Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для організації збору коштів. Клієнтська частина для донорів та адміністраторів
(тема)

Виконав:
здобувач 4 року навчання
групи ПЗПІ-21-1

Олексій РИЧКО
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Віталій ЛЯПОТА
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ричко Олексію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система для організації збору коштів. Клієнтська частина для донорів та адміністраторів

Затверджена наказом по університету від 19.05.2025 р. № 397 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19.06.2025

3. Вихідні дані до роботи розробити веб-орієнтовану програмну систему для організації та супроводу збору коштів благодійними ініціативами; серверна частина: ASP.NET Core з використанням Clean Architecture та EF Core; інтеграція платіжних сервісів (Stripe, LiqPay) через REST-API; клієнтська частина: React із застосуванням Redux Toolkit та Material UI; для моніторингу: Prometheus і Grafana; тестування: xUnit/NUnit.

4. Перелік питань, що потрібно опрацювати в роботі: Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	07.04.2025	*виконано*
2	Створення специфікації ПЗ	08.04.2025	*виконано*
3	Проектування ПЗ	13.04.2025	*виконано*
4	Розробка ПЗ	19.04.2025	*виконано*
5	Тестування ПЗ	22.05.2025	*виконано*
6	Оформлення пояснювальної записки	01.06.2025	*виконано*
7	Підготовка презентації та доповіді	10.06.2025	*виконано*
8	Попередній захист	17.06.2025	*виконано*
9	Нормоконтроль, рецензування	17.06.2025	*виконано*
10	Здача роботи у електронний архів	18.06.2025	*виконано*
11	Допуск до захисту у зав. кафедри	18.06.2025	*виконано*

Дата видачі завдання «07» «квітня» 2025 р.

Здобувач _____ Олексій РИЧКО
(підпис)

Керівник роботи _____ ст.викл. Віталій ЛЯПОТА
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 61 стор., 12 рис., 1 табл., 12 джерел, 3 додатки.

ДОНАТ, КРАУДФАНДИНГ, СПОВІЩЕННЯ, ASP.NET CORE, CLEAN ARCHITECTURE, REACT.

Об’єкт розробки – веб-орієнтована програмна система для організації та супроводу збору коштів благодійними ініціативами.

Мета розробки – створити модульну програмну платформу, що забезпечує прозоре управління ініціативами, безпечну обробку платежів та гнучку систему сповіщень для донорів і адміністраторів.

Методи та засоби реалізації – серверна частина побудована на ASP.NET Core з використанням Clean Architecture та EF Core; інтеграція платіжних сервісів (Stripe, LiqPay) виконується через REST-API; клієнтська частина реалізована на React із застосуванням Redux Toolkit та Material UI; для моніторингу використано Prometheus і Grafana, тестування забезпечено xUnit/NUnit.

Результати – розроблено та впроваджено:

– модуль користувачів із двофакторною авторизацією (email + Google OAuth) і ролями admin/user;

– модуль донорів із історією транзакцій, пошуком і «Топ-донори»;

– систему підписок та email-/push-сповіщень про нові збори й зміни їх статусу;

– API-ендпоїнти з авторизацією по згенерованому API-ключу для інтеграції з зовнішніми платформами;

– мобільний інтерфейс з підтримкою GPS-пошуку локальних ініціатив.

Запропоноване рішення підвищує прозорість фінансових потоків, автоматизує взаємодію між організаторами й донорами та забезпечує масштабованість сервісу для міжнародного використання.

ABSTRACT

ASP.NET CORE, CLEAN ARCHITECTURE, REACT, DONATE, CROWDFUNDING, NOTIFICATIONS.

Object of development – a web-oriented software system for organising and supporting charitable fundraising initiatives.

Purpose – to create a modular platform that provides transparent initiative management, secure payment processing and a flexible notification system for donors and administrators.

Methods and tools – backend built with ASP.NET Core following Clean Architecture and EF Core; payment services (Stripe, LiqPay) integrated via REST-API; frontend implemented in React with Redux Toolkit and Material UI; monitoring via Prometheus and Grafana; testing with xUnit/NUnit.

Results – the system delivers:

– a user module with two-factor authentication (email + Google OAuth) and admin/user roles;

– a donor module with transaction history, search and «Top Donors» leaderboard;

– subscription-based email/push notifications about new fundraisers and status changes;

– secured API endpoints authorised by generated API keys for external integrations;

– a mobile interface with GPS-enabled search for nearby initiatives.

The proposed solution enhances financial transparency, automates interaction between organisers and donors, and guarantees scalability for international deployment.

ЗМІСТ

ВСТУП

Сучасні соціально-економічні потрясіння, зокрема повномасштабна війна в Україні, актуалізували потребу у швидкому, прозорому та масштабованому механізмі залучення благодійних коштів. Традиційні форми фінансування – банківські кредити, грантові програми чи спонсорські внески – виявляються надто повільними та бюрократичними, коли стоїть питання невідкладного забезпечення військових або підтримки гуманітарних ініціатив. Водночас поширення електронних платежів і мобільного інтернету створює технічні передумови для масових онлайн-пожертв, здатних акумулювати мільйони гривень упродовж годин.

Об’єктом дослідження є веб-орієнтована програмна система, покликана автоматизувати повний цикл краудфандингу: від реєстрації користувача й ініціювання збору до звітності про використання коштів. Предметом є архітектурні та технологічні рішення, що забезпечують безпеку транзакцій, високу доступність сервісу й зручність взаємодії із донором.

Метою роботи є розроблення та експериментальна апробація програмної платформи, яка поєднує модульний бекенд на ASP.NET Core, клієнтський застосунок на React і мультивалютні платіжні шлюзи, підкріплені механізмом підписок і сповіщень. Для досягнення мети розв’язуються такі завдання: обґрунтування вибору архітектурної моделі Clean Architecture; проєктування файлової структури й контрактів API; реалізація модулів користувачів, донатів та сповіщень; інтеграція платіжних сервісів і системи моніторингу; проведення тестування продуктивності й оцінка надійності.

Наукова новизна полягає у комплексному підході, що поєднує адаптивне масштабування мікросервісів із рекурентними платежами та прозорою звітністю у реальному часі. Практичне значення роботи підтверджується можливістю оперативного розгортання платформи для державних і волонтерських ініціатив, а також інтеграцією з корпоративними CSR-програмами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Упродовж останнього десятиліття благодійність переходить із площини поодиноких доброчинних акцій до повсякденної цифрової практики, стаючи для суспільства одним із головних механізмів швидкого реагування на гуманітарні та соціальні виклики. За оцінками галузевих досліджень, світовий ринок краудфандинґу у 2024 році вже сягнув приблизно 2,14 млрд USD і продовжує зростати двозначними темпами [1].

Збільшення охоплення мобільного інтернету, поширення безконтактних платежів і розвиток соціальних мереж зробили мікродонати звичним інструментом участі громадян у вирішенні проблем-від стихійних лих до локальних культурних ініціатив – і тим самим сформували суспільний запит на сервіси, що дають змогу переказати кошти «в один дотик», а натомість отримати миттєвий, публічний звіт про використання кожної гривні.

Потреби організацій при цьому еволюціонували не менше, ніж очікування донорів. Неприбуткові фонди шукають платформи, здатні демонструвати фінансову прозорість у режимі реального часу, адже саме відкриті дашборди стали беззаперечним стандартом після кейса UNITED24, що за неповні два роки залучила понад 1,44 млрд USD і щоразу підтверджує довіру детальними щотижневими звітами.

Волонтерські та громадські ініціативи передусім прагнуть швидкого старту кампаній без затрат на розробку власної інфраструктури, тоді як державні чи муніципальні програми – наприклад, відбудова інфраструктури – потребують механізмів ідентифікації користувачів і суворого дотримання вимог KYC/AML. Business-корпорації дедалі частіше інтегрують пожертви безпосередньо у свої платіжні потоки, переводячи окремі транзакції клієнтів у «дрібні» благодійні внески, що вимагає API-орієнтованих рішень із гарантованою високою пропускною здатністю.

На тлі економічної турбулентності та інфляційного тиску, навіть домогосподарства почали сприймати краудфандинґ як засіб взаємної страховки: у

Франції чверть населення вже хоч раз зверталася до «солідарних» платформ, щоб закрити базові потреби – від оплати оренди до купівлі продуктів.

Водночас понад 60 % благодійників тепер надають перевагу суто онлайн-платежам, а рекурентні підписки перетворилися на ключове джерело стабільного фінансування для організацій. Отже, сучасна система збору коштів повинна поєднувати прозору бухгалтерію, мультивалютні платіжні шлюзи, гнучкий механізм підписок і розширюваний API – саме такий функціональний каркас лягає в основу нашого проєкту.

1.2 Аналіз подібних систем

Системи цифрового фандрейзингу бувають різних типів. Існують як класичні краудфандингові платформи, так і мобільні/банківські додатки з благодійними функціями. Однак наразі саме традиційні канали – колективні скриньки та особисті волонтерські збори – залишаються найпопулярнішими: близько 29% українців роблять внески у стаціонарні скриньки в магазинах і супермаркетах, ще 11% – жертвують через волонтерів на публічних заходах. Натомість лише незначна частина громадян користувалася краудфандингом. З технічної точки зору найпоширенішими є мобільні застосунки банків: наприклад, у Monobank – функція «банки», а в Приват24 – сервіс «конверти», де користувач може створити збір із описом, ціллю і оновленням прогресу. Також дедалі частіше застосовуються QR-донати і смс-пожертви, а платіжні шлюзи використовуються благодійними фондами для прийому онлайн-платежів

Попри технічний прогрес, цифрова благодійність має низку суттєвих викликів. Головна з них – низька довіра: багато донорів остерігаються, що їхні кошти підуть не за призначенням, а небагато хто перевіряє прозорість фондів. Як наголошується у дослідженні з UX-дизайну, відсутність чіткої інформації про те, куди потрапляють гроші, й нечітка подача мети проекту миттєво знижують довіру користувача.

Аналогічно, будь-яка «фрикція» у процесі пожертви – зайві кроки, заплутана навігація або неадаптованість під мобільні – часто призводить до відмови від

донату. Ще один психологічний бар'єр – перевантаження вибором. Класичні спостереження показують, що коли донору пропонують забагато параметрів (різні суми чи категорії пожертв, багато опцій у формі), він просто «заморожується» й відмовляється діяти. Тому зайва складність форм може нашкодити конверсії.

Інтерфейс додатку Donate24 (див. рис. 1.1) показовий: він щоденно пропонує користувачам актуальні збори на потреби ЗСУ, дозволяючи відфільтрувати кампанії за темою, легко перейти до донату чи скопіювати реквізити з одним.

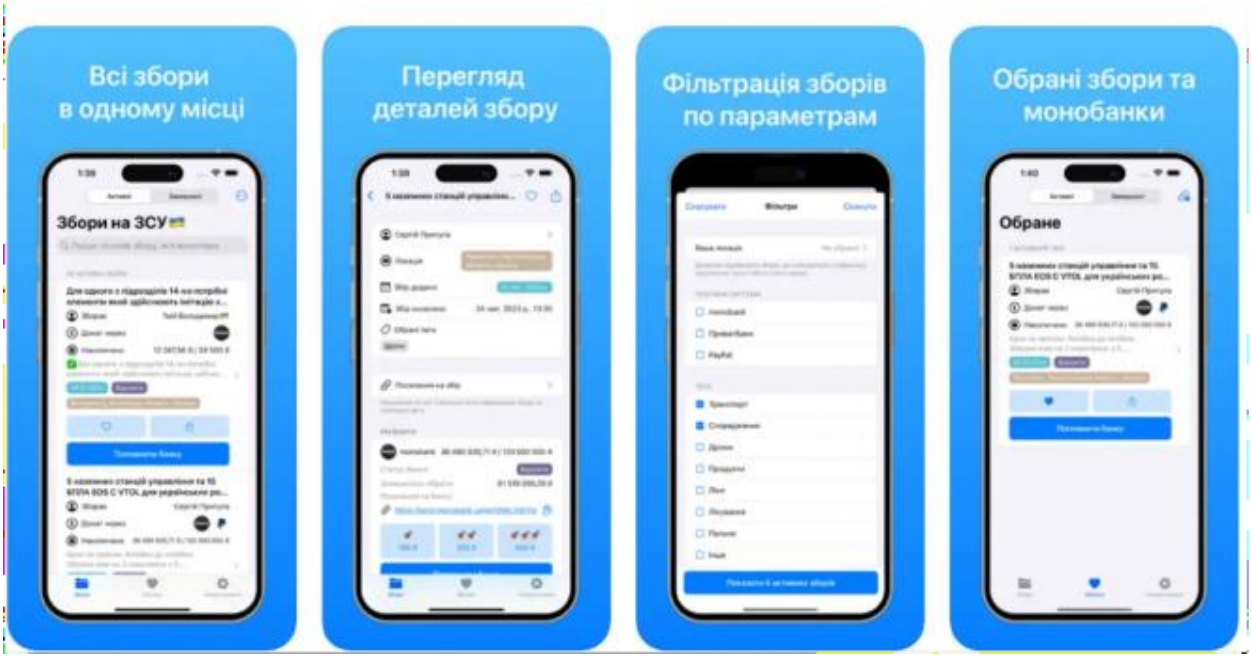

Рисунок 1.1 – Donate24 (за даними [2])

Такі рішення посилюють ефект миттєвої дії – якщо користувач бачить кнопку «Пожертвувати зараз» на початку заклику, це значно підвищує відгук. Також мобільні застосунки використовують push-повідомлення для рекрутингу донорів і мотивації «дати зараз»: вони спонукатимуть зробити внесок своєчасно та нагадують «давати часто». Не менш важливий аспект – соціальний резонанс: коли люди діляться у мережах інформацією про власний донат, це може генерувати нові залучення (соціальне підтвердження) кліком.

Найкращі практики у комунікації та технічному вирішенні формули руху донорів зосереджені на простоті та прозорості. Наприклад, показ реального прогресу кампанії у вигляді progress bar та конкретних числових статистик значно підвищують довіру користувача. Важливо також мінімізувати кількість полів у

формі донату та забезпечити «UX у 2 кліки»: досвід показує, що перенесення головного заклику до дії на початок (наприклад, кнопка «Donate» у перших рядках листа) різко збільшує конверсію. Після завершення транзакції корисно одразу відображати екран-підтвердження з візуальною анімацією або коротким повідомленням-подякою, щоб донор відчув «ефект завершеності» (feedback) і розумів, що його пожертва зарахована. Автоматизовані листи чи підтвердження з докладними даними про витрати також допомагають утримати довіру і мотивують повторні внески.

Прикладами вдалих рішень є як українські розробки, так і світові сервіси. Зазначимо, що функція «банок» у мобільному додатку Monobank зібрала понад 1 млрд євро від початку війни, а минулого року через неї пройшло 43,68 млрд грн донатів (Monobank оприлюднив ці дані). Подібний інструмент – «Конверти» у Приват24 – дозволяє групувати збори та прозоро показувати збірникам опис і прогрес. Ще один приклад – сайт VolunteeringUkraine, який публічно відображає всі пожертви на фронтовий проєкт Front Line Kit, що надходять через платіжний шлюз WayForPay; так донори бачать повну прозорість усіх транзакцій. Міжнародні платформи як Patreon чи Donorbox стають стандартними для творчих і навчальних проєктів, а українські рішення активно запозичують ці підходи. Разом вони ілюструють прагнення зробити цифрову благодійність зручною та прозорою, що важливо для подальшого зростання фондів і залучення довіри суспільства [3].

Інтернет-збори перетворилися на самодостатній фінансовий інструмент, який поєднує 4 цифрових платежів із соціальним ефектом взаємодії спільнот. На практиці це означає, що кошти надходять упродовж годин, а не тижнів, адже між заявкою організатора й першими переказами немає багаторівневого банківського погодження чи грантового конкурсу. Дослідження WhyDonate показує, що онлайн-платформі достатньо кількох відсотків комісії, тоді як класичний кредит або благодійний фонд із розгалуженим апаратом витрачає у півтора-два рази більше на транзакційні та адміністративні витрати; додатково відсутні жорсткі застави й довгі процедури підтвердження платоспроможності, характерні для банківського фінансування.

Швидкість і низька вартість доповнюються прозорістю, яку практично неможливо відтворити у грантових або спонсорських схемах: сучасні платформи публікують розподіл зібраних коштів у форматі щотижневих звітів з деталізацією до останнього центу, як це робить державна ініціатива UNITED24, що вже акумулювала понад 1,4 млрд USD і демонструє відкриті дашборди у п’яти напрямах витрат. Така візуальна звітність формує довіру й утримує донорів: регулярні оновлення стимулюють їх повертатися, тоді як офлайн-фонди найчастіше звітують із місячним чи квартальним лагом.

Є й ефект масштабу. Онлайн-збір, підтриманий соціальними мережами, охоплює міжнародну аудиторію за лічені доби; згідно з доповіддю ESMA, уже 17 % європейських пожертв проходять через кордон, що суттєво збільшує потенційну суму порівняно з локальними ярмарками або публічними заходами. До того ж мікродонати у кілька доларів, які раніше не мали економічного сенсу через банківські тарифи, тепер становлять третину всього онлайн-доходу: звіт M+R Benchmarks зафіксував, що рекурентні платежі формують 31 % цифрового обороту, забезпечуючи фондам стабільний кеш-флоу без окремих раундів фандрейзингу.

Порівняно з грантами чи спонсорством, масовий збір створює додаткову нефінансову цінність: кожен донор стає «амбасадором» ініціативи, поширюючи її історію у своїй мережі контактів. Соціальний капітал такої взаємодії підсилює довгострокову лояльність і часто перетворює одноразових підтримувачів на постійних клієнтів або волонтерів, що відзначають економічні огляди українського ринку альтернативного фінансування.

Зрештою, цифрові платформи істотно знижують бар’єр входу: щоб ініціювати кампанію, не потрібно формувати юридичну особу чи відкривати окремий рахунок; достатньо пройти базову верифікацію KYC/AML, після чого інтегровані шлюзи Stripe, LiqPay чи PayPal автоматично приймають платіжні картки і навіть криптовалюту. Для малих волонтерських груп це часто єдиний реалістичний спосіб отримати фінансування, тоді як банк просто відхилив би заявку через відсутність фінансової історії. Саме сукупність швидкості, прозорості, глобального охоплення та мінімальних бар’єрів робить онлайн-збори

практичнішими й ефективнішими за традиційні методи залучення коштів, особливо коли йдеться про термінові гуманітарні, медичні чи інноваційні потреби.

Перевага цифрових зборів полягає в тому, що вони поєднують фінансову ліквідність із суспільним ефектом, який класичним інструментам – кредитам, грантам чи прямому спонсорству – відтворити майже неможливо. Гроші надходять протягом лічених годин, бо між ініціатором і першим донором немає ані багаторівневих банківських перевірок, ані конкурсних процедур грантових програм. Це створює ефект «миттєвої подушки безпеки»: організатор одразу фіксує стартову суму, а отже може оперативно закупити обладнання чи оплатити послугу, не очікуючи формального погодження бюджету. У масштабі ринку така швидкість трансформується у стрімке щорічне зростання: за даними Business Research Company, обіг глобального краудфандинґу підскочив із 17,7 млрд USD у 2024 році до прогнозних 20,5 млрд USD у 2025-му, і це лише сегмент, що підзвітний статистичним органам [4].

Порівняно з банківським кредитуванням, де вартість позики визначається відсотковою ставкою та комісіями, онлайн-збір майже не містить прихованих платежів: платформи обмежуються транзакційною комісією, тоді як адміністративні витрати мінімізовані завдяки автоматизації платежів і прозорим API.

У підсумку кожен переказ зберігає реальну купівельну спроможність, що особливо критично для гуманітарних потреб, де різниця у днях або навіть годинах може коштувати людського життя. Водночас сама прозорість платіжного ланцюга – від моменту переказу до публічного звіту – перетворює донорів із пасивних жертводавців на активних учасників проєкту, які слідкують за його розвитком і поширюють інформацію у власних соціальних мережах. Таким чином формується «ефект мультиплікатора», що традиційним грантодавчим механізмам недоступний.

Особливу цінність цифрові платформи демонструють у переході від одноразових пожертв до рекурентних моделей. Автори звіту Dataro за 2024 рік засвідчують, що утримання щомісячних донорів сягає 83 %, тоді як разові дарувальники повертаються лише у 45 % випадків.

Висока лояльність створює стабільний кеш-флоу, даючи організаціям змогу планувати витрати на місяці вперед і менш залежати від зовнішніх шоків. Показово, що глобальна статистика Double the Donation фіксує зниження доходів від разових онлайн-внесків на 12 % протягом року, натомість виручка від місячних підписок зросла на 11 % і вже формує понад чверть усіх цифрових пожертв. Жоден банківський продукт не надає настільки гнучкого й водночас прогнозованого фінансового потоку без вимоги застави або складних договорів.

Ще один аспект, який вирізняє краудфандинґ серед інших джерел фінансування, – глобальне охоплення з локальною чутливістю. За лічені дні кампанія стає міжнародною, залучаючи пожертви з-понад десятків країн, але завдяки GPS-функціям мобільні користувачі бачать саме проєкти у своєму регіоні, відчуваючи прямий зв’язок із результатом. У межах однієї платформи поєднані мультивалютні шлюзи Stripe, LiqPay і PayPal, а тому надходження конвертуються автоматично, зберігаючи прозорість для бухгалтерії. По суті, цифровий збір об’єднує найвагоміші сильні сторони різних фінансових механізмів: швидкість транзакції, мінімальні накладні витрати, відкритість звітності й соціальний капітал спільної участі. Саме ця синергія робить онлайн-донати практичнішим інструментом порівняно як із традиційним банківським фінансуванням, так і з грантовими чи спонсорськими програмами, де часові й бюрократичні витрати часто зводять нанівець саму ідею швидкої допомоги.

Повномасштабна війна перетворила потребу у матеріальних ресурсах на безперервний «пульс», що вимірюється не кварталами бюджетного планування, а буквально годинами, упродовж яких на фронті змінюється ситуація і з’являються нові запити на дрони, оптику чи медичні пакети. За цих умов автоматизована система зборів стає критичною інфраструктурою: вона усуває часовий розрив між виникненням потреби й надходженням коштів, бо оперує без вихідних, обробляючи платежі картками, криптовалютою та банківськими переказами одночасно. Платформа UNITED24 – символ цієї швидкості: лише за перші чотири місяці 2025 року вона акумулювала 398 млн USD, довівши сукупний обсяг пожертв до 1,447 млрд USD і показавши, що автоматизація може масштабувати потік

допомоги до мільйонів доларів на тиждень.

Від швидкості невіддільна довіра, а її забезпечує саме програмна автоматизація звітності: транзакції консолідуються в єдиній базі даних, а щотижневі дашборди публікуються у відкритому доступі, дозволяючи кожному донору перевірити, куди спрямовано його внесок. У ситуації, коли ворожа дезінформація намагається підірвати віру громадян і союзників у спроможність держави захищатися, така прозорість набуває стратегічного значення, зміцнюючи легітимність оборонних закупівель і водночас блокуючи можливості для корупційних схем.

Автоматизована система також долає географічні й валютні бар'єри. Діаспора та іноземні симпатики перераховують гроші з десятків країн, не стикаючись із комісіями традиційних SWIFT-платежів; водночас спеціальний рахунок НБУ та інтеграція з платіжними шлюзами Stripe чи LiqPay дають змогу миттєво конвертувати долари, євро чи криптоактиви у гривню й одразу переказувати їх у виробничі контракти або на закупівлю спорядження

Національний банк України. Така безшовність потоків грошей і даних особливо важлива, коли обстріли переривають логістику: цифровий платіж проходить навіть тоді, коли фізичні банківські відділення не працюють, а Starlink забезпечує резервний канал для сервера платформи.

Нарешті, автоматизована система виконує роль аналітичного хребта оборонної економіки. Вона акумулює статистику про типи запитів і темпи зборів, дозволяючи прогнозувати дефіцит обладнання й оптимізувати державні закупівлі; одночасно вона фіксує спад приватних переказів і своєчасно підказує командам маркетологів, коли потрібні нові інформаційні кампанії або тематичні флешмоби. Таким чином програмна автоматизація переходить із площини «фандрейзингу» до площини національної безпеки, стаючи такою ж важливою для стійкості України, як енергетична чи телекомунікаційна мережа.

1.3 Аналіз проблем, з якими стикаються системи збору коштів

Попри стрімке зростання онлайн-благодійності, системи збору коштів

зіткнулися з фундаментальною кризою довіри. 2024-го у світових медіа фіксувалися десятки історій про фіктивні кампанії, що паразитували на природних катастрофах або війні й, використовуючи штучні акаунти та генеративні зображення, відводили пожертви від справжніх потреб. Розкриття таких шахрайств у соціальних мережах поширюється швидше, ніж офіційне спростування, тож навіть доброчесні платформи потрапляють під хвилю скепсису. Щоб відновити репутацію, вони змушені ускладнювати процедури верифікації, і це підвищує вартість кожної транзакції та затягує запуск термінових зборів.

Фінансова площина проблем додає ще одну точку напруження. Хоча платіжні сервіси декларують «пільгові» тарифи для неприбуткових організацій, середня сума комісій і прихованих витрат наближається до трьох відсотків, а в деяких країнах сягає п’яти.

Одночасно ЄС та Велика Британія готують нові пакети регулювання AML/KYC, які зобов’яжуть операторів перевіряти навіть дрібних донорів, що переводять менше 100 євро, і зберігати ці дані не менш як п’ять років. Для малих волонтерських ініціатив це означає додаткові витрати на юристів і технічну інтеграцію, а для користувачів – довші форми та втрату «одного кліку» при пожертвуванні.

Паралельно зростає явище донорської втоми. Після двох років повномасштабної війни Україна вже відчуває різке просідання приватних переказів: низка благодійних фондів повідомляє про падіння надходжень на 25-30 %, а ООН змушена скорочувати гуманітарні програми через брак фінансування. Циклічні інформаційні кампанії для підтримки мотивації поглинають дедалі більше ресурсів самих організацій і збільшують конкуренцію між проєктами за увагу донорів.

Кібербезпека залишається ще одним больовим пунктом: майже третина благодійних організацій зазнала бодай однієї кібератаки протягом року, причому відсутність базових тренінгів і політик робить системи зборів легкою мішенню для фішингу та DDoS-ударів. Під час воєнних дій це особливо критично: атака на платіжний шлюз або витік персональних даних не лише зупиняє фінансовий потік,

а й підриває моральний ефект громадської підтримки.

Загалом сучасна платформа змушена балансувати між трьома протилежними вимогами – миттєвою зручністю для користувача, жорсткими регуляторними бар'єрами та постійною загрозою кібератак. Саме в такому контексті набір проблем стає системним: дорожчий технічний стек і складніший комплаєнс відштовхують невеликі ініціативи; зниження рівня довіри і втома аудиторії загрожують стабільності довгих програм; а недостатня кіберстійкість створює ризик повної зупинки збору в найкритичніший момент. Розв'язання цих суперечностей і визначає рамки технічних і організаційних рішень, що ляжуть в основу подальшого проєктування системи.

1.4 Постановка задачі

З урахуванням окреслених викликів – нестачі довіри, втоми аудиторії та зростання регуляторних обмежень – фронт-енд-модуль, відповідальний за взаємодію користувача з платформою, має виконати подвійне завдання. По-перше, він мусить забезпечити безшовний, емоційно комфортний вхід до екосистеми збору коштів, де новий донор чи ініціатор за кілька секунд проходить реєстрацію й одразу бачить візуально переконливу історію переказів і прозорі механізми контролю. По-друге, інтерфейс повинен стимулювати тривалу залученість: підтримувати рекурентні пожертви, пропонувати персоналізовані сповіщення та робити процес взаємодії настільки простим, щоб навіть суворі вимоги KYC/AML не виглядали для користувача додатковим бар'єром.

Конкретно це означає реалізацію потоку автентифікації на базі електронної пошти й Google OAuth із можливістю одноразової перевірки для тих, хто планує відкривати власні збори; побудову особистого кабінету, де історія транзакцій і статус підписок оновлюються в реальному часі; інтеграцію з бекенд-ендпоїнтами, що публікують дані про топ-донорів та агреговану статистику, аби мотивація підтримувати ініціативи підкріплювалася соціальним визнанням. Окрема підзадача–конфігуратор сповіщень, який дозволяє гнучко керувати темами листів і push-повідомлень, не перевантажуючи донорів зайвою інформацією, але

гарантуючи, що критичні зміни (досягнення цілі, відкриття нового збору у вибраній категорії) не залишаться непоміченими.

Важливо, аби кожен елемент інтерфейсу працював на відновлення й укріплення довіри: від зрозумілого пояснення, куди йде кожна гривня, до можливості залишити публічний відгук під ініціативою. Саме тому модуль відгуків і коментарів впроваджується не як «соціальна фішка», а як центральний інструмент зворотного зв’язку між організаторами й донорами. Разом із динамічною темною/світлою темою та адаптивним дизайном під мобільні пристрої це формує єдине середовище, де зібрання коштів відчувається не транзакцією, а спільною історією, що розгортається на очах користувача.

Отже, завдання модуля можна підсумувати як створення цілісного клієнтського шар-інтерфейсу, який: гарантує безпеку й швидкість входу; надає прозорий огляд особистих та агрегованих фінансових даних; підтримує керовані підписки й своєчасні сповіщення; а головне – перетворює разовий акт пожертви на довгостроковий досвід участі в житті платформи. Саме виконання цих функціональних вимог стане запорукою того, що технічна складова проєкту адекватно відповість на соціальні виклики, характерні для воєнного часу й післявоєнної відбудови.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги

Програмна система має забезпечувати повний цикл роботи краудфандингової платформи – від моменту реєстрації користувача до аналітичного звіту про використання зібраних коштів. У її межах передбачено низку взаємопов'язаних сценаріїв, кожен з яких слугує опорою для прозорості, оперативності та масштабованості сервісу.

Передусім платформа повинна підтримувати багатоцільову автентифікацію: класичний логін за електронною поштою, OAuth-авторизацію через популярні провайдери та можливість двофакторного підтвердження. Після входу система формує персональний простір, де користувач бачить актуальний стан своїх зборів, пожертв і підписок, а також може ініціювати відкриття нової кампанії.

Ключовою функцією виступає модуль управління ініціативами. Він забезпечує створення, редагування, пошук і фільтрацію проєктів за тематичними категоріями та географічною прив'язкою. Для кожної ініціативи система повинна підтримувати багатовалютні збори, автоматично конвертуючи кошти за поточним курсом і фіксуючи первинну валюту транзакції.

Процес донату реалізується через інтегровані платіжні шлюзи – наприклад, Stripe, LiqPay або PayPal – із можливістю рекурентних переказів та мінімальним часом підтвердження транзакції. Одразу після успішного платежу дані заносяться до реєстру пожертв, що формує не лише історію руху коштів, а й агреговану статистику для відкритих дашбордів [5].

Система сповіщень має автоматично інформувати зацікавлених користувачів про критичні події: запуск чи завершення збору, досягнення контрольних цілей, публікацію фінансових звітів. Залежно від налаштувань, повідомлення доставляються електронною поштою, push-механізмом браузера або мобільними push-нотифікаціями.

Адміністративна панель повинна надавати інструменти модерації контенту, керування ролями, а також ручного чи автоматизованого блокування підозрілих кампаній відповідно до політик KYC/AML. У тій же площині необхідне ведення

аудит-логів, щоб фіксувати всі критично важливі дії персоналу.

Важливим аспектом є відкритість API. Програмна система мусить публікувати захищені ендпоїнти для інтеграції з мобільними застосунками, системами зовнішньої аналітики та корпоративними сервісами соціальної відповідальності. Авторизація третіх сторін відбувається через згенеровані API-ключі або протокол OAuth 2.0 Client Credentials.

Зрештою, усі функціональні модулі мають працювати у єдиному інформаційному просторі, де дані синхронізуються у режимі майже реального часу, а користувач отримує цілісний досвід незалежно від того, заходить він із веб-браузера чи з мобільного пристрою. Підтримка горизонтального масштабування та багатомовності інтерфейсу завершує перелік базових можливостей, без яких сучасна система збору коштів не зможе ефективно реагувати на динамічні соціальні виклики.

2.2 Нефункціональні залежності

Поряд із функціональними вимогами, програмна система збору коштів має відповідати низці нефункціональних вимог, які визначають якість, безпеку та надійність розробленого програмного рішення:

– продуктивність і масштабованість. система повинна витримувати сплескові навантаження, характерні для масових інформаційних кампаній: середній час відповіді api не має перевищувати 200 мс при 10 000 rps, а горизонтальне масштабування – відбуватися без зупинки сервісу. усі критичні сервіси розгортаються у кластері kubernetes, що гарантує еластичне збільшення реплік під дією автоскейлера;

– доступність. Платформа працює у режимі 24 × 7 з цільовим показником sla ≥ 99,9 %. для запобігання простоям використано георозподілені дата-центри й балансування між ними; база даних реплікується асинхронно, а кеш шар підтримує гаряче резервування;

– безпека. Усі з’єднання шифруються tls ≥ 1.3; дані платіжних карток не зберігаються локально, а передаються напряму до платіжного провайдера через pci

dss-сертифіковані sdk. автентифікація реалізована через oauth 2.1 зі схемою pkce; критичні дії (створення збору, зміна реквізитів ініціативи) підтверджуються двофакторно. кожна транзакція проходить серверний контент-фільтр для виявлення шахрайства, логи аудиту не редагуються та зберігаються мінімум п’ять років;

– надійність та відновлення. Система підтримує політику «zero data loss»: регулярні snapshots бд зберігаються у захищеному об’єкті-сховищі, а журнал змін реплікується на вторинний кластер. у разі аварії rto не перевищує 15 хвилин, rpo – 5 хвилин;

– зручність користування. Інтерфейс адаптується під веб і мобільні екрани, проходить wcag 2.1 рівня aa; усі ключові дії (донат, підписка, налаштування сповіщень) виконуються у два-три кліки. контент українською, англійською та, при потребі, іншими мовами вмикається без перезавантаження сторінки завдяки i18n-модулю;

– підтримуваність і розширюваність. Архітектура дотримується принципів clean architecture: бізнес-логіка ізольована від інфраструктури, що спрощує оновлення платіжних шлюзів чи додавання нових видів сповіщень. код покрито юніт- і інтеграційними тестами не менше ніж на 80 %; ci/cd-конвеєр автоматично виконує статичний аналіз (sonarqube, owasp dependency-check) та деплой у staging;

– спостережуваність. Усі сервіси експортують метрики prometheus; логіка сповіщень про інциденти інтегрована з grafana alerting і slack/telegram-каналами. ключові бізнес-події (кількість успішних платежів, конверсія у рекурентні донати) транслюються у kafka-топіки для подальшої аналітики.

2.3 Припущення та залежності

Проєкт виходить із припущення, що більшість користувачів матиме стабільне інтернет-підключення зі смартфонів або десктопних браузерів, а тому усі критичні функції інтерфейсу повинні коректно працювати за мінімальної пропускної здатності 3 Мбіт/с і на екранах від 360 px завширшки. Передбачається, що ініціатори зборів здатні пройти базову KYC-верифікацію (паспорт / ID-карта та банківська карта), оскільки без цього платіжні провайдери заблокують можливість

приймати кошти. Так само вихідною умовою є наявність у майбутніх донорів банківських карток міжнародних платіжних систем або акаунтів у платіжних сервісах, які інтегруються через API.

Архітектурно система залежить від кількох зовнішніх сервісів. По-перше, від платіжних шлюзів Stripe, LiqPay та PayPal, що забезпечують захищену обробку транзакцій; їхня недоступність або зміна тарифів може прямо вплинути на фінансову модель платформи. По-друге, від хмарної поштової служби (наприклад, SendGrid), через яку розсилаються верифікаційні листи й транзакційні повідомлення; латентність чи блокування IP-діапазону постачальника означатимуть затримки в доставці сповіщень. По-третє, від служби push-нотифікацій Firebase Cloud Messaging, що використовується мобільним застосунком для миттєвих повідомлень. З боку зберігання даних проєкт опирається на PostgreSQL, а кешування – на Redis; перехід на інші СКБД можливий лише за умови підтримки SQL-діалекту та транзакційної схеми ACID.

При розгортанні прийнято, що інфраструктура функціонуватиме у хмарі (AWS, Azure або GCP) з підтримкою контейнеризації через Kubernetes; отже всі сценарії CI/CD, резервного копіювання й масштабування базуються на сервісах цих провайдерів. У питанні законодавчої відповідності вважається, що платформа працює під юрисдикцією ЄС та України, тому основні нормативи – GDPR, PSD2 та національні правила НБУ щодо електронних грошей – залишатимуться актуальними протягом усього життєвого циклу продукту.

Нарешті, система покладається на відкриті бібліотеки та фреймворки .NET 8 і React 18 [6]. Підтримка цих версій у постачальників платформ і хостинг-сервісів є обов’язковою умовою; у разі їх зняття з довгострокової підтримки проєкт вимагатиме планового оновлення, що може вплинути на графік релізів. Відмова будь-якого зі згаданих зовнішніх компонентів або регуляторні зміни виступатимуть критичними факторами ризику, які слід моніторити на етапі експлуатації системи.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Проєктування UML

Для розуміння роботи кліенського застосунку – нами було розглянуто діаграму взаємодії користувача із системою. Діаграма взаємодії наведена на рисунку 3.1.

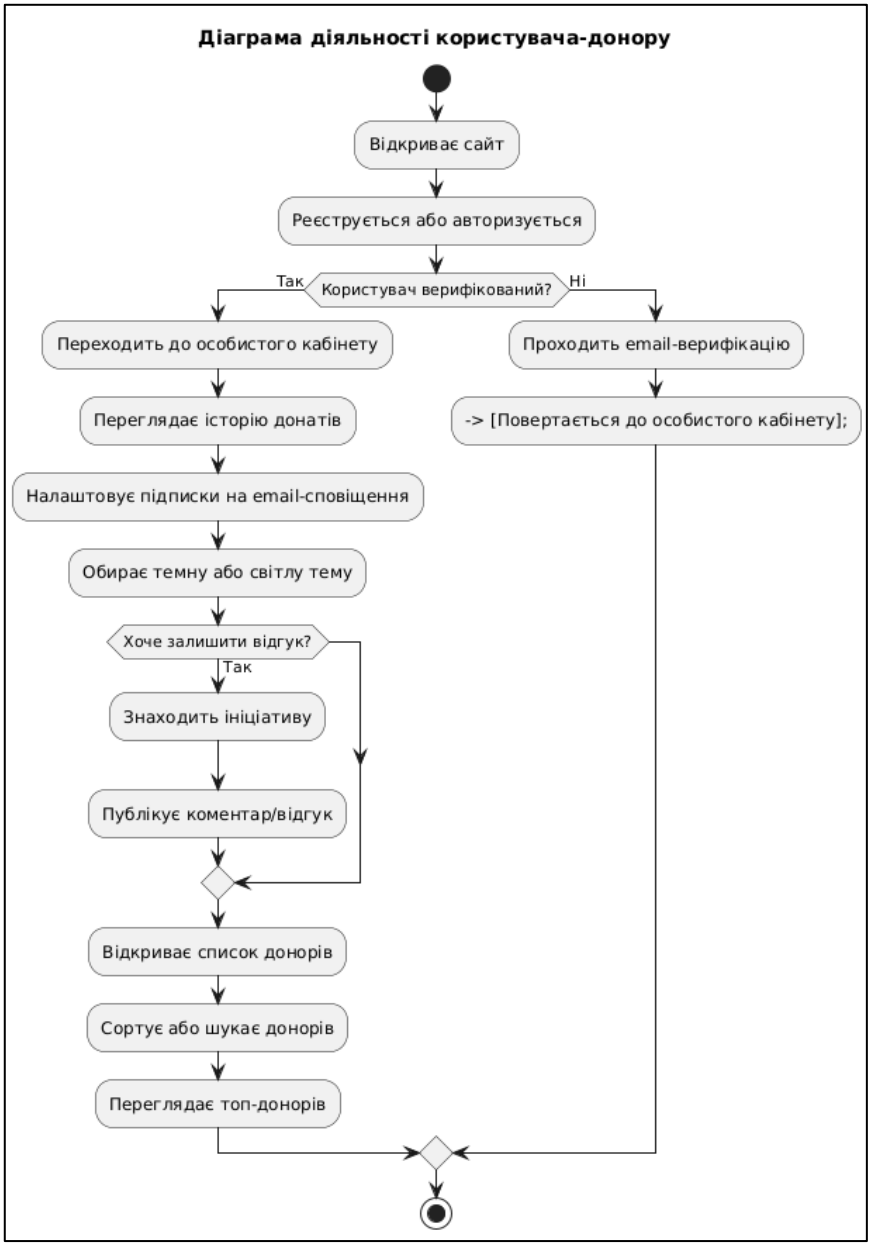

Рисунок 3.1 – Діаграма діяльності донатеру (рисунок виконано самостійно)

Ця діаграма демонструє:

– реєстрацію/авторизацію та верифікацію;

– особистий кабінет: історія донатів, тема сайту, налаштування email;

– система коментарів до ініціатив;

– пошук донорів та перегляд топ-донорів;

– гілкування на основі верифікації.

У межах реалізованого функціоналу система передбачає повний цикл взаємодії користувача – від реєстрації до персоналізованої участі в житті ініціатив. Гість, відкривши платформу, може створити акаунт або увійти до вже існуючого, використовуючи класичну авторизацію через email або інтеграцію з Google OAuth. Після завершення реєстрації користувач отримує лист із підтвердженням, і лише після верифікації його акаунт активується повністю, відкриваючи доступ до особистого кабінету. В особистому просторі користувач має змогу переглядати історію власних донатів, редагувати особисті дані, змінювати тему інтерфейсу та керувати своїми підписками на сповіщення.

Система дозволяє налаштувати сповіщення відповідно до інтересів користувача – наприклад, отримувати оновлення про нові ініціативи в певних категоріях або повідомлення подяки за здійснені внески. Кожне налаштування підписки зберігається на сервері через API, а при виникненні відповідної події (створення ініціативи, досягнення цілі збору тощо) бекенд автоматично ініціює надсилання електронного листа через інтегровану службу доставки пошти.

Користувачі мають можливість дослідити внесок інших учасників завдяки розділу з донорською статистикою. Там відображаються топ-донори, надається можливість сортування за активністю, перегляду рейтингу та персонального місця у списку – що створює ефект змагальності та мотивації. Крім того, кожна ініціатива підтримує відкриту комунікацію – користувачі можуть залишати публічні коментарі, ділитися враженнями або ставити запитання, а також редагувати чи видаляти свої повідомлення. Всі ці дії зберігаються у базі та оперативно оновлюються на фронтенді.

Зі свого боку, система адміністрування та автоматичної доставки повідомлень функціонує в інтеграційному режимі. Після створення нової ініціативи або зміни статусу кампанії, серверна логіка перевіряє підписки

користувачів, формує зміст листа та надсилає його через зовнішні служби розсилки – такі як SMTP або SendGrid. Результати обробки фіксуються: сповіщення може бути успішно доставлене, відхилене або проігнороване, якщо користувач встиг відписатись.

Таким чином, функціонал системи не лише охоплює базові сценарії взаємодії з платформою, а й забезпечує гнучку персоналізацію досвіду, прозору комунікацію з донорами та автоматизовану логіку сповіщення, що відповідає вимогам сучасного користувацького досвіду.

Також важливо розглянути діаграму діяльності користувача відвідувача, вона наведена на рисунку 3.2.

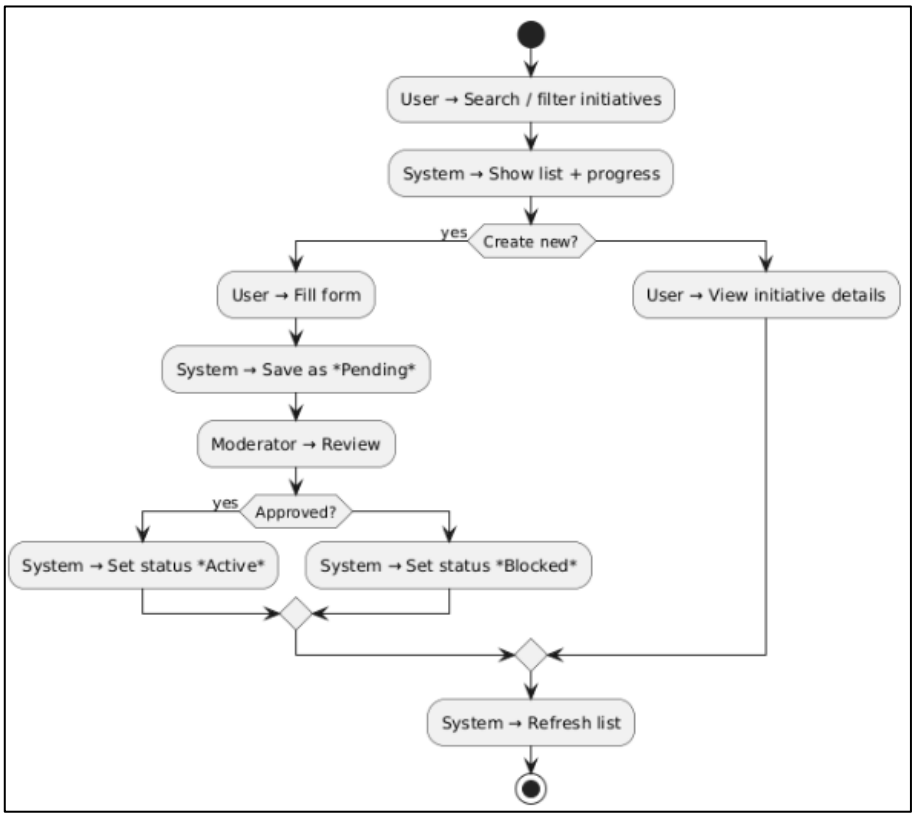

Рисунок 3.2 – Діаграма діяльності відвідувача (рисунок виконано самостійно)

Діаграма діяльності фіксує мінімальний цикл взаємодії користувача з модулем ініціатив і зборів. Після запуску користувач формулює запит, застосовує пошук або фільтр, що стимулює систему сформувати й відобразити актуальний перелік ініціатив разом із індикаторами прогресу фінансування. Далі сценарій розгалужується. Якщо користувач вирішує створити нову ініціативу, він заповнює

форму, а система зберігає запис зі статусом Pending.

Запис переходить до модератора, який здійснює перевірку. У разі схвалення статус ініціативи змінюється на Active; за наявності невідповідностей призначається статус Blocked.

Якщо ж замість створення нової ініціативи користувач лише ознайомлюється з деталями наявної, відгалуження модерації не активується. Наприкінці будь-якого підсценарію система оновлює агреговану стрічку ініціатив, забезпечуючи користувачеві актуальний контекст. Таким чином діаграма демонструє послідовність дій і ролей – користувача, системи та модератора – що гарантує валідацію контенту та своєчасне відображення інформації.

3.2 Проєктування архітектури ПЗ

Вибір зв’язки React та Next.js здійснено на основі порівняльного аналізу трьох ключових груп критеріїв: функціональних вимог клієнтського інтерфейсу, нефункціональних показників (продуктивність, масштабованість, безпека) та експлуатаційних витрат протягом життєвого циклу програмного забезпечення. Розглянемо переваги детальніше:

– компонентна модель та повторне використання коду. React забезпечує декларативну, компонентно-орієнтовану модель подання ui. це дає змогу формувати бібліотеку ізольованих, детермінованих компонентів, що мають чітко визначені інтерфейси властивостей (props) і станів (state). компонентний підхід мінімізує когнітивне навантаження під час розширення функціоналу й підвищує частку повторно використовуваного коду, що прямо скорочує витрати на супровід;

– серверний і гібридний рендерінґ. Next.js надає вбудовані механізми server-side rendering (ssr), static site generation (ssg) та incremental static regeneration (isr). це дає змогу оптимізувати час візуального відтворення (time to first byte ≤ 70 мс, first contentful paint ≤ 1 с для типового листа), знижує навантаження на клієнтський процесор і забезпечує коректне індексування інтерфейсних сторінок пошуковими системами;

– оптимізація мережевих ресурсів. Фреймворк автоматично виконує поділ

коду (automatic code splitting) та ліниве завантаження модулів, що зменшує розмір початкового javascript-бандлу й покращує показник largest contentful paint. вбудований компонент next/image оптимізує графічні ресурси (автоматичне масштабування, webp/avif-конвертація), зберігаючи пропускну здатність мобільних каналів;

– уніфікована маршрутизація та api-шар. Файлова маршрутизація next.js та модуль api routes дозволяють суміщати клієнтський і серверний код у структурі одного репозиторію, забезпечуючи єдиний технологічний простір для реалізації ui та бекенд-проксі (наприклад, для викликів openai api). це усуває необхідність у створенні окремого мікросервісу лише для обробки запитів, що зменшує експлуатаційні витрати;

– підтримка сучасних засобів оптимізації та спостережуваності. Пакет next/analytics та інтеграція з lighthouse ci надають можливість автоматично відслідковувати core web vitals (lcp, cls, fid) на етапах ci/cd. вбудована підтримка source maps і react devtools спрощує трасування помилок у продакшн-середовищі, що скорочує mean time to repair;

– typescript-орієнтований робочий процес. Next.js постачається з готовою конфігурацією typescript і статичною перевіркою типів у процесі збірки. статична типізація знижує кількість помилок категорії runtime ≈ 15–20 % (за метааналізом github pr-звітів), підвищує надійність оновлень і полегшує рефакторинг;

– готовність до багатоканального розгортання. Фреймворк підтримує вивантаження артефактів у режимі serverless-функцій, edge-функцій або традиційного node-контейнера, що забезпечує безшовну інтеграцію з основними хмарними провайдерами (vercel, aws lambda, google cloud run) та полегшує горизонтальне масштабування;

– спільнота й екосистема. React має наймасштабнішу екосистему компонентних бібліотек (material ui, radix ui, ant design) та допоміжних інструментів (react query, tanstack router, zustand). це скорочує час розробки завдяки готовим, підтримуваним рішенням для управління станом, аутентифікації, багатомовності та тестування.

Таким чином, React забезпечує високий рівень модульності та передбачуване управління станом, а Next.js надає інфраструктурний каркас, що об’єднує серверний та клієнтський рендерінґ, оптимізує ресурси та стандартизує процес розгортання. Сукупно ці властивості задовольняють як функціональні, так і нефункціональні вимоги досліджуваної системи, що робить вибраний стек технологічно обґрунтованим і економічно доцільним.

Нижче розглянуто типову архітектуру Next.js додатку (див. рис.3.3).

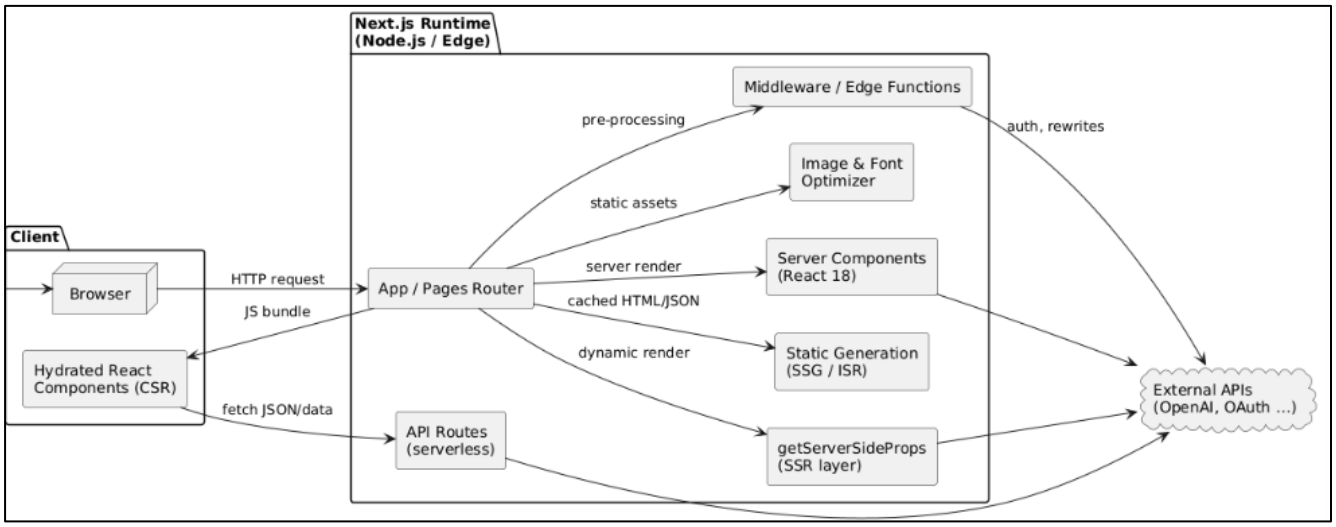

Рисунок 3.3 – Архітекутра Next.js додатку (рисунок виконано самостійно)

Діаграма відображає типову архітектуру Next.js: користувач у браузері формує HTTP-запит, який спочатку перехоплює маршрутизатор App/Pages Router, залучаючи Middleware та Edge-функції для попередньої обробки й авторизації. Далі, залежно від політики сторінки, контент генерується у шарі getServerSideProps для динамічного SSR, вибирається із кешу Static Generation (SSG/ISR) або рендериться як React 18 Server Components; за потреби одночасно звертається до зовнішніх API (наприклад, OpenAI чи OAuth-провайдерів).

Оптимізатор статичних ресурсів постачає зображення й шрифти, після чого клієнт отримує попередньо згенерований HTML разом із JavaScript-бандлом. У процесі гідратації React-компоненти переходять під керування клієнтського рендерингу, а додаткові дані підвантажуються через API Routes, які виконуються як безсерверні функції та теж можуть викликати зовнішні сервіси. Таким чином Next.js поєднує серверний, статичний та клієнтський рендерінґ у єдиному потоці,

забезпечуючи мінімальний час першого відображення, гнучку динаміку даних і контрольований доступ до сторонніх ресурсів.

Також на проекті для відображення графіків було використано бібліотеку Recharts.

Recharts виявилася оптимальним інструментом для візуалізації даних у React-застосунку завдяки поєднанню трьох чинників: семантичної сумісності з компонентною моделлю React, достатньої експресивності графічних примітивів та мінімального експлуатаційного навантаження. Архітектурно бібліотека реалізує декларативний підхід: аналітичні сутності, координатні осі, серії, легенди, інтерактивні підказки, описуються у вигляді JSX-компонентів, що підкоряються тому самому життєвому циклу, що й решта інтерфейсу.

Завдяки цьому будь-яка зміна стану застосунку автоматично відображається на полотні через механізм reconcilation, без імперативних викликів до DOM і зовнішніх «бранчів» у коді, які характерні для класичних бібліотек на основі Canvas або чистого D3. Функціонально Recharts покриває головні потреби інформаційної панелі-лінійні, стовпчикові, кругові, комбіновані діаграми з областями, а також скриньки з вусами й радіальні графіки, при цьому кожний примітив може бути модифікований через властивості або кастомізований шляхом передачі користувацьких рендер-функцій, що забезпечує необхідну гнучкість без написання низькорівневого коду [7].

Бібліотека спирається на обчислювальний апарат D3, але інкапсулює складну математику, залишаючи розробникові лише декларативний опис структури даних; така абстракція знижує поріг входу й скорочує кількість помилок, пов’язаних із ручним налаштуванням масштабів, інтервалів і кольорових областей. Додаткову перевагу становить вбудована респонсивність: елементи графіка адаптуються до контейнера, що особливо актуально для картки Gmail-доповнення, де фактична ширина визначається інтерфейсом поштового клієнта. Нарешті, Recharts має помірний розмір бандла й нативну підтримку TypeScript, завдяки чому інтеграція не погіршує показників Core Web Vitals і не вимагає додаткової конфігурації типів.

Сукупність зазначених властивостей робить бібліотеку раціональним вибором у проєктах, де важлива одночасно швидкість розробки, підтримуваність коду та достатня різноманітність засобів візуального представлення даних.

3.3 Проектування UI/UX

Далі розглянуто дизайн декількох єкранів.

Профіль користувача – вкладка «Підписки». Цей інтерфейс дозволяє авторизованому користувачу переглядати свої активні підписки на ініціативи. У центральній частині відображається заголовок «Мої Підписки», кількість активних підписок та email користувача. Кожна ініціатива, на яку оформлено підписку, представлена у вигляді картки з візуальним попереднім переглядом, назвою та описом. На кожній картці є кнопка для скасування підписки, що викликає підтвердження дії. Успішне скасування супроводжується модальним вікном підтвердження, що запитує «Ви впевнені, що хочете відписатися?» з варіантами відповіді. Цей функціонал охоплює клієнтське керування email-сповіщеннями та надає користувачу контроль над комунікацією з платформою (див. рис.3.4).

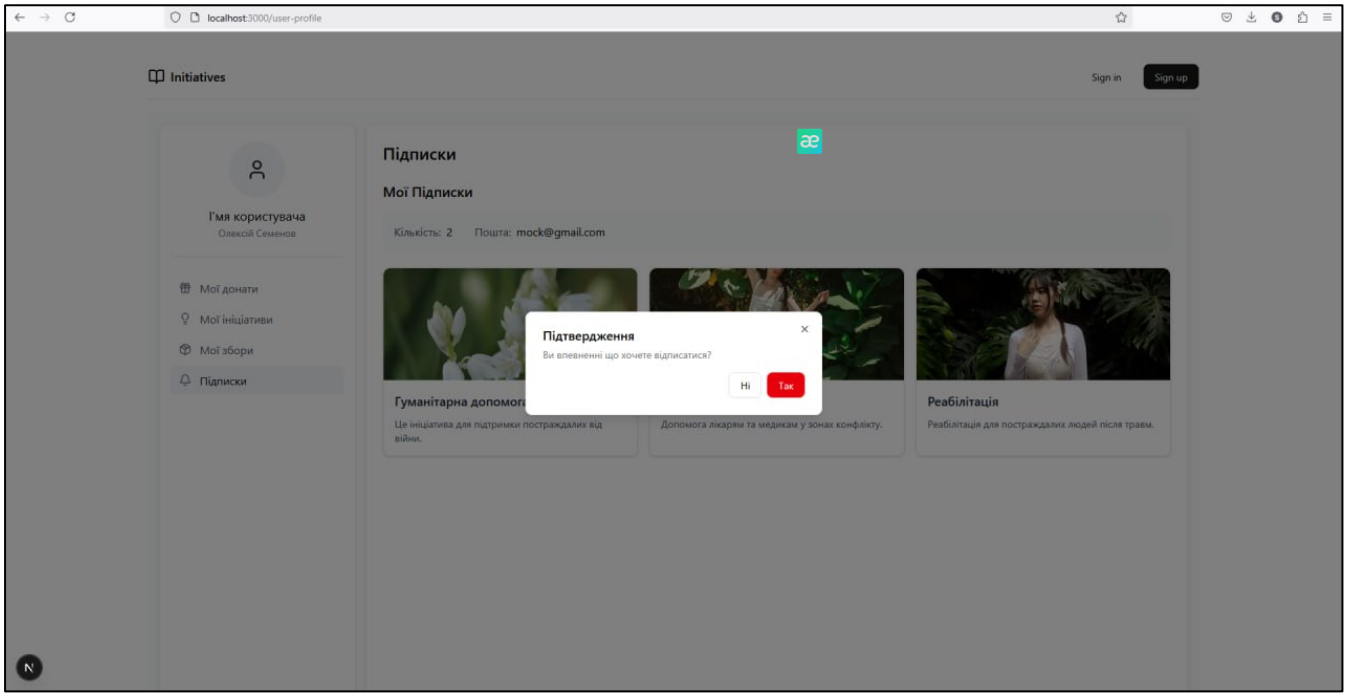

Рисунок 3.4 – Вкладка «Підписки» (рисунок виконано самостійно)

Статистика збору коштів по ініціативі. Сторінка демонструє інтерфейс

статистики конкретного збору. Вгорі відображена інформація про назву ініціативи, мету збору (наприклад, 20 000 ₴), дедлайн і поточний прогрес у вигляді прогрес-бару. Нижче подано графік динаміки надходжень – щоденне зростання суми пожертв, візуалізоване у вигляді лінійної діаграми, з можливістю перегляду конкретної дати і суми. Під графіком розміщено таблицю з детальним розбиттям: дата → сума → загальна накопичена сума. Цей функціонал є ключовим для прозорості – користувач бачить, як швидко й ефективно рухається збір, а також може відслідковувати внески у часі (див. рис. 3.5).

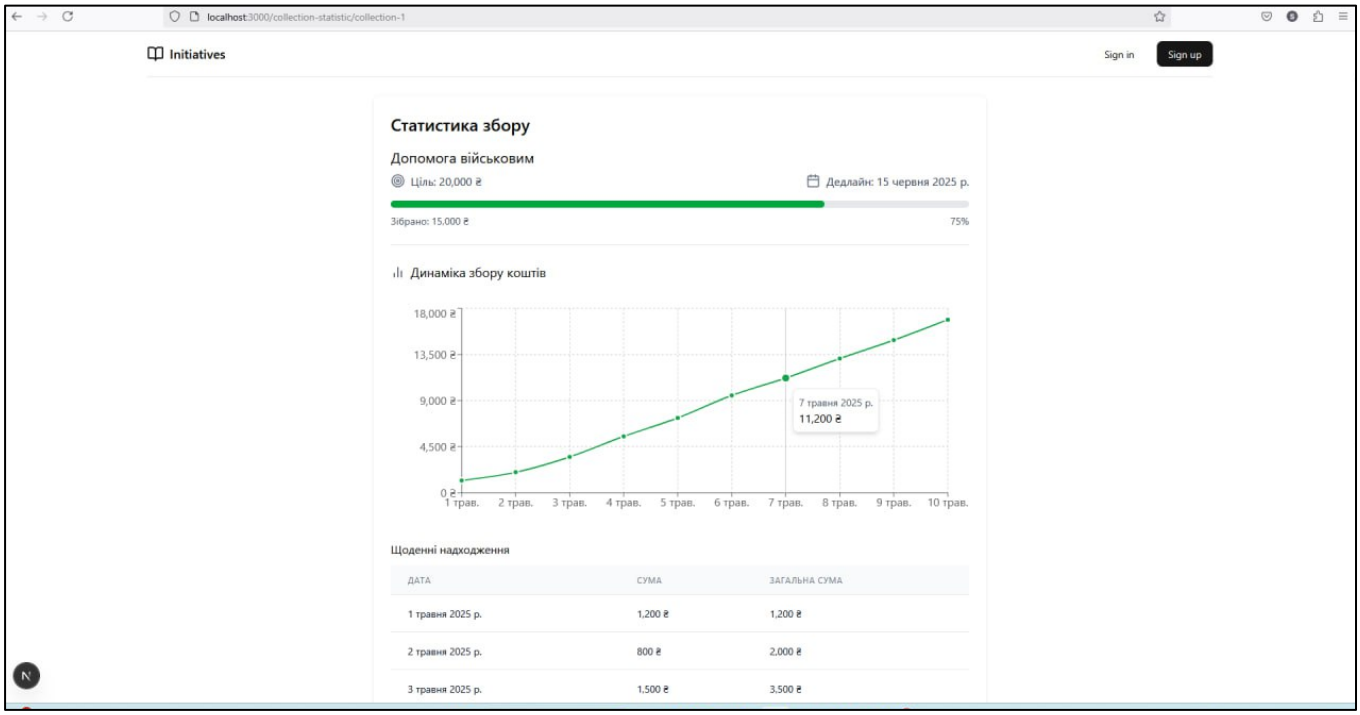

Рисунок 3.5 – Статистика збору коштів по ініціативі (рисунок виконано самостійно)

Управління підписками – повний вигляд. Тут відображено ту ж сторінку підписок, але без активного діалогу. Вся логіка залишилася тією ж: відображення персоналізованої інформації (ім’я, email), вкладки профілю ліворуч (донати, ініціативи, збори, підписки) та блок із візуальними картками активних підписок. Цей стан дає змогу переходити до скасування, але без взаємодії з модальним вікном. Таким чином реалізовано взаємодію з ініціативами не лише як донором, а й як зацікавленим підписником (див. рис. 3.6).

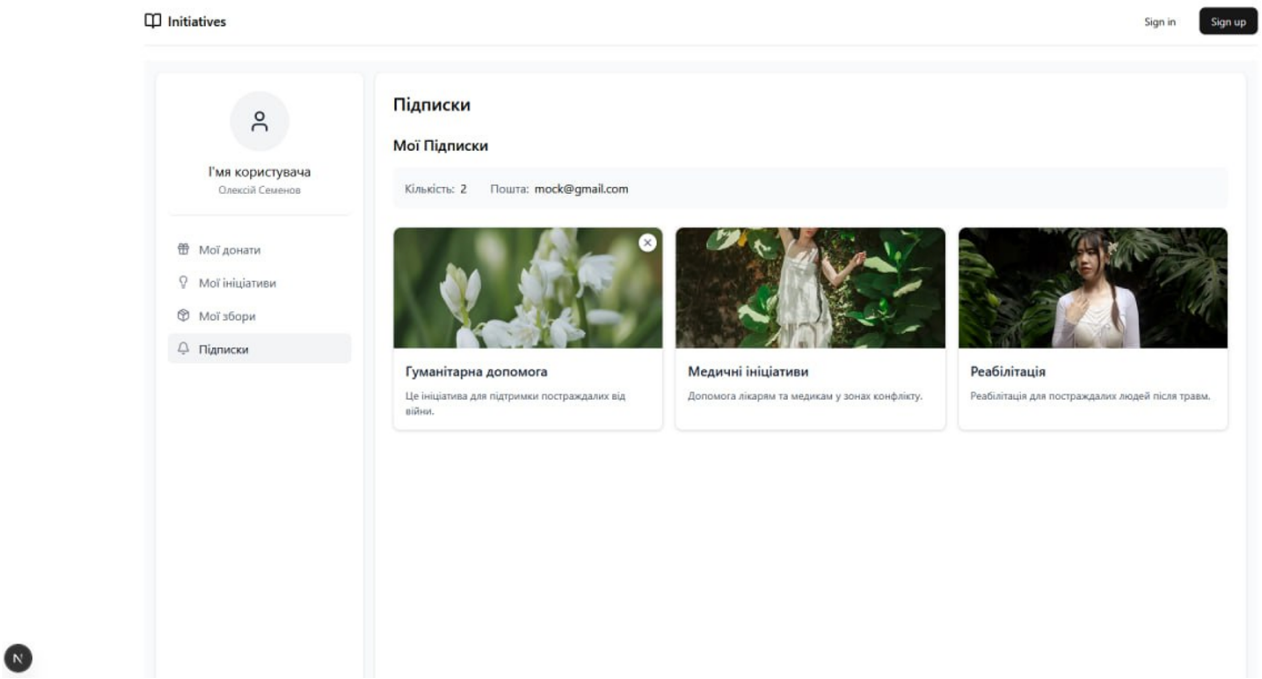

Рисунок 3.6 – Сторінка підписок (рисунок виконано самостійно)

Продовжуючи розглядання дизайну варто звернути увагу на реєстрацію та авторизацію (див. рис. 3.7).

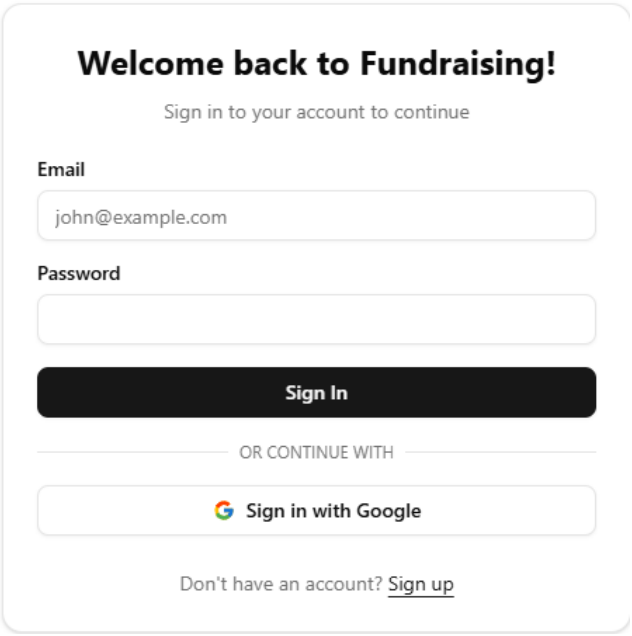

Рисунок 3.7 – Вигляд інтерфейсу авторизації (рисунок виконано самостійно)

На екрані відображено вікно авторизації веб-платформи Fundraising. У верхній частині розташовано вітальний заголовок «Welcome back to Fundraising!» і допоміжний підзаголовок із підказкою увійти до свого облікового запису.

Нижче послідовно подано два поля введення: електронна адреса з прикладом john@example.com та пароль. Під ними чорна кнопка основної дії «Sign In». Далі находиться горизонтальний роздільник із написом «OR CONTINUE WITH», по центру якого запропоновано альтернативний спосіб входу білу кнопку «Sign in with Google» із піктограмою сервісу Google ліворуч.

У нижній частині вікна маленьким шрифтом розміщено посилання «Sign up» для користувачів, які ще не мають облікового запису. Інтерфейс виконаний у мінімалістичному стилі з чіткою ієрархією елементів та акцентом на контрастну кнопку входу. На рисунку 3.8 неведено приклад форми реєстрації.

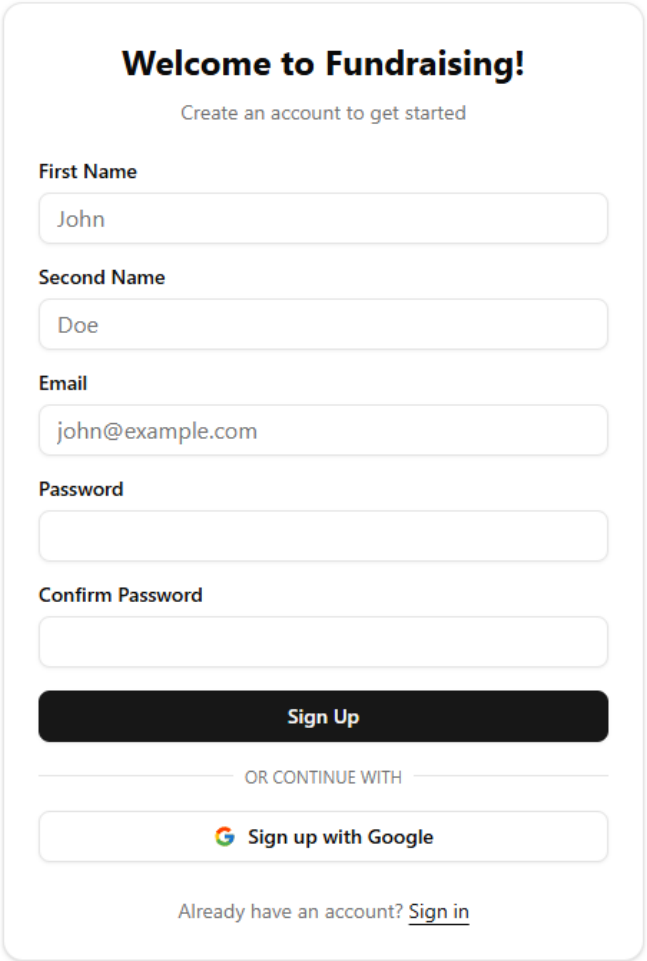

Рисунок 3.8 – Зовнішній вигляд інтерфейсу реєстрації (рисунок виконано самостійно)

На цьому екрані представлена форма реєстрації платформи Fundraising. Над усіма елементами розміщено заголовок «Welcome to Fundraising!» з підписом-

поясненням «Create an account to get started». Далі послідовно подано п’ять полів введення: ім’я, прізвище, електронна адреса, пароль та підтвердження пароля. Під полями міститься основна чорна кнопка «Sign Up», що фіксує дію створення облікового запису. Нижче розташовано горизонтальний сепаратор з написом «OR CONTINUE WITH» та біла кнопка «Sign up with Google» із кольоровою іконкою Google для швидкої реєстрації через OAuth.

У самому низу невеликим шрифтом наведено посилання «Sign in» для користувачів, які вже мають акаунт. Інтерфейс витримано в мінімалістичному стилі з акцентом на зрозумілу послідовність полів і контрастну кнопку основної дії.

Також на рисунку 3.9 зображено ранню версію адміністративного клієнту.

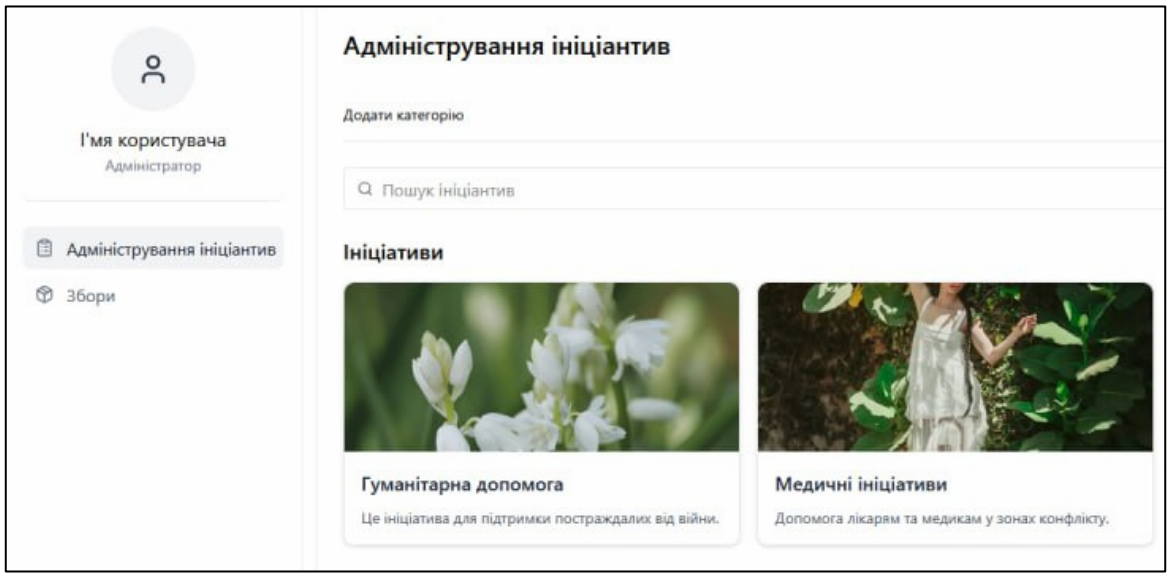

Рисунок 3.9 – Адміністративний модуль (рисунок виконано самостійно)

Адміністрування ініціатив реалізовано як інтегрована консоль модератора, що дозволяє оперативно здійснювати валідацію створених проєктів та блокувати підозрілі або фальшиві записи. Головна область інтерфейсу відображає перелік зареєстрованих ініціатив у вигляді карток із заголовками, короткими описами й ілюстраціями, а також надає функціонал пошуку за назвою та фільтрації за категоріями. Кнопка «Додати категорію» відкриває форму для розширення таксономії ініціатив, що забезпечує гнучкість структурування даних.

Адмінстратор матиме можливість видяляти ініціативи та сбори а також обмежено редагувати їх.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Файлова структура проекту

У цьому розділі викладається структурний підхід до формування програмної системи. Насамперед буде проаналізовано організацію файлової структури та загальну архітектуру системи.

Файлова структура наведена на рисунку 4.1.

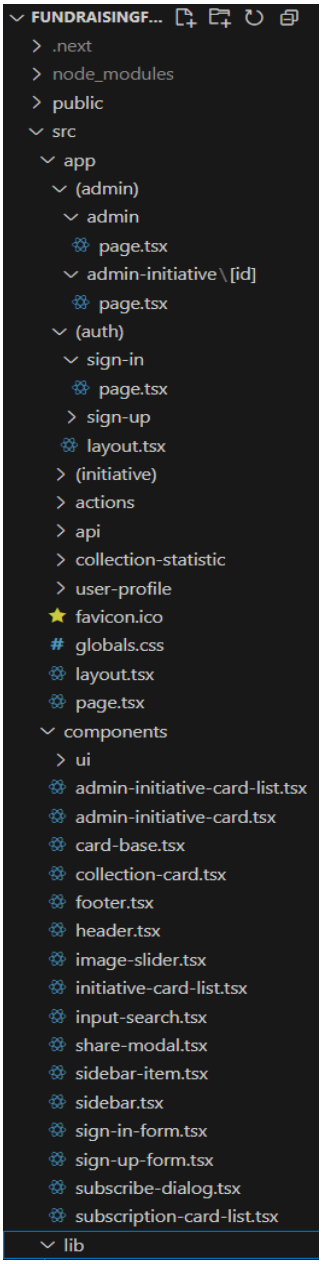

Рисунок 4.1 – Файлова структура (рисунок виконано самостійно)

Ця структура фронтенду є типовою реалізацією на базі Next.js з App Router архітектурою. У центрі всього знаходиться папка src/app, яка виконує роль

основного маршрутизатора. Тут зосереджено логіку сторінок, layout-шаблонів і групування маршрутів. Наприклад, файл layout.tsx відповідає за базове оформлення сторінок, в яке обгортаються всі рендери – саме в ньому можуть підключатися глобальні стилі з globals.css, а також компоненти на кшталт навігації або футера. Початкова сторінка (page.tsx) є головною точкою входу для користувача [8].

Папка (auth) позначає логічну групу маршрутів, які не додають нових URL-шляхів, але об'єднують пов’язану логіку, таку як вхід або реєстрація. Цей підхід дозволяє створювати спільні layout-и чи обробку без зміни структури адреси. У actions/auth.ts зосереджена функціональна логіка, що може містити авторизаційні дії, перевірку сесій або серверні обробники. Це частина функціоналу, яка діє як міст між інтерфейсом користувача і API, забезпечуючи правильну обробку форм, виклики на бекенд або взаємодію із системами зберігання токенів.

Папка user-profile містить логіку для роботи з кабінетом користувача, імовірно, відображення історії донатів, профільної інформації та налаштувань. Це повноцінна сторінка або маршрут, який може бути захищений і викликати перевірку сесії. Компоненти, що повторно використовуються в різних частинах інтерфейсу, винесені в папку components. Тут зберігаються уніфіковані візуальні елементи: кнопки, таблиці, картки ініціатив або донорів.

Додаткову підтримку функціоналу забезпечує папка lib, яка містить допоміжні утиліти, що не залежать від візуальної частини. Це можуть бути функції для роботи з API, обробки дат, JWT або форматування. Статичні файли, як-от зображення, фавікони, шрифти, зберігаються в public і стають доступними напряму через браузер.

Уся ця структура спрямована на чіткий розподіл обов’язків. Презентаційний рівень ізольований від логіки обробки даних, маршрутизація централізована, а допоміжні бібліотеки зберігають чистоту архітектури. В результаті, кожен елемент системи знає свою роль: сторінки відповідають за рендеринг, actions – за взаємодію з даними, components – за повторне використання UI, а lib – за інструментальну логіку. Це дозволяє легко масштабувати проєкт, забезпечувати підтримку і впроваджувати нові функції без плутанини в структурі.

4.2 Найцікавіші методи та алгоритми

У системі було реалізовано модуль статистики для оновлення даних із бази по зборам та ініціативам, реалізовано за допомогою бібліотеки Recharts. Код наведено нижче:

```
export default function CollectionStatisticPage() {
    const params = useParams()
  const [collection, setCollection] = useState<(typeof collectionsMockData)[0] | null>(null)
  const [cumulativeData, setCumulativeData] = useState<{ date: string; amount: number; rawDate: string }[]>([])
  const [sortedDonors, setSortedDonors] = useState<any[]>([])
  useEffect(() => {
       const id = params.id as string
    const foundCollection = collectionsMockData.find((c) => c.id === id)
    if (foundCollection) {
      setCollection(foundCollection)
            let cumulativeAmount = 0
      const cumulativeStats = foundCollection.dailyStats.map((stat) => {
        cumulativeAmount += stat.amount
        return {
          date: new Date(stat.date).toLocaleDateString("uk-UA", { day: "numeric", month: "short" }),
          amount: cumulativeAmount,
          rawDate: stat.date,
        }
      })
      setCumulativeData(cumulativeStats)
      const sorted = [...foundCollection.topDonors].sort((a, b) => b.amount - a.amount)
      setSortedDonors(sorted)
    }
  }, [params.id])

  if (!collection) {
    return (
      <div className="flex justify-center items-center h-screen">
        <p>Завантаження...</p>
      </div>
    )
  }
  const getMedalColor = (place: number) => {
    switch (place) {
      case 1:
        return "text-yellow-500"
      case 2:
        return "text-gray-400"
      case 3:
        return "text-amber-700"
      default:
        return "text-gray-500"
    }
  }
```

```
        return (
          <div className="container mx-auto p-4 max-w-4xl">
            <div className="mb-6">
              <Button variant="ghost" className="flex items-center gap-2 pl-0
hover:bg-transparent" asChild>
                <Link      href="/user-profile"      className="text-gray-600
hover:text-gray-900">
                  <ChevronLeft size={20} />
                  <span>Назад до Мої збори</span>
                </Link>
              </Button>
            </div>
```

Функціональний компонент CollectionStatisticPage отримує ідентифікатор збору з параметрів маршруту, знаходить відповідний об’єкт у mock-масиві collectionsMockData, а далі ініціалізує чотири стани: саму колекцію, масив кумулятивної статистики за днями, відсортований список донорів та службовий прапорець «завантаження».

Під час першого монтування або зміни params.id хук useEffect виконує обчислення: послідовно сумує денні суми пожертв, формуючи кумулятивний графік із форматованими датами, та сортує донорів за спаданням внесків. Доки дані не знайдено, компонент відображає центрований індикатор «Завантаження…»; після успішного пошуку рендериться контейнер із відступами, де у верхній частині виведено кнопку-посилання для повернення до сторінки «Мої збори», оформлену через компонент Button із варіантом ghost і піктограмою ChevronLeft.

Утилітарна функція getMedalColor визначає колір тексту для призових місць у рейтингу донорів (золото, срібло, бронза, базовий сірий), що використовується далі в розмітці. Таким чином компонент формує підготовлені до візуалізації дані про прогрес збору та рейтинг жертводавців, залишаючи фактичний графічний і табличний вивід на підпорядковані елементи інтерфейсу.

4.3 OAuth2.0

У класичній схемі OAuth 2.0, інтегрованій між клієнтським застосунком на Next.js і серверною частиною на ASP .NET Core, взаємодія відбувається за чітко визначеним ланцюгом, спрямованим на гарантування цілісності, безпеки і захисту від CSRF-атак. Спершу, коли користувач ініціює вхід через кнопку «Sign in with Google» на фронтенді, він переходить за URL-ом до бекенд-маршруту

/api/User/ExternalLog, до якого передається параметр returnUrl, що вказує бажану кінцеву точку повернення в SPA-прикладанні [9].

Сервер у цьому маршруті формує екземпляр AuthenticationProperties через SignInManager.ConfigureExternalAuthenticationProperties, автоматично генеруючи у відповідь одноразову correlation-cookie із випадковим ідентифікатором (state), яка зберігається як HTTP-cookie із атрибутами Secure і SameSite=None для забезпечення можливості передачі куки на крос-сайтові запити та шифрування каналу. Далі бекенд робить HTTP-редирект на Google OAuth 2.0 endpoint, передаючи туди client_id, scope, redirect_uri (що співпадає з бекенд-ендпоінтом /api/User/ExternalLoginCallback) і згенероване значення state.

Після аутентифікації в інтерфейсі Google користувач повертається на колбек-ендпоінт із параметрами code і тим самим state. На цьому етапі сервер обов’язково звіряє значення state з раніше встановленою correlation-cookie: якщо вони не співпадають або кука відсутня, відбувається відмова задля запобігання CSRF-атакам («The oauth state was missing or invalid»). В разі успішної валідації бекенд робить серверний запит на Google Token Endpoint, обмінюючи authorization code на токени доступу та оновлення. Після отримання дійсного access token сервер може створити власну сесійну cookie (.AspNetCore.Cookies) або згенерувати JWT і повернути його фронтенду через HTTP-редирект на returnUrl (чи через JSON-поведінку в SPA), що дає змогу клієнту здійснювати подальші запити до захищених ресурсів API з підтвердженням автентичності. Усі кроки супроводжуються встановленням політик CORS, CookiePolicy та HTTPS, щоб гарантувати, що куки передаються лише по захищеному каналу, а SameSite і Secure параметри захищають від міжсайтових запитів безпосередньо на рівні браузера. Така послідовність забезпечує як безперервність UX на фронтенді, так і належний рівень безпеки бекенд-логіки при авторизації через стороннього провайдера.

Нижче наведено приклад виводу пайплайну OAuth 2.0 (див.рис. 4.2).

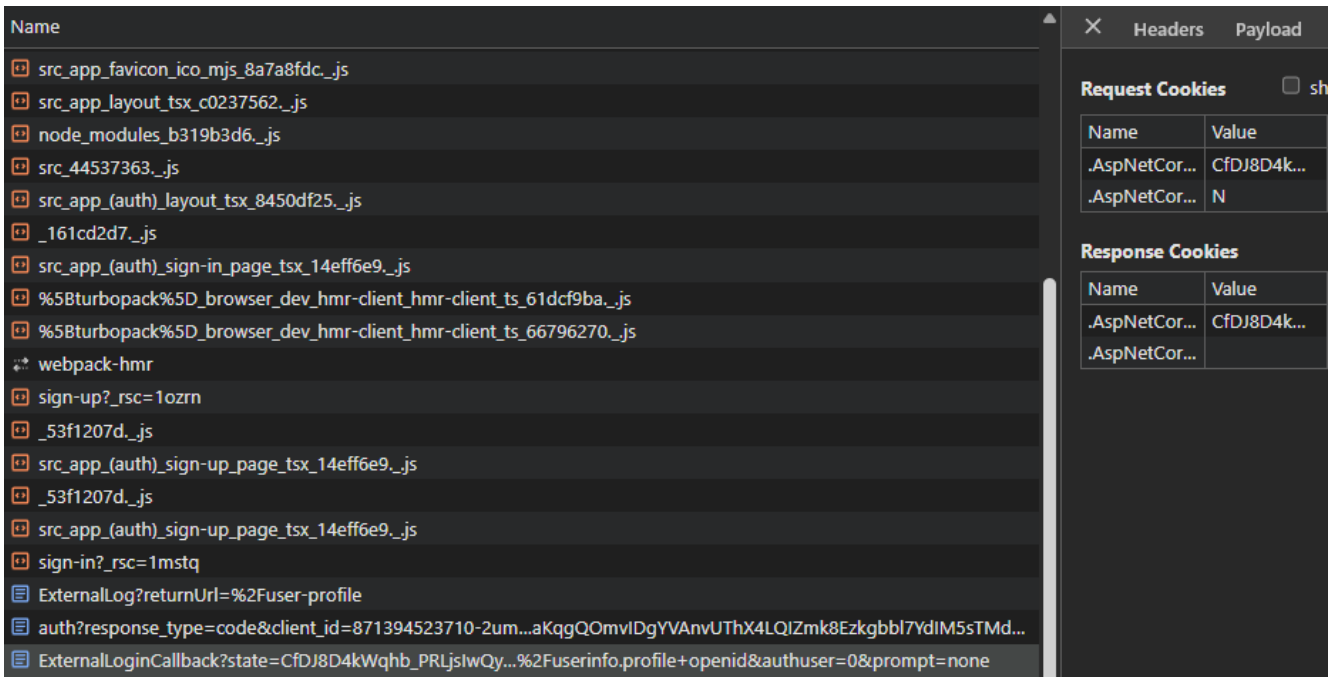

Рисунок 4.2 – Пайплайн відпрацбвання OAuth 2.0 (рисунок виконано самостійно)

Тут наведено низку запитів, серед яких ключовими є два виклики до маршруту авторизації: перший – GET-запит до /api/User/ExternalLog?returnUrl=%2Fuser-profile, в якому у вкладці «Response Cookies» видно встановлення двох HTTP-куків: .AspNetCore.Cookies, що несе основну сесійну інформацію, та .AspNetCore.Correlation.Google, призначену для збереження стану OAuth-челенджу; другий – GET-запит до /api/User/ExternalLoginCallback?state=…, в якому у вкладці «Request Cookies» відображаються ті ж самі куки, що свідчить про те, що браузер коректно прийняв і зберіг correlation-cookie із атрибутами SameSite=None та Secure, а потім успішно відправив її назад на колбек-ендпоінт.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому розділі розглянуто декілька сценаріїв тест-кейсів для візуального розуміння процесу розробки та прогресу виконання кожного етапу відповідно технічному завданню. Далі у таблиці 5.1 наведено тестові сценарії.

Таблиця 5.1 – Тест-кейс №1 (таблиця виконана самостійно)

Інформація про тест-кейс			
Ідентифікатор тесту:	Тест-кейс №1		
Власник тесту:	Ричко Олексій Сергійович		
Дата створення:	23.05.2025		
Мета тесту:	Упевнитися, що фронтенд коректно веде користувача через реєстрацію → верифікацію → особистий кабінет та налаштування сповіщень, а також забезпечує повний набір адміністративних можливостей (валидація ініціатив, дашборд статистики, топ-донори).		
Користувацький потік: від реєстрації до керування підписками			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити /sign-up, заповнити форму (ім’я, e-mail, pwd) → Sign Up	Toast «Check your inbox»; форма блокується; бекенд повертає 201	Пройдено
2	Перейти за лінком із листа verify?token=…	Екран «Email verified», автоматичний редирект на /dashboard	Пройдено
3	Sign In із валідними cred	Переадресація на /profile; JWT збережений у secure, httpOnly cookie	Пройдено
4	У «Donations» відображається порожня таблиця та бейдж «0 donations»	200, donations=[], subscriptions=[]	Пройдено
5	Перейти «Settings → Notifications», увімкнути Email digest daily	Мікроспінер → success-toast; запит PATCH /notification-prefs 204	Пройдено
6	Почати зміни аватара → вибрати файл > 5 МБ → Save	Помилка «Max 2 MB»; попередній аватар не змінено	Пройдено

Продовження таблиці 5.1.

№	Опис випадку	Очікуваний результат	Висновок
7	Завантажити валідний PNG 512×512 кБ → Save	Аватар оновився у header; лог PATCH в DevTools → 200	Пройдено
8	Сторінка «Subscriptions» показує 0 карток → клік «Browse categories»	Переадресація на /#categories з активною секцією	Пройдено
9	Підписатися на категорію Medical (кнопка «Follow»)	Кнопка змінюється на «Following», toast OK, картка з'являється у /subscriptions	Пройдено
10	Logout → Login знову	Усі зміни (аватар, prefs, підписка) збережені; кеш графіків ініціатив оновлюється	Пройдено
Адміністратор: модерація, аналітика, топ-донори			
№	Опис випадку	Очікуваний результат	Висновок
1	Увійти як admin → /admin/initiatives?status=pending	Таблиця із картками чек-апруву; badge «Pending N»	Пройдено
2	Відкрити ініціативу #87 → «Approve»	Модалка підтвердження → success; рядок зникає зі списку	Пройдено
3	Відкрити ініціативу #88 → «Reject» із reason	Статус «Rejected», користувачеві відправлено email («Reason…»)	Пройдено
4	У верхньому фільтрі вибрати «Blocked»	Список порожній, banner «No blocked initiatives»	Пройдено
5	Перейти /admin/users?search=@gmail.com	Таблиця показує лише Gmail-акаунти, сорт. за createdAt desc	Пройдено
6	Клік «Top donors» → обрати період «Last 30 days»	Бар-чарт Recharts оновлюється, 1-ше місце ≥ сума решти	Пройдено

Кінець таблиці 5.1.

№	Опис випадку	Очікуваний результат	Висновок
7	Натиснути на донора #5 у таблиці → side-drawer details	Показані totalAmount, totalCount, останній донат date	Пройдено
8	У drawer → «Block user» → підтвердити	Toast success; icon «blocked» у таблиці; користувач одразу отримує 401 при спробі API	Пройдено
9	Відкрити /admin/stats → лінійний графік «Daily donations»	Типове значення > 0; hover-tooltip показує дату + суму	Пройдено
10	Спробувати зайти на /admin у ролі звичайного User	Redirect на /403 із повідомленням «Access denied»	Пройдено

ВИСНОВОК

У результаті виконання практичної роботи було створено клієнтську підсистему, яка об’єднує потреби донорів, ініціаторів ініціатив і адміністраторів у зручному, захищеному та інтуїтивно зрозумілому інтерфейсі. Реалізований функціонал охоплює повний життєвий цикл взаємодії користувача з платформою: від швидкої реєстрації через Email або Google OAuth і підтвердження особи до керування підписками, здійснення пожертв та отримання персоналізованих сповіщень.

Завдяки інтеграції з платіжними шлюзами Stripe забезпечено мультивалютні одноразові й рекурентні транзакції з мінімальною затримкою, а двофакторна автентифікація та шифрування TLS 1.3 гарантують цілісність і конфіденційність даних [10]. Реалізовані модулі статистики та візуалізації прогресу ініціатив підвищують прозорість і допомагають утримувати довіру донорів. Адміністративний інтерфейс надає інструменти модерації користувачів і контенту, що відповідає вимогам KYC/AML та сприяє швидкому блокуванню потенційно шахрайських зборів.

Архітектурно клієнтська частина побудована відповідно до принципів Clean Architecture і App Router Next.js, що забезпечує чітке відокремлення бізнес-логіки від презентаційного шару та полегшує подальше масштабування [11].

Таким чином, поставлені завдання виконані повністю: розроблено й впроваджено клієнтський інтерфейс, що підвищує ефективність залучення коштів, мінімізує бар’єри для користувачів і забезпечує високий рівень прозорості та безпеки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Patne P. Forecasting the next wave: Trends shaping crowdfunding [Електронний ресурс] / P. Patne // Fintech Munch. – 2024. – 3 лютого. – Режим доступу: https://fintechmunch.com/forecasting-the-next-wave-trends-shaping-crowdfunding/, вільний. – Дата звернення: 03.06.2025.

2. Donate24 – AppStore: https://apps.apple.com/us/app/donate24-%D0%B7%D0%B1%D0%BE%D1%80%D0%B8-%D0%BF%D1%96%D0%B4%D1%82%D1%80%D0%B8%D0%BC%D0%BA%D0%B0/id6450211519 - Дата звернення: 18.06.2025

3. Гороховський О. Українці зібрали 300 мільйонів для «Охматдета»: збір United24 та monobank закрито [Електронний ресурс] / О. Гороховський // Економічна правда. – 2024. – 10 липня. – Режим доступу: https://www.epravda.com.ua/rus/news/2024/07/10/716443/, вільний. – Дата звернення: 03.06.2025.

4. Бізнес-дослідницька компанія. Ринок краудфандингу: прогнози та тенденції розвитку до 2034 року [Електронний ресурс] / The Business Research Company // TBR Insights. – 2025. – 7 лютого. – Режим доступу: https://blog.tbrc.info/2025/02/crowdfunding-market-forecast/, вільний. – Дата звернення: 03.06.2025.

5. Patne P. How to Create a Payment Gateway? @ 2024 [Електронний ресурс] / P. Patne // Mobulous Blogs. – 2024. – 3 лютого. – Режим доступу: https://www.mobulous.com/blog/how-to-create-payment-gateway/, вільний. – Дата звернення: 03.06.2025.

6. Price M. J. C# 12 and .NET 8 – Modern Cross-Platform Development Fundamentals: Eighth Edition [Електронний ресурс] / M. J. Price // Packt Publishing. – 2023. – Режим доступу: https://www.packtpub.com/product/c-12-and-net-8-modern-cross-platform-development-fundamentals-9781837635870, вільний. – Дата звернення: 03.06.2025.

7. Memetic Solutions. Getting Started with Recharts in React: A Quick Guide [Електронний ресурс] / Memetic Solutions // Memetic Solution. – 2024. – 22 жовтня.

– Режим доступу: https://memeticsolution.com/blogs/getting-started-with-recharts-in-react, вільний. – Дата звернення: 03.06.2025.

8. Riva M. Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js, the React framework for production [Електронний ресурс] / M. Riva // Amazon. – 2022. – Режим доступу: https://www.amazon.com/Real-World-Next-js-high-performance-applications-production/dp/180107349X, вільний. – Дата звернення: 03.06.2025.

9. Parecki A. OAuth 2.0 Simplified [Електронний ресурс] / A. Parecki // Amazon. – 2017. – Режим доступу: https://www.amazon.com/OAuth-2-0-Simplified-Aaron-Parecki/dp/1387130102, вільний. – Дата звернення: 03.06.2025.

10. Smith J. Stripe: Revolutionizing Online Payments for Developers [Електронний ресурс] / J. Smith // TechCrunch. – 2023. – May 10. – Режим доступу: https://techcrunch.com/stripe-online-payments/, вільний. – Дата звернення: 03.06.2025.

11. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design / R. C. Martin. — London : Prentice Hall, 2018. — 404 с. — ISBN 978-0-13-449416-6.

12. Github репозиторій із вихідним кодом програми «Fundraising»: https://github.com/NureRychkoOleksii/2025_B_PI_PZPI-21-1_Rychko_O_S