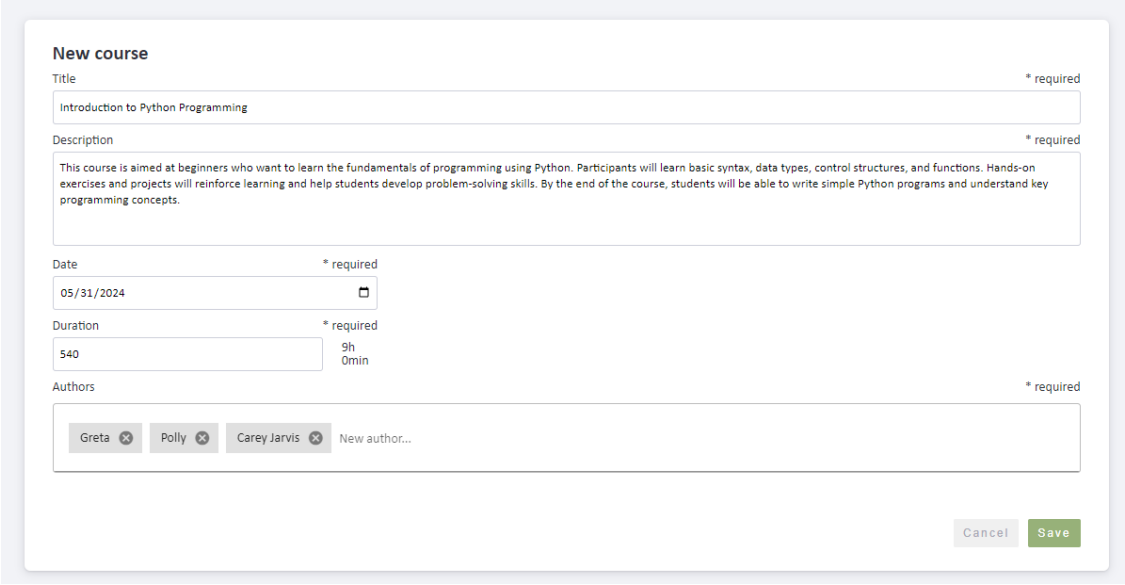


Рисунок 4.28 – Інтерфейсу редагування курсу

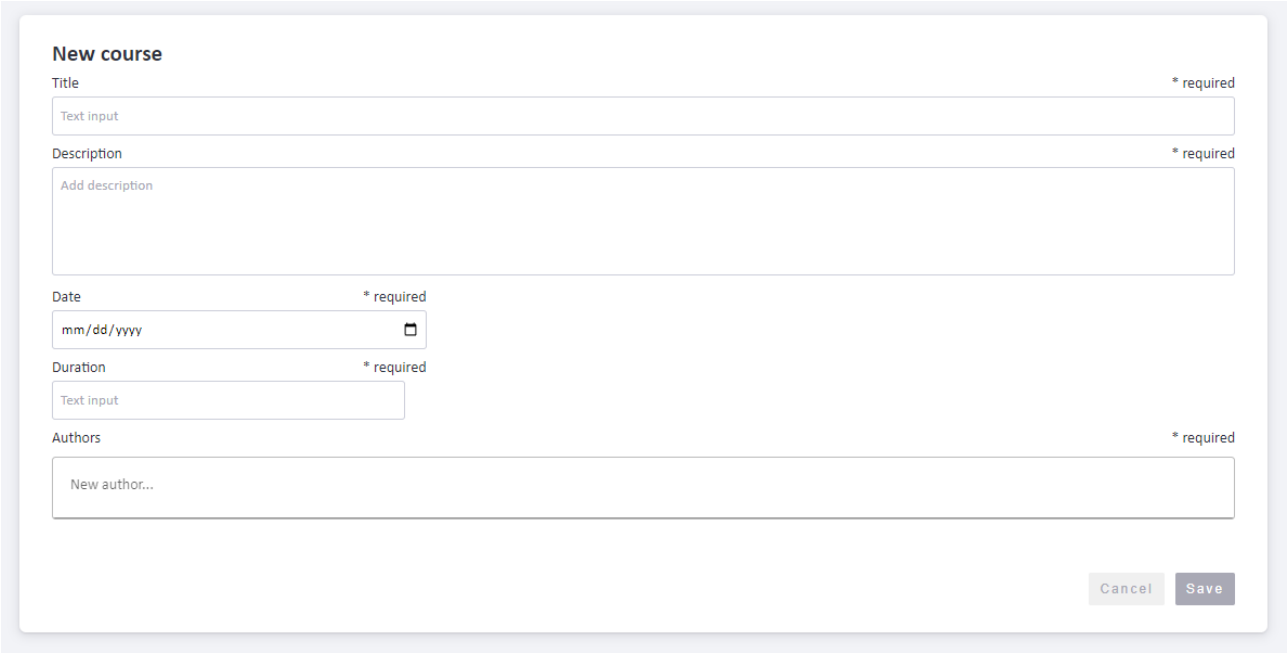


Рисунок 4.29 – Інтерфейсу створення курсу

4.3.4 Розробка сторінки авторизації користувача

Компонент “LoginComponent” відповідає за реалізацію форми авторизації користувача. Його HTML-шаблон складається з кількох ключових елементів: заголовка форми "Login", повідомлення про помилку, полів введення для електронної пошти та пароля, посилання для відновлення пароля та кнопки "Login" для надсилання форми, яка стає неактивною у разі некоректного заповнення форми. Повідомлення про помилку відображається, якщо введені дані неправильні.

Компонент div з класом content слугує контейнером для всього вмісту форми. Всередині цього контейнера знаходиться секція з класом login, яка містить саму форму для входу.

Форма має атрибут formGroup, який пов'язує її з формою Angular Reactive Forms, визначеною в TypeScript-коді. Подія ngSubmit на формі викликає метод onSubmit, коли користувач натискає кнопку відправки форми.

У заголовку форми з класом “.login__header” міститься заголовок з текстом "Login" та блок для відображення повідомлення про помилку. Якщо під час

входу виникає помилка, цей блок відображає повідомлення "Wrong e-mail or password". Використовується директива “*ngIf” для умовного відображення цього блоку лише тоді, коли є помилка. Далі йдуть два блоки “div” з класом “login__input-container”, кожен з яких містить мітку для відповідного поля вводу (електронна пошта та пароль) та індикатор обов'язкового поля .

Поля вводу для електронної пошти та пароля мають атрибут “formControlName”, який зв'язує їх з відповідними контролерами у формі. Обидва поля мають атрибути “required”, що означає, що ці поля є обов'язковими для заповнення.

В кінці форми розташований блок “div” з класом “.login__row”, який містить посилання "Forgot password?" та кнопку для відправки форми. Посилання має клас “login__forgotPswd” і служить для переходу на сторінку відновлення пароля. Кнопка з класом “login__btn btn” має атрибут “[disabled]="loginForm.invalid", що означає, що вона буде неактивною, якщо форма заповнена неправильно або не заповнена взагалі. З цілим HTML-кодом можна ознайомитись на рисунку 4.30.

```
<div class="content">
    <section class="login">
        <form  [formGroup]="loginForm" (ngSubmit)="onSubmit(loginForm)" action="">
            <div class="login__header">
                <h2 class="login__title">Login</h2>
                <div class="login__error"  *ngIf="error$ | async as error">Wrong e-mail or password</div>
            </div>
            <div class="login__input-container">
                <label for="login__input-text">Email:</label>
                <span class="required-indicator">* required</span>
            </div>
            <input formControlName="email" type="text" id="login__input-text" placeholder="Enter email" name="password" required>
            <div class="login__input-container">
                <label for="login__input-text">Password:</label>
                <span class="required-indicator">* required</span>
            </div>
            <input formControlName="password" type="password" id="login__input-pswd" placeholder="Enter password" required class="login__password" name="password"
                required />
            <div class="login__row">
                <a href="" class="login__forgotPswd">Forgot password?</a>
                <button type="submit" class="login__btn btn" [disabled]="loginForm.invalid">Login</button>
            </div>
        </form>
        You, 5 months ago • Finished project
    </section>
</div>
```

Рисунок 4.30 – HTML-код компоненту Login

TypeScript-код компонента реалізує логіку роботи форми. У конструкторі компонента здійснюється підключення необхідних сервісів: AuthService для автентифікації, Router для навігації та Store для управління станом додатка через

NgRx. За допомогою FormBuilder створюється форма з полями для електронної пошти та пароля, для яких встановлені відповідні валідатори. У методі ngOnInit формується сама форма. Метод onSubmit обробляє подію відправки форми: здійснюється команда loginStart з даними форми, після чого відбувається підписка на зміну статусу автентифікації. Якщо автентифікація пройшла успішно, користувача перенаправляють на сторінку курсів. Скріншот файлу з typescript кодом наведено на рисунку 4.31.

Основні стилі включають білий фон форми, закруглені кути, тінь для створення ефекту об'ємності та стилізацію кнопок і полів введення. Кнопка "Login" стає сірою, якщо вона неактивна. Також визначені стилі для повідомлення про помилку, яке відображається червоним кольором, та стилі для посилання "Forgot password?", який підкреслюється і має синій колір. Приклад не заповненої і заповненої форми зображено на рисунках 4.32 і 4.33 відповідно.

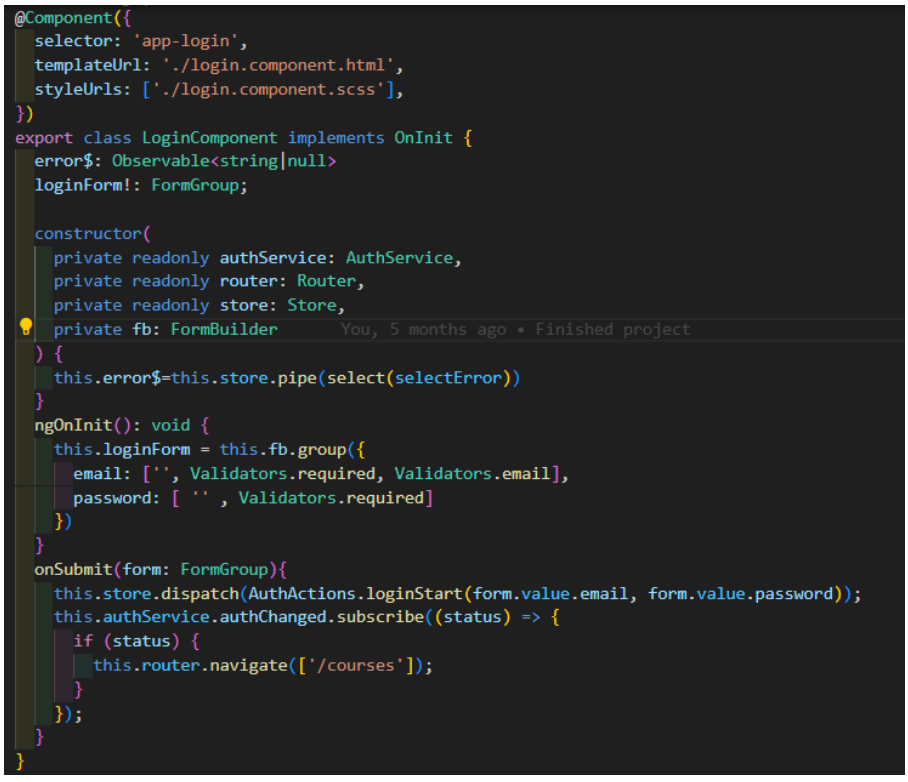


Рисунок 4.31 – Typescript-код компоненту Login

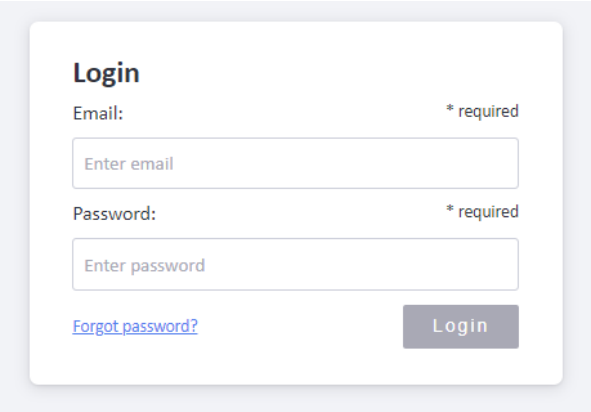


Рисунок 4.32 – Незаповнена форма Login

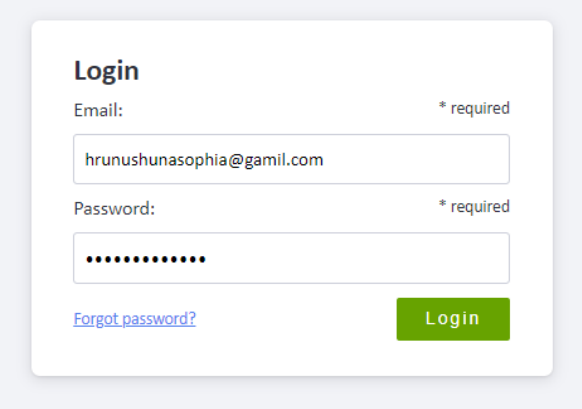


Рисунок 4.33 – Заповнена форма Login

4.3.5 Розробка функції пошуку курсів

Функція пошуку курсів використовується для введення користувачем текстового запиту та передачі цього запиту для пошуку в системі. Компонент складається з поля введення, де користувач може вводити текст для пошуку, та логіки для передачі цього тексту для обробки.

У шаблоні компонента маємо форму з полем введення, яке прив'язане до моделі данних з допомогою Angular Reactive Forms. З HTML-кодом можна ознайомитись на рисунку 4.34.

```
<form [formGroup]="searchForm"  class="searchBar">
    <input formControlName="searchName"  (keyup)="onSubmit()" name="searchName" type="text" placeholder="Text to search" />
</form>      You, 5 months ago • Finished project
```

Рисунок 4.34 – HTML-код компонента SearchBar

Коли користувач вводить текст у поле введення, відбувається виклик методу onSubmit(), який відправляє введений текст для обробки. При цьому також використовується реактивний підхід з debounceTime() для затримки відправки запиту певний час після останньої зміни.

У класі компонента ми створюємо об'єкт типу Subject, який використовується для відстеження змін в полі введення. Підписка на цей Subject відбувається у методі ngAfterViewInit(), де також встановлюється затримка за допомогою debounceTime(). При отриманні нового значення з поля введення, викликається обробник, який перевіряє довжину введеного тексту. Якщо текст має довжину більше або рівну 3 символам, він передається за допомогою події search для пошуку курсів. Якщо користувач видаляє текст з поля введення (довжина тексту дорівнює 0), також відбувається пошук і користувачеві знов виводяться всі курси.

При знищенні компонента у методі ngOnDestroy() відбувається відписка від підписки на Subject для уникнення витоку пам'яті. Код компонента зображено на рисунку 4.35.

```
ngAfterViewInit(): void {
  this.inputSubscription=this.searchNameChanged
  .pipe(debounceTime(500))
  .subscribe(searchName=>{
    if(searchName.length>=3 || searchName.length===0){
      this.search.emit(searchName)
    }
  })
}
onSubmit() {
  const searchName = this.searchForm.get('searchName')?.value;
  this.searchNameChanged.next(searchName);

}
ngOnDestroy(): void {
  this.inputSubscription.unsubscribe()
}
```

Рисунок 4.35 –Typescript-код компонента SearchBar

Таким чином користувач може здійснювати пошуковий запит за назвою або частиною опису курсу. На сторінку буде виведено список всіх матеріалів, опис або назва яких містять введене слово. На рисунку 4.36 зображено результат пошуку курсу за ключовим словом “Development”

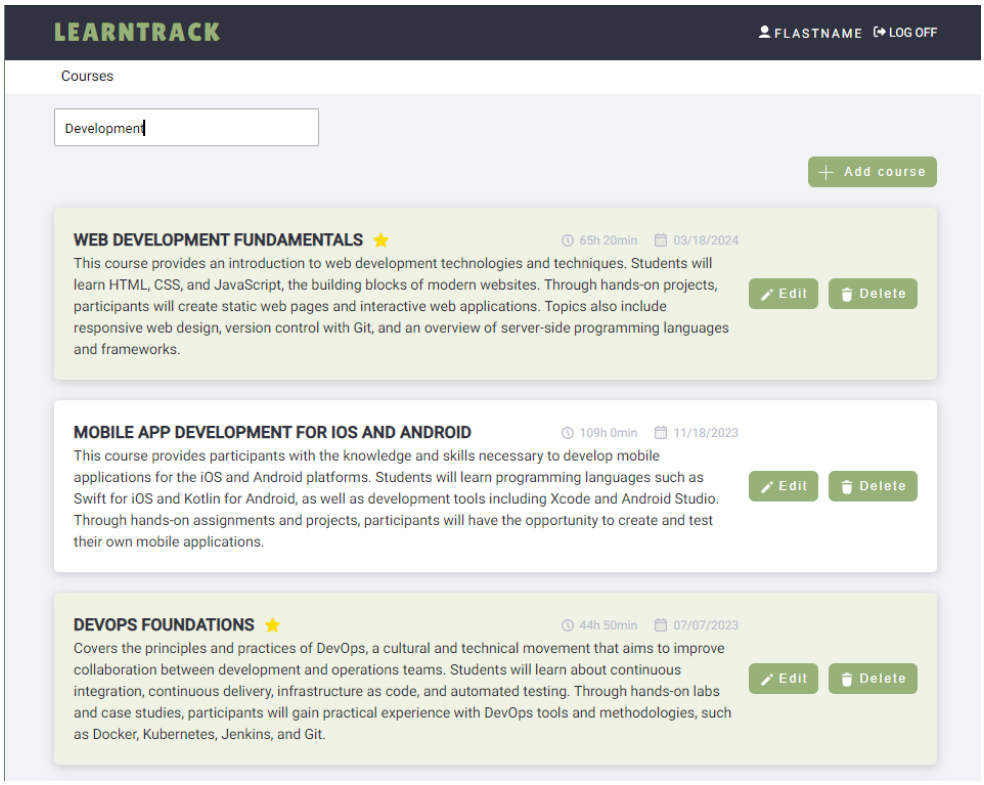


Рисунок 4.36 – Результат пошуку курсу за ключовим словом “Development”

У разі відсутності жодного курсу на екран буде виведено повідомлення “No data. Feel free to add new course”. З прикладом можна ознайомитись на рисунку 4.37.

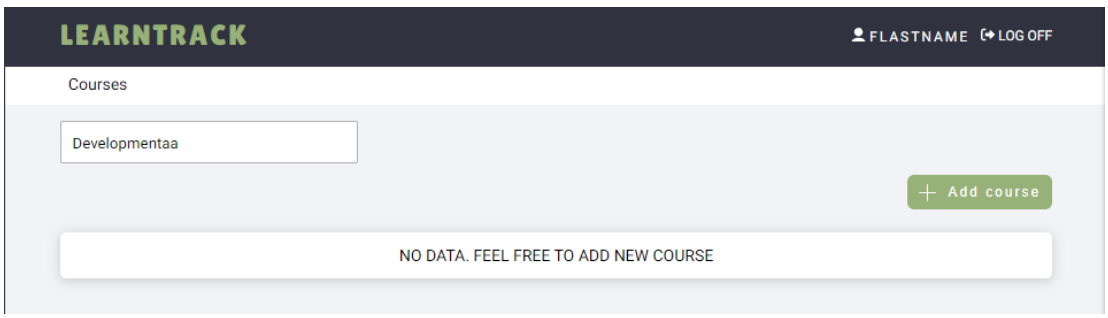


Рисунок 4.37 – Результат невдалого пошуку курсу

4.3.6 Розробка breadcrumbs

Компонент breadcrumbs призначений для відображення шляху навігації на веб-сайті, щоб користувачі могли легко переходити між різними сторінками. Він використовується для створення "хлібних крихт" - невеликих посилань, які показують користувачеві його поточне місцезнаходження та шлях до нього.

У шаблоні компонента маємо список <ul> з посиланнями <li>, які генеруються за допомогою директиви *ngFor для кожного об'єкта breadcrumb в масиві breadcrumbs. Кожне посилання містить ярлик (label) і шлях (path), де label відображається як текст, а path використовується для встановлення посилання. Детально з шаблоном можна ознайомитись на рисунке 4.38.

```
<ul class="breadcrumbs">
  <li *ngFor="let breadcrumb of breadcrumbs">
    <a [href]="breadcrumb.path">   {{breadcrumb.label}} </a>
  </li>
</ul>      You, 5 months ago • Finished project
```

Рисунок 4.38 – Результат невдалого пошуку курсу

У класі компонента визначені властивості і методи для оновлення крихтовин, підписки на зміни маршрутів і вибірки курсів. При ініціалізації компонента відбувається підписка на зміни маршрутів за допомогою Router events, які спрацьовують при завантаженні нової сторінки. Під час кожної зміни маршруту викликається метод updateBreadcrumbs(), який оновлює список крихт.

Метод updateBreadcrumbs(), з яким можна ознайомитись на рисунку 4.39, аналізує поточний шлях (URL) та генерує відповідні крихти на основі поточної сторінки та даних про курси. Наприклад, для сторінки "Courses/new" відображається "Courses / New", а для сторінки конкретного курсу відображається "Courses / Назва курсу". Всі крихти додаються до масиву breadcrumbs, який використовується для відображення у шаблоні. Приклади breadcrumbs при переході на курс з назвою “Introduction to Python Programming”

та на сторінку створення нового курсу зображено на рисунках 4.40 та 4.41 відповідно.

```
updateBreadcrumbs() {
  const url = this.router.routerState.snapshot.url;
  this.breadcrumbs.length = 0;
  const routerList = url.slice(1).split('/');
  if (routerList[0] === 'login') return;
  routerList.forEach((router, index) => {
    let label = '';
    let path = '';
    if (index === 0) {
      label = 'Courses';
      path = router;
    } else if (router === 'new') {
      label = '/ New';
    } else {
      const id = +router;
      const course: Course | undefined = this.courses.find(
        (course) => course.id === id
      );
      label = '/ ' + course?.name;
    }
    this.breadcrumbs.push({ label, path });
  });
  this.cdr.detectChanges();
}
```

Рисунок 4.39 – Метод updateBreadcrumbs

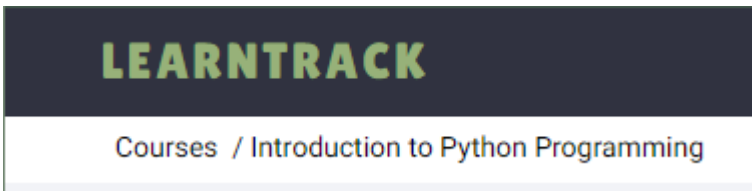


Рисунок 4.40 – Приклад breadcrumbs при переході на курс з назвою “Introduction to Python Programming”

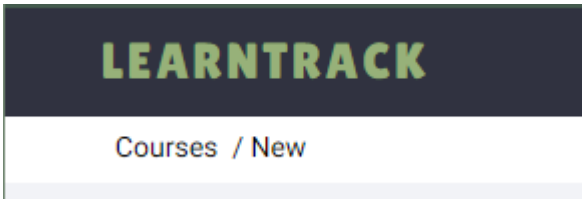


Рисунок 4.41 – Приклад breadcrumbs при переході на сторінку створення курсу

5 РОЗРОБКА BACK-END ЧАСТИНИ ПЛАТФОРМИ

5.1 Структура back-end частини

Back-end частина платформи організована таким чином, щоб забезпечити чітку ієрархію файлів та легке керування різними аспектами системи. Коренева директорія проекту містить кілька основних папок і файлів, які забезпечують функціонування сервера і його взаємодію з клієнтською частиною. У кореневій директорії знаходяться такі важливі елементи: каталог node_modules/, який містить всі встановлені залежності Node.js, необхідні для роботи проекту; каталог public/ для зберігання статичних файлів, таких як зображення та стилі, які можуть бути доступні безпосередньо з клієнтської сторони. У ньому розташовані підкаталоги images/ для зберігання зображень і stylesheets/ для зберігання файлів стилів, включаючи style.css, який містить стилі для клієнтської частини, а також favicon.ico – файл, який визначає іконку сайту. Основний HTML-файл index.html є стартовою точкою для клієнтської частини, а README.md містить загальний опис проекту, інструкції з встановлення та використання.

Директорія services/ містить піддиректорію core/, яка включає всі основні функціональні модулі сервера. Вона містить кілька підкаталогів: auth/ для аутентифікації користувачів, authors/ для управління авторами курсів, і courses/ для управління курсами. Кожен з цих підкаталогів має власні файли баз даних, проміжного програмного забезпечення та маршрутизації. Наприклад, підкаталог auth/ містить файли auth.db.json, auth.middleware.js, та auth.routes.js, що відповідають за зберігання даних аутентифікації, логіку проміжного програмного забезпечення та маршрутизацію відповідно. Аналогічно організовані підкаталоги authors/ і courses/. У загальній директорії core/ також є файли db.js для роботи з базою даних, index.js для ініціалізації сервера, middleware.js для загального проміжного програмного забезпечення, і routes.js для загальної маршрутизації.

Додатково, директорія utils/ містить утилітарні файли, такі як cors.js для налаштування CORS, walk.js для обробки файлової системи. У кореневій директорії також знаходяться файли .gitignore для визначення файлів та директорій, які повинні бути ігноровані системою контролю версій Git, index.js для запуску сервера, package-lock.json і package.json для управління залежностями проекту.

Таким чином, структура back-end частини платформи забезпечує модульність, організованість та легкість у керуванні різними аспектами системи, що сприяє більш ефективному розробленню та підтримці проекту. На рисунку 5.1 зображено скріншот зі структурою серверу.

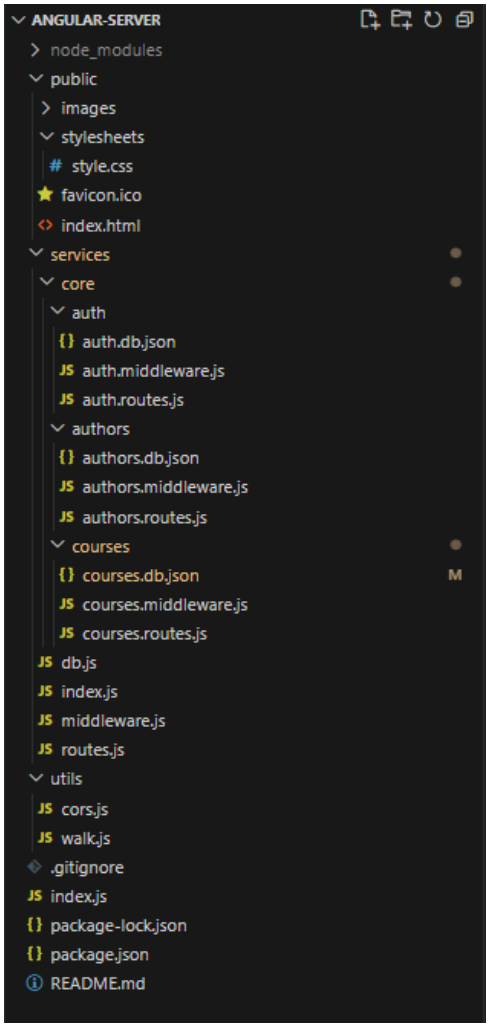


Рисунок 5.1 – Структура серверної частини

5.2 Розробка маршрутизатора запитів автентифікації користувача

Файл auth.middleware.js є модулем, який використовує Express для створення маршрутизатора, що обробляє запити, пов'язані з аутентифікацією користувачів. Він включає два основних маршрути: /auth/login для входу користувача і /auth/userinfo для отримання інформації про користувача. Функція, зображена на рисунку 5.2, приймає сервер як аргумент, що дозволяє отримати доступ до бази даних сервера. Спершу створюється маршрутизатор за допомогою express.Router(). Потім визначаються маршрути для обробки POST-запитів.

```
module.exports = (server) => {

    router.post('/auth/login', (req, res, next) => {
        let users = server.db.getState().users,
            matchedUser = users.find((user) => {
                return user.login.toUpperCase() === req.body.login.toUpperCase();
            });

        if(!matchedUser) {
            res.status(401).send('Wrong username');
        } else if(matchedUser.password === req.body.password) {
            res.json({ token: matchedUser.fakeToken});
        } else {
            res.status(401).send("Wrong password");
        }
    });

    router.post('/auth/userinfo', (req, res, next) => {
        let users = server.db.getState().users,
            matchedUser = users.find((user) => {
                return user.fakeToken === req.body.token;//('Authorization');
            });

        if(!matchedUser) {
            res.status(401).send('Unauthorized');
        } else {
            res.json(matchedUser);
        }
    });

    return router;
};
```

Рисунок 5.2 – Функція обробки запиту на авторизацію

Маршрут /auth/login обробляє запити на вхід користувача. При надходженні POST-запиту на цей маршрут з бази даних сервера отримуються всі користувачі. Шукається користувач, чиє ім'я користувача (логін) відповідає тому, що було надіслано в запиті. Порівняння імен користувачів проводиться без урахування регістру. Якщо користувача з таким логіном не знайдено, повертається статус 401 з повідомленням "Wrong username". Якщо логін знайдено, але пароль не збігається, повертається статус 401 з повідомленням "Wrong password". Якщо логін і пароль збігаються, користувачу повертається токен.

Маршрут /auth/userinfo обробляє запити на отримання інформації про користувача. При надходженні POST-запиту на цей маршрут з бази даних сервера отримуються всі користувачі. Шукається користувач, чий токен відповідає тому, що було надіслано в запиті. Якщо користувача з таким токеном не знайдено, повертається статус 401 з повідомленням "Unauthorized". Якщо токен збігається, повертається інформація про користувача.

Обидва маршрути використовують метод find для пошуку відповідного користувача в базі даних, а також перевіряють наявність помилок аутентифікації та повертають відповідні статуси і повідомлення. Цей файл створює і експортує маршрутизатор для обробки аутентифікаційних запитів. Він обробляє логін користувача, повертаючи токен у випадку успішної аутентифікації, і надає інформацію про користувача на основі токена, що дозволяє забезпечити базовий рівень безпеки і керування доступом у додатку.

5.3 Розробка маршрутизатора запитів, пов’язаних з авторами

Папка, яка відповідає за обробку запитів, пов'язаних з авторами, складається з кількох ключових файлів: authors.routes.js, authors.db.json та rewrite-example.js. Файл authors.routes.js реалізує маршрутизацію для обробки запитів до серверу. Використовуючи фреймворк Express, створюється маршрутизатор за допомогою express.Router(). Далі експортована функція, яка

приймає сервер як аргумент, налаштовує маршрути для отримання списку авторів. Зокрема, на маршрут GET /authors можна надсилати запити для отримання списку авторів з можливістю фільтрації за фрагментом тексту в їхніх іменах. Для цього обробляється URL запиту, витягуються параметри і, якщо присутній параметр “textFragment”, автори фільтруються за цим фрагментом тексту, причому пошук відбувається без урахування регістру. З функцією можна ознайомитись на рисунку 5.3.

Рисунок 5.3 – Функція обробки запиту “GET /authors”

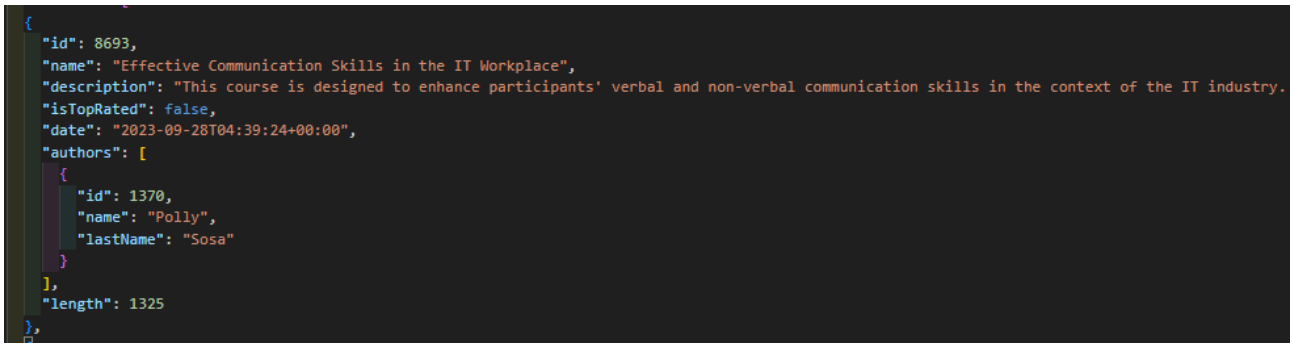
Файл authors.db.json містить статичні дані про авторів у форматі JSON, де кожен автор представлений об'єктом з полями id та name. Цей файл слугує джерелом даних для маршрутизатора, який обробляє запити на отримання авторів.

5.4 Розробка маршрутизатора запитів, пов’язаних з курсами

Так само як і попередні, папка “courses” складається з файлів з даними про курси у форматі JSON та маршрутизаторів для обробки HTTP-запитів до цих даних.

Файл courses.json містить дані про курси, включаючи ідентифікатори курсів, їхні назви, описи, рейтинг, дати створення, авторів та тривалість курсів. Дані про кожен курс структуровані у вигляді об'єктів з різними властивостями. Наприклад, курс "Effective Communication Skills in the IT Workplace" має

ідентифікатор 8693, описує навички ефективної комунікації в IT-індустрії, і його автором є Polly Sosa. На рисунку 5.4 зображено цей об’єкт цього курсу.



```
{
  "id": 8693,
  "name": "Effective Communication Skills in the IT Workplace",
  "description": "This course is designed to enhance participants' verbal and non-verbal communication skills in the context of the IT industry.
  "isTopRated": false,
  "date": "2023-09-28T04:39:24+00:00",
  "authors": [
    {
      "id": 1370,
      "name": "Polly",
      "lastName": "Sosa"
    }
  ],
  "length": 1325
},
```


Рисунок 5.4 – Об’єкт курсу "Effective Communication Skills in the IT Workplace"

Інші курси охоплюють різні теми, такі як програмування на Python, основи кібербезпеки, веб-розробку, машинне навчання, Інтернет речей (IoT), основи хмарних обчислень, принципи DevOps, штучний інтелект, етичний хакінг, цифровий маркетинг, розробку мобільних додатків і аналіз великих даних.

Файл “courses.routes.js” реалізує серверні маршрути аналогічно до файлів “auth.routes.js” та “authors.routes.js”. основний маршрут GET /courses дозволяє отримати список курсів з можливістю фільтрації за фрагментом тексту, сортування за різними критеріями (дата, рейтинг тощо) та пагінації (метод розбивки великого набору даних на менші частини). Якщо в запиті присутній параметр textFragment, курси фільтруються за наявністю цього фрагмента в їхніх назвах та описах. Якщо вказано параметр “sort”, курси сортуються за вказаним критерієм. Параметри “start” та “count” використовуються для реалізації пагінації. Якщо вказано параметр “id”, фільтрується конкретний курс за його ідентифікатором. Зі скріншотом функції можна ознайомитись на рисунку 5.5.

```
router.get('/courses', (req, res, next) => {
    let url_parts = url.parse(req.originalUrl, true);
    const query = url_parts.query;
    const from = parseInt(query.start, 10) || 0;
    let to = from + parseInt(query.count, 10);
    const sort = query.sort;
    const id = query.id;
    let courses = server.db.getState().courses;

    if (!!query.textFragment) {
        courses = courses.filter((course) => course.name.concat(course.description).toUpperCase().indexOf(query.textFragment.toUpperCase()) >= 0);
    }

    if(sort) {
        courses = [...courses].sort((a, b) => {
            if (sort === 'date') {
                const c = moment(b.date);
                const d = moment(a.date);
                return c.valueOf() - d.valueOf();
            } else {
                return b[sort] - a[sort];
            }
        });
    }

    if (courses.length < to || !to) {
        to = courses.length;
    }

    if(!id) {
        courses = courses.slice(from, to);
    } else {
        courses = courses.filter((item) => {
            return item.id == id;
        });
    }

    res.json(courses);
});
```

Рисунок 5.5 – Функція обробки запиту “GET /courses”

Також реалізовано маршрут “GET/error”, який приймає параметр “code” і повертає відповідь з вказаним кодом помилки. Приклад цієї функції зображено на рисунку 5.6

```
router.get('/error', function(req, res, next) {
    let url_parts = url.parse(req.originalUrl, true);
    let query = url_parts.query;
    res.status(parseInt(query.code, 10)).send({message: 'Error'});
});
```

Рисунок 5.6 – Функція обробки запиту “GET / error”

Файл “db.json” зчитує всі файли бази даних та зливає їх у єдиний об'єкт db за допомогою бібліотеки lodash. Потім він створює та повертає роутер jsonServer, який обробляє запити до даних. Це дозволяє централізовано керувати базою даних, зберігаючи всі дані в одному місці. Скріншот файлу наведено на рисунку 5.7.

```
const jsonServer = require('json-server');
const _ = require('lodash');

module.exports = (files) => {
    let db = {};

    for (let file of files) {
        if (file.match(/db\.json$/gi)) {
            let data = require(file);
            db = _.merge(db, data);
        }
    }

    return jsonServer.router(db);
```

Рисунок 5.7 – Файл db.json

Модуль проміжних шарів “middleware.js” зчитує всі файли, які містять проміжні шари (.middleware.js), та підключає їх до роутера Express. Кожен проміжний шар отримує доступ до об'єкту бази даних db, що дозволяє їм взаємодіяти з даними та виконувати необхідні операції. З кодом файлу можна ознайомитись на рисунку 5.8.

```
const express = require('express');
const router = express.Router();

module.exports = (files, db) => {

    for (let file of files) {
        if (file.match(/\.middleware\.js$/gi)) {
            let middleware = require(file)(db);
            router.use(middleware);
        }
    }

    return router;
```

Рисунок 5.8 – Файл middleware.js

Модуль маршрутів “routes.js” зчитує всі файли, що містять маршрути (.routes.js), та підключає їх до роутера Express. Це дозволяє легко додавати нові маршрути, просто створюючи нові файли у відповідній директорії. Код файлу зображено на рисунку 5.9.

```
const express = require('express');
const router = express.Router();

module.exports = (files) => {

    for (let file of files) {
        if (file.match(/\.routes\.js$/gi)) {
            let route = require(file);
            router.use(route);
        }
    }

    return router;
```

Рисунок 5.9 – Файл routes.js

Далі, головний модуль “index.js” експортує три основні компоненти сервера: базу даних (db), маршрути (routes) та проміжні шари (middleware). Цей модуль імпортує відповідні функції та передає їм масив файлів для налаштування. Таким чином, він забезпечує з'єднання всіх необхідних компонентів для роботи сервера. Скріншот файлу наведено на рисунку 5.10.

```
1  module.exports = (files) => {
2      const db = require('./db')(files);
3      const routes = require('./routes')(files);
4      const middleware = require('./middleware')(files, db);
5      return {middleware, db, routes};
```

Рисунок 5.10 – Файл index.js

Завдяки описаним вище модулям та методам з'єднання файлів, була створена ефективна та масштабована інфраструктура back-end сервера,яка дозволяє зручно керувати базою даних, маршрутами та проміжними шарами, що робить процес розробки більш гнучким і ефективним.

Таким чином, завдяки інтеграції front-end та back-end, створено цілісну систему, яка може надавати користувачам зручний інтерфейс для взаємодії з платформою курсів.

ВИСНОВКИ

В результаті виконаної кваліфікаційної роботи була створена платформа для онлайн-курсів, яка відповідає сучасним вимогам щодо функціональності, безпеки та масштабованості. Отримані результати демонструють, що розроблена система є надійною основою для подальшого розвитку та вдосконалення в сфері онлайн-освіти. Платформа сприяє підвищенню якості навчання та відкриває нові можливості для організації освітнього процесу в цифровому середовищі.

У цій роботі було розглянуто процес розробки та впровадження платформи для онлайн-курсів. В результаті виконаної роботи було досягнуто наступних ключових результатів:

– проведено детальний аналіз вимог до платформи для курсів, що дозволило сформувати чітке бачення необхідних функціональних можливостей та технічних характеристик системи. Визначено основні компоненти платформи, включаючи базу даних, маршрутизацію, інтерфейси користувача та проміжні шари;

– створено інтерфейс користувача для платформи курсів з використанням сучасних технологій веб-розробки. Використано фреймворки та бібліотеки для створення інтерактивного та зручного інтерфейсу, який дозволяє користувачам легко взаємодіяти з платформою. Забезпечено адаптивний дизайн для коректного відображення на різних пристроях;

– описано процес створення ефективної та масштабованої інфраструктури back-end сервера. Використано модулі для управління базою даних, маршрутизації та проміжними шарами, що забезпечило гнучкість і ефективність розробки. Завдяки цьому вдалося забезпечити зручне керування даними, обробку запитів та інтеграцію з іншими компонентами системи;

– впроваджено заходи для забезпечення безпеки даних та аутентифікації користувачів. Також було створено умови для подальшої масштабованості системи, що дозволяє легко додавати нові функціональні можливості та компоненти без значних змін у базовій архітектурі;

– розроблено та впроваджено основні функції платформи, такі як управління курсами, реєстрація користувачів, відстеження прогресу та взаємодія між студентами та викладачами. Ці функції забезпечують зручність використання системи та підвищують ефективність навчального процесу;

Ця робота може стати основою для подальших досліджень та розробок в галузі інформаційних технологій та онлайн-освіти, сприяючи впровадженню інноваційних підходів до навчання та розвитку цифрових платформ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Історія та перспектива розвитку дистанційного навчання [Електронний ресурс] – URL: https://dspace.nuft.edu.ua (Дата звернення 02.05.2023).

2. Про онлайн навчання на платформі Coursera. [Електронний ресурс] – URL: https://ode.dcz.gov.ua/publikaciya/pro-onlayn-navchannya-na-platformi-coursera (Дата звернення 16.05.2024).

3. Angular. [Електронний ресурс] – URL: https://angular.dev/overview (Дата звернення 26.05.2024).

4. Основні концепції Angular: компоненти. [Електронний ресурс] – URL:https://blog-it.com.ua/vvedennya-do-frejmvorku-angular/ (Дата звернення 26.05.2024).

5. Основні концепції Angular: сервіси. [Електронний ресурс] – URL:https://blog-it.com.ua/vvedennya-do-frejmvorku-angular/ (Дата звернення 26.05.2024).

6. Набір функціональності для розробки веб додатків. [Електронний ресурс] – URL: https://blog-it.com.ua/vvedennya-do-frejmvorku-angular/ (Дата звернення 26.05.2024).

7. Фреймворк Jasmine [Електронний ресурс] – URL:https://github.com/jasmine/jasmine (Дата звернення 27.05.2024).

8. Препроцесор SCSS [Електронний ресурс] – URL: https://mate.academy/blog/front-end-and-js/css-preprocessors/ (Дата звернення 27.05.2024).