НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Фандрайзингова платформа EFund (комплексна тема)

Виконав студент IV курсу, групи ІП-11
(шифр групи)
Веремчук Ігор Ігорович _____
(прізвище, ім'я, по батькові) (підпис)

Виконав студент IV курсу, групи ІП-11
(шифр групи)
Головатюк Владислав Ігорович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доц. каф.ІСТ, к.т.н., доц., Писаренко А. В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Київ – 2025

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Веремчуку Ігорю Ігоровичу та Головатюку Владиславу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Фандрайзингова платформа EFund (комплексна тема)
керівник проєкту Ліщук Катерина Ігорівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «23» травня 2025 р. №1705-с

2. Термін подання студентом проєкту «16» червня 2025 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.
4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань _____

2) Схема структурна компонентів програмного забезпечення _____

3) Схема бази даних _____

4) Креслення вигляду екранних форм _____

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «15» березня 2025 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв’язання задачі	25.03.2025	
3	Постановка та формалізація задачі	05.04.2025	
4	Розробка інформаційного забезпечення	15.04.2025	
5	Алгоритмізація задачі	20.04.2025	
6	Обґрунтування вибору використаних технічних засобів	25.04.2025	
7	Розробка програмного забезпечення	15.05.2025	
8	Налагодження програми	20.05.2025	
9	Виконання графічних документів	26.05.2025	
10	Оформлення пояснювальної записки	28.05.2025	
11	Подання ДП на попередній захист	04.06.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	16.06.2025	

Студент _____ (підпис) Ігор ВЕРЕМЧУК (ініціали, прізвище)

Студент _____ (підпис) Владислав ГОЛОВАТЮК (ініціали, прізвище)

Керівник _____ (підпис) Катерина ЛІЩУК (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 35 таблиць, 35 рисунків та 20 джерел – загалом 67 сторінок.

Дипломний проєкт присвячений розробці вебзастосунку для управління фандрейзинговими кампаніями.

Мета: створення вебзастосунку, котрий підвищить ефективність збору коштів шляхом створення платформи, яка дозволить користувачам оперативно створювати кампанії, моніторити надходження, публічно звітувати про використання ресурсів та мати систему модерації.

У першому розділі наведено аналіз предметної області сфери працевлаштування в ІТ, порівняльний аналіз між реалізованим програмним застосунком та аналогами, виділено переваги та недоліки перед кожним з них та наведено 4 основних бізнес-процеси.

У другому розділі наведено основні варіанти використання програмного забезпечення, функціональні та нефункціональні вимоги до програмного забезпечення та сформульовано системні вимоги до нього, наведено результати аналізу економічних показників ПЗ та наведено його приблизну собівартість.

У третьому розділі наведено архітектуру програмного забезпечення, наведено обґрунтування прийнятих основних архітектурних рішень та застосування використаних засобів розробки.

У четвертому розділі наведено опис контрольного прикладу.

У п’ятому розділі описано процес розгортання та супроводу програмного забезпечення.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, ЗБОРИ, ФАНДРЕЙЗИНГ, КАМПАНІЇ, .NET, REACT.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 35 tables, 35 figures and 20 sources – in total 67 pages.

The diploma project is dedicated to the development of a web application for managing fundraising campaigns.

The purpose of the diploma project is to create a web application that increases the efficiency of fundraising by providing a platform that allows users to quickly launch campaigns, monitor donations, publicly report on funds usage, and operate within a system of moderation.

The first section provides an analysis of the subject area, including a comparison between the implemented application and existing alternatives, outlining their respective advantages and disadvantages, and describes four core business processes.

The second section presents the main use cases of the software, the functional and non-functional requirements, and defines system-level requirements. It also includes an analysis of the software's economic indicators and an estimate of its development cost.

The third section describes the software architecture, provides justifications for the key architectural decisions made, and outlines the development tools used.

The fourth section contains a description of the test scenario.

The fifth section explains the deployment and maintenance process of the software.

KEYWORDS: WEB APPLICATION, FUNDRAISING, CAMPAIGNS, .NET, REACT.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-1103.ІП-1106.045440.01.91	Технічне завдання	18	
3	А4	КПІ.ІП-1103.ІП-1106.045440.02.81	Пояснювальна записка	67	
4	А4	КПІ.ІП-1103.ІП-1106.045440.05.34	Керівництво користувача	13	
5	А3	КПІ.ІП-1103.ІП-1106.045440.06.99	Графічний матеріал	4	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-1103.ІП-1106.045440.00.90		
Розроб.		Веремчук І. І.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
		Головатюк В.. І.					
Перевір.		Ліщук К.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-11	
Н.контр.		Ліщук К.І.					
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2025 р.

Фандрайзингова платформа EFund (комплексна тема)

Технічне завдання

КПІ.ІП-1103.ІП-1106.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Катерина ЛІЩУК

Нормоконтроль:

_____ Катерина ЛІЩУК

Виконавці:

_____ Ігор ВЕРЕМЧУК

_____ Владислав ГОЛОВАТЮК

Київ – 2025

Зміст

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Фандрайзингова платформа EFund (комплексна тема).

Галузь застосування:

Наведене технічне завдання поширюється на розробку вебзастосунку EFund, котрий використовується для збору коштів на потреби користувачів та призначена для волонтерів, благодійних організацій та стартаперів.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки EFund є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для організації фандрейзингових кампаній.

Метою розробки є підвищення ефективності зборів коштів шляхом створення платформи, яка дозволить користувачам оперативно створювати кампанії, моніторити надходження, публічно звітувати про використання ресурсів та мати систему модерації.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

– форма реєстрації користувача (рисунок 4.1);

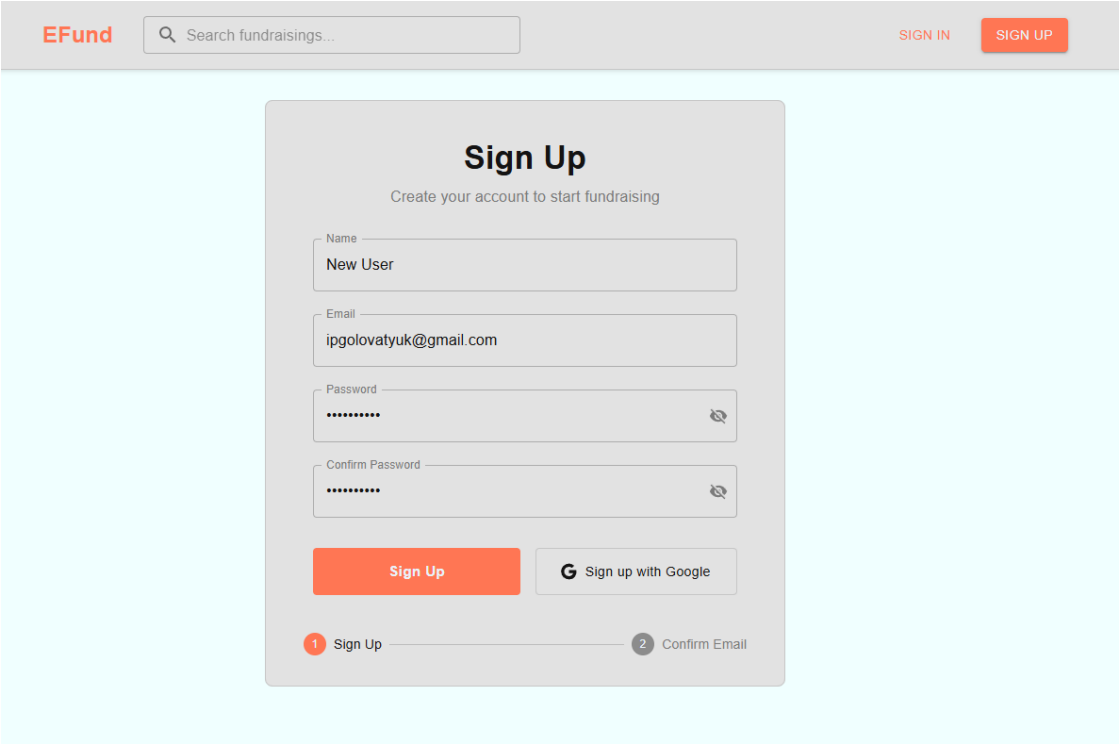


Рисунок 4.1 – Форма реєстрації користувача

– форма входу користувача (рисунок 4.2);

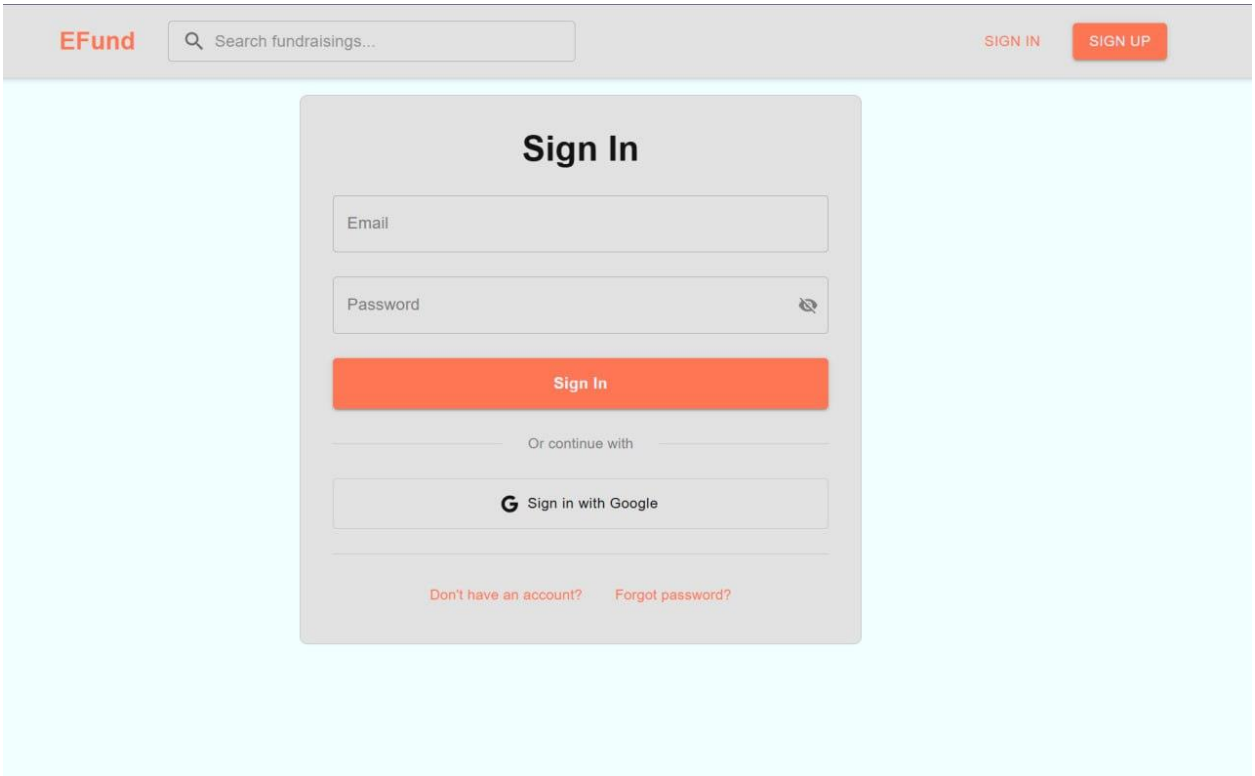


Рисунок 4.2 – Форма входу користувача

– сторінка профілю користувача (рисунок 4.3);

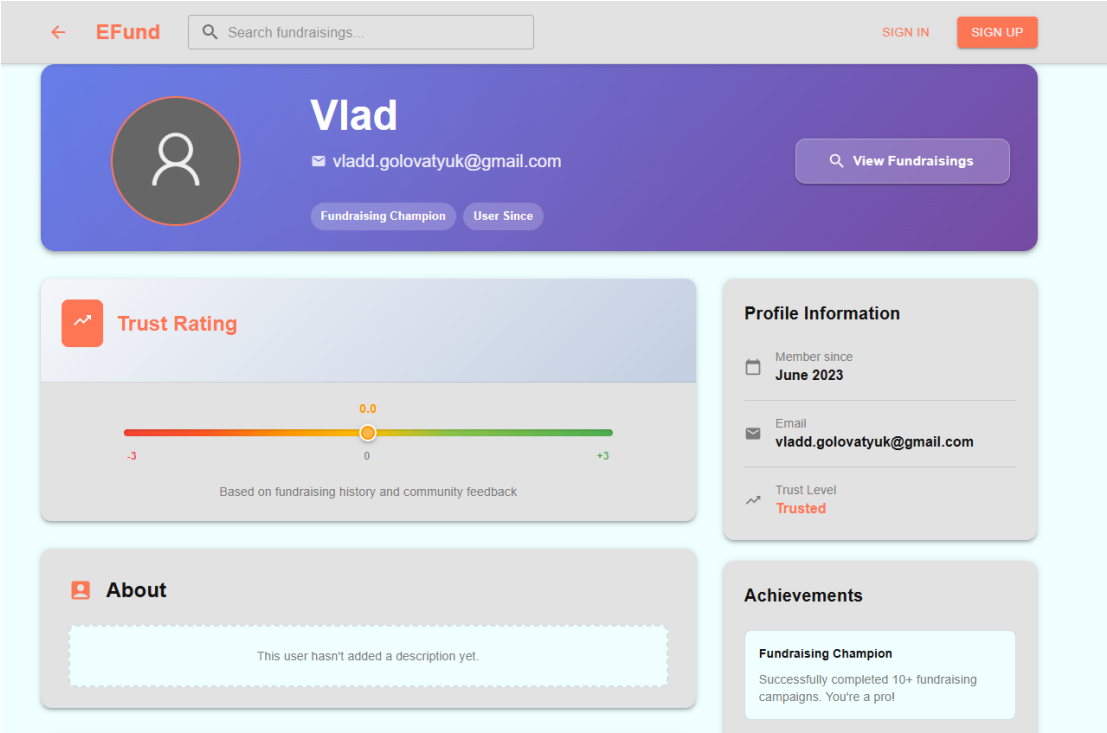


Рисунок 4.3 – Сторінка профілю користувача

– сторінка пошуку зборів (рисунок 4.4);

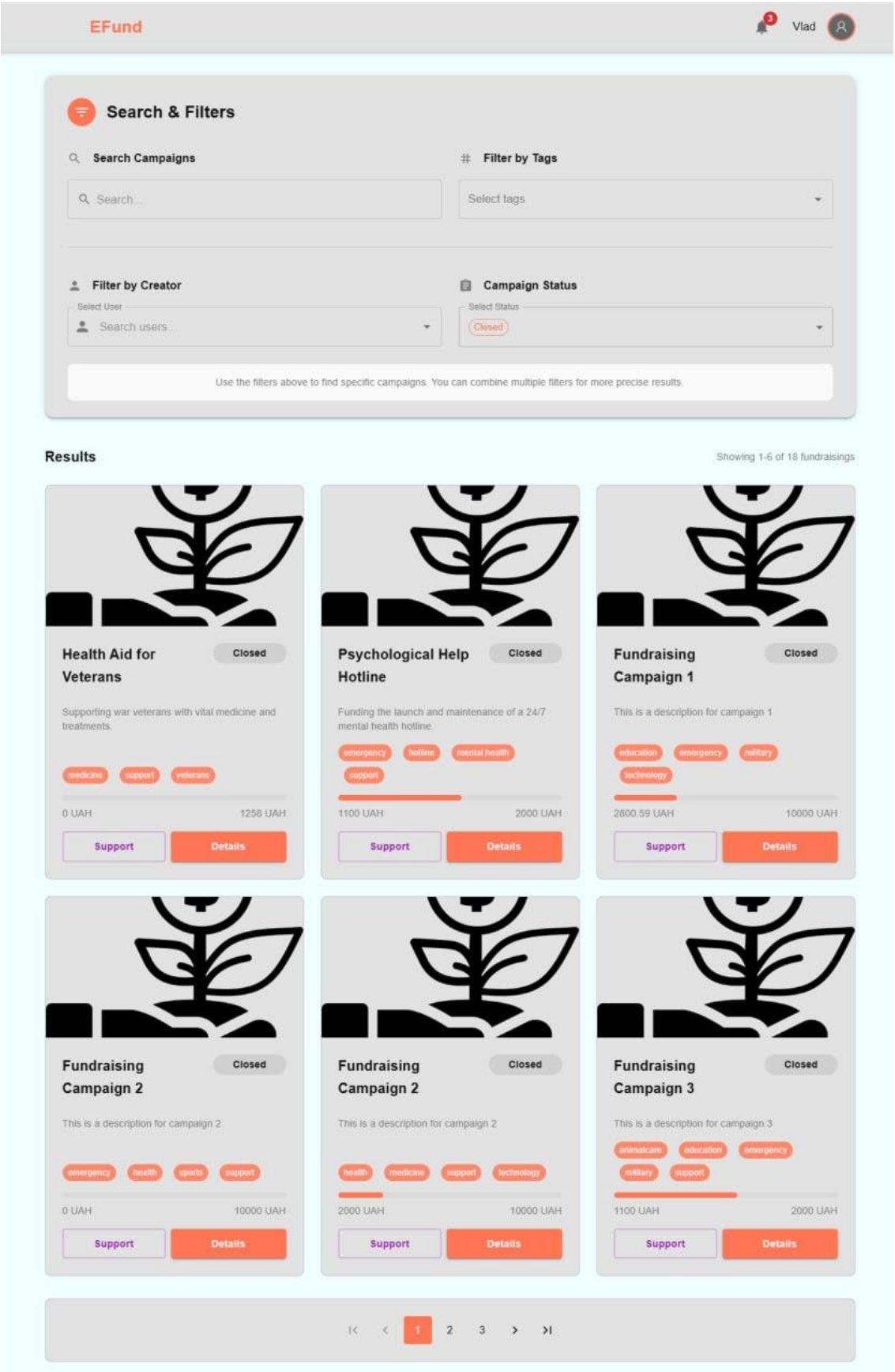


Рисунок 4.4 – Сторінка пошуку зборів

– сторінка деталей збору (рисунок 4.5);

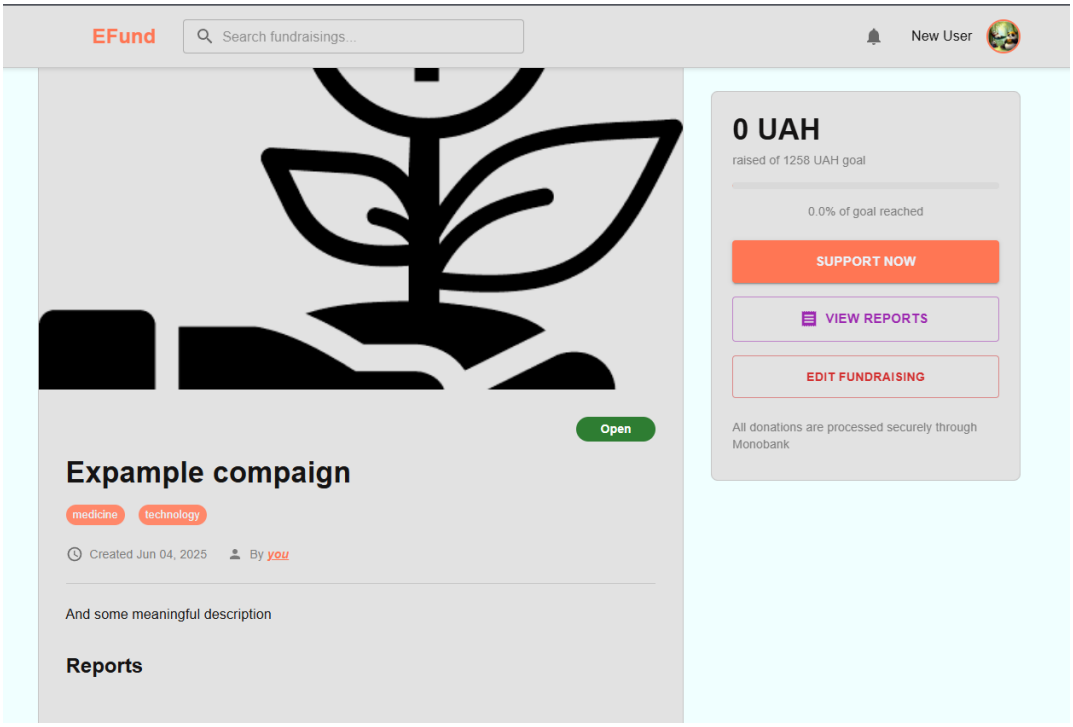


Рисунок 4.5 – Сторінка деталей збору

– форма створення збору (рисунок 4.6);

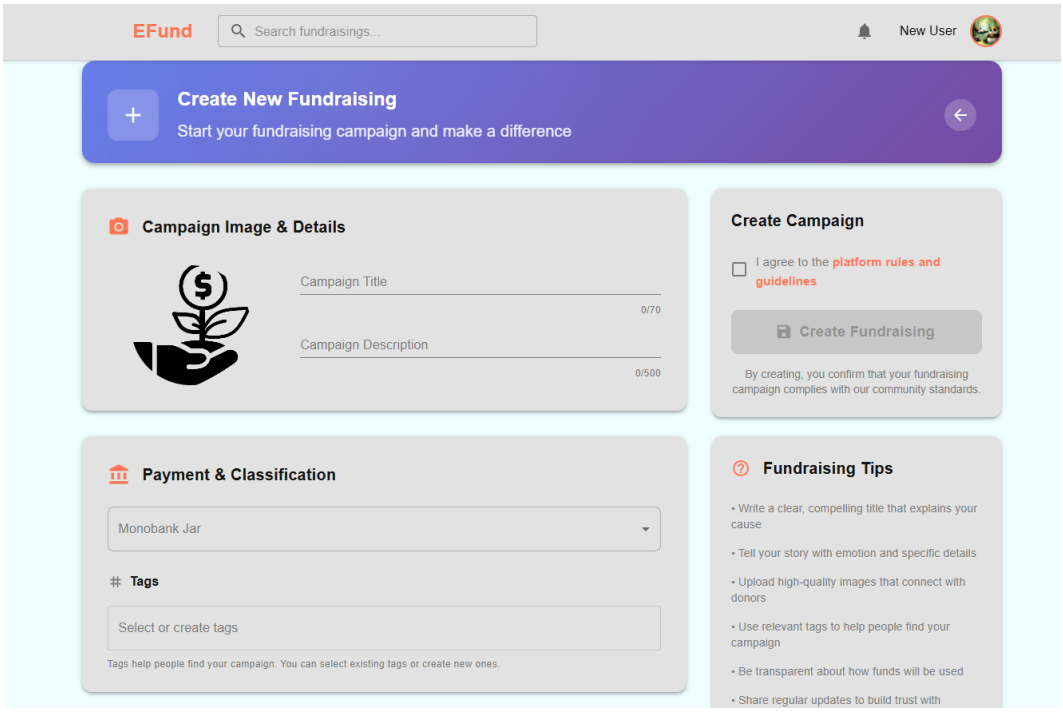


Рисунок 4.6 – Форма створення збору

– форма редагування збору (рисунок 4.7);

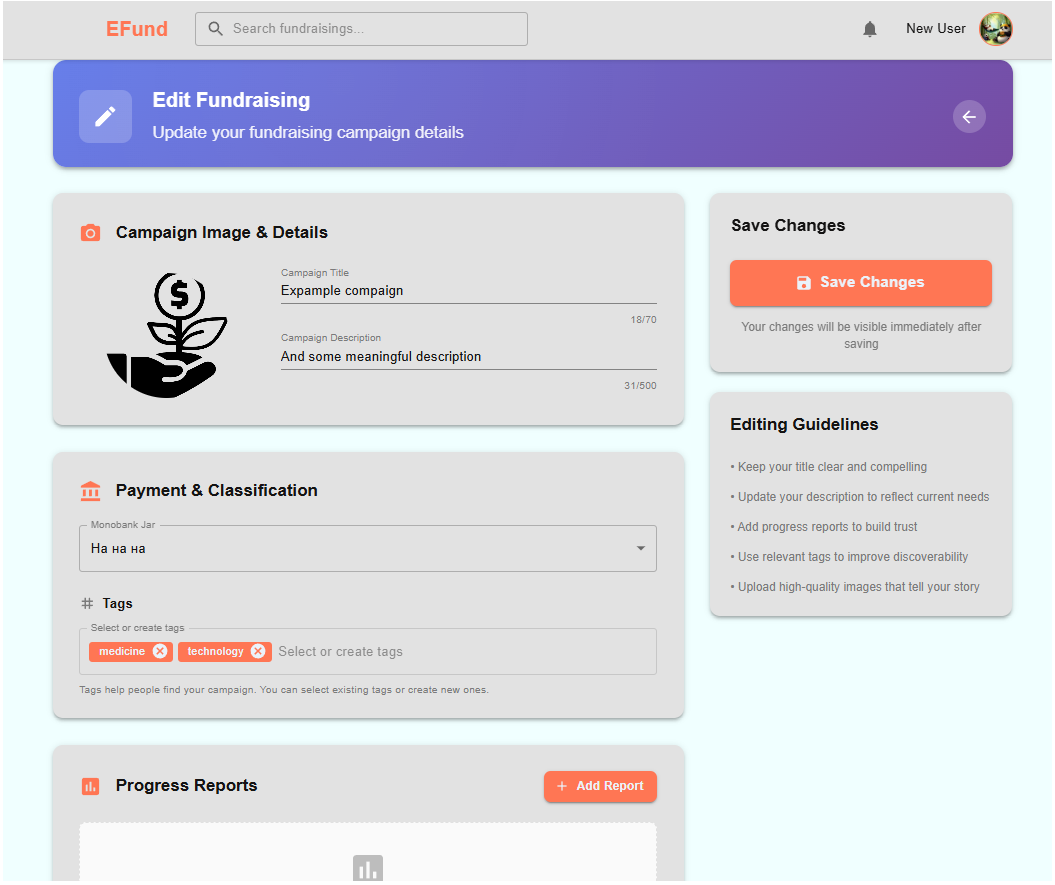


Рисунок 4.7 – Форма редагування збору

– сторінка пошуку користувачів (рисунок 4.8);

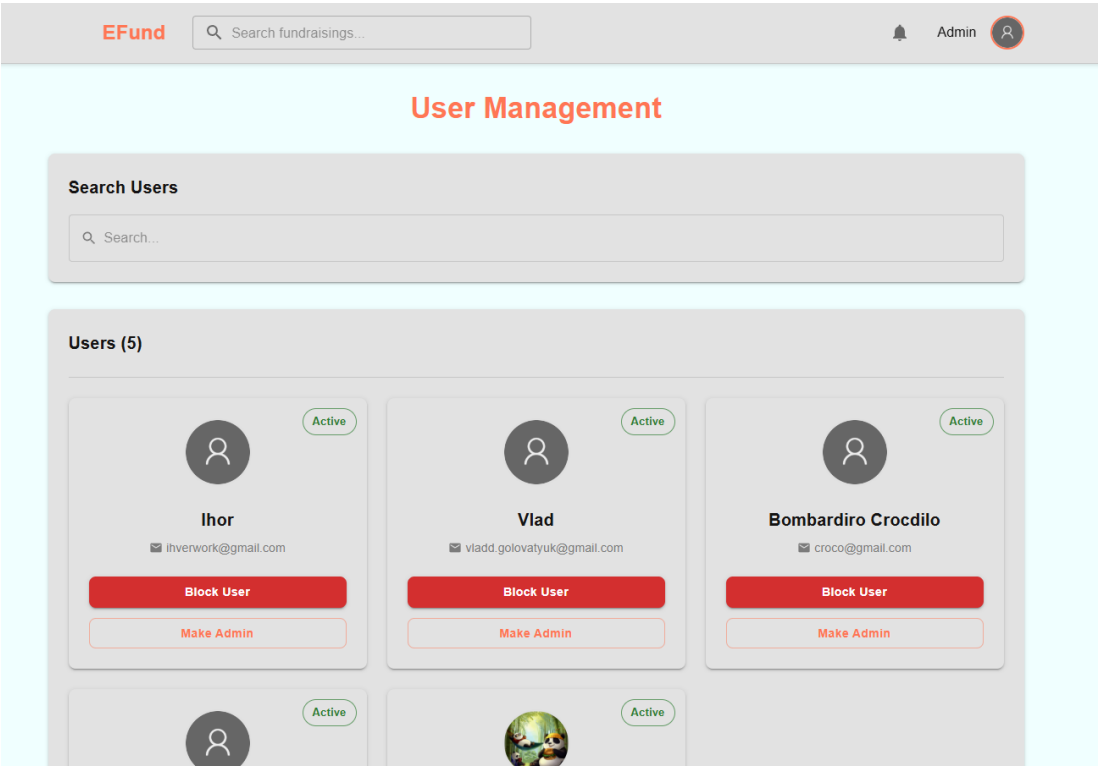


Рисунок 4.8 – Сторінка пошуку користувачів

– сторінка списку скарг (рисунок 4.9);

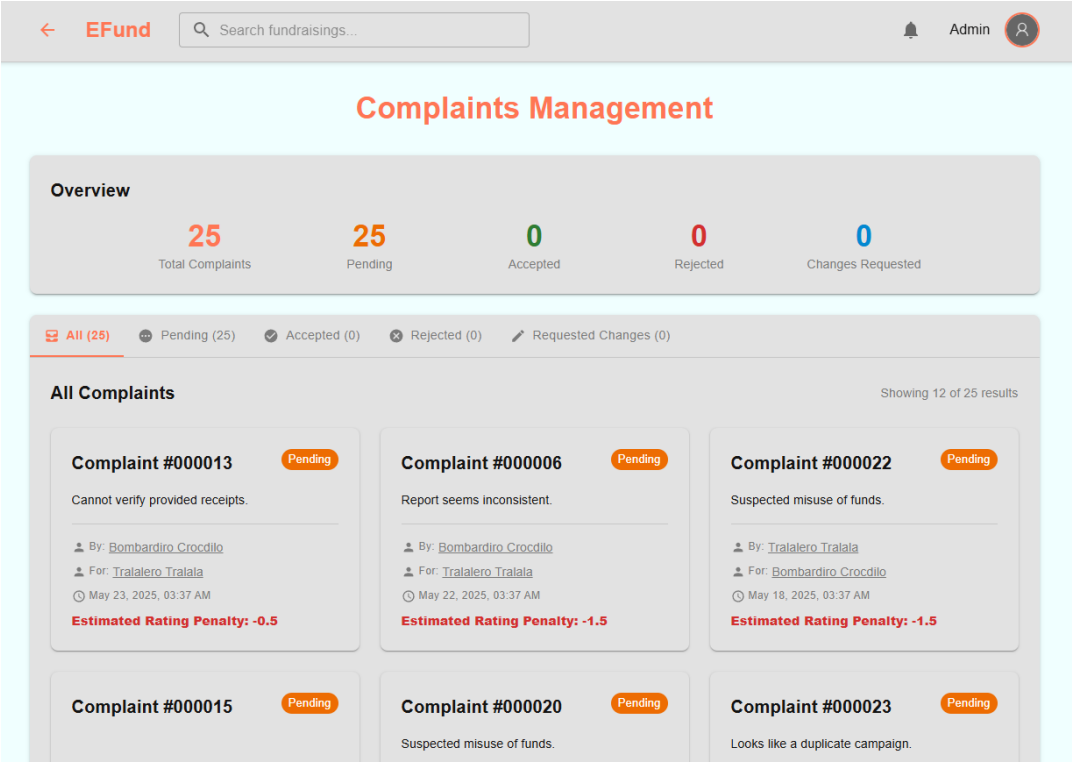


Рисунок 4.9 – Сторінка списку скарг

– сторінка деталей скарги та дій над нею (рисунок 4.10);

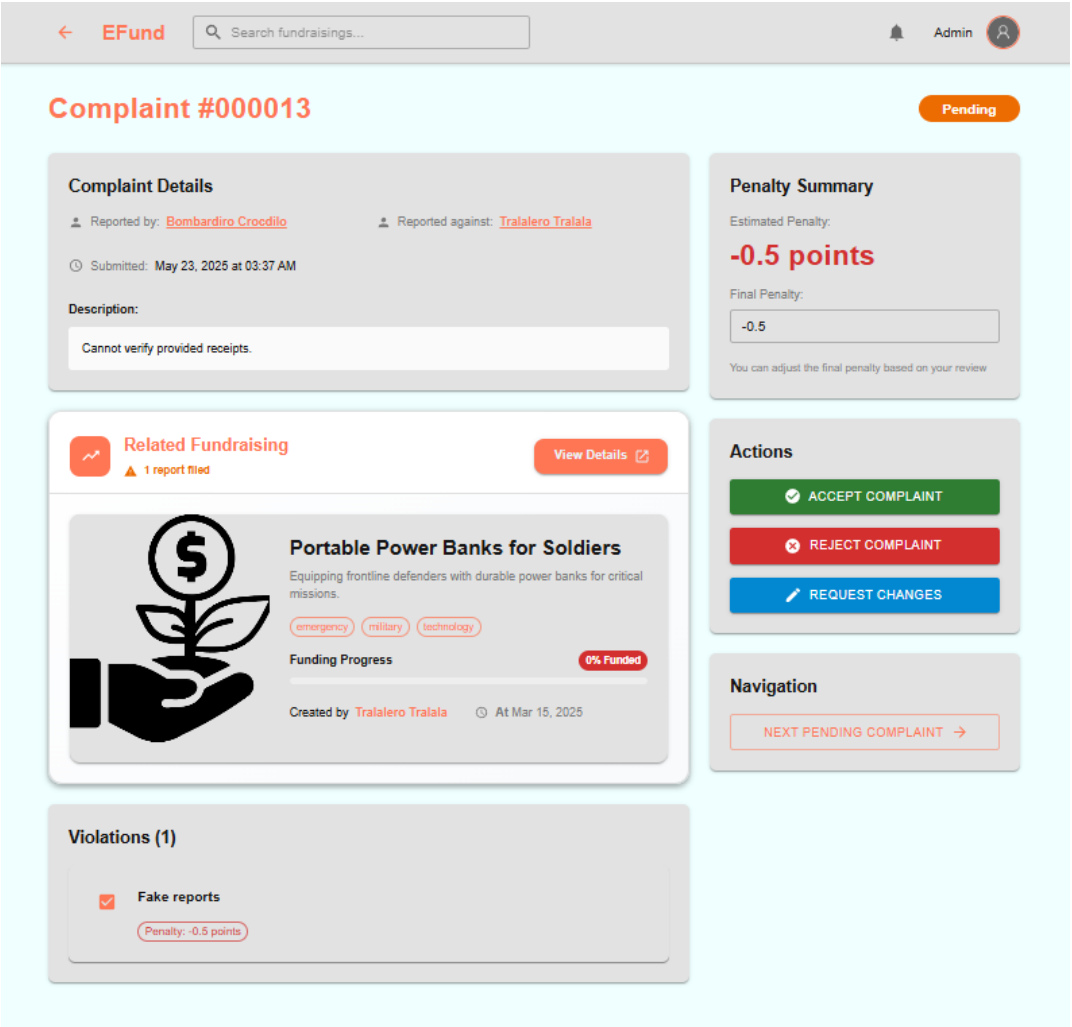


Рисунок 4.10 – Сторінка деталей скарги та дій над нею

4.1.2 Для користувача:

- реєстрація з використанням пошти та паролю;
- реєстрація через Google;
- вхід з використанням пошти та паролю;
- вхід через Google;
- відображення публічного профілю користувача з його аватаром, ім’ям, описом, рейтингом та досягненнями;
- рейтинг довіри користувача на основі його завершених зборів та скарг на нього;
- можливість редагувати профіль користувача, змінюючи ім’я, пароль, пошту, опис та аватар;
- пошук зборів за назвою, тегами, статусом та автором;
- створення зборів з назвою, описом тегами та автаром;
- можливість прив’язки банки monobank до збору;
- редагування даних збору;
- відображення прогресу збору, актуальної накопиченої та цільової суми;
- переадресування на банку monobank для надсилання коштів;
- можливість додавання звітів по збору з назвою, описом та прикріпленими файлами;
- можливість редагування та видалення звітів;
- можливість скаржитися на збори, з вибором порушення та описом;
- можливість закривати збори по досягненню запланованої суми;
- можливість передавати збори на перевірку адміністратору після завершення реалізації коштів;
- відображення системних сповіщень про оновлення збору та дії адміністрації.

4.1.3 Для адміністратора системи:

- пошук користувачів у системі;
- можливість блокувати та розблоковувати користувачів;
- можливість приховувати та видаляти збори;
- можливість перевіряти завершені збори та змінювати рейтинг користувача за результатами перевірки;
- можливість перегляду списку скарг з фільтрацією по статусу;
- можливість відхиляти скарги;
- можливість приймати скарги та знижувати рейтинг користувача на визначене значення;
- можливість надсилання запиту на зміни по збору до користувача з надсиланням системного сповіщення та імейлу;
- можливість відновлювати приховані збори;
- можливість надавати права адміністратора користувачам;
- можливість запрошувати нових адміністраторів через пошту.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних. Забезпечити належну безперебійність.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Адміністрування користувачів та фандрейзингових кампаній

4.3.2 Обслуговуючий персонал

Обслуговуючий персонал повинен мати базові навички роботи з комп'ютером. Зокрема, адміністратор повинен вміти працювати з веббраузером, вміти користуватися пошуком та фільтрацією.

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i3;
– об‘єм ОЗП: 2 Гб;
– підключення до мережі Інтернет зі швидкістю від 5 Мбіт/с;

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;
– об‘єм ОЗП: 4 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 Мбіт/с;

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням усіх операційних систем, які забезпечують з'єднання з Інтернетом і можуть надати доступ до веб-застосунку через браузер.

4.5.1 Вимоги до вхідних даних

Вхідні дані надаються користувачами за допомогою інтерфейсу

4.5.2 Вимоги до вихідних даних

Результат повертається в графічному форматі на інтерфейсі

4.5.3 Вимоги до мови розробки

Розробку виконати на мовах програмування C#, TypeScript

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформах Rider, VS Code

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми серверної частини додатку має бути представлений у вигляді гітхаб репозиторію за посиланням https://github.com/vrmchk/EFund

Вихідний код програми клієнтської частини додатку має бути представлений у вигляді гітхаб репозиторію за посиланням https://github.com/aquaprogit/EFund

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема структурна компонентів програмного забезпечення;
– схема бази даних;
– архітектура програмного забезпечення;
– креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1	Вивчення рекомендованої літератури	15.03.2025	
2	Аналіз існуючих методів розв’язання задачі	25.03.2025	Технічне завдання
3	Постановка та формалізація задачі	05.04.2025	Специфікації програмного забезпечення
4	Розробка інформаційного забезпечення	15.04.2025	
5	Алгоритмізація задачі	20.04.2025	Формалізовані алгоритми програмного забезпечення
6	Обґрунтування вибору використаних технічних засобів	25.04.2025	
7	Розробка програмного забезпечення	15.05.2025	Тексти програмного забезпечення
8	Налагодження програми	20.05.2025	Тести, результати тестування
9	Виконання графічних документів	26.05.2025	Графічний матеріал проєкту
10	Оформлення пояснювальної записки	28.05.2025	Пояснювальна записка
11	Подання ДП на попередній захист	04.06.2025	
12	Подання ДП рецензенту	10.06.2025	
13	Подання ДП на основний захист	16.06.2025	

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Фандрайзингова платформа EFund (комплексна тема)

КПІ.ІП-1103.ІП-1106.045440.02.81

Київ – 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс.
SDK	–	Software development kit.
IT	–	Інформаційні технології.
ER	–	Entity-Relation diagram.
ОС	–	Операційна система.
БД	–	База даних.

ВСТУП

У сучасних умовах цифровізації суспільства та розвитку інформаційних технологій особливої актуальності набуває створення інструментів, що сприяють ефективній взаємодії між громадянами, волонтерами та благодійними організаціями. В умовах повномасштабної війни в Україні потреба у прозорих, надійних та швидких механізмах збору коштів на гуманітарні, військові та соціальні потреби зросла до безпрецедентного рівня. Незважаючи на активність волонтерських спільнот, значна частина фандрейзингових кампаній досі проводиться вручну, через особисті банківські рахунки, що ускладнює процес перевірки достовірності та знижує рівень довіри серед потенційних донорів.

Світові тенденції демонструють стрімке зростання популярності цифрових фандрейзингових платформ, зокрема в таких сферах як благодійність, медицина, освіта, культура та військова підтримка. Платформи на зразок GoFundMe та Kickstarter вже довели свою ефективність у забезпеченні масової фінансової підтримки ініціатив з боку широкої громадськості. Водночас в Україні спостерігається дефіцит локалізованих рішень, що адаптовані до реалій війни, підтримують інтеграцію з популярними платіжними сервісами, такими як monobank, і забезпечують високий рівень прозорості та підзвітності.

В рамках даної дипломної роботи буде розроблено вебзастосунок — фандрейзингова платформа EFund, що об'єднає волонтерів і донорів у єдиній екосистемі для прозорої та ефективної взаємодії. Платформа дозволить створювати, поширювати та супроводжувати кампанії зі збору коштів, інтегрувати їх із банківськими сервісами, зокрема через API monobank, а також вести детальне звітування з можливістю додавання фото, відео та документів.

Призначенням даного проєкту є створення універсального цифрового інструменту для організації фандрейзингових кампаній, який забезпечить простоту, зручність, прозорість та надійність збору коштів на волонтерські потреби.

Метою роботи є підвищення ефективності зборів шляхом створення платформи, яка дозволить користувачам оперативно створювати кампанії, моніторити надходження та публічно звітувати про використання ресурсів.

Сфера застосування розробленого рішення охоплює як індивідуальні волонтерські ініціативи, так і діяльність організованих громадських, благодійних та гуманітарних організацій. У перспективі платформа може бути масштабована для підтримки національних кампаній та міжнародного співробітництва.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання дипломного проєктування

Дипломне проєктування передбачає виконання наступних завдань:

– аналіз предметної області та опис її ключових бізнес-процесів, визначення загального завдання розробки у рамках ДП;

– аналіз існуючих рішень (як програмних продуктів, так і алгоритмічних чи технічних рішень) обраного завдання розробки у рамках ДП;

– аналіз та моделювання бізнес-процесів;

– розроблення функціональних, нефункціональних та системних вимог до програмного забезпечення;

– аналіз економічних показників програмного забезпечення ДП;

– постановка завдання на розробку програмного забезпечення ДП;

– розроблення архітектури програмного забезпечення;

– розроблення архітектурних рішень та обґрунтування вибору засобів розробки програмного забезпечення;

– конструювання та розроблення програмного забезпечення;

– аналіз безпеки даних програмного забезпечення;

– аналіз якості та тестування програмного забезпечення;

– розгортання та супровід програмного забезпечення;

– створення супроводжувальної документації до розробленого програмного забезпечення.

1.2 Аналіз предметної області

Фандрейзингові платформи — це цифрові сервіси, призначені для організації збору коштів з метою підтримки соціальних, благодійних, медичних, освітніх або інших ініціатив. Вони є важливим інструментом сучасного суспільства, який забезпечує прозору, швидку та ефективну взаємодію між донорами та ініціаторами зборів. Завдяки стрімкому розвитку

цифрових технологій, фандрейзинг із традиційного офлайн-формату перейшов у площину вебдодатків та мобільних сервісів [1].

Сучасні вебдодатки для фандрейзингу представляють собою складні багаторівневі системи. Вони складаються з інтерфейсу користувача, бекенд-сервера, системи збереження даних, інтеграції з платіжними сервісами, модулів авторизації, звітності. Вони також часто включають механізми адміністрування та модерації зборів та користувачів, аналітики в режимі реального часу, фільтрації та категоризації кампаній. У контексті війни в Україні такі системи набули особливої актуальності для підтримки військових, гуманітарних ініціатив та волонтерських організацій [2].

Основні переваги фандрейзингових додатків:

− централізація зборів — єдина платформа для управління пожертвами, звітністю, користувачами;

− доступність 24/7 з будь-якого пристрою — критично важливо в умовах війни;

− швидкість та простота переказів — мінімальна кількість кроків для здійснення пожертви;

− можливість поширення кампаній через соціальні мережі, месенджери тощо;

Незважаючи на всі вищезгадані переваги та весь розвиток і прогрес у даній сфері, сучасні рішення мають певні проблеми:

− відсутність зручних локалізованих рішень, адаптованих до умов війни в Україні;

− обмежені можливості звітності по кампаніях. У багатьох рішеннях реалізована лише базова текстова звітність без вбудованої підтримки фото-, відео- та інших доказових матеріалів;

− нестача підтримки для волонтерських ініціатив з низьким технічним досвідом. Процес створення кампаній зазвичай є дуже складним, та включає багато складних кроків. Існуючі інтерфейси часто орієнтовані на досвідчених користувачів або великі організації;

− складність обслуговування багатьох кампаній одночасно. Відсутність дашборду для адміністраторів та волонтерів з можливістю масового керування кампаніями, комунікацією, пожертвами [3].

Можливі шляхи покращення ситуації:

− реалізація інтеграції з monobank API [4] з можливістю прив’язки "банок" безпосередньо до кампаній та автоматичної синхронізації суми збору;

− реалізація модуля звітів з підтримкою завантаження фото, відео, документів, які автоматично публікуються на сторінці кампанії;

− спрощення алгоритму створення кампаній, з наданням зручного та зрозумілого інтерфейсу зі зрозумілими підказками;

− реалізація централізованої адмін-панелі з групуваннями, фільтрами, груповими діями.

1.3 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації фандрейзингової платформи EFund. Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.3.1 Аналіз відомих програмних продуктів

Kickstarter [5] — це краудфандингова платформа, орієнтована переважно на творчі та стартап-проєкти. Користувачі можуть створювати кампанії з описом, відео, цілями збору коштів і винагородами для донорів. Платформа має строгі правила щодо затвердження кампаній, але не підтримує благодійні ініціативи. Доступна як у вебверсії, так і для мобільних пристроїв. Основний функціонал передбачає створення кампаній, перегляд статистики зборів та інтеграцію з платіжними системами (рисунки 1.1 – 1.2).

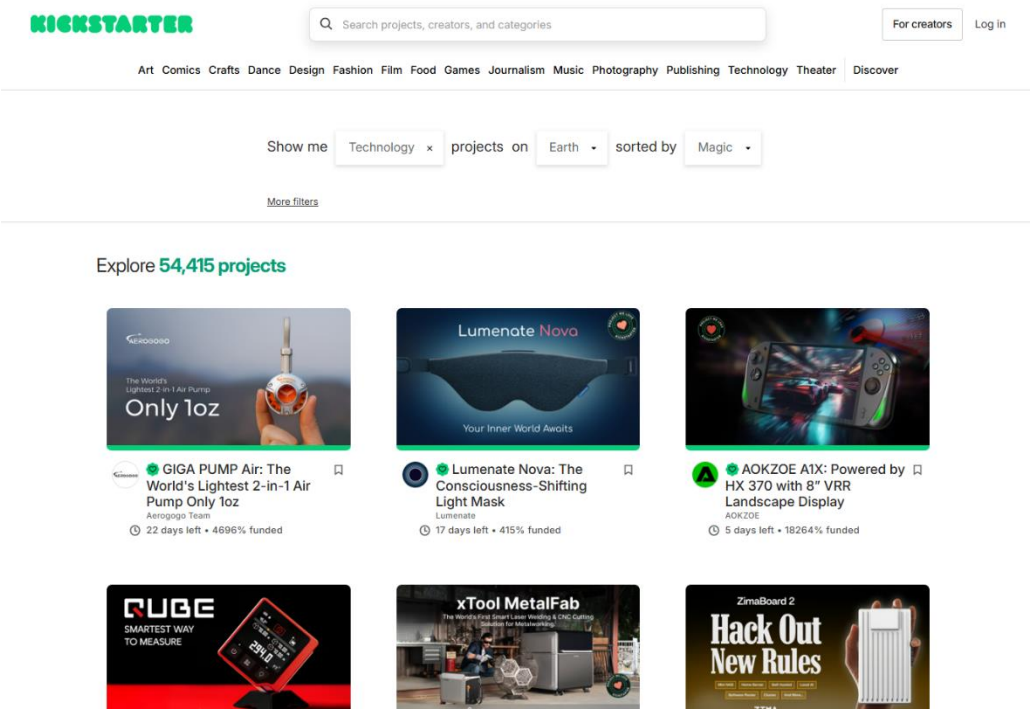


Рисунок 1.1 – Інтерфейс застосунку Kickstarter

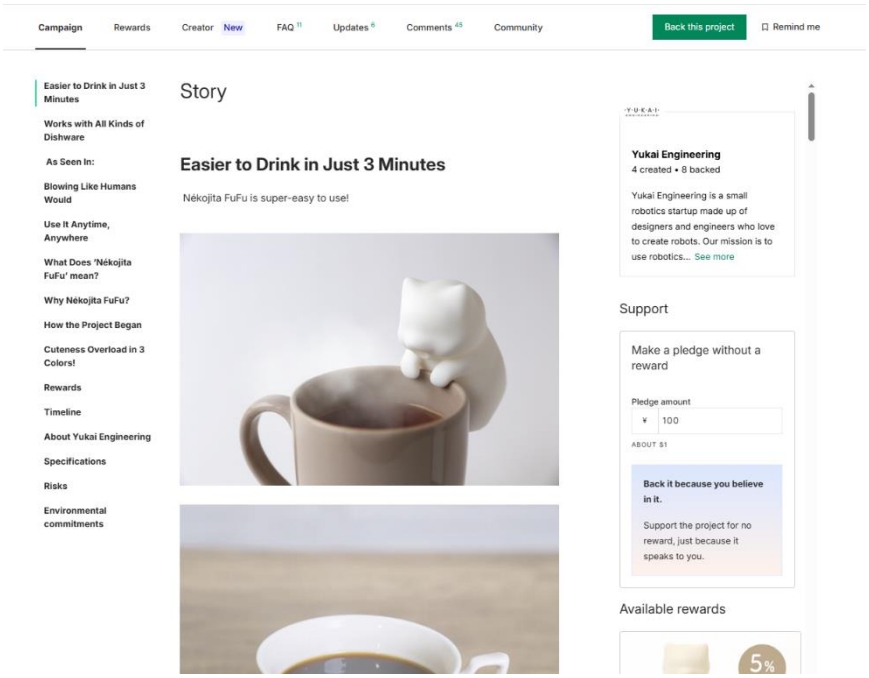


Рисунок 1.2 – Інтерфейс кампанії в застосунку Kickstarter

GoFundMe [6] — одна з найпопулярніших платформ для збору коштів на соціальні, медичні та благодійні потреби. Користувачі можуть створювати персональні кампанії, ділитися ними через соціальні мережі та отримувати пожертви напряму на рахунок. Сервіс підтримує функції звітності, завантаження фото та оновлень. GoFundMe активно використовується в США

та інших англомовних країнах, але має обмежену локалізацію для України (рисунки 1.3 – 1.4).

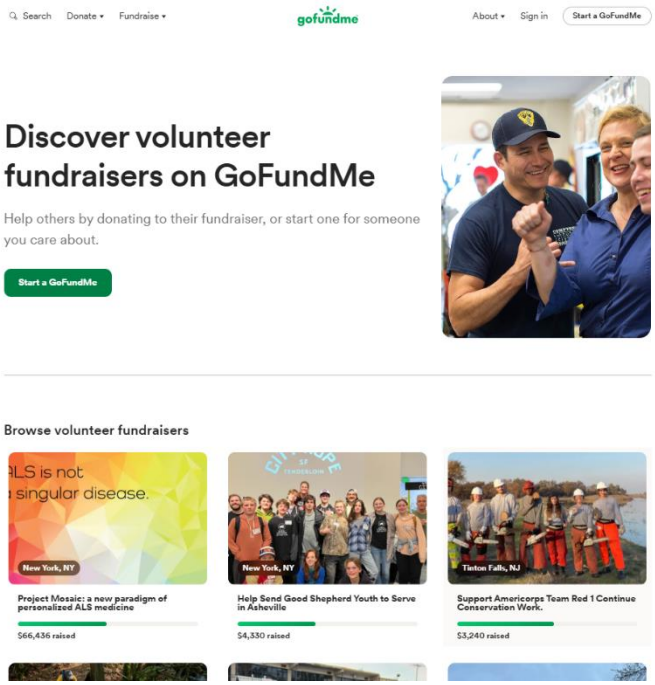


Рисунок 1.3 – Інтерфейс застосунку GoFundMe

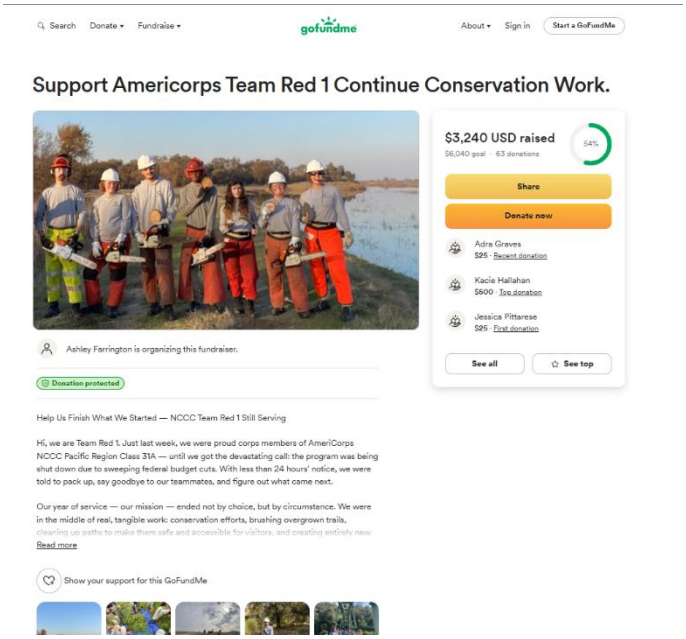


Рисунок 1.4 – Інтерфейс кампанії в застосунку GoFundMe

Fundly [7] — онлайн-платформа для створення благодійних кампаній, яка орієнтована як на окремих користувачів, так і на некомерційні організації. Fundly який надає можливість відстежити цільові суми, статус зборів та підтрмуя інтеграції з Facebook. Платформа дозволяє збирати кошти без потреби в офіційній реєстрації організації, однак має комісію за транзакції.

Додаток доступний у вебверсії, мобільного застосунку не передбачено (рисунки 1.5 – 1.6).

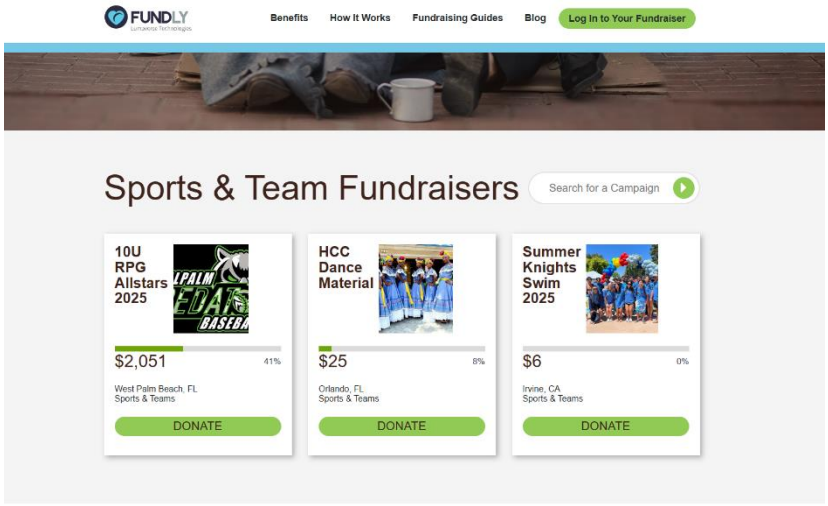


Рисунок 1.5 – Інтерфейс застосунку Fundly

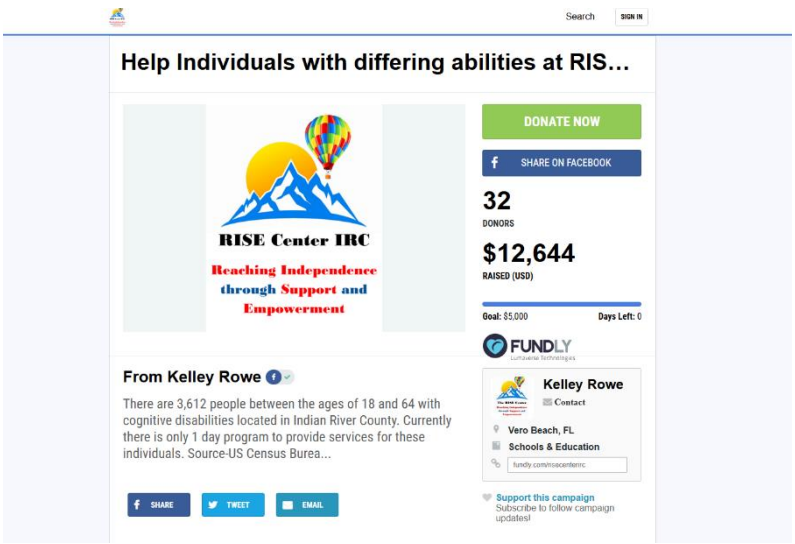


Рисунок 1.6 – Інтерфейс кампанії в застосунку Fundly

Банка від monobank [8] — популярний інструмент збору коштів в Україні, що дозволяє створювати публічні або приватні "банки" з унікальним посиланням. Сервіс підтримує мобільні платежі, миттєві перекази та відображення суми збору в реальному часі. Основний недолік — відсутність централізованої платформи для зборів, кампаній і звітності. Проте інтеграція з цією системою може бути критично важливою для локального рішення (рисунки 1.7 – 1.8).

Рисунок 1.7 – Інтерфейс сторінки зборів застосунку monobank

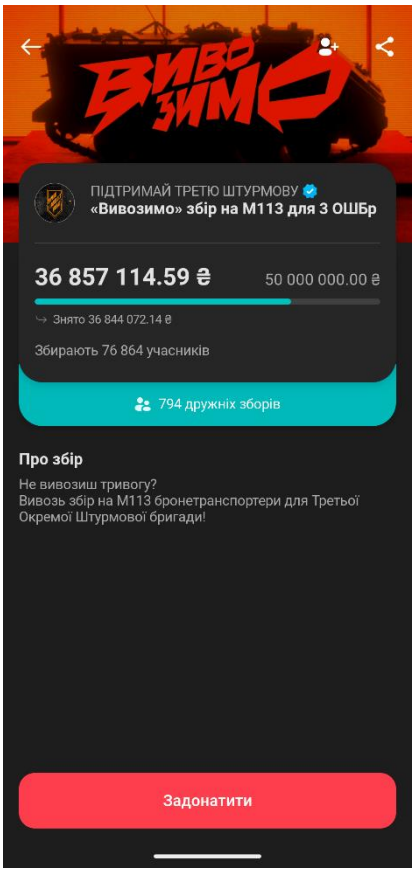


Рисунок 1.8 – Інтерфейс кампанії в застосунку monobank

Для порівняння проєкту з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогами

Функціонал	Фандрейзингова платформа EFund	Аналог для порівняння 1 (Kickstarter)	Аналог для порівняння 2 (GoFundMe)	Аналог для порівняння 3 (Fundly)	Аналог для порівняння 4 (monobank	Пояснення
Благодійні ініціативи	+	-	+	+	+	Можливість відкриття кампанійна благодійні та гуманітарні потреби
Військові потреби	+	-	-	-	+	Можливість відкриття кампанійна військові потреби

Продовження таблиці 1.1

Пошук зборів по категоріях	+	+	+	+	-	Наявність категорій, що спрощують пошук для донорів
Можливість додавання звітів	+	+	+	-	-	Можливість показувати донорам результати кампаній та реалізацію їх коштів

Продовження таблиці 1.1

Можливість прикріпляти файли до звітів	+	+	-	-	-	Можливість прикріпляти фото, відео, таблиці та текстові файли

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

Під час проєктування програмного забезпечення було здійснено аналіз сучасних архітектурних та технічних підходів до побудови вебзастосунків. Метою цього аналізу було обрати таку архітектуру, яка забезпечить ефективну розробку, прозорість, гнучкість і подальше масштабування системи за умов обмежених ресурсів команди розробників.

У результаті аналізу було обрано клієнт-сервісну архітектуру, яка передбачає чіткий розподіл між інтерфейсом користувача та серверною логікою. Такий підхід дозволяє розробляти й обслуговувати кожну частину застосунку незалежно: користувацький інтерфейс працює автономно, надсилаючи запити до серверної частини через відкритий програмний інтерфейс. Це не тільки спрощує обслуговування системи, а й відкриває можливості для використання одного серверного ядра різними клієнтськими додатками — наприклад, мобільною або десктопною версією.

Клієнт-сервісна модель має низку переваг над альтернативами, зокрема над мікросервісною архітектурою. Остання хоч і забезпечує високу модульність та масштабованість, але вимагає значно складнішої інфраструктури: оркестрації, розподіленого моніторингу, логування,

маршрутизації тощо. У випадку невеликої команди це створює зайве навантаження, що негативно впливає на швидкість і стабільність розробки.

Натомість клієнт-сервісний підхід забезпечує просте та ефективне розгортання, централізовану обробку логіки на сервері, можливість масштабування по окремих компонентах (наприклад, лише інтерфейс чи лише API), зменшує дублювання логіки та моделей даних, а також дозволяє централізовано реалізовувати механізми авторизації, звітності та безпеки.

Одним із ключових технічних рішень, реалізованих у межах цієї архітектури, є інтеграція із зовнішніми банківськими сервісами. Така інтеграція дозволяє користувачам прив’язувати до кампаній свої рахунки в онлайн-банкінгу. Система виконує періодичні запити до банківського API, автоматично оновлюючи інформацію про стан рахунку. Це забезпечує максимальну прозорість для донорів і значно зменшує навантаження на організаторів зборів, адже всі звіти оновлюються автоматично.

Таким чином, обрана архітектура дозволяє зберегти баланс між простотою обслуговування, продуктивністю та функціональністю. Вона забезпечує необхідний рівень гнучкості для подальшого розвитку системи без потреби в дорогих чи складних інфраструктурних рішеннях.

1.4 Аналіз та моделювання бізнес-процесів

Для моделювання бізнес-процесів використовується BPMN модель.

Опис моделі бізнес-процесу реєстрації користувача (рисунок 1.9):

- незареєстрований користувач відкриває сторінку реєстрації;
- у відповідних полях вводить імейл, пароль, підтвердження пароля;
- якщо дані некоректні (не відповідають вимогам формату), система відображає повідомлення про помилку;
- у випадку успішної валідації дані зберігаються у базі даних, і система надсилає код підтвердження на вказану електронну пошту;
- користувач переходить на сторінку введення отриманого коду;

- якщо код введено невірно — знову виводиться повідомлення про помилку;
- якщо код підтвердження коректний — користувач авторизується та завершує процес реєстрації.

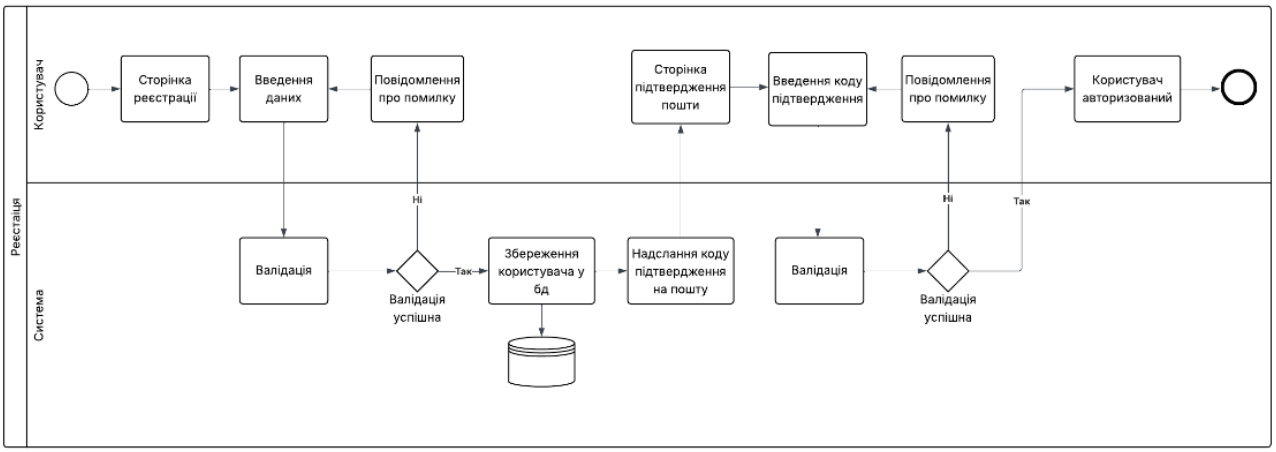


Рисунок 1.9 – Схема бізнес-процесу реєстрації користувача

Опис моделі бізнес-процесу входу в обліковий запис користувача (рисунок 1.10):

- користувач переходить на сторінку входу;
- у відповідних полях вводить імейл та пароль;
- якщо дані неправильні або неповні, система виводить повідомлення про помилку;
- якщо валідація успішна — користувач авторизується в системі та отримує доступ до функціональності.

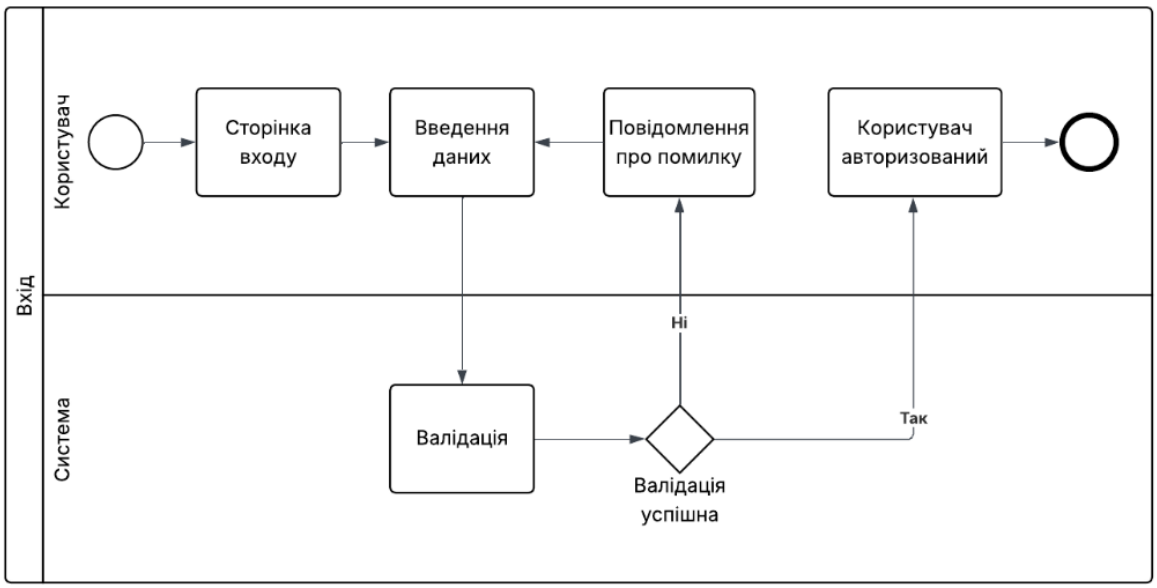


Рисунок 1.10 – Схема бізнес-процесу входу користувача

Опис моделі бізнес-процесу пошуку зборів (рисунок 1.11):

- користувач переходить на сторінку перегляду зборів;
- система завантажує першу сторінку всіх активних зборів з використанням пагінації;
- користувач може переглядати список або скористатись фільтрами категорії або назви для вибірки зборів;
- після фільтрації система завантажує відповідний набір зборів, який виводиться на екран.

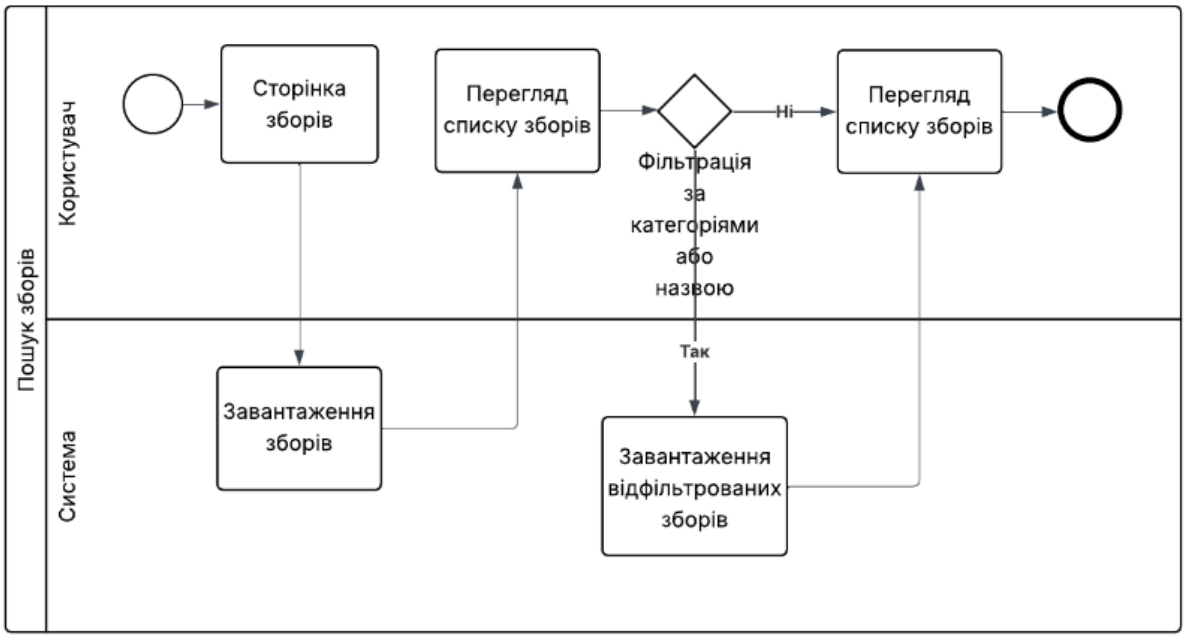


Рисунок 1.11 – Схема бізнес-процесу пошуку зборів

Опис моделі бізнес-процесу створення збору (рисунок 1.12):

- користувач відкриває сторінку створення нового збору;
- вводить усі необхідні дані: назву збору, цільову суму, обирає банку з monobank, яка буде прив’язана до збору;
- якщо якісь поля заповнено некоректно — з’являється повідомлення про помилку;
- у разі правильно введених даних, вони проходять валідацію та зберігаються у базі даних;
- після успішного створення збору користувача перенаправляють на сторінку перегляду зборів.

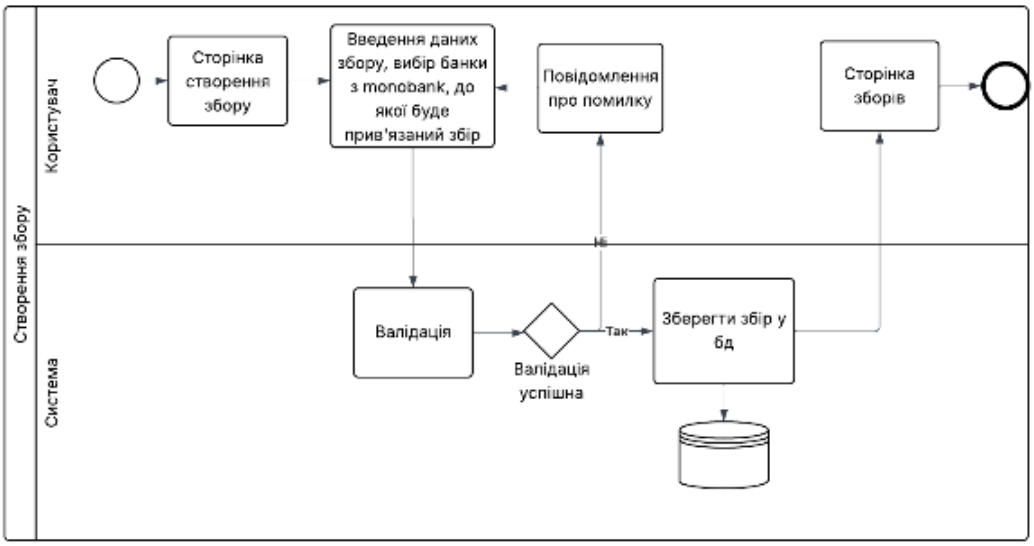


Рисунок 1.12 – Схема бізнес-процесу створення збору

Висновки до розділу

У даному розділі було проведено всебічний аналіз предметної області програмного забезпечення, що використовується у сфері благодійної діяльності та збору коштів. Було визначено ключову мету розробки – створення вебзастосунку, який дозволяє користувачам ініціювати, переглядати та адмініструвати збори коштів, з можливістю інтеграції з банківською системою monobank.

На початку розділу було описано організаційно-економічну характеристику проєкту, включаючи актуальність розробки, соціальну

важливість та потенційний вплив платформи на ефективність волонтерської діяльності. Далі проведено аналіз основних функцій, які має реалізовувати система: автентифікацію користувачів, створення та перегляд зборів, перевірку стану рахунків через зовнішній API.

Важливою частиною розділу стало моделювання ключових бізнес-процесів, які реалізуються у програмному забезпеченні. Було створено та описано BPMN-діаграми для чотирьох процесів: реєстрації користувача, авторизації, створення збору та перегляду списку зборів. У кожному з цих процесів визначено ролі учасників, логіку перевірки даних, взаємодію із системою, обробку помилок та успішні сценарії завершення.

Виконане моделювання дозволило наочно представити логіку роботи системи, що є основою для подальшої реалізації архітектури вебзастосунку, вибору технологічного стеку та побудови інтерфейсів. Усі процеси були описані з урахуванням вимог до зручності використання, безпеки, масштабованості та інтеграції з зовнішніми сервісами.

У результаті, розділ забезпечив концептуальне та функціональне обґрунтування майбутньої інформаційної системи, що стало фундаментом для наступного етапу – безпосередньої розробки програмного забезпечення.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головною функцією програмного забезпечення є створення зборів для накопичення коштів на потреби користувачів, більше функцій можна побачити на рисунку 2.1

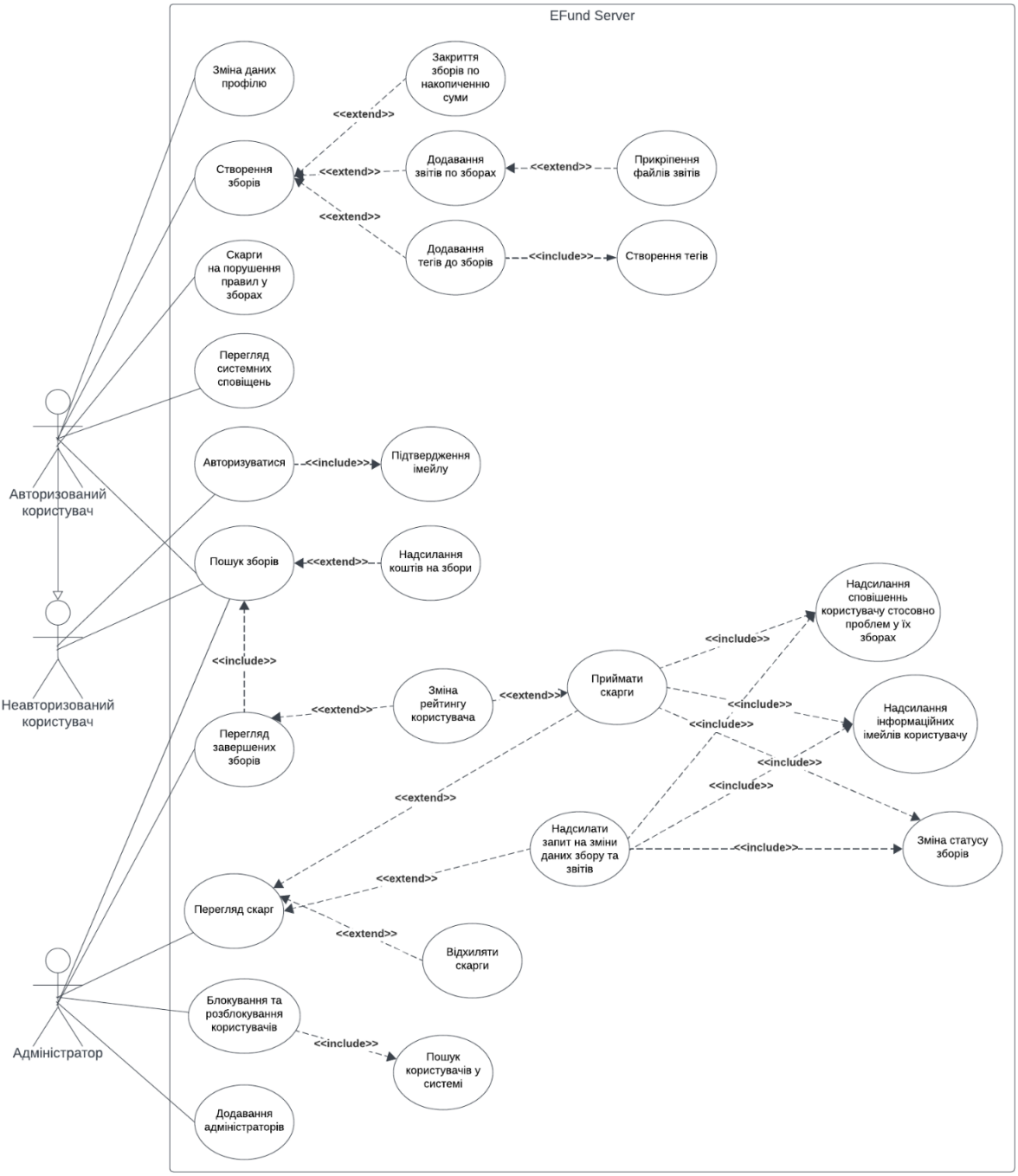


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1-2.12 наведено опис варіантів використання програмного забезпечення.

Таблиця 2.1 – Варіант використання UC-01

Use case name	Пошук фандрейзингів
Use case ID	UC-01
Goals	Отримати релевантний список зборів за критеріями пошуку
Actors	Гість, користувач, адміністратор
Trigger	Перехід на сторінку зборів або ініціація пошуку
Pre-conditions	-
Flow of Events	Користувач переходить на сторінку зборів. У відповідних полях форми він може ввести ключові слова з назви чи опису збору, обрати теги, статус збору (наприклад, відкритий, суму зібрано або завершений) або вказати фандрайзера (тобто користувача, який створив збір). Після натискання кнопки пошуку система формує список зборів, що відповідають введеним критеріям
Extension	Користувач може комбінувати декілька критеріїв одночасно — наприклад, тег і статус або ім’я автора та ключове слово. У разі некоректного введення або відсутності результатів, користувач отримує відповідне повідомлення
Post-Condition	Користувач отримує список зборів, які відповідають вказаним параметрам пошуку

Таблиця 2.2 – Варіант використання UC-02

Use case name	Надсилання коштів на збір
Use case ID	UC-02
Goals	Надіслати кошти на збір
Actors	Гість, користувач, адміністратор
Trigger	Користувач натискає на кнопку надіслати кошти
Pre-conditions	Користувач знаходиться на сторінці збору

Продовження таблиці 2.2

Flow of Events	Користувач натискає кнопку надсилання коштів на сторінці збору. Система відкриває сторінку пов’язаної банки monobank, де користувач здійснює переказ коштів на відповідний рахунок
Extension	У разі помилки або відсутності з’єднання з банківським сервісом користувач отримує повідомлення про неможливість здійснення операції
Post-Condition	Кошти надсилаються на рахунок, пов’язаний зі збором

Таблиця 2.3 – Варіант використання UC-03

Use case name	Створення збору
Use case ID	UC-03
Goals	Створити новий збір
Actors	Користувач
Trigger	Користувач натискає на кнопку створення збору
Pre-conditions	Користувач повинен бути авторизованим та мати створену банку збору у monobank
Flow of Events	Користувач переходить на форму створення збору Він заповнює назву, опис, завантажує зображення, вибирає банку з monobank. Користувач додає теги до збору, вибираючи їх з існуючих або створюючи нові, після натискання кнопки створення, система створює збір та зберігає пов’язані теги
Extension	У випадку відсутності доступних банок у monobank відображається відповідне повідомлення
Post-Condition	Збір створено та прив’язано до обраної банки і тегів

Таблиця 2.4 – Варіант використання UC-04

Use case name	Додавання звітів до збору
Use case ID	UC-04
Goals	Додавання звітів до збору
Actors	Користувач

Продовження таблиці 2.4

Trigger	Користувач натискає кнопку додавання звітів
Pre-conditions	Користувач авторизований, збір створений
Flow of Events	Користувач додає до збору звіт, зазначаючи його назву, опис та обираючи файли які він хоче додати до звіту. Після натискання кнопки додати до збору додається звіт
Extension	У випадку завантаження файлу з неприйнятним форматом відображається повідомлення з помилкою
Post-Condition	Звіт додано до відповідного збору

Таблиця 2.5 – Варіант використання UC-05

Use case name	Пошук користувачів у системі
Use case ID	UC-05
Goals	Отримати релевантний список користувачів за заданими критеріями
Actors	Адміністратор
Trigger	Перехід на сторінку керування користувачами або ініціація пошуку
Pre-conditions	Адміністратор успішно авторизований у системі
Flow of Events	Адміністратор переходить на спеціальну сторінку керування користувачами. У відповідних полях він може ввести критерії для пошуку: ім’я, електронну пошту, унікальний ідентифікатор користувача або інші атрибути, залежно від доступного інтерфейсу. Після введення інформації адміністратор натискає кнопку пошуку, і система формує перелік користувачів, які відповідають заданим фільтрам. Цей список відображається у вигляді таблиці з можливістю взаємодії (наприклад, блокування, редагування прав)
Extension	Адміністратор може комбінувати декілька фільтрів одночасно для точнішого пошуку. У випадку, якщо жоден користувач не відповідає критеріям, система повідомляє про відсутність результатів

Продовження таблиці 2.5

Post-Condition	Адміністратор отримує список користувачів, які відповідають заданим параметрам, з можливістю подальшої взаємодії з ними

Таблиця 2.6 – Варіант використання UC-06

Use case name	Контроль користувачів у системі
Use case ID	UC-06
Goals	Здійснення адміністративного контролю над обліковими записами користувачів
Actors	Адміністратор
Trigger	Адміністратор відкриває профіль користувача або список користувачів для керування
Pre-conditions	Адміністратор авторизований та має відповідні права доступу
Flow of Events	Адміністратор заходить на сторінку профілю конкретного користувача або у загальний список користувачів. Він переглядає поточний статус облікового запису та обирає одну з доступних дій — заблокувати чи розблокувати користувача. Після натискання відповідної кнопки зміна статусу негайно застосовується. Інтерфейс оновлюється для відображення нового стану користувача
Extension	-
Post-Condition	Користувач отримує новий статус. Заблоковані користувачі втрачають можливість створювати та редагувати збори, а також додавати або змінювати звіти

Таблиця 2.7 – Варіант використання UC-07

Use case name	Редагування даних профілю
Use case ID	UC-07
Goals	Надання користувачу можливості змінювати свої персональні дані
Actors	Користувач, адміністратор

Продовження таблиці 2.7

Trigger	Користувач переходить до власного профілю та ініціює редагування
Pre-conditions	Користувач має активну авторизацію у системі
Flow of Events	Користувач відкриває сторінку редагування профілю. У доступних полях він може змінити ім’я, імейл, пароль, аватар, опис про себе або додати токен з monobank для подальшої інтеграції. Після внесення змін користувач натискає кнопку збереження, і дані оновлюються у системі. У випадку успішного збереження виводиться повідомлення про успіх
Extension	У випадку некоректного формату токену або інших помилок валідації система підсвічує відповідні поля і забороняє збереження до усунення проблем
Post-Condition	Дані користувача оновлено у системі. Зміни відображаються при наступному вході або оновленні сторінки профілю

Таблиця 2.8 – Варіант використання UC-08

Use case name	Надсилання скарги на збір
Use case ID	UC-01
Goals	Повідомлення адміністрації про порушення або підозру щодо збору
Actors	Користувач
Trigger	Користувач натискає кнопку «Поскаржитись» на сторінці збору
Pre-conditions	Користувач авторизований
Flow of Events	Користувач переглядає сторінку збору, який викликає підозру. Він натискає кнопку «Поскаржитись» і заповнює форму скарги, де зазначає причину та додає при потребі додаткову інформацію. Після натискання кнопки «Надіслати» скарга зберігається у системі та стає доступною для перегляду модератором
Extension	Якщо форма заповнена некоректно або причина не обрана, кнопка надсилання неактивна. Користувач отримує повідомлення з інструкцією щодо коригування

Продовження таблиці 2.8

Post-Condition	Скарга надіслана до адміністративної частини системи для подальшого розгляду

Таблиця 2.9 – Варіант використання UC-09

Use case name	Перегляд та обробка скарг
Use case ID	UC-09
Goals	Надання адміністратору інструментів для модерації скарг
Actors	Адміністратор
Trigger	Адміністратор відкриває розділ скарг у панелі адміністратора
Pre-conditions	Адміністратор авторизований
Flow of Events	Адміністратор відкриває список скарг, обирає одну з них та переглядає її зміст разом із деталями збору та історією автора. Після аналізу інформації він може: – відхилити скаргу; – прийняти скаргу, що призводить до блокування чи архівації збору; – зробити запит на зміни, який надсилається автору збору з інструкціями щодо редагування. Автор збору отримає системне сповіщення та імейл з поясненнями щодо скарги. У разі грубого порушення адміністратор може знизити рейтинг автора
Extension	-
Post-Condition	Стан збору змінено відповідно до рішення адміністратора. Автор збору повідомлений про результат через email та сповіщення. Рейтинг автора оновлено за необхідності. Користувачі, які подали скаргу, отримують сповіщення з описом змін її статусу

Таблиця 2.10 – Варіант використання UC-10

Use case name	Перевірка завершених зборів
Use case ID	UC-10

Продовження таблиці 2.10

Goals	Надання адміністратору змоги перевіряти завершені збори та змінювати рейтинг користувача
Actors	Адміністратор
Trigger	Адміністратор переходить до розділу завершених зборів
Pre-conditions	Збір має статус завершеного
Flow of Events	Адміністратор відкриває розділ завершених зборів, обирає конкретний збір та аналізує його дані, перебіг та звіти. За потреби адміністратор приймає рішення про підвищення або зниження довіри автора, яке відображається в рейтингу користувача
Extension	Якщо по збору вже подано скаргу, вона враховується при ухваленні рішення
Post-Condition	Збір позначається як перевірений, оцінка рейтингу автора оновлюється

Таблиця 2.11 – Варіант використання UC-11

Use case name	Зміна статусу збору користувачем
Use case ID	UC-11
Goals	Керування статусом збору користувачем (закриття, завершення або видалення)
Actors	Користувач
Trigger	Користувач відкриває інтерфейс редагування збору
Pre-conditions	Користувач має створений збір

Продовження таблиці 2.11

Flow of Events	Користувач переходить до сторінки редагування збору. В залежності від етапу кампанії, він може: – закрити збір, якщо зібрано цільову суму. Після цього збір позначається як закритий і більше не приймає пожертви; – завершити збір після реалізації коштів, наприклад, придбання товарів чи завершення проєкту. Такий збір може містити фінальний звіт; – видалити збір, якщо він не був активним або був створений помилково. Користувач підтверджує дію, і статус збору оновлюється
Extension	У разі спроби змінити статус збору, який уже був завершений або модерується адміністратором, користувач отримує попередження, і зміни блокуються
Post-Condition	Статус збору оновлений, а зміни відображаються у списку зборів і на його сторінці. Якщо збір завершено, адміністратор отримує доступ до його модерації

Таблиця 2.12 – Варіант використання UC-12

Use case name	Зміна статусу збору адміністратором
Use case ID	UC-12
Goals	Адміністративне керування статусом зборів
Actors	Адміністратор
Trigger	Адміністратор відкриває інтерфейс управління зборами
Pre-conditions	Адміністратор авторизований

Продовження таблиці 2.12

Flow of Events	Адміністратор відкриває сторінку збору, що потребує модерації або дій. Він має можливість: − видалити збір, якщо виявлено порушення або за запитом користувача; − приховати збір від публічного доступу у випадку тимчасового порушення правил або підозр; − відновити видимість прихованого збору після внесення змін користувачем. Всі зміни супроводжуються поясненням дій, які можуть автоматично знижувати рейтинг користувача
Extension	Якщо збір вже має статус «Перевірено», адміністратор не може внести зміни до даного збору
Post-Condition	Статус збору оновлено. Користувач повідомлений через email і систему сповіщень

2.2 Розроблення нефункціональних вимог

Вебзастосунок має забезпечувати швидке завантаження основного вмісту, зокрема списків кампаній, сторінок перегляду та форм створення зборів. Час очікування користувача на виконання основних дій не повинен перевищувати двох секунд за умов середнього навантаження.

Система повинна бути здатна обробляти значну кількість одночасних запитів, не допускаючи критичних затримок або збоїв у роботі. Усі користувацькі дії, що змінюють дані, мають бути гарантовано оброблені або повністю скасовані у разі помилки.

Дані користувачів, включаючи облікову інформацію та банківські ідентифікатори, мають передаватися виключно через захищене з'єднання. Сторонні особи не повинні мати можливості перехопити або змінити передану інформацію. Також має бути виключена можливість несанкціонованого доступу до ресурсів системи.

Програмне забезпечення має стабільно функціонувати у всіх сучасних браузерах, таких як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge та

Opera. Інтерфейс повинен бути адаптований як для десктопних, так і для мобільних пристроїв, з урахуванням різних розмірів екранів.

У разі виникнення помилок або непередбачених ситуацій, застосунок повинен повертати коректні HTTP-статуси, а користувач має бачити інформативні повідомлення без втрати доступу до інших функцій.

2.3 Аналіз системних вимог

Мінімальні вимоги до настільних комп'ютерів та ноутбуків:

- тип процесору: Intel Core i3;
- не менше 2 ГБ оперативної пам'яті;
- стабільне підключення до мережі зі швидкістю від 5 Мбіт/с;
- веббраузери останніх версій: Google Chrome, Mozilla Firefox, Microsoft Edge або Opera.

Рекомендовані вимоги для повного користувацького досвіду:

- тип процесору: Intel Core i7;
- 4 ГБ або більше оперативної пам'яті;
- стабільне підключення до мережі зі швидкістю від 20 Мбіт/с.

2.4 Аналіз економічних показників програмного забезпечення

Для оцінки основних економічних характеристик створеного програмного забезпечення було використано базову модель COCOMO. Вихідними параметрами стали обсяг коду, кількість виконавців і рівень їхньої кваліфікації.

Загальний обсяг написаного коду становить приблизно 37000 рядків (LOC = 37000), тобто 37 тис. логічних рядків (kLOC = 37). Відповідно до класифікації COCOMO, проєкт такого масштабу відноситься до категорії Semi-detached — тобто проміжної між простими (Organic) та складними (Embedded) системами. До цієї категорії входять застосунки середньої складності, які розробляються невеликими командами з помірною взаємодією та із залученням досвідчених спеціалістів.

Команду розробки складали два спеціалісти з рівнем кваліфікації middle, зарплата кожного з розробників взята на рівні середньої зарплати розробників рівня middle і становить 1500 $ на місяць. Фактична тривалість розробки — 4 місяці.

Базова модель Basic COCOMO має наступні коефіцієнти для класу Semi-detached: $a_b = 3.0,\ b_b = 1.12$. Проведемо розрахунок трудомісткості за наступною формулою:

$$E = a_b \cdot Size^{b_b} \quad (2.1)$$

Де $Size = kLOC$ – це розмір проєкту, тож підставивши значення отримаємо $E = 3 * 37^{1.12} \approx 181.65$ людино місяців. Фактична трудомісткість команди (2 розробники × 4 місяці) становила лише 8 людино-місяців, що суттєво менше розрахункової. Таке відхилення можна пояснити широким використанням готових бібліотек, зокрема UI-фреймворків, ORM, які генерують велику частину коду.

Далі обчислюємо прогнозовану тривалість проєкту за формулою:

$$T = c_b \cdot E^{d_b} \quad (2.2)$$

Підстановка коефіцієнтів $c_b = 2.5$ та $d_b = 0.35$ дає наступне значення тривалості: $T \approx 16.22$ міс. Така оцінка в 4 рази перевищує реальну тривалість, що ще раз підкреслює спрощення проєкту за рахунок повторного використання модулів, шаблонів і зовнішніх сервісів (наприклад, інтеграції з банківським API без потреби власного платіжного процесингу).

Собівартість розробки складається із зарплат та платежів за хостиг. Основна частина витрат у межах проєкту — це оплата праці. При зарплаті 1500 $ за місяць і двох розробниках протягом 4 місяців отримаємо $1500 \cdot 4 = 12000\$$.

Додатково слід врахувати витрати на хостинг середня ціна мінімальної конфігурації якого становить близько 50$ за місяць, тобто сумарна вартість виходить 200$. Таким чином, собівартість розрбки $Cdev = 12000 + 200 = 12200\$$. .

Висновки до розділу

У межах другого розділу пояснювальної записки було розглянуто сценарії взаємодії користувачів із системою та описано їх у діаграмі варіантів використання та таблицях з описом варіантів використання. Це дозволило чітко визначити основний функціонал програмного забезпечення з точки зору кінцевого користувача. Було сформульовано нефункціональні та системні вимоги. Також було здійснено оцінку економічної ефективності розробки — проведено розрахунок трудомісткості, тривалості виконання та приблизної собівартості створення початкової версії системи.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Для реалізації програмного забезпечення, було обрано клієнт-серверну архітектуру [9]. Такий підхід дозволяє розділити систему на дві основні частини: клієнтську, що відповідає за взаємодію з користувачем, та серверну, яка виконує обробку запитів, реалізує бізнес-логіку й забезпечує роботу з базою даних. Перші 2 рівні діаграми C4 зображені на рисунках 3.1 та 3.2:

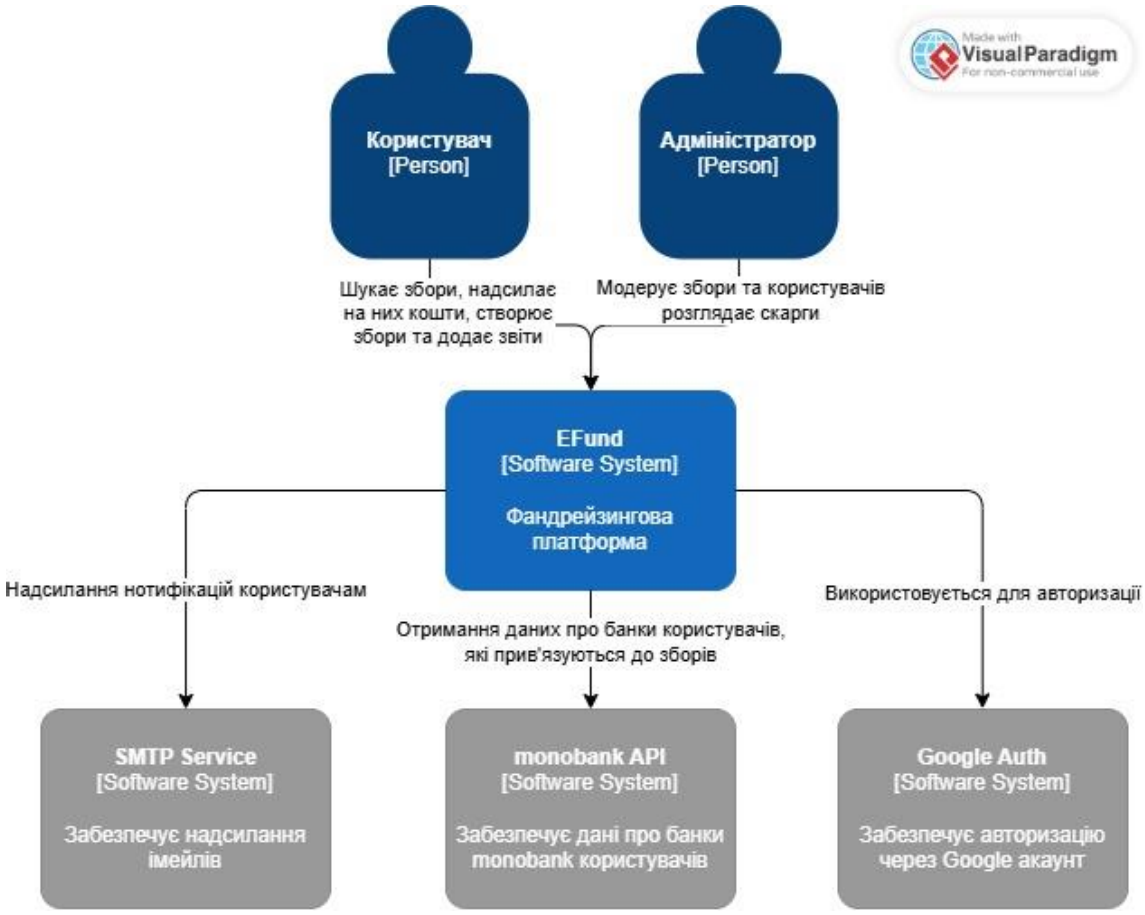


Рисунок 3.1 – Перший рівень діаграми C4

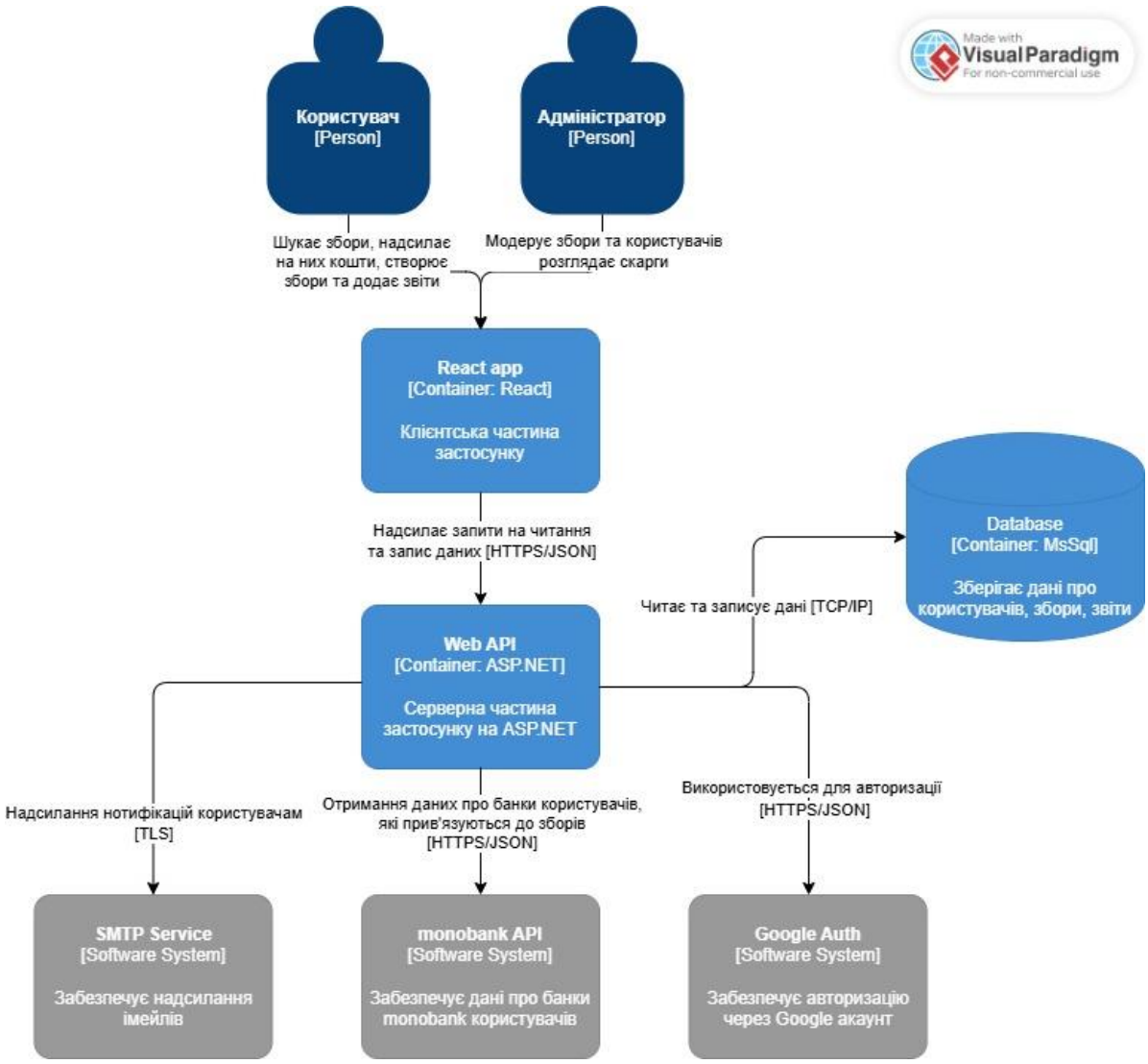


Рисунок 3.2 – Другий рівень діаграми C4

На рівні обох компонентів — клієнта і сервера — застосовано багаторівневу архітектуру [10]. Це забезпечує чітке розмежування обов’язків між окремими частинами системи, полегшує супровід, тестування та подальший розвиток програмного продукту.

Клієнтська частина відповідає за відображення інтерфейсу, взаємодію з користувачем, передавання запитів до сервера та обробку відповідей. Вона реалізує механізми навігації, авторизації, заповнення форм зборів, перегляду звітів та інші функції, доступні для гостей, користувачів або адміністраторів.

Серверна частина забезпечує реалізацію основної логіки платформи: створення та редагування зборів, керування звітами, модерацію, обробку скарг, управління ролями користувачів, відправку імейлів та багато іншого.

Також сервер опрацьовує запити, проводить валідацію вхідних даних, управляє авторизацією, виконує бізнес-правила та взаємодіє з базою даних.

Взаємодія між клієнтом і сервером відбувається через HTTP-запити. Для ідентифікації та авторизації користувачів застосовується JWT токени [11], що забезпечує захищений доступ до ресурсів відповідно до ролі користувача.

Кожен компонент — як клієнтський, так і серверний — побудований з урахуванням принципів багаторівневої архітектури. Такий підхід дозволяє ізолювати представлення даних, логіку обробки та доступ до ресурсів, що забезпечує:

- модульність і гнучкість при внесенні змін;
- можливість незалежного масштабування частин системи;
- підвищену безпеку при обробці запитів;
- зручність тестування та відлагодження окремих рівнів.

У підсумку, клієнт-серверна архітектура в поєднанні з багаторівневою структурою обох частин платформи дозволяє досягти високої надійності, підтримуваності та масштабованості розробленого застосунку. Вона забезпечує ефективне виконання усіх необхідних функцій платформи для збору коштів, модерації контенту, керування звітністю та користувачами.

3.2 Архітектурні рішення та обґрунтування вибору засобів розробки

У процесі розробки програмного забезпечення для організації зборів коштів було прийнято низку архітектурних і технологічних рішень, що забезпечують стабільну, масштабовану та безпечну роботу системи. Основною моделлю взаємодії обрано клієнт-серверну архітектуру з розділенням на клієнтську та серверну частини. Обидві реалізовані за принципами багаторівневої побудови, що сприяє кращому структуруванню коду, полегшує тестування й підтримку.

На стороні сервера для реалізації логіки, обробки запитів і доступу до бази даних застосовано сучасні інструменти з .NET-екосистеми [12], зокрема середовище ASP.NET Core [13]. Цей фреймворк було обрано з огляду на його

продуктивність, гнучкість конфігурації, підтримку масштабування та безпеки, а також можливість створювати RESTful API для зручної взаємодії з клієнтами. Крім того, він добре інтегрується з ORM-рішеннями, такими як Entity Framework [14], що було використано для роботи з реляційною базою даних. Такий підхід дозволяє зменшити потребу в написанні SQL-запитів і забезпечує обробку даних у вигляді об'єктів.

У якості мови програмування обрано C# [15], яка відзначається високим рівнем стабільності, підтримкою об'єктно-орієнтованого підходу, асинхронного виконання та широким спектром бібліотек для роботи з вебресурсами, базами даних, мережею та безпекою.

Для клієнтської частини було використано технології на базі TypeScript [16]. Основною бібліотекою для побудови інтерфейсу став React [17] — гнучке рішення, що дозволяє створювати динамічні, адаптивні та багаторазові компоненти інтерфейсу. Вибір React обумовлений його широкою підтримкою у спільноті, високою продуктивністю та зручністю інтеграції з сучасними API.

Окремо варто зазначити вибір типу бази даних. На етапі проєктування розглядалися два основні варіанти: реляційна та нереляційна модель зберігання даних. Нереляційні бази (наприклад, документоорієнтовані системи) мають переваги в умовах гнучкої схеми даних і горизонтального масштабування. Однак для даного проєкту, де основні об'єкти мають чіткі зв’язки (збори, користувачі, звіти, транзакції) та важлива підтримка складних запитів і транзакцій, реляційна база даних виявилася оптимальним вибором. Вона дозволяє зберігати структуровані зв’язки між сутностями, забезпечує надійність, підтримку цілісності даних і добре інтегрується з ORM-інструментами.

Загалом, обраний набір технологій та архітектурних рішень забезпечує баланс між продуктивністю, простотою розробки, масштабованістю й підтримуваністю. Він дозволяє реалізовувати функціональність у відповідності до вимог, забезпечуючи при цьому гнучкість для подальшого розвитку та розширення системи.

3.3 Конструювання програмного забезпечення

3.3.1 Опис структури бази даних

В якості системи управління базами даних використовується MsSQL Server. База даних серверу призначена для зберігання користувачів, ролей, а також даних зборів, тегів, звітів та скарг. Опис таблиць бази даних наведено у таблицях 3.1-3.22. Модель бази даних наведена на рисунку 3.3

Таблиця 3.1 – Опис таблиці AspNetUsers

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор користувача
DisplayName	nvarchar	повне ім’я користувача
RefreshToken	nvarchar	рефреш токен користувача
RefreshTokenExpiresAt	datetime	час коли рефреш токен стане недійсним
Email	nvarchar	імейл користувача
NormalizedEmail	nvarchar	нормазіований імейл користувача
EmailConfirmed	bit	чи підтверджений імейл
PasswordHash	nvarchar	захешований пароль
SecurityStamp	nvarchar	штамп безпеки користувача
ConcurrencyStamp	nvarchar	штамп безпеки конкурентних задач користувача
AvatarPath	nvarchar	шлях до аватару користувача
CreatedByAdmin	bit	чи був користувач створений адміністратором

Продовження таблиці 3.1

IsBlocked	bit	чи заблокований користувач
CreatedAt	datetimeoffset	час створення користувача

Таблиця 3.2 – Опис таблиці AspNetUserLogins

Назва поля	Тип даних	Опис
ProviderDisplayName	nvarchar	ім’я стороннього логіну
UserId	uniqueidentifier	ідентифікатор користувача
LoginProvider	nvarchar	провайдер стороннього логіну
ProviderKey	nvarchar	ключ стороннього логіну

Таблиця 3.3 – Опис таблиці AspNetUserClaims

Назва поля	Тип даних	Опис
Id	int	ідентифікатор клеімс
UserId	uniqueidentifier	ідентифікатор користувача
ClaimType	nvarchar	тип клеімс
ClaimValue	nvarchar	значення клеімс

Таблиця 3.4 – Опис таблиці AspNetRoles

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор ролі
Name	nvarchar	ім’я ролі
NormalazideName	nvarchar	Нормалізоване ім’я ролі
ConcurrencyStamp	nvarchar	штамп безпеки конкурентних задач ролі

Таблиця 3.5 – Опис таблиці AspNetUserRoles

Назва поля	Тип даних	Опис
UserId	uniqueidentifier	ідентифікатор користувача
RoleId	uniqueidentifier	ідентифікатор ролі

Таблиця 3.6 – Опис таблиці AspNetUserRoleClaims

Назва поля	Тип даних	Опис
Id	int	ідентифікатор
RoleId	uniqueidentifier	ідентифікатор ролі
ClaimType	nvarchar	тип клеімс
ClaimValue	nvarchar	значення клеімс

Таблиця 3.7 – Опис таблиці UserRegistrations

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор
Code	Int	код підтвердження імейлу
CreatedAt	datetimeoffset	час створення
ExpiresAt	datetimeoffset	час коли код стане недійсним
IsCodeRegenerated	bit	чи був код перегенерований
UserId	uniqueidentifier	ідентифікатор користувача

Таблиця 3.8 – Опис таблиці UserMonobanks

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор
UserId	uniqueidentifier	ідентифікатор користувача

Продовження таблиці 3.8

MonobankToken	varbinary	захешований токен monobank

Таблиця 3.9 – Опис таблиці Fundraisings

Назва поля	Тип даних	Опис
Id	Uniqueidentifier	ідентифікатор збору
Provider	Int	провайдер збору
Title	nvarchar	назва збору
Description	nvarchar	опис
UserId	uniqueidentifier	ідентифікатор користувача
AvatarPath	nvarchar	шлях до аватару збору
CreatedAt	datetimeoffset	час створення збору
ClosedAt	datetimeoffset	час закриття збору
ReviewedAt	datetimeoffset	час перевірки збору
ReadyForReviewAt	datetimeoffset	час коли збір став готови до перевірки

Таблиця 3.10 – Опис таблиці MonobankFundraisings

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор
Jarld	nvarchar	ідентифікатор банки з monobank
FundraisingId	uniqueidentifier	ідентифікатор збору

Таблиця 3.11 – Опис таблиці FundraisingReports

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор звіту

Продовження таблиці 3.11

Title	nvarchar	назва звіту
Description	nvarchar	опис
FundraisingId	uniqueidentifier	ідентифікатор збору
CreatedAt	datetimeoffset	час створення звіту

Таблиця 3.12 – Опис таблиці Tags

Назва поля	Тип даних	Опис
Name	nvarchar	назва тегу

Таблиця 3.13 – Опис таблиці FundraisingTag

Назва поля	Тип даних	Опис
FundraisingId	uniqueidentifier	ідентифікатор збору
TagsName	nvarchar	назва тегу

Таблиця 3.14 – Опис таблиці ReportAttachments

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор файлу
FilePath	Nvarchar	шлях до файлу звіту
FundraisingReportId	uniqueidentifier	ідентифікатор звіту
Name	nvarchar	назва файлу

Таблиця 3.15 – Опис таблиці Badge

Назва поля	Тип даних	Опис
Type	int	тип значка

Таблиця 3.16 – Опис таблиці BadgeUser

Назва поля	Тип даних	Опис

Продовження таблиці 3.16

BadgesType	int	тип значка
UsersId	uniqueidentifier	ідентифікатор користувача

Таблиця 3.17 – Опис таблиці Complaint

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор скарги
Status	int	статус скарги
Comment	nvarchar	коментар
RequestedAt	datetimeoffset	час створення скарги
ReviewedAt	datetimeoffset	час перегляду скарги адміністратором
FundraisingId	uniqueidentifier	ідентифікатор збору на який була створена скарга
RequestedBy	uniqueidentifier	ідентифікатор користувача що поскаржився
RequestedFor	uniqueidentifier	ідентифікатор користувача на якого поскаржилися
ReviewedBy	uniqueidentifier	ідентифікатор адміністратора що переглянув скаргу
Number	int	унікальний номер скарги

Таблиця 3.18 – Опис таблиці ComplaintViolation

Назва поля	Тип даних	Опис
ComplaintsId	uniqueidentifier	ідентифікатор скарги
ViolationsId	uniqueidentifier	ідентифікатор порушення

Таблиця 3.19 – Опис таблиці FundraisingReview

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор перевірки збору адміністратором
RatingChange	decimal	зміна рейтингу користувача

Продовження таблиці 3.19

Comment	nvarchar	коментар адміністратора по результатах перевірки
CreatedAt	datetimeoffset	час створення перевірки
ReviewedBy	uniqueidentifier	ідентифікатор адміністратора що здійснив перевірку
FundraisingId	uniqueidentifier	ідентифікатор збору, який був перевірений

Таблиця 3.20 – Опис таблиці Notification

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор сповіщення
UserId	uniqueidentifier	ідентифікатор користувача
Reason	int	причина сповіщення
IsRead	bit	чи прочитане сповіщення
Args	nvarchar	JSON поле з додатковими параметрами сповіщення
CreatedAt	datetimeoffset	час створення сповіщення

Таблиця 3.21 – Опис таблиці ViolationGroups

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор групи порушень
Title	nvarchar	назва групи порушень
IsDeleted	bit	чи видалена група порушен

Таблиця 3.22 – Опис таблиці Violations

Назва поля	Тип даних	Опис
Id	uniqueidentifier	ідентифікатор порушення
Title	nvarchar	назва порушень
ViolationGroupId	uniqueidentifier	ідентифікатор групи порушень
IsDeleted	bit	чи видалене порушення

Продовження таблиці 3.22

RatingImpact	decimal	рекомендований вплив на рейтинг користувача

Рисунок 3.3 – Схема бази даних

3.4 Аналіз безпеки даних

Безпека даних є критично важливою складовою функціонування даного програмного забезпечення, зважаючи на обробку персональних даних

користувачів, включно з паролями, токенами доступу до банківських сервісів, інформацією про збори та звітами про використання коштів.

Для автентифікації та управління обліковими записами використовується ASP.NET Core Identity [18], що забезпечує зберігання паролів у захищеному вигляді. Паролі хешуються за допомогою стійкого алгоритму PBKDF2 з використанням криптографічної «солі», що робить неможливим відновлення оригінального пароля навіть у разі компрометації бази даних.

Особливу увагу приділено зберіганню токенів monobank, які є чутливою інформацією. Для їх шифрування використовується симетричний криптоалгоритм AES (Advanced Encryption Standard). Розшифрування таких токенів можливе лише у межах серверного середовища. Сам шифрувальний ключ не зберігається у відкритому вигляді — він зчитується під час запуску застосунку зі змінних середовища або з .env файлу Docker-контейнера, що значно знижує ризик витоку конфіденційних даних.

На рівні взаємодії з базою даних застосовується Entity Framework Core, який за замовчуванням реалізує параметризовані запити. Це виключає можливість SQL-ін’єкцій, оскільки будь-які дані, передані користувачем, обробляються в межах безпечних шаблонів запитів.

З боку клієнта (на React) забезпечено захист від XSS-атак — завдяки автоматичному екрануванню значень у JSX-шаблонах. Усі динамічно вставлені фрагменти проходять додаткову валідацію та санітизацію, що унеможливлює виконання шкідливих скриптів. Компоненти інтерфейсу, що залежать від ролей, динамічно керуються, а доступ до чутливих елементів обмежується відповідно до авторизаційних політик.

Уся комунікація між клієнтом і сервером здійснюється через HTTPS, що гарантує цілісність і конфіденційність переданих даних. HTTP-клієнт (Axios) [19] на клієнтській стороні налаштований на автоматичне оновлення токенів, обробку помилок авторизації (401) та забезпечує збереження сесії без потреби в повторній аутентифікації до моменту завершення терміну дії токена.

Таким чином, у системі реалізовано сучасні підходи до захисту інформації — від хешування паролів і шифрування чутливих даних до безпечної передачі інформації через HTTPS і належного управління доступом на рівні інтерфейсу. Це забезпечує високий рівень довіри до системи та мінімізує ризики несанкціонованого доступу.

Висновки до розділу

У третьому розділі пояснювальної записки було представлено загальну архітектуру програмного забезпечення за допомогою архітектурної діаграми, що відображає взаємодію основних компонентів системи. Надано обґрунтування вибору архітектурного підходу та програмних засобів, використаних під час реалізації. Також було побудовано діаграму структури бази даних, надано опис таблиць і здійснено аналіз рішень щодо захисту даних, зокрема методів шифрування та безпечного зберігання конфіденційної інформації.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Опис контрольного прикладу

Далі наведено контрольний приклад використання програмного забезпечення, розробленого в рамках дипломного проєкту.

При переході на вебсайт користувач направляється на головну сторінку з описом функціоналу вебсайту (рисунок 4.1).

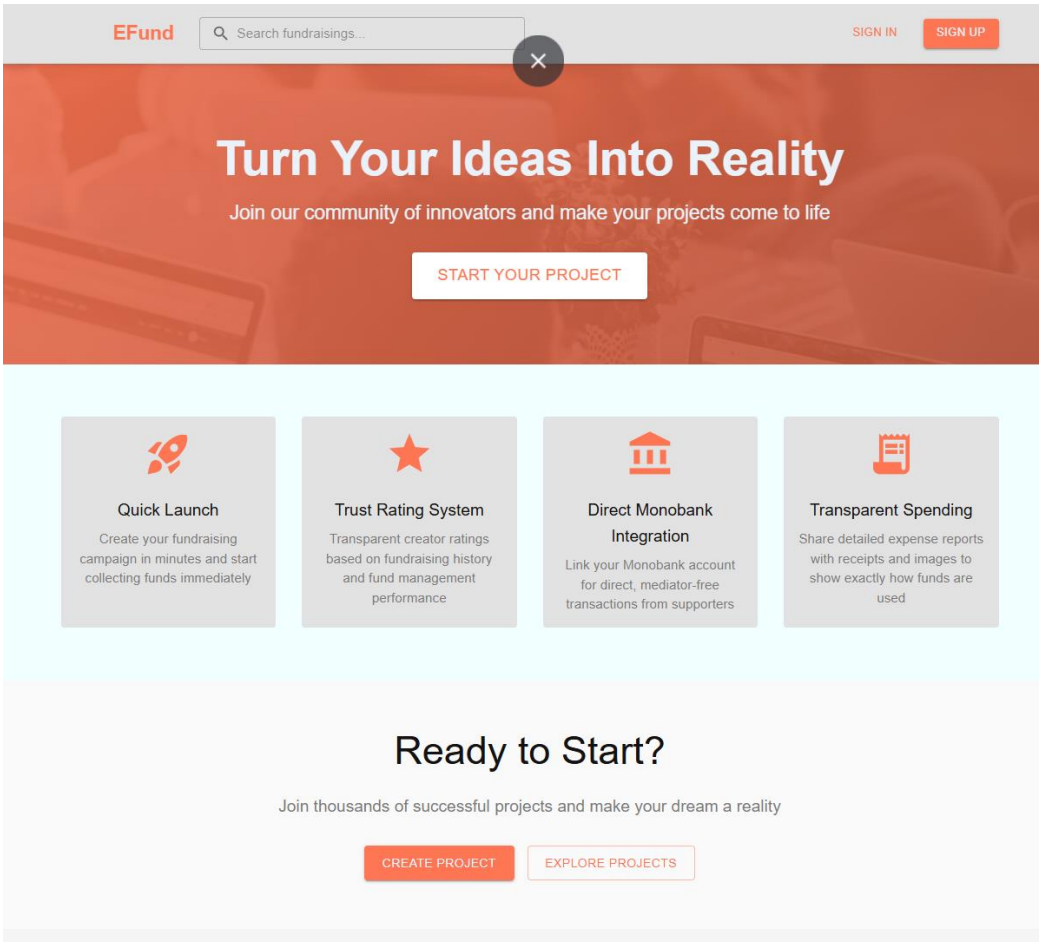


Рисунок 4.1 – Головна сторінка

З головної сторінки користувач може потрапити на загальний пошук фандрейзингів з секцією фільтрів пошуку (рисунок 4.2) та результатами пошуку (рисунок 4.3)

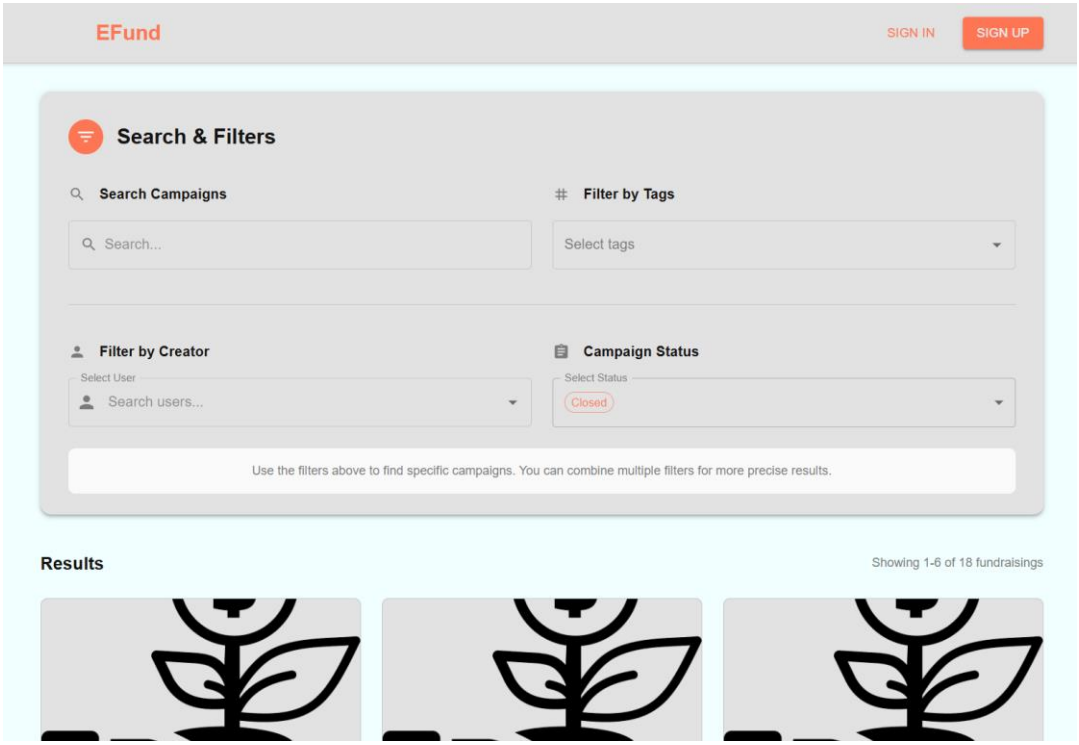


Рисунок 4.2 – Секція фільтрів пошуку

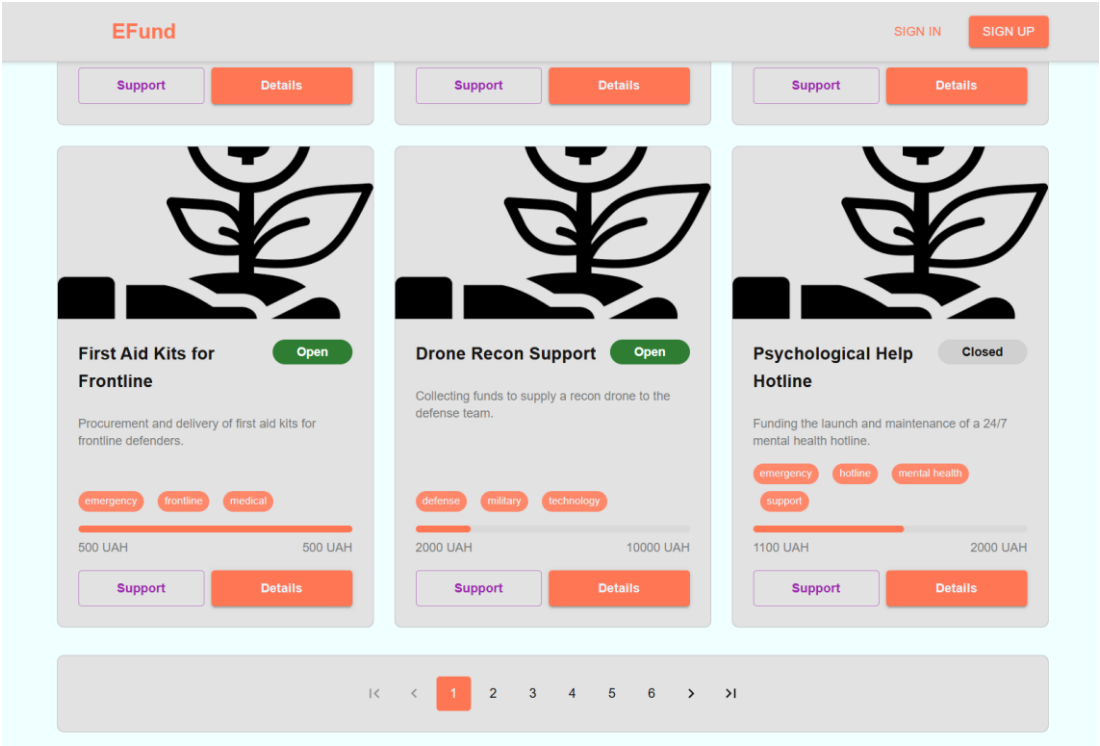


Рисунок 4.3 – Результати пошуку

Користувач може відкрити деталі будь якого фандрейзингу натиснувши на відповідну кнопку «Details» на картці фандрейзингу (рисунок 4.4)

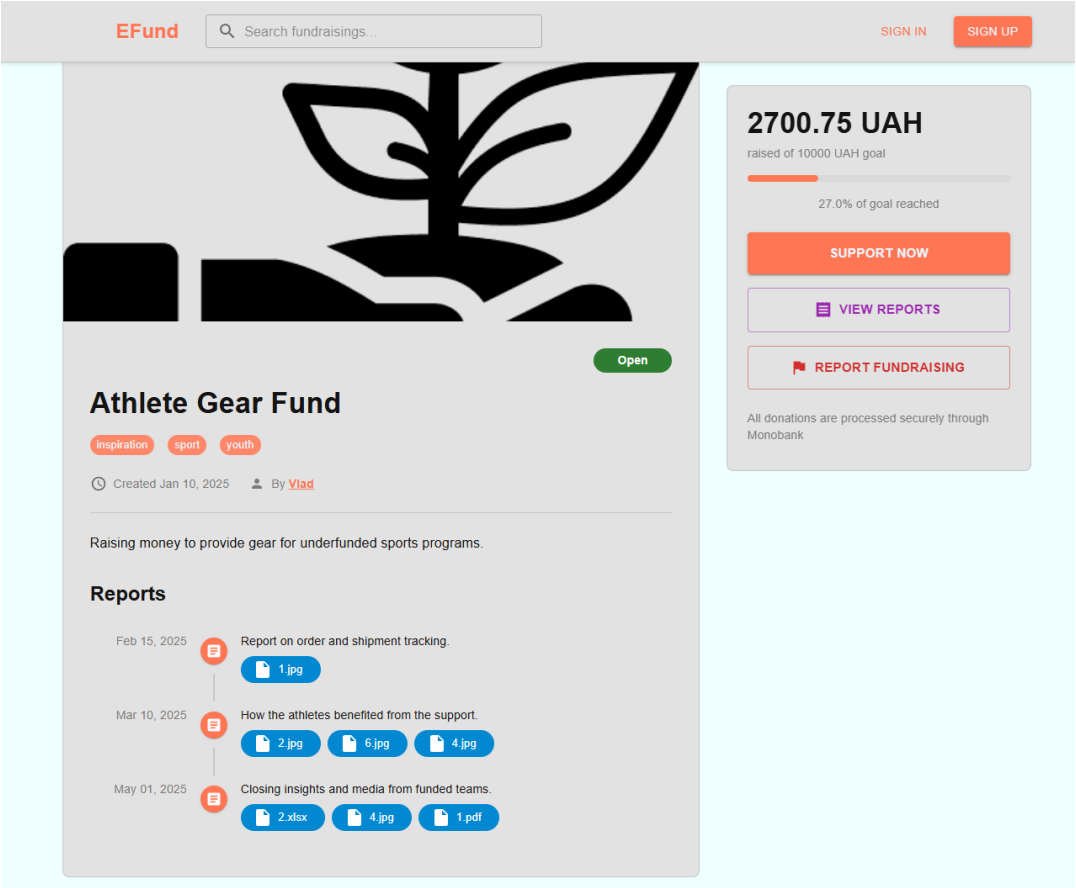


Рисунок 4.4 – Деталі

На сторінці деталей, користувач може бачити опис, дату створення, автора, теги фандрайзингу, а також секцію звітування по цьому збору.

На цій сторінці користувач може перейти до банки monobank, щоб поповнити збір власними коштами (рисунок 4.5), а також має можливість поскаржитись на збір, якщо вважає, що фандрейзинг порушує внутршні правила створення зборів (рисунок 4.6)

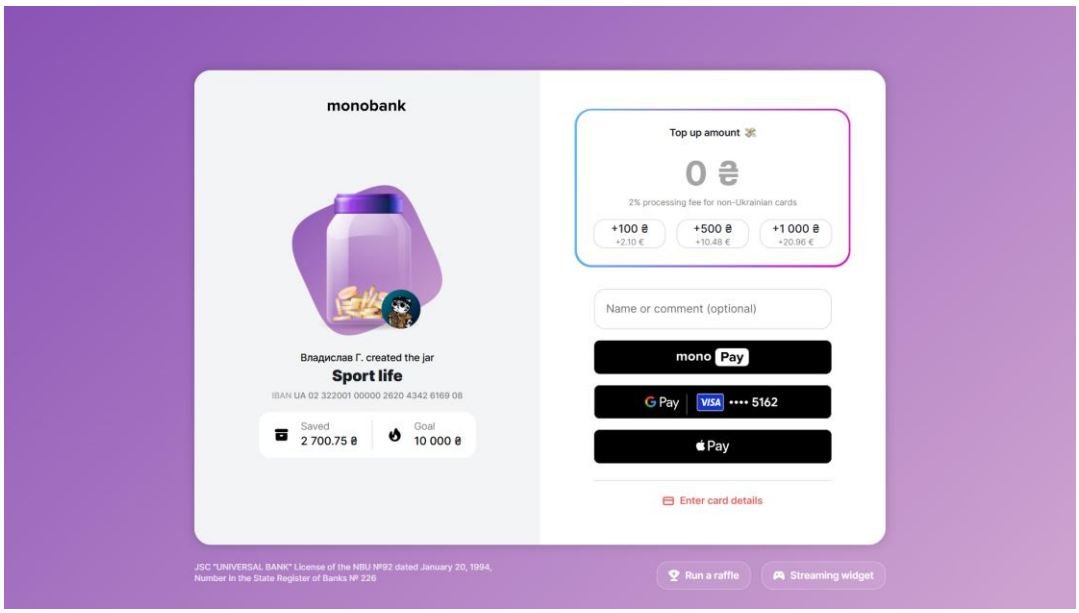


Рисунок 4.5 – Сторінка поповнення збору на сайті monobank

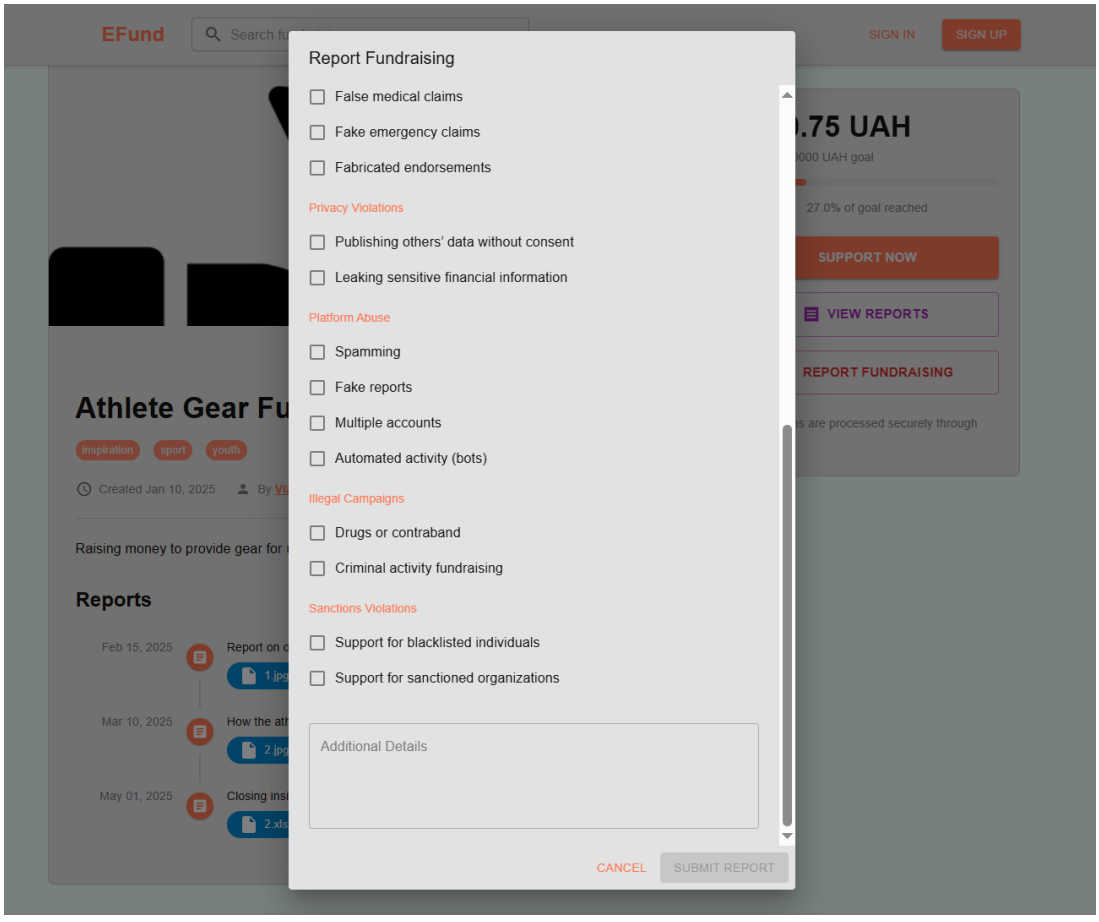


Рисунок 4.6 – Вікно створення скарги на фандрейзинг

Також, зі сторінки деталей фандрейзингу, користувач має змогу побачити короткий опис профіля автора збору (рисунок 4.7), який включає ім’я, пошту та поточний рейтинг користувача, та перейти на профіль автора збору для більш детальної інформації (рисунок 4.8)

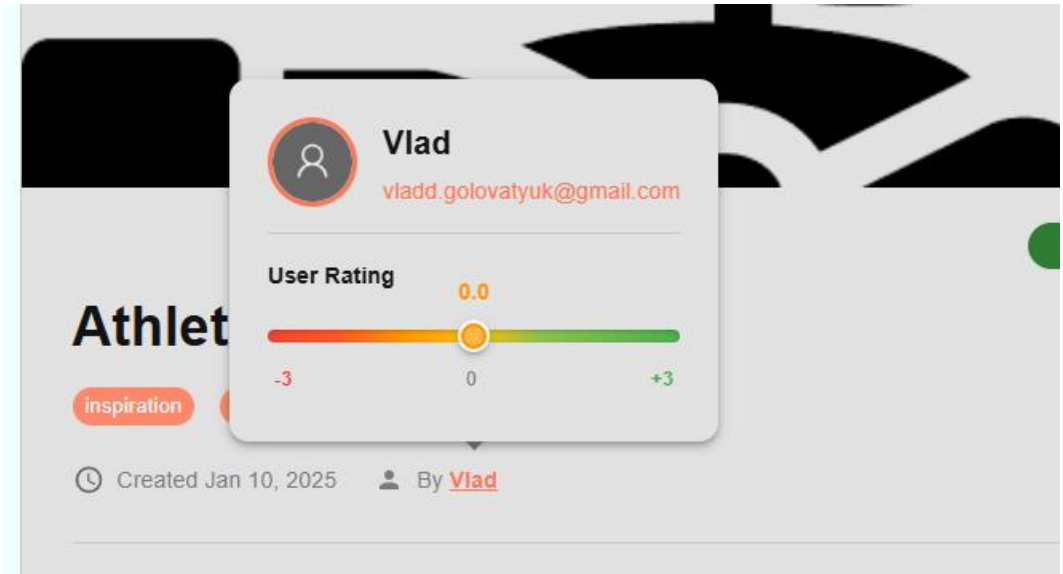


Рисунок 4.7 – Короткий опис профіля автора збору

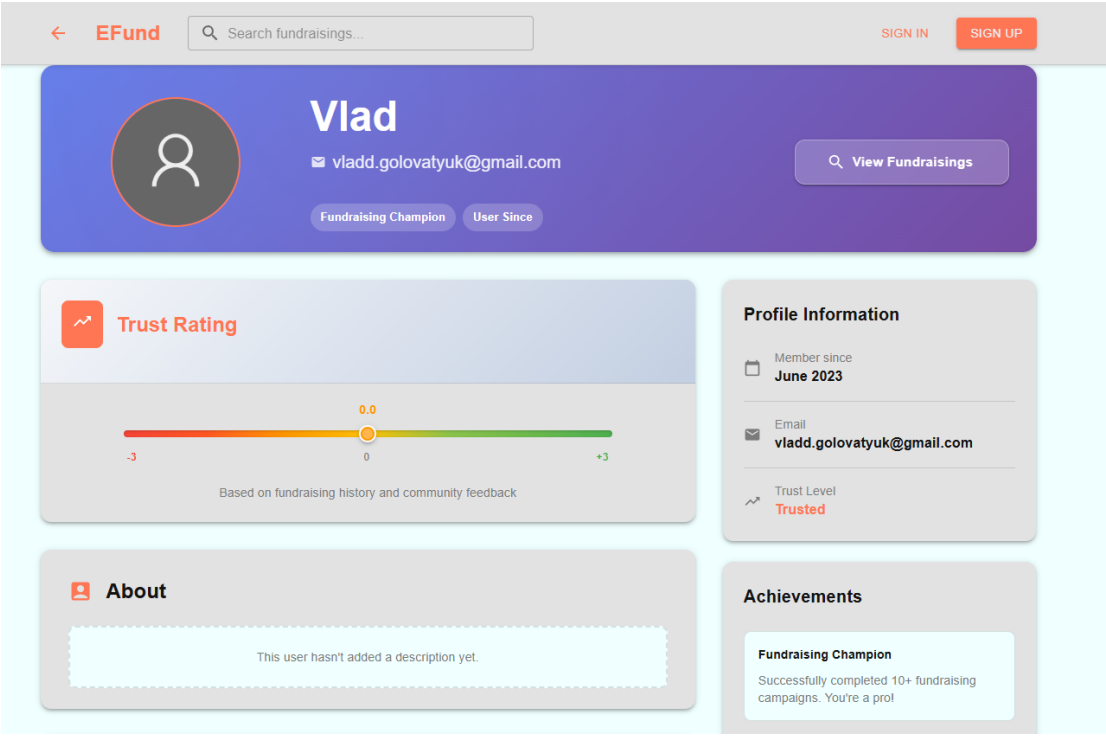


Рисунок 4.8 – Детальний опис профілю користувача

У разі бажання користувача створювати власні збори, у нього є можливість зареєструватись, натиснувши відповідну кнопку «SIGN UP» на верхній частині застосунку (рисунок 4.9)

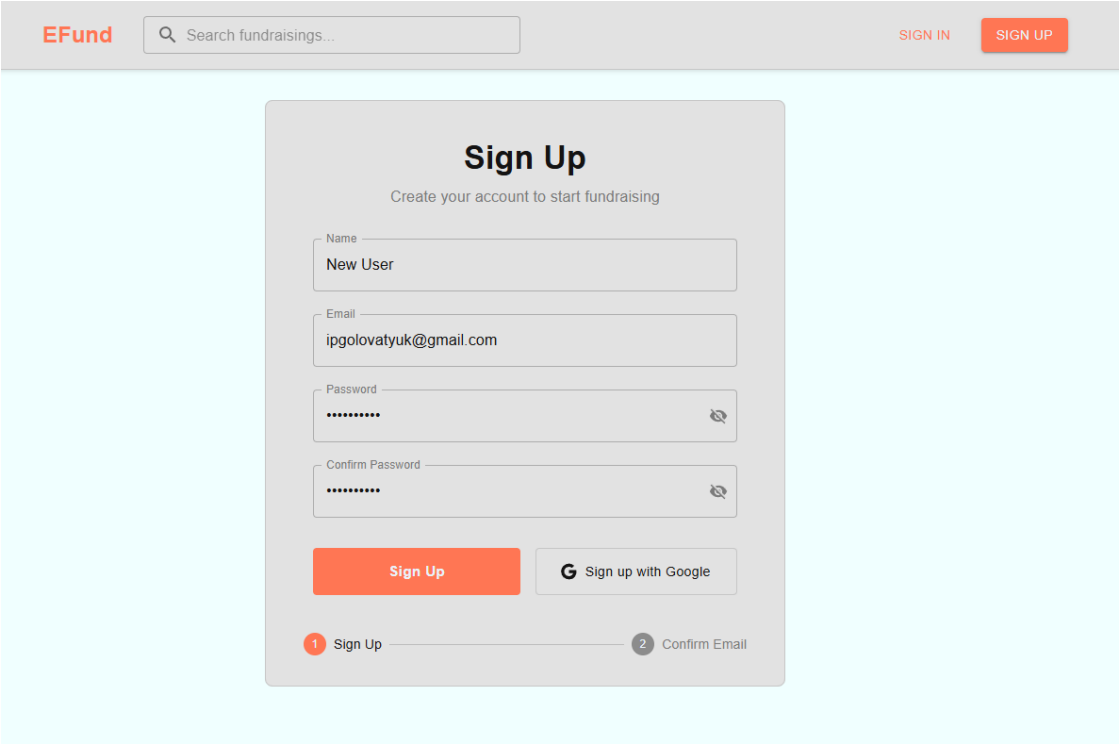


Рисунок 4.9 – Форма реєстрації

Після введеня даних користувача, на його пошту надходить лист з кодом підтвердження (рисунок 4.10), який потрібно ввести на наступний етап реєстрації, підтвердження пошти (рисунок 4.11)

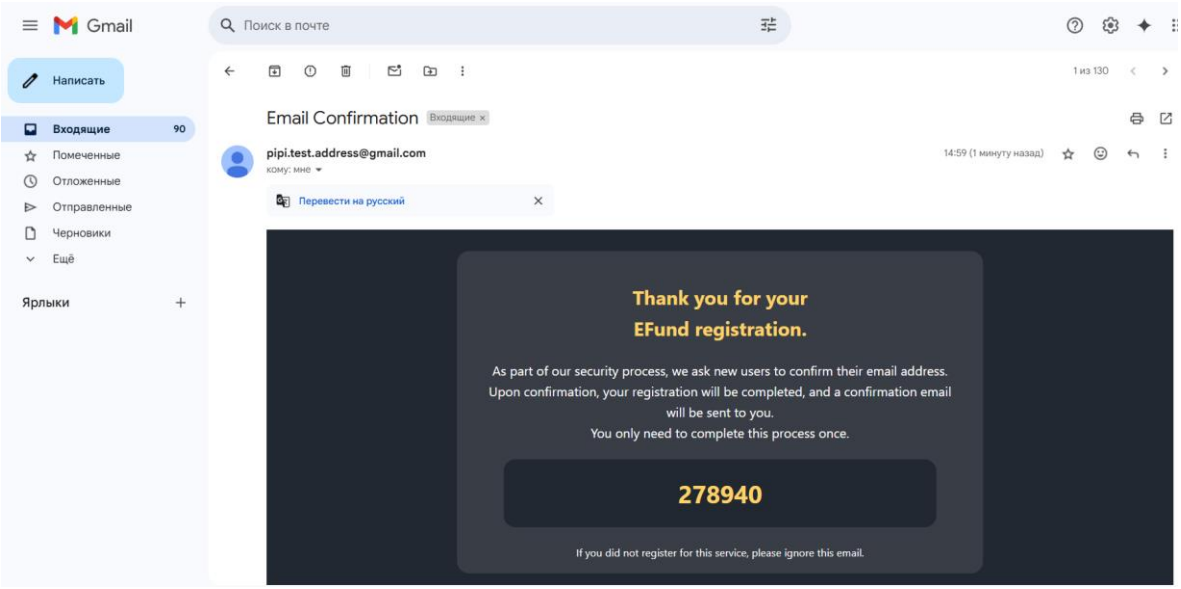


Рисунок 4.10 – Лист з кодом підтвердження пошти

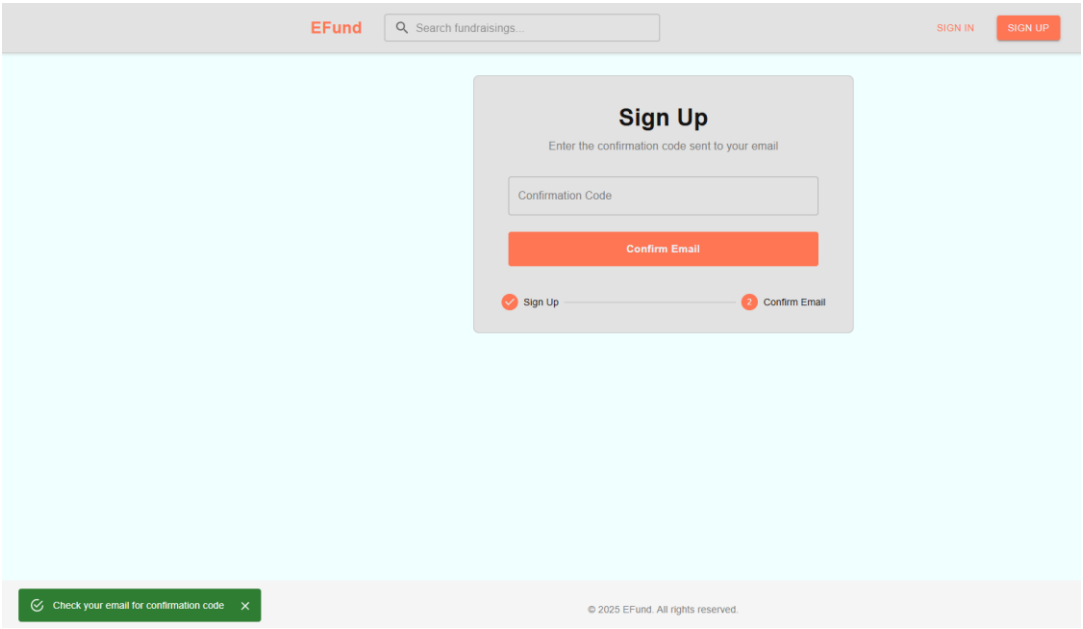


Рисунок 4.11 – Форма підтвердження пошти користувача

Після підтвердження пошти, користувач має перейти в редагування власного профілю (рисунок 4.12)

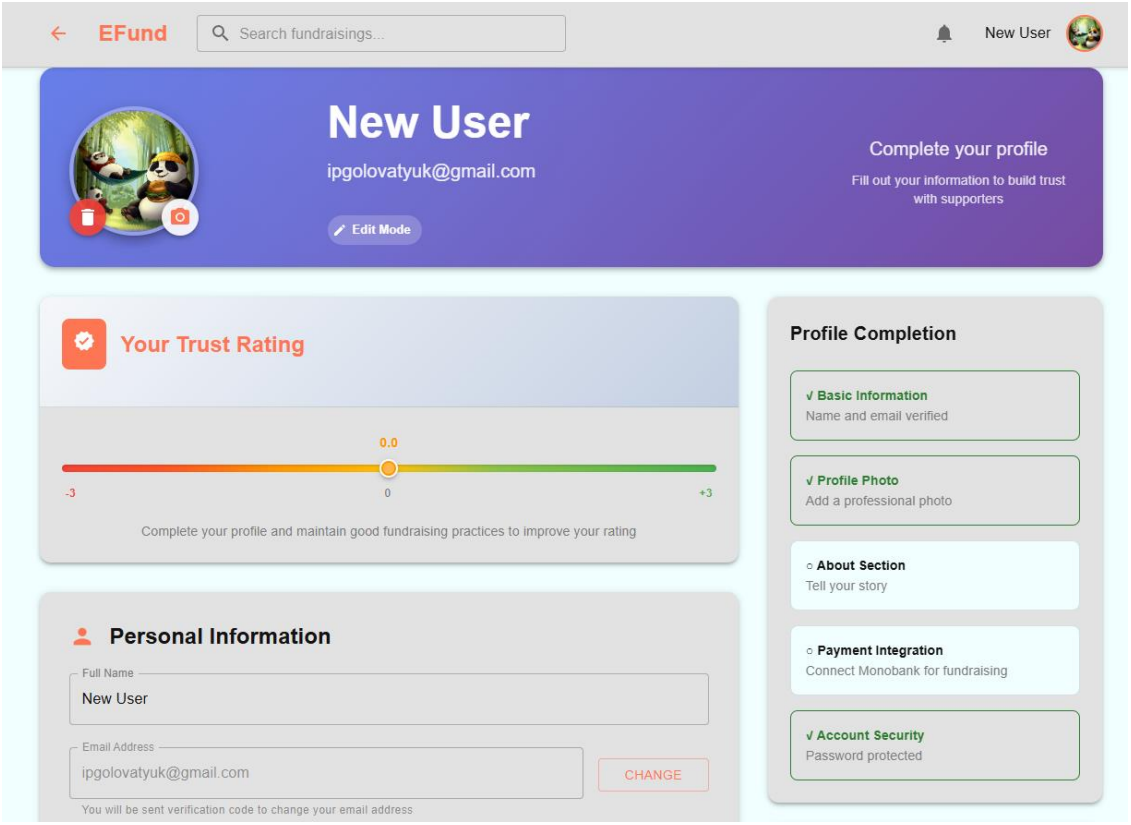


Рисунок 4.12 – Сторінка редагування профілю користувача

Для того, щоб почати створювати фандрейзинги, потрібно під’єднати особистий аккаунт monobank до застосунку (рисунок 4.13), щоб мати можливість підв’язувати «банки» до зборів

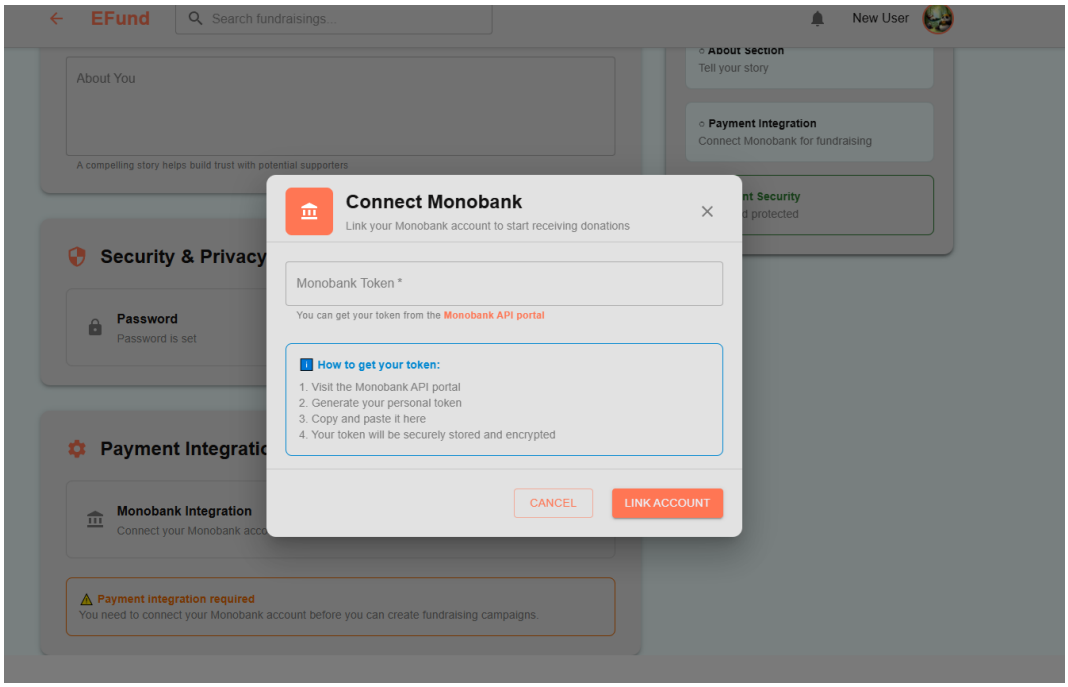


Рисунок 4.13 – Форма під’єднання особистого аккаунту monobank

Після цього користувач отримує змогу створювати власні фандрейзинги заповнивши форму (рисунок 4.14)

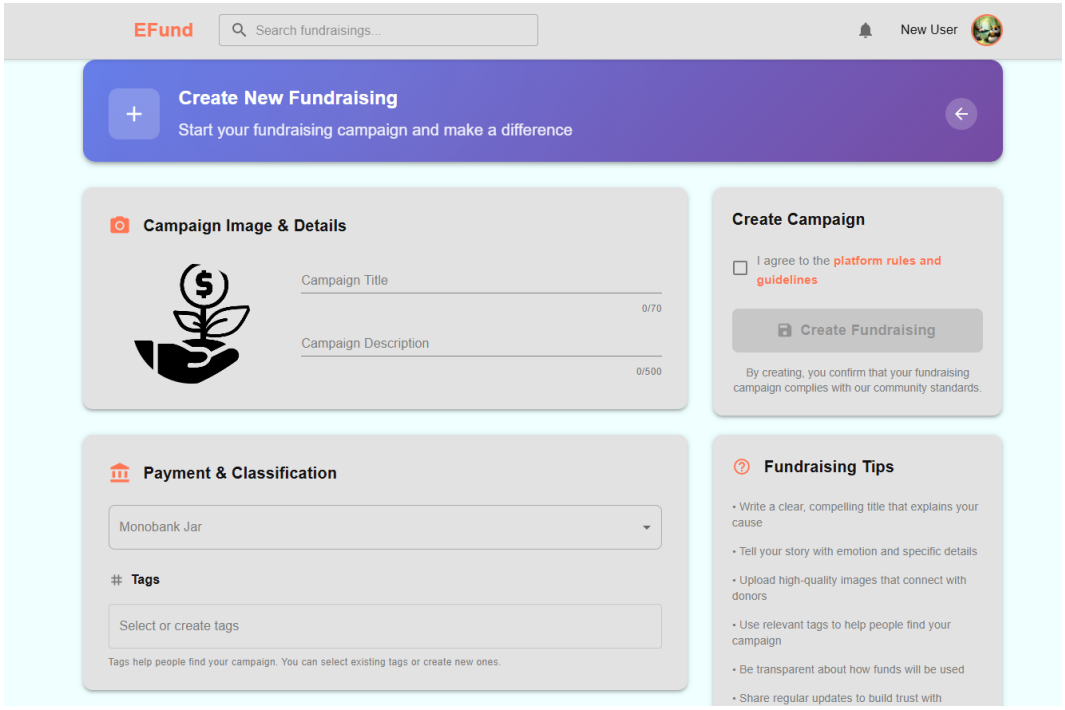


Рисунок 4.14 – Форма створення нового фандрейзингу

Знайти створенні збори користувач може у секції «My Fundraisings», де він може побачити деталі свої зборів (рисунок 4.15) та перейти на їх редагування (рисунок 4.16)

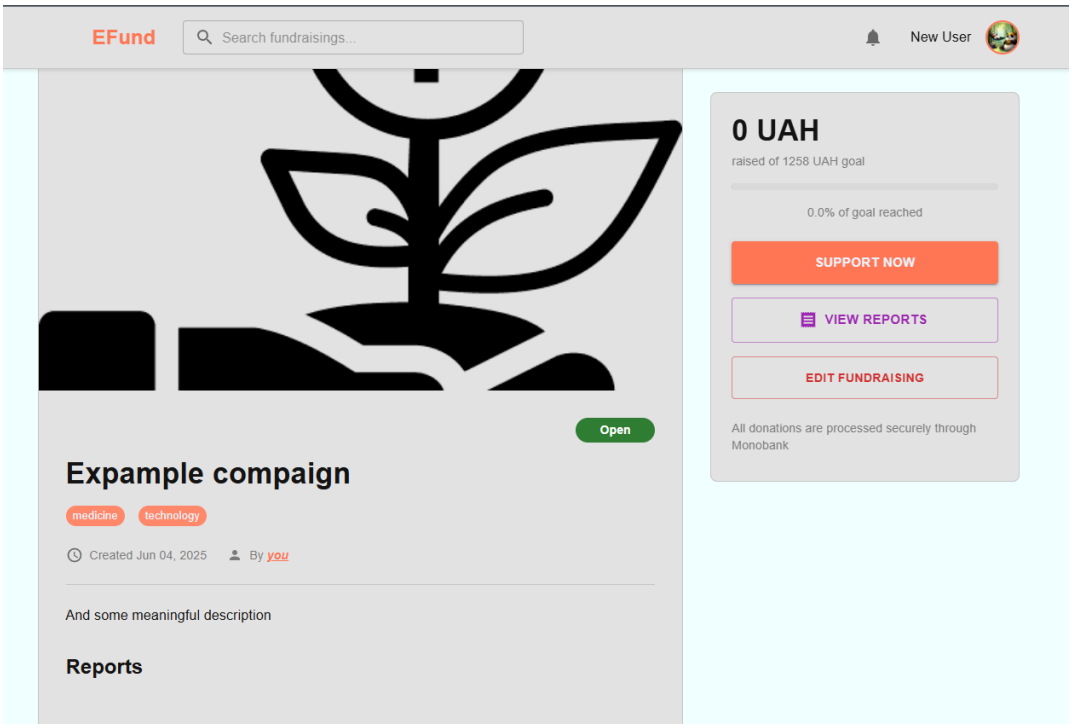


Рисунок 4.15 – Деталі особистого збору

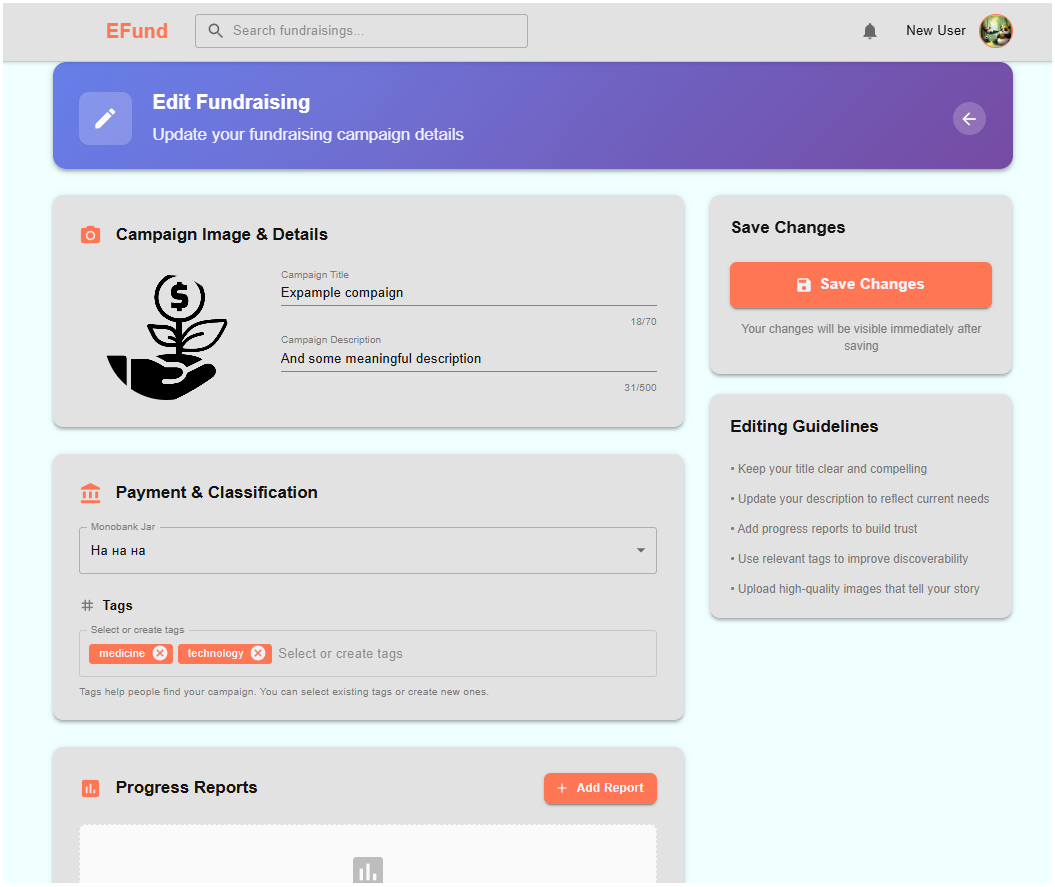


Рисунок 4.16 – Форма редагування фандрейзингу

Також, у користувачів з правами адміністратора, є можливість бачити всіх користувачів, зареєстрованих в системі у меню «Users» (рисунок 4.17)

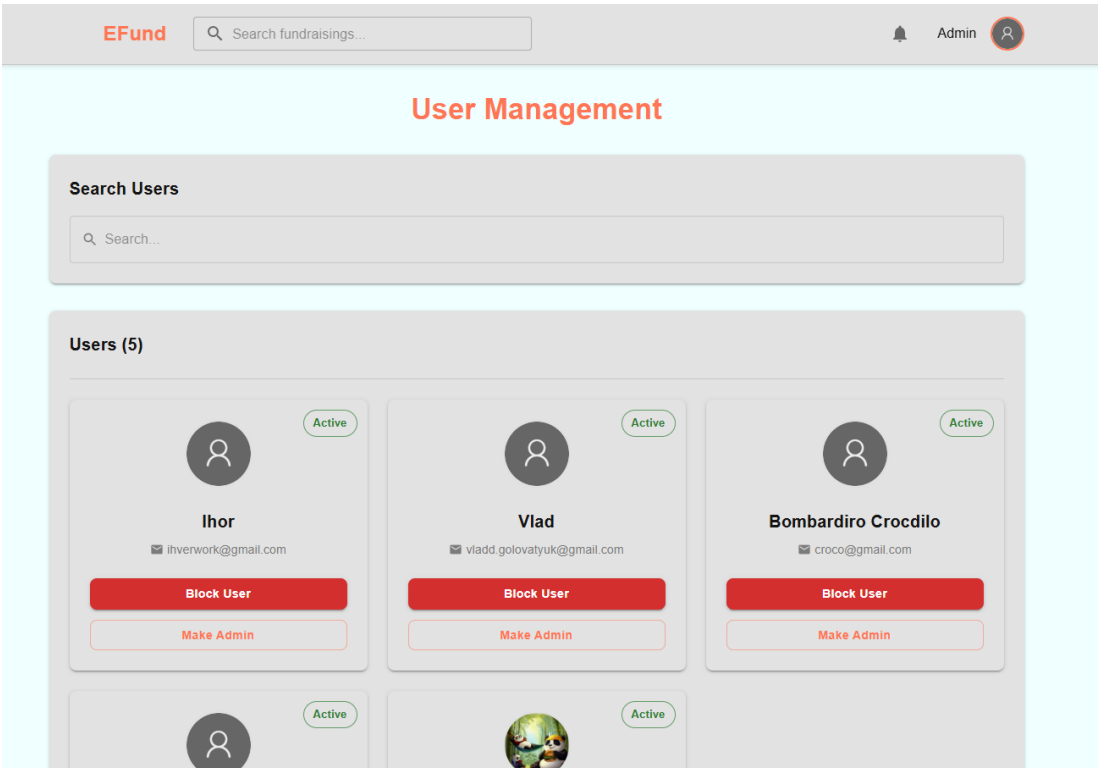


Рисунок 4.17 – Сторінка усіх користувачів системи

Адміністрація також повинна переглядати надіслані користувачами скарги на створенні фандрейзинги, щоб приймати рішення відповідно до скарги (рисунок 4.18)

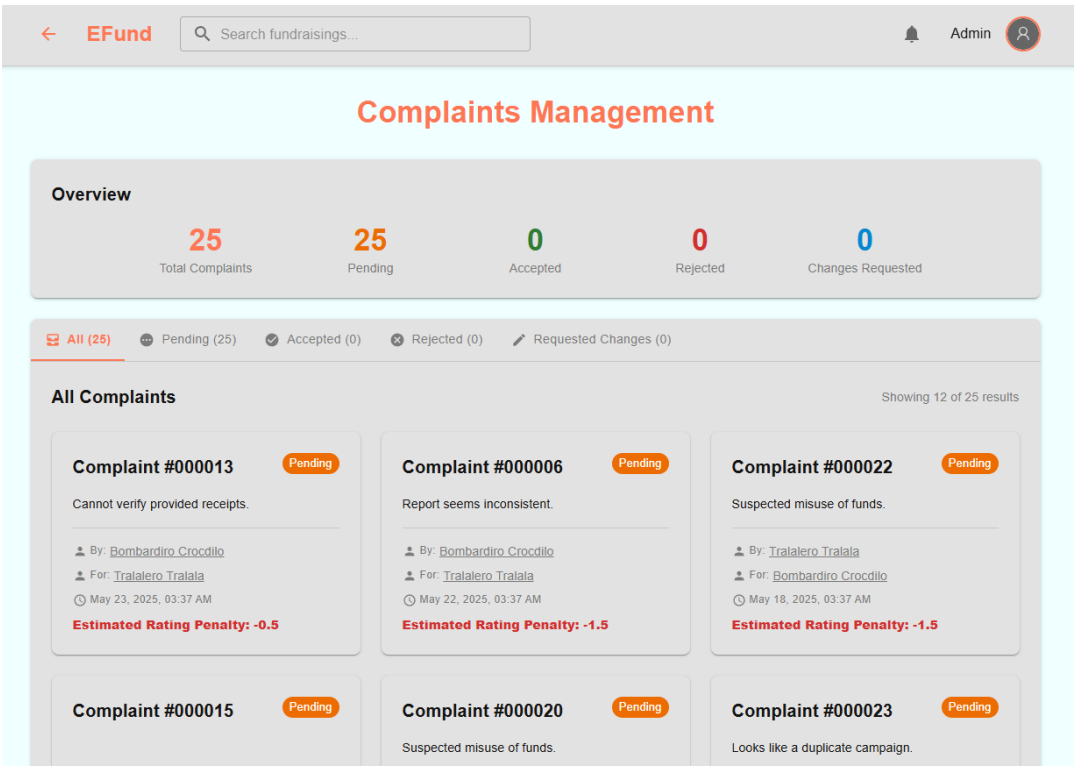


Рисунок 4.18 – Список скарг від користувачів

Натиснувши на скаргу, адміністратор може переглянути її деталі, і приймати відповідне рішення у вигляді зняття рейтингу довіри автора збору, або письмове попередження з проханням виправити порушення (рисунок 4.19)

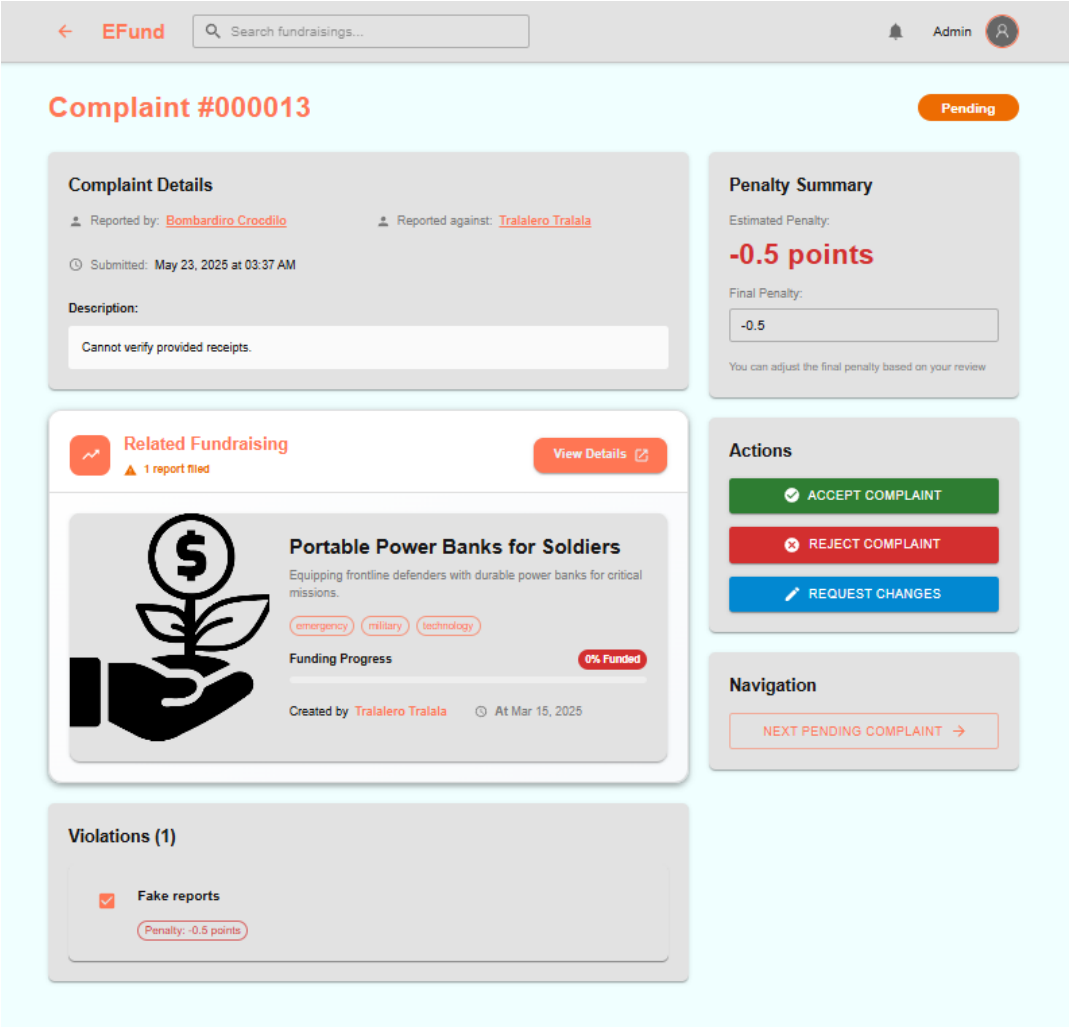


Рисунок 4.19 – Деталі скарги з можливістю залишити свій вердикт

Висновки до розділу

В рамках розділу було наведено контрольний приклад до розробленого програмного забезпечення, в рамках якого було наведено повний функціонал, що включає реєстрацію, пошук та фільтрація фандрейзингів, перегляд їх детальних інформацій, процес створення власних зборів та їх редагування. А також функціонал адміністрування, який включає в себе керування користувачами та перегляд їх скарг на існуючі фандрейзинги, задля прийняття рішення за скаргою.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Процес розгортання програмного забезпечення фандрейзингової платформи EFund включає кілька послідовних етапів, які забезпечують правильну конфігурацію всіх компонентів системи та її готовність до роботи. Основними компонентами системи є база даних Microsoft SQL Server, серверна частина застосунку на базі ASP.NET Core, клієнтська частина застосунку на React, а також інфраструктура, яка реалізована за допомогою Docker [20] та Docker Compose.

Після встановлення Docker та Docker Compose використовується файл docker-compose.yml, який містить усі необхідні налаштування для запуску системи. У цьому файлі описані три основні сервіси – база даних, серверна частина застосунку та клієнтська частина застосунку. Визначено образи контейнерів, порти, змінні середовища, а також залежності між сервісами.

Початковим етапом є налаштування бази даних Microsoft SQL Server. У конфігураційному файлі визначено сервіс db, який використовує офіційний образ Microsoft SQL Server. Вказуються порти для з’єднання із сервісом, встановлюються змінні середовища, зокрема пароль адміністратора та згода з умовами ліцензії. Цей сервіс є критично важливим, оскільки серверна частина застосунку залежить від його доступності під час запуску.

Наступним кроком є конфігурація серверної частини застосунку. Сервіс api збирається локально з використанням Dockerfile, який складається з кількох послідовних шарів. Спочатку створюється базовий шар на основі офіційного образу ASP.NET 9.0, у якому визначено робочу директорію та відкриті порти для обробки вхідних запитів. Потім застосунок збирається на основі SDK-образу, у якому послідовно копіюються всі проєкти рішення, виконується відновлення залежностей та складання застосунку у режимі Release.

Після цього відбувається публікація зібраного застосунку, копіюються необхідні конфігураційні файли, шаблони листів, вихідні дані для ініціалізації бази, а також створюються системні директорії для зберігання логів, файлів користувачів, звітів, вкладень та іншої інформації, яка використовується у процесі роботи застосунку. Останнім кроком є формування фінального шару, який об’єднує все опубліковане та забезпечує запуск серверної частини застосунку за допомогою команди dotnet EFund.WebAPI.dll.

Також, у складі docker-compose визначено сервіс client, який відповідає за клієнтську частину застосунку, реалізовану на React. Цей сервіс будується на базі Node.js-образу та виконує збірку інтерфейсу користувача у вигляді статичних ресурсів, що розміщуються на вебсервері. У конфігурації вказуються необхідні порти для обробки HTTP-запитів з боку користувачів, а також залежність від серверної частини застосунку, що гарантує доступність API під час старту клієнтської частини. Клієнтська частина застосунку взаємодіє із серверною через HTTP-запити, використовуючи відповідну адресу, яка передається через змінні середовища.

Особливу увагу приділено налаштуванню взаємозв’язків між сервісами. У docker-compose.yml це реалізовано за допомогою параметра depends_on, який гарантує правильний порядок запуску: база даних запускається першою, далі серверна частина застосунку, після чого — клієнтська частина застосунку. Це забезпечує стабільну взаємодію компонентів системи.

Після завершення конфігурації усіх компонентів виконується команда docker-compose up, яка автоматично завантажує образи, створює та запускає контейнери відповідно до заданої конфігурації. У процесі запуску Docker Compose створює мережу між сервісами та забезпечує їхню взаємодію, згідно із вказаними параметрами. Завдяки параметру restart: always, як серверна частина застосунку, так і клієнтська частина застосунку автоматично перезапускаються у разі збою або перезапуску системи, що підвищує загальну надійність платформи.

5.2 Супровід програмного забезпечення

Після завершення роботи над певним компонентом і внесення змін до репозиторію, автоматично запускається CI/CD пайплайн, налаштований у GitHub Actions. Цей процес відповідає за автоматичне тестування, компіляцію та збирання програмного забезпечення.

Застосунок збирається у вигляді Docker-образів, які публікуються в Docker Hub або інше обране сховище контейнерів. Це дозволяє централізовано зберігати збірки та забезпечує зручний доступ до них для подальшого розгортання як на локальних серверах, так і в хмарних середовищах.

Оновлення застосунку для користувачів відбувається максимально просто: достатньо завантажити нові образи контейнерів та виконати їх розгортання через Docker Compose. Усі налаштування компонентів уже прописані у конфігураційному файлі, тому немає потреби у ручному втручанні під час оновлення системи.

Висновки до розділу

У розділі було детально розглянуто процес розгортання фандрейзингової платформи EFund із використанням Docker та Docker Compose. Було описано поетапне налаштування ключових компонентів системи, зокрема бази даних Microsoft SQL Server, серверної частини застосунку на базі ASP.NET Core та клієнтської частини на React. Особливу увагу приділено конфігурації взаємодії між сервісами, механізмам запуску контейнерів та організації файлової структури в межах Dockerfile.

Окремо висвітлено принципи супроводу та оновлення програмного забезпечення за допомогою CI/CD пайплайнів у GitHub Actions, які автоматизують збірку, тестування та публікацію Docker-образів до централізованого сховища. Розгортання нових версій для користувачів значно спрощено завдяки готовим конфігураціям у docker-compose.yml, що усуває потребу у ручному налаштуванні кожного компонента.

Таким чином, застосування Docker-інфраструктури у поєднанні з автоматизованими CI/CD процесами забезпечує ефективне, надійне та масштабоване розгортання платформи EFund як у локальних, так і у хмарних середовищах.

ВИСНОВКИ

У ході виконання було розроблено програмне забезпечення для автоматизації створення та супроводу благодійних зборів із можливістю інтеграції платіжних систем, ведення звітності та адміністративного контролю, що забезпечує прозорість фандрейзингу, зручність для користувачів і ефективне управління платформою.

В першому розділі дипломного проєкту наведено аналіз предметного середовища фандрейзингових онлайн-платформ, порівняльний аналіз між реалізованим програмним застосунком та аналогами і виділено переваги та недоліки перед кожним з них, з використанням BPMN-нотації наведено 4 основних бізнес-процеси, що будуть реалізовані в дипломному проєкті.

В другому розділі наведено основні функціональні та нефункціональні вимоги до програмного забезпечення та сформульовано системні вимоги до нього, а також наведено результати аналізу економічних показників ПЗ та його приблизну собівартість.

В третьому розділі наведено архітектуру програмного забезпечення, наведено обґрунтування прийнятих основних архітектурних рішень та застосування використаних засобів розробки.

В четвертому розділі наведено опис контрольного прикладу.

В п’ятому розділі наведено результати порівняльного аналізу можливих способів розгортання програмного застосунку, наведено обґрунтування обраного варіанту.

У ході тестування розробленого програмного забезпечення встановлено, що мета дипломного проєкту, а саме автоматизація та спрощення процесів створення, ведення та перевірки фандрейзингових кампаній з інтеграцією з банківськими сервісами та системою звітності, досягнута. Реалізована система дозволяє користувачам ефективно організовувати збори, звітувати про їх перебіг, а адміністрації — забезпечувати контроль і прозорість процесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) European Center for Not-for-Profit Law. The Potential and Risks of Using Digital Technologies in Fundraising: A Comparative Research [Електронний ресурс]. – Hague : ECNL, 2021. – 40 с. – Режим доступу: https://ecnl.org/sites/default/files/2021-05/ECNL%20Comparative%20research%20on%20digital%20fundraising%202021%20FINAL.pdf

2) Khoma N. Crowdfunding and Fundraising in the Peacebuilding System: Ukraine's Case [Електронний ресурс] / N. Khoma // Lithuanian Annual Strategic Review. – 2023. – Vol. 20. – P. 53–75. – Режим доступу: https://journals.lka.lt/journal/lasr/article/2100/file/pdf

3) European Center for Not-for-Profit Law. Good Practices in Digital Fundraising [Електронний ресурс]. – Hague : ECNL, 2021. – 32 с. – Режим доступу: https://ecnl.org/sites/default/files/2021-12/ECNL%20Good%20Practices%20of%20Digital%20Fundraising_0.pdf

4) monobank API [Електронний ресурс]. – Режим доступу: https://api.monobank.ua/docs/

5) Kickstarter [Електронний ресурс]. – Режим доступу: https://www.kickstarter.com/

6) GoFundMe [Електронний ресурс]. – Режим доступу: https://www.gofundme.com/

7) Fundly [Електронний ресурс]. – Режим доступу: https://fundly.com/

8) monobank [Електронний ресурс]. – Режим доступу: https://www.monobank.ua/

9) Bass L., Clements P., Kazman R. Software Architecture in Practice [Електронний ресурс] / L. Bass, P. Clements, R. Kazman. – 3rd ed. – Boston : Addison-Wesley, 2012. – 624 с. – Режим доступу: https://ptgmedia.pearsoncmg.com/images/9780321815736/samplepages/0321815734.pdf

10) Microsoft. N-Tier Architecture Styles [Електронний ресурс] // Microsoft Docs. – 2023. – Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier

11) Jones M. B., Bradley J., Sakimura N. JSON Web Token (JWT) [Електронний ресурс] : RFC 7519 / M. B. Jones, J. Bradley, N. Sakimura // The Internet Engineering Task Force (IETF). – 2015. – 32 с. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7519

12) .NET [Електронний ресурс]. – Режим доступу: https://dotnet.microsoft.com/en-us/

13) ASP.NET Core [Електронний ресурс]. – Режим доступу: https://dotnet.microsoft.com/en-us/apps/aspnet

14) Entity Framework [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/ef/

15) C# [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/

16) TypeScript [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org/

17) React [Електронний ресурс]. – Режим доступу: https://react.dev/

18) ASP.NET Identity [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/identity/

19) Axios [Електронний ресурс]. – Режим доступу: https://axios-http.com/docs/intro

20) Docker [Електронний ресурс]. – Режим доступу: https://www.docker.com/

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

StrikePlagiarism.com

Дата звіту 6/9/2025
Дата редагування 6/9/2025

Документ прийнятий

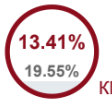
Звіт подібності

метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-11_Веремчук_Головатюк_ПЗ
Автор
ІП-11_Веремчук_Головатюк
Науковий керівник / Експерт
Ліщук К.І.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

10	7620	60097
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2025 р.

Фандрайзингова платформа EFund (комплексна тема)

Керівництво користувача

КПІ.ІП-1103.ІП-1106.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Катерина ЛІЩУК

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавці:
_____ Ігор ВЕРЕМЧУК
_____ Владислав ГОЛОВАТЮК

Київ – 2025

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«EFund» – це вебзастосунок, що забезпечує створення, управління та супровід фандрейзингових кампаній із підтримкою інтеграції з банківськими сервісами, зокрема monobank. Користувачі можуть створювати збори, додавати по них звіти та прикріпляти до звітів файли. Функціонал також включає фільтрацію кампаній за статусами, тегами або авторами, надсилання скарг та перегляд звітності.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Мінімальні вимоги до настільних комп’ютерів та ноутбуків:

- двоядерний процесор із базовою частотою від 1.6 ГГц;
- не менше 2 ГБ оперативної пам’яті;
- стабільне підключення до мережі зі швидкістю від 5 Мбіт/с;
- веббраузери останніх версій: Google Chrome, Mozilla Firefox, Microsoft Edge або Opera.

Рекомендовані вимоги для повного користувацького досвіду:

- чотириядерний процесор із частотою від 2.4 ГГц;
- 4 ГБ або більше оперативної пам’яті;
- стабільне підключення до мережі зі швидкістю від 20 Мбіт/с.

2.2 Завантаження застосунку

Застосунок запускається локально за допомогою Docker, використовуючи файл Docker Compose, який визначає конфігурацію всіх необхідних сервісів.

2.3 Перевірка коректної роботи

Після завершення завантаження застосунку у браузері повинна відобразитися головна сторінка з описом доступного функціоналу вебсайту. Успішне завантаження інтерфейсу свідчить про коректну роботу сервісу. У разі проблем зі з’єднанням або відображенням сторінки слід перевірити підключення до Інтернету або сумісність браузера.

3 ВИКОНАННЯ ПРОГРАМИ

При переході на вебсайт користувач направляється на головну сторінку з описом функціоналу вебсайту (рисунок 3.1).

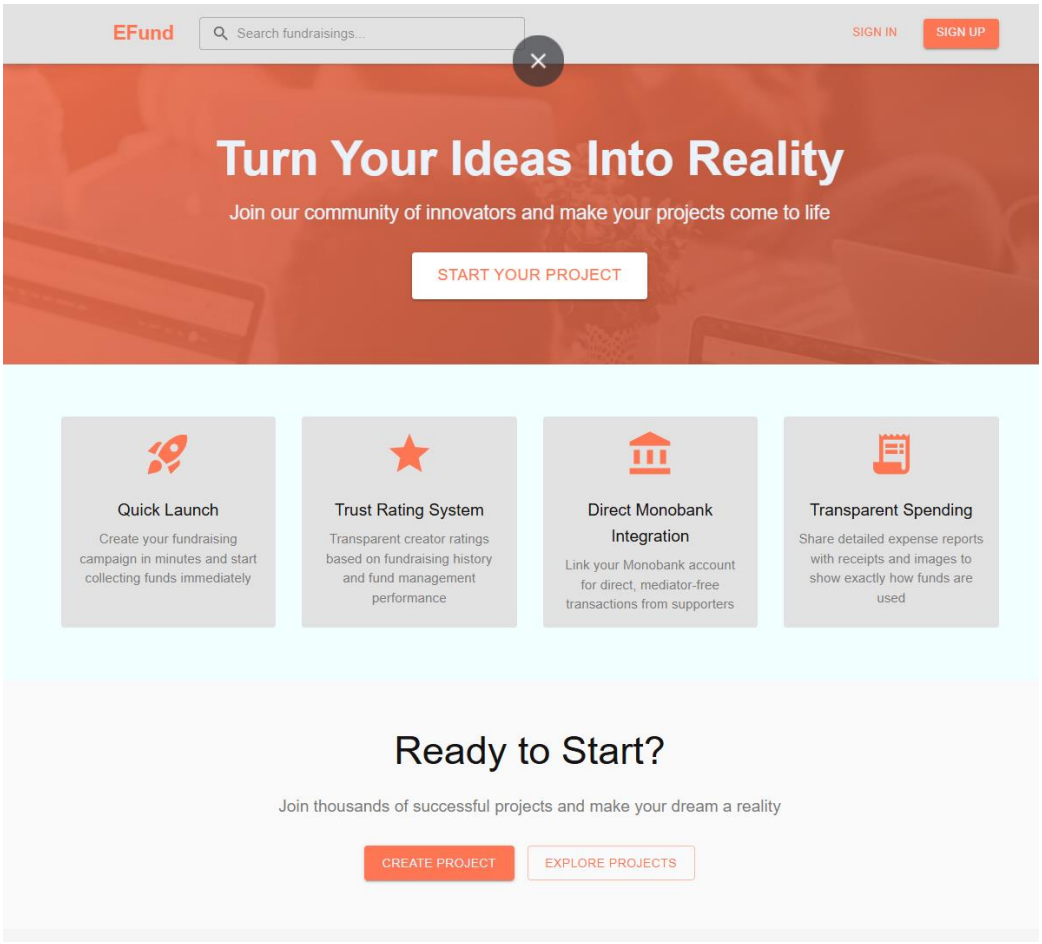


Рисунок 3.1 – Головна сторінка

З головної сторінки користувач може потрапити на загальний пошук фандрейзингів з секцією фільтрів пошуку (рисунок 3.2) та результатами пошуку (рисунок 3.3)

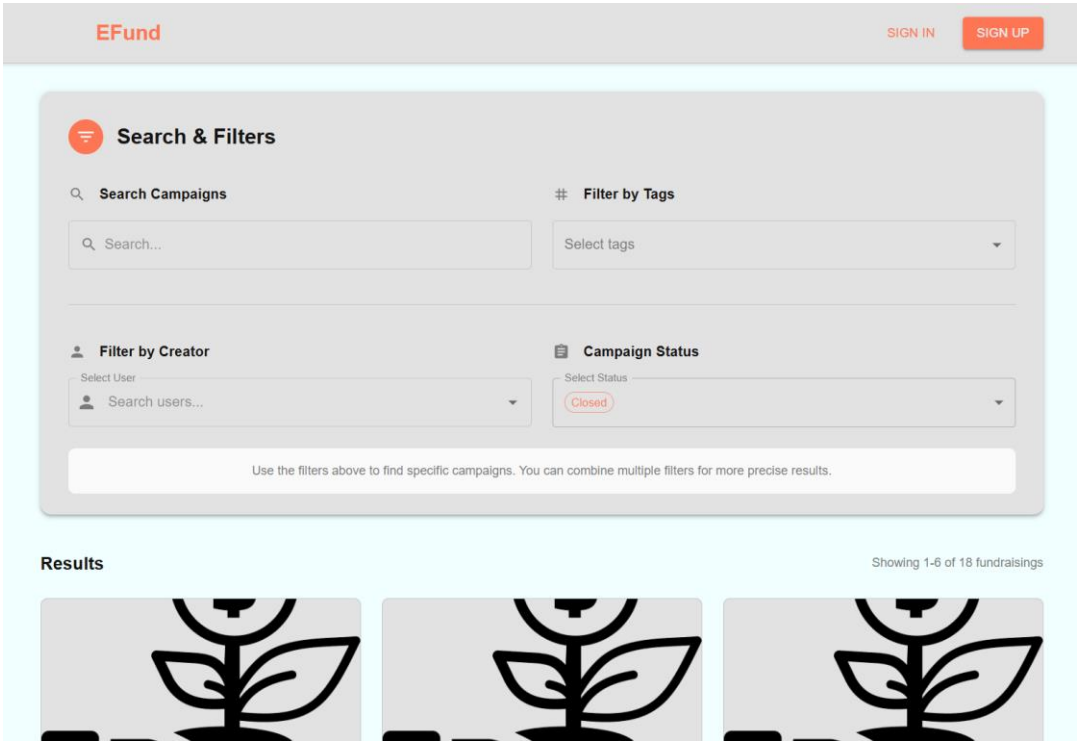


Рисунок 3.2 – Секція фільтрів пошуку

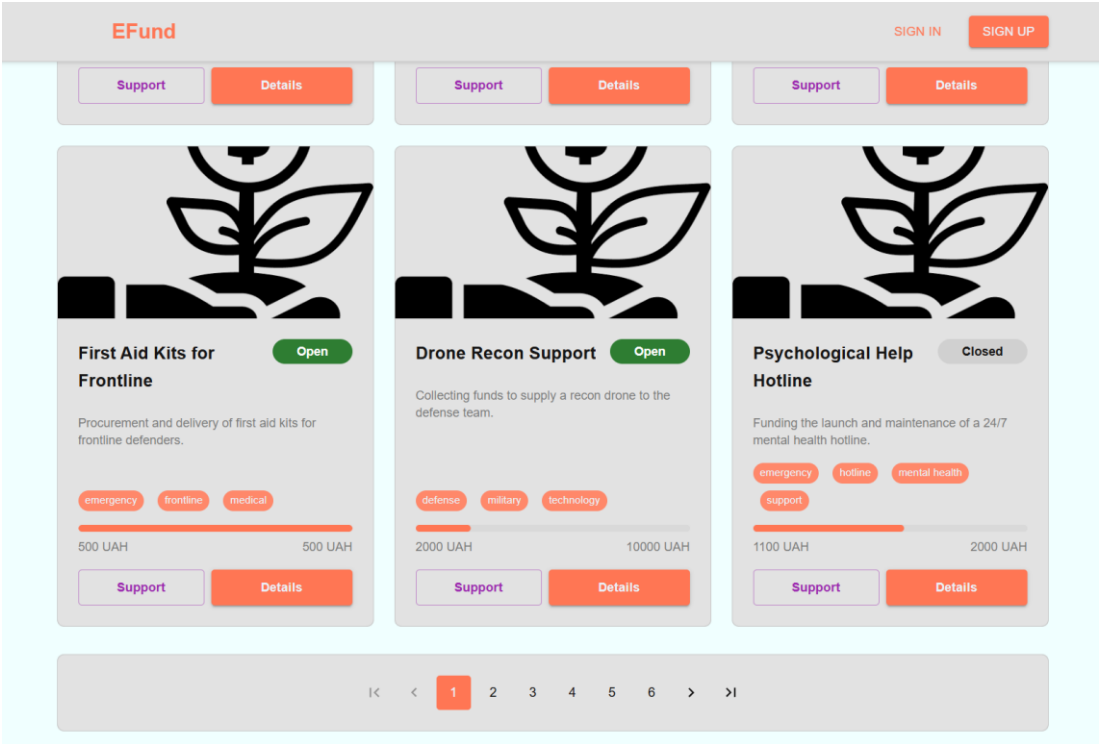


Рисунок 3.3 – Результати пошуку

Користувач може відкрити деталі будь якого фандрейзингу натиснувши на відповідну кнопку «Details» на картці фандрейзингу (рисунок 4.4)

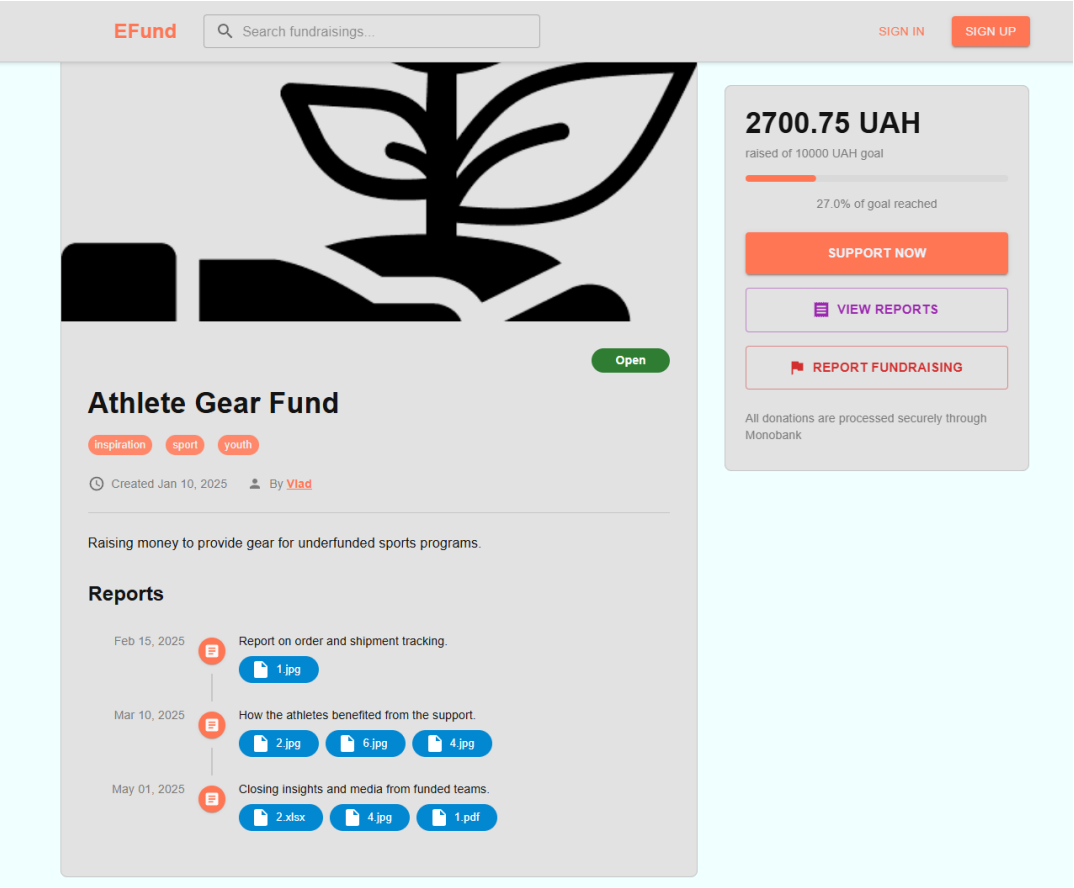


Рисунок 3.4 – Деталі

На сторінці деталей, користувач може бачити опис, дату створення, автора, теги фандрайзингу, а також секцію звітування по цьому збору.

На цій сторінці користувач може перейти до банки monobank, щоб поповнити збір власними коштами (рисунок 3.5), а також має можливість поскаржитись на збір, якщо вважає, що фандрейзинг порушує внутршні правила створення зборів (рисунок 3.6)

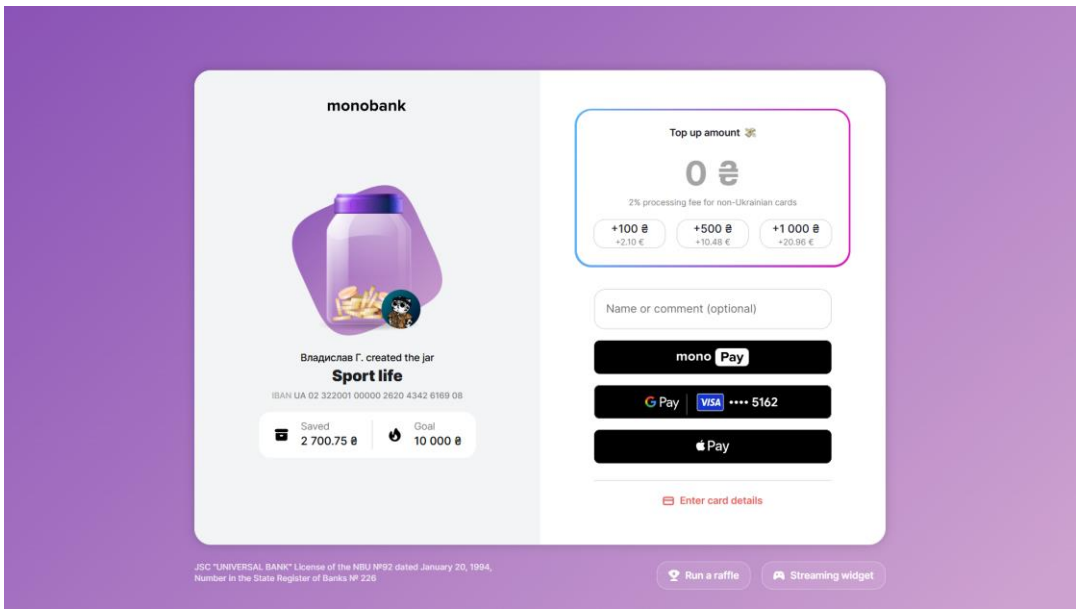


Рисунок 3.5 – Сторінка поповнення збору на сайті monobank

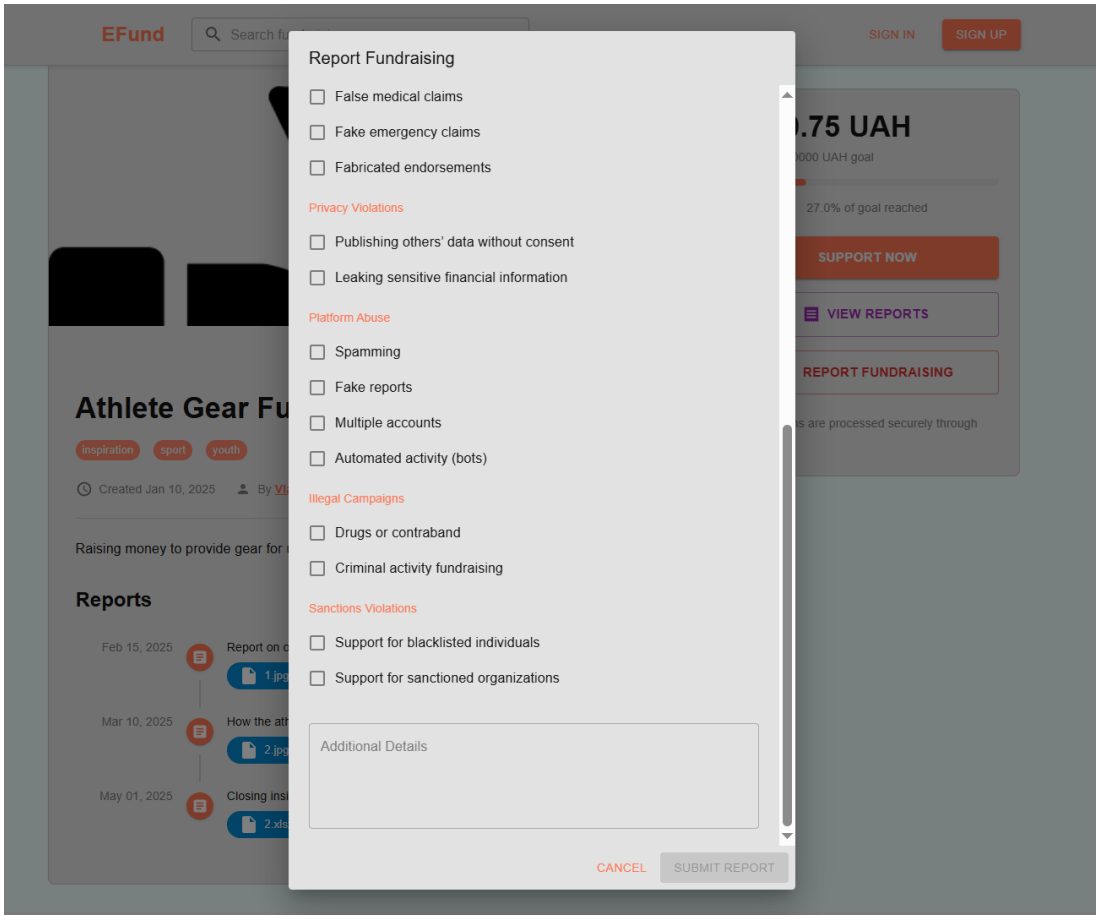


Рисунок 3.6 – Вікно створення скарги на фандрейзинг

Також, зі сторінки деталей фандрейзингу, користувач має змогу побачити короткий опис профіля автора збору (рисунок 3.7), який включає

ім’я, пошту та поточний рейтинг користувача, та перейти на профіль автора збору для більш детальної інформації (рисунок 3.8)

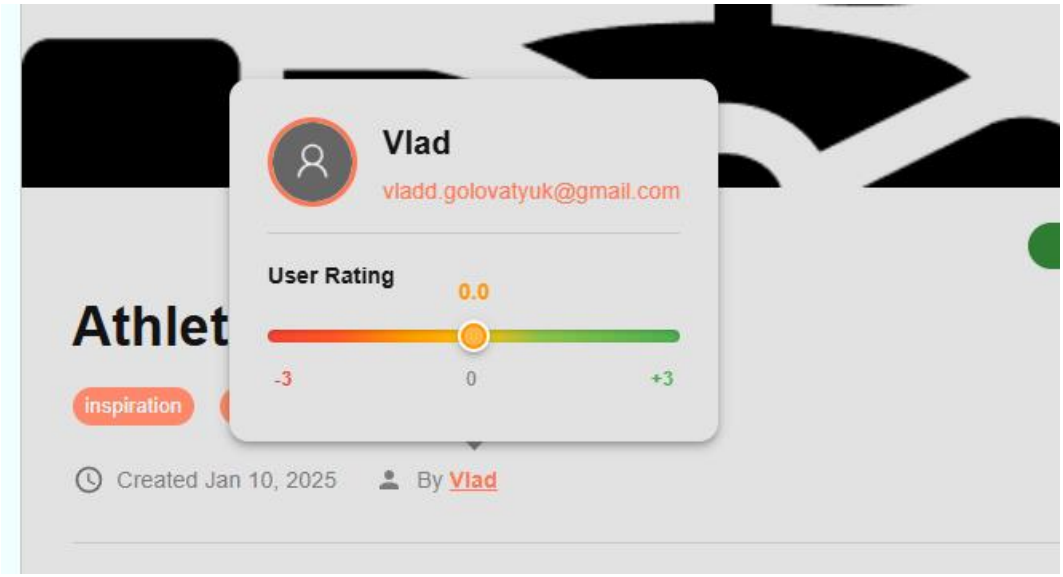


Рисунок 3.7 – Короткий опис профіля автора збору

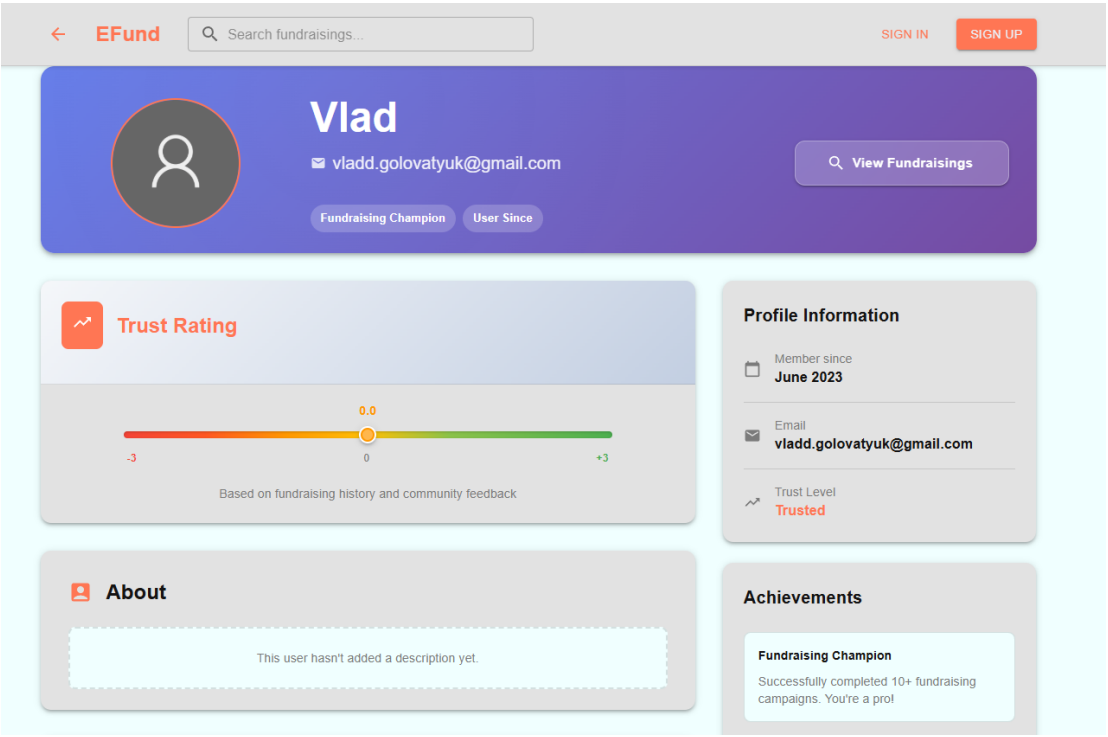


Рисунок 3.8 – Детальний опис профілю користувача

У разі бажання користувача створювати власні збори, у нього є можливість зареєструватись, натиснувши відповідну кнопку «SIGN UP» на верхній частині застосунку (рисунок 3.9)

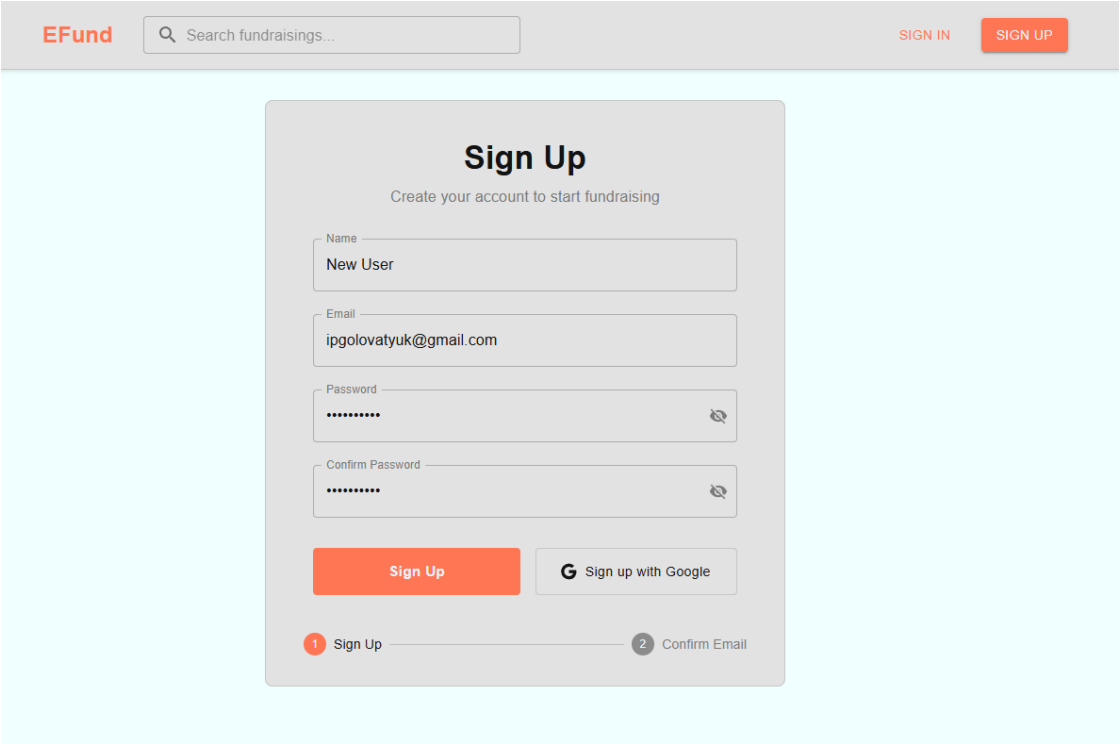


Рисунок 3.9 – Форма реєстрації

Після введеня даних користувача, на його пошту надходе лист з кодом підтвердження (рисунок 3.10), який потрібно ввести на наступний етап реєстрації, підтвердження пошти (рисунок 3.11)

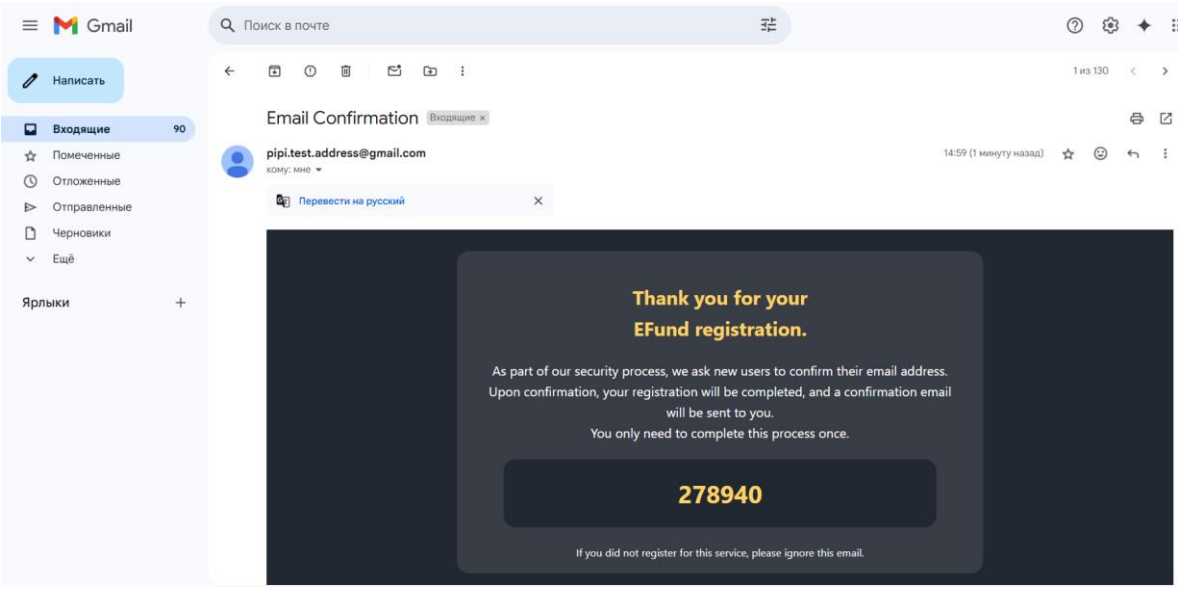


Рисунок 3.10 – Лист з кодом підтвердження пошти

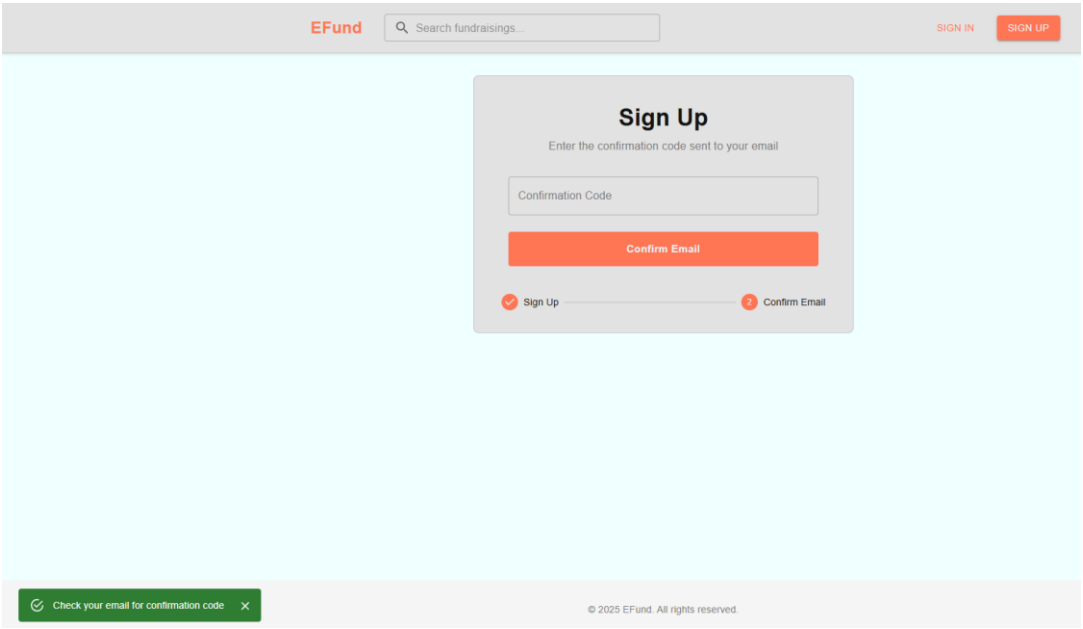


Рисунок 3.11 – Форма підтвердження пошти користувача

Після підтвердження пошти, користувач має перейти в редагування власного профілю (рисунок 3.12)

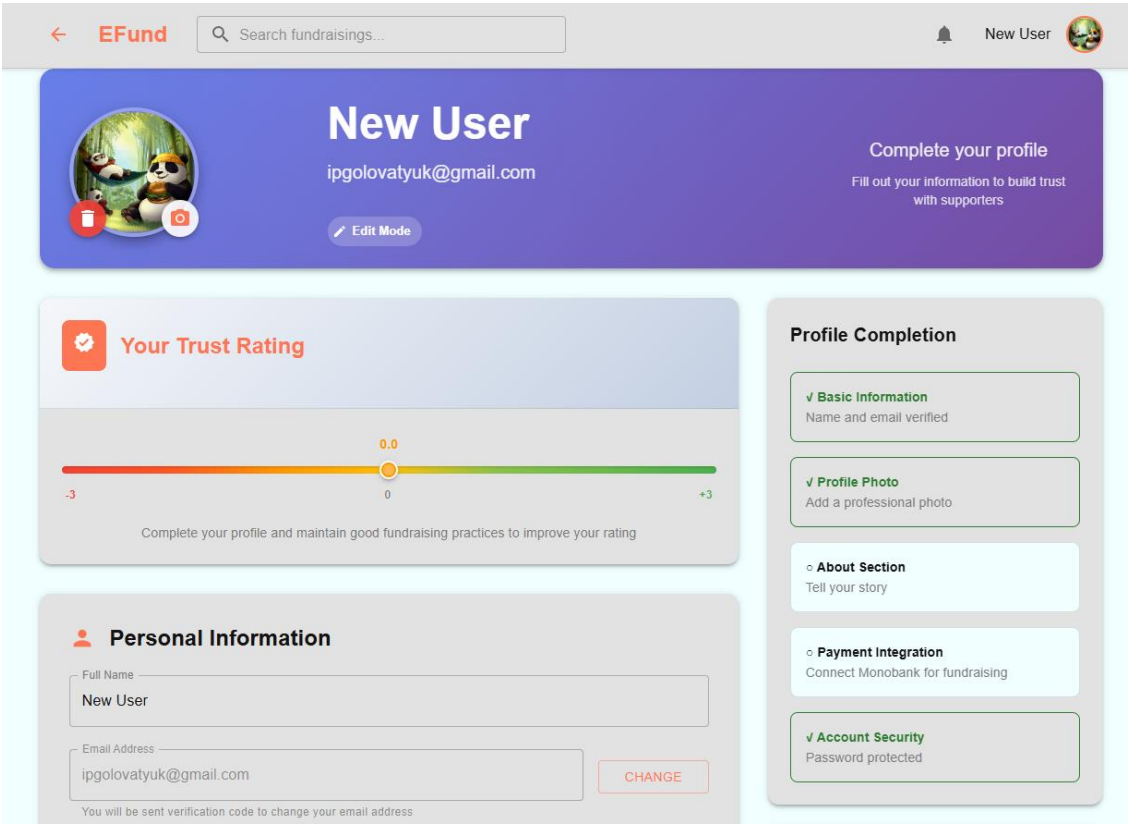


Рисунок 3.12 – Сторінка редагування профілю користувача

Для того, щоб почати створювати фандрейзинги, потрібно під’єднати особистий аккаунт monobank до застосунку (рисунок 3.13), щоб мати можливість підв’язувати «Банки» до зборів

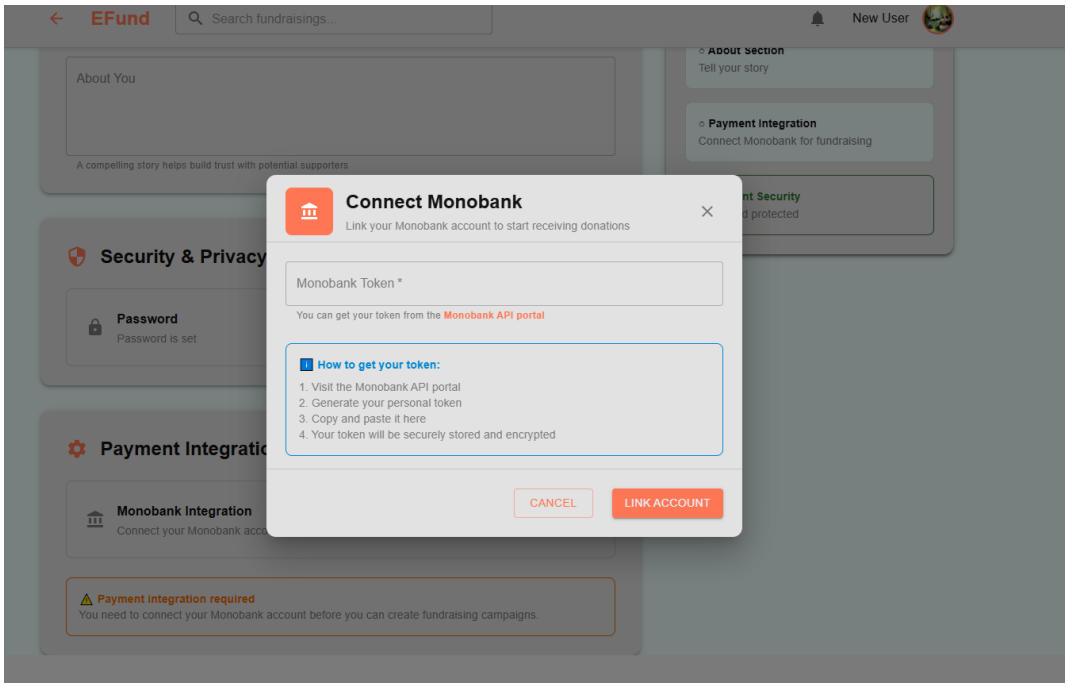


Рисунок 3.13 – Форма під’єднання особистого аккаунту monobank

Після цього користувач отримує змогу створювати власні фандрейзинги заповнивши форму (рисунок 3.14)

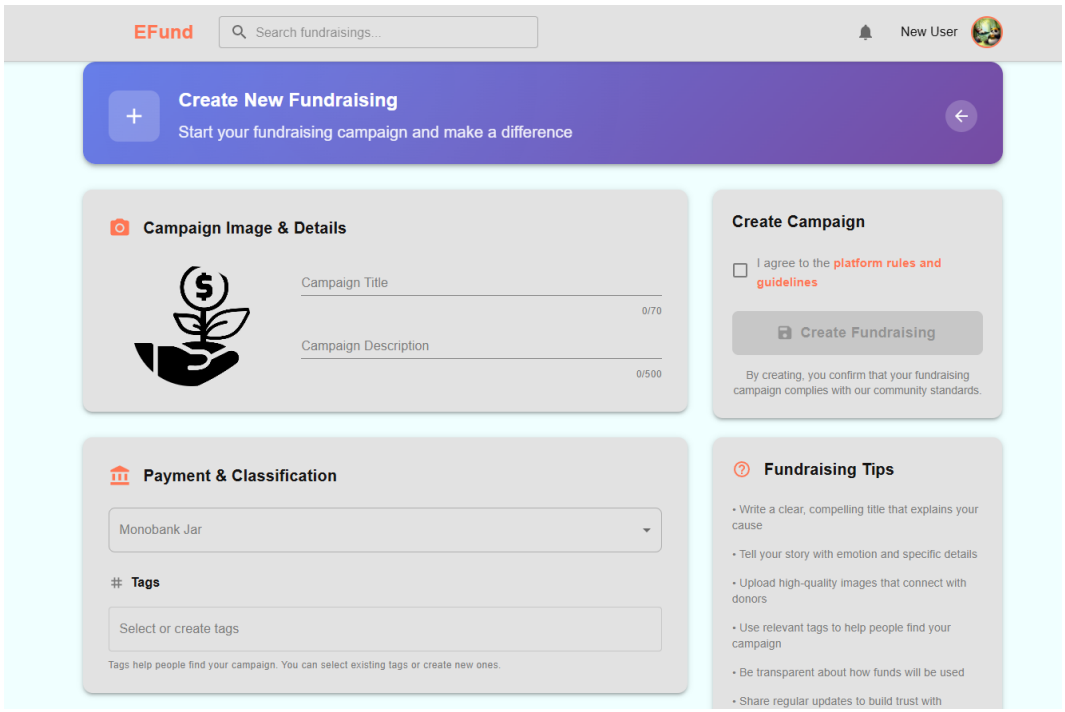


Рисунок 3.14 – Форма створення нового фандрейзингу

Знайти створенні збори користувач може у секції «My Fundraisings», де він може побачити деталі свої зборів (рисунок 3.15) та перейти на їх редагування (рисунок 3.16)

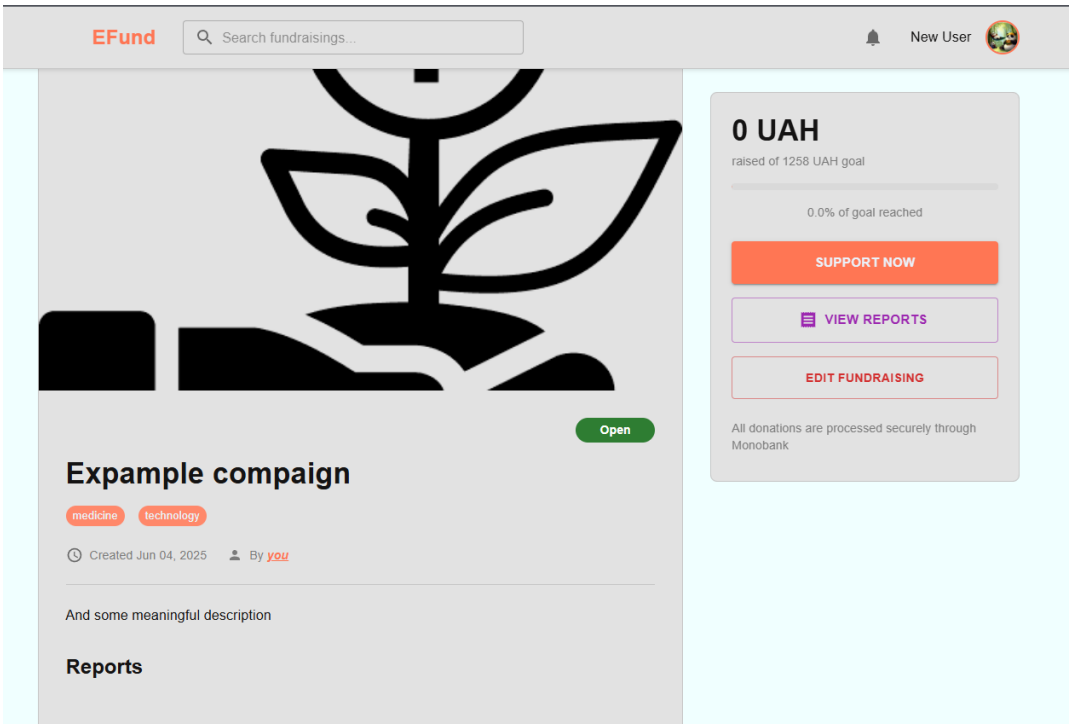


Рисунок 3.15 – Деталі особистого збору

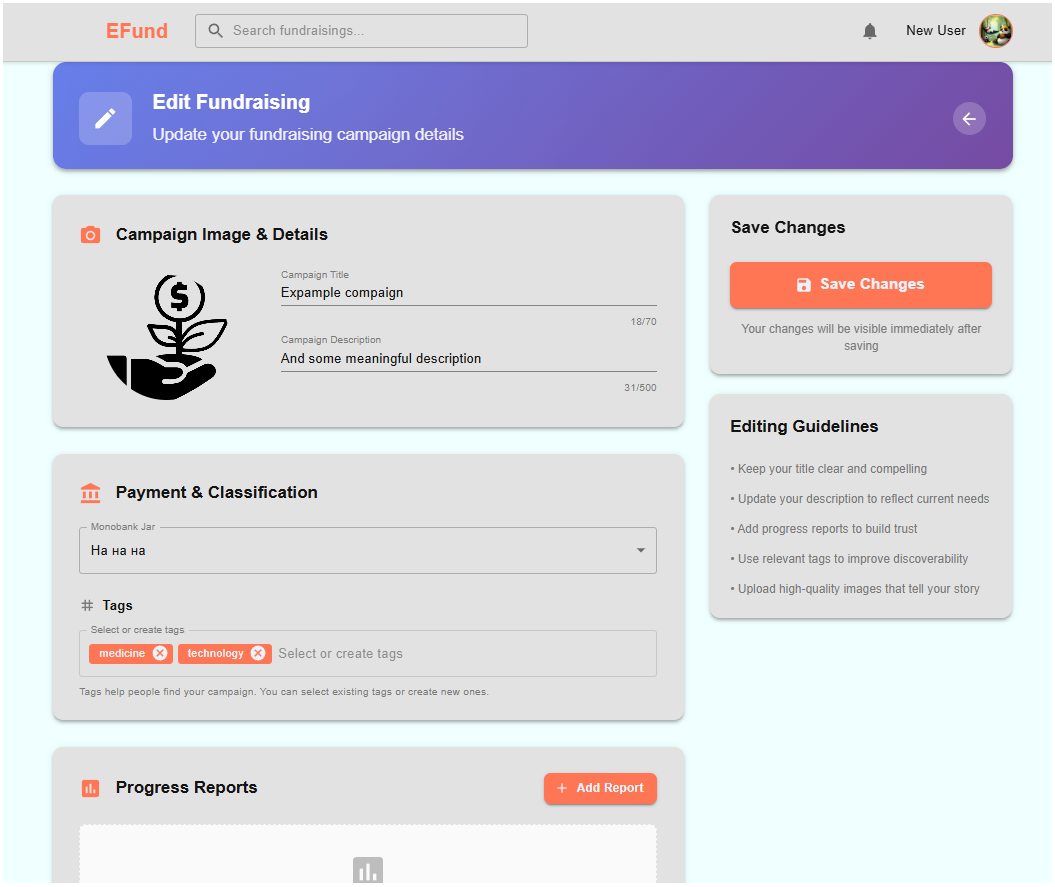


Рисунок 3.16 – Форма редагування фандрейзингу

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2025 р.

Фандрайзингова платформа EFund (комплексна тема)

Графічний матеріал

КПІ.ІП-1103.ІП-1106.045440.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Катерина ЛІЩУК

Нормоконтроль:

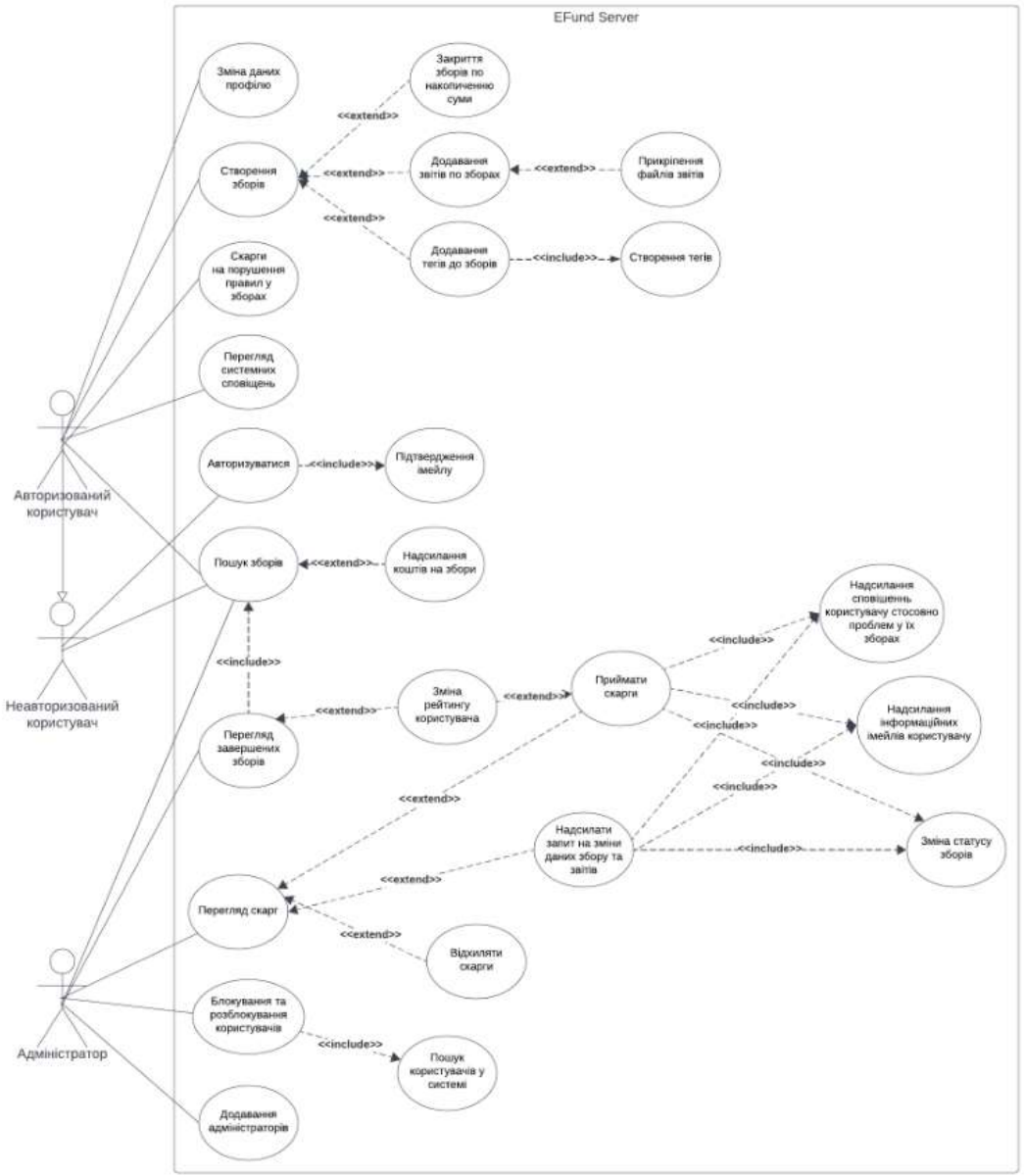
_____ Катерина ЛІЩУК

Виконавці:

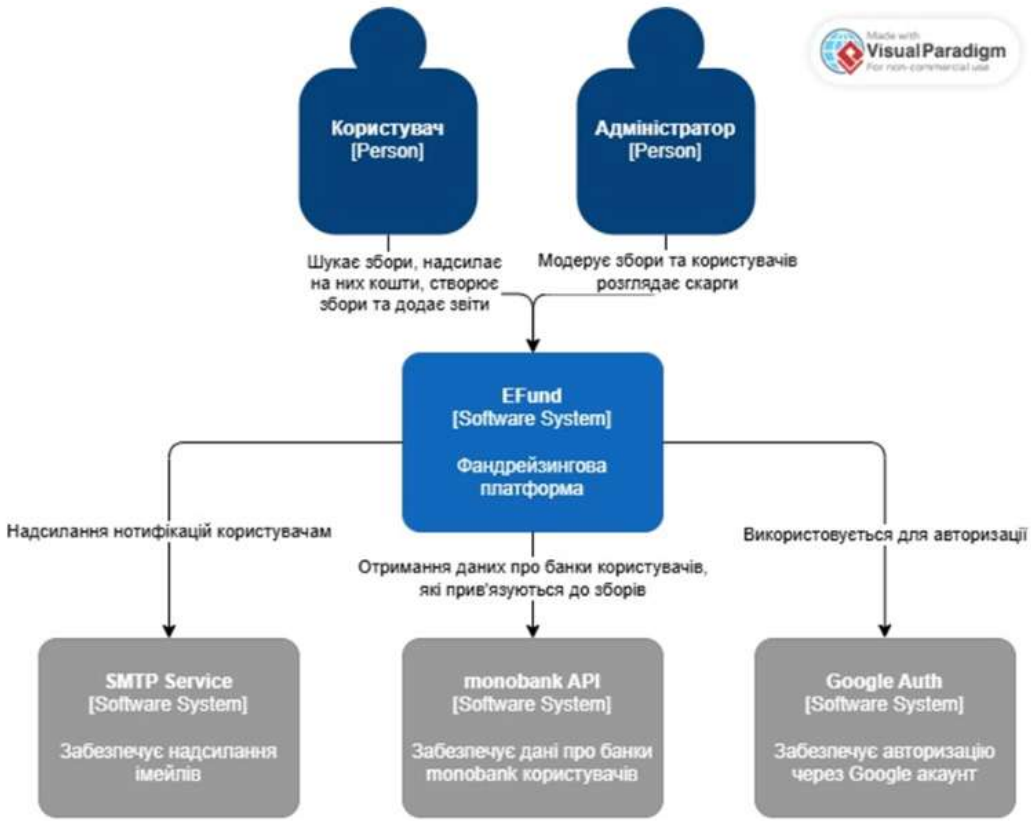
_____ Ігор ВЕРЕМЧУК

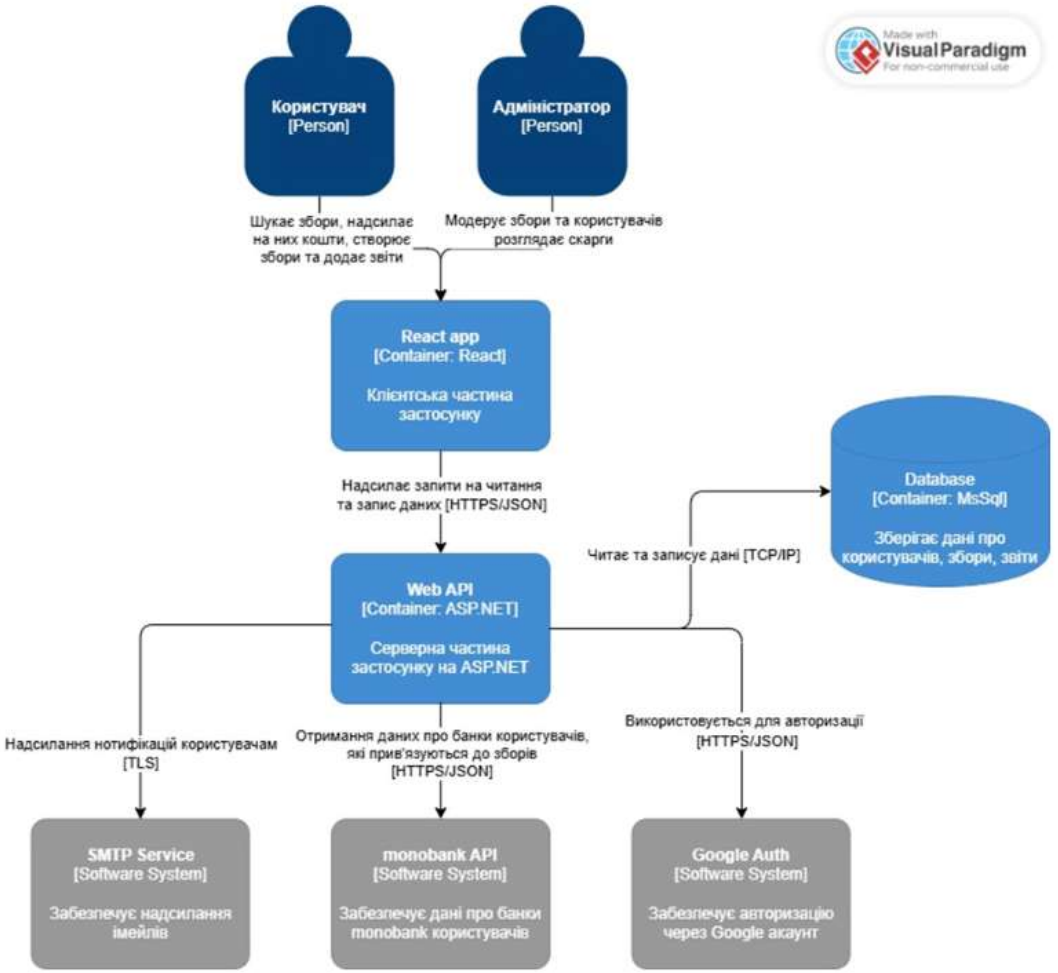
_____ Владислав ГОЛОВАТЮК

Київ – 2025



					КПІ.ІП-1103.ІП-1106.045440.06.99.ССВ	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використань	Літера / Маса / Масштаб
Розробив		Веремчук І. І.				
		Головатюк В. І.				
Перевірив		Ліщук К.І.				Аркуш 1 / Аркушів 4
Т. контр.					Фандрайзингова платформа EFund (комплексна тема)	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11
Н. контр.		Ліщук К.І.				
Затвердив		Жаріков Е.В.				



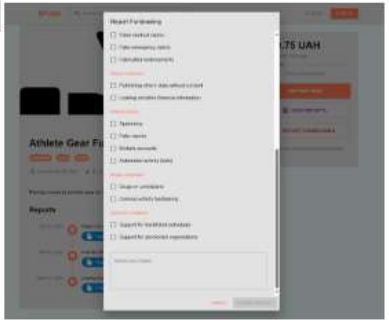


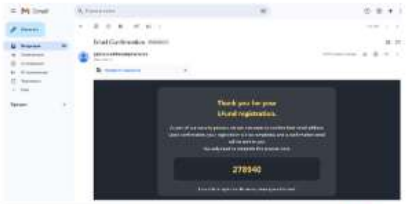
					КПІ.ІП-1103.ІП-1106.045440.06.99.ССМ	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Веремчук І. І.				
		Головатюк В. І.				
Перевірив		Ліщук К.І.				Аркуш 2 / Аркушів 4
Т. контр.					Фандрайзингова платформа EFund (комплексна тема)	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11
Н. контр.		Ліщук К.І.				
Затвердив		Жаріков Е.В.				

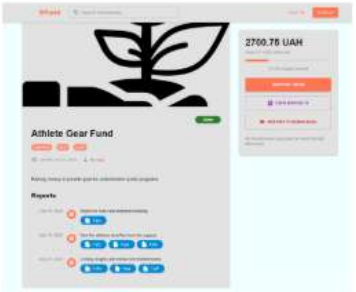
КПІ.ІП-1103.ІП-1106.045440.06.99.СБД

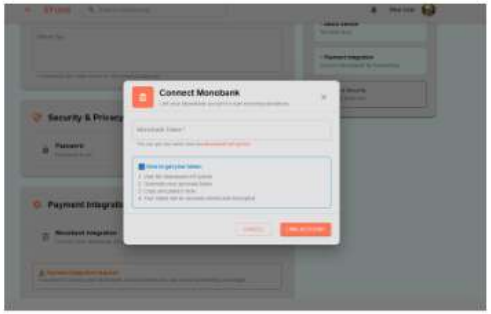
					КПІ.ІП-1103.ІП-1106.045440.06.99.СБД	
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних	Літера / Маса / Масштаб
Розробив		Веремчук І. І.				
		Головатюк В. І.				
Перевірив		Ліщук К.І.				Аркуш 3 / Аркушів 4
Т. контр.					Фандрайзингова платформа EFund (комплексна тема)	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11
Н. контр.		Ліщук К.І.				
Затвердив		Жаріков Е.В.				

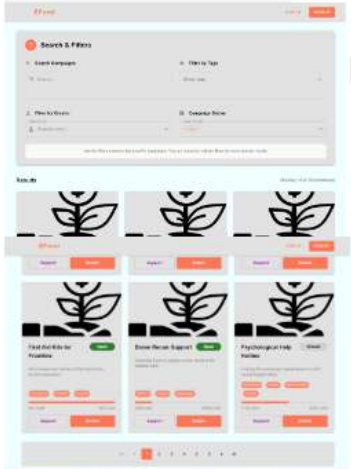
КПІ.ІП-1103.ІП-1106.045440.06.99.КЕ

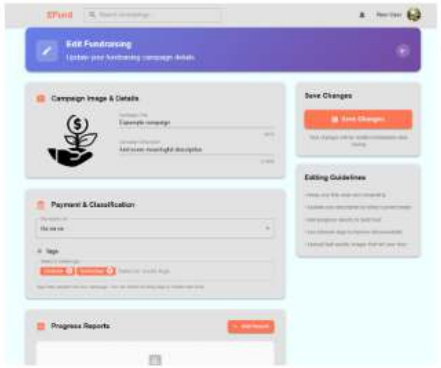


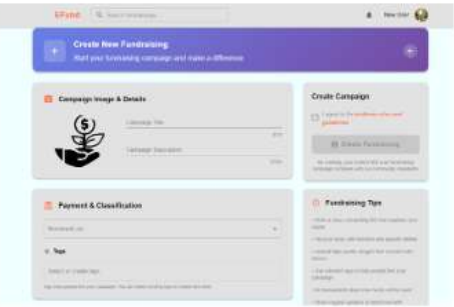


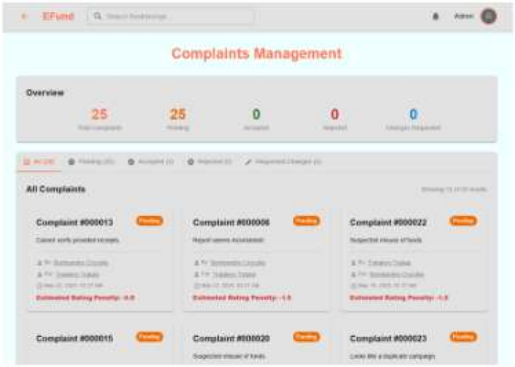


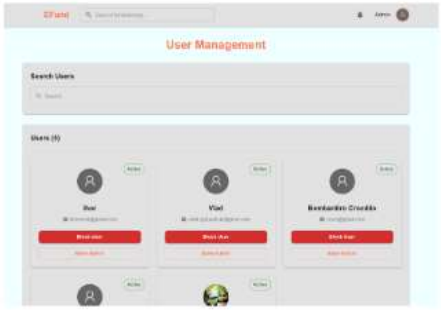


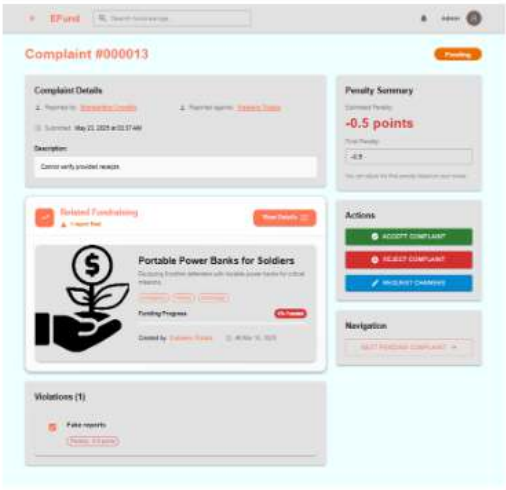


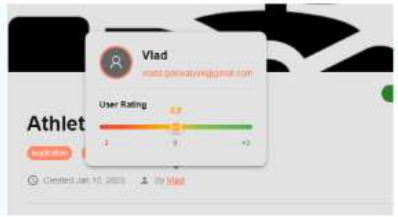


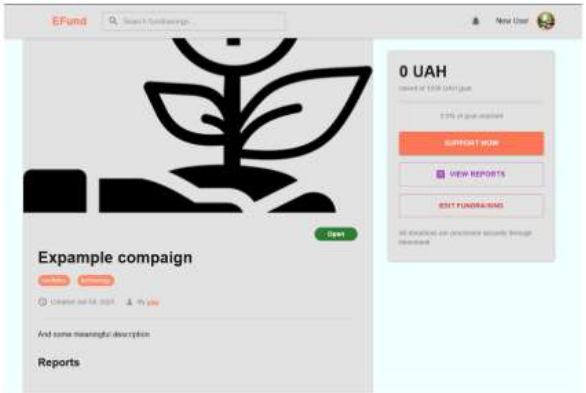












					КПІ.ІП-1103.ІП-1106.045440.06.99.КЕ	
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм	Літера / Маса / Масштаб
Розробив		Веремчук І. І.				
		Головатюк В. І.				
Перевірив		Ліщук К.І.				Аркуш 4 / Аркушів 4
Т. контр.						
Н. контр.		Ліщук К.І.			Фандрайзингова платформа EFund (комплексна тема)	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-11
Затвердив		Жаріков Е.В.				