АНОТАЦІЯ

Коваленко Д.М., Кулик І.В. (керівник). Застосунок для проведення голосувань з використанням технології блокчейну. Бакалаврська кваліфікаційна робота, спеціальність F2 «Інженерія програмного забезпечення». – Національний університет «Львівська політехніка», Львів, 2025. – 65 с.

Актуальність теми дослідження зумовлена потребою у створенні прозорих, безпечних та децентралізованих систем електронного голосування, які можуть забезпечити довіру до результатів та запобігти фальсифікаціям. Традиційні централізовані системи мають низку недоліків, таких як вразливість до атак, відсутність прозорості та залежність від довірених органів. Використання блокчейн-технологій дозволяє усунути ці проблеми, забезпечуючи незмінність даних, децентралізацію та автоматизовану перевірку результатів [1].

Метою роботи було розроблення функціонального вебзастосунку для електронного голосування на основі блокчейну Solana. Застосунок дозволяє організаторам створювати голосування, а учасникам — подавати голоси через інтуїтивно зрозумілий інтерфейс. Результати голосування зберігаються у блокчейні, що гарантує їхню незмінність та прозорість. Для реалізації системи використано мову програмування Rust для смарт-контрактів, фреймворк Anchor для їх розробки, а також TypeScript, React та Tailwind CSS для клієнтської частини [2].

Серед *аналогічних рішень* можна відзначити систему Voatz, яка також використовує блокчейн для голосування, але критикується за закритість коду [3], та естонську систему i-Voting, що базується на електронній ідентифікації [4]. На відміну від них, запропонований застосунок поєднує відкритість, високу продуктивність Solana та зручний інтерфейс.

Середовищем функціонування застосунку є браузер із підключенням до блокчейн-мережі Solana (локальна мережа або devnet) та наявність криптогаманця Phantom

Wallet. Застосунок може використовуватися для проведення локальних опитувань, корпоративних голосувань або як основа для подальших досліджень у сфері децентралізованих виборчих систем [5].

Структура пояснювальної записки БКР. Анотація, вступ, п'ять розділів, висновки, список літератури, додатки. Загальний обсяг роботи становить 78 сторінок, у тому числі 53 сторінки основного тексту, 14 рисунки, 15 таблиць та 6 додатків. Бібліографічний опис містить посилання на 11 використаних літературних джерел.

Ключові слова. Solana, смарт-контракт, вебзастосунок, електронне голосування, децентралізований застосунок.

[1] Awosika E. What Is Blockchain Voting?. URL: https://second-pocket-shoot-73.hashnode.dev/what-is-blockchain-voting.

[2] Network Performance Report - October 2022. *Solana*. URL: https://solana.com/news/network-performance-report-october-2022.

[3] Security and technology. *Voatz*. URL: https://voatz.com/security-and-technology/.

[4] Introduction to i-voting. *Valimised*. URL: https://www.valimised.ee/en/internet-voting/more-about-i-voting/introduction-i-voting.

[5] Almeida, R. L., Baiardi, F., Maesa, D. D. F., & Ricci, L. Impact of Decentralization on Electronic Voting Systems: A Systematic Literature Survey. *IEEE Access*. 2023. Ст. 31. DOI: 10.1109/ACCESS.2023.3336593.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, СИМВОЛІВ І СПЕЦІАЛЬНИХ ТЕРМІНІВ

BFT Byzantine Fault Tolerance.

DPoS Delegated Proof of Stake.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

P2P Peer-to-Peer.

PBFT Practical Byzantine Fault Tolerance.

PDA Program Derived Address.

PoA Proof of Authority.

PoC Proof of Capacity.

PoET Proof of Elapsed Time.

PoH Proof of History.

PoS Proof of Stake.

PoSpace Proof of Space.

PoW Proof of Work.

RPC Remote Procedure Call.

SDK Software Development Kit.

TPS транзакції на секунду.

ВСТУП

У сучасному світі цифрові технології відіграють ключову роль у забезпеченні прозорості, достовірності та довіри в різних сферах суспільного життя. Однією з таких сфер є електронне голосування — інноваційний підхід до організації виборів, який дедалі частіше використовується як альтернатива традиційним паперовим методам. Електронне голосування дозволяє зменшити витрати часу та ресурсів, підвищити зручність участі, а також розширити коло учасників. Однак широке впровадження таких систем стикається з рядом викликів, серед яких — забезпечення цілісності даних, запобігання фальсифікаціям, гарантування конфіденційності виборців, а також підтвердження легітимності результатів.

Сучасні централізовані електронні системи голосування залишаються вразливими до низки загроз, включно з технічними збоями, зовнішніми атаками, внутрішніми маніпуляціями та недовірою з боку користувачів. Відсутність повної прозорості процесу створює передумови для спотворення результатів, що у свою чергу знижує рівень довіри суспільства до інституцій, які організовують такі голосування. Одним із перспективних напрямів розв'язання цих проблем є використання технології блокчейн. Блокчейн знаходить застосування в багатьох галузях, включаючи фінанси, логістику, охорону здоров'я та управління ланцюгами поставок. У контексті електронного голосування він дозволяє забезпечити прозорість, конфіденційність і захищеність голосів: кожен голос у такій системі зберігається у вигляді транзакції, яка є незмінною та відкритою для перевірки всіма учасниками. Завдяки цьому досягається високий рівень довіри до процесу голосування навіть без централізованого контролю.

Актуальність теми цієї роботи зумовлена потребою у створенні сучасних електронних систем голосування, які забезпечуватимуть високий рівень безпеки та довіри. З розвитком технологій все більше організацій, компаній та державних

установ прагнуть впроваджувати цифрові рішення для голосування, що потребує детального аналізу існуючих підходів, а також розробки нових моделей, що враховуватимуть їх переваги та недоліки.

Метою роботи є проєктування системи для проведення голосувань та опитувань, яка відповідатиме вимогам прозорості, безпеки та достовірності результатів. Для забезпечення поставленої мети розроблено архітектуру системи голосування, яка відповідає вимогам безпеки, надійності та ефективності.

Для досягнення поставленої мети необхідно виконати низку задач, зокрема провести комплексне дослідження існуючих рішень у сфері електронного голосування, зосередивши увагу на системах, побудованих із використанням технології блокчейн. На основі цього слід визначити ключові переваги та недоліки таких рішень і сформулювати вимоги до безпечної, прозорої та зручної у використанні платформи для голосування. Далі потрібно спроєктувати архітектуру майбутньої системи, що передбачає механізми створення голосувань, автентифікації користувачів, захисту даних і прозорої обробки результатів. Після цього необхідно реалізувати основні компоненти системи та провести тестування з метою перевірки її функціональності, стабільності роботи й відповідності визначеним вимогам безпеки.

Результати цієї роботи можуть бути використані для подальшого розвитку електронних систем голосування, а також як основа для впровадження подібних рішень у різних сферах діяльності — від державного управління до корпоративних голосувань та соціальних опитувань.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ОРГАНІЗАЦІЇ ЕЛЕКТРОННИХ ГОЛОСУВАНЬ

1.1. Основи блокчейн-технології

Блокчейн є децентралізованою технологією зберігання даних, яка дозволяє створювати надійні розподілені системи без потреби в довіреному централізованому органі. Основною її особливістю є здатність забезпечувати незмінність і прозорість інформації завдяки використанню криптографічних методів та особливої структури зберігання. На рис 1.1 зображено загальну структуру блокчейну, що ілюструє зв'язки між блоками, їхній вміст та хеш-значення попередніх блоків [1].

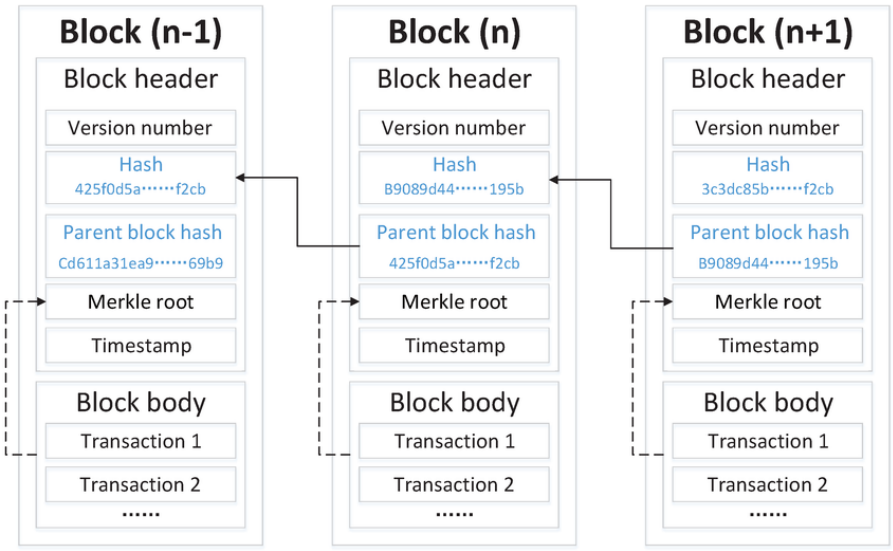

Рис. 1.1 Схема структури блокчейну

Кожен блок у блокчейні містить перелік транзакцій, хеш попереднього блоку та часову позначку [2]. Транзакції в мережі спочатку накопичуються, після чого об'єднуються в блоки — це дозволяє підвищити ефективність обробки та масштабованість системи. Блоки пов'язані між собою у послідовний ланцюг. Якщо змінити хоча б одну транзакцію в блоці, це призведе до зміни його хеша, порушить цілісність усього ланцюга й така спроба буде одразу виявлена. Завдяки

цьому блокчейн є стійким до фальсифікацій і несанкціонованого втручання.

Функціонування блокчейну забезпечується одноранговою Peer-to-Peer (P2P) мережею, в якій кожен вузол має власну копію ланцюга і з'єднаний безпосередньо з іншими учасниками. Вузли взаємодіють між собою, обмінюючись інформацією про транзакції та нові блоки, синхронізуючи свої копії ланцюга. Така архітектура усуває потребу в центральному сервері, забезпечуючи стійкість до збоїв, а також підвищуючи надійність і прозорість системи: усі транзакції доступні для перегляду кожному учаснику, що дозволяє перевіряти достовірність інформації без довіри до окремого адміністратора.

Таким чином, децентралізація, прозорість і незмінність — це основні риси блокчейн-систем, які роблять їх надійним фундаментом для побудови цифрових інфраструктур, включно з фінансовими платформами, ланцюгами постачання та електронним голосуванням.

1.2. Методи досягнення консенсусу у розподілених системах

Однією з ключових складових технології блокчейн є механізм досягнення консенсусу, необхідний для узгодження єдиного порядку транзакцій та додавання нових блоків у децентралізовану мережу. У розподілених системах, де дані зберігаються та обробляються незалежними вузлами без центрального органу управління, виникає ризик розбіжностей між учасниками через одночасне надходження кількох транзакцій або через можливість недобросовісної поведінки окремих вузлів. Без механізму консенсусу система може стати вразливою до атак або втратити свою цілісність, оскільки вузли не зможуть дійти згоди щодо єдиного стану ланцюжка блоків. Ефективний механізм консенсусу вирішує ці проблеми, забезпечуючи синхронізацію дій вузлів, незмінність даних та довіру між учасниками мережі [3].

Один із найпоширеніших алгоритмів консенсусу — Proof of Work (PoW), або доказ виконаної роботи [4]. У цьому алгоритмі вузли мережі змагаються за право додати новий блок до ланцюжка, розв'язуючи складну криптографічну задачу.

Перший вузол, який успішно знаходить розв'язок, отримує винагороду, а його блок додається до ланцюжка. PoW забезпечує безпеку системи завдяки високій обчислювальній складності, проте має значні недоліки, включаючи високе енергоспоживання та обмежену масштабованість.

Інший популярний алгоритм — Proof of Stake (PoS), або доказ частки володіння [5]. У цьому підході вузол, що додає новий блок, обирається пропорційно до кількості криптовалюти, яку він утримує. PoS значно знижує енергоспоживання порівняно з PoW та забезпечує вищу продуктивність, проте може викликати занепокоєння щодо концентрації влади у власників великої кількості криптовалюти.

Серед сучасних альтернативних підходів виділяється Proof of History (PoH) [6]. PoH впроваджує механізм впорядкування подій у часі за допомогою криптографічних міток часу. Це дозволяє вузлам мережі перевіряти порядок транзакцій незалежно від інших вузлів, що значно підвищує швидкість обробки даних і забезпечує високу пропускну здатність. PoH унікальний тим, що додає часовий компонент до процесу консенсусу, що знижує необхідність тривалих узгоджень між вузлами.

Таблиця 1.1

Порівняння алгоритмів консенсусу

Критерій	**PoW**	**PoS**	**PoH**
Безпека	Висока	Висока	Висока
Енергоспоживання	Високе	Низьке	Низьке
Масштабованість	Низька	Середня	Висока
Швидкість	Низька	Середня	Висока

Таким чином, різні алгоритми консенсусу забезпечують баланс між безпекою, продуктивністю та енергоспоживанням. Вибір конкретного механізму залежить від особливостей застосування блокчейну. У контексті електронного голосування важливими факторами є швидкість обробки транзакцій, масштабованість, низькі

витрати та захищеність, що впливає на вибір найбільш відповідного алгоритму для створення ефективної системи. З огляду на це, для реалізації системи електронного голосування найбільш доцільним є використання алгоритму PoH. На відміну від PoW і PoS, PoH забезпечує високу швидкість обробки транзакцій, високу масштабованість та низьке енергоспоживання завдяки вбудованому часовому компоненту, який дозволяє вузлам незалежно та швидко верифікувати порядок подій. Це зменшує затримки при досягненні консенсусу та є критично важливим для забезпечення швидкодії та доступності системи.

Окрім основних методів консенсусу, існують інші підходи, як-от Delegated Proof of Stake (DPoS), де делегати обираються для прийняття рішень; Practical Byzantine Fault Tolerance (PBFT), який застосовується в приватних блокчейнах для забезпечення консенсусу при наявності шахраїв; Proof of Authority (PoA), де валідатори обираються на основі їх авторитету; Proof of Elapsed Time (PoET), який використовує випадковий час очікування для визначення лідера блоку; Proof of Space (PoSpace) і Proof of Capacity (PoC), що залучають вільне місце на диску для досягнення консенсусу з меншими енергетичними витратами. Проте ці алгоритми не набули широкого використання, оскільки використовуються переважно в специфічних умовах.

1.3. Інноваційні підходи у технологіях електронного голосування

Електронне голосування є одним із ключових напрямів цифровізації суспільства, який дозволяє підвищити зручність та доступність виборчих процесів. Традиційні електронні системи, хоч і широко використовуються, стикаються з численними викликами, зокрема щодо забезпечення прозорості, безпеки та захищеності від маніпуляцій. Сучасні інноваційні підходи у цій сфері спрямовані на вирішення цих проблем за рахунок використання новітніх технологій.

Одним із таких підходів є застосування блокчейн-технології, яка забезпечує децентралізоване зберігання даних і гарантує незмінність записів. У системах голосування на основі блокчейну кожен голос реєструється як транзакція, яка

зберігається у ланцюжку блоків. Це дозволяє кожному учаснику перевірити правильність підрахунку голосів, зберігаючи при цьому конфіденційність виборців. Крім того, блокчейн унеможливлює зміну результатів голосування без відома більшості учасників, що підвищує рівень довіри до системи.

Іншим важливим напрямом є використання криптографічних методів для забезпечення безпеки та анонімності виборців. Зокрема, технології шифрування даних і цифрового підпису дозволяють гарантувати, що голос може бути зарахований лише від авторизованого виборця, а його зміст залишається недоступним для третіх сторін. Ці методи також забезпечують захист від дублювання голосів або спроб втручання у виборчий процес.

Застосування інноваційних підходів дозволяє вирішити ключові проблеми, притаманні традиційним електронним системам голосування. Проте впровадження цих технологій вимагає подолання ряду викликів, таких як висока вартість розробки, необхідність адаптації до юридичних норм та забезпечення масштабованості системи. Незважаючи на ці труднощі, інноваційні рішення відкривають нові можливості для створення прозорих, безпечних та доступних виборчих процесів.

1.4. Огляд та аналіз існуючих рішень для організації голосувань

Системи голосування постійно еволюціонують, охоплюючи як традиційні офлайн та онлайн централізовані моделі, так і новітні децентралізовані рішення. Вибір підходу залежить від цілей, масштабу, вимог до безпеки та ступеня довіри між учасниками процесу.

Одним із найпоширеніших підходів є централізовані офлайн-системи голосування, які використовуються на виборчих дільницях. Вони базуються на спеціалізованому апаратному та програмному забезпеченні (електронні урни, сканери бюлетенів), контрольованому єдиним оператором або групою довірених органів. Такі системи забезпечують високу швидкість обробки голосів і зручність для виборчих комісій. Проте вони мають суттєві недоліки: обмежена прозорість для виборців, потреба в повній довірі до адміністратора системи, ризики технічних

збоїв або втручання в ланцюг постачання обладнання.

Іншим різновидом є централізовані онлайн-системи голосування, які забезпечують дистанційну участь виборців через інтернет. Вони реалізуються як веб- або мобільні застосунки і особливо корисні в умовах пандемій чи для виборців за кордоном. Попри зручність, ці системи мають свої виклики: захист від кібератак, гарантія цілісності та конфіденційності голосів, запобігання повторному голосуванню, а також складність незалежної верифікації результатів.

У зв'язку з викликами, притаманними централізованим рішенням, зростає зацікавленість у децентралізованих системах голосування, зокрема на основі блокчейн-технологій. У таких системах дані про голоси зберігаються в розподіленому реєстрі, змінити який без консенсусу учасників мережі практично неможливо. Платформи на кшталт Voatz декларують використання блокчейну для досягнення прозорості, незмінності та довіри [7]. Водночас Voatz критикували за закритість вихідного коду та виявлені вразливості [8], що підкреслює важливість відкритості й незалежного аудиту в таких рішеннях.

Ще одним прикладом масштабного впровадження електронного голосування є система i-Voting в Естонії [9], яка з 2017 року дає громадянам змогу голосувати онлайн із використанням електронного підпису. Хоча ця система не базується на блокчейні, вона спирається на перевірену інфраструктуру електронної ідентифікації та шифрування й продемонструвала на практиці свою ефективність у загальнодержавному масштабі.

Для кращого розуміння відмінностей між централізованими та децентралізованими підходами у сфері електронного голосування, у таблиці 1.2 наведено порівняння ключових характеристик обох типів систем за основними критеріями, що впливають на їхню ефективність, надійність та придатність до практичного використання. Оскільки існують різні реалізації блокчейнів, кожен з яких має свої характеристики, показники децентралізованих систем можуть суттєво варіюватися в залежності від вибраної платформи.

Таблиця 1.2

Порівняння централізованих та децентралізованих систем голосування

Критерій	Централізовані системи	Децентралізовані системи
Прозорість	Низька	Висока
Безпека	Середня	Висока
Масштабованість	Висока	Залежить від реалізації
Енергоспоживання	Низьке	Залежить від реалізації
Швидкість	Висока	Залежить від реалізації

Аналіз демонструє, що централізовані рішення переважають у зрілості та практичності, проте обмежені в прозорості та довірі. Натомість децентралізовані рішення пропонують привабливу альтернативу, що вимагає вирішення проблем масштабованості, складності впровадження, зручності для користувачів та юридичного регулювання [10]. Зважаючи на ці фактори, дослідження в напрямку побудови блокчейн-орієнтованих систем електронного голосування є актуальним і перспективним.

1.5. Висновки до розділу

У цьому розділі було розглянуто основні аспекти технології блокчейн, методи досягнення консенсусу у розподілених системах, інноваційні підходи до електронного голосування та існуючі рішення для організації виборчих процесів. Аналіз існуючих рішень продемонстрував, що децентралізовані платформи на основі блокчейн-технології мають значний потенціал для вирішення ключових проблем електронного голосування. Подальший розвиток цих технологій, спрямований на подолання технічних і організаційних бар'єрів, дозволить створити прозорі, безпечні та доступні системи голосування, які відповідають сучасним вимогам.

РОЗДІЛ 2. ПОСТАНОВКА ЗАВДАННЯ РОЗРОБЛЕННЯ ЗАСТОСУНКУ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

2.1. Мета та завдання розробки

Метою роботи є створення функціональної та надійної системи електронного голосування, яка забезпечить підвищення прозорості, безпеки та довіри до процесу волевиявлення шляхом застосування технології блокчейн. Розроблена система має вирішити існуючі проблеми централізованих рішень, зокрема залежність від операторів, потенційний ризик маніпуляцій з даними голосування та необхідність забезпечення високого рівня безпеки.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Розробити деталізовану архітектуру системи електронного голосування, визначивши ключові модулі, їхню взаємодію та відповідальність. Архітектурне рішення має враховувати вимоги до безпеки, масштабованості та зручності використання.
2. Реалізувати програмну частину системи, включаючи смарт-контракти на блокчейн-платформі, а також клієнтський вебінтерфейс для взаємодії користувачів із системою.
3. Забезпечити інтеграцію з криптовалютним гаманцем для безпечної аутентифікації користувачів та підписання транзакцій голосування.
4. Створити та провести тестування розробленої системи на відповідність визначеним функціональним та нефункціональним вимогам, включаючи перевірку безпеки, продуктивності та зручності використання.
5. Підготувати технічну документацію, що описує процес розробки, архітектуру системи, інструкції з використання та результати тестування.

2.2. Характеристика об'єкту проєктування

Об'єктом проєктування є децентралізована система електронного голосування, що функціонує на основі блокчейн-технології.

Вхідними даними для розробленого програмного забезпечення є:

- Реєстраційна інформація користувачів (ідентифікатори, публічні ключі гаманців), що отримується з підключеного гаманця. Формат представлення – дані, що передаються через вебформи та API гаманця.
- Параметри голосування, що задаються організатором через вебінтерфейс (назва, опис, перелік варіантів вибору, тривалість голосування). Формат представлення – структуровані дані, що передаються через вебформи.
- Подані голоси користувачів, що вибираються ними через вебінтерфейс та підписуються за допомогою їхніх приватних ключів. Формат представлення – ідентифікатор обраного варіанту та криптографічний підпис транзакції.

Вихідними даними розробленого програмного забезпечення є:

- Захищені та незмінні записи про проведені голосування та подані голоси, що зберігаються у блокчейні. Формат представлення – транзакції та дані у відповідних облікових записах блокчейну.
- Результати голосування, що агрегуються на основі даних з блокчейну та відображаються користувачам через вебінтерфейс у текстовому та графічному форматах.
- Сповіщення про статус голосу (успішно подано, помилка), що відображаються користувачам у вебінтерфейсі. Формат представлення – текстові повідомлення та візуальні елементи інтерфейсу.

2.3. Обґрунтування вибору технологій та методів

Як видно з таблиці 1.2, децентралізовані системи голосування мають значні переваги порівняно з централізованими, зокрема високу прозорість та безпеку. Саме тому для розробки системи обрано блокчейн як технологічну основу. Однак для ефективного використання цих переваг необхідно обрати блокчейн-платформу,

яка забезпечує високу масштабованість, низьке енергоспоживання та швидкість обробки транзакцій. Саме тому для реалізації системи було обрано платформу Solana, яка завдяки інноваційному механізму консенсусу PoH демонструє високу продуктивність і стабільну роботу навіть за великого навантаження, що є критично важливим для системи голосування з великою кількістю учасників. Solana здатна обробляти до 65,000 транзакції на секунду (TPS) [11], що значно перевищує показники багатьох інших блокчейн-платформ.

Мову програмування для розробки смарт-контрактів було обрано Rust. Однією з основних причин вибору Rust є його висока продуктивність та безпека. Rust дозволяє контролювати управління пам'яттю без використання збирача сміття, що дозволяє досягати кращої ефективності та передбачуваності в порівнянні з іншими мовами. Мова забезпечує гарантії безпеки пам'яті через систему власності і запозичення, що дозволяє уникати багатьох типових помилок, таких як витоки пам'яті або доступ до неініціалізованої пам'яті, зберігаючи високу продуктивність. Такий підхід є особливо важливим при розробці смарт-контрактів для блокчейн-платформи, де потрібно обробляти великі обсяги транзакцій і зберігати ефективність виконання програм при обмежених ресурсах. Завдяки цьому Rust є ідеальним вибором для розробки смарт-контрактів на Solana.

Для спрощення процесу розробки смарт-контрактів на Solana було обрано використання бібліотеки Anchor. Anchor - це фреймворк для розробки смарт-контрактів, який забезпечує більш високий рівень абстракції і значно спрощує роботу з Solana, дозволяючи швидше і безпечніше створювати смарт-контракти. Anchor також допомагає в автоматизації багатьох аспектів розробки, таких як перевірка транзакцій, підписання і виконання, що дозволяє зосередитися на бізнес-логіці, а не на низькорівневих деталях взаємодії з блокчейном.

Важливою частиною застосунку є інтеграція з криптовалютним гаманцем для забезпечення безпечного доступу користувачів до системи. Вибір Phantom Wallet зумовлений не лише його популярністю серед користувачів Solana, а й зручністю для розробників. Phantom надає простий інтерфейс для взаємодії зі смарт-

контрактами, що значно спрощує інтеграцію з блокчейном і пришвидшує розробку. Він також підтримує безпечне зберігання ключів і легке підтвердження транзакцій, що робить його оптимальним вибором для системи голосування.

Для клієнтської частини застосунку обрано React, що дозволяє створювати зручні та інтерактивні вебзастосунки. Завдяки своїй популярності і широкій підтримці бібліотек, React забезпечує гнучкість і масштабованість, необхідні для розробки інтерфейсу користувача для системи голосувань.

2.4. Аналіз вимог

На етапі аналізу вимог було проведено ретельне дослідження предметної області електронного голосування. Це дозволило визначити ключові потреби основних зацікавлених сторін: організаторів голосувань та учасників (виборців). Метою аналізу було формування чіткого переліку функціональних та нефункціональних вимог до розроблюваної системи, які стали підґрунтям для подальшої детальної специфікації.

Організатори голосувань мають потребу у зручному інструменті для створення та гнучкого налаштування голосувань, включаючи визначення назви, опису, варіантів вибору та терміну дії. Вони також зацікавлені у можливості контролювати процес голосування та, за необхідності (з урахуванням анонімності), переглядати проміжну статистику. Ключовою вимогою є забезпечення прозорості та незмінності результатів для зміцнення довіри до процесу волевиявлення, а також надання простого способу оголошення підсумків після завершення голосування. У перспективі, за потреби, можлива інтеграція системи з іншими платформами або системами обліку користувачів.

Учасники голосувань мають потребу в інтуїтивно зрозумілому інтерфейсі для зручної участі в голосуваннях. Для них критично важливим є забезпечення анонімності їхнього волевиявлення та захист від розкриття зробленого вибору. Виборці також зацікавлені у можливості ознайомлення з результатами голосування після його завершення. Безпека системи та неможливість фальсифікації результатів є

для них першочерговими вимогами, так само як і зручний спосіб ідентифікації та автентифікації для участі у голосуванні.

2.5. Специфікація вимог

Призначенням застосунку є забезпечення прозорого, безпечного та незмінного процесу голосування завдяки використанню технології блокчейну. Він дозволяє організаторам ініціювати голосування, визначати варіанти вибору та переглядати результати, а учасникам безпечно віддавати свої голоси.

2.5.1. Загальний опис

2.5.1.1. Характеристики продукту

Розроблена система електронного голосування надає наступні основні можливості:

- Організатори можуть ініціювати голосування, визначаючи назву, опис, перелік варіантів вибору та термін дії.
- Зареєстровані учасники можуть безпечно віддавати свої голоси за один із запропонованих варіантів протягом визначеного терміну.
- Після завершення голосування всі користувачі можуть переглядати підсумки голосування у наочному форматі.

2.5.1.2. Класи користувачів та їх характеристики

Користувачі системи поділяються на два класи:

- Організатор голосування – користувач, який має право створювати, налаштовувати та контролювати параметри голосування (назва, опис, варіанти, час завершення). Організатор також може брати участь у створеному ним голосуванні як звичайний виборець.
- Учасник голосування (виборець) – користувач, який має право брати участь в активних голосуваннях, віддаючи свій голос за один із варіантів. Для участі виборець повинен мати встановлений та налаштований криптовалютний гаманець.

Як організатори, так і учасники голосування є пріоритетним класом користу-

вачів, оскільки без їхньої взаємодії функціонування системи є неможливим.

2.5.1.3. Середовище функціонування

Програмний продукт функціонує у децентралізованому середовищі блокчейну Solana. Смарт-контракти, що реалізують основну логіку голосування, виконуються в мережі Solana. Клієнтська частина системи розроблена як вебзастосунок, що працює у сучасних браузерах та взаємодіє зі смарт-контрактами через Remote Procedure Call (RPC)-сервіси Solana. Для взаємодії з користувачами використовується криптовалютний гаманець.

2.5.2. Характеристики системи

2.5.2.1. Створення голосування

Опис: Організатор може створювати голосування з заданими параметрами.

Пріоритет: Високий.

Послідовність дія/відгук:

1. Організатор входить у застосунок.
2. Вибирає опцію створення голосування.
3. Вводить параметри голосування (назва, опис, варіанти, час до завершення).
4. Підтверджує створення голосування.
5. Система перевіряє дані та публікує голосування у блокчейні.
6. Система повідомляє організатора про успішне створення.

Функціональні вимоги:

REQ-1.1: Система має забезпечувати форму для параметрів голосування.

REQ-1.2: Система перевіряє коректність введених даних.

REQ-1.3: Система інтегрується зі смарт-контрактом для запису даних у блокчейн.

2.5.2.2. Участь у голосуванні

Опис: Виборець може подати свій голос за один із варіантів голосування.

Пріоритет: Високий.

Послідовність дія/відгук:

1. Виборець входить у систему.
2. Вибирає активне голосування.

3. Обирає варіант голосування.
4. Підтверджує вибір.
5. Система перевіряє валідність голосу (уникнення повторного голосування).
6. Система записує голос у блокчейн.
7. Виборець отримує підтвердження успішного голосування.

Функціональні вимоги:

REQ-2.1: Система має дозволяти вибір варіанту голосування.

REQ-2.2: Система перевіряє валідність голосу перед його подачею.

REQ-2.3: Система записує голос через смарт-контракт у блокчейн.

2.5.2.3. Перегляд результатів голосування

Опис: Учасники можуть переглядати результати після завершення голосування.

Пріоритет: Середній.

Послідовність дія/відгук:

1. Користувач входить у систему.
2. Вибирає завершене голосування.
3. Вибирає опцію перегляду результатів.
4. Система зчитує дані з блокчейну.
5. Відображає результати голосування у графічному вигляді.

Функціональні вимоги:

REQ-3.1: Система має дозволяти доступ до завершених голосувань.

REQ-3.2: Система інтегрується зі смарт-контрактом для отримання даних.

REQ-3.3: Система візуалізує результати у зручному форматі.

2.5.3. Вимоги зовнішніх інтерфейсів

2.5.3.1. Користувацькі інтерфейси

Користувач взаємодіє із системою через вебінтерфейс, що включає:

- Екран створення голосування (форма з введенням параметрів голосування).
- Екран участі у голосуванні (вибір варіанта і підтвердження).
- Екран перегляду результатів (графічна візуалізація результатів голосуван-

ня).

- Екран авторизації за допомогою Phantom Wallet для підтвердження особи користувача перед участю у голосуванні.
- Екран вибору кластеру мережі Solana.
- Екран управління акаунтом у мережі Solana.

2.5.3.2. Програмні інтерфейси

Система взаємодіє з наступними програмними компонентами:

- Блокчейн Solana — через RPC-сервіси для запису та зчитування даних смарт-контрактів.
- Криптовалютний гаманець Phantom Wallet — через його API для аутентифікації користувачів та підписання транзакцій.
- Браузери (сучасні версії Chrome, Firefox, Safari, Edge, тощо).
- Бібліотека Anchor для спрощення взаємодії зі смарт-контрактами Solana.
- Бібліотека React для розробки клієнтського вебінтерфейсу.

2.5.3.3. Комунікаційні інтерфейси

Вебзастосунок використовує стандартні протоколи Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS) для взаємодії з RPC-сервісами Solana. Для взаємодії з гаманцем Phantom Wallet використовуються специфічні методи, передбачені його API. Безпека комунікацій забезпечується використанням протоколу HTTPS.

2.5.4. Інші нефункційні вимоги

2.5.4.1. Вимоги продуктивності

- Час підтвердження голосу не повинен перевищувати 3 секунд (Не враховуючи час на взаємодію з гаманцем Phantom Wallet).
- Час відображення результатів голосування не повинен перевищувати 5 секунд.
- Система повинна підтримувати одночасну участь 1000 користувачів без значного зниження продуктивності (час відгуку на дії користувача не повинен збільшуватися більше ніж на 20

2.5.4.2. Вимоги безпеки

- Доступ до функціональності створення та редагування голосувань повинен бути обмежений лише авторизованими організаторами.
- Система повинна гарантувати, що кожен користувач може проголосувати лише один раз в межах одного голосування.
- Голоси, подані користувачами, повинні бути перевірені на валідність перед записом у блокчейн.

2.5.4.3. Атрибути якості програмного продукту

- Кодова база повинна мати модульну та читабельну структуру для полегшення подальшої розробки, тестування та підтримки.
- Система повинна бути стійкою до тимчасових втрат з'єднання з блокчейном та автоматично відновлювати свою роботу після відновлення з'єднання.
- Основна функціональність системи повинна бути покрита автоматизованими тестами (юніт-тести, інтеграційні тести) для забезпечення відповідності вимогам та виявлення потенційних помилок.
- Інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним у використанні для користувачів з різним рівнем технічної підготовки.

2.6. Висновки до розділу

У цьому розділі було сформульовано мету та завдання розробки, а також обґрунтовано вибір технологій, алгоритмів і інструментів, які використовуються для реалізації системи голосування. Проаналізовано вимоги до системи, включаючи функціональні та нефункціональні аспекти, що дозволило визначити ключові параметри, необхідні для забезпечення її надійності, безпеки та зручності використання та сформувати специфікацію вимог.

РОЗДІЛ 3. ПРОЄКТУВАННЯ ДЕЦЕНТРАЛІЗОВАНОГО ЗАСТОСУНКУ ДЛЯ ПРОВЕДЕННЯ ГОЛОСУВАНЬ

3.1. Проєктування загальної архітектури застосунку

Архітектура застосунку для проведення голосувань базується на принципах децентралізації, прозорості та безпеки, що є фундаментальними вимогами для сучасних електронних систем голосування. Застосунок складається з чотирьох основних компонентів: клієнтської частини, гаманця Phantom Wallet, смарт-контракту та блокчейн-платформи Solana. Кожен з них виконує окремі функції у процесі голосування, забезпечуючи ефективну, безпечну та прозору взаємодію між користувачем і блокчейном. Схематичне відображення взаємодії між компонентами представлено на діаграмі компонентів на рисунку 3.1.

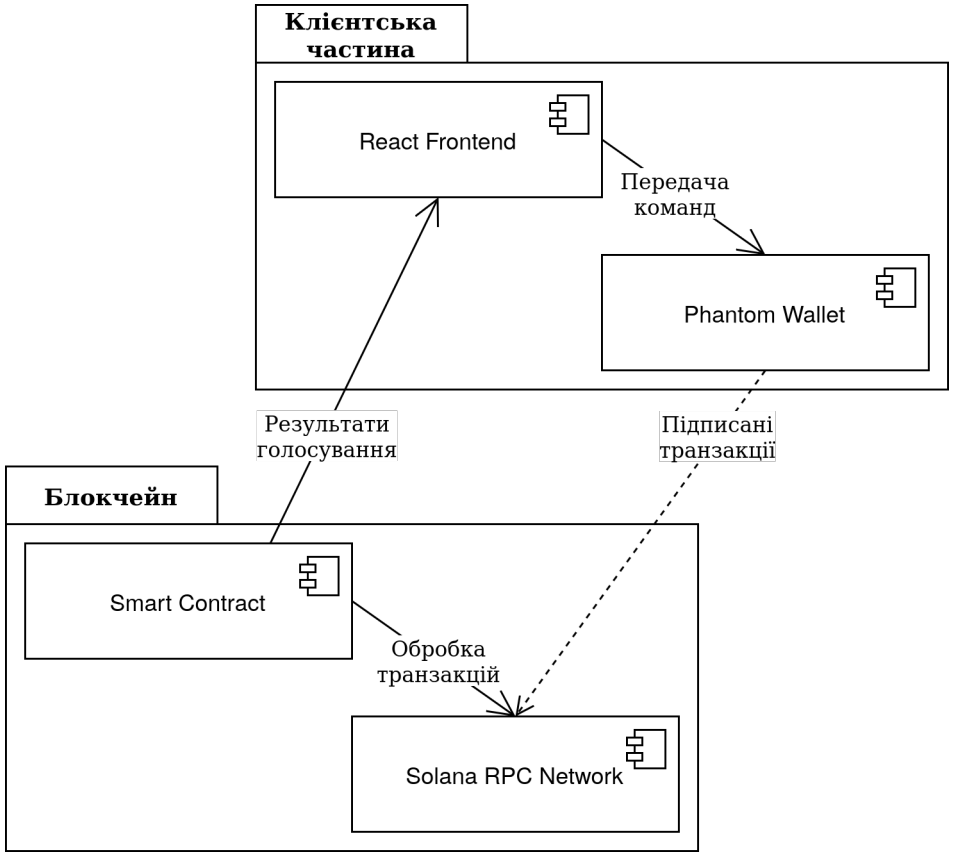

Рис. 3.1 Діаграма компонентів

Клієнт є основним інтерфейсом взаємодії користувача із системою. Реалізована за допомогою фреймворку React, вона забезпечує доступ до функціоналу як для організаторів, так і для учасників голосування. Організатори можуть створювати нові голосування, встановлювати варіанти відповідей і визначати дедлайни. Учасники мають змогу переглядати доступні голосування, обирати варіанти та відстежувати результати. Кожна дія користувача (створення голосування, голосування, перегляд результатів) ініціює транзакцію, яка формується у клієнті, підписується через Phantom Wallet та надсилається до смарт-контракту. Таким чином, клієнт виступає посередником між користувачем і блокчейн-інфраструктурою, забезпечуючи зручність та безпеку взаємодії.

Phantom Wallet — це браузерне розширення, яке забезпечує безпечне зберігання приватних ключів користувача та підпис транзакцій. Його використання є критично важливим для автентифікації та авторизації дій у системі. Під час голосування або створення нового голосування клієнтська частина формує транзакцію, яку користувач повинен підписати своїм приватним ключем через Phantom Wallet. Це гарантує, що жодна дія не може бути виконана від імені користувача без його згоди. Гаманець також забезпечує взаємодію з мережею Solana, виступаючи у ролі шлюзу між користувачем і блокчейном.

Смарт-контракт є логічним ядром системи голосування, розгорнутим у мережі Solana. Він реалізує повний життєвий цикл голосування: створення опитування, реєстрацію голосів, перевірку права голосу, виключення можливості повторного голосування, підрахунок голосів та збереження результатів. Смарт-контракт є детермінованим і не залежить від зовнішніх сервісів, що гарантує чесність та прозорість виконання логіки. Всі записи зберігаються безпосередньо в блокчейні, що унеможливлює їх зміну або видалення. Це забезпечує довіру до системи і дозволяє будь-кому перевірити достовірність результатів голосування.

У системі голосування блокчейн Solana виконує роль середовища для зберігання та обробки транзакцій. Кожна дія користувача — створення голосування, голос чи перегляд результатів — перетворюється на транзакцію, яка надсилається

до мережі Solana. Ці транзакції перевіряються, обробляються та фіксуються у блокчейні, що гарантує їхню незмінність і публічну доступність. Solana забезпечує високу швидкість обробки та масштабованість, що дозволяє системі працювати ефективно навіть при великій кількості користувачів.

Узгоджена взаємодія клієнта, гаманця Phantom, смарт-контракту та блокчейну Solana формує цілісну архітектуру, що забезпечує безпечний, прозорий і незмінний процес голосування. Такий підхід дозволяє реалізувати децентралізовану систему, якій можуть довіряти всі учасники, незалежно від їх ролі, забезпечуючи високий рівень надійності та масштабованості.

3.2. Проєктування функціональних модулів клієнтської частини

Функціональність клієнтської частини застосунку організована навколо шести ключових модулів, які забезпечують повний цикл взаємодії користувачів із системою електронного голосування, починаючи від перегляду інформації про власний обліковий запис до безпосередньої участі у голосуваннях та ознайомлення з їхніми результатами. Ці модулі побудовані таким чином, щоб забезпечити інтуїтивно зрозумілий процес для різних категорій користувачів, використовуючи при цьому можливості блокчейн-платформи Solana для гарантування безпеки та прозорості. Взаємодія між користувачами різних ролей та функціональними можливостями системи наочно представлена на діаграмі прецедентів, відображеній на рисунку 3.2.

Діаграма прецедентів ілюструє, як дві основні ролі користувачів – "Виборець"та "Організатор"– взаємодіють із застосунком. "Виборець"має можливість переглядати інформацію про свій обліковий запис, включаючи баланс та історію транзакцій, обирати кластер мережі Solana для роботи, переглядати результати вже завершених голосувань, додавати власну адресу кластеру та, що є його основною функцією, брати участь у голосуваннях шляхом вибору одного з доступних варіантів. Процес голосування є складною дією, яка включає попередній вибір варіанту.

Роль "Організатор"об’єднує в собі всі можливості "Виборця"та додає ключову функцію створення нових голосувань. Цей процес передбачає введення необхідних даних, таких як назва голосування, опис, перелік варіантів вибору та тривалість. Таким чином, діаграма прецедентів демонструє розподіл функціональних можливостей між різними ролями користувачів, підкреслюючи основні сценарії використання системи електронного голосування.

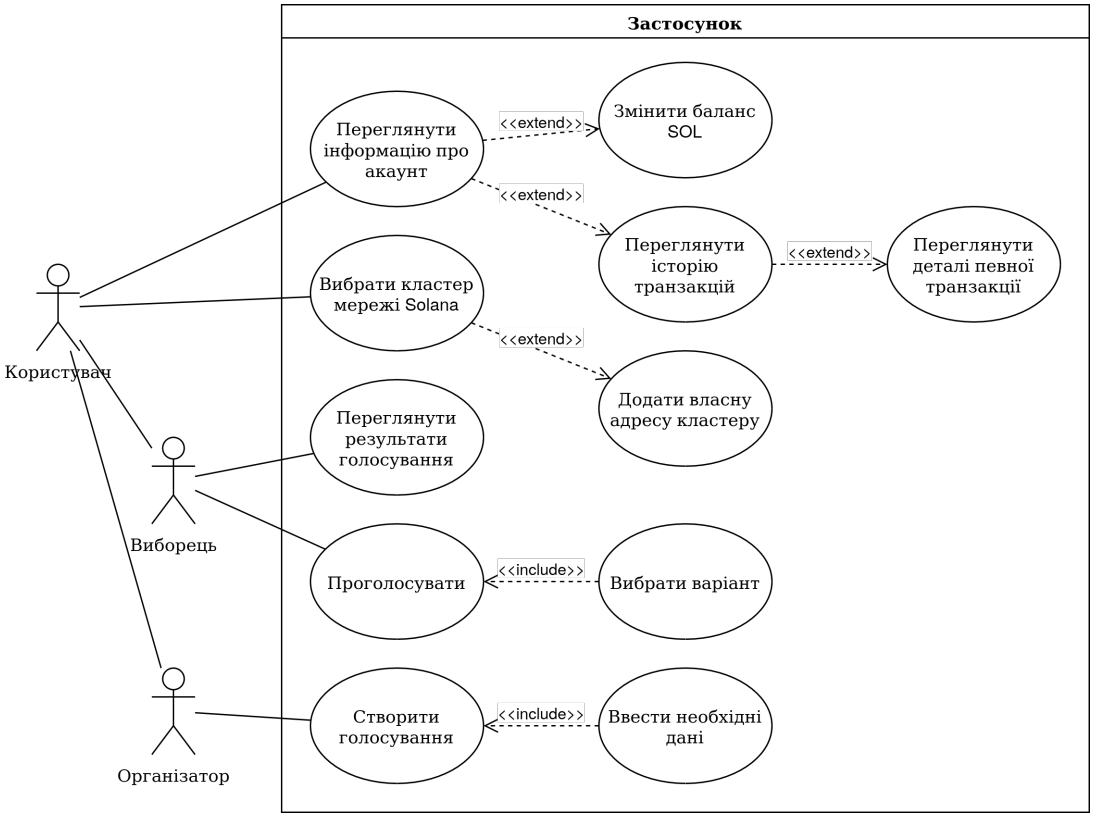

Рис. 3.2 Діаграма прецедентів

3.3. Проєктування інтерфейсу користувача та його елементів

Інтерфейс клієнтської частини застосунку розроблено з фокусом на зручність та інтуїтивність використання, щоб забезпечити ефективну взаємодію користувачів на всіх етапах роботи з системою електронного голосування.

На рисунку А.1 зображено головну сторінку застосунку, де відображаються доступні голосування. У верхній частині сторінки розміщено заголовок, та кнопки для вибору між управлінням акаунтом, вибором кластеру та застосунком для голосувань. Нижче розташована коротка інструкція користувача для створе-

ння голосування та інформація про обліковий запис користувача, зокрема його ідентифікатор, що підкреслює важливість автентифікації у системі голосування. Центральне місце займає кнопка створення, яка є ключовим елементом для ініціювання процесу створення нового голосування організатором. Важливою частиною цієї сторінки є список доступних голосувань, представлених за їхніми назвами. Користувачі можуть переглядати цей список, щоб знаходити потрібні їм голосування. При натисканні на назву конкретного голосування відкривається діалогове вікно, що відображає детальну інформацію про це голосування та надає можливість взяти в ньому участь. Дизайн сторінки мінімалістичний, з акцентом на простоті та функціональності, щоб користувачі могли легко орієнтуватися та швидко знаходити потрібні опції.

На рисунку А.2 представлено форму для створення нового голосування. Для того, щоб потрапити на цю сторінку, користувач повинен натиснути кнопку "Create Poll"на головній сторінці застосунку. У верхній частині форми розташовані поля для введення назви голосування та його опису, що дозволяє організатору надати контекст та інформацію про предмет голосування. Нижче знаходиться поле для визначення тривалості голосування, де організатор вказує час, протягом якого голосування буде активним. Тривалість голосування може бути вказана у хвилинах, годинах та днях, що забезпечує гнучкість у плануванні голосування. Далі розташовані поля для введення кандидатів або варіантів вибору та кнопка для додавання нових варіантів. У нижній частині форми знаходяться дві кнопки: для підтвердження створення голосування та для закриття форми. Дизайн форми інтуїтивно зрозумілий, з чіткими підписами до полів введення, що полегшує процес заповнення.

На рисунку А.3 зображено форму, призначену для подачі голосу учасником голосування. У верхній частині форми відображається назва та опис голосування. Нижче розміщено таймер, який спливає в реальному часі та показує час, що залишився до завершення голосування, що створює відчуття терміновості та стимулює учасників до активності. Після завершення таймера проголосувати більше не мо-

жна. Далі представлені варіанти голосування з інформацією про кількість поданих голосів за кожен варіант. Праворуч від кожного варіанту розташована кнопка для подачі голосу. У нижній частині форми знаходиться кнопка для закриття форми. Якщо голосування переглядає організатор, йому також доступна кнопка для зміни параметрів голосування, а саме можливість збільшення тривалості часу до завершення. Дизайн форми зосереджений на простоті та зручності використання, з чітким відображенням варіантів вибору та таймером, що дозволяє учасникам швидко і легко зробити свій вибір.

На рисунку А.4 представлено вікно з результатами голосування. У верхній частині вікна розміщений заголовок, який одразу інформує користувача про призначення цього екрана. Нижче наведено варіанти відповідей. Для кожного варіанту відображається кількість отриманих голосів та їхній відсоток від загальної кількості. Візуально кількість голосів представлена горизонтальними смугами, довжина яких пропорційна кількості відданих голосів, що полегшує швидке порівняння результатів. Під результатами голосування підсумовується загальна кількість поданих голосів та вказано лідера голосування. У нижній частині вікна розміщено кнопку, яка дозволяє користувачеві закрити вікно з результатами та повернутися до попереднього екрана. Дизайн цього вікна лаконічний та інформативний, основний акцент зроблено на чіткому відображенні результатів голосування.

На рисунку А.5 відображено інтерфейс для управління кластерами Solana. У верхній частині сторінки знаходиться заголовок та коротка інструкція користувача, що пояснює призначення сторінки – управління та вибір кластерів Solana. Нижче розташована кнопка, що дозволяє користувачам додавати нові кластери. Основну частину сторінки займає таблиця з переліком доступних кластерів, де для кожного кластера відображаються його назва, мережа та кінцева точка. Кластери Solana – це різні мережі, на яких розгортаються та функціонують програми Solana. Дизайн сторінки структурований, з використанням таблиці для зручного представлення інформації про кластери.

На рисунку А.6 представлено інтерфейс, що відображає інформацію про облі-

ковий запис користувача. У верхній частині сторінки розташовані основні функції, пов'язані з управлінням обліковим записом, такі як отримання та передача криптовалюти. Нижче знаходиться інформація про історію транзакцій. Список транзакцій дозволяє користувачам відстежувати всі операції, здійснені з їхнім обліковим записом. Деталі кожної транзакції, такі як хеш, час, статус, можна переглянути на сторонньому сервісі (наприклад, Solana Explorer), що забезпечує прозорість та можливість перевірки операцій. Дизайн сторінки функціональний, з чітким розділенням інформації на блоки.

Таким чином, представлені візуалізації ключових екранів клієнтської частини застосунку демонструють інтуїтивно зрозумілий та функціональний дизайн. Кожен інтерфейс спроєктовано з урахуванням потреб користувачів на різних етапах роботи із системою голосування: від створення опитування та участі в ньому до перегляду результатів та управління власним обліковим записом і вибором кластерів Solana. Простота навігації, чітке представлення інформації та логічне розташування елементів керування сприяють зручній та ефективній взаємодії користувача з застосунком.

3.4. Проєктування смарт-контракту

Смарт-контракт являє собою самовиконуваний код, що зберігається в блокчейні та автоматично виконує умови, прописані в ньому. У контексті цього застосунку, смарт-контракт відповідає за створення опитувань, реєстрацію голосів та забезпечення чесності процесу голосування. Інструкція в смарт-контракті – це дія, яку користувач може ініціювати для взаємодії з контрактом. Кожна інструкція виконує певну функцію, наприклад, створення нового опитування або подача голосу. Транзакція – це підписане повідомлення, яке користувач надсилає в мережу блокчейн для виконання певної інструкції смарт-контракту. Транзакція включає в себе дані інструкції та підпис користувача, що підтверджує його автентичність.

У блокчейні Solana для зберігання даних використовується акаунти. Існують два основних типи акаунтів: акаунти, що належать системній програмі, які ви-

користовуються для базових операцій, таких як переказ коштів або виділення місця для даних, та акаунти, що належать програмам, які використовуються для зберігання стану смарт-контрактів та пов'язаних з ними даних.

Для функціонування смарт-контракту застосунку для проведення голосувань необхідні наступні акаунти:

- Акаунт підписувача – це акаунт користувача, який ініціює транзакцію. Він потрібен для оплати комісій за транзакції та підписання інструкцій.
- Акаунт опитування – зберігає інформацію про конкретне опитування, включаючи його назву, опис, час завершення, список кандидатів та кількість голосів за кожного кандидата. Оскільки кожне опитування є унікальним, для кожного нового опитування створюється окремий акаунт опитування.
- Акаунт виборця – відіграє ключову роль у запобіганні повторному голосуванню. Коли користувач вперше голосує в певному опитуванні, для нього створюється унікальний акаунт виборця, пов'язаний як з самим опитуванням, так і з публічним ключем користувача. Коли той самий користувач спробує проголосувати в цьому ж опитуванні вдруге, смарт-контракт перевірить, чи існує акаунт виборця, пов'язаний з його публічним ключем та цим опитуванням. Якщо такий акаунт вже існує, це означає, що користувач вже віддав свій голос, і транзакція на голосування буде відхилена.

Архітектурно, смарт-контракт розроблений таким чином, щоб забезпечити чітке розділення між логікою створення опитування та логікою голосування. Взаємодія з блокчейном відбувається через надсилання підписаних транзакцій. Клієнт створює транзакцію, яка містить необхідну інструкцію та дані, підписує її своїм приватним ключем і надсилає в мережу Solana. Мережа верифікує підпис та виконує інструкцію смарт-контракту, оновлюючи стан відповідних акаунтів.

Для створення опитування клієнт ініціює транзакцію, яка викликає інструкцію у смарт-контракті. У цій транзакції клієнт передає назву опитування, його опис, час завершення та список кандидатів. Смарт-контракт перевіряє ці дані на валідність та створює новий акаунт опитування, зберігаючи в ньому передану ін-

формацію. Адреса цього акаунта опитування детерміновано генерується на основі назви опитування, що дозволяє унікально ідентифікувати кожне опитування.

Для голосування клієнт ініціює транзакцію, яка викликає інструкцію vote у смарт-контракті. У цій транзакції клієнт передає назву опитування та ім'я кандидата, за якого він хоче проголосувати. Смарт-контракт перевіряє, чи існує опитування з такою назвою, чи не закінчився час голосування, чи існує кандидат з вказаним ім'ям, і чи не голосував вже даний користувач у цьому опитуванні (перевіряючи наявність відповідного акаунта виборця). Якщо всі перевірки проходять успішно, смарт-контракт збільшує лічильник голосів для обраного кандидата в акаунті опитування та створює або оновлює акаунт виборця, фіксуючи факт голосування цього користувача в даному опитуванні.

Смарт-контракт, у свою чергу, зберігає в блокчейні стан опитувань (назва, опис, час завершення, кандидати, кількість голосів) та інформацію про те, хто і в якому опитуванні вже проголосував. Усі ці дані є публічними та незмінними після запису в блокчейн, що забезпечує прозорість та чесність процесу голосування.

Схематично роботу смарт-контракту зображено на діаграмі послідовності на рисунку 3.3,

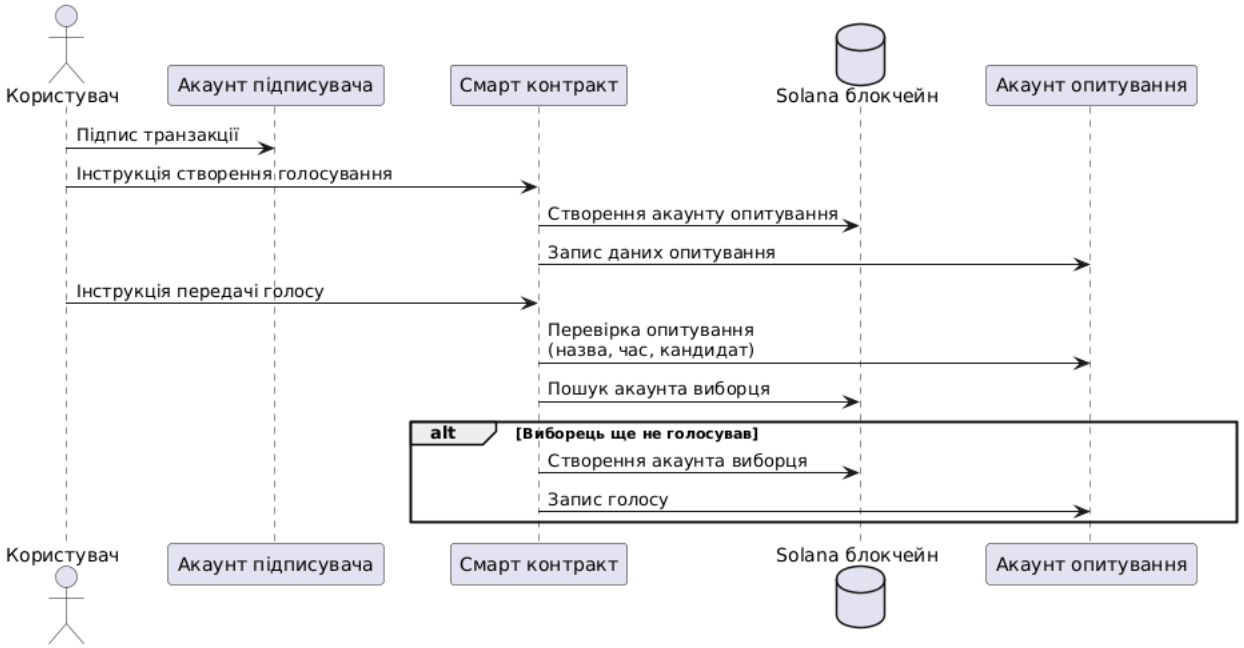

Рис. 3.3 Діаграма послідовності

3.5. Проєктування безпеки та конфіденційності

Забезпечення безпеки та конфіденційності є фундаментальним елементом архітектури застосунку для голосувань, оскільки саме ці аспекти формують базову довіру користувачів до легітимності та незворотності результатів. У рамках розробки системи було впроваджено багатошарову модель захисту, яка використовує передові методи криптографії, протоколи шифрування та алгоритмічні механізми валідації даних.

Для унеможливлення повторного голосування застосовується система валідації унікальності транзакцій. Кожен голос супроводжується унікальним криптографічним маркером, що генерується на основі комбінації відкритого ключа користувача, даних про голосування та випадкового доповнення. Смарт-контракт на рівні Solana Runtime Environment аналізує ці маркери і здійснює детерміновану валідацію автентичності кожного голосу у режимі реального часу.

Щодо перевірки валідності голосу, перед його зарахуванням виконується перевірка підпису транзакції приватним ключем користувача. Додатково, перед надсиланням транзакції у мережу, відбувається локальна верифікація за допомогою Phantom Wallet Software Development Kit (SDK), що гарантує збереження приватного ключа у зашифрованому середовищі пристрою користувача без жодної можливості його витоку в мережу.

Анонімність учасників забезпечується шляхом використання псевдонімізації, де кожному користувачеві призначається унікальний криптографічний ідентифікатор. Ідентифікатор створюється на основі випадково згенерованого ентропійного значення та додаткових криптографічних операцій, що не дозволяють встановити зв'язок із особистими даними користувача. Відповідність голосу певному ідентифікатору перевіряється без розкриття особистості голосуючого, що забезпечує приватність даних у процесі голосування.

Блокчейн-технологія забезпечує високий рівень захисту від кібератак та маніпуляцій з даними завдяки своїм властивостям. Після того як голос зареєстрований

у блокчейні, його неможливо змінити або видалити. Це забезпечується завдяки структурі блокчейну, що гарантує незмінність і стійкість до маніпуляцій завдяки криптографічному зв'язку блоків через унікальні хеші. Кожна зміна будь-якого елемента даних миттєво призводить до розриву зв'язності у мережі, що робить підробку практично неможливою без повного контролю над переважною більшістю вузлів (чого досягти надзвичайно складно через географічну та інституційну децентралізацію мережі).

Крім того, блокчейн Solana застосовує унікальну модель консенсусу PoH у комбінації з Tower Byzantine Fault Tolerance (BFT) - гібридний механізм, що забезпечує високу пропускну здатність транзакцій без зниження безпеки. PoH синхронізує події у часовій шкалі за допомогою криптографічних часових міток, що унеможливлює атаки типу "подвійного витрачання"навіть при високих навантаженнях на мережу. Ця властивість дозволяє системі підтримувати мілісекундні затримки обробки та мільйонні обсяги транзакцій на годину.

3.6. Висновки до розділу

У цьому розділі було розглянуто архітектуру децентралізованого застосунку для проведення голосувань, яка побудована на основі блокчейн-технології Solana. Кожен компонент системи взаємодіє з іншими через смарт-контракт, що забезпечує прозорість, безпеку та незмінність результатів голосування. Клієнтська частина застосунку, розроблена з використанням React, надає користувачам інтуїтивно зрозумілий інтерфейс для участі в голосуванні, а інтеграція з Phantom Wallet дозволяє здійснювати безпечну авторизацію та підпис транзакцій.

Важливими аспектами проєктування є забезпечення високої безпеки та конфіденційності голосів. Використання криптографічних методів і анонімних ідентифікаторів гарантує, що голоси не можуть бути підроблені чи змінені після їх реєстрації в блокчейні. Усі транзакції зберігаються в Solana, що дозволяє забезпечити їх незмінність. Це робить застосунок ефективним і надійним інструментом для проведення прозорих та безпечних голосувань.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ДЕЦЕНТРАЛІЗОВАНОГО ЗАСТОСУНКУ ДЛЯ ПРОВЕДЕННЯ ГОЛОСУВАНЬ

4.1. Реалізація смарт-контракту

Архітектура смарт-контракту для системи голосування побудована на платформі Solana з використанням фреймворку Anchor, що забезпечує безпеку та ефективність роботи. Основна логіка реалізована у програмному модулі votingdapp, який надає дві ключові функції: створення опитування (create_poll) та процес голосування (vote).

Стан програми організований через три основні структури даних. Структура Poll містить інформацію про творця голосування, назву, опис, час завершення та список кандидатів. Кожен кандидат представлений структурою Candidate, яка зберігає ім'я та лічильник голосів. Для запобігання повторному голосуванню використовується структура Voter, яка фіксує зв'язок між учасником голосування та конкретним опитуванням.

Процес створення нового опитування включає перевірку унікальності імені голосування, валідацію вхідних параметрів та ініціалізацію стану. Код обробки створення опитування, окрім перевірки даних з форми, виконує перевірку чи акаунт голосування вже існує:

```
require!(account.is_empty(), Error::PollAlreadyExists);
```

Механізм голосування реалізований з урахуванням обмежень на час голосування та унікальність голосу учасника. Перед обробкою голосу система перевіряє чи відкрите голосування та чи не проголосував учасник раніше:

```
require!(voter.id != signer.key(), Error::AlreadyVoted);
require!(poll.is_open()?, Error::VotingEnded);
```

Для зберігання даних використовуються Program Derived Address (PDA), що дозволяє програмі контролювати доступ до даних без необхідності зовнішніх під-

писів. Кожне опитування зберігається за адресою, що формується з префіксу та ідентифікатора голосування, що забезпечує унікальність та передбачуваність розміщення даних.

Важливою особливістю реалізації на блокчейні Solana є необхідність заздалегідь визначати розмір даних, що зберігаються. Це обумовлено архітектурою блокчейну, де кожен акаунт має фіксований виділений розмір, і зміна цього розміру є неможливою після створення акаунту. Отже, для забезпечення передбачуваності та ефективності використання ресурсів, було встановлено такі обмеження:

- Назва опитування та кандидата обмежена 32 символами.
- Опис опитування обмежений 64 символами.
- Кількість кандидатів обмежена 8.

Ці обмеження дозволяють розрахувати необхідний обсяг пам'яті для зберігання даних про опитування та заздалегідь виділити його на блокчейні. Наприклад, якщо ми зарезервуємо місце під 8 кандидатів з іменами до 32 символів, блокчейн знатиме, скільки точно місця потрібно для даного Poll-акаунту, навіть якщо фактично кандидатів буде менше. Це мінімізує витрати на зберігання, запобігає переповненню та оптимізує продуктивність смарт-контракту. Без таких обмежень, динамічно змінювані розміри даних могли б призвести до неефективного використання пам'яті або до неможливості запису даних, які перевищують виділений простір.

Результатом розробки є функціональний смарт-контракт, який забезпечує створення опитувань, процес голосування з обмеженням на час та унікальність участі, а також прозорий підрахунок результатів. Контракт включає комплексну обробку помилок, що охоплює всі можливі сценарії некоректної роботи. Вихідний код частини смарт-контракту наведено в додатку В

4.2. Реалізація клієнтської частини

Клієнтську частину побудовано з використанням сучасного фреймворку React із застосуванням типізації через TypeScript, що значно покращує надійність і

передбачуваність поведінки інтерфейсу. Уся логіка взаємодії з блокчейном, користувачем і візуальними компонентами розділена на відповідні модулі.

Перший модуль реалізує доступ до бізнес-логіки застосунку. У ньому використано хук, що інкапсулює функції для створення, пошуку та взаємодії з голосуваннями на блокчейні. Для реалізації створення голосування застосовано функцію, яка очікує введення назви голосування, варіантів відповідей, а також вартості ставки. Створення голосування відбувається через асинхронну функцію, яка працює з Anchor-клієнтом і відправляє транзакцію до програми Solana, враховуючи публічний ключ користувача. Отримання списку голосувань реалізовано через функцію, яка виконує запит до RPC-інтерфейсу і десеріалізує дані у зручний для відображення формат.

У другому модулі реалізовано основну логіку інтерфейсу користувача. Компонент за допомогою хуку отримує поточний стан усіх голосувань, а також дозволяє користувачу створити нове голосування або проголосувати за обраний варіант. Стан форми зберігається у локальному стані, що забезпечує зручну інтерактивну роботу з формами без зайвих оновлень екрану. Особливу увагу приділено валідації введених даних — наприклад, створення голосування неможливе, якщо не вказано хоча б два варіанти відповідей. Голосування реалізовано як виклик методу, який передає вибраний індекс варіанту до смарт-контракту.

У третьому модулі реалізовано візуальне оформлення, де визначено інтерфейсні компоненти, зокрема поля введення, кнопки та контейнери для виведення списку голосувань. Використано Tailwind CSS для стилізації, що дозволяє швидко створювати адаптивний і чистий інтерфейс без надлишкової верстки. Кожне голосування виводиться як окремий компонент, що містить його назву, варіанти відповідей та кнопку для голосування. Особливістю інтерфейсу є те, що користувач бачить лише ті голосування, які ще відкриті для голосування, а завершені позначаються окремо.

Уся логіка побудована на асинхронних функціях з обробкою можливих помилок через try/catch, що дозволяє інформувати користувача про успішність або

невдачу дії. Наприклад, при створенні голосування у разі помилки виводиться повідомлення про неможливість створення, а при успішному створенні — оновлюється список голосувань.

Таким чином, застосунок поєднує сучасні вебтехнології з інфраструктурою Solana. Реалізовані рішення забезпечують користувачу прозору, динамічну та безпечну взаємодію з голосуваннями, з мінімальною затримкою й гарантованим збереженням даних у блокчейні. Вихідний код частини клієнтської частини наведено в додатку В

4.3. Звіт про тестування

Метою тестування було перевірити функціональність, надійність, безпеку та продуктивність розробленої системи електронного голосування. Усі тестування проводились в контрольованому середовищі з використанням попередньо підготовлених сценаріїв.

4.3.1. Розробка тестів

У ході розробки тестових випадків було створено сценарії для:

- функціонального тестування;
- тестування графічного інтерфейсу користувача;
- тестування безпеки;
- тестування продуктивності.

4.3.1.1. Функціональне тестування

Функціональні тести призначені для перевірки коректної роботи застосунку для голосування, моделюючи типові дії користувачів та обробку помилок. Під час розробки функціональних тестів було спроєктовано 9 тестових сценаріїв, що охоплюють основні операції, такі як створення нових опитувань та їхніх дублікатів, а також процес голосування за кандидатів, включаючи спроби голосувати за неіснуючих. Також тести перевіряють, як система реагує на некоректні вхідні дані, наприклад, невалідні назви опитувань, їхні описи, а також невірну кількість або імена кандидатів.

4.3.1.2. Тестування безпеки

Тести безпеки, зосереджені на критично важливих аспектах безпеки та цілісності процесу голосування у застосунку. Було спроєктовано 4 тестові сценарії, що включають перевірку коректного підрахунку голосів, навіть якщо імена кандидатів ідентичні, що є важливим для уникнення колізій та забезпечення точності результатів. Крім того, було реалізовано перевірку механізмів запобігання голосуванню з невірним PDA, що є ключовим для захисту від несанкціонованого доступу. Також були успішно протестовані сценарії, що забезпечують неможливість голосування після завершення опитування та запобігання повторному голосуванню, що гарантує принцип одного голосу від одного користувача і загальну справедливість системи.

4.3.1.3. Тестування графічного інтерфейсу

Тести графічного інтерфейсу, сфокусовані на перевірці взаємодії користувача з інтерфейсом створення опитування. Було спроєктовано 9 тестів, що охоплювали перевірку функціоналу відкриття та закриття модального вікна створення опитування, успішне створення опитування з валідними даними, а також валідацію вхідних даних. Зокрема, було перевірено запобігання створенню опитувань з нульовою тривалістю, відображення помилок для порожніх або занадто довгих заголовків опитування, а також обмеження кількості кандидатів та відхилення порожніх або занадто довгих імен кандидатів.

4.3.1.4. Тестування продуктивності

Тестування продуктивності застосунку проводилось з використанням утиліти solana-bench-tps, яка є стандартним інструментом для вимірювання пропускної здатності кластерів у мережі Solana. Утиліта генерує навантаження, надсилаючи велику кількість транзакцій на зазначений RPC-ендпоінт, після чого вимірює максимальний та середній рівень TPS, загальну кількість оброблених транзакцій, відсоток втрат транзакцій. Це дозволило оцінити масштабованість та стабільність системи.

4.3.2. Функціональне тестування

Результат тестування

Тестові сценарії функціонального тестування наведено в Додатку Г. На рисунку 4.1 зображено результати функціонального тестування застосунку.

```
FAIL tests/votingdapp.spec.ts
  votingdapp
    ✓ should create a new poll (477 ms)
    ✓ should fail to create duplicate poll (19 ms)
    ✓ should vote for a candidate (395 ms)
    ✓ should fail to vote for non-existent candidate (445 ms)
    ✓ should fail with invalid poll name (2 ms)
    ✓ should fail with invalid poll description (21 ms)
    ✕ should count vote sent in last moment (750 ms)
    ✓ should fail with invalid candidates count (17 ms)
    ✓ should fail with invalid candidate name (13 ms)

  ● votingdapp › should count vote sent in last moment
```

Рис. 4.1 Результати функціонального тестування

Підсумок тестування

За результатами проведеного функціонального тестування було виявлено, що з 9 запланованих тестових сценаріїв 8 було успішно виконано. Це свідчить про частковий успіх функціонального тестування із показником 89% успішності.

Відомі дефекти

Дефект: Як описано в тестовому сценарії №5.1 — якщо користувач подасть голос за гранично можливий час до завершення голосування — голос не було зараховано.

Причина: Проблема полягає в недостатній кількості часу для передачі даних та успішного оброблення транзакції. Мережеві затримки, навантаження на мережу, час обробки транзакцій, а також внутрішня робота в застосунку займають час, протягом якого транзакція не може бути завершена.

Статус: пріоритет присвоєно низький оскільки граничний час до завершення голосування протягом якого проявляється дефект є малим (менше 10мс) та не буде впливати на користувачів, що подають голос заздалегідь.

4.3.3. Тестування безпеки

Результат тестування

Результати тестування безпеки наведено в Додатку Д. На рисунку 4.2 зображено результати тестування безпеки застосунку.

```
PASS tests/votingdapp.spec.ts
  votingdapp
    ✓ should correctly count votes when candidate names are identical (857 ms)
    ✓ should prevent voting with wrong PDA (457 ms)
    ✓ should fail to vote after poll has ended (357 ms)
    ✓ should fail to vote twice (827 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        3.584 s, estimated 4 s
Ran all test suites.
```

Рис. 4.2 Результати безпекового тестування

Підсумок тестування

За результатами проведеного безпекового тестування було виявлено, що всі 4 заплановані тестові сценарії було успішно виконано. Це свідчить про успіх безпекового тестування із показником 100% успішності.

4.3.4. Тестування графічного інтерфейсу

Результат тестування

Результати тестування графічного інтерфейсу наведено в Додатку Е. На рисунку 4.3 зображено результати тестування графічного інтерфейсу застосунку.

```
✓ Poll Creation UI Tests
  ✓ Should open the create poll modal (87 ms)
  ✓ Should successfully create a poll with valid inputs (312 ms)
  ✓ Should prevent poll creation with zero duration (15 ms)
  ✓ Should show error for empty poll title (12 ms)
  ✓ Should display error for excessively long poll title (18 ms)
  ✓ Should limit candidates to a maximum of 8 (22 ms)
  ✓ Should reject empty candidate names (10 ms)
  ✓ Should reject overly long candidate names (14 ms)
  ✓ Should close the poll creation modal (55 ms)

All GUI tests passed!
Tests:       9 passed, 9 total
Time:        0.555 s
```

Рис. 4.3 Результати тестування графічного інтерфейсу

Підсумок тестування

За результатами проведеного тестування графічного інтерфейсу було виявлено, що всі 9 запланованих тестових сценаріїв було успішно виконано. Це свідчить про успіх тестування графічного інтерфейсу із показником 100% успішності.

4.3.5. Тестування продуктивності

Результат тестування

На рисунку 4.4 зображено результати тестування продуктивності застосунку.

```
[2025-05-26T13:59:51.618113176Z INFO  solana_bench_tps::bench]  Node address        |       Max TPS | Total Transactions
[2025-05-26T13:59:51.618116111Z INFO  solana_bench_tps::bench] ---------------------+---------------+--------------------
[2025-05-26T13:59:51.618121381Z INFO  solana_bench_tps::bench] http://127.0.0.1:8899 |       1098.65 | 68453
[2025-05-26T13:59:51.618127543Z INFO  solana_bench_tps::bench]
    Average max TPS: 1098.65, 0 nodes had 0 TPS
[2025-05-26T13:59:51.618131821Z INFO  solana_bench_tps::bench]
    Highest TPS: 1098.65 sampling period 1s max transactions: 68453 clients: 1 drop rate: 0.02
[2025-05-26T13:59:51.618137071Z INFO  solana_bench_tps::bench]  Average TPS: 987.32
```

Рис. 4.4 Результати тестування продуктивності

Підсумок тестування

Під час тестування продуктивності застосунку було досягнуто максимальний TPS: 1 098.65, середній TPS: 987.32, рівень втрат транзакцій: 0.02%. Ці показники свідчать про стабільну роботу кластера з високою пропускною здатністю та мінімальними втратами при значному навантаженні.

4.3.6. Метрики

Для оцінки якості та успішності тестування системи електронного голосування було використано декілька ключових метрик. Відсоток успішності тестових сценаріїв показує, наскільки коректно реалізовано функціонал, вимірюючи частку успішно виконаних тестів у кожній категорії. Для оцінки продуктивності системи були важливими метрики пропускної здатності — максимальна та середня кількість транзакцій за секунду, а також відсоток втрат транзакцій, що свідчить про стабільність системи під навантаженням.

4.3.7. Критерій прийняття/відхилення релізу

Рішення про випуск системи електронного голосування має базуватися на чітких критеріях:

- Система буде прийнята, якщо функціонал застосунку буде покритий автоматичними тестами, з ступенем успішності понад 89%, а всі тести безпеки та графічного інтерфейсу будуть успішними на 100% та 90% відповідно. Також продуктивність системи має відповідати очікуванням, демонструючи стабільний TPS понад 1 000 та втрати транзакцій менше 0.05%. Ключовим є те, що всі критичні дефекти мають бути виправлені.
- Реліз буде відхилено, якщо виявлено будь-які критичні недоліки, що впливають на основний функціонал, безпеку чи стабільність системи, або якщо відсоток успішності тестів не відповідає встановленим мінімальним значенням.

4.4. Висновки до розділу

У цьому розділі було реалізовано та протестовано децентралізований застосунок для голосувань на Solana з використанням Anchor. Смарт-контракт забезпечує створення опитувань та голосування, ефективно керуючи даними через PDA та встановлюючи обмеження на розмір даних для оптимізації.

Клієнтська частина, розроблена на React з TypeScript, надає інтуїтивний інтерфейс, що взаємодіє з блокчейном.

Проведене тестування підтвердило високу якість застосунку:

- Функціональні тести (89% успіху) виявили лише один незначний дефект.
- Тести безпеки (100% успіху) підтвердили надійність системи.
- Тести графічного інтерфейсу (100% успіху) засвідчили зручність використання.
- Тестування продуктивності показало високий TPS та мінімальні втрати транзакцій, що свідчить про стабільність під навантаженням.

Застосунок відповідає критеріям випуску та готова до впровадження.

РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА

5.1. Економічна характеристика проєктного рішення

Метою роботи є розроблення застосунку для проведення голосувань з використанням технології блокчейну, що буде забезпечувати прозорість, незмінність результатів та виключати можливість фальсифікацій, що особливо важливо в умовах недовіри до традиційних способів голосування. Економічна доцільність проєкту полягає у зниженні витрат на організацію голосувань за рахунок автоматизації процесів, виключення паперових носіїв та мінімізації людського фактору. Система також не потребує додаткових витрат на перевірку достовірності результатів, оскільки дані зберігаються у захищеному блокчейні. Маркетингова оцінка свідчить про високу актуальність продукту: потенційними користувачами можуть бути державні органи, громадські організації, бізнес та освітні заклади. Завдяки прозорості, безпеці, гнучкості налаштувань і зручному інтерфейсу система має конкурентні переваги та перспективи комерціалізації через платну підписку або ліцензування.

5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення товару

Сфера децентралізованих технологій стрімко зростає, і водночас зростає суспільний запит на прозорі, захищені та недорогі способи прийняття колективних рішень. Існуючі системи голосування охоплюють централізовані офлайн-моделі з електронними урнами та сканерами, а також вебзастосунки для дистанційної участі, які мають обмеження у прозорості, довірі та безпеці. У свою чергу, децентралізовані рішення на базі блокчейн-технологій, як Voatz, пропонують підвищену прозорість і незмінність даних, хоча стикаються з викликами відкритості коду та аудиту. Водночас система i-Voting в Естонії, хоча і не базується на блокчейні,

демонструє ефективність електронної ідентифікації та шифрування у масштабах держави. Основними споживачами запропонованої системи є організатори опитувань, локальні спільноти, державні установи та проєкти, що потребують прозорості і довіри. Аналіз ринку та конкурентів підтверджує необхідність створення нової системи на основі блокчейну Solana, яка поєднує зручність, безпеку та інноваційні функції.

5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ

Для оцінки доцільності розроблення запропонованого продукту доцільно розглянути та проаналізувати ключові фактори зовнішнього та внутрішнього середовищ, які впливають на перспективи впровадження системи. Нижче наведено результати експертного оцінювання впливу цих факторів, що дозволяють зробити комплексну оцінку умов для реалізації проєкту.

Таблиця 5.1

Результати експертного оцінювання впливу факторів зовнішнього та внутрішнього середовищ

Фактори	**Середня експертна оцінка, бали**	**Середня вагомість факторів**	**Зважений рівень впливу, бали**
1	**2**	**3**	**4**
Фактори зовнішнього середовища			
Споживачі	4	0,11	0,44
Постачальники	1	0,10	0,10
Конкуренти	-2	0,10	-0,20
Державні органи влади	-1	0,05	-0,05
Інфраструктура	2	0,06	0,12
Законодавчі акти	-2	0,10	-0,20
Профспілки, партії та інші громадські організації	0	0,05	0,00

Продовження табл. 5.1

Система економічних відносин в державі	1	0,06	0,06
Організації-сусіди	1	0,01	0,01
Міжнародні події	0	0,01	0,00
Міжнародне оточення	0	0,03	0,00
Науково-технічний прогрес	3	0,07	0,21
Політичні обставини	-1	0,06	-0,06
Соціально-культурні обставини	1	0,05	0,05
Рівень техніки та технологій	3	0,04	0,12
Особливості міжнародних економічних відносин	0	0,02	0,00
Стан економіки	-1	0,08	-0,08
Загальна сума		**1**	**1,62**
Фактори внутрішнього середовища			
Цілі	4	0,11	0,44
Структура	3	0,16	0,48
Завдання	3	0,07	0,21
Технологія	4	0,20	0,80
Працівники	4	0,21	0,84
Ресурси	3	0,25	0,75
Загальна сума		**1**	**3,52**

Проаналізувавши наведені дані, можна відзначити, що серед зовнішніх факторів найбільше позитивного впливу на діяльність системи надають споживачі, науково-технічний прогрес та рівень розвитку технологій. Водночас конкуренти, чинне законодавство та економічна ситуація створюють певні перешкоди. Що сто-

сується внутрішніх чинників, найкраще оцінені працівники, технологічна база та доступні ресурси, що свідчить про високий потенціал для реалізації проєкту. Таким чином, результати експертної оцінки свідчать про наявність необхідності у розробці продукту та сприятливі умови для його виходу на ринок, що підтверджує обґрунтованість вибору напрямку для дипломної роботи.

5.4. Формування стратегічних альтернатив

Для програми обрано стратегію розвитку нового продукту, оскільки вона дозволяє створити унікальне програмне забезпечення, здатне задовольнити нові потреби користувачів та ринку. Враховуючи, що існуючі рішення не повністю відповідають сучасним вимогам, розроблення нового продукту відкриває можливості для інновацій та забезпечує конкурентні переваги. Такий підхід також передбачає потенційне розширення функціональності через супутні послуги, що підвищує цінність рішення для кінцевих споживачів.

5.5. Бюджетування

У таблиці 5.2 наведено бюджет витрат на матеріали та комплектуючі.

Таблиця 5.2

Бюджет витрат матеріалів та комплектуючих виробів

Назва матеріалів та комплектуючих	**Марка, тип, модель**	**Кількість, шт.**	**Ціна за одиницю, грн.**	**Разом, грн.**
Зовнішній SSD-диск для зберігання резервних копій	Samsung T7 Shield 1TB USB 3.2	1	3 400.00	3 400.00
Набір канцелярії	А4, ручки, стікери	1 комплект	300.00	300.00
Разом				**3 700.00**

У таблиці 5.3 показано бюджет на купівлю основних засобів, де вказані кількість одиниць, ціна, термін експлуатації амортизація.

Таблиця 5.3

Бюджет витрат на купівлю основних засобів

Витрати на куповані засоби	**Марка, тип, модель**	**Кіль-кість, шт.**	**Ціна за оди-ницю, грн.**	**Разом, грн.**	**Термін екс-плуа-тації, років**	**Місячна сума аморти-зації, грн.**
Ноутбук розро-бника	ASUS Vivobook Pro 15 OLED	3	20 000.00	60 000.00	4	1 250.00
Смартфон для те-стування мобіль-ного засто-сунку	Samsung Galaxy A34 5G 6/128GB	1	10 500.00	10 500.00	3	291.67
Монітор зовні-шній	LG 24MP60G-B 24” IPS FHD	1	5 000.00	5 000.00	5	83.33
Разом				**75 500.00**		**1 625.00**

Таблиця 5.4 містить бюджет витрат на оплату праці, розділений на основну зарплату та додаткову (премії 10%), враховано кількість працівників і робочі дні.

Таблиця 5.4

Бюджет витрат на оплату праці

Посада, спеціальність	**Кількість працівни-ків, осіб**	**Час роботи, дні**	**Денна заро-бітна плата працівни-ків, грн.**	**Сума витрат на оплату праці, грн.**
Основна заробітна плата				
Керівник проєкту	1	22	1 000.00	22 000.00

Продовження табл. 5.4

Розробник ПЗ	1	10	2 000.00	20 000.00
Менеджер зі збуту	1	22	1 000.00	22 000.00
Разом				**64 000**
Додаткова заробітна плата (премії, надбавки)				
Премія керівнику (10%)	1	–	–	2 200.00
Премія розробнику (10%)	1	–	–	2 000.00
Премія менеджеру зі збуту (10%)	1	–	–	2 200.00
Разом				**6 400.00**

В таблиці 5.5 наведено розрахунок обов'язкових відрахувань і податків, зокрема єдиного внеску на соціальне страхування, який становить 22% від зарплати.

Таблиця 5.5

Бюджет обов'язкових відрахувань та податків

Посада, спеціальність	**Сума основної заробітної плати**	**Сума додаткової заробітної плати**	**Разом витрат на оплату праці**	**Сума єдиного внеску на соціальне страхування, грн.**
Керівник проєкту	22 000.00	2 200.00	24 200.00	5 324.00
Розробник ПЗ	20 000.00	2 000.00	22 000.00	4 840.00
Менеджер зі збуту	22 000.00	2 200.00	24 200.00	5 324.00
Разом			**70 400.00**	**15 488.00**

У таблиці 5.6 представлено бюджет загальновиробничих витрат, що включає змінні й постійні складові; додаткові витрати на допоміжний персонал та інші статті не передбачені, оскільки всі необхідні функції забезпечуються вже залученими

працівниками без потреби у додаткових ресурсах.

Таблиця 5.6

Бюджет загальновиробничих витрат

Статті витрат	Сума, грн.
Змінні загальновиробничі витрати	
Заробітна плата допоміжного персоналу	0.00
Витрати на МШП	3 700.00
Витрати на електроенергію та технологічні цілі	1 000.00
Витрати на ремонт	0.00
Інші змінні витрати	0.00
Разом змінних витрат	**4 700.00**
Постійні загальновиробничі витрати	
Заробітна плата допоміжного персоналу	0.00
Комунальні послуги	1 000.00
Витрати на оренду	10 000.00
Витрати на ремонт	0.00
Інші постійні витрати	0.00
Разом постійних витрат	11 000.00
Разом загальновиробничих витрат	**15 700.00**

Таблиця 5.7 містить дані про адміністративні витрати та витрати на збут.

Таблиця 5.7

Бюджет адміністративних витрат та витрат на збут

Статті витрат	Сума, грн.
1	**2**
Адміністративні витрати	
Заробітна плата адміністративного персоналу	22 000.00
Витрати на МШП	3 700.00
Витрати на відрядження	0.00
Витрати на ремонт	0.00
Витрати на паливно-мастильні матеріали	0.00
Витрати на сплату податків і зборів	4 840.00
Знос адміністративного обладнання	0.00
Разом адміністративних витрат	**30 540.00**

Продовження табл. 5.7

Витрати на збут	
Заробітна плата менеджерів зі збуту	22 000.00
Витрати на гарантійний ремонт	0.00
Витрати на відрядження	0.00
Витрати на гарантійне обслуговування	10 000.00
Витрати на налагодження і експлуатацію	10 000.00
Витрати на паливно-мастильні матеріали	0.00
Витрати на сплату податків і зборів	4 840.00
Витрати на рекламу	40 000.00
Разом витрат на збут	**86 840.00**

У таблиці 5.8 підсумовано всі основні витрати на розробку проєкту, утворюючи зведений кошторис. Витрати на сировину і матеріали та зворотні відходи не передбачені.

Таблиця 5.8

Зведений кошторис витрат на розробку проєктного рішення (продукту)

Статті витрат	Разом, грн.
Купівельні напівфабрикати та комплектуючі вироби	3 700.00
Паливо та електроенергія на технологічні цілі	1 000.00
Амортизаційні відрахування основних фондів	1 625.00
Основна заробітна плата	64 000.00
Додаткова заробітна плата	6 400.00
Відрахування на соціальне страхування	15 488.00
Витрати на утримання й експлуатацію устаткування	0.00
Змінні загальновиробничі витрати	4 700.00
Постійні загальновиробничі витрати	11 000.00
Разом загальновиробничих витрат	**107 913.00**
Адміністративні витрати	30 540.00
Витрати на збут	86 840.00
Інші операційні витрати	0.00
Разом виробничих і операційних витрат	**225 293.00**

В таблиці 5.9 наведено розрахунок собівартості продукції і планованої ціни

продажу з урахуванням рентабельності 30% при обсязі продажу 5 одиниць.

Таблиця 5.9

Бюджет фінансових результатів

Показники	Сума, грн.
1	**2**
Дохід від реалізації продукції	701 434.50
Податок на додану вартість	140 286.90
Чистий дохід від реалізації продукції	561 147.60
Собівартість реалізованої продукції	107 913.00
Валовий прибуток	453 234.60
Адміністративні витрати	30 540.00
Витрати на збут	86 840.00
Інші операційні витрати	0.00
Фінансовий результат від операційної діяльності	335 854.60
Податок на прибуток	60 453.83
Чистий прибуток (збиток)	275 400.77

5.6. Висновки до розділу

У роботі проведено аналіз ринку, середовища, стратегічних альтернатив та економічного обґрунтування, що дозволило сформувати чітке уявлення про перспективи впровадження програмного забезпечення. Враховано основні характеристики продукту, його технічні специфікації та вигоди для кінцевих користувачів, що забезпечує конкурентні переваги на ринку.

Обрана стратегія — стратегія нового продукту, що передбачає впровадження інноваційного рішення з унікальними властивостями. Такий підхід відповідає ринковим тенденціям і дозволяє максимально задовольнити потреби цільової аудиторії.

Ціна програмного забезпечення становить 140 286.90 грн. За прогнозом, реалізація обраної стратегії забезпечить чистий прибуток у розмірі 275 400.77 грн, що свідчить про економічну ефективність проєкту та його перспективність для інвесторів.

ВИСНОВКИ

У дипломній роботі було розроблено децентралізований застосунок для проведення електронного голосування на основі блокчейн-технології Solana. Головною метою роботи було створення системи, яка забезпечує прозорість, безпеку та незмінність результатів голосування, усуваючи недоліки традиційних централізованих рішень. Для досягнення цієї мети було виконано низку завдань, включаючи аналіз існуючих підходів, проектування архітектури, реалізацію смарт-контракту та клієнтської частини, а також тестування системи.

Теоретичний аналіз показав, що блокчейн-технології є перспективним рішенням для організації голосувань, оскільки вони забезпечують децентралізацію, прозорість та захист від фальсифікацій. Архітектура застосунку базується на чотирьох основних компонентах: клієнтській частині, гаманці, смарт-контракті та блокчейні. Така структура забезпечує зручність для користувачів, безпеку даних та ефективну взаємодію з мережею. Тестування системи підтвердило її функціональність, безпеку та продуктивність. Економічне обґрунтування підтвердило доцільність розробки, оскільки система здатна знизити витрати на організацію голосувань та забезпечити високий рівень довіри.

Перспективи подальшого розвитку включають розширення функціональності, наприклад, додавання можливості створення різних типів голосувань. Покращення масштабованості для підтримки великої кількості учасників. Інтеграція з іншими блокчейн-платформами та сервісами для розширення можливостей.

Таким чином, розроблений застосунок є ефективним рішенням для організації електронного голосування на базі блокчейн, що відповідає сучасним вимогам до безпеки та прозорості. Досягнуті результати є міцною основою для подальших досліджень та практичних впроваджень у сфері децентралізованих систем голосування.

СПИСОК ЛІТЕРАТУРИ

[1] Blockchain structure. URL: https://www.researchgate.net/publication/369352496/figure/fig1/AS:11431281414772432@1746017302739/Blockchain-structure.tif.

(дата звернення 13.05.2025)

[2] Awosika E. What Is Blockchain Voting?. URL: https://second-pocket-shoot-73.hashnode.dev/what-is-blockchain-voting.

(дата звернення 22.01.2025)

[3] PoH, PoS, PoW - Explained. *Helius*. URL: https://www.helius.dev/blog/proof-of-history-proof-of-stake-proof-of-work-explained.

(дата звернення 22.01.2025)

[4] Porat, A., Pratap, A., Shah, P. Blockchain Consensus: An analysis of Proof-of-Work and its applications. *Stanford University*. URL: https://www.scs.stanford.edu/17au-cs244b/labs/projects/porat_pratap_shah_adkar.pdf.

(дата звернення 22.01.2025)

[5] C. T. Nguyen, D. T. Hoang, D. N. Nguyen. Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities. *IEEE Access*. 2019. Ст. 5-9. DOI: 10.1109/ACCESS.2019.2925010.

[6] Victor S. Proof of history: what is it good for? URL: https://www.shoup.net/papers/poh.pdf.

(дата звернення 22.01.2025)

[7] Security and technology. *Voatz*. URL: https://voatz.com/security-and-technology/.

(дата звернення 22.01.2025)

[8] Specter M. A., Koppel J., Weitzner D. The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz, the First Internet Voting Application Used in U.S. Federal Elections. *MIT*. URL: https://internetpolicy.mit.edu/wp-content/uploads/2020/02/SecurityAnalysisOfVoatz_Public.pdf.

(дата звернення 13.05.2025)

[9] Introduction to i-voting. *Valimised*. URL: https://www.valimised.ee/en/internet-voting/more-about-i-voting/introduction-i-voting.

(дата звернення 17.01.2025)

[10] Almeida, R. L., Baiardi, F., Maesa, D. D. F., & Ricci, L. Impact of Decentralization on Electronic Voting Systems: A Systematic Literature Survey. *IEEE Access*. 2023. Ст. 31. DOI: 10.1109/ACCESS.2023.3336593.

[11] Network Performance Report - October 2022. *Solana*. URL: https://solana.com/news/network-performance-report-october-2022.

(дата звернення 18.03.2025)

ДОДАТОК А. ПРИКЛАД ІНТЕРФЕЙСУ КОРИСТУВАЧА

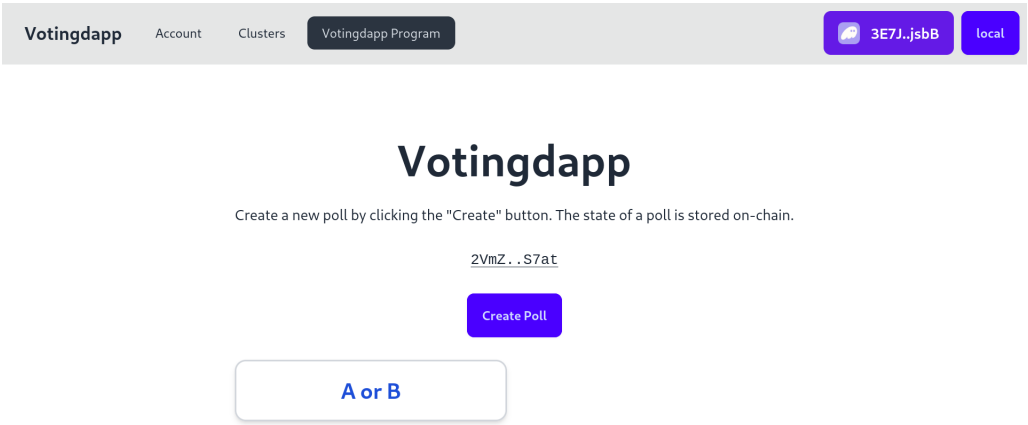

Рис. А.1 Перегляд списку голосувань

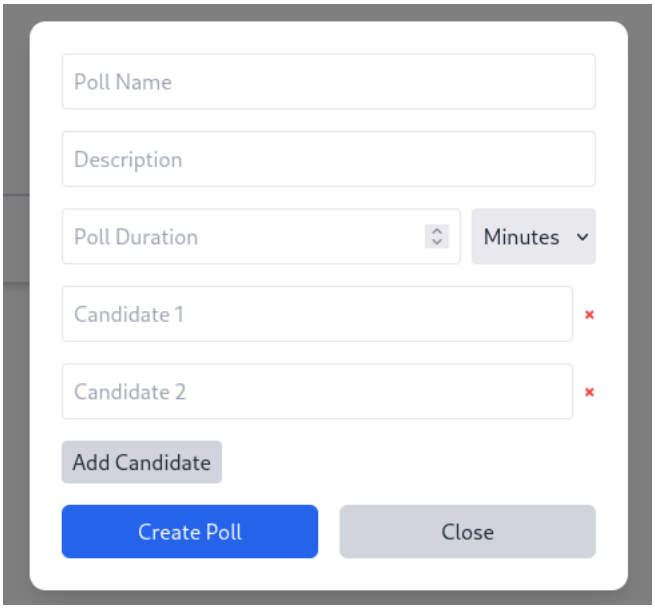

Рис. А.2 Форма для створення голосування

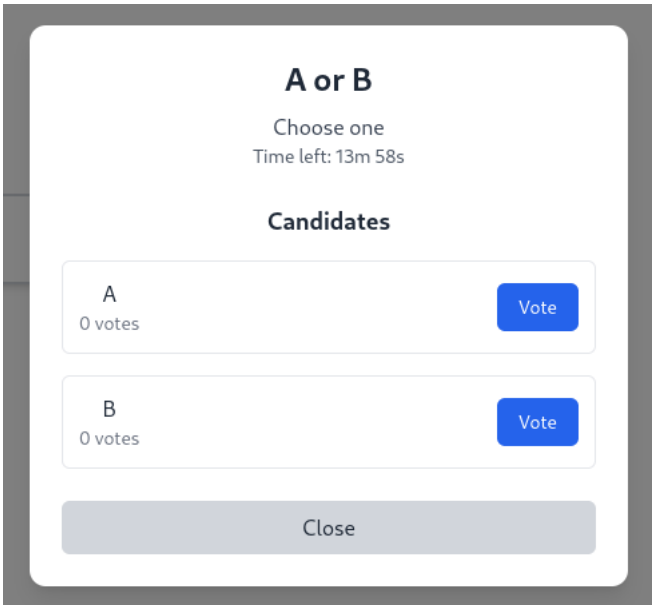

Рис. А.3 Форма для подачі голосу

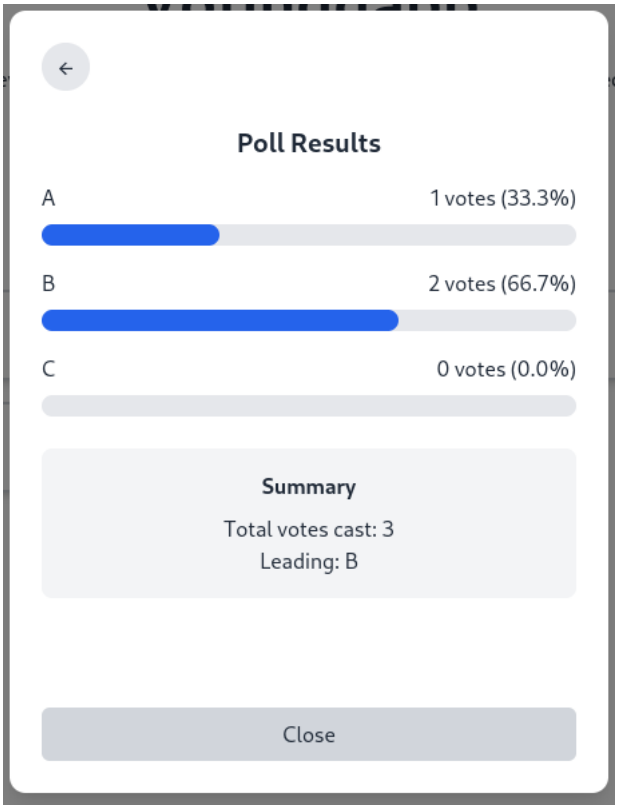

Рис. А.4 Перегляд результатів голосування

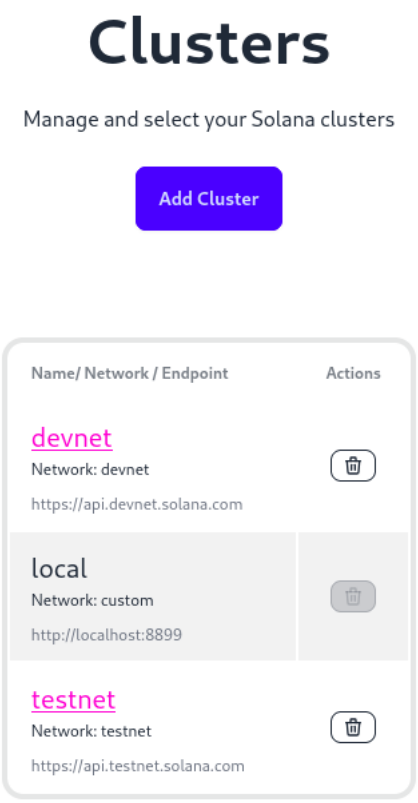

Рис. А.5 Сторінка зі списком кластерів мережі

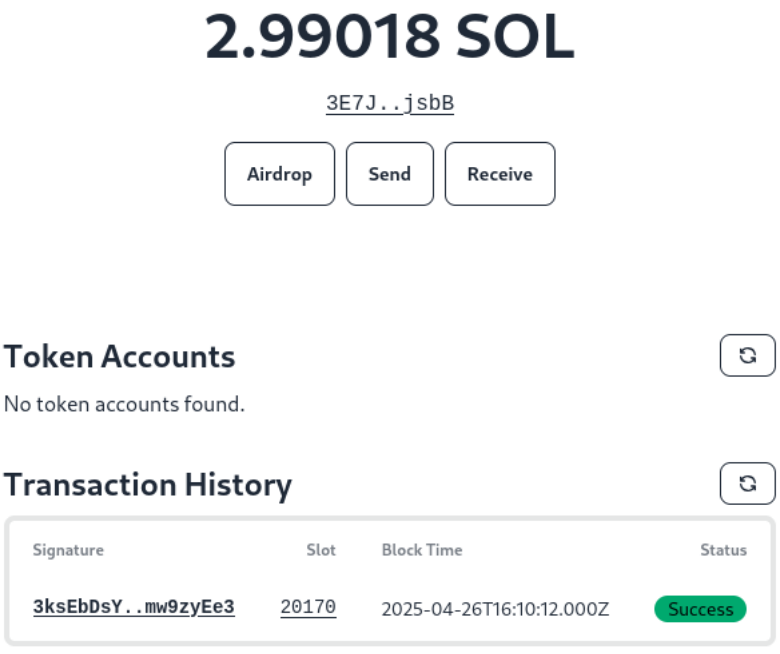

Рис. А.6 Відображення інформації про баланс акаунту та історії транзакцій

ДОДАТОК Б. ІНСТРУКЦІЯ КОРИСТУВАЧА

Компоненти ПЗ

Розроблене програмне забезпечення реалізовано з використанням мови програмування Rust. Як середовище розробки застосовано редактор коду Helix та термінал fish. Було використано систему керування пакетами Cargo для збирання проєкту, а також бібліотеки, пов'язані із взаємодією з блокчейн-платформою Solana. Серверна логіка реалізована як смарт-контракт, який виконується на блокчейні, а клієнтська частина – у вигляді React-застосунку.

Програмне забезпечення потребує мінімальні апаратні ресурси. Достатньо персонального комп'ютера з двоядерним процесором, 4 ГБ оперативної пам'яті, встановленою ОС на базі Linux, macOS або Windows та доступом до інтернету. Для роботи потрібна попередньо встановлена Solana CLI (версія 2.1.17), Anchor CLI (версія 0.30.1), Rust toolchain (версія 1.83.0) та Node.js (версія 22.14.0). Список основних файлів проєкту необхідних для розгортання застосунку із поясненням їхнього призначення наведено в таблиці Б.1.

Таблиця Б.1

Основні файли проєкту

№	Файл	Призначення
1	anchor/Anchor.toml	Файл конфігурації Anchor, що містить налаштування проєкту, адресу програми, параметри середовища та кластерів для розгортання.
2	anchor/target/idl/*.json	Опис інтерфейсу смарт-контракту у форматі JSON, який використовується для взаємодії клієнтської частини з програмою.

Продовження табл. Б.1

№	Файл	Призначення
3	anchor/target/deploy/*.so	Скомпільований байткод смарт-контракту, готовий для розгортання в мережі Solana.
4	keypair.json	Файл із закритим ключем програми, необхідний для розгортання смарт-контракту.

Встановлення ПЗ

Для встановлення застосунку необхідно мати встановлені Solana CLI (версія 2.1.17), Anchor CLI (версія 0.30.1) та Node.js (версія 22.14.0).

Перед розгортанням смарт-контракту потрібно згенерувати файл з парою ключів програми keypair.json і вказати його шлях у anchor/Anchor.toml.

Для розгортання смарт-контракту слід задати адресу потрібного кластера Solana у файлі anchor/Anchor.toml, після чого виконати команду npm run anchor deploy.

Для розгортання клієнтської частини достатньо виконати команду npm install && npm run build && npm start.

Налаштування ПЗ

Для коректної роботи програми необхідно налаштувати підключення до мережі Solana, вказати адресу кластера, а також забезпечити наявність достатнього балансу SOL на акаунті користувача. Крім того, слід правильно розгорнути смарт-контракт.

Базові функції ПЗ

Розроблене програмне забезпечення дозволяє створювати та проводити голосування на базі блокчейну Solana. Один зі способів доступу до функцій — через клієнтський веб-інтерфейс, який забезпечує взаємодію користувача з системою.

Аналіз помилок

Якщо клієнтський інтерфейс відображає помилку доступу до програми, необхідно впевнитися в успішному розгортанні смарт-контракту.

Якщо клієнтський інтерфейс відображає помилку підключення до кластера Solana, необхідно перевірити стабільність інтернет-з’єднання та доступність самого кластера.

Якщо на балансі акаунту недостатньо SOL, потрібно скористатися функцією airdrop для поповнення балансу.

ДОДАТОК В. ПРОГРАМНИЙ КОД

Файл create_poll.rs

```
use anchor_lang::prelude::*;

use crate::error::Error;
use crate::state::poll::Poll;

#[derive(Accounts)]
#[instruction(name: String)]
pub struct CreatePoll<'info> {
    #[account(mut)]
    signer: Signer<'info>,
    #[account(
        init_if_needed,
        payer = signer,
        space = 8 + Poll::INIT_SPACE,
        seeds = [b"poll".as_ref(), name.as_ref
    ()],
        bump
    )]
    poll: Account<'info, Poll>,
    system_program: Program<'info, System>,
}

pub fn handle_create_poll(
    ctx: Context<CreatePoll>,
    name: String,
    description: String,
    timestamp: u64,
    candidates: Vec<String>,
) -> Result<()> {
    let account = &mut ctx.accounts.poll;
    let creator = ctx.accounts.signer.key;

    require!(account.is_empty(), Error::
    PollAlreadyExists);

    account.create(*creator, name, description,
     timestamp, candidates)?;

    Ok(())
}
```

Файл vote.rs

```
use anchor_lang::prelude::*;

use crate::error::Error;
use crate::state::{poll::Poll, voter::Voter};

#[derive(Accounts)]
#[instruction(name: String, candidate: String)]
pub struct Vote<'info> {
    #[account(mut)]
    signer: Signer<'info>,
    #[account(
        mut,
        seeds = [b"poll".as_ref(), name.as_ref
    ()],
        bump
    )]
    poll: Account<'info, Poll>,
    #[account(
        init_if_needed,
        payer = signer,
        space = 8 + Voter::INIT_SPACE,
        seeds = [b"voter".as_ref(), name.as_ref
    (), signer.key().as_ref()],
        bump
    )]
    voter: Account<'info, Voter>,
    system_program: Program<'info, System>,
}

pub fn handle_vote(ctx: Context<Vote>,
    candidate: String) -> Result<()> {
    let signer = &mut ctx.accounts.signer;
    let poll = &mut ctx.accounts.poll;
    let voter = &mut ctx.accounts.voter;

    require!(voter.id != signer.key(), Error::
    AlreadyVoted);
    require!(poll.is_open()?, Error::
    VotingEnded);
```

```
    poll.vote(candidate)?;

    voter.id = signer.key();

    Ok(())
}
```

Файл votingsdapp-ui.tsx

```
'use client'

import { Keypair, PublicKey } from '@solana/
    web3.js'
import { useWallet } from '@solana/wallet-
    adapter-react'
import { useMemo, useState, useEffect } from '
    react'
import { ellipsify } from '../ui/ui-layout'
import { ExplorerLink } from '../cluster/
    cluster-ui'
import { useVotingdappProgram,
    useVotingdappProgramAccount } from './
    votingdapp-data-access'
import BN from 'bn.js'

function useCountdown(targetTimestamp: number |
     undefined) {
  const [timeLeft, setTimeLeft] = useState<
    number>(0)

  useEffect(() => {
    if (!targetTimestamp) return

    const update = () => {
      const now = Math.floor(Date.now() / 1000)
      setTimeLeft(Math.max(0, targetTimestamp -
     now))
    }

    update()
    const interval = setInterval(update, 1000)

    return () => clearInterval(interval)
  }, [targetTimestamp])

  return timeLeft
}

function formatTime(seconds: number) {
  const m = Math.floor(seconds / 60)
  const s = seconds % 60
  return `${m}m ${s}s`
}

function useEditPoll() {
  const [error, setError] = useState("")
  const [duration, setDuration] = useState("")
  const [durationUnit, setDurationUnit] =
    useState("min")

  const isValid = () => {
    const durationValue = parseInt(duration)
    if (isNaN(durationValue) || durationValue <
     1) return "Duration must be greater than
    0"
    return ""
  }

  const handleEdit = async () => {
    const validationError = isValid()
    if (validationError) {
      setError(validationError)
      return false
    } else {
      setError("")
    }

    const durationValue = parseInt(duration)
    let durationInSeconds = durationValue * 60
    if (durationUnit === "hour")
    durationInSeconds *= 60
    if (durationUnit === "day")
    durationInSeconds *= 60 * 60 * 24

    const timestamp = new BN(Math.floor(Date.
    now() / 1000) + durationInSeconds)
    return timestamp
  }

  return {
    error,
    duration,
    setDuration,
    durationUnit,
    setDurationUnit,
    handleEdit
  }
```

```
}

export function VotingdappCreate() {
  const [isModalOpen, setIsModalOpen] =
    useState(false)

  const openModal = () => {
    setIsModalOpen(true)
  }

  const closeModal = () => {
    setIsModalOpen(false)
  }

  return (
    <div>
      <button
        className="btn btn-xs lg:btn-md btn-
    primary"
        onClick={openModal}
      >
        Create Poll
      </button>
      {isModalOpen && <VotingdappCreatePopup
    onClose={closeModal} />}
    </div>
  )
}

export function VotingdappCreatePopup({ onClose
    }: { onClose: () => void }) {
  const { createPollMutation } =
    useVotingdappProgram()

  const [name, setName] = useState("")
  const [description, setDescription] =
    useState("")
  const [candidates, setCandidates] = useState<
    string[]>([""])
  const [error, setError] = useState("")
  const [duration, setDuration] = useState("")
  const [durationUnit, setDurationUnit] =
    useState("min")

  const isValid = () => {
    if (!name.trim()) return "Poll name is
    required"
    if (new TextEncoder().encode(name).length >
     64) return "Poll name is too long"
    if (!description.trim()) return "
    Description is required"
    if (new TextEncoder().encode(description).
    length > 64) return "Description is too
    long"
    const durationValue = parseInt(duration)
    if (isNaN(durationValue) || durationValue <
     1) return "Duration must be greater than
    0"
    if (candidates.length === 0) return "At
    least one candidate is required"
    if (candidates.length > 8) return "Maximum
    8 candidates allowed"
    for (const c of candidates) {
      if (!c.trim()) return "Candidate name can
    't be empty"
      if (new TextEncoder().encode(c).length >
    32) return "Candidate name too long"
    }
    return ""
  }

  const handleCreate = async () => {
    const validationError = isValid()
    if (validationError) {
      setError(validationError)
      return
    }

    const durationValue = parseInt(duration)
    let durationInSeconds = durationValue * 60
    if (durationUnit === "hour")
    durationInSeconds *= 60
    if (durationUnit === "day")
    durationInSeconds *= 60 * 60 * 24

    const timestamp = new BN(Math.floor(Date.
    now() / 1000) + durationInSeconds)

    await createPollMutation.mutateAsync({ name
    , description, timestamp, candidates })
    onClose()
  }

  return (
    <div className="fixed inset-0 z-50 flex
    items-center justify-center bg-black bg-
```

```
opacity-50">
  <div className="bg-white rounded-xl p-6 w
-full max-w-md shadow-lg space-y-4">
    <input
      className="w-full border p-2 rounded"
      placeholder="Poll Name"
      value={name}
      onChange={e => setName(e.target.value
)}
    />
    <input
      className="w-full border p-2 rounded"
      placeholder="Description"
      value={description}
      onChange={e => setDescription(e.
target.value)}
    />
    <div className="flex gap-2">
      <input
        type="number"
        className="w-full border p-2
rounded"
        placeholder="Poll Duration"
        value={duration}
        onChange={e => setDuration(e.target
.value)}
        min="1"
      />
      <select
        className="border p-2 rounded"
        value={durationUnit}
        onChange={e => setDurationUnit(e.
target.value)}
      >
        <option value="min">Minutes</option
>
        <option value="hour">Hours</option>
        <option value="day">Days</option>
      </select>
    </div>

    {candidates.map((c, i) => (
      <div key={i} className="flex items-
center gap-2">
        <input
          className="flex-1 border p-2
rounded"
          placeholder={`Candidate ${i +
1}`}
          value={c}
          onChange={e => {
            const copy = [...candidates]
            copy[i] = e.target.value
            setCandidates(copy)
          }}
        />
        {candidates.length > 1 && (
          <button
            className="text-red-500 font-
bold"
            onClick={() => setCandidates(
candidates.filter((_, j) => j !== i))}
            title="Remove candidate"
          >
            ×
          </button>
        )}

      </div>
    ))}
    <div className="flex justify-between">
      {candidates.length < 8 && (
        <button
          className="px-2 py-1 bg-gray-300
rounded"
          onClick={() => setCandidates([...
candidates, ""])}
        >
          Add Candidate
        </button>
      )}
    </div>
    {error && <div className="text-red-600
text-sm">{error}</div>}
    <div className="flex justify-between mt
-6 space-x-4">
      <button
        className="flex-1 px-4 py-2 bg-blue
-600 text-white rounded-md disabled:opacity
-50"
        onClick={handleCreate}
        disabled={createPollMutation.
isPending}
      >
        Create Poll {createPollMutation.
isPending && '...'}
```

```
          </button>
          <button
            className="flex-1 px-4 py-2 bg-gray
    -300 text-gray-800 rounded-md"
            onClick={onClose}
          >
            Close
          </button>
        </div>
      </div>
    </div>
  )
}

export function VotingdappList() {
  const { accounts, getProgramAccount } =
    useVotingdappProgram()

  if (getProgramAccount.isLoading) {
    return <div className="flex justify-center
    items-center h-full">
      <span className="loading loading-spinner
    loading-lg"></span>
    </div>
  }
  if (!getProgramAccount.data?.value) {
    return (
      <div className="alert alert-info flex
    justify-center">
        <span>Program account not found. Make
    sure you have deployed the program and are
    on the correct cluster.</span>
      </div>
    )
  }

  return (
    <div className={'space-y-6'}>
      {accounts.isLoading ? (
        <div className="flex justify-center
    items-center h-full">
          <span className="loading loading-
    spinner loading-lg"></span>
        </div>
      ) : accounts.data?.length ? (
        <div className="grid md:grid-cols-2 gap
    -4">
          {accounts.data?.map((account) => (
            <VotingdappCard key={account.
    publicKey.toString()} account={account.
    publicKey} />
          ))}
        </div>
      ) : (
        <div className="text-center">
          <h2 className={'text-2xl'}>No polls</
    h2>
          No polls found. Create one to get
    started.
        </div>
      )}
    </div>
  )
}

function VotingdappCard({ account }: { account:
     PublicKey }) {
  const { accountQuery } =
    useVotingdappProgramAccount({ account })

  const name = useMemo(() => accountQuery.data
    ?.name, [accountQuery.data?.name])
  const [isModalOpen, setIsModalOpen] =
    useState(false)

  const openModal = () => {
    setIsModalOpen(true)
  }

  const closeModal = () => {
    setIsModalOpen(false)
  }

  return accountQuery.isLoading ? (
    <span className="loading loading-spinner
    loading-lg"></span>
  ) : (
    <div className="card border-2 border-gray
    -300 rounded-xl shadow-md p-4 w-full max-w-
    md mx-auto">
      <div className="flex flex-col items-
    center space-y-4 text-center">
        <h2
          className="text-2xl font-semibold
    text-blue-700 hover:underline cursor-
    pointer"
```

```
            onClick={openModal}
          >
            {name}
          </h2>
        </div>
        {isModalOpen && (
          <VotingdappCardPopup
            account={account}
            onClose={closeModal}
          />
        )}
      </div>
    )
}

function VotingdappCardPopup({ account, onClose
    }: { account: PublicKey, onClose: () =>
    void }) {
  const { voteMutation } = useVotingdappProgram
    ()
  const { accountQuery } =
    useVotingdappProgramAccount({ account })

  type LoadingState = { [key: string]: boolean
    }
  const [loadingState, setLoadingState] =
    useState<LoadingState>({})
  const [viewMode, setViewMode] = useState<'
    main' | 'edit' | 'results'>('main')
  const { publicKey } = useWallet()

  const isCreator = useMemo(() => {
    const creator = accountQuery.data?.creator
    ?.toString()
    return creator && publicKey ? creator ===
    publicKey.toBase58() : false
  }, [accountQuery.data?.creator, publicKey])

  const name = useMemo(() => accountQuery.data
    ?.name ?? '', [accountQuery.data?.name])
  const description = useMemo(() =>
    accountQuery.data?.description, [
    accountQuery.data?.description])
  const timestamp = accountQuery.data?.
    timestamp?.toNumber()
  const timeLeft = useCountdown(timestamp)
  const candidates = useMemo(() => accountQuery
    .data?.candidates, [accountQuery.data?.
    candidates])

  const {
    error,
    duration,
    setDuration,
    durationUnit,
    setDurationUnit,
    handleEdit
  } = useEditPoll()

  const handleVoteClick = async (candidate: {
    name: string; votesCount: BN }) => {
    setLoadingState((prevState) => ({
      ...prevState,
      [candidate.name]: true,
    }))

    try {
      await voteMutation.mutateAsync({ name:
    name, candidate: candidate.name })
      await accountQuery.refetch()
    } catch (e) {
      console.error(e)
    } finally {
      setLoadingState((prevState) => ({
        ...prevState,
        [candidate.name]: false,
      }))
    }
  }

  const onSaveEdit = async () => {
    const timestamp = await handleEdit()
    if (timestamp) {
      // await editPollMutation.mutateAsync({
    timestamp })
      setViewMode('main')
    }
  }

  return (
    <div className="fixed inset-0 z-50 flex
    items-center justify-center bg-black bg-
    opacity-50">
      <div className="bg-white rounded-xl w-
    full max-w-md shadow-lg relative overflow-
    hidden">
```

```
    <div
      className="flex w-[300%] transition-
transform duration-500 ease-in-out"
      style={{
        transform: viewMode === 'edit' ? '
translateX(0)' :
          viewMode === 'main' ? 'translateX
(-33.33%)' :
            'translateX(-66.66%)'
      }}
    >
      <PollEditView
        onSave={onSaveEdit}
        onBack={() => setViewMode('main')}
        error={error}
        duration={duration}
        setDuration={setDuration}
        durationUnit={durationUnit}
        setDurationUnit={setDurationUnit}
      />

      <div className="w-full p-6" style={{
flex: '0 0 33.33%' }}>
        <h2 className="text-2xl font-bold
text-center">{name}</h2>
        <p className="text-gray-700 text-
center mt-2">
          {description}
          <div className="text-gray-600
text-sm">
            {timeLeft === 0 ? 'Voting has
ended' : `Time left: ${formatTime(timeLeft)
}`}
          </div>
        </p>
        <div className="mt-6 space-y-4">
          <h3 className="text-lg font-
semibold">Candidates</h3>
          {candidates?.map((candidate: {
name: string; votesCount: BN }) => (
            <div
              key={candidate.name}
              className="flex items-center
justify-between border p-3 rounded-md"
            >
              <div>
                <p className="font-medium
">{candidate.name}</p>
                <p className="text-sm text-
gray-500">{candidate.votesCount.toString()}
 votes</p>
              </div>
              <button
                className="bg-blue-600
hover:bg-blue-700 text-white text-sm
rounded-md px-4 py-2 disabled:opacity-50
disabled:cursor-not-allowed"
                onClick={() =>
handleVoteClick(candidate)}
                disabled={loadingState[
candidate.name] || timeLeft === 0}
              >
                {loadingState[candidate.
name] ? (
                  <div className="w-5 h-5
border-2 border-white border-t-transparent
rounded-full animate-spin" />
                ) : (
                  "Vote"
                )}
              </button>
            </div>
          ))}
        </div>
        <div className="flex justify-
between mt-4">
          {isCreator && (
            <button
              className="px-4 py-2 bg-
yellow-500 text-white rounded-md hover:bg-
yellow-600"
              onClick={() => setViewMode('
edit')}
            >
              Edit Poll
            </button>
          )}
          {timeLeft === 0 && (
            <button
              className="px-4 py-2 bg-
purple-500 text-white rounded-md hover:bg-
purple-600"
              onClick={() => setViewMode('
results')}
            >
              View Results
```

```
                </button>
              )}
            </div>
          </div>

          <PollResultsView
            candidates={candidates || []}
            onBack={() => setViewMode('main')}
          />
        </div>

        <div className="p-6">
          <button
            className="w-full px-4 py-2 bg-gray
    -300 text-gray-800 rounded-md"
            onClick={onClose}
          >
            Close
          </button>
        </div>
      </div>
    </div>
  )
}

function PollEditView({
  onSave,
  onBack,
  error,
  duration,
  setDuration,
  durationUnit,
  setDurationUnit
}: {
  onSave: () => void,
  onBack: () => void,
  error: string,
  duration: string,
  setDuration: (value: string) => void,
  durationUnit: string,
  setDurationUnit: (value: string) => void
}) {
  return (
    <div className="w-full p-6">
      <div className="flex items-center mb-6">
        <button
          onClick={onBack}
          className="text-lg bg-gray-200 hover:
bg-gray-300 text-gray-800 rounded-full px-3
 py-1"
        >
          ←
        </button>
      </div>
      <div className="space-y-4">
        <h3 className="text-xl font-bold">Edit
Poll Duration</h3>
        <div className="flex gap-2">
          <input
            type="number"
            className="w-full border p-2
rounded"
            placeholder="Poll Duration"
            value={duration}
            onChange={e => setDuration(e.target
.value)}
            min="1"
          />
          <select
            className="border p-2 rounded"
            value={durationUnit}
            onChange={e => setDurationUnit(e.
target.value)}
          >
            <option value="min">Minutes</option
>
            <option value="hour">Hours</option>
            <option value="day">Days</option>
          </select>
        </div>
        {error && <div className="text-red-600
text-sm">{error}</div>}
        <button
          className="mt-6 w-full px-4 py-2 bg-
green-600 text-white rounded-md hover:bg-
green-700"
          onClick={onSave}
        >
          Save Changes
        </button>
      </div>
    </div>
  )
}

function PollResultsView({
```

```
  candidates,
  onBack
}: {
  candidates: { name: string; votesCount: BN
    }[],
  onBack: () => void
}) {
  const totalVotes = useMemo(() => {
    return candidates.reduce((sum, candidate)
    => sum + candidate.votesCount.toNumber(),
    0)
  }, [candidates])

  return (
    <div className="w-full p-6">
      <div className="flex items-center mb-6">
        <button
          onClick={onBack}
          className="text-lg bg-gray-200 hover:
    bg-gray-300 text-gray-800 rounded-full px-3
     py-1"
        >
          ←
        </button>
      </div>

      <h3 className="text-xl font-bold mb-4">
    Poll Results</h3>

      <div className="space-y-4 mb-6">
        {candidates.map((candidate) => {
          const votes = candidate.votesCount.
    toNumber()
          const percentage = totalVotes > 0 ? (
    votes / totalVotes * 100) : 0

          return (
            <div key={candidate.name} className
    ="space-y-2">
              <div className="flex justify-
    between">
                <span className="font-medium">{
    candidate.name}</span>
                <span>{votes} votes ({
    percentage.toFixed(1)}%)</span>
              </div>
              <div className="w-full bg-gray
    -200 rounded-full h-4">
                <div
                  className="bg-blue-600 h-4
    rounded-full"
                  style={{ width: `${percentage
    }%` }}
                ></div>
              </div>
            </div>
          )
        })}
      </div>

      <div className="bg-gray-100 p-4 rounded-
    lg">
        <h4 className="font-semibold mb-2">
    Summary</h4>
        <p>Total votes cast: {totalVotes}</p>
        {totalVotes > 0 && (
          <p>
            Leading: {candidates.reduce((prev,
    current) =>
              prev.votesCount.gt(current.
    votesCount) ? prev : current
            ).name}
          </p>
        )}
      </div>
    </div>
  )
}
```

ДОДАТОК Г. ТЕСТОВІ ВИПАДКИ ФУНКЦІОНАЛЬНИХ ТЕСТІВ

Таблиця Г.1

Тестові випадки функціональних тестів

ID | Name	**Опис**	
№1.1 | Створення опитування	Перевірка успішного створення нового опитування	
Кроки	**Очікування**	**Результат**
1. Викликати метод createPoll 2. Вказати коректні параметри	Опитування створено з правильними даними	Passed
№1.2 | Дублювання опитування	Спроба створити опитування з існуючою назвою	
Кроки	**Очікування**	**Результат**
1. Викликати метод createPoll 2. Використати існуючу назву опитування	Помилка "Poll already exists"	Passed
№2.1 | Голосування	Успішне голосування за кандидата	
Кроки	**Очікування**	**Результат**
1. Викликати метод vote 2. Вказати існуючого кандидата	Кількість голосів збільшена на 1	Passed
№2.2 | Голосування за неіснуючого кандидата	Спроба проголосувати за неіснуючого кандидата	
Кроки	**Очікування**	**Результат**

Продовження табл. Г.1

1. Викликати метод vote 2. Вказати неіснуюче ім’я кандидата	Помилка "Candidate not found"	Passed
№3.1 | Некоректна назва опитування	Спроба створити опитування з надто довгою або порожньою назвою	
Кроки	**Очікування**	**Результат**
1. Викликати метод createPoll 2. Вказати порожню назву або довше 32 символів	Помилка "Invalid poll name"	Passed
№3.2 | Некоректний опис опитування	Спроба створити опитування з надто довгим або порожнім описом	
Кроки	**Очікування**	**Результат**
1. Викликати метод createPoll 2. Вказати порожній опис або довше 64 символів	Помилка "Invalid poll description"	Passed
№4.1 | Некоректна назва кандидата	Спроба створити кандидата з надто довгою або порожньою назвою	
Кроки	**Очікування**	**Результат**
1. Викликати метод createPoll 2. Вказати порожню назву кандидата або довше 32 символів	Помилка "Invalid candidate name"	Passed
№4.2 | Некоректна кількість кандидатів	Спроба створити опитування з надто великою кількістю кандидатів	
Кроки	**Очікування**	**Результат**

Продовження табл. Г.1

1. Викликати метод createPoll 2. Вказати кандидатів, кількість яких більша 8	Помилка "Invalid candidate count"	Passed
№5.1	Голосування в останній момент перед завершенням	Спроба проголосувати в останній момент перед завершенням голосування
Кроки	**Очікування**	**Результат**
1. Створити опитування з часом до завершення 2. Викликати метод vote за 10мс до завершення	Успішне зарахування голосу	Failed

ДОДАТОК Д. ТЕСТОВІ ВИПАДКИ ТЕСТІВ БЕЗПЕКИ

Таблиця Д.1

Тестові випадки тестів безпеки

ID | Name	**Опис**	
№1.1 | Голосування з невірним PDA	Спроба проголосувати, використовуючи невірний PDA	
Кроки	**Очікування**	**Результат**
1. Створити опитування 2. Згенерувати некоректний PDA для виборця 3. Викликати метод vote з некоректним PDA	Транзакція відхилена; кількість голосів не змінюється	Passed
№2.1 | Повторне голосування	Спроба проголосувати вдруге	
Кроки	**Очікування**	**Результат**
1. Викликати метод vote 2. Використати той самий гаманець	Помилка "Already voted"	Passed
№2.2 | Голосування після завершення	Спроба проголосувати у завершеному опитуванні	
Кроки	**Очікування**	**Результат**
1. Створити опитування з коротким часом до завершення 2. Зачекати завершення 3. Викликати метод vote	Помилка "Voting ended"	Passed

Продовження табл. Д.1

№3.1	Неунікальні назви кандидатів	Спроба проголосувати в опитуванні з кандидатами, що мають однакові назви	
Кроки		**Очікування**	**Результат**
1. Створити опитування з кандидатами з однаковими назвами 2. Викликати метод vote для другого кандидата в списку		Голос зараховується другому кандидату	Passed

ДОДАТОК Е. ТЕСТОВІ ВИПАДКИ ТЕСТІВ ГРАФІЧНОГО ІНТЕРФЕЙСУ

Таблиця Е.1

Тестові випадки тестів графічного інтерфейсу

ID | Name	**Опис**	
№1.1 | Відкриття модального вікна	Перевірка відкриття вікна створення опитування	
Кроки	**Очікування**	**Результат**
1. Натиснути кнопку "Create Poll"	Відображається модальне вікно з формою опитування	Passed
№1.2 | Успішне створення опитування	Заповнення всіх полів коректними значеннями	
Кроки	**Очікування**	**Результат**
1. Заповнити поля: назва, опис, 2 кандидати, тривалість = 10, одиниця = хвилини 2. Натиснути кнопку створення	Опитування успішно створено, модальне вікно закрито	Passed
№1.3 | Некоректна тривалість	Спроба створити опитування з тривалістю = 0	
Кроки	**Очікування**	**Результат**
1. Вказати тривалість = 0 2. Натиснути кнопку створення	Повідомлення про помилку "Duration must be greater than 0"	Passed
№1.4 | Порожнє поле назви	Спроба створити опитування без назви	

Продовження табл. Е.1

Кроки	**Очікування**	**Результат**
1. Очистити поле назви 2. Натиснути кнопку створення	Повідомлення про помилку "Poll name is required"	Passed
№1.5 | Довга назва	Спроба ввести назву довжиною понад 64 байти	
Кроки	**Очікування**	**Результат**
1. Ввести назву довжиною понад 64 символи 2. Натиснути кнопку створення	Повідомлення про помилку "Poll name is too long"	Passed
№1.6 | Більше 8 кандидатів	Спроба додати більше 8 кандидатів	
Кроки	**Очікування**	**Результат**
1. Додати 9 полів кандидатів	Повідомлення про помилку "Maximum 8 candidates allowed"	Passed
№1.7 | Порожнє поле кандидата	Один з кандидатів має порожню назву	
Кроки	**Очікування**	**Результат**
1. Один з кандидатів має порожню назву 2. Натиснути кнопку створення	Повідомлення про помилку "Candidate name can’t be empty"	Passed
№1.8 | Довге ім’я кандидата	Спроба ввести назву кандидата понад 32 символи	
Кроки	**Очікування**	**Результат**

Продовження табл. Е.1

1. Додати кандидата з назвою понад 32 символи	Повідомлення про помилку "Candidate name too long"	Passed
№1.9	Закриття модального вікна	Перевірка закриття модального вікна	
Кроки	**Очікування**	**Результат**
1. Натиснути кнопку "×"або "Cancel"	Модальне вікно закрите	Passed