

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 «Веб-технології»)

на тему:

«Вебплатформа для організації зборів з елементами гейміфікації»

Виконав студент групи ВТ-22-1
БАЛАН Дмитро Олегович

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
ФАНТ Микола Олександрович

Житомир -2026

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

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БАЛАН Дмитро Олегович**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Вебплатформа для організації зборів з елементами гейміфікації»**,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/С**

Вихідні дані для роботи:

- об'єкт дослідження: процес організації та проведення благодійних зборів у цифровому середовищі
- предмет дослідження: методи та технології розробки вебзастосунків для організації благодійних зборів з елементами гейміфікації із використанням сучасних JavaScript-фреймворків на рівні клієнтської частини (Next.js, React) і серверної частини (NestJS, PostgreSQL, Prisma).

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	ВАКАЛЮК Т.А.	01.03.2026р.	01.03.2026р.
2	ВАКАЛЮК Т.А.	01.04.2026р.	01.04.2026р.
3	ВАКАЛЮК Т.А.	01.05.2026р.	01.05.2026р.

РЕФЕРАТ

У роботі розглянуто розробку вебплатформи для організації зборів з елементами гейміфікації. Метою роботи є створення вебзастосунку, що забезпечує зручне створення та управління зборами коштів, прозорість благодійної діяльності і підвищення залученості користувачів через ігрові механіки.

Об'єктом дослідження є процес організації та проведення благодійних зборів у цифровому середовищі, а предметом - методи та засоби розробки вебзастосунків із використанням сучасних технологій веброзробки.

Під час виконання роботи проаналізовано існуючі платформи для збору коштів, зокрема United24, HelpUA Foundation та Donatello, виявлено проблеми недостатньої прозорості використання коштів, відсутності структурованої звітності та низького рівня залученості користувачів. Для реалізації системи обрано стек технологій Next.js, React та Tailwind CSS для клієнтської частини, а також NestJS, Prisma та PostgreSQL для серверної логіки та роботи з даними. Автентифікацію реалізовано на основі JWT з передачею через httpOnly cookie та підтримкою входу через Google OAuth2. Для розмежування доступу використовується модель ролей: користувач, волонтер та адміністратор.

У системі передбачено механізм верифікації волонтерів: користувач подає заявку з персональними даними та підтверджуючими документами, після чого адміністрація здійснює перевірку та надає відповідний статус. Верифіковані волонтери отримують можливість створювати благодійні збори і супроводжувати їх звітністю.

Для підвищення залученості користувачів реалізовано систему гейміфікації, що включає 62 досягнення, які нараховуються за здійснення донатів, активність на платформі та взаємодію з профілем. Передбачено

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебплатформа для організації зборів з елементами гейміфікації	Лім.	Арк.	Аркушів
Розроб.		Балан Д.О.						
Перевір.		Вакалюк Т.А.					3	132
Керівник						ФІКТ Гр. ВТ-22-1		
Н. контр.								
Зав. каф.								

чотири рівні донора - bronze, silver, gold та platinum - а також таблицю лідерів і кастомізацію картки донора.

Обробку платежів реалізовано через платіжну систему WayForPay з використанням HMAC-MD5 підпису для верифікації транзакцій. Для забезпечення сповіщень у реальному часі реалізовано WebSocket-з'єднання, через яке користувачі отримують повідомлення про нові бейджі, відповіді адміністратора та зміну статусу верифікації.

Процес розробки організовано з використанням системи контролю версій Git, конвеєрів CI/CD на базі GitHub Actions та GitLab CI, а також контейнеризації засобами Docker. Якість коду перевіряється автоматично за допомогою SonarCloud. Застосунок розгорнуто у хмарному середовищі Railway та доступний за публічним посиланням.

Повний обсяг роботи становить 76 сторінок основного тексту, 4 таблиці, 31 ілюстрацію в основному тексті, 1 рисунок у додатку, 2 додатки та 30 літературних джерел.

Результатом роботи є повнофункціональний вебзастосунок, розгорнутий у виробничому середовищі, що забезпечує централізоване управління благодійними зборами, прозорість фінансових операцій та інструменти підвищення залученості користувачів.

Ключові слова: ВЕБПЛАТФОРМА, БЛАГОДІЙНІ ЗБОРИ, ГЕЙМІФІКАЦІЯ, ВЕРИФІКАЦІЯ, WAYFORPAY.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ABSTRACT

This paper presents the development of a web platform for organizing charitable fundraising campaigns with gamification elements. The aim of the work is to create a web application that simplifies the creation and management of fundraising campaigns, ensures transparency of charitable activities, and increases user engagement through game-like mechanics.

The object of the research is the process of organizing and conducting charitable fundraising in a digital environment, while the subject is the methods and tools for developing web applications using modern web technologies.

During the research, existing fundraising platforms were analyzed, including United24, HelpUA Foundation, and Donatello. Key problems identified were insufficient transparency in fund usage, lack of structured reporting, and low user engagement. The system is implemented using Next.js, React, and Tailwind CSS for the frontend, along with NestJS, Prisma, and PostgreSQL for the backend and data management. Authentication is based on JWT stored in httpOnly cookies with support for Google OAuth2. Role-based access control is implemented with three roles: user, volunteer, and administrator.

The platform includes a volunteer verification mechanism: users submit an application with personal information and supporting documents, which is reviewed by an administrator before granting volunteer status. Verified volunteers can create fundraising campaigns and publish financial reports.

A gamification system is implemented to increase user engagement, featuring 62 achievements awarded for donations, platform activity, and profile interactions. Four donor levels are provided -- bronze, silver, gold, and platinum -- along with a leaderboard and donor card customization.

Payment processing is handled through the WayForPay payment system using HMAC-MD5 signature verification. Real-time notifications are delivered via WebSocket connections, informing users about new achievements, administrator messages, and verification status updates.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		5

The development process is organized using Git version control, CI/CD pipelines built on GitHub Actions and GitLab CI, and containerization with Docker. Code quality is automatically analyzed using SonarCloud. The application is deployed in the Railway cloud environment and is accessible via a public URL.

The total volume of the work is 76 pages of main text, 4 tables, 31 illustrations in the main text, 1 figure in the appendix, 2 appendices, and 30 references.

The result of the work is a fully functional web application deployed in a production environment, providing centralized management of fundraising campaigns, financial transparency, and tools for increasing user engagement.

Keywords: WEB PLATFORM, CHARITABLE FUNDRAISING, GAMIFICATION, VERIFICATION, WAYFORPAY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	8
ВСТУП	9
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБ-ПЛАТФОРМИ БЛАГОДІЙНИХ ЗБОРІВ	12
1.1 Постановка задачі	12
1.2 Аналіз аналогів програмного продукту	14
1.3 Вибір архітектури веб-платформи для організації благодійних зборів з елементами гейміфікації	25
1.4 Обґрунтування вибору технологій реалізації платформи	27
Висновки до першого розділу	29
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ ДЛЯ ОРГАНІЗАЦІЇ БЛАГОДІЙНИХ ЗБОРІВ	31
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	31
2.2 Проєктування бази даних та взаємодія з API	35
2.3 Проєктування та реалізація алгоритмів роботи системи.....	40
2.4 Реалізація функціональних можливостей вебплатформи	45
Висновки до другого розділу.....	49
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ДЛЯ ОРГАНІЗАЦІЇ БЛАГОДІЙНИХ ЗБОРІВ	51
3.1 Порядок встановлення та налаштування параметрів системи.....	51
3.2 Структура та опис інтерфейсу вебплатформи	56
3.3 Тестування вебплатформи	65
Висновки до третього розділу	69
ВИСНОВКИ	71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	73
ДОДАТКИ	76

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

IC - інформаційна система
UI - User Interface
UX - User Experience
UI/UX - User Interface / User Experience
API - Application Programming Interface
REST - Representational State Transfer
DB - Database
SQL - Structured Query Language
ORM - Object-Relational Mapping
SSR - Server-Side Rendering
CSR - Client-Side Rendering
JWT - JSON Web Token
HTTP - HyperText Transfer Protocol
RBAC - Role-Based Access Control
MVP - Minimum Viable Product
CRUD - Create, Read, Update, Delete
KPI - Key Performance Indicators
CI/CD - Continuous Integration / Continuous Delivery
OAuth2 - Open Authorization 2.0
HMAC - Hash-based Message Authentication Code
MD5 - Message-Digest Algorithm 5

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

ВСТУП

З початком повномасштабного вторгнення 2022 року благодійна діяльність в Україні набула безпрецедентних масштабів. Мільйони громадян регулярно долучаються до зборів коштів на потреби армії, медичну допомогу, гуманітарну підтримку та відновлення постраждалих громад [1]. Значна частина таких зборів організовується через соціальні мережі та месенджери, що забезпечує швидке поширення інформації, але водночас створює низку проблем, пов'язаних із недостатньою структурованістю звітності, складністю перевірки достовірності організаторів і відсутністю єдиного механізму відстеження стану збору. Централізовані платформи на зразок United24 підвищують рівень довіри до благодійної діяльності, однак не завжди дають змогу користувачу простежити зв'язок між конкретним внеском і конкретним результатом.

Актуальність розробки вебплатформи для організації благодійних зборів зумовлена потребою у програмному рішенні, яке поєднує створення окремих кампаній, верифікацію волонтерів, прозоре відображення інформації про збори та інструменти підвищення залученості користувачів. Особливе значення має можливість формування довіри між організаторами зборів і донорами, оскільки саме прозорість, регулярне оновлення інформації та наявність підтверджуючих матеріалів впливають на готовність користувачів підтримувати благодійні ініціативи. Використання елементів гейміфікації у вигляді досягнень, рівнів донора, таблиці лідерів і кастомізації картки користувача є обґрунтованим підходом до підвищення мотивації та формування сталої участі в благодійній діяльності [2; 3; 4; 5].

Метою роботи є розробка та програмна реалізація вебплатформи для організації благодійних зборів з елементами гейміфікації, що забезпечує створення та супровід зборів, верифікацію волонтерів, прозору обробку фінансових операцій і підвищення залученості користувачів.

Досягнення поставленої мети передбачає аналіз предметної області та існуючих програмних аналогів платформ для збору коштів, визначення їх

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

функціональних обмежень і формування вимог до власного програмного продукту. У межах роботи необхідно обґрунтувати архітектуру системи та вибір технологічного стеку, спроектувати схему бази даних з урахуванням зв'язків між сутностями, реалізувати автентифікацію та авторизацію користувачів на основі JWT з підтримкою входу через Google OAuth2, а також розробити модуль верифікації волонтерів із можливістю обміну повідомленнями між користувачем та адміністратором. Також робота передбачає інтеграцію з платіжною системою WayForPay для обробки донатів, впровадження системи гейміфікації з досягненнями, рівнями донора та таблицею лідерів, реалізацію сповіщень у реальному часі на основі WebSocket, налаштування процесів CI/CD з використанням GitHub Actions і GitLab CI, контейнеризацію застосунку засобами Docker та тестування основних сценаріїв роботи платформи.

Об'єктом дослідження є процес організації та проведення благодійних зборів у цифровому середовищі.

Предметом дослідження є методи та технології розробки вебзастосунків для організації благодійних зборів з елементами гейміфікації із використанням сучасних JavaScript-фреймворків на рівні клієнтської частини та серверної частини.

Під час виконання роботи застосовано методи системного аналізу для дослідження предметної області та наявних програмних рішень, методи об'єктно-орієнтованого проєктування для побудови структури програмного комплексу, методи реляційного моделювання для проєктування схеми бази даних, методи компонентного проєктування для організації клієнтської частини та методи модульного тестування для перевірки коректності реалізованих компонентів.

Практичне значення одержаних результатів полягає в тому, що розроблений застосунок є повнофункціональною вебплатформою, розгорнутою у хмарному середовищі Railway та доступною за публічним посиланням. Платформа може використовуватися як основа для реального

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

сервісу організації благодійних зборів, а також як інфраструктурне рішення для волонтерських організацій, яким потрібні інструменти управління кампаніями, взаємодії з донорами, перевірки волонтерів і підвищення довіри до благодійної діяльності.

Випускна кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Повний обсяг роботи становить 76 сторінок основного тексту. Робота містить 4 таблиці, 31 ілюстрацію в основному тексті, 1 рисунок у додатку та 30 літературних джерел.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				11
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБ-ПЛАТФОРМИ БЛАГОДІЙНИХ ЗБОРІВ

1.1 Постановка задачі

Веб-платформа для організації благодійних зборів є складною інформаційною системою, яка поєднує функціональні можливості для створення, управління та контролю зборів коштів у цифровому середовищі. В умовах активної цифровізації суспільства такі системи набувають особливої актуальності, оскільки дозволяють значно спростити процес залучення фінансової допомоги та забезпечити доступ до благодійної діяльності широкому колу користувачів незалежно від їх місцезнаходження.

Подібні платформи об'єднують у собі інструменти взаємодії між користувачами, механізми перевірки достовірності зборів, а також засоби обробки фінансових операцій. Важливою особливістю є забезпечення прозорості процесу збору коштів, що досягається через надання звітності, підтверджуючих матеріалів та постійного оновлення інформації про стан збору. Таким чином, система виступає як єдина цифрова екосистема, що охоплює повний цикл благодійної діяльності -від створення збору до його завершення та звітування.

Архітектурно система складається з декількох взаємопов'язаних модулів, кожен з яких виконує окрему функціональну роль. Основним є модуль управління благодійними зборами, який забезпечує створення, редагування та перегляд кампаній. У межах цього модуля користувачі можуть додавати опис збору, встановлювати цільову суму, змінювати статус, а також публікувати оновлення та звітні матеріали.

Модуль управління користувачами відповідає за реєстрацію, автентифікацію та розмежування прав доступу. У системі передбачено декілька ролей користувачів, зокрема звичайні користувачі, волонтери та адміністратори. Це дозволяє обмежувати доступ до функціоналу відповідно до ролі та забезпечує контроль над діями користувачів у системі.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Окрему роль відіграє модуль верифікації волонтерів, який реалізує процес подання заявки на отримання відповідного статусу. Користувач надає інформацію про себе та підтверджуючі документи, після чого адміністрація здійснює перевірку та приймає рішення. Такий підхід дозволяє підвищити довіру до платформи та мінімізувати ризики шахрайства.

Модуль фінансових операцій забезпечує можливість здійснення донатів через інтеграцію з платіжною системою WayForPay. При цьому платформа не зберігає банківські дані користувачів, що підвищує рівень безпеки та знижує ризики витоку конфіденційної інформації.

Для підвищення активності користувачів у системі реалізовано модуль гейміфікації, який включає механізм досягнень. Користувачі отримують винагороди за виконання певних дій, таких як здійснення донатів, регулярність участі, перегляд зборів або взаємодія з профілем. Такий підхід базується на принципах сучасних цифрових сервісів та ігор, що дозволяє підвищити рівень залученості користувачів.

Інтерфейс користувача розробляється з урахуванням сучасних підходів UI/UX, що забезпечує зручність використання та адаптивність для різних типів пристроїв. Це дозволяє створити інтуїтивно зрозуміле середовище для взаємодії з платформою.

Серверна частина системи реалізується із використанням сучасних технологій, що забезпечують обробку запитів, роботу з базою даних та реалізацію бізнес-логіки. Взаємодія між клієнтською та серверною частинами здійснюється через API, що дозволяє забезпечити гнучкість та масштабованість системи.

Таким чином, у межах даної роботи ставиться задача розробки веб-платформи, яка забезпечує ефективну організацію благодійних зборів, прозорість фінансових операцій, контроль доступу користувачів та підвищення їхньої активності за рахунок впровадження сучасних технологічних рішень.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

1.2 Аналіз аналогів програмного продукту

Під час проєктування веб-платформи для організації благодійних зборів доцільно проаналізувати вже існуючі рішення, які використовуються в Україні для збору коштів на військові, гуманітарні, медичні та соціальні потреби. З 2022 року онлайн-донати стали одним із важливих інструментів підтримки армії, волонтерських ініціатив та постраждалих громадян, тому відповідні сервіси повинні забезпечувати не лише зручність переказу коштів, а й прозорість, довіру та зрозумілу систему звітності.

До основних аналогів можна віднести спеціалізовані благодійні платформи, окремі сервіси для прийому донатів, а також соціальні мережі та месенджери, які на практиці часто використовуються як основний канал для поширення зборів.

Одним із найбільш відомих прикладів централізованої благодійної платформи є United24. Це офіційний сервіс, через який користувачі можуть здійснювати донати на оборону, медичну допомогу та відновлення України. На платформі подано напрями зборів, звіти та загальну інформацію про використання залучених коштів; у звітах за напрямом Defense, зокрема, відображаються загальні суми надходжень і основні категорії закупівель, серед яких БПЛА, морські дрони, спеціальний транспорт та інше обладнання.

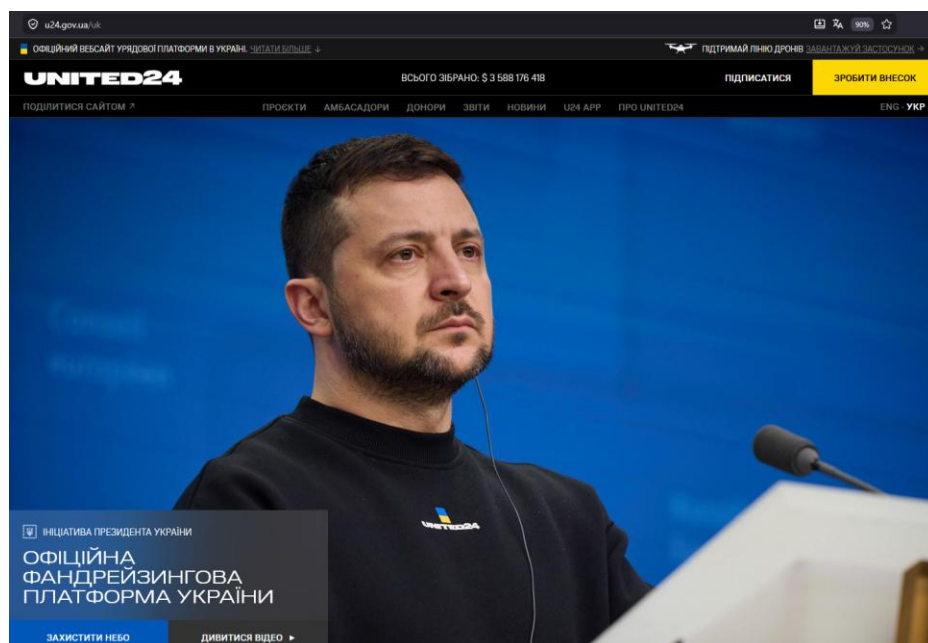


Рис. 1.1. Головна сторінка United24

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Перевагою United24 є високий рівень довіри, офіційний статус, зручний інтерфейс та можливість швидко здійснити платіж. Користувач може обрати напрям допомоги та зробити внесок без складної реєстрації. Також платформа регулярно публікує новини та звіти щодо результатів зборів.

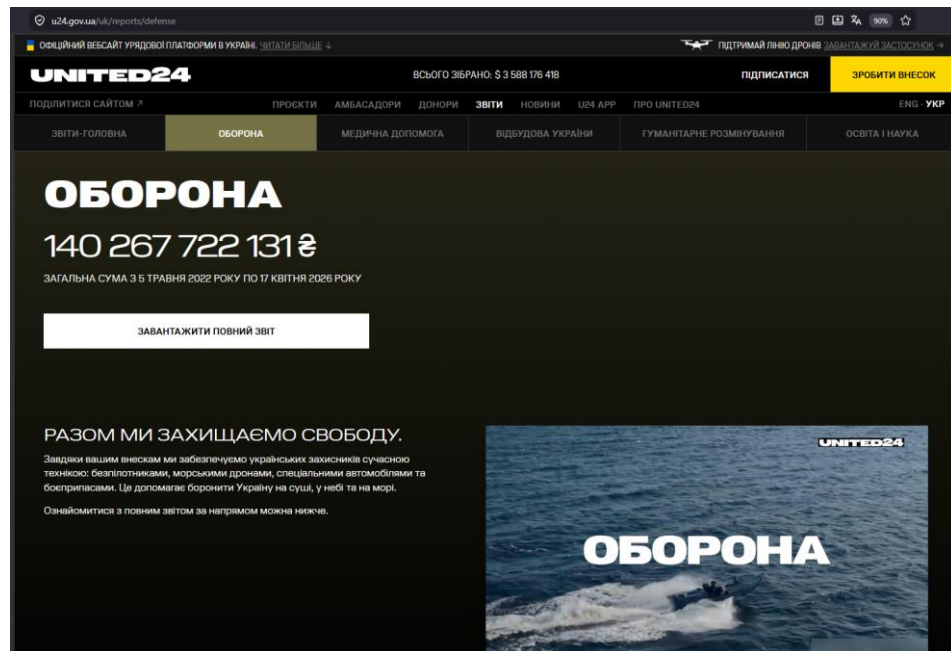


Рис. 1.2. Сторінка звіту щодо збору на оборону

Водночас такий підхід має певні обмеження. Збори на платформі часто мають агрегований характер, тобто кошти багатьох користувачів об'єднуються в межах великого напрямку або програми. Через це звичайний користувач не завжди може чітко простежити зв'язок між власним донатом і конкретним результатом, наприклад з тим, яка саме частина внеску була спрямована на певну закупівлю. Звіти надають загальне уявлення про використання коштів, однак не завжди дозволяють детально відстежити шлях кожного окремого внеску.

Це не зменшує значення платформи, однак показує, що для менших волонтерських зборів може бути корисною система з більш деталізованою структурою, яка передбачає окрему сторінку збору, конкретну ціль, регулярні оновлення, документи, підтвердження закупівель та фінальну звітність.

Платформа HelpUA Foundation орієнтована на підтримку волонтерських ініціатив і подає збори у більш наближеному до практичної благодійної

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

діяльності форматі. На сайті відображаються активні кампанії, загальна сума зібраних коштів, кількість закритих зборів та інша статистика, що дає користувачу загальне уявлення про поточний стан роботи сервісу.

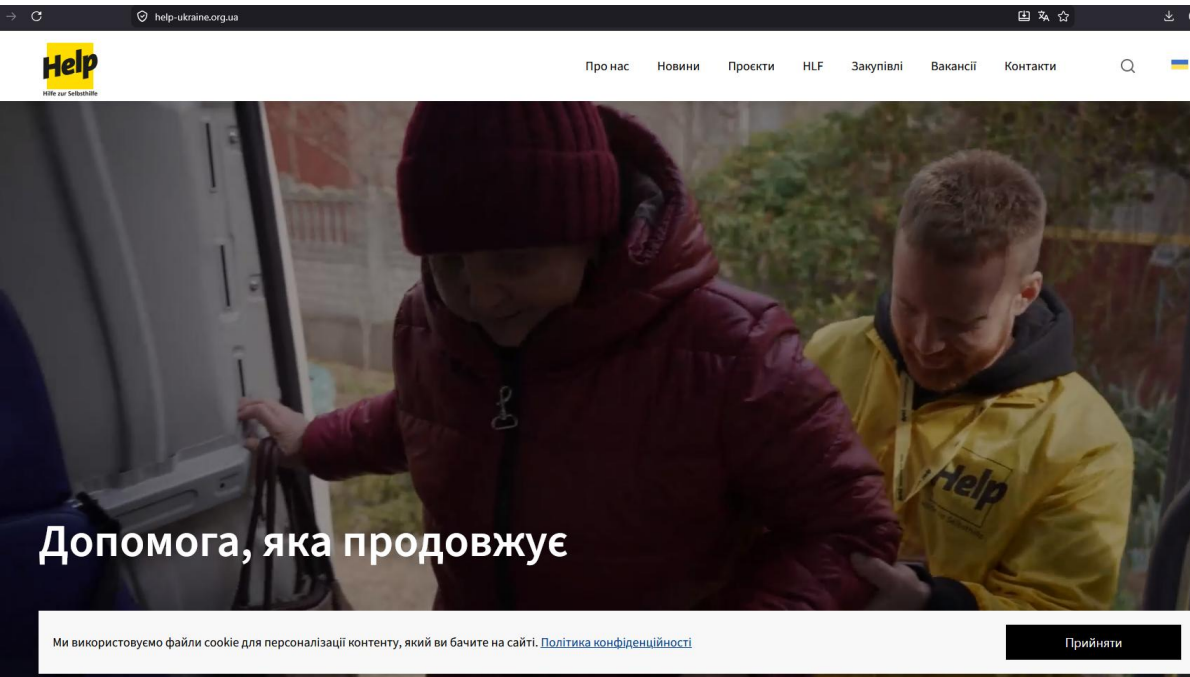


Рис. 1.3. Головна сторінка HelpUA Foundation

За концепцією HelpUA Foundation ближча до розроблюваної системи, оскільки вона працює не лише як загальний фонд, а як платформа з окремими зборами. Користувач може переглядати конкретні кампанії, обирати потрібний збір та бачити певну інформацію про його стан.

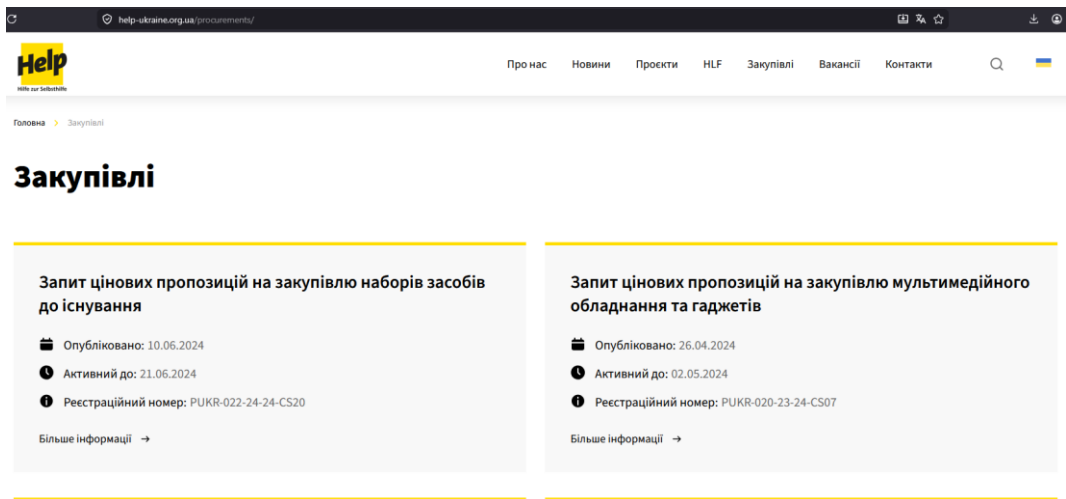


Рис. 1.4. Сторінка активних зборів на платформі

Сильними сторонами такого рішення є зрозуміла структура, наявність окремих зборів і модель роботи, наближена до волонтерської діяльності.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Водночас для користувача, який хоче простежити повний шлях збору, звітність не завжди є достатньо деталізованою, оскільки важливо бачити автора кампанії, надані документи, додані оновлення, здійснені закупівлі, строки їх виконання та прикріплені підтвердження.

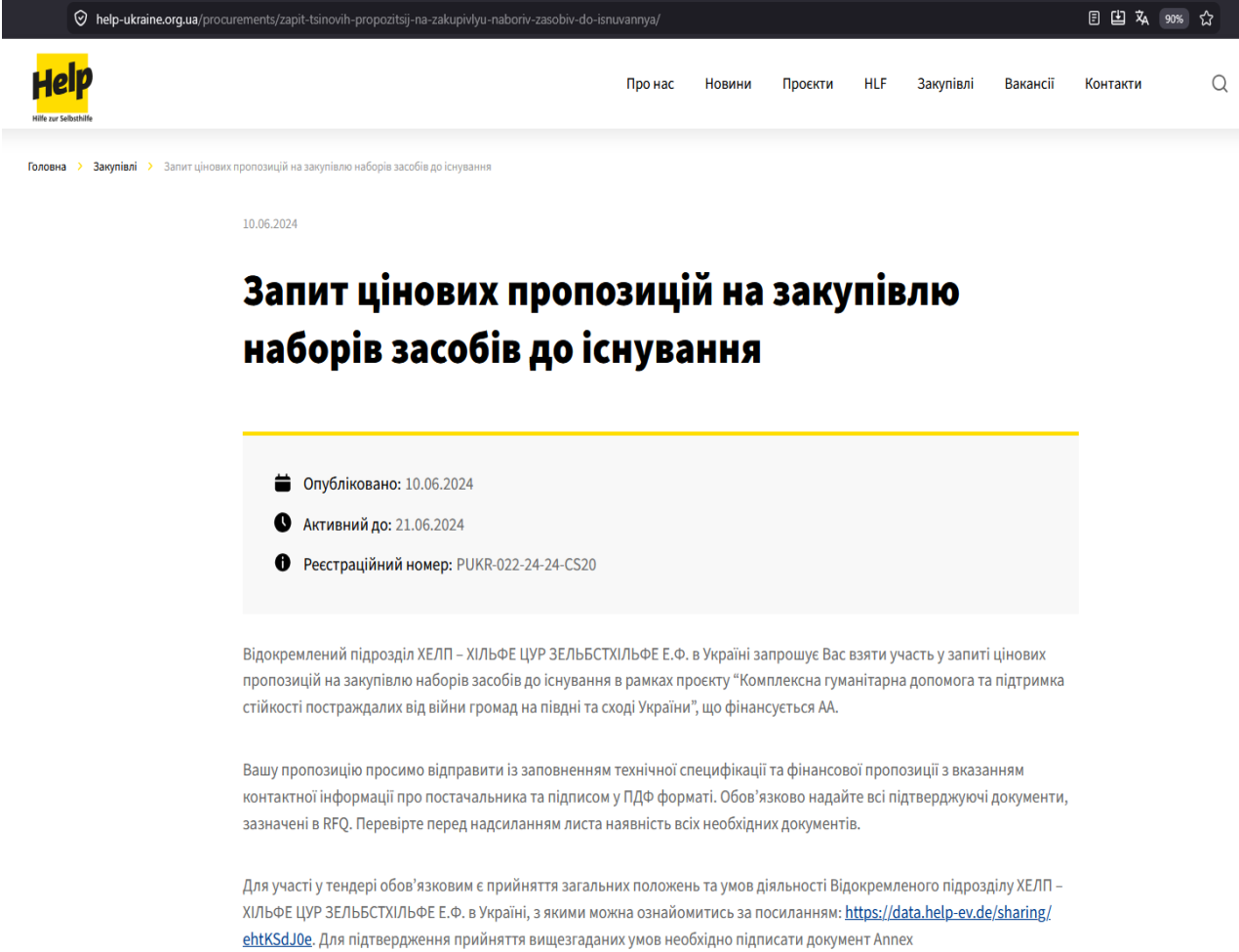


Рис. 1.5. Сторінка конкретного збору на платформі

Таким чином, HelpUA Foundation можна розглядати як близький аналог, однак розроблювана платформа має бути орієнтована на більш структуровану звітність, чітке розмежування ролей користувачів та додаткову мотивацію через елементи гейміфікації.

Окрему групу аналогів становлять сервіси для прийому донатів, серед яких можна виділити Donatello. Це українська платформа донатів і регулярних підписок, орієнтована переважно на підтримку авторів, блогерів, стрімерів та медіаконтенту.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		17

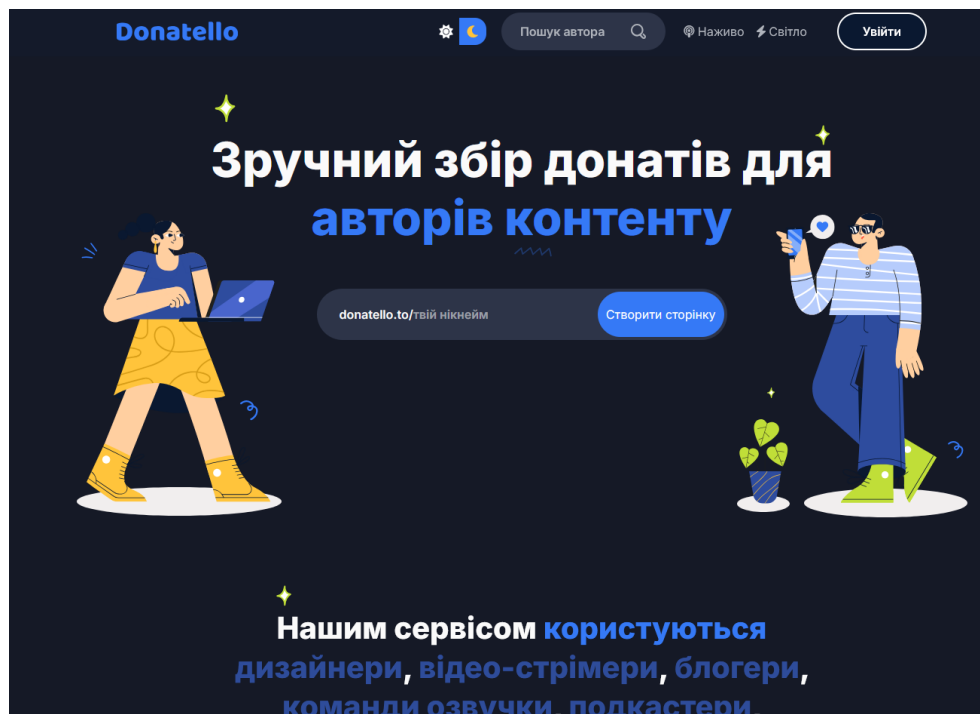


Рис. 1.6. Головна сторінка Donatello

Donatello вирізняється простотою використання, оскільки користувач може створити сторінку для отримання донатів, а платники мають змогу швидко переказати кошти. У деяких випадках сторінки також містять рейтинги донатів, повідомлення від користувачів та елементи взаємодії з аудиторією.

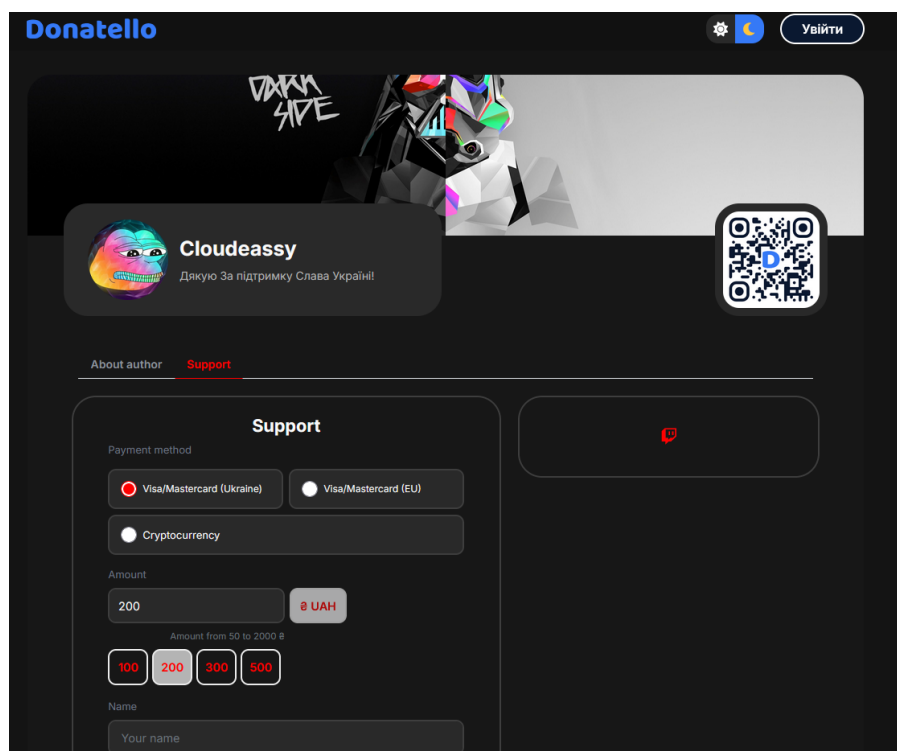


Рис. 1.7. Приклад сторінки донату

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Однак Donatello не є повноцінною платформою для організації благодійних зборів. Його основне призначення - прийом фінансової підтримки, а не управління кампаніями. У сервісі відсутня повноцінна система верифікації волонтерів, модуль звітності за конкретними зборами, контроль статусів кампаній та структура, орієнтована саме на благодійні потреби.

Тому Donatello можна розглядати не як прямий аналог, а як приклад окремого платіжного інструменту, який вирішує лише частину задачі - прийом донатів.

Для обробки платежів у благодійних системах часто використовуються спеціалізовані платіжні сервіси. Одним із таких рішень є LiqPay, який надає платіжні інструменти для благодійних організацій, підтримує донати на довільну суму, можливість регулярних платежів та оплату різними способами.

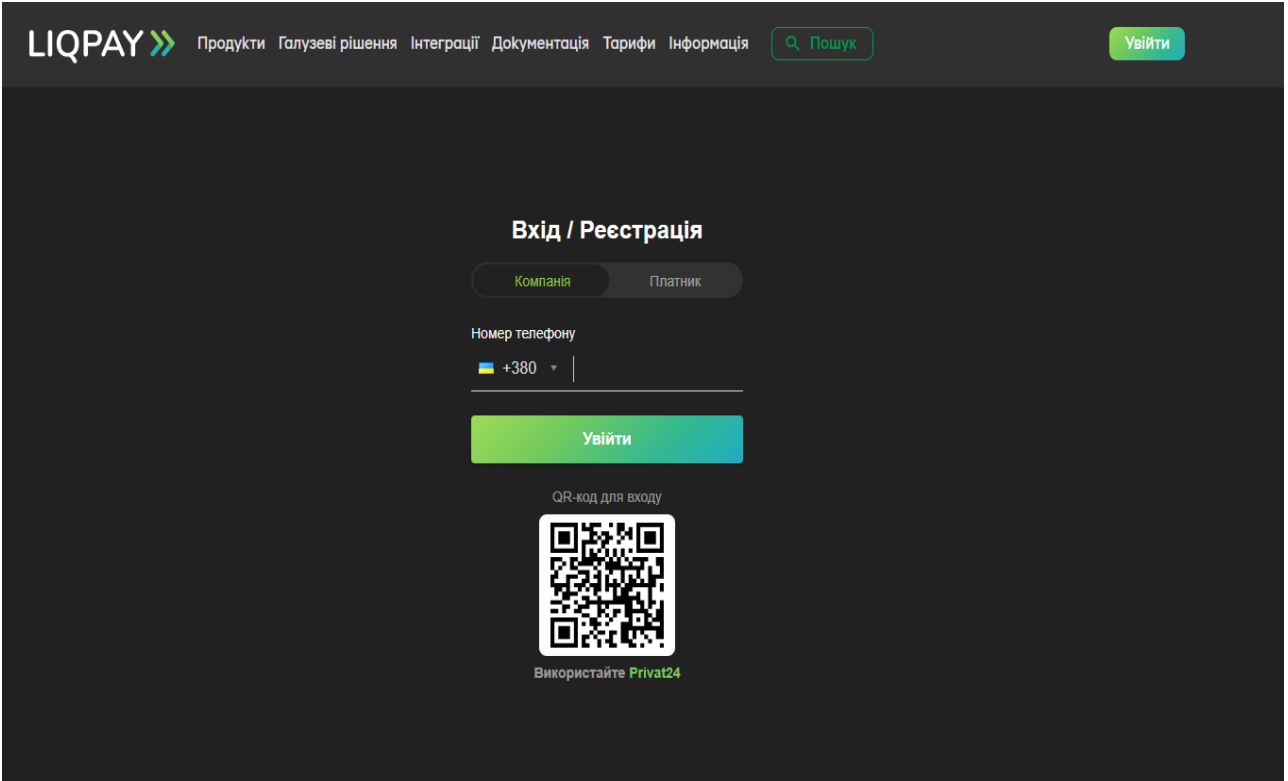


Рис. 1.8. Головна сторінка LiqPay

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		19

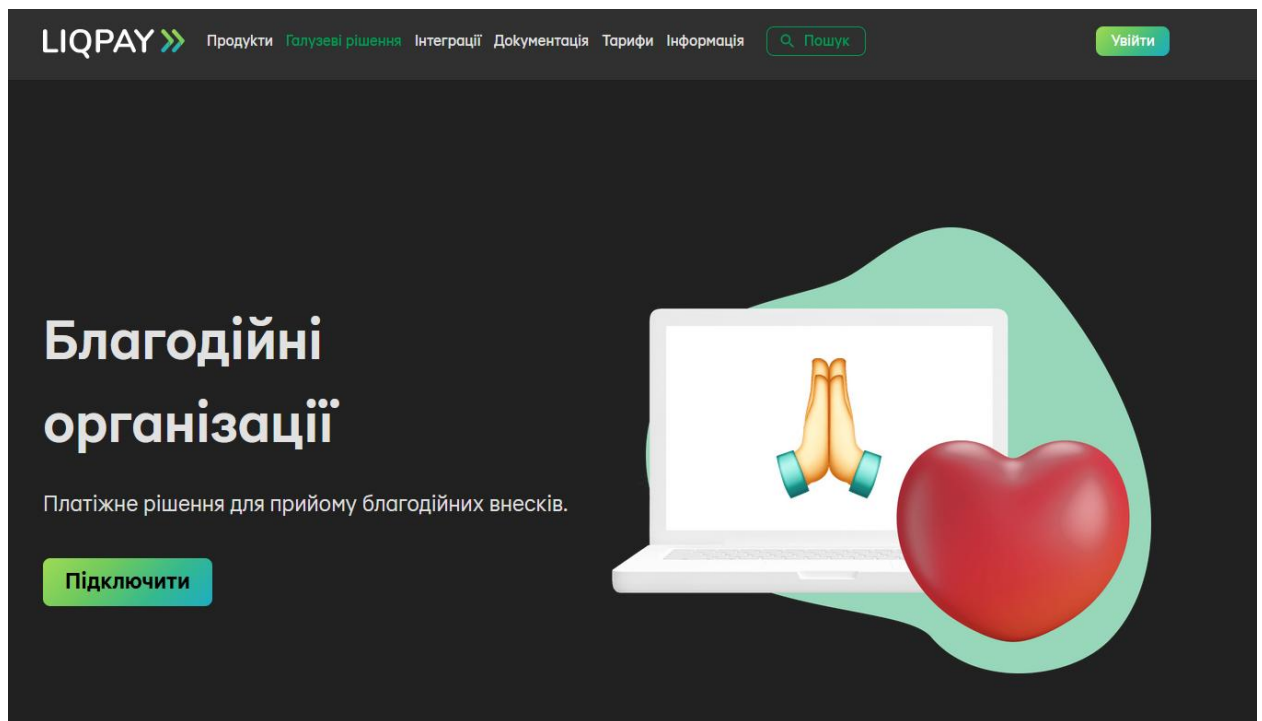


Рис. 1.9. Сторінка для підключення благодійних внесків

LiQPay доцільно використовувати не як окрему благодійну платформу, а як інструмент інтеграції платежів у власну систему. Його перевагою є те, що платформа не повинна самостійно зберігати банківські дані користувачів. Це підвищує рівень безпеки та зменшує технічні ризики під час роботи з фінансовими операціями.

Водночас LiQPay не вирішує задачі управління зборами, перевірки волонтерів, публікації звітів або взаємодії між користувачами. Саме тому для розроблюваної платформи LiQPay доцільно розглядати як допоміжний сервіс для безпечної обробки платежів.

На практиці значна частина зборів в Україні проводиться через Telegram-канали. Волонтери, військові, громадські діячі або організації публікують пости з описом потреби, сумою збору, банківськими реквізитами або посиланням на банку. Такий формат є поширеним, оскільки Telegram дозволяє швидко поширювати інформацію та оперативно оновлювати аудиторію.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		20

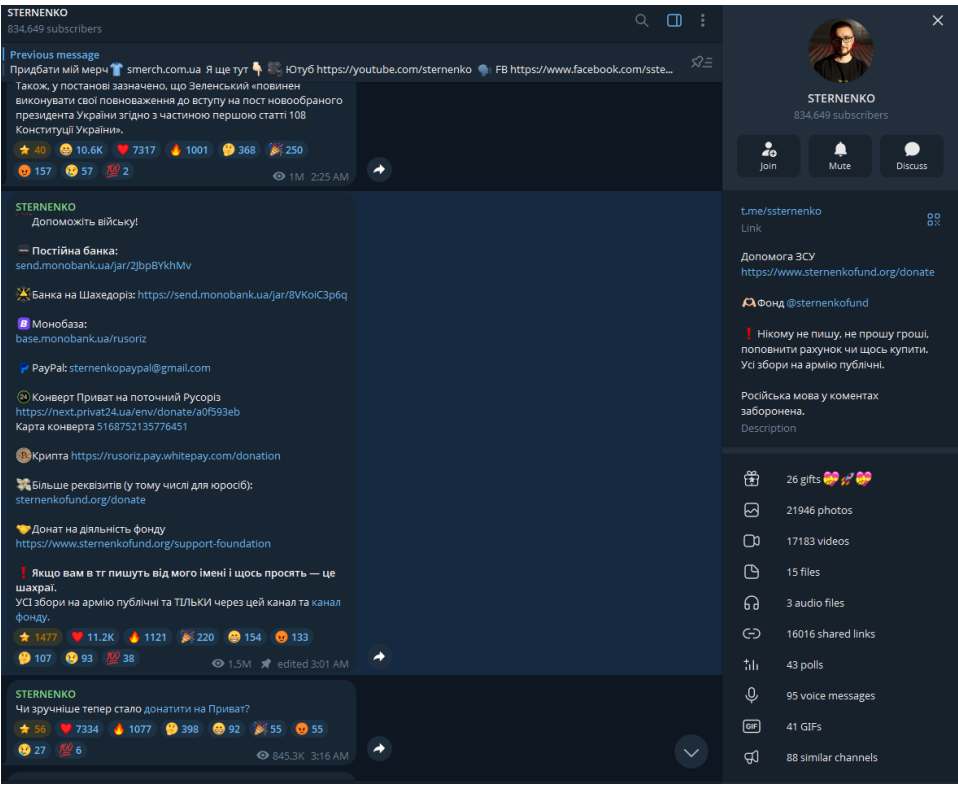


Рис. 1.10. Приклад збору в телеграм каналі

Перевагою Telegram є швидкість комунікації. Автор збору може швидко повідомити аудиторію про нову потребу, додати фото, відео, документи або проміжний звіт. Також користувачі можуть поширювати пости між каналами, що допомагає швидше закривати збори.

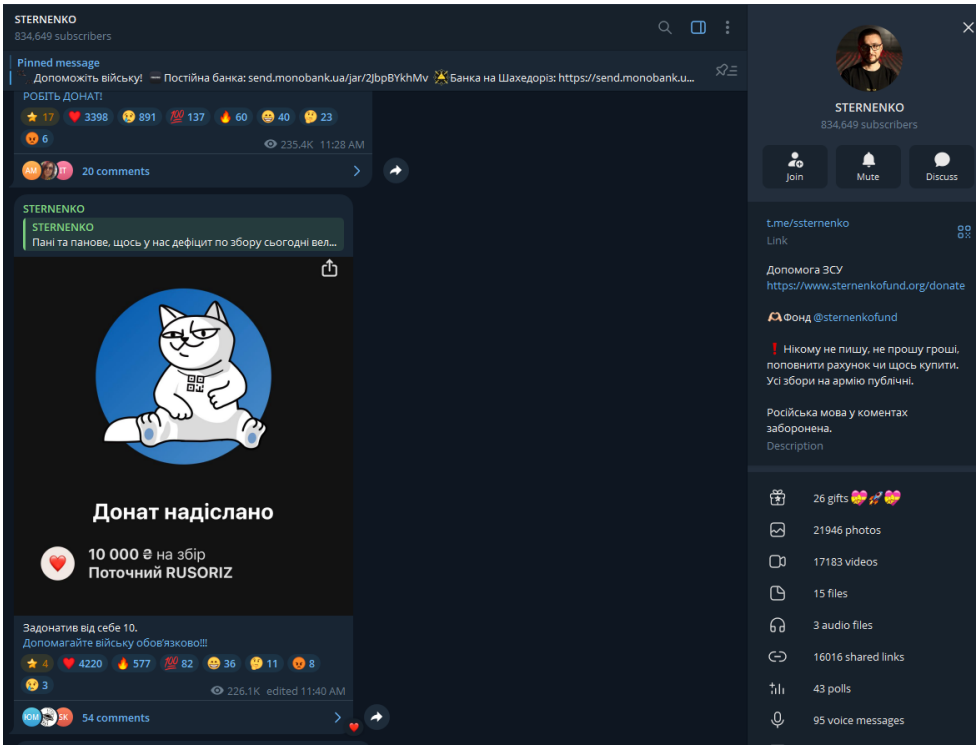


Рис. 1.11. Приклад звітності телеграм збору

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Однак Telegram не є спеціалізованою системою для благодійних зборів. Інформація подається у довільному вигляді, структура збору залежить від самого автора, а звітність може бути розкидана між різними постами. Користувачу складно швидко знайти всі оновлення, перевірити статус збору або зрозуміти, які саме документи підтверджують витрати.

Крім того, Telegram не має вбудованого механізму верифікації організаторів зборів саме в контексті благодійності. Довіра зазвичай формується на основі репутації автора або каналу. Це працює для відомих волонтерів, але може бути недостатнім для нових або менш відомих ініціатив.

Соціальні мережі, зокрема Instagram та YouTube, також активно використовуються для поширення благодійних зборів. В Instagram відповідні матеріали часто публікуються у вигляді постів, сторіс або посилань у профілі, а на YouTube донати поширюються через опис відео, закріплені коментарі або звернення автора до аудиторії.

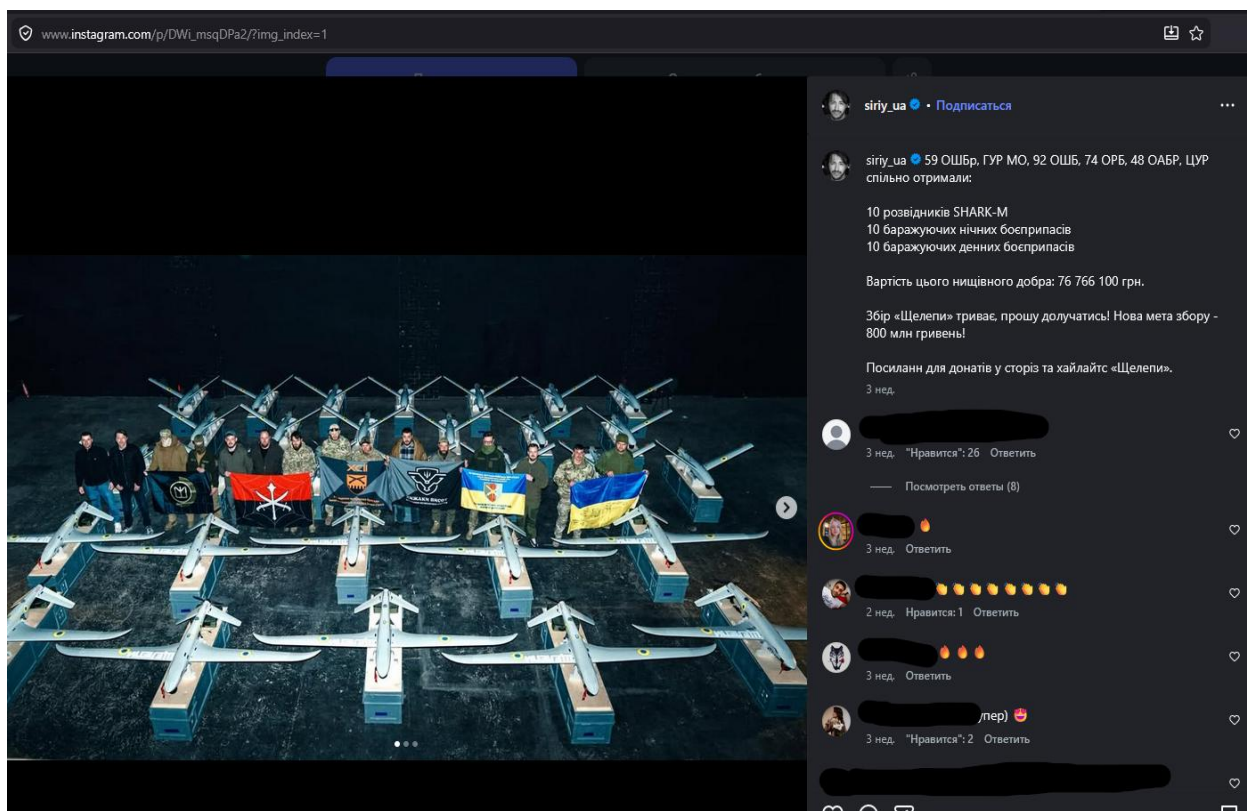


Рис. 1.12. Приклад звіту в Інстаграм

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

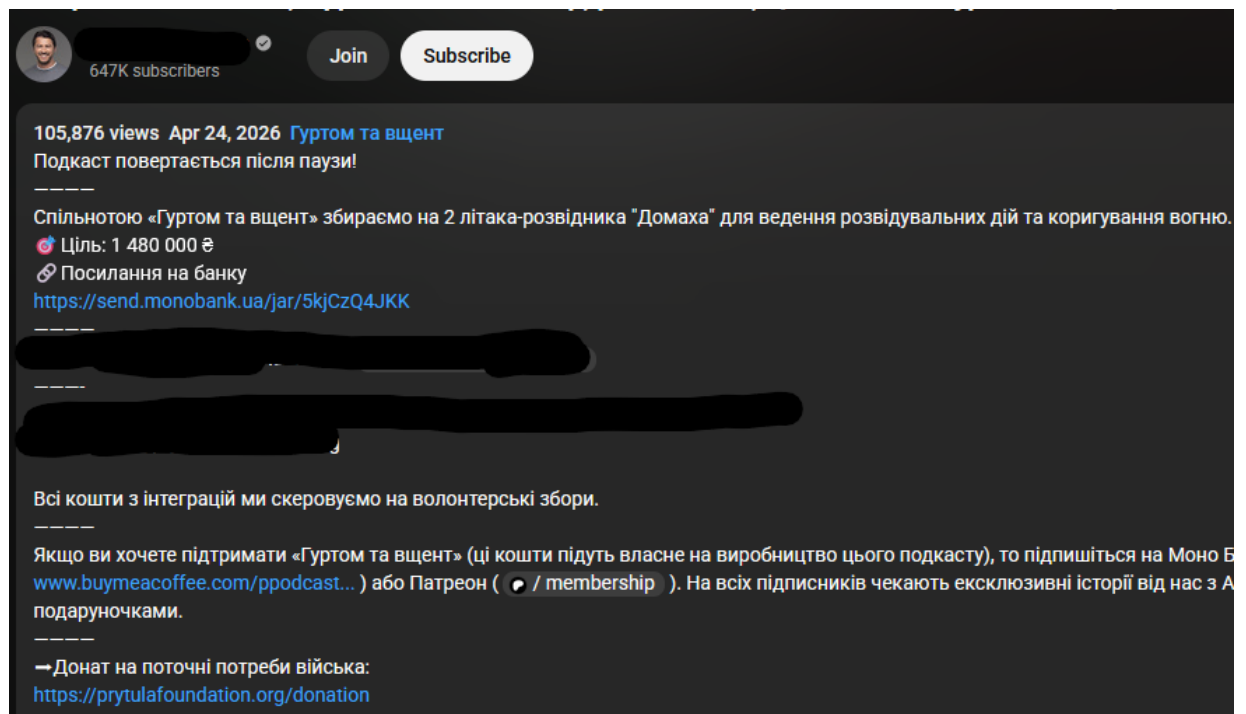


Рис. 1.13. Приклад збору під YouTube-відео

Перевагою таких платформ є велике охоплення аудиторії. Якщо автор має довіру підписників, збір може швидко отримати значну підтримку. Також фото - та відеоформат дозволяє емоційно пояснити мету збору, показати проблему та мотивувати користувачів до участі.

Недоліком є те, що Instagram та YouTube не надають повноцінної структури для ведення зборів. Вони виступають переважно як канали поширення інформації, а не як системи управління благодійними кампаніями. У таких платформах відсутні єдині статуси зборів, система ролей, верифікація волонтерів, централізована звітність та зручне відстеження історії оновлень.

Таблиця 1.1

Порівняльний аналіз існуючих рішень

Критерій	United24	HelpUA Foundation
Тип платформи	Централізована державна платформа	Платформа зборів
Наявність окремих зборів	Частково	Так

Прозорість використання коштів	Середня (узагальнені звіти)	Середня
Прив'язка донату до результату	Відсутня або обмежена	Часткова
Верифікація користувачів	Висока	Часткова
Наявність звітності по збору	Загальна	Обмежена
Управління зборами	Обмежене	Часткове
Зручність донату	Висока	Середня
Структурованість даних	Висока	Середня
Гейміфікація	Відсутня	Відсутня
Наявність окремих зборів	Частково	Так
Прозорість використання коштів	Середня (узагальнені звіти)	Середня

Проведений аналіз показує, що існуючі рішення вирішують окремі задачі, пов'язані з благодійними зборами, проте не завжди поєднують їх у межах єдиної системи. United24 забезпечує високий рівень довіри та централізований збір коштів, однак не завжди дозволяє користувачу простежити зв'язок між конкретним донатом і конкретним результатом. HelpUA Foundation є ближчою за концепцією до розроблюваного проєкту, однак потребує більш структурованого підходу до звітності та взаємодії з користувачами.

Донат-сервіси, такі як Donatello, зручні для прийому платежів, але не забезпечують повного циклу роботи зі зборами. LiqPay, у свою чергу, є корисним інструментом для обробки платежів, однак самостійно не вирішує задачі управління благодійними кампаніями.

Соціальні мережі та месенджери залишаються важливим каналом поширення зборів, але вони не мають спеціалізованої структури для перевірки організаторів, ведення звітності та контролю за станом збору.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Таким чином, актуальною є розробка веб-платформи, яка поєднує переваги існуючих підходів: зручність створення зборів, безпечну оплату, верифікацію волонтерів, структуровану звітність, систему ролей та елементи гейміфікації для підвищення залученості користувачів.

1.3 Вибір архітектури веб-платформи для організації благодійних зборів з елементами гейміфікації

Для реалізації веб-платформи благодійних зборів обрано клієнт-серверну архітектуру з розділенням фронтенд та бекенд частин. Такий підхід зумовлений необхідністю забезпечення гнучкості розробки, масштабованості системи та чіткого розмежування відповідальності між відображенням інтерфейсу та обробкою бізнес-логіки. Система проєктується як модульний застосунок, у якому функціонал поділено на окремі логічні блоки: управління користувачами, управління благодійними зборами, система ролей, модуль верифікації, обробка донатів та елементи гейміфікації.

Архітектура передбачає використання підходу Full-Stack[6] розробки з розділенням клієнтської та серверної логіки у межах єдиної системи. Клієнтська частина відповідає за відображення даних, взаємодію з користувачем та відправку запитів до серверу, тоді як серверна частина реалізує обробку запитів, бізнес-логіку, авторизацію, перевірку прав доступу та взаємодію з базою даних.[7]

Взаємодія між клієнтською та серверною частинами здійснюється через REST API, що забезпечує стандартизований обмін даними та дозволяє масштабувати систему незалежно від конкретної реалізації фронтенду або бекенду. Для спрощення взаємодії та уникнення проблем із політикою CORS використовується підхід проксування запитів через єдиний маршрут /api, який перенаправляється на серверну частину застосунку.

Система реалізована у вигляді окремих сервісів: клієнтського застосунку та серверного API. Такий підхід дозволяє розгортати та масштабувати компоненти незалежно один від одного, а також спрощує тестування та подальший розвиток системи.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

Важливим аспектом архітектури є організація системи автентифікації та авторизації. У платформі використовується механізм JWT, який передається через httpOnly cookie. Це дозволяє забезпечити захист токена від доступу через клієнтські скрипти та зменшити ризики, пов'язані з XSS-атаками. Контроль доступу до функціоналу реалізується на основі ролей користувачів, що відповідає моделі RBAC. У системі передбачено ролі користувача, волонтера та адміністратора, кожна з яких має обмежений набір дозволених дій.

Окрему роль в архітектурі відіграє модуль верифікації волонтерів. Він забезпечує процес подання заявки користувачем, її обробку адміністрацією та надання відповідного статусу. Це дозволяє обмежити доступ до створення зборів лише перевіреними користувачами та підвищити загальний рівень довіри до платформи.

Система роботи з благодійними зборами передбачає створення окремих кампаній, які мають власні атрибути: опис, цільову суму, статус, список оновлень та звітних матеріалів. Такий підхід дозволяє структурувати інформацію та забезпечити можливість відстеження повного життєвого циклу збору - від створення до завершення та публікації фінального звіту.

Інтеграція платіжної системи реалізована через використання зовнішнього сервісу WayForPay. Такий підхід дозволяє делегувати обробку фінансових операцій сторонньому сервісу та уникнути необхідності зберігання банківських даних у системі. Це підвищує рівень безпеки та спрощує відповідність вимогам щодо обробки платіжної інформації.

Архітектура також враховує можливість впровадження механізмів гейміфікації. Дані про активність користувачів, зокрема кількість донатів, їх сума, регулярність участі та взаємодія з платформою, зберігаються у системі та використовуються для нарахування досягнень. Це дозволяє реалізувати додатковий рівень взаємодії з користувачем та підвищити його мотивацію до участі у благодійних ініціативах.

Таким чином, обрана архітектура забезпечує гнучкість, безпеку та масштабованість веб-платформи, дозволяє ефективно реалізувати основні

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

функціональні можливості системи та створює основу для подальшого розвитку і розширення функціоналу.

1.4 Обґрунтування вибору технологій реалізації платформи

Реалізація веб-платформи для організації благодійних зборів з елементами гейміфікації, сформульованої у попередніх розділах, потребує використання сучасного технологічного стеку, здатного забезпечити ефективну роботу з користувачами, благодійними кампаніями, фінансовими операціями та системою взаємодії між учасниками платформи. Основною вимогою до системи є створення єдиного середовища, яке об'єднує функції створення зборів, їх адміністрування, верифікацію користувачів, обробку донатів та ведення звітності.

[2]Клієнтська частина платформи реалізується з використанням бібліотеки React у складі фреймворку Next.js. Такий вибір зумовлений необхідністю створення динамічного та адаптивного інтерфейсу користувача, який забезпечує швидку взаємодію без перезавантаження сторінок. Використання компонентного підходу дозволяє ефективно організувати структуру інтерфейсу, повторно використовувати елементи та спростити підтримку коду. Next.js[8] забезпечує додаткові можливості оптимізації, зокрема серверний рендеринг та маршрутизацію, що покращує продуктивність і загальний досвід користувача.[9; 10]

Для стилізації інтерфейсу використовується TailwindCSS[11], який дозволяє швидко створювати адаптивний дизайн без необхідності написання великої кількості окремих CSS-файлів. Такий підхід забезпечує гнучкість у розробці інтерфейсу та спрощує його масштабування.

Серверна частина системи реалізована на базі фреймворку NestJS[12], який орієнтований на побудову масштабованих серверних застосунків із використанням модульної архітектури. Це дозволяє розділити систему на логічні компоненти, зокрема модулі користувачів, благодійних зборів, ролей та адміністративного управління. NestJS забезпечує чітку структуру проєкту, що є важливим для підтримки та подальшого розвитку системи.[13]

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		27

Для збереження даних обрано реляційну систему управління базами даних PostgreSQL. Це рішення обумовлене необхідністю роботи зі складними зв'язками між сутностями, такими як користувачі, збори, ролі, донати та звітні матеріали. PostgreSQL[14] забезпечує цілісність даних, підтримку транзакцій та стабільну роботу при високому навантаженні.

Взаємодія з базою даних реалізується через ORM Prisma[15], яка виступає проміжним шаром доступу до даних. Використання Prisma дозволяє працювати з базою даних у типобезпечному середовищі, зменшує кількість помилок та спрощує процес розробки за рахунок автоматичної генерації моделей і запитів.

У якості основної мови програмування використовується TypeScript[16], що застосовується як на клієнтській, так і на серверній частині. Це дозволяє створити єдину типізовану модель даних та забезпечити узгодженість між різними частинами системи. Типізація значно знижує ймовірність виникнення помилок та підвищує надійність програмного продукту.

Для реалізації автентифікації використовується механізм JWT, який передається через httpOnly cookie. Такий підхід дозволяє підвищити безпеку системи, оскільки токен недоступний для клієнтських скриптів і не зберігається у відкритому вигляді. Це знижує ризики, пов'язані з XSS-атаками, та забезпечує надійний контроль доступу до ресурсів.

Для реалізації сповіщень у реальному часі використовується протокол WebSocket засобами бібліотеки Socket.io[17]. Це дозволяє без перезавантаження сторінки повідомляти користувача про нові досягнення, відповіді адміністратора та зміну статусу верифікаційної заявки. На відміну від підходу з регулярними HTTP-запитами (polling), WebSocket забезпечує постійне з'єднання між клієнтом і сервером, що знижує навантаження та зменшує затримку доставки повідомлень.

Обробка фінансових операцій реалізується через інтеграцію з платіжною системою WayForPay. Це дозволяє здійснювати донати без необхідності збереження банківських даних користувачів у системі. Використання

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		28

зовнішнього платіжного сервісу підвищує рівень безпеки та спрощує реалізацію фінансової логіки.

Для забезпечення стабільного середовища розробки та розгортання використовується Docker[18] . Контейнеризація дозволяє ізолювати залежності проєкту, забезпечити однакову роботу системи на різних етапах розробки та спростити процес розгортання. Зокрема, база даних PostgreSQL запускається у вигляді окремого контейнера, що забезпечує зручність керування та масштабування.[19]

Для автоматизації процесів збірки, тестування та розгортання використовуються конвеєри CI/CD на базі GitHub Actions та GitLab CI. Репозиторій розміщено на університетській платформі git.ztu.edu.ua з дзеркалом на GitHub[20] . При кожному коміті до основної гілки автоматично запускаються юніт-тести, збірка Docker-образу та перевірка стану розгорнутого застосунку. Статичний аналіз якості коду виконується засобами SonarCloud.

Таким чином, обраний технологічний стек у складі Next.js, React, NestJS, PostgreSQL, Prisma, TypeScript, TailwindCSS , WayForPay[21] та Docker є обґрунтованим і повністю відповідає вимогам до сучасної веб-платформи для організації благодійних зборів. Він дозволяє реалізувати єдину систему, що поєднує управління зборами, користувачами, фінансовими операціями та елементами гейміфікації, забезпечуючи при цьому високу продуктивність, безпеку та можливість подальшого розвитку.

Висновки до першого розділу

У першому розділі було розглянуто предметну область та сформовано основні підходи до розробки веб-платформи для організації благодійних зборів. На початковому етапі було визначено постановку задачі, у межах якої окреслено ключові вимоги до системи. Зокрема, платформа повинна забезпечувати створення та управління зборами, обробку донатів, взаємодію між користувачами, а також надання звітності щодо використання коштів.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		29

У процесі аналізу предметної області встановлено, що в умовах сучасної цифрової інфраструктури благодійні збори значною мірою здійснюються через інтернет. Це дозволяє швидко залучати кошти та охоплювати широку аудиторію, однак одночасно створює проблеми, пов'язані з довірою, прозорістю та перевіркою достовірності зборів. Окрему увагу було приділено питанням контролю використання коштів, верифікації волонтерів та зручності взаємодії користувачів із платформою.

У межах аналізу існуючих рішень, зокрема таких як United24, HelpUA Foundation, а також використання соціальних мереж (Telegram, Instagram, YouTube), було виявлено ряд обмежень. Частина платформ має централізований характер і не завжди дозволяє простежити зв'язок між конкретним донатом і результатом його використання. Інші рішення, навпаки, не забезпечують достатнього рівня контролю та структурованості, що ускладнює перевірку зборів і звітності.

Таким чином, було визначено доцільність створення веб-платформи, яка поєднує зручність здійснення донатів, зрозумілу структуру зборів, механізми верифікації користувачів та можливість ведення детальної звітності. Додатково передбачено впровадження елементів гейміфікації, що дозволяє підвищити рівень залученості користувачів та стимулювати їх до участі у благодійній діяльності.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		30

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ ДЛЯ ОРГАНІЗАЦІЇ БЛАГОДІЙНИХ ЗБОРІВ

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

На етапі проєктування вебплатформи для організації благодійних зборів було здійснено формалізацію функціональних вимог шляхом визначення варіантів використання, а також побудови об'єктно-орієнтованої структури, що відображає основні сутності предметної області та їх взаємозв'язки.

У системі передбачено три основні ролі користувачів: звичайний користувач, волонтер та адміністратор. Такий поділ забезпечує розмежування доступу до функціональних можливостей платформи відповідно до обсягу відповідальності кожного учасника.

Звичайний користувач є основним учасником платформи. Він може проходити реєстрацію та автентифікацію, зокрема через Google OAuth2, переглядати список благодійних зборів, відкривати детальну інформацію про кожен збір та здійснювати донати. Користувач має доступ до власного профілю з історією донатів, досягнень і рівня донора. Також передбачено можливість перегляду таблиці лідерів та профілів інших учасників.

Волонтер - це користувач, який пройшов процедуру верифікації адміністратором. Після отримання відповідного статусу він отримує можливість створювати власні благодійні збори, редагувати їх та вести переписку з адміністратором через вбудований чат.

Адміністратор здійснює управління системою. До його функцій належать перевірка заявок на отримання статусу волонтера, управління користувачами, перегляд усіх чатів, видача та відкликання бейджів вручну, а також видалення облікових записів із зазначенням причини.

На основі визначених ролей сформовано сценарії використання, які охоплюють реєстрацію та автентифікацію користувачів через email або Google OAuth2, перегляд і пошук благодійних зборів, здійснення донату через

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		31

WayForPay, роботу з профілем, досягненнями, карткою донора та таблицею лідерів.

Для волонтера передбачено подання заявки на верифікацію з документами, листування з адміністратором у реальному часі, а після підтвердження статусу - створення та редагування власних благодійних зборів. Адміністратор у межах своїх сценаріїв переглядає та обробляє заявки, керує користувачами, видає або відкликає бейджі, а також відповідає у чатах підтримки.

Побудова варіантів використання виконувалася з урахуванням того, що платформа повинна підтримувати повний шлях користувача від першого перегляду збору до здійснення донату, отримання досягнення та подальшої взаємодії з профілем. Тому сценарії не обмежуються тільки авторизацією або переглядом сторінок, а охоплюють послідовність дій, які формують основну цінність системи. Звичайний користувач спочатку знайомиться з доступними кампаніями, аналізує їх опис, цільову суму, актуальний прогрес і дані про організатора, після чого може перейти до здійснення внеску. Якщо користувач авторизований, система пов'язує фінансову операцію з його профілем, оновлює статистику донора та перевіряє умови для нарахування нових досягнень.

Окрему увагу під час визначення сценаріїв приділено переходу користувача між ролями. Платформа не надає можливість створювати збори будь-якому зареєстрованому користувачу одразу після реєстрації, оскільки це могло б знизити рівень довіри до сервісу. Для отримання розширених прав користувач повинен подати заявку на верифікацію, додати інформацію про себе та підтверджуючі матеріали. Після цього адміністратор перевіряє заявку, може уточнити інформацію через чат і лише після позитивного рішення надає користувачу роль волонтера. Така логіка забезпечує контроль за створенням кампаній і дозволяє зменшити ризик появи сумнівних зборів.

Для ролі волонтера основним сценарієм є створення та супровід благодійної кампанії. Волонтер формує назву збору, описує його мету,

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		32

визначає категорію, цільову суму, терміновість, додає зображення та за потреби вказує інформацію про бенефіціара. Після публікації кампанії він повинен підтримувати актуальність даних, оновлювати статус, додавати проміжні повідомлення та звітні матеріали. У межах такого сценарію важливо, щоб система зберігала зв'язок між автором, збором, донатами та звітністю, оскільки саме ця зв'язність забезпечує прозорість благодійної діяльності.

Адміністраторські сценарії пов'язані не лише з технічним керуванням користувачами, а й із підтримкою довіри до платформи. Адміністратор переглядає заявки на верифікацію, оцінює надані документи, відповідає на повідомлення, ухвалює рішення щодо надання або відхилення статусу волонтера, а також може змінювати ролі користувачів і керувати досягненнями. У разі порушення правил або некоректної поведінки адміністратор має можливість видалити обліковий запис із зазначенням причини. Таке розмежування відповідальності дозволяє підтримувати працездатність системи й одночасно контролювати якість контенту, що публікується на платформі.

Під час аналізу варіантів використання також враховано граничні ситуації, які можуть виникати під час роботи системи. Користувач може намагатися відкрити адміністративний розділ без відповідної ролі, створити збір без статусу волонтера, повторно подати заявку на верифікацію або здійснити донат із некоректними параметрами. Для таких випадків серверна частина повинна виконувати перевірку прав доступу, валідацію вхідних даних і повертати зрозуміле повідомлення про помилку. Це дозволяє уникнути неконсистентного стану даних і забезпечити передбачувану поведінку інтерфейсу.

Важливим елементом функціональної моделі є зв'язок між донатом, профілем донора та системою гейміфікації. Після успішного підтвердження платежу система не лише оновлює поточну суму збору, а й перераховує показники активності користувача. На основі цих показників визначається рівень донора, перевіряються умови отримання бейджів і формується

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		33

інформація для таблиці лідерів. Завдяки цьому гейміфікація не є ізольованим декоративним елементом, а безпосередньо інтегрується в основний бізнес-процес платформи.

Діаграма варіантів використання відображає взаємодію між акторами системи та відповідними сценаріями (див. Додаток А)

Об'єктна модель платформи складається з одинадцяти основних сутностей, які відображають користувачів, благодійні збори, фінансові операції, елементи гейміфікації, верифікацію волонтерів, чати та перегляди профілів.

Центральною сутністю є користувач (User), що містить ідентифікатор, електронну пошту, ім'я, хеш пароля, роль, посилання на аватар та спосіб автентифікації. З користувачем пов'язані всі інші сутності системи.

Сутність благодійного збору (Campaign) описує окрему кампанію зі збору коштів. Вона містить назву, короткий та повний опис, цільову та поточну суму, категорію, статус, ознаку терміновості та прив'язку до автора.

Для фіксації фінансових операцій використовується сутність донату (Donation), що зберігає суму внеску, статус транзакції, унікальний номер замовлення у платіжній системі та зв'язки з користувачем і збором.

Профіль донора (DonorProfile) зберігає накопичену інформацію про активність користувача: загальну суму донатів, кількість операцій, поточний рівень (bronze, silver, gold, platinum) та параметри кастомізації картки донора.

Система гейміфікації представлена двома сутностями: Badge описує наявні досягнення з ключем, назвою, іконкою та елементами кастомізації, що вони розблоковують. UserBadge є зв'язуючою таблицею між користувачем та отриманим досягненням.

Процес верифікації волонтерів реалізовано через дві сутності. VerificationRequest зберігає заявку користувача з описом досвіду та прикріпленими документами. VerificationMessage містить повідомлення переписки між користувачем та адміністратором у межах конкретної заявки.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Для підтримки чату між волонтерами та адміністраторами використовуються сутності Chat та ChatMessage. Кожен волонтер має один чат, до якого прив'язані всі повідомлення з відміткою відправника та статусом прочитання.

Сутність ProfileView фіксує факт перегляду одного профілю іншим користувачем і використовується для нарахування відповідних досягнень у системі гейміфікації.

Діаграма класів (рис. 2.1.) відображає зв'язки між основними сутностями системи.

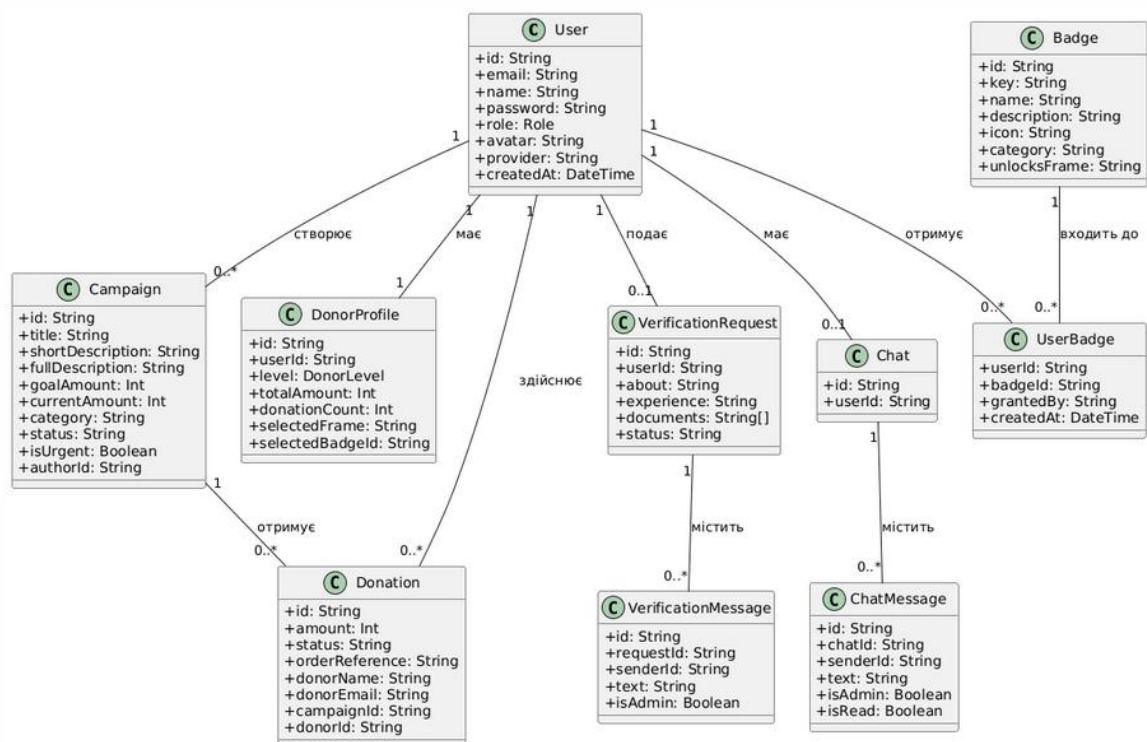


Рис. 2.1. Діаграма класів

2.2 Проектування бази даних та взаємодія з API

Для зберігання даних платформи обрано реляційну систему управління базами даних PostgreSQL. Вибір обумовлений кількома факторами. [22]

По-перше, дані платформи мають складну структуру зв'язків: користувачі пов'язані з донатами, зборами, профілями, досягненнями, верифікаційними заявками та чатами. Реляційна модель дозволяє описати ці зв'язки декларативно і забезпечити цілісність даних на рівні бази.

По-друге, PostgreSQL підтримує транзакції, що критично для операцій з донатами: зарахування суми до збору та створення запису донату повинні виконуватись як єдина атомарна операція.

По-третє, PostgreSQL надає вбудовану підтримку масивів, що використовується для зберігання списків зображень та документів у зборах без введення додаткових таблиць.

База даних містить одинадцять взаємопов'язаних таблиць, кожна з яких відповідає окремій частині бізнес-логіки платформи та використовується для збереження даних про користувачів, збори, донати, досягнення, верифікаційні заявки, повідомлення та перегляди профілів.

Таблиця User є центральною сутністю системи. Вона містить такі поля: унікальний ідентифікатор, електронну пошту (з обмеженням унікальності), ім'я, хеш пароля, роль (user, volunteer або admin), посилання на аватар, спосіб автентифікації (local або google), ідентифікатор у зовнішньому провайдері та дати створення і оновлення запису. Хеш пароля зберігається з використанням алгоритму bcrypt. Для користувачів, що авторизувались через Google OAuth2, поле пароля залишається порожнім.

Таблиця Campaign описує благодійний збір. Містить назву, короткий та повний опис, цільову та поточну суму, категорію (military, medical, humanitarian, general), статус (active, completed), масив посилань на зображення, масив шляхів до прикріплених документів, ознаку терміновості, дату закінчення терміновості та зовнішній ключ до автора збору.

Таблиця Donation фіксує кожну фінансову операцію. Зберігає суму донату, статус транзакції (pending, approved, declined, refunded), унікальний номер замовлення у системі WayForPay, ім'я та email донора (заповнюються при анонімному донаті), а також зовнішні ключі до збору і донора. Поле donorId допускає значення null, що дозволяє реалізувати анонімні донати без прив'язки до облікового запису.

Таблиця DonorProfile зберігає накопичену статистику активності користувача на платформі. Включає загальну суму всіх успішних донатів, їх

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		36

кількість, поточний рівень донора (bronze, silver, gold, platinum), а також параметри кастомізації картки донора: вибрану рамку, фон, шрифт та відображуваний бейдж. Запис створюється автоматично при першому успішному донаті користувача.

Таблиця Badge містить опис кожного з 62 досягнень системи. Зберігає унікальний текстовий ключ, назву, опис, іконку та категорію. Також містить поля, що описують елементи кастомізації, які розблоковуються при отриманні цього досягнення: рамку, фон та шрифт картки донора.

Таблиця UserBadge є зв'язуючою між User та Badge і реалізує відношення багато-до-багатьох. Містить зовнішні ключі до користувача та досягнення, поле grantedBy для фіксації адміністратора у разі ручної видачі, а також дату отримання. Комбінація userId та badgeId є унікальною, що виключає дублювання досягнень.

Таблиця VerificationRequest зберігає заявку користувача на отримання статусу волонтера. Включає поля з описом досвіду волонтерської діяльності, масив шляхів до завантажених підтверджуючих документів, посилання на соціальні мережі та поточний статус заявки (pending, approved, rejected). Кожен користувач може мати лише одну заявку, що забезпечується обмеженням унікальності на поле userId.

Таблиця VerificationMessage зберігає повідомлення переписки між користувачем та адміністратором у межах верифікаційної заявки. Містить зовнішній ключ до заявки, ідентифікатор відправника, текст повідомлення та ознаку того, чи є відправник адміністратором.

Таблиця Chat реалізує чат між волонтером та адміністратором. Кожен волонтер має рівно один чат, що забезпечується відношенням один-до-одного з таблицею User.

Таблиця ChatMessage зберігає окремі повідомлення чату. Містить зовнішній ключ до чату, ідентифікатор відправника, текст, ознаку адміністраторського повідомлення та ознаку прочитання. Ознака прочитання використовується для формування лічильника непрочитаних повідомлень.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Таблиця ProfileView фіксує факт перегляду профілю одного користувача іншим. Використовується виключно у системі гейміфікації для нарахування досягнень, пов'язаних із соціальною активністю.

Між таблицями визначено зв'язки, які забезпечують цілісність даних і відображають логіку роботи платформи. Користувач пов'язаний зі зборами відношенням один-до-багатьох, оскільки один користувач може бути автором кількох кампаній. Аналогічно, один збір може отримати багато донатів, а один користувач може здійснити багато фінансових внесків. Зв'язки User - DonorProfile, User - Chat та User - VerificationRequest реалізовано як один-до-одного, а зв'язок між користувачами й досягненнями реалізовано через проміжну таблицю UserBadge як відношення багато-до-багатьох.

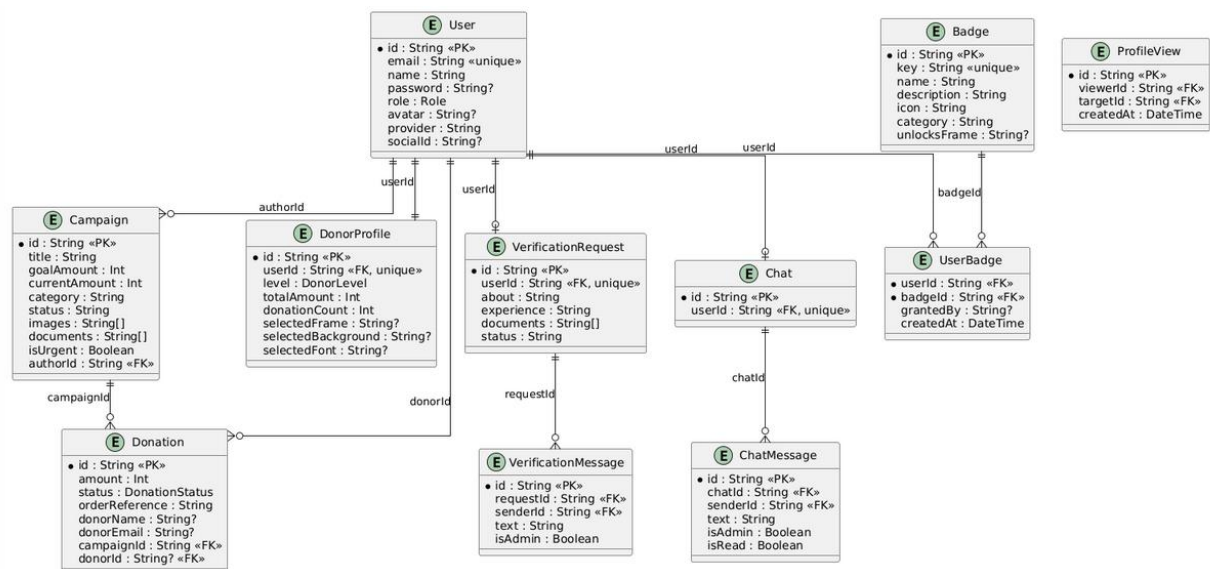


Рис. 2.2. ER-діаграма бази даних

Для взаємодії серверної частини з PostgreSQL використовується ORM Prisma[5]. Схема бази даних описується декларативно у файлі schema.prisma, на основі якого Prisma генерує типізований клієнт для TypeScript. Це дозволяє виконувати запити до бази даних без написання SQL-коду вручну та отримувати повну підтримку типів на етапі компіляції.

Порівняно з TypeORM, який також є поширеним вибором для NestJS-проектів, Prisma має суттєву перевагу у вигляді автоматично згенерованого типізованого клієнта. TypeORM використовує декоратори безпосередньо в entity-класах, що призводить до змішування опису схеми та бізнес-логіки в

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		38

одному файлі. Prisma натомість зберігає схему окремо від коду застосунку, що спрощує її читання та супровід. Перелік усіх сутностей системи з їх призначенням та основними полями наведено у таблиці 2.1.

Таблиця 2.1

Сутності бази даних

Сутність	Призначення	Основні поля
User	Обліковий запис користувача	id, email, name, password, role, avatar, provider
Campaign	Благодійний збір	id, title, goalAmount, currentAmount, category, status, authorId
Donation	Фінансова операція	id, amount, status, orderReference, campaignId, donorId
DonorProfile	Профіль та статистика донора	userId, level, totalAmount, donationCount, selectedFrame
Badge	Досягнення системи	id, key, name, icon, category, unlocksFrame, unlocksBackground
UserBadge	Отримані досягнення	userId, badgeId, grantedBy, createdAt
VerificationRequest	Заявка на волонтера	id, userId, about, experience, documents, status
VerificationMessage	Повідомлення верифікації	id, requestId, senderId, text, isAdmin
Chat	Чат волонтера	id, userId
ChatMessage	Повідомлення чату	id, chatId, senderId, text, isAdmin, isRead
ProfileView	Перегляд профілю	id, viewerId, targetId, createdAt

Для управління змінами схеми бази даних використовуються міграції Prisma. Кожна зміна моделі фіксується командою `prisma migrate dev`, яка створює SQL-файл міграції та застосовує його до бази. Усього в проєкті

сформовано 14 міграцій, які послідовно створюють таблиці користувачів, зборів, донатів, досягнень, профілів донорів, верифікаційних заявок, чатів та допоміжних сутностей. У виробничому середовищі запускається команда `prisma migrate deploy`, яка застосовує всі незастосовані міграції. Це забезпечує відтворюваність стану бази даних на будь-якому середовищі та дозволяє відстежувати історію змін схеми разом з кодом застосунку у системі контролю версій.

2.3 Проєктування та реалізація алгоритмів роботи системи

Функціонування платформи забезпечується сукупністю алгоритмів, що реалізують основні бізнес-процеси. У межах цього підрозділу розглянуто алгоритми автентифікації, обробки донатів, верифікації волонтерів, гейміфікації та доставки сповіщень у реальному часі.

Система підтримує два способи автентифікації: через email та пароль і через Google OAuth2. Обидва варіанти входу приводять до створення безпечної сесії користувача з використанням JWT-токена.[23; 24; 25]

При стандартній автентифікації користувач надсилає email та пароль на ендпоінт `POST /auth/login`. Сервер шукає користувача за email у базі даних. Якщо запис не знайдено або хеш пароля не збігається з наданим значенням, повертається помилка 401. При успішній перевірці сервер генерує JWT-токен, що містить у корисному навантаженні ідентифікатор користувача, email та роль. Токен передається клієнту через `httpOnly cookie` з параметром `sameSite: lax`, що унеможлиблює його зчитування клієнтськими скриптами та знижує ризик XSS-атак.

При авторизації через Google OAuth2 користувач перенаправляється на сторінку авторизації Google. Після підтвердження Google повертає профіль на `callback-ендпоінт` сервера. Система перевіряє наявність облікового запису за Google ID або email. Якщо обліковий запис не знайдено, створюється новий з роллю `user`. Після цього формується JWT-токен і передається клієнту через параметри URL перенаправлення, де клієнтська частина зберігає його у `cookie`

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		40

через ендпоінт POST /auth/set-session. Діаграма активності процесу автентифікації наведена на рис. 2.3.

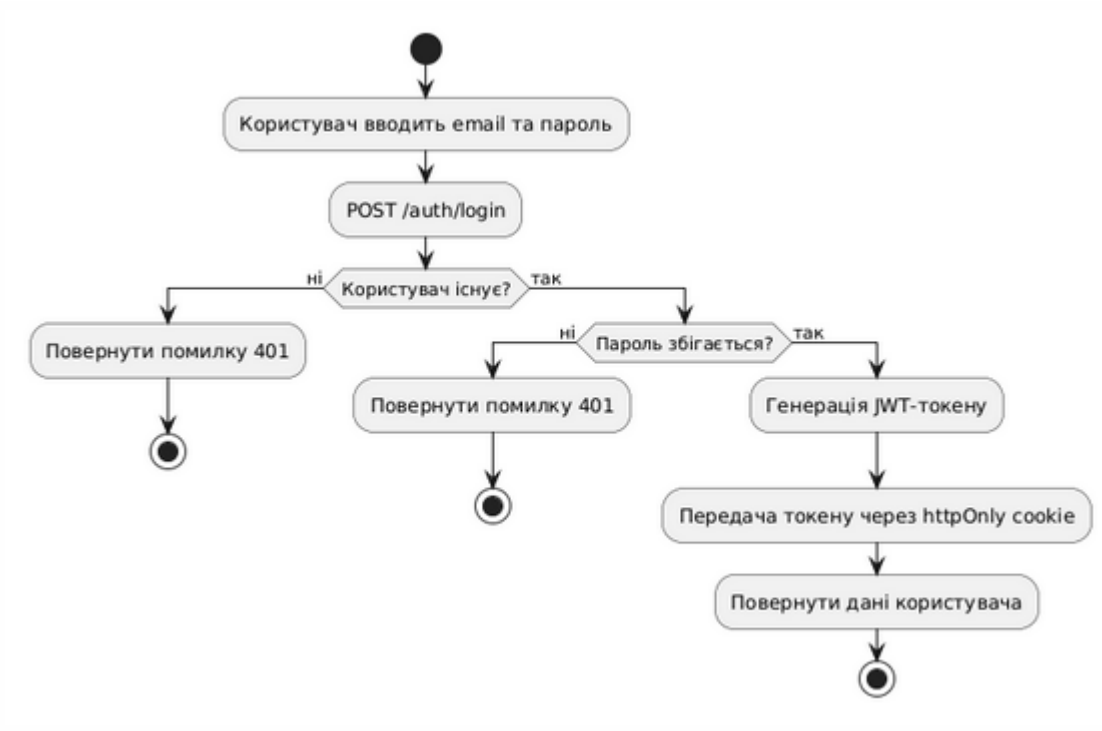


Рис. 2.3. Діаграма активності процесу автентифікації

Процес обробки донату складається з двох основних етапів: ініціювання платежу та обробки результату після відповіді платіжної системи.

На першому етапі клієнт надсилає запит з сумою та ідентифікатором збору. Сервер перевіряє існування збору та його активний статус. Далі генерується унікальний номер замовлення у форматі NEXUSAID_{timestamp}_{random}. Сервер формує підпис запиту за алгоритмом HMAC-MD5 з параметрів транзакції та секретного ключа WayForPay. Підписаний запит передається клієнту, який ініціює платіжну форму WayForPay.

На другому етапі WayForPay надсилає webhook-повідомлення на ендпоінт POST /donations/webhook. Сервер перевіряє підпис отриманого запиту для підтвердження його автентичності. При успішній перевірці оновлюється статус донату на approved, збільшується поточна сума збору та запускається алгоритм гейміфікації для відповідного користувача. Діаграма активності процесу донату наведена на рис. 2.4.

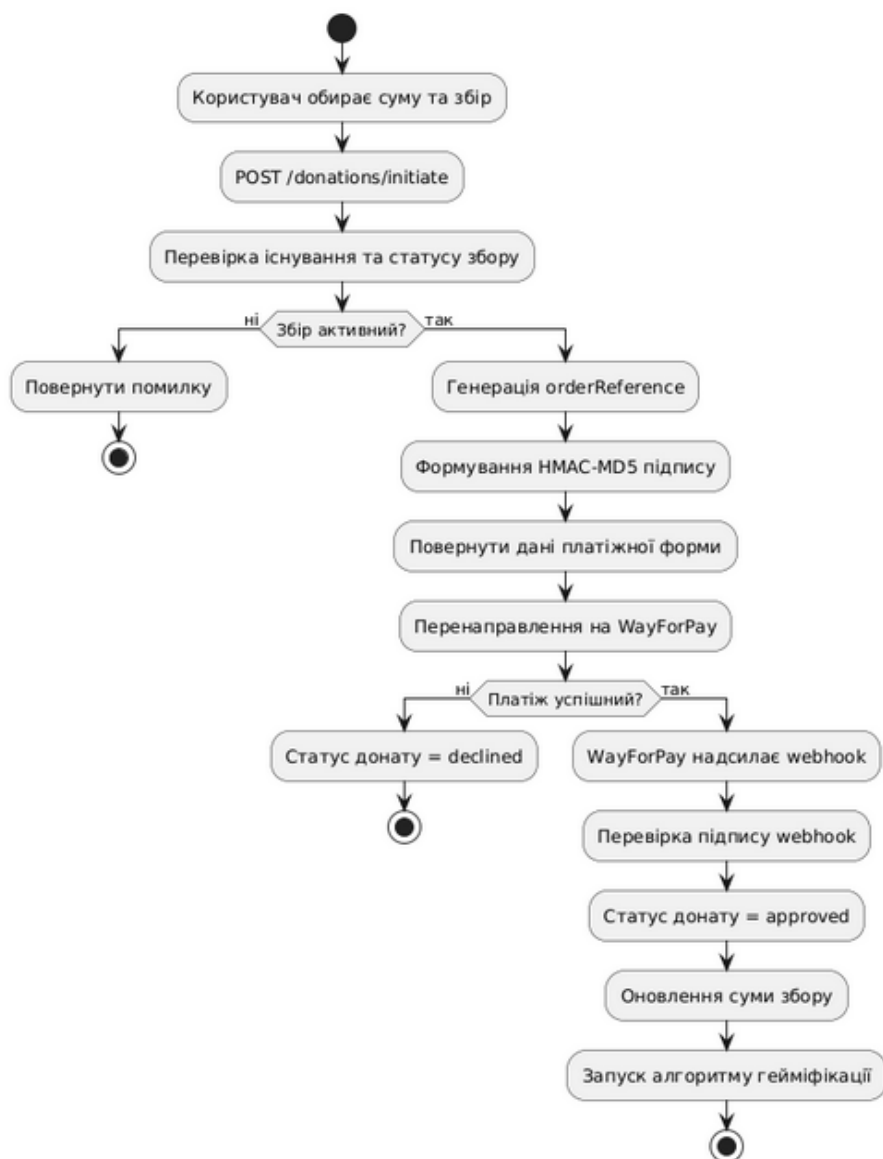


Рис. 2.4. Діаграма активності процесу донату

Під час верифікації волонтера користувач заповнює форму заявки з описом досвіду та прикріплює підтверджуючі документи. Сервер перевіряє, чи немає вже активної або схваленої заявки від цього користувача. При відсутності попередньої заявки створюється запис `VerificationRequest` зі статусом `pending`, після чого адміністратор отримує сповіщення про нову заявку.

Адміністратор переглядає заявку в панелі керування, може надіслати повідомлення користувачу через вбудований чат верифікації та приймає рішення. При схваленні роль користувача змінюється з `user` на `volunteer`, а

йому надходить сповіщення через WebSocket. При відхиленні статус заявки оновлюється і користувач також отримує сповіщення.

Діаграма активності процесу верифікації наведена на рис. 2.5.

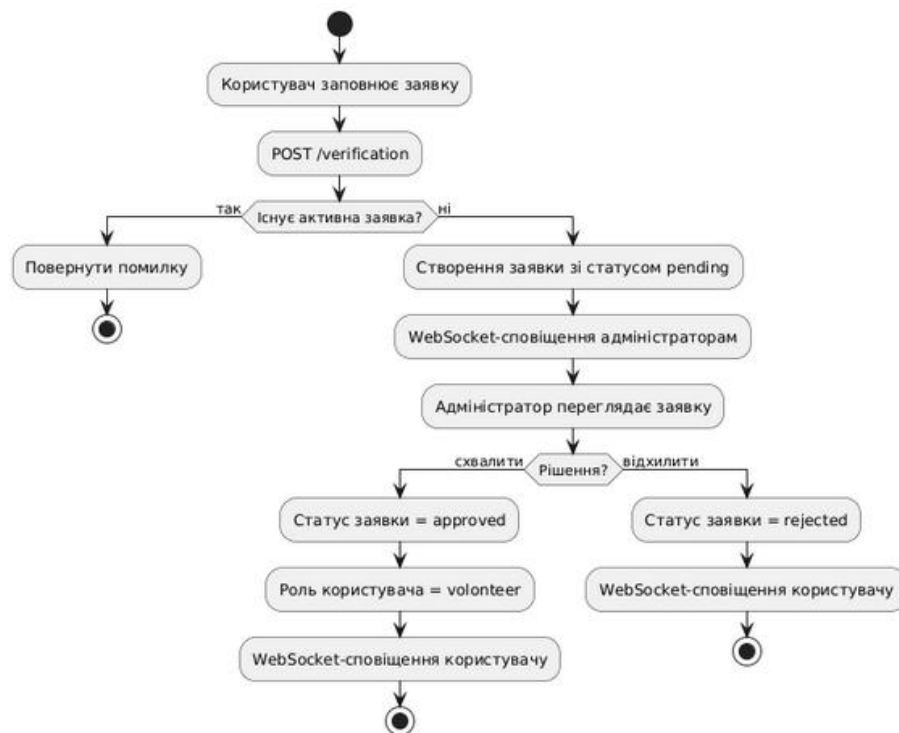


Рис. 2.5. Діаграма активності процесу верифікації

Алгоритм гейміфікації запускається після кожної успішної дії користувача на платформі. Система аналізує статистику конкретного користувача та порівнює її з умовами отримання досягнень.

Умови поділяються на кілька груп. Перша група пов'язана із донатами: перший донат, певна кількість донатів (5, 10, 25, 50), досягнення порогових значень загальної суми (1000, 5000, 10000, 100000 грн), регулярність (7 та 30 послідовних днів). Друга група враховує категорії зборів: донати у військову, медичну, гуманітарну або загальну категорії. Третя група пов'язана з активністю на платформі: заповнення профілю, кастомізація картки донора, перегляд профілів інших користувачів, активність у чаті.

Для кожного досягнення, умови якого виконані, система перевіряє, чи не отримувач вже користувач це досягнення раніше, і при відсутності запису створює новий у таблиці UserBadge. Після нарахування нового досягнення клієнт отримує WebSocket-сповіщення.

Окремо виконується перерахунок рівня донора: bronze (від 0 грн), silver (від 5000 грн), gold (від 25000 грн), platinum (від 100000 грн). Якщо загальна сума донатів досягає наступного порогу, рівень у таблиці DonorProfile оновлюється.

Діаграма активності алгоритму гейміфікації наведена на рис. 2.6.

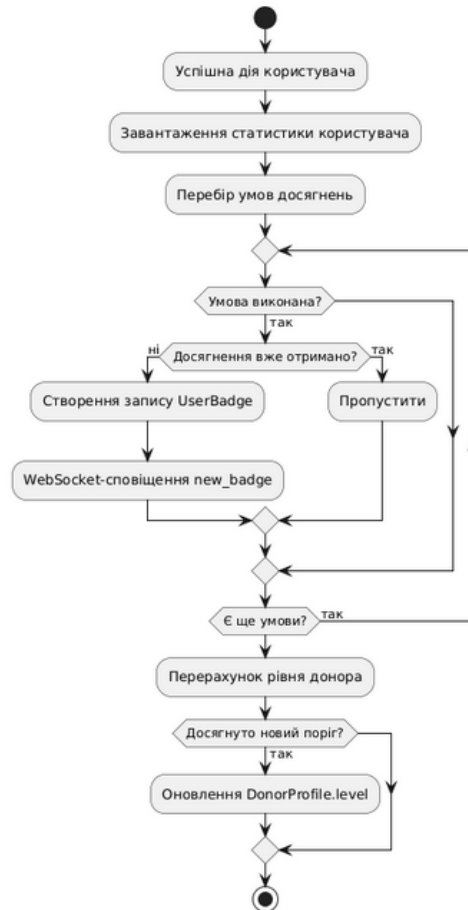


Рис. 2.6. Діаграма активності алгоритму гейміфікації

Для доставки сповіщень у реальному часі використовується протокол WebSocket через бібліотеку Socket.io. При завантаженні застосунку авторизований користувач отримує короткостроковий WebSocket-токен через GET /auth/ws-token і використовує його для встановлення з'єднання з сервером. Після перевірки токена сервер додає клієнта до особистої кімнати користувача user: {id}, а користувачів з роллю адміністратора також підключає до кімнати admin_room.

Набір WebSocket-подій охоплює основні сценарії взаємодії користувачів із платформою. Подія new_badge використовується для повідомлення про

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

отримання нового досягнення, new_message передає нові повідомлення в чаті підтримки, verification_message доставляє повідомлення в межах верифікаційної заявки, verification_approved інформує про схвалення заявки та оновлює роль користувача без перезавантаження сторінки, а new_verification_request надсилається адміністраторам при появі нової заявки на верифікацію.

При розриві з'єднання клієнт автоматично повторює спроби підключення з наростаючим інтервалом затримки. Діаграма активності процесу доставки сповіщень наведена на рис. 2.7.

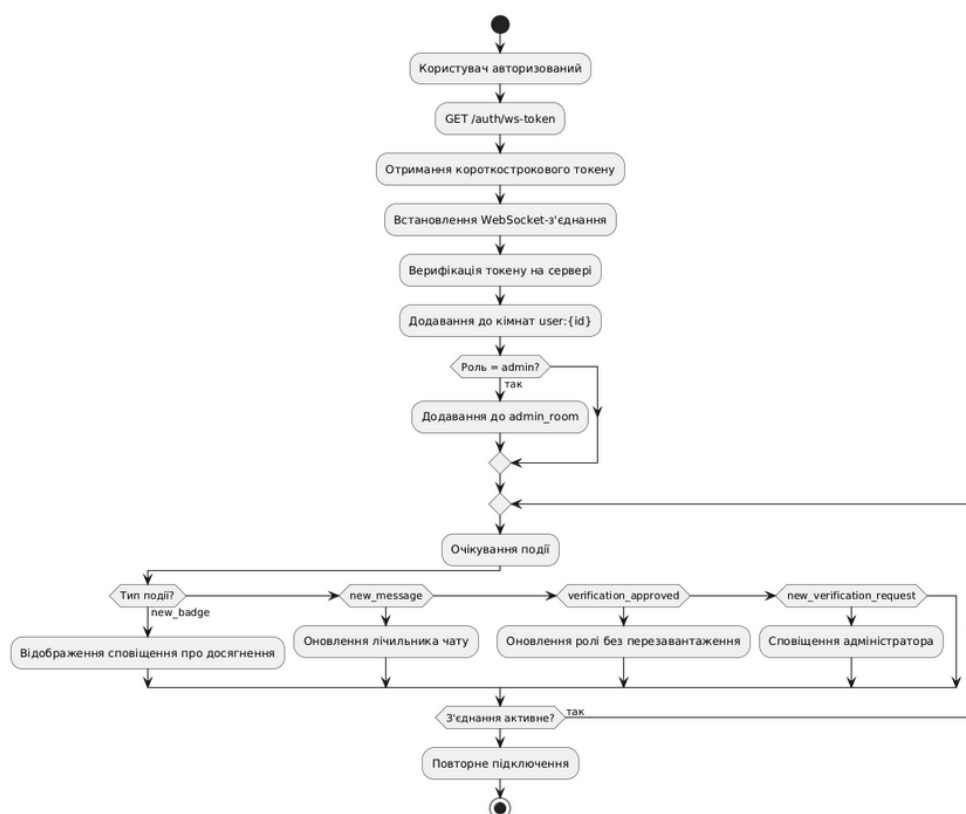


Рис. 2.7. Діаграма активності доставки сповіщень

2.4 Реалізація функціональних можливостей вебплатформи

У межах реалізації вебплатформи впроваджено повний набір функціональних модулів, що охоплюють усі ролі користувачів та основні бізнес-процеси системи.

Головна сторінка є точкою входу в платформу та формує перше враження користувача. Вона складається з кількох блоків. Блок статистики відображає лічильники кількості активних зборів, загальної суми залучених

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		45

коштів та кількості учасників. Значення завантажуються з ендпоінтів `/api/campaigns` та `/api/gamification/leaderboard` і оновлюються при кожному відкритті сторінки.

Нижче розташований блок термінових зборів, де для кожної з чотирьох категорій виводиться один найбільш терміновий активний збір з таймером зворотного відліку до дати закінчення. Дані надходять з ендпоінту `/api/campaigns/urgent`. При відсутності термінових зборів у певній категорії відповідна картка не відображається.

Таблиця лідерів на головній сторінці відображає топ донорів платформи і формується на основі даних ендпоінту `/api/gamification/leaderboard`.

Сторінка «Про нас» описує концепцію платформи та спосіб роботи з нею. Вона містить секцію з анімованими лічильниками статистики, що активуються при прокрутці до відповідного блоку за допомогою Intersection Observer API. Також представлено секцію «Як це працює» з трьома кроками: створення збору, здійснення донату та отримання досягнень. Окремо виведено цінності платформи - прозорість, гейміфікацію та спільноту - і перелік використаних технологій.

У системі реалізовано реєстрацію через email і пароль та через Google OAuth2. При реєстрації виконується перевірка унікальності email та хешування пароля алгоритмом bcrypt. JWT-токен передається через httpOnly cookie. Для розмежування доступу кожен захищений ендпоінт перевіряє токен через JwtAuthGuard, а ендпоінти, що вимагають ролі адміністратора або волонтера, додатково перевіряють поле role у корисному навантаженні токена.

Функціонал управління благодійними зборами охоплює перегляд списку зборів з фільтрацією за категоріями та текстовим пошуком через ендпоінт `GET /campaigns`. Кожен збір має детальну сторінку з описом, прогрес-баром зібраної суми та переліком донатів. Для термінових зборів відображається таймер зворотного відліку.

Верифіковані волонтери мають доступ до форми створення збору з можливістю завантаження документів. Завантажені файли зберігаються на

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		46

сервері у папці uploads і доступні за публічним URL. Редагування збору реалізовано через ендпоінт PATCH /campaigns/:id. Право на редагування перевіряється на рівні серверної логіки: дозволено автору збору та адміністратору.

Для здійснення донату клієнт надсилає запит на ендпоінт POST /donations/initiate із сумою та ідентифікатором збору. Сервер генерує унікальний номер замовлення та формує підписаний запит до WayForPay. Після підтвердження оплати WayForPay надсилає webhook, сервер верифікує підпис та оновлює статус донату і суму збору. Передбачено підтримку як авторизованих, так і анонімних донатів.

Верифікація волонтерів починається з подання користувачем заявки через форму з полями опису досвіду та завантаженням документів. Сервер перевіряє відсутність активної заявки та створює запис зі статусом pending. Адміністратор отримує WebSocket-сповіщення та може переглянути заявку і вести переписку з користувачем. Після прийняття рішення система автоматично змінює роль користувача.

У межах гейміфікації реалізовано 62 досягнення, розподілені за категоріями. Алгоритм гейміфікації запускається після кожного успішного донату та перевіряє виконання умов усіх досягнень. Реалізовано чотири рівні донора з пороговими значеннями 0, 5000, 25000 та 100000 гривень. Кожне досягнення розблоковує елементи кастомізації картки донора, а таблиця лідерів кешується на сервері з оновленням кожні п'ять хвилин для зниження навантаження на базу даних.

Після авторизації клієнт отримує WebSocket-токен та встановлює з'єднання з сервером. Реалізовано п'ять типів WebSocket-подій: new_badge, new_message, verification_message, verification_approved та new_verification_request. Лічильники непрочитаних сповіщень відображаються в навігаційному заголовку і оновлюються без перезавантаження сторінки.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		47

Адміністратор має доступ до окремого розділу з управління користувачами, де реалізовано пошук, фільтрацію за роллю, зміну ролей, видачу та відкликання досягнень і видалення облікових записів із зазначенням причини. Окремий розділ відображає список усіх чатів з волонтерами з індикатором непрочитаних повідомлень. Перелік основних ендпоінтів REST API платформи наведено у таблиці 2.2.

Таблиця 2.2

Основні ендпоінти REST API[26]

Модуль	Метод	Ендпоінт	Призначення
Auth	POST	/auth/register	Реєстрація
Auth	POST	/auth/login	Вхід в акаунт
Auth	GET	/auth/google	Авторизація через Google
Auth	GET	/auth/me	Поточний користувач
Auth	GET	/auth/ws-token	Токен для WebSocket
Campaigns	GET	/campaigns	Список зборів
Campaigns	POST	/campaigns	Створення збору
Campaigns	PATCH	/campaigns/:id	Редагування збору
Campaigns	DELETE	/campaigns/:id	Видалення збору
Campaigns	GET	/campaigns/urgent	Термінові збори
Donations	POST	/donations/initiate	Ініціювання донату
Donations	POST	/donations/webhook	Webhook WayForPay
Gamification	GET	/gamification/badges	Список досягнень
Gamification	GET	/gamification/leaderboard	Таблиця лідерів
Gamification	GET	/gamification/me	Профіль гейміфікації
Users	GET	/users/me	Профіль користувача
Users	GET	/users	Список користувачів (admin)
Users	DELETE	/users/:id	Видалення користувача

Verification	POST	/verification	Подача заявки
Verification	GET	/verification/me	Моя заявка
Verification	PATCH	/verification/:id/status	Зміна статусу заявки
Chat	GET	/chat/me	Чат волонтера
Chat	POST	/chat/me/message	Надіслати повідомлення

Висновки до другого розділу

У другому розділі виконано повне проєктування та реалізацію вебплатформи для організації благодійних зборів з елементами гейміфікації.

У підрозділі 2.1 визначено три ролі користувачів системи - користувач, волонтер та адміністратор - і сформовано перелік з 14 варіантів використання. Побудовано діаграму варіантів використання та діаграму класів, що відображає зв'язки між основними сутностями предметної області.

У підрозділі 2.2 спроектовано схему бази даних з 11 взаємопов'язаних таблиць. Визначено відношення між сутностями: один-до-одного між User і DonorProfile, User і Chat, User і VerificationRequest. Відношення один-до-багатьох між User і Campaign, Campaign і Donation, Chat і ChatMessage, VerificationRequest і VerificationMessage. Відношення багато-до-багатьох між User і Badge через проміжну таблицю UserBadge. Для роботи з базою даних обрано PostgreSQL та ORM Prisma. Управління змінами схеми реалізовано через систему міграцій, що містить 14 послідовних файлів.

У підрозділі 2.3 описано та формалізовано п'ять ключових алгоритмів роботи системи: автентифікації з підтримкою JWT та Google OAuth2, обробки донатів через WayForPay з HMAC-MD5 підписом, верифікації волонтерів, гейміфікації з 62 досягненнями та чотирма рівнями донора, а також доставки сповіщень у реальному часі через WebSocket. Для кожного алгоритму побудовано діаграму активності.

У підрозділі 2.4 описано реалізацію всіх функціональних модулів платформи: автентифікації, управління зборами, обробки донатів, верифікації волонтерів, системи гейміфікації, сповіщень у реальному часі та

адміністративної панелі. Взаємодія між клієнтською та серверною частинами здійснюється через REST API, що містить 25 ендпоінтів, описаних у таблиці 2.2.

Результатом другого розділу є повністю спроектована та реалізована система, готова до розгортання у виробничому середовищі.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ДЛЯ ОРГАНІЗАЦІЇ БЛАГОДІЙНИХ ЗБОРІВ

3.1 Порядок встановлення та налаштування параметрів системи

Для управління вихідним кодом використовується система контролю версій Git. Основний репозиторій розміщено на університетській платформі git.ztu.edu.ua, додатково створено дзеркало на GitHub для інтеграції з хмарними сервісами CI/CD[27] та аналізу якості коду.

У проєкті застосовується гілкова стратегія з двома основними гілками. Гілка main містить стабільний код, що відповідає стану виробничого середовища. Кожен коміт до цієї гілки автоматично запускає повний конвеєр CI/CD та ініціює розгортання на Railway[28]. Гілка develop використовується для поточної розробки. Під час реалізації окремих функціональних модулів створювались тимчасові feature-гілки, які після завершення роботи зливались до develop, а потім до main.

На університетській платформі git.ztu.edu.ua налаштовано конвеєр GitLab[29] CI, що запускається при кожному коміті до основної гілки. Конвеєр складається з двох етапів. На етапі test виконується встановлення залежностей, генерація Prisma-клієнта та запуск юніт-тестів з формуванням звіту про покриття коду. На етапі build виконується компіляція TypeScript-коду серверної частини. Конфігурація конвеєра описана у файлі .gitlab-ci.yml у корені репозиторію.

На GitHub налаштовано три workflow. Перший, CI/CD, є основним конвеєром і запускається при кожному коміті до гілок main та develop. Він складається з трьох послідовних задач. Задача Tests виконує встановлення залежностей, генерацію Prisma-клієнта, запуск 125 юніт-тестів з формуванням coverage-звіту та статичний аналіз коду через SonarCloud. Задача Docker Build збирає production Docker-образ серверної частини з перевіркою коректності багатоетапного Dockerfile. Задача Verify Deploy виконується тільки при коміті до main та перевіряє стан розгорнутого застосунку через HTTP-запит до ендпоінту /stats.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Другий workflow, Mirror, виконує автоматичну синхронізацію коду з GitHub на git.ztu.edu.ua при кожному коміті до main. Це забезпечує актуальність університетського репозиторію без ручних дій. Усі конфігурації конвеєрів зберігаються у директорії .github/workflows у вигляді YAML-файлів.

Для автоматизованого аналізу якості коду використовується сервіс SonarCloud. Аналіз запускається автоматично в межах задачі Tests конвеєра GitHub Actions після кожного коміту до main. SonarCloud перевіряє код на наявність потенційних помилок, вразливостей безпеки та відхилень від стандартів якості. За результатами аналізу проєкт отримав статус Quality Gate Passed та рейтинг безпеки A. Звіт доступний на sonarcloud.io.[30]

Для забезпечення відтворюваності середовища виконання серверна частина контейнеризована засобами Docker. Dockerfile реалізовано за багатоетапним підходом і містить три етапи. Перший етап builder встановлює всі залежності та компілює TypeScript-код. Другий етап production копіює скомпільований код та встановлює тільки виробничі залежності, що зменшує розмір фінального образу. Третій етап local призначений для локального запуску через docker-compose і включає запуск міграцій бази даних.

Для локального розгортання повного стеку реалізовано конфігурацію docker-compose.yml, що описує три сервіси: база даних PostgreSQL, серверна частина та клієнтська частина. Запуск усього стеку виконується однією командою docker-compose up --build.

Застосунок розгорнуто у хмарному середовищі Railway. Серверна та клієнтська частини розгорнуті як окремі сервіси, база даних PostgreSQL надається як керований сервіс Railway. Для кожного сервісу налаштовано змінні середовища, що містять параметри підключення до бази даних, секрети JWT, ключі WayForPay, Google OAuth2 та налаштування CORS.

У результаті порядок встановлення й налаштування параметрів системи охоплює підготовку середовища, конфігурацію змінних, створення структури бази даних, запуск клієнтської та серверної частин, налаштування інтеграцій, перевірку безпеки та контроль працездатності після розгортання. Такий підхід

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		52

забезпечує відтворюваність розгортання і дозволяє переносити систему між локальним, тестовим та виробничим середовищами без зміни її програмної логіки.

Оновлення системи виконується через внесення змін до гілки розробки, перевірку працездатності та подальше злиття зі стабільною гілкою main. Після цього автоматично запускається конвеєр, який встановлює залежності, виконує тести, аналізує якість коду та ініціює оновлення розгорнутого середовища. Така схема зменшує ризик випадкового потрапляння неперевірених змін у виробничу версію платформи та дозволяє підтримувати стабільний стан застосунку.

Після розгортання важливо перевірити не лише факт запуску, а й повний цикл роботи системи. Перевіряється авторизація, відкриття сторінок, отримання даних з API, створення тестового збору, подання заявки на верифікацію, робота чату та коректність сповіщень. Для виявлення помилок використовуються журнали Railway, результати виконання CI/CD і звіти SonarCloud. Якщо під час запуску виникають помилки, першочергово перевіряються змінні середовища, доступність бази даних і відповідність версій залежностей.

Процес розгортання у Railway виконується окремо для клієнтської та серверної частин. Для кожного сервісу задаються власні змінні середовища, команда збірки та команда запуску. База даних PostgreSQL створюється як керований сервіс, після чого її connection string передається backend-застосунку. Після першого розгортання необхідно застосувати міграції Prisma, щоб структура виробничої бази даних відповідала структурі, яка використовується в коді застосунку.

Налаштування інтеграції з Google OAuth2 передбачає створення клієнта в консолі Google та вказання дозволених redirect URI. Ці адреси повинні відповідати доменам, на яких працює застосунок. Якщо адреса в конфігурації Google не збігається з реальною адресою сервера, авторизація буде завершуватися помилкою. Аналогічно інтеграція з WayForPay потребує

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

правильного налаштування merchant-параметрів, секретного ключа та адреси callback, на яку платіжна система надсилає результат обробки транзакції.

Особливе значення має налаштування параметрів безпеки. Оскільки автентифікація використовує httpOnly cookie, сервер повинен коректно налаштовувати політики CORS, домени, режим secure-cookie та SameSite. У локальному середовищі частина цих параметрів може бути менш суворою, однак у виробничому середовищі вони повинні відповідати публічному домену клієнтської частини. Неправильне налаштування цих параметрів може призвести до того, що користувач зможе пройти вхід, але сесія не буде зберігатися між запитами.

Після локального запуску обов'язково перевіряється доступність основних сервісів. Для серверної частини перевіряється відповідь API, коректність підключення до бази даних і можливість виконання запитів автентифікації. Для клієнтської частини перевіряється відкриття головної сторінки, завантаження списку зборів і відсутність помилок у консолі браузера. Така первинна перевірка дозволяє переконатися, що конфігурація виконана правильно, а компоненти системи взаємодіють між собою без порушень.

Для спрощення запуску всього програмного комплексу використовується Docker Compose. Конфігурація docker-compose.yml описує взаємодію трьох основних сервісів: бази даних PostgreSQL, серверної частини та клієнтського застосунку. Перевага такого підходу полягає в тому, що розробнику не потрібно окремо встановлювати та налаштовувати всі компоненти системи вручну. Після запуску контейнерів база даних, backend і frontend працюють в одному середовищі та можуть взаємодіяти між собою через визначені мережеві адреси.

Клієнтська частина налаштовується окремо, оскільки вона має власний набір залежностей та параметрів збірки. У конфігурації клієнта вказується адреса серверного API, що використовується для отримання даних про збори, користувачів, профілі, досягнення та повідомлення. Під час локальної

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

розробки ця адреса зазвичай вказує на сервер, запущений на локальному порту, а у виробничому середовищі — на публічну адресу backend-сервісу. Після встановлення залежностей клієнтська частина може запускатися в режимі розробки або збиратися для виробничого використання.

Після налаштування змінних середовища виконується підготовка бази даних. Для роботи з PostgreSQL використовується Prisma, тому перед запуском сервера необхідно згенерувати клієнт Prisma та застосувати міграції. Міграції створюють таблиці користувачів, зборів, донатів, досягнень, заявок на верифікацію, повідомлень і допоміжних сутностей. Такий підхід забезпечує відтворюваність структури бази даних, оскільки однаковий набір міграцій може бути застосований як у локальному середовищі, так і в хмарному середовищі Railway.

Підготовка серверної частини починається з отримання вихідного коду з репозиторію та встановлення залежностей. Після цього створюється файл змінних середовища, у якому задаються параметри підключення до PostgreSQL, секрет JWT, адреси клієнтської частини, параметри Google OAuth2, ключі WayForPay і службові налаштування WebSocket. Ці дані не повинні зберігатися у відкритому репозиторії, тому в проєкті використовується приклад конфігураційного файлу без реальних секретів, а робочі значення додаються безпосередньо в середовищі запуску.

Перед встановленням системи необхідно підготувати середовище виконання, яке відповідає технологічному стеку застосунку. Для серверної частини потрібні Node.js, менеджер пакетів npm, доступ до бази даних PostgreSQL та набір змінних середовища, що визначають параметри підключення, секрети автентифікації і дані зовнішніх сервісів. Для клієнтської частини необхідне середовище Node.js та можливість виконання збірки Next.js. Якщо застосунок розгортається локально без Docker, база даних створюється окремо, після чого в конфігурації вказується рядок підключення до неї.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

3.2 Структура та опис інтерфейсу вебплатформи

Клієнтська частина платформи реалізована у вигляді односторінкового застосунку на основі Next.js 16 з використанням App Router. Навігація між сторінками здійснюється без повного перезавантаження сторінки. Інтерфейс адаптований для роботи на різних типах пристроїв завдяки використанню Tailwind CSS.

Головна сторінка є точкою входу на платформу. У верхній частині розміщено блок з лічильниками статистики, що відображають кількість активних зборів та учасників платформи. Нижче розташовано блок термінових зборів, де для кожної категорії виводиться один найбільш терміновий збір з таймером зворотного відліку. Завершує сторінку блок таблиці лідерів з топом донорів платформи (рис. 3.1).

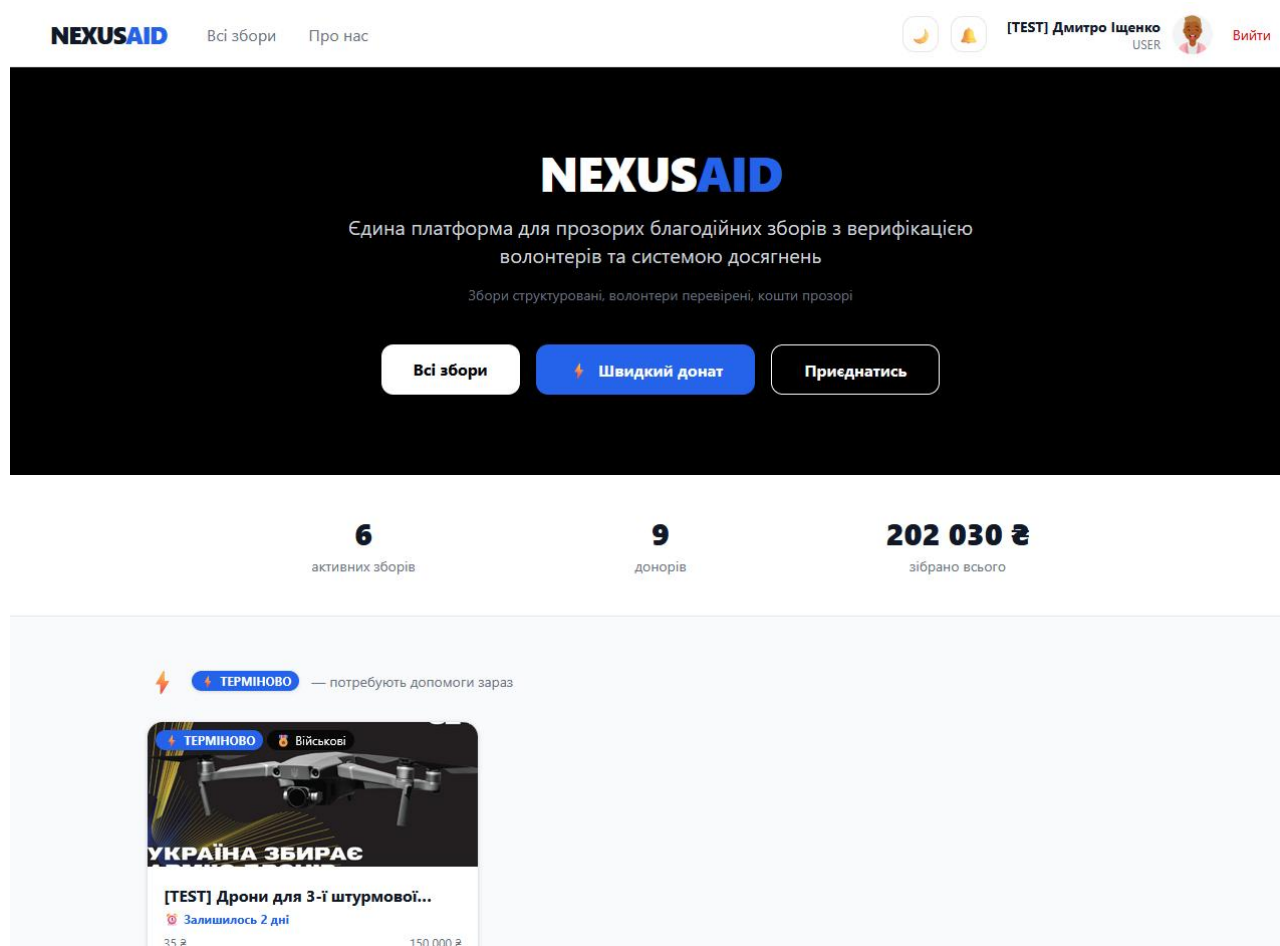


Рис. 3.1. Головна сторінка платформи

Сторінка зборів відображає список усіх активних кампаній. У верхній частині розміщено панель фільтрації за категоріями та поле текстового

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		56

пошуку. Кожен збір представлено карткою з назвою, коротким описом, прогрес-баром зібраної суми та кнопкою переходу до детальної сторінки (рис. 3.2.).

Фільтрація та пошук на сторінці зборів дають змогу користувачу швидко знайти кампанію за тематикою або ключовими словами. Це особливо важливо для платформи з великою кількістю зборів, де просте відображення всіх кампаній без інструментів пошуку ускладнювало б взаємодію. Користувач може перейти від загального перегляду до конкретної категорії, порівняти прогрес різних зборів і вибрати той, який відповідає його намірам.

Картка збору подає інформацію у стислому, але достатньому для попереднього вибору вигляді. Вона містить назву, короткий опис, зображення, категорію, суму, прогрес виконання та ознаку терміновості. Такий формат дозволяє поєднати інформативність і зручність, оскільки користувач не перевантажується зайвими даними, але отримує достатньо контексту для переходу на детальну сторінку.

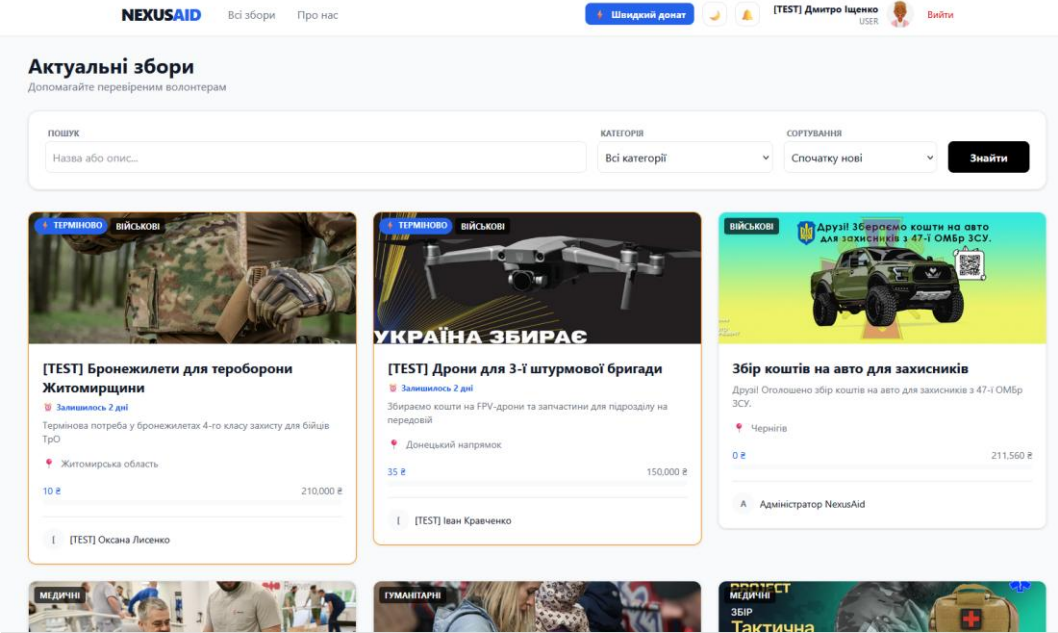


Рис. 3.2. Сторінка списку зборів

Детальна сторінка збору містить повний опис кампанії, зображення, інформацію про організатора та бенефіціара, прогрес-бар із зібраною та цільовою сумою, а також форму для здійснення донату (рис. 3.3.).

Детальна сторінка є центральним елементом взаємодії користувача з конкретною благодійною кампанією. На ній користувач отримує повну інформацію про мету збору, організатора, бенефіціара, необхідну суму та поточний стан накопичення коштів. Важливо, що сторінка поєднує інформаційний блок і функціональний блок донату, тому користувач може одразу перейти від ознайомлення з кампанією до практичної підтримки.

Форма здійснення донату передбачає введення суми, вибір способу участі та подальше перенаправлення на платіжну систему. Для авторизованого користувача результат успішного платежу впливає на профіль, статистику, рівень донора та досягнення. Для неавторизованого користувача зберігається можливість зробити внесок без створення акаунта, що знижує поріг участі та відповідає практиці швидких благодійних зборів.

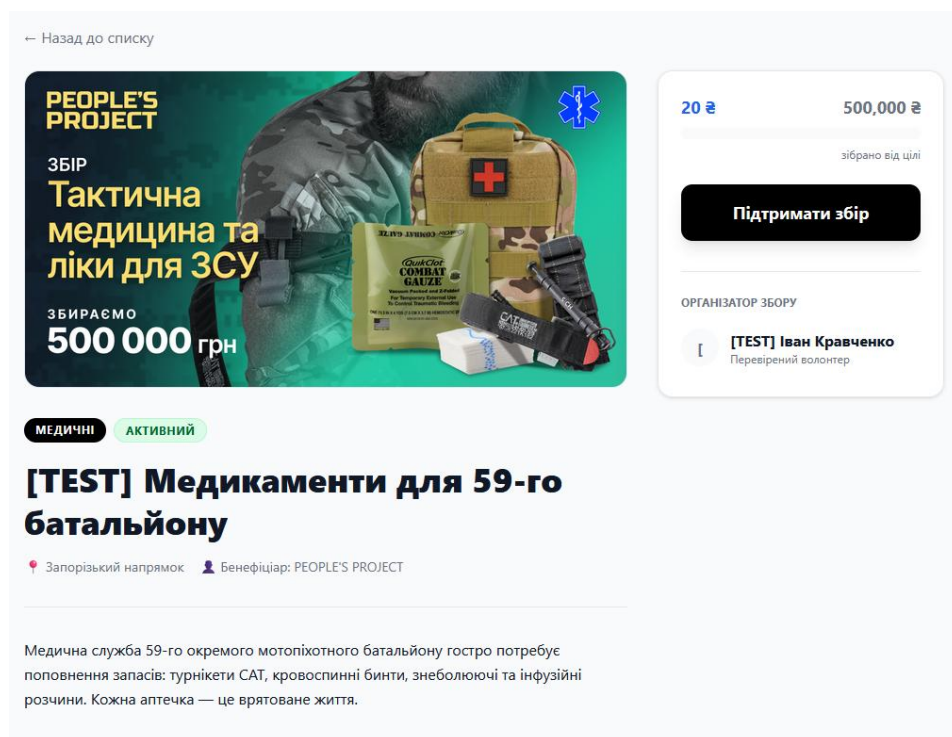


Рис. 3.3. Детальна сторінка збору

Сторінка профілю відображає персональні дані користувача, поточний рівень донора, статистику донатів та картку донора з обраною кастомізацією. Передбачено можливість редагування імені та аватара (рис. 3.4.).

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Профіль користувача об'єднує персональну інформацію та результати його активності на платформі. Через цю сторінку користувач може оцінити власний внесок у благодійні ініціативи, переглянути рівень донора, кількість здійснених донатів і отримані досягнення. Такий підхід робить участь у зборах більш персоналізованою, оскільки користувач бачить накопичений результат своєї активності.

Картка донора в профілі виконує роль візуального представлення участі користувача у платформі. Вона може використовуватися як елемент мотивації, оскільки користувач бачить власний рівень, оформлення та відкриті елементи кастомізації. Завдяки цьому профіль не обмежується технічними налаштуваннями акаунта, а стає частиною гейміфікованої взаємодії.

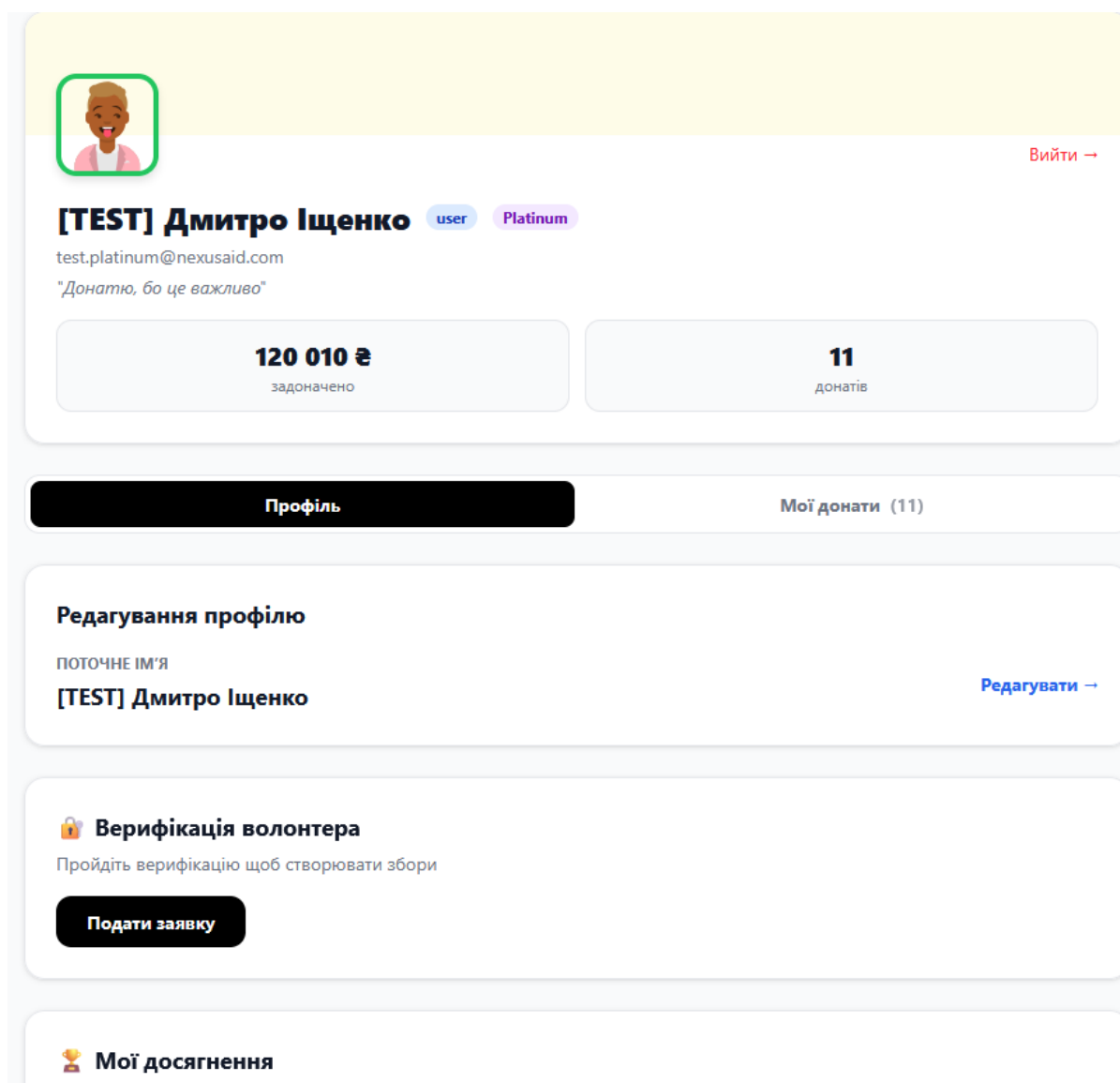


Рис. 3.4. Сторінка профілю користувача

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		59

Сторінка досягнень відображає повний перелік бейджів платформи, розподілених за категоріями. Отримані досягнення виділені кольором, заблоковані відображаються у приглушеному вигляді з описом умови отримання. Окремо показано поточний рівень донора та прогрес до наступного рівня (рис. 3.5.).

Сторінка досягнень дозволяє користувачу бачити не лише вже отримані бейджі, а й потенційні цілі для подальшої активності. Заблоковані досягнення відображаються окремо, тому користувач розуміє, які дії можуть привести до їх відкриття. Такий механізм формує додаткову мотивацію повертатися на платформу, робити нові внески або взаємодіяти з профілем.

Досягнення пов'язані з різними типами активності: сумою донатів, кількістю внесків, регулярністю участі, переглядом профілів і використанням елементів платформи. Це дозволяє заохочувати не одну конкретну дію, а ширшу поведінку користувача. У результаті гейміфікація підтримує основну мету платформи — підвищення залученості до благодійної діяльності.

Мої досягнення

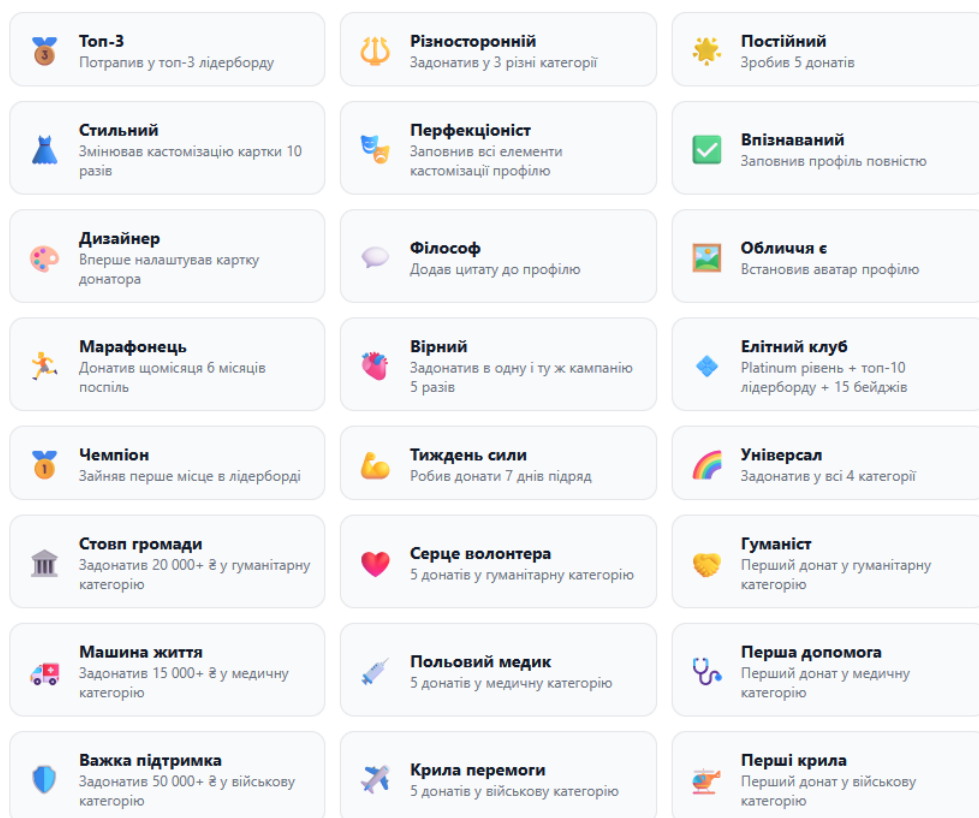


Рис. 3.5. Сторінка досягнень

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		60

Сторінка кастомізації дозволяє обирати розблоковані елементи оформлення картки донора, зокрема рамку, фон та шрифт. Після збереження генерується картка донора, яку можна завантажити як зображення (рис. 3.6.).

Кастомізація картки донора побудована так, щоб користувач міг застосовувати тільки ті елементи, які були розблоковані через активність на платформі. Це створює зв'язок між реальними діями користувача та зовнішнім виглядом його профілю. Рамки, фони, шрифти та інші елементи оформлення стають не просто декоративними налаштуваннями, а видимим результатом участі у зборах.

Після вибору елементів оформлення користувач зберігає налаштування, а система оновлює відображення картки. Згенеровану картку можна використовувати для демонстрації участі в благодійній діяльності, що додатково сприяє поширенню інформації про платформу. Такий механізм поєднує персоналізацію, гейміфікацію та соціальну мотивацію.

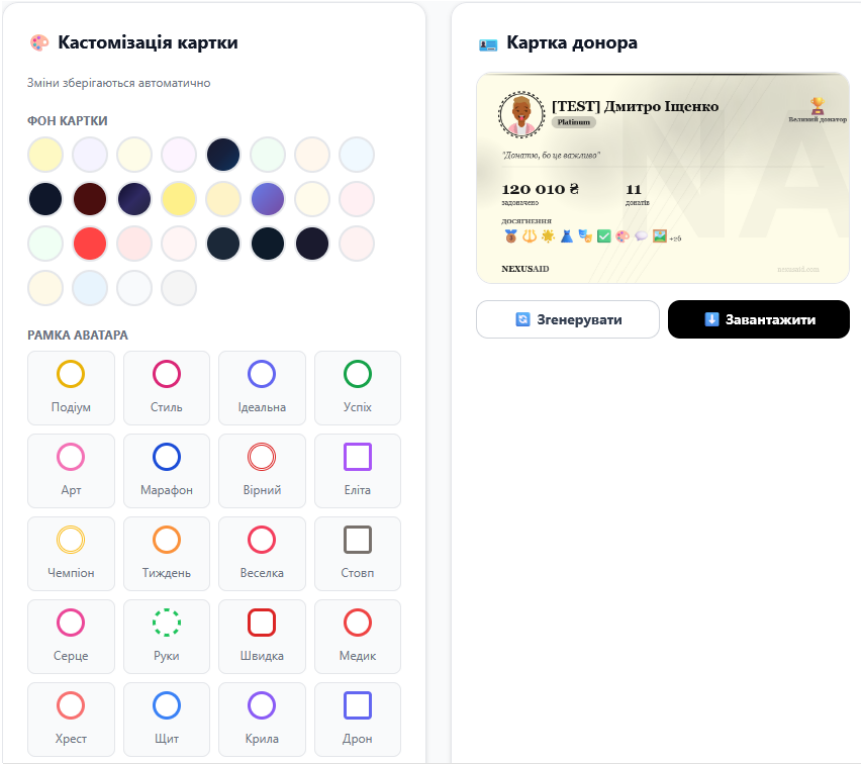


Рис. 3.6. Кастомізація та картка донора

Сторінка таблиці лідерів відображає топ-20 донорів платформи з їх рівнем, загальною сумою донатів та кількістю досягнень (рис. 3.7.).

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		61

Таблиця лідерів підтримує змагальний елемент платформи та дозволяє користувачам бачити найактивніших донорів. Під час формування рейтингу враховується загальна сума донатів, рівень користувача та кількість досягнень. При цьому таблиця повинна залишатися зрозумілою і не перевантажувати користувача технічними деталями, тому основні показники подаються у стислому форматі.

Наявність таблиці лідерів важлива для формування спільноти навколо платформи. Користувач бачить, що благодійність є колективною активністю, а не ізольованою дією окремої людини. Водночас рейтинг не замінює основну мету зборів, а лише доповнює її, створюючи додатковий стимул для регулярної участі.

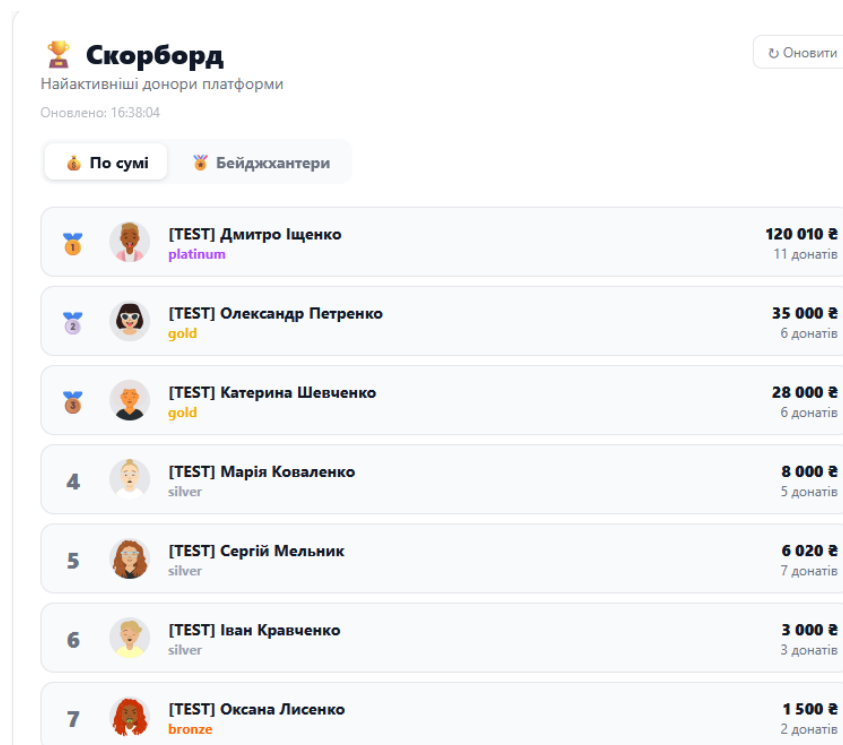


Рис. 3.7. Таблиця лідерів

Сторінка верифікації містить форму подачі заявки на отримання статусу волонтера. Після подачі заявки відображається її поточний статус та чат з адміністратором (рис. 3.8.).

Сторінка верифікації волонтера реалізує один із ключових механізмів довіри в системі. Користувач заповнює заявку, описує власний досвід, додає документи або інші підтверджуючі матеріали, після чого заявка переходить на

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		62

розгляд адміністратора. Такий процес дозволяє відокремити звичайних донорів від користувачів, які мають право створювати збори від власного імені.

Після подання заявки користувач бачить її актуальний статус і може спілкуватися з адміністратором через чат. Це важливо, оскільки в процесі перевірки можуть виникати уточнення щодо документів, досвіду або мети майбутніх зборів. Відображення статусу заявки робить процес прозорим для користувача та зменшує невизначеність під час очікування рішення.

Рис. 3.8. Сторінка верифікації волонтера

Адміністративна панель доступна тільки для користувачів з роллю admin і містить кілька розділів. Розділ управління користувачами відображає повний список облікових записів з можливістю пошуку та фільтрації за роллю. Для кожного користувача доступні зміна ролі, видача або відкликання досягнень та видалення з вказанням причини (рис. 3.9.). Розділ чатів відображає список переписок з усіма волонтерами (рис. 3.10.), а розділ верифікації містить список поданих заявок на статус волонтера з можливістю перегляду деталей та прийняття рішення (рис. 3.11.).

Адміністративна панель об'єднує функції контролю за користувачами, заявками та комунікацією. У розділі управління користувачами адміністратор

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		63

може переглядати облікові записи, змінювати ролі, видавати досягнення вручну або видаляти користувача із зазначенням причини. Такі можливості необхідні для підтримки порядку в системі та реагування на порушення правил платформи.

Панель чатів дає адміністратору змогу переглядати звернення користувачів і відповідати на повідомлення, пов'язані з верифікацією або роботою платформи. Окремий розділ заявок дозволяє швидко оцінювати дані, подані користувачами, приймати рішення щодо підтвердження або відхилення статусу волонтера. Завдяки цьому адміністративна частина підтримує не лише технічне керування системою, а й організаційний процес перевірки довіри.

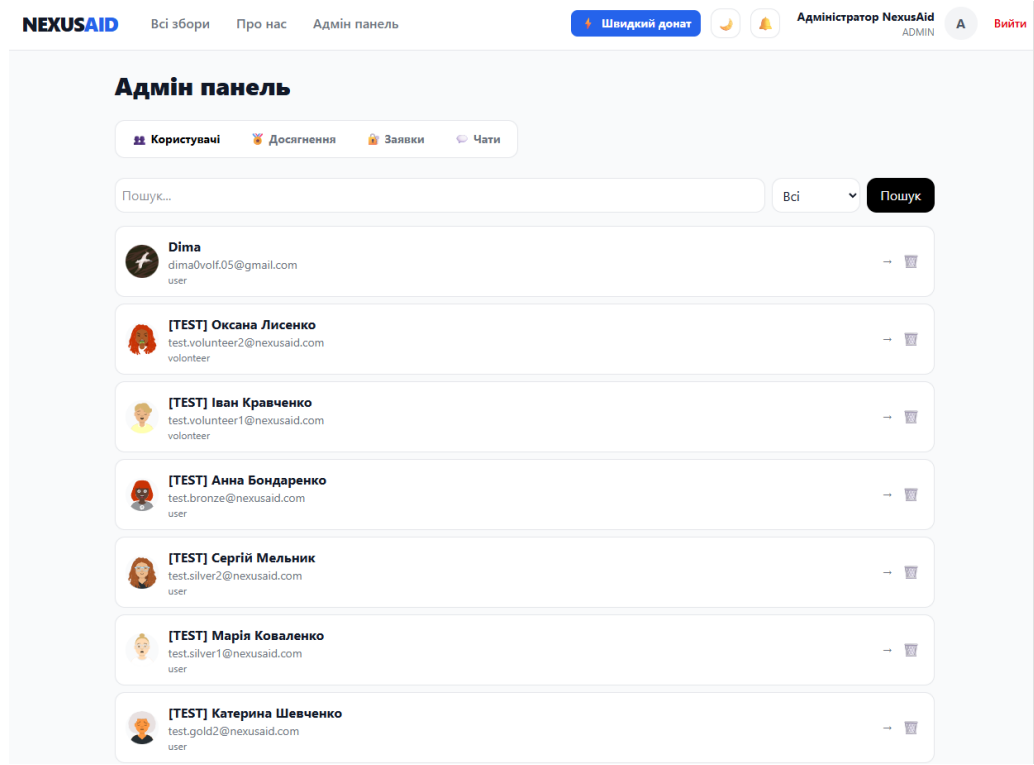


Рис. 3.9. Управління користувачами

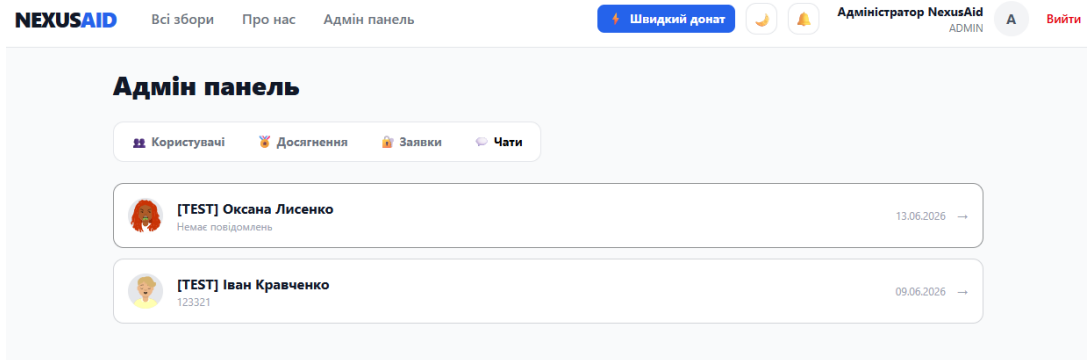


Рис. 3.10. Панель чатів адміністратора

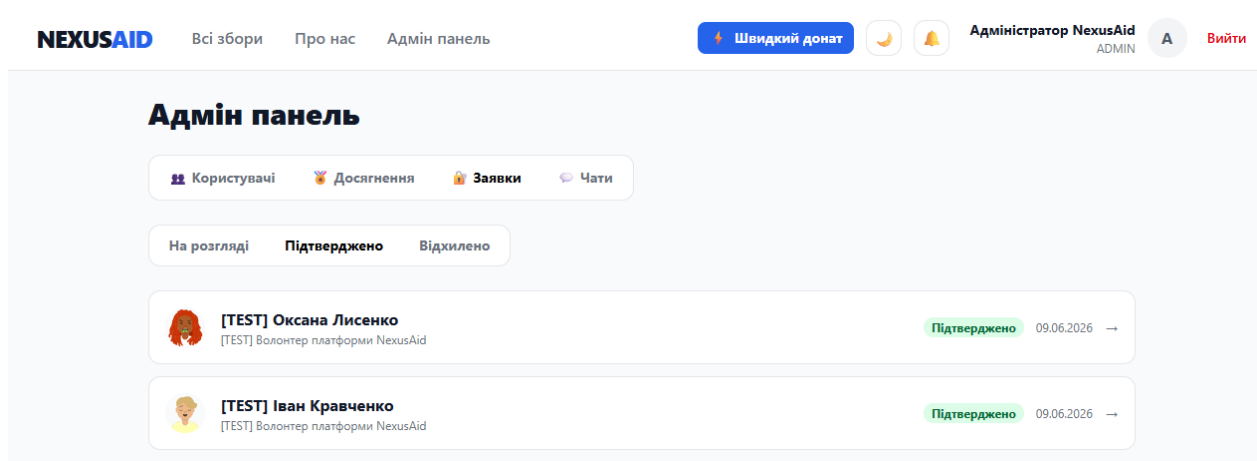


Рис. 3.11. Панель верифікації заявок

3.3 Тестування вебплатформи

Тестування платформи проводилось на двох рівнях: автоматизоване юніт-тестування серверної частини та ручне функціональне тестування повного стеку застосунку.

Для автоматизованого тестування серверної частини використовується фреймворк Jest у поєднанні з вбудованими інструментами тестування NestJS. Тести розміщені поруч з відповідними файлами вихідного коду у файлах з суфіксом .spec.ts.

Розроблено 125 юніт-тестів, що охоплюють такі модулі: AuthService, AppService, AppController, CampaignsService, UsersService, ChatService, VerificationService, GamificationService та DonationsService. Зовнішні залежності (PrismaService, сторонні сервіси) замінюються моками, що дозволяє тестувати бізнес-логіку кожного модуля ізольовано.

Звіт про покриття коду формується автоматично при виконанні команди `npm run test:cov`. Підсумок покриття наведено у таблиці 3.1.

Таблиця 3.1

Покриття коду юніт-тестами за модулями

Модуль	Покриття рядків	Покриття функцій
auth.service.ts	97%	83%
app.controller.ts	91%	67%

продовж. табл. 3.1

app.service.ts	62%	50%
prisma.service.ts	83%	0%
campaigns.service.ts	100%	100%
users.service.ts	93%	90%
donations.service.ts	37%	70%
gamification.service.ts	15%	31%
chat.service.ts	97%	100%
verification.service.ts	100%	100%

Загальне покриття коду становить 26%. Низький відсоток загального покриття пояснюється тим, що значна частина кодової бази складається з контролерів, модулів, DTO та стратегій автентифікації, які є конфігураційними компонентами і перевіряються в умовах реального середовища. Сервісні модулі з бізнес-логікою мають значно вищий відсоток покриття.

Конвеєр CI/CD автоматично запускає всі 125 тестів при кожному коміті. Результати виконання та звіт про покриття інтегровані з SonarCloud та GitLab CI.

Ручне тестування проводилось на розгорнутому виробничому середовищі Railway. Перевірено коректність роботи основних сценаріїв використання системи, які охоплюють не лише вхід у систему, а й основний функціонал платформи.

Під час перевірки реєстрації та автентифікації протестовано створення облікового запису через email і Google OAuth2, вхід в акаунт, збереження сесії через httpOnly cookie та розмежування доступу для різних ролей.

Під час перевірки здійснення донату протестовано ініціювання платежу авторизованим та анонімним користувачем, перенаправлення на платіжну

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

форму WayForPay, коректне оновлення статусу донату та суми збору після підтвердження webhook.

Під час перевірки верифікації волонтера протестовано подачу заявки, отримання сповіщення адміністратором через WebSocket, переписку між користувачем та адміністратором, схвалення заявки та автоматичну зміну ролі без перезавантаження сторінки.

Під час перевірки гейміфікації протестовано автоматичне нарахування досягнень після донатів, підвищення рівня донора при досягненні порогових значень, відображення спливаючого сповіщення через WebSocket та оновлення картки донора.

Під час перевірки адміністративного управління протестовано зміну ролі користувача, видачу досягнення вручну, видалення облікового запису, перегляд повідомлень та відповідь у чаті.

Результати тестування показали, що реалізований застосунок виконує основні функціональні вимоги: забезпечує реєстрацію та автентифікацію, дозволяє переглядати й супроводжувати збори, обробляє донати через платіжний сервіс, підтримує верифікацію волонтерів, нараховує елементи гейміфікації та надає адміністратору інструменти керування системою. Поєднання автоматизованих юніт-тестів і ручної перевірки повних сценаріїв дозволило підтвердити працездатність основних компонентів платформи.

Додатково перевірялася адаптивність основних сторінок платформи. Головна сторінка, список зборів, детальна сторінка, профіль, сторінка досягнень і адміністративна панель переглядалися на різній ширині екрана. Основною вимогою було збереження доступності ключових дій: перегляду інформації, переходу між сторінками, натискання кнопок, заповнення форм і читання повідомлень. Виявлені дрібні недоліки вирівнювання усувалися під час ручного тестування.

Під час перевірки інтерфейсу оцінювалася узгодженість відображення даних на різних сторінках. Після створення або оновлення збору перевірялося, чи правильно він відображається у списку кампаній і на детальній сторінці.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		67

Після здійснення донату перевірялося оновлення прогресу збору, статистики профілю, сторінки досягнень і таблиці лідерів. Така перевірка дозволила переконатися, що клієнтська частина коректно відображає зміни, які відбуваються на сервері.

Тестування платіжного сценарію проводилося з урахуванням того, що WayForPay є зовнішнім сервісом. Перевірялося формування платіжного запиту, передача необхідних параметрів, обробка callback-повідомлення та оновлення статусу донату після отримання підтвердження. Також перевірялася ситуація, коли платіж не завершується успішно: у такому випадку запис про операцію не повинен впливати на суму збору та статистику користувача.

Окремо перевірялася робота WebSocket-сповіщень, оскільки вони впливають на оперативність взаємодії між користувачем і системою. Під час тестування перевірялося отримання повідомлення про нове досягнення, відповідь адміністратора в чаті та зміну статусу верифікації. Для цього одночасно відкривалися сесії користувача й адміністратора, після чого виконувалися дії, які повинні були спричинити появу сповіщення без перезавантаження сторінки.

Функціональне тестування верифікації волонтера включало повний шлях від подання заявки до ухвалення рішення адміністратором. Перевірялося створення заявки, прикріплення даних, зміна статусу, надсилання повідомлення в чаті, отримання відповіді адміністратора та автоматична зміна ролі після схвалення. Також перевірялося, що користувач без підтвердженого статусу не може створювати благодійні збори, навіть якщо він намагається звернутися до відповідного API напрямку.

Під час тестування модуля гейміфікації перевірялася логіка нарахування досягнень і визначення рівня донора. Система повинна коректно реагувати на зміну загальної суми донатів, кількості внесків і виконання інших умов. Якщо користувач уже має певне досягнення, повторне виконання умови не повинно

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		68

створювати дублікати. Це дозволяє зберігати цілісність профілю та забезпечувати правильне відображення нагород у клієнтській частині.

Для модуля благодійних зборів перевірялися створення кампанії, отримання списку зборів, пошук за параметрами, оновлення даних і обробка ситуацій, коли користувач не має достатніх прав. Окремо перевірялося, що зміна суми збору відбувається тільки після успішного підтвердження донату. Така перевірка є важливою, оскільки фінансовий прогрес кампанії не повинен змінюватися через помилкові, скасовані або непідтверджені платежі.

Автоматизовані тести серверної частини перевіряли роботу бізнес-логіки без необхідності вручну виконувати однакові дії після кожної зміни коду. Для сервісів автентифікації перевірялися створення токена, обробка некоректних даних, пошук користувача та поведінка при неправильному паролі. Для сервісів користувачів перевірялися сценарії отримання профілю, зміни ролі та обробки випадків, коли користувача не знайдено. Це дозволило переконатися, що основні серверні функції повертають очікувані результати та коректно реагують на помилки.

Під час тестування особлива увага приділялася тому, щоб перевірити не окремі елементи інтерфейсу, а повні користувацькі сценарії. Для цього використовувалися тестові облікові записи з різними ролями: звичайний користувач, волонтер та адміністратор. Такий підхід дозволив перевірити, чи правильно система обмежує доступ до функцій, чи коректно змінюється поведінка інтерфейсу після авторизації та чи не отримує користувач можливості, які не відповідають його ролі.

Висновки до третього розділу

У третьому розділі описано організацію процесу розробки, структуру інтерфейсу та результати тестування вебплатформи.

У підрозділі 3.1 описано застосовану гілкову стратегію на основі Git з розмежуванням гілок main та develop. Налаштовано два конвеєри CI/CD: на університетській платформі GitLab та GitHub Actions. Конвеєр GitHub Actions охоплює автоматичне тестування, статичний аналіз коду SonarCloud, збірку

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

Docker-образу та перевірку стану розгорнутого застосунку. Реалізовано автоматичну синхронізацію репозиторіїв. Застосунок розгорнуто у хмарному середовищі Railway та доступний за публічним посиланням.

У підрозділі 3.2 описано структуру інтерфейсу вебплатформи. Розглянуто головну сторінку, сторінку зборів, детальну сторінку збору, профіль користувача, систему досягнень, кастомізацію картки донора, таблицю лідерів, сторінку верифікації та адміністративну панель.

У підрозділі 3.3 проведено тестування застосунку на двох рівнях. Розроблено 125 юніт-тестів для дев'яти сервісних модулів, які автоматично виконуються в межах конвеєра CI/CD. Ручним тестуванням перевірено коректність роботи п'яти основних сценаріїв використання системи в умовах виробничого середовища.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

ВИСНОВКИ

У кваліфікаційній роботі вирішено задачу розробки вебплатформи для організації благодійних зборів з елементами гейміфікації. Результатом роботи є повнофункціональний застосунок, розгорнутий у хмарному середовищі Railway та доступний за публічним посиланням.

У ході виконання роботи вирішено такі завдання:

Проведено аналіз предметної області та існуючих платформ для збору коштів. Розглянуто United24, HelpUA Foundation та Donatello. Виявлено основні проблеми: недостатня прозорість використання коштів, відсутність механізмів верифікації організаторів та низький рівень залученості користувачів.

Спроектовано архітектуру системи за клієнт-серверним принципом з розділенням серверної та клієнтської частин на незалежні сервіси. Обґрунтовано вибір технологічного стеку: NestJS і PostgreSQL для серверної частини, Next.js і Tailwind CSS для клієнтської частини.

Спроектовано схему бази даних з 11 взаємопов'язаних таблиць. Реалізовано схему засобами Prisma ORM з управлінням змінами через систему міграцій. Описано відношення між сутностями системи: один-до-одного, один-до-багатьох та багато-до-багатьох.

Реалізовано систему автентифікації на основі JWT з передачею через httpOnly cookie та підтримкою входу через Google OAuth2. Впроваджено модель розмежування доступу на основі ролей: користувач, волонтер та адміністратор.

Розроблено модуль верифікації волонтерів з підтримкою обміну повідомленнями між користувачем та адміністратором у реальному часі через WebSocket.

Реалізовано інтеграцію з платіжною системою WayForPay з використанням HMAC-MD5 підпису для верифікації транзакцій. Підтримується здійснення донатів як авторизованими, так і анонімними користувачами.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

Впроваджено систему гейміфікації з 62 досягненнями, чотирма рівнями донора та таблицею лідерів. Реалізовано кастомізацію картки донора з можливістю її завантаження.

Налаштовано конвеєри CI/CD на базі GitHub Actions та GitLab CI з автоматичним тестуванням, аналізом якості коду через SonarCloud та верифікацією розгортання.

Розроблено 125 юніт-тестів, що охоплюють дев'ять сервісних модулів. Проведено ручне тестування п'яти основних сценаріїв використання системи.

Практичне значення роботи полягає у тому, що розроблений застосунок є готовим до використання рішенням для централізованого управління благодійними зборами з підтримкою прозорих фінансових операцій та інструментами підвищення залученості донорів. Платформа може бути розширена шляхом додавання модуля публікації фінансових звітів, інтеграції з додатковими платіжними системами та підтримки мобільного застосунку.

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		72

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. UNITED24. Official fundraising platform of Ukraine : електронний ресурс. URL: <https://u24.gov.ua> (дата звернення: 14.06.2026).
2. Kusuma F. P. Proposed Gamification Concept to Increase Donor Engagement for Green Fund Digital Philanthropy Platform: Case Study: Greeneration Foundation / F. P. Kusuma, N. N. Arief // International Journal of Current Science Research and Review. 2023. Vol. 6, Issue 1.
3. Zhang Y. Impact of Gamification on Intention to Donate Using Crowdfunding Platform / Y. Zhang et al. // International Journal of Information and Education Technology. 2023. Vol. 14, № 2. P. 45-58.
4. Golrang H. Applying Gamification Design to a Donation-Based Crowdfunding Platform for Improving User Engagement / H. Golrang, E. Safari // Entertainment Computing. 2021. Vol. 38. Article 100425. DOI: 10.1016/j.entcom.2021.100425.
5. Xi N. Does gamification satisfy needs? A study on the relationship between gamification features and intrinsic need satisfaction / N. Xi, J. Hamari // International Journal of Information Management. 2019. Vol. 46. P. 210–221. DOI: 10.1016/j.ijinfomgt.2018.12.002.
6. MDN Web Docs. Learn web development : електронний ресурс. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development (дата звернення: 14.06.2026).
7. Khanna A. Full-Stack Web Development with TypeScript 5 / A. Khanna. Birmingham : Packt Publishing, 2024. 318 p.
8. Vercel Inc. Next.js Documentation : електронний ресурс. URL: <https://nextjs.org/docs> (дата звернення: 11.06.2026).
9. React Team. React Documentation : електронний ресурс. URL: <https://react.dev> (дата звернення: 14.06.2026).
10. Wieruch R. The Road to React / R. Wieruch. Leanpub, 2024. 380 p.
11. Tailwind Labs Inc. Tailwind CSS Documentation : електронний ресурс. URL: <https://tailwindcss.com/docs> (дата звернення: 13.06.2026).

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		73

12. Myśliwiec K. NestJS Documentation : электронный ресурс / K. Myśliwiec. URL: <https://docs.nestjs.com> (дата звернення: 09.06.2026).
13. Casciaro M. Node.js Design Patterns: Design and Implement Production-Grade Node.js Applications Using Proven Patterns and Techniques / M. Casciaro, L. Mammino. 3rd ed. Birmingham : Packt Publishing, 2021. 660 p.
14. PostgreSQL Global Development Group. PostgreSQL 16 Documentation : электронный ресурс. URL: <https://www.postgresql.org/docs/16> (дата звернення: 12.06.2026).
15. Prisma Data Inc. Prisma ORM Documentation : электронный ресурс. URL: <https://www.prisma.io/docs/orm> (дата звернення: 12.06.2026).
16. Microsoft Corporation. TypeScript Handbook : электронный ресурс. URL: <https://www.typescriptlang.org/docs/handbook> (дата звернення: 11.06.2026).
17. Socket.IO contributors. Socket.IO Documentation : электронный ресурс. URL: <https://socket.io/docs/v4> (дата звернення: 13.06.2026).
18. Docker Inc. Docker Documentation : электронный ресурс. URL: <https://docs.docker.com> (дата звернення: 12.06.2026).
19. Poulton N. Docker Deep Dive: Zero to Docker in a Single Book / N. Poulton. Nigel Poulton Ltd, 2023. 425 p.
20. GitHub Inc. GitHub Actions Documentation : электронный ресурс. URL: <https://docs.github.com/en/actions> (дата звернення: 13.06.2026).
21. WayForPay LLC. WayForPay API Documentation : электронный ресурс. URL: <https://wiki.wayforpay.com/uk> (дата звернення: 14.06.2026).
22. Ferrari L. Learn PostgreSQL: Build and manage high-performance database solutions using PostgreSQL 12 and 13 / L. Ferrari, E. Pirozzi. Birmingham : Packt Publishing, 2020. 650 p.
23. Hanson J. Passport.js Documentation : электронный ресурс / J. Hanson. URL: <https://www.passportjs.org/docs> (дата звернення: 13.06.2026).
24. Google LLC. OAuth 2.0 for Web Server Applications : электронный ресурс. URL: <https://developers.google.com/identity/protocols/oauth2/web-server> (дата звернення: 14.06.2026).

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

25. Open Web Application Security Project. OWASP Top Ten 2021 : електронний ресурс. URL: <https://owasp.org/Top10> (дата звернення: 14.06.2026).
26. SmartBear Software. OpenAPI Specification 3.1 : електронний ресурс. URL: <https://swagger.io/specification> (дата звернення: 14.06.2026).
27. Humble J. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation / J. Humble, D. Farley. Boston : Addison-Wesley, 2010. 512 p.
28. Railway Corporation. Railway Documentation : електронний ресурс. URL: <https://docs.railway.app> (дата звернення: 13.06.2026).
29. GitLab Inc. GitLab CI/CD Documentation : електронний ресурс. URL: <https://docs.gitlab.com/ee/ci> (дата звернення: 14.06.2026).
30. SonarSource SA. SonarCloud Documentation : електронний ресурс. URL: <https://docs.sonarcloud.io> (дата звернення: 13.06.2026).

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

ДОДАТКИ

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

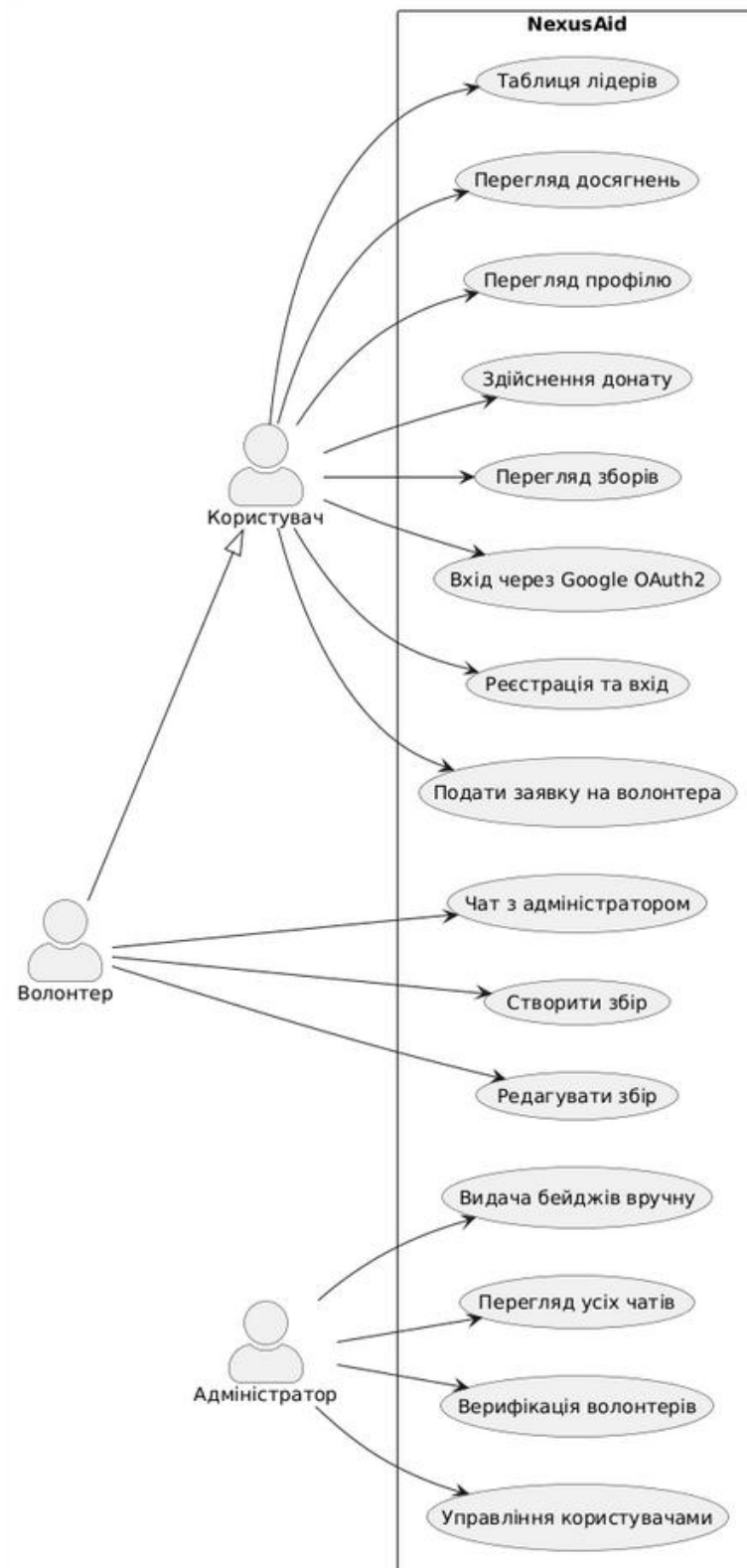


Рис. А.1. Use Case Діаграма

auth.services.ts:

```
import { Injectable, BadRequestException } from '@nestjs/common';
import { PrismaService } from '../prisma.service';
import { JwtService } from '@nestjs/jwt';
import { RegisterDto } from '../dto/register.dto';
import { GamificationService } from '../gamification/gamification.service';
import * as bcrypt from 'bcrypt';
```

@Injectable()

export class AuthService {

constructor(

private prisma: PrismaService,

private jwtService: JwtService,

private gamification: GamificationService,

) {}

async validateUser(email: string, pass: string): Promise<any> {

const user = await this.prisma.user.findUnique({ where: { email } });

if (user && user.password && (await bcrypt.compare(pass, user.password))) {

const { password, ...result } = user;

return result;

}

return null;

}

async login(user: any) {

const payload = { username: user.email, sub: user.id, role: user.role };

return {

access_token: this.jwtService.sign(payload),

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

    userId: user.id,
    role: user.role,
    name: user.name,
  };
}

async register(dto: RegisterDto) {
  const oldUser = await this.prisma.user.findUnique({
    where: { email: dto.email },
  });
  if (oldUser)
    throw new BadRequestException('Користувач з таким email вже існує');

  const salt = await bcrypt.genSalt(10);
  const hashedPassword = await bcrypt.hash(dto.password, salt);
  const avatarSeed = encodeURIComponent(dto.name);

  const user = await this.prisma.user.create({
    data: {
      email: dto.email,
      name: dto.name,
      password: hashedPassword,
      avatar: `https://api.dicebear.com/7.x/avataaars/svg?seed=${avatarSeed}`,
      isVerified: true,
    },
  });

  await this.gamification.grantBadgeSystem(user.id, 'welcome');
  return this.login(user);
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

async validateOAuthUser(data: {
  email: string;
  name: string;
  avatar?: string;
  provider: string;
  socialId: string;
}) {
  let user = await this.prisma.user.findFirst({
    where: { provider: data.provider, socialId: data.socialId },
  });

  if (!user && data.email) {
    user = await this.prisma.user.findUnique({
      where: { email: data.email },
    });
    if (user) {
      user = await this.prisma.user.update({
        where: { id: user.id },
        data: { provider: data.provider, socialId: data.socialId },
      });
    }
  }

  if (!user) {
    const avatarSeed = encodeURIComponent(data.name || data.email || 'user');
    user = await this.prisma.user.create({
      data: {
        email: data.email,
        name: data.name,
        avatar:

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

        data.avatar ||
        `https://api.dicebear.com/7.x/avataaars/svg?seed=${avatarSeed}`,
        provider: data.provider,
        socialId: data.socialId,
        isVerified: true,
    },
    });
    await this.gamification.grantBadgeSystem(user.id, 'welcome');
}

return user;
}

async getWsToken(userId: string, role: string): Promise<string> {
    return this.jwtService.sign({ sub: userId, role }, { expiresIn: '24h' });
}
}

campaigns.controller.ts:

import {
    Injectable,
    ForbiddenException,
    NotFoundException,
} from '@nestjs/common';
import { CreateCampaignDto } from '../dto/create-campaign.dto';
import { PrismaService } from '../prisma.service';
import { GamificationService } from '../gamification/gamification.service';

@Injectable()
export class CampaignsService {
    constructor(

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

private prisma: PrismaService,
private gamification: GamificationService,
) {}

async create(createCampaignDto: CreateCampaignDto, authorId: string) {
  const campaign = await this.prisma.campaign.create({
    data: {
      title: createCampaignDto.title,
      shortDescription: createCampaignDto.shortDescription,
      fullDescription: createCampaignDto.fullDescription,
      goalAmount: createCampaignDto.goalAmount,
      location: createCampaignDto.location,
      beneficiary: createCampaignDto.beneficiary,
      category: createCampaignDto.category,
      images: createCampaignDto.images,
      documents: createCampaignDto.documents,
      isUrgent: createCampaignDto.isUrgent || false,
      urgentUntil: createCampaignDto.urgentUntil
        ? new Date(createCampaignDto.urgentUntil + 'T23:59:59.000Z')
        : null,
      status: 'active',
      authorId,
    },
  });

  await this.gamification.grantBadgeSystem(authorId, 'campaign_creator');

  return campaign;
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

async delete(id: string) {
  return this.prisma.campaign.delete({ where: { id } });
}

async findAll(search?: string, sort?: 'asc' | 'desc', category?: string) {
  const where: any = {};

  if (search) {
    where.OR = [
      { title: { contains: search, mode: 'insensitive' } },
      { shortDescription: { contains: search, mode: 'insensitive' } },
    ];
  }

  if (category && category !== 'all') {
    where.category = category;
  }

  return this.prisma.campaign.findMany({
    where,
    orderBy: { createdAt: sort === 'asc' ? 'asc' : 'desc' },
    include: {
      author: { select: { name: true, email: true } },
    },
  });
}

async findOne(id: string) {
  return this.prisma.campaign.findUnique({
    where: { id },
  });
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

include: {
  author: { select: { id: true, name: true, email: true, avatar: true } },
},
});
}

```

```

async getUrgentByCategory() {
  const categories = ['military', 'medical', 'humanitarian', 'general'];
  const now = new Date();

```

```

const results = await Promise.all(
  categories.map(async (cat) => {
    const campaign = await this.prisma.campaign.findFirst({
      where: {
        category: cat,
        status: 'active',
        isUrgent: true,
        urgentUntil: { gte: now },
      },
      orderBy: [{ urgentUntil: 'asc' }, { createdAt: 'asc' }],
      include: {
        author: { select: { id: true, name: true, avatar: true } },
      },
    });
    return { category: cat, campaign };
  }),
);

```

```

return results.filter((r) => r.campaign !== null);
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

async getRandomByCategory(category?: string) {
  const where: any = { status: 'active' };
  if (category) where.category = category;

  const count = await this.prisma.campaign.count({ where });
  if (count === 0) return null;

  const skip = Math.floor(Math.random() * count);
  const campaigns = await this.prisma.campaign.findMany({
    where,
    skip,
    take: 1,
    include: {
      author: { select: { id: true, name: true, avatar: true } },
    },
  });

  return campaigns[0] || null;
}

async update(id: string, updateData: any, userId: string) {
  const currentCampaign = await this.prisma.campaign.findUnique({
    where: { id },
  });

  if (!currentCampaign) throw new NotFoundException('Збір не знайдено');

  if (currentCampaign.authorId !== userId) {
    const user = await this.prisma.user.findUnique({ where: { id: userId } });

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```

if (!user || user.role !== 'admin') {
  throw new ForbiddenException('Ви не маєте права редагувати цей збір');
}
}

const newImages =
  updateData.images && updateData.images.length > 0
    ? [...updateData.images, ...currentCampaign.images.slice(1)]
    : currentCampaign.images;

const newDocuments = updateData.documents
  ? [...currentCampaign.documents, ...updateData.documents]
  : currentCampaign.documents;

return this.prisma.campaign.update({
  where: { id },
  data: {
    title: updateData.title,
    shortDescription: updateData.shortDescription,
    fullDescription: updateData.fullDescription,
    goalAmount: updateData.goalAmount
      ? Number(updateData.goalAmount)
      : undefined,
    location: updateData.location,
    beneficiary: updateData.beneficiary,
    category: updateData.category,
    status: updateData.status,
    isUrgent:
      updateData.isUrgent !== undefined
        ? updateData.isUrgent === 'true' || updateData.isUrgent === true

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

        : undefined,
    urgentUntil: updateData.urgentUntil
    ? new Date(
        updateData.urgentUntil.includes('T')
        ? updateData.urgentUntil
        : updateData.urgentUntil + 'T23:59:59.000Z',
    )
    : undefined,
    images: newImages,
    documents: newDocuments,
},
});
}
}

```

donations.service.ts:

```

import {
    Injectable,
    NotFoundException,
    BadRequestException,
} from '@nestjs/common';
import { PrismaService } from '../prisma.service';
import { InitiateDonationDto } from '../dto/initiate-donation.dto';
import * as crypto from 'crypto';
import { ConfigService } from '@nestjs/config';
import { GamificationService } from '../gamification/gamification.service';

```

```

@Injectable()
export class DonationsService {
    constructor(
        private prisma: PrismaService,

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		87

```

private config: ConfigService,
private gamification: GamificationService,
) {}

```

```

private getSecretKey(): string {
    const key = this.config.get<string>('WFP_SECRET_KEY');
    if (!key) throw new Error('WFP_SECRET_KEY не вказано в .env');
    return key;
}

```

```

private generateOrderReference(): string {
    const timestamp = Date.now();
    const random = Math.random().toString(36).substring(2, 8).toUpperCase();
    return `NEXUSAID_${timestamp}_${random}`;
}

```

```

private generateSignature(params: string[]): string {
    return crypto
        .createHmac('md5', this.getSecretKey())
        .update(params.join(';'))
        .digest('hex');
}

```

```

private async checkWayForPayStatus(orderReference: string): Promise<string> {
    try {
        const merchantAccount =
this.config.get<string>('WFP_MERCHANT_ACCOUNT')!;
        const signature = this.generateSignature([
            merchantAccount,
            orderReference,

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		88

```
]);
```

```
const response = await fetch('https://api.wayforpay.com/api', {  
  method: 'POST',  
  headers: { 'Content-Type': 'application/json' },  
  body: JSON.stringify({  
    transactionType: 'CHECK_STATUS',  
    merchantAccount,  
    orderReference,  
    merchantAuthType: 'SimpleSignature',  
    merchantSignature: signature,  
    apiVersion: 1,  
  })),  
});
```

```
const data = (await response.json()) as any;  
console.log('[CHECK_STATUS] response:', JSON.stringify(data));  
return data.transactionStatus || 'Unknown';  
} catch (e: any) {  
  console.error('[CHECK_STATUS] error:', e?.message);  
  return 'Unknown';  
}  
}
```

```
async initiate(dto: InitiateDonationDto, userId?: string) {  
  const campaign = await this.prisma.campaign.findUnique({  
    where: { id: dto.campaignId },  
  });  
}
```

```
if (!campaign) throw new NotFoundException('Кампанію не знайдено');
```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		89

```

if (campaign.status !== 'active')
    throw new BadRequestException('Кампанія не активна');

const orderReference = this.generateOrderReference();
const orderDate = Math.floor(Date.now() / 1000);
const merchantAccount =
this.config.get<string>('WFP_MERCHANT_ACCOUNT')!;
const merchantDomain =
this.config.get<string>('WFP_MERCHANT_DOMAIN')!;
const returnUrl = this.config.get<string>('WFP_RETURN_URL')!;
const serviceUrl = this.config.get<string>('WFP_SERVICE_URL')!;

const progressAtDonation =
    campaign.goalAmount > 0
        ? Math.round((campaign.currentAmount / campaign.goalAmount) * 100)
        : 0;

await this.prisma.donation.create({
    data: {
        amount: dto.amount,
        orderReference,
        status: 'pending',
        campaignId: dto.campaignId,
        donorId: userId || null,
        donorName: dto.donorName || null,
        donorEmail: dto.donorEmail || null,
        campaignProgressAtDonation: progressAtDonation,
    },
});

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		90

```

const merchantSignature = this.generateSignature([
    merchantAccount,
    merchantDomain,
    orderReference,
    String(orderDate),
    String(dto.amount),
    'UAH',
    campaign.title,
    '1',
    String(dto.amount),
]);

```

```

return {
    orderReference,
    merchantAccount,
    merchantDomain,
    merchantSignature,
    orderDate,
    amount: dto.amount,
    currency: 'UAH',
    productName: campaign.title,
    returnUrl,
    serviceUrl,
};
}

```

```

async verifyReturn(params: Record<string, string>) {
    const { orderReference } = params;

```

```

    if (!orderReference)

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		91

```

throw new BadRequestException('Missing orderReference');

const donation = await this.prisma.donation.findUnique({
  where: { orderReference },
  include: { campaign: { select: { category: true } } },
});

if (!donation) throw new NotFoundException('Донат не знайдено');
if (donation.status === 'approved') {
  return { success: true, alreadyProcessed: true };
}

const transactionStatus = await this.checkWayForPayStatus(orderReference);
console.log('[VERIFY_RETURN] status from WFP:', transactionStatus);

if (transactionStatus === 'Approved') {
  await this.prisma.$transaction([
    this.prisma.donation.update({
      where: { orderReference },
      data: { status: 'approved' },
    }),
    this.prisma.campaign.update({
      where: { id: donation.campaignId },
      data: { currentAmount: { increment: donation.amount } },
    }),
  ]);

  if (donation.donorId) {
    try {
      await this.gamification.processAfterDonation(

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		92

```

        donation.donorId,
        donation.campaign.category,
        donation.amount,
        donation.campaignId,
    );
} catch (e: any) {
    console.error('[VERIFY_RETURN] Gamification error:', e?.message);
}
}

try {
    const updatedCampaign = await this.prisma.campaign.findUnique({
        where: { id: donation.campaignId },
        select: { currentAmount: true, goalAmount: true },
    });
    if (
        updatedCampaign &&
        updatedCampaign.currentAmount >= updatedCampaign.goalAmount
    ) {
        await this.gamification.processAfterCampaignFunded(
            donation.campaignId,
        );
    }
} catch (e: any) {
    console.error('[VERIFY_RETURN] Campaign funded error:', e?.message);
}

return { success: true };
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		93

```

if (['Declined', 'Expired'].includes(transactionStatus)) {
  await this.prisma.donation.update({
    where: { orderReference },
    data: { status: 'declined' },
  });
  return { success: false, status: transactionStatus };
}

```

```

return { success: false, status: transactionStatus };
}

```

```

async handleWebhook(body: any) {
  const expectedSignature = this.generateSignature([
    String(body.merchantAccount),
    String(body.orderReference),
    String(body.amount),
    String(body.currency),
    String(body.authCode ?? ''),
    String(body.cardPan ?? ''),
    String(body.transactionStatus),
    String(body.reasonCode ?? ''),
  ]);

```

```

if (expectedSignature !== body.merchantSignature) {
  return { status: 'error', message: 'Invalid signature' };
}

```

```

const donation = await this.prisma.donation.findUnique({
  where: { orderReference: body.orderReference },
  include: { campaign: { select: { category: true } } },

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		94

```

});

if (!donation) return { status: 'error', message: 'Donation not found' };
if (donation.status === 'approved') {
  const responseTime = Math.floor(Date.now() / 1000);
  return {
    orderReference: body.orderReference,
    status: 'accept',
    time: responseTime,
    signature: this.generateSignature([
      body.orderReference,
      'accept',
      String(responseTime),
    ]),
  };
}

```

```

if (body.transactionStatus === 'Approved') {
  await this.prisma.$transaction([
    this.prisma.donation.update({
      where: { orderReference: body.orderReference },
      data: {
        status: 'approved',
        transactionId: String(body.transactionId),
      },
    }),
    this.prisma.campaign.update({
      where: { id: donation.campaignId },
      data: { currentAmount: { increment: donation.amount } },
    }),
  ],

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		95

```

]);

if (donation.donorId) {
  try {
    await this.gamification.processAfterDonation(
      donation.donorId,
      donation.campaign.category,
      donation.amount,
      donation.campaignId,
    );
  } catch (e: any) {
    console.error('[WEBHOOK] Gamification error:', e?.message);
  }
}

try {
  const updatedCampaign = await this.prisma.campaign.findUnique({
    where: { id: donation.campaignId },
    select: { currentAmount: true, goalAmount: true },
  });
  if (
    updatedCampaign &&
    updatedCampaign.currentAmount >= updatedCampaign.goalAmount
  ) {
    await this.gamification.processAfterCampaignFunded(
      donation.campaignId,
    );
  }
} catch (e: any) {
  console.error('[WEBHOOK] Campaign funded error:', e?.message);
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		96

```

    }
  } else if (['Declined', 'Expired'].includes(body.transactionStatus)) {
    await this.prisma.donation.update({
      where: { orderReference: body.orderReference },
      data: { status: 'declined' },
    });
  }
}

```

```

const responseTime = Math.floor(Date.now() / 1000);
return {
  orderReference: body.orderReference,
  status: 'accept',
  time: responseTime,
  signature: this.generateSignature([
    body.orderReference,
    'accept',
    String(responseTime),
  ]),
};
}

```

```

async findByUser(userId: string) {
  return this.prisma.donation.findMany({
    where: { donorId: userId, status: 'approved' },
    select: {
      id: true,
      amount: true,
      createdAt: true,
      campaign: {
        select: {

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		97

```

        id: true,
        title: true,
        category: true,
        images: true,
    },
    },
    },
    orderBy: { createdAt: 'desc' },
  });
}

```

```

async findByCampaign(campaignId: string) {
  return this.prisma.donation.findMany({
    where: { campaignId, status: 'approved' },
    select: {
      id: true,
      amount: true,
      donorName: true,
      createdAt: true,
      donor: {
        select: {
          name: true,
          avatar: true,
        },
      },
    },
    orderBy: { createdAt: 'desc' },
  });
}
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		98

gamification.service.ts:

```
import {
  Injectable,
  NotFoundException,
  ForbiddenException,
} from '@nestjs/common';
import { PrismaService } from '../prisma.service';
import { DonorLevel } from '@prisma/client';

const LEVEL_THRESHOLDS: Record<DonorLevel, number> = {
  bronze: 0,
  silver: 5000,
  gold: 25000,
  platinum: 100000,
};

const BADGES = [
  {
    key: 'welcome',
    name: 'Ласкаво просимо',
    description: 'Приєднався до платформи NexusAid',
    icon: '🎉',
    category: 'general',
    unlocksFrame: 'frame_star',
    unlocksBackground: '#8fafc',
    unlocksFont: 'font_default',
  },
  {
    key: 'first_donation',
    name: 'Перший крок',
```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		99

```

description: 'Зробив свій перший донат',
icon: '★',
category: 'general',
unlocksFrame: 'frame_star',
unlocksBackground: '#f5f5f5',
unlocksFont: 'font_default',
},
{
key: 'total_10k',
name: 'Меценат',
description: 'Загальна сума донатів понад 10 000 ₪',
icon: '💎',
category: 'general',
unlocksFrame: 'frame_diamond',
unlocksBackground: '#e8f4fd',
unlocksFont: 'font_elegant',
},
{
key: 'total_100k',
name: 'Легенда',
description: 'Загальна сума донатів понад 100 000 ₪',
icon: '👑',
category: 'general',
unlocksFrame: 'frame_crown',
unlocksBackground: '#fef9e7',
unlocksFont: 'font_bold',
},
{
key: 'first_drone',

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		100

```

name: 'Перші крила',
description: 'Перший донат у військову категорію',
icon: '🚁',
category: 'military',
unlocksFrame: 'frame_drone',
unlocksBackground: '#1a1a2e',
unlocksFont: 'font_military',
},
{
key: 'drone_x5',
name: 'Крила перемоги',
description: '5 донатів у військову категорію',
icon: '✈️',
category: 'military',
unlocksFrame: 'frame_wings',
unlocksBackground: '#0d1b2a',
unlocksFont: 'font_bold',
},
{
key: 'heavy_support',
name: 'Важка підтримка',
description: 'Задонатив 50 000+ ₪ у військову категорію',
icon: '🛡️',
category: 'military',
unlocksFrame: 'frame_shield',
unlocksBackground: '#1b2838',
unlocksFont: 'font_military',
},
{

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		101

```

key: 'first_aid',
name: 'Перша допомога',
description: 'Перший донат у медичну категорію',
icon: '🩹',
category: 'medical',
unlocksFrame: 'frame_cross',
unlocksBackground: '#fff5f5',
unlocksFont: 'font_default',
},
{
key: 'field_medic',
name: 'Польовий медик',
description: '5 донатів у медичну категорію',
icon: '💉',
category: 'medical',
unlocksFrame: 'frame_medic',
unlocksBackground: '#ffe8e8',
unlocksFont: 'font_elegant',
},
{
key: 'ambulance',
name: 'Машина життя',
description: 'Задонатив 15 000+ ₪ у медичну категорію',
icon: '🚑',
category: 'medical',
unlocksFrame: 'frame_ambulance',
unlocksBackground: '#ff4444',
unlocksFont: 'font_bold',
},

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		102

```

{
  key: 'first_step_humanitarian',
  name: 'Гуманіст',
  description: 'Перший донат у гуманітарну категорію',
  icon: '🤝',
  category: 'humanitarian',
  unlocksFrame: 'frame_hands',
  unlocksBackground: '#0fff4',
  unlocksFont: 'font_default',
},
{
  key: 'volunteer_heart',
  name: 'Серце волонтера',
  description: '5 донатів у гуманітарну категорію',
  icon: '❤️',
  category: 'humanitarian',
  unlocksFrame: 'frame_heart',
  unlocksBackground: '#fff0f3',
  unlocksFont: 'font_elegant',
},
{
  key: 'pillar',
  name: 'Стовп громади',
  description: 'Задонатив 20 000+ ₪ у гуманітарну категорію',
  icon: '🏠',
  category: 'humanitarian',
  unlocksFrame: 'frame_pillar',
  unlocksBackground: '#fffbef',
  unlocksFont: 'font_bold',
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		103

```
},  
];
```

```
@Injectable()
```

```
export class GamificationService {
```

```
  constructor(private prisma: PrismaService) {}
```

```
  private leaderboardCache: { data: any; expiresAt: number } | null = null;
```

```
  invalidateLeaderboardCache() {
```

```
    this.leaderboardCache = null;
```

```
  }
```

```
  async seedBadges() {
```

```
    for (const badge of BADGES) {
```

```
      await this.prisma.badge.upsert({
```

```
        where: { key: badge.key },
```

```
        update: badge,
```

```
        create: badge,
```

```
      });
```

```
    }
```

```
  }
```

```
  private calculateLevel(totalAmount: number): DonorLevel {
```

```
    if (totalAmount >= LEVEL_THRESHOLDS.platinum) return 'platinum';
```

```
    if (totalAmount >= LEVEL_THRESHOLDS.gold) return 'gold';
```

```
    if (totalAmount >= LEVEL_THRESHOLDS.silver) return 'silver';
```

```
    return 'bronze';
```

```
  }
```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		104

```

async processAfterDonation(
  userId: string,
  campaignCategory: string,
  donationAmount: number,
  campaignId: string,
) {
  const allDonations = await this.prisma.donation.findMany({
    where: { donorId: userId, status: 'approved' },
    include: {
      campaign: {
        select: { category: true, isUrgent: true, urgentUntil: true },
      },
    },
    orderBy: { createdAt: 'desc' },
  });

  const totalAmount = allDonations.reduce((s, d) => s + d.amount, 0);
  const totalCount = allDonations.length;
  const newLevel = this.calculateLevel(totalAmount);
  const now = new Date();
  const today = new Date(now.getFullYear(), now.getMonth(), now.getDate());

  const currentProfile = await this.prisma.donorProfile.findUnique({
    where: { userId },
  });

  let currentDayStreak = 1;
  let longestDayStreak = currentProfile?.longestDayStreak ?? 1;
  let currentMonthStreak = 1;
  let isComeback = false;

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		105

```

if (currentProfile?.lastDonationDate) {
  const last = new Date(currentProfile.lastDonationDate);
  const lastDay = new Date(
    last.getFullYear(),
    last.getMonth(),
    last.getDate(),
  );
  const diffDays = Math.floor(
    (today.getTime() - lastDay.getTime()) / 86400000,
  );

  if (diffDays === 0) {
    currentDayStreak = currentProfile.currentDayStreak;
  } else if (diffDays === 1) {
    currentDayStreak = (currentProfile.currentDayStreak ?? 0) + 1;
  } else {
    currentDayStreak = 1;
  }

  longestDayStreak = Math.max(longestDayStreak, currentDayStreak);

  const monthDiff =
    (now.getFullYear() - last.getFullYear()) * 12 +
    (now.getMonth() - last.getMonth());

  if (monthDiff === 0) {
    currentMonthStreak = currentProfile.currentMonthStreak;
  } else if (monthDiff === 1) {
    currentMonthStreak = (currentProfile.currentMonthStreak ?? 0) + 1;
  }
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		106

```

    } else {
        currentMonthStreak = 1;
    }

    isComeback = diffDays >= 30;
}

await this.prisma.donorProfile.upsert({
    where: { userId },
    update: {
        level: newLevel,
        totalAmount,
        donationCount: totalCount,
        currentDayStreak,
        longestDayStreak,
        currentMonthStreak,
        lastDonationDate: now,
    },
    create: {
        userId,
        level: newLevel,
        totalAmount,
        donationCount: totalCount,
        currentDayStreak: 1,
        longestDayStreak: 1,
        currentMonthStreak: 1,
        lastDonationDate: now,
    },
});

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		107

```

const badges: string[] = [];

if (totalCount >= 1) badges.push('first_donation');
if (totalCount >= 5) badges.push('donations_5');
if (totalCount >= 10) badges.push('donations_10');
if (totalCount >= 25) badges.push('donations_25');
if (totalCount >= 50) badges.push('donations_50');
if (totalCount >= 100) badges.push('donations_100');

if (totalAmount >= 1000) badges.push('total_1k');
if (totalAmount >= 5000) badges.push('total_5k');
if (totalAmount >= 10000) badges.push('total_10k');
if (totalAmount >= 50000) badges.push('total_50k');
if (totalAmount >= 100000) badges.push('total_100k');

if (donationAmount >= 1000) badges.push('big_one_1k');
if (donationAmount >= 5000) badges.push('big_one_5k');
if (donationAmount >= 10000) badges.push('big_one_10k');

if (donationAmount % 1000 === 0) badges.push('round_number');
if (String(donationAmount).includes('7')) badges.push('lucky_7');

const amountStr = String(donationAmount);
if (
    amountStr.length > 1 &&
    amountStr === amountStr.split('').reverse().join('')
) {
    badges.push('palindrome');
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		108

```

if (donationAmount === 777 || donationAmount === 1337)
  badges.push('phantom');

const byCategory = (cat: string) =>
  allDonations.filter((d) => d.campaign.category === cat);

const military = byCategory('military');
if (military.length >= 1) badges.push('first_drone');
if (military.length >= 5) badges.push('drone_x5');
if (military.reduce((s, d) => s + d.amount, 0) >= 50000)
  badges.push('heavy_support');

const medical = byCategory('medical');
if (medical.length >= 1) badges.push('first_aid');
if (medical.length >= 5) badges.push('field_medic');
if (medical.reduce((s, d) => s + d.amount, 0) >= 15000)
  badges.push('ambulance');

const humanitarian = byCategory('humanitarian');
if (humanitarian.length >= 1) badges.push('first_step_humanitarian');
if (humanitarian.length >= 5) badges.push('volunteer_heart');
if (humanitarian.reduce((s, d) => s + d.amount, 0) >= 20000)
  badges.push('pillar');

const uniqueCategories = new Set(
  allDonations.map((d) => d.campaign.category),
);
if (uniqueCategories.size >= 3) badges.push('triple_category');
if (uniqueCategories.size >= 4) badges.push('all_categories');

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		109

```

const hour = now.getHours();
if (hour >= 0 && hour < 5) badges.push('night_owl');
if (hour >= 6 && hour < 8) badges.push('early_bird');
if (now.getMonth() === 1 && now.getDate() === 24)
    badges.push('ukraine_day');
if (now.getMonth() === 7 && now.getDate() === 24)
    badges.push('independence_day');

const todayDonations = allDonations.filter((d) => {
    const d2 = new Date(d.createdAt);
    return (
        d2.getFullYear() === today.getFullYear() &&
        d2.getMonth() === today.getMonth() &&
        d2.getDate() === today.getDate()
    );
});
if (todayDonations.length >= 3) badges.push('speed_donor');

const weekendDonations = allDonations.filter((d) => {
    const day = new Date(d.createdAt).getDay();
    return day === 0 || day === 6;
});
if (weekendDonations.length >= 5) badges.push('weekend_warrior');

if (currentDayStreak >= 3) badges.push('streak_3');
if (currentDayStreak >= 7) badges.push('streak_7');
if (isComeback) badges.push('comeback');
if (currentMonthStreak >= 3) badges.push('monthly_regular');
if (currentMonthStreak >= 6) badges.push('marathon');

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		110

```

const byCampaign = allDonations.reduce(
  (acc, d) => {
    acc[d.campaignId] = (acc[d.campaignId] || 0) + 1;
    return acc;
  },
  {} as Record<string, number>,
);
if (Object.values(byCampaign).some((count: number) => count >= 5))
  badges.push('loyal');

const urgentDonations = allDonations.filter(
  (d) =>
    d.campaign.isUrgent &&
    d.campaign.urgentUntil &&
    new Date(d.createdAt) <= new Date(d.campaign.urgentUntil),
);
const uniqueUrgentCampaigns = new Set(
  urgentDonations.map((d) => d.campaignId),
);
if (uniqueUrgentCampaigns.size >= 5) badges.push('crisis_savior');

const lastStandDonations = allDonations.filter((d) => {
  if (!d.campaign.isUrgent || !d.campaign.urgentUntil) return false;
  const urgentUntilMs = new Date(d.campaign.urgentUntil).getTime();
  const donMs = new Date(d.createdAt).getTime();
  return urgentUntilMs - donMs <= 3 * 60 * 60 * 1000;
});
if (lastStandDonations.length >= 3) badges.push('last_stand');

const leaderboard = await this.prisma.donorProfile.findMany({

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		111

```

    orderBy: { totalAmount: 'desc' },
    take: 10,
    select: { userId: true },
  });

const position = leaderboard.findIndex((p) => p.userId === userId);
if (position >= 0 && position < 3) badges.push('top3_leaderboard');
if (position === 0) badges.push('top1_leaderboard');

const badgeCount = await this.prisma.userBadge.count({ where: { userId } });
if (
  newLevel === 'platinum' &&
  position >= 0 &&
  position < 10 &&
  badgeCount >= 15
) {
  badges.push('elite_club');
}

for (const key of badges) {
  await this.grantBadgeSystem(userId, key);
}

this.invalidateLeaderboardCache();
}

async processAfterProfileUpdate(userId: string) {
  const user = await this.prisma.user.findUnique({ where: { id: userId } });
  const profile = await this.prisma.donorProfile.findUnique({
    where: { userId },
  });
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		112

```

if (user?.avatar) await this.grantBadgeSystem(userId, 'avatar_set');
if (profile?.quote) await this.grantBadgeSystem(userId, 'quote_master');

if (
  user?.avatar &&
  profile?.quote &&
  profile?.selectedFrame &&
  profile?.selectedBackground &&
  profile?.selectedFont &&
  profile?.selectedBadgeId
) {
  await this.grantBadgeSystem(userId, 'perfectionist');
}
}

async processProfileView(viewerId: string, targetId: string) {
  if (viewerId === targetId) return;

  await this.prisma.profileView.create({
    data: { viewerId, targetId },
  });

  const uniqueViewed = await this.prisma.profileView.groupBy({
    by: ['targetId'],
    where: { viewerId },
  });

  if (uniqueViewed.length >= 20) {
    await this.grantBadgeSystem(viewerId, 'social_butterfly');
  }
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		113

```

const totalViews = await this.prisma.profileView.count({
  where: { targetId },
});

if (totalViews >= 50) {
  await this.grantBadgeSystem(targetId, 'inspiration');
}
}

async processAfterCampaignFunded(campaignId: string) {
  const campaign = await this.prisma.campaign.findUnique({
    where: { id: campaignId },
    select: { authorId: true, currentAmount: true, goalAmount: true },
  });

  if (!campaign) return;
  if (campaign.currentAmount < campaign.goalAmount) return;

  await this.grantBadgeSystem(campaign.authorId, 'campaign_funded');
  await this.grantBadgeSystem(campaign.authorId, 'architect');

  const earlyDonations = await this.prisma.donation.findMany({
    where: {
      campaignId,
      status: 'approved',
      campaignProgressAtDonation: { lt: 10 },
    },
    select: { donorId: true },
  });
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		114

```

for (const d of earlyDonations) {
  if (d.donorId) {
    await this.grantBadgeSystem(d.donorId, 'early_supporter');
  }
}

async grantBadgeManually(
  targetUserId: string,
  badgeKey: string,
  adminId: string,
) {
  const admin = await this.prisma.user.findUnique({ where: { id: adminId } });
  if (!admin || admin.role !== 'admin')
    throw new ForbiddenException('Only admin');

  const badge = await this.prisma.badge.findUnique({
    where: { key: badgeKey },
  });
  if (!badge) throw new NotFoundException('Badge not found');

  return this.prisma.userBadge.upsert({
    where: { userId_badgeId: { userId: targetUserId, badgeId: badge.id } },
    update: {},
    create: { userId: targetUserId, badgeId: badge.id, grantedBy: adminId },
  });
}

async grantBadgeSystem(targetUserId: string, badgeKey: string) {
  const badge = await this.prisma.badge.findUnique({

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		115

```

    where: { key: badgeKey },
  });
  if (!badge) return;

  return this.prisma.userBadge.upsert({
    where: { userId_badgeId: { userId: targetUserId, badgeId: badge.id } },
    update: {},
    create: { userId: targetUserId, badgeId: badge.id, grantedBy: null },
  });
}

async getUserProfile(userId: string) {
  const profile = await this.prisma.donorProfile.findUnique({
    where: { userId },
    include: {
      user: { select: { id: true, name: true, avatar: true } },
    },
  });
}

const userBadges = await this.prisma.userBadge.findMany({
  where: { userId },
  include: { badge: true },
  orderBy: { createdAt: 'desc' },
});

return { profile, badges: userBadges.map((ub) => ub.badge) };
}

async updateCustomization(
  userId: string,

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		116

```

data: {
  selectedFrame?: string;
  selectedBackground?: string;
  selectedFont?: string;
  selectedBadgeId?: string;
  quote?: string;
},
) {
  const userBadges = await this.prisma.userBadge.findMany({
    where: { userId },
    include: { badge: true },
  });

  const unlockedFrames = userBadges
    .map((ub) => ub.badge.unlocksFrame)
    .filter(Boolean);
  const unlockedBackgrounds = userBadges
    .map((ub) => ub.badge.unlocksBackground)
    .filter(Boolean);
  const unlockedFonts = userBadges
    .map((ub) => ub.badge.unlocksFont)
    .filter(Boolean);

  if (data.selectedFrame && !unlockedFrames.includes(data.selectedFrame))
    throw new ForbiddenException('Цю рамку ще не розблоковано');
  if (
    data.selectedBackground &&
    !unlockedBackgrounds.includes(data.selectedBackground)
  )
    throw new ForbiddenException('Цей фон ще не розблоковано');

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		117

```

if (data.selectedFont && !unlockedFonts.includes(data.selectedFont))
  throw new ForbiddenException('Цей шрифт ще не розблоковано');

if (data.selectedBadgeId) {
  const badge = await this.prisma.badge.findFirst({
    where: {
      OR: [{ key: data.selectedBadgeId }, { id: data.selectedBadgeId }],
    },
  });
  if (!badge) throw new ForbiddenException('Цей бейдж не розблоковано');
  const hasBadge = userBadges.some((ub) => ub.badgeId === badge.id);
  if (!hasBadge) throw new ForbiddenException('Цей бейдж не
розблоковано');
  data.selectedBadgeId = badge.id;
}

const currentProfile = await this.prisma.donorProfile.findUnique({
  where: { userId },
});
const newCount = (currentProfile?.customizationChangeCount ?? 0) + 1;

const result = await this.prisma.donorProfile.upsert({
  where: { userId },
  update: { ...data, customizationChangeCount: newCount },
  create: { userId, ...data, customizationChangeCount: 1 },
});

await this.grantBadgeSystem(userId, 'card_customizer');
if (newCount >= 10) await this.grantBadgeSystem(userId, 'style_icon');
if (data.quote) await this.grantBadgeSystem(userId, 'quote_master');

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		118

```

    await this.processAfterProfileUpdate(userId);

    return result;
}

async createBadge(data: any) {
    return this.prisma.badge.create({ data });
}

async updateBadge(key: string, data: any) {
    return this.prisma.badge.update({ where: { key }, data });
}

async deleteBadge(key: string) {
    return this.prisma.badge.delete({ where: { key } });
}

async revokeBadge(userId: string, badgeKey: string) {
    const badge = await this.prisma.badge.findUnique({
        where: { key: badgeKey },
    });
    if (!badge) throw new NotFoundException('Badge not found');

    return this.prisma.userBadge.delete({
        where: { userId_badgeId: { userId, badgeId: badge.id } },
    });
}

async getLeaderboard() {

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		119

```

const now = Date.now();

if (this.leaderboardCache && this.leaderboardCache.expiresAt > now) {
  return this.leaderboardCache.data;
}

const profiles = await this.prisma.donorProfile.findMany({
  orderBy: { totalAmount: 'desc' },
  take: 20,
  include: {
    user: { select: { id: true, name: true, avatar: true } },
  },
});

const withBadges = await Promise.all(
  profiles.map(async (p) => {
    const badgeCount = await this.prisma.userBadge.count({
      where: { userId: p.userId },
    });
    return { ...p, badgeCount };
  }),
);

this.leaderboardCache = {
  data: withBadges,
  expiresAt: now + 5 * 60 * 1000,
};

return withBadges;
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		120

```

async getAllBadges() {
  return this.prisma.badge.findMany({ orderBy: { category: 'asc' } });
}

```

verification.service.ts:

```

import {
  Injectable,
  NotFoundException,
  ForbiddenException,
  BadRequestException,
} from '@nestjs/common';
import { PrismaService } from '../prisma.service';
import { UsersService } from '../users/users.service';
import { ChatGateway } from '../chat/chat.gateway';

```

```

@Injectable()
export class VerificationService {
  constructor(
    private prisma: PrismaService,
    private usersService: UsersService,
    private chatGateway: ChatGateway,
  ) {}

```

```

async submitRequest(
  userId: string,
  data: {
    about: string;
    experience: string;
    socialLinks?: string;

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		121

```

    documents: string[];
  },
) {
  const existing = await this.prisma.verificationRequest.findUnique({
    where: { userId },
  });

  if (existing) {
    if (existing.status === 'approved')
      throw new BadRequestException('Ви вже верифіковані');
    if (existing.status === 'pending')
      throw new BadRequestException('Заявка вже на розгляді');
    return this.prisma.verificationRequest.update({
      where: { userId },
      data: { ...data, status: 'pending' },
    });
  }

  const request = await this.prisma.verificationRequest.create({
    data: { userId, ...data },
  });
  this.chatGateway.emitNewVerificationRequest();
  return request;
}

async getMyRequest(userId: string) {
  return this.prisma.verificationRequest.findUnique({
    where: { userId },
    include: {
      messages: {

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		122

```

include: {
  sender: {
    select: { id: true, name: true, avatar: true, role: true },
  },
},
orderBy: { createdAt: 'asc' },
},
});
}

async getAllRequests(status?: string) {
  return this.prisma.verificationRequest.findMany({
    where: status ? { status: status as any } : {},
    include: {
      user: { select: { id: true, name: true, email: true, avatar: true } },
      messages: { orderBy: { createdAt: 'desc' }, take: 1 },
    },
    orderBy: { createdAt: 'desc' },
  });
}

async getRequestById(id: string) {
  const request = await this.prisma.verificationRequest.findUnique({
    where: { id },
    include: {
      user: {
        select: {
          id: true,
          name: true,

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		123

```

        email: true,
        avatar: true,
        role: true,
    },
},
messages: {
    include: {
        sender: {
            select: { id: true, name: true, avatar: true, role: true },
        },
    },
    orderBy: { createdAt: 'asc' },
},
});
if (!request) throw new NotFoundException('Заявку не знайдено');
return request;
}

```

```

async sendMessage(
    requestId: string,
    senderId: string,
    text: string,
    isAdmin: boolean,
) {
    const request = await this.prisma.verificationRequest.findUnique({
        where: { id: requestId },
    });
    if (!request) throw new NotFoundException('Заявку не знайдено');
    if (!isAdmin && request.userId !== senderId)

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		124

```

throw new ForbiddenException('Немає доступу');

const message = await this.prisma.verificationMessage.create({
  data: { requestId, senderId, text, isAdmin },
  include: {
    sender: { select: { id: true, name: true, avatar: true, role: true } },
  },
});

if (isAdmin) {
  this.chatGateway.emitVerificationMessage(request.userId, message);
} else {
  this.chatGateway.emitNewVerificationRequest();
}

return message;
}

async updateStatus(
  id: string,
  status: 'approved' | 'rejected',
  adminId: string,
) {
  const request = await this.prisma.verificationRequest.findUnique({
    where: { id },
    include: { user: true },
  });
  if (!request) throw new NotFoundException('Заявку не знайдено');

  await this.prisma.verificationRequest.update({

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		125

```

    where: { id },
    data: { status },
  });

```

```

if (status === 'approved') {
  await this.prisma.user.update({
    where: { id: request.userId },
    data: { role: 'volunteer' },
  });
  this.chatGateway.emitVerificationApproved(request.userId);
}

```

```

await this.prisma.verificationMessage.create({
  data: {
    requestId: id,
    senderId: adminId,
    isAdmin: true,
    text:
      status === 'approved'
        ? '✅ Вашу заявку підтверджено. Вітаємо, ви тепер верифікований
        волонтер!'
        : '❌ На жаль, вашу заявку відхилено. Ви можете подати нову заявку з
        оновленою інформацією.',
  },
});

return { status };
}
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		126

NotificationSocket.tsx

```
'use client';

import { useEffect, useRef } from 'react';
import { io, Socket } from 'socket.io-client';
import { useAuth } from '@context/AuthContext';
import { useNotification } from '@context/NotificationContext';

const BACKEND_URL = process.env.NEXT_PUBLIC_API_URL ||
'https://nexus-aid-production.up.railway.app';

export default function NotificationSocket() {
  const { user, refreshProfile } = useAuth();
  const { addToToast, setUnreadChatCount, setNewBadgeCount,
setAdminUnreadCount, setSocket, isChatOpen } = useNotification();
  const socketRef = useRef<Socket | null>(null);
  const badgeIntervalRef = useRef<ReturnType<typeof setInterval> | null>(null);
  const isChatOpenRef = useRef(isChatOpen);

  useEffect(() => { isChatOpenRef.current = isChatOpen; }, [isChatOpen]);

  useEffect(() => {
    if (!user) {
      socketRef.current?.disconnect();
      socketRef.current = null;
      setSocket(null);
      if (badgeIntervalRef.current) clearInterval(badgeIntervalRef.current);
      return;
    }
  })
```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		127

```

const knownRole = localStorage.getItem('known_role');
if (knownRole && knownRole !== user.role && user.role === 'volunteer') {
    addToast('🎉 Вашу заявку схвалено! Ви тепер волонтер', 'success',
    '🎉');
}
localStorage.setItem('known_role', user.role);

const checkBadges = async () => {
    try {
        const res = await fetch('/api/gamification/me', { credentials: 'include' });
        if (!res.ok) return;
        const data = await res.json();
        const badges: { key: string; name: string; icon: string }[] = data.badges ||
        [];

        const stored = localStorage.getItem('known_badge_ids');
        const known = stored !== null ? JSON.parse(stored) as string[] : null;
        if (known !== null) {
            const newBadges = badges.filter(b => !known.includes(b.key));
            if (newBadges.length > 0) {
                newBadges.forEach(b => addToast('🏆 Новий бейдж:
                ${b.name}', 'badge', b.icon));
                setNewBadgeCount(prev => prev + newBadges.length);
            }
        }
        localStorage.setItem('known_badge_ids', JSON.stringify(badges.map(b
        => b.key)));
    } catch {}
};

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		128

```

const initAdminCounts = async () => {
  try {
    const [chatsRes, verRes] = await Promise.all([
      fetch('/api/chat/admin/all', { credentials: 'include' }),
      fetch('/api/verification?status=pending', { credentials: 'include' }),
    ]);
    const chats = chatsRes.ok ? await chatsRes.json() : [];
    const vers = verRes.ok ? await verRes.json() : [];
    const chatCount = Array.isArray(chats) ? chats.filter((c: any) =>
c.unreadCount > 0).length : 0;
    setAdminUnreadCount(chatCount + (Array.isArray(vers) ? vers.length :
0));
  } catch {}
};

const connect = async () => {
  try {
    const res = await fetch('/api/auth/ws-token', { credentials: 'include' });
    const { token } = await res.json();

    const socket = io(`${BACKEND_URL}/chat`, {
      auth: { token },
      transports: ['websocket', 'polling'],
    });

    socket.on('connect', () => {
      console.log('[WS] Connected/Reconnected');
      setSocket(socket);
    });
  }
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		129

```

socket.on('disconnect', () => {
    console.log('[WS] Disconnected');
    setSocket(null);
});

socket.on('new_message', (data: any) => {
    if (data.isAdmin === true && user.role === 'volunteer' &&
!isChatOpenRef.current) {
        setUnreadChatCount(prev => prev + 1);
        addToast('💬 Нове повідомлення від адміністратора', 'message');
    }
    if (data.chatUserId && data.message && user.role === 'admin') {
        setAdminUnreadCount(prev => prev + 1);
    }
});

socket.on('verification_approved', async () => {
    addToast('🎉 Вашу заявку схвалено! Ви тепер волонтер', 'success',
'🎉');
    localStorage.setItem('known_role', 'volunteer');
    await refreshProfile();
});

socket.on('verification_message', () => {
    addToast('📄 Нове повідомлення щодо вашої заявки', 'message');
});

socket.on('new_verification_request', () => {
    if (user.role === 'admin') {

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		130

```

        setAdminUnreadCount(prev => prev + 1);
    }
});

    socketRef.current = socket;
} catch {}
};

if (user.role === 'volunteer') {
    fetch('/api/chat/me/unread', { credentials: 'include' })
        .then(r => r.json())
        .then(data => { if (typeof data === 'number') setUnreadChatCount(data);
    })
        .catch(() => {});
}

if (user.role === 'admin') initAdminCounts();

connect();
checkBadges();
badgeIntervalRef.current = setInterval(checkBadges, 10000);

return () => {
    socketRef.current?.disconnect();
    socketRef.current = null;
    setSocket(null);
    if (badgeIntervalRef.current) {
        clearInterval(badgeIntervalRef.current);
        badgeIntervalRef.current = null;
    }
}

```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А..				
Змн.	Арк.	№ докум.	Підпис	Дата		131

```
};  
}, [user]);  
  
return null;  
}
```

		Балан Д.О.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		132