МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:

Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д. е. н., проф. П. М. Грицюк

_____

«____» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА НА ЗДОБУТТЯ ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ «БАКАЛАВР»

Проектування та розробка користувацької частини веб-платформи для краудфандингу соціальних проєктів

Виконав:
здобувач вищої освіти за ОПП «Інформаційні системи та технології» спеціальності 126 «Інформаційні системи та технології», групи ІСТ-41
Гарват Костянтин Олександрович

Керівник:
д.е.н., професор Грицюк П.М.

Рецензент:
к.е.н., доцент Волошин В. С.

Рівне – 2025

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 73с., 55 рис., 2 табл., додатки на 20 стор., 20 літературних джерел.

Актуальність теми даної бакалаврської роботи полягає в тому, що в сучасному суспільстві краудфандинг став одним із провідних інструментів для фінансування соціально важливих ініціатив, включаючи освітні, культурні, медичні та волонтерські проєкти. Особливої ваги він набув у період повномасштабної війни в Україні, коли громадянське суспільство активно використовує онлайн-платформи для підтримки Збройних Сил, евакуації постраждалих, забезпечення гуманітарних потреб. Такий контекст вимагає надійних, гнучких та зручних у користуванні веб-рішень, що забезпечують прозорість збору коштів і ефективну взаємодію з донорами. Саме тому розробка сучасної користувацької частини краудфандингової платформи з урахуванням технологічних вимог і реальних сценаріїв використання є актуальним напрямом дослідження.

Об’єкт дослідження бакалаврської роботи – краудфандинг соціальних проєктів, тобто процес колективного фінансування ініціатив соціального спрямування за допомогою спеціалізованих онлайн-платформ, що об’єднують потенційних благодійників, волонтерів та організаторів проєктів з метою досягнення суспільно корисних результатів.

Предметною областю бакалаврської роботи є веб-платформа краудфандингу, зокрема її функціональні компоненти, архітектура та принципи роботи, а також методи і засоби побудови зручного, інтуїтивно зрозумілого та ефективного користувацького інтерфейсу, який забезпечує взаємодію між користувачами платформи.

Метою бакалаврської роботи є проєктування та розробка користувацької частини веб-платформи для краудфандингу соціальних проєктів з використанням сучасних технологій веб-розробки, зокрема фреймворка Angular, а також аналіз доцільності його використання у

порівнянні з іншими популярними інструментами, такими як React і jQuery, на основі прикладного функціоналу реальної системи.

У бакалаврській роботі було: здійснено аналіз предметної області, визначений набір інструментів, що сприяють проектування і створення інформаційної системи, розроблено проект інформаційної системи, який включає в себе зображення функціональної структури; реалізовано інтерфейс веб-платформи для краудфандингу з використанням Angular (адміністративна частина) та jQuery (донорська частина), з підтримкою Peer-to-Peer кампаній; створено приклад конфігурації кампанії збору коштів, адаптованої до українського контексту, включаючи її візуальні, функціональні та аналітичні елементи.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, КРАУДФАНДИНГ, ВЕБ-ПЛАТФОРМА, DESKTOP-ЗАСТОСУНОК, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС.

ЗМІСТ

ВСТУП

У сучасному суспільстві стрімко зростає роль громадянських ініціатив, соціальної взаємодії та незалежного фінансування через цифрові канали. Платформи для краудфандингу стали важливим інструментом, що дозволяє небайдужим громадянам підтримувати гуманітарні, культурні, освітні та військові проєкти. Однією з найактуальніших моделей є Peer-to-Peer (P2P) кампанії, які дозволяють кожному користувачу самостійно ініціювати збір коштів у рамках загальної мети.

Особливої актуальності ці платформи набули в умовах повномасштабної війни в Україні, коли оперативність збору коштів на потреби армії, волонтерських ініціатив чи допомоги переселенцям є критично важливою. У таких умовах надзвичайно важливою стає зручна та функціональна користувацька частина платформи, що дозволяє донорам швидко взаємодіяти з системою та здійснювати внески.

На практиці у веб-платформах часто поєднуються різні підходи до розробки інтерфейсу. Наприклад, донорські частини багатьох краудфандингових систем реалізуються за допомогою jQuery або інших легких бібліотек, тоді як адміністративні модулі – із застосуванням фреймворків, таких як Angular. У цій роботі було реалізовано умовний інтерфейс користувача з підтримкою Peer-to-Peer кампаній на основі таких технологій.

Метою бакалаврської роботи є проектування та розробка користувацької частини веб-платформи для краудфандингу соціальних проєктів з урахуванням сучасних вимог до UX/UI, масштабованості та технологічної ефективності.

Об’єктом дослідження є краудфандинг соціальних проєктів. **Предметом** дослідження є веб-платформа краудфандингу, зокрема її функціональні

компоненти, архітектура та принципи роботи реалізовані за допомогою Angular та jQuery, а також архітектурні рішення для підтримки Peer-to-Peer кампаній.

Завданнями бакалаврської роботи є:

- проаналізувати предметну область краудфандингу та сучасні моделі його реалізації;
- дослідити функціональні особливості Peer-to-Peer кампаній на прикладі платформи PayBee;
- здійснити порівняльний аналіз Angular та React як технологій для створення користувацьких інтерфейсів, а також обґрунтувати вибір Angular як основного фреймворка для розробки;
- спроєктувати функціональну структуру користувацької частини платформи та реалізувати інтерфейс користувача з урахуванням UX/UI принципів;
- сформувати вихідні інтерфейсні елементи та протестувати їхню взаємодію в системі.

Інструменти, використані під час виконання роботи: Angular, jQuery, TypeScript, JavaScript, HTML/CSS, Bootstrap, Visual Studio Code, Git, Google Chrome DevTools.

РОЗДІЛ 1. ІНФОРМАЦІЙНА ПІДТРИМКА ФУНКЦІОНУВАННЯ СОЦІАЛЬНИХ ПРОЄКТІВ

1.1. Поняття краудфандингу

Краудфандинг (англ. crowdfunding) – це форма колективного фінансування, яка передбачає добровільне надання коштів великою кількістю людей (donors, backers) для підтримки проєктів, ідей або ініціатив через інтернет-платформи. Сам термін утворений із двох англійських слів: crowd (натовп, громада) і funding (фінансування), що буквально означає «фінансування громадою» [1].

Перші краудфандингові кампанії з'явилися на початку 2000-х років. Однією з перших відомих платформ стала ArtistShare (2003 рік), яка дозволяла фанатам фінансувати записи музичних альбомів. Згодом на ринку з'явилися такі платформи, як Kickstarter, Indiegogo, GoFundMe, що значно розширили сферу застосування краудфандингу. У сучасному вигляді краудфандинг перетворився на глобальну галузь із мільярдними обігами, охоплюючи не лише творчі ініціативи, а й бізнес, медицину, соціальні проєкти, волонтерство та політичну активність.

Класична модель краудфандингу передбачає участь чотирьох ключових сторін: автора кампанії (ініціатора), платформи (як технічного середовища для розміщення та адміністрування кампаній), донорів (користувачів, які здійснюють внески), а також механізму звітності або винагород. Успіх краудфандингової кампанії залежить від прозорості умов, довіри до організаторів, якості подачі інформації та ефективності використання соціальних мереж.

Залежно від очікувань і мотивації донорів, розрізняють кілька основних типів краудфандингу (рис. 1.1) [3, 4].

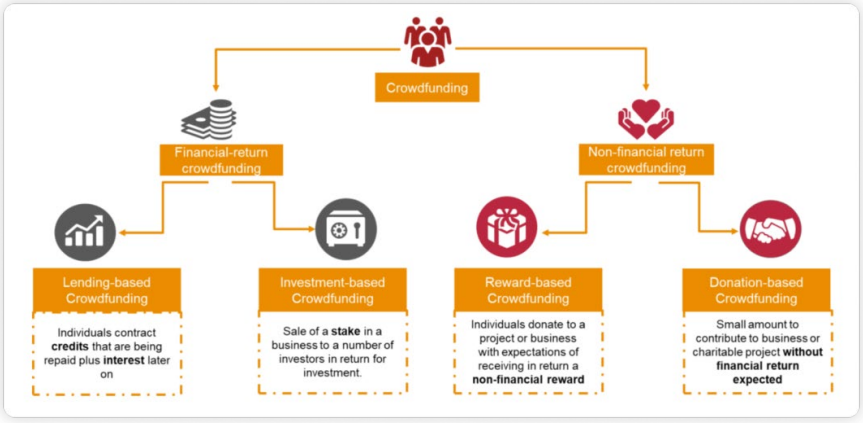

Рис.1.1. Класифікація основних моделей краудфандингу

1. Благодійний (donation-based). Це найпростіша форма краудфандингу, де люди роблять добровільні пожертви на підтримку ідей, ініціатив або допомоги без очікування матеріальної винагороди. Такий тип фінансування широко використовується у сфері благодійності, соціальних проєктів, збору коштів на лікування або підтримку Збройних Сил України.
2. Винагородний (reward-based). Учасники проєкту отримують винагороду за свій внесок - це може бути готовий продукт, сувенір, доступ до послуги чи інша форма подяки. Така модель найчастіше використовується у творчих та підприємницьких проєктах.
3. Інвестиційний (equity-based). Інвестори вкладають кошти у стартап чи компанію в обмін на частку в бізнесі – акції або інші корпоративні права. Такий тип потребує юридичного оформлення інвестицій і дотримання фінансового регулювання.
4. Позиковий (debt-based). У цій моделі користувачі позичають кошти проєкту або фізичній особі з очікуванням повернення з відсотками. Такий механізм подібний до банківського кредиту, але без участі фінансової установи.

5. Peer-to-Peer (P2P). Ця модель посідає особливе місце, у ній користувачі не лише підтримують, але й самостійно ініціюють кампанії в рамках загальної мети, стаючи так званими фандрейзерами.

Peer-to-Peer краудфандинг дозволяє масштабувати збір завдяки особистим ініціативам звичайних учасників, які діляться кампанією серед свого оточення. Це сприяє персоналізації зборів, розширенню аудиторії та формуванню довіри за рахунок знайомих людей. У межах такої моделі можна делегувати відповідальність фандрейзерам, ефективніше залучати спільноти та формувати гнучкі сценарії фінансування.

Краудфандинг як інструмент фінансування має численні сильні сторони, що сприяють його популярності у сучасному цифровому середовищі. Однією з ключових переваг є відносна простота запуску кампанії – для початку збору коштів зазвичай достатньо мати ідею, підготувати опис, візуальні матеріали та зареєструватися на відповідній платформі. На відміну від традиційних механізмів фінансування, таких як банківські кредити чи венчурні інвестиції, краудфандинг не потребує застав, складних фінансових процедур або доступу до вузького кола інвесторів.

Краудфандинг дозволяє охопити широку аудиторію: завдяки інтернету та соціальним мережам інформація про кампанію може швидко поширюватися, що дає змогу залучати підтримку навіть з інших країн. Цей глобальний характер поєднується з високою прозорістю: більшість платформ надають відкритий доступ до статистики збору коштів, що формує довіру з боку потенційних донорів. До того ж, кампанія на краудфандинговій платформі нерідко виконує й маркетингову функцію, оскільки активно поширюється самими учасниками та може привертати увагу медіа або блогерів.

Ще однією перевагою є можливість перевірки життєздатності ідеї ще до її реалізації. Кампанія виконує роль ринку в мінімальному масштабі: вона демонструє, чи готові люди підтримати певний проєкт, продукт або ініціативу, що дає автору зворотний зв’язок і шанси на адаптацію стратегії.

Втім, попри численні позитивні аспекти, краудфандинг має і свої недоліки. Одним із головних викликів є висока конкуренція: на великих платформах щодня запускаються тисячі кампаній, тому вирізнитися серед них стає дедалі складніше. Успіх значною мірою залежить від якісного маркетингу, добре сформованого бренду та активного просування – без цих компонентів навіть перспективна ідея може залишитися непоміченою. Крім того, деякі платформи працюють за принципом «усе або нічого»: якщо цільова сума не зібрана, кошти повертаються донорам, і автор не отримує фінансування взагалі. [2, 4]

Існують також юридичні та операційні труднощі. Залежно від обраної моделі (особливо у випадку інвестиційного чи позикового краудфандингу), ініціатори повинні враховувати фінансове законодавство, регуляції, а іноді – й податкові зобов'язання. Навіть у випадках donation- чи reward-based кампаній важливим є своєчасне інформування аудиторії, виконання зобов'язань і ведення публічної звітності. Відповідно, краудфандинг вимагає не лише енергії на старті, а й регулярного залучення ресурсу протягом усього періоду збору, а також після нього – особливо, якщо обіцяно винагороди.

Таким чином, краудфандинг – це ефективний, гнучкий, але вимогливий інструмент. Його успіх залежить не лише від якості самої ідеї, а й від здатності ініціатора комунікувати, будувати довіру та мобілізувати спільноту навколо свого проєкту.

В Україні краудфандинг набув особливого значення з 2014 року, після Революції Гідності, і суттєво посилився в період повномасштабного вторгнення у 2022 році. Він став одним із ключових інструментів підтримки Збройних Сил України, волонтерських ініціатив, медичних та евакуаційних проєктів. Прикладами масштабного збору є ініціативи через платформу «Банка Монобанку», численні кампанії на дрони, транспорт, оптику та амуніцію, організовані фондами «Повернись живим», Сергія Притули, KOLO, United24 тощо. Ці приклади демонструють ефективність цифрових каналів взаємодії в умовах національної кризи [17, 18, 19].

Сучасні краудфандингові платформи функціонують як складні інформаційні системи. Вони включають фронтенд (інтерфейс користувача), який забезпечує доступність і зручність взаємодії; бекенд - для безпечного збереження даних і обробки транзакцій; а також системи управління кампаніями – для адміністрування, аналітики, звітності та комунікації з донорами.

Таким чином, краудфандинг – це не просто механізм збору коштів, а складний соціотехнічний процес, який поєднує технології, довіру та соціальну енергію. Ефективність таких систем напряму залежить від якісного проєктування їхньої інформаційної архітектури та користувацького інтерфейсу, особливо в умовах активної громадської взаємодії.

1.2. Порівняння фреймворка Angular та бібліотеки React для створення користувацьких інтерфейсів

Створення ефективного, адаптивного і масштабованого користувацького інтерфейсу є одним із ключових етапів розробки сучасних веб-застосунків. У процесі проєктування інтерфейсної частини веб-платформи для краудфандингу важливо обрати відповідний інструмент розробки, який забезпечить високу продуктивність, підтримку типових архітектурних шаблонів і добрий користувацький досвід. Найпоширенішими рішеннями для побудови інтерфейсів є фреймворк Angular та бібліотека React (рис. 1.1).

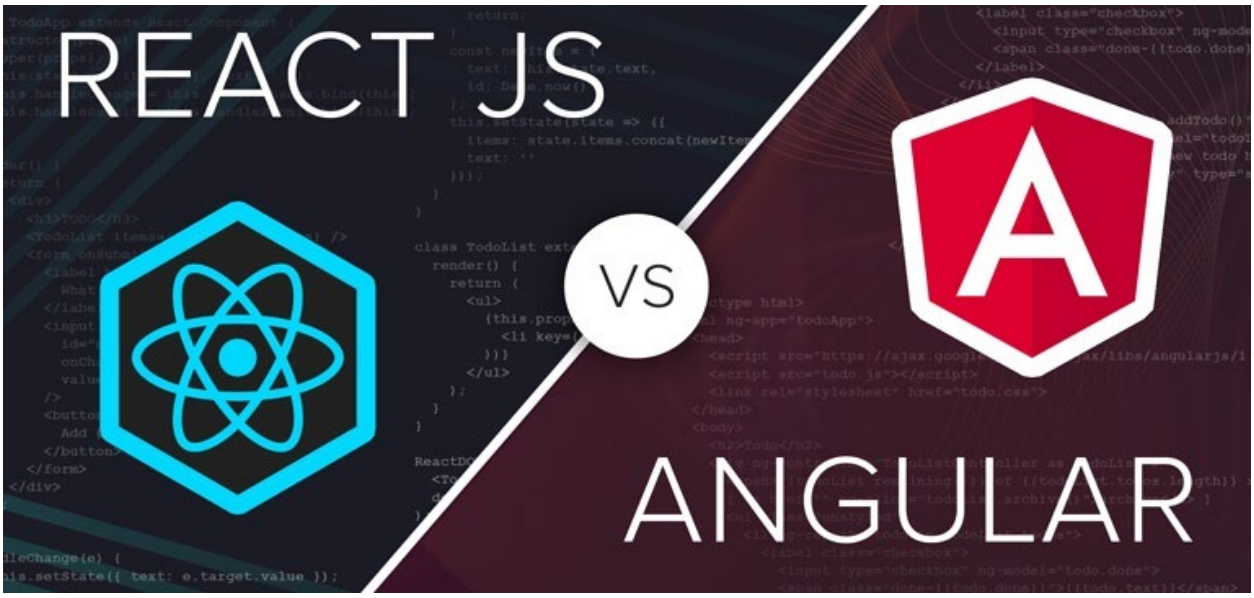

Рис.1.2. Бібліотека React та фреймворк Angular

Angular і React є одними з найпопулярніших інструментів для розробки користувацьких інтерфейсів веб-застосунків. Вони широко використовуються як у невеликих проєктах, так і у великих корпоративних рішеннях. Хоча обидві технології застосовуються для досягнення схожих цілей - створення динамічних веб-інтерфейсів, їхня архітектура, підхід до реалізації та філософія суттєво відрізняються.

Angular – це повноцінний фреймворк для розробки клієнтських веб-додатків. Його перша версія була представлена компанією Google у 2010 році під назвою AngularJS. Згодом, у 2016 році, було випущено повністю переписану версію фреймворку під назвою Angular 2+, яка й надалі активно розвивається. Angular побудований на мові TypeScript, що забезпечує строго типізовану модель програмування, високу стабільність коду та зручність у масштабних проєктах. Однією з основних переваг Angular є наявність усіх необхідних інструментів «з коробки»: двобічне зв’язування даних, вбудована маршрутизація, система залежностей, форми, модулі, компоненти, сервіси та інше. [5, 7]

React, натомість, є бібліотекою, розробленою компанією Meta (раніше Facebook) у 2013 році. Вона призначена виключно для побудови інтерфейсів

(UI) і реалізує так званий view layer (шар подання) у архітектурі Model-View-Controller (MVC). React написаний на мові JavaScript та використовує JSX - синтаксичне розширення, яке дозволяє описувати структуру інтерфейсу в JavaScript-коді. Завдяки своїй гнучкості React не нав’язує структуру застосунку, а дозволяє розробнику самостійно обирати інструменти для управління станом, маршрутизації тощо. Це робить React особливо привабливим для стартапів, навчальних проєктів і застосунків із невеликим або середнім рівнем складності. [8]

Angular позиціонується як «фреймворк повного циклу», оскільки пропонує все необхідне для повноцінного життєвого циклу застосунку - від генерації компонентів до їхнього тестування та компіляції. React, у свою чергу, потребує підключення додаткових бібліотек (наприклад, React Router, Redux, Axios) для повного функціонування. Ця різниця впливає на вибір технології залежно від цілей проєкту, рівня підготовки команди та вимог до масштабованості.

В обох інструментів є активна спільнота, регулярні оновлення, потужна документація та приклади використання в проєктах світового рівня - наприклад, Angular використовується у Google Cloud Console, Gmail, Microsoft Office Web; React - у Facebook, Instagram, Netflix, Airbnb тощо.

Таким чином, Angular і React мають спільну мету - розробку інтерактивних інтерфейсів - проте реалізують її різними способами: Angular - шляхом комплексного підходу, React - через мінімалістичну та гнучку структуру. Обидва інструменти залишаються актуальними у 2025 році та активно розвиваються відповідно до потреб розробників і нових тенденцій у веб-технологіях.

Незважаючи на подібну мету використання - створення динамічних веб-інтерфейсів - Angular і React мають суттєві технічні відмінності, що впливають на архітектуру застосунку, досвід розробки та підтримку проєктів. Розглянемо ключові технічні аспекти, які відрізняють ці інструменти один від одного.

Angular побудований на мові TypeScript - надбудові над JavaScript, яка забезпечує сувору типізацію, покращене автодоповнення, рефакторинг та виявлення помилок на етапі компіляції. Це дозволяє створювати масштабовані, безпечні та підтримувані застосунки.React, хоча і підтримує TypeScript, за замовчуванням використовує JavaScript у поєднанні з JSX - синтаксичним розширенням, яке дозволяє вписувати HTML-подібний код безпосередньо у JavaScript. JSX робить структуру інтерфейсу компактною, але вимагає звикання від розробників, які звикли до шаблонів HTML у окремих файлах.

Angular базується на чітко визначеній модульній архітектурі, яка включає компоненти, сервіси, модулі, ін'єкцію залежностей (Dependency Injection) та двобічне зв'язування даних. Цей підхід дозволяє стандартизувати структуру проєкту, що є перевагою для командної роботи.React, навпаки, надає лише представницький рівень (view layer) і не накладає жорстких вимог до структури. У React можна організовувати архітектуру довільно, що дає гнучкість, але водночас вимагає досвіду та чіткої дисципліни в команді.

У Angular компоненти реалізуються як класи з декораторами (@Component), що мають прив'язку до окремих HTML-шаблонів і CSS-стилів. Така модель дозволяє жорстко розмежовувати логіку, шаблон і стилі.У React компонент - це, зазвичай, функція або клас, яка повертає JSX. Компонентна структура в React є більш гнучкою, з меншою кількістю шаблонного коду, однак усі частини інтерфейсу можуть бути змішані в одному файлі, що ускладнює підтримку у великих застосунках.

Управління станом.Angular використовує сервіси та бібліотеку RxJS для реактивного управління станом. Це дозволяє ефективно обробляти асинхронні потоки даних (наприклад, з API або форм). Крім того, Angular має централізовані рішення на зразок NgRx.React реалізує керування станом через внутрішні механізми компонентів (useState, useReducer) або

сторонні бібліотеки: Redux, Recoil, Zustand, Context API. Такий підхід є гнучким, але часто потребує додаткової конфігурації [20].

У Angular маршрутизація реалізується через вбудований модуль Angular Router. Це дає змогу визначати шляхи, вкладені маршрути, lazy-loading та захищені маршрути.У React роутінг реалізується через сторонню бібліотеку React Router, яка хоч і є де-факто стандартом, однак не є частиною ядра бібліотеки.

Angular пропонує два підходи до створення форм - template-driven (шаблонні) та reactive (реактивні), з вбудованими механізмами валідації.У React форми реалізуються вручну або через сторонні рішення, наприклад, Formik, React Hook Form, що забезпечує більшу гнучкість, але вимагає додаткових зусиль для базових речей.

Angular підтримує двобічне зв’язування (two-way binding) - коли зміни в моделі автоматично оновлюють представлення, і навпаки.React працює за принципом одностороннього потоку даних (one-way binding), що вважається більш передбачуваним і стабільним у великих проєктах, але потребує додаткових дій для двобічної синхронізації.

Angular має вбудований CLI (Angular CLI), який дозволяє швидко створювати модулі, компоненти, сервіси, будувати та тестувати застосунок.У React для початку розробки найчастіше використовується Create React App, Vite, або кастомні збірки. React також має багаті DevTools, але загалом вимагає більше ручної конфігурації.

Таким чином, Angular забезпечує комплексну та структуровану платформу для створення застосунків "під ключ", тоді як React - це гнучка бібліотека, яку розробник може налаштувати під свої конкретні потреби. Вибір між ними залежить від рівня складності проєкту, досвіду команди, вимог до масштабування, продуктивності та підтримки коду.

Продуктивність веб-застосунку є ключовим фактором для забезпечення позитивного користувацького досвіду. Швидке завантаження, миттєва взаємодія з інтерфейсом, плавна навігація - усе це є критично важливим як

для кінцевого користувача, так і для бізнесу. Angular та React мають власні підходи до досягнення високої продуктивності, використовуючи різні механізми оптимізації на рівні рендерингу, компіляції, маршрутизації тощо.

Однією з найбільших інновацій React є використання віртуального DOM (Virtual DOM) – легковагового представлення дерева елементів інтерфейсу. Замість того щоб одразу оновлювати реальний DOM браузера (що є повільною операцією), React спочатку виконує зміни у Virtual DOM, порівнює нову та попередню версію (diffing), а вже потім мінімально змінює реальний DOM. Такий підхід значно підвищує ефективність відображення інтерфейсу, особливо у складних і динамічних застосунках.

Angular підтримує так звану Ahead-of-Time (AOT) компіляцію, яка дозволяє перетворювати шаблони HTML та TypeScript-код у оптимізований JavaScript ще до того, як застосунок буде запущено в браузері. Завдяки цьому браузер не витрачає час на компіляцію - сторінка швидше завантажується, а помилки виявляються ще на етапі збірки.

Обидві технології підтримують рендеринг на стороні сервера (SSR) - метод, за якого HTML-сторінка генерується на сервері та надсилається до браузера вже в готовому вигляді.

У React ця функція реалізується за допомогою Next.js - популярного фреймворку, що базується на React і оптимізований для SEO та продуктивності.

У Angular SSR реалізується через Angular Universal - офіційну платформу для серверного рендерингу.

SSR особливо корисний для покращення першого рендеру сторінки (First Contentful Paint), оптимізації SEO та підтримки повільних пристроїв або нестабільних мереж.

У Angular використання бібліотеки RxJS дозволяє реактивно обробляти події та асинхронні потоки, що знижує навантаження на рендеринг.У React можна вручну оптимізувати рендери за допомогою React.memo(), useMemo, useCallback, а також контролювати глибину оновлень компонента.

Інструменти вимірювання продуктивності. Обидві платформи мають власні DevTools:

- Angular DevTools дозволяє переглядати зміну стану, структуру компонентів, рендери тощо.
- React DevTools - візуалізація компонентів, перевірка продуктивності, побудова профілів рендерингу.

Таким чином, Angular і React мають різні, але ефективні підходи до оптимізації продуктивності. Angular орієнтований на попередню компіляцію, структурованість і рендеринг за допомогою Ivy та AOT. React - на гнучкість, ефективне оновлення DOM за допомогою Virtual DOM та потужні екосистемні рішення на зразок Next.js. Обидві технології дозволяють створювати швидкі, масштабовані та продуктивні веб-застосунки за умови правильного підходу до архітектури та оптимізації.

Масштабованість - це здатність технології підтримувати ефективну розробку, супровід та розширення застосунку у міру зростання його функціональності, кількості користувачів і складності внутрішньої логіки. Обидві платформи - Angular і React - підтримують масштабування, але підходять до цього по-різному. Вибір між ними залежить від структури команди, тривалості проєкту, вимог до підтримки та темпів розробки.

CLI, автоматизація та підтримка. Angular має вбудовану командну утиліту Angular CLI, яка дозволяє створювати нові модулі, компоненти, сервіси та конфігурації всього за одну команду. Це значно спрощує підтримку проєкту, уніфікує структуру та пришвидшує розробку. Angular CLI також інтегрується з тестуванням, збіркою, лінтингом і оптимізацією.

React не має офіційного CLI такого рівня - замість нього зазвичай використовують Create React App, Vite, Next.js, або конфігурують Webpack вручну. Це надає більше свободи, однак потребує глибшого технічного досвіду.

Angular підтримується корпорацією Google з чітким циклом оновлень (двічі на рік) і довготривалою підтримкою (Long-Term Support) для кожної

стабільної версії. Це особливо цінно для enterprise-середовища, де важливо мати стабільну платформу на кілька років.

React, підтримуваний Meta (Facebook), має гнучкіший, менш формалізований підхід до оновлень. Попри активну спільноту, важливі зміни (наприклад, впровадження Hooks або Server Components) впливають на кодову базу, і міграція між версіями може потребувати адаптації.

Отже, Angular забезпечує високу масштабованість завдяки своїй строгій архітектурі, CLI та модульності, що робить його ідеальним вибором для великих і довготривалих проєктів із чіткими вимогами до підтримки. React, натомість, надає більшу гнучкість, дозволяючи будувати структуру за індивідуальним підходом, однак для ефективного масштабування потребує досвіду, зваженого вибору бібліотек і послідовного дотримання архітектурних патернів.

Angular і React - це два провідних технологічних рішення у сфері frontend-розробки, що мають багато спільного в цілях використання, але суттєво відрізняються за структурою, підходами до проєктування та функціональністю. Обидва інструменти дають змогу створювати динамічні односторінкові застосунки (SPA) та широко використовуються в проєктах світового рівня.

Кожен з них використовує DOM - спосіб представлення HTML-документа як об'єктів. Але React використовує Virtual DOM, а Angular - справжній DOM. У чому різниця: якщо ви хочете змінити якісь дані в тегах HTML-документа, Virtual DOM оновлюється лише потрібний фрагмент. При цьому в DOM оновлення відбувається у всіх тегах, доки не знайдеться потрібний шматок коду. Це знижує продуктивність у деяких моментах. Але не треба думати, що у битві React проти Angular одразу перемагає перший. Справа в тому, що такий метод обробки даних дозволяє створювати на Angular великі корпоративні проекти та використовувати складні архітектурні рішення. Безліч функцій дозволяють створювати практично будь-який продукт. При цьому ймовірність помилок є мінімальною.

Один із ключових аспектів, який слід враховувати під час порівняння Angular і React, - це ступінь їхньої складності. Angular, будучи повноцінним фреймворком, має крутішу криву навчання, особливо для новачків у веб-розробці. У той час як React, зосереджений на компонентах, надає більш плавний старт для нових розробників [5, 9, 10].

Таблиця 1.1. Порівняльна характеристика фреймворку Angular та бібліотеки React

Критерій	Angular	React
Тип	Повноцінний фреймворк	Бібліотека для створення користувацьких інтерфейсів
Розробник/Підтримка	Google, підтримка LTS, активні оновлення	Meta (Facebook), велика спільнота, часті оновлення
Мова розробки	TypeScript (строго типізована)	JavaScript (із підтримкою TypeScript через конфігурацію)
Архітектура	Модульна, побудована на DI, MVC-шаблон	Компонентна, функціонально-орієнтована
Старт і крива навчання	Складний старт, велика кількість концепцій	Простий старт, поступова складність
Компоненти	Окремі файли шаблонів (HTML), логіки (TS) і стилів (CSS/SCSS)	JSX - шаблон і логіка в одному файлі
Управління станом	RxJS, сервіси, BehaviorSubject	Redux, Context API, Recoil - сторонні бібліотеки
Маршрутизація	Вбудована (Angular Router)	Потребує сторонніх рішень (React Router)
Підтримка форм	Вбудовані Template-driven і Reactive форми	Форми реалізуються кастомно або через сторонні бібліотеки
Зв'язування даних	Двобічне (two-way) через [(ngModel)]	Одностороннє (one-way), двобічне

		можливо вручну
Розмір застосунку	Більший, але оптимізується AOT-компіляцією	Менший, швидший на старті
Продуктивність	Висока, особливо з оптимізаціями SSR та Ivy	Висока, завдяки Virtual DOM
Застосування	Складні enterprise-застосунки, платформи	Гнучкі SPA, інтерактивні інтерфейси, мобільні застосунки (React Native)
Інструменти розробки	Angular CLI, Angular DevTools	Create React App, Vite, React DevTools
Масштабованість	Висока: чітка структура, DI, розбиття на модулі	Середня: потрібна дисципліна та архітектурні підходи
Екосистема	Централізована, інтегровані рішення	Гнучка, на основі сторонніх бібліотек
Спільнота	Професійна спільнота корпоративного рівня	Найбільша фронтенд-спільнота, безліч прикладів та плагінів
Мобільна розробка	Angular Native (обмежена підтримка)	React Native (популярний фреймворк для мобільних застосунків)

Для наочного порівняння популярності фреймворків Angular та React у 2025 році доцільно використати статистичні дані та графіки, що відображають їхнє поширення у веб-розробці.

Статистика використання за даними W3Techs станом на червень 2025 року:

- React використовується на 5.5% всіх веб сайтів, що становить 6.7% ринку JavaScript-бібліотек [11].
- Angular застосовується на 0.2% веб сайтів, з ринковою часткою 0.3% серед JavaScript-бібліотек [12].

Ці дані свідчать про значну перевагу React у загальному використанні.

Популярність серед розробників згідно з опитуванням Stack Overflow Developer Survey 2024:

- React є найпопулярнішим фреймворком серед розробників, з показником використання 39.5%.
- Angular займає друге місце з показником 17.1%.

Це підтверджує високий рівень популярності React серед професійної спільноти.

На графіках нижче представлено статистику використання JavaScript-бібліотеки React та фреймворку Angular серед найпопулярніших веб-сайтів за даними платформи BuiltWith Trends. Вони ілюструють суттєві зміни у популярності цих технологій протягом останнього десятиліття.

Згідно з рис. 1.2, починаючи з 2013 року React демонструє стабільне зростання популярності. Його застосування особливо інтенсивно розвивалося з 2016 року і досягло піку приблизно у 2024 році серед топ-1 мільйона сайтів. У цьому сегменті кількість сайтів, що використовують React, зросла з кількох тисяч до понад 180 тисяч. Незначне зниження на початку 2025 року може бути пов’язане з оновленням даних або переходом частини проєктів на інші технології. Варто зазначити, що серед топ-10 тисяч сайтів React також утримує стабільну присутність, хоча в абсолютних цифрах цей сегмент значно менший.

У той час як React продовжував зростати, Angular, навпаки, зазнав зниження популярності, що чітко видно на рис. 1.3. Після швидкого підйому з 2017 по 2020 роки Angular досяг пікових значень, але згодом його використання почало помітно зменшуватись. Це особливо виражено серед топ-1 мільйона сайтів, де Angular втратив близько половини позицій протягом 2021–2025 років. Причинами цього тренду, ймовірно, є складність архітектури Angular, а також перехід частини розробників на легші або більш популярні бібліотеки, зокрема той самий React.

Таким чином, порівняльний аналіз свідчить про те, що React утвердився як домінуюча технологія для розробки інтерфейсів, тоді як Angular поступово

втрачає позиції, хоча й залишається актуальним у багатьох корпоративних середовищах [13, 14].

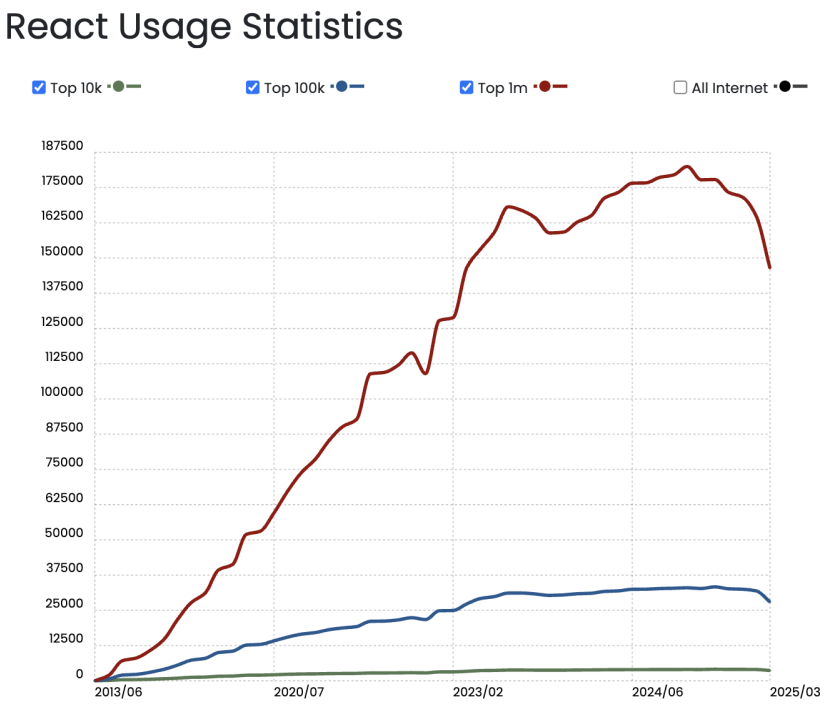

Рис.1.3. Графік використання React серед найпопулярніших веб-сайтів у 2013-2025 рр. (джерело: BuiltWith Trends)

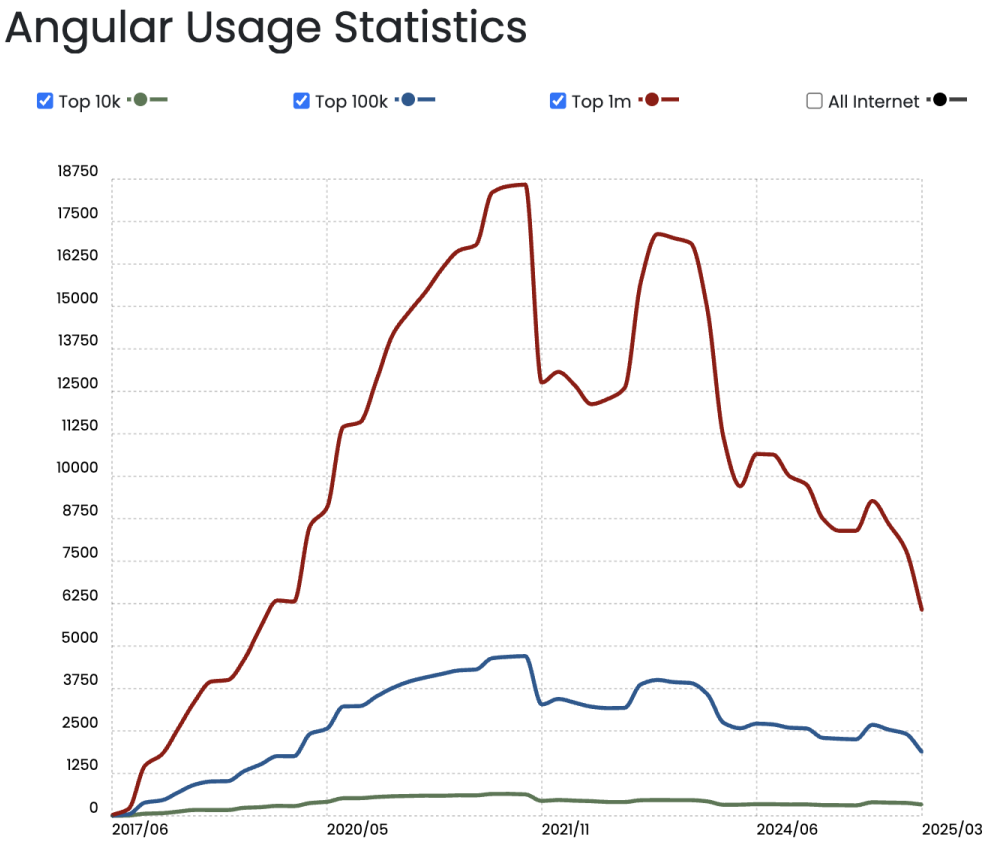

Рис.1.4. Графік використання Angular серед найпопулярніших веб-сайтів у 2017-2025 рр. (джерело: BuiltWith Trends)

Додаткове уявлення про популярність фреймворків React і Angular можна отримати, проаналізувавши динаміку пошукових запитів у сервісі Google Trends. Ці дані відображають не лише реальне використання технологій, а й рівень інтересу до них серед розробників і технічної спільноти загалом.

На рис. 1.4 показано, що пошукова популярність React демонструвала стабільне зростання у 2020–2021 роках, досягнувши піку наприкінці 2021 року. Після цього рівень інтересу зберігався на відносно високому рівні до середини 2023 року, після чого почалося поступове, але помітне зниження. Така динаміка може свідчити про зрілість технології та її широку впровадженість: зниження кількості запитів не обов'язково означає втрату актуальності, а радше стабілізацію у сфері знань і матеріалів [15].

У свою чергу, рис. 1.5 демонструє подібну, але дещо менш позитивну картину для Angular. Попри те, що у 2020–2021 роках фреймворк мав високі показники інтересу, вже з 2022 року спостерігається хвилеподібне, але в цілому знижувальне значення пошукових запитів. На відміну від React, падіння популярності Angular є більш вираженим, що корелює з відповідними тенденціями використання технології на практиці [16].

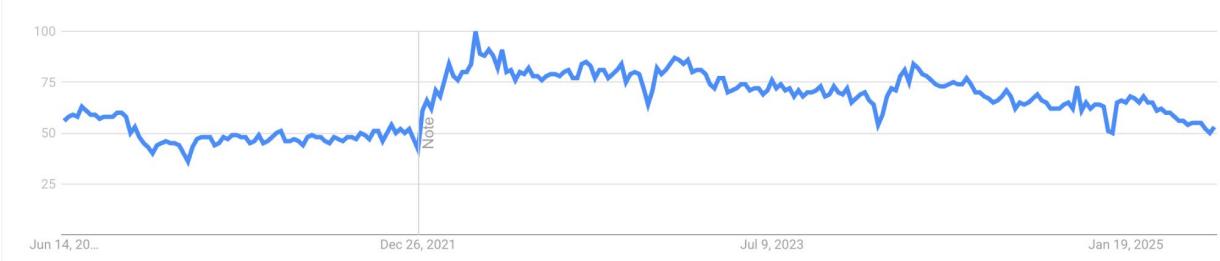

Рис.1.5. Графік динаміки популярності пошукових запитів "React" серед найпопулярніших веб-сайтів у 2020-2025 рр за даними Google Trends

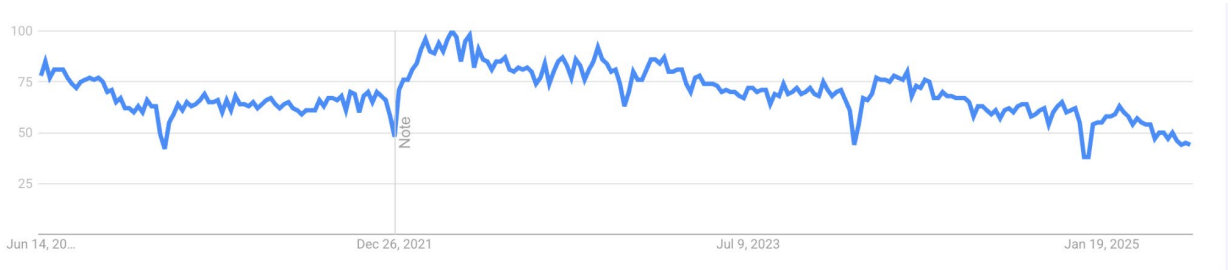

Рис.1.6. Графік динаміки популярності пошукових запитів "Angular" серед найпопулярніших веб-сайтів у 2020-2025 рр за даними Google Trends

У межах цієї бакалаврської роботи було прийнято рішення використовувати саме Angular для реалізації адміністративної частини користувацького інтерфейсу. Це зумовлено потребою у структурованому підході, підтримці компонентно-модульної архітектури, зручному зв'язуванні даних, наявності вбудованих сервісів для роботи з формами та злагодженості у командній розробці. Angular забезпечує комплексне рішення з вбудованими інструментами для маршрутизації, управління станом, валідації форм та HTTP-запитів, що дозволяє зменшити залежність від сторонніх бібліотек та спростити підтримку коду.

Водночас донорська частина інтерфейсу реалізована із застосуванням jQuery – як легшого інструменту, що добре підходить для реалізації простих сценаріїв внеску. Такий підхід дозволяє оптимізувати ресурси та забезпечити ефективну взаємодію з користувачами.

1.3. Web-ресурси краудфандингу соціальних проєктів

Краудфандингові платформи – це спеціалізовані web-ресурси, що забезпечують організацію збору коштів широким колом користувачів. Вони виступають у ролі посередника між ініціаторами кампаній (проєктів) та донорами (спонсорами, інвесторами, благодійниками), надаючи технічні, платіжні та комунікаційні засоби для ефективної реалізації фінансових ініціатив.

Платформи можуть значно відрізнятися за моделлю фінансування, функціональністю, цільовою аудиторією та юридичним статусом. Залежно від принципів взаємодії між сторонами, краудфандингові сервіси класифікуються на кілька основних типів [2, 3]:

1. Благодійні (donation-based)

Це найпоширеніший тип краудфандингу в соціальній сфері. Донори надають кошти без очікування винагороди чи повернення. Найчастіше такі платформи використовуються для гуманітарної допомоги, медичних зборів, військових ініціатив, освітніх і культурних проєктів.

Приклади**:** GoFundMe, JustGiving, KOLO, Повернись живим.

2. Винагородні (reward-based)

Передбачають, що донори отримають певну нематеріальну або матеріальну винагороду у разі успіху кампанії. Такий тип краудфандингу характерний для креативних або інноваційних проєктів: мистецтво, технології, мода, музика.

Приклади**:** Kickstarter, Indiegogo, Велика Ідея.

3. Інвестиційні (equity-based)

Донори виступають у ролі інвесторів, які в обмін на кошти отримують частку бізнесу, дивіденди або інші форми прибутку. Такий тип вимагає юридичної легалізації інвестицій, прозорості структури бізнесу, аналітики та звітності.

Приклади**:** StartEngine, SeedInvest.

4. Позикові (debt-based)

Цей формат полягає у наданні мікропозик або кредитів з фіксованою ставкою чи очікуваним поверненням коштів. Зазвичай використовується для фінансування соціальних або підприємницьких ініціатив у країнах, що розвиваються.

Приклади**:** Kiva, LendingClub.

5. Peer-to-Peer (P2P-fundraising)

Особливий формат, у якому не лише організація, але й окремі учасники (фандрейзери) можуть створювати персональні кампанії в межах однієї

великої ініціативи. Це розширює охоплення аудиторії, дозволяє делегувати поширення збору серед друзів і партнерів, забезпечує "вірусний" ефект.

Приклади**:** Donorbox P2P, PayBee P2P, JustGiving P2P.

Попри популярність цих сервісів, українські ініціативи стикаються з низкою обмежень:

- Платіжні обмеження: деякі платформи не підтримують перекази з українських карток або не працюють з гривнею.
- Вимоги до резидентства: для верифікації потрібна юридична адреса у США або Великобританії.
- Юридичні бар'єри: обмеження щодо фіскалізації, звітності та використання благодійних коштів.

В умовах війни ці фактори створюють потребу в локалізованих рішеннях, що адаптовані під українське законодавство, мову, платіжну інфраструктуру та специфіку гуманітарної допомоги.

Розвиток краудфандингу в Україні почався активно після Революції Гідності у 2014 році, однак справжній прорив у цій сфері відбувся з початком повномасштабної війни у 2022 році. В умовах воєнного часу саме онлайн-збір коштів став основним інструментом оперативного фінансування армії, гуманітарних місій, волонтерських ініціатив і соціальних проєктів. У цьому контексті в Україні сформувалась своя екосистема платформ, адаптованих до локальних умов, фінансових інструментів та потреб аудиторії [4].

Таблиця 1.2. Порівняльна таблиця українських Web-ресурсів для краудфандингу соціальних проєктів

Критерій	Спільнокошт (Big Idea)	Повернись живим	Фонд Сергія Притули	KOLO
Тип	Reward-based + Donation	Donation-based	Donation-based, P2P	Donation-based, технологічний фокус
Рік запуску	2012	2014	2020	2022
Фокус	Культурні, освітні, соціальні	Допомога ЗСУ	Допомога ЗСУ, P2P кампанії	Високотехнологічне обладнання для фронту

	проєкти			
Платформа	Big Idea	okou.to	prytulafoundation.org	koloua.com
Модель винагород	Так (мерч, участь)	Ні	Ні, але активна публічна комунікація	Ні
Форма звітності	Публічна, через сторінки проєктів	Звітність на сайті + фінансові виписки	Відеозвіти, щотижневі пости	Google Sheets, GitHub
Мобільна адаптивність	Так	Так	Так	Так
Інтеграція з банківськими системами	LiqPay, Приват24	Monobank, Приват24	Monobank, карткові збори	Monobank, криптовалюти
Модель P2P	Обмежено	Ні	Так, фандрейзери, амбасадори	Так, через індивідуальні збори
UX та інтерфейс	Класичний, багатосторінковий	Простий, з фокусом на звітність	Інтуїтивний, емоційно-залучаючий	Технічний, лаконічний
Довіра користувачів	Стабільна, перевірена роками	Найвища в армійському сегменті	Довіра через персональний бренд	Довіра через прозорість

Українські платформи не лише адаптовані до національного контексту, а й продемонстрували високу ефективність у кризових умовах. Вони поєднують простоту використання, довіру аудиторії та гнучкість. Це доводить, що локальні технічні рішення можуть бути не менш ефективними, ніж великі міжнародні сервіси, особливо за наявності прозорої структури, звітності та активної цифрової комунікації [3, 4].

Проєктована користувацька частина web-платформи у цій роботі спирається на найкращі практики згаданих рішень:

- реалізація інтерфейсу з використанням Angular - як технології, що дозволяє побудувати структурований, гнучкий та безпечний інтерфейс адміністрування;

- застосування jQuery у публічній частині - як легкого рішення для забезпечення донорських сценаріїв без перевантаження системи;
- акцент на P2P-фандрейзингу як надійній моделі соціальної взаємодії;
- фокус на мінімалістичному, зрозумілому дизайні, оптимізованому під українську аудиторію.

РОЗДІЛ 2. ІНФОРМАЦІЙНА СИСТЕМА КРАУДФАНДИНГУ СОЦІАЛЬНИХ ПРОЄКТІВ

2.1. Front end розробка з використанням бібліотеки JQuery

У процесі розробки донорської частини веб-платформи було вирішено використати бібліотеку jQuery, як основний інструмент для реалізації інтерактивної поведінки інтерфейсу. Цей вибір є обґрунтованим з огляду на характер задач, простоту в реалізації, швидкість прототипування, а також вже наявну інтеграцію з внутрішніми модулями платформи.

jQuery – це легка бібліотека JavaScript, яка дозволяє швидко працювати з DOM-деревом, обробляти події, виконувати анімації та надсилати AJAX-запити. Донорська частина системи не потребує складної логіки станів чи маршрутизації, тому впровадження повноцінного фреймворку (наприклад, React чи Angular) на цьому етапі вважалося недоцільним [6].

Платформа підтримує сценарій швидкого донату (Quick Pledge) – тобто внесення платежу за декілька кліків, без повної реєстрації. Для таких завдань jQuery є ефективним інструментом, який забезпечує:

- зчитування та вставлення даних з полів;
- зміну вигляду форм у відповідь на дії користувача;
- керування блоками DOM (.hide(), .show(), .toggleClass());
- спрощену інтеграцію з backend-логікою через API-запити.

Код інтерфейсу зберігається у файлі quickpay.html. Основна взаємодія з користувачем відбувається через вказані секції (див. наступні підпункти), які поступово змінюються в процесі заповнення. Ці секції представлені у вигляді -елементів з відповідними id, і показуються/приховуються динамічно. Використані техніки:

- Подієве програмування: кнопки, вибір суми, поля вводу реагують на події click, change, keyup.

- Перехід між етапами форми: реалізований через .hide() / .show() та .scrollTo(), що дозволяє зберігати єдину сторінку з покроковим процесом.
- Передача стану між секціями: дані, введені на першому кроці, динамічно зберігаються в JavaScript-змінних, а потім вставляються у підтвердження платежу (#quickPledgeConfirm) чи додаткові поля (#additionalInfo).
- Анімації: використовуються при переході між етапами (ефект появи/зникнення, фокусування на полях).

Вибір jQuery дозволив:

- швидко реалізувати сценарій донату без складної архітектури;
- уникнути перевантаження DOM-процесів завдяки прямому управлінню;
- легко підтримувати код без потреби у зборці, конфігурації та маршрутизації.

Однак, варто зазначити й обмеження:

- відсутність реактивності (на відміну від Angular/React);
- громіздкість при роботі з великими структурами даних;
- складність підтримки при масштабуванні.

Таким чином, jQuery виправдовує себе у рамках локального, лінійного сценарію швидкої оплати, що не вимагає складних станів або компонентного розділення. Надалі, при розвитку або рефакторингу проєкту, можливий перехід на сучасний фреймворк.

Секція #quickPledge є першою і найважливішою частиною процесу здійснення пожертви. Саме тут користувач вперше взаємодіє з інтерфейсом платформи, обирає суму, тип внеску, а також може додати власний коментар або інші параметри, якщо це дозволено кампанією.

У реалізації модуля #quickPledge беруть участь численні функції, зокрема loadOptionsWithData(), getActivePresetOptions(), getGridViewOptions(), getListViewOptions(), initOptionListViewSwitcher(),

optionSelected(), loadDonationConfirm() та інші. Вони забезпечують повний цикл взаємодії користувача з інтерфейсом: від вибору суми до підготовки даних для передачі до Stripe та сервера.

Блок #quickPledge представлений у HTML як із низкою підсекцій:

- Головний заголовок кампанії.
- Група кнопок із можливими опціями для донату.
- Поле для введення кастомної суми.
- Кнопка переходу до наступного етапу.

Усі ці елементи ініціалізуються за допомогою jQuery одразу після завантаження сторінки.

Далі у секції #quickPledgeConfirm відбувається візуальне узагальнення вибраних даних перед переходом введення додаткової інформації або до платежу, у разі її відсутності.

У побудові секції бере участь функція loadDonationConfirm(), яка викликає низку допоміжних методів: hideAll(), setPageState(), getCampaignType(), loadStreetAddress(), initGoogleMapsAutoComplete(), getPhoneNumRequired(), getHideSelfContribution(), loadTeamSelect(), getSelectedItemsHtml(), getDonationButtonLabel(), showNextButton(), validateDonationConfirm(), та showOneTimePayment(). Ці функції відповідають за динамічне формування інтерфейсу підтвердження, обробку типів внесків і адаптацію до умов Peer-to-Peer кампанії.

У межах цього блоку відображається:

- Обрана сума пожертви з попереднього етапу #quickPledge.
- Поле для введення суми, яку ви хочете задонатити від себе одразу.
- Поле для введення справжнього ім’я того, хто створює фандрейзера.
- Поле для введення імені самого фандрейзера, яке буде відображатися для донорів.
- Поля для введення електронної пошти.
- Поле для введення номеру, якщо зазначено в налаштуваннях кампанії.

- Поле для введення адресу, якщо зазначено в налаштуваннях кампанії.
- Кнопка переходу до наступного етапу.

Уся ця інформація оновлюється динамічно за допомогою jQuery. При переході зі сторінки #quickPledge дані передаються у JavaScript-об’єкт і використовуються для формування вмісту секції #quickPledgeConfirm.

Секція #additionalInfo виконує важливу допоміжну функцію в процесі оформлення донату в кампанії. Саме тут платформа запитує в користувача додаткову інформацію, яка може бути важливою для організаторів, фандрейзера або навіть для майбутньої аналітики.

У формуванні секції #additionalInfo, яка відповідає за збір додаткових даних користувача в межах Peer-to-Peer кампанії, беруть участь функції loadAdditionalInfo(), initSelectedItems(), loadAdditionalInfoText(), loadAdditionalInfoSelect(), showOneTimePayment() та validateAdditionalInfo(). Ці методи забезпечують динамічне формування відповідних полів (текстових, випадаючих списків, чекбоксів), ініціалізацію попередньо вибраних значень, відображення секції одноразового платежу за потреби, а також перевірку коректності введеної інформації перед фіналізацією транзакції.

Ця частина інтерфейсу зазвичай містить:

- Поле для будь-якої додаткової текстової інформації, яка може бути введена.
- Поле для будь-якої текстової інформації, яку потрібно вибрати серед запропонованих варіантів.

Ця секція відображається лише у випадку, якщо попередні кроки були пройдені успішно. Всі динамічні поля формуються залежно від налаштувань кампанії, які підтягуються з об'єкта конфігурації:

Секція #rightContainerCampaignInfo є правобічним фіксованим блоком, що супроводжує користувача протягом усього процесу оформлення донату в межах кампанії. Її головна мета - нагадувати, до якої кампанії належить

донат та кому саме адресовано збір, підсилюючи мотивацію користувача та підвищуючи емоційний вплив.

Відображення цієї секції здійснюється через загальні функції оновлення DOM, без виділення окремих методів із відповідними назвами. Наповнення блоку динамічно відбувається на основі даних, отриманих з об’єктів організації, кампанії та обраної опції. Ці дані вставляються у відповідні елементи DOM за допомогою функцій типу html(), text(), attr() та інших, що формують візуальне представлення інформації.

До складу #rightContainerCampaignInfo входить в залежності від контексту:

- Інформація про вибрану кампанію.
- Інформація про орагізацію, до якої належить кампанія.
- Інформація про вибрану опцію
- Прогрес виконання цілі

Позиціонування цього блоку не змінюється при переходах між секціями форми, що створює ефект контекстної підтримки - користувач завжди бачить, що він робить та кому адресований донат.

Секція #oneTimePay є останнім етапом оформлення пожертви - саме тут відбувається ініціація реального платежу. У контексті Peer-to-Peer кампанії цей крок також включає остаточну фіксацію пожертви за конкретним фандрейзером, а всі параметри транзакції вже заповнено на основі попередніх кроків.

Для обробки платежів на платформі використовується платіжний сервіс Stripe – один із найпоширеніших і найбезпечніших платіжних процесорів у світі. Його підтримка включена безпосередньо у quickPledge.js через об’єкти StripePaymentRequest та StripePaymentElement.

Як працює Stripe у quickPledge.js:

1. Ініціалізація Stripe. Stripe ініціалізується з публічного ключа а допомого функції initialize(). Далі створюється elements API, після чого

генерується платіжна форма card, яка виводиться в DOM. Поле для індексу (postalCode) приховане.

```
StripePaymentElement.prototype.initialize = function () {
    this.stripe = Stripe(api.getStripePublishableKey());
    this.elements = this.stripe.elements();
    this.card = this.elements.create('card', {
        hidePostalCode: true,
        style: style
    });

    this.card.on('change', function (event) {
        const errors = $("#stripePaymentElementErrors");
        if (event.error) {
            errors.text(event.error.message).show();
        } else {
            errors.hide();
        }
    });
};
```

2. Рендер елементу введення картки. Функція mount відповідає за безпечне виведення платіжного поля (елементу типу card) на сторінку. Вона також підключає кнопку підтвердження, обробляє подію натискання, генерує Stripe-токен із введених даних та, у разі успішної генерації, передає його для обробки на бекенд. У разі помилок (наприклад, недійсна картка), користувач бачить відповідне повідомлення.

```
StripePaymentElement.prototype.mount = function() {
    this.card.mount('#stripePaymentElementBody');
```

```
    const button = $("#stripePaymentElementSubmit");
    let submitted = false;

    button.off("click");
    button.on("click", (event) => {
        event.preventDefault();
        if(submitted) { return; }
        submitted = true;

        const text = button.text();
        button.text('Processing...');
        button.attr('disabled', true);

        this.stripe.createToken(this.card).then((result) => {
            submitted = false;

            button.text(text);
            button.attr('disabled', false);

            if (result.error) {
                const errors = $("#stripePaymentElementErrors");
                const message = result?.error?.message || ""
                return errors.text(message);
            }

            return this.events["token"].call(this, result.token);
        });
    });
}
```

У результаті реалізації донорської частини веб-платформи було успішно застосовано jQuery як оптимальний інструмент для створення лінійного сценарію донату з покроковою взаємодією. Такий підхід виявився ефективним у контексті завдань, що не потребують складної архітектури чи реактивного керування станом.

jQuery забезпечла швидку обробку подій, маніпуляцію DOM-елементами, динамічну побудову інтерфейсу та гнучке підключення до backend-сервісів. Завдяки цьому вдалося реалізувати інтуїтивно зрозумілий сценарій швидкого внесення пожертви, який включає заповнення даних, підтвердження вибору та інтеграцію з платіжною системою Stripe.

Попри відомі обмеження jQuery — такі як складність масштабування, відсутність реактивності та громіздкість при роботі з великою кількістю станів — вибір цієї бібліотеки цілком виправданий для даного прикладу використання. Розроблений функціонал може бути легко адаптований або переписаний на сучасний фреймворк у майбутньому, якщо виникне потреба в розширенні або переосмисленні архітектури.

Таким чином, технічна реалізація донорської частини платформи показала, що навіть класичні інструменти на зразок jQuery можуть бути ефективними у вузьких, добре визначених сценаріях взаємодії, особливо коли пріоритетом є швидкість розробки, стабільність та доступність.

2.2. Застосування фреймворка Angular для розробки Web-платформи

Для розробки веб-платформи краудфандингу соціальних проєктів було обрано фреймворк Angular. Цей вибір був зроблений з урахуванням численних переваг, які надає цей фреймворк. Angular пропонує модульну структуру, що дозволяє ефективно розділити функціонал на логічні блоки, та компонентний підхід, який сприяє повторному використанню коду.

Використання TypeScript як основної мови програмування забезпечує надійну типізацію та кращу підтримку середовища розробки.

Фреймворк надає потужні вбудовані інструменти для розробки, включаючи систему роутингу для навігації між сторінками, реактивні форми для валідації даних, HTTP-клієнт для роботи з API та ефективну систему змін для оптимізації продуктивності.

З бізнес-перспективи, Angular має велику спільноту розробників та надає довгострокову підтримку від Google, що гарантує стабільність та безпеку проекту. Наявність готових UI-компонентів, таких як Material Design та MDB Angular, значно прискорює процес розробки та забезпечує професійний вигляд інтерфейсу.

У проекті використовується сучасний стек технологій, включаючи Angular Material для UI компонентів, NgRx для управління станом додатку, RxJS для роботи з асинхронними операціями та різні спеціалізовані бібліотеки, такі як ngx-pagination, ngx-quill, тощо. Цей технічний стек дозволяє створювати масштабований, підтримуваний та продуктивний додаток, що відповідає сучасним вимогам до веб-розробки та забезпечує зручний користувацький досвід.

Архітектура веб-платформи побудована на основі модульного підходу Angular, що дозволяє ефективно організувати код та забезпечити його масштабованість. Основний додаток розділений на логічні модулі, кожен з яких відповідає за певний функціонал системи. Головний модуль додатку (app.module.ts) виступає точкою входу та об'єднує всі інші модулі системи.
Структура проекту організована таким чином, що кожен функціональний модуль (наприклад, manage-p2p) має свою власну внутрішню структуру, яка включає компоненти, сервіси, моделі даних та специфічні для модуля утиліти. Така організація дозволяє підтримувати принцип інкапсуляції та забезпечує чітке розділення відповідальності між різними частинами додатку.

Особливу увагу при проектуванні архітектури було приділено компонентному підходу. Кожен компонент відповідає за конкретну частину інтерфейсу користувача та має чітко визначені вхідні та вихідні дані. Компоненти організовані ієрархічно, що дозволяє ефективно керувати потоком даних та подіями в додатку. Наприклад, модуль manage-p2p містить компоненти для відображення списку P2P кампаній та їх детальної інформації, які можуть бути перевикористані в різних частинах додатку.

Система роутингу реалізована через app-routing.module.ts та модульні роути, що забезпечує ефективну навігацію між різними частинами додатку. Використання lazy loading для модулів дозволяє оптимізувати початкове завантаження додатку, завантажуючи код тільки коли він дійсно потрібен.

Для управління станом додатку використовується NgRx, що дозволяє централізовано керувати даними та забезпечує передбачувану поведінку додатку. Сервіси, такі як ті, що знаходяться в директорії services, відповідають за взаємодію з бекендом та обробку бізнес-логіки, забезпечуючи розділення відповідальності між компонентами та бізнес-логікою.

Архітектура додатку також включає спільні компоненти та сервіси, які використовуються в різних частинах системи. Наприклад, модуль core містить базові сервіси та компоненти, які використовуються в усьому додатку, а модуль shared містить перевикористовувані компоненти та директиви.

Така архітектура забезпечує високу модульність, масштабованість та підтримуваність коду, що є критично важливим для великого проекту краудфандингової платформи. Вона дозволяє легко додавати нові функції, модифікувати існуючі та підтримувати код у довгостроковій перспективі.

Система роутингу в Angular-додатку реалізована з використанням вбудованого модуля RouterModule, що забезпечує ефективну навігацію між різними частинами додатку. Головний роутинг модуль (app-routing.module.ts) налаштований з використанням сучасних практик Angular,

включаючи підтримку прив'язки до компонентів (bindToComponentInputs: true) та оновлення при навігації на ту саму URL-адресу (onSameUrlNavigation: "reload").

Для P2P функціоналу реалізовано два основних роути:

1. /manage/p2p/list - для відображення списку P2P кампаній.
2. /manage/p2p/id - для відображення детальної інформації про конкретну P2P кампанію

Компоненти, відповідальні за відображення P2P функціоналу, організовані в окремій директорії manage-p2p/components та включають:

- manage-peer-to-peer-page - головний компонент для списку P2P кампаній;
- manage-peer-to-peer-details-page - компонент для детального перегляду кампанії;
- peer-to-peer-list - компонент для відображення списку кампаній;
- peer-to-peer-details-card - компонент для відображення деталей кампанії.

Для управління станом P2P функціоналу використовується NgRx, що включає:

- p2p.actions.ts - визначення дій для управління станом;
- p2p.effects.ts - обробка побічних ефектів;
- p2p.reducer.ts - логіка оновлення стану;
- p2p.selectors.ts - селектори для отримання даних зі стану.

Така організація роутингу забезпечує:

- Чітке розділення відповідальності між компонентами.
- Ефективну навігацію між різними частинами P2P функціоналу.
- Можливість легкого розширення функціоналу.
- Оптимізоване завантаження компонентів.
- Зручне управління станом додатку.

Роутинг реалізований з урахуванням принципів lazy loading, що дозволяє оптимізувати початкове завантаження додатку. Компоненти

завантажуються тільки коли вони дійсно потрібні, що покращує продуктивність та користувацький досвід.

Кожен компонент в Angular-додатку складається з чотирьох основних файлів:

- .component.ts - файл з логікою компонента.
- .component.html - файл з шаблоном відображення.
- .component.scss - файл зі стилями компонента.
- .component.spec.ts - файл з тестами компонента.

Компонент peer-to-peer-list відповідає за відображення списку P2P кампаній. У його структурі присутні всі чотири стандартні файли, де основний файл peer-to-peer-list.component.ts містить логіку для відображення списку кампаній, фільтрації та сортування даних. HTML-шаблон реалізує табличне відображення даних з підтримкою пагінації, а SCSS-файл містить стилі для оформлення таблиці та інтерактивних елементів.

Компонент інтегрується з NgRx для управління станом та взаємодіє з API для отримання списку кампаній, оновлення їх статусу та фільтрації даних на стороні сервера. Реалізована обробка помилок та відображення статусів завантаження для покращення користувацького досвіду.

Компонент peer-to-peer-details-page також складається з чотирьох стандартних файлів і забезпечує детальний перегляд та управління окремою P2P кампанією. У файлі peer-to-peer-details-page.component.ts реалізована логіка для відображення детальної інформації про кампанію, форми редагування параметрів та валідацію даних. HTML-шаблон організує відображення інформації, а SCSS-файл містить стилі для оформлення сторінки.

Для розширення функціоналу використовуються додаткові компоненти, кожен з яких також має стандартну структуру з чотирьох файлів:

- peer-to-peer-details-card - для відображення детальної інформації.

- p2p-donations-report-tab - для звіту про пожертви.
- p2p-fundraisers-report-tab - для звіту про фандрайзерів.
- p2p-pledges-report-tab - для звіту про зобов'язання.
- p2p-donor-details - для відображення інформації про донорів.

Компонент взаємодіє з API для отримання детальної інформації про кампанію, оновлення параметрів, отримання статистики та управління учасниками. Додатково реалізовані функції експорту даних, генерації звітів, відправки сповіщень та управління доступом.

Всі компоненти розроблені з використанням сучасних практик Angular, включаючи реактивні форми для валідації даних, асинхронні операції з використанням RxJS, компонентний підхід з чітким розділенням відповідальності, інтеграцію з NgRx для управління станом.

Сервіси в Angular-додатку відіграють ключову роль у взаємодії з бекендом та обробці бізнес-логіки. Вони реалізують паттерн Singleton, що забезпечує єдиний екземпляр сервісу для всього додатку. У модулі manage-p2p реалізовано наступні сервіси:

1. Основні сервіси для роботи з P2P кампаніями:
    - peer-to-peer-manage - основний сервіс для управління P2P кампаніями.
    - fundraiser-list - сервіс для роботи зі списком фандрайзерів.
2. Сервіси для роботи з модальними вікнами:
    - modal-p2p-detail - для відображення деталей кампанії.
    - modal-p2p-edit - для редагування параметрів кампанії.
    - modal-p2p-add-fundrasing - для додавання нових фандрайзерів.
    - modal-p2p-add-donations - для додавання пожертв.
    - modal-p2p-move-transactions - для переміщення транзакцій.
    - modal-p2p-add-milestone - для додавання етапів кампанії.
    - modal-p2p-add-bulk-milestones - для масового додавання етапів.
    - modal-p2p-dowload-qr-cards - для генерації QR-карток.
    - modal-street-address - для роботи з адресами.

- modal-fulfill-bulk-donor-pledges - для масового виконання зобов'язань донорів.
- modal-fullfill-all-donors - для виконання всіх зобов'язань донорів.

Кожен сервіс відповідає за конкретну частину функціоналу та забезпечує:

- Взаємодію з API через HTTP-запити.
- Обробку відповідей від сервера.
- Валідацію даних.
- Обробку помилок.
- Кешування даних при необхідності.

Сервіси інтегруються з NgRx для управління станом додатку, використовуючи actions та effects для обробки асинхронних операцій. Це забезпечує передбачувану поведінку додатку та спрощує тестування.

Моделі даних представлені у вигляді TypeScript інтерфейсів, що забезпечує типізацію та перевірку типів на етапі компіляції. Це допомагає уникнути помилок при роботі з даними та покращує підтримку коду.

Сервіси також реалізують паттерн Repository, що абстрагує доступ до даних та забезпечує єдиний інтерфейс для роботи з різними джерелами даних. Це дозволяє легко змінювати реалізацію доступу до даних без зміни коду, який використовує ці сервіси.

Реалізація P2P функціоналу з використанням Angular фреймворку дозволила створити сучасну, масштабовану та підтримувану веб-платформу для краудфандингу соціальних проєктів. Використання модульної архітектури та компонентного підходу забезпечило чітке розділення відповідальності між різними частинами додатку, що значно полегшило розробку та подальшу підтримку.

Основними перевагами обраного підходу є:

- Ефективна організація коду завдяки модульній структурі.
- Зручна навігація між різними частинами додатку через систему роутингу.

- Можливість легкого розширення функціоналу.
- Висока продуктивність завдяки оптимізації завантаження компонентів.
- Надійне управління станом додатку через NgRx.

Реалізація P2P функціоналу включає всі необхідні компоненти для ефективної роботи з кампаніями, фандрайзерами та донорами, а також забезпечує зручний інтерфейс для користувачів. Використання сучасних практик розробки, таких як реактивні форми, асинхронні операції та адаптивний дизайн, дозволило створити продуктивний та зручний у використанні додаток.

У майбутньому можливі такі покращення:

- Впровадження нових функцій для аналітики та звітності.
- Оптимізація продуктивності для роботи з великими наборами даних.
- Розширення можливостей для роботи з мобільними пристроями.
- Впровадження нових інструментів для взаємодії з донорами.
- Покращення системи безпеки та валідації даних.

Загалом, використання Angular для розробки P2P функціоналу виявилося успішним рішенням, яке дозволило створити сучасну та ефективну платформу для краудфандингу соціальних проєктів.

2.3. Функціональні можливості інформаційної системи зі сторони користувача

Функціональні можливості платформи зі сторони користувача реалізовані у вигляді покрокового інтерфейсу, що забезпечує зручність, прозорість та мінімальну кількість бар’єрів для здійснення пожертви.

Уявімо типову ситуацію: користувач отримує посилання на активну кампанію - наприклад, через соціальні мережі, електронну пошту, або шляхом сканування QR-коду на події, де організація представлена фізично.

Перше, що бачить користувач – це сторінка збору, створена некомерційною організацією у форматі Peer-to-Peer. На цій сторінці розміщено ключову інформацію про мету збору, суму, яка вже зібрана,

візуальні матеріали та кнопку для приєднання до кампанії як донор або як фандрейзер (рис. 2.1).

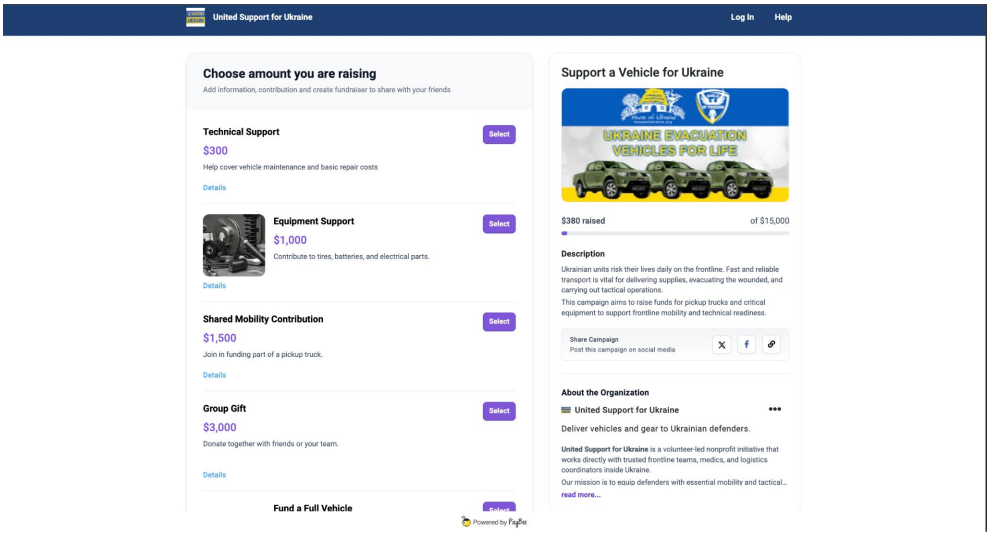

Рис.2.1. Головна сторінка збору кампанії

При натисканні на назву опції або кнопку “Details” відкривається модальне вікно з більш детальною інформацією про конкретну опцію (рис 2.2).

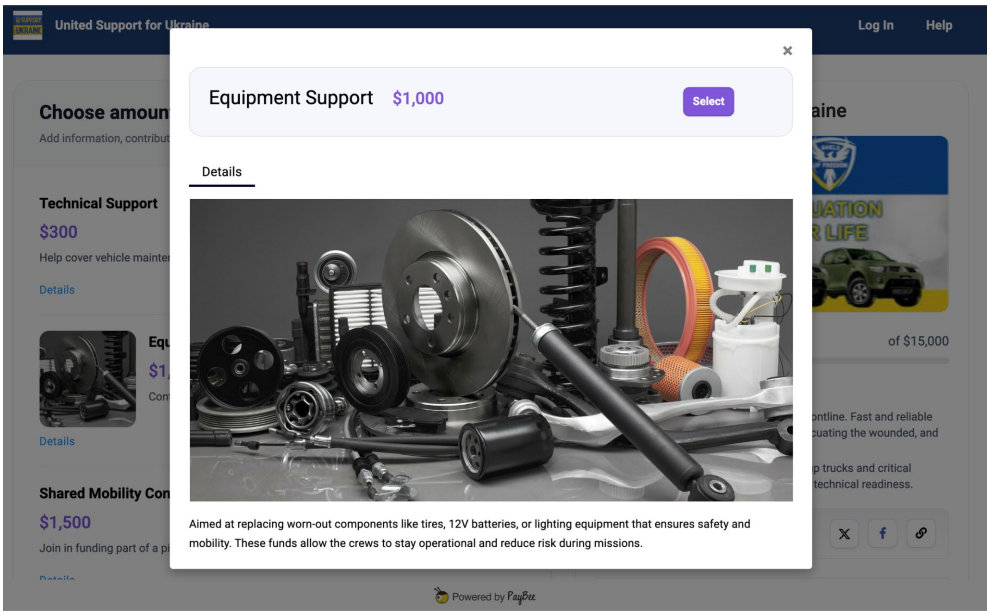

Рис.2.2. Модальне вікно з деталями конкретної опції

Після натискання кнопки “Next” відбувається перехід на сторінку для введення основної інформації. Тут користувачеві потрібно ввести суму донату одразу до свого ж фандрейзера (за бажанням), своє ім’я, ім’я того, дл

кого створюється фандрейзер (якщо потрібно), ім’я фандрейзера, свою електронну пошту та адрес, якщо цього вимагає кампанія (рис. 2.3).

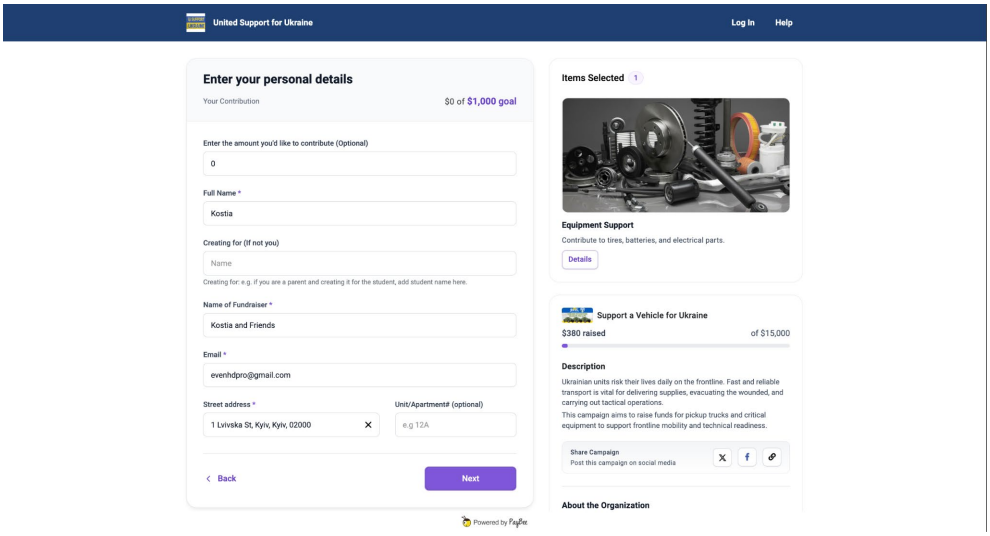

Рис.2.3. Сторінка вибраної опції для введення основної інформації

Після натискання кнопки “Next” відбувається перехід на сторінку для введення додаткової інформації. Тут користувачеві потрібно ввести додаткову інформацію у вигляді текстової відповіді на питання та (або) відповіді на питання у вигляді вибору варіанту зі списку (рис. 2.4).

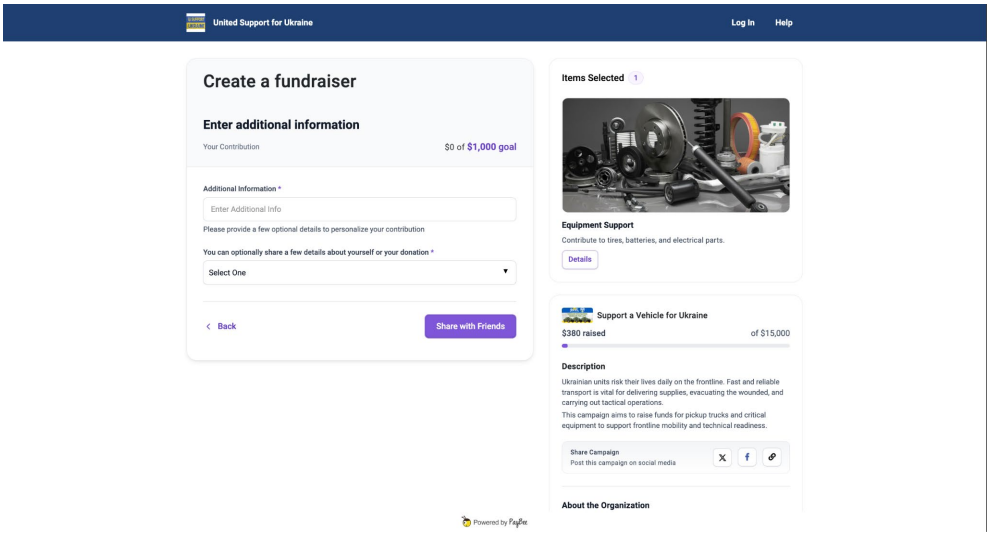

Рис.2.4. Сторінка вибраної опції для введення додаткової інформації

Після натискання кнопки “Share with friends” відбувається перехід на сторінку для оплати вибраної опції. На цьому етапі потрібно ввести дані

карти, з якої буде проводитися оплата та вибрати чи бажаєте покривати платіжну комісію замість організації (рис. 2.5).

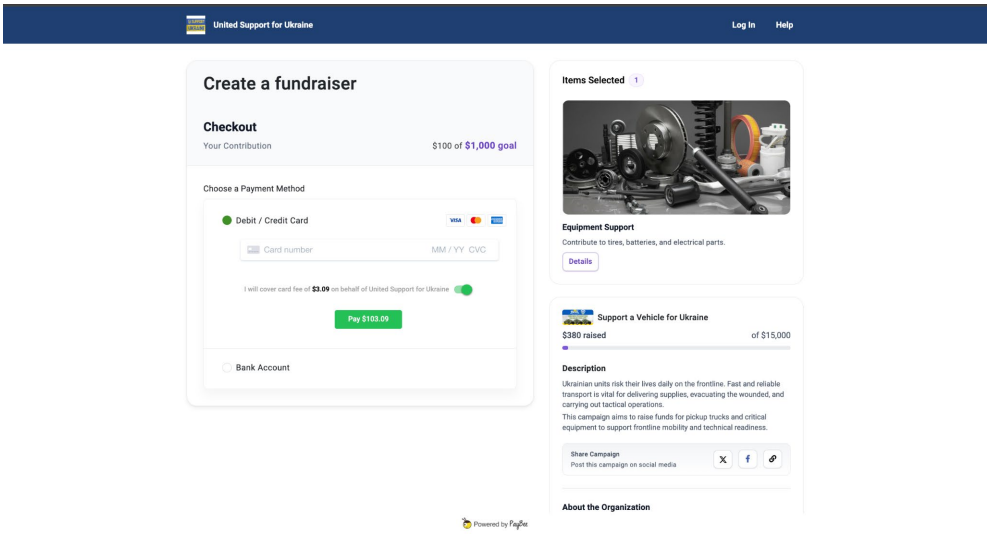

Рис.2.5. Сторінка оплати вибраної опції

Після успішної оплати відбувається перехід на сторінку створеного фандрейзера. На цій сторінці зображена уся інформація про створеного фандрейзера, можливість її редагування, поширення та запрошення друзів (рис. 2.6).

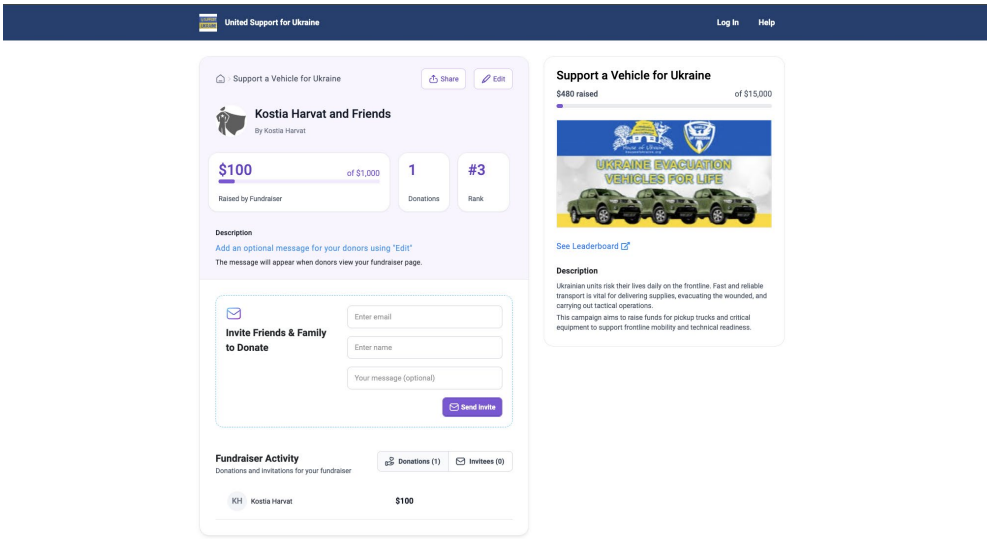

Рис.2.6. Сторінка створеного фандрейзера

При натисканні на кнопку “Share” відкривається список варіантів, як саме можна поділитися своїм фандрейзером: скачати qr-коди та роздрукувати

їх, поділитися у X чи Facebook або ж скопіювати посилання на сторінку (рис. 2.7).

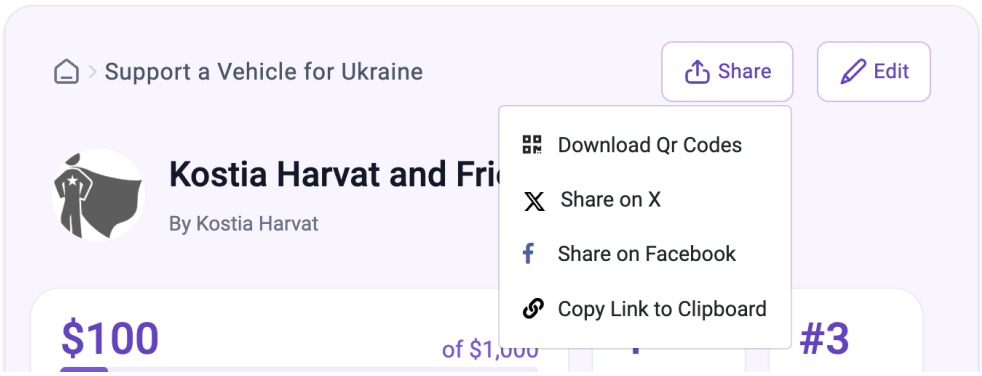

Рис.2.7. Вміст кнопки "Share"

При натисканні на варіант "Download Qr Codes" відкривається модальне вікно для скачування, у якому також описується як саме виглядають qr-коди та як їх потрібно друкувати (рис. 2.8).

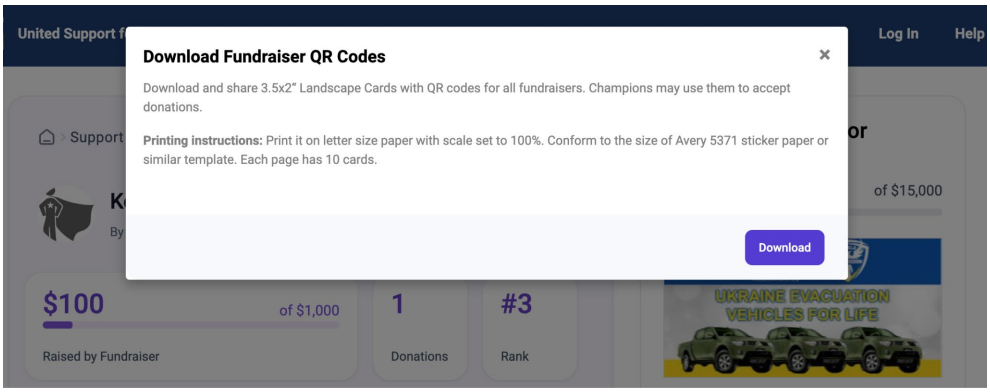

Рис. 2.8. Модальне вікно "Download QR Codes"

При натисканні на кнопку "Edit" показується секція для редагування інформації фандрейзера. Тут можна змінити ім'я фандрейзера, ім'я донора, фото, ціль та опис (рис. 2.9).

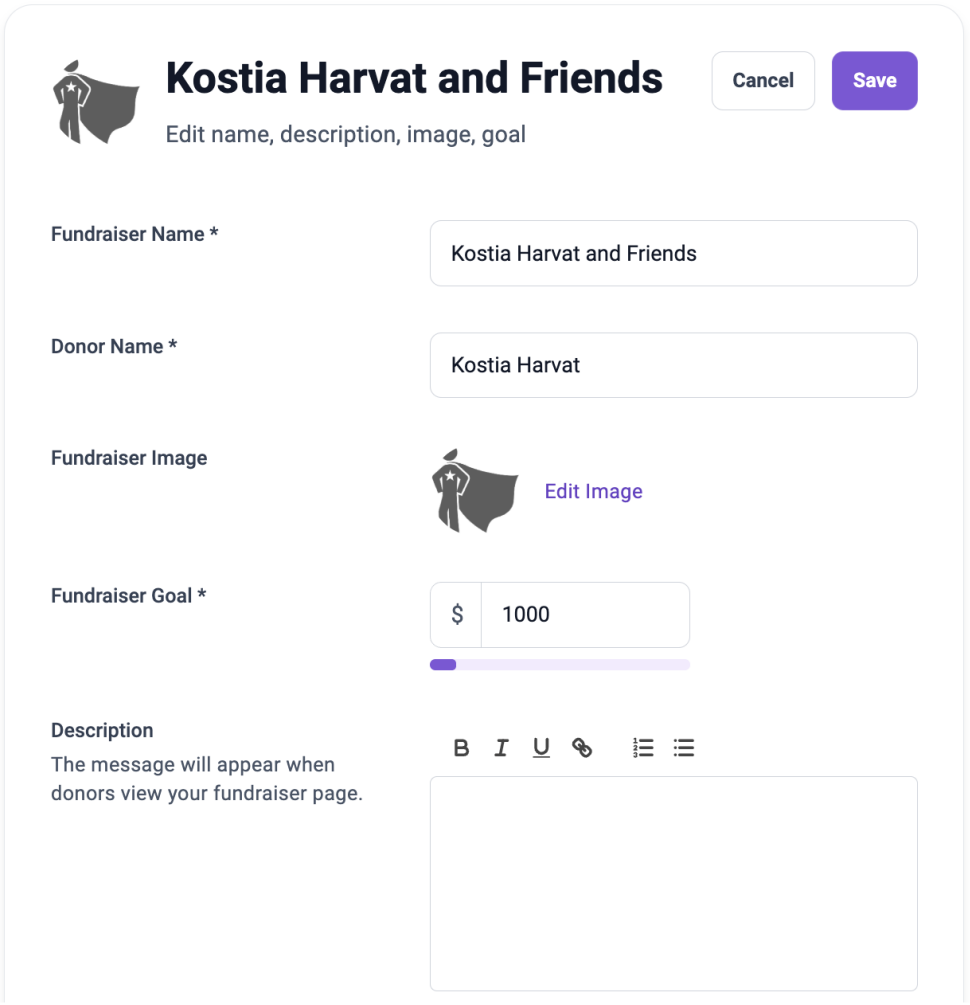

Рис. 2.9. Вікно редагування фандрейзера

Для відправлення запрошення на пошту використовується секція “Invite Friends and Family to Donate”. Потрібно заповнити пошту, ім’я та повідомлення (за бажанням) і натиснути кнопку “Send Invite”, після чого запрошення буде відправлено та показано відповідне повідомлення у куті екрану (рис. 2.10, рис 2.11).

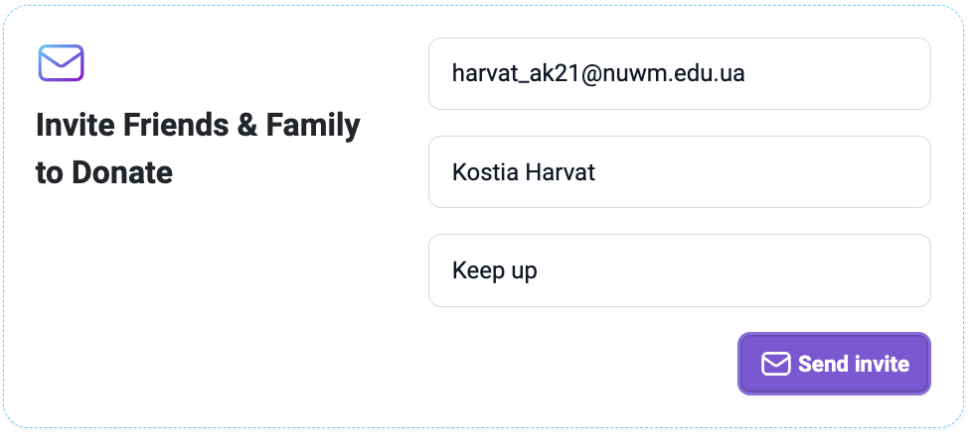

Рис. 2.10. Форма для заповнення даних для запрошення на пошту

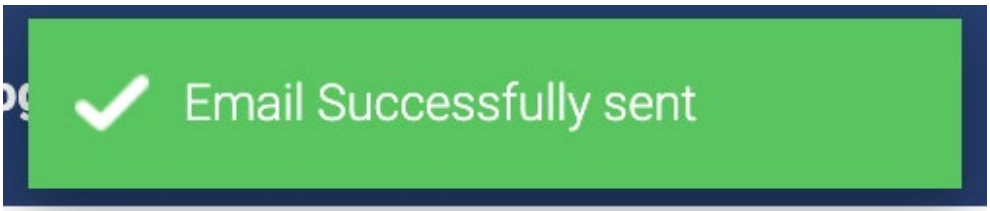

Рис. 2.11. Відображення повідомлення (toast) про успішну відправку електронного листа у верхньому правому куті інтерфейсу

У секції “Fundraiser Activity” на вкладці “Donations” відображаються донати до фандрейзера від інших користувачів, а саме: ім’я користувача, сума донату, повідомлення, яке залишив користувач під час донату та кнопка “Thank Them” для відправлення вдячності за донат (рис. 2.12).

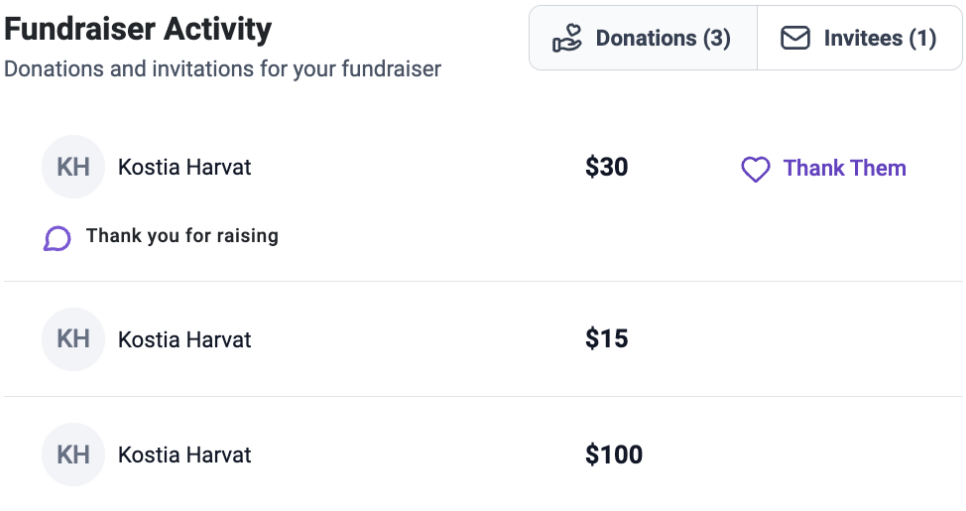

Рис. 2.12. Секція з інформацією про отримані донати

При натисканні кнопки “Thank Them” відкривається модальне вікно для відправлення подяки у якому знаходиться поле для тексту подяки, який можна ввести (рис. 2.13).

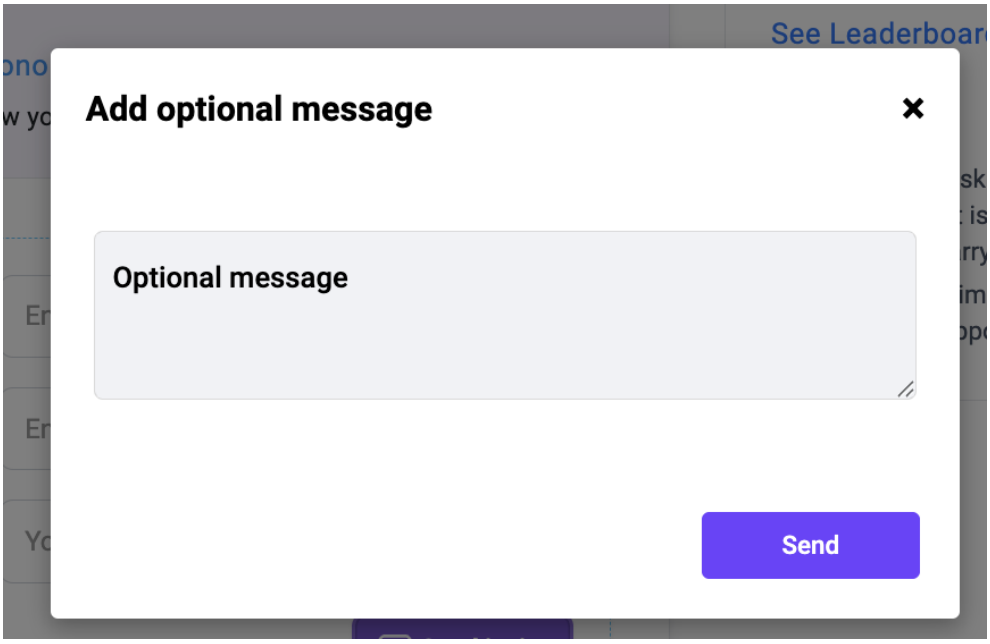

Рис. 2.13. Модальне вікно Thank Them

Після після успішного виконання “Thank Them”, текст кнопки змінюється на “Thanked” та, при наведенні курсором на інконку, показується повідомлення та дата подяки (рис. 2.14).

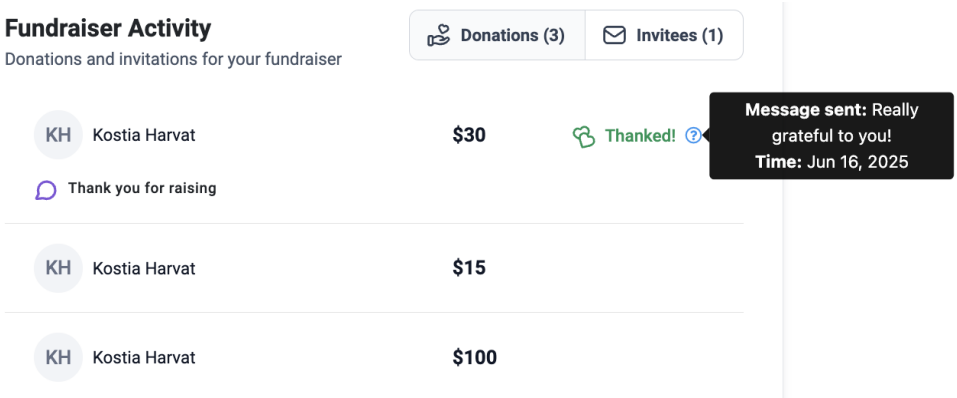

Рис. 2.14. Вигляд секції з інформацією про отримані донати після відправлення подяки

У секції “Fundraiser Activity” на вкладці “Invitees” відображаються запрошені користувачі для донату до фандрейзера та кнопка “Send Reminder” біля кожного (рис. 2.15).

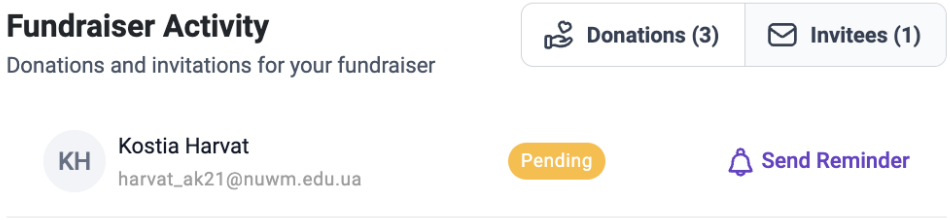

Рис. 2.15. Відображення секції з відправленими запрошеннями

При натисканні кнопки “Send Reminder” відправляється нагадування запрошеному користувачу, а під кнопкою з’являється текст з датою останнього нагадування (рис. 2.16).

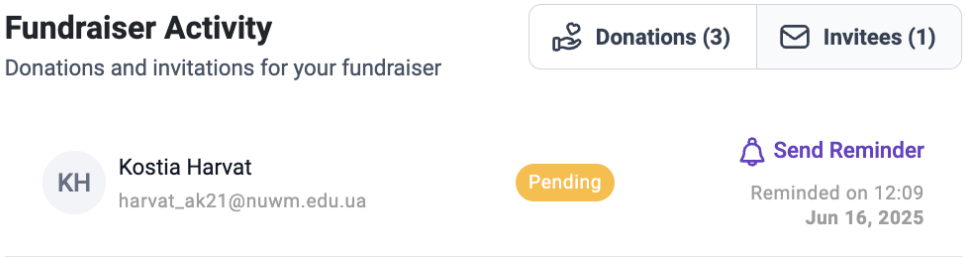

Рис. 2.16. Вигляд секції з відправленими запрошеннями після відправлення нагадування

Після того, як фандрейзер був створений можна використовувати посилання на сторінку донату. На цій сторінці відображається ім’я фандрейзера та хто його створив, прогрес збору, кількість донатів, опції для донату та список останніх до антів користувачів (рис. 2.17).

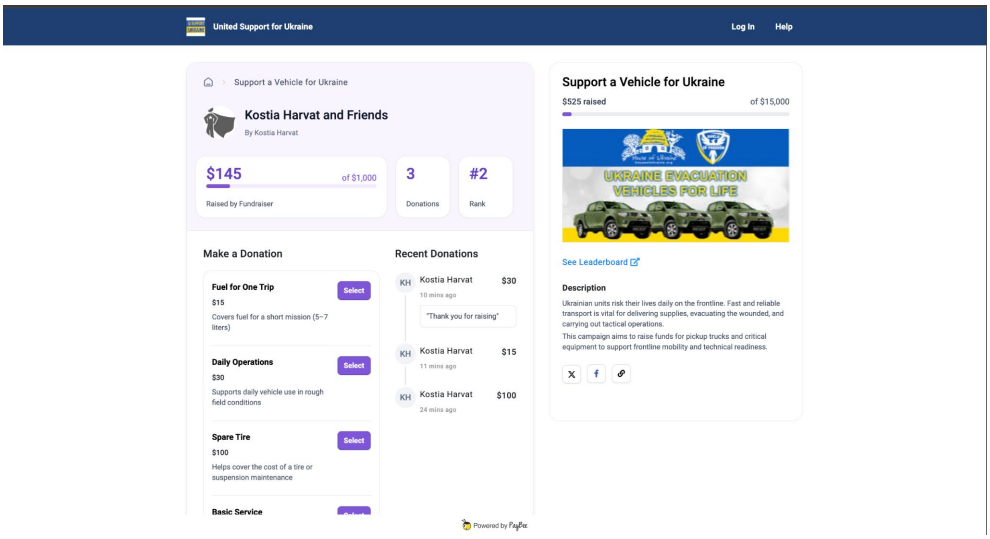

Рис. 2.17. Інформація про розробника ІС

При натисканні кнопки “Select” біля вибраної опції, користувач переходить на сторінку для введення інформації для донату, а саме: своє ім’я, електронну пошту та повідомлення (за бажанням) (рис. 2.18).

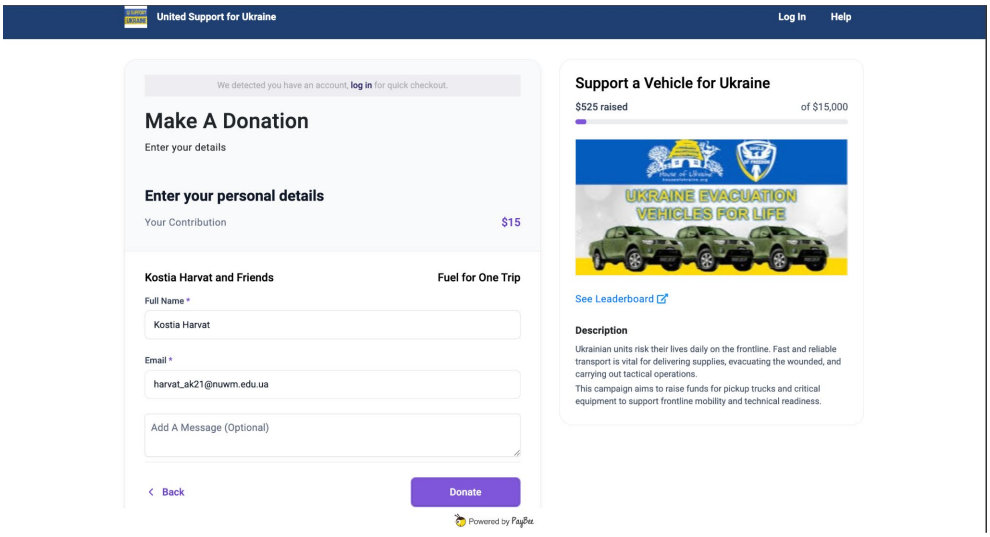

Рис. 2.18.Сторінка для введення основної інформації для донату фандрейзеру

При натисканні кнопки “Donate” відбувається перехід на сторінку для оплати вибраної опції. На цьому етапі потрібно ввести дані карти, з якої буде проводитися оплата та вибрати чи бажаєте покривати платіжну комісію замість організації (рис. 2.19).

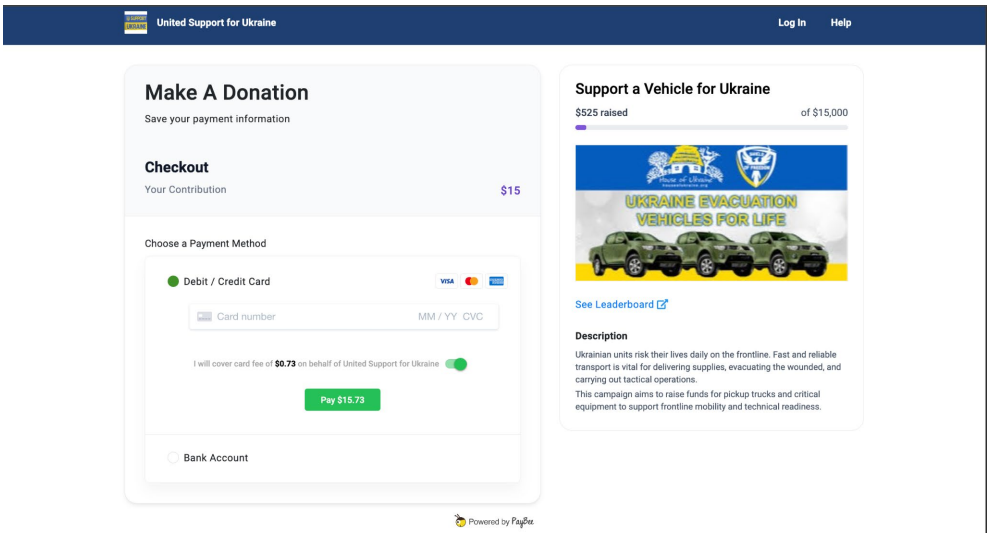

Рис. 2.19. Сторінка оплати доната для фандрейзера

Вигляд сторінки після успішної оплати (рис. 2.20).

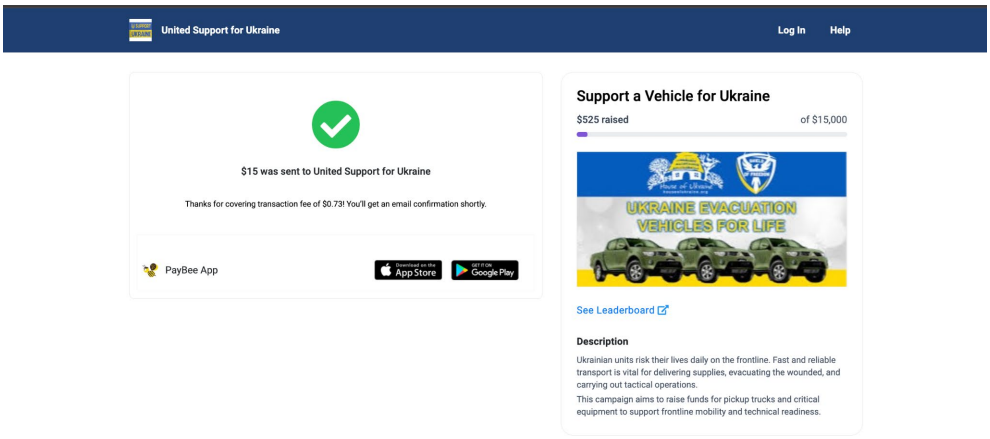

Рис. 2.20. Сторінка успішної оплати доната для фандрейзера

Тепер передемо до адміністративної частини сайту. Для управління кампанією зі сторони організації є розроблений спеціальний інтерфейс. На рисунку 2.21 зображена сторінка зі списком усіх Peer To Peer кампаній конкретної організації.

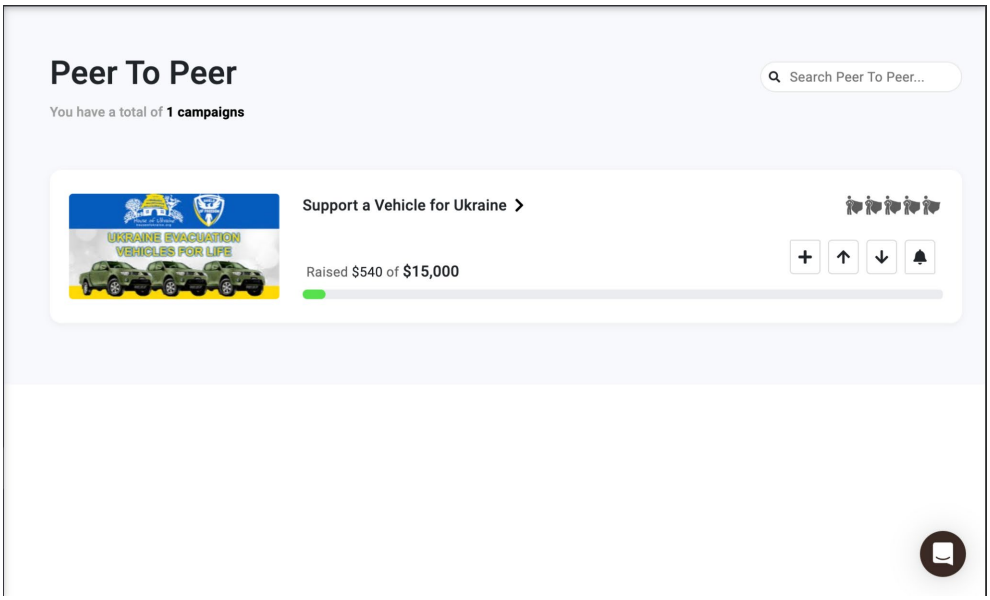

Рис. 2.21. Список усіх Peer To Peer кампаній організації

Біля кожної кампанії зображені чотири кнопки з швидкими діями (рис. 2.22). Кнопка “Create Fundraiser” відкриває модальне вікно для створення фандрейзера (рис. 2.23). Кнопка “Import Fundraiser” відкриває модальне вікно для імпорту фандрейзерів через Excel файл (рис. 2.24). Кнопка “Download Qr Codes” відкриває модальне вікно для завантаження qr-кодів з посиланням на фандрейзера на стороні донора (рис. 2.25). Кнопка “Notify All” відкриває модальне вікно для посилання нагадуванням усім фандрейзерам про поточний статус збору (рис. 2.26).

Рис. 2.22. Кнопки швидкої дії для P2P кампанії

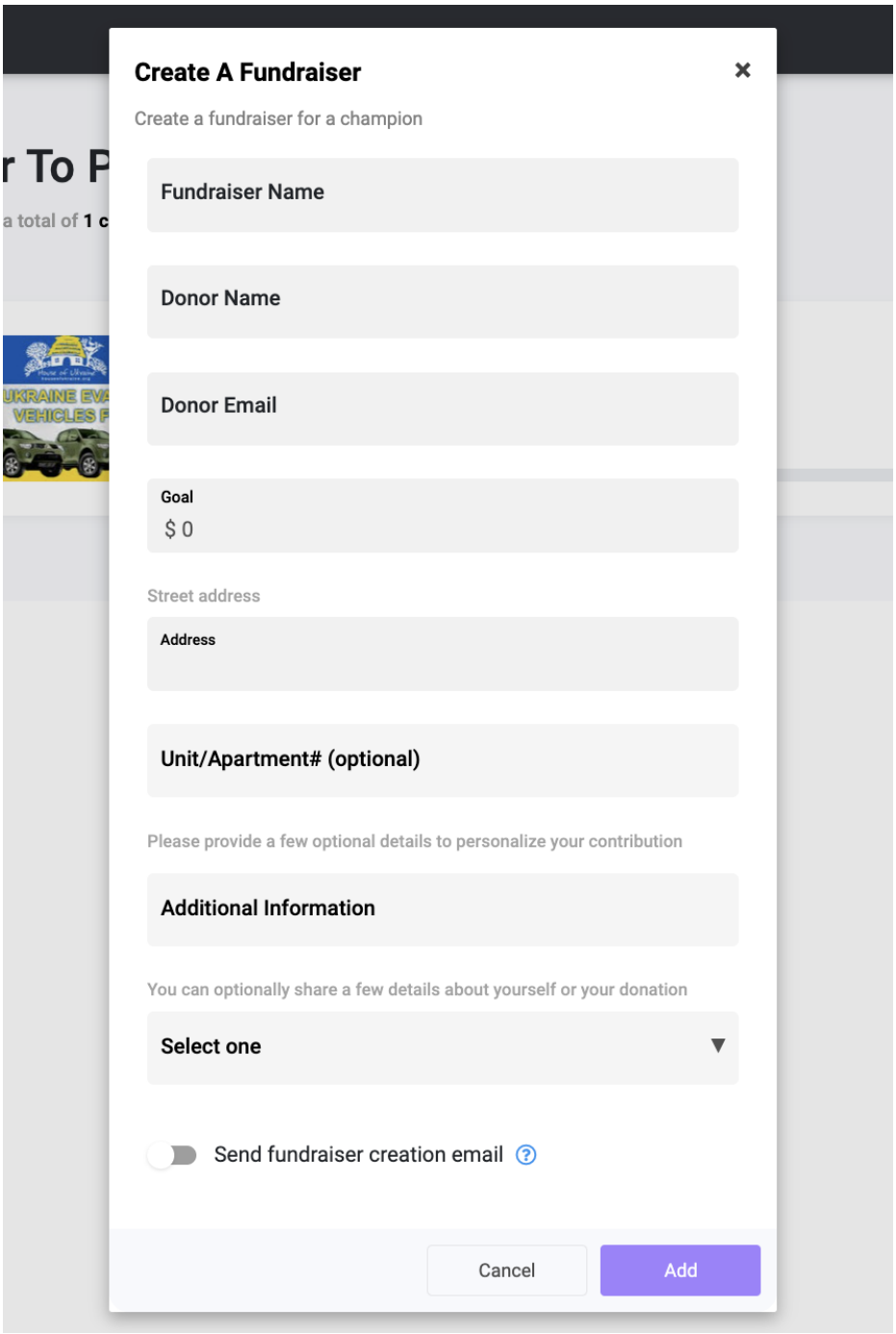

Рис. 2.23. Модальне вікно для створення фандрейзера

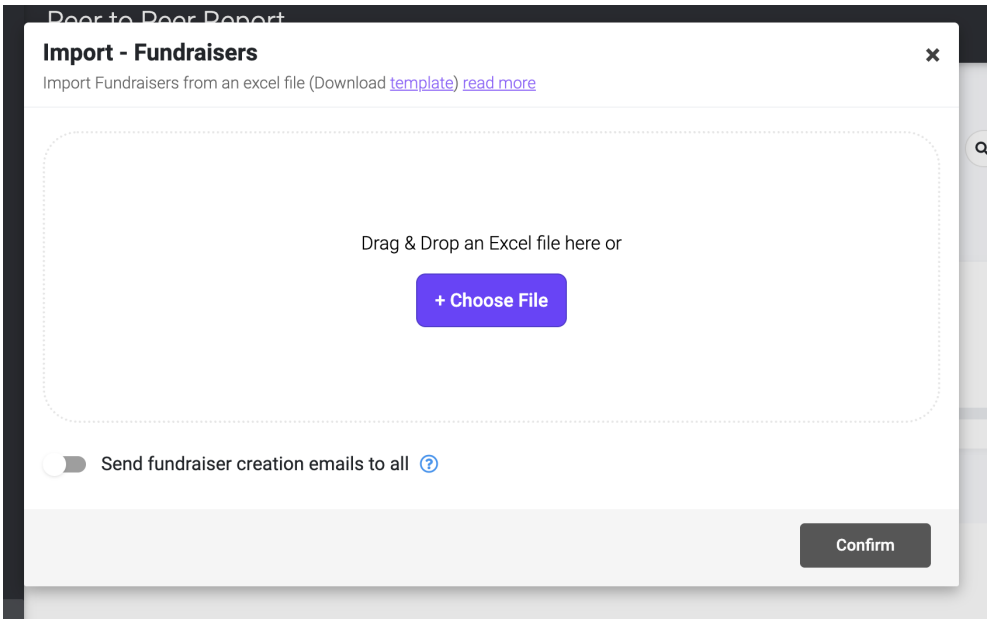

Рис. 2.24. Модальне вікно для імпорту фандрейзерів через Excel

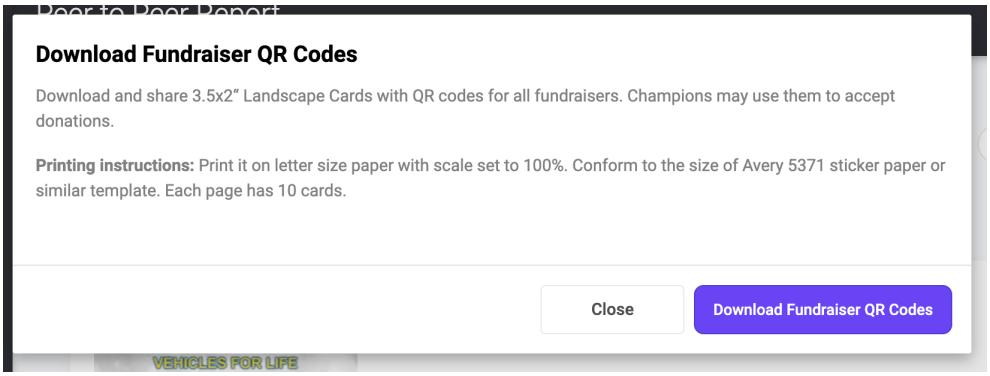

Рис. 2.25. Модальне вікно для завантаження qr-кодів

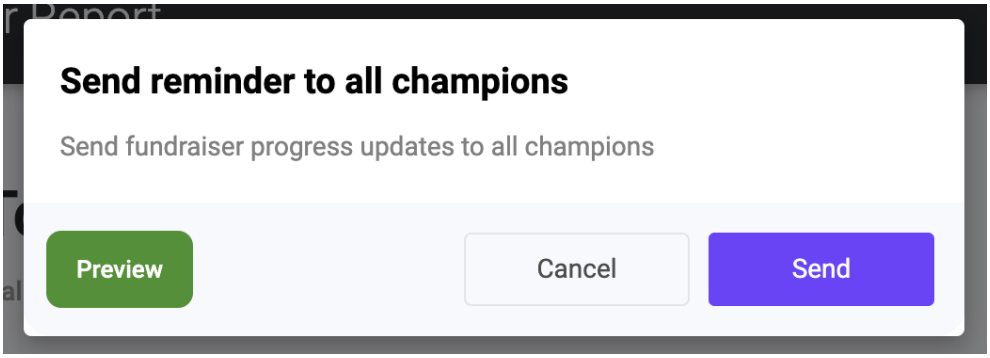

Рис. 2.26. Модальне вікно для нагадування усіх фандрейзерів

Результат роботи кнопки “Preview” (рис. 2.27).

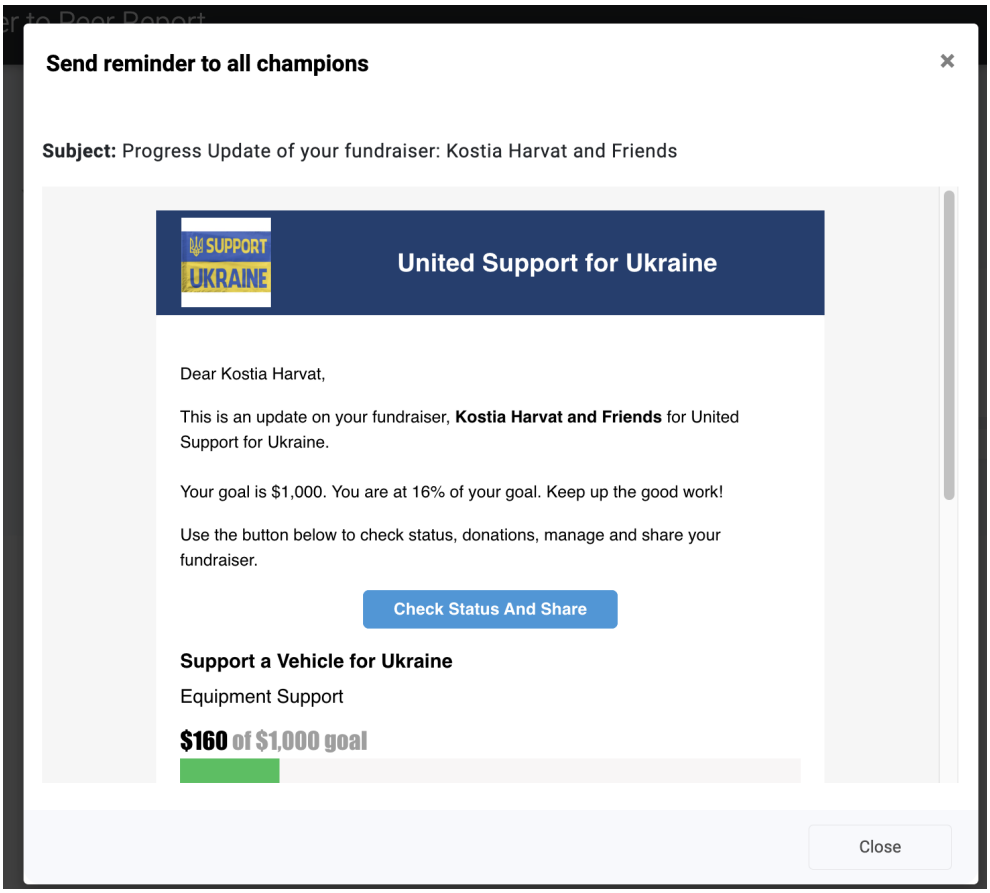

Рис. 2.27. Модальне вікно попереднього перегляду нагадування

Після натискання на вибрану кампанію користувач переходить на сторінку управління нею (рис 2.28). На цій знаходиться уся інформація про кампанію: ім’я, фото, прогрес, аналогічні кнопки швидких дій, які були зображені на рисунку 2.22. Також присутні вкладки “Fundraisers”, “Reminders”, “Fundraisers Report”, “Donations Report”

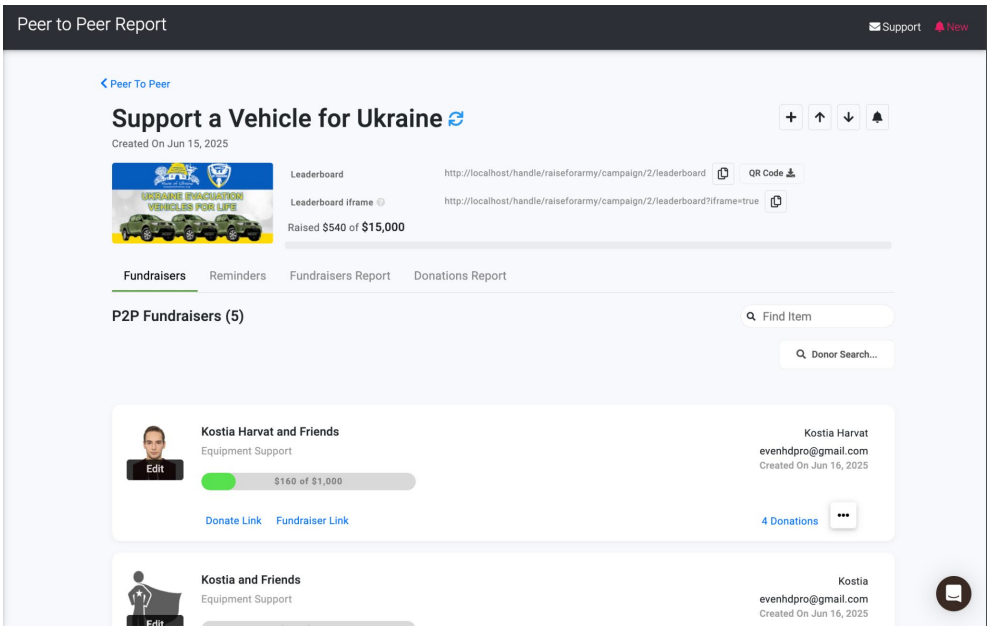

Рис. 2.28. Сторінка вибраної кампанії

Результат роботи пошуку по фандрейзерах (рис. 2.29).

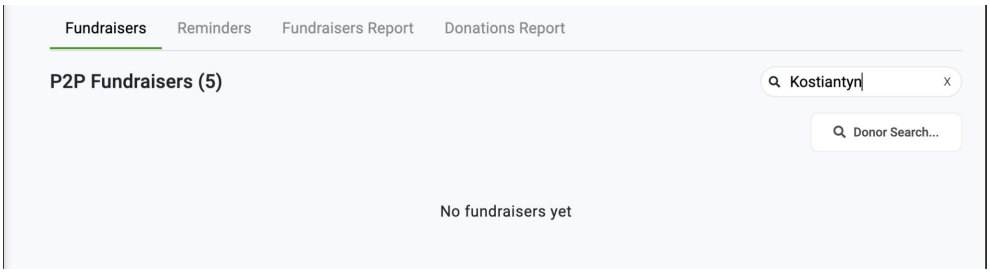

Рис. 2.29. Результат роботи пошуку по фандрейзерах

При натисканні кнопки “Donor Search” відкривається модальне вікно з пошуком по всіх донорах кампанії. Біля кожного донора відображається його ім’я та пошта, а також кількість зроблених пожертв(рис. 2.30).

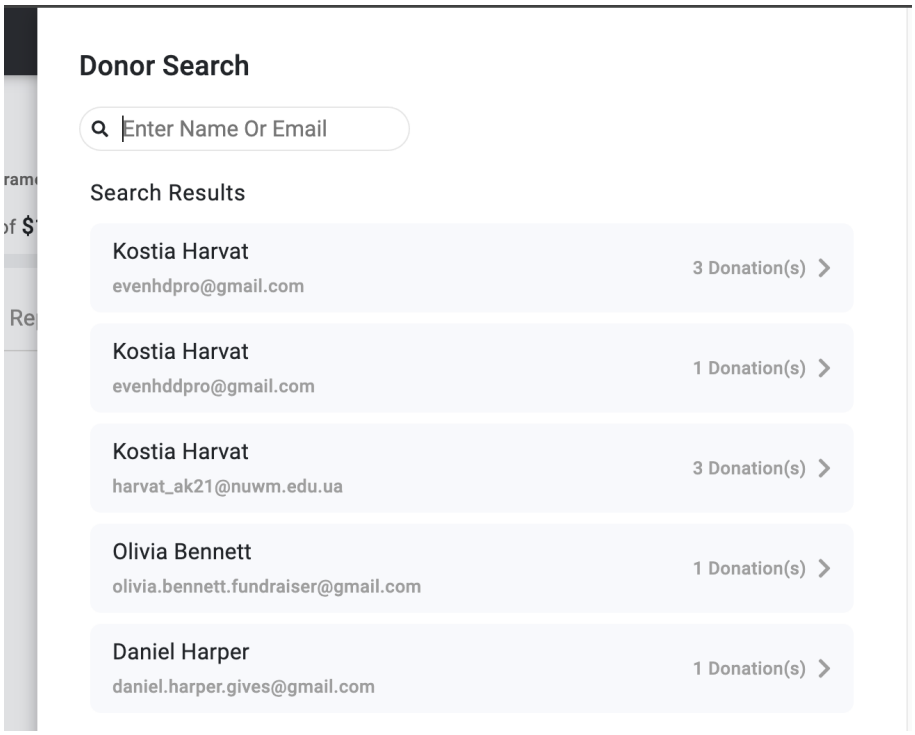

Рис. 2.30. Модальне вікно "Donor Search"

При натисканні на конкретного донора відкривається екран в модальному вікні для відображення інформації про вибраного донора (рис. 2.31). Тут можна побачити більш розширену інформацію про донора, а також список фандрейзерів, яким донор робив пожертви. Біля кожного фандрейзера видно його ціль, прогрес та кількість донатів конкретно від вибраного донора.

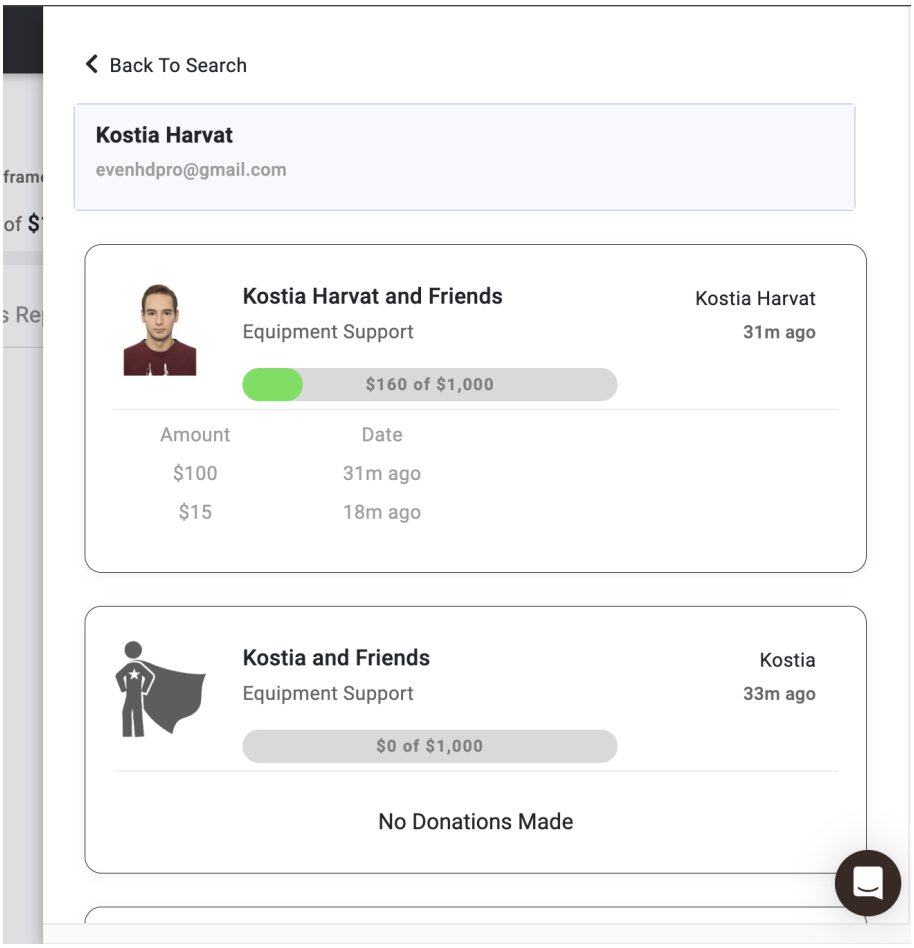

Рис. 2.31. Модальне вікно з інформацією про конкретного донора

При натисканні на фандрейзера відкривається модальне вікно з інформацією про вибраного фандрейзера. Тут можна побачити вибрану опцію фандрейзера, прогрес, ціль, кнопки “Add Donation” та “Export Donations” і головне - список усіх донорів даного фандрейзера з сумою та датою донату (рис. 2.32).

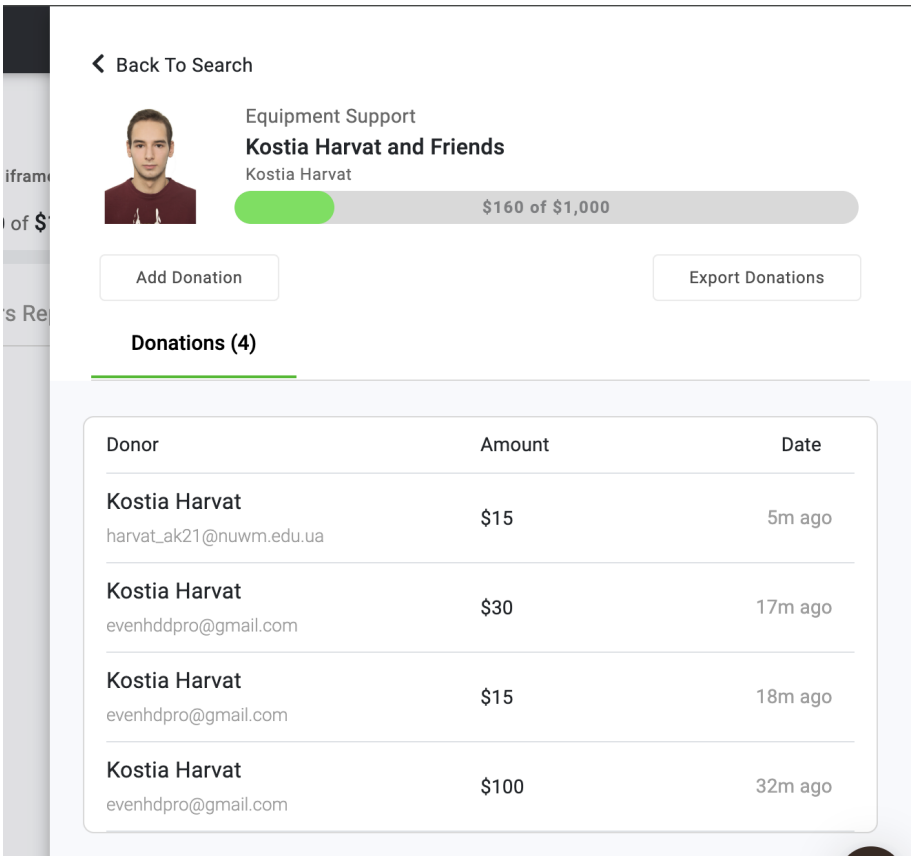

Рис. 2.32. Модальне вікно з інформацією про конкретного фандрейзера

При натисканні кнопки “Add Donation” відкривається модальне вікно для додавання донату до фандрейзера вручну. Для цього необхідно заповнити поля ім’я, електронна адреса, сума донату, дата донату та платіжний спосіб. За бажанням можна додати memo (рис. 2.33).

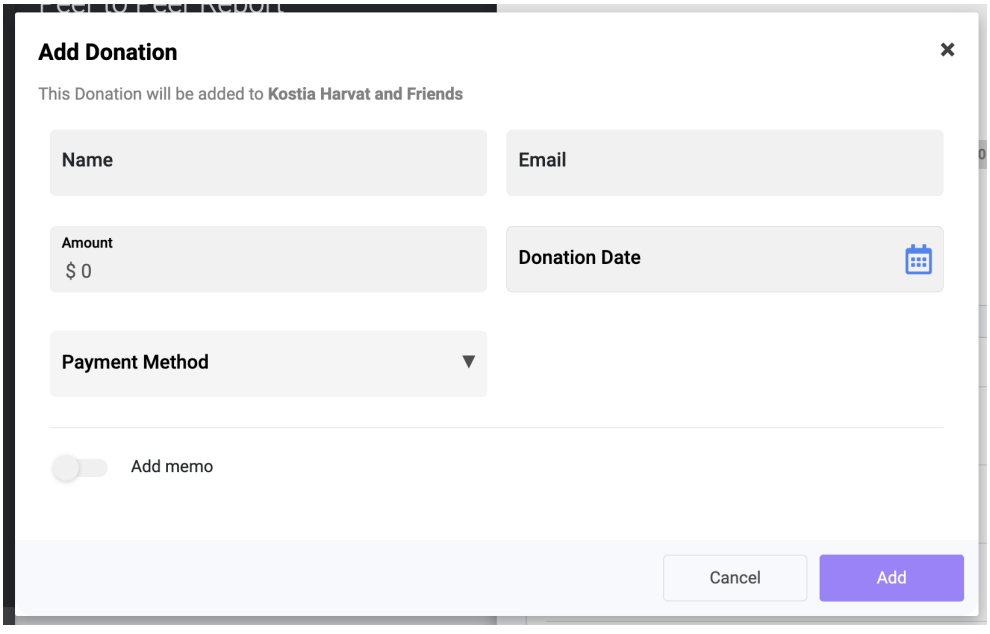

Рис. 2.33. Модальне вікно "Add Donation"

При натисканні на текст "Donate Link" або "Fundraiser Link" у конкретного фандрейзера відкривається список зі швидкими діями, який дозволяє відкрити, скопіювати або ж скачати qr-коди з посиланням на сторінку донора (рис. 2.34, рис. 2.35).

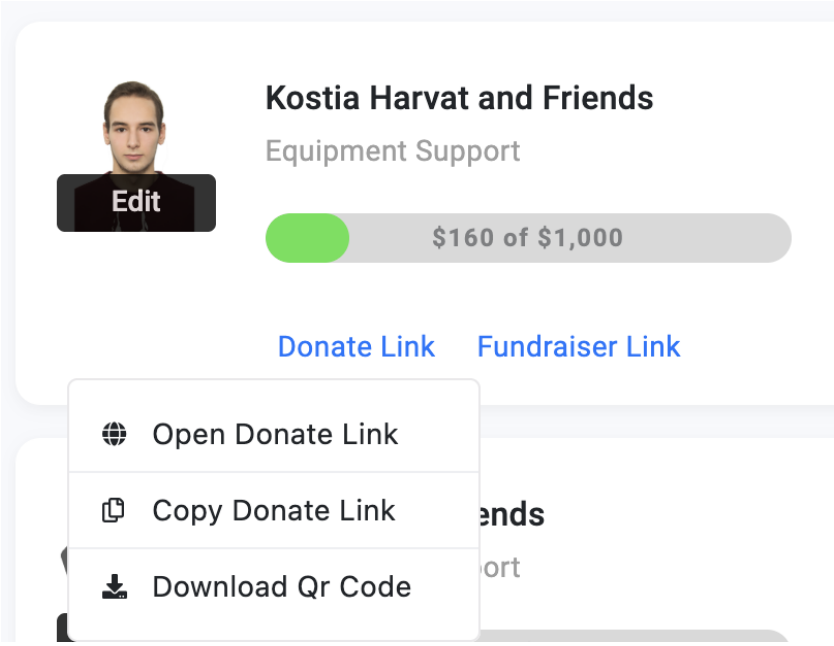

Рис. 2.34. Список швидких дій для "Donate Link"

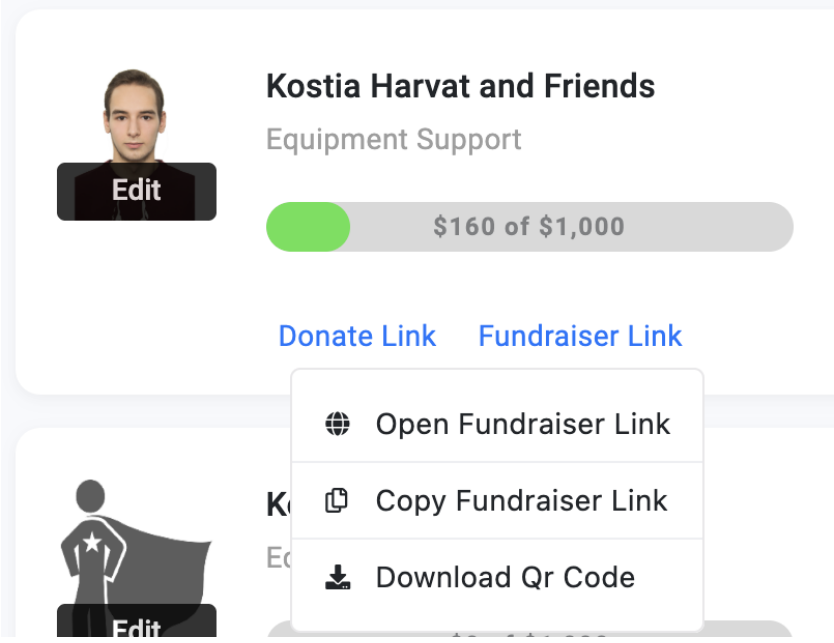

Рис. 2.35. Список швидких дій для Fundraiser Link"

Біля кожного фандрейзера присутня кнопка “...” з швидкими діями (рис. 2.36). Кнопка “Edit” відкриває модальне вікно для редагування фандрейзера (рис. 2.37). Кнопка “Notify” відкриває модальне вікно для посилання нагадуванням фандрейзеру про поточний статус збору (рис. 2.38) Кнопка “Export Donations” завантажує Excel файл з усіма донатами фандрейзера. Кнопка “Move Transactions” відкриває модальне вікно для перенесення трансакцій (донатів) від одного фандрейзера до іншого (рис. 2.39, рис. 2.40). Кнопка “Download Qr Cards” відкриває модальне вікно для завантаження qr-кодів з посиланням на фандрейзера на стороні донора. Кнопка “Delete” видаляє фандрейзера.

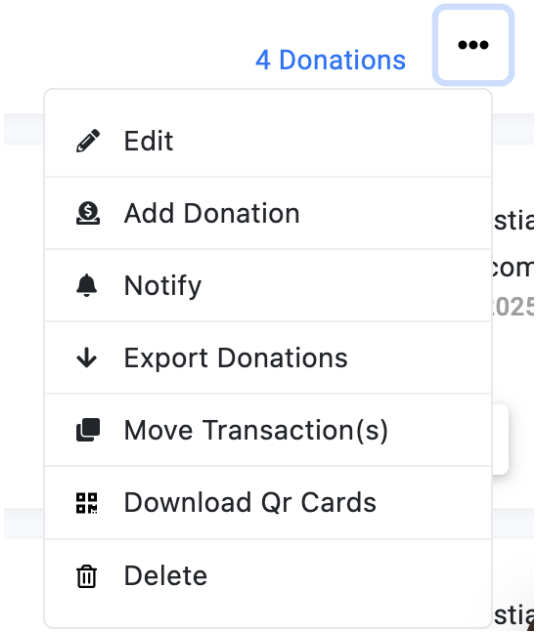

Рис. 2.36. Список дій для кнопки “...”

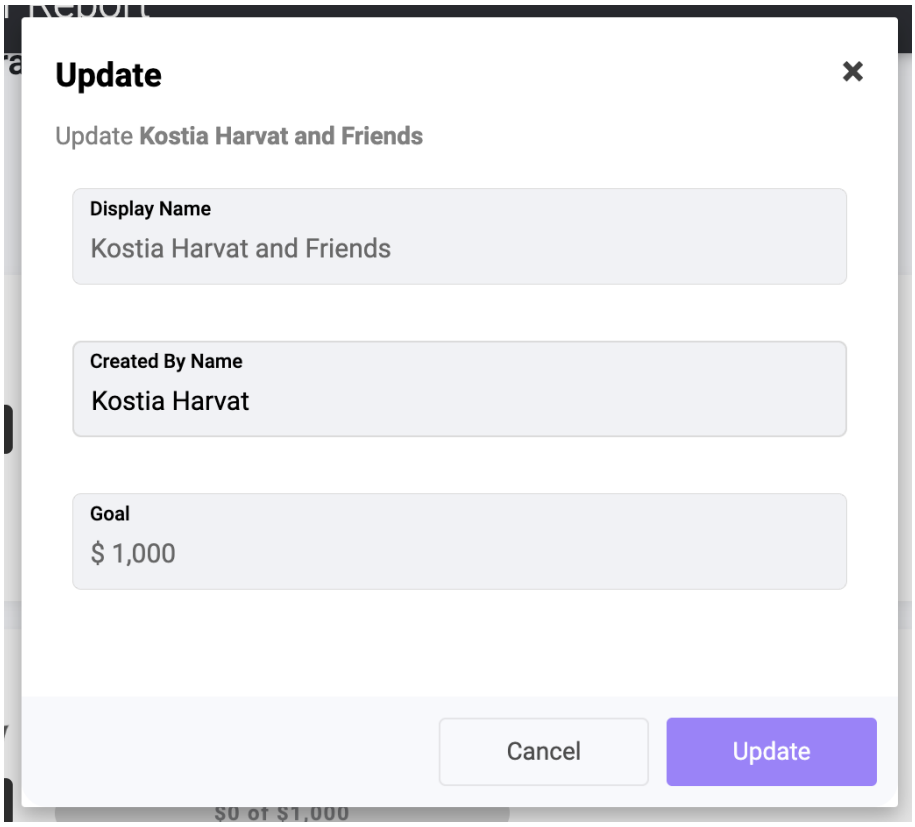

Рис. 2.37. Модальне вікно "Edit"

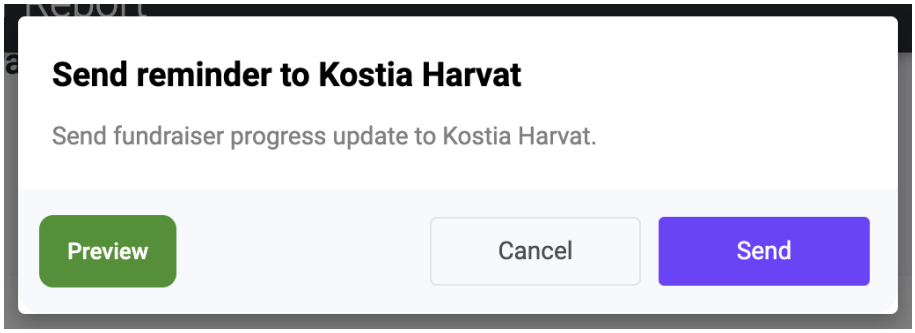

Рис. 2.38. Модальне вікно "Notify"

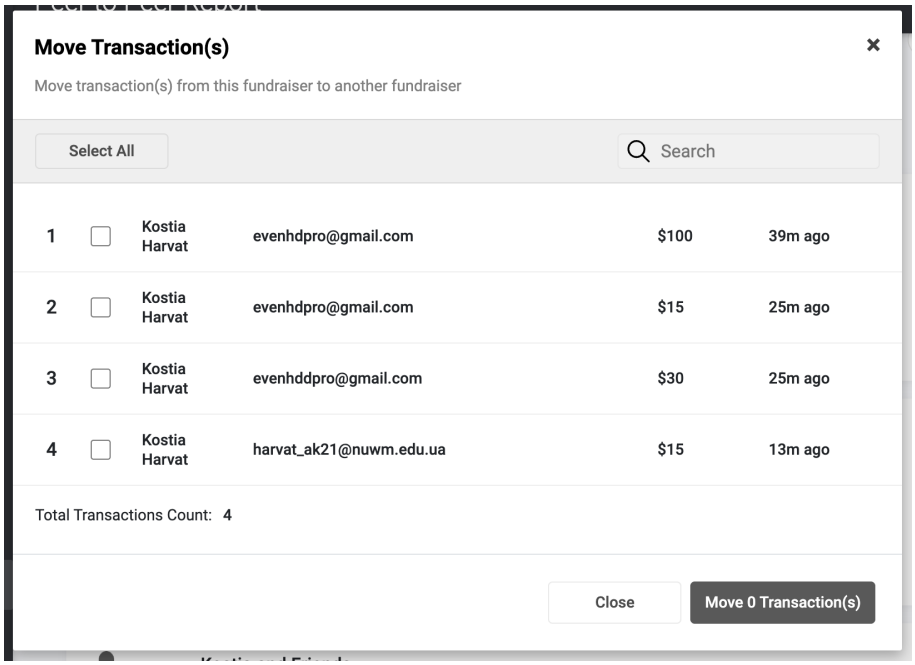

Рис. 2.39. Модальне вікно "Move transactions"

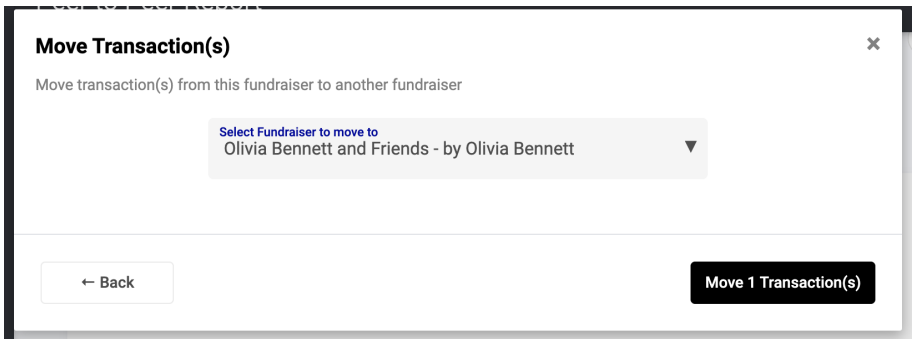

Рис. 2.40. Модальне вікно "Move transactions"

У вкладці "Reminders" знаходиться інформація про заплановані нагадування для цієї кампанії. Для планування нагадування використовується кнопка "Reminders" та вибір потрібного нагадування. У списку запланованих нагадувань відображається інформація про тип нагадування, його дату, час та часовий пояс, є кнопка для попереднього перегляду та видалення нагадування (рис 2.41).

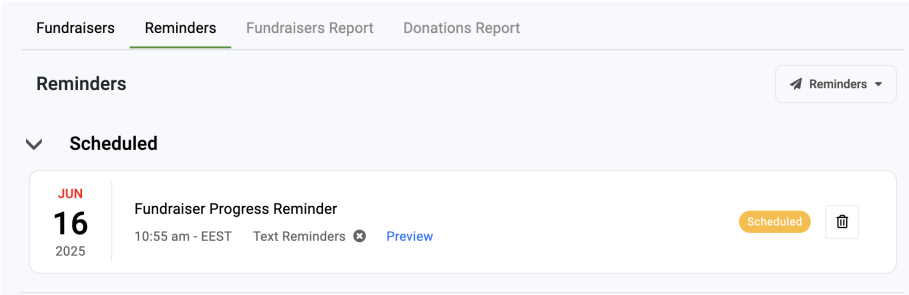

Рис. 2.41. Вкладка "Remidners"

При натисканні на кнопку "Preview" відкривається модальне вікно для попереднього перегляду нагадування (рис. 2.42).

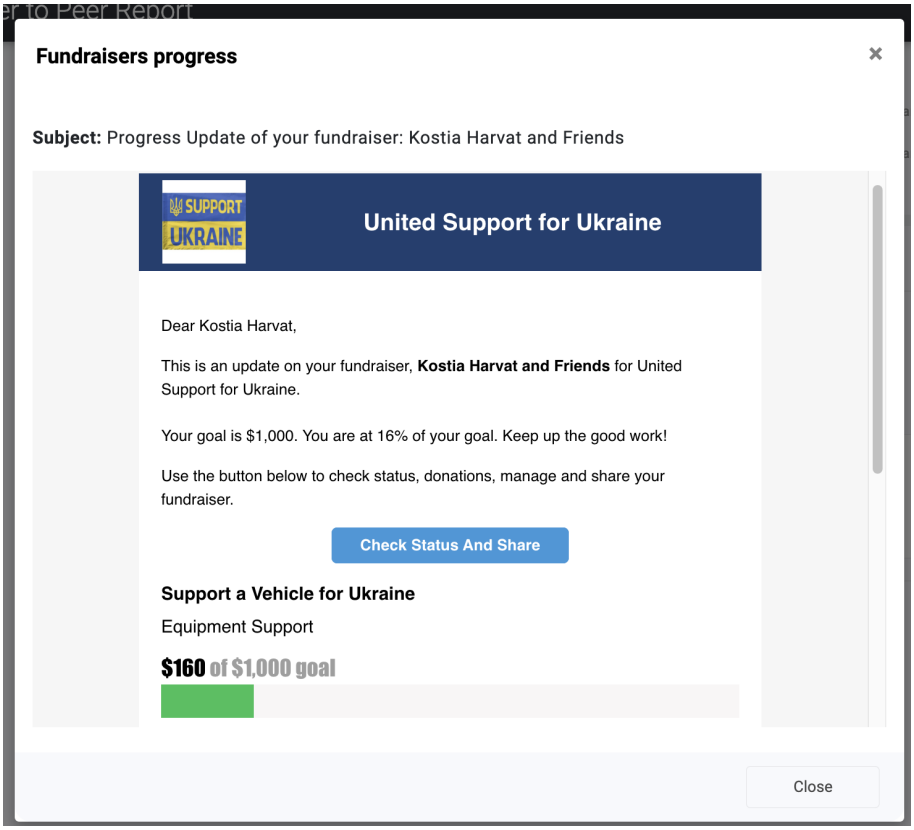

Рис. 2.42. Модальне вікно для попереднього перегляду нагадування

При натисканні на кнопку "Delete" відкривається модальне вікно з підтвердженням для видалення нагадування (рис. 2.43).

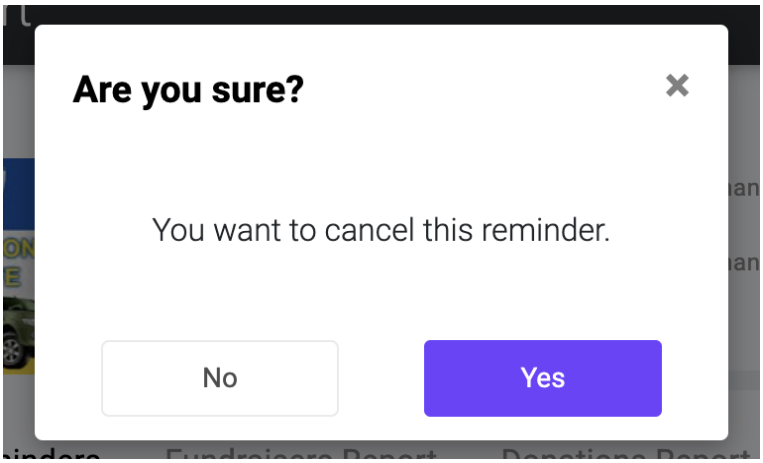

Рис. 2.43. Модальне вікно підтвердження видалення нагадування

При натисканні кнопки “Reminders” відкривається список з вибором типу нагадування, яке потрібно відправити (рис. 2.44).

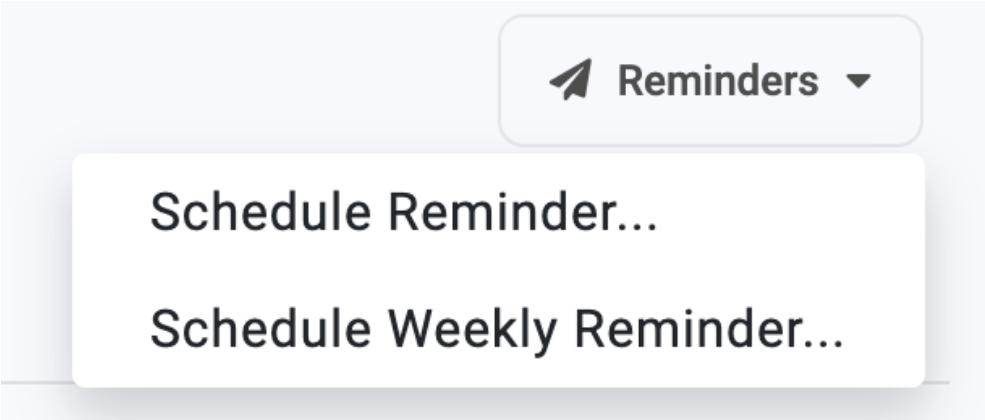

Рис. 2.44. Список дій кнопки “Reminders”

При натисканні на кнопку “Schedule Reminder” відкривається модальне вікно для планування нагадування. У ньому потрібно зазначити дату та час, часовий пояс, тему та зміст листа (рис. 2.45).

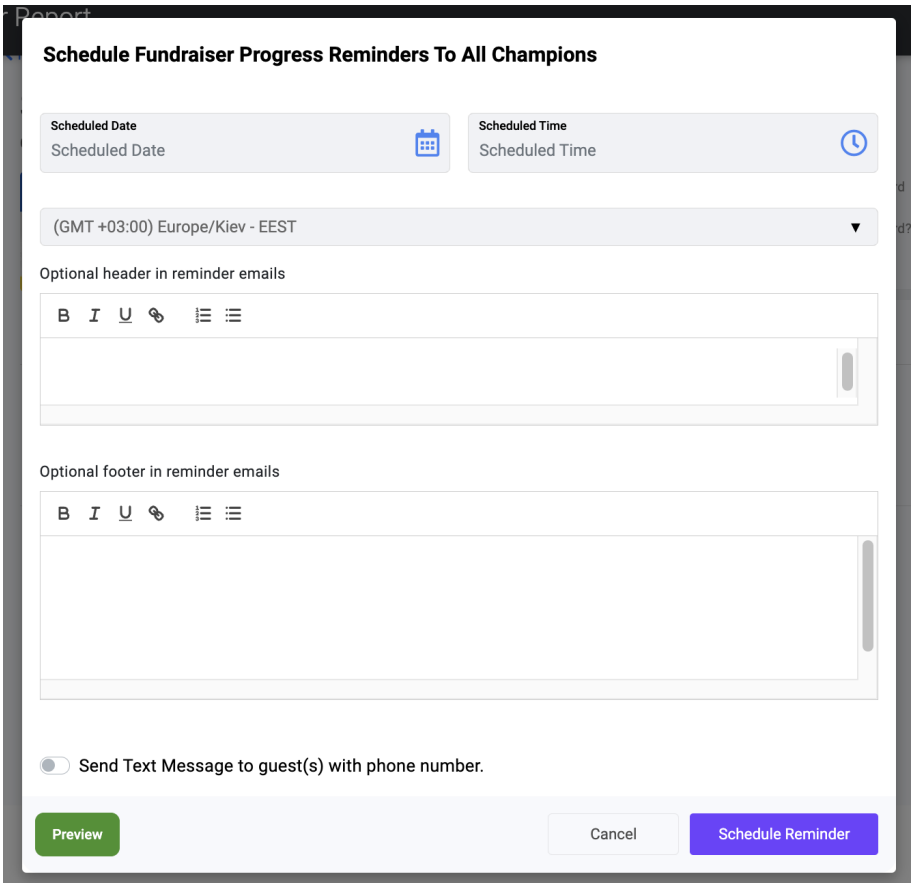

Рис. 2.45. Модальне вікно планування нагадування на конкретну дату та час

У вкладці “Fundraisers Report” представлений звіт для всіх фандрейзерів вибраної кампанії. У звіті відображається ім’я фандрейзера, ім’я та електронна адреса донора, назва опції під якою було створено фандрейзера, ціль, прогрес, дата створення, посилання на фандрейзера та додаткова інформація, яку потребує кампанія (рис 2.46).

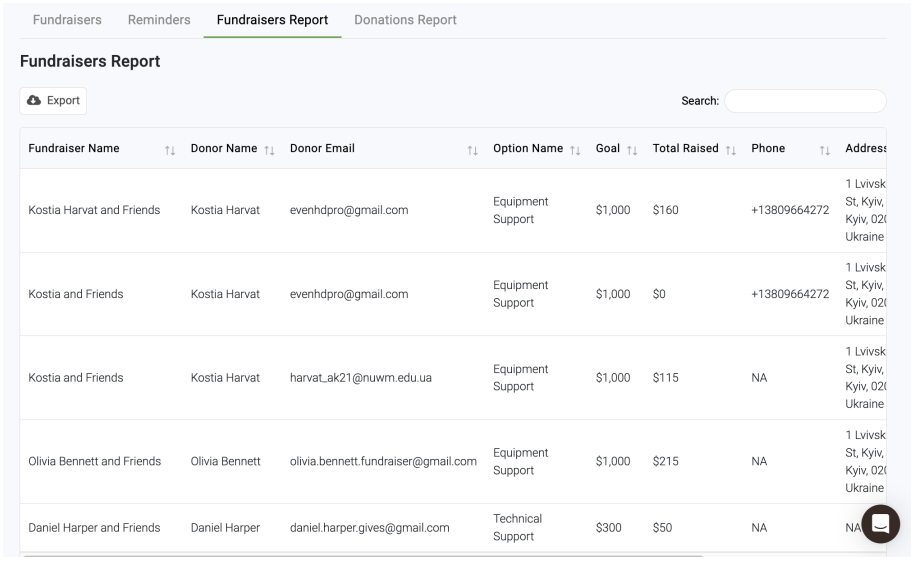

Рис. 2.46. Вигляд вкладки "Fundraiser Report"

У вкладці "Donations Report" представлений звіт для всіх пожертв вибраної кампанії. У звіті відображається ім'я та електронна адреса фандрейзера, ім'я та електронна адреса донора, сума донату, дата та додаткова введена інформація (рис 2.47).

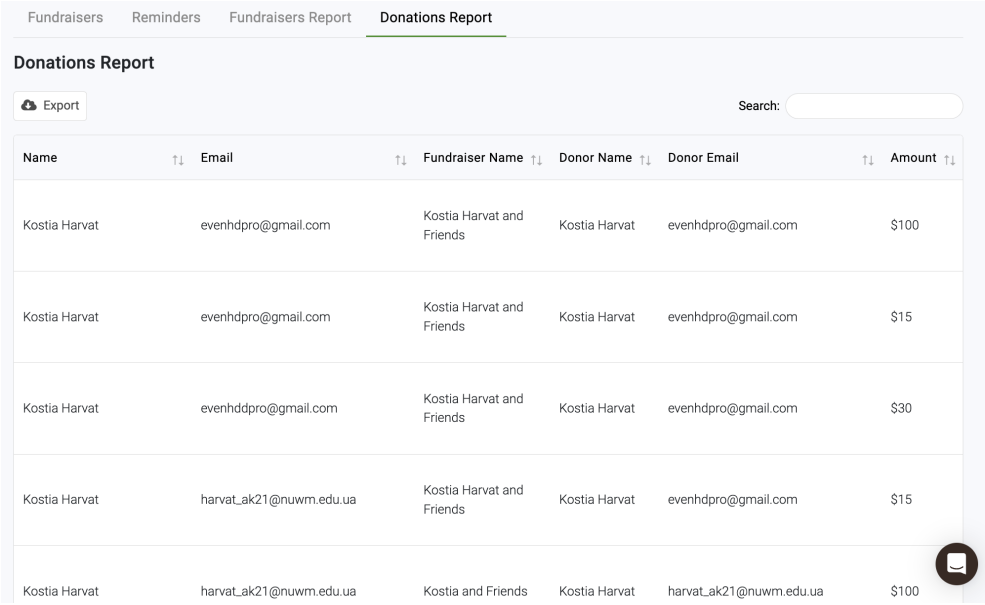

Рис. 2.47. Вигляд вкладки "Donations Report"

У верхній секції сторінки з інформацією про кампанію також відображається посилання на Leaderboard цієї кампанії. Він призначений для відображення усіх фандрейзерів у вигляді списку, щоб кожен міг вільно подивитися прогрес загального збору. При наведенні мишкою на це посилання показується додатковий текст (tooltip), див рис. 2.48.

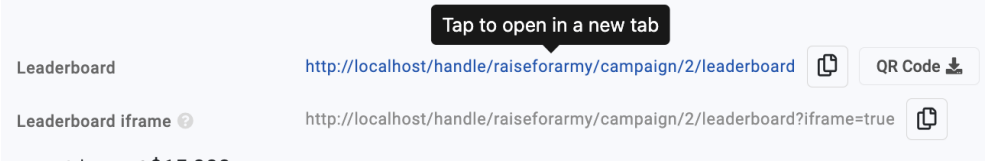

Рис. 2.48. Вигляд підказки при наведенні на текст

При натисканні на посилання користувач переходить на сторінку “Leaderboard”, де можна побачити загальний прогрес та ціль кампанії, список усіх фандрейзерів (рис. 2.50). Біля кожного фандрейзера відображається ім’я донора, прогрес та ціль збору, кількість донатів та кнопка “Donate”, при натисканні на яку можна перейти на сторінку донату для фандрейзера (рис. 2.18). Біля топ 3 фандрейзерів відображається відповідний ранг (1, 2, 3).

З правого боку можна побачити основну інформацію про кампанію.

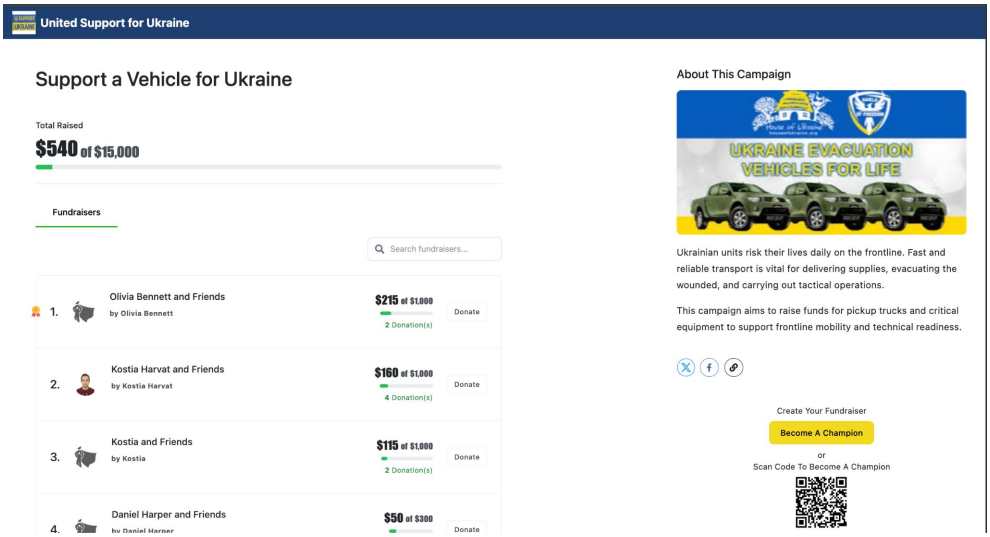

Рис. 2.49. Вигляд сторінки “Leaderboard”

Описаний сценарій взаємодії демонструє ефективність покрокового інтерфейсу, що реалізований на платформі для збору пожертв. Особливість цього підходу полягає у мінімізації когнітивного навантаження на користувача: кожен етап візуально відокремлений, логічно структурований та супроводжується чіткими підказками.

Розглянутий приклад Peer-to-Peer кампанії показує, як користувач переходить від ознайомлення зі збором до фіналізації внеску через послідовні інтерфейсні кроки: ознайомлення з опціями, вибір імен, підтвердження деталей, введення додаткової інформації та фінальну оплату. Таке поетапне ведення користувача знижує ймовірність помилок, підвищує

конверсію та забезпечує індивідуалізацію внеску - зокрема в контексті фандрейзерських ініціатив.

Таким чином, застосування покрокового підходу в інтерфейсі користувача не лише покращує UX, але й відіграє ключову роль у реалізації прозорого, адаптивного та ефективного механізму взаємодії в межах благодійної платформи.

ВИСНОВКИ

У ході виконання бакалаврської роботи було здійснено повний цикл дослідження, проєктування та реалізації користувацької частини веб-платформи для краудфандингу соціальних проєктів, з фокусом на підтримку моделі Peer-to-Peer кампаній. Проведено глибокий аналіз предметної області, що дозволив виявити актуальні виклики та вимоги, характерні для сучасних краудфандингових ініціатив, зокрема в українському контексті.

У теоретичній частині дослідження було узагальнено поняття краудфандингу, охарактеризовано його основні моделі (donation, reward, equity, debt, peer-to-peer) та проаналізовано тенденції розвитку галузі, включаючи динаміку популярності ключових фреймворків Angular і React. Виконано порівняльний аналіз зазначених інструментів на основі архітектурних, технічних, UX/UI та продуктивнісних характеристик, що дозволило обґрунтувати вибір Angular для створення адміністративної частини платформи, а jQuery – для реалізації фронтенду донорської частини.

Практична частина передбачала проєктування інформаційної системи, створення її функціональної структури, а також реалізацію інтерфейсів з урахуванням вимог до масштабованості, зручності та ефективності користувацької взаємодії. Були розроблені окремі модулі для обробки пожертв, керування кампаніями, відображення фандрейзерів, обробки нагадувань та звітів з необхідною інформацією. Особливу увагу було приділено створенню зручного покрокового інтерфейсу внесків, що є ключовим у моделі Peer-to-Peer.

Використання Angular дозволило реалізувати модульну архітектуру з підтримкою маршрутизації, управлінням станом за допомогою NgRx, ефективною валідацією форм та інтеграцією з API. jQuery, своєю чергою, забезпечила швидке прототипування публічної частини з оптимізованим DOM-керуванням. У результаті вдалося створити функціональну платформу, що поєднує технічну стабільність, сучасний дизайн і підтримку фандрейзерських сценаріїв.

Таким чином, поставлену мету роботи було досягнуто: створено прототип користувацької частини веб-платформи для краудфандингу соціальних проєктів з підтримкою Peer-to-Peer моделі, що відповідає сучасним технологічним, соціальним і UX-вимогам.

У перспективі розробку можна доповнити такими напрямами:

- інтеграція аналітичного модуля зі статистикою фандрейзерів і донатів;
- адаптація інтерфейсу до мобільних платформ;
- впровадження багатомовності та локалізації;
- розширення функціоналу для взаємодії з донорами (системи подяки, бейджі, інтеграції з соцмережами);
- удосконалення системи безпеки та верифікації кампаній.

Отримані результати можуть бути використані як основа для розгортання реальної платформи соціального фінансування, а також як приклад інженерного проєкту з реалізацією сучасних підходів до frontend-розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Краудфандинг. URL: https://w.wiki/ESGB (дата звернення 07.06.2025).
2. Краудфандинг: що це таке та навіщо потрібен. URL: https://www.zen.com/ua/blog/business-uk/crowdfunding-definition-and-purpose/ (дата звернення 07.06.2025).
3. Краудфандинг: стратегії залучення коштів та перспективи краудфандингу. URL: https://welfare.green/kraudfanding-strategii-zaluchennya-koshtiv-ta-perspektivi-kraudfandingu/ (дата звернення 07.06.2025).
4. Краудфандинг: як отримати інвестиції для старту бізнесу. URL: https://rates.fm/ua-uk/invest/kraudfanding-yak-otrimati-investiciyi-dlya-startu-biznesu/ (дата звернення 07.06.2025).
5. Angular проти React. URL: https://foxminded.ua/angular-vs-react/ (дата звернення 07.06.2025).
6. Офіційна документація jQuery API. URL: https://api.jquery.com/ (дата звернення 07.06.2025).
7. Офіційна документація Angular 17. URL: https://angular.io/docs (дата звернення 07.06.2025).
8. Офіційна документація React. URL: https://react.dev/learn (дата звернення 07.06.2025).
9. Angular vs React: Which to Choose for Your Front End in 2024?. URL: https://www.simform.com/blog/angular-vs-react/ (дата звернення 07.06.2025).
10. Angular vs React: A Detailed Side-by-Side Comparison. URL: https://kinsta.com/blog/angular-vs-react/ (дата звернення 07.06.2025).
11. Usage statistics and market share of Angular for websites. URL: https://w3techs.com/technologies/details/js-angularjs (дата звернення 07.06.2025).
12. Usage statistics and market share of React for websites. URL: https://w3techs.com/technologies/details/js-react (дата звернення 07.06.2025).
13. Angular Usage Statistics. URL: https://trends.builtwith.com/framework/Angular (дата звернення 07.06.2025).

14. React Usage Statistics. URL: https://trends.builtwith.com/javascript/React (дата звернення 07.06.2025).
15. Google Trends - React. URL: https://trends.google.com/trends/explore?date=now%201-d&q=React&hl=en (дата звернення 07.06.2025).
16. Google Trends - Angular. URL: https://trends.google.com/trends/explore?date=now%201-d&q=Angular&hl=en (дата звернення 07.06.2025).
17. Платформа KOLO. URL: https://www.koloua.com/ (дата звернення 07.06.2025).
18. Фонд Сергія Притули. URL: https://www.koloua.com/ (дата звернення 07.06.2025).
19. Що таке «Банка» в Monobank та як вона працює. URL: https://rates.fm/ua-uk/banks/sho-take-banka-v-monobank-ta-yak-vona-pracyuye/ (дата звернення 07.06.2025).
20. NgRx Docs. URL: https://ngrx.io/docs (дата звернення 07.06.2025).

ДОДАТКИ

Додаток А Код функції для відображення опцій кампані на стороні користувача

```
QuickPledge.prototype.loadOptionsWithData = function (selectedCampaign, push) {
   const self = this
   const presetOptionListFilterService = window.presetOptionListFilterService
   let presetOptions = selectedCampaign.campaignTypeData.presetOptions || [];
   const searchParams = new URLSearchParams(window.location.search);

   quickPledge.embedded = searchParams.get("embedded") === "true";
   quickPledge.theme = searchParams.get("theme") || null;

   const { gridView: showGridView } = selectedCampaign.campaignTypeData;
   quickPledge.showTicketingGridView = showGridView;
   quickPledge.campaignForm = [[]];

   const checkForLoadDonationConfirm = function() {
      const totalOptionsCount = self.isCustomOn() ? presetOptions.length + 1 :
presetOptions.length;

      if (totalOptionsCount === 1) {
         if (self.isCustomOn() && selectedCampaign.type ==
CAMPAIGN_TYPE_DONATION) {
            return true;
         }

         if (selectedCampaign.type == CAMPAIGN_TYPE_PEER_TO_PEER) {
            return true;
         }
      }

      return false;
```

```
    }

    presetOptions = self.getActivePresetOptions(presetOptions);
    self.setOptionListHeader(selectedCampaign, presetOptions);
      if (selectedCampaign.type == CAMPAIGN_TYPE_PEER_TO_PEER) {
       $("#p2pOptionListFundraiserNote").hide();

       const p2pData = self.peerToPeerData.id ? self.peerToPeerData: {
          id: null
       };
       if(p2pData.id) {
          $("#optionListTitleBlock").hide();
          const { pledgePerEvent } = p2pData.campaignData.campaignTypeData;
          $("#p2pOptionListFundraiserBlock").show();
          $("#p2pOptionListFundraiserTitle").text(`Make A ${pledgePerEvent ? 'Pledge':
'Donation'}`);

          const datetime = moment(closingDateTimeUnix*1000)
          const datetimeFormat = `${datetime.format("ddd, MMM DD YYYY")} at
${datetime.format("h:mma")}`;
          $("#p2pOptionListFundraiserClosingDateTime").text(datetimeFormat);

          if(pledgePerEvent) {
             $("#p2pOptionListFundraiserNote").show();
          }
       } else {
          $("#optionListTitleBlock").show();
          $("#p2pOptionListFundraiserBlock").hide();
       }
       if($("#optionListHeaderBlock").hasClass('d-sm-flex')) {
          $("#optionListHeaderBlock").addClass('d-sm-block').removeClass('d-sm-flex');
       } else {
```

```
            $("#optionListHeaderBlock").removeClass('d-sm-block').removeClass('d-sm-flex');
        }
    } else {
        $("#optionListTitleBlock").hide();
        $("#p2pOptionListFundraiserBlock").hide();
    }
    if(self.isCampaignViewGrid) {
        renderPresetOptionList = () => {
            const presetOptionList = self.getGridViewOptions(selectedCampaign,
presetOptionListFilterService.filtered);

            if (presetOptionList.html) {
                $("#presetButtons").off("click");
                $("#presetButtons").click(({ target }) => {
                    presetOptionList.clickCallback({ target });
                });
            }

            $("#presetButtons").html(`
                ${presetOptionList.html}

                <hr class="my-4">

                ${customAmountCard}
            `);

            for (let i = 0; i < self.multiTicketsSelected.length; i++) {
                const {quantity, option} = self.multiTicketsSelected[i];
                self.setTicketingCounterValue(option.id, quantity);
            }
            attachCounterEvents();
```

```
            finalHtml += presetOptionList.html;
            finalHtml += `<hr class="my-4">`
        }
        renderPresetOptionList();

    } else {
        renderPresetOptionList = () => {
            finalHtml = self.getListViewOptions(selectedCampaign,
presetOptionListFilterService.filtered);

            $("#presetButtons").html(`
                ${finalHtml}
                ${customAmountCard}
            `);

            for (let i = 0; i < self.multiTicketsSelected.length; i++) {
                const {quantity, option} = self.multiTicketsSelected[i];
                self.setTicketingCounterValue(option.id, quantity);
            }
            attachCounterEvents();

            $("#presetButtons").off("click", `[id^='openPresetOptionDetailsModal_']`);
            $("#presetButtons").on("click", `[id^='openPresetOptionDetailsModal_']`, (event) => {
                const [, presetId] = event.currentTarget.id.split("_");
                self.openPresetOptionDetailsModal(selectedCampaign, presetId);
            });
        }
        renderPresetOptionList();
    }

    $("#presetButtons").html(`
        ${finalHtml}
```

```
    ${customAmountCard}
  `);
   self.initShareTwitter('data-id', 'id', callback, "Check out this auction item and bid.");
  self.initShareFacebook('data-id', 'id', callback, "Check out this auction item and bid.");
  self.initCopyLink('data-id', 'id', callback);
```

Додаток Б. Код функції для сторінки створеного фандрейзера

```
QuickPledge.prototype.showPeerToPeerShare = function () {
  var self = this
  self.hideAll()

  self.setRightOrgBox(true)
  $("#rightContainerCampaignInfo").removeClass("col-lg-5").addClass("col-lg-4");

  self.setTopNav(true)
  self.showPage();

  self.setBannerColors(self.payeeInfo.bannerBkColor, self.payeeInfo.bannerTextColor)
  self.setGoalProgressBars()

  window.scrollTo(0, 0)

  const isP2PPledgeOn =
self.peerToPeerData.campaignData.campaignTypeData.pledgePerEvent;

  const p2pLeaderboardLink =
`${window.location.protocol}//${window.location.hostname}/handle/${self.peerToPeerData.pay
eeInfo.handle}/campaign/${self.peerToPeerData.campaignData.intid}/leaderboard`
  $("#p2pBackToAllChampions_share").attr('href',
p2pLeaderboardLink).html(self.peerToPeerData.campaignData.name);

  $("#peerToPeerShareLink").prop("href", self.peerToPeerData.url)
```

```
    $("#peerToPeerShare").show()


    $("#p2pShareCampName").html(self.peerToPeerData.campaignData.name)


    let p2pCampHref =
`${window.location.protocol}//${window.location.hostname}/handle/${self.peerToPeerData.pay
eeInfo.handle}/campaign/${self.peerToPeerData.campaignData.intid}/leaderboard`
    $('#p2pShareCampNameHref').attr('href', p2pCampHref);


    const pledges = self.peerToPeerData.pledges || [];
    pledges.sort((a, b) => Date.parse(b.updated) - Date.parse(a.updated));


    const transactions = self.peerToPeerData.transactions || [];
    const raisedText = utils.getAmountDisplay(self.peerToPeerData.raised, true);
    const goalText = utils.getAmountDisplay(self.peerToPeerData.goal);
    let rank = self.peerToPeerData.rank;


    $("#p2pShareFundRaised").text(raisedText)
    $("#p2pShareFundGoal").text(`of ${goalText}`)


    $("#fundraiserShareDonationNum").text(isP2PPledgeOn ? pledges.length :
transactions.length);
    const donorName = self.peerToPeerData.createdByName ||
self.peerToPeerData.userInfo.name;
    $("#p2pShareDonorName").html(`By ${donorName}`);
    const displayName = utils.transformString(self.peerToPeerData.displayName);
    $("[id^='p2pShareFundraiserName_']").html(displayName);
    const setProgressBarValue = function(raised, goal, id ) {
        var completed = (raised / goal) * 100


        if(!raised){
            completed = 0
        }
```

```
        if (completed > 100) {
            $(id).prop("aria-valuenow", 100)
            $(id).css("width", "100%")
        } else {
            $(id).prop("aria-valuenow", completed)
            $(id).css("width", completed.toFixed(0) + "%")
        }
    }

    getProgressText = function(raised, goal, transactions) {
        const raisedText = utils.getAmountDisplay(raised, true);
        const goalText = utils.getAmountDisplay(goal)
        const raisedOf = goal ? `of <b>${ goalText }</b>` : `raised`;
        return `  <div class="campaign-goal__raised--lg headline-7 mr-1">Raised <b>${ raisedText
}</b></div>
                <div class="campaign-goal__raised--lg headline-7"> ${ raisedOf }</div>`
    }

    setProgressBarValue(self.peerToPeerData.raised, self.peerToPeerData.goal,
"#peerGoalProgressShareFund");

    let isTeamEnabled = self.peerToPeerData.campaignData.campaignTypeData.teamEnabled;
    let teams = self.peerToPeerData.campaignData.campaignTypeData.teams;
    let p2pId = self.peerToPeerData.id;
      let shownTransactions = 5;
    if (p2pTransactions.length > shownTransactions) {
        $("#showMoreP2pTransactionsShare").show();
        $("#showMoreP2pTransactionsShare").off("click")
        $("#showMoreP2pTransactionsShare").click(function () {
            shownTransactions += 5;
```

```
$("#p2pRecentDonationsList").html(getRecentDonationsListWithThankHtml(shownTransaction
s));
        initP2pThankBtnClick();

        if (p2pTransactions.length <= shownTransactions) {
            $("#showMoreP2pTransactionsShare").hide();
        }
    })
}

if(p2pTransactions.length > 0) {
    initP2pThankBtnClick();
}
   const initP2PShareByEmail = function () {
    quickPledge.p2pInviteDonorsList = [];

    const validateP2PShareByEmail = function () {
        const email = utils.getElementValueTrimmed('p2pShareViaEmailInputEmail')
        if (email == "") {
            orgUtil.setInputError("p2pShareViaEmailInputEmail", "Empty Email")
            return false
        }
        if (!utils.validateEmail(email)) {
            orgUtil.setInputError("p2pShareViaEmailInputEmail", "Invalid Email")
            return false
        }

        const isEmailExist = quickPledge.p2pShareEmaisList.find((donor)=>(donor.email ===
email));
        if (isEmailExist) {
            orgUtil.setInputError("p2pShareViaEmailInputEmail", "This email is already added to
list")
```

```
        return false;
    }

    const name = utils.getElementValueTrimmed('p2pShareViaEmailInputName');
    if (name == '') {
        orgUtil.setInputError("p2pShareViaEmailInputName", "Name cannot be empty");
        return false;
    }

    return true;
}

const addInviteeToList = function (name_ = '', email_ = '') {
    const name = name_ || utils.getElementValueTrimmed('p2pShareViaEmailInputName');
    const email = email_ || utils.getElementValueTrimmed('p2pShareViaEmailInputEmail')

    quickPledge.p2pShareEmaisList.push({name,email})
}

$("#p2pShareViaEmailInvite").off("click")
$("#p2pShareViaEmailInvite").click(function () {
    if (quickPledge.p2pShareEmaisList.length === 0) {
        if (!validateP2PShareByEmail()) {
            return;
        }
        addInviteeToList();
    }

    const params = {
        "peerToPeerId": self.peerToPeerData.id,
        "shareTo": quickPledge.p2pShareEmaisList,
```

```
            "message": utils.getElementValueTrimmed("p2pShareViaEmailInputMessage") || '',
        }

        api.makeApiRequest({
            type: "p2pshare",
            data: { "p": JSON.stringify(params) },
            success: function () {
                quickPledge.p2pShareEmaisList = [];
                toastr.success("Email Successfully sent");
                $('#p2pShareViaEmailInputName').val('');
                $('#p2pShareViaEmailInputEmail').val('');
                $('#p2pShareViaEmailInputMessage').val('');
                $('#p2pShareViaEmailList').html('');
                self.showPeerToPeerShare();
            },
            error: function (error) {
                toastr.error("Error in trying to send email try again")
            },
            expired: function () { quickPledge.onInvalidAccessToken() },
            extend: function () { quickPledge.extendCookieLife() }
        })
    })
}
initP2PShareByEmail()

$("#twitterShareButton").off("click")
$("#twitterShareButton").click(function () {
    const twitUrl = "https://twitter.com/intent/tweet?text="
    + escape("I am raising funds for " + self.peerToPeerData.payeeInfo.name + ". Tap on the link below to know more or donate.")
    + "&url="
    + escape(self.peerToPeerData.url)
```

```
    window.open(twitUrl, '_blank');
  })

  $("#fbShareButton").off("click")
  $("#fbShareButton").click(function () {
    FB.ui({
      method: 'share',
      href: self.peerToPeerData.url,
      quote: "I am raising funds for " + self.peerToPeerData.payeeInfo.name + ". Tap on the
link below to know more or donate.",
    }, function (response) {
    });
  })

  $('#copyLinkButtonUrl').text(self.peerToPeerData.url)
  const callback = () => {
    return self.peerToPeerData.url;
  }
  this.initCopyLink('data-copy-link', 'copyLink', callback)
```

Додаток В. Код функції для сторінки донату до фандрейзера

```
QuickPledge.prototype.loadPeerToPeerDonate = function(push) {
  var self = this
  self.hideAll()

  setPageState("peerToPeerDonate")
  if (push) {
    pushState("peerToPeerDonate")
  }
   window.scrollTo(0, 0)
  api.makeApiRequest({
```

```
type: "getPeerToPeerInfo",
data: { "peerid": self.peerId },
success: function (respData) {
    self.peerToPeerData = respData
    self.payeeInfo = respData.payeeInfo
    self.payeeInfo.selectedCampaign = respData.campaignData
    self.selectedCampaign = respData.campaignData
    self.setBannerColors(self.payeeInfo.bannerBkColor, self.payeeInfo.bannerTextColor)
    self.setRightOrgBox(true)
    $("#rightContainerCampaignInfo").removeClass("col-lg-5").addClass("col-lg-4");

    $('#shareButtonsBlock').addClass('d-sm-block')
    $('#shareButtonsDropdown').removeClass('d-none')

    const { campaignData, userInfo } = self.peerToPeerData;

    const { campaignTypeData,
        campaignTypeData: { pledgePerEvent, teamEnabled, useAmounts, usePercentage }
    } = campaignData;

    if(pledgePerEvent){
        $('#contributeSubtitle').show();
        $('#contributeTitle').text('Make a Pledge');
        $('#milestoneName').text(campaignTypeData.eventName)
        $('#pledgeDonateOtherAmount').text("Pledge Other Amount");
        $('#contributionsPledges').text("Recent Pledges");
        $('#donationCountDonorScreen').text("Pledges");
    }
    self.setTopNav(true)
    self.showPage();
```

```
        const p2pLeaderboardLink =
`${window.location.protocol}//${window.location.hostname}/handle/${self.peerToPeerData.pay
eeInfo.handle}/campaign/${campaignData.intid}/leaderboard`

        $("#p2pBackToAllChampions_donate").attr('href',
p2pLeaderboardLink).html(campaignData.name);


        let rank = self.peerToPeerData.rank

        $("#peerToFundraiserRankNum").text(`#${rank}`);

        if(rank == 1){

           $('#p2pRankMedalIcon').show()

        }


        if (rank === 0) {

           $("#rankDonateScreen").hide()

        } else {

           $("#rankDonateScreen").show()

        }


        const pledges = self.peerToPeerData.pledges || [];

        pledges.sort((a, b) => Date.parse(b.updated) - Date.parse(a.updated));


        const transactions = self.peerToPeerData.transactions || [];

        const teams = campaignTypeData.teams || [];

        const enableTeam = teamEnabled;

        const fundraiserTeamId = self.peerToPeerData.teamId || null;

        const fundraiserTeam = teams.find(team => team.id === fundraiserTeamId);

        const displayName = utils.transformString(self.peerToPeerData.displayName);

        const donorName = self.peerToPeerData.createdByName ||
self.peerToPeerData.userInfo.name;

        const p2pFundraiserImage = self.peerToPeerData.profileImage ||
self.peerToPeerData.userInfo.image;


        $("#p2pDonateFundraiserName, #p2pDonateFundraiserNameSm").html(displayName);
```

```
        $("#fundraiserDonationNum").text(pledgePerEvent ? pledges.length :
transactions.length);
        $('#p2pDonateChampionName').html(`By ${donorName}`);
        $("#p2pUserImageDonate").prop("src", p2pFundraiserImage);
        $("#fundraiserTeamName").hide();

        if(self.peerToPeerData.desc !== ''){

$('#p2pDonateMessageForDonors').html(self.getDescription(self.peerToPeerData.desc, {classes:
'quill-description option-description mt-2', webkitLineClamp: [2,2,2,2,2]}));
            $('#p2pDonateMessageForDonorsBlock').addClass("d-flex");
        }

        if(enableTeam && teams.length > 0){
            $("#fundraiserTeamName").show();
        }

        if(fundraiserTeam) {
            $("#fundraiserTeamName").html(fundraiserTeam.name);
        }

        setProgressBarValue = function(raised, goal, id ) {
            var completed = (raised / goal) * 100

            if(!raised){
                completed = 0
            }

            if (completed > 100) {
                $(id).prop("aria-valuenow", 100)
                $(id).css("width", "100%")
            } else {
```

```
            $(id).prop("aria-valuenow", completed)
            $(id).css("width", completed.toFixed(0) + "%")
        }
    }

    setProgressBarValue(self.peerToPeerData.raised, self.peerToPeerData.goal,
"#peerGoalProgressDonateFund");

    $("#p2pFundRaised").html(utils.getAmountDisplay(self.peerToPeerData.raised));
    $("#p2pFundGoal").text(`of ${utils.getAmountDisplay(self.peerToPeerData.goal)}`);

    const remainingAmount = self.peerToPeerData.goal - self.peerToPeerData.raised;

    $("#goalReached").hide();
    if(remainingAmount <= 0) {
        $("#goalReached").show();
    }

    let p2pOptionListHtml = '';
    if (useAmounts) {
        p2pOptionListHtml =
self.getP2POptionListHTML(campaignTypeData.amountOptions, 0);
    } else if(usePercentage) {
        p2pOptionListHtml =
self.getP2POptionListHTML(campaignTypeData.percentOptions, 1, remainingAmount);
    }

    $("#p2pOptionList").html(p2pOptionListHtml);

    $("#p2pOptionList").off("click");
    $("#p2pOptionList").on("click", ".p2p-donate-option", ({ currentTarget }) => {
        self.p2pDonateOptionName = currentTarget.dataset.option;
```

```
            self.amount = utils.convertCurrencyToFloat(currentTarget.dataset.amount);
            self.loadDonationConfirm(true);
        });

        $("#peerCustom").off("click")
        $("#peerCustom").click(function () {
            self.p2pDonateOptionName = "";
            self.amount = 0;
            self.loadDonationConfirm(true)
        })

        const p2pTransactions = pledgePerEvent ? pledges : transactions;

        const getP2pRecentDonationsHtml = function (sliceCount) {
            let html = '';

            if (p2pTransactions.length > 0) {
                html = self.getRecentDonationsListHTML(p2pTransactions.slice(0, sliceCount),
userInfo.id, pledgePerEvent);
            } else {
                html = `<span class="body-2" style="color: #667085;">There are no
${pledgePerEvent ? 'pledges' : 'donations'} for this fundraiser yet. Be the first!</span>`
            }

            return html;
        }

        $("#p2pRecentDonationList").html(getP2pRecentDonationsHtml(5));

        let shownTransactions = 5;
        if (p2pTransactions.length > shownTransactions) {
            $("#showMoreP2pTransactions").show();
```

```
            $("#showMoreP2pTransactions").off("click")
            $("#showMoreP2pTransactions").click(function () {
                shownTransactions += 5;

$("#p2pRecentDonationList").html(getP2pRecentDonationsHtml(shownTransactions));

                if (p2pTransactions.length <= shownTransactions) {
                    $("#showMoreP2pTransactions").hide();
                }
            })
        }

        $("#peerToPeerDonateOptions").show()
    },
    error: function (error) {
        switch (error.errorCode) {
            case "CampaignStatusInactive":
                self.showError("This campaign is no longer active")
                return
            case "InvalidPayee" :
                self.showError("Invalid Organization")
                return
            default:
                self.showError()
                return
        }
    },
    expired: function () { quickPledge.onInvalidAccessToken() },
    extend: function () { quickPledge.extendCookieLife() }
})
```

Додаток Г. Код сторінки списку усіх Peer To Peer кампаній на стороні адміна

```
<div class="container pl-5 pr-4">
  <header>
    <div class="d-flex align-items-center mt-3">
      <h1 class="headline-4 pis">Peer To Peer</h1>
      <div class="card__right-side d-flex justify-content-end search ml-auto">
        <div class="search__icon-wrapper">
          <i class="fas fa-search  input-prefix"></i>
        </div>
        <input [formControl]="search" class="form-control--search body-2"
placeholder="Search Peer To Peer..." />
        <div class="search__icon-wrapper--cross px-1">
          @if(search.value.length > 0) {
            <span class="cursor--pointer" (click)="resetSearch()">X</span>
          }
        </div>
      </div>
    </div>
    <span class="grey-text subtitle-2">You have a total of
      <b class="black-text">
        {{P2PCampaigns.length}} campaigns
      </b>
    </span>
  </header>
  <div class="py-5">
    @if(filteredP2PCampaigns.length > 0) {
      <app-peer-to-peer-card *ngFor="let p2pCampaign of filteredP2PCampaigns; trackBy:
trackById" [peerToPeer]="p2pCampaign"></app-peer-to-peer-card>
    } @else {
      <div class="d-flex justify-content-center">No Matching Campaigns</div>
```

```
      }
    </div>
</div>
```

Додаток Д. Код логічної частини сторінки списку усіх Peer To Peer кампаній на стороні адміна

```
import * as _ from 'lodash';
import { Component, DestroyRef, inject, OnInit } from '@angular/core';
import { select, Store } from '@ngrx/store';
import { filter } from 'rxjs/operators';
import { SharedService } from 'app/virtual-event/services/shared.service';
import { PeerToPeerManageService } from '../../services/peer-to-peer-manage/peer-to-peer-manage.service';
import { FormsModule, ReactiveFormsModule, UntypedFormControl } from '@angular/forms';
import { loadP2PCampaignList } from '../../features/p2p.actions';
import { selectP2PCampaignsList } from '../../features/p2p.selectors';
import { PeerToPeerCardComponent } from '../peer-to-peer-card/peer-to-peer-card.component';
import { CommonModule } from '@angular/common';
import { takeUntilDestroyed } from '@angular/core/rxjs-interop';

@Component({
  standalone:true,
  selector: 'app-peer-to-peer-list',
  templateUrl: './peer-to-peer-list.component.html',
  styleUrls: ['./peer-to-peer-list.component.scss'],
  imports:[PeerToPeerCardComponent,FormsModule,ReactiveFormsModule,CommonModule]

})
export class PeerToPeerListComponent implements OnInit {
  destroyRef = inject(DestroyRef)
  public newP2PCampaigns$: any = this.store.pipe(
    select(selectP2PCampaignsList)
```

```
);

public P2PCampaigns = []
public filteredP2PCampaigns = []
public search: UntypedFormControl = new UntypedFormControl('', []);

constructor(
  public store: Store,
  public sharedService: SharedService,
  public peerToPeerManageService:PeerToPeerManageService
) { }

ngOnInit() {
  this.store.dispatch({
    type: loadP2PCampaignList.type,
    payload: this.sharedService.token
  });

  this.newP2PCampaigns$.pipe(
    filter((list:any)=> list.length > 0),
    takeUntilDestroyed(this.destroyRef)
  ).subscribe((P2PCampaigns)=>{
    this.P2PCampaigns = [...P2PCampaigns];
    this.filteredP2PCampaigns = [...P2PCampaigns];
  });

  this.search.valueChanges.pipe(
    takeUntilDestroyed(this.destroyRef)
  ).subscribe(searchQuery => {
    this.filteredP2PCampaigns = _.filter(this.P2PCampaigns, (p2p: any) => {
      return _.includes(p2p.name.toLowerCase(), searchQuery.toLowerCase())
```

```
      });
    });
  }

  trackById(index: number, item: any): number {
    return item.id;
  }

  resetSearch() {
    this.search.setValue('');
  }
}
```