НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
В.о. завідувача кафедри
_____ Михайло НОВОТАРСЬКИЙ
(підпис)

“__” _____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

на тему: Веб-платформа для навчання операторів БПЛА та збірки FPV дронів

Виконав : студент __4__ курсу, групи _ІО-16
(шифр групи)

_____Павлуш Анастасія Юріївна_____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____ас. Череватенко О. В._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) ст. вик., док. філ. комп. інж. Міщенко Л. Д.__
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ас. каф. ІСТ д.філ. Нікітін В. А._____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2025 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“ Комп'ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Михайло НОВОТАРСЬКИЙ
(підпис)

“__” _____2025 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Павлуш Анастасії Юріївни

1. Тема проєкту *Веб-платформа для навчання операторів БПЛА та збірки FPV дронів*
керівник проєкту Череватенко Олексій Володимирович, асистент,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 31 грудня 2024 року №НОД/996/24

2. Термін здачі студентом закінченого проєкту *2 червня 2025 р.*

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Аналіз сучасного стану та особливостей предметної області

Розділ 2. Огляд та порівняння технологій розробки.

Розділ 3. Реалізація веб-застосунку.

Розділ 4. Аналіз розробленого веб-застосунку.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна схема системи, функціональна схема (діаграма класів), алгоритм дій програмного забезпечення.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Міщенко Л. Д.	*10.02.2025*	*02.06.2025*

7. Дата видачі завдання «02» лютого 2025 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.12.2024-15.12.2024*	
2.	*Вивчення та аналіз завдання*	*15.12.2024-15.03.2025*	
3.	*Розробка архітектури та загальної структури системи*	*15.03.2025-25.03.2025*	
4.	*Розробка структур окремих підсистем*	*25.03.2025-5.04.2025*	
5.	*Програмна реалізація системи*	*5.04.2025-15.04.2025*	
6.	*Оформлення пояснювальної записки*	*15.04.2025-20.05.2025*	
7.	*Захист програмного продукту*	*25.04.2025*	
8.	*Передзахист*	*30.05.2025*	
9.	*Захист*	*16.06.2025*	

Студент-дипломник _____ Павлуш Анастасія
(підпис)

Керівник проєкту _____ Олексій Череватенко
(підпис)

АНОТАЦІЯ

У даному проєкті було розроблено та реалізовано користувацький інтерфейс веб-платформи для навчання операторів БПЛА та збірки FPV дронів. Ця платформа забезпечує зручний доступ до навчальних матеріалів, інструкцій та інтерактивних модулів, що сприяють ефективному засвоєнню як теоретичних знань, так і практичних навичок. Для реалізації фронтенду використано сучасні технології веброзробки — HTML, CSS та JavaScript. Особливу увагу приділено створенню інтуїтивно зрозумілого, доступного та функціонального інтерфейсу, зручного для користувачів із різним рівнем підготовки. Зі сторони бекенду платформа реалізована з використанням мови програмування Python. Для зберігання та обробки даних застосовується система управління базами даних PostgreSQL.

Ключові слова: навчальна веб-платформа, БПЛА, FPV, фронтенд, бекенд, база даних, HTML, CSS, JavaScript, Python, SQL.

ANNOTATION

In this project, the user interface of a web platform for training UAV operators and assembling FPV drones was developed and implemented. This platform provides convenient access to training materials, instructions, and interactive modules that contribute to the effective acquisition of both theoretical knowledge and practical skills. Modern web development technologies were used to implement the frontend — HTML, CSS, and JavaScript. Particular attention was paid to creating an intuitive, accessible, and functional interface that is convenient for users with different levels of training. On the backend side, the platform was implemented using the Python programming language. The PostgreSQL database management system is used to store and process data.

Keywords: educational web platform, UAV, FPV, frontend, backend, database, HTML, CSS, JavaScript, Python, SQL.

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпля	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.045440.002 ТЗ***	Веб-платформа для	3		
			навчання операторів БПЛА			
			та збірки FPV дронів			
			Технічне завдання			
	А4	***ІАЛЦ.045440.003 ПЗ***	Веб-платформа для	61		
			навчання операторів БПЛА			
			та збірки FPV дронів			
			Пояснювальна записка			
	А4	***ІАЛЦ.045440.004 Д1***	Веб-платформа для	1		
			навчання операторів БПЛА			
			та збірки FPV дронів			
			Структурна схема			
			системи			
	А4	***ІАЛЦ.045440.005 Д2***	Веб-платформа для	1		
			навчання операторів БПЛА			
			та збірки FPV дронів			
			Функціональна схема			
	А4	***ІАЛЦ.045440.006 Д3***	Веб-платформа для	1		
			навчання операторів БПЛА			
			та збірки FPV дронів			
			Схема принципова			
			(Схема взаємодії			
			користувача)			
	А4	***ІАЛЦ.045440.007 Д4***	Веб-платформа для	61		
			навчання операторів БПЛА			
			та збірки FPV дронів			
			Текст програмного коду			

					ІАЛЦ.045440.001 ОА	
Зм	*Лист*	*№ докум.*	*Підп*	*Дата*		
Розроб		Павлуш А.Ю.			*Веб-платформа для навчання операторів БПЛА та збірки FPV дронів* Опис альбому	Літ. / Аркуш 1 / Аркушів 1
Перев		Череватенко О.В.				***НТУУ "КПІ" ім. Ігоря Сікорського, ФІОТ ІО-16***

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Веб-платформа для навчання операторів БПЛА та збірки FPV дронів»*

Київ – 2025

ЗМІСТ

					ІАЛЦ.045440.002 ТЗ		
		№ докум.	Підпис	Дата			
Розробив		Павлуш А.Ю			**Веб-платформа для навчання операторів БПЛА та збірки FPV дронів** Технічне завдання	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Череватенко О.В					
						НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-16	
Н. Контр.		Міщенко Л. Д.					
Затвердив							

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку веб-платформи для навчання операторів безпілотних літальних апаратів (БПЛА) та збірки FPV дронів, а також на подальшу підтримку, розвиток і вдосконалення функціоналу платформи.

Область застосування цієї платформи охоплює дистанційне навчання та самопідготовку операторів БПЛА, технічне навчання з конструювання FPV дронів, використання у військовій сфері, де потрібне швидке та доступне оволодіння навичками керування безпілотними системами.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної веб-платформи є актуальні потреби, зумовлені збройною агресією проти України та широким застосуванням безпілотних літальних апаратів у бойових діях. У зв’язку з цим виникла гостра необхідність у підготовці операторів БПЛА та освоєнні технологій FPV-дронів для оперативного застосування в умовах сучасної війни.

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Платформа має на меті забезпечити доступне, структуроване та ефективне навчання для військовослужбовців, волонтерів та всіх, хто залучений до оборонного сектору. Реалізація проєкту також відповідає стратегічному завданню цифрової трансформації військової підготовки та розвитку інженерно-технічних компетенцій у критичних умовах.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки проєкту стали технічна документація, наукова література та матеріали з Інтернету на відповідну тематику.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена веб-платформа має відповідати таким вимогам:

- Простий, інтуїтивно зрозумілий та адаптивний інтерфейс.
- Наявність структурованого доступу до навчальних матеріалів.
- Підтримка інтерактивних модулів, що сприятимуть закріпленню практичних навичок.
- Можливість адміністрування та оновлення контенту платформи.
- Забезпечення багатокористувацького режиму з авторизацією та особистими кабінетами.

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux.
- Visual Studio Code IDE версії або вище.

5.3. Вимоги до апаратної частини

Веб-застосунок повинен функціонувати на сервері з наступними характеристиками:

- Процесор — щонайменше 4 ядра.
- Оперативна пам’ять — не менше 16 ГБ.
- Місце на жорсткому диску — не менше 100 ГБ вільного простору.
- Підключення до мережі Інтернет зі швидкістю не менше 10 Мбіт/с.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	10.12.2024-15.12.2024
Вивчення та аналіз завдання	15.12.2024-15.03.2025
Розробка архітектури та загальної структури системи	15.03.2025-25.03.2025
Розробка структур окремих частин системи	25.03.2025-5.04.2025
Програмна реалізація системи	5.04.2025-15.04.2025
Виправлення помилок	15.04.2025-20.05.2025
Оформлення пояснювальної записки	25.04.2025

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Веб-платформа для навчання операторів БПЛА та збірки FPV дронів»*

Київ – 2025

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БПЛА	(Безпілотний Літальний Апарат) Літальний апарат без екіпажу на борту
API	(Application Programming Interface) Інтерфейс прикладного програмування
CI/CD	(Continuous Integration / Continuous Deployment) Безперервна інтеграція та безперервне розгортання
CRUD	(Create, Read, Update, Delete) Створення, зчитування, оновлення та видалення даних
CSS	(Cascading Style Sheets) Каскадні таблиці стилів
DOM	(Document Object Model) Модель об’єктів документа
FPV	(First Person View) Вид від першої особи
HTML	(HyperText Markup Language) Мова розмітки гіпертексту
HTTP	(HyperText Transfer Protocol) Протокол передавання гіпертексту
JSON	(JavaScript Object Notation) Нотація об'єктів JavaScript
JS	(JavaScript) Мова програмування
JWT	(JSON Web Token) Веб-токен на основі JSON.
ORM	(Object-Relational Mapping) Об'єктно-реляційне відображення
REST	(Representational State Transfer) Архітектурний стиль для створення веб сервісів
SQL	(Structured Query Language) Мова структурованих запитів
UAV	(Unmanned Aerial Vehicle) Безпілотний літальний апарат
UI	(User Interface) Інтерфейс користувача
UUID	(Universally Unique Identifier) Універсальний унікальний ідентифікатор

ВСТУП

У контексті сучасних викликів, пов’язаних із військовими діями в Україні, зростає потреба у швидкій адаптації технологій до потреб безпеки, оборони та підготовки спеціалістів. Одним із важливих напрямів є розвиток безпілотних літальних апаратів (БПЛА), зокрема FPV-дронів (First Person View), які стали важливими інструментами ведення розвідки, коригування вогню та виконання спеціальних завдань. Ефективне використання таких технологій напряму залежить від рівня підготовки операторів та їхньої здатності збирати, обслуговувати і застосовувати FPV-дрони у реальних умовах.

На цьому тлі розробка сучасної веб-платформи для навчання операторів БПЛА стає не лише актуальною, а й критично важливою. Такий веб-застосунок має забезпечити централізований, доступний і структурований освітній процес, що охоплює як теоретичну, так і практичну складову. Платформа дозволяє користувачам ознайомлюватися з навчальними матеріалами, відеоінструкціями, покроковими гайдами зі збірки дронів, а також виконувати інтерактивні завдання для закріплення знань.

Під час реалізації проєкту застосовано сучасний і ефективний технологічний стек. Фронтенд реалізовано з використанням HTML, CSS, JavaScript та фреймворку React, що забезпечує високу швидкодію, адаптивність до різних пристроїв та зручність для користувачів. Бекенд частину побудовано на Python із застосуванням фреймворку FastAPI, що дозволяє створювати продуктивні, надійні та масштабовані веб-сервіси. Для збереження і обробки даних обрана реляційна база даних PostgreSQL, яка гарантує стабільність, цілісність інформації та високу продуктивність у роботі з великими обсягами даних.

Під час проєктування було враховано специфіку роботи майбутніх користувачів — зокрема, те, що більшість із них можуть мати різний рівень технічної підготовки. Це зумовило створення інтуїтивно зрозумілого інтерфейсу з чіткою логікою взаємодії та структурованою подачею навчального контенту. Особливу увагу приділено модульності системи — це дозволяє масштабувати платформу, додавати нові навчальні блоки, інтегрувати додаткові функціональні модулі тощо.

Наукова новизна даної роботи полягає в розробці універсального веб-застосунку для дистанційного навчання у військово-технічній галузі, який поєднує доступність, гнучкість і практичну спрямованість. Практична значущість полягає у можливості впровадження цієї платформи у навчальні процеси військових частин, волонтерських проєктів, освітніх закладів, а також у її потенціалі для підготовки цивільних операторів дронів.

Таким чином, створений веб-застосунок може стати дієвим інструментом для масової підготовки операторів FPV-дронів, сприяти розвитку технологічної спроможності країни в умовах війни та стати частиною стратегічної ініціативи з підтримки обороноздатності України. Подальший розвиток системи передбачає її розширення, інтеграцію з тренажерами та системами тестування, а також адаптацію під інші напрямки підготовки фахівців у галузі новітніх технологій.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНОГО СТАНУ ТА ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність розробки веб-застосунку у сфері навчання операторів БПЛА

Сьогодні Україна переживає період великих трансформацій, спричинених війною, мобілізацією суспільства та переосмисленням ролі сучасних технологій у питаннях безпеки, оборони, логістики та розвитку економіки. Однією з ключових галузей, яка набула особливого значення, стала сфера безпілотних літальних апаратів (БПЛА) [1], зокрема FPV-дронів [2]. Ці дрони відіграють усе важливішу роль як на фронті — у розвідці, атаках, коригуванні вогню — так і в мирному житті — у сфері моніторингу, картографії, сільського господарства, пошуково-рятувальних операцій тощо.

З цим стрімким розвитком пов'язана гостра потреба у якісному, масовому навчанні операторів БПЛА [3]. Сьогодні вже недостатньо мати лише базові навички пілотування — оператор повинен розуміти принципи зборки дронів, налаштування FPV-систем, техніку безпеки, роботу з телеметрією та програмне забезпечення, зокрема для автономного польоту. Але, незважаючи на попит, доступ до систематизованих навчальних матеріалів залишається серйозною проблемою.

В інтернеті справді є велика кількість ресурсів, присвячених темі FPV-дронів, однак вони часто:

- розкидані по різних платформах, форумах, YouTube-каналах та Telegram-групах;
- дублюють або суперечать одне одному;
- не мають чіткої структури або подання інформації для новачків;

- доступні переважно англійською мовою, що ускладнює розуміння для багатьох користувачів;
- швидко застарівають через стрімку зміну технічних стандартів.

Саме тому важливо створити єдину веб-платформу, яка стане точкою доступу до перевіреної, актуальної та адаптованої українською мовою інформації. Такий веб-застосунок дозволить новачкам без технічного бекграунду швидко зануритися у тему, а досвідченим операторам — поглибити знання чи знайти технічну інформацію в одному місці без необхідності витрачати години на пошук по різних джерелах.

Додатково, більшість існуючих курсів із підготовки операторів БПЛА проводяться у форматі очного навчання, що створює кілька бар'єрів:

- географічний — курси зазвичай проводяться у великих містах, що унеможливлює участь жителів малих населених пунктів;
- фінансовий — вартість офлайн-програм коливається від 5 000 до 25 000 гривень, що не завжди є прийнятним варіантом, особливо для ветеранів, ВПО або людей, які шукають перекваліфікацію;
- часовий — офлайн-навчання часто проходить у фіксований час, що не підходить тим, хто працює, служить чи доглядає за родиною.

Веб-застосунок, у свою чергу, дає змогу навчатись у власному темпі, у зручний час, з будь-якого пристрою. Крім того, платформа буде мати модульну структуру навчання — це дозволить користувачеві крок за кроком проходити навчання, не губитися в інформації й завжди мати доступ до потрібного блоку: від базових знань до схем підключення, налаштування польотного контролера чи огляду моделей.

Суттєвою перевагою платформи є її гнучкість та здатність до масштабування. Завдяки продуманій архітектурі, адміністратор зможе легко оновлювати навчальні матеріали, додавати нові модулі або курси, змінювати структуру контенту відповідно до появи нових технологій чи потреб користувачів. Навчання супроводжуватиметься відеоуроками, тестуванням,

тематичними блоками у форматі інструкцій, що сприятиме кращому засвоєнню матеріалу. Це забезпечить не лише зручний доступ до знань, а й створення динамічного освітнього простору навколо теми FPV-дронів — відкритого, підтримуючого та орієнтованого на практичний результат.

Створення веб-застосунку для навчання операторів FPV-дронів — це не просто освітній проєкт. Це інфраструктурна ініціатива, яка поєднує технологічну доступність, соціальну відповідальність та практичну користь у час, коли суспільство особливо потребує нових навичок, згуртованості та фокусу на спільне майбутнє.

1.2 Огляд і порівняльний аналіз наявних веб-рішень у сфері FPV-дронів та дистанційного навчання

Сфера підготовки операторів безпілотних літальних апаратів (БПЛА) стрімко розвивається, що зумовило появу низки навчальних платформ і онлайн-курсів. Водночас, більшість із них або орієнтовані на зарубіжну аудиторію, або є надто фрагментованими та спеціалізованими. Станом на 2024 рік найпопулярніші рішення в цій галузі — це платформи, як Drone Pilot Ground School, UAV Coach, Drone Dojo, Pilot Institute (США), а також кілька ініціатив в Україні, пов’язаних із громадськими організаціями та волонтерськими проєктами.

Drone Pilot Ground School та Pilot Institute пропонують сертифікацію відповідно до стандартів FAA (Federal Aviation Administration), а також доступ до відеоуроків, тестів і симуляторів. Однак основна проблема полягає в тому, що ці платформи орієнтовані виключно на англомовну аудиторію, вартість курсів часто перевищує $100, а подані приклади не завжди мають практичне застосування в умовах бойових дій чи волонтерських ініціатив, які актуальні для України.

В українському сегменті існують спроби створення навчального контенту: наприклад, YouTube-канали типу «Дронарня», окремі телеграм-канали з порадами щодо FPV-дронів, а також навчальні ініціативи на базі ЗСУ або добровольчих організацій. Проте ці ресурси розрізнені, не мають єдиного навчального плану та не завжди зручно структуровані для новачків. У деяких випадках інформація подається хаотично, без належної систематизації або оновлення, а також без можливості зворотного зв’язку чи тестування знань.

Крім того, у відкритому доступі переважна більшість якісних інструкцій зі збирання FPV-дронів, налаштування автопілотів, оновлення прошивок чи вибору комплектуючих — публікується лише англійською мовою. Це створює додатковий бар’єр для багатьох охочих опанувати цю сферу, особливо для тих, хто не володіє іноземними мовами на достатньому рівні.

Таким чином, попри наявність окремих спроб надати доступ до знань у цій галузі, наразі відсутнє комплексне рішення українською мовою, що:

- об’єднує теоретичні й практичні блоки;
- забезпечує адаптацію контенту під різний рівень користувача;
- має гнучку адміністративну систему для регулярного оновлення навчальних матеріалів;
- є безкоштовним або доступним для широкої аудиторії.

Саме ця потреба й мотивувала створення цілісної веб-платформи, яка систематизує навчальні матеріали, інструкції та практичні поради з експлуатації й зборки FPV-дронів, подаючи їх у зручній, зрозумілій та актуальній формі.

1.3 Ключові переваги та обмеження сучасних веб-застосунків освітнього призначення

Сучасні веб-застосунки у сфері освіти вже давно зарекомендували себе як ефективний інструмент передачі знань. Їх популярність зросла особливо після пандемії COVID-19, яка суттєво активізувала розвиток дистанційного навчання. У контексті підготовки операторів БПЛА, освітні веб-платформи відкривають нові можливості для доступного, структурованого та гнучкого процесу навчання. Проте, як і будь-який інструмент, такі платформи мають свої сильні сторони та певні обмеження.

Переваги:

- Доступність з будь-якого місця та пристрою Онлайн-застосунки дозволяють навчатися з дому, військових частин або польових умов, що особливо актуально для тих, хто має обмежений доступ до очного навчання чи проживає у віддалених регіонах.
- Гнучкість навчального процесу. Користувач може проходити курс у зручному темпі, повертатися до складних тем та повторювати матеріал.
- Централізованість знань. Платформа може зібрати в одному місці всю ключову інформацію: відеоінструкції, текстові пояснення, схеми, списки необхідного обладнання, часті помилки — усе це робить навчання системним і зручним.
- Регулярне оновлення контенту. Адміністратор або викладач має змогу оперативно додавати нові курси, оновлювати інструкції відповідно до змін у технологіях (наприклад, нові версії прошивок, контролерів або FPV-гарнітур).
- Економічність. Онлайн-формат дозволяє уникнути витрат на оренду приміщень, друк матеріалів чи транспорт, що робить освітній продукт більш доступним для широкого кола користувачів.

Обмеження:

- Відсутність практичної частини "наживо". Жодна платформа не замінить фізичне керування дроном. Без практики в реальних умовах користувач не зможе повноцінно засвоїти навички керування, хоча платформа може дати міцну теоретичну базу.
- Відсутність інтерактивних симуляторів у деяких реалізаціях. Розробка якісного симулятора потребує великих ресурсів. У межах цього проєкту симулятори не передбачаються, проте платформа може посилатись на сторонні рекомендовані варіанти.
- Проблеми з мотивацією та самодисципліною. Самостійне онлайн-навчання вимагає високого рівня самоорганізації, що може бути складно для деяких користувачів без зовнішнього контролю.
- Залежність від інтернету. У деяких регіонах України, особливо у прифронтових зонах, доступ до стабільного інтернету може бути ускладнений. Проте платформа може бути частково адаптована для офлайн-режимів (наприклад, завантаження PDF-інструкцій).

Освітні веб-застосунки — це сучасний і ефективний інструмент передачі знань, особливо у тих сферах, де є великий попит на оперативне й доступне навчання. За правильного підходу до розробки, такі системи можуть стати ключовим елементом підготовки нових операторів БПЛА в умовах сучасної війни, технологічного розвитку й потреби у швидкій перекваліфікації фахівців.

1.4 Формулювання завдання та характеристика предметної області

Завданням даного проєкту є розробка веб-застосунку, призначеного для теоретичної підготовки операторів безпілотних літальних апаратів (БПЛА), зокрема дронів типу FPV. Основна мета проєкту полягає у створенні сучасного, зручного та доступного засобу дистанційного навчання, який дозволить широкому колу користувачів отримати систематизовані знання щодо основ будови, експлуатації та обслуговування дронів без необхідності відвідування очних курсів [4].

Предметна область проєкту охоплює сферу дистанційної технічної освіти з фокусом на розвиток навичок у сфері управління та використання БПЛА. Це включає розуміння конструктивних особливостей дронів, принципів їх функціонування, основ програмування контролерів, правил безпеки, а також базових підходів до польотів у FPV-форматі. Особливу актуальність проєкту зумовлює зростаюча потреба у підготовці операторів дронів як для цивільного, так і для військового використання в умовах сучасних реалій України.

Значна частина існуючих освітніх ініціатив у цій галузі передбачає очну форму навчання, що часто є недоступною через географічні, фінансові чи часові обмеження. Водночас інформація в мережі Інтернет представлена фрагментарно, значною мірою англомовна, часто без належної структури, перевірки чи локалізації. Це створює додаткові бар’єри для користувачів-початківців, які прагнуть здобути якісні та практичні знання українською мовою.

Розробка веб-застосунку сприятиме вирішенню проблеми інформаційної розпорошеності та нерівного доступу до знань. Вона дозволить сконцентрувати актуальні навчальні матеріали в єдиному середовищі, що систематично оновлюється та підтримується. Також платформа відкриє можливості для

самостійної перекваліфікації, підвищення технічної обізнаності або підготовки до практичних курсів.

Предметна область даного проєкту лежить на перетині інформаційних технологій, технічної освіти та сучасних підходів до професійної підготовки. Вона передбачає використання веб-технологій для поширення знань, підвищення доступності освітніх ресурсів і стимулювання самостійного навчання у динамічно зростаючій сфері експлуатації БПЛА.

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі дипломної роботи було здійснено всебічний аналіз завдань і предметної області, пов’язаних із розробкою веб-застосунку для дистанційного навчання операторів безпілотних літальних апаратів, зокрема FPV-дронів. Обґрунтовано актуальність проєкту в умовах зростаючого попиту на підготовку кваліфікованих фахівців у цій галузі, що особливо важливо в контексті сучасної ситуації в Україні, де БПЛА відіграють важливу роль у військовій, гуманітарній та цивільній сферах.

Було встановлено, що існуючі освітні рішення не повністю задовольняють запити потенційних користувачів. Частина матеріалів є розпорошеною, доступною переважно англійською мовою, або ж представлена у вигляді окремих відеоуроків, форумів чи каналів у соціальних мережах. Крім того, значна частина курсів має виключно очний формат, вартість участі в яких є суттєвим бар’єром для широких верств населення. Це знижує доступність освіти в умовах, коли багато громадян потребують перекваліфікації або прагнуть долучитися до технічної підтримки оборонного сектору.

Також веб-застосунок може стати ефективним рішенням, яке дозволить централізовано зібрати, впорядкувати та подати українською мовою всю необхідну інформацію для теоретичної підготовки операторів FPV-дронів. Такий формат навчання забезпечить гнучкість, доступність з будь-якого пристрою, можливість самостійного проходження курсів у зручному темпі та регулярне оновлення контенту. Водночас особливу увагу було приділено тому, щоб система залишалася простою у використанні, адаптивною до рівня користувача та відкритою для розширення навчального матеріалу адміністраторами платформи.

Серед переваг веб-формату також виділено економічність, відсутність територіальних обмежень, можливість масштабування проєкту та мінімальні

вимоги до технічного обладнання. Водночас було виявлено і певні обмеження дистанційного навчання — передусім, неможливість відпрацювання практичних навичок керування дроном у реальному середовищі. Однак платформа здатна закріпити фундаментальні знання, що необхідні перед початком практичної частини підготовки.

Узагальнення предметної області дозволило виокремити ключові технічні, освітні та соціальні аспекти, на які має реагувати система. Визначено, що майбутній застосунок має відповідати вимогам гнучкості, простоти, розширюваності, безпечного доступу та зручності для різних категорій користувачів.

Підсумовуючи, перший розділ заклав концептуальну та методичну основу для подальшої розробки веб-застосунку. Отримані результати аналізу дають змогу сформулювати технічні вимоги до системи, визначити оптимальні підходи до її реалізації та забезпечити створення ефективного інструменту для масового доступу до знань у сфері експлуатації БПЛА. Наступні розділи будуть присвячені етапам проєктування, розробки, впровадження та оцінки ефективності розробленої платформи.

РОЗДІЛ 2. ОГЛЯД ТА ПОРІВНЯННЯ ТЕХНОЛОГІЙ РОЗРОБКИ.

2.1 Опис функціональних можливостей навчальної веб-платформи.

Цей пункт пропонує вичерпний опис функціональної поведінки освітньої веб-платформи, присвяченої систематичному навчанню експлуатації дронів. Він містить такий набір функцій: загальнодоступні інформаційні сторінки, сторінка автентифікації та реєстрації, закритий каталог курсів, переглядач контенту, орієнтованим на урок, редактор профілю користувача. Є два типи користувачів: викладачі та студенти. Студент може тільки переглядати навчальний матеріал. Викладач контролює створення, зміну та видалення курсів та уроків. Всі пояснення, наведені нижче, зосереджені на тому, що має робити кожна складова системи і як з нею працює учень або викладач. Особлива увага приділяється вплетенню домену дронів у кожну поверхню, щоб мета платформи - підготовка компетентних пілотів дронів, обізнаних з правилами - залишалася безпомилковою від першого кліку до останнього уроку.

Домашня сторінка, виконує три взаємопов'язані функції. По-перше, вона представляє безпосередню тематику навчальної веб-платформи - відвідувачі відразу розуміють призначення цього сайту. По-друге, сторінка функціонує як основний вузол маршрутизації. Фіксована навігаційна панель видна перед будь-яким прокручуванням, гарантуючи, що новачок завжди може перейти на сторінки «Про нас», «Контакти» або «Логін» одним кліком миші. По-третє, хоча і менш помітно, це закладає психологічну основу для конверсії. Тизерні обкладинки висвітлюють передові теми, але розділ курси залишається не доступним, доки користувач не пройде аутентифікацію. Ця навмисна

недоступність натякає на глибину доступної підготовки, яка підштовхує гостя до реєстрації.

Сторінка «Про нас» розширює розповідь про довіру до платформи, використовуючи прозу, а не маркетингові слогани. Важливо, що сторінка пояснює, чому структурована освіта з використання дронів має значення: безпечна експлуатація вимагає теоретичних та технічних навичок. Висвітлюючи навчальний контент як шлях до особистої майстерності та запоруку оволодінні навчиок, які необхідні пілоту, сторінка виправдовує закриті частини сайту, що йдуть далі.

Оскільки початківці часто стикаються з проблемами та питаннями, сторінка «Контакти» містить всі необхідні контакти для зворотного зв’язку. Вона містить стислий абзац із закликом надсилати свої запити.

Інтерфейс для входу в систему навмисно мінімальний. Два марковані поля введення - електронна пошта та пароль - розташовані поруч, візуально відокремлені від декоративних елементів, так що погляд потрапляє безпосередньо на них. Перевірка на стороні клієнта миттєво спрацьовує в разі очевидних помилок, наприклад, якщо в адресі відсутній символ «@». Правильне введення пари облікових даних призводить до перенаправлення до каталогу «Курси». Невдала спроба призводить до того що, з'являється повідомлення про те, що комбінація є недійсною.

Якщо в користувача ще не створенно профіль, він може його створити в розділі реєстрації. Є чотири поля, які потрібно заповнити: ім’я, призвіще, електронна пошта та пароль. Також, при реєстрації потрібно обрати один із двох типів користувачів: студент або викладач. Тип користувача визначає можливості та достопу на навчальній веб-платформі.

Після успішної аутентифікації панель навігації непомітно і швидко змінюється. Елемент «Логін» зникає, натомість вкладка «Курси» стають доступними і з’являється новий елемент «Профіль». Ця заміна, виконана без

повного оновлення сторінки, тонко сигналізує про те, що гість перетнув поріг від спостерігача до учасника.

Кожен курс представлений у вигляді картки, яка займає велику горизонтальну площу, що дозволяє ілюстративному зображенню банера та реченню з описовим текстом співіснувати без тісноти. Наприклад, курс під назвою «Основи FPV» має підпис «Навчіться налаштовувати петлі P-I-D і підтримувати ситуаційну обізнаність під час польоту в акробатичному режимі».

Натиснувши на картку курсу, учень потрапляє до спеціального детального перегляду, верхнім елементом якого є широкий баннер, що відповідає темі. У центральній колонці нижче перераховані уроки, кожному з яких передує цифровий індекс, що вказує на послідовність і прогрес. Умовні назви уроків підкреслюють дію, наприклад, «Калібрування електронного регулятора швидкості» або «Визначення висоти повернення додому». Кожен заголовок уроку функціонує як внутрішнє гіперпосилання, що перенаправляє на повну сторінку уроку, формуючи таким чином основу робочого процесу учня.

Сторінка уроку дотримується незмінної візуальної ієрархії. Вона починається з фотографії з високою роздільною здатністю. Одразу за нею йде заголовок - назва уроку, надрукована великим шрифтом, який затьмарює навколишній текст, щоб позначити початок основного змісту. Оповідний текст. Прив'язка сторінки - це вбудований фрейм YouTube з демонстраційним відео. Ця структура - зображення, заголовок, текст, відео - ніколи не змінюється, навчаючи учнів швидко знаходити інформацію і підкреслюючи, що навчальна програма є продуманою і послідовною.

Вибравши пункт «Профіль», учень потрапляє на сторінку профілю. Тут відображається вся інформація про користувача: його ім’я, прізвище, електронна пошта та його тип: студент чи викладач. Є кнопка для редагувати профілю, яка дає змогу змінити всю інформацію включно з паролем, окрім типу користувача. Після натискання кнопки «Зберегти» клієнт надсилає серверу

запит, і в разі успіху нас перенаправляє назад на сторінку з профілем, але уже із зміненими даними користувача.

Після входу до системи під обліковим записом викладача, користувач перенаправляється до розділу «Курси», де відображається перелік наявних курсів. Над списком з’являється кнопка «Створити курс», натискання якої відкриває нову сторінку з формою, що містить два поля: «Назва курсу» та «Короткий опис». Після заповнення цих полів і надсилання форми сервер створює новий запис у таблиці «Курси» та повертає оновлений список курсів.

При переході на будь-який курс на сторінці з’являються іконки «Редагувати» та «Видалити». Редагування здійснюється у тій формі: поля форми автоматично заповнюються поточними значеннями назви та опису, і викладач може змінити їх за потреби. Після збереження оновлених даних відбувається оновлення відповідного рядка в базі даних. Видалення курсу супроводжується каскадним видаленням всіх пов’язаних уроків.

Відкривши деталі будь-якого курсу, викладач бачить перелік його уроків та кнопку «Створити урок». Форма створення уроку містить чотири елементи:

- Завантаження зображення — при виборі файлу у формі показується мініатюрний preview перед відправленням.
- Назва уроку — текстове поле.
- Опис (контент) — багаторядкове текстове поле.
- Посилання на YouTube — текстове поле.

Після валідації форма надсилається, сервер зберігає файл зображення у статичну папку під унікальним UUID-іменем, повертає шлях до нього, а потім створює запис у таблиці «Уроки», вказавши всі чотири поля. Редагування уроку повторно відкриває цю ж форму з поточними даними, заміна зображення за бажанням викликає видалення старого файлу на сервері та завантаження нового. Після натиснення кнопки про видалення уроку, видаляється і запис у базі, і файл-зображення зі сховища.

Ролі на платформі чітко розділені — гість, студент та викладач. Гості бачать лише маркетингові сторінки; студенти після входу отримують доступ до перегляду курсів та редагування інформації в своєму профілі; викладачі ж мають повне право створювати, редагувати та видаляти курси й уроки. Кожен запит на зміну ресурсів захищений перевіркою JWT-токена та перевіркою ролі на сервері — у разі спроби несанкціонованої операції сервер повертає HTTP 403, а клієнт відображає дружнє повідомлення про відсутність доступу.

2.2 Вибір архітектури веб-додатку

Коли будь-який новий програмний проект переходить від задуму до реалізації, одне з перших - і найбільш філософських - рішень стосується загальної структурної форми. Чи повинні всі проблеми, пов'язані з виконанням, жити разом всередині одного розгортаємого модуля (моноліту), чи система повинна бути розбита на багато вузькоспрямованих, пов'язаних між собою мережею виконуваних файлів (мікросервісів)? Нижче я намагатимусь надати об'ємний аналіз обох підходів з висвітленням їхніх переваг і недоліків [5], після чого будуть наведені аргументи на користь монолітного дизайну для платформи для навчання про безпілотники, про яку йшлося раніше.

2.2.1 Монолітна архітектура

Монолітна архітектура є традиційним підходом до створення програмного забезпечення, коли вся система функціонує як єдине ціле.

Переваги:

- Єдина ментальна модель. Весь вихідний код, конфігурація та інфраструктура знаходяться в одному місці, тому розробник може

відстежувати запит від точки входу HTTP до SQL-запиту без переходів між репозиторіями або контейнерами.

- Комунікація в процесі роботи. Оскільки кожен компонент використовує один і той самий простір пам'яті, виклики є простими викликами функцій.
- Єдиний конвеєр побудови та розгортання. CI/CD - це просто: один раз скомпілюйте, один раз запустіть юніт-тести. Відкотити систему так само просто.
- Низькі операційні витрати. Немає потреби в сітці сервісів. Для продуктів на ранніх стадіях розробки це означає скорочення витрат на хмарні технології та зменшення проблем при роботі з клієнтами.
- Швидке підключення та рефакторинг. Нові розробники запускають всю платформу двома-трьома командами, а наскрізні зміни - перейменування об'єкта домену, введення нового рівня перевірки - можна зробити за допомогою одного pull-запиту.

Недоліки:

- Масштабованість. Якщо одна кінцева точка (скажімо, генерація мініатюр відео) вимагає багато процесорних ресурсів, вам доведеться масштабувати весь контейнер програми, витрачаючи ресурси на менш завантажені модулі.
- Потенційне розростання кодової бази. У міру накопичення функцій чіткі межі можуть розмиватися, час юніт-тестування може збільшуватися. Дисципліна і модульний дизайн всередині моноліту мають важливе значення.
- Технологічна прив'язка. Оскільки кожен модуль повинен мати спільне середовище виконання, використання декількох технологій є складним.

2.2.2 Мікросервісна архітектура

Мікросервісна архітектура — це підхід до розробки програмного забезпечення, при якому система складається з окремих, незалежних сервісів, кожен із яких виконує одну конкретну функцію. У цьому розділі розглянуто ключові переваги та виклики мікросервісного підходу в контексті побудови масштабованих і гнучких платформ.

Переваги:

- Гранульована масштабованість та відмовостійкість. Кожен сервіс може бути реплікований незалежно, що дозволяє команді завантажувати процесор тільки там, де дійсно є навантаження. Збій в аналітичному сервісі не призводить до автоматичного припинення входу в систему або відтворення уроків, наприклад.
- Незалежна каденція розгортання. Команди, відповідальні за різні домени, можуть оновлювати, відкочувати та експериментувати, не координуючи вікна релізу для всієї платформи, що прискорює впровадження інновацій у великих організаціях.
- Гнучкість вибору мови. Сервіс обробки зображень, що вимагає великих обчислень, може бути написаний на Go, тоді як API, орієнтований на студентів, залишається на Python, дозволяючи кожній команді обирати інструментарій, який найкраще підходить для її проблемного простору.
- Чіткі межі володіння. Оскільки кожен сервіс має явний контракт (зазвичай це інтерфейс HTTP), організаційні команди чітко відображають код, уникаючи багато конфліктів при злитті у мега-репозиторії.

Недоліки:

- Значна операційна складність. Запуск навіть півдюжини сервісів, як правило, вимагає оркестровки контейнерів, виявлення сервісів, розподіленого ведення журналів, трасування і надійних конвеєрів CI/CD - механізмів, які повинна вивчити і підтримувати невелика команда.
- Розподілені режими збоїв. Кожен міжсервісний виклик може зазнавати стрибків затримок, невідповідності версій або часткових відмов, що змушує застосовувати повторні спроби, автоматичні вимикачі та моделі узгодженості.
- Вищі базові витрати. Додаткові балансувальники навантаження, більша кількість контейнерів, плата за міжсервісну пропускну здатність і кілька сховищ даних збільшують щомісячні рахунки за хмару задовго до того, як трафік їх виправдовує.
- Складніше налагоджувати. Відстеження періодичної помилки в п'яти сервісах, вимагає дисциплінованого інструментарію і може сповільнити аналіз першопричини, особливо в умовах дефіциту часу.

2.2.3 Обгрунтування вибору монолітної архітектури

Вибір архітектури напряму залежить від масштабу проєкту, ресурсів команди та технічних вимог на старті. У цьому розділі обґрунтовано доцільність використання монолітної архітектури для першої версії платформи.

- Розмір команди та пропускна здатність зв'язку. Один або декілька розробників можуть поділитися одним репозиторієм, переглянути

запити на злиття один одного та розкрутити ідентичні локальні середовища за лічені хвилини.

- Обсяг роботи з першим запуском. У першому запуску будуть розміщені статичні зображення уроків, потік вбудованих відео і виконувати скромні запити PostgreSQL. Ніщо в списку вимог не передбачає асиметрію масштабування на кінцеву точку настільки драматичною, що одному модулю потрібно потрібно дуже багато ресурсів.
- Простота транзакцій. Адміністративні операції - «створити курс і його перший урок» або «видалити курс» - підходить для монолітної архітектури. Моноліт може відкривати одну транзакцію Postgres, виконувати всі вставки або видаляти та фіксувати. У мікросервісній архітектурі таке завдання вимагало б занадто багато ресурсів для такої простої задачі.
- Швидке вирішення дефектів. У монолітній архітектурі знайти і виправити помилку в одному репозиторії набагто легше, якщо проект не дуже складний.
- Попередній моноліт не виключає переходу на мікросервісну архітектуру. Добре структурована монолітна архітектруа зменшує проблеми для переходу на мікросервісну архітектуру при потребі у майбутньому [6].

2.3 Інструменти для розробки та тестування

Вибір правильних інструментів розробки - це більше, ніж питання особистих уподобань; він визначає, наскільки ефективно команда може втілювати ідеї в надійне програмне забезпечення, яке легко підтримувати. Редактор або платформа для розробки формує щоденний ритм проекту,

визначаючи час налаштування, швидкість рефакторингу, тощо. Добре підібрані інструменти спрощують робочі процеси та виявляють помилки на ранній стадії. Серед повнофункціональних редакторів, придатних для мого Full-stack проекту, PyCharm та Visual Studio Code домінують у більшості порівнянь, проте вони відрізняються за своїм характером. В цьому розділі я коротко розгляну переваги та недоліки цих двох редакторів та аргументую свій вибір.

Рисунок 2.1. – PyCharm

Серед переваг PyCharm можна виділити такі:

- Спеціально розроблений для Python. Його статичний аналіз коду, дії рефактора на місці та підказки «намірів» налаштовані на граматику та ідіоми Python, зменшуючи кількість малопомітних помилок (невикористаний імпорт, затінена змінна) ще до того, як вони потраплять до інтерпретатора.
- Найкраща у своєму класі підтримка Django, Flask, FastAPI та SQLAlchemy. Вбудовані інспектори розуміють ORM-моделі і навіть пропонують автозаповнення для імен полів всередині необроблених SQL-рядків.

- Потужний налагоджувач і запуск тестів. Точки зупинки, умовні годинники та перегляд вбудованих змінних доступні з коробки; інтеграція з pytest дозволяє запускати окремі параметризовані кейси одним клацанням миші.

Так такі недоліки:

- Важкість редактора. Час запуску та споживання оперативної пам'яті помітно вищі, ніж у інших редакторах коду. На ноутбуці з обмеженою пам'яттю робота з кількома проектами може бути повільною.
- Вартість ліцензії для повного набору функцій. Ліцензія Professional передбачає щорічну оплату.
- Світогляд, орієнтований на Python. Мій стек включає значну частину роботи з React/JavaScript, інструментарій JavaScript редактора відстає від автозаповнення VS Code, а інтеграція з ESLint відчувається менш плавною.
- Екосистема плагінів менша і повільніше розвивається, ніж у VS Code, особливо для новітніх фронтенд-фреймворків.

Рисунок 2.2. – VS Code

Тепер розглянемо переваги та недоліки VSCode [7].

Переваги:

- Легкість застосунку. Базовий додаток швидко запускається і споживає скромні ресурси. Ви встановлюєте лише ті розширення, які вам потрібні.
- Першокласна багатомовність. Підказки коду, підсвічування JSX-синтаксису та перевірка інтерфейсу в React відчуваються як природна частина середовища, а розширення для Python (з Pylance) майже не поступається PyCharm за швидкістю автозавершення.
- Ринок розширень у веб-масштабі. Тисячі активно підтримуваних плагінів і навіть попередній перегляд файлів Figma - часто випущені через кілька днів після того, як новий інструмент потрапляє на GitHub.
- Застосунок є повністю безкоштовний та ніяких ліцензійних платежів.

Недоліки:

- Потребує ручного вибору розширень. Ви повинні зібрати, налаштувати і періодично перевіряти власний набір плагінів, щоб уникнути конфліктів і зниження продуктивності.
- Менш упереджена структура проекту. VS Code не нав'язує жодної конкретної схеми віртуального середовища або тестового середовища.

Тому VS Code є прагматичним вибором для моєї навчальної веб-платформи. Код проекту поєднує FastAPI (Python) та React (JavaScript). Єдиний крос-платформний редактор, який одночасно повноцінно підтримує обидві мови, дозволяє комфортно розробляти все в одному середовищі. Цей редактор коду є безкоштовний. Нарешті, швидкий темп розвитку ринку розширень

означає, що в міру розширення веб-платформи, можна з легкістю додавати і працювати з іншими технологіями. У сукупності VS Code забезпечує найширше охоплення технологій, що робить його розумною основою для розробки [5].

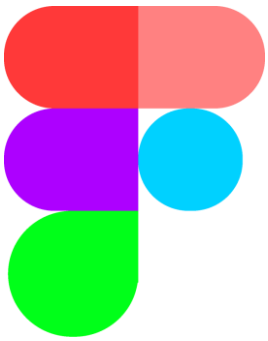


Рисунок 2.3. – Figma

Figma слугує моїм єдиним хмарним центром дизайну [8], де зберігаються каркаси, відшліфовані макети інтерфейсу та інтерактивні прототипи.
Я обрала Figma, тому що вона надає спільне полотно на основі браузера, де в одному місці живуть каркаси, високоточні макети та прототипи, які можна клікнути. Дизайнери та розробники можуть коментувати та ітерації в режимі реального часу з будь-якого пристрою, скорочуючи цикл зворотного зв'язку до декількох хвилин. У Dev Mode відображаються точні кольори, інтервали та фрагменти CSS, тому перехід до коду React відбувається без проблем, а безкоштовний рівень охоплює все, що потрібно для моєї навчально платформи без встановлення, без затримок з ліцензуванням, лише миттєва співпраця та ідеальні специфікації.

2.4 Вибір технологій для розробки

У цьому розділі буде описано, які технології було обрано для розробки навчальної веб-платформи, зокрема – front-end та back-end частин та обґрунтування причин такого вибору.

2.4.1 Вибір технологій для фронтенд частини

Для розробки front-end частини я обрала React [9]. Коротко, я опишу чому саме React, а не інші фреймворки чи бібліотеки на JS.

React пропонує поєднання гнучкості, зрілості та глибини екосистеми, що майже ідеально відповідає потребам моєї платформи для навчання. Як бібліотека, а не жорсткий фреймворк з повним стеком, він дозволяє нам починати з малого - один інтерактивний компонент курсу тут, панель профілю користувача там - і масштабувати його до багатшого односторінкового додатку без переписування початкового коду. Завдяки моделі Virtual DOM інтерфейс адаптивно реагує навіть на ноутбуках середнього рівня, що важливо, коли студенти одночасно переглядають навчальні відео на YouTube і прокручують текст уроку. Оскільки поверхня API React навмисно мінімальна - по суті, це useState, useEffect та JSX - нові учасники розуміють основи за лічені дні, а навколишня екосистема (React Router, TanStack Query, Zustand, Next.js) надає перевірені рішення, коли нам потрібна маршрутизація, синхронізація станів або рендеринг на стороні сервера. Завдяки цим перевагам та популярності екосистеми, потрібну інформацію чи вирішення проблеми можна знайти набагато простіше, так як спільнота, яка використовує цю бібліотеку є великою [10].

Vue JS, безумовно, забезпечує елегантну реактивність і приємний синтаксис однофайлових компонентів, але його спільнота, залишається меншою, тому сторонні віджети, безголові бібліотеки компонентів або навчальний контент з'являються рідше - і часто пізніше - ніж їхні React-еквіваленти. З іншого боку, Angular нав'язує комплексну архітектуру з декораторами, ін'єкціями залежностей та суворими модульними конвенціями. Ця сила чудово працює у великих корпоративних командах, але може сповільнити роботу для невеликого проекту під шарами шаблонів. React

знаходить золоту середину: достатньо легкий, щоб підтримувати високу початкову швидкість, але достатньо усталений, щоб кожна інтеграція вже мала першокласні зв'язки з React. З цих причин - гнучке поетапне впровадження, неперевершена широта екосистеми та найнижча сукупна вартість навчання та розширення - React є прагматичним фронтенд-фундаментом для цього проекту, випереджаючи як Vue JS, так і Angular для специфічних обмежень мого проекту [11].

2.4.2 Вибір технологій для бекенд частини

Для back-end частини було обрано FastAPI [12]. FastAPI поєднує в собі швидкість сучасного інструментарію Python зі зручним для розробників дизайном, який підходить для нашої платформи навчання дронів краще, ніж Flask або Django. Нижче розглянуто причини такого вибору детальніше.

FastAPI підходить для бекенду нашої платформи для навчання дронів, оскільки поєднує сучасні підказки типів Python з мінімалістичним, високопродуктивним ядром. Виражаючи кожен запит і схему відповіді у вигляді моделей Pydantic з анотаціями за допомогою стандартного синтаксису, ми отримуємо автоматичну перевірку даних, генерацію JSON-схем і автозавершення редактора без зайвих шаблонів. Ці ж анотації перетікають прямо в інтерактивну документацію: в момент написання кінцевої точки FastAPI відкриває самооновлювані сторінки інтерфейсу Swagger UI і ReDoc. Це дуже допомагає швидше розробляти веб-застосунки разом з front-end бібліотеками, такими як React.

Не менш важливо, що FastAPI є асинхронним з самого початку. Побудований на Starlette і обслуговується Uvicorn, він пропонує нам оголошувати маршрути async def, які вивільняють цикл обробки подій під час очікування бази даних або зовнішніх API. Ця неблокуюча модель забезпечує швидкий час відгуку, коли десятки учнів одночасно отримують метадані уроків

або завантажують зображення курсів - всі операції, які б зв'язали потоки у традиційному синхронному стеку. Бенчмарки постійно ставлять FastAPI на перше місце серед фреймворків Python і впритул до Node або Go, а це означає, що ми можемо відкласти горизонтальне масштабування до того моменту, коли реальні сплески трафіку виправдають це.

Вибір FastAPI також допомагає нам уникнути компромісів, притаманних Flask та Django. Flask залишається напрочуд легким, але для того, щоб відповідати валідації FastAPI на основі типів, підтримці асинхронізації та автоматичного документообігу, нам довелося б прикручувати сторонні розширення та з'єднувати їх вручну. Django, навпаки, постачається як моноліт з шаблонами, ORM та конвеєром рендерингу на стороні сервера - ресурси, від яких ми б одразу відмовилися, оскільки інтерфейс платформи повністю живе в React. Невикористання цього механізму сповільнило б час завантаження, ускладнило б конфігурацію і змусило б нас боротися з проміжним програмним забезпеченням за замовчуванням. FastAPI знаходиться посередині: достатньо стрункий, щоб зменшити наші розумові витрати, але достатньо розумний, щоб забезпечити надійні будівельні блоки «з коробки».

Імпульс фреймворку в Python-спільноті гарантує достатню кількість плагінів і прикладів. Вже існують підтримувані бібліотеки, задокументовані в тому ж стилі, що і сам FastAPI. Така зрілість екосистеми скорочує цикли розробки. Взяті разом - безпека типів даних, жива документація, вбудована асинхронізація, висока продуктивність, мінімальний поріг входження і безліч готових розширень - FastAPI пропонує саме той баланс швидкості і надійності, якого вимагає наш бекенд, перевершуючи Flask за сучасними можливостями і обходячи важку структуру Django [13].

ВИСНОВОК ДО РОЗДІЛУ 2

У підсумку, функціональна поверхня платформи може здаватися вузькою через те, що публічні сторінки, автентифікація, перегляд курсів, редагування профілю та адміністративна консоль, але кожен елемент чітко визначений для того, щоб служити головній меті - підвищенню кваліфікації пілотів безпілотних літальних апаратів. Зображення і відео відтворюють реалістичність технічного обслуговування безпілотних літальних апаратів. Адміністратори, озброєні невеликим, але потужним інструментарієм, можуть наповнювати навчальну програму з нуля, не вдаючись до надмірної складності. Учні, в свою чергу, насолоджуються безперешкодним шляхом від початківця до майстра: вони входять в систему, обирають курс, засвоюють ретельно впорядковані уроки. Зосереджуючись на цьому дисциплінованому наборі функцій - без форумів, без гейміфікації прогресу, без платіжного шлюзу - проект гарантує, що його мінімальний життєздатний продукт можна буде створити в єдиному сховищі, протестувати за короткі ітерації і зробити корисним для пілотів-початківців, які потребують надійних орієнтирів у такий непростий час.

Враховуючи невелику інженерну команду, скромний початковий набір функцій (розпізнавання користувача, управління курсом, тощо) та передбачувані схеми трафіку, моноліт є кращим варіантом. Він мінімізує інфраструктурні рішення. Важливо відзначити, що вибрана монолітна архітектура не виключає майбутнього переходу на мікросервісну архітектуру у разі ускладнення бізнес-логіки веб-платформи.

Для розробки веб-застосунку було обрано VS Code через його гнучкість, легкість та через те, що цей редактор коду є безкоштовним.

React було обрано, тому що його легка, компонентна модель та широка екосистема дозволяють невеликій команді швидко створювати адаптивний інтерфейс, використовуючи величезну кількість готових бібліотек.

Для бекенду FastAPI забезпечує валідацію на основі підказок типів даних, автоматичну документацію Swagger та власну асинхронну продуктивність, що пришвидшує розробку без шкоди для безпеки та масштабованості.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ

3.1 Фронтенд частина

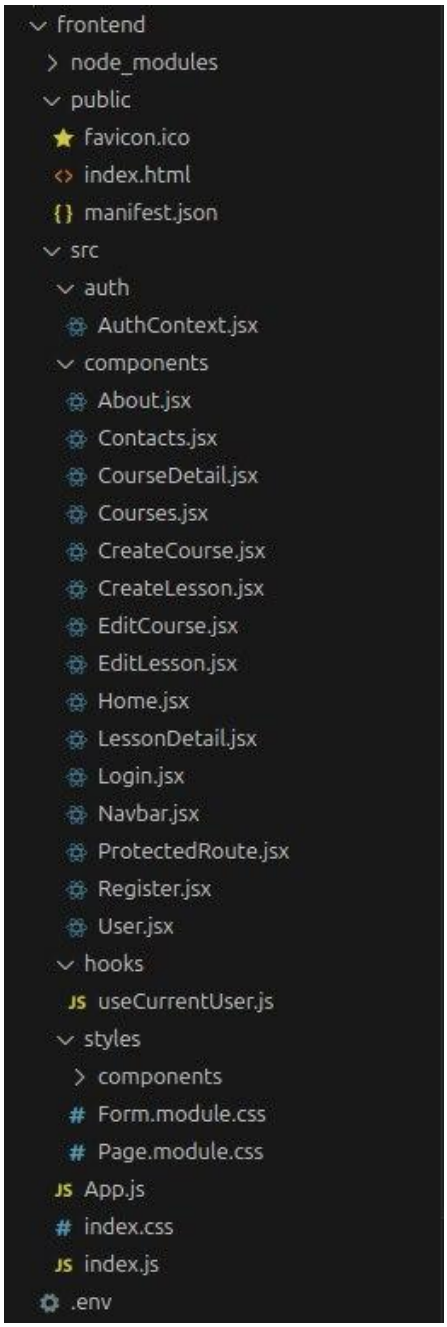


Рисунок 3.1. – Структура фронтенд частини

Клієнтська частина проєкту побудована на React з акцентом на компонентну архітектуру та чітке розділення обов’язків. Основна ідея полягала в тому, щоб кожен UI-блок був якомога незалежнішим: компоненти мало

взаємодіють один з одним напряму, а всі загальні сервіси — мережеві запити, авторизація, теми — винесені в окремі контексти або хоки.

У точці входу *src/index.js* підключається глобальний CSS-ресет *index.css*, який встановлює базові значення для шрифтів, відступів та коробкового моделювання. Після цього ReactDOM рендерить головний компонент <*App/*>, розташований в *src/App.js*. Сам <*App/*> містить лише дві речі: компонент *Navbar* і набір маршрутів, організованих через *react-router-dom*.

Визначення маршрутів виконано з використанням <*Routes*> та <*Route*>, причому частина із захищеним доступом обгорнута у компонент *ProtectedRoute*. Цей компонент звертається до контексту *AuthContext* і вирішує: якщо токена немає — перенаправити на сторінку логіну, інакше відобразити дочірній контент.

Авторизація реалізована через окремий контекст *src/auth/AuthContext.js*, який зберігає JWT-токен у localStorage та надає методи *login()* і *logout()*. Перехоплення та додавання заголовка *Authorization* відбувається двома шляхами: або компоненти самі витягують контекст і формують заголовки при кожному *fetch*, або застосовується невеликий хук *useCurrentUser*, який автоматично надсилає GET-запит за профілем поточного користувача.

Всі форми (реєстрація, логін, створення/редагування курсів та уроків, редагування профілю) обгорнуті стилями з *Form.module.css*. Цей CSS-модуль надає уніфіковані класи *.form*, *.input*, *.textarea*, *.buttonPrimary* і т. д., що забезпечує єдиний вигляд усіх елементів введення та кнопок. Кожна сторінка або велика секція оточена класом .container з *Page.module.css*, який відповідає за ширину макету, фоновий білий прямокутник із тінню та відступи.

У директорії *src/components* кожен компонент має свій власний CSS-модуль (наприклад, *Courses.module.css*, *CourseDetail.module.css*), у якому зібрані специфічні для цього компонента стилі. Така схема дозволяє змінювати розмітку чи кольори окремого екрану, не впливаючи на інші.

Обмін даними з сервером побудовано на стандартизованих REST-ендпоінтах. Наприклад, компонент *Courses.jsx* надсилає GET-запит до */courses*, потім відображає перелік курсів, а якщо роль поточного користувача — *teacher*, показує кнопку «+ Створити курс», що веде на маршрут */courses/new*. У деталях курсу *CourseDetail.jsx* додатково відбувається запит */lessons/by_course/{id}*, і для власника курсу доступні кнопки «Редагувати» та «Видалити», які викликають PUT і DELETE відповідно.

3.2 Бекенд частина

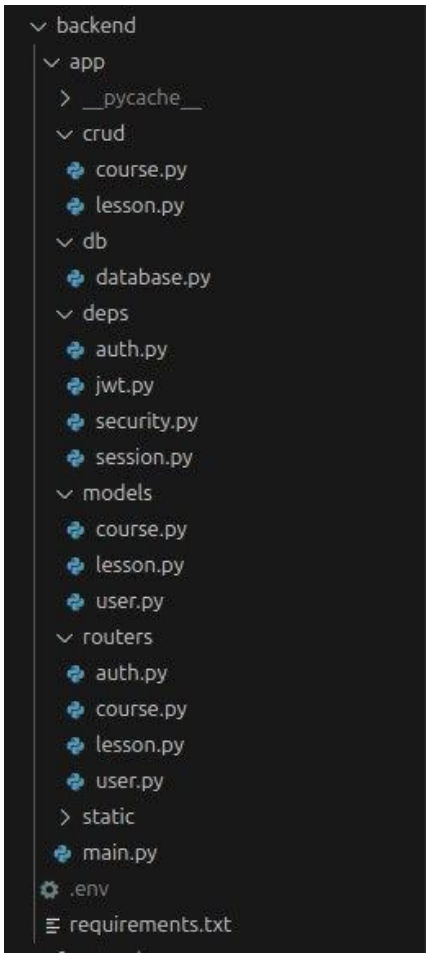


Рисунок 3.1. – Структура бекенд частини

Серверна частина реалізована на FastAPI з використанням SQLModel — ORM, що поєднує SQLAlchemy і Pydantic. При запуску застосунку

backend/app/main.py у секції lifespan викликається функція *init_db()*, яка створює всі таблиці в базі даних.

Конфігурація підключення до PostgreSQL [14][15] зчитується через python-dotenv із файлу *.env*: параметри *DATABASE_URL* та *SECRET_KEY* потрібні для створення двигуна SQLAlchemy та підписування JWT. FastAPI налаштовано з CORS-політикою, що дозволяє запити зі *http://localhost:3000*, та монтує статичні файли *app/static/uploaded_images* під шляхом */static*.

Аутентифікація: у модулі *app/deps/jwt.py* є методи *create_access_token()* та *decode_token()*, які створюють і перевіряють JWT з алгоритмом HS256. Безпосередньо у *app/deps/auth.py* визначено дві залежності: *get_current_user*, яка декодує токен, завантажує користувача з базою даних і повертає його або кидає відповідь 401; та *get_current_teacher*, яка перевіряє, що роль користувача — «teacher», інакше кидає відповідь 403.

Логіка CRUD винесена в окрему папку *app/crud*. Наприклад, у *app/crud/course.py* містяться функції *get_courses()*, *get_course()*, *create_course()*, *update_course()* і *delete_course()*. У кожній з них здійснюється перевірка прав: напр., при оновленні чи видаленні передається *teacher_id* поточного користувача, і якщо курс не належить цьому вчителю, функція повертає *None* або *False*, що в роутері перетворюється на *HTTPException* із кодом 403.

У папці *app/routers* кожний модуль відповідає за REST-ендпоінти. Так, у *auth.py* реалізовані */auth/register* та */auth/login*, де перший приймає JSON із *name*, *surname*, *email*, *password* і роль, хешує пароль через *passlib[bcrypt]* та зберігає користувача; другий приймає форму *OAuth2*, перевіряє пароль, повертає *access_token*.

Далі у *user.py* під шляхом /users/me реалізовані два методи: GET — повернути поточного користувача, PUT — оновити ймена, прізвище, email та пароль за бажанням. При зміні *email* є обробка помилки *IntegrityError*, яка повертає 400 із деталлю «Email already registered».

Аналогічно для уроків у *lesson.py* є маршрути POST */lessons/by_course/{course_id}*, GET */lessons/{id}*, PUT */lessons/{id}*, DELETE */lessons/{id}*, з обробкою завантаження зображення: файл приймається як *UploadFile*, зберігається в *static/uploaded_images*, а шлях зберігається в полі *image_path*.

3.3 Маршрутизація

Маршрути на фронт-енді реалізовані за допомогою react-router-dom. Параметричні маршрути типу */courses/:id* або */lessons/:id/edit* автоматично передають відповідні *id* у компоненти через хук *useParams*.

На бек-енді маршрути організовані в окремі роутери:

- */auth* — реєстрація та логін,
- */users/me* — отримання та оновлення даних поточного користувача,
- */courses* — CRUD для курсів,
- */lessons* — CRUD для уроків та вибірка уроків за курсом.

Ця чітка двоступенева маршрутизація (фронт із захищеними маршрутами та бекенд із REST-ендпоінтами) забезпечує зручну навігацію в UI й однозначність URL у всьому застосунку.

3.4 Схема взаємодії

1. При вході користувач заповнює форму в *Login.jsx*, запит відправляється на POST */auth/login*.
2. Сервер перевіряє облікові дані, повертає JWT.
3. Клієнт зберігає токен у контексті й *localStorage*, далі всі запити до захищених маршрутів (*/courses*, */lessons*) надсилають заголовок *Authorization*.

4. На бек-енді залежність *get_current_user* декодує токен, перевіряє валідність та повертає модель *User*.
5. Для ендпоінтів створення/редагування потрібна зв’язка *get_current_teacher*, що гарантує дозвіл.

Цей механізм легко масштабується: додавання нового ресурсу (наприклад, категорій або коментарів) вимагатиме лише створення моделі, CRUD-шару, роутера та відповідних компонентів на фронті.

3.5 База даних

Три основні таблиці: *users*, *courses* і *lessons*, пов’язані між собою відношеннями один-до-багатьох. Користувачі можуть бути студентами або вчителями (enum-поле *role*). Кожен курс належить одному вчителю (*teacher_id*), а кожен урок — одному курсу (*course_id*) з *ON DELETE CASCADE*, щоб при видаленні курсу автоматично видалялися всі пов’язані уроки.

Таким чином, архітектура забезпечує чіткий поділ відповідальностей між Front-end і Back-end, гарантує безпеку через JWT-аутифікацію, а також зручний інтерфейс і гнучку адмінку для викладачів та студентів.

ВИСНОВОК ДО РОЗДІЛУ 3

Розроблена архітектура поєднує сучасні підходи до створення односторінкових застосунків і надійної серверної логіки, що дає змогу досягти високої продуктивності, гнучкості та безпеки. На рівні клієнтської частини використано React із чітким компонентним поділом та CSS Modules, що гарантує ізоляцію стилів і зменшує ризик конфліктів у великих командах. Використання react-router-dom і власного ProtectedRoute дозволяє контролювати доступ до ресурсів, одночасно зберігаючи логіку маршрутизації максимально декларативною. Вбудований AuthContext забезпечує централізоване управління токеном і простий механізм авторизації на всіх рівнях інтерфейсу.

Серверна частина на FastAPI із SQLModel і PostgreSQL демонструє відмінні показники швидкодії та масштабованості. Завдяки чіткому розділенню відповідальностей між депенденсами, CRUD-шаром і роутерами, код залишився зрозумілим і легко тестується. JWT-автентифікація зводить до мінімуму залежність від сесійного стану, а вбудований Swagger UI дозволяє миттєво ознайомитися з усіма доступними ендпоінтами. Наявність ON DELETE CASCADE для уроків виключає можливість «висячих» записів і забезпечує цілісність даних.

Важливо, що обрана архітектура не лише відповідає поточним вимогам, а й легко адаптується до подальших розширень. Наприклад, інтеграція WebSocket для реального часу, впровадження кешування (Redis), або перехід на мікросервісну модель для складніших бізнес-процесів може бути реалізована без кардинальної перебудови коду. Додавання юніт- та інтеграційних тестів на рівні crud-функцій і компонентів React забезпечить безперебійну фронт-ту-бек перевірку під час CI/CD.

Таким чином, представлений стек технологій і структура коду забезпечують надійну платформу для освітнього веб-застосунку: від швидкої розробки інтерфейсу до безпечного та масштабованого бекенду. Висока модульність, автоматизація процесів і чітке дотримання принципів розділення відповідальностей створюють міцну основу для подальшого розвитку, інтеграцій та покращення користувацького досвіду.

РОЗДІЛ 4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБ-ЗАСТОСУНКУ

Веб-застосунок містить кілька ключових сторінок, кожна з яких виконує унікальні завдання та має власний функціонал. У цьому розділі буде докладно розглянуто кожну зі сторінок, проаналізовано її будову, цілі та особливості роботи.

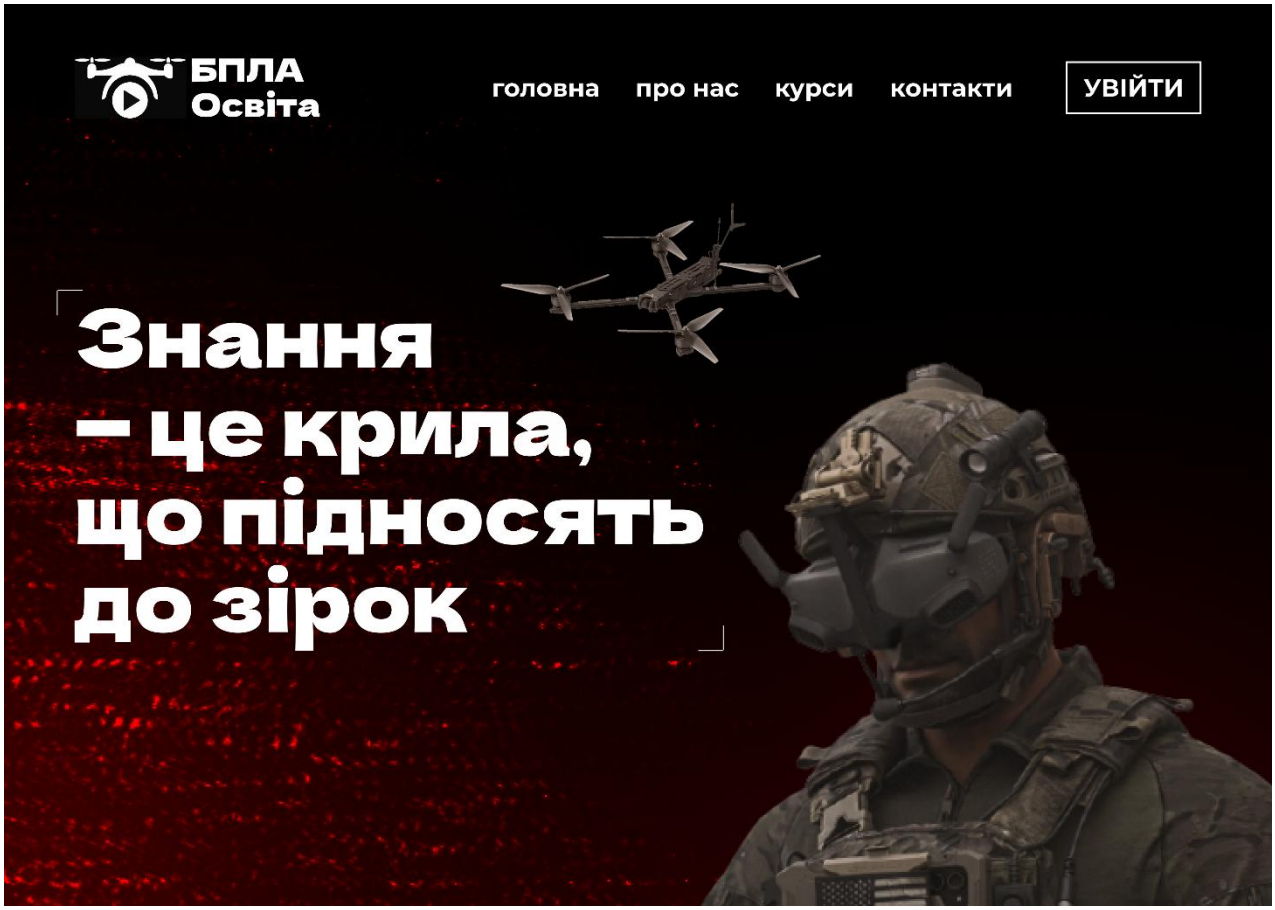


Рисунок 4.1. – Головна сторінка

Головна сторінка є вхідною точкою веб-застосунку та виконує навігаційно-інформаційну функцію. У верхній частині розміщено меню з посиланнями на основні розділи: «Головна», «Про нас», «Курси», «Контакти», а також кнопку «Увійти» для авторизації користувачів.

Центральний елемент сторінки — мотиваційне гасло, яке підкреслює освітню спрямованість проєкту. Візуальні компоненти акцентують увагу на сучасних технологіях та практичному спрямуванні навчання.

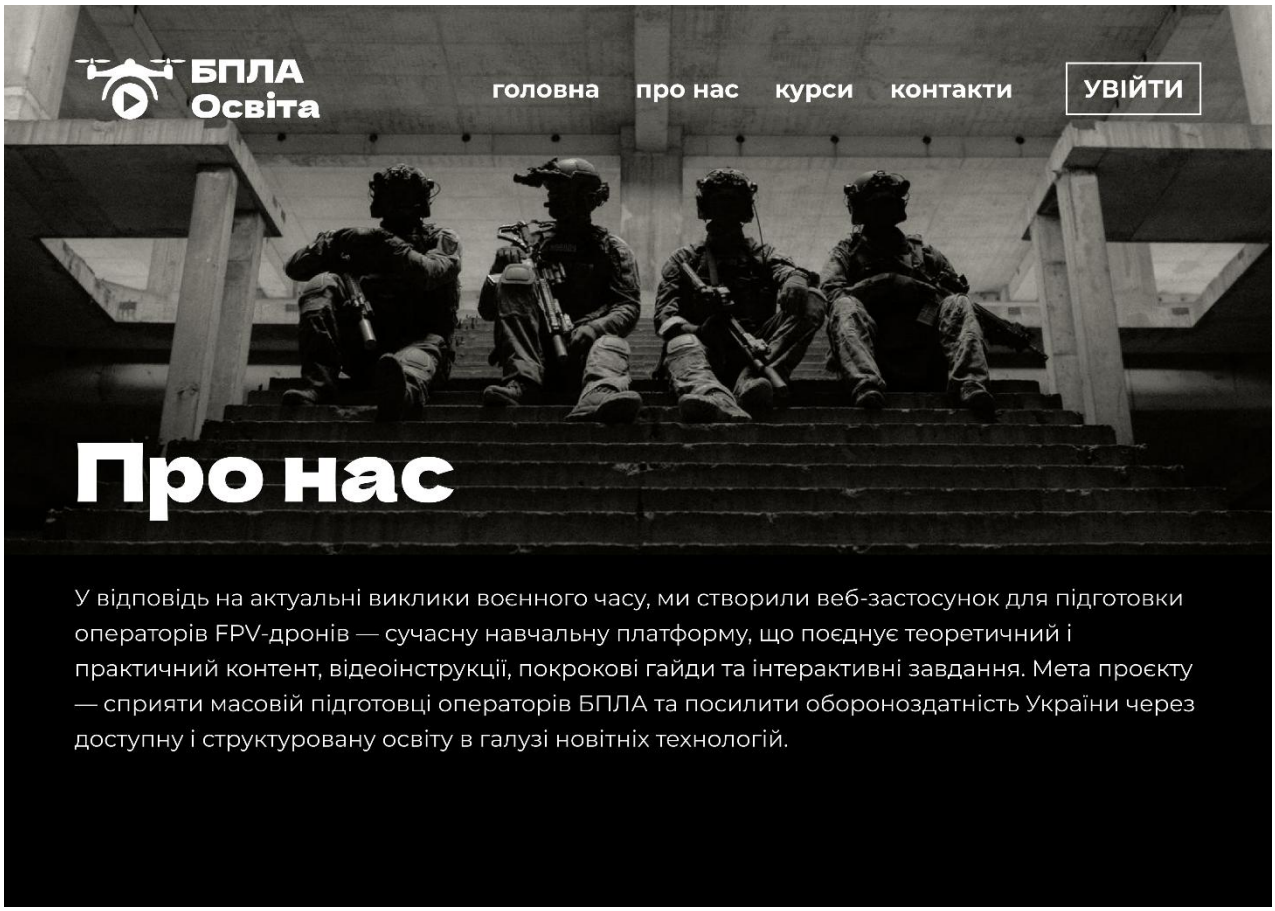


Рисунок 4.2. – Сторінка «Про нас»

Сторінка «Про нас» виконує інформативну функцію, окреслюючи концептуальні засади та цільове призначення освітнього веб-застосунку «БПЛА Освіта».

Матеріали сторінки пояснюють мотиваційні чинники створення сучасної навчальної платформи для підготовки операторів FPV-дронів в умовах воєнного часу. Текстовий контент висвітлює основні напрямки діяльності проєкту, зокрема поєднання теоретичного й практичного компонентів навчання, інтерактивних інструкцій та покрокових методичних матеріалів.

Ключовою метою ресурсу декларується сприяння масовій підготовці фахівців у сфері безпілотних технологій з метою посилення обороноздатності України. Підкреслюється орієнтація на доступність, структурованість і прикладний характер освітнього контенту.

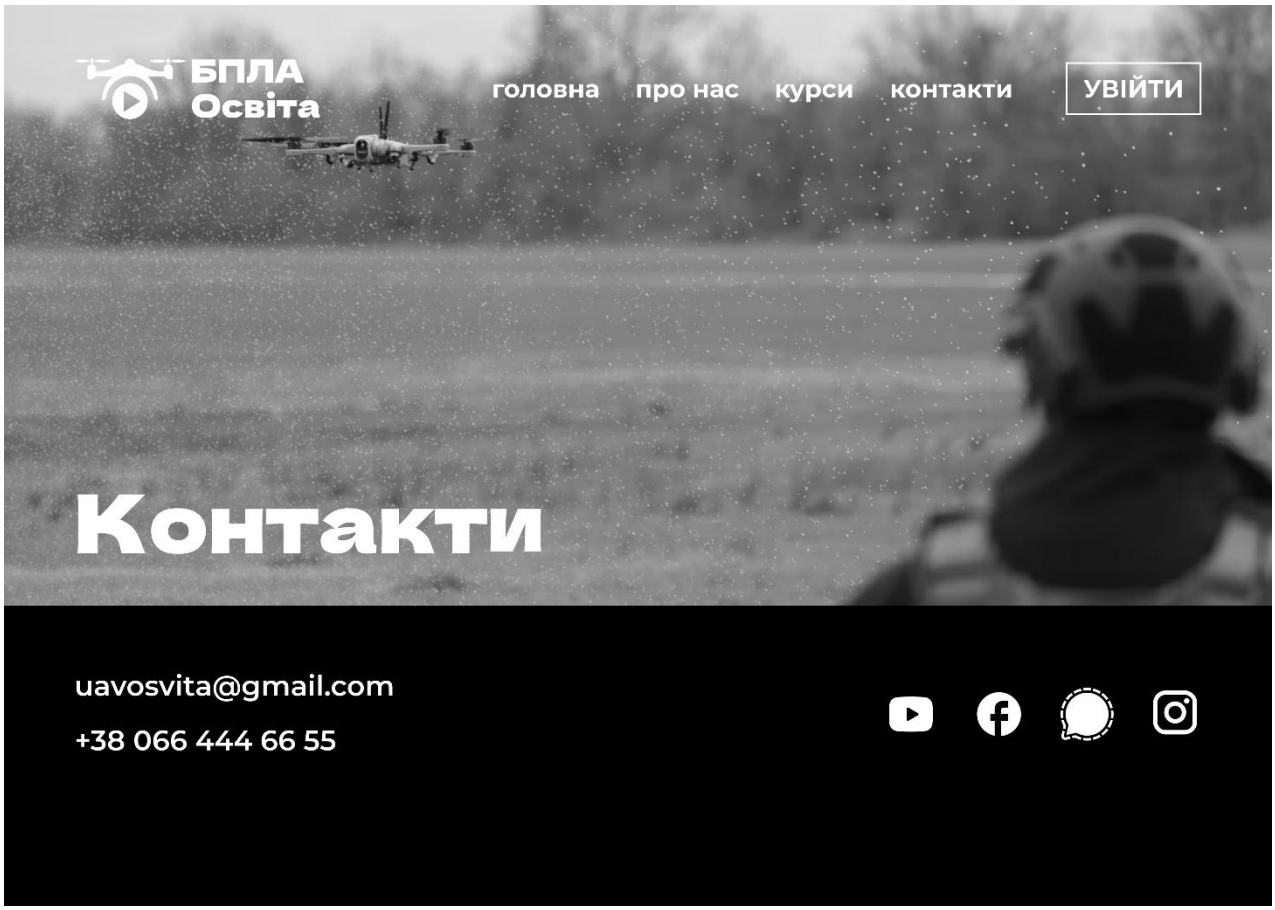


Рисунок 4.3. – Сторінка «Контакти»

Сторінка «Контакти» виконує комунікаційну функцію та надає користувачам доступ до офіційних каналів зв’язку з командою проєкту.

Її зміст сформовано з урахуванням потреб цільової аудиторії й охоплює електронну адресу, форму зворотного зв’язку, а також, за наявності, посилання на соціальні мережі або інші засоби комунікації. Така структура сприяє забезпеченню оперативної взаємодії з користувачами, партнерами та іншими зацікавленими сторонами.

Сторінка також виконує представницьку роль, підкреслюючи відкритість і доступність проєкту, що відповідає принципам прозорості, довіри та ефективної взаємодії в освітньому просторі.

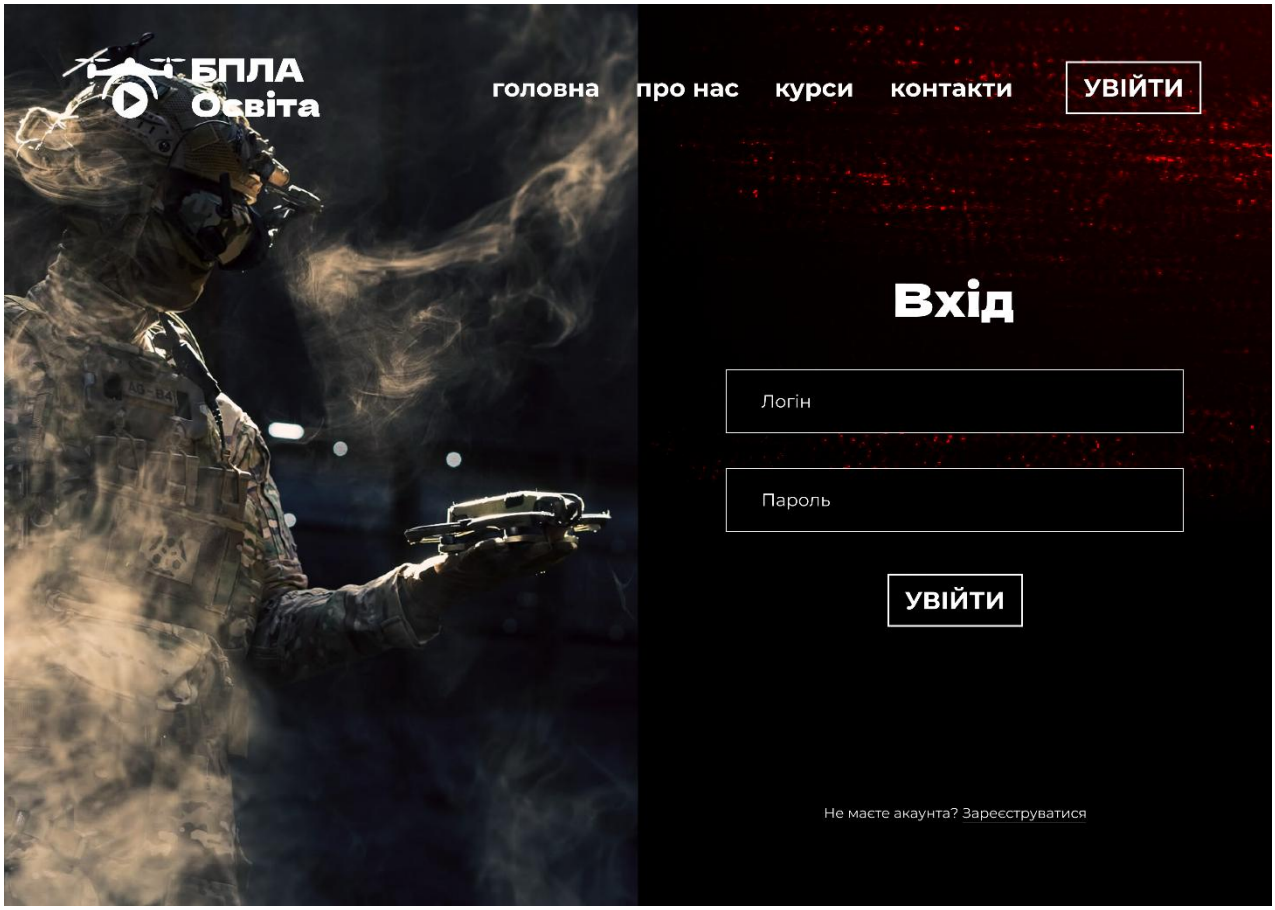


Рисунок 4.4. – Сторінка «Вхід»

Сторінка «Вхід» виконує функцію авторизації користувачів у системі веб-застосунку та забезпечує доступ до персоналізованого освітнього контенту відповідно до ролі зареєстрованого учасника (студента або викладача).

Інтерфейс сторінки побудовано з урахуванням принципів зручності та безпеки: реалізовано поля для введення логіна і пароля, а також передбачено можливість переходу до реєстрації для нових користувачів. Візуальний супровід підкреслює технічну тематику ресурсу та формує асоціації з високотехнологічним навчанням. Авторизація є необхідним етапом для

забезпечення персоніфікованої взаємодії з платформою, захисту даних та управління доступом до навчальних матеріалів.

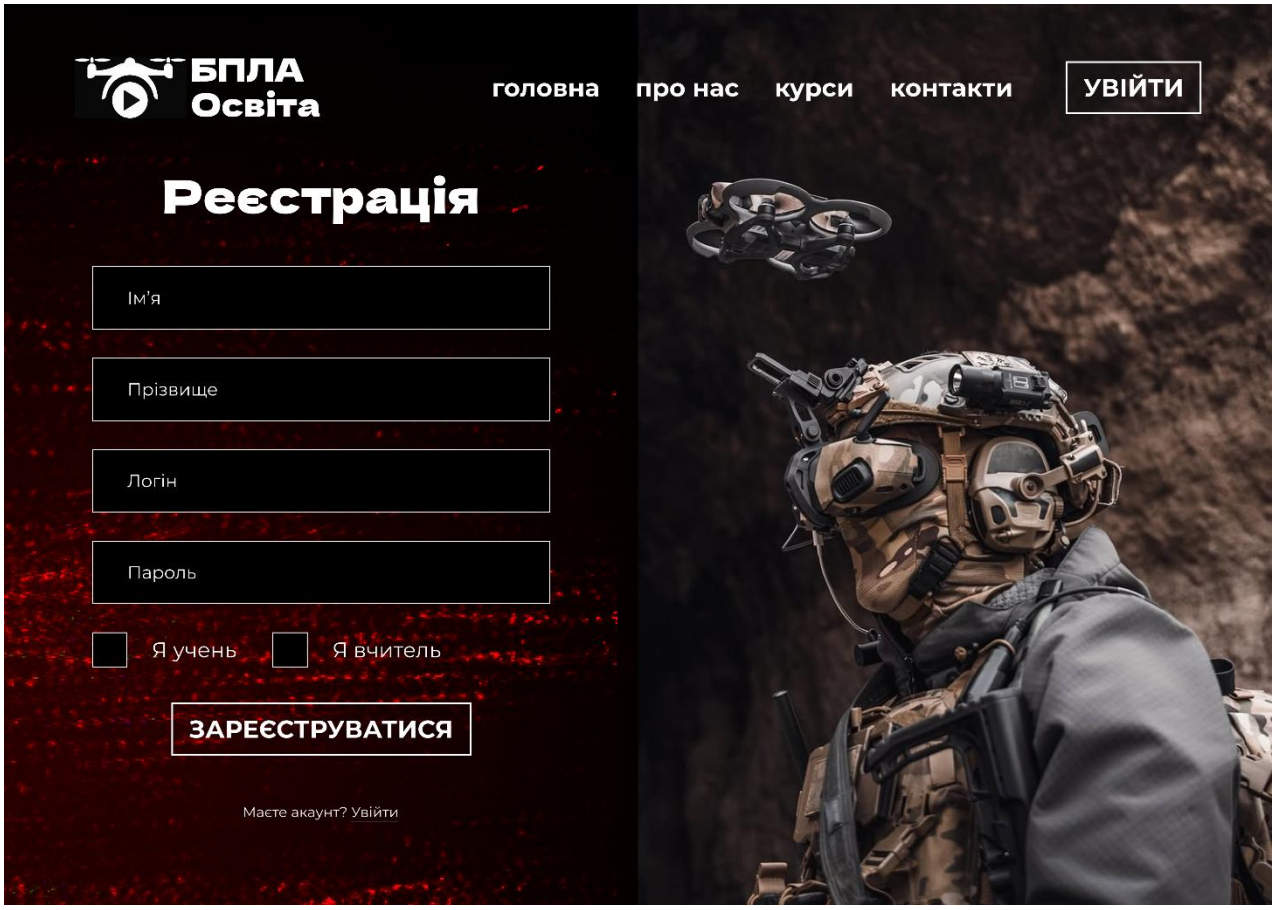


Рисунок 4.4. – Сторінка «Реєстрація»

Сторінка «Реєстрація» забезпечує можливість створення облікового запису для нових користувачів освітнього веб-застосунку. Форма реєстрації містить обов'язкові поля для введення особистих даних (ім'я, прізвище), облікової інформації (логін, пароль), а також передбачає вибір ролі користувача (учень або вчитель), що дозволяє диференціювати доступ до функціональних можливостей платформи.

Інтерфейс побудовано відповідно до принципів інтуїтивної зрозумілості та зручності взаємодії, що сприяє залученню користувачів та оптимізації процесу реєстрації.

Сторінка відіграє ключову роль у формуванні бази користувачів платформи, забезпечуючи доступ до персоналізованого освітнього контенту та підтримуючи безперебійну комунікацію в межах навчального середовища.

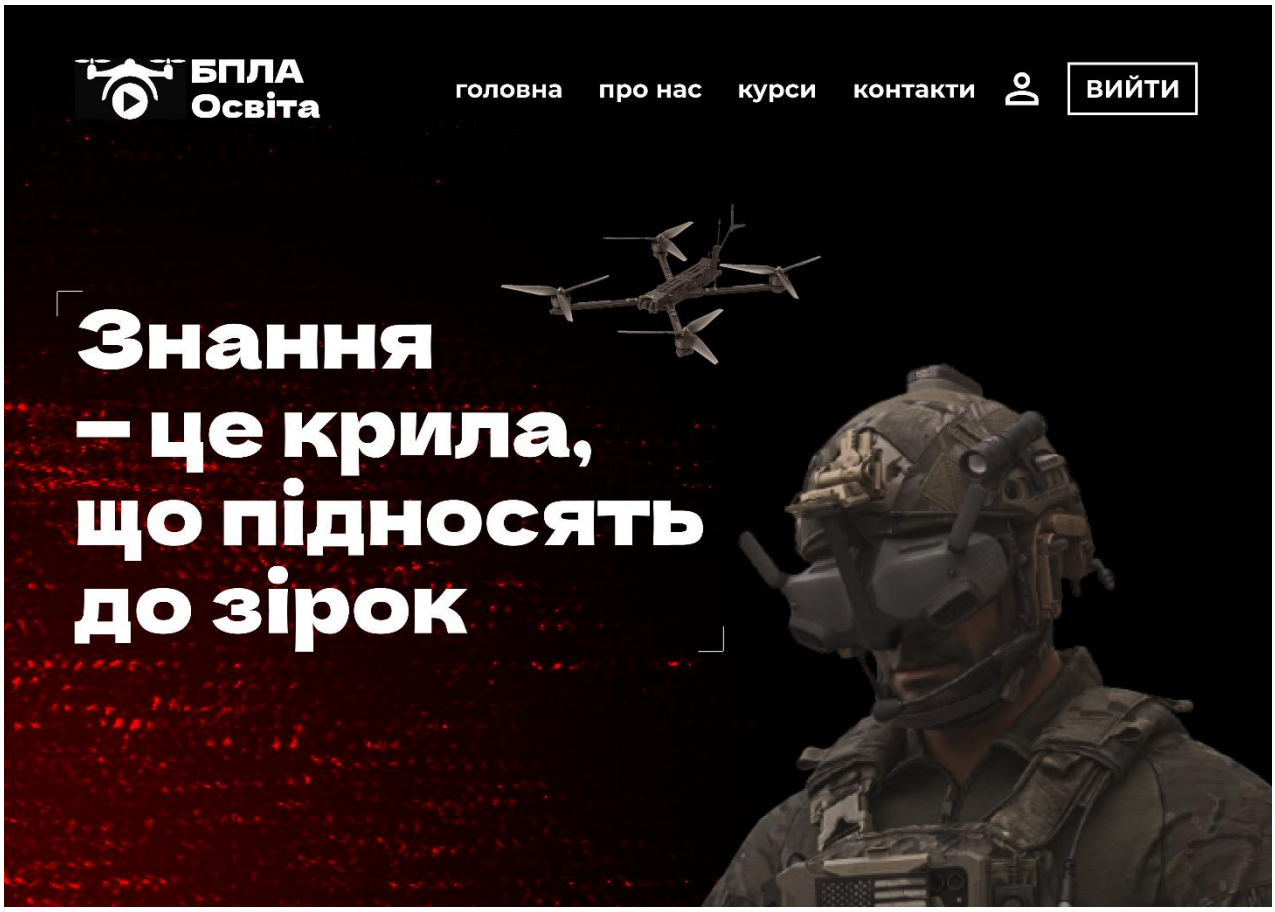


Рисунок 4.5. – Головна сторінка користувача

Після входу або реєстрації інтерфейс сторінки доповнюється іконкою користувача, що вказує на активний обліковий запис і слугує зручним навігаційним елементом для переходу до персональних даних.

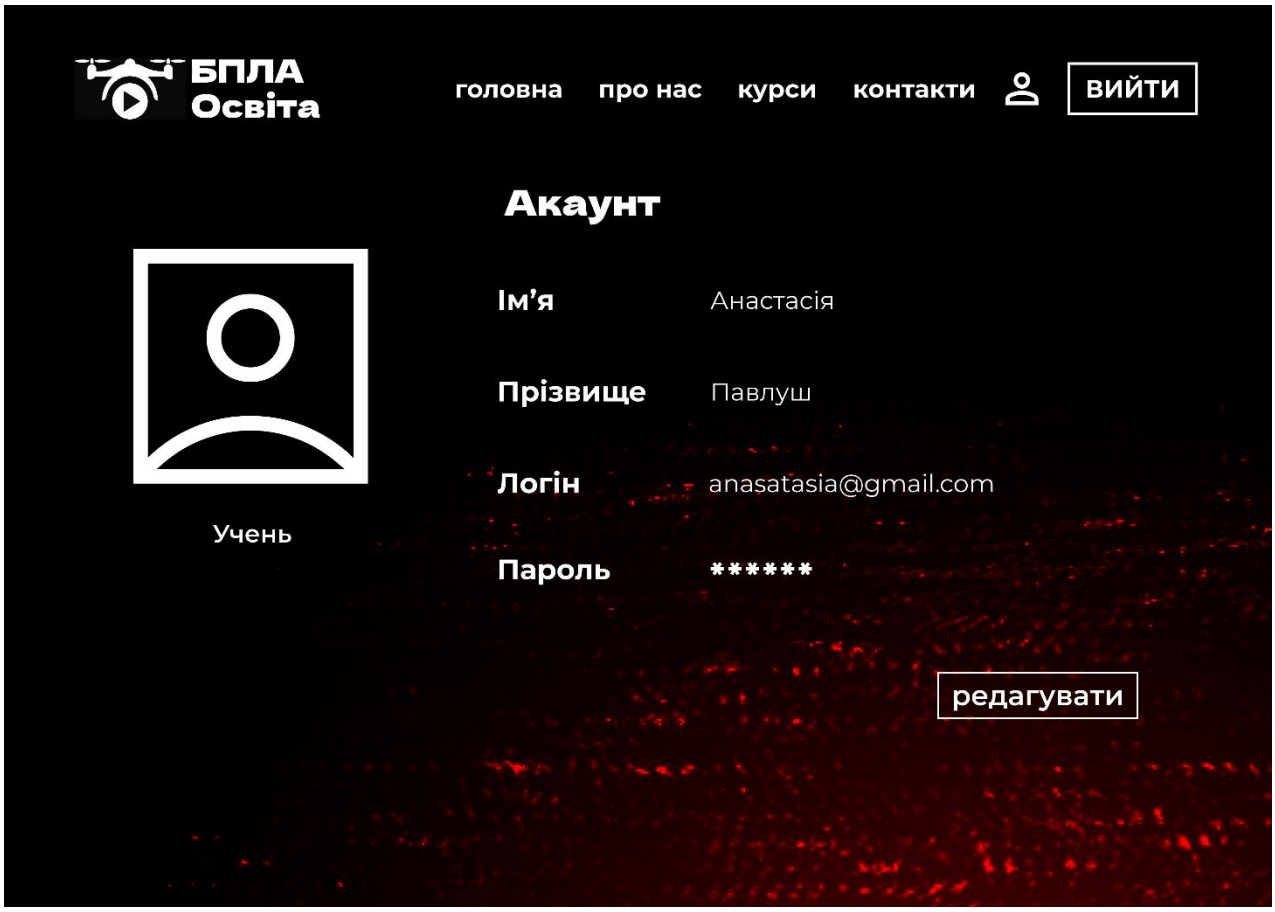


Рисунок 4.6. – Сторінка «Акаунт»

Сторінка «Акаунт» виконує функцію відображення та редагування особистих даних зареєстрованого користувача освітнього веб-застосунку. Вона містить інформацію, що була вказана під час реєстрації, зокрема ім’я, прізвище, логін (електронну адресу), роль на платформі (учень або вчитель), а також замаскований пароль.

Інтерфейс передбачає можливість редагування персональних даних через відповідну функцію, що сприяє підтримці актуальності облікової інформації. Наявність іконки користувача в навігаційній панелі забезпечує зручний перехід до сторінки акаунта та підкреслює персоніфікований характер взаємодії з платформою. Сторінка слугує інструментом для самостійного керування обліковими записами, забезпечуючи користувачам контроль над своїми персональними даними в межах освітнього середовища.

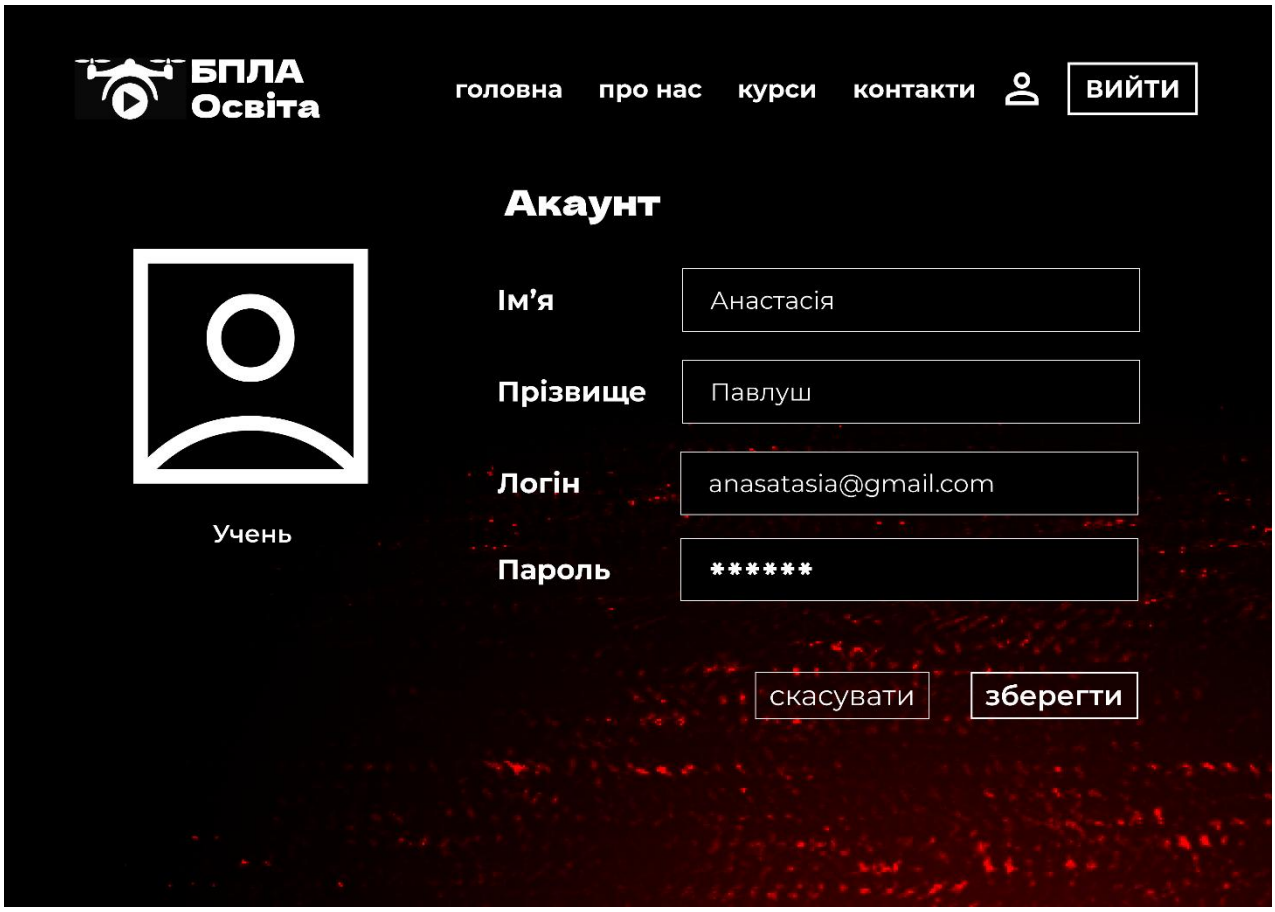


Рисунок 4.7. – Редагування сторінки «Акаунт»

Сторінка редагування акаунта забезпечує користувачам можливість оновлення особистих даних. Вона є продовженням функціоналу облікового запису та сприяє підтримці актуальності інформації в межах персоналізованого освітнього середовища. Інтерфейс сторінки передбачає редагування ключових параметрів облікового запису: імені, прізвища, логіна (електронної пошти) та пароля. Зміни супроводжуються варіантами збереження або скасування введених даних, що підвищує зручність користування та забезпечує контроль над змінами. Сторінка реалізована відповідно до принципів конфіденційності, інтуїтивної доступності й технічної безпеки, що є важливими складовими цифрової взаємодії в сучасному освітньому просторі.

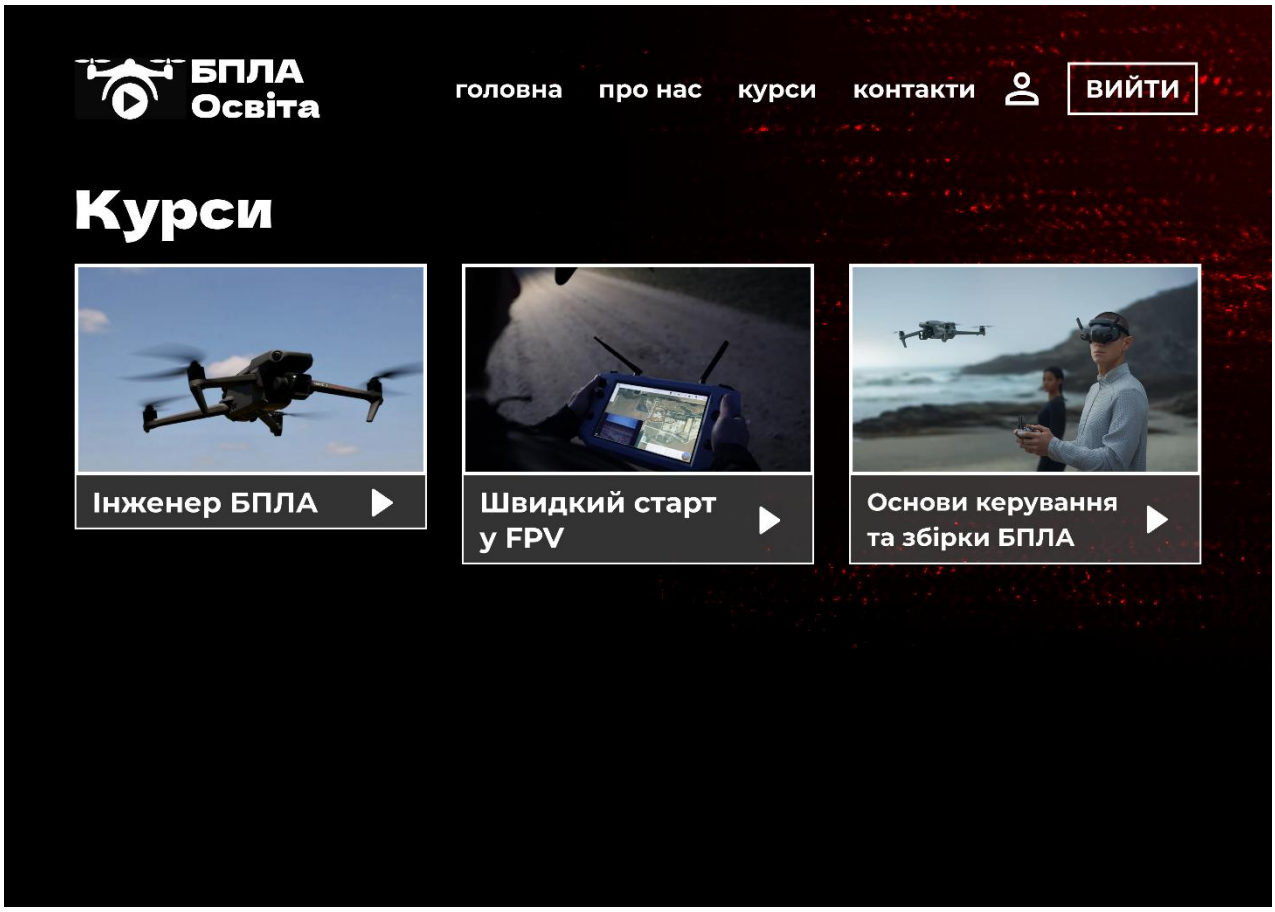


Рисунок 4.8. – Сторінка «Курси»

Сторінка «Курси» є структурованим навчальним простором, що надає зареєстрованим користувачам доступ до освітніх програм, спрямованих на підготовку операторів безпілотних літальних апаратів різних рівнів.

Інтерфейс сторінки представлено у форматі інтерактивного каталогу, що містить візуально виокремлені блоки з короткими назвами доступних курсів. Кожен блок виконує функцію навігаційного елемента, що веде до відповідного навчального модуля. Така подача сприяє швидкій орієнтації користувача в освітньому контенті та вибору курсу відповідно до власного рівня підготовки чи професійних інтересів.

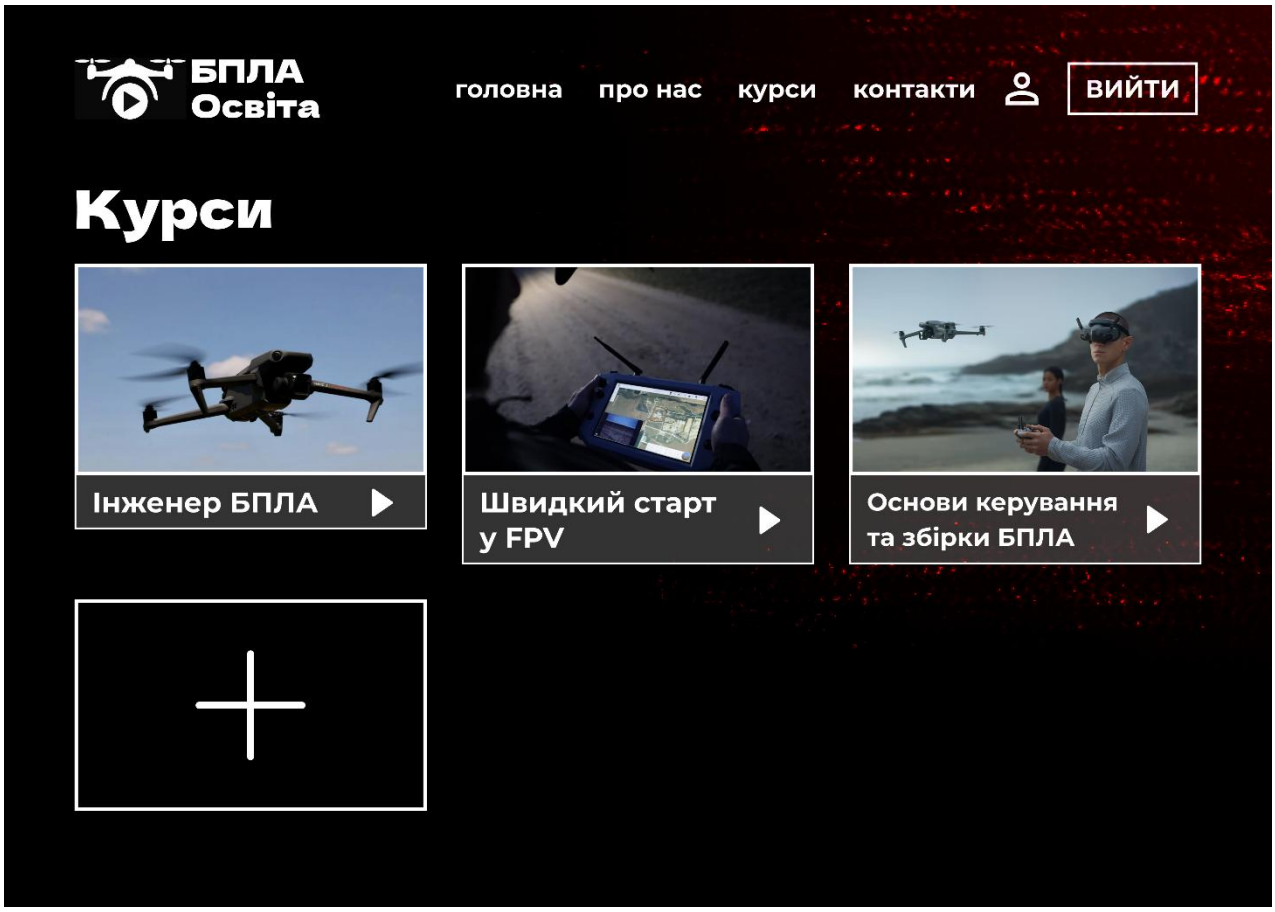


Рисунок 4.9 – Сторінка «Курси (режим викладача)»

Сторінка «Курси» в режимі викладача виконує функцію управління навчальним контентом у межах освітнього веб-застосунку. Вона забезпечує доступ до переліку вже створених курсів і розширюється інструментами для додавання нових навчальних програм.

Інтерфейс сторінки включає інтерактивний візуальний каталог курсів, аналогічний учнівському режиму, а також додатковий елемент — кнопку створення нового курсу, позначену символом «плюс». Така структура надає викладачеві можливість швидко орієнтуватися в існуючому контенті й оперативно ініціювати розробку нових освітніх модулів. Сторінка реалізує функціональну підтримку педагогічної діяльності в цифровому середовищі платформи, сприяючи розширенню навчальної пропозиції та адаптації освітнього процесу до потреб сучасної підготовки операторів БПЛА.

Рисунок 4.10 – Сторінка «Про курс»

Сторінка виконує ознайомчу функцію та надає користувачам структуровану інформацію про зміст окремого навчального курсу, його цілі, тематичне наповнення та цільову аудиторію.

Основний текстовий блок містить опис курсу з акцентом на рівень підготовки, необхідні технічні компетентності та практичні вміння, які здобуваються в процесі навчання. У поясненні також вказано, для яких категорій слухачів курс є найбільш релевантним — зокрема для новачків, операторів FPV-дронів, технічних фахівців і волонтерів.

Нижче розміщено послідовний перелік навчальних уроків, кожен з яких має заголовок, що окреслює його зміст. Такий формат забезпечує логічну структуру подання матеріалу, полегшує орієнтацію у змісті та сприяє послідовному опануванню навчального курсу. Сторінка виконує важливу роль у процесі прийняття рішення про вибір курсу, сприяє формуванню обґрунтованих освітніх очікувань і підтримує цілісне уявлення про навчальну програму.

Рисунок 4.10 – Сторінка «Урок»

Сторінка «Урок» є функціональним елементом навчального модуля, що надає доступ до змісту конкретного заняття в межах обраного курсу. Вона поєднує текстовий супровід і відеоматеріал, спрямовані на формування цілісного розуміння теми уроку.

Текстова частина містить вступне пояснення до теми, визначає основні поняття, що розглядаються, а також окреслює змістовні акценти уроку. Особливу увагу приділено переліку технічних компонентів, логіці їхньої взаємодії, а також прикладному аспекту використання знань у реальних умовах.

Інтегроване відео виконує роль основного навчального ресурсу, що візуалізує матеріал, демонструє технічні елементи дронів, порядок їхньої збірки, принципи роботи систем та інші ключові аспекти.

ВИСНОВОК ДО РОЗДІЛУ 4

У четвертому розділі представлений розроблений веб-застосунок «БПЛА Освіта», який демонструє високий рівень організації освітнього процесу та відповідає актуальним викликам часу, пов’язаним із підготовкою операторів безпілотних літальних апаратів. Структура платформи охоплює повний цикл взаємодії користувача з навчальним середовищем — від реєстрації та авторизації до перегляду курсів, проходження уроків, керування особистим профілем і створення нового контенту (викладачами). Кожна сторінка виконує чітко визначену функцію та має логічно побудовану структуру, що сприяє зручності користування.

Інтерфейс застосунку вирізняється інтуїтивною навігацією, візуальною послідовністю й адаптацією до потреб різних ролей користувачів — студентів і викладачів. Відеоуроки, текстові пояснення, розділення на модулі та персоналізовані сторінки забезпечують ефективне засвоєння матеріалу, мотивацію до навчання та підтримку індивідуального освітнього маршруту.

Узагальнюючи, платформа «БПЛА Освіта» є потужним освітнім інструментом, що поєднує технічну інноваційність із педагогічною структурованістю. Вона не лише підвищує якість підготовки операторів БПЛА, але й сприяє зміцненню обороноздатності країни через доступ до актуальних знань та навичок.

Подальший розвиток веб-застосунку може включати впровадження аналітичних інструментів для відстеження прогресу користувачів, розширення функцій зворотного зв’язку, підтримку мобільних платформ та інтеграцію з іншими освітніми системами. Це дозволить зробити платформу ще більш гнучкою, масштабованою та ефективною в умовах динамічного інформаційного середовища та зростаючих потреб оборонної галузі.

ВИСНОВКИ

Під час виконання дипломної роботи «Розробка веб-застосунку для дистанційного навчання операторів БПЛА» було послідовно вирішено чотири ключові завдання, що забезпечили створення функціональної, масштабованої та безпечної платформи.

У першому розділі проведено глибокий аналіз завдань і предметної області дистанційного навчання операторів FPV-дронів. Визначено прогалини існуючих рішень — розпорошеність матеріалів, висока вартість очних курсів, недостатня локалізація. Обґрунтовано, що веб-формат українською мовою забезпечить гнучкість, доступність і економічність навчання, заклавши методичну основу для технічних вимог системи.

У другому розділі детально описано набір мінімальних функцій — публічні сторінки, автентифікація, перегляд курсів, редагування профілю, адміністративна консоль. Пояснено архітектурний вибір моноліту для проекту із подальшою можливістю еволюції в мікросервіси. Вибір React забезпечує компонентність і швидкий UI-розвиток, FastAPI з Swagger-документацією — безпеку та асинхронну продуктивність бекенду.

У третьому розділі розроблено архітектуру односторінкового застосунку на React із CSS Modules і react-router-dom, а також серверну логіку на FastAPI із SQLModel і PostgreSQL. Запроваджено JWT-автентифікацію, чітке розділення CRUD-шару, роутерів і залежностей. Структура коду залишилася прозорою та тестованою, з можливістю подальшого розширення — WebSocket, кешування, мікросервіси.

Загалом, виконана робота забезпечила створення потужного та адаптивного інструмента для масового дистанційного навчання операторів БПЛА. Запропонований стек технологій і архітектурні рішення дають змогу

подальшому розширенню функціоналу — впровадження аналітики прогресу, мобільної підтримки, розширених зворотних зв’язків та інтеграції з іншими освітніми системами. Це закладає міцну основу для розвитку платформи в умовах динамічних вимог оборонної та цивільної сфер.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Henri Eisenbeiss. A mini unmanned aerial vehicle (UAV): System overview and image acquisition [Електронний ресурс] - Режим доступу до ресурсу: https://www.academia.edu/download/38864677/UAV_White_Paper.pdf
2. Matyas Palik, Máté Nagy. Brief history of UAV development [Електронний ресурс] - Режим доступу до ресурсу: https://folyoirat.ludovika.hu/index.php/reptudkoz/article/download/246/42
3. Dante Tezza, Denis Laesker, Marvin Andujar. The Learning Experience of Becoming a FPV Drone Pilot [Електронний ресурс] - Режим доступу до ресурсу: https://dl.acm.org/doi/abs/10.1145/3434074.3447167
4. Ahmed Fawzi Otoom,Ghadeer AL Kateb,Maen Hammad ,Rateb J. Sweis, Haneen Hijazi. Success Factors Importance Based on Software Project Organization Structure [Електронний ресурс] - Режим доступу до ресурсу: https://www.mdpi.com/2078-2489/10/12/391
5. Vamsi Krishna Thatikonda, Hemavantha Rajesh Varma Mudunuri. Microservices vs. Monoliths: Choosing the Right Architecture for Your Project [Електронний ресурс] - Режим доступу до ресурсу: https://journalspub.com/wp-content/uploads/2024/10/31-38-Microservices-vs-Monoliths-Choosing-the-Right-Architecture_Vamsi_Revised-1.pdf
6. Miika Kalske. Transforming monolithic architecture towards microservice architecture [Електронний ресурс] - Режим доступу до ресурсу: https://helda.helsinki.fi/server/api/core/bitstreams/e88caa20-70ea-496b-8125-5d6488ebc343/content
7. Visual Studio Code Documentation [Електронний ресурс] - Режим доступу до ресурсу: https://code.visualstudio.com/docs

8. Figma Learn. Figma Documentation [Електронний ресурс] - Режим доступу до ресурсу: https://help.figma.com/hc/en-us/categories/360002051613-Get-started
9. Quick Start. React Documentation [Електронний ресурс] - Режим доступу до ресурсу: https://react.dev/learn
10. Jelica Cincović, Marija Punt. Comparison: Angular vs. React vs. Vue. Which framework is the best choice? [Електронний ресурс] - Режим доступу до ресурсу: https://www.eventiotic.com/eventiotic/files/Papers/URL/50173409-699e-4b17-8edb-9764ecc53160.pdf
11. Arjun Naik. The Front-End Dilemma: How to Choose the Perfect TECHNOLOGY for Your Application. [Електронний ресурс] - Режим доступу до ресурсу: https://www.neliti.com/publications/589873/the-front-end-dilemma-how-to-choose-the-perfect-technology-for-your-application
12. Learn FastAPI. FastAPI Documentation [Електронний ресурс] - Режим доступу до ресурсу: https://fastapi.tiangolo.com/learn/
13. Солохін, А. Є. Дослідження ефективності використання фреймворків веб-розробки на продуктивність веб-додатку: порівняльний аналіз Django, Flask, та FastAPI [Електронний ресурс] - Режим доступу до ресурсу: https://openarchive.nure.ua/entities/publication/d55b29de-4cf2-4ba2-9f3c-64b598de0912
14. PostgreSQL Documentation [Електронний ресурс] - Режим доступ до ресурсу: https://www.postgresql.org/docs/
15. Antonios Makris, Konstantinos Tserpes, Dimosthenis Anagnostopoulos. MongoDB Vs PostgreSQL: A comparative study on performance aspects [Електронний ресурс] - Режим доступу до ресурсу: https://link.springer.com/article/10.1007/s10707-020-00407-w#author-information

ДОДАТОК 1

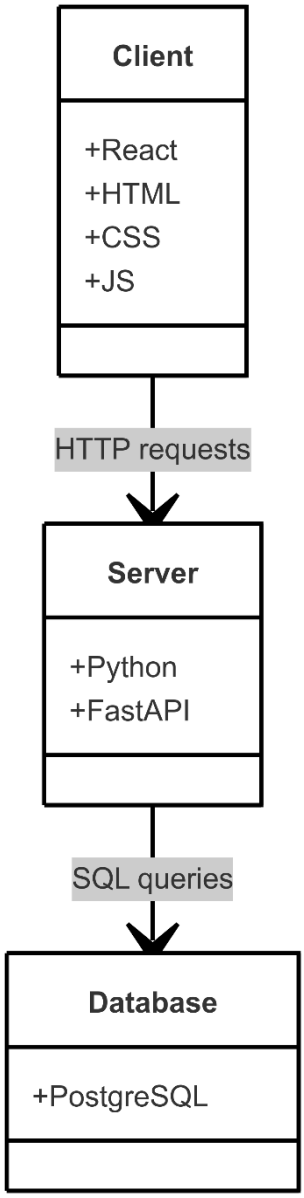
Веб-платформа для навчання операторів БПЛА та збірки FPV дронів

Структурна схема системи

ІАЛЦ.045440.004 Д1

Аркушів 1

Київ 2025



					ІАЛЦ.045440.004 Д1		
		№ докум.	Підпис	Дата			
Розробив		Павлуш А. Ю.			**Веб-платформа для навчання операторів БПЛА та збірки FPV дронів** Структурна схема системи	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Череватенко О.В					
						НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-16	
Н. Контр.		Міщенко Л. Д.					
Затвердив							

ДОДАТОК 2

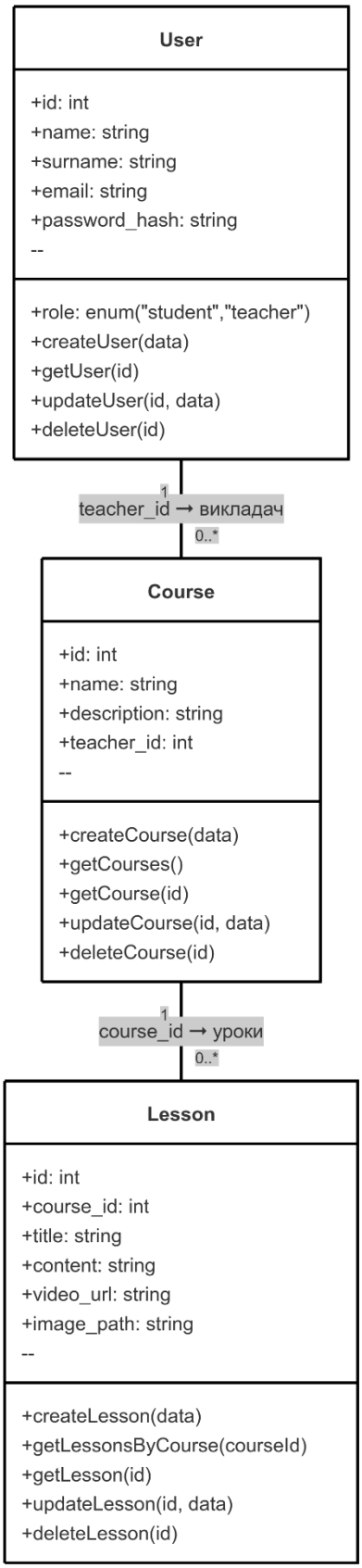
Веб-платформа для навчання операторів БПЛА та збірки FPV дронів

Функціональна схема

ІАЛЦ.045440.005 Д2

Аркушів 1

Київ 2025



					ІАЛЦ.045440.005 Д2	
		№ докум.	Підпис	Дата		
Розробив		Павлуш А. Ю.			**Веб-платформа для навчання операторів БПЛА та збірки FPV дронів** Функціональна схема	Літ. / Аркуш: 1 / Аркушів: 1
Перевірив		Череватенко О.В				
						НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-16
Н. Контр.		Міщенко Л. Д.				
Затвердив						

ДОДАТОК 3

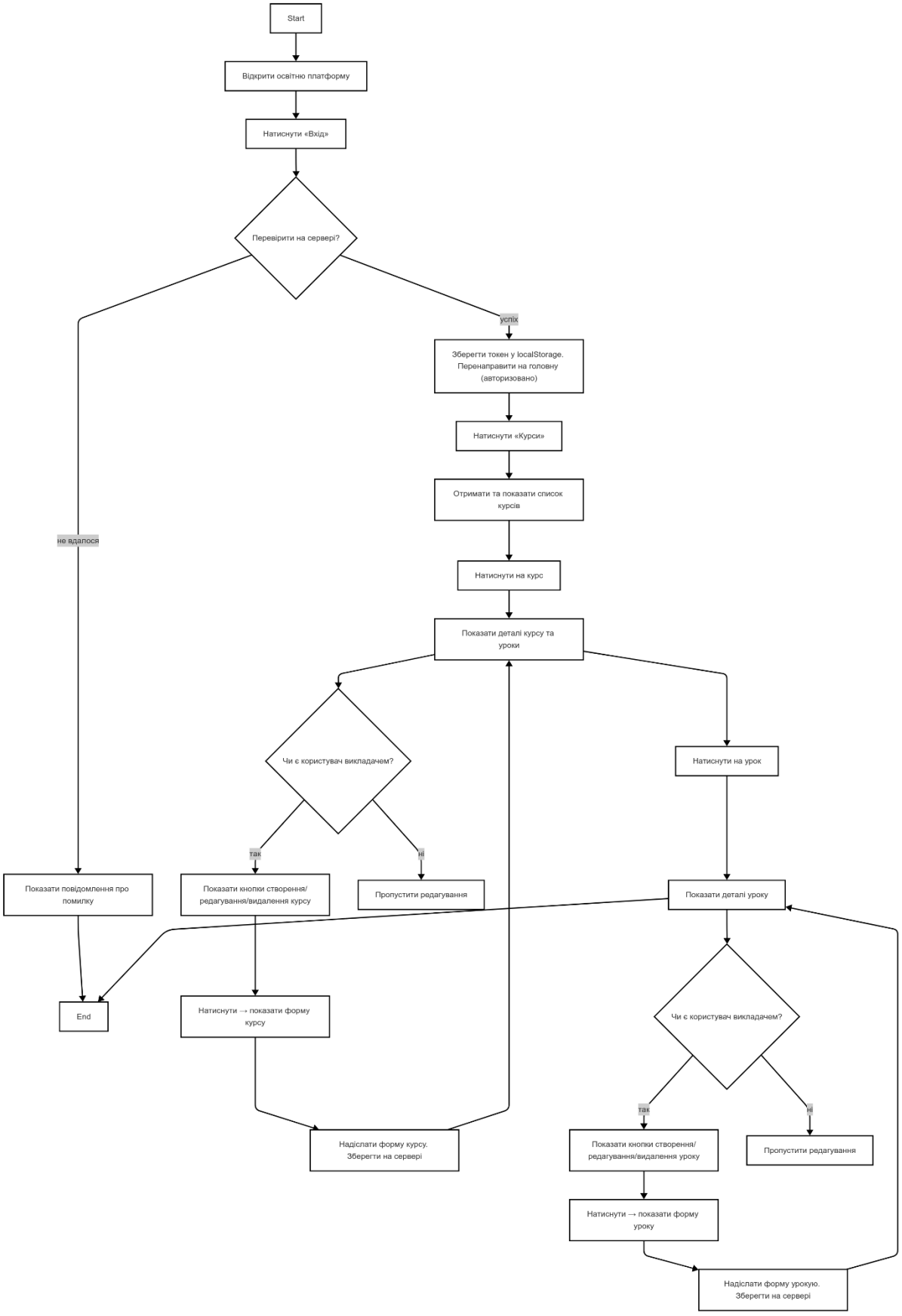
Веб-платформа для навчання операторів БПЛА та збірки FPV дронів

Алгоритм дій програмного забезпечення

ІАЛЦ.045440.006 Д3

Аркушів 1

Київ 2025



					ІАЛЦ.045440.006 Д3		
		№ докум.	Підпис	Дата			
Розробив		Павлуш А. Ю.			**Веб-платформа для навчання операторів БПЛА та збірки FPV дронів** Алгоритм дій програмного забезпечення	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Череватенко О.В					
Н. Контр.		Міщенко Л. Д.				НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-16	
Затвердив							

ДОДАТОК 4

Веб-платформа для навчання операторів БПЛА та збірки FPV дронів

Текст програмного коду

ІАЛЦ.045440.007 Д4

Аркушів 61

Київ 2025

```python
# app/crud/course.py
from sqlmodel import Session, select
from ..models.course import Course


def get_courses(db: Session) -> list[Course]:
    """Retrieve all courses from the database.

    Args:
        db: SQLModel session for interacting with the database.

    Returns:
        A list of all Course objects.
    """
    return db.exec(select(Course)).all()


def get_course(db: Session, course_id: int) -> Course | None:
    """Retrieve a course by its ID.

    Args:
        db: SQLModel session for interacting with the database.
        course_id: The ID of the course to retrieve.

    Returns:
        The Course object if found, otherwise None.
    """
    return db.get(Course, course_id)


def create_course(
    db: Session,
    name: str,
    description: str,
    teacher_id: int,
) -> Course:
    """Create a new course and save it to the database.

    Args:
        db: SQLModel session for interacting with the database.
        name: Name of the new course.
        description: Description of the new course.
        teacher_id: ID of the teacher creating the course.

    Returns:
        The newly created Course object.
    """
    course = Course(name=name, description=description, teacher_id=teacher_id)
    db.add(course)
    db.commit()
    db.refresh(course)
    return course
```

```
def update_course(
    db: Session,
    course_id: int,
    name: str | None = None,
    description: str | None = None,
) -> Course | None:
    """Update an existing course's name and/or description.

    Args:
        db: SQLModel session for interacting with the database.
        course_id: ID of the course to update.
        name: New name for the course (optional).
        description: New description for the course (optional).

    Returns:
        The updated Course object, or None if the course was not found.
    """
    course = db.get(Course, course_id)
    if not course:
        return None
    if name is not None:
        course.name = name
    if description is not None:
        course.description = description
    db.add(course)
    db.commit()
    db.refresh(course)
    return course


def delete_course(db: Session, course_id: int) -> None:
    """Delete a course by its ID.

    Args:
        db: SQLModel session for interacting with the database.
        course_id: ID of the course to delete.

    Returns:
        None
    """
    course = db.get(Course, course_id)
    if not course:
        return
```

```
    db.delete(course)
    db.commit()

# app/crud/lesson.py
from sqlmodel import Session, select
from ..models.lesson import Lesson

def get_lessons_by_course(db: Session, course_id: int) -> list[Lesson]:
    return db.exec(select(Lesson).where(Lesson.course_id == course_id)).all()

def get_lesson(db: Session, lesson_id: int) -> Lesson | None:
    return db.get(Lesson, lesson_id)

def create_lesson(
    db: Session,
    course_id: int,
    title: str,
    content: str,
    video_url: str,
    image_path: str,
) -> Lesson:
    lesson = Lesson(
        course_id=course_id,
        title=title,
        content=content,
        video_url=video_url,
        image_path=image_path,
    )
    db.add(lesson)
    db.commit()
    db.refresh(lesson)
    return lesson

def update_lesson(
    db: Session,
    lesson_id: int,
    title: str | None = None,
    content: str | None = None,
    video_url: str | None = None,
    image_path: str | None = None,
) -> Lesson | None:
    lesson = db.get(Lesson, lesson_id)
    if not lesson:
```

```
            return None
        if title is not None:
            lesson.title = title
        if content is not None:
            lesson.content = content
        if video_url is not None:
            lesson.video_url = video_url
        if image_path is not None:
            lesson.image_path = image_path
        db.add(lesson)
        db.commit()
        db.refresh(lesson)
        return lesson

    def delete_lesson(db: Session, lesson_id: int) -> None:
        lesson = db.get(Lesson, lesson_id)
        if not lesson:
            return
        db.delete(lesson)
        db.commit()

    # app/crud/lesson.py
    from sqlmodel import Session, select
    from ..models.lesson import Lesson


    def get_lessons_by_course(db: Session, course_id: int) -> list[Lesson]:
        """Retrieve all lessons that belong to a specific course.

        Args:
            db: SQLModel session for interacting with the database.
            course_id: ID of the course whose lessons are to be fetched.

        Returns:
            A list of Lesson objects that belong to the given course.
        """
        return db.exec(select(Lesson).where(Lesson.course_id == course_id)).all()


    def get_lesson(db: Session, lesson_id: int) -> Lesson | None:
        """Retrieve a lesson by its ID.

        Args:
```

```
        db: SQLModel session for interacting with the database.
        lesson_id: The ID of the lesson to retrieve.

    Returns:
        The Lesson object if found, otherwise None.
    """
    return db.get(Lesson, lesson_id)

# app/deps/auth.py
import logging
from fastapi import Depends, HTTPException, status
from fastapi.security import OAuth2PasswordBearer
from jose import JWTError
from sqlmodel import select, Session

from .jwt import decode_token
from ..deps.session import get_session
from ..models.user import User

logger = logging.getLogger("uvicorn.error")
oauth2_scheme = OAuth2PasswordBearer(tokenUrl="/auth/login")


def get_current_user(
    token: str = Depends(oauth2_scheme),
    session: Session = Depends(get_session),
) -> User:
    """Get the current authenticated user based on the JWT token.

    Args:
        token: JWT token from the OAuth2 scheme.
        session: SQLModel database session.

    Raises:
        HTTPException: If the token is invalid or the user cannot be found.

    Returns:
        The authenticated User object.
    """
    credentials_exc = HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Could not validate credentials",
    )
```

```
    logger.debug("raw token   → %s", token)

    try:
        payload = decode_token(token)
        logger.debug("decoded    → %s", payload)
        sub = payload.get("sub")
        if sub is None:
            logger.debug("no sub claim")
            raise credentials_exc

        if not isinstance(sub, str):
            logger.debug("sub claim is not a string: %s", sub)
            raise credentials_exc

        try:
            user_id = int(sub)
        except ValueError:
            logger.debug("invalid sub claim, not an integer: %s", sub)
            raise credentials_exc

    except JWTError as e:
        logger.debug("JWTError   → %s", e)
        raise credentials_exc

    user = session.exec(select(User).where(User.id == user_id)).one_or_none()
    logger.debug("DB lookup   → %s", user)
    if not user:
        logger.debug("no user for id=%s", user_id)
        raise credentials_exc

    return user


def get_current_teacher(
    user: User = Depends(get_current_user),
) -> User:
    """Ensure that the current user has a teacher role.

    Args:
        user: The currently authenticated user.

    Raises:
```

```
            HTTPException: If the user is not a teacher.

        Returns:
            The User object, if the user is a teacher.
        """
        if user.role != user.role.__class__.teacher:
            raise HTTPException(
                status_code=status.HTTP_403_FORBIDDEN,
                detail="Teacher access only",
            )
        return user

    # app/deps/jwt.py
    import os
    from datetime import datetime, timedelta, timezone
    from jose import jwt

    SECRET_KEY = os.getenv("SECRET_KEY", "change-this")
    ALGORITHM = "HS256"
    ACCESS_TOKEN_EXPIRE_MINUTES = 60


    def create_access_token(data: dict) -> str:
        """Create a JWT access token.

        Args:
            data: Dictionary of data to encode in the token. Must include 'sub' as user
ID.

        Returns:
            A signed JWT token as a string.
        """
        to_encode = data.copy()
        if "sub" in to_encode:
            to_encode["sub"] = str(to_encode["sub"])
        expire                =                datetime.now(timezone.utc)                +
timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES)
        to_encode.update({"exp": expire})
        return jwt.encode(to_encode, SECRET_KEY, algorithm=ALGORITHM)


    def decode_token(token: str) -> dict:
        """Decode a JWT token.
```

```
    Args:
        token: A signed JWT token string.

    Returns:
        A dictionary containing the decoded token payload.
    """
    return jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])


# app/deps/security.py
from collections.abc import Generator
from sqlmodel import Session
from ..db.database import engine


def get_session() -> Generator[Session, None, None]:
    """Dependency to provide a SQLModel session for FastAPI.

    Yields:
        A database session that is automatically closed after use.
    """
    with Session(engine) as session:
        yield session

# app/deps/session.py
from collections.abc import Generator
from sqlmodel import Session
from ..db.database import engine


def get_session() -> Generator[Session, None, None]:
    """Dependency to provide a SQLModel session for FastAPI.

    Yields:
        A database session that is automatically closed after use.
    """
    with Session(engine) as session:
        yield session
```

```
from typing import Optional
from sqlmodel import SQLModel, Field, Relationship
from sqlalchemy.orm import relationship

# Forward declaration to avoid circular import issues
from typing import TYPE_CHECKING
if TYPE_CHECKING:
    from .lesson import Lesson


class CourseBase(SQLModel):
    """Shared fields for all course schemas."""
    name: str
    description: str


class Course(CourseBase, table=True):
    """Course model for the database.

    Attributes:
        id: Unique identifier for the course.
        teacher_id: ID of the teacher who owns the course.
        lessons: List of associated lessons (one-to-many relationship).
    """
    id: Optional[int] = Field(default=None, primary_key=True)
    teacher_id: int = Field(foreign_key="user.id")
    lessons: list["Lesson"] = Relationship(
        back_populates="course",
        sa_relationship_kwargs={"cascade": "all, delete-orphan"}
    )


class CourseCreate(CourseBase):
    """Data required to create a new course (input schema)."""
    pass


class CourseRead(CourseBase):
    """Data returned when reading course information (output schema).

    Attributes:
        id: Unique identifier of the course.
        teacher_id: ID of the course's teacher.
```

```
    """
    id: int
    teacher_id: int


class CourseUpdate(SQLModel):
    """Fields that can be updated for a course (partial update).

    Attributes:
        name: New name for the course.
        description: New description for the course.
    """
    name: Optional[str] = None
    description: Optional[str] = None

from typing import Optional
from sqlmodel import SQLModel, Field, Relationship
from sqlalchemy import Column, ForeignKey

# Forward declaration to avoid circular import issues
from typing import TYPE_CHECKING
if TYPE_CHECKING:
    from .course import Course


class LessonBase(SQLModel):
    """Shared fields for all lesson schemas."""
    title: str
    content: str
    video_url: str


class Lesson(LessonBase, table=True):
    """Lesson model for the database.

    Attributes:
        id: Unique identifier of the lesson.
        course_id: Foreign key referencing the associated course.
        image_path: Path to the image related to the lesson.
        course: The Course object this lesson belongs to.
    """
    id: Optional[int] = Field(default=None, primary_key=True)
    course_id: int = Field(
```

```
            sa_column=Column(
                ForeignKey("course.id", ondelete="CASCADE"),
                nullable=False
            )
        )
        image_path: str
        course: "Course" = Relationship(back_populates="lessons")


    class LessonCreate(SQLModel):
        """Data required to create a new lesson (input schema)."""
        title: str
        content: str
        video_url: str


    class LessonRead(LessonBase):
        """Data returned when reading lesson information (output schema).

        Attributes:
            id: Unique identifier of the lesson.
            course_id: ID of the course the lesson belongs to.
            image_path: Path to the image associated with the lesson.
        """
        id: int
        course_id: int
        image_path: str


    class LessonUpdate(SQLModel):
        """Fields that can be updated for a lesson (partial update).

        Attributes:
            title: New title for the lesson.
            content: New content for the lesson.
            video_url: New video URL.
            image_path: New image path.
        """
        title: Optional[str] = None
        content: Optional[str] = None
        video_url: Optional[str] = None
        image_path: Optional[str] = None
```

```
# app/models/user.py
from sqlmodel import SQLModel, Field, Column
from pydantic import EmailStr
from enum import Enum
from sqlalchemy import Enum as SAEnum


class UserRole(str, Enum):
    """Enumeration for user roles."""
    student = "student"
    teacher = "teacher"


class UserBase(SQLModel):
    """Shared fields for all user schemas."""
    name: str
    surname: str
    email: EmailStr
    role: UserRole


class User(UserBase, table=True):
    """User model for the database.

    Attributes:
        id: Unique identifier for the user.
        password_hash: Hashed user password.
        role: Role of the user (student or teacher).
    """
    id: int = Field(default=None, primary_key=True)
    password_hash: str
    role: UserRole = Field(
        sa_column=Column(
            SAEnum(UserRole, name="userrole"),
            default=UserRole.student,
            nullable=False
        )
    )


class UserCreate(UserBase):
    """Data required to register a new user (input schema).
```

```
    Attributes:
        password: Plain text password (to be hashed).
    """
    password: str


class UserRead(UserBase):
    """Data returned when reading user information (output schema).

    Attributes:
        id: Unique identifier of the user.
    """
    id: int


# app/routers/auth.py
from fastapi import APIRouter, Depends, HTTPException
from fastapi.security import OAuth2PasswordRequestForm
from sqlalchemy.exc import IntegrityError
from sqlmodel import select, Session

from ..deps.session import get_session
from ..models.user import User, UserCreate, UserRead
from ..deps.security import hash_password, verify_password
from ..deps.jwt import create_access_token

router = APIRouter()


@router.post("/register", response_model=UserRead, status_code=201)
def register(
    user_in: UserCreate,
    session: Session = Depends(get_session),
) -> User:
    """Register a new user account.

    Args:
        user_in: The data needed to create a new user.
        session: Database session dependency.

    Raises:
        HTTPException: If the email is already registered.
```

```
    Returns:
        The newly created user (excluding sensitive fields).
    """
    user = User(
        **user_in.model_dump(exclude={"password"}),
        password_hash=hash_password(user_in.password),
    )
    session.add(user)
    try:
        session.commit()
        session.refresh(user)
    except IntegrityError:
        raise HTTPException(status_code=400, detail="Email already registered")
    return user


@router.post("/login")
def login(
    form_data: OAuth2PasswordRequestForm = Depends(),
    session: Session = Depends(get_session),
) -> dict[str, str]:
    """Authenticate a user and return a JWT access token.

    Args:
        form_data: Form data containing username (email) and password.
        session: Database session dependency.

    Raises:
        HTTPException: If the credentials are invalid.

    Returns:
        A dictionary with the access token and token type.
    """
    user = session.exec(
        select(User).where(User.email == form_data.username)
    ).one_or_none()

    if not user or not verify_password(form_data.password, user.password_hash):
        raise HTTPException(status_code=401, detail="Invalid credentials")

    token = create_access_token({"sub": user.id, "role": user.role.value})
    return {"access_token": token, "token_type": "bearer"}
```

```python
# app/routers/course.py
import os
from fastapi import APIRouter, Depends, HTTPException, status
from typing import List
from sqlmodel import Session

from ..deps.session import get_session
from ..deps.auth import get_current_user, get_current_teacher
from ..models.course import CourseRead, CourseCreate, CourseUpdate
from ..models.user import User
from ..crud.course import (
    get_courses,
    get_course,
    create_course,
    update_course,
    delete_course,
)

router = APIRouter()


@router.get("/", response_model=List[CourseRead])
def read_courses(
    session: Session = Depends(get_session),
    user: User = Depends(get_current_user),
) -> list[CourseRead]:
    """Get a list of all courses.

    Args:
        session: SQLModel DB session.
        user: Authenticated user (any role).

    Returns:
        A list of CourseRead schemas.
    """
    return get_courses(session)


@router.post("/", response_model=CourseRead, status_code=status.HTTP_201_CREATED)
def create_new_course(
    course_in: CourseCreate,
    session: Session = Depends(get_session),
    teacher: User = Depends(get_current_teacher),
```

```
) -> CourseRead:
    """Create a new course (teacher only).

    Args:
        course_in: The input data for the new course.
        session: SQLModel DB session.
        teacher: Currently authenticated teacher.

    Returns:
        The created course data.
    """
    return create_course(
        session,
        name=course_in.name,
        description=course_in.description,
        teacher_id=teacher.id,
    )


@router.get("/{course_id}", response_model=CourseRead)
def read_course(
    course_id: int,
    session: Session = Depends(get_session),
    user: User = Depends(get_current_user),
) -> CourseRead:
    """Get a specific course by its ID.

    Args:
        course_id: The ID of the course to retrieve.
        session: SQLModel DB session.
        user: Authenticated user (any role).

    Raises:
        HTTPException: If the course is not found.

    Returns:
        The CourseRead schema of the course.
    """
    course = get_course(session, course_id)
    if not course:
        raise HTTPException(status.HTTP_404_NOT_FOUND, "Course not found")
    return course
```

```
@router.put("/{course_id}", response_model=CourseRead)
def update_existing_course(
    course_id: int,
    course_up: CourseUpdate,
    session: Session = Depends(get_session),
    teacher: User = Depends(get_current_teacher),
) -> CourseRead:
    """Update an existing course (teacher only).

    Args:
        course_id: ID of the course to update.
        course_up: Fields to update.
        session: SQLModel DB session.
        teacher: Authenticated teacher user.

    Raises:
        HTTPException: If course is not found or belongs to another teacher.

    Returns:
        The updated course data.
    """
    course = get_course(session, course_id)
    if not course:
        raise HTTPException(status.HTTP_404_NOT_FOUND, "Course not found")
    if course.teacher_id != teacher.id:
        raise HTTPException(status.HTTP_403_FORBIDDEN, "Not your course")
    updated = update_course(
        session,
        course_id,
        name=course_up.name,
        description=course_up.description,
    )
    return updated


@router.delete("/{course_id}", status_code=status.HTTP_204_NO_CONTENT)
def delete_existing_course(
    course_id: int,
    session: Session = Depends(get_session),
    teacher: User = Depends(get_current_teacher),
) -> None:
    """Delete an existing course (teacher only).
```

```
        Args:
            course_id: ID of the course to delete.
            session: SQLModel DB session.
            teacher: Authenticated teacher user.

        Raises:
            HTTPException: If course is not found or belongs to another teacher.

        Returns:
            None
        """
        course = get_course(session, course_id)
        if not course:
            raise HTTPException(status.HTTP_404_NOT_FOUND, "Course not found")
        if course.teacher_id != teacher.id:
            raise HTTPException(status.HTTP_403_FORBIDDEN, "Not your course")
        delete_course(session, course_id)

# app/routers/lesson.py
"""Lesson router: defines endpoints for creating, reading, updating, and deleting
lessons."""

import os
from uuid import uuid4
from fastapi import APIRouter, Depends, HTTPException, status, UploadFile, File, Form
from sqlmodel import Session

from ..deps.session import get_session
from ..deps.auth import get_current_user, get_current_teacher
from ..models.lesson import LessonRead
from ..models.user import User
from ..models.course import Course
from ..crud.lesson import (
    get_lessons_by_course,
    get_lesson,
    create_lesson,
    update_lesson,
    delete_lesson,
)

router = APIRouter()
```

```
@router.get("/by_course/{course_id}", response_model=list[LessonRead])
def read_lessons(
    course_id: int,
    session: Session = Depends(get_session),
    user: User = Depends(get_current_user),
) -> list[LessonRead]:
    """Retrieve all lessons for a specific course.

    Args:
        course_id: The ID of the course.
        session: SQLModel DB session.
        user: Authenticated user.

    Returns:
        A list of lessons belonging to the course.
    """
    return get_lessons_by_course(session, course_id)


@router.get("/{lesson_id}", response_model=LessonRead)
def read_lesson(
    lesson_id: int,
    session: Session = Depends(get_session),
    user: User = Depends(get_current_user),
) -> LessonRead:
    """Get a lesson by its ID.

    Args:
        lesson_id: The ID of the lesson to retrieve.
        session: SQLModel DB session.
        user: Authenticated user.

    Raises:
        HTTPException: If the lesson is not found.

    Returns:
        The lesson object.
    """
    lesson = get_lesson(session, lesson_id)
    if not lesson:
        raise HTTPException(status.HTTP_404_NOT_FOUND, "Lesson not found")
    return lesson
```

```
@router.post(
    "/by_course/{course_id}",
    response_model=LessonRead,
    status_code=status.HTTP_201_CREATED,
)
def create_new_lesson(
    course_id: int,
    title: str = Form(...),
    content: str = Form(...),
    video_url: str = Form(...),
    file: UploadFile = File(...),
    session: Session = Depends(get_session),
    teacher: User = Depends(get_current_teacher),
) -> LessonRead:
    """Create a new lesson under a course (teacher only).

    Args:
        course_id: ID of the course the lesson belongs to.
        title: Title of the lesson.
        content: Text content of the lesson.
        video_url: YouTube video URL.
        file: Uploaded image file for the lesson.
        session: SQLModel DB session.
        teacher: Authenticated teacher user.

    Raises:
        HTTPException: If the course doesn't exist or is not owned by the teacher.

    Returns:
        The created lesson object.
    """
    course = session.get(Course, course_id)
    if not course:
        raise HTTPException(status.HTTP_404_NOT_FOUND, "Course not found")
    if course.teacher_id != teacher.id:
        raise HTTPException(status.HTTP_403_FORBIDDEN, "Not your course")

    ext = os.path.splitext(file.filename)[1]
    filename = f"{uuid4().hex}{ext}"
    upload_dir = os.path.join("app", "static", "uploaded_images")
    path = os.path.join(upload_dir, filename)
```

```
    with open(path, "wb") as out:
        out.write(file.file.read())

    image_path = f"uploaded_images/{filename}"
    lesson = create_lesson(
        session,
        course_id=course_id,
        title=title,
        content=content,
        video_url=video_url,
        image_path=image_path,
    )
    return lesson


@router.put("/{lesson_id}", response_model=LessonRead)
def update_existing_lesson(
    lesson_id: int,
    title: str | None = Form(None),
    content: str | None = Form(None),
    video_url: str | None = Form(None),
    file: UploadFile | None = File(None),
    session: Session = Depends(get_session),
    teacher: User = Depends(get_current_teacher),
) -> LessonRead:
    """Update a lesson (teacher only).

    Args:
        lesson_id: ID of the lesson to update.
        title: Optional new title.
        content: Optional new content.
        video_url: Optional new video URL.
        file: Optional new image file.
        session: SQLModel DB session.
        teacher: Authenticated teacher user.

    Raises:
        HTTPException: If the lesson is not found or owned by another teacher.

    Returns:
        The updated lesson object.
    """
    lesson = get_lesson(session, lesson_id)
```

```
    if not lesson:
        raise HTTPException(status.HTTP_404_NOT_FOUND, "Lesson not found")
    course = session.get(Course, lesson.course_id)
    if course.teacher_id != teacher.id:
        raise HTTPException(status.HTTP_403_FORBIDDEN, "Not your lesson")

    image_path = None
    if file:
        ext = os.path.splitext(file.filename)[1]
        filename = f"{uuid4().hex}{ext}"
        upload_dir = os.path.join("app", "static", "uploaded_images")
        path = os.path.join(upload_dir, filename)
        with open(path, "wb") as out:
            out.write(file.file.read())
        image_path = f"uploaded_images/{filename}"

    updated = update_lesson(
        session,
        lesson_id,
        title=title,
        content=content,
        video_url=video_url,
        image_path=image_path,
    )
    return updated


@router.delete("/{lesson_id}", status_code=status.HTTP_204_NO_CONTENT)
def delete_existing_lesson(
    lesson_id: int,
    session: Session = Depends(get_session),
    teacher: User = Depends(get_current_teacher),
) -> None:
    """Delete a lesson (teacher only).

    Args:
        lesson_id: ID of the lesson to delete.
        session: SQLModel DB session.
        teacher: Authenticated teacher user.

    Raises:
        HTTPException: If the lesson is not found or not owned by the teacher.
```

```
    Returns:
        None.
    """
    lesson = get_lesson(session, lesson_id)
    if not lesson:
        raise HTTPException(status.HTTP_404_NOT_FOUND, "Lesson not found")
    course = session.get(Course, lesson.course_id)
    if course.teacher_id != teacher.id:
        raise HTTPException(status.HTTP_403_FORBIDDEN, "Not your lesson")
    delete_lesson(session, lesson_id)

# app/routers/user.py
"""User router: endpoints for retrieving and updating the authenticated user's data."""

from fastapi import APIRouter, Depends, HTTPException, status
from sqlmodel import Session
from pydantic import BaseModel, EmailStr
from sqlalchemy.exc import IntegrityError

from ..deps.session import get_session
from ..deps.auth import get_current_user
from ..deps.security import hash_password
from ..models.user import User, UserRead

router = APIRouter()


class UserUpdate(BaseModel):
    """Schema for user profile update input.

    Attributes:
        name: Optional new first name.
        surname: Optional new last name.
        email: Optional new email.
        password: Optional new password (will be hashed).
    """
    name: str | None = None
    surname: str | None = None
    email: EmailStr | None = None
    password: str | None = None


@router.get("/me", response_model=UserRead)
```

```
def read_current_user(user: User = Depends(get_current_user)) -> User:
    """Get the current authenticated user's profile.

    Args:
        user: The currently authenticated user (injected by dependency).

    Returns:
        The user profile as a `UserRead` schema.
    """
    return user


@router.put("/me", response_model=UserRead)
def update_current_user(
    update: UserUpdate,
    session: Session = Depends(get_session),
    user: User = Depends(get_current_user),
) -> User:
    """Update the current user's profile.

    Args:
        update: The fields to update.
        session: SQLModel DB session.
        user: The currently authenticated user.

    Raises:
        HTTPException: If the updated email is already in use.

    Returns:
        The updated user profile.
    """
    if update.name is not None:
        user.name = update.name
    if update.surname is not None:
        user.surname = update.surname
    if update.email is not None:
        user.email = update.email
    if update.password is not None:
        user.password_hash = hash_password(update.password)

    session.add(user)
    try:
        session.commit()
```

```
            session.refresh(user)
        except IntegrityError:
            session.rollback()
            raise HTTPException(
                status_code=status.HTTP_400_BAD_REQUEST,
                detail="Email is already in use",
            )

        return user

    # app/main.py
    """Main application entry point for FastAPI app setup, middleware, and route registration."""

    from typing import Annotated
    from contextlib import asynccontextmanager

    from fastapi import Depends, FastAPI
    from fastapi.middleware.cors import CORSMiddleware
    from fastapi.staticfiles import StaticFiles
    from sqlmodel import Session

    from .db.database import engine, init_db
    from .routers import auth, user, course, lesson

    # Allowed CORS origins (e.g. frontend running on localhost:3000)
    origins = ["http://localhost:3000"]


    @asynccontextmanager
    async def lifespan(app: FastAPI):
        """Initialize resources (e.g., database) when app starts.

        Args:
            app: The FastAPI application instance.

        Yields:
            None. Allows FastAPI to continue with startup.
        """
        init_db()
        yield
```

```
# Create the FastAPI app
app = FastAPI(lifespan=lifespan)

# Apply CORS middleware
app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Serve static files from /static
app.mount("/static", StaticFiles(directory="app/static"), name="static")

# Register routers
app.include_router(auth.router,   prefix="/auth",    tags=["auth"])
app.include_router(user.router,   prefix="/users",   tags=["users"])
app.include_router(course.router, prefix="/courses", tags=["courses"])
app.include_router(lesson.router, prefix="/lessons", tags=["lessons"])
```

```
// src/auth/AuthContext.jsx
import { createContext, useContext, useState } from "react";

const AuthContext = createContext();

export function AuthProvider({ children }) {
  const [token, setToken] = useState(localStorage.getItem("token"));

  const login = (newToken) => {
    localStorage.setItem("token", newToken);
    setToken(newToken);
  };
  const logout = () => {
    localStorage.removeItem("token");
    setToken(null);
  };

  return (
    <AuthContext.Provider value={{ token, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
```

```
}

export const useAuth = () => useContext(AuthContext);


// src/components/About.jsx
import React from "react";
import pageStyles from "../styles/Page.module.css";
import styles from "../styles/components/About.module.css";

export default function About() {
  return (
    <div className={`${pageStyles.container} ${styles.container}`}>
      <h2 className={styles.title}>Про нас</h2>
      <p className={styles.text}>
        У відповідь на актуальні виклики  воєнного часу, ми створили веб-застосунок для підготовки
        операторів FPV-дронів - сучасну навчальну платформу, що поєднує теоретичний і практичний контент,
         відеоінструкції,  покрокові гайди та посилити  обороноздатність України через доступну доступну
         і структуровану освіту в галузі новітніх технологій.
      </p>
    </div>
  );
}


// src/components/Contacts.jsx
import React from "react";
import pageStyles from "../styles/Page.module.css";
import styles from "../styles/components/Contacts.module.css";

export default function Contacts() {
  return (
    <div className={`${pageStyles.container} ${styles.container}`}>
      <h2 className={styles.title}>Контакти</h2>
      <p className={styles.info}>
        uavosvita@gmail.com
      </p>
      <p className={styles.info}>
        +38 066 444 66 55
      </p>
```

```
    </div>
  );
}


// src/components/CourseDetail.jsx
import React, { useEffect, useState } from 'react';
import pageStyles from "../styles/Page.module.css";
import styles     from "../styles/components/CourseDetail.module.css";
import formStyles from "../styles/Form.module.css";
import { Link, useNavigate, useParams } from 'react-router-dom';
import { useAuth } from '../auth/AuthContext';
import useCurrentUser from '../hooks/useCurrentUser';

const API = process.env.REACT_APP_API_URL;

export default function CourseDetail() {
  const { token } = useAuth();
  const user = useCurrentUser();
  const navigate = useNavigate();
  const { id } = useParams();

  const [course, setCourse] = useState(null);
  const [lessons, setLessons] = useState([]);
  const [error, setError] = useState('');

  useEffect(() => {
    fetch(`${API}/courses/${id}`, {
      headers: { Authorization: `Bearer ${token}` },
    })
      .then(res => res.json())
      .then(setCourse);

    fetch(`${API}/lessons/by_course/${id}`, {
      headers: { Authorization: `Bearer ${token}` },
    })
      .then(res => res.json())
      .then(setLessons);
  }, [id, token]);

  if (!course) return <p>Loading...</p>;

  const isOwner = user?.role === 'teacher' && user.id === course.teacher_id;
```

```
        const handleDelete = async () => {
          setError('');
          const res = await fetch(`${API}/courses/${id}`, {
            method: 'DELETE',
            headers: { Authorization: `Bearer ${token}` },
          });
          if (res.status === 403) {
            setError("You don't have permission to delete this course.");
          } else if (!res.ok) {
            setError('Failed to delete course.');
          } else {
            navigate('/courses');
          }
        };

        const handleEdit = () => {
          if (!isOwner) {
            setError("You don't have permission to edit this course.");
          } else {
            navigate(`/courses/${id}/edit`);
          }
        };

        return (
          <div className={`${pageStyles.container} ${styles.container}`}>
            <h2 className={styles.title}>{course.name}</h2>
            <p className={styles.description}>{course.description}</p>
            {error && <p style={{ color: 'red' }}>{error}</p>}

            {user?.role === 'teacher' && (
              <div className={styles.actions}>
                <button      className={formStyles.buttonPrimary}      onClick={handleEdit}
disabled={!isOwner}>
                  Редагувати курс
                </button>
                <button     className={formStyles.buttonPrimary}     onClick={handleDelete}
disabled={!isOwner}>
                  Видалити курс
                </button>
              </div>
            )}
```

```
            <h3>Уроки</h3>
            {isOwner && (
              <button       className={formStyles.buttonPrimary}       onClick={()       =>
navigate(`/courses/${id}/lessons/new`)}>
                + Створити урок
              </button>
            )}
            <ul className={styles.lessonList}>
              {lessons.map(l => (
                <li key={l.id} className={styles.lessonItem}>
                  <Link to={`/lessons/${l.id}`}>{l.title}</Link>
                </li>
              ))}
            </ul>
          </div>
        );
      }


      // src/components/Courses.jsx
      import React, { useEffect, useState } from 'react';
      import { Link, useNavigate } from 'react-router-dom';
      import { useAuth } from '../auth/AuthContext';
      import useCurrentUser from '../hooks/useCurrentUser';
      import pageStyles from "../styles/Page.module.css";
      import styles     from "../styles/components/CourseDetail.module.css";
      import formStyles from "../styles/Form.module.css";

      const API = process.env.REACT_APP_API_URL;

      export default function Courses() {
        const { token } = useAuth();
        const navigate = useNavigate();
        const user = useCurrentUser();
        const [courses, setCourses] = useState([]);

        useEffect(() => {
          if (!token) return navigate('/login', { replace: true });

          fetch(`${API}/courses`, {
            headers: { Authorization: `Bearer ${token}` },
          })
            .then(res => res.json())
```

```
            .then(setCourses)
            .catch(console.error);
        }, [token, navigate]);

        return (
          <div className={`${pageStyles.container} ${styles.container}`}>
            <h1 className={styles.title}>Курси</h1>
            {user?.role === 'teacher' && (
              <button       className={formStyles.buttonPrimary}       onClick={()       =>
navigate('/courses/new')}>
                + Створити курс
              </button>
            )}
            <ul className={styles.lessonList}>
              {courses.map(c => (
                <li className={styles.lessonItem} key={c.id}>
                  <Link to={`/courses/${c.id}`}>{c.name}</Link>
                </li>
              ))}
            </ul>
          </div>
        );
      }


      // src/components/CreateCourse.jsx
      import React, { useState } from 'react';
      import pageStyles from "../styles/Page.module.css";
      import styles     from "../styles/components/CreateCourse.module.css";
      import formStyles from "../styles/Form.module.css";
      import { useAuth } from '../auth/AuthContext';
      import { useNavigate } from 'react-router-dom';

      const API = process.env.REACT_APP_API_URL;

      export default function CreateCourse() {
        const { token } = useAuth();
        const navigate = useNavigate();
        const [form, setForm] = useState({ name: '', description: '' });

        const handleChange = e =>
          setForm({ ...form, [e.target.name]: e.target.value });
```

```
  const handleSubmit = async e => {
    e.preventDefault();
    const res = await fetch(`${API}/courses`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        Authorization: `Bearer ${token}`,
      },
      body: JSON.stringify(form),
    });
    if (!res.ok) {
      alert('Failed to create course');
      return;
    }
    const course = await res.json();
    navigate(`/courses/${course.id}`);
  };

  return (
    <div className={`${pageStyles.container} ${styles.container}`}>
      <h2 className={styles.title}>Створити курс</h2>
      <form className={formStyles.form} onSubmit={handleSubmit}>
        <input
          name="name"
          className={formStyles.input}
          value={form.name}
          onChange={handleChange}
          placeholder="Назва курсу"
          required
        />
        <textarea
          name="description"
          className={formStyles.input}
          value={form.description}
          onChange={handleChange}
          placeholder="Опис курсу"
          required
        />
        <button                                    className={formStyles.buttonSecondary}
type="submit">Зберегти</button>
        <button  className={formStyles.buttonSecondary}  type="button"  onClick={()  =>
navigate('/courses')}>
          Скасувати
```

```
        </button>
      </form>
    </div>
  );
}


// src/components/CreateLesson.jsx
import React, { useState } from 'react';
import { useAuth } from '../auth/AuthContext';
import { useNavigate, useParams } from 'react-router-dom';
import pageStyles from "../styles/Page.module.css";
import styles     from "../styles/components/CreateLesson.module.css";
import formStyles from "../styles/Form.module.css";

const API = process.env.REACT_APP_API_URL;

export default function CreateLesson() {
  const { token } = useAuth();
  const navigate = useNavigate();
  const { id: courseId } = useParams();
  const [form, setForm] = useState({
    title: '',
    content: '',
    video_url: '',
    file: null,
  });

  const handleChange = e => {
    const { name, value, files } = e.target;
    setForm(prev => ({
      ...prev,
      [name]: files ? files[0] : value,
    }));
  };

  const handleSubmit = async e => {
    e.preventDefault();
    const data = new FormData();
    data.append('title', form.title);
    data.append('content', form.content);
    data.append('video_url', form.video_url);
    data.append('file', form.file);
```

```
    const res = await fetch(`${API}/lessons/by_course/${courseId}`, {
      method: 'POST',
      headers: { Authorization: `Bearer ${token}` },
      body: data,
    });
    if (!res.ok) {
      alert('Failed to create lesson');
      return;
    }
    const lesson = await res.json();
    navigate(`/lessons/${lesson.id}`);
  };

  return (
    <div className={`${pageStyles.container} ${styles.container}`}>
      <h2 className={styles.title}>Створити урок</h2>
      <form className={formStyles.form} onSubmit={handleSubmit}>
        <input
          name="title"
          className={formStyles.input}
          value={form.title}
          onChange={handleChange}
          placeholder="Напишіть назву уроку"
          required
        />
        <textarea
          name="content"
          className={formStyles.input}
          value={form.content}
          onChange={handleChange}
          placeholder="Напишіть контент уроку"
          required
        />
        <input
          name="video_url"
          className={formStyles.input}
          value={form.video_url}
          onChange={handleChange}
          placeholder="Введіть YouTube Embed URL"
          required
        />
        <input
```

```
                name="file"
                className={formStyles.input}
                type="file"
                accept="image/*"
                onChange={handleChange}
                required
              />
              <div style={{ display: 'flex', gap: '0.5rem' }}>
                <button className={formStyles.buttonPrimary} type="submit">Зберети</button>
                <button  className={formStyles.buttonPrimary}  type="button"  onClick={()  =>
navigate(-1)}>Скасувати</button>
              </div>
            </form>
          </div>
        );
      }


      // src/components/EditCourse.jsx
      import React, { useEffect, useState } from 'react';
      import { useAuth } from '../auth/AuthContext';
      import { useNavigate, useParams } from 'react-router-dom';
      import useCurrentUser from '../hooks/useCurrentUser';
      import pageStyles from "../styles/Page.module.css";
      import styles     from "../styles/components/CreateCourse.module.css";
      import formStyles from "../styles/Form.module.css";

      const API = process.env.REACT_APP_API_URL;

      export default function EditCourse() {
        const { token } = useAuth();
        const user = useCurrentUser();
        const navigate = useNavigate();
        const { id } = useParams();

        const [form, setForm] = useState({ name: '', description: '' });
        const [teacherId, setTeacherId] = useState(null);
        const [error, setError] = useState('');
        const [loading, setLoading] = useState(true);

        useEffect(() => {
          fetch(`${API}/courses/${id}`, {
            headers: { Authorization: `Bearer ${token}` },
```

```
  })
    .then(res => res.json())
    .then(data => {
      setForm({ name: data.name, description: data.description });
      setTeacherId(data.teacher_id);
    })
    .catch(console.error)
    .finally(() => setLoading(false));
}, [id, token]);

if (loading) return <p>Loading...</p>;

const isOwner = user?.role === 'teacher' && user.id === teacherId;
if (!isOwner) {
  return (
    <div className={`${pageStyles.container} ${styles.container}`}>
      <p style={{ color: 'red' }}>
        Ви не маєте права змінювати курс, так як не його власником.
      </p>
      <button onClick={() => navigate(`/courses/${id}`)}>
        Назад
      </button>
    </div>
  );
}

const handleChange = e =>
  setForm({ ...form, [e.target.name]: e.target.value });

const handleSubmit = async e => {
  e.preventDefault();
  setError('');
  const res = await fetch(`${API}/courses/${id}`, {
    method: 'PUT',
    headers: {
      'Content-Type': 'application/json',
      Authorization: `Bearer ${token}`,
    },
    body: JSON.stringify(form),
  });
  if (res.status === 403) {
    setError("You don't have permission to update this course.");
  } else if (!res.ok) {
```

```
            setError('Failed to update course.');
          } else {
            navigate(`/courses/${id}`);
          }
        };

        return (
          <div className={`${pageStyles.container} ${styles.container}`}>
            <h2 className={styles.title}>Редагувати курс</h2>
            {error && <p style={{ color: 'red' }}>{error}</p>}
            <form className={formStyles.form} onSubmit={handleSubmit}>
              <input
                className={formStyles.input}
                name="name"
                value={form.name}
                onChange={handleChange}
                required
              />
              <textarea
                name="description"
                className={formStyles.input}
                value={form.description}
                onChange={handleChange}
                required
              />
              <div style={{ display: 'flex', gap: '0.5rem' }}>
                <button                                    className={formStyles.buttonPrimary}
type="submit">Зберегти</button>
                <button
                  type="button"
                  className={formStyles.buttonPrimary}
                  onClick={() => navigate(`/courses/${id}`)}
                >
                  Скасувати
                </button>
              </div>
            </form>
          </div>
        );
      }



      // src/components/EditLesson.jsx
```

```
import React, { useEffect, useState } from 'react';
import { useAuth } from '../auth/AuthContext';
import { useNavigate, useParams } from 'react-router-dom';
import useCurrentUser from '../hooks/useCurrentUser';
import pageStyles from "../styles/Page.module.css";
import styles     from "../styles/components/CreateLesson.module.css";
import formStyles from "../styles/Form.module.css";

const API = process.env.REACT_APP_API_URL;

export default function EditLesson() {
  const { token } = useAuth();
  const user = useCurrentUser();
  const navigate = useNavigate();
  const { id } = useParams();

  const [form, setForm] = useState({
    title: '',
    content: '',
    video_url: '',
    file: null,
    existingImage: '',
    course_id: null,
  });
  const [teacherId, setTeacherId] = useState(null);
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    // fetch lesson + course in one go
    fetch(`${API}/lessons/${id}`, {
      headers: { Authorization: `Bearer ${token}` },
    })
      .then(res => res.json())
      .then(lesson => {
        setForm(f => ({
          ...f,
          title: lesson.title,
          content: lesson.content,
          video_url: lesson.video_url,
          existingImage: lesson.image_path,
          course_id: lesson.course_id,
        }));
```

```
      return fetch(`${API}/courses/${lesson.course_id}`, {
        headers: { Authorization: `Bearer ${token}` },
      });
    })
    .then(res => res.json())
    .then(course => {
      setTeacherId(course.teacher_id);
    })
    .catch(console.error)
    .finally(() => setLoading(false));
}, [id, token]);

if (loading) return <p>Loading...</p>;

const isOwner = user?.role === 'teacher' && user.id === teacherId;
if (!isOwner) {
  return (
    <div className={`${pageStyles.container} ${styles.container}`}>
      <p style={{ color: 'red' }}>
        Ви не маєте доступу, щоб редагувати урок.
      </p>
      <button onClick={() => navigate(`/lessons/${id}`)}>
        Повернутися
      </button>
    </div>
  );
}

const handleChange = e => {
  const { name, value, files } = e.target;
  setForm(prev => ({ ...prev, [name]: files ? files[0] : value }));
};

const handleSubmit = async e => {
  e.preventDefault();
  setError('');
  const data = new FormData();
  data.append('title', form.title);
  data.append('content', form.content);
  data.append('video_url', form.video_url);
  if (form.file) data.append('file', form.file);

  const res = await fetch(`${API}/lessons/${id}`, {
```

```
      method: 'PUT',
      headers: { Authorization: `Bearer ${token}` },
      body: data,
    });
    if (res.status === 403) {
      setError("You don't have permission to update this lesson.");
    } else if (!res.ok) {
      setError('Failed to update lesson.');
    } else {
      navigate(`/lessons/${id}`);
    }
  };

  return (
    <div className={`${pageStyles.container} ${styles.container}`}>
      <h2 className={styles.title}>Редагувати урок</h2>
      {error && <p style={{ color: 'red' }}>{error}</p>}
      <form
        className={formStyles.form}
        onSubmit={handleSubmit}
      >
          <input
            name="title"
            className={formStyles.input}
            value={form.title}
            onChange={handleChange}
            required
          />
          <textarea
            name="content"
            className={formStyles.input}
            value={form.content}
            onChange={handleChange}
            required
          />
          <input
            name="video_url"
            className={formStyles.input}
            value={form.video_url}
            onChange={handleChange}
            required
          />
        <div>
```

```
                <p>Поточне фото:</p>
                <img
                  className={styles.imagePreview}
                  src={`${API}/static/${form.existingImage}`}
                  alt="Current"
                  style={{ maxWidth: '200px' }}
                />
              </div>
              <label>
                Змінити фото (опційно):
                <input
                  name="file"
                  className={formStyles.input}
                  type="file"
                  accept="image/*"
                  onChange={handleChange}
                />
              </label>
              <div style={{ display: 'flex', gap: '0.5rem' }}>
                <button                                  className={formStyles.buttonPrimary}
type="submit">Зберегти</button>
                <button  className={formStyles.buttonPrimary}  type="button"  onClick={()  =>
navigate(-1)}>
                  Скасувати
                </button>
              </div>
            </form>
          </div>
        );
      }


      // src/components/Home.jsx
      import pageStyles from "../styles/Page.module.css";
      import styles from "../styles/components/Home.module.css";


      export default function Home() {
        return (
          <div className={pageStyles.container + " " + styles.container}>
            <h1 className={styles.title}>Знання - це крила, що  підносять до зірок</h1>
          </div>
        );
```

```
}


// src/components/LessonDetail.jsx
import React, { useEffect, useState } from 'react';
import { useAuth } from '../auth/AuthContext';
import { useNavigate, useParams } from 'react-router-dom';
import useCurrentUser from '../hooks/useCurrentUser';
import pageStyles from "../styles/Page.module.css";
import styles     from "../styles/components/LessonDetail.module.css";
import formStyles from "../styles/Form.module.css";


const API = process.env.REACT_APP_API_URL;

export default function LessonDetail() {
  const { token } = useAuth();
  const user = useCurrentUser();
  const navigate = useNavigate();
  const { id } = useParams();

  const [lesson, setLesson] = useState(null);
  const [courseTeacherId, setCourseTeacherId] = useState(null);
  const [error, setError] = useState('');

  // fetch lesson
  useEffect(() => {
    fetch(`${API}/lessons/${id}`, {
      headers: { Authorization: `Bearer ${token}` },
    })
      .then(res => res.json())
      .then(data => {
        setLesson(data);
        // then fetch course to get teacher_id
        return fetch(`${API}/courses/${data.course_id}`, {
          headers: { Authorization: `Bearer ${token}` },
        });
      })
      .then(res => res.json())
      .then(courseData => {
        setCourseTeacherId(courseData.teacher_id);
      })
      .catch(console.error);
```

```
}, [id, token]);

if (!lesson) return <p>Loading...</p>;

const isOwner =
  user?.role === 'teacher' && user.id === courseTeacherId;

const handleDelete = async () => {
  setError('');
  const res = await fetch(`${API}/lessons/${id}`, {
    method: 'DELETE',
    headers: { Authorization: `Bearer ${token}` },
  });
  if (res.status === 403) {
    setError("You don't have permission to delete this lesson.");
  } else if (!res.ok) {
    setError('Failed to delete lesson.');
  } else {
    navigate(`/courses/${lesson.course_id}`);
  }
};

const handleEdit = () => {
  if (!isOwner) {
    setError("You don't have permission to edit this lesson.");
  } else {
    navigate(`/lessons/${id}/edit`);
  }
};

return (
  <div className={`${pageStyles.container} ${styles.container}`}>
    <h2 className={styles.title}>{lesson.title}</h2>

    {error && <p style={{ color: 'red' }}>{error}</p>}
    <img className={styles.image}
      src={`${API}/static/${lesson.image_path}`}
      alt={lesson.title}
      style={{ maxWidth: '100%', marginBottom: '1rem' }}
    />
    <p>{lesson.content}</p>
    <div className={styles.videoWrapper}>
      <iframe
```

```
                width="560"
                height="315"
                src={lesson.video_url}
                title={lesson.title}
                frameBorder="0"
                allowFullScreen
              />
            </div>

            {user?.role === 'teacher' && (
              <div style={{ display: 'flex', gap: '0.5rem' }}>
                <button      className={formStyles.buttonPrimary}      onClick={handleEdit}
disabled={!isOwner}>
                  Редагувати урок
                </button>
                <button     className={formStyles.buttonPrimary}     onClick={handleDelete}
disabled={!isOwner}>
                  Видалити урок
                </button>
              </div>
            )}
          </div>
        );
      }


      // src/components/Login.jsx
      import { useState } from "react";
      import { useNavigate } from "react-router-dom";
      import { useAuth } from "../auth/AuthContext";
      import formStyles from "../styles/Form.module.css";
      import pageStyles from "../styles/Page.module.css";
      const API = process.env.REACT_APP_API_URL;

      export default function Login() {
        const [form, setForm] = useState({ username: "", password: "" });
        const [error, setError] = useState("");
        const { login } = useAuth();
        const navigate = useNavigate();

        const handleSubmit = async e => {
          e.preventDefault();
          const params = new URLSearchParams(form);
```

```
    try {
      const res = await fetch(`${API}/auth/login`, {
        method: "POST",
        headers: { "Content-Type": "application/x-www-form-urlencoded" },
        body: params,
      });
      const data = await res.json();
      if (!res.ok) throw new Error(data.detail || "Login failed");
      login(data.access_token);
      navigate("/courses");
    } catch (err) {
      setError(err.message);
    }
  };

  return (
    <div className={pageStyles.container}>
      <form onSubmit={handleSubmit} className={formStyles.form}>
        {["username","password"].map(f => (
          <input className={formStyles.input}
            key={f}
            name={f}
            type={f === "password" ? "password" : "text"}
            placeholder={f}
            value={form[f]}
            onChange={e => setForm({ ...form, [f]: e.target.value })}
            required
          />
        ))}
        <button className={formStyles.buttonPrimary} type="submit">Увійти</button>
        {error && <p className={formStyles.errorMsg}>{error}</p>}
      </form>
    </div>
  );
}


// src/components/Navbar.jsx
import { Link, useNavigate } from "react-router-dom";
import { useAuth } from "../auth/AuthContext";
import styles from "../styles/components/Navbar.module.css"

export default function Navbar() {
```

```
  const { token, logout } = useAuth();
  const navigate = useNavigate();

  const onLogout = () => {
    logout();
    navigate("/");
  };

  return (
    <nav className={styles.navbar}>
      <div className={styles.menu}>
        <Link className={styles.link} to="/">БПЛА Освіта</Link>
        <Link className={styles.link} to="/">головна</Link>
        <Link className={styles.link} to="/courses">курси</Link>
        <Link className={styles.link} to="/about">про нас</Link>
        <Link className={styles.link} to="/contacts">контакти</Link>

        {token ? (
          <>
            <Link className={styles.link} onClick={onLogout}>вийти</Link>
            <Link className={styles.link} to="/user">профіль</Link>
          </>
        ) : (
          <>
            <Link className={styles.link} to="/login">увійти</Link>
            <Link className={styles.link} to="/register">зареєструватися</Link>
          </>
        )}
      </div>
    </nav>
  );
}


// src/components/ProtectedRoute.jsx
import { Navigate } from "react-router-dom";
import { useAuth } from "../auth/AuthContext";

export default function ProtectedRoute({ children }) {
  const { token } = useAuth();
  return token ? children : <Navigate to="/login" replace />;
}
```

```
// src/components/Register.jsx
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import formStyles from "../styles/Form.module.css";
import pageStyles from "../styles/Page.module.css";

const API = process.env.REACT_APP_API_URL;

export default function Register() {
  const [form, setForm] = useState({ name: "", surname: "", email: "", password: "",
role: "student" });
  const [error, setError] = useState("");
  const navigate = useNavigate();

  const handleChange = e => setForm({ ...form, [e.target.name]: e.target.value });

  const handleSubmit = async e => {
    e.preventDefault();
    try {
      const res = await fetch(`${API}/auth/register`, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(form),
      });
      if (!res.ok) {
        const text = await res.text();
        throw new Error(text || "Registration failed");
      }
      navigate("/login");
    } catch (err) {
      setError(err.message);
    }
  };

  return (
    <div className={pageStyles.container}>
      <form onSubmit={handleSubmit} className={formStyles.form}>
        {["name","surname","email","password"].map(f => (
          <input className={formStyles.input}
            key={f}
            name={f}
            type={f === "password" ? "password" : "text"}
```

```
                placeholder={f}
                value={form[f]}
                onChange={handleChange}
                required
              />
            ))}
            <select    className={formStyles.select}    name="role"    value={form.role}
onChange={handleChange}>
              <option value="student">Я учень</option>
              <option value="teacher">Я вчитель</option>
            </select>
            <button                                    className={formStyles.buttonPrimary}
type="submit">Зареєструватися</button>
            {error && <p className={formStyles.errorMsg}>{error}</p>}
          </form>
        </div>
      );
    }


    // src/components/User.jsx
    import React, { useEffect, useState } from 'react';
    import { useAuth } from '../auth/AuthContext';
    import { useNavigate } from 'react-router-dom';
    import pageStyles from "../styles/Page.module.css";
    import formStyles from "../styles/Form.module.css";
    import styles     from "../styles/components/User.module.css";

    const API = process.env.REACT_APP_API_URL;

    export default function User() {
      const { token } = useAuth();
      const navigate = useNavigate();
      const [user, setUser] = useState(null);
      const [edit, setEdit] = useState(false);

      // Now track email & a new password field
      const [form, setForm] = useState({
        name: '',
        surname: '',
        email: '',
        password: '',       // blank ⇒ no change
      });
```

```
  useEffect(() => {
    if (!token) return navigate('/login', { replace: true });

    const headers = { Authorization: `Bearer ${token}` };

    fetch(`${API}/users/me`, { headers })
      .then(res => {
        if (res.status === 401) {
          navigate('/login', { replace: true });
          throw new Error('Unauthorized');
        }
        if (!res.ok) throw new Error('Fetch failed');
        return res.json();
      })
      .then(data => {
        setUser(data);
        // initialize all fields
        setForm({
          name: data.name,
          surname: data.surname,
          email: data.email,
          password: '',
        });
      })
      .catch(console.error);
  }, [token, navigate]);

  const handleChange = e => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };

  const handleSubmit = async e => {
    e.preventDefault();

    // Build payload; always send name/surname/email, only include password if non-
empty
    const payload = {
      name: form.name,
      surname: form.surname,
      email: form.email,
    };
    if (form.password) {
```

```
            payload.password = form.password;
          }

          const headers = {
            'Content-Type': 'application/json',
            Authorization: `Bearer ${token}`,
          };

          try {
            const res = await fetch(`${API}/users/me`, {
              method: 'PUT',
              headers,
              body: JSON.stringify(payload),
            });
            if (!res.ok) throw new Error('Update failed');
            const updated = await res.json();
            setUser(updated);
            // clear password field after success
            setForm(f => ({ ...f, password: '' }));
            setEdit(false);
          } catch (err) {
            console.error(err);
          }
        };

        if (!user) return <p>Loading...</p>;

        return (
          <div className={`${pageStyles.container} ${styles.container}`}>
            <h2 className={styles.title}>Мій профіль</h2>

            {!edit ? (
              <>
                <p className={styles.field}>Ім'я: {user.name}</p>
                <p className={styles.field} >Прізвище: {user.surname}</p>
                <p className={styles.field}>Email: {user.email}</p>
                <p className={styles.field}>Роль: {user.role}</p>
                <button      className={formStyles.buttonPrimary}      onClick={()      =>
setEdit(true)}>Редагувати профіль</button>
              </>
            ) : (
              <form  className={formStyles.form}  onSubmit={handleSubmit}  style={{  display:
'flex', flexDirection: 'column', gap: '0.5rem' }}>
```

```
                <input
                  name="name"
                  className={formStyles.input}
                  value={form.name}
                  onChange={handleChange}
                  placeholder="Ім'я"
                  required
                />
                <input
                  name="surname"
                  className={formStyles.input}
                  value={form.surname}
                  onChange={handleChange}
                  placeholder="Прізвище"
                  required
                />
                <input
                  name="email"
                  className={formStyles.input}
                  type="email"
                  value={form.email}
                  onChange={handleChange}
                  placeholder="Email"
                  required
                />
                <input
                  name="password"
                  className={formStyles.input}
                  type="password"
                  value={form.password}
                  onChange={handleChange}
                  placeholder="Новий пароль (Залиште пустим, щоб не змінювати)"
                />
                <div style={{ display: 'flex', gap: '0.5rem' }}>
                  <button                                 className={formStyles.buttonPrimary}
type="submit">Зберегти</button>
                  <button className={formStyles.buttonPrimary}  type="button" onClick={() =>
setEdit(false)}>
                    Скасувати
                  </button>
                </div>
              </form>
            )}
```

```
    </div>
  );
}


// src/hooks/useCurrentUser.js
import { useState, useEffect } from 'react';
import { useAuth } from '../auth/AuthContext';

const API = process.env.REACT_APP_API_URL;

export default function useCurrentUser() {
  const { token } = useAuth();
  const [user, setUser] = useState(null);

  useEffect(() => {
    if (!token) return;
    fetch(`${API}/users/me`, {
      headers: { Authorization: `Bearer ${token}` },
    })
      .then(res => (res.ok ? res.json() : null))
      .then(setUser)
      .catch(() => setUser(null));
  }, [token]);

  return user;
}


// src/App.js
import React from "react";
import { Routes, Route } from "react-router-dom";

import Navbar from "./components/Navbar";
import Home from "./components/Home";
import About from "./components/About";
import Contacts from "./components/Contacts";

import Courses from "./components/Courses";
import CreateCourse from "./components/CreateCourse";
import EditCourse from "./components/EditCourse";
import CourseDetail from "./components/CourseDetail";
```

```
import CreateLesson from "./components/CreateLesson";
import LessonDetail from "./components/LessonDetail";
import EditLesson from "./components/EditLesson";

import Login from "./components/Login";
import Register from "./components/Register";
import User from "./components/User";
import ProtectedRoute from "./components/ProtectedRoute";

export default function App() {
  return (
    <>
      <Navbar />
      <Routes>
        {/* public */}
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/contacts" element={<Contacts />} />
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Register />} />

        {/* protected */}
        <Route
          path="/courses"
          element={
            <ProtectedRoute>
              <Courses />
            </ProtectedRoute>
          }
        />
        <Route
          path="/courses/new"
          element={
            <ProtectedRoute>
              <CreateCourse />
            </ProtectedRoute>
          }
        />
        <Route
          path="/courses/:id"
          element={
            <ProtectedRoute>
              <CourseDetail />
```

```
      </ProtectedRoute>
    }
  />
  <Route
    path="/courses/:id/edit"
    element={
      <ProtectedRoute>
        <EditCourse />
      </ProtectedRoute>
    }
  />
  <Route
    path="/courses/:id/lessons/new"
    element={
      <ProtectedRoute>
        <CreateLesson />
      </ProtectedRoute>
    }
  />

  <Route
    path="/lessons/:id"
    element={
      <ProtectedRoute>
        <LessonDetail />
      </ProtectedRoute>
    }
  />
  <Route
    path="/lessons/:id/edit"
    element={
      <ProtectedRoute>
        <EditLesson />
      </ProtectedRoute>
    }
  />

  <Route
    path="/user"
    element={
      <ProtectedRoute>
        <User />
      </ProtectedRoute>
```

```
          }
        />
      </Routes>
    </>
  );
}

// src/index.js
import "./index.css"
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router-dom";
import { AuthProvider } from "./auth/AuthContext";
import App from "./App";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <AuthProvider>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </AuthProvider>
);



<!-- public/index.html -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Educational platform front-end powered by React"
    />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>My Educational Platform</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
```

```
    <div id="root"></div>
    <!-- The bundled React scripts will be injected here by the build tool -->
  </body>
</html>



/* src/index.css */
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
body {
  font-family: Arial, sans-serif;
  background: #f7f8fa;
  color: #333;
  line-height: 1.6;
}
a {
  color: #007bff;
  text-decoration: none;
}
button {
  cursor: pointer;
  border: none;
  border-radius: 4px;
}


/* src/styles/Form.module.css */
.form {
  display: flex;
  flex-direction: column;
  gap: 0.75rem;
}
.input, .textarea, .select {
  padding: 0.5rem;
  font-size: 1rem;
  border: 1px solid #ccc;
  border-radius: 4px;
}
.textarea {
```

```
  resize: vertical;
  min-height: 100px;
}
.buttonPrimary {
  background: #007bff;
  color: white;
  padding: 0.5rem 1rem;
}
.buttonSecondary {
  background: #6c757d;
  color: white;
  padding: 0.5rem 1rem;
}
.errorMsg {
  color: red;
  font-size: 0.9rem;
}


/* src/styles/Page.module.css */
.container {
  max-width: 800px;
  margin: 2rem auto;
  background: white;
  padding: 2rem;
  border-radius: 8px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
}
.title {
  font-size: 2rem;
  margin-bottom: 1rem;
}


/* src/components/About.module.css */
.container {
  text-align: left;
}
.title {
  font-size: 1.8rem;
  margin-bottom: 1rem;
}
.text {
```

```
  font-size: 1rem;
  color: #555;
}

/* src/components/Contacts.module.css */
.container {
  text-align: left;
}
.title {
  font-size: 1.8rem;
  margin-bottom: 1rem;
}
.info {
  font-size: 1rem;
  line-height: 1.5;
}

/* src/components/CourseDetail.module.css */
.container {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}
.title {
  font-size: 1.8rem;
}
.description {
  font-size: 1rem;
  color: #444;
}
.actions {
  display: flex;
  gap: .5rem;
}
.lessonList {
  list-style: none;
  padding-left: 0;
}
.lessonItem {
  padding: .5rem 0;
  border-bottom: 1px solid #eee;
}
```

```
/* src/components/Courses.module.css */
.container {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}
.title {
  font-size: 2rem;
  margin-bottom: .5rem;
}
.list {
  list-style: none;
  padding-left: 0;
}
.item {
  padding: .5rem;
  border-bottom: 1px solid #eee;
}
.createBtn {
  align-self: flex-start;
}


/* src/components/CreateCourse.module.css */
.container {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}
.title {
  font-size: 1.6rem;
}

/* src/components/CreateLesson.module.css */
.container {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}
.title {
  font-size: 1.6rem;
}
.imagePreview {
```

```
  max-width: 200px;
  margin-bottom: 1rem;
}

/* src/components/Home.module.css */
.container {
  text-align: center;
  padding: 4rem 1rem;
}
.title {
  font-size: 2.5rem;
  margin-bottom: 1rem;
}
.subtitle {
  font-size: 1.2rem;
  color: #555;
}

/* src/components/LessonDetail.module.css */
.container {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}
.title {
  font-size: 1.8rem;
}
.image {
  max-width: 100%;
  border-radius: 4px;
}
.videoWrapper {
  position: relative;
  padding-bottom: 56.25%;
  height: 0;
}
.videoWrapper iframe {
  position: absolute;
  width: 100%;
  height: 100%;
}
.actions {
  display: flex;
```

```
  gap: .5rem;
  margin-top: 1rem;
}

/* src/components/Navbar.module.css */
.navbar {
  background: #333;
  color: white;
  padding: 1rem;
  display: flex;
  align-items: center;
  justify-content: space-between;
}
.menu {
  display: flex;
  gap: 1rem;
}
.link {
  color: white;
  font-weight: bold;
}
.link:hover {
  text-decoration: underline;
}

/* src/components/User.module.css */
.container {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}
.field {
  font-size: 1rem;
}
.actions {
  display: flex;
  gap: .5rem;
}
```