

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

«Платформа для організації благодійних подій»

Виконав студент групи ПЗ-21-3
Дячук Денис Олександрович

Керівник роботи:
Годлевський Юрій Олександрович

Рецензент:
Лобанчикова Надія Миколаївна

Житомир – 2025

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Дячук Денис Олександрович**

Керівник роботи: **Годлевський Юрій Олександрович**

Тема роботи: **«Платформа для організації благодійних подій»,**

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

Вихідні дані для роботи: **Процес краудфандингу та збору команд для підтримки благодійних подій. Робота охоплює інтеграцію ролей користувачів (організатор, інвестор, волонтер)**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Годлевський Ю.О.	27.02.2025р.	27.02.2025р.
2	Годлевський Ю.О.	27.02.2025р.	27.02.2025р.
3	Годлевський Ю.О.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна праця бакалавра складається з вебзастосунку — краудфандингової платформи для влаштування благодійних заходів — та пояснювальної записки.

Пояснювальна записка до роботи містить 58 сторінок, 32 ілюстрації та 19 таблиць.

Метою роботи є розробка сучасної, зручної та функціональної вебплатформи, що дозволяє організаторам створювати благодійні проекти, волонтерам долучатися до їх реалізації, а інвесторам — підтримувати ініціативи шляхом фінансових внесків.

Застосунок орієнтований на різні категорії користувачів, що підвищує ефективність координації та залучення ресурсів для благодійних заходів в онлайн-середовищі.

Об'єктом дослідження є процес організації та фінансування благодійних заходів за допомогою краудфандингових технологій. Предметом дослідження є методи розробки вебзастосунку для краудфандингової платформи, зокрема реалізація багаторівневих ролей користувачів і також принципи побудови зручного інтерфейсу та ефективного управління проектами і пожертвами.

У праці застосовано методи об'єктно-орієнтованого програмування, веброзробки, створення REST API, роботу з базами даних, адаптивний дизайн. Особливу увагу приділено реалізації системи керування ролями користувачів, механізмам створення та підтримки благодійних проектів, а також забезпеченню безпеки та зручності взаємодії на платформі.

КЛЮЧОВІ СЛОВА: КРАУДФАНДИНГ, БЛАГОДІЙНІ ЗАХОДИ, ВЕБЗАСТОСУНОК, ВЕБ-ТЕХНОЛОГІЇ, РОЛІ КОРИСТУВАЧІВ, ОРГАНІЗАЦІЯ ПОДІЙ

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ABSTRACT

The bachelor's thesis comprises a web application — a crowdfunding platform for coordinating charity events — and an explanatory note.

The explanatory note contains 58 pages, 32 illustrations, and 19 tables.

The goal of the work is to build a contemporary, user-friendly, and functional web platform that allows organizers to start charity projects, volunteers to engage in their execution, and investors to back initiatives through financial contributions.

The application is tailored for different categories of users, improving the effectiveness of coordination and resource mobilization for charity events in an online environment.

The object of investigation is the process of organizing and funding charity events utilizing crowdfunding web technologies. The subject of investigation is the methodologies of developing a web application for a crowdfunding platform, particularly the implementation of multi-level user roles, integration with calendars and video communication systems, and also the principles of creating a convenient interface and effective handling of projects and donations.

The work utilizes methods of object-oriented programming, web development, REST API creation, database management, responsive design. Particular emphasis is given to the implementation of user role management, mechanisms for originating and supporting charity projects, and ensuring the security and convenience of interaction on the platform.

KEYWORDS: CROWD FUNDING, CHARITY EVENTS, WEB APPLICATION, WEB TECHNOLOGIES, USER ROLES, EVENT ORGANIZATION

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

РЕФЕРАТ.....	4
ABSTRACT	5
ЗМІСТ	6
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	8
ВСТУП.....	10
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ.....	12
1.1 Постановка задачі	12
1.2 Аналіз аналогів програмного продукту.....	14
1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	18
Висновки до першого розділу.	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПЛАТФОРМИ ДЛЯ ОРГАНІЗАЦІЇ БЛАГОДІЙНИХ ПОДІЙ	22
2.1 Визначення варіантів використання та структури системи..	22
2.2 Розробка бази даних системи.....	31
2.3 Проектування та реалізація алгоритмів роботи	33
2.4 Реалізація функціоналу платформи для організації подій.....	35
Висновки до другого розділу	38
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ДЛЯ ОРГАНІЗАЦІЇ ПОДІЙ.....	39
3.1 Порядок встановлення та налаштування параметрів платформи	39
3.2 Структура інтерфейсу платформи для організації подій.....	41
3.3 Тестування платформи	49
Висновки до третього розділу	54

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	56
ДОДАТКИ.....	59

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД — база даних

ПЗ — програмне забезпечення

ІТ — інформаційні технології

СУБД — система управління базами даних

UI — користувацький інтерфейс (англ. User Interface)

UX — користувацький досвід (англ. User Experience)

HTML — мова розмітки гіпертексту (англ. HyperText Markup Language)

CSS — каскадні таблиці стилів (англ. Cascading Style Sheets)

JS — мова програмування JavaScript

API — інтерфейс прикладного програмування (англ. Application Programming Interface)

REST — архітектурний стиль для створення вебсервісів (англ. Representational State Transfer)

HTTPS — захищений протокол передавання гіпертексту

JWT — вебтокен JSON (англ. JSON Web Token)

Node.js — серверна платформа для виконання JavaScript-коду

Express – фреймворк для Node.js

React — бібліотека JavaScript для побудови UI

MongoDB — документно-орієнтована база даних

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

Краудфандинг — спосіб залучення фінансування від великої кількості людей через інтернет

Фреймворк – бібліотека з набором інструментів, що спрощує розробку ПЗ

Middleware – програмні засоби для пов'язування різних компонентів та прикладних програм

Фронтенд – клієнтська частина додатку

Бекенд – серверна частина додатку

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

ВСТУП

Актуальність теми. У наш час, коли країна потопає у війні, неабияку важливу роль має відігравати майбутнє наших дітей. Ми маємо піклуватись не лише про їх матеріальне забезпечення, навчання, виховання, а і про відпочинок. В час, коли по всій країні будь-де може приховуватись небезпека, неабияку увагу потрібно приділити тому, щоб забезпечити дітям безпечний відпочинок. Але щоб організовувати такі заходи, необхідно врахувати наступні аспекти: робоча команда, інвестиції, залучення допомоги від інших людей. Саме для цього необхідна платформа для об'єднання усіх аспектів в одному місці. Організатори таких подій зможуть знайти для себе як працівників, що зможуть допомогти провести захід, так і інвесторів, що мають бажання зробити пожертву на такі заходи. Таким чином, суттєво спрощується процес налагодження відносин між трьома сторонами одночасно та вирішується проблема організації заходів відпочинку для дітей. Адже вони – це майбутнє країни та наше завдання – зробити все, щоб вони могли відволіктись від усієї ситуації, що проходить в країні та відновитись психологічно.^[1]

Мета випускної роботи. Розробка вебсайту платформи для організації благодійних подій, що дозволяє з'єднати інвесторів, які бажають вкласти кошти в розвиток майбутнього дітей, волонтерів, що хочуть працювати та людей, що організовують такі заходи.

Для того, щоб досягти цієї мети необхідно виконати наступні завдання:

1. Провести аналіз функціоналу благодійних платформ та інших додатків, що займаються збором фінансів.
2. Обрати засоби для проектування платформи.
3. Розробити платформу для організації благодійних подій.

Об'єктом дослідження є процес організації та фінансування благодійних заходів за допомогою вебтехнологій, зокрема через краудфандингову платформу, що забезпечує взаємодію між організаторами, волонтерами та інвесторами.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Предметом дослідження є сучасні методи та інструменти розробки веб-застосунків для краудфандингу, зокрема механізми багаторівневої аутентифікації, управління ролями користувачів, а також принципи побудови зручного користувацького інтерфейсу та ефективної системи керування проектами та пожертвами.

Практичне значення одержаних результатів. Практичне значення результатів випускної роботи полягає у створенні функціональної краудфандингової платформи, яка може бути впроваджена для автоматизації організації благодійних заходів, покращення комунікації між учасниками та залучення фінансової підтримки від інвесторів. Реалізовані рішення сприятимуть оптимізації процесів координації проектів, підвищенню прозорості та ефективності збору коштів, а також розширенню можливостей для взаємодії волонтерів та організаторів.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Постановка задачі

Розробка платформи для організації благодійних подій передбачає створення зручної платформи, що сприятиме ефективному налагодженню зв'язку між різними користувачами та продуктивній організації відпочинку для дітей через зручний інтерфейс, інтуїтивно зрозумілий для користувачів різного віку, забезпечивши зручність налагодження зв'язку. Платформа обов'язково має враховувати потреби усіх користувачів: тих, хто ще не зареєстрований, організаторів подій, волонтерів та інвесторів, забезпечуючи необхідний функціонал для кожного користувача.

Платформа має надавати декілька модулів, що будуть забезпечувати наступний функціонал для різних користувачів:

Модуль створення подій дозволяє користувачам з роллю організатора подій створювати, розплановувати та керувати подіями. Цей модуль дозволяє організатору надати всю інформацію про подію, що планується, та яких ресурсів потребує подія. Також забезпечення організації зв'язку між користувачами для подальшого обговорення подій.

Модуль волонтерства забезпечує роботу з профілями волонтерів. Обробка заявок, відгуки про конкретного користувача, його рейтингова оцінка, історія участі в подіях, контактні дані. Усе це має оброблятися одним модулем.[\[2\]](#)

Модуль інвестицій робить своєю ціллю обробити всі потреби щодо фінансування. Цілі проектів, на що піде фінансування та забезпечення усіх необхідних способів для безпечного внесення коштів інвесторами. Його ціль забезпечити безпечність даних користувачів, піклуватись про те, щоб усі операції використовували захист і не були легкими до ураження.

Модуль відображення викорусі необхідні сторінки платформи, щоб забезпечити зручність користування сайтом та легкість знаходження потрібної інформації.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

В результаті повинен бути створений сайт, що дозволяє людям дізнатись про події, які мають відбуватись, знайти працівників на ці події та забезпечити можливість пожертв на ці події людьми. Процес реалізації платформи вимагає проходження декількох важливих етапів. Розглянемо їх детальніше.

Розробка дизайну сайту вимагає декілька кроків, таких як: створити загальний дизайн і кольорову гаму, розробити UI компоненти та дизайн сторінок платформи. Побудувати архітектуру клієнтської частини. Далі розробити і побудувати систему авторизації і ролей. Наступний етап це головний функціонал платформи, а саме система ролей, створення база даних для зберігання даних, імплементація серверної частини в клієнтську. Після цього потрібно проробити профіль користувача, створивши унікальні профілі для кожної ролі, додати можливість редагування своєї інформації та оновлення фото користувача. Далі йде система подій, тобто необхідно розробити методи створення подій, внесення необхідно інформації про події та можливість її редагування і організацію процесу фінансування проекту. І в кінці необхідно протестувати платформу, провівши функціональне, навантажувальне та тестування користувацького інтерфейсу. Загалом ці етапи формують план, при дотриманні якого можна реалізувати повноцінну платформу для організації благодійних подій.

В результаті виконання плану дій буде створення платформи, за допомогою якої користувачі можуть допомагати один одному в організації подій відпочинку для дітей, залученні коштів на ці події та збору команд працівників, що загалом спрощує процес створення нових дозволів в країні. А це в свою чергу дозволить піклуватись про майбутнє своєї країни для більшої кількості людей, пропагувати благодійність людям та зверне увагу людей на проблеми, що дійсно є важливими зараз. Саме тому важливо дотримуватись цього плану, реалізуючи платформу для організації благодійних подій.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

1.2 Аналіз аналогів програмного продукту

Розробка платформи для організації благодійних подій з можливостями краудфандингу вимагає дослідження конкурентів та аналогів задля того, щоб зрозуміти плюси та мінуси вже існуючих проектів та забезпечити максимально комфортний користувацький досвід. Саме тому необхідно провести цей аналіз не лише на повноцінних аналогах майбутнього додатку, а й на різнопланових схожих платформах, щоб зрозуміти, що саме необхідно забезпечити в нашій платформі. Одним із таких додатків є платформа Caritas Ukraine (рис. 1.1).[3]

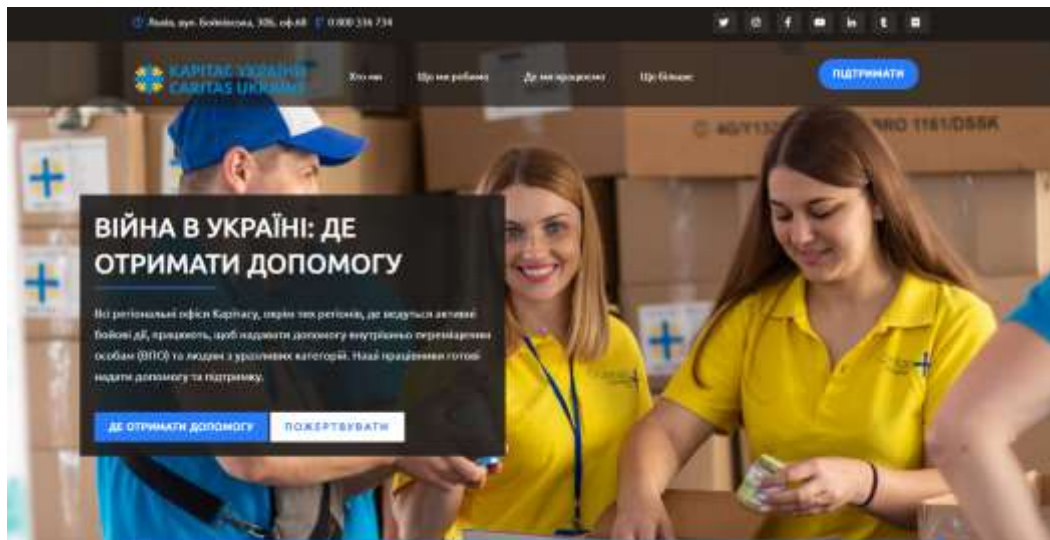


Рис. 1.1. Платформа краудфандингу Caritas Ukraine

Платформа займається збором пожертв зі всього світу. Перевагами цієї платформи є прозорість інформації подій. Доступна інформація про зібрані кошти, детальна інформація про події та для чого будуть використані кошти.

До основних переваг Caritas Ukraine можна віднести прозорість інформації, доступ до пожертв будь-кому без профілю користувача та події поділені по категоріях, завдяки чому швидше можна знайти потрібну подію. З недоліків є поганий адаптивний дизайн та загальний вигляд сайту, що означає, що нам потрібно попіклуватись про це, адже загальний вигляд платформи неабияк впливає на враження і відношення користувачів до платформи.

JustGiving - Це міжнародний онлайн-майданчик для збору коштів, що надає можливість як фізичним особам, так і добродійним організаціям

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

організовувати кампанії, залучати благодійні внески, проводити заходи та координувати підтримку(Рис. 1.2).[4]

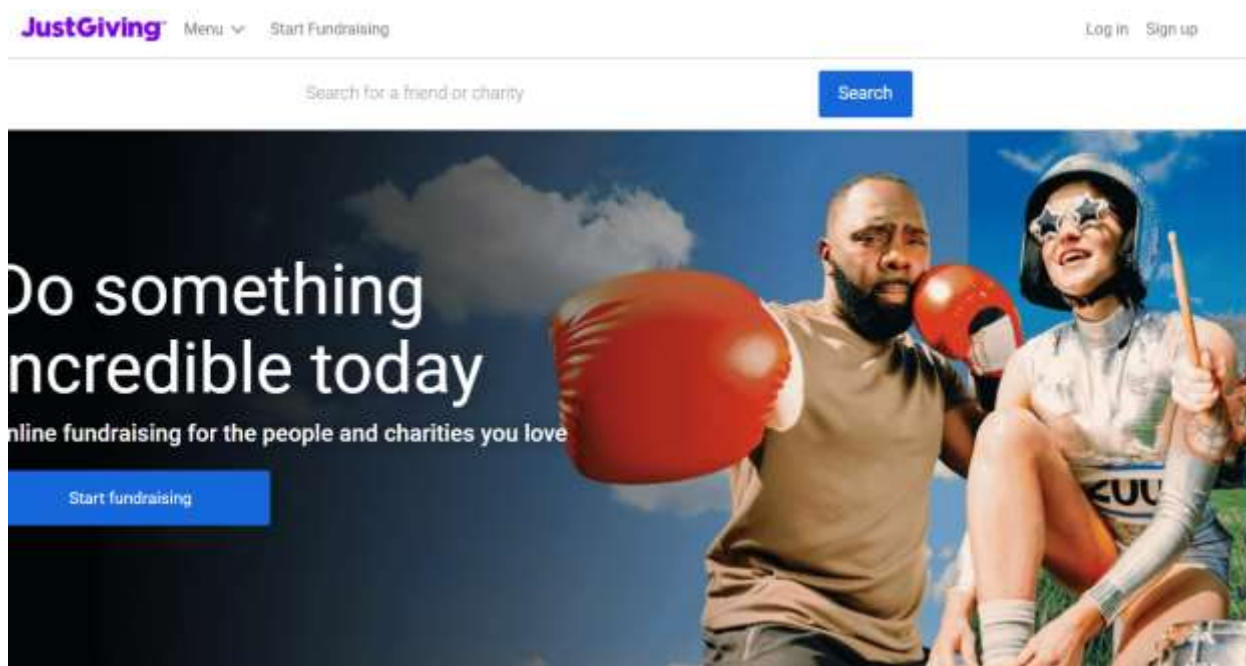


Рис. 1.2. Платформа організації благодійних подій JustGiving

Переваги платформи виявляються у легкому і швидкому запуску кампаній на платформі, зручному поширенню інформації про події через соціальні мережі, такі як Facebook та Twitter, широкий вибір способів оплати, включаючи міжнародні картки та раурал, прозорість статистики пожертв, доступність в багатьох країнах, можливість створювати брендові сторінки для організацій та автоматичні листи подяки для жертводавців. [5-6]

До недоліків платформи можна віднести комісію з кожної пожертви, відсутню інтеграція з відеозв'язком, а також платформа не орієнтована спеціально під волонтерів — більша схильність до жертводавців та організаторів подій.

Zeffy – платформа без комісій, що дозволяє публікувати події для всіх та збирати кошти двома методами: помісячна підписка та одноразові пожертви. З переваг можна виділити приємний простий та разом з цим динамічний інтерфейс, що допомагає заохотити користувачів ознайомитись з платформою ближче.[7]

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

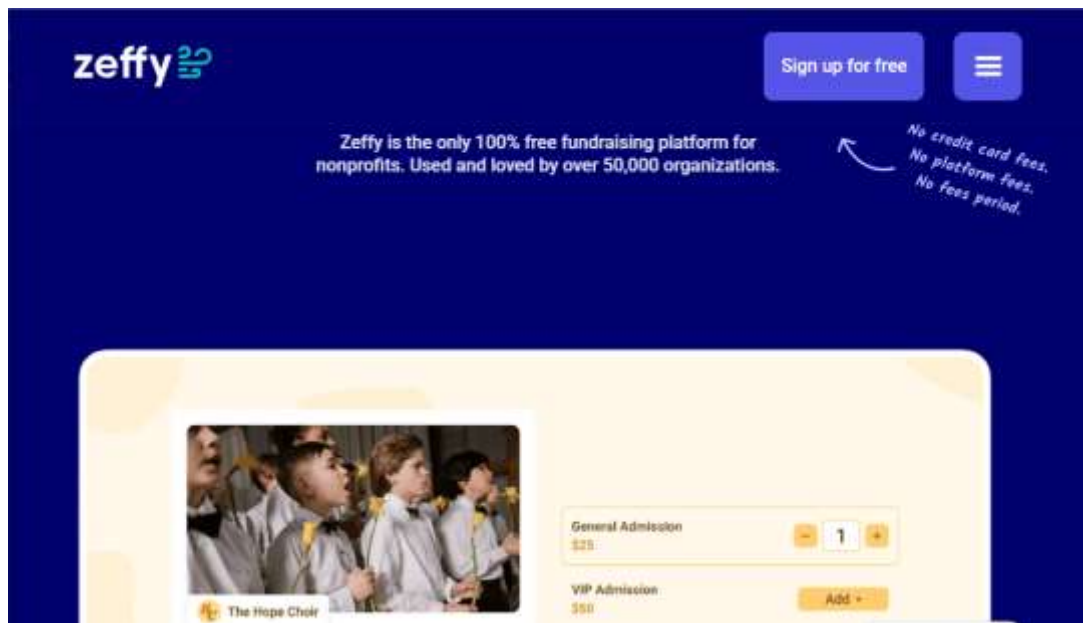


Рис. 1.3. Благодійна платформа Zeffy

Серед переваг можна виділити гарний дизайн, простоту інтерфейсу та відсутність комісій. А до недоліків можемо віднести відсутність української мови та неможливість використання платформи без облікового запису.

Patreon – платформа помісячної підтримки різних авторів та виконавців. Вона не розроблена саме для благодійних подій та благодійності, а задля підтримки різних авторів в інтернеті, а ті в свою чергу часто віддячують додатковим контентом. Але сама концепція платформи може бути використана саме для благодійних подій, щоб запрошувати помісячну підтримку.[8]



Рис. 1.4. Платформа Patreon

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Серед переваг платформи Patreon є зрозумілий інтерфейс, система проста у використанні, а також підтримка декількох мов. А до недоліків Patreon ми можемо внести лише помісячну оплату і проблеми з адаптивним дизайном

Орієнтуючись на інформацію про переваги та недоліки різноманітних платформ, що було отримано завдяки аналізу кожної з них, можемо створити наступну порівняльну таблицю недоліків та переваг(табл. 1.1):

Таблиця 1.1

Таблиця порівнянь благодійних платформ

Характеристика	Caritas Ukraine	JustGiving	Zeffy	Patreon
Доступ для пожертв всім	+	+	-	+/-
Події по категоріях	+	+	-	-
Привабливість дизайну	-	+	+	-
Простота інтерфейсу	+/-	+	+	+
Відсутність комісій	-	-	+	-
Доступ до платформи без аккаунта	+	+	-	-
Щомісячні оплати	-	+	+	+

Оглянувши отримані результати аналізу різних платформ, ми можемо зробити висновки відносно особливостей кожної платформи та виділити певні спорідненості і відмінності у кожної з них. Виділимо спільні особливості, що будуть важливими під час розробки платформи.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

Важливим аспектом має бути зручний і простий процес створення облікового запису, щоб будь-хто міг зручно та без зайвих дій скористатись цією можливістю. Дизайн має бути привабливим, інакше користувачі можуть відлякуватись через поганий дизайн. Також потрібно приділити увагу простоті цього інтерфейсу, щоб користувачі будь-якого віку могли користуватись платформою не відчуючи дискомфорту. Також необхідно дотримуватись відсутності комісій платформи на переказах, адже ціль майбутньої платформи – сприяти благодійності, а не заробляти кошти. Оглядаючи аналоги, в багатьох була можливість помісячної підписки, але зважаючи на сутність нашої майбутньої платформи ми можемо відмовитись від цієї опції, адже вона не є близькою до нашої ситуації.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Платформа буде побудована на основі клієнт-серверної архітектури. Для створення серверної частини у вигляді API використається сучасний бекенд фреймворк Express для серверної платформи Node.js. Клієнтська частина буде побудована з використанням бібліотеки ReactJS у форматі SPA(Single Page Application). Для зберігання даних використається нереляційна база даних MongoDB.[\[9-12\]](#)

Для реалізації серверної частини додатку було вирішено використовувати архітектурний патерн MVC: Model, View, Controller. Цей патерн було адаптовано під RESTful API-варіант, для того щоб зберегти зручність використання і читабельність коду. Таким чином структура зручна для читання та підтримує легкість і зрозумілість реалізації RESTful API – прості запити Get,Post,Put,Delete. [\[13\]](#)

Щоб створити структуровану клієнтську частину буде використано патерн Container-Presenter і забезпечити SPA(Single Page Application). Вирішено було розділити сторінки, відділивши компоненти що повторюються і містять більшість запитів до API. Завдяки цьому зменшується повторення

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

коду та додається структуризація файлів, що забезпечує ефективне використання сховища та читабельність структури файлів і коду.

Express.js або ж просто Express – легкий та зручний у використанні фреймворк для Node.js, що якісно спрощує процес розробки серверних додатків та дозволяє просто створювати REST API. Фреймворк, що має широке використання завдяки своїй простоті, можливості до розширення та гнучкості. До переваг Express можна назвати декілька важливих пунктів, а саме гнучкість і простоту використання, підтримку middleware, REST API і швидкий запуск проектів. Далі буде детальніше розписано в чому полягають його переваги, щоб краще обґрунтувати вибір саме Express. Мінімалістичний підхід до розробки серверних застосунків дозволяє розробникам будувати тільки той функціонал, що їм справді необхідний, оминаючи усе інше. Фреймворк надає прості можливості до підключення проміжних обробників, що обробляють запити, завдяки чому з'являється можливість створювати логічно поділену структуру всього застосунку. Це чудовий інструмент для побудови RESTful API, що надає можливість створювати вебсервіси для різноманітних додатків. Через свою мінімальну конфігурацію Express надає можливості до швидкого початку розробки. Наявність великої кількості готових бібліотек, завдяки чому можливість розширювати наявний функціонал додатку стає максимально легкою і доступною. [\[14-16\]](#)

Підводячи підсумки, Express.js – потужний, простий, та швидкий у реалізації фреймворк одночасно, завдяки чому розробники можуть зручно і швидко розробити необхідні серверні застосунки, REST API та інші мережеві сервіси на базі Node.js.

React це бібліотека JS створена для розробки інтерфейсів користувача, розроблена компанією Meta. Її функціонал дозволяє будувати інтерактивні, динамічні і найголовніше, масштабовані вебдодатки розділяючи інтерфейс на компоненти. Переваги React виявляються у багатьох аспектах, наприклад компонента архітектура, Virtual DOM, наявність готових бібліотек. Кожна частина інтерфейсу реалізується як окремий компонент, завдяки чому

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

відкривається можливість повторного використання коду, без його дублювання. React використовує віртуальний DOM для оновлення інтерфейсу, завдяки чому покращується продуктивність роботи готового додатку. Завдяки своїй популярності, React вже містить в собі багато різних корисних бібліотек, що спрощують написання коду та допомагають швидше реалізовувати потрібні рішення. Завдяки цим перевагам, React надає можливість створення швидких, адаптивних і легко-масштабованих веб-інтерфейсів, які разом складатимуть повноцінну майбутню платформу.[17-19]

Node.js представляє собою середовище виконання JavaScript на сервері. Дозволяє запускати усі необхідні процеси поза браузером, а також створювати серверну логіку тією ж мовою, що була використана для сторони клієнта. До переваг Node.js можна віднести асинхронність, швидкодію, масштабованість. Завдяки асинхронності, що зараз є край необхідним для будь-якого додатку, забезпечує підтримку одночасного під'єднання великої кількості клієнтів. Завдяки V8-рушію Node.js виконує код дуже швидко. React легко розширюється, що дозволяє нам легко і зручно будувати як невеликі проекти, так і масштабні додатки, що вимагають багато функціоналу. [20]

MongoDB це документоорієнтована NoSQL база даних зберігає інформацію в форматі BSON (розширений JSON). Її гнучкість у роботі з напівструктурованими даними робить її чудовим вибором для веб-додатків та стартапів. Щодо переваг цієї БД можна виділити гнучкість структури даних, простоту інтеграції та продуктивність цієї бази даних. Завдяки відсутності жорсткої схеми даних надається можливість легкої зміни структури даних додатку без потреб у додаткових міграціях. Легко інтегрується в код з допомогою однієї бібліотеки та не вимагає приділення особливої уваги до себе. MongoDB оптимізована для швидкісних операцій запису і читання.[21]

Висновки до першого розділу.

Було проведено аналіз, під час якого оглянули предметну область, змогли визначити головні вимоги до майбутньої краудфандингової благодійної платформи, що як головну ціль, несе в собі завдання створити платформу для

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

організації та збору коштів для благодійних подій. Протягом аналізу дослідили схожі за своєю суттю та ціллю платформи, що використовують в собі різні способи і принципи збору коштів та загальної організації подій. Оглянувши їх функціонал та принципи, було сформовано необхідний функціонал та методи збору коштів для власної платформи. Відповідно до вимог нашої платформи сформували список головних засобів і технологій, які будуть використані для реалізації платформи.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПЛАТФОРМИ ДЛЯ ОРГАНІЗАЦІЇ БЛАГОДІЙНИХ ПОДІЙ

2.1 Визначення варіантів використання та структури системи

Благодійна платформа пропонує веб-інтерфейс, за допомогою якого користувачі можуть створювати заходи, долучатися як волонтери, інвестувати гроші та спілкуватися між собою. Функціональність системи залежить від активності її користувачів, які мають змогу вибрати одну з доступних ролей, а саме організатора, волонтера або інвестора та виконувати дії, властиві кожній з них. Саме такий поділ користувачів та функціоналу для них обраний оптимальним та буде опрацьований в цьому розділі. Логічне розділення функціоналу для кожної категорії забезпечує зберігання простоти інтерфейсу для усіх користувачів. Таким чином волонтер не буде бачити лишніх кнопок і профілів, які буде бачити організатор і навпаки. Так само і інвестор не буде бачити усього функціоналу який його не цікавить. Завдяки такому підходу платформа залишиться логічно сформованою та оптимальною для використання.

Користувачі платформи розподіляються на декілька категорій: організатори, що створюють благодійні події, керують заявками волонтерів та підтримують опис та фінансову інформацію про проєкт, волонтери, які можуть подавати заявки на участь у заходах, залишати або отримувати відгуки, інвестори, що мають змогу підтримати події фінансово та контролювати власні внески. Кожен захід містить інформацію про його ціль, дату проведення, опис, місцезнаходження, необхідну суму збору коштів, галерею зображень тощо. Це дає можливість залучити більше користувачів до участі або підтримки ініціативи. Для наочності та кращого розуміння взаємодії між користувачами та компонентами системи було розроблено діаграму варіантів використання (рис. 2.1), що показує функціональні можливості платформи відповідно до ролі користувача.[\[22\]](#)

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

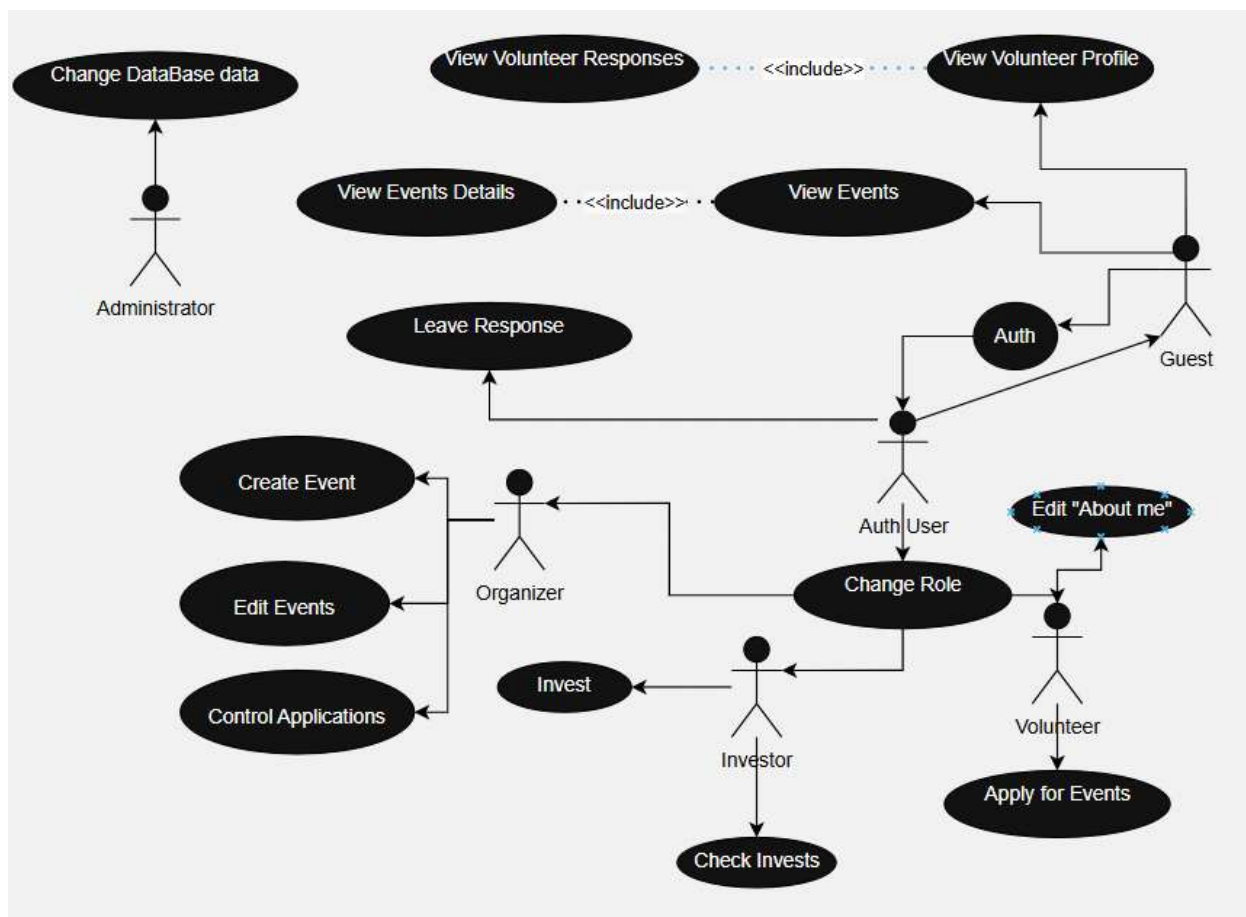


Рис. 2.1 Діаграма варіантів використання

Опис діаграми використання наведено у таблицях 2.1 - 2.11

Таблиця 2.1

Опис прецеденту «Вхід/Реєстрація»

Назва варіанту використання	Авторизація / Реєстрація
Основні актори	Гість
Короткий опис	Користувачі можуть зареєструватися або увійти в систему для отримання доступу до розширених можливостей платформи, включаючи зміну ролі, створення подій, участь у заходах або інвестування.

Таблиця 2.2

Опис прецеденту «Зміна ролі»

Назва варіанту використання	Зміна ролі
Основні актори	Авторизований користувач
Короткий опис	Після входу в систему користувач може змінити активну роль (Організатор, Волонтер, Інвестор), щоб отримати доступ до відповідної функціональності.

Таблиця 2.3

Опис прецеденту «Створення та редагування події»

Назва варіанту використання	Створення та редагування події
Основні актори	Організатор
Короткий опис	Організатор може створювати благодійні події, редагувати їхній вміст, оновлювати медіа та деталі події.

Таблиця 2.4

Опис прецеденту «Подання заявки волонтером»

Назва варіанту використання	Подання заявки волонтером
Основні актори	Волонтер
Короткий опис	Волонтери можуть подавати заявки на участь у заходах, які будуть розглядатись організаторами.

Таблиця 2.5

Опис прецеденту «Редагування «Про себе»»

Назва варіанту використання	Редагування "Про себе"
Основні актори	Волонтер
Короткий опис	Волонтер має змогу додати або редагувати власний опис (біо), який відображається в його профілі.

Таблиця 2.6

Опис прецеденту «Перегляд подій і деталей»

Назва варіанту використання	Перегляд подій та деталей
Основні актори	Гість, Авторизований користувач
Короткий опис	Користувачі можуть переглядати перелік подій, їх описи, зображення, дати та інформацію про зібрані кошти.

Таблиця 2.7

Опис прецеденту «Інвестування у подію»

Назва варіанту використання	Інвестування у подію
Основні актори	Інвестор
Короткий опис	Інвестори можуть фінансово підтримувати події через форму оплати та відстежувати власні інвестиції.

Таблиця 2.8

Опис прецеденту «Залишити відгук»

Назва варіанту використання	Залишити відгук
Основні актори	Організатор
Короткий опис	Організатор має можливість залишити відгук про волонтера після завершення події.

Таблиця 2.9

Опис прецеденту «Залишити відгук»

Назва варіанту використання	Залишити відгук
Основні актори	Волонтер
Короткий опис	Волонтер може переглядати свій профіль, біографію та отримані відгуки.

Таблиця 2.10

Опис прецеденту «Керування заявками волонтерів»

Назва варіанту використання	Керування заявками волонтерів
Основні актори	Організатор
Короткий опис	Організатор може переглядати подані заявки волонтерів до власних подій, приймати або відхиляти їх.

Таблиця 2.11

Опис прецеденту «Адміністрування бази даних»

Назва варіанту використання	Адміністрування бази даних
Основні актори	Адміністратор

Короткий опис	Адміністратор має можливість вносити зміни до бази даних платформи для забезпечення її коректної роботи.
---------------	--

Головна ціль, якої має досягнути розробка платформи, полягає у створенні онлайн-середовища, через яке зможуть ефективно взаємодіяти один з одним волонтери, інвестори і організатори таких подій. Напрямок розробки направлено на реалізацію зручного інтерфейсу, зрозумілості і простоти користування додатком.

Метою ж розробки є допомогти створювати, поширювати та зробити доступнішими різноманітні благодійні цілі через набір працівників і збір коштів для благодійних подій. Організаторам надається можливість розписати, що їм необхідно для реалізації їхнього заходу, які ресурси вони потребують та чого намагаються досягнути. Люди, що не мають коштів, але хочуть допомогти для інших можуть виступити волонтерами, а платформа надасть їм можливість знайти подію, яка підійде для них. Ті, хто можуть підтримати такі ініціативи фінансово, отримують для себе засоби щодо реалізації цього бажання. Завдяки цим аспектам платформа створює місце, що об'єднує різних людей для допомоги суспільству. Опишемо детальніше вимоги користувачів, що будуть користуватись платформою, а саме адміністратора, клієнтів, організаторів, волонтерів та інвесторів.

Адміністратор потребує доступ до даних бази даних для керування подіями і користувачами, що є на платформі. Клієнт потребує такі функції як авторизація, перегляд подій, перегляд профілів користувачів, редагування власного профілю, додавання відгуків про волонтерів. Серед вимог організатора можна виділити створення, редагування, видалення подій, керування заявками волонтерів, огляд статистики зібраних коштів. Волонтерам потрібно мати можливість редагування інформації профілю

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

волонтера, подавати заявки на приєднання до подій, доступ до статусу поданих заяв. Інвесторам необхідно задовольнити вимоги доступу до пожертвування коштів, доступу до особистої інформації інвестора для контролю своїх дій на платформі.

Формуючи вимоги до системи, потрібно виділити функціональні та нефункціональні вимоги. Формуючи функціональні вимоги, можна сказати що система має містити можливість авторизації звичним усім способом, забезпечити безпеку даних користувачів, відображення подій, можливість вибору ролей і адаптивного профілю, забезпечити організаторів подій доступом до заявок волонтерів, забезпечити механіку створення подій організаторами, процес інвестицій коштів для інвесторів. Складаючи нефункціональні вимоги, потрібно описати час, за який система буде відповідати на поставлене їй завдання. Отож час, за який користувач освоює сайт має складати 3-4 хвилини, досвідчені користувачі потребують 1-2 хвилини. Час, за який вебсайт відповідає на дії користувача не більше 2-х секунд. Складні операції мають оброблятися не довше 10-ти секунд. Інтерфейс вебсайту має бути зрозумілим, без зайвих елементів і зручним для користування. Дані користувачів мають бути захищені. Сайт має бути доступним постійно без перебоїв. Сам вебсайт повинен витримувати навантаження великої кількості користувачів. Далі важливо описати системні вимоги, тобто характеристики пристроїв, яким вони мають відповідати для успішного запуску платформи у себе. Вимоги до середовища виконання вимагають для повноцінного і безпроблемного запуску платформи, пристрій який має задовільняти наступні вимоги: 2 Гб оперативної пам'яті, процесор з тактовою частотою не нижче 1.8 Ghz, вихід в мережу інтернет. Для того щоб виконувати усі базові функції серверу, пристрій має мати наступні мінімальні характеристики: процесор з тактовою частотою не нижче 1.4 Ghz, оперативна пам'ять 2Гб, місце на диску: 10Гб.[23]

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Завдяки аналізу функціональних вимог ми змогли виділити наступні сутності, які дозволять реалізувати додаток. Рисунок 2.2 наведе Діаграму класів, які будуть використані для роботи додатку.[24]

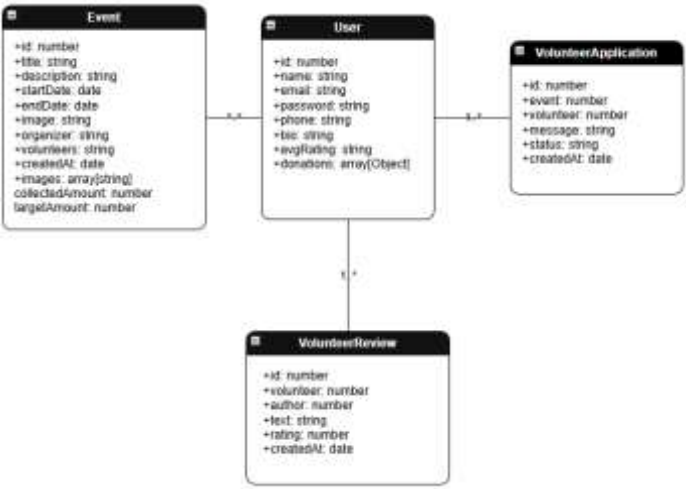


Рис.2.2 Діаграма класів

Опишемо ж діаграму класів детальніше у таблиці 2.12 (див. Рис. 2.2)

Таблиця 2.12

Опис діаграми класів

Назва	Опис
User	Центральний клас, відповідальний за дані користувача і взаємодіє з усіма іншими класами.
Event	Клас, що займається зберіганням даних подій.
VolunteerApplication	Зберігання даних заявок волонтерів.
VolunteerReview	Відгуки про волонтерів.

Також для обробки даних, що зберігаються в цих класах серверна частина додатку містить контролери, оглянемо ж їх головні методи, дослідивши таблицю 2.13.

Опис методів і контроллерів

Назва	Опис
AuthController	обробка даних користувача
getMe()	отримання даних про користувача
updateProfile()	оновлення даних користувача
uploadUserPhoto()	завантаження аватару користувача
register()	реєстрація нового користувача
login()	вхід користувача до платформи
logout()	вихід користувача з системи
EventController	CRUD-операції над класом Event
createEvent()	створити нову подію
getAllEvents()	отримати всі події, що є на платформі
getEventById()	пошук конкретної події по id
updateEvent()	оновити інформацію події
deleteEvent()	видалити подію за її id
PaymentController	обробка процесу пожертвувань
donate()	додавання коштів з пожертви
getMyDonations()	пожертви конкретного інвестора
VolunteersController	обробка заяв і відгуків волонтерів
createApplication()	створити заяву волонтера
getAllMyApplications()	отримання всіх заявок, що стосуються подій конкретного організатора
getApplications()	отримання заявок по одній події
updateApplicationStatus()	оновлення статусу заявки
getReviews()	отримати відгуки по конкретному волонтеру
createReview()	створити відгук

2.2 Розробка бази даних системи

Для зберігання даних платформи використовується MongoDB, нереляційна база даних. Для зручного зберігання даних системи використовується 4 таблиці. Users – таблиця з даними користувачів. Events – таблиця з інформацією про події. Volunteerapplications – таблиця, що відповідає за зберігання заяв волонтерів. Volunteerreviews – таблиця, що зберігає відгуки користувачів. Детальний опис таблиць наведено у таблицях 2.14-2.17.[\[25\]](#)

Таблиця 2.14

Опис таблиці Users

Поле	Тип	Призначення
id	number	Ідентифікатор
name	string	Ім'я користувача
email	string	Електронна пошта
password	string	Хешований пароль
phone	string	Номер телефону
bio	string	Опис користувача
avgRating	string	середній рейтинг
donations	array[Object]	Список пожертв

Таблиця 2.15

Опис таблиці Events

Поле	Тип	Призначення
id	number	Ідентифікатор
title	string	Назва події
description	string	Опис
startDate	date	Дата початку
endDate	date	Дата завершення

Продовження таблиці 2.15

image	string	Посилання на зображення події
organizer	string	Id організатора
volunteers	array[string]	Масив id волонтерів
createdAt	date	Дата створення
images	array[string]	Масив посилань на зображення в галерею
collectedAmount	number	Зібрані кошти
targetAmount	number	Ціль збору

Таблиця 2.16

Опис таблиці VolunteerApplication

Поле	Тип	Призначення
id	number	Ідентифікатор
event	string	Id події
volunteer	string	Id волонтера
message	string	Повідомлення до заяви
status	string	Статус заявки
createdAt	date	Дата створення

Таблиця 2.17

Опис таблиці VolunteerReviews

Поле	Тип	Призначення
id	number	Ідентифікатор
volunteer	string	Id волонтера
author	number	Id автора

text	string	Текст відгуку
rating	number	Оцінка
createdAt	date	Дата створення

2.3 Проектування та реалізація алгоритмів роботи

Платформа проста у своєму використанні, містить основні і найважливіші функції, завдяки чому користувачам легше взаємодіяти з платформою. Рисунок 2.3 показує алгоритм роботи платформи, також нижче описано детальніше його пункти.

Зайшовши на головну сторінку платформи, користувачу надається можливість ввійти в наявний або ж зареєструвати новий аккаунт. Після успішного входу до аккаунту платформи функціонал платформи, які заплановано було раніше, розподіляється на декілька варіантів залежно від того, з якими ролями користувач ввійшов на платформу.

Якщо авторизувався Організатор, він може створювати події та керувати поданими заявами. Тобто надаються можливості створення подій, редагування їх інформації, контролю збору коштів через профіль організатора. На цьому головний функціонал для організатора завершується.

Якщо авторизувався Волонтер, він може подавати заявки на події, додаючи в заявки інформацію про себе та причини, чому саме його потрібно взяти та яку користь він може принести цій події. Далі для відслідковування того, чи прийняли його заявку волонтер може знайти потрібну йому подію та оглянути список прийнятих волонтерів.

Якщо авторизувався Інвестор, то він може робити внески та оглянути доступ до статистики його внесків. Будь-який авторизований користувач має доступ до залишання відгуків, незалежно від його ролі. Таким чином платформа зможе виконати усі необхідні від неї вимоги її користувачів, забезпечивши їх зручним у користуванні інтерфейсом.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

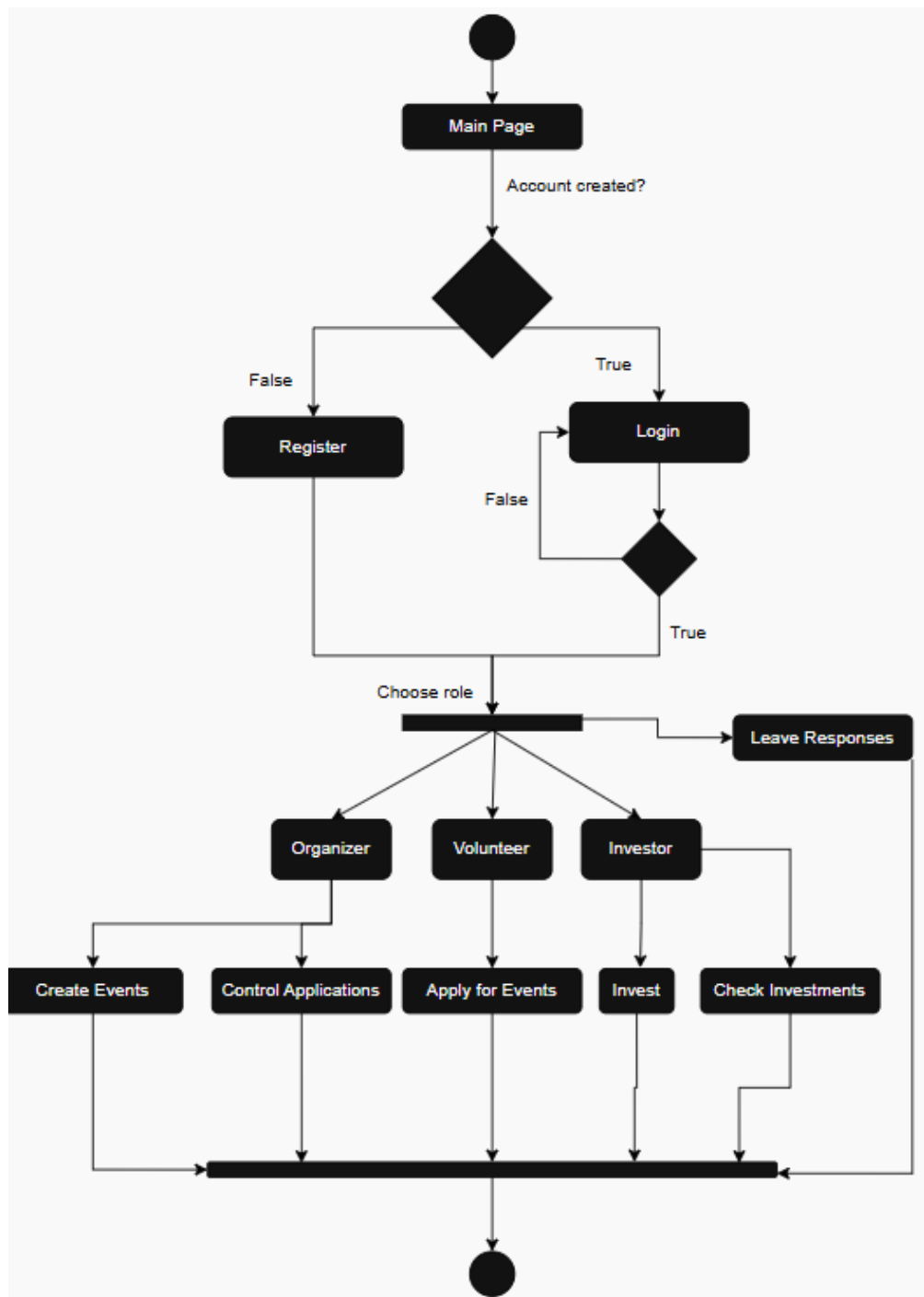


Рис. 2.3. Діаграма активності

Дослідивши детальніше кожен пункт діаграми, можна зробити висновок що платформа задовільняє поставлені до неї вимоги, які вона повинна дотримувати. Тому далі можна переходити до реалізації головного функціоналу платформи, реалізуючи поетапно головні частини платформи одна за одною, допоки не складеться готова платформа для організації благодійних подій.

2.4 Реалізація функціоналу платформи для організації подій

Розглянемо основні етапи створення збору на платформі та головні дії, що виконуються над ним. Першочергово користувач після входу в аккаунт, має створити цей збір. Для цього є функція createEvent(Рис. 2.4).

```
exports.createEvent = async (req, res) => {
  try {
    const event = new Event(req.body);
    await event.save();
    res.status(201).json(event);
  } catch (err) {
    console.error('Помилка при створенні події:', err);
    res.status(400).json({ error: err.message });
  }
};
```

Рис.2.4. функція createEvent

Після цього волонтери можуть подавати заявки на приєднання до події, для чого реалізовано наступну функцію createApplication(Рис.2.5). Ця функція отримує дані про подію, на яку робиться заявка, дані волонтера, створює об'єкт заявки, де передає інформацію про волонтера, подію та повідомлення самого волонтера, яке він написав у заяві.

```
exports.createApplication = async (req, res) => {
  try {
    const { eventId, message } = req.body;
    const volunteerId = req.user._id;

    if (!eventId || !message) {
      return res.status(400).json({ message: 'eventId та message обов'язкові' });
    }

    // Перевірити чи вже є подана заявка на цю подію
    const existing = await VolunteerApplication.findOne({ event: eventId, volunteer: volunteerId });
    if (existing) {
      return res.status(400).json({ message: 'Заявка вже подана' });
    }

    const application = new VolunteerApplication({
      event: eventId,
      volunteer: volunteerId,
      message,
    });
    await application.save();

    res.status(201).json(application);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Помилка сервера' });
  }
};
```

Рис.2.5. createApplication

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

Інвестори в свою чергу мають робити внески на подію. В умовах виконання Кваліфікаційної роботи, реалізація повноцінної системи збору коштів через такі сервіси, як LiqPay є неможливою, адже вимагає наявності ФОП або зареєстрованої Релігійної/Благодійної організації. Тому на час виконання кваліфікаційної роботи було реалізовано аналог системи внесення коштів, а функція, відповідальна за внески зображена на Рисунку 2.6[26-27]

```
exports.donate = async (req, res) => {
  try {
    const { amount } = req.body;
    const { id } = req.params;
    const userId = req.user._id;

    if (!amount || amount <= 0)
      return res.status(400).json({ message: 'Некоректна сума' });
    const event = await Event.findById(id);
    if (!event) return res.status(404).json({ message: 'Подія не знайдена' });
    event.collectedAmount += Number(amount);
    await event.save();
    await User.findByIdAndUpdate(
      userId,
      { $push: { donations: { event: id, amount } } },
      { new: true }
    );
    res.json({
      message: 'Оплата успішна',
      raisedAmount: event.raisedAmount,
      targetAmount: event.targetAmount,
    });
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Помилка сервера' });
  }
};
```

Рис.2.6. Реалізація функції внесків

Після чого, організатор має доступ до аналітики збору фінансів через свій профіль Організатора, а також може керувати статусами заявок волонтерів: відхиляючи їх або приймаючи. Для цього реалізовано ряд важливих функцій, що детальніше описані у таблиці 2.18. Кожна функція подана зі своїм описом, щоб було зрозуміліше сутність того, які дії може робити організатор з подіями.

Опис головних функцій платформи

Назва	Опис
Отримати всі події	Запит повертає усі події для відображення. Може опрацювати id організатора або події, щоб вивести усі події організатора, або певну конкретну подію
Отримати заявки	Функція повертає усі заявки волонтерів, що стосуються подій одного конкретного організатора, чим надає доступ до керування заявками волонтерів
Публікація відгуку	Функція створює відгук про конкретного волонтера
Оновлення профілю	Використовується для оновлення профілю користувача

На основі отриманих даних, було створено діаграму компонентів платформи для організації благодійних подій. Рисунок 2.7 зображає повністю цю діаграму з усіма її компонентами.

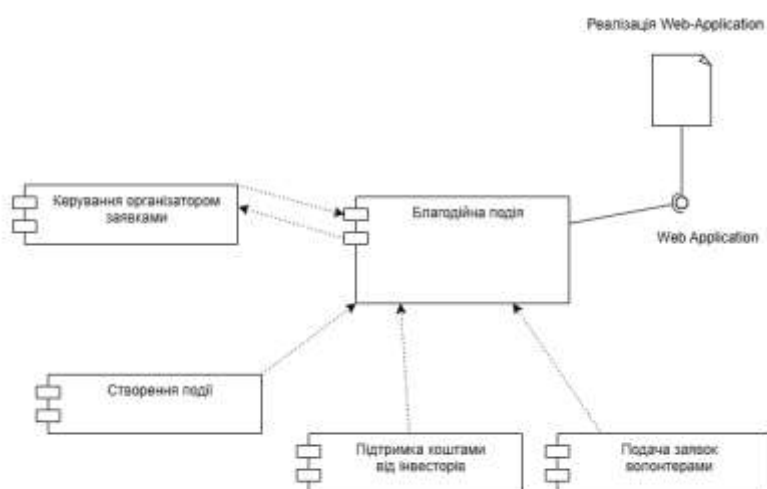


Рис.2.7. Діаграма компонентів

На цій діаграмі зображено наступні елементи: благодійна подія – компонент події. Створення події – процес створення події організатором. Підтримка коштами від інвесторів – процес передачі коштів від інвесторів до організаторів з допомогою сервісу пожертв. Подача заявок волонтерами – волонтери надсилають запити на участь у події з допомогою відповідної форми. Керування організатором заявками – двосторонній процес обміну даними організатора і події, через який модеруються заявки волонтерів.

Висновки до другого розділу

В процесі проектування платформи для організації благодійних подій було розглянуто усі варіанти як користувачі можуть використовувати готову платформу, а також на основі програмного забезпечення. На основі отриманих даних спроектовано структуру платформи, що включало в собі моделювання структури проекту і його взаємодію між складовими частинами.

Також спроектували базу даних, яка буде зберігати дані платформи, а саме користувачі, події, інформація про пожертвування і відгуки користувачів.

Описали алгоритм роботи з додатком, описали послідовні пункти взаємодії користувача з платформою, як він її може використати.

В кінці наведено було опис головного функціоналу додатку – взаємодії користувачів з подією. А саме його створення, подача заявок волонтерів, їх модерація і пожертви інвесторів.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ДЛЯ ОРГАНІЗАЦІЇ ПОДІЙ

3.1 Порядок встановлення та налаштування параметрів платформи

Процес встановлення і налаштування готової платформи вимагає розгортання усіх компонентів та їх налаштування – як кожного окремо, так і між собою. Щоб краще описати архітектуру готової платформи було створено діаграму розгортання.[\[28\]](#)

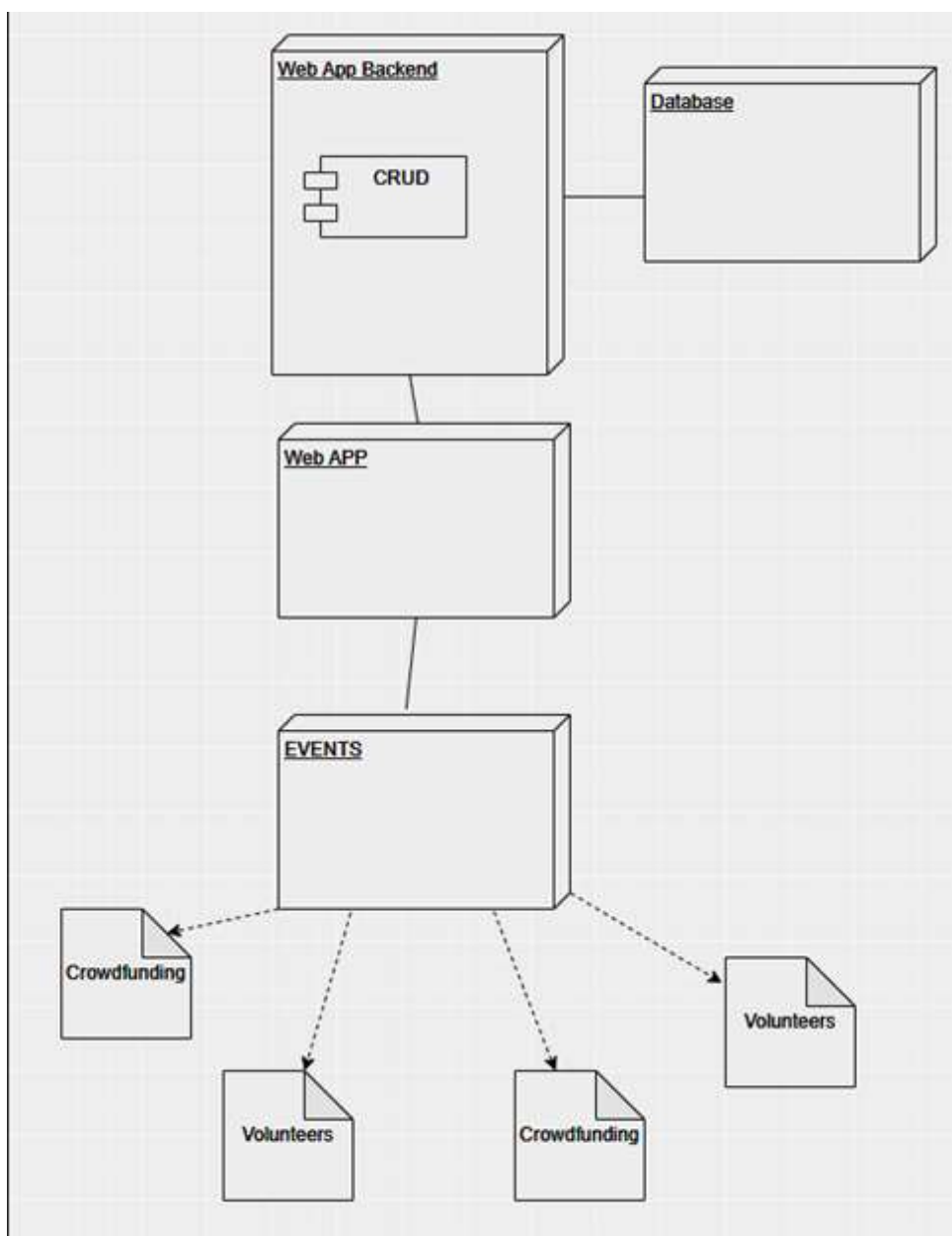


Рис.3.1. Діаграма розгортання платформи

Опишемо весь процес розгортання детальніше. Database це база даних, що відповідає за зберігання даних платформи. В ній буде зберігатись абсолютно вся інформація щодо користувачів, подій, заявок та історія внесків інвесторів. Web app backend зображує RESTful API, що приймає запити платформи, обробляє їх, взаємодіє з БД та повертає відповідь у додаток. Це як ядро системи, що поєднує між собою користувацький інтерфейс та Базу Даних, де зберігається вся інформація. Web APP уособлює саму платформу, через яку користувачі взаємодіють з усіма компонентами. Тобто весь користувацький інтерфейс, який при взаємодії з користувачем звертається до бекенду з запитами, а далі обробляє отриману відповідь та виводить необхідну інформацію. EVENTS це компонент подій, що є центром усіх аспектів платформи. Тобто це елемент, що містить в собі зв'язок між інвесторами, волонтерами та організаторами. Центр логіки платформи, навколо якого будується вся система. Crowdfunding показує збір коштів з інвесторів, що роблять внески у розвиток подій. В платформі на даний момент це шаблон, що буде заміщений на повноцінну систему прийому платежів LiqPay. Volunteers це волонтери, що подають заявки на вступ до подій. Взаємодіють разом з системою через головний компонент EVENTS, подаючи заявки на вступ до подій, отримуючи відповіді від організаторів. Кожен елемент є важливим у розгортанні платформи і без якогось одного платформа не зможе нормально функціонувати. Саме тому потрібно було приділити увагу для схеми розгортання платформи в цілому.

Таким чином було зображено загальну схему розгортання платформи та її процес роботи, зв'язки між компонентами та сутність кожного елемента схеми. В загальному вони складають повноцінну платформу, що відповідає усім вимогам описаним раніше і надає усі необхідні можливості для її використання.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

3.2 Структура інтерфейсу платформи для організації подій

Для того, щоб користувачі могли скористатись платформою не потрібно нічого довстановлювати або завантажувати, вся взаємодія відбувається через їх браузер. Зайшовши на платформу користувача чекатиме головна сторінка, де він може ознайомитись з суттю платформи і побачити, скільки вже користувачів зареєстровано на платформі.

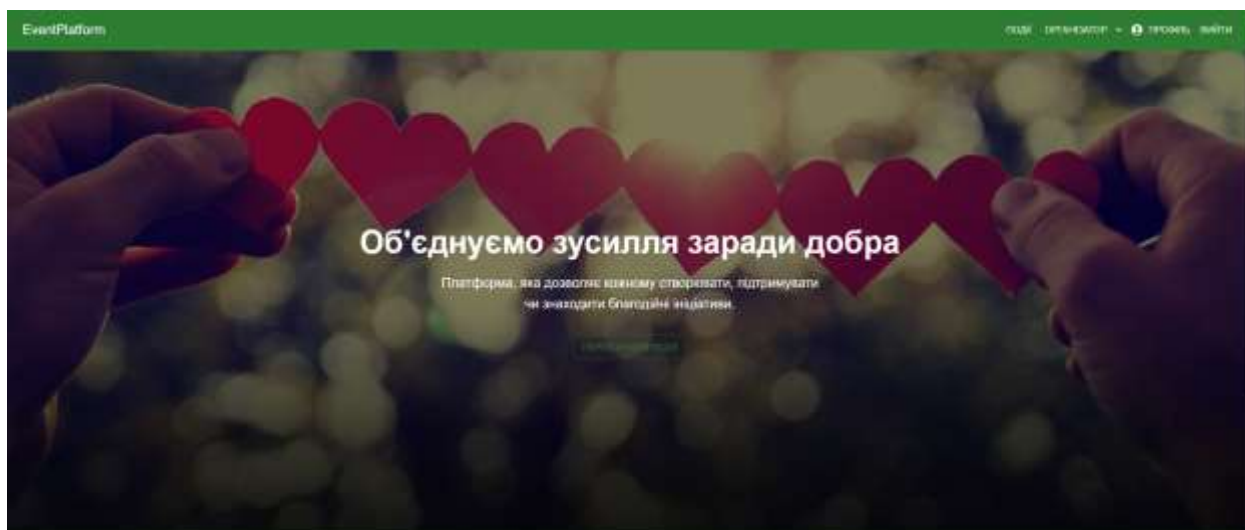


Рис.3.2. Головна сторінка 1

Прогорнувши донизу, платформа ознайомлює користувача з її суттю – для кого вона була створена. Нижче відображається кількість користувачів, що вже використовують платформу та заклик до реєстрації.

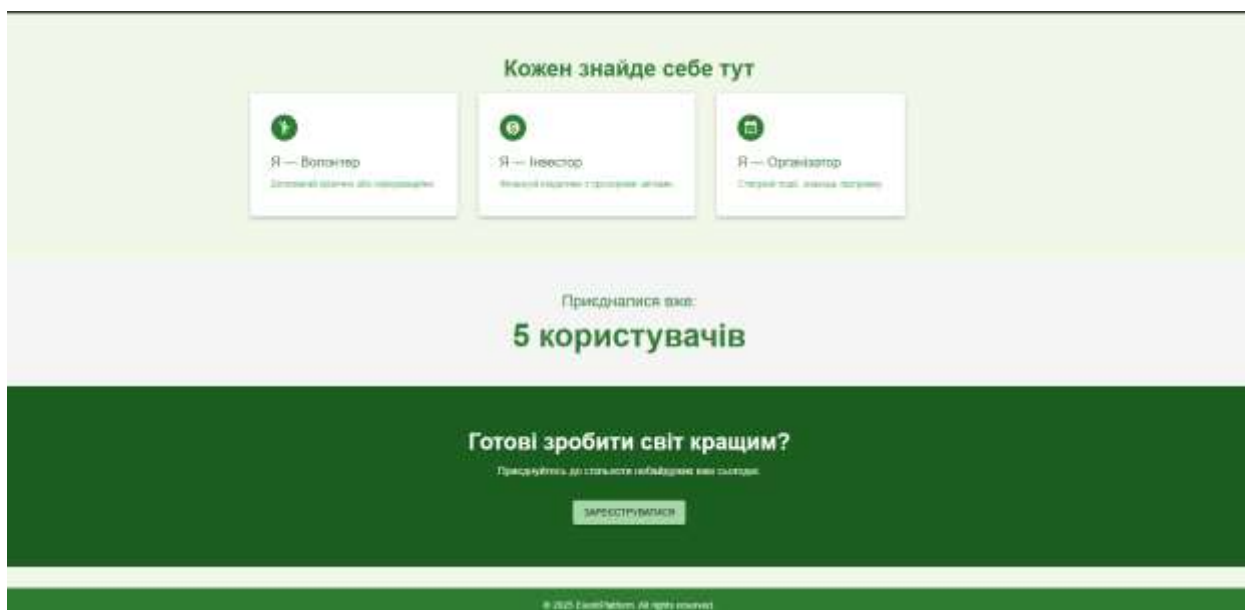


Рис.3.3. Головна сторінка 2

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

Далі у користувача постає вибір – зайти ознайомитись з усіма подіями, що зараз є на платформі, або ж перейти до входу і реєстрації сторінки. Розглянемо користувача у ролі організатора подій. Для початку необхідно перейти до реєстрації або входу до вже наявного акаунту. Для проходження усіх етапів створимо нового користувача організатора. Для цього потрібно перейти по кнопці реєстрації у навігаційній панелі.

Рис.3.4. Реєстрація

Ввівши всі необхідні дані користувача переносить до його профілю(Рис. 3.5).

Рис.3.5. Профіль користувача

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

Перейдемо до процесу створення нової події. Клікнувши по відповідній кнопці нас переносить на сторінку створення подій. Відразу зауважимо, що при відсутності необхідної ролі користувача не пропустить до входу на цю сторінку. Ввівши усі необхідні поля, а саме назву, опис події, локацію де проводиться, дати проведення (якщо подія триває один день, є відповідний прапорець), необхідну суму збору а також посилання на зображення, що буде відображатись на сторінці всіх подій. Так як в більшості організаторів своїх подій вже є свої сайти, такий метод був обраний для економії місця на БД, а також для пришвидшення роботи додатку. Після натискання на кнопку, при умові правильного введення всіх даних, подія створюється і користувача переносить на сторінку з усіма подіями. Рисунки 3.6 і 3.7 покажуть ці 2 сторінки.

Рис.3.6. Створення події

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

Після успішного створення події, тобто за умови відсутності отриманих помилок, користувач бачить сторінку з усіма подіями, як і говорилося раніше. Тут через пошук він може знайти свою подію та оглянути інформацію про неї за бажанням. Детальніше про цей процес буде розписано далі.

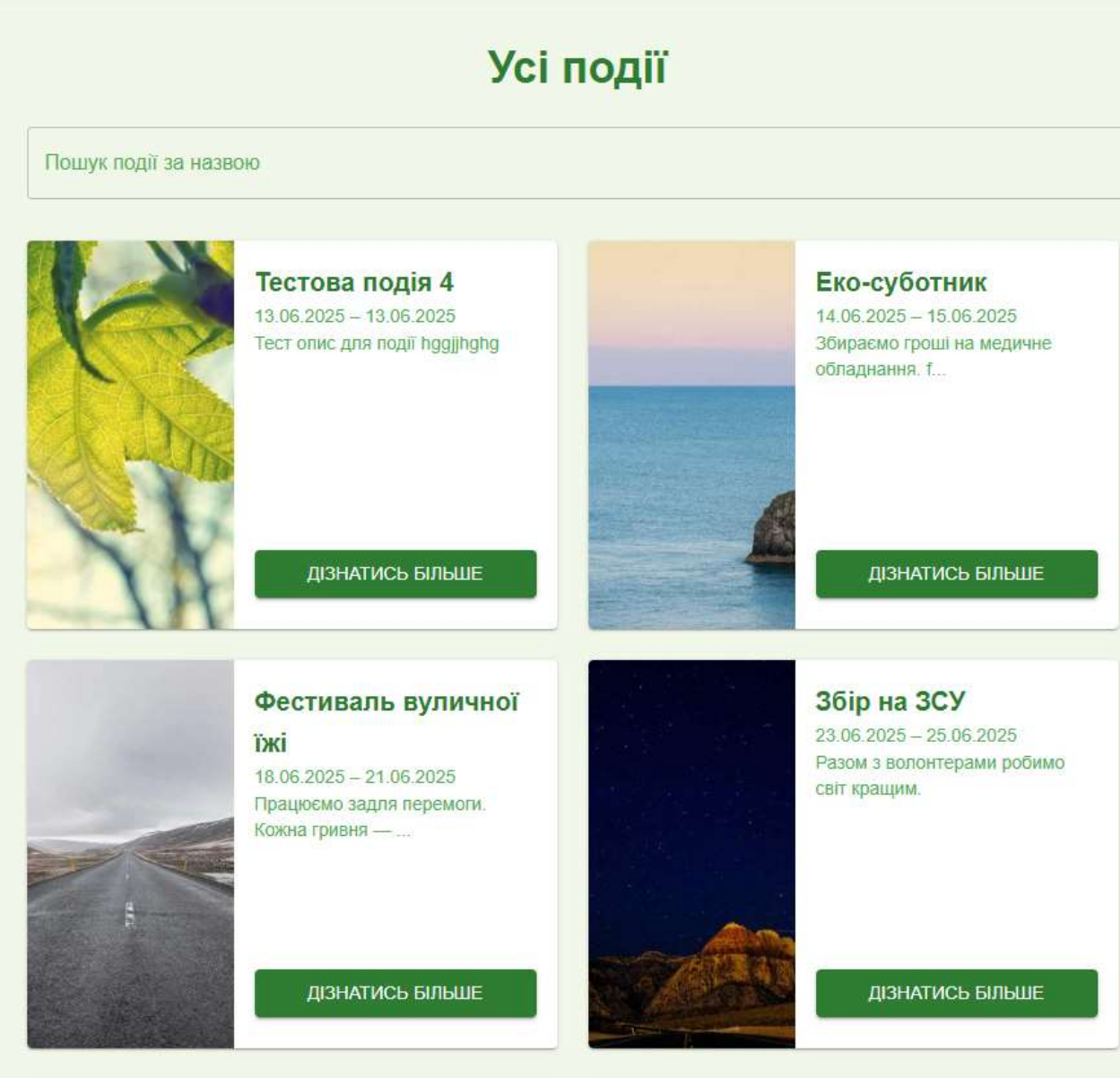


Рис.3.7. Сторінка подій

Далі користувач може ознайомитись з іншими подіями, або оглянути детальніше свою, знайшовши її в списку або з допомогою зручного пошуку. Обравши подію, що його зацікавила, він може побачити детальніше інформацію про цю подію, а саме побачити стан збору, весь опис, прийнятих волонтерів на подію(Рис.3.8).

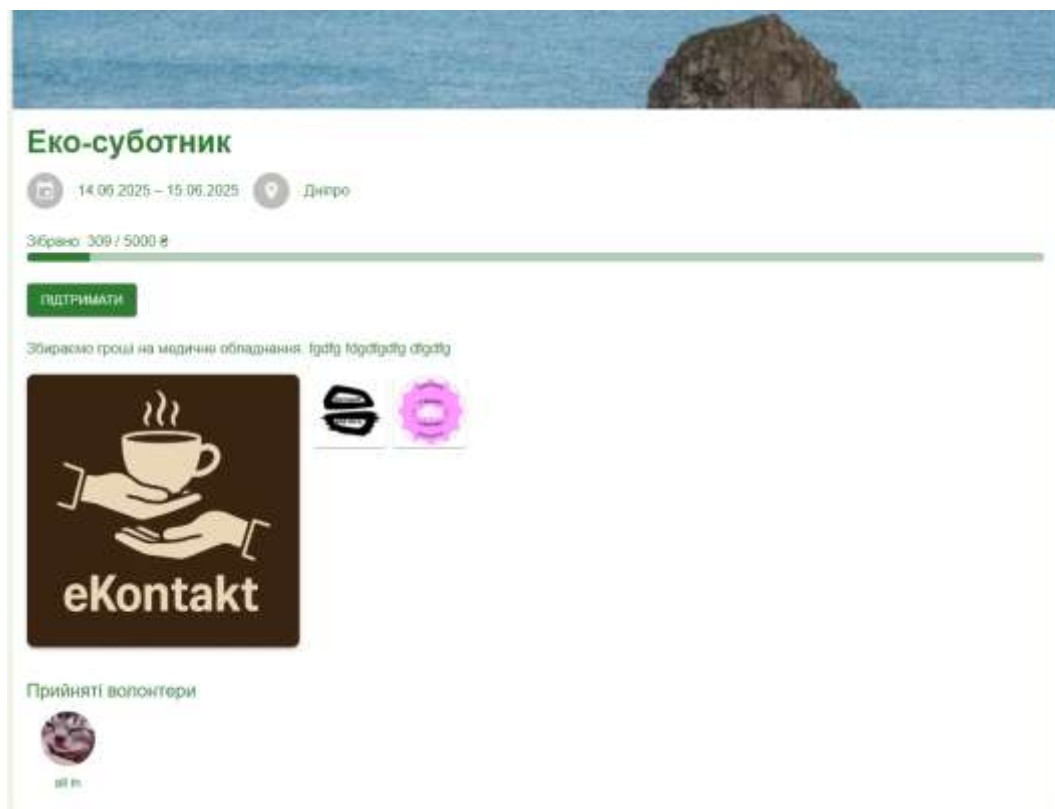


Рис.3.8. Сторінка події

Далі користувач, як організатор може перейти знову до свого профілю для надання додаткової інформації про свої події, побачити зручно стан його зборів та перейти до модерування заявок від волонтерів.

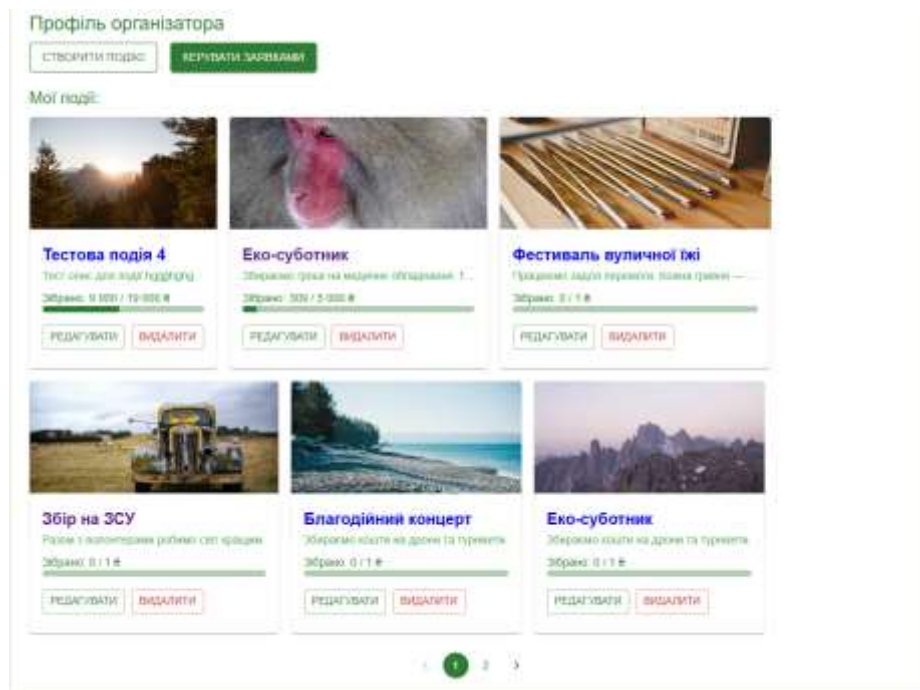


Рис.3.9. Панель подій організатора

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

Перейшовши до редагування події користувач може оновити опис події та додати необхідні фото до його події. Цей процес було винесено в окрему форму для того, щоб під час створення події при умові збоїв не втратити важливу інформацію, саме тому весь цей процес винесено окремо(Рис.3.10).

Рис.3.10. Редагування події

Керування заявками відбувається через таблицю з кнопками Прийняти або відхилити. Початковий статус заявок – очікується відповідь. Після натискання на одну з кнопок статус змінюється на «Прийнято» або «Відхилено». Якщо організатор змінив свою думку про того чи іншого волонтера, то може змінити статус заявки, натиснувши на іншу кнопку.

Рис.3.11. Керування заявками

Також натиснувши на ім'я волонтера відкриється його публічний профіль, де користувачі можуть залишати про нього оцінки. Це допомагає організатору зробити вибір – потрібен йому цей волонтер чи ні. Рисунок 3.12

покаже як виглядає цей профіль і відгуки. Після цього основний функціонал організатора завершується, далі розглянемо функціонал для інвестора.

Рис.3.12. Сторінка волонтера

На рисунку 3.8 можна побачити відповідну кнопку, що відображається користувачам з профілем організатора. Натиснувши на неї відображається меню для проведення оплати інвестором.

Рис.3.13. Меню пожертви

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

Після успішного введення даних і натискання кнопки оплатити, при правильно введених даних користувач отримує відповідне повідомлення і перенаправляється назад на сторінку події. В своєму профілі організатора він може побачити куди і скільки було внесено коштів.

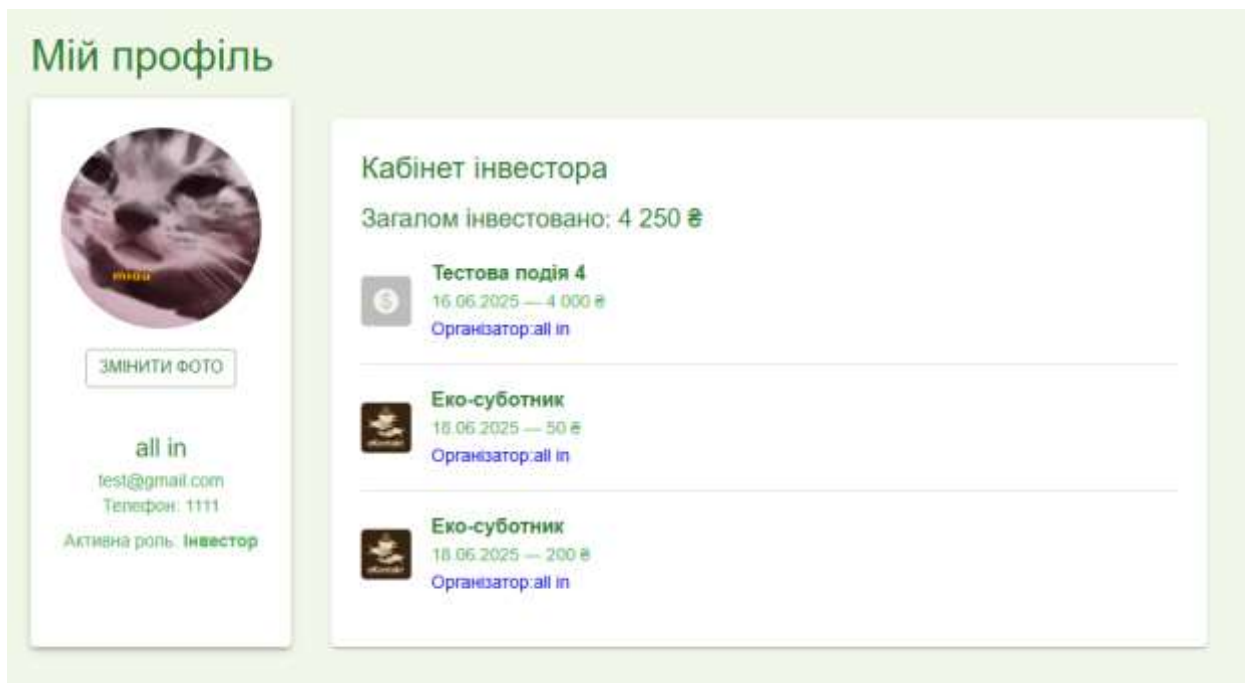


Рис.3.14. Профіль інвестора

Далі оглянемо процес подачі заявки волонтером. Так само на сторінці події з рисунку 3.8, за умови що користувач не є власником події і має роль «Волонтер», йому покажеться відповідна кнопка «Подати заяву». На рисунку 3.15 буде зображено вигляд сторінки подання заявки. Як і раніше, доступ до цієї сторінки є лише у користувачів, які мають відповідну роль.

Рис.3.15. Заявка волонтера

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

Автоматично до неї підтягуються дані користувача: ім'я, номер телефону і електронна пошта для зв'язку. А також текстове поле, куди користувач може внести необхідну інформацію, чому його потрібно обрати як волонтера на цю подію.

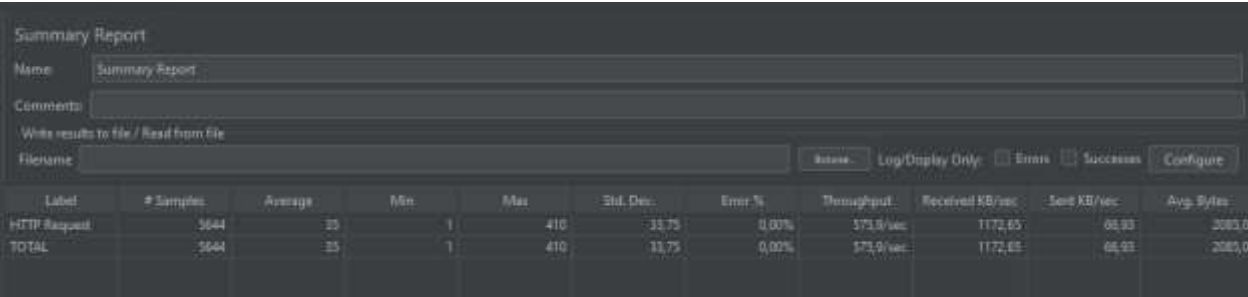
3.3 Тестування платформи

Для кінцевого тестування платформи було використано ПЗ JMeter та Chrome DevTools. Для початку протестуємо нашу платформу навантаженнями. Для цього з допомогою JMeter проведемо тестування системи з наступними параметрами (Рис 3.16). [29-30]



Рис.3.16. Параметри тестування

Далі запускаємо наше тестування і на початку бачимо, що платформа справляється нормально. Перших 5500 запитів пройшли без проблем, 575 запитів в секунду витримала система без помилок. Це хороший знак, можемо рухатись далі, проводячи наступні тестування.



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	5644	35	1	410	33.75	0.00%	573.9/sec	1172.65	66.93	2083.0
TOTAL	5644	35	1	410	33.75	0.00%	573.9/sec	1172.65	66.93	2083.0

Рис.3.17. Початок тестування

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Але на більшій дистанції в тих самих умовах, а саме приблизно 500 запитів за секунду платформа видала 0,22% помилок, що є нормальним. Така кількість помилок при умові стабільного навантаження платформи є навіть більш, ніж задовільною.

Рис.3.18. Кінець тестування

Після цього було вирішено провести більш довге тестування з меншим одночасним навантаженням, завдяки чому побачили, що система здатна довгий час обробляти запити без помилок. На рисунку 3.19 можна ознайомитись з результатами тривалого тестування. Платформа показує себе з хорошої сторони під час кожного тестування, що є добре.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	155000	229	0	7351	301,35	0,11%	140,0/sec	303,48	17,30	2085,4
TOTAL	155000	229	0	7351	301,35	0,11%	140,0/sec	303,48	17,30	2085,4

Рис.3.19. Тестування на протязі часу

На кінець було вирішено запустити останнє тестування на більший проміжок часу з меншою активністю, щоб ще раз перевірити, як платформа реагуватиме на різні кількості користувачів одночасно. Результати тестування просто чудові, 0.03% помилок на 500 тисяч запитів це просто чудовий результат.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput
HTTP Request	512729	137	0	7351	184,85	0,03%	290,1/sec
TOTAL	512729	137	0	7351	184,85	0,03%	290,1/sec

Рис.3.20. Результати тривалого навантаження

Тепер протестуємо платформу з допомогою інструментів від Google DevTools. Воно допоможе нам побачити, як себе веде платформа в різних

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

умовах: під час навантаження і в стандартній роботі. Час завантаження сторінок, швидкість переходу між сторінками та інше. Усі ці дані важливі під час тестування, адже впливають на те, як буде відображатись платформа в пошуковій стрічці, наскільки вона буде приваблива для користувача і приємна в користуванні. Для початку порівняємо показники якості під час навантаження на платформу та без навантажень.

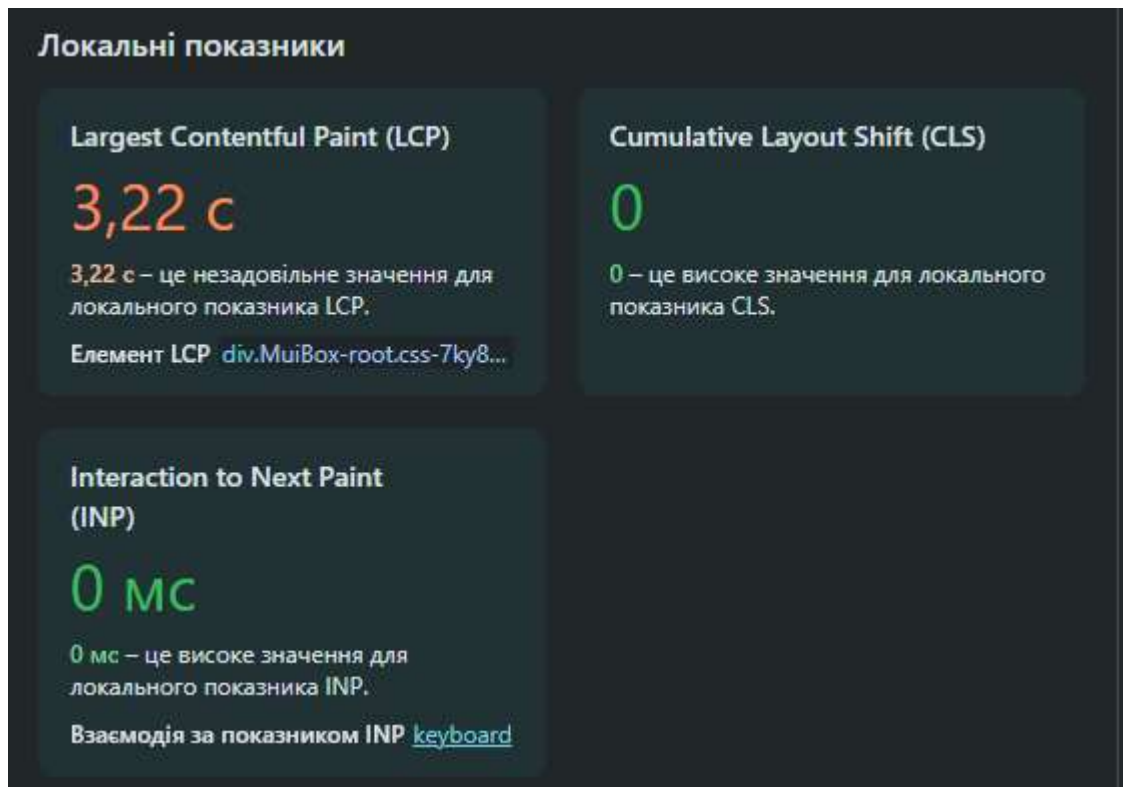


Рис.3.21. Показники головної сторінки під навантаженням

Тепер для продовження збору даних, потрібно користуватись платформою під час отримання нею різних запитів, тоді можна буде отримати кращу загальну картину якості платформи. Чим більше ми використовуємо платформу для переходу між сторінками, взаємодіємо з кнопками на платформі, тим краще статистичні дані від Google опишуть дійсну картину того, яким показникам насправді відповідає платформа. Саме тому було проведено певний час у використанні платформи як реальний користувач. Перехід по подіях, пожертви, повертання на попередню сторінку, профіль, перемикання ролей.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

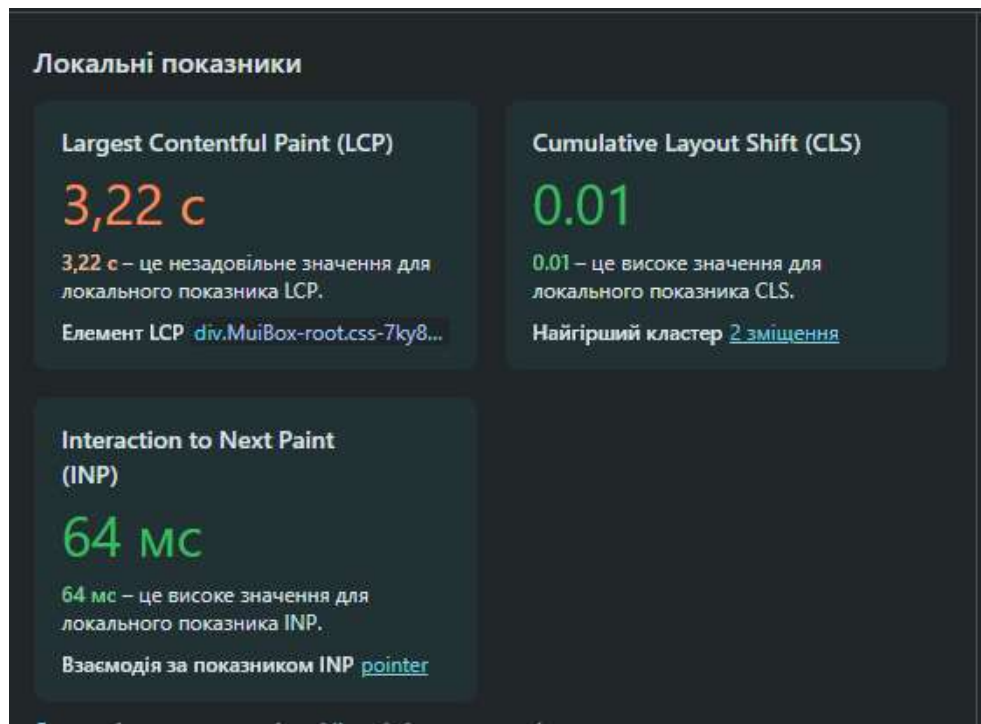


Рис.3.22. Результати після переходу на сторінку подій

Далі було пройдено на сторінку події, звідти на сторінку пожертвувань і проведено інвестицію в подію. Після такого ряду дій показники стали частково незадовільними, а в усьому іншому високими, що є добре(Рис.3.23).



Рис.3.23. Показники після певного ряду дій

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

Продовжуючи перевіряти платформу під навантаженнями неодноразово перевірялись оновлення різних сторінок і в моменти навантажень оновлення сторінки займає дійсно багато часу. Цього нажаль ніяк не уникнути через головний недолік SPA, а саме довший час завантаження сторінки.

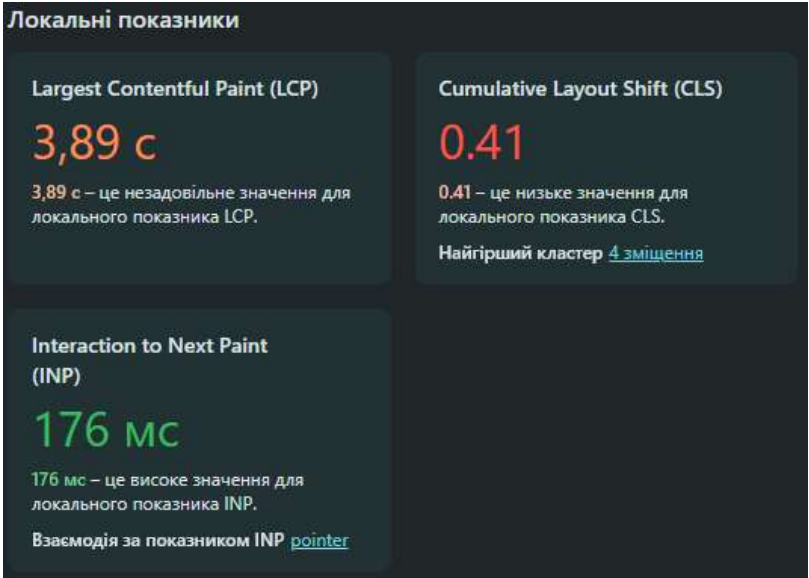


Рис.3.24. Показники після оновлення сторінки подій

Але завдяки використанню SPA платформа має одноразовий процес завантаження сторінки, а далі все відображається в рази швидше, без додаткових промальовувань. Дочекавшись кінця тестування платформи, можна було перейти до повторного тестування швидкості оновлення головної сторінки сайту. Результат виявився більш ніж задовільним.

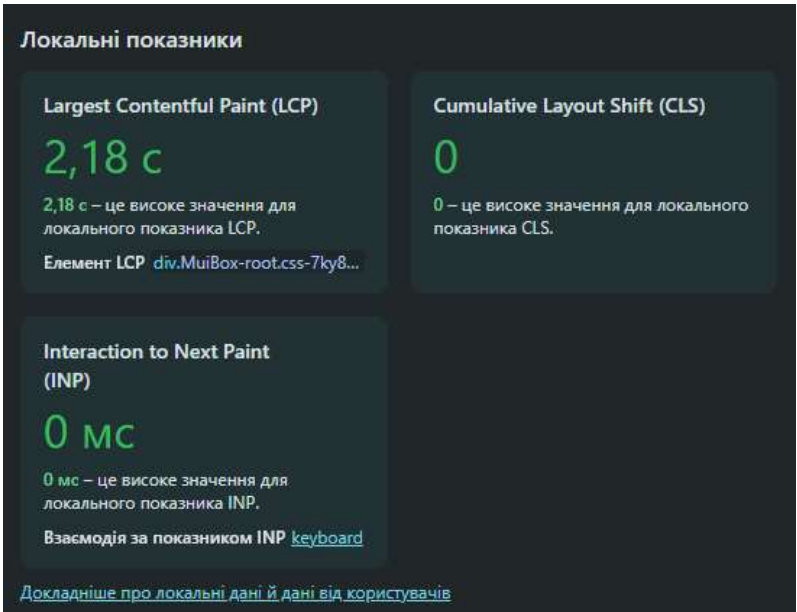


Рис.3.25. Показники без навантаження

Завдяки цим тестуванням було проведено якісний аналіз поведінки платформи під час навантажень, швидкість завантаження сторінок як під час навантаження на платформу, так і без нього. Усі ці дані важливі під час проведення оптимізації платформи та її подальшому розвитку.

Висновки до третього розділу

В цьому розділі було проведено роботу з розгортання платформи, а саме складено схему розгортання, наведено архітектуру готового продукту по компонентах, описано кожен структурний елемент і показано його взаємодію з іншими елементами. Наступним кроком було описання інтерфейсу вже розробленого програмного продукту. Оглянули головні сценарії використання платформи усіма користувачами, які можуть відвідати цю платформу. Кожен користувач має спільні і відмінні особливості користування платформою і кожному з них було пройдено і перевірено. Також оглянули зрозумілість інтерфейсу і на практиці перевірили, чи зручно користуватись платформою. На кінець було проведено важливі тестування для перевірки життєздатності платформи під час навантажень, як великих одночасних, так і стабільних навантажень на протязі довго часу. Після проаналізували якість і швидкість відображення інтерфейсу в різних умовах, а також ще раз підтвердили важливість і зручність SPA на фронтенді.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

ВИСНОВКИ

Виконуючи кваліфікаційну роботу бакалавра було спроектовано та реалізовано платформу для організації благодійних подій, проведено аналіз предметної області і порівняно між собою різноманітні платформи, що займаються схожими напрямленнями. Кожна з них мала свої особливості, спільні риси з іншими, переваги та недоліки. Протягом цього аналізу було виділено найважливіші пункти та розроблено архітектуру, основні підходи та технології для реалізації додатку.

Проектуючи платформу було проведено аналіз усіх варіантів використання платформи і спроектовано структуру, включаючи моделювання взаємодії складових платформи між собою. Після було спроектовано майбутню базу даних для зберігання даних користувачів і системи в цілому.

Далі було описано алгоритм роботи з додатком та пройдено послідовно усі пункти варіантів використання додатку. Оглянуто головний функціонал платформи, як кожен користувач може використати готову платформу, а саме проведення пожертв, створення подій, подача заявок та керування ними зі сторінки організатора.

Наступним кроком було надання детальної схеми розгортання платформи, описано як структурні компоненти платформи взаємодіють між собою і описано весь інтерфейс платформи. Платформа має простий, зручний і інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам легко створювати події, інвестувати в них, подавати заявки на вступ, керувати цими заявками та залишати відгуки про користувачів.

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Війна в Україні – Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Російсько-українська_війна
2. Волонтерство – Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Волонтерство>
3. Caritas Ukraine [Електронний ресурс] – Режим доступу до ресурсу: <https://caritas.ua/>
4. JustGiving [Електронний ресурс] – Режим доступу до ресурсу: <https://www.justgiving.com/>
5. Facebook [Електронний ресурс] – Режим доступу до ресурсу: <https://www.facebook.com/>
6. Twitter [Електронний ресурс] – Режим доступу до ресурсу: <https://twitter.com/>
7. Zeffy [Електронний ресурс] – Режим доступу до ресурсу: <https://www.zeffy.com/>
8. Patreon [Електронний ресурс] – Режим доступу до ресурсу: <https://www.patreon.com/>
9. Frontend Development [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Front-end_web_developer
10. Backend Development [Електронний ресурс] – Режим доступу до ресурсу: <https://roadmap.sh/backend>
11. Single Page Application (SPA) [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>
12. Model-View-Controller (MVC) [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>
13. CRUD Operations [Електронний ресурс] – Режим доступу до ресурсу: <https://restfulapi.net/http-methods/>
14. Express.js [Електронний ресурс] – Режим доступу до ресурсу: <https://expressjs.com/>

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

15. Middleware in Express [Електронний ресурс] – Режим доступу до ресурсу:
<https://expressjs.com/en/guide/using-middleware.html>
16. REST API [Електронний ресурс] – Режим доступу до ресурсу:
<https://restfulapi.net/>
17. ReactJS [Електронний ресурс] – Режим доступу до ресурсу:
<https://react.dev/>
18. Material UI [Електронний ресурс] – Режим доступу до ресурсу:
<https://mui.com/>
19. Virtual DOM [Електронний ресурс] – Режим доступу до ресурсу:
<https://react.dev/learn/rendering-elements#the-virtual-dom>
20. Node.js [Електронний ресурс] – Режим доступу до ресурсу:
<https://nodejs.org/en>
21. MongoDB [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.mongodb.com/>
22. Draw.io (diagrams.net) [Електронний ресурс] – Режим доступу до ресурсу:
<https://app.diagrams.net/>
23. Системні вимоги до серверів [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.ibm.com/docs/en/was/9.0.5?topic=requirements-hardware-operating-system>
24. Класи в об'єктно-орієнтованому програмуванні (ООП) [Електронний ресурс] – Режим доступу до ресурсу:
https://uk.wikipedia.org/wiki/Об'єктно-орієнтоване_програмування
25. Реляційні та нереляційні бази даних [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.mongodb.com/relational-vs-nosql>
26. LiqPay [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.liqpay.ua/>
27. Фізична особа підприємств (ФОП) – Дія [Електронний ресурс] – Режим доступу до ресурсу:
<https://diia.gov.ua/services/reestraciya-fop>

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

28. Архітектура вебсайтів [Електронний ресурс] – Режим доступу до ресурсу:
https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Website_architecture
29. Chrome DevTools [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.chrome.com/docs/devtools/>
30. Apache JMeter [Електронний ресурс] – Режим доступу до ресурсу:
<https://jmeter.apache.org/>

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

ДОДАТКИ

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

```

const mongoose = require('mongoose');

const connectDB = async () => {
  try {
    await mongoose.connect('mongodb://localhost:27017/Charity-platform', {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    });
    console.log('✅ MongoDB connected');
  } catch (error) {
    console.error('❌ MongoDB connection error:', error.message);
    process.exit(1);
  }
};

module.exports = connectDB;

const User = require('../models/User');
const path = require('path');
const jwt = require('jsonwebtoken');

const generateToken = (user) => {
  return jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
    expiresIn: '7d',
  });
};

exports.uploadUserPhoto = async (req, res) => {
  try {
    if (!req.file) {
      return res.status(400).json({ message: 'Фото не завантажено' });
    }

    const user = await User.findById(req.user._id);

    if (!user) {
      return res.status(404).json({ message: 'Користувача не знайдено' });
    }

    // Зберігаємо ім'я файлу в базу
    user.photo = req.file.filename;
    await user.save();

    res.json({
      message: 'Фото оновлено',
      photo: user.photo,
    });
  }
};

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Помилка сервера' });
  }
};

exports.updateProfile = async (req, res) => {
  try {
    const user = await User.findById(req.user._id);

    if (!user) {
      return res.status(404).json({ message: 'Користувача не знайдено' });
    }

    // Оновлюємо поля, які прийшли в req.body (наприклад: name, phone, bio і т.д.)
    // Захищаємо від оновлення email та пароля через цей метод (окрім окремого роуту)
    const updatableFields = ['name', 'phone', 'bio'];
    updatableFields.forEach(field => {
      if (req.body[field] !== undefined) {
        user[field] = req.body[field];
      }
    });

    // Якщо є роль зі специфічними даними, можна їх оновити теж, наприклад volunteerInfo.bio
    if (user.role === 'Волонтер' && req.body.volunteerInfo?.bio !== undefined) {
      user.volunteerInfo.bio = req.body.volunteerInfo.bio;
    }

    // Зберігаємо оновленого користувача
    await user.save();

    res.json({
      message: 'Профіль оновлено',
      user: {
        id: user._id,
        name: user.name,
        email: user.email,
        role: user.role,
        phone: user.phone,
        photo: user.photo,
        volunteerInfo: user.volunteerInfo,
        investorInfo: user.investorInfo,
        organizerInfo: user.organizerInfo,
      },
    });
  } catch (error) {
    console.error(error);
  }
};

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    res.status(500).json({ message: 'Помилка сервера' });
  }
};

exports.getMe = async (req, res) => {
  try {
    const user = await User.findById(req.user._id);

    if (!user) {
      return res.status(404).json({ message: 'Користувача не знайдено' });
    }

    res.json({
      user: {
        id: user._id,
        name: user.name,
        email: user.email,
        bio: user.bio,
        roles: user.roles,
        phone: user.phone || null,
        photo: user.photo || null,
        volunteerInfo: user.volunteerInfo,
        investorInfo: user.investorInfo,
        organizerInfo: user.organizerInfo,
      }
    });
  } catch (error) {
    console.error('✗ Помилка у getMe:', error);
    res.status(500).json({ message: 'Помилка сервера' });
  }
};

exports.register = async (req, res) => {
  console.log('🔗 [POST /register] Тіло запиту:', req.body);
  console.log('🔗 req.file:', req.file);

  const { name, email, password, roles, phone } = req.body;

  try {
    const userExists = await User.findOne({ email });
    console.log('🔍 Перевірка існуючого користувача:', !!userExists);
    if (userExists) {
      console.log('⚠️ Користувач з такою поштою вже існує');
      return res.status(400).json({ message: 'User already exists' });
    }

    const user = await User.create({ name, email, password, roles, phone });
    console.log('✅ Користувача створено:', user._id);

    const token = generateToken(user);
    console.log('🔑 JWT створено:', token);
  }
};

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

res
  .cookie('token', token, {
    httpOnly: true,
    secure: process.env.NODE_ENV === 'production',
    sameSite: 'lax',
    maxAge: 7 * 24 * 60 * 60 * 1000,
  })
  .status(201)
  .json({
    user: {
      id: user._id,
      name: user.name,
      email: user.email,
      role: user.role,
      photo: user.photo || null,
      phone: user.phone || null,
    },
  });
} catch (err) {
  console.error('✗ Помилка при реєстрації:', err);
  res.status(500).json({ message: 'Server error', error: err.message });
}
};

exports.login = async (req, res) => {
  const { email, password } = req.body;

  try {
    const user = await User.findOne({ email });
    if (!user) return res.status(400).json({ message: 'User not found' });

    const isMatch = await user.matchPassword(password);
    if (!isMatch) return res.status(400).json({ message: 'Invalid password' });

    const token = generateToken(user);

    res
      .cookie('token', token, {
        httpOnly: true,
        secure: process.env.NODE_ENV === 'production',
        sameSite: 'lax',
        maxAge: 7 * 24 * 60 * 60 * 1000,
      })
      .status(200)
      .json({
        user: {
          id: user._id,
          name: user.name,
          email: user.email,

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

```

    roles: user.roles,
    phone: user.phone || null,
    photo: user.photo || null,
    volunteerInfo: user.volunteerInfo,
    investorInfo: user.investorInfo,
    organizerInfo: user.organizerInfo,
  }
});
} catch (err) {
  res.status(500).json({ message: 'Server error', error: err.message });
}
};

exports.logout = (req, res) => {
  res
    .cookie('token', '', {
      httpOnly: true,
      expires: new Date(0),
    })
    .json({ message: 'Вихід виконано' });
};

const Event = require('../models/Event');
const fs = require('fs');
const path = require('path');
const VolunteerApplication = require('../models/VolunteerApplication');
// Створити подію
exports.createEvent = async (req, res) => {
  try {
    const event = new Event(req.body);
    await event.save();
    res.status(201).json(event);
  } catch (err) {
    console.error('Помилка при створенні події:', err);
    res.status(400).json({ error: err.message });
  }
};

// Отримати всі події
exports.getAllEvents = async (req, res) => {
  try {
    const { page = 1, limit = 6, search = '', organizer } = req.query;

    const query = {};

    if (search) {
      query.title = { $regex: search, $options: 'i' };
    }

    if (organizer) {
      query.organizer = organizer;
    }
  }
};

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

```

    }

    const total = await Event.countDocuments(query);
    const events = await Event.find(query)
      .skip((page - 1) * limit)
      .limit(parseInt(limit))
      .sort({ startDate: 1 });

    res.json({
      events,
      total,
      page: Number(page),
      totalPages: Math.ceil(total / limit),
    });
  } catch (error) {
    console.error('Помилка при отриманні подій:', error);
    res.status(500).json({ message: 'Серверна помилка при отриманні подій' });
  }
};

// Отримати подію за ID
exports.getEventById = async (req, res) => {
  try {
    const eventId = req.params.id;

    /* 1. сама подія */
    const event = await Event
      .findById(eventId)
      .populate('organizer', 'name') // залишаємо як було
      .lean();                       // .lean() => plain object

    if (!event) {
      return res.status(404).json({ message: 'Подія не знайдена' });
    }

    /* 2. прийняті (accepted) заявки на цю подію */
    const apps = await VolunteerApplication.find({
      event : eventId,
      status: 'accepted',
    })
      .populate('volunteer', 'name photo') // тягнемо дані волонтера
      .lean();

    /* 3. додаємо у відповідь масив волонтерів */
    event.approvedVolunteers = apps.map(a => a.volunteer);

    res.json(event);
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Помилка сервера' });
  }
}

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

```

};

// Оновити подію
exports.updateEvent = async (req, res) => {
  try {
    const event = await Event.findById(req.params.id);
    if (!event) return res.status(404).json({ message: 'Подія не знайдена' });

    /* 1. Масив, який залишив користувач (після можливого видалення картинок) */
    let imgs = event.images; // стартуємо з того, що було

    if (req.body.existingImages) {
      imgs = JSON.parse(req.body.existingImages); // залишені з фронту
    }

    /* 2. Додаємо нові файли, якщо вони є */
    if (req.files?.length) {
      const add = req.files.map(f => `uploads/${f.filename}`); // обов'язково зі
      слешем!
      imgs = [...imgs, ...add];
    }

    /* 3. Оновлюємо поля */
    event.description = req.body.description ?? event.description;
    event.images      = imgs;

    await event.save();
    res.json(event);
  } catch (err) {
    console.error('updateEvent error:', err);
    res.status(500).json({ error: 'Помилка сервера' });
  }
};

// Видалити подію
exports.deleteEvent = async (req, res) => {
  try {
    const deleted = await Event.findByIdAndDelete(req.params.id);
    if (!deleted) return res.status(404).json({ message: 'Подія не знайдена' });
    res.json({ message: 'Подію видалено' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

// controllers/paymentController.js
const Event = require('../models/Event');
const User = require('../models/User');

```

```

exports.getMyDonations = async (req, res) => {
  try {
    const user = await User.findById(req.user._id)
      .populate({
        path: 'donations.event',
        select: 'title organizer images',
        populate: { path: 'organizer', select: 'name' },
      });

    res.json(user.donations);
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Помилка сервера' });
  }
};

exports.donate = async (req, res) => {
  try {
    const { amount } = req.body;
    const { id } = req.params;
    const userId = req.user._id;

    if (!amount || amount <= 0)
      return res.status(400).json({ message: 'Некоректна сума' });
    const event = await Event.findById(id);
    if (!event) return res.status(404).json({ message: 'Подія не знайдена' });
    event.collectedAmount += Number(amount);
    await event.save();
    await User.findByIdAndUpdate(
      userId,
      { $push: { donations: { event: id, amount } } },
      { new: true }
    );
    res.json({
      message: 'Оплата успішна',
      raisedAmount: event.raisedAmount,
      targetAmount: event.targetAmount,
    });
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Помилка сервера' });
  }
};

const User = require("../models/User");

exports.getUsers = async (req, res) => {
  try {
    const count = await User.countDocuments();
    res.json({ count });
  }
};

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

```

    } catch (err) {
      res.status(500).json({ message: "Server error" });
    }
  }
}

const VolunteerApplication = require('../models/VolunteerApplication');
const VolunteerReview = require('../models/VolunteerReview');
const User = require('../models/User');
const Event = require('../models/Event');

// Подати заявку волонтером
exports.createApplication = async (req, res) => {
  try {
    const { eventId, message } = req.body;
    const volunteerId = req.user._id;

    if (!eventId || !message) {
      return res.status(400).json({ message: 'eventId та message обов'язкові' });
    }

    // Перевірити чи вже є подана заявка на цю подію
    const existing = await VolunteerApplication.findOne({ event: eventId, volunteer: volunteerId });
    if (existing) {
      return res.status(400).json({ message: 'Заявка вже подана' });
    }

    const application = new VolunteerApplication({
      event: eventId,
      volunteer: volunteerId,
      message,
    });
    await application.save();

    res.status(201).json(application);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Помилка сервера' });
  }
};

exports.getAllMyApplications = async (req, res) => {
  try {
    const userId = req.user._id;

    // 1. Отримуємо події, які створив цей організатор
    const myEvents = await Event.find({ organizer: userId }).select('_id title');

    const eventIds = myEvents.map(e => e._id);

    // 2. Отримуємо заявки на ці події
    const applications = await VolunteerApplication.find({ event: { $in: eventIds } });
  }
}

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

```

        .populate('volunteer', 'name email phone photo')
        .populate('event', 'title dateStart');

    res.json({ applications });
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Помилка при завантаженні заявок' });
  }
};

// Список заявок організатору для події
exports.getApplications = async (req, res) => {
  try {
    const { event } = req.query;
    const userId = req.user._id;

    if (!event) {
      return res.status(400).json({ message: 'Потрібен ID події' });
    }

    // Перевірити, що організатор має право бачити заявки (володіє подією)
    const eventDoc = await Event.findById(event);
    if (!eventDoc) return res.status(404).json({ message: 'Подія не знайдена' });
    if (eventDoc.organizer.toString() !== userId.toString()) {
      return res.status(403).json({ message: 'Немає доступу' });
    }

    const applications = await VolunteerApplication.find({ event })
      .populate('volunteer', 'name email phone photo')
      .sort({ createdAt: -1 });

    res.json(applications);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Помилка сервера' });
  }
};

// Прийняти / Відхилити заявку
exports.updateApplicationStatus = async (req, res) => {
  try {
    const { id } = req.params;
    const { status } = req.body; // 'accepted' | 'rejected'
    const userId = req.user._id;

    if (!['accepted', 'rejected'].includes(status)) {
      return res.status(400).json({ message: 'Невірний статус' });
    }

    const application = await VolunteerApplication.findById(id);
    if (!application) return res.status(404).json({ message: 'Заявка не знайдена' });
  }
});

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const eventDoc = await Event.findById(application.event);
if (!eventDoc) return res.status(404).json({ message: 'Подія не знайдена' });

// Перевірка прав організатора
if (eventDoc.organizer.toString() !== userId.toString()) {
  return res.status(403).json({ message: 'Немає доступу' });
}

application.status = status;
await application.save();

if (status === 'accepted') {
  // Додати волонтера в масив volunteers події, якщо його там ще нема
  if (!eventDoc.volunteers.includes(application.volunteer)) {
    eventDoc.volunteers.push(application.volunteer);
    await eventDoc.save();
  }
}

res.json(application);
} catch (error) {
  console.error(error);
  res.status(500).json({ message: 'Помилка сервера' });
}
};

// Отримати відгуки про волонтера (публічно)
exports.getReviews = async (req, res) => {
  try {
    const { volunteer } = req.query;
    if (!volunteer) {
      return res.status(400).json({ message: 'Потрібен ID волонтера' });
    }
    const reviews = await VolunteerReview.find({ volunteer })
      .populate('author', 'name photo')
      .sort({ createdAt: -1 });
    res.json(reviews);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Помилка сервера' });
  }
};

// Залишити відгук про волонтера (авторизований користувач)
exports.createReview = async (req, res) => {
  try {
    const { volunteerId, text, rating } = req.body;
    const authorId = req.user._id;

    if (!volunteerId || !text || !rating) {

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

        return res.status(400).json({ message: 'volunteerId, text та rating
обов'язкові' });
    }

    if (volunteerId.toString() === authorId.toString()) {
        return res.status(400).json({ message: 'Ви не можете залишити відгук самому
про себе' });
    }

    // Створити новий відгук
    const review = new VolunteerReview({
        volunteer: volunteerId,
        author: authorId,
        text,
        rating,
    });
    await review.save();

    // Оновити avgRating у User
    const agg = await VolunteerReview.aggregate([
        { $match: { volunteer: review.volunteer } },
        { $group: { _id: '$volunteer', avgRating: { $avg: '$rating' } } }
    ]);

    if (agg.length > 0) {
        await User.findByIdAndUpdate(review.volunteer, { avgRating: agg[0].avgRating });
    }

    res.status(201).json(review);
} catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Помилка сервера' });
}
};

const jwt = require('jsonwebtoken');
const User = require('../models/User');

const JWT_SECRET = process.env.JWT_SECRET;

const protect = async (req, res, next) => {
    try {
        const token = req.cookies.token; // <-- беремо токен з cookie

        if (!token) {
            return res.status(401).json({ message: 'Токен відсутній' });
        }

        const decoded = jwt.verify(token, JWT_SECRET);

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

req.user = await User.findById(decoded.id).select('-password');

if (!req.user) {
  return res.status(401).json({ message: 'Неавторизований доступ' });
}

next();
} catch (error) {
  console.error(error);
  res.status(401).json({ message: 'Неавторизований доступ' });
}
};

const authorizeRoles = (...allowed) => {
  return (req, res, next) => {
    if (!req.user) {
      return res.status(401).json({ message: 'Неавторизований доступ' });
    }

    // roles у твоїй схемі - масив рядків
    const userRoles = req.user.roles || [];

    const hasRole = userRoles.some(r => allowed.includes(r));
    if (!hasRole) {
      return res.status(403).json({ message: 'Недостатньо прав' });
    }

    next();
  };
};

module.exports = { protect, authorizeRoles };
const multer = require('multer');
const path = require('path');

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, 'uploads/'); // ✅ Зберігає у папку uploads
  },
  filename: function (req, file, cb) {
    const uniqueName = `${Date.now()}-${file.originalname}`; // ✅ Унікальна назва
    cb(null, uniqueName);
  }
});

const upload = multer({ storage });

module.exports = upload;

const mongoose = require('mongoose');

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const eventSchema = new mongoose.Schema({
  title: { type: String, required: true },
  description: String,
  location: String,
  startDate: { type: Date, required: true },
  endDate: { type: Date, required: true },
  image: { type: String },
  organizer: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  volunteers: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }],
  createdAt: { type: Date, default: Date.now },
  images: [{ type: String }],
  collectedAmount: {
    type: Number,
    default: 0,
  },
  targetAmount: {
    type: Number,
    required: true,
  },
});

module.exports = mongoose.model('Event', eventSchema);

```

Payment.js

```

const mongoose = require('mongoose');

const paymentSchema = new mongoose.Schema({
  investor: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  event: { type: mongoose.Schema.Types.ObjectId, ref: 'Event', required: true },
  amount: { type: Number, required: true },
  message: String,
  createdAt: { type: Date, default: Date.now },
  goal: { type: Number }
});

module.exports = mongoose.model('Payment', paymentSchema);

```

Rating.js

```

const mongoose = require('mongoose');

const ratingSchema = new mongoose.Schema({
  volunteer: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  event: { type: mongoose.Schema.Types.ObjectId, ref: 'Event', required: true },
  score: { type: Number, required: true, min: 1, max: 5 },
  comment: String,
});

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

    createdAt: { type: Date, default: Date.now }
  });

module.exports = mongoose.model('Rating', ratingSchema);

```

User.js

```

const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');

const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },

  phone: { type: String, required: true }, //  нове поле

  roles: {
    type: [String],
    enum: ['Організатор', 'Волонтер', 'Інвестор'],
    required: true
  },

  photo: {
    type: String,
    default: ''
  },

  bio: { type: String, default: '' },
  avgRating: { type: Number, default: 0 },

  donations: [{
    event : { type: mongoose.Schema.Types.ObjectId, ref: 'Event' },
    amount : Number,
    date   : { type: Date, default: Date.now }
  }]
});

userSchema.pre('save', async function (next) {
  if (!this.isModified('password')) return next();
  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
  next();
});

userSchema.methods.matchPassword = async function (enteredPassword) {
  return await bcrypt.compare(enteredPassword, this.password);
};

module.exports = mongoose.model('User', userSchema);

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

VolunteerApplication.js

```
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');

const VolunteerApplicationSchema = new mongoose.Schema({
  event      : { type: mongoose.Schema.Types.ObjectId, ref: 'Event', required:
true },
  volunteer  : { type: mongoose.Schema.Types.ObjectId, ref: 'User',  required:
true },
  message    : { type: String, required: true },
  status     : { type: String, enum: ['pending','accepted','rejected'], default:
'pending' },
  createdAt  : { type: Date, default: Date.now },
});
module.exports = mongoose.model('VolunteerApplication', VolunteerApplica-
tionSchema);
```

VolunteerReview.js

```
const mongoose = require('mongoose');
const { Schema, model } = mongoose;

const VolunteerReviewSchema = new Schema({
  volunteer : { type: mongoose.Schema.Types.ObjectId, ref: 'User',  required:
true },
  author     : { type: mongoose.Schema.Types.ObjectId, ref: 'User',  required:
true }, // хто залишив
  text       : { type: String,  required: true },
  rating     : { type: Number,  min: 1, max: 5 },
  createdAt  : { type: Date, default: Date.now },
});
module.exports = mongoose.model('VolunteerReview', VolunteerReviewSchema);
```

auth.js

```
const express = require('express');
const router = express.Router();
const upload = require('../middleware/uploadMiddleware');

const { register, login, getMe, updateProfile, uploadUserPhoto, logout } = re-
quire('../controllers/authController');
const { protect } = require('../middleware/authMiddleware');

router.post('/register', register);
router.post('/login', login);
router.post('/logout', protect, logout);

router.get('/me', protect, getMe);
router.put('/me', protect, updateProfile);
```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```
router.post('/upload-photo', protect, upload.single('photo'), uploadUserPhoto);

module.exports = router;
```

events.js

```
const express = require('express');
const router = express.Router();
const EventController = require('../controllers/eventController');
const upload = require('../middleware/uploadMiddleware');

// Створення події
router.post('/createEvent', EventController.createEvent);

// Отримання всіх подій
router.get('/', EventController.getAllEvents);

// Отримання однієї події
router.get('/:id', EventController.getEventById);

// Видалення події
router.delete('/:id', EventController.deleteEvent);

router.put('/:id', upload.array('images', 5), EventController.updateEvent);

module.exports = router;
```

payment.js

```
const express = require('express');
const router = express.Router();
const paymentC = require('../controllers/paymentController');
const { protect } = require('../middleware/authMiddleware');

// POST /api/payments/:id (id = eventId)
router.post('/:id', protect, paymentC.donate);
router.get('/my', protect, paymentC.getMyDonations);

module.exports = router;
```

stats.js

```
const express = require("express");
const router = express.Router();
const User = require("../models/User"); // твоя модель користувача
const { getUsers } = require("../controllers/statsController");

router.get("/user-count", getUsers);
```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```
module.exports = router;
```

users.js

```
const express = require('express');
const router = express.Router();
const User = require('../models/User');
const Event = require('../models/Event');
const Auth = require('../controllers/authController');
const { protect } = require('../middleware/authMiddleware');

// GET /api/users/:id/public
router.get('/:id/public', async (req, res) => {
  try {
    const user = await User.findById(req.params.id).select('name photo roles');
    if (!user) return res.status(404).json({ message: 'Користувача не знайдено' });

    const events = await Event.find({ organizer: req.params.id })
      .select('title startDate endDate images collectedAmount targetAmount');

    res.json({ user, events });
  } catch (e) {
    console.error(e);
    res.status(500).json({ message: 'Помилка сервера' });
  }
});
router.patch('/:id', protect, Auth.updateProfile);

module.exports = router;
```

volunteers.js

```
const express = require('express');
const router = express.Router();
const volunteersController = require('../controllers/volunteersController');
const { protect, authorizeRoles } = require('../middleware/authMiddleware'); // приклад middleware авторизації

// Волонтер подає заявку (треба бути залогіненим і мати роль 'Волонтер')
router.post('/applications', protect, authorizeRoles('Волонтер'), volunteersController.createApplication);

// Організатор дивиться заявки на подію (ролі перевіряються в контролері)
router.get('/applications', protect, volunteersController.getApplications);

router.get(
  '/applications/all',
  protect,
```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    authorizeRoles('Організатор'),
    volunteersController.getAllMyApplications
  );

  // Організатор приймає/відхиляє заявку
  router.patch('/applications/:id', protect, authorizeRoles('Організатор'), volunteersController.updateApplicationStatus);

  // Публічний перегляд відгуків про волонтера
  router.get('/reviews', volunteersController.getReviews);

  // Залишити відгук (залогінені користувачі)
  router.post('/reviews', protect, volunteersController.createReview);

  module.exports = router;

```

index.js

```

const express = require('express');
const app = express();
const morgan = require('morgan');
const cors = require('cors');
const dotenv = require('dotenv');
dotenv.config();
const connectDB = require('./config/db');
const authRoutes = require('./routes/auth');
const cookieParser = require('cookie-parser');
const statsRoutes = require('./routes/stats');
const eventRoutes = require('./routes/events');
const path = require('path');
const paymentRoutes = require('./routes/payments');
const volunteersRouter = require('./routes/volunteers');
app.use(cors({
  origin: 'http://localhost:3000',
  credentials: true
}));

// Завантаження змінних оточення
app.use(morgan('dev'));
app.use(cookieParser());
app.use(express.json());
app.use('/uploads', express.static(path.join(__dirname, 'uploads')));
connectDB(); // Підключення до бази даних
app.use('/api/stats', statsRoutes);

app.use('/api/auth', authRoutes); // Підключення маршруту авторизації та реєстрації

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

app.get('/', (req, res) => {
  res.send('API працює 🚀');
});
app.use('/api/events', eventRoutes);
app.use('/api/payments', paymentRoutes);
app.use('/api/users', require('./routes/users'));
app.use('/api/volunteers', volunteersRouter);

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`🚀 Сервер на порті ${PORT}`));

```

investorprofile.jsx

```

import { useEffect, useState } from 'react';
import axios from 'axios';
import {
  Paper, Typography, Box, Avatar, Stack, Divider, CircularProgress, Link as
  MuiLink
} from '@mui/material';
import { Paid as PaidIcon } from '@mui/icons-material';
import { Link } from 'react-router-dom'; // ➡️ додали

const InvestorProfile = () => {
  const [donations, setDonations] = useState(null);
  const [error, setError] = useState('');

  useEffect(() => {
    axios
      .get(`${process.env.REACT_APP_API_URL}/api/payments/my`, { withCredentials:
true })
      .then(res => setDonations(res.data))
      .catch(err => {
        console.error(err);
        setError('Не вдалося отримати ваші донати');
      });
  }, []);

  if (error) return <Typography color="error">{error}</Typography>;
  if (donations === null) return <CircularProgress sx={{ mt: 4 }} />;

  const total = donations.reduce((sum, d) => sum + d.amount, 0);

  return (
    <Paper sx={{ p: 3, mt: 2, minWidth: 700 }}>
      <Typography variant="h5" gutterBottom>Кабінет інвестора</Typography>

      <Typography variant="h6" sx={{ mb: 2 }}>
        Загалом інвестовано: {total.toLocaleString()} ₪
      </Typography>

      {donations.length === 0 ? (

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    <Typography color="text.secondary">Ви ще не робили внесків.</Typography>
  ) : (
    <Box>
      {donations.map((d, idx) => {
        const event = d.event; // короткий доступ
        const hasEvent = Boolean(event && event._id);

        return (
          <Box key={idx} sx={{ mb: 2 }}>
            <Stack direction="row" spacing={2} alignItems="center">
              <Avatar
                src={event?.images?.[0] ? `${process.env.REACT_APP_API_URL}/${event.images[0]}` : undefined}
                variant="rounded"
              >
                <PaidIcon />
              </Avatar>

              <Box>
                {hasEvent ? (
                  <MuiLink
                    component={Link}
                    to={` /event/${event._id}`}
                    underline="hover"
                    fontWeight={600}
                  >
                    {event.title}
                  </MuiLink>
                ) : (
                  <Typography fontWeight={600}>Подію видалено</Typography>
                )}

                <Typography variant="body2" color="text.secondary">
                  {new Date(d.date).toLocaleDateString('uk-UA')} –
                  {d.amount.toLocaleString()} ₪
                </Typography>

                {event?.organizer && (
                  <Typography variant="body2" component={Link} to={` /organ-
изер/${d.event.organizer._id}`} style={{ textDecoration: 'none' }}>
                    Організатор:{d.event.organizer.name}
                  </Typography>
                )}
              </Box>
            </Stack>

            {idx !== donations.length - 1 && <Divider sx={{ mt: 2 }} />}
          </Box>
        );
      })}
    </Box>
  );
}
}
</Box>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    })
  </Paper>
);
};

export default InvestorProfile;

```

organizerprofile.jsx

```

import { useEffect, useState } from 'react';
import axios from 'axios';
import {
  Typography,
  Paper,
  Button,
  Box,
  Grid,
  Pagination
} from '@mui/material';
import { AccessElement } from '../AccessControl';
import { useAuth } from '../context/AuthContext';
import EventCardForOrganizer from '../EventCardForOrganizer';
import { Link } from 'react-router-dom';

const OrganizerProfile = () => {
  const { user } = useAuth();
  const [events, setEvents] = useState([]);
  const [page, setPage] = useState(1);
  const [totalPages, setTotalPages] = useState(1);

  const LIMIT = 6; // 6 подій = 2 в ряд * 3 ряди

  const handleDelete = async (eventId) => {
    if (!window.confirm('Ви впевнені, що хочете видалити цю подію?')) return;

    try {
      await axios.delete(`${process.env.REACT_APP_API_URL}/api/events/${eventId}`);
      setEvents((prevEvents) => prevEvents.filter((event) => event._id !== eventId));
    } catch (error) {
      console.error('Помилка при видаленні події:', error);
      alert('Не вдалося видалити подію. Спробуйте пізніше.');
```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

        organizer: user.id,
        page,
        limit: LIMIT,
      },
    ));

    setEvents(res.data.events);
    setTotalPages(res.data.totalPages);
  } catch (err) {
    console.error('Помилка при завантаженні подій організатора:', err);
  }
};

if (user?.id) {
  fetchEvents();
}
}, [user, page]);

return (
  <Paper sx={{ p: 3, mt: 2, minHeight: '600px' }}>
    <Typography variant="h5" gutterBottom>Профіль організатора</Typography>

    <AccessElement allowedRoles="Організатор">
      <Box sx={{ display: 'flex', gap: 2, mb: 2 }}>
        <Button color="inherit" variant="outlined" component={Link}
to='../events/createEvent'>
          Створити Подію
        </Button>
        <Button color="primary" variant="contained" component={Link} to='../or-
ganizer/applications'>
          Керувати заявками
        </Button>
      </Box>
    </AccessElement>

    <Typography variant="h6" sx={{ mt: 2 }}>Мої події:</Typography>

    {events.length === 0 ? (
      <Typography color="text.secondary">Подій поки немає</Typography>
    ) : (
      <>
        <Grid container spacing={2} mt={1}>
{events.map((event) => (
  <Grid
    item
    xs={12}
    sm={6}
    md={4}
    key={event._id}
    sx={{ display: 'flex' }}
    >

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    <EventCardForOrganizer
      event={event}
      onDelete={handleDelete}
      sx={{ flexGrow: 1 }}
    />
  </Grid>
  )}}
</Grid>

    {totalPages > 1 && (
      <Box mt={3} display="flex" justifyContent="center">
        <Pagination
          count={totalPages}
          page={page}
          onChange={(_, value) => setPage(value)}
          color="primary"
        />
      </Box>
    )}
  </>
  )}
</Paper>
);
};

export default OrganizerProfile;

```

Volunteerprofile

```

// src/components/profile/VolunteerProfile.jsx
import React, { useState, useEffect } from 'react';
import {
  Paper, Typography, TextField, Button,
  Box, Rating, Avatar, Stack,
} from '@mui/material';
import dayjs from 'dayjs';
import axios from 'axios';
import { useAuth } from '../../context/AuthContext';

const BASE = process.env.REACT_APP_API_URL;

export default function VolunteerProfile() {
  const { user, refreshUser, setUser } = useAuth();

  /* чернетка для редагування */
  const [draftBio, setDraftBio] = useState('');
  const [editing, setEditing] = useState(false);
  const [reviews, setReviews] = useState([]);

  /* коли user приходить або змінився – синхронізуємо чернетку */
  useEffect(() => {

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

    setDraftBio(user?.bio || '');
  }, [user?.bio]);

  /* Відгуки */
  useEffect(() => {
    if (!user?._id) return;
    axios.get(`${BASE}/api/volunteers/reviews`, {
      params: { volunteer: user._id },
      withCredentials: true,
    })
      .then(res => setReviews(res.data))
      .catch(console.error);
  }, [user?._id]);

  /* зберегти */
  const saveBio = async () => {
    try {
      await axios.patch(
        `${BASE}/api/users/${user._id}`,
        { bio: draftBio },
        { withCredentials: true }
      );
    }

    // МИТТЕВО ВНОСИМО ЗМІНИ У КОНТЕКСТ
    setUser((prev) => ({ ...prev, bio: draftBio }));

    // option-2: далі можете _не_ чекати refreshUser,
    // але корисно підтягти найсвіжіші дані з БД
    refreshUser();

    setEditing(false);
  } catch (err) {
    console.error(err);
    alert('Не вдалося зберегти біо');
  }
};

if (!user) return null;

return (
  <Paper sx={{ p: 3, mt: 2 }}>
    <Typography variant="h5" gutterBottom>
      Волонтер
    </Typography>

    {/* ----- Про себе ----- */}
    <Box sx={{ mb: 3 }}>
      <Typography variant="h6">Про себе</Typography>

      {editing ? (
        <>

```

```

        <TextField
          multiline
          rows={4}
          fullWidth
          value={draftBio}
          onChange={(e) => setDraftBio(e.target.value)}
          sx={{ my: 1 }}
        />
        <Stack direction="row" spacing={1}>
          <Button variant="contained" onClick={saveBio}>Зберегти</Button>
          <Button color="inherit" onClick={() => setEditing(false)}>Скасувати</Button>
        </Stack>
      </>
    ) : (
      <>
        <Typography sx={{ whiteSpace: 'pre-wrap', my: 1 }}>
          {user.bio || 'Поки що немає опису.'}
        </Typography>
        <Button variant="outlined" size="small" onClick={() => setEditing(true)}>
          Редагувати
        </Button>
      </>
    )}
  </Box>

  { /* ----- Відгуки ----- */ }
  <Typography variant="h6" gutterBottom>Відгуки</Typography>

  {reviews.length === 0 && (
    <Typography color="text.secondary">Поки що немає відгуків.</Typography>
  )}

  {reviews.map((rv) => (
    <Paper key={rv._id} sx={{ p: 2, my: 1 }}>
      <Stack direction="row" alignItems="center" gap={1}>
        <Avatar
          src={rv.author?.photo ? `${BASE}/uploads/${rv.author.photo}` : undefined}
        />
        <Typography variant="subtitle2">{rv.author?.name || '-'}</Typography>
        <Rating value={rv.rating} size="small" readOnly />
        <Typography variant="caption" sx={{ ml: 'auto' }}>
          {dayjs(rv.createdAt).format('DD.MM.YYYY')}
        </Typography>
      </Stack>
      <Typography sx={{ mt: 1 }}>{rv.text}</Typography>
    </Paper>
  )))
</Paper>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```
);
}
```

Accesscontro.jsx

```
import React from 'react';
import { useAuth } from '../context/AuthContext';
import { useNavigate } from 'react-router-dom';
import { Card, CardContent, Typography, Button } from '@mui/material';

export const AccessElement = ({ allowedRoles, children }) => {
  const { user } = useAuth();

  const hasAccess = user?.roles?.some(role => allowedRoles.includes(role));

  if (!hasAccess) {
    return null; // нічого не рендериться, якщо немає доступу
  }

  return <>{children}</>;
};

export const HideLoginElement = ({ allowedRoles, children }) => {
  const { user } = useAuth();

  const hasAccess = user?.roles?.some(role => allowedRoles.includes(role));

  if (hasAccess) {
    return null; // нічого не рендериться, якщо немає доступу
  }

  return <>{children}</>;
};

export const AccessControl = ({ allowedRoles, children }) => {
  const { user } = useAuth();
  const navigate = useNavigate();

  const hasAccess = user?.roles?.some(role => allowedRoles.includes(role));

  if (!hasAccess) {
    const missingRolesText = allowedRoles.join(', ');

    return (
      <Card sx={{ maxWidth: 600, margin: 'auto', mt: 5, p: 3, textAlign: 'center' }}>
        <CardContent>
          <Typography variant="h5" gutterBottom>
```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Доступ заборонено
      </Typography>
      <Typography variant="body1" gutterBottom>
        Ця сторінка доступна лише користувачам з наступними ролями:
      </Typography>
      <strong>{missingRolesText}</strong>
    </Typography>
    <Button
      variant="contained"
      color="primary"
      sx={{ mt: 2 }}
      onClick={() => navigate('/events')}
    >
      На головну
    </Button>
  </CardContent>
</Card>
);
}

return <>{children}</>;
};

```

ErrorBoundary.jsx

```

import React from 'react';
import { Button, Box, Typography } from '@mui/material';

export default class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false };
  }

  static getDerivedStateFromError() {
    return { hasError: true };
  }

  componentDidCatch(err, info) {
    // можна логувати в Sentry / console
    console.error('UI error:', err, info);
  }

  handleReload = () => window.location.reload();

  render() {
    if (!this.state.hasError) return this.props.children;

    /* кастомна сторінка */
    return (
      <Box sx={{ p: 4, textAlign: 'center' }}>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    <Typography variant="h4" gutterBottom>
      Упс... Щось пішло не так 😞
    </Typography>
    <Typography sx={{ mb: 3 }}>
      Спробуйте перезавантажити сторінку або поверніться на головну.
    </Typography>
    <Button variant="contained" onClick={this.handleReload}>
      Перезавантажити
    </Button>
  </Box>
);
}
}

```

Eventcard.jsx

```

import React from 'react';
import { Card, Button, Box, CardContent, Typography } from '@mui/material';
import { Link } from 'react-router-dom';

const EventCard = ({ event }) => {
  return (
    <Card
      sx={{
        display: 'flex',
        flexDirection: 'row',
        width: 410,
        height: 300,
        maxWidth: 410,
        maxHeight: 300,
        overflow: 'hidden',
      }}
    >
      </* Зображення зліва */>
      <Box
        component="img"
        src={event.image}
        alt={event.title}
        sx={{
          width: 160,
          height: '100%',
          objectFit: 'cover',
        }}
      />

      </* Контент справа */>
      <CardContent
        sx={{
          display: 'flex',

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

```

        flexDirection: 'column',
        justifyContent: 'space-between',
        flexGrow: 1,
        p: 2,
      }}
    >
    <Box>
      <Typography variant="h6" fontWeight={600}>
        {event.title}
      </Typography>
      <Typography color="text.secondary" fontSize={14}>
        {new Date(event.startDate).toLocaleDateString()} -{' '}
        {new Date(event.endDate).toLocaleDateString()}
      </Typography>
      <Typography variant="body2" color="text.secondary">
        {event.description.length > 40
          ? `${event.description.slice(0, 40)}...`
          : event.description}
      </Typography>

    </Box>

    {/* Кнопка "Дізнатись більше" */}
    <Box mt={2}>
      <Button
        variant="contained"
        color="primary"
        component={Link}
        to={` /event/${event._id}`}
        fullWidth
      >
        Дізнатись більше
      </Button>
    </Box>
  </CardContent>
</Card>
);
};

export default EventCard;

```

EventCardForOrganizer.jsx

```

// EventCardForOrganizer.jsx
import {
  Card, CardMedia, CardContent,
  Typography, Button, Box, LinearProgress, Stack,
} from '@mui/material';
import { Link } from 'react-router-dom';

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

const EventCardForOrganizer = ({
  event,
  onDelete,
  hideActions = false,
}) => {
  // ► нові поля (за замовчуванням порожні)
  const {
    _id,
    title = 'Без назви',
    description = '',
    collectedAmount = 0,
    targetAmount = 1,
    image = '', // ① ← одне фото-URL
    images = [], // ② ← масив (якщо залишився)
  } = event;

  /** ① Вибираємо прев'ю
   *   пріоритет: image (рядок) → перший із images */
  const preview =
    image?.trim() ||
    (Array.isArray(images) && images.length ? images[0] : '');

  /** ② Якщо прев'ю відносно («uploads/...») — додаємо хост */
  const previewSrc = preview.startsWith('http')
    ? preview
    : preview
    ? `${process.env.REACT_APP_API_URL}/${preview}`
    : ''; // порожній ⇒ не рендеримо <CardMedia/>

  const progress = Math.min(100, (collectedAmount / targetAmount) * 100);

  return (
    <Card sx={{ height: '100%', display: 'flex', flexDirection: 'column', }}>
      {previewSrc && (
        <CardMedia
          component="img"
          height="140"
          image={previewSrc}
          alt={title}
        />
      )}

      <CardContent sx={{ flexGrow: 1 }}>
        <Typography variant="h6" fontWeight={600}>
          <Link to={`/event/${_id}`} style={{ textDecoration: 'none' }}>
            {title}
          </Link>
        </Typography>

        <Typography variant="body2" color="text.secondary" sx={{ mb: 1 }}>

```

```

        {description.length > 40 ? `${description.slice(0, 40)}...` : descrip-
tion}
      </Typography>

      <Box sx={{ mb: 2 }}>
        <Typography variant="body2">
          Зібрано: {collectedAmount.toLocaleString()} / {targetAmount.toLocale-
String()} ₪
        </Typography>
        <LinearProgress
          variant="determinate"
          value={progress}
          sx={{ height: 8, borderRadius: 5 }}
        />
      </Box>

      {!hideActions && (
        <Stack direction="row" spacing={1}>
          <Button
            variant="outlined"
            size="small"
            component={Link}
            to={` /event/${_id}/edit`}
          >
            Редагувати
          </Button>
          <Button
            variant="outlined"
            size="small"
            color="error"
            onClick={() => onDelete?.(_id)}
          >
            Видалити
          </Button>
        </Stack>
      )}
    </CardContent>
  </Card>
);
};

export default EventCardForOrganizer;

```

Footer.jsx

```

import { Box, Typography } from '@mui/material';

const Footer = () => {
  return (
    <Box sx={{ bgcolor: 'primary.main', color: 'white', textAlign: 'center', p:
2, mt: 4 }}>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <Typography variant="body2">© {new Date().getFullYear()} EventPlatform. All
rights reserved.</Typography>
      </Box>
    );
  };

export default Footer;

```

Navbar.jsx

```

import React, { useState } from 'react';
import {
  AppBar,
  Toolbar,
  Typography,
  Button,
  Box,
  IconButton,
  Drawer,
  List,
  ListItem,
  ListItemText,
  useMediaQuery,
  Menu,
  MenuItem,
  ListItemIcon
} from '@mui/material';
import MenuIcon from '@mui/icons-material/Menu';
import AccountCircleIcon from '@mui/icons-material/AccountCircle';
import CheckIcon from '@mui/icons-material/Check';
import ExpandMoreIcon from '@mui/icons-material/ExpandMore';
import { useTheme } from '@mui/material/styles';
import { Link } from 'react-router-dom';
import { useAuth } from '../context/AuthContext'; // твій кастомний хук

const Navbar = () => {
  const { user, logout, activeRole, changeRole } = useAuth();
  if (user) var availableRoles = user.roles;
  const [anchorEl, setAnchorEl] = useState(null);
  const [drawerOpen, setDrawerOpen] = useState(false);
  const theme = useTheme();
  const isMobile = useMediaQuery(theme.breakpoints.down('sm'));
  const open = Boolean(anchorEl);
  const handleClick = (event) =>{

    setAnchorEl(event.currentTarget);
  }
  const handleClose = () => setAnchorEl(null);

```

```

const handleRoleSelect = (role) => {
  changeRole(role);
  handleClose();
};

const toggleDrawer = (open) => () => {
  setDrawerOpen(open);
};

const drawerContent = (
  <Box sx={{ width: 250 }} role="presentation" onClick={toggleDrawer(false)}
  onKeyDown={toggleDrawer(false)}>
    <List>
      <ListItem button component={Link} to="/">
        <ListItemText primary="Головна" />
      </ListItem>
      {!user ? (
        <>
          <ListItem button component={Link} to="/login">
            <ListItemText primary="Увійти" />
          </ListItem>
          <ListItem button component={Link} to="/register">
            <ListItemText primary="Реєстрація" />
          </ListItem>
        </>
      ) : (
        <>
          <ListItem button component={Link} to="/profile">
            <AccountCircleIcon sx={{ mr: 1 }} />
            <ListItemText primary="Профіль" />
          </ListItem>
          <ListItem button onClick={logout}>
            <ListItemText primary="Вийти" />
          </ListItem>
        </>
      )}
    </List>
  </Box>
);

return (
  <AppBar position="static" color="primary">
    <Toolbar sx={{ justifyContent: 'space-between' }}>
      <Typography variant="h6" component={Link} to="/" sx={{ textDecoration:
'none', color: 'white' }}>
        EventPlatform
      </Typography>
      {isMobile ? (
        <>
          <IconButton edge="end" color="inherit" onClick={toggleDrawer(true)}>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <MenuIcon />
      </IconButton>
      <Drawer anchor="right" open={drawerOpen} onClose={toggle-
Drawer(false)}>
        {drawerContent}
      </Drawer>
    </>
  ) : (
    <Box>
      <Button color='inherit' component={Link} to="/events">Події</Button>
      {!user ? (
        <>
          <Button color="inherit" component={Link} to="/login">Увійти</But-
ton>
          <Button color="inherit" component={Link} to="/regis-
ter">Реєстрація</Button>
        </>
      ) : (
        <>
          <Button color="inherit" onClick={handleClick} endIcon={<Expand-
MoreIcon />}>
            {activeRole || 'Оберіть роль'}
          </Button>
          <Menu anchorEl={anchorEl} open={open} onClose={handleClose}>
            {availableRoles.map((role) => (
              <MenuItem key={role} onClick={() => handleRoleSelect(role)}>
                <ListItemIcon>
                  {activeRole === role && <CheckIcon fontSize="small" />}
                </ListItemIcon>
                <ListItemText>{role}</ListItemText>
              </MenuItem>
            ))}
          </Menu>
          <Button color="inherit" component={Link} to="/profile" startI-
con={<AccountCircleIcon />}>
            Профіль
          </Button>
          <Button color="inherit" onClick={logout}>Вийти</Button>
        </>
      )}
    </Box>
  )}
</Toolbar>
</AppBar>
);

};

export default Navbar;

```

AuthContext.jsx

```
// src/context/AuthContext.jsx
import { createContext, useContext, useState, useEffect } from 'react';
import axios from 'axios';
import Cookies from 'js-cookie';

export const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);
  const [activeRole, setActiveRole] = useState(null);

  /* ----- */
  /* helpers                                     */
  /* ----- */

  /** Завантажує дані поточного користувача із cookie-токеном */
  const fetchMe = async () => {
    try {
      const { data } = await axios.get(
        `${process.env.REACT_APP_API_URL}/api/auth/me`,
        { withCredentials: true }
      );
      setUser(data.user);
    } catch {
      setUser(null);
    }
  };

  /** Публічний метод – можна викликати після PATCH-ів */
  const refreshUser = async () => {
    await fetchMe();
  };

  /* ----- */
  /* ініціалізація                             */
  /* ----- */

  useEffect(() => {
    (async () => {
      await fetchMe();
      setLoading(false);
    })();

    const savedRole = Cookies.get('activeRole');
    if (savedRole) setActiveRole(savedRole);
  }, []);

  /* ----- */
}
```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

/* auth actions */
/* ----- */

const changeRole = (role) => {
  setActiveRole(role);
  Cookies.set('activeRole', role);
};

const register = async (payload) => {
  try {
    const { data } = await axios.post(
      `${process.env.REACT_APP_API_URL}/api/auth/register`,
      payload,
      { withCredentials: true }
    );
    setUser(data.user);
    return true;
  } catch { return false; }
};

const login = async (payload) => {
  try {
    const { data } = await axios.post(
      `${process.env.REACT_APP_API_URL}/api/auth/login`,
      payload,
      { withCredentials: true }
    );
    setUser(data.user);
    return true;
  } catch { return false; }
};

const logout = async () => {
  try {
    await axios.post(
      `${process.env.REACT_APP_API_URL}/api/auth/logout`,
      {},
      { withCredentials: true }
    );
  } catch { /* ignore */ }
  setUser(null);
};

const uploadPhoto = async (file) => {
  try {
    const formData = new FormData();
    formData.append('photo', file);

    const { data } = await axios.post(
      `${process.env.REACT_APP_API_URL}/api/auth/upload-photo`,
      formData,

```

Дячук Д.О.

Годлевський Ю.О.

ІПЗ.КР.Б-121-25-ПЗ

Арк.

96

Змн.	Арк.	№ докум.	Підпис	Дата
------	------	----------	--------	------

```

        {
          withCredentials: true,
          headers: { 'Content-Type': 'multipart/form-data' },
        }
      );
      setUser((p) => ({ ...p, photo: data.photo }));
    } catch (err) {
      console.error('Помилка при завантаженні фото:', err);
    }
  };

  /* ----- */
  /* render */
  /* ----- */

  if (loading) return null; // тут можна вставити спінер

  return (
    <AuthContext.Provider
      value={{
        user,
        register,
        login,
        logout,
        setUser,
        uploadPhoto,
        activeRole,
        changeRole,
        refreshUser, // ← додаємо у контекст
      }}
    >
      {children}
    </AuthContext.Provider>
  );
};

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

```

AllEventsPage

```

import React, { useState, useEffect } from 'react';
import {
  Box,
  Container,
  TextField,
  Typography,
  Link,
  Grid,
  Pagination,
  CircularProgress,
  Button,

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

} from '@mui/material';
import axios from 'axios';
import EventCard from '../components/EventCard';
import { AccessElement } from '../components/AccessControl';

const AllEventsPage = () => {
  const [events, setEvents] = useState([]);
  const [search, setSearch] = useState('');
  const [page, setPage] = useState(1);
  const [totalPages, setTotalPages] = useState(1);
  const [loading, setLoading] = useState(false);
  const limit = 6;

  const fetchEvents = async () => {
    try {
      setLoading(true);
      const res = await axios.get(
        `${process.env.REACT_APP_API_URL}/api/events`,
        {
          params: {
            search,
            page,
            limit,
          },
        }
      );
      setEvents(res.data.events);
      setTotalPages(res.data.totalPages);
    } catch (err) {
      console.error('Помилка при завантаженні подій', err);
    } finally {
      setLoading(false);
    }
  };

  useEffect(() => {
    fetchEvents();
  }, [search, page]);

  const handleSearchChange = (e) => {
    setSearch(e.target.value);
    setPage(1); // повертаємося на першу сторінку при новому пошуку
  };

  return (

    <Container maxWidth="md" sx={{ py: 5 }}>
      <Typography variant="h4" fontWeight={600} mb={3} align="center">
        Усі події
      </Typography>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<TextField
  label="Пошук події за назвою"
  variant="outlined"
  fullWidth
  value={search}
  onChange={handleSearchChange}
  sx={{ mb: 4 }}
/>

{loading ? (
  <Box display="flex" justifyContent="center" py={6}>
    <CircularProgress />
  </Box>
) : events.length > 0 ? (
  <>
    <Grid container spacing={3}>
      {events.map((event) => (
        <Grid item xs={12} sm={6} key={event._id}>
          <EventCard event={event} />
        </Grid>
      ))}
    </Grid>

    <Box display="flex" justifyContent="center" mt={4}>
      <Pagination
        count={totalPages}
        page={page}
        onChange={(e, value) => setPage(value)}
        color="primary"
      />
    </Box>
  </>
) : (
  <Typography align="center" color="text.secondary">
    Подій не знайдено.
  </Typography>
)}
</Container>
);
};

export default AllEventsPage;

```

ApplicationsManagePage.jsx

```

import React, { useEffect, useState } from 'react';
import {
  Box,
  Paper,
  TextField,

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				99
Змн.	Арк.	№ докум.	Підпис	Дата		

```

MenuItem,
Select,
InputLabel,
FormControl,
Typography,
Button,
Stack,
} from '@mui/material';
import { DataGrid } from '@mui/x-data-grid';
import dayjs from 'dayjs';
import axios from 'axios';
import { useAuth } from '../context/AuthContext';
import { Link } from 'react-router-dom';
const BASE = process.env.REACT_APP_API_URL;

export default function ApplicationsManagePage() {
  const { user } = useAuth();
  const [applications, setApplications] = useState([]);
  const [filteredApps, setFilteredApps] = useState([]);
  const [eventOptions, setEventOptions] = useState([]);

  // Фільтри
  const [statusFilter, setStatusFilter] = useState('');
  const [eventFilter, setEventFilter] = useState('');
  const [searchName, setSearchName] = useState('');
  const statusMap = {
    pending: 'Очікує',
    accepted: 'Прийнято',
    rejected: 'Відхилено',
  };
  useEffect(() => {
    const fetchApplications = async () => {
      try {
        const res = await axios.get(`${BASE}/api/volunteers/applications/all`, {
          withCredentials: true,
        });
        setApplications(res.data.applications);
        setFilteredApps(res.data.applications);

        // Витягуємо список унікальних подій
        const uniqueEvents = [
          ...new Map(
            res.data.applications.map((app) => [app.event._id, app.event])
          ).values(),
        ];
        setEventOptions(uniqueEvents);
      } catch (err) {
        console.error('Помилка при завантаженні заявок:', err);
      }
    };
  });

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    fetchApplications();
  }, []);

  // Фільтрація
  useEffect(() => {
    let filtered = [...applications];

    if (statusFilter) {
      filtered = filtered.filter((app) => app.status === statusFilter);
    }

    if (eventFilter) {
      filtered = filtered.filter((app) => app.event._id === eventFilter);
    }

    if (searchName) {
      filtered = filtered.filter((app) =>
        (app.volunteer?.name || '')
          .toLowerCase()
          .includes(searchName.toLowerCase())
      );
    }

    setFilteredApps(filtered);
  }, [statusFilter, eventFilter, searchName, applications]);

  // Обробка оновлення статусу заявки
  const updateStatus = async (id, newStatus) => {
    try {
      await axios.patch(
        `${BASE}/api/volunteers/applications/${id}`,
        { status: newStatus },
        { withCredentials: true }
      );
      // Оновити локальний стан, щоб одразу побачити зміни
      setApplications((prev) =>
        prev.map((app) =>
          app._id === id ? { ...app, status: newStatus } : app
        )
      );
    } catch (error) {
      console.error('Помилка оновлення статусу:', error);
      alert('Не вдалося оновити статус. Спробуйте пізніше.');
```

```

const columns = [
  {
    field: 'volunteerName', // ← вигадане поле
    headerName: 'Волонтер',
  }
];

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

```

flex: 1,
sortable: false,
renderCell: ({ row }) =>
  row.volunteer ? (
    <Link
      to={`../../volunteer/${row.volunteer._id}`}
      style={{ textDecoration: 'none' }}
    >
      {row.volunteer.name}
    </Link>
  ) : '-',
},
{
  field: 'eventTitle', // ← вигадане поле
  headerName: 'Подія',
  flex: 1,
  sortable: false,
  renderCell: ({ row }) =>
    row.event ? (
      <Link
        to={`../../event/${row.event._id}`}
        style={{ textDecoration: 'none' }}
      >
        {row.event.title}
      </Link>
    ) : '-',
  },
{
  field: 'message',
  headerName: 'Повідомлення',
  flex: 2,
},
{
  field: 'status',
  headerName: 'Статус',
  flex: 1,
},
{
  field: 'createdAt',
  headerName: 'Дата',
  flex: 1,
  valueGetter: (p) =>
    p?.row?.createdAt ? dayjs(p.row.createdAt).format('DD.MM.YYYY') : '',
},
{
  field: 'actions',
  headerName: 'Дії',
  flex: 1,
  sortable: false,
  filterable: false,
  renderCell: (p) => (

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<Stack direction="row" spacing={1}>
  <Button
    variant="contained"
    color="success"
    size="small"
    disabled={p.row.status === 'accepted'}
    onClick={() => updateStatus(p.row._id, 'accepted')}
  >
    Прийняти
  </Button>
  <Button
    variant="outlined"
    color="error"
    size="small"
    disabled={p.row.status === 'rejected'}
    onClick={() => updateStatus(p.row._id, 'rejected')}
  >
    Відхилити
  </Button>
</Stack>
),
},
];

return (
  <Box sx={{ p: 4 }}>
    <Typography variant="h5" gutterBottom>
      Заявки волонтерів на всі події
    </Typography>

    { /* Фільтри */ }
    <Box sx={{ display: 'flex', gap: 2, mb: 3, flexWrap: 'wrap' }}>
      <TextField
        label="Пошук за ім'ям"
        variant="outlined"
        value={searchName}
        onChange={(e) => setSearchName(e.target.value)}
      />

      <FormControl sx={{ minWidth: 160 }}>
        <InputLabel>Статус</InputLabel>
        <Select
          value={statusFilter}
          onChange={(e) => setStatusFilter(e.target.value)}
          label="Статус"
        >
          <MenuItem value="">Всі</MenuItem>
        {Object.entries(statusMap).map(([code, ui]) => (
          <MenuItem key={code} value={code}>
            {ui}
          </MenuItem>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				103
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    )))
    </Select>
  </FormControl>

  <FormControl sx={{ minWidth: 200 }}>
    <InputLabel>Подія</InputLabel>
    <Select
      value={eventFilter}
      onChange={(e) => setEventFilter(e.target.value)}
      label="Подія"
    >
      <MenuItem value="">Bci</MenuItem>
      {eventOptions.map((e) => (
        <MenuItem key={e._id} value={e._id}>
          {e.title}
        </MenuItem>
      ))}
    </Select>
  </FormControl>
</Box>

{/* Таблиця */}
<Paper>
  <DataGrid
    rows={filteredApps}
    getRowId={(r) => r._id}
    columns={columns}
    autoHeight
    pageSize={10}
    rowsPerPageOptions={[10, 20]}
    disableSelectionOnClick
  />
</Paper>
</Box>
);
}

```

CreateEventPage.jsx

```

import React, { useState } from 'react';
import axios from 'axios';
import {
  Box,
  Button,
  Container,
  TextField,
  Typography,
  Paper,
  Checkbox,

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		104

```

Alert,
FormControlLabel,
Card,
CardContent
} from '@mui/material';
import { useNavigate } from 'react-router-dom';

import { useAuth } from '../context/AuthContext';
import { AccessControl } from '../components/AccessControl';
const CreateEventPage = () => {
  const [form, setForm] = useState({
    title: '',
    description: '',
    location: '',
    date: '',
    endDate: '',
    image: '',
    targetAmount: '',
  });
  const [errors, setErrors] = useState([]);
  const [startDate, setStartDate] = useState('');
  const [endDate, setEndDate] = useState('');
  const [oneDayOnly, setOneDayOnly] = useState(true);

  const { user } = useAuth(); // ← витягуємо користувача з AuthContext
  const navigate = useNavigate();

  const handleChange = (e) => {
    setErrors([]);
    setForm({
      ...form,
      [e.target.name]: e.target.value,
    });
  };
  const handleStartDateChange = (e) => {
    const newStartDate = e.target.value;
    setStartDate(newStartDate);
    if (oneDayOnly) setEndDate(newStartDate);
    setErrors([]);
  };
  const handleSubmit = async (e) => {
    e.preventDefault();
    const validationErrors = [];
    if (!form.title.trim()) validationErrors.push('Назва події обов'язкова.');
    if (!form.description.trim()) validationErrors.push('Опис події обов'язковий.');
    if (!form.location.trim()) validationErrors.push('Локація події обов'язкова.');
    if (!startDate) validationErrors.push('Дата початку обов'язкова.');
    if (!endDate) validationErrors.push('Дата завершення обов'язкова.');
    if (new Date(endDate) < new Date(startDate)) {

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				105
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        validationErrors.push('Дата завершення не може бути раніше за дату по-
чатку.');
```

```

    }

    if (!user || !user.id) {
        validationErrors.push('Користувача не знайдено або він неавторизований.');
```

```

    }

    if (validationErrors.length > 0) {
        setErrors(validationErrors);
        return;
    }

    try {
        if (!user || !user.id) {
            throw new Error('Користувача не знайдено або він неавторизований');
```

```

        }
        const eventData = {
            ...form,
            startDate,
            endDate,
            organizer: user.id,
        };

        await axios.post(`${process.env.REACT_APP_API_URL}/api/events/createEvent`,
eventData);
        navigate('/events');
    } catch (error) {
        const serverError =
            error.response?.data?.message || 'Помилка при створенні події на сер-
вері.';
        setErrors([serverError]);
    }
};

return (
    <AccessControl allowedRoles={['Організатор']}>
    <Container maxWidth="sm" sx={{ py: 6 }}>
    <Paper elevation={3} sx={{ p: 4 }}>
    <Typography variant="h5" fontWeight={600} mb={3} align="center">
        Створити нову подію
    </Typography>

    {errors.length > 0 && (
        <Alert severity="error" sx={{ mb: 2 }}>
        <ul style={{ margin: 0, paddingLeft: '1.2em' }}>
        {errors.map((err, index) => (
            <li key={index}>{err}</li>
        ))}
        </ul>
        </Alert>
    )}

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				106
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    })

    <Box component="form" onSubmit={handleSubmit}>
      <TextField
        fullWidth
        label="Назва події"
        name="title"
        value={form.title}
        onChange={handleChange}
        required
        margin="normal"
      />
      <TextField
        fullWidth
        multiline
        minRows={3}
        label="Опис"
        name="description"
        value={form.description}
        onChange={handleChange}
        margin="normal"
      />
      <TextField
        fullWidth
        label="Локація"
        name="location"
        value={form.location}
        onChange={handleChange}
        margin="normal"
      />

      <TextField
        label="Дата початку"
        type="date"
        value={startDate}
        onChange={handleStartDateChange}
        InputLabelProps={{ shrink: true }}
        required
        fullWidth
        margin="normal"
      />

      <FormControlLabel
        control={
          <Checkbox
            checked={oneDayOnly}
            onChange={(e) => {
              setOneDayOnly(e.target.checked);
              if (e.target.checked) setEndDate(startDate);
            }}
          />
        />

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				107
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    label="Подія триває один день"
  />

  <TextField
    label="Дата завершення"
    type="date"
    value={endDate}
    onChange={(e) => setEndDate(e.target.value)}
    InputLabelProps={{ shrink: true }}
    disabled={oneDayOnly}
    required
    fullWidth
    margin="normal"
  />
  <TextField
    fullWidth
    label="Необхідна сума (грн)"
    name="targetAmount"
    type="number"
    value={form.targetAmount}
    onChange={handleChange}
    required
    margin="normal"
  />

  <TextField
    fullWidth
    label="Посилання на зображення"
    name="image"
    value={form.image}
    onChange={handleChange}
    margin="normal"
  />

  <Button
    type="submit"
    fullWidth
    variant="contained"
    color="primary"
    sx={{ mt: 3 }}
  >
    Створити подію
  </Button>
</Box>
</Paper>
</Container>
</AccessControl>
);
};

```

```
export default CreateEventPage;
```

EditEventPage.jsx

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { TextField, Button, IconButton, Typography, Box } from '@mui/material';
import { useParams, useNavigate } from 'react-router-dom';
import DeleteIcon from '@mui/icons-material/Delete';

const EditEventPage = () => {
  const { id } = useParams();
  const navigate = useNavigate();
  const [description, setDescription] = useState('');
  const [images, setImages] = useState([]);
  const [existingImages, setExistingImages] = useState([]);
  const base = process.env.REACT_APP_API_URL;
  useEffect(() => {
    axios.get(`${base}/api/events/${id}`)
      .then(res => {
        setDescription(res.data.description);

        // одразу нормалізуємо шляхи при завантаженні
        const imgs = (res.data.images || []).map(img =>
          img.startsWith('http') ? img : `${base}/${img}`
        );
        setExistingImages(imgs);
      })
      .catch(err => console.error(err));
  }, [id, base]);
  const handleRemoveImage = (indexToRemove) => {
    setExistingImages(prev =>
      prev.filter((_, index) => index !== indexToRemove)
    );
  };
  const handleSubmit = async (e) => {
    e.preventDefault();
    const formData = new FormData();
    formData.append('description', description);
    formData.append('existingImages', JSON.stringify(existingImages));
    images.forEach((image) => formData.append('images', image));

    try {
      await axios.put(`${process.env.REACT_APP_API_URL}/api/events/${id}`,
        formData, {
          headers: { 'Content-Type': 'multipart/form-data' },
        });
      navigate(`/event/${id}`);
    } catch (error) {
      console.error('Помилка при оновленні:', error);
    }
  };
};
```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		109

```

    }
  };

  return (
    <Box component="form" onSubmit={handleSubmit} sx={{ p: 2 }}>
      <Typography variant="h5">Редагувати подію</Typography>
      <TextField
        fullWidth
        label="Опис"
        value={description}
        multiline
        onChange={(e) => setDescription(e.target.value)}
        sx={{ mt: 2 }}
      />
      <Button variant="contained" component="label" sx={{ mt: 2 }}>
        Завантажити фото
        <input type="file" hidden multiple onChange={(e) => setImages([...e.target.files])} />
      </Button>

      <Box sx={{ mt: 2 }}>
        <Typography>Існуючі фото:</Typography>
        <Box sx={{ display: 'flex', gap: 2, flexWrap: 'wrap' }}>
          {existingImages.map((img, index) => {
            const src = img;
            console.log(src);
            return (
              <Box
                key={index}
                sx={{
                  position: 'relative',
                  width: 100,
                  height: 100,
                  overflow: 'hidden',
                  borderRadius: 2,
                  '&:hover .overlay': { opacity: 1 },
                }}
              >
                <img
                  src={src}
                  alt={`event-${index}`}
                  style={{ width: '100%', height: '100%', objectFit: 'cover' }}
                />

                {/* затемнення + кнопка видалення */}
                <Box
                  className="overlay"
                  onClick={() => handleRemoveImage(index)}
                  sx={{
                    position: 'absolute',
                    inset: 0,

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				110
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        bgcolor: 'rgba(0,0,0,0.55)',
        color: 'common.white',
        display: 'flex',
        alignItems: 'center',
        justifyContent: 'center',
        gap: 1,
        opacity: 0,
        transition: 'opacity .25s',
        cursor: 'pointer',
      }}
    >
      <DeleteIcon fontSize="small" />
      <Typography variant="caption">Видалити</Typography>
    </Box>
  </Box>
);
}}
</Box>
</Box>

  <Button type="submit" variant="contained" sx={{ mt: 3 }}>
    Оновити подію
  </Button>
</Box>
);
};

export default EditEventPage;

```

Event.jsx

```

import React, { useEffect, useState } from 'react';
import { useParams, Link } from 'react-router-dom';
import axios from 'axios';
import {
  Box,
  Typography,
  Container,
  CircularProgress,
  Card,
  CardMedia,
  CardContent,
  Avatar,
  Stack,
  Button,
  Paper,
} from '@mui/material';
import EventIcon from '@mui/icons-material/Event';
import LocationOnIcon from '@mui/icons-material/LocationOn';
import LinearProgress from '@mui/material/LinearProgress';
import { AccessElement } from '../components/AccessControl';

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		111

```

const BASE = process.env.REACT_APP_API_URL;

const EventPage = () => {
  const { id } = useParams();
  const [event, setEvent] = useState(null);
  const [loading, setLoading] = useState(true);

  // Стан для вибраного фото у галереї (нижній блок)
  const [selectedPhoto, setSelectedPhoto] = useState(null);

  useEffect(() => {
    axios.get(`${BASE}/api/events/${id}`)
      .then(res => {
        setEvent(res.data);
        setLoading(false);
        if (res.data.images && res.data.images.length > 0) {
          setSelectedPhoto(res.data.images[0]);
        }
      })
      .catch(err => {
        console.error('Помилка при завантаженні події', err);
        setLoading(false);
      });
  }, [id]);

  if (loading) return <CircularProgress sx={{ mt: 4 }} />;
  if (!event) return <Typography variant="h6">Подію не знайдено</Typography>;

  const approvedVolunteers = event.approvedVolunteers || [];

  // Всі фото крім вибраного
  const otherPhotos = event.images?.filter(img => img !== selectedPhoto) || [];

  return (
    <Container sx={{ py: 6 }}>
      <Card>
        {/* Верхнє велике зображення події – НЕ міняється */}
        {event.image && (
          <CardMedia
            component="img"
            height="400"
            image={event.image.startsWith('http') ? event.image :
`${BASE}/${event.image}`}
            alt={event.title}
            sx={{ objectFit: 'cover' }}
          />
        )}

        <CardContent>
          <Typography variant="h4" fontWeight={700} gutterBottom>

```

Дячук Д.О.

Годлевський Ю.О.

ІПЗ.КР.Б-121-25-ПЗ

Арк.

112

Змн. Арк. № докум. Підпис Дата

```

        {event.title}
      </Typography>

      <Stack direction="row" spacing={2} alignItems="center" mb={2}
flexWrap="wrap">
        <Avatar>
          <EventIcon />
        </Avatar>

        <Typography variant="body1">
          {new Date(event.startDate).toLocaleDateString('uk-UA')} -{' '}
          {new Date(event.endDate).toLocaleDateString('uk-UA')}
        </Typography>

        {event.location && (
          <>
            <Avatar>
              <LocationOnIcon />
            </Avatar>
            <Typography variant="body1">{event.location}</Typography>
          </>
        )}
      </Stack>

      <Box sx={{ my: 3 }}>
        <Typography>
          Зібрано: {event.collectedAmount} / {event.targetAmount} ₴
        </Typography>
        <LinearProgress
          variant="determinate"
          value={Math.min(100, (event.collectedAmount / event.targetAmount) *
100)}

          sx={{ height: 10, borderRadius: 5 }}
        />
      </Box>

      <Stack direction="row" spacing={2} mb={3}>
        <Button variant="contained" component={Link} to="./payment">
          Підтримати
        </Button>
        <AccessElement allowedRoles="Волонтер">
          <Button color="inherit" variant="outlined" component={Link}
to="./apply">
            Подати заявку
          </Button>
        </AccessElement>
      </Stack>

      <Typography variant="body1" mb={2}>
        {event.description}
      </Typography>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				113
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{/* Галерея: ліворуч вибране фото, праворуч – решта */}
<Box sx={{ mt: 2, display: 'flex', gap: 2, flexWrap: 'wrap' }}>
  {/* Ліворуч - вибране фото */}
  <Paper
    elevation={3}
    sx={{
      width: 300,
      height: 300,
      borderRadius: 2,
      overflow: 'hidden',
      flexShrink: 0,
    }}
  >
    {selectedPhoto ? (
      <img
        src={`/${BASE}/${selectedPhoto}`}
        alt="selected"
        style={{ width: '100%', height: '100%', objectFit: 'cover' }}
        loading="lazy"
      />
    ) : (
      <Box
        sx={{
          width: '100%',
          height: '100%',
          display: 'flex',
          justifyContent: 'center',
          alignItems: 'center',
          bgcolor: '#eee',
        }}
      >
        Немає фото
      </Box>
    )}
  </Paper>

  {/* Праворуч – інші фото */}
  <Box
    sx={{
      display: 'flex',
      flexWrap: 'wrap',
      gap: 1,
      flexGrow: 1,
      maxHeight: 200,
      overflowY: 'auto',
    }}
  >
    {otherPhotos.length === 0 && (
      <Typography variant="body2" color="text.secondary" sx={{ p: 1 }}>
        Інших фото немає
      </Typography>
    )}
  </Box>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				114
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    </Typography>
  ))

  {otherPhotos.map((img, i) => (
    <Paper
      key={i}
      elevation={1}
      onClick={() => setSelectedPhoto(img)}
      sx={{
        width: 80,
        height: 80,
        borderRadius: 1,
        overflow: 'hidden',
        cursor: 'pointer',
        '&:hover': { boxShadow: 6, borderColor: 'primary.main' },
        border: '2px solid transparent',
      }}
      component="img"
      src={`/${BASE}/${img}`}
      alt={`thumb-${i}`}
      loading="lazy"
    </>
  ))}
</Box>
</Box>

{/* Список прийнятих волонтерів */}
<Box sx={{ mt: 4 }}>
  <Typography variant="h6" gutterBottom>
    Прийняті волонтери
  </Typography>

  {approvedVolunteers.length === 0 && (
    <Typography variant="body2" color="text.secondary">
      Ще немає підтверджених волонтерів.
    </Typography>
  )}

  <Stack direction="row" spacing={2} flexWrap="wrap">
    {approvedVolunteers.map((v) => (
      <Link key={v._id} to={`/volunteer/${v._id}`} style={{ textDecoration: 'none',
color: 'inherit' }}>
        <Stack alignItems="center" spacing={1} sx={{ width: 90 }}>
          <Avatar
            src={v.photo ? `/${BASE}/uploads/${v.photo}` : undefined}
            alt={v.name}
            sx={{ width: 60, height: 60 }}
          </>
          <Typography variant="body2" align="center" noWrap>
            {v.name}
          </Typography>
        </Stack>
      </Link>
    ))}
  </Stack>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		115

```

        </Stack>
      </Link>
    )))}
      </Stack>
    </Box>
  </CardContent>
</Card>
</Container>
);
};

export default EventPage;

```

Home.jsx

```

import React, { useEffect, useState } from 'react';
import axios from 'axios';
import {
  Box,
  Typography,
  Container,
  Grid,
  Button,
  Card,
  CardContent,
  Avatar,
  CircularProgress,
} from '@mui/material';
import { VolunteerActivism, MonetizationOn, EventAvailable, EmojiPeople } from '@mui/icons-material';
import { HideLoginElement, AccessElement } from '../components/AccessControl';
import { Link } from 'react-router-dom';

const FeatureCard = ({ icon, title, description }) => (
  <Card elevation={3} sx={{ p: 2 }}>
    <CardContent>
      <Avatar sx={{ bgcolor: 'primary.main', mb: 2 }}>{icon}</Avatar>
      <Typography variant="h6" gutterBottom>{title}</Typography>
      <Typography variant="body2" color="text.secondary">{description}</Typogra-
phy>
    </CardContent>
  </Card>
);

const heroImage = '../Images/carousel1.jpg';

const HomePage = () => {
  const [userCount, setUserCount] = useState(null);

  useEffect(() => {

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		116

```

    axios.get(`${process.env.REACT_APP_API_URL}/api/stats/user-count`)
      .then(res => setUserCount(res.data.count))
      .catch(err => console.error('Помилка отримання кількості користувачів',
err));
  }, []);

  return (
    <Box>
      { /* Hero Section with Single Image */ }
      <Box>
        sx={{
          height: '80vh',
          backgroundImage: `url(https://imglife.pravda.com.ua/life/im-
ages/doc/1/6/45928/16e20e3-blagodiynt-1320.jpg)`,
          backgroundSize: 'cover',
          backgroundPosition: 'center',
          position: 'relative',
          display: 'flex',
          alignItems: 'center',
          justifyContent: 'center',
        }}
      >
      <Box>
        sx={{
          position: 'absolute',
          width: '100%',
          height: '100%',
          backgroundColor: 'rgba(0, 0, 0, 0.5)',
          top: 0,
          left: 0,
        }}
      />
      <Container sx={{ position: 'relative', textAlign: 'center', color:
'white' }}>
        <Typography variant="h3" fontWeight={700} gutterBottom>
          Об'єднуємо зусилля заради добра
        </Typography>
        <Typography variant="h6" sx={{ maxWidth: '600px', mx: 'auto' }}>
          Платформа, яка дозволяє кожному створювати, підтримувати чи знаходити
          благодійні ініціативи.
        </Typography>
        <Box mt={4} display="flex" justifyContent="center" gap={2}>
          <HideLoginElement al-
lowedRoles={["Організатор", "Волонтер", "Інвестор"]}>
            <Button variant="contained" color="primary" component={Link}
to="/register">Стати волонтером</Button>
          </HideLoginElement>
            <Button variant="outlined" color="primary" component={Link}
to="/events">Переглянути події</Button>
          </Box>
        </Container>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				117
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</Box>

{/* Кожен знайде себе тут */}
<Container sx={{ py: 8 }}>
  <Typography variant="h4" fontWeight={600} align="center" gutterBottom>
    Кожен знайде себе тут
  </Typography>
  <Grid container spacing={4} mt={2}>
    <Grid item xs={12} md={4}>
      <FeatureCard icon={<EmojiPeople />} title="Я – Волонтер" description="Допомагай фізично або інформаційно." />
    </Grid>
    <Grid item xs={12} md={4}>
      <FeatureCard icon={<MonetizationOn />} title="Я – Інвестор" description="Фінансуй ініціативи з прозорими звітами." />
    </Grid>
    <Grid item xs={12} md={4}>
      <FeatureCard icon={<EventAvailable />} title="Я – Організатор" description="Створюй події, знаходь підтримку." />
    </Grid>
  </Grid>
</Container>

{/* Лічильник користувачів */}
<Box bgcolor="grey.100" py={6} textAlign="center">
  <Container>
    <Typography variant="h5" gutterBottom>
      Приєдналися вже:
    </Typography>
    {userCount !== null ? (
      <Typography variant="h3" color="primary" fontWeight={700}>
        {userCount} користувачів
      </Typography>
    ) : (
      <CircularProgress color="primary" />
    )}
  </Container>
</Box>

<HideLoginElement allowedRoles={['Організатор', 'Волонтер', 'Інвестор']}>
  <Box bgcolor="primary.dark" color="white" py={8} textAlign="center">
    <Container>
      <Typography variant="h4" fontWeight={700} gutterBottom>
        Готові зробити світ кращим?
      </Typography>
      <Typography variant="body1" mb={4}>
        Приєднуйтесь до спільноти небайдужих вже сьогодні.
      </Typography>
      <Button variant="contained" color="secondary">
        Зареєструватися
      </Button>
    </Container>
  </Box>
</HideLoginElement>

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				118
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </Container>
      </Box>
    </HideLoginElement>
  </Box>
);
};

export default HomePage;

```

Login.jsx

```

import React, { useState } from 'react';
import { Container, Typography, TextField, Button, Box } from '@mui/material';
import { useAuth } from '../context/AuthContext';
import { useNavigate } from 'react-router-dom';

const Login = () => {
  const [formData, setFormData] = useState({ email: '', password: '' });
  const { login } = useAuth();
  const navigate = useNavigate();

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

  const handleSubmit = async e => {
    e.preventDefault();
    const success = await login(formData);
    if (success) {
      navigate('/profile');
    } else {
      alert('Невірний email або пароль');
    }
  };

  return (
    <Container maxWidth="sm">
      <Box sx={{ mt: 6 }}>
        <Typography variant="h4" gutterBottom>Login</Typography>
        <form onSubmit={handleSubmit}>
          <TextField
            fullWidth
            margin="normal"
            label="Email"
            name="email"
            value={formData.email}
            onChange={handleChange}
            required
          />
          <TextField
            fullWidth
            margin="normal"

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				119
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        type="password"
        label="Password"
        name="password"
        value={formData.password}
        onChange={handleChange}
        required
      />
      <Button variant="contained" type="submit" fullWidth sx={{ mt: 2 }}>
        Login
      </Button>
    </form>
  </Box>
</Container>
);
};

export default Login;

```

NotFound.jsx

```

// src/pages/NotFound.jsx
import React from 'react';
import { Box, Typography, Button } from '@mui/material';

const NotFound = () => (
  <Box sx={{ p: 4, textAlign: 'center' }}>
    <Typography variant="h3" gutterBottom>
      404
    </Typography>
    <Typography variant="h5" gutterBottom>
      Сторінку не знайдено
    </Typography>

    <Button
      variant="contained"
      href="/"
      sx={{ mt: 3 }}
    >
      На головну
    </Button>
  </Box>
);

export default NotFound;

```

Paymentpage.jsx

```

// src/pages/PaymentPage.jsx
import React, { useState } from 'react';
import {
  Box, TextField, Button, Typography, Alert, CircularProgress

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				120
Змн.	Арк.	№ докум.	Підпис	Дата		

```

} from '@mui/material';
import axios from 'axios';
import { useNavigate, useParams } from 'react-router-dom';
import { useAuth } from '../context/AuthContext';

const PaymentPage = () => {
  const { id } = useParams(); // id події
  const navigate = useNavigate();
  const { user } = useAuth(); // потрібен JWT-cookie
  const [loading, setLoading] = useState(false);
  const [msg, setMsg] = useState({ type: '', text: '' });

  /* локальний state форми */
  const [cardNumber, setCardNumber] = useState('');
  const [expiryDate, setExpiryDate] = useState('');
  const [cvv, setCvv] = useState('');
  const [amount, setAmount] = useState('');

  /** невеличка перевірка картки (дуже спрощена) */
  const isCardDataValid = () =>
    cardNumber.replace(/\s/g, '').length === 16 &&
    /^d{2}\/d{2}$/.test(expiryDate) &&
    /^d{3}$/.test(cvv) &&
    Number(amount) > 0;

  const handlePayment = async (e) => {
    e.preventDefault();
    setMsg({ type: '', text: '' });

    if (!isCardDataValid()) {
      setMsg({ type: 'error', text: 'Перевірте коректність введених даних.' });
      return;
    }

    try {
      setLoading(true);
      await axios.post(
        `${process.env.REACT_APP_API_URL}/api/payments/${id}`,
        { amount },
        { withCredentials: true } //  передає cookie-JWT
      );

      setMsg({ type: 'success', text: 'Дякуємо! Платіж успішно проведено.' });
      setTimeout(() => navigate(`/event/${id}`), 1500); // редирект через 1½ с
    } catch (err) {
      console.error(err);
      const m = err.response?.data?.message || 'Помилка на сервері';
      setMsg({ type: 'error', text: m });
    } finally {
      setLoading(false);
    }
  }
}

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				121
Змн.	Арк.	№ докум.	Підпис	Дата		

```

};

return (
  <Box component="form"
    onSubmit={handlePayment}
    sx={{ maxWidth: 420, mx: 'auto', mt: 5, p: 3,
      border: '1px solid #ccc', borderRadius: 2 }}>

    <Typography variant="h6" gutterBottom>Зробити внесок</Typography>

    {msg.text && (
      <Alert severity={msg.type} sx={{ mb: 2 }}>{msg.text}</Alert>
    )}

    <TextField
      label="Номер картки"
      placeholder="0000 0000 0000 0000"
      fullWidth required sx={{ mb: 2 }}
      value={cardNumber}
      onChange={(e) => setCardNumber(e.target.value)}
    />

    <Box sx={{ display: 'flex', gap: 2, mb: 2 }}>
      <TextField
        label="MM/YY"
        placeholder="12/30"
        required fullWidth
        value={expiryDate}
        onChange={(e) => setExpiryDate(e.target.value)}
      />
      <TextField
        label="CVV"
        placeholder="123"
        required fullWidth
        value={cvv}
        onChange={(e) => setCvv(e.target.value)}
      />
    </Box>

    <TextField
      label="Сума (₾)"
      type="number" fullWidth required sx={{ mb: 3 }}
      value={amount}
      onChange={(e) => setAmount(e.target.value)}
    />

    <Button type="submit" fullWidth variant="contained" disabled={loading}>
      {loading ? <CircularProgress size={26} /> : 'Оплатити'}
    </Button>
  </Box>
);

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				122
Змн.	Арк.	№ докум.	Підпис	Дата		

```
};
```

```
export default PaymentPage;
```

Profile.jsx

```
import { React, useRef } from 'react';
import {
  Container,
  Typography,
  Box,
  Avatar,
  Button,
  Grid,
  Paper,
} from '@mui/material';
import { useAuth } from '../context/AuthContext';
import OrganizerProfile from '../components/profile/OrganizerProfile';
import VolunteerProfile from '../components/profile/VolunteerProfile';
import InvestorProfile from '../components/profile/InvestorProfile';

const Profile = () => {
  const { user, activeRole, uploadPhoto } = useAuth();

  const fileInputRef = useRef();

  const handlePhotoChange = (e) => {
    const file = e.target.files[0];
    if (file && file.type.startsWith('image/')) {
      uploadPhoto(file); // <- функція з контексту або локально
    }
  };

  if (!user || !activeRole) {
    return (
      <Container>
        <Typography variant="h5" sx={{ mt: 4 }}>
          Ви не авторизовані або не обрано роль
        </Typography>
      </Container>
    );
  }

  return (
    <Container maxWidth="lg" sx={{ mt: 4 }}>
      <Typography variant="h4" gutterBottom>
        Мій профіль
      </Typography>

      <Grid container spacing={4}>
        </* LEFT SIDE – Avatar + Info */>
```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				123
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<Grid item xs={12} md={4}>
  <Paper elevation={3} sx={{ p: 3, height: '100%' }}>
    <Box display="flex" flexDirection="column" alignItems="center"
gap={2}>
      <Avatar
        src={user.photo ? `http://localhost:5000/uploads/${user.photo}` :
undefined}
        alt={user.name}
        sx={{ width: 160, height: 160 }}
      />
      <Button onClick={() => fileInputRef.current?.click()} variant="out-
lined" size="small">
        Змінити фото
      </Button>
      <input
        ref={fileInputRef}
        type="file"
        accept="image/*"
        style={{ display: 'none' }}
        onChange={handlePhotoChange}
      />

      <Box textAlign="center" mt={2}>
        <Typography variant="h6">{user.name}</Typography>
        <Typography variant="body2" color="text.secondary">
          {user.email}
        </Typography>
        {user.phone && (
          <Typography variant="body2" color="text.secondary">
            Телефон: {user.phone}
          </Typography>
        )}
        <Typography variant="body2" color="text.secondary" mt={1}>
          Активна роль: <strong>{activeRole}</strong>
        </Typography>
      </Box>
    </Paper>
  </Grid>

  /* RIGHT SIDE – Role-Specific */
  <Grid item xs={12} md={12}>

    {activeRole === 'Організатор' && <OrganizerProfile />}
    {activeRole === 'Волонтер' && <VolunteerProfile />}
    {activeRole === 'Інвестор' && <InvestorProfile />}

  </Grid>
</Grid>
</Container>
);

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				124
Змн.	Арк.	№ докум.	Підпис	Дата		

```
};

export default Profile;
```

PublicOrganizerPage.jsx

```
import { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import axios from 'axios';
import {
  Container, Typography, Avatar, Grid, CircularProgress, Paper, Box, LinearPro-
  gress
} from '@mui/material';
import EventCardForOrganizer from '../components/EventCardForOrganizer';

const PublicOrganizerPage = () => {
  const { id } = useParams();
  const [profile, setProfile] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    axios.get(`${process.env.REACT_APP_API_URL}/api/users/${id}/public`)
      .then(res => {
        setProfile(res.data);
        setLoading(false);
      })
      .catch(() => setLoading(false));
  }, [id]);

  if (loading) return <CircularProgress sx={{ mt: 6 }} />;
  if (!profile) return <Typography variant="h6">Організатора не
  знайдено</Typography>;

  const { user, events } = profile;

  return (
    <Container sx={{ py: 4 }}>
      <Paper sx={{ p: 3, mb: 4 }}>
        <Box sx={{ display: 'flex', alignItems: 'center', gap: 2 }}>
          <Avatar src={user.photo ? `${process.env.REACT_APP_API_URL}/${user.photo}` : undefined}
            sx={{ width: 64, height: 64 }} />
          <Box>
            <Typography variant="h5">{user.name}</Typography>
            <Typography color="text.secondary">Організатор</Typography>
          </Box>
        </Box>
      </Paper>
    </Container>
  );
};
```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				125
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<Typography variant="h6" gutterBottom>Події організатора</Typography>
{events.length === 0 ? (
  <Typography color="text.secondary">Подій ще немає.</Typography>
) : (
  <Grid container spacing={2}>
    {events.map(ev => (
      <Grid item xs={12} sm={6} md={4} key={ev._id}>
        <EventCardForOrganizer
          event={ev}
          onDelete={() => {}}
          hideActions
        />
      </Grid>
    ))}
  </Grid>
)}
</Container>
);
};

export default PublicOrganizerPage;

```

Register.jsx

```

import React, { useState } from 'react';
import {
  Container, Typography, TextField, Button, Box,
  FormGroup, FormControlLabel, Checkbox, InputAdornment, IconButton, FormHelp-
  erText
} from '@mui/material';
import { useAuth } from '../context/AuthContext';
import { useNavigate } from 'react-router-dom';
import { Visibility, VisibilityOff } from '@mui/icons-material';

const roleOptions = ['Організатор', 'Волонтер', 'Інвестор'];

const Register = () => {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    password: '',
    confirmPassword: '',
    phone: '',
    roles: []
  });

  const [showPassword, setShowPassword] = useState(false);
  const [errors, setErrors] = useState({});
  const { register } = useAuth();
  const navigate = useNavigate();

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		126

```

const isValidPassword = password => /^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$/ .test(password);
const isValidPhone = phone => /^+380\d{9}$/ .test(phone);

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

const handlePhoneChange = e => {
  const digits = e.target.value.replace(/\D/g, '').slice(0, 9);
  setFormData(prev => ({ ...prev, phone: '+380' + digits }));
};

const handleRoleChange = e => {
  const { value, checked } = e.target;
  setFormData(prev => ({
    ...prev,
    roles: checked
      ? [...prev.roles, value]
      : prev.roles.filter(role => role !== value)
  }));
};

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

  if (formData.password !== formData.confirmPassword) {
    newErrors.confirmPassword = 'Паролі не збігаються!';
  }

  if (!isValidPassword(formData.password)) {
    newErrors.password = 'Пароль має містити мінімум 8 символів, хоча б одну літеру та одну цифру';
  }

  if (!isValidPhone(formData.phone)) {
    newErrors.phone = 'Телефон має бути у форматі +380XXXXXXXXX';
  }

  setErrors(newErrors);
  if (Object.keys(newErrors).length > 0) return;

  const payload = {
    name: formData.name,
    email: formData.email,
    password: formData.password,
    phone: formData.phone,
    roles: formData.roles
  };
};

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				127
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const success = await register(payload);
if (success) {
  navigate('/profile');
} else {
  alert('Реєстрація не вдалася');
}
};

return (
  <Container maxWidth="sm">
    <Box sx={{ mt: 6 }}>
      <Typography variant="h4" gutterBottom>Реєстрація</Typography>
      <form onSubmit={handleSubmit}>
        <TextField
          fullWidth
          margin="normal"
          label="Ім'я"
          name="name"
          value={formData.name}
          onChange={handleChange}
          required
        />
        <TextField
          fullWidth
          margin="normal"
          label="Email"
          name="email"
          value={formData.email}
          onChange={handleChange}
          required
        />
        <TextField
          fullWidth
          margin="normal"
          label="Номер телефону"
          name="phone"
          defaultValue="+380"
          onChange={handlePhoneChange}
          required
          error={!errors.phone}
          helperText={errors.phone}
        />
        <TextField
          fullWidth
          margin="normal"
          type={showPassword ? 'text' : 'password'}
          label="Пароль"
          name="password"
          value={formData.password}

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				128
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        onChange={handleChange}
        required
        error={!errors.password}
        helperText={errors.password}
        InputProps={{
          endAdornment: (
            <InputAdornment position="end">
              <IconButton onClick={() => setShowPassword(p => !p)}
edge="end">
                {showPassword ? <VisibilityOff /> : <Visibility />}
              </IconButton>
            </InputAdornment>
          )
        }}
      />

      <TextField
        fullWidth
        margin="normal"
        type={showPassword ? 'text' : 'password'}
        label="Повторіть пароль"
        name="confirmPassword"
        value={formData.confirmPassword}
        onChange={handleChange}
        required
        error={!errors.confirmPassword}
        helperText={errors.confirmPassword}
      />

      <FormGroup sx={{ mt: 2 }}>
        <Typography variant="subtitle1" gutterBottom>Обери роль (можна
декілька)</Typography>
        {roleOptions.map(role => (
          <FormControlLabel
            key={role}
            control={
              <Checkbox
                checked={formData.roles.includes(role)}
                onChange={handleRoleChange}
                value={role}
              />
            }
            label={role}
          />
        ))}
      </FormGroup>

      <Button variant="contained" type="submit" fullWidth sx={{ mt: 3 }}>
        Зареєструватись
      </Button>
    </form>
  )}

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				129
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    </Box>
  </Container>
);
};

export default Register;

```

VolunteerApplyPage.jsx

```

import { useParams, useNavigate } from 'react-router-dom';
import { useAuth } from '../context/AuthContext';
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { Paper, TextField, Typography, Button, Box } from '@mui/material';
const BASE = process.env.REACT_APP_API_URL;
export default function VolunteerApplyPage() {
  const { id: eventId } = useParams();
  const { user } = useAuth();
  const navigate = useNavigate();
  const [msg, setMsg] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post(`${BASE}/api/volunteers/applications`, {
        eventId,
        message: msg,
      }, {withCredentials: true});
      navigate(`/event/${eventId}`); // або toast + назад
    } catch (err) {
      console.error(err);
      alert(err.response?.data?.message || 'Помилка');
    }
  };

  return (
    <Paper sx={{ p: 3, maxWidth: 600, mx: 'auto', mt: 4 }}>
      <Typography variant="h5" gutterBottom>Заявка волонтера</Typography>

      <Box component="form" onSubmit={handleSubmit}>
        <TextField
          label="Ім'я"
          fullWidth
          margin="normal"
          value={user.name}
          disabled
        />
        <TextField
          label="Телефон"

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		130

```

        fullWidth
        margin="normal"
        value={user.phone}
        disabled
      />
      <TextField
        label="Е-пошта"
        fullWidth
        margin="normal"
        value={user.email}
        disabled
      />
      <TextField
        label="Чому (і ким) хочу допомагати"
        fullWidth
        margin="normal"
        multiline
        rows={4}
        required
        value={msg}
        onChange={(e) => setMsg(e.target.value)}
      />

      <Button type="submit" variant="contained" sx={{ mt: 2 }}>
        Надіслати
      </Button>
    </Box>
  </Paper>
);
}

```

VolunteerPublicProfile.jsx

```

import React, { useState, useEffect } from 'react';
import { useParams } from 'react-router-dom';
import axios from 'axios';
import {
  Avatar,
  Typography,
  Paper,
  Box,
  Rating,
  TextField,
  Button,
} from '@mui/material';
import dayjs from 'dayjs';

const BASE = process.env.REACT_APP_API_URL;

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		131

```

const VolunteerPublicProfile = () => {
  const { id } = useParams();
  const [vol, setVol] = useState(null);
  const [reviews, setReviews] = useState([]);
  const [text, setText] = useState('');
  const [rate, setRate] = useState(5);

  /* ----- */
  /* helpers                                     */
  /* ----- */

  const fetchReviews = async () => {
    const { data } = await axios.get(
      `${BASE}/api/volunteers/reviews`,
      { params: { volunteer: id }, withCredentials: true }
    );
    setReviews(data);
  };

  /* ----- */
  /* load profile + reviews                     */
  /* ----- */

  useEffect(() => {
    (async () => {
      // публічний профіль
      const u = await axios.get(
        `${BASE}/api/users/${id}/public`,
        { withCredentials: true }
      );
      // бекенд може повернути { user: {...} } або просто {...}
      setVol(u.data.user ?? u.data);

      // відгуки
      await fetchReviews();
    })();
  }, [id]);

  /* ----- */
  /* submit review                             */
  /* ----- */

  const submitReview = async () => {
    if (!text.trim()) return;
    await axios.post(
      `${BASE}/api/volunteers/reviews`,
      { volunteerId: id, text, rating: rate },
      { withCredentials: true }
    );
    setText('');
    await fetchReviews();
  };

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				132
Змн.	Арк.	№ докум.	Підпис	Дата		

```

};

if (!vol) return null;

/* ----- */
/* render */
/* ----- */

return (
  <Paper sx={{ p: 3, maxWidth: 800, mx: 'auto', mt: 4 }}>
    <Box display="flex" gap={2}>
      <Avatar
        src={vol.photo ? `${BASE}/uploads/${vol.photo}` : undefined}
        sx={{ width: 120, height: 120 }}
      />
      <Box>
        <Typography variant="h5">{vol.name}</Typography>
        <Rating value={vol.avgRating || 0} precision={0.5} readOnly />
        {vol.bio && (
          <Typography variant="body2" sx={{ mt: 1 }}>
            {vol.bio}
          </Typography>
        )}
      </Box>
    </Box>

    <Typography variant="h6" sx={{ mt: 3 }}>
      Відгуки
    </Typography>

    {reviews.map((rv) => (
      <Paper key={rv._id} sx={{ p: 2, my: 1 }}>
        <Box display="flex" alignItems="center" gap={1}>
          <Avatar
            src={
              rv.author?.photo
                ? `${BASE}/uploads/${rv.author.photo}`
                : undefined
            }
          />
          <Typography variant="subtitle2">
            {rv.author?.name || '-'}
          </Typography>
          <Rating value={rv.rating} size="small" readOnly />
          <Typography variant="caption" sx={{ ml: 'auto' }}>
            {dayjs(rv.createdAt).format('DD.MM.YYYY')}
          </Typography>
        </Box>
        <Typography variant="body2" sx={{ mt: 1 }}>
          {rv.text}
        </Typography>
      )
    )}
  </Paper>
);

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				133
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </Paper>
      )))}

      <Typography variant="h6" sx={{ mt: 3 }}>
        Залишити відгук
      </Typography>
      <Rating value={rate} onChange={(_, v) => setRate(v || 1)} />
      <TextField
        fullWidth
        multiline
        rows={3}
        sx={{ my: 1 }}
        value={text}
        onChange={(e) => setText(e.target.value)}
      />
      <Button variant="contained" onClick={submitReview}>
        Надіслати
      </Button>
    </Paper>
  );
};

export default VolunteerPublicProfile;

```

App.css

```

.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				134
Змн.	Арк.	№ докум.	Підпис	Дата		

```

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

```

App.jsx

```

import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import { Box } from '@mui/material';

import Navbar from './components/Navbar';
import Footer from './components/Footer';
import { AccessControl } from './components/AccessControl.jsx';
import Home from './pages/Home';
import Register from './pages/Register';
import Login from './pages/Login';
import Profile from './pages/Profile';
import CreateEventPage from './pages/CreateEventPage';
import EventPage from './pages/Event';
import AllEventsPage from './pages/AllEventsPage';
import EditEventPage from './pages/EditEventPage';
import PaymentPage from './pages/PaymentPage';
import PublicOrganizerPage from './pages/PublicOrganizerPage.jsx';
import VolunteerApplyPage from './pages/VolunteerApplyPage';
import ApplicationsManagePage from './pages/ApplicationsManagePage';
import VolunteerPublicProfile from './pages/VolunteerPublicProfile';
import NotFound from './pages/NotFound';

function App() {
  return (
    <Router>
      <Box display="flex" flexDirection="column" minHeight="100vh">
        <Navbar />
        <Box component="main" flex={1}>
          <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/register" element={<Register />} />
            <Route path="/login" element={<Login />} />
            <Route path="/profile" element={<Profile />} />
            <Route path="/events" element={<AllEventsPage />} />
            <Route path="/events/createEvent" element={<CreateEventPage />} />
            <Route path="/event/:id/edit" element={<EditEventPage />} />
          </Routes>
        </Box>
      </Box>
    </Router>
  );
}

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		135

```

        <Route path="/event/:id/payment" element={<PaymentPage />} />
        <Route path="/event/:id" element={<EventPage />} />
        <Route path="/organizer/:id" element={<PublicOrganizerPage />} />
        <Route path="/event/:id/apply" element={
          <AccessControl allowedRoles={['Волонтер']}><VolunteerApplyPage />
        </AccessControl>
        } />
        <Route path="/organizer/applications" element={
          <AccessControl allowedRoles={['Організатор']}><ApplicationsMan-
agePage /></AccessControl>
        } />
        <Route path="/volunteer/:id" element={<VolunteerPublicProfile
/>} />

        <Route path="*" element={<NotFound />} />
      </Routes>
    </Box>

    <Footer />
  </Box>
</Router>
);
}

export default App;

```

index.css

```

body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}

```

Index.js

```

import { AuthProvider } from '../context/AuthContext';
import React from 'react';
import ReactDOM from 'react-dom/client';

```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				136
Змн.	Арк.	№ докум.	Підпис	Дата		

```
import App from './App';
import { ThemeProvider, CssBaseline } from '@mui/material';
import theme from './theme';
import ErrorBoundary from './components/ErrorBoundary';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <ThemeProvider theme={theme}>
      <CssBaseline />
      <ErrorBoundary>
        <AuthProvider>
          <App />
        </AuthProvider>
      </ErrorBoundary>
    </ThemeProvider>
  </React.StrictMode>
);
```

Theme.js

```
// theme.js
import { createTheme } from '@mui/material/styles';

const theme = createTheme({
  palette: {
    primary: {
      main: '#2e7d32', // темно-зелений (подібний до "миру")
      light: '#60ad5e',
      dark: '#1b5e20',
      contrastText: '#fff',
    },
    secondary: {
      main: '#a5d6a7', // світло-зелений
      contrastText: '#000',
    },
    background: {
      default: '#f1f8e9', // дуже світлий зелений/білий
      paper: '#ffffff', // білий для паперових контейнерів
    },
    text: {
      primary: '#2e7d32', // темно-зелений для тексту
      secondary: '#4caf50', // світло-зелений
    },
  },
  typography: {
    fontFamily: "'Roboto', 'Helvetica', 'Arial', sans-serif",
  },
});

export default theme;
```

		Дячук Д.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Годлевський Ю.О.				137
Змн.	Арк.	№ докум.	Підпис	Дата		