Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Магістерська кваліфікаційна робота на тему:
«Метод та засіб генерування смарт-контрактів для захищеного голосування»

Виконав: студент 2 курсу групи 2БС-22м
спеціальності 125 Кібербезпека
_____Олег НАГИРНЯК

Керівник: к. т. н., доцент каф. ЗІ
_____Юрій БАРИШЕВ
«12» 12 2023 р.

Опонент: к. т. н., доцент каф. ПЗ
_____Денис КАТЄЛЬНІКОВ
«13» 12 2023 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«14» 12 2023 р.

Вінниця ВНТУ – 2023 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ,
д. т. н., проф.
Володимир ЛУЖЕЦЬКИЙ
«19» 09 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Нагирняку Олегу Юрійовичу

1. Тема роботи: «Метод та засіб генерування смарт-контрактів для захищеного голосування»
керівник роботи: Баришев Юрій Володимирович, к. т. н., доцент кафедри ЗІ, затверджені наказом ректора ВНТУ від 18 вересня 2023 №247.

2. Строк подання студентом роботи: 13 грудня 2023р.

3. Вихідні дані до роботи:
   - блокчейн – Sepolia;
   - мова програмування інтерфейсу користувача – JavaScript;
   - мова програмування смарт-контрактів - Solidity;
   - покриття смарт-контрактів блоковими автоматичними тестами.

4. Зміст текстової частини: Вступ. 1. Аналіз методів захищеного електронного голосування. 2. Метод генерування смарт-контрактів для захищеного електронного голосування. 3. Засіб генерування смарт-контрактів. 4. Тестування засобу генерування смарт-контракту. 5. Економічна частина. Висновки. Перелік використаних джерел. Додатки.

5. Перелік ілюстративного матеріалу: Актуальність, мета та задачі МКР (плакат А4). Наукова новизна та практична цінність (плакат А4). Вибір параметрів генерування голосування (плакат А4). Модуль генерування голосування (плакат А4). Модуль проведення голосування (плакат А4). Алгоритм взаємодії з модулем генерування голосування (плакат А4). Процедура налаштування параметру фінансування (плакат А4). Процедура генерування голосування (плакат А4). Результати тестування модуля генерування голосування (плакат А4). Робоча область інтерфейсу користувача (плакат А4). Результат статичного тестування (плакат А4). Економічна частина (плакат А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Юрій БАРИШЕВ, к. т. н., доц. каф. ЗІ	01.09	15.09
2	Юрій БАРИШЕВ, к. т. н., доц. каф. ЗІ	01.09	21.09
3	Юрій БАРИШЕВ, к. т. н., доц. каф. ЗІ	01.09	26.10
4	Юрій БАРИШЕВ, к. т. н., доц. каф. ЗІ	01.09	23.11
5	Ольга РАТУШНЯК, к. т. н., доц. каф. ЕПВМ	01.09	17.11

7. Дата видачі завдання 1 вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітки
1	Аналіз завдання. Вступ	01.09.2023 – 10.09.2023	
2	Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи	10.09.2023 – 15.09.2023	
3	Розробка рішень	16.09.2023 – 22.09.2023	
4	Проектування програмного засобу	23.09.2023 – 29.09.2023	
5	Практична реалізація, експериментування, результати	30.09.2023 – 12.10.2023	
6	Розробка розділу тестування і обґрунтування доцільності розробки	14.10.2023 – 10.11.2023	
7	Розробка розділу економічного обґрунтування доцільності розробки	11.11.2023 – 17.11.2023	
8	Аналіз виконання ІЗ, висновки	18.11.2023 – 24.11.2023	
9	Оформлення пояснювальної записки	25.11.2023 – 30.11.2023	
10	Попередній захист та доопрацювання МКР	28.11.2023 – 10.12.2023	
11	Перевірка магістерської роботи на наявність плагіату	02.12.2023 – 10.12.2023	
12	Представлення МКР до захисту	11.12.2023 – 14.12.2023	
13	Захист МКР	14.12.2023 – 21.12.2023	

Студент _____ Олег НАГИРНЯ

Керівник роботи _____ Юрій БАРИШЕ

АНОТАЦІЯ

Магістерська кваліфікаційна робота складається з 84 сторінок формату А4, на яких є 34 рисунка, 9 таблиць, 29 формул, список використаних джерел містить 34 найменування.

Магістерська кваліфікаційна робота присвячена розробці методу та засобу генерування смарт-контрактів для захищеного голосування. У роботі проаналізовано сучасні системи електронного голосування, а також основні недоліки та загрози кібербезпеці для таких систем. У межах роботи виконано математичний опис системи захищених голосувань, описано метод генерування голосувань та правила розмежування прав доступу. Спроектовано архітектуру засобу генерування смарт-контрактів, а також описано основні процедури роботи системи. Після розробки алгоритмів функціонування програмного засобу та його окремих модулів здійснено програмну реалізацію. Проведено автоматизоване блокове тестування смарт-контрактів та статичне тестування безпеки.

Ключові слова: захищене електронне голосування, інформаційна безпека, смарт-контракт, система голосування, рольова модель, прозорість та гнучкість

ABSTRACT

The master's qualification work consists of 84 pages of A4 format, which contains 34 figures, 9 tables, 29 formulas, the list of references contains 34 items.

The master's qualification work is devoted to the development of a method and tool for generating smart contracts for secure voting. The work analyzes modern electronic voting systems, as well as the main shortcomings and threats to cyber security for such systems. Within the framework of the work, a mathematical description of the protected voting system was performed, the method of votings generating and the rules for distinguishing access rights were described. The architecture of the tool for generating smart contracts is designed, and the main procedures of the system are also described. After the development of algorithms for the functioning of the software tool and its individual modules, software implementation was carried out. Automated unit testing of smart contracts and static security testing were performed.

Keywords: secure electronic voting, information security, smart contract, voting system, role model, transparency and flexibility

ЗМІСТ

ВСТУП

В світі інформаційних технологій, незважаючи на значні досягнення у багатьох сферах, деякі процеси залишаються недостатньо ефективними та піддаються ризику порушень та несанкціонованого втручання. Один із таких процесів - це голосування, яке є фундаментальним елементом демократичних суспільств. Традиційні методи голосування часто є недосконалими та піддаються фальсифікації. Незважаючи на постійні зусилля з їх покращення, досі зустрічаються виклики, пов'язані з безпекою, достовірністю та доступністю голосування.

Технологія блокчейн пропонує новий спосіб проведення голосування, який є більш безпечним та надійним - смарт-контракти. Даний підхід може бути використаний для створення голосування, що є стійким до фальсифікації, забезпечує конфіденційність та доступність для всіх учасників, та гарантує, що голоси будуть підраховані правильно.

Актуальність теми обумовлена важливістю голосування як елементу суспільства при потребі зробити певний вибір. Традиційні методи фізичного та електронного голосування залишаються вразливими перед загрозами (хакерські атаки, підробка голосів, порушення конфіденційності), мають низьку зручність процесів, та не пропонують універсальні рішення. Використання блокчейну та смарт-контрактів, поруч з гнучкою системою генерації голосувань, пропонує безпечний та ефективний підхід, відповідаючи сучасним потребам.

Метою дипломної роботи є покращення спостережності електронного голосування шляхом використання смарт-контрактів.

Задачі роботи:

- провести аналіз існуючих методів голосування, визначити їх переваги та недоліки
- оцінити економічний ефект розробки
- виконати математичний опис системи захищеного голосування
- розробити власний метод генерації смарт-контрактів для голосування, спрямований на підвищення ефективності та гнучкості

– реалізувати засіб генерації голосування на основі розробленого методу.
– здійснити тестування та отримати результати ефективності розробленого методу та засобу

Об'єктом дослідження є захист процесу голосування.

Предметом дослідження є метод та засіб генерування голосування.

Наукова новизна: отримано подальший розвиток методів генерування голосування на основі смарт-контрактів, що відрізняється від відомих високою адаптивністю до умов використання, що дозволило збільшити кількість потенційних застосувань до 160 варіацій.

Практична цінність розробки полягає в смарт-контрактах для генерування захищених голосувань, інтерфейсі користувача для взаємодії зі смарт-контрактами, та контрольному списку блокового тестування системи.

Результати отримані у магістерській кваліфікаційній роботі були представлені на L Науково-технічній конференції ФІТКІ 2021 року, Вінницького національного технічного університету [1].

1 АНАЛІЗ МЕТОДІВ ЗАХИЩЕНОГО ЕЛЕКТРОННОГО ГОЛОСУВАННЯ

1.1 Аналіз систем електронного голосування

Багато компаній, в якості своїх послуг та продукту, пропонують організаціям системи онлайн-голосування, щоб спростити їх процеси при потребі певного вибору.

Мова може йти про вибір нового старости класу, наступного генерального директора компанії або про голосування за важливе комерційне рішення. Системи онлайн-голосування - чудовий спосіб заощадити час, зусилля, гроші, та значно спростити організаційні процеси [2].

Система електронного голосування - це платформа, яка дозволяє членам організації голосувати в електронному вигляді, через веб-сайт, мобільний додаток або будь-який пристрій, підключений до Інтернету [3].

Через систему онлайн-голосування можна проводити різні види виборів. Наприклад, можна просто використовувати його для голосування більшістю, де перемагає варіант або кандидат, який набрав найбільшу кількість голосів. Або ж, можливо використовувати даний спосіб для більш складної системи голосування, наприклад пропорційного представництва, де кожен голос має вагу відповідно до переваг виборця [4].

Щоденне використання систем електронного-голосування включає [5]:

– вибори ради директорів;
– збори акціонерів;
– вибори правління товариства домовласників;
– голосування керівництва профспілки;
– вибори учнівського самоврядування.

Типовий сеанс онлайн-голосування виглядає так [4, 5]:

– виборець входить до системи голосування за допомогою свого унікального імені користувача та пароля;
– учасники обирають кандидатів або варіанти, за які хочуть голосувати;

– голосуючі подають свій голос, і система підраховує результати.

Серед переваг електронного голосування можна виділити [6-8]:

– Ефективність процесу та прийняття рішень: електронне голосування дозволяє швидко збирати голоси та приймати рішення в комерційних справах без необхідності проводити фізичні зустрічі. Завдяки онлайн-системі наявна можливість надіслати електронні бюлетені всім виборцям лише за кілька кліків. А коли період голосування закінчиться, система автоматично підрахує результати, без потреби робити це вручну, що заощадить організації багато часу та грошей.
– Покращена точність: ще одна перевага систем електронного-голосування полягає в тому, що вони, як правило, більш точні, ніж традиційні паперові системи. Завжди існує ймовірність людської помилки з паперовими бюлетенями, будь то неправильний підрахунок голосів або переплутування бюлетенів. Завдяки системі онлайн-голосування голоси підраховуються автоматично, що унеможливлює шанси на людську помилку та дає впевненість в точності результатів.
– Більша явка та залучення виборців: електронне голосування може підвищити явку виборців, оскільки їм буде зручніше голосувати онлайн, ніж брати участь в процесі фізично.

До недоліків електронного голосування, можна віднести [7-9]:

– Безпека: онлайн-голосування не такі безпечні, як традиційні паперові системи, так як зловмисники завжди можуть підробити результати. Подібні системи постійно піддаються загрозам кібербезпеки, і необхідно вживати заходів для захисту процесу голосування та інформації. Слід шукати систему, яка використовує шифрування для захисту даних, та яку перевірили незалежні експерти з безпеки.
– Анонімність та приватність: комерційне голосування може вимагати більш високого рівня анонімності та приватності, особливо коли йдеться про фінансові питання;
– Відсутність прозорості: при традиційному паперовому голосуванні виборці можуть бачити, як люди підраховують бюлетені. Але в

онлайн-голосуванні процес повністю електронний, що ускладнює перевірку результатів. Деякі системи надають сторінку з результатами виборів у реальному часі, де виборці можуть бачити результати, коли вони надходять.

Аналіз систем електронного голосування показує, що вони ефективні для проведення різних видів виборів та прийняття рішень в організаціях, проте вимагають уваги до питань безпеки, анонімності та приватності.

1.2 Аналіз загроз системам електронного голосування

В умовах швидкої цифрової трансформації, електронне голосування стає важливим інструментом для організацій та підприємств, які приймають стратегічні рішення через голосування в комерційних цілях. Такий підхід значно спрощує процеси прийняття рішень та зменшує витрати часу та ресурсів.

Проте, зростаюча цифрова залежність таких систем призводить до нових загроз і ризиків, які можуть підірвати їх ефективність та довіру до них [7].

Серед критичних загроз можна виділити:

– можливість хакерських атак на системи голосування;
– підробка голосів або втручання в процес голосування;
– загроза конфіденційності особистих даних голосуючих та результатів голосування;
– прослуховування голосів голосуючих та пов’язування їх з конкретними особами;
– можливість зловмисного впливу на процес голосування або на результати самого голосування зсередини системи.

Можливість хакерських атак на системи голосування представляє серйозний ризик для надійності і безпеки електронного голосування в комерційних цілях.

Хакери можуть запустити розподілені атаки на сервери системи голосування з метою їх перевантаження і припинення роботи. DDoS-атаки

можуть перервати доступ до системи голосування, що призводить до припинення процесу голосування та втрати часу [8].

Зловмисники можуть відправляти електронні листи або повідомлення, які видаватимуться за легітимні, і намагатися отримати облікові дані виборців, які потім можна використовувати для незаконного голосування або підробки голосів.

Хакери можуть спробувати запустити зловмисні програми в систему голосування для отримання несанкціонованого доступу та впливу на результати.

Підробка голосів або втручання в процес голосування становить серйозний ризик для легітимності та точності систем електронного голосування.

Існує ризик порушення ідентичності реальних виборців, які можуть бути використані для голосування без їхнього дозволу. Зловмисники можуть намагатися підмінити голоси реальних виборців або додати фіктивні голоси.

Можливість підробки або підміни голосів виборців, а також використання облікових даних реальних виборців для голосування без їхнього дозволу може призвести до недійсних голосів та порушити легітимність голосування.

Випадки підробки голосів, в свою чергу, можуть спричинити втрату довіри виборців до системи голосування.

Окрім цього, зловмисники можуть намагатися вплинути на рішення виборців, надсилаючи їм дезінформацію чи пропагуючи певні варіанти голосування [9].

Збереження конфіденційності особистих даних голосуючих та результатів голосування є важливою складовою систем електронного голосування.

Хакери можуть намагатися отримати доступ до бази даних, де зберігаються дані голосуючих та результати голосування. Несанкціонований доступ до особистих даних призводить до порушення приватності та може бути використаний для недобросовісних цілей.

Зловмисники можуть намагатися витягти конфіденційні дані та результати голосування шляхом порушення безпеки системи, після чого використовувати цю інформацію для маніпуляцій та інших умислів.

Прослуховування голосів голосуючих та зв'язку їх з конкретними особами надає зловмисникам можливість виявити, які особи проголосували за певних кандидатів або варіанти.

Процес голосування має бути анонімним, і будь-яке порушення цієї анонімності може вплинути на довіру до системи голосування.

Знання, які голоси належать конкретним особам, може бути використано для маніпуляцій або тиску на голосуючих.

Можливість зловмисного впливу на процес голосування або на результати самого голосування зсередини системи включає в себе дії осіб, які мають прямий доступ до системи голосування.

Особи, які мають доступ до системи голосування, можуть спробувати маніпулювати голосами або результатами голосування.

Випадки зловмисного впливу можуть призвести до підвищення сумнівів у чесності системи голосування та порушити довіру виборців [7-9].

1.3 Аналіз технології блокчейн

Блокчейн технологія відкриває нові можливості для реалізації безпечних та надійних систем голосування, зокрема для комерційних цілей.

Блокчейн - це розподілена база даних, що складається з ланцюжка блоків, кожен з яких містить кілька транзакцій. Кожен блок підтверджується мережею вузлів, і, коли новий блок додається до ланцюжка, він стає неможливим до змін. Це забезпечує надійність та стійкість до втручань [10].

Блокчейн гарантує незмінність, доступність, та анонімність. Це робить його привабливим для розгляду в якості платформи для створення електронного голосування та вирішення наявних проблем.

Однією з відомих властивостей та переваг блокчейну є децентралізація. Блокчейн працює по принципу розподіленої мережі вузлів, що сильно

ускладнює будь-які спроби втручання або маніпуляції зловмисників. Голосування може відбуватися без посереднього контролю централізованої організації.

Кожен голос або транзакція включається в блокчейн назавжди, і не може бути змінений чи видалений. Це забезпечує надійність результатів голосування та значно зменшує ризик підробки [11].

Блокчейн може забезпечити високий рівень анонімності для голосів учасників, дозволяючи їм залишати свої голоси без розкриття особистої інформації.

Всі транзакції в блокчейні є публічною інформацією, що дозволяє стежити за процесом голосування та перевіряти результати в режимі реального часу.

Блокчейн може забезпечити безпеку і легітимність комерційних голосувань, завдяки високому рівню захисту даних і прозорості процесу [12].

Використання блокчейн технології дозволяє проводити голосування швидше та ефективніше, зменшуючи час і ресурси, потрібні для організації голосувань [13].

Окремо слід приділити увагу смарт-контрактам. Смарт-контракти - це програми, які виконуються автоматично на блокчейні при виконанні певних умов. Вони можуть визначити логіку голосування та автоматизувати процеси, забезпечуючи чесність та точність [14].

Смарт-контракти можуть автоматично реєструвати та підраховувати голоси, гарантуючи точність і уникнення підробки. Також, дані програми на блокчейні можуть виконувати голосування за складними системами, такими як пропорційне представництво, де кожен голос має вагу відповідно до переваги виборця [15].

Смарт-контракти можуть забезпечувати автоматичне виявлення та реагування на незвичайні або недостовірні дії під час голосування.

Отож, технологія блокчейн та смарт-контракти мають значний потенціал для застосування в електронних голосуваннях, забезпечуючи безпеку, легітимність і ефективність процесу. Вони можуть бути ключовими

інструментами для захисту прав і інтересів учасників онлайн-голосувань та підвищення рівня довіри до них, відкриваючи широкі можливості для створення надійних, безпечних і чесних систем голосування [16].

1.4 Аналіз твірних шаблонів проєктування

Шаблони проектування - це рекомендації, які визначають структуру та організацію проекту з метою розв'язання певних проблем або виконання конкретних завдань. Вони надають загальні рекомендації та правила для створення програмного забезпечення або системи, що спрощує процес розробки [17].

Існує кілька видів шаблонів проєктування:

– Твірні - визначають процес створення об'єктів. Наприклад, шаблон "Singleton" використовується для створення класу, який має лише один екземпляр.
– Структурні - визначають структуру об'єктів та взаємодію між ними. Наприклад, шаблон "Adapter" використовується для з'єднання двох несумісних інтерфейсів.
– Поведінкові - визначають спосіб взаємодії об'єктів та способи обробки змін у стані. Наприклад, шаблон "Observer" дозволяє об'єктам сповіщати інші об'єкти про свої зміни [18].

В свою чергу, до твірних шаблонів проектування відпоносяться:

– Фабричний метод;
– Абстрактна фабрика;
– Будівельник;
– Одинак;
– Прототип.

Фабричний метод - визначає загальний інтерфейс для створення об’єктів у суперкласі, дозволяючи підкласам змінювати тип створюваних об’єктів.

Патерн Фабричний метод пропонує відмовитись від безпосереднього створення об’єктів, замінивши його викликом особливого фабричного методу.

Може бути застосований у випадках відсутності попередньої інформації про типи і залежності об’єктів, надання можливість іншим розширювати частини існуючої бібліотеки, потреби в економії системних ресурсів та повторного використання вже створених об’єктів.

Абстрактна фабрика - дає змогу створювати сімейства пов’язаних об’єктів, не прив’язуючись до конкретних класів створюваних об’єктів.

Патерн Абстрактна фабрика пропонує виділити загальні інтерфейси для окремих продуктів, що складають одне сімейство, і описати в них спільну для цих продуктів поведінку.

Може бути застосований у випадках потреби працювати з різними видами пов’язаних один з одним продуктів, та як ступінь еволюції фабричного методу, коли чергові зміни потребують створення нових типів.

Будівельник - дає змогу створювати складні об’єкти крок за кроком, а також можливість використовувати один і той самий код будівництва для отримання різних відображень об’єктів.

Патерн Будівельник пропонує винести конструювання об’єкта за межі його власного класу, доручивши цю справу окремим об’єктам, які називаються будівельниками. Шаблон пропонує розбити процес конструювання об’єкта на окремі кроки, а при потребі створити об’єкт - по черзі викликати методи будівельника, до того ж, лише ті, що необхідні для певної конфігурації.

Одинак - гарантує, що клас має лише один екземпляр, та надає глобальну точку доступу до нього.

Всі реалізації Одинака зводяться до того, аби приховати типовий конструктор та створити публічний статичний метод, який і контролюватиме життєвий цикл об’єкта-одинака. При наявності доступу до класу одинака можна бути впевненим в доступі до його статичного методі, а отже і в незмінності створюваного об’єкту при будь-якій точці виклику.

Прототип - дає змогу копіювати об’єкти, не вдаючись у подробиці їхньої реалізації.

Патерн Прототип доручає процес копіювання самим об’єктам, які треба скопіювати. Він вводить загальний інтерфейс для всіх об’єктів, що

підтримують клонування. Метод створює новий об’єкт поточного класу й копіює в нього значення всіх полів власного об’єкта [17].

У контексті розробки системи електронного голосування, використання твірних шаблонів проектування може бути дуже корисним.

Шаблони дозволяють додавати новий функціонал без необхідності редагувати вже наявний код системи голосування. Це робить розвиток системи більш гнучким і дозволяє швидко впроваджувати нові можливості голосування.

Використання патернів дозволяє легко створювати різні види голосувань, включаючи голосування більшістю, пропорційне голосування та інші, без переписування коду системи.

Твірні шаблони спрощують процес створення нових об'єктів та відділяють створення об'єктів від їх подальшого використання, що допомагає запобігти помилкам при роботі з об'єктами голосування [17, 18].

Твірні шаблони дозволяють створювати окремі компоненти, які можна використовувати в інших проектах або навіть в межах одного проекту для різних видів голосувань.

1.5 Постановка задачі

Аналіз систем електронного голосування виявив, що ці системи мають значні переваги, такі як ефективність у прийнятті рішень, точність підрахунку голосів, покращені зручність та залучення виборців. Однак, вони також супроводжуються загрозами безпеки, анонімності та приватності.

На основі проведеного аналізу, та для досягнення мети дипломної роботи, було вирішено поєднати переваги електронного голосування та технології блокчейн.

Також, слід приділити увагу гнучкості використання голосувань та смарт-контрактів, та розробити метод генерації екземплярів голосування, що забезпечить значну кількість можливих варіацій голосування.

Крім того, в цілях підвищення безпеки системи голосувань та смарт-

контрактів, необхідно розробити правила розмежування прав доступу до функціоналу системи.

1.6 Висновки з розділу

В даному розділі було проведено аналіз систем електронного голосування, технології блокчейн, шаблонів проектування, а також визначено ключові задачі для розробки системи захищеного електронного голосування.

Досліджено існуючі системи електронного голосування та виявлено їхні переваги та недоліки. Системи електронного голосування мають великий потенціал для поліпшення виборчих процесів, але вимагають вирішення проблем безпеки та довіри.

Розглянуто потенційні загрози та вразливості систем електронного голосування. Безпека є критичним аспектом розробки системи та засобів онлайн-голосування, і потребує ретельного аналізу та заходів її забезпечення.

Висвітлено основні переваги технології блокчейн для створення систем електронного голосування, такі як децентралізація, незмінність та анонімність. Блокчейн може стати ключовим елементом безпечних та надійних систем голосування.

Розглянуто важливість використання твірних шаблонів проектування для створення системи електронного голосування.

Сформульовано задачі, необхідні для реалізації системи захищеного електронного голосування.

2 МЕТОД ГЕНЕРУВАННЯ СМАРТ-КОНТРАКТІВ ДЛЯ ЗАХИЩЕНОГО ЕЛЕКТРОННОГО ГОЛОСУВАННЯ

2.1 Математичний опис системи захищеного голосування

Блокчейн гарантує незмінність, доступність, та безпеку. З допомогою смарт-контрактів можна гарантувати автоматичне прийняття та підрахунок голосів, точність та відсутність помилок внаслідок людського фактору, підтримку складних процедур голосування, захист від стороннього зловмисного впливу, та як наслідок - довіру до системи голосування та впевненість в чесності та достовірності процесу.

Враховуючи всі переваги даних технологій, було вирішено виконати розробку системи захищеного електронного голосування на основі смарт-контрактів.

Процес онлайн-голосування за допомогою смарт-контракту передбачає попереднє створення даного контракту, що планується виконувати автоматично та з детальним налаштуванням параметрів, та подальшим викликом функцій що представляють відповідні можливості голосування [14].

Вхідні дані системи захищеного електронного голосування:

- можливість перегляду голосів учасників: відкрите або закрите голосування (disc);
- багаторазовість: одноразове або повторюване голосування (rep);
- наявність ваги голосу певного учасника: демократичне або рейтингове голосування (pow);
- обмеження кількості учасників: максимальна кількість учасників голосування (lim);
- обмеження можливості участі в голосуванні: приватне або публічне голосування (priv);
- спосіб підрахунку голосів та результату: мажоритарне, з певною часткою голосів необхідною для перемоги (part);
- наявність права вето (veto);

– кількість виборів в голосуванні: так або ні, більше 2 виборів (choice);
– можливість фінансування: надання учасникам голосування суми криптовалюти, необхідної для покриття витрат на подання голосу (fund).

Розглянувши можливі входи, отримано наступну множину:

$$I = \{disc, rep, pow, lim, priv, part, veto, choice, fund\} \quad (2.1)$$

Окремий смарт-контракт голосування, при наявності відповідних налаштувань, буде можливо активувати, що дозволить розпочати голосування, та вимкнути для його завершення.

Перетворення, а саме процес голосування включає в себе наступні етапи:

– прийняття параметрів голосування, з усіма необхідними даними, від модуля генерації голосувань (accept);
– запуск голосування (start);
– процес подання голосів учасниками (vote);
– закінчення голосування та підрахунок голосів (finish);
– оприлюднення результатів (result).

Розглянувши можливі перетворення, отримано таку множину:

$$T = \{accept, start, vote, finish, result\} \quad (2.2)$$

Результати, а тобто вихідні дані в процесі генерування та проведення голосування містять:

– адреса екземпляру голосування на блокчейні (addr);
– переможний вибір голосування (win);
– кількість голосів переможного вибору (voteW);
– загальна кількість поданих голосів (voteT).

В результаті, отримано наступну множину:

$$O = \{addr, win, voteW, voteT\} \quad (2.3)$$

Як результат, при виборі параметрів для генерування смарт-контракту голосування можливо створити аж 160 унікальних комбінацій голосування, з можливістю гнучко налаштувати створюване голосування під власні потреби та специфіку використання, а також безпечно та ефективно провести сам процес голосування.

2.2 Узагальнений опис методу

Враховуючи кількість імовірних варіантів окремого створеного голосування, та подальшу можливість масштабування та додавання нових типів параметрів, є досить недоцільним реалізувати кожну з варіацій вручну.

Подібний прямолінійний підхід є дуже неоптимізованим по очікуваних затратах часу, ресурсів, кількості ітерацій та повторюваних процесів. Окрім цього, існує великий ризик помилки, некоректного імплементації функціоналу, людського фактору. Важко уявити кількість окремих класів, величину кодової бази, документації та потребу в безлічі зайвих повторюваних дій в випадку необхідності додати бодай один додатковий параметр при даному непрактичному підході.

На допомогу приходять твірні шаблони проектування, які вклають в себе: фабричний метод, абстрактна фабрика, будівельник, прототип. Використання патернів дозволяє легко та гнучно створювати всі можливі та підтримувані види голосувань, уникаючи при цьому копіювання коду, величезних затрат часу та ресурсів, та високої вірогідності помилки. При цьому даний підхід надає безмежні можливості масштабування та дозволяє в подальшому легко редагувати вже наявні параметри [17].

Враховуючи кількість можливих варіацій голосування, та специфіку блокчейну, для проектування методу генерування смарт-контрактів захищеного голосування було обрано патерн Будівельник [18].

Процес генерування смарт-контрактів голосування можна розділити на наступні кроки:

1) автентифікація адміністратора, вповноваженого до налаштування параметрів та підготовки до генерування нового голосування;
2) налаштування параметрів голосування відповідно до підтримуваних типів голосування;
3) генерування нового смарт-контракту голосування;
4) автентифікація адміністратора, вповноваженого до перевірки встановлених параметрів та підготовки до початку голосування;
5) автентифікація адміністратора, вповноваженого до запуску голосування, активація смарт-контракту голосування.

В результаті, метод передбачає можливість швидко, безпечно, компактно, масштабовано, гнучко, та без ризику помилки, налаштувати, створити, та запустити смарт-контракт електронного голосування.

2.3 Архітектура засобу генерування смарт-контрактів

Архітектура засобу є дуже важливою, коли мова йде про розуміння роботи системи, легкість в знаходженні помилок та їх виправленні, легкість масштабування та, що найголовніше, про безпеку, стійкість, та ефективність засобу (рис. 2.1).

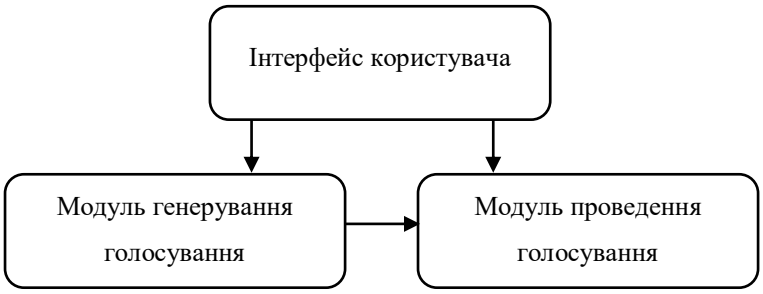


Рисунок 2.1 – Архітектура засобу генерування смарт-контрактів

Враховуючи зазначені важливі аспекти побудови структури системи, а також обраний шаблон проектування, можна розділити систему електронного голосування на два структурні модулі: модуль генерування голосування та модуль проведення голосування [15].

Модуль генерування голосування – це окремий смарт-контракт, що є батьківським в системі, може створювати екземпляри іншого смарт-контракту (рис. 2.2). Даний модуль дозволяє гнучко налаштовувати параметри

голосування, відповідно до підтримуваних типів, та створювати нові екземпляри смарт-контракту голосування (рис. 2.3).

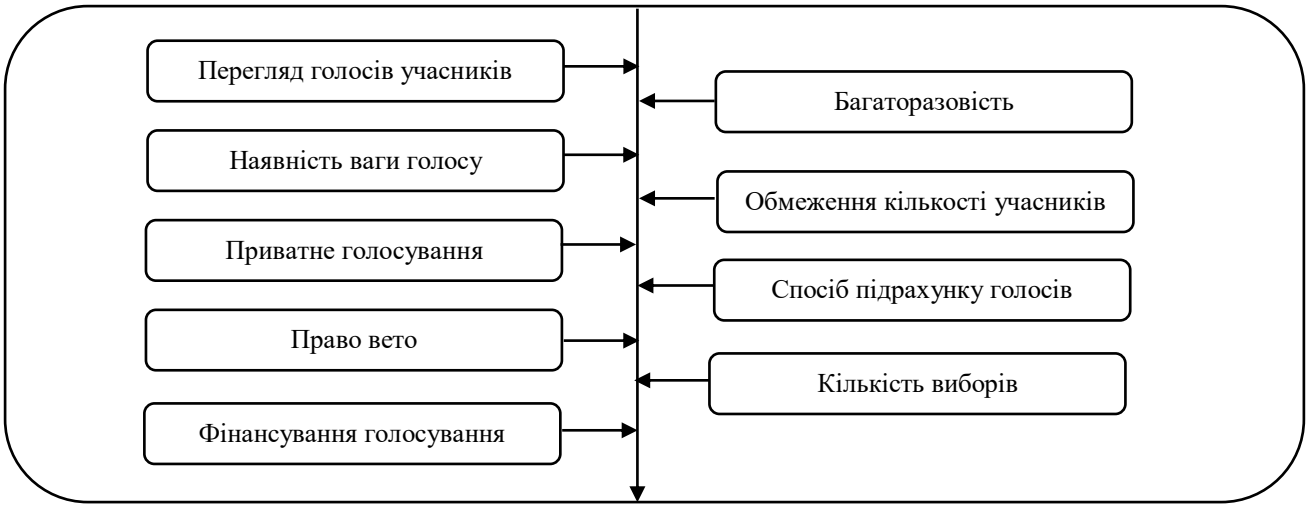


Рисунок 2.2 – Візуалізація вибору параметрів генерування голосування

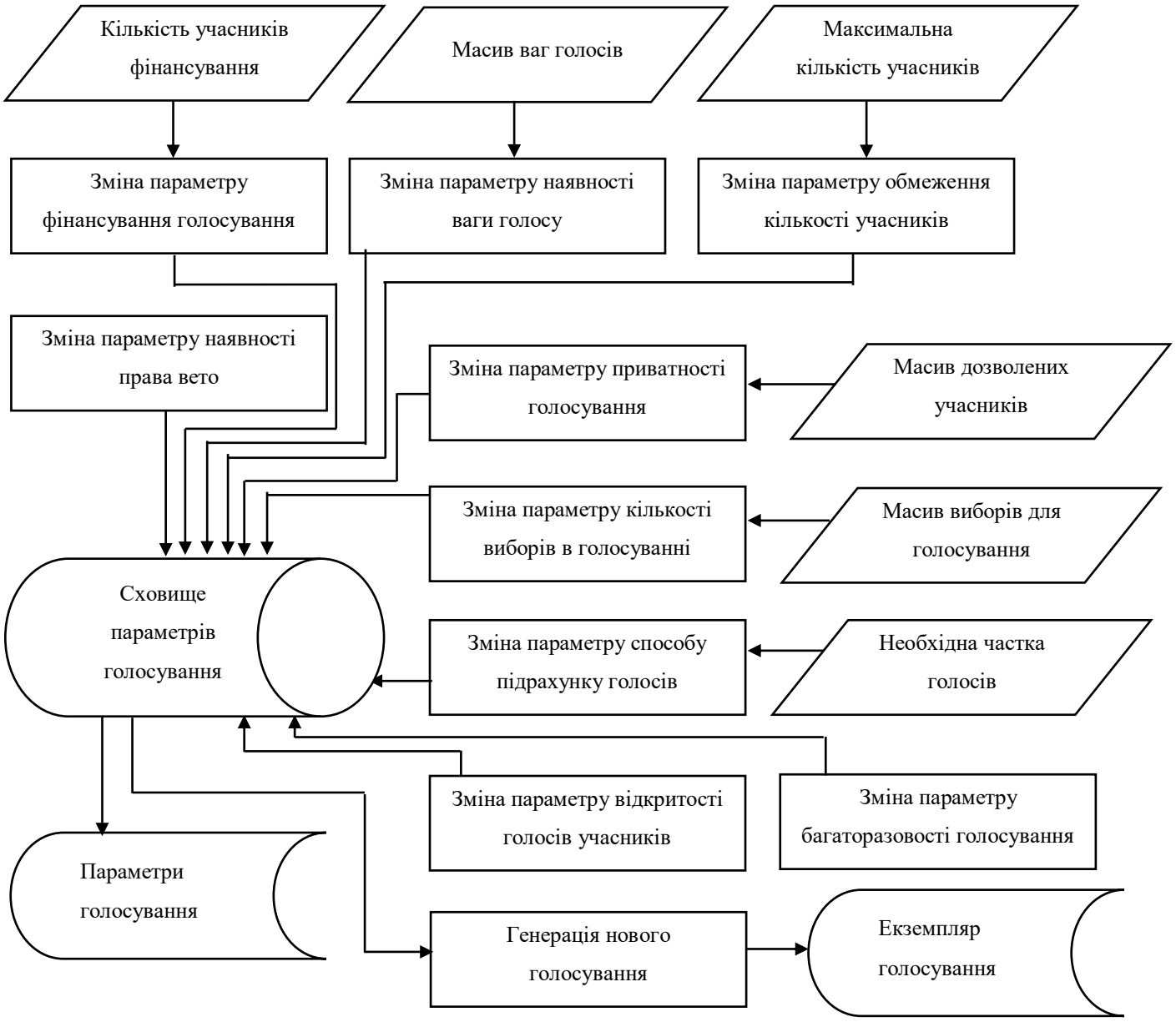


Рисунок 2.3 – Модуль генерування голосування

Модуль проведення голосування: окремий смарт-контракт, що є дочірнім в системі, тобто його екземпляри генеруються іншим смарт-контрактом, модулем генерування. Однак, даний модуль є головним по значенню, так як саме він вміщує в себе повний функціонал голосування, з усіма можливими його типами (рис. 2.4).

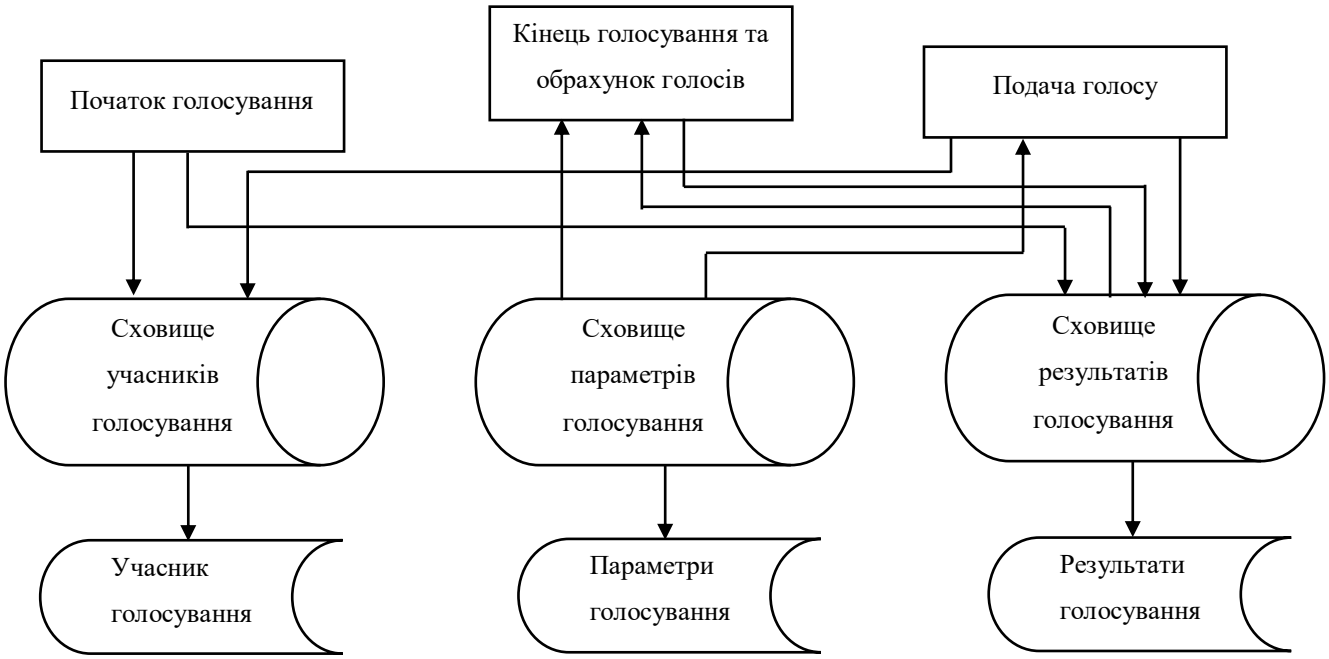


Рисунок 2.4 – Модуль проведення голосування

Модуль голосування являє собою смарт-контракт, що містить в собі функціонал усіх підтримуваних типів голосування. При створенні екземпляру, він приймає параметри налаштування від модуля генерування.

Відповідно до налаштувань, конкретний екземпляр смарт-контракту голосування буде мати встановлені властивості, тобто частина його функціоналу буде задіяна в його роботі, а інша частина, що не була активована параметрами - заблокована для використання [16].

Планується, що розроблена система електронного голосування, серед своїх переваг буде мати користувацький інтерфейс, для покращення зручності використання засобу.

Втім, для виконання учасником будь-якої зміни стану блокчейну, в нашому випадку - подання голосу, необхідно підписати транзакцію, для чого потрібно мати в наявності на балансі своєї адреси на блокчейні достатню суму криптовалюти, а саме рідної монети блокчейну, в якому виконується дія [20].

В цілях покращення комфорту учасників та їх досвіду використання системи електронного голосування, було вирішено при їх реєстрації як учасників голосування за допомогою певного екземпляру смарт-контракту, давати можливість автоматично надавати їм в власність певну кількість рідної криптовалюти блокчейну, достатню для подальшої можливості подати свій голос [21].

В результаті, єдина дія, яку потрібно буде виконати учаснику голосування для подачі свого голосу – підключити свій гаманець для можливості взаємодії зі смарт-контрактом, що є невід'ємною частиною роботи з децентралізованими застосунками.

2.4 Правила розмежування прав доступу до смарт-контрактів голосування

Підготовка та використання системи голосування та смарт-контрактів електронного голосування, як її складових, передбачає виконання ряду визначених критично важливих дій та налаштувань.

До таких дій можна віднести:

– налаштування параметрів майбутнього нового голосування в модулі генерації голосування;
– розгортання нового екземпляру контракту голосування;
– перевірка параметрів та даних створеного нового голосування;
– запуск голосування;
– подача голосів;
– зупинка голосування;
– оприлюднення результатів.

Міри безпеки системи голосування та смарт-контрактів передбачають використання ролей користувачів та розмежування прав доступу до виконання певних дій по підготовці та використанню системи голосування [22].

Можливо виокремити основі ролі та їх зони відповідальності:

– власник: має право призначати та відчужувати інші ролі, передавати власність;
– адміністратор: має доступ до налаштування параметрів майбутнього нового голосування, розгортання нового екземпляру контракту голосування, та перевірки даних новоствореного голосування;
– оператор: має доступ до запуску та завершення голосування, отримання даних про учасників;
– учасник голосування: має доступ до подачі голосів, та отримання результатів голосування [23].

Було розглянуто процеси підготовки та використання системи голосування та смарт-контрактів, та на їх основі виконано розмежування прав доступу за ролями користувачів, що значно підсилює безпеку та ефективність процесу голосування та відповідає стандартам безпеки та захисту інформації.

2.5 Висновки з розділу

В даному розділі було виконано математичний опис системи захищеного електронного голосування, виокремлено вхідні дані системи та кроки процесу голосування. Виділено та наведено підтримувані варіації та типи голосування.

Досліджено проблеми процесу розробки та генерування смарт-контрактів електронного голосування, розглянуто рішення проблеми в вигляді застосування твірного шаблону проектування. Наведено ключові кроки створення екземплярів смарт-контракту голосування згідно з обраним патерном.

Розроблено архітектуру засобу генерування голосувань, виконано поділ системи на ключові функціональні модулі, описано їх призначення, а також продумано заходи та особливості системи з метою покращення досвіду використання засобу.

Проведено аналіз критично важливих дій в системі, сформульовано правила розмежування прав доступу до смарт-контрактів системи голосування, розроблено основі ролі та їх зони відповідальності.

3 ЗАСІБ ГЕНЕРУВАННЯ СМАРТ-КОНТРАКТІВ

3.1 Узагальнений алгоритм роботи засобу

Програмна реалізація розробленого засобу генерування голосування включає в себе три основі структурні модулі: інтерфейс користувача, смарт-контракт генерування голосування, смарт-контракт проведення голосування.

Інтерфейс користувача є відправною точкою роботи учасника процесу голосування з застосунком. Для забезпечення підготовки до коректної взаємодії з смарт-контрактами на блокчейні реалізовано можливість підключити криптовалютний web3-гаманець до сторінки веб-інтерфейсу.

При успішному підключенні до інтерфейсу користувача, стає наявною можливість обрати смарт-контракт для взаємодії, що відобразить доступний функціонал обраного модуля.

Спосіб взаємодії користувача зі смарт-контрактами на блокчейні подібний до схеми взаємодії в клієнт-серверній архітектурі, та включає два сценарії.

В першому – користувач за допомогою інтерфейсу здійснює виклик функції смарт-контракту з метою зміни стану блокчейну, а в якості відповіді отримує інформацію про новостворену транзакцію. Отримана відповідь, окрім іншої технічної інформації, містить дані про успішність виконаної зміни, згенеровані події, повернені дані, та помилки, якщо такі стались.

В другому сценарії – користувач надсилає запит на отримання даних зі сховища смарт-контракту. Як результат, користувач отримує в відповідь запитувані дані, якщо вони присутні, або ж помилку, якщо запит було здійснено некоректно або не вистачає повноважень згідно вимог смарт-контракту.

Відповідно, користувач при взаємодії зі смарт-контрактами надсилає виклики та запити в блокчейн, а після їх обробки та відповіді – отримує результат відображеним на сторінці інтерфейсу.

Важливою властивістю такого підходу є стійкість до несанкціонованого копіювання [1], оскільки кожен екземпляр смарт-контаркту однозначно ідентифікується адресою, а тому голосування запущене зловмисником буде мати іншу адресу смарт-контракту на блокчейні.

Першим кроком алгоритму взаємодії з модулем генерування голосування є підготовка налаштувань - користувач почергово надсилає запити на зміну стану параметрів бажаного голосування. При цьому деякі з параметрів потребують передачі від користувача додаткових даних, необхідних для точного встановлення налаштувань, як то максимальна кількість учасників, список дозволених учасників, або ж список доступних виборів в майбутньому голосуванні.

Для коректної зміни стану параметрів, користувачу необхідно надавати дані в корекній формі та виконати умови зміни параметрів відповідно до актуального стану інших налаштувань. В разі невідповідності запиту вимогам смарт-контракту, усі зміни скасовуються, а користувач повідомляється про причину відмови.

Після закінчення процесу зміни стану параметрів майбутнього голосування, користувач може пересвідчитись в їх коректності виконавши запит на отримання актуального стану налаштувань. В разі невідповідності актуального стану параметрів бажаним для голосування налаштуванням – користувач може змінити необхідні йому параметри. Даний процес є ітеративним та може повторюватись аж до моменту досягнення необхідних налаштувань.

При умові досягнення мети коректного налаштування параметрів майбутнього голосування, користувач має можливість виконати генерацію бажаного голосування. В разі такого запиту, смарт-контракт виконає розгортку нового екземпляру смарт-контракту голосування на блокчейні, враховуючи усі налаштовані раніше параметри голосування, які також будуть передані в сам створений екземпляр голосування. В разі, якщо користувач задав параметр фінансування голосування як позитивний, йому також буде необхідно, разом з

транзакцією створення нового екземпляру, надати необхідну суму криптовалютних активів.

Останнім кроком взаємодії з модулем генерування голосування залишається отримати адресу новоствореного екземпляру смарт-контракту голосування на блокчейні для подальшого його використання (рис. 3.1).

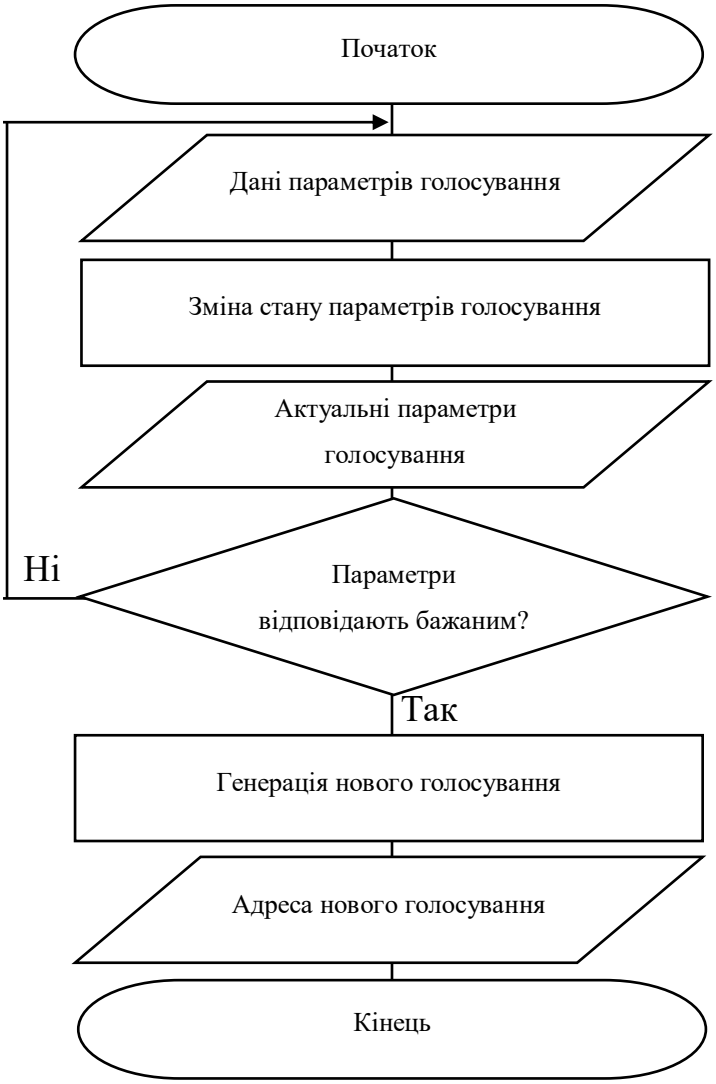


Рисунок 3.1 – Алгоритм взаємодії з модулем генерування голосування

Далі доцільно розглянути алгоритм взаємодії з модулем проведення голосування. Для виконання дій з новим екземпляром смарт-контракту голосування необхідно в першу чергу перемкнути інтерфейс в режим роботи з проведенням голосування, після чого використати отриману адресу відповідного голосування, та вставити її в відповідне поле користувацького інтерфейсу. В результаті інтерфейс підключиться до смарт-контракту проведення голосування за наданою йому адресою на блокчейні.

Першим кроком взаємодії з новим голосуванням є перевірка встановлених параметрів для пересвідчення в їх коректності та готовності до проведення процесу голосування.

В разі коректності налаштувань голосування відповідальна особа може виконати запит на старт процесу голосування. При успішному виклиці даного функціоналу, учасники голосування матимуть можливість подати свій голос.

Після відкриття можливості подачі голосів, учасники голосування можуть виконати запит реєстрації голосу, передавши номер свого вибору. В залежності від налаштованих параметрів голосування, учасники можуть отримати різний результат спроби свого воєвиявлення: помилка про досягнення ліміту голосів або відсутність дозволу подачі голосу певного учасника, успішна подача звичайного голосу, врахування сили голосу певного учасника.

В момент потреби закінчення голосування, оператор виконує запит на завершення процесу подачі голосів. Дана дія забороняє можливість подачі голосу, автоматично підраховує голоси та записує результати завершеного голосування, відповідно до встановлених параметрів.

І фінальним кроком процесу проведення голосування є оприлюднення результатів (рис. 3.2).

Рисунок 3.2 – Алгоритм взаємодії з модулем проведення голосування

В результаті, було розроблено загальний алгоритм роботи засобу генерування смарт-контрактів для захищеного голосування, реалізовано підключення користувача до інтерфейсу та до самих смарт-контрактів, описано алгоритми взаємодії користувача з головними функціональними модулями, та закладено ключовий функціонал кожного з смарт-контрактів системи.

3.2 Процедура налаштування параметрів голосування

Налаштування параметрів голосування є ключовим процесом модуля генерування голосування, адже саме ці дії дозволяють досягти гнучкості та високої варіативності можливих створених екземплярів голосувань.

Налаштування деяких параметрів вимагає високої уваги до безпеки, врахування взаємодії з іншими налаштуваннями та особливостей роботи смарт-контрактів та блокчейну як платформи для їх виконання. Доцільно перейти до налаштування деяких важливих параметрів детальніше.

Параметр фінансування голосування – вимагає в якості вхідних даних кількість учасників голосування, витрати яких на подачу голосів будуть профінансовані. Даний параметр для його активації потребує активованих параметрів максимальної кількості учасників та приватного голосування, при цьому кількість профінансованих учасників повинна відповідати кількості дозволених учасників, для забезпечення контролю фінансування та коректної передачі криптовалютних активів на конкретні адреси (рис. 3.3).

Параметр сили голосу – вимагає в якості вхідних даних масив сил голосу кожного учасника, в числовому представленні. Даний параметр для його активації потребує активованих параметрів максимальної кількості учасників та приватного голосування, при цьому кількість елементів в масиві сил голосу повинна відповідати кількості дозволених учасників, для забезпечення відповідності певної сили голосу певному учаснику. Сила голосу повинна бути не менше 1, а деактивація цього параметру очищає збережний в параметри масив сил голосу для можливості повторної активації (рис. 3.4).

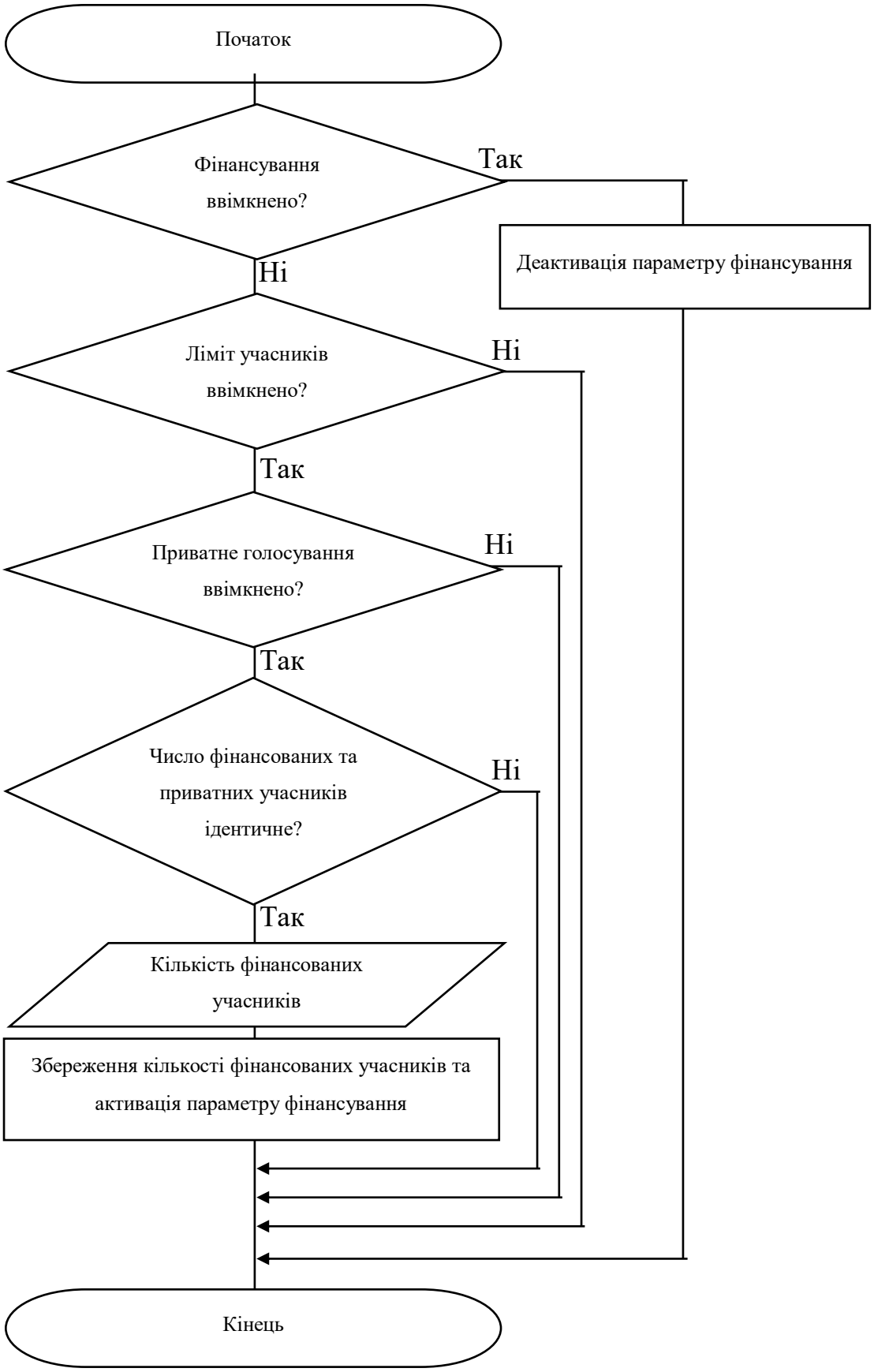


Рисунок 3.3 – Процедура налаштування параметру фінансування

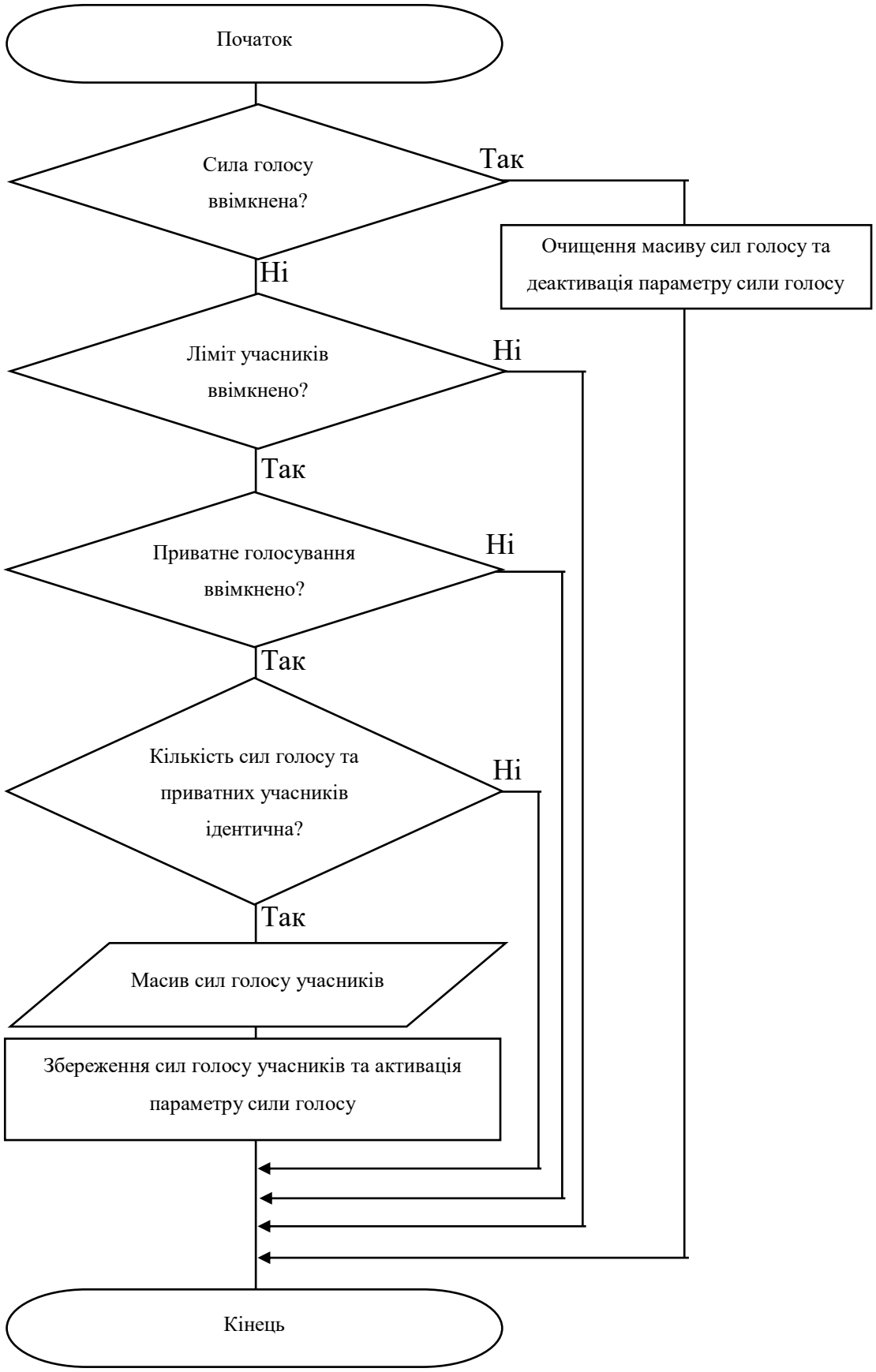


Рисунок 3.4 – Процедура налаштування параметру сили голосу

Параметр ліміту кількості учасників – вимагає в якості вхідних даних число максимальної кількості учасників голосування, при активації зберігає надане число в параметри голосування, в свою чергу при деактивації – даний параметр також деактивує параметри фінансування, сили голосу, приватного голосування в цілях уникнути можливих конфліктів параметрів та ризиків збою системи.

Параметр приватного голосування - вимагає в якості вхідних даних масив адрес дозволених учасників. При активації даний параметр потребує аби кількість дозволених учасників була обмежена, в цілях контролю над ситуацією при роботі на блокчейні, а також не була більшою ніж загальний ліміт учасників. В результаті активації надані дозволені адреси зберігаються в параметрах голосування, а при деактивації очищаються, а також деактивуються параметри фінансування та сили голосу, по причині їх залежності від активного параметру приватного голосування.

Параметр способу підрахунку голосів – потребує в якості вхідних даних число порогу, що буде необхідним для досягнення для вибору в голосуванні, аби стати переможним. Даний параметр вимагає аби наданий поріг був числом від 0 до 100, та якщо число від 0 до 50 включно – встановалює підрахунок результату за мажоритарним принципом – перемагає вибір що набрав більшість голосів, а якщо число від 51 до 100 – наданий поріг встановалюється як необхідний для досягення для перемоги (рис. 3.5).

Параметр кількості виборів в голосуванні – вимагає в якості вхідних даних масив назв виборів, що будуть запропоновані для подачі голосів. Даний параметр потребує обмеження можливої кількості виборів, в цілях контролю поведінки та забезпення стабільності смарт-контракту голосування. У випадку, якщо було надано масив з 0 або 1 елементу, автоматично встановлюються вибори Так/Ні в параметри голосування, а параметр кількісного вибору вважається деактивованим. Якщо ж надано масив виборів довжиною в 2 і більше елементи – надані елементи встановлюються як вибори для налаштовуваного голосування, а саме воно вважається голосуванням з багатьма виборами.

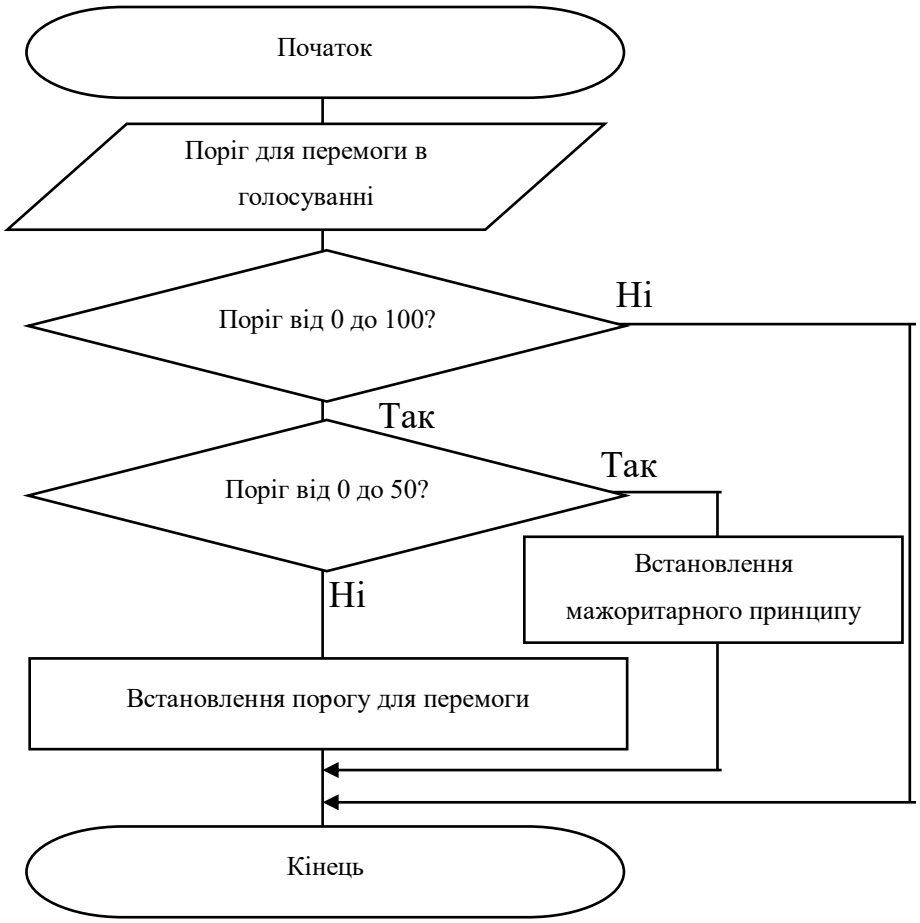


Рисунок 3.5 – Процедура налаштування параметру підрахунку голосів

В результаті було розроблено процедури налаштування різних параметрів голосування, з урахуванням їх взаємодії між собою, а також особливостей логіки голосування та процесу виконання смарт-контрактів.

3.3 Процедура генерування голосування

Після закінчення процесу налаштування голосування та отримання задовільних заданих параметрів, наступним кроком є генерування нового екземпляру смарт-контракту голосування.

Основною вимогою до можливості створення нового голосування є наявність заданого параметру виборів для даного екземпляру. В залежності від інших заданих налаштувань процес генерації нового голосування може вимагати виконання додаткових умов та дій.

В разі активного параметру фінансування процесу подачі голосів для учасників, процес генерування потребує передачі, разом з транзакцією генерації

нового екземпляру, суми криптовалютних активів для проведення фінансування.

Якщо передана сума криптовалюти більша 0 – починається процес розподілення коштів та відправки їх на адреси дозволених учасників, тобто їх фінансування. Загальна сума переданих активів рівнозначно розподіляється між визначеними учасниками та переказується на їх збережені адреси. Відповідальність за контроль загальної суми криптовалюти та достатньою її кількістю для подальшої можливості усіх учасників подати свій голос лежить безпосередньо на вповноваженій особі, що виконує запит на створення нового голосування, оскільки необхідна сума для оплати комісії блокчейну для подачі голосу постійно змінюється, залежить від актуальних даних блокчейну, та може бути вирахувана лише поза блокчейном (рис. 3.6).

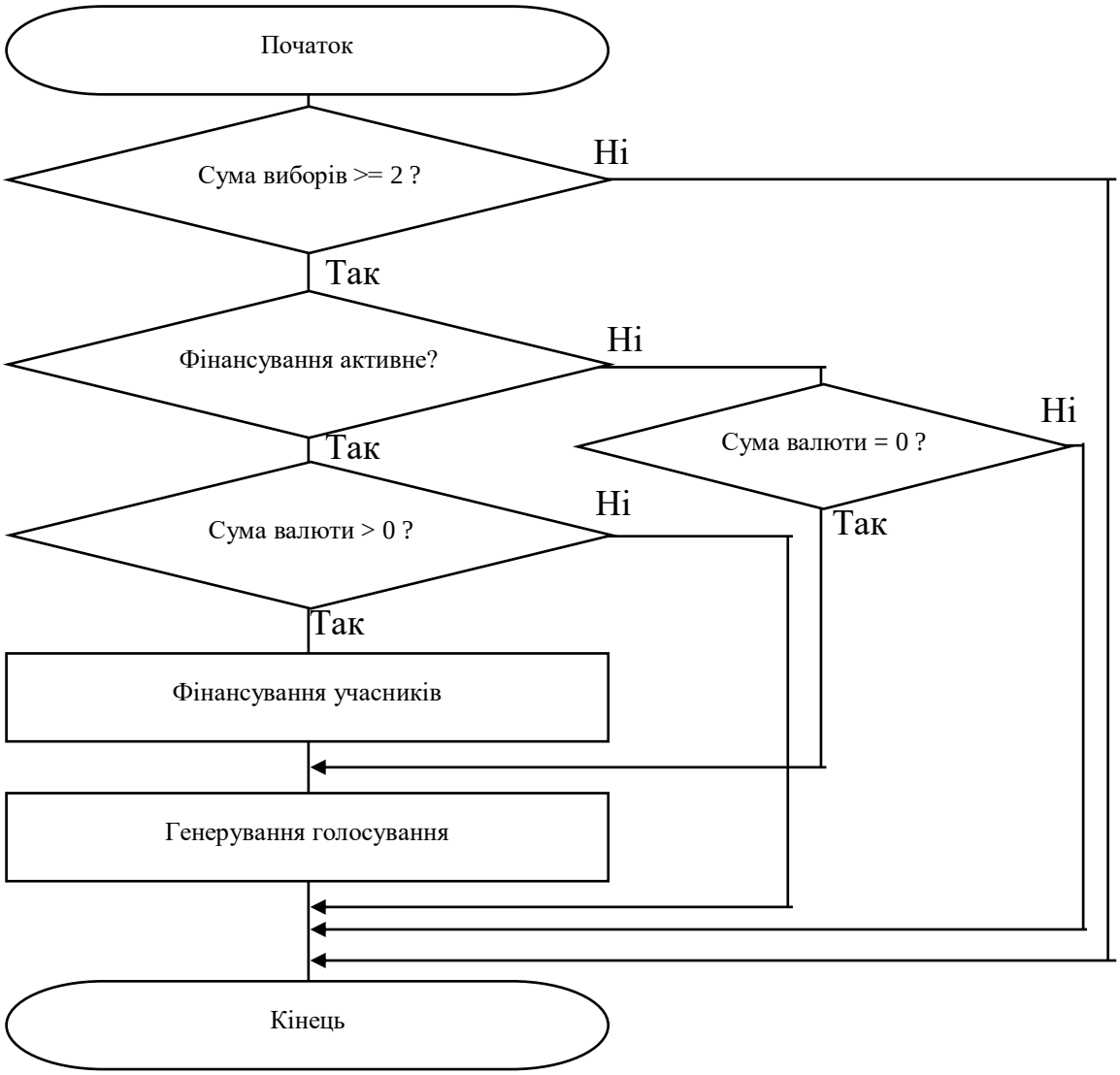


Рисунок 3.6 – Процедура генерування голосування

За іншим сценарієм, якщо параметр фінансування деактивовано – контракт навпаки вимагатиме аби сума переданих разом з транзакцією криптовалютних активів була рівна 0, адже в разі передачі криптовалюти без її розподілення між учасниками – вона просто залишиться на балансі смарт-контракту генерування голосування, звідки її буде неможливо ніколи повернути.

І фінальним кроком даного процесу є безпосередня розгортка нового екземпляру смарт-контракту голосування на блокчейні, та передача параметрів голосування в його конструктор.

В результаті, отримано новий екземпляр голосування з налаштованими параметрами, та можливістю дістати адресу нового смарт-контракту для подальшої взаємодії.

3.4 Процедура старту голосування

Маючи адресу смарт-контракту створеного екземпляру голосування, є доступною можливість перевірити налаштування параметрів, отриманих від модуля генерування. При умові, що екземпляр голосування було створено коректно з відповідними налаштуваннями – можна переходити до проведення голосування.

Процедура старту голосування є першим кроком в процесі проведення голосування, та потребує аби статус активації обраного екземпляру був негативним.

Також, в процесі виконання запиту на старт, враховується параметр повторюваності голосування. Якщо голосування одноразове – контракт перевіряє чи не було голосування вже виконане, і якщо ні – активує можливість подачі голосів. В разі ж багаторазового голосування – смарт-контракт враховує сценарій, при якому старт голосування не є першим за час його існування. Для забезпечення можливості повторного використання конкретного екземпляру голосування, передбачено при повторному запуску обнуляти статус голосу кожного учасника що брав участь в минулій ітерації, очищати загальну

кількість голосів, а також кількість голосів за кожен вибір та переможний результат минулого голосування. І лише після підготовки до повторного використання, активується можливість подачі голосів (рис. 3.7).

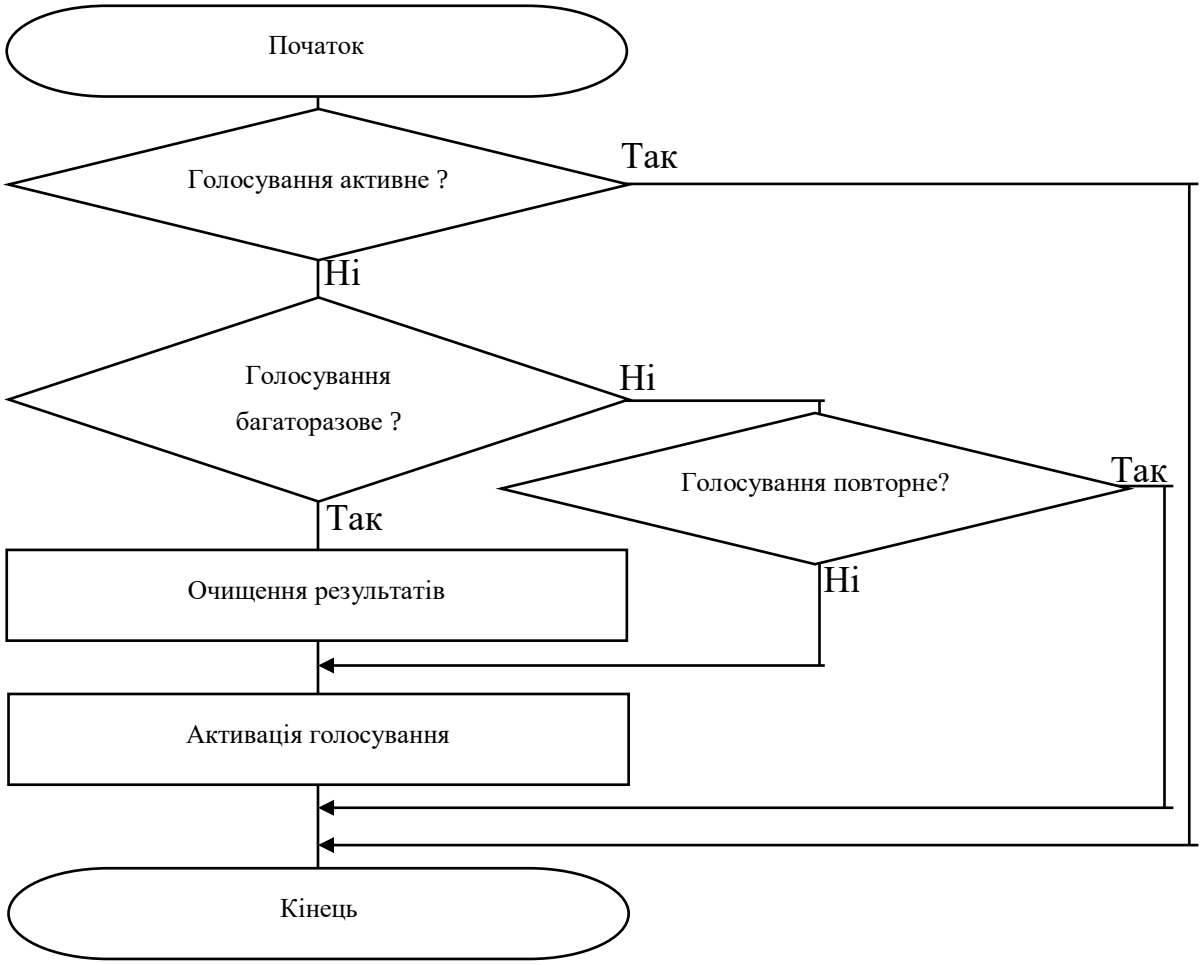


Рисунок 3.7 – Процедура старту голосування

В результаті розроблено процедуру старту голосування, що враховує можливість повторного використання одного екземпляру, та активує можливість подачі голосів учасниками.

3.5 Процедура подачі голосів

В разі успішної активації процесу голосування, стає доступною для учасників можливість подачі голосів.

Дана процедура вимагає передачі в якості вхідного параметру порядкового номера вибору, за який учасник бажає віддати свій голос. Також, процедура перевіряє чи активне голосування на момент спроби подачі голосу,

чи подавав вже даний учасник свій голос в даній ітерації голосування, та чи входить наданий номер вибору до списку наявних виборів в параметрах голосування.

Окрім цього, якщо голосування має ліміт учасників – смарт-контракт перевіряє чи не перевищує ліміт спроба подати голос учасником. Якщо голосування приватне – процедура перевіряє чи входить учасник-відправник голосу до списку дозволених учасників (рис. 3.8).

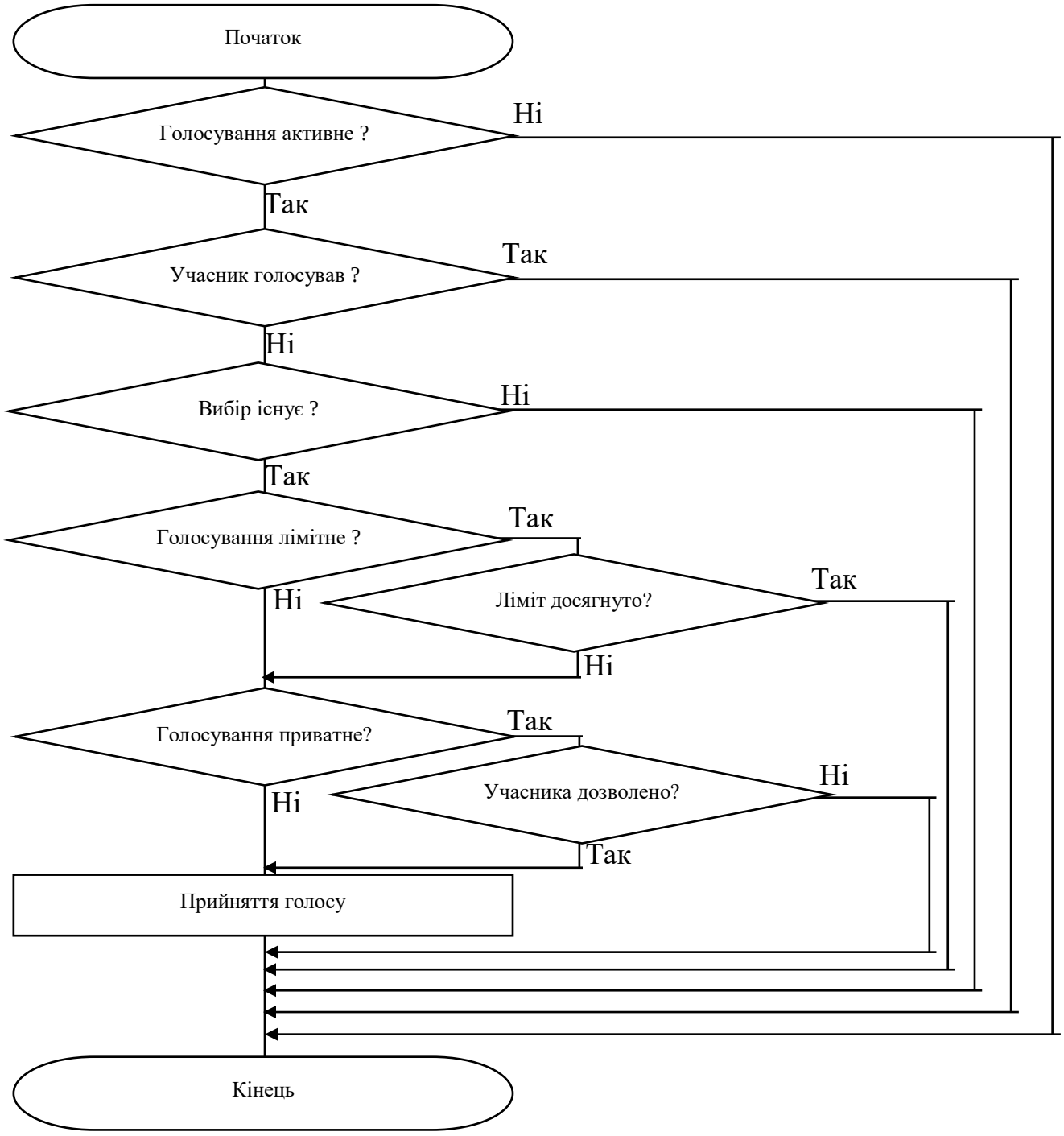


Рисунок 3.8 – Процедура подачі голосу

В разі виконання всіх вимог, учасник позначається як той що подав свій голос, а його вибір зберігається в сховищі. В разі якщо параметр сили голосу активовано – сила голосу учасника враховується при підрахунку загальної кількості голосів та голосів за кожен доступний вибір.

В результаті, розроблено процедуру подачі голосу, що пильно слідкує за коректністю процесу, дотриманням усіх вимог зазначених в параметрах, та зберігає голос учасника.

3.6 Процедура завершення голосування та підрахунку результатів

В будь-який момент після активації голосування наявна можливість його завершити та запустити підрахунок результатів. Завершення голосування вимагає аби воно було активовано.

При завершенні голосування, процедура підраховує отримані на цей момент голоси учасників, визначає максимальну кількість голосів за один вибір, вибір що отримав найбільше голосів, та чи існує декілька таких виборів.

На основі обрахованих даних, смарт-контракт обирає переможний вибір. Однак, процедура передбачає ряд сценаріїв, при яких переможний вибір не буде визначено в результаті проведення голосування: якщо переможних виборів більше 1, не було подано жодного голосу, було застосовано право вето, жоден вибір не досяг необхідного порогу голосів від загальної кількості.

В разі ж, якщо було подано хоча б один голос, переможний вибір лише один, вибір отримав 100% голосів при активному параметрів права вето, вибір досяг необхідного порогу голосів при активному параметрі необхідного порогу, або ж просто переміг при інших сценаріях налаштувань голосування – даний вибір та його кількість голосів зберігаються в результатах, а можливість подачі голосів деактивується (рис. 3.9).

В будь-який момент голосування кожен учасник може переглянути доступні для подачі голосу вибори, загальну кількість голосів, а також кількість голосів що вже подано за кожен вибір. Після деактивації подачі голосів

результати голосування також будуть містити переможний вибір та його кількість голосів.

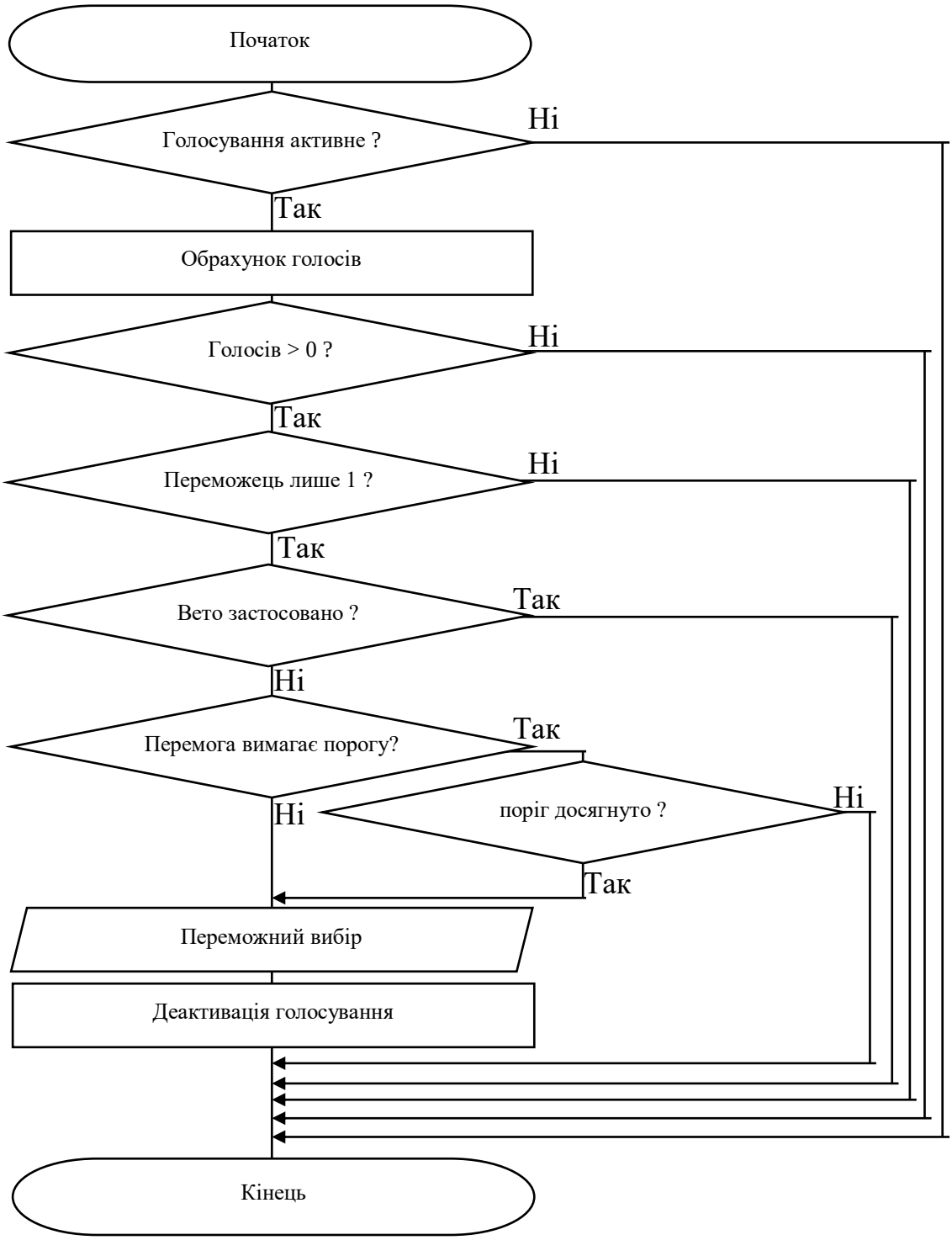


Рисунок 3.9 – Процедура завершення голосування та підрахунку результатів

В результаті, було розроблено процедуру завершення голосування та підрахунку результатів голосування, з урахуванням усіх імовірних сценаріїв результатів та варіацій налаштувань параметрів.

3.7 Процедури розмежування прав доступу

В цілях підвищення безпеки смарт-контракту, контролю над його поведінкою та виконанням на блокчейні, та попередження несанкіонованого доступу до функціоналу критично важливо розробити рольову модель системи генерування голосування.

Оскільки обидва модулі системи, як генерування голосування, так і проведення голосування, потребують наявності моделі ролей, її розроблено в окремому смарт-контракті, від якого наслідуються головні смарт-контракти системи, та в результаті обоє містять функціонал рольової моделі.

В рольову модель входять три ролі: власник, адміністратор, оператор. Власник є головною роллю в системі та задається при розгортці екземпляру смарт-контракту в системі як адреса, що виконала розгортку. Інші ролі не є призначеними одразу при розгортці смарт-контракту, та навіть власник не є одразу адміністратором та оператором.

Рольова модель містить наступні процедури: передача власництва, призначення адміністратора, призначення оператора, відчуження адміністратора, відчуження оператора. Окрім основних процедур, смарт-контракт рольової моделі також дозволяє переглянути власника, та перевірити чи є певна адреса адміністратором або оператором.

І найважливішими та результуючими елементами рольової моделі є модифікатори доступу. Використання модифікаториів доступу в якості параметрів функцій дозволяє чітко контролювати доступ до виклику певних процедур.

Модифікатори доступу існують відповідно до наявних ролей, та кожен з них передбачається бути застовованих до певних процедур системи:

– onlyOwner - обмежує виклики лише від власника, застосовується до процедур призначення та відчуження ролей, передачі власництва;
– onlyAdmin - обмежує виклики лише від адміністратора, застосовується до процедур перегляду параметрів голосування, налаштування усіх

параметрів голосування, генерації нового екзмепляру голосування, та отримання адреси нового екземпляру;

– onlyOperator - обмежує виклики лише від оператора, застосовується до процедур перегляду активності голосування, перегляду інформації певного учасника, старту та завершення голосування.

В результаті, було розроблено рольову модель для застосування в смарт-контрактах системи, що дозволяє покращити безпеку, контроль над діями та поведінкою системи, захист від несанкіонованого доступу до функціоналу. Саме особливість модифікаторів мови програмування Solidity дозволила ефективно здійснити контроль розмежування прав доступу.

3.8 Висновки з розділу

В даному розділі було розроблено загальний алгоритм роботи засобу генерування смарт-контрактів для захищеного голосування та описано алгоритми взаємодії користувача з головними функціональними модулями.

Також, було розроблено процедури налаштування різних параметрів голосування, генерації нового екземпляру голосування, старту голосування, процедуру подачі голосу, завершення голосування та підрахунку результатів, та розроблено рольову модель для застосування в смарт-контрактах системи.

4 ТЕСТУВАННЯ ЗАСОБУ ГЕНЕРУВАННЯ СМАРТ-КОНТРАКТУ

4.1 Обгрунтування засобів розробки та тестування

Вибір засобів розробки та тестування є досить важливим в процесі реалізації алгоритму та процедур засобу генерування голосування.

Remix IDE - це інтегроване середовище розробки для смарт-контрактів на мові Solidity, яке працює у веб-браузері. До його переваг можна віднести широкі можливості тестування, що дозволяють тестувати та відлагоджувати смарт-контракти безпосередньо у середовищі розробки.

Remix включає в себе різноманітні інструменти, такі як статичний аналіз, що допомагають виявляти потенційні помилки у коді ще до виконання контракту, а також має широкі можливості компіляції - дозволяє налаштовувати оптимізацію при компіляції коду смарт-контрактів, переглядати ABI та байт-код [24].

Remix дозволяє підключатись до різноманітних локальних, тестових, та публічних блокчейнів. Також надає зручну візуалізацію підключених адрес, балансів, параметрів транзакції, розгортки та імпорту смарт-контрактів, виклику функцій та перегляду повернень, консоль для більш детальної інформації [25].

Visual Studio Code (VS Code) є одним із найпопулярніших текстових редакторів у розробці програмного забезпечення. Для розробки смарт-контрактів на мові Solidity у VS Code використовується плагін Solidity, який надає додаткові можливості.

VS Code має велику кількість розширень, що дозволяє інтегрувати його з різними інструментами для роботи з блокчейном та смарт-контрактами [26]. Плагін Solidity розширює можливості VS Code, дозволяючи використовувати автодоповнення, перевірку синтаксису, відлагодження коду та інші корисні функції спеціально для Solidity [27].

Truffle та Ganache - це два інструменти, які широко використовуються для розробки, тестування та впровадження смарт-контрактів на блокчейні Ethereum.

Truffle надає набір інструментів для розробки, тестування та взаємодії з Ethereum. Включає в себе компіляцію, відлагодження, розгортання контрактів, а також надає можливість писати та запускати автоматизовані тести для смарт-контрактів та відлагоджувати їх для виявлення помилок [28].

Ganache - це осередок блокчейну Ethereum, який дозволяє створювати та використовувати власний локальний блокчейн для розробки та тестування. Завдяки локальному блокчейну, розгортання та взаємодія з контрактами відбувається швидше, що сприяє швидкій розробці та тестуванню [29].

Hardhat є засобом розробки та тестування смарт-контрактів на блокчейні Ethereum, а Mocha - це популярний фреймворк для написання тестів на JavaScript. Поєднуючи Hardhat з Mocha, отримуємо потужний інструментарій для розробки, тестування та відлагодження смарт-контрактів.

Hardhat надає багато корисних інструментів для розробки смарт-контрактів, включаючи компіляцію, тестування, відлагодження та розгортання контрактів. Також, забезпечує можливість створення локального блокчейну для тестування контрактів, що допомагає в розробці та тестуванні без реального блокчейну [30].

Jest - це популярний фреймворк для тестування у JavaScript, який надає можливості для написання й виконання різноманітних тестів. Він відомий своєю простотою, швидкодією та зручністю використання, має багато корисних функцій для тестування, таких як автоматична підтримка запуску й організації тестів, можливість визначення тестів, використання різних типів асерцій [31].

Mocha - це популярний фреймворк для написання тестів, який надає гнучкість у структурі та організації тестових сценаріїв. Підтримує різні стилі написання тестів, такі як BDD (Behavior-Driven Development) та TDD (Test-Driven Development) [32].

Поєднуючи Hardhat з Mocha, отримується потужний інструмент для автоматизованого тестування смарт-контрактів на Ethereum, що дозволяє швидко й ефективно створювати, тестувати та вдосконалювати контракти. Hardhat надає базову інфраструктуру для розробки, в той час як Mocha забезпечує можливість створення різноманітних тестів, що допомагає виявляти

помилки та забезпечувати високу якість продукту. Виходячи з переваг даного поєднання, було обрано саме Hardhat та Mocha для подальшої реалізації засобу.

Серед інших використовуваних засобів розробки можна виокремити мову програмування для Ethereum - Solidity. Solidity - це об’єктно-орієнтована високорівнена мова програмування, спеціально створена для Ethereum Virtual Machine (EVM) та розробки смарт-контрактів на блокчейні Ethereum. Solidity, поміж іншого, має статичну типізацію, підтримує успадкування, бібліотеки та складні типи, визначені користувачем [33].

4.2 Розгортка та тестування смарт-контрактів

Розробка смарт-контрактів вимагає максимальної точності, продуманості, уваги до безпеки та захисту від можливих атак, оскільки ці програмні засоби є незмінними після розгортки на публічному блокчейні.

Одним з ключових методів забезпечення якості та надійності розроблених смарт-контрактів є розробка та виконання блокових функціональних та безпекових тестів.

Розроблену систему смарт-контрактів в розрізі тестування можна розділити на чотири основні модулі: генерування голосування, проведення голосування, рольова модель в генеруванні голосування, та рольова модель в проведенні голосування (рис. 4.1).

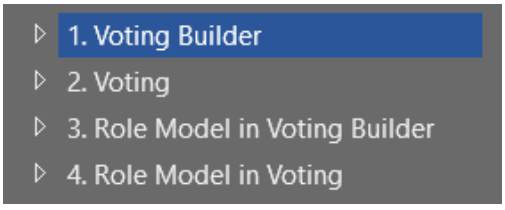


Рисунок 4.1 – Основні модулі тестування

Кожен з наведених модулів містить менші блоки, що необхідно покрити ефективними тестами на функціональність та безпеку.

Модуль генерування голосування містить наступні блоки для тестування:

– загальні параметри голосування;

– параметр ліміту учасників;

– параметр розголошення даних учасника;

– параметр сили голосу;

– параметр права вето;

– параметр відсотка для перемоги;

– параметр багаторазовості голосування;

– параметр приватності голосування;

– параметр кількості виборів в голосуванні;

– параметр фінансування голосування;

– створення голосування.

На основі наявних блоків модуля генерування голосування створено тест-план та заголовки тест-кейсів для подальшого виконання (рис. 4.2).

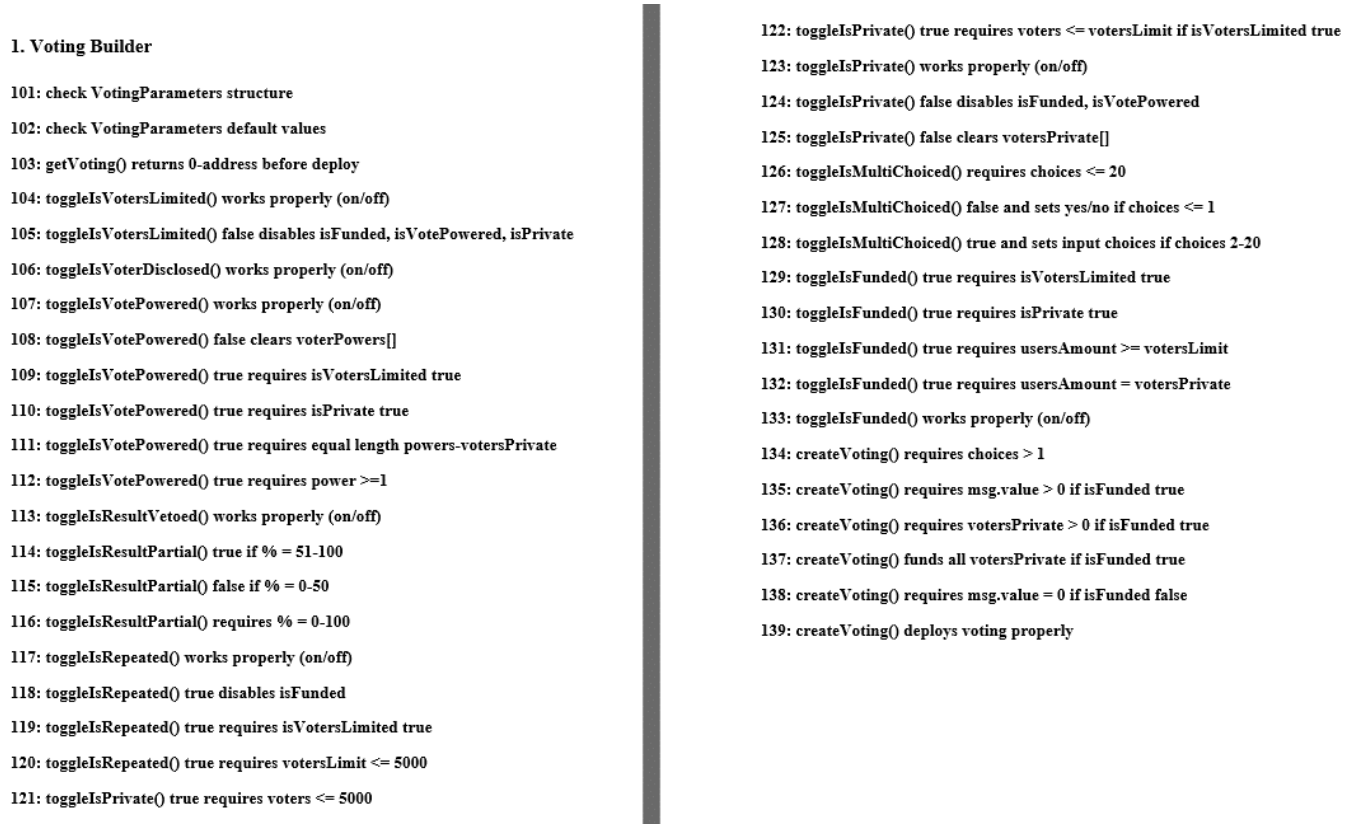


Рисунок 4.2 – Тест-план модуля генерування голосування

Розроблені тест-кейси реалізовано за допомогою середовища Hardhat та фреймворку Mocha, та отримано результати автоматизованого тестування блоків модуля генерування голосування, що підтверджують його коректність функціонування та обробку усіх необхідних умов та запобіжників (рис. 4.3).

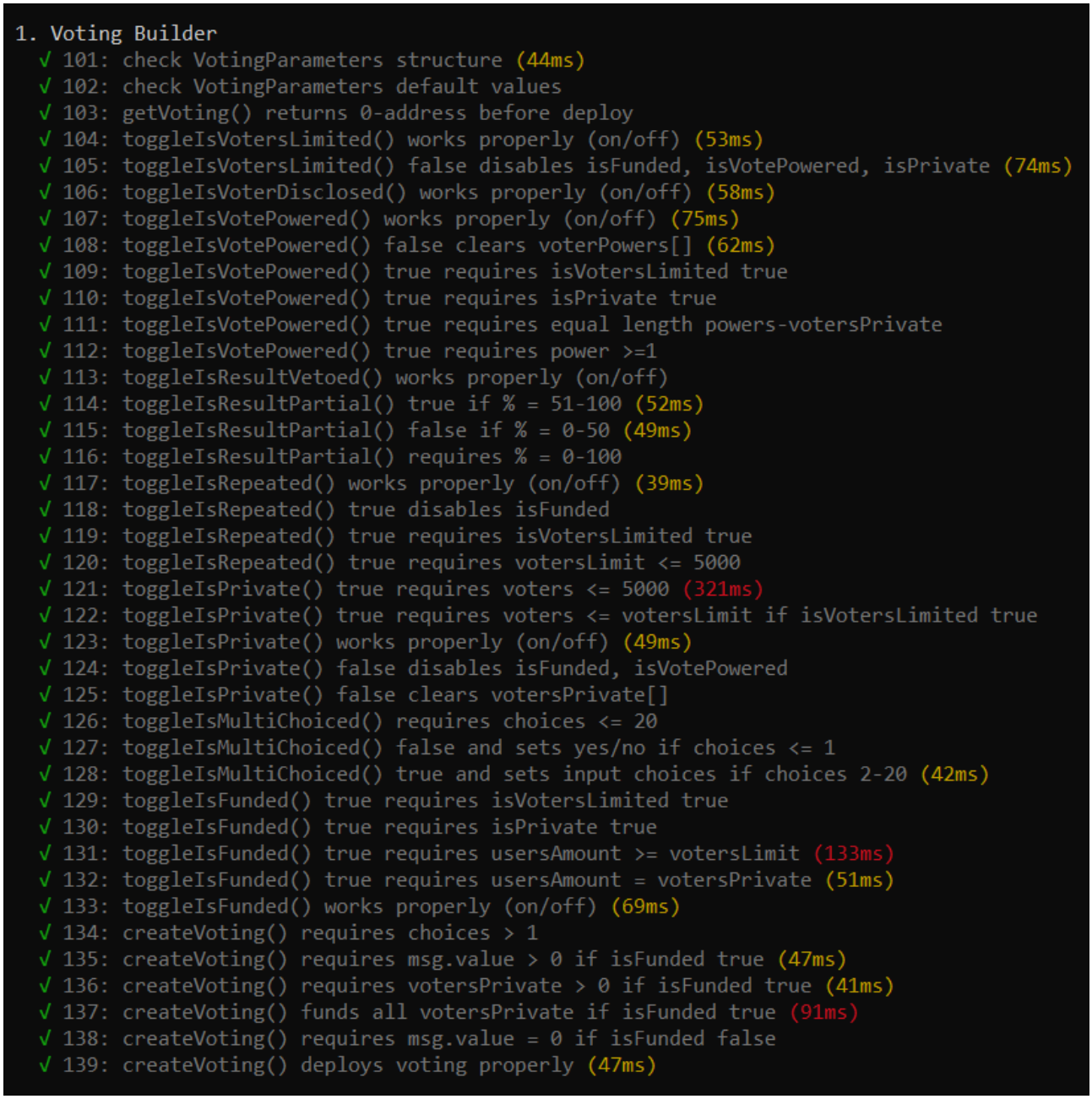


Рисунок 4.3 – Результати тестування модуля генерування голосування

Наступний модуль для тестування – проведення голосування, містить наступні блоки для тестування:

– параметри голосування;

– результати голосування;

– учасник голосування;

– обробка параметрів при розгортці екземпляру;

– подача голосів;

– старт голосування;

– завершення голосування та обрахунок результатів.

На основі наявних блоків модуля проведення голосування створено тест-план та заголовки тест-кейсів для подальшого виконання (рис. 4.4).

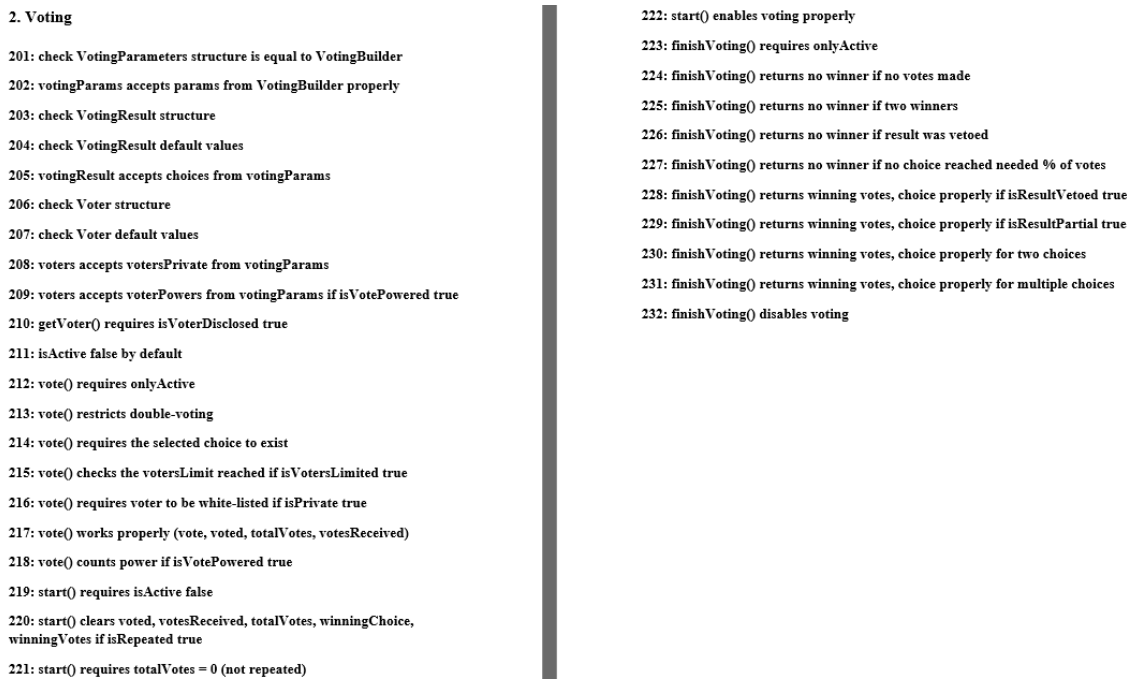


Рисунок 4.4 – Тест-план модуля проведення голосування

Розроблені тест-кейси реалізовано та отримано результати автоматизованого тестування блоків модуля проведення голосування, що підтверджують його коректність функціонування та обробку усіх необхідних умов та запобіжників (рис. 4.5).

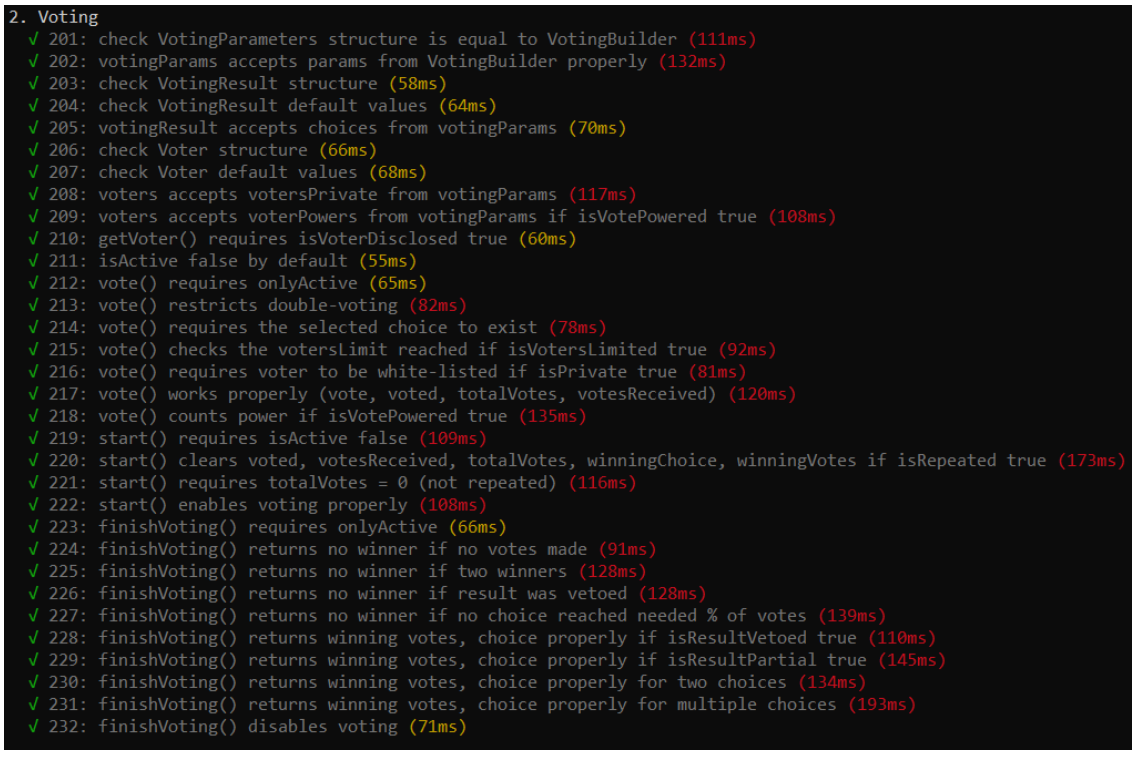


Рисунок 4.5 – Результати тестування модуля проведення голосування

Наступний модуль для тестування – рольова модель в генеруванні голосування, містить наступні блоки для тестування:

– загальні параметри голосування;

– власник смарт-контракту;

– параметри голосування та адміністратор;

– присвоєння та відчуження ролей;

– створення голосування.

На основі наявних блоків модуля рольової моделі в генеруванні голосування створено тест-план та заголовки тест-кейсів для подальшого виконання (рис. 4.6).

3. Role Model in Voting Builder
301: getVotingParams() requires onlyAdmin
302: getVoting() requires onlyAdmin
303: owner was set properly
304: owner is not operator by default
305: owner is not admin by default
306: owner cannot transfer Ownership to himself
307: transferOwnership() requires onlyOwner
308: transferOwnership() works properly
309: toggleIsVotersLimited() requires onlyAdmin
310: toggleIsVoterDisclosed() requires onlyAdmin
311: toggleIsVotePowered() requires onlyAdmin
312: toggleIsResultVetoed() requires onlyAdmin
313: toggleIsResultPartial() requires onlyAdmin
314: toggleIsRepeated() requires onlyAdmin
315: toggleIsPrivate() requires onlyAdmin
316: toggleIsMultiChoiced() requires onlyAdmin
317: toggleIsFunded() requires onlyAdmin
318: grantRoleAdmin() requires onlyOwner
319: grantRoleAdmin() requires address is not admin
320: grantRoleOperator() requires onlyOwner
321: grantRoleOperator() requires address is not operator
322: revokeRoleAdmin() requires onlyOwner
323: revokeRoleAdmin() requires address is admin
324: revokeRoleOperator() requires onlyOwner
325: revokeRoleOperator() requires address is operator
326: createVoting() requires onlyAdmin

Рисунок 4.6 – Тест-план модуля рольової моделі в генеруванні голосування

Розроблені тест-кейси реалізовано та отримано результати автоматизованого тестування блоків модуля рольової моделі в генеруванні голосування, що підтверджують його коректність функціонування та обробку усіх необхідних умов та запобіжників (рис. 4.7).

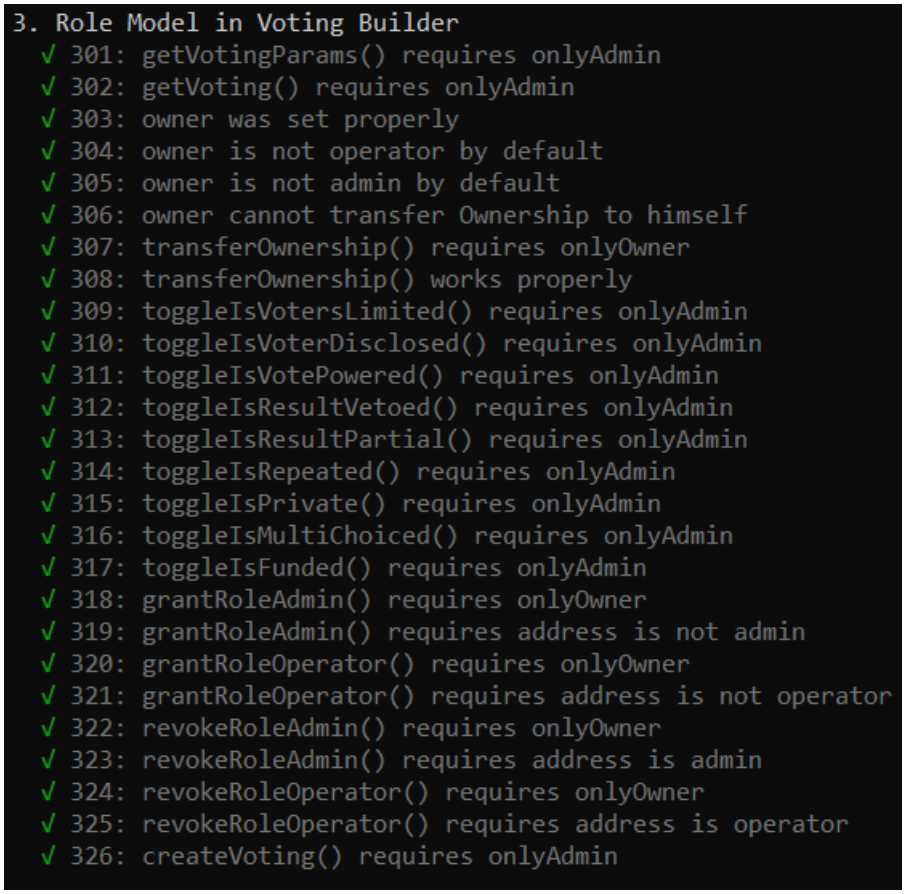


Рисунок 4.7 – Результати тестування модуля рольової моделі в генеруванні голосування

І останній модуль для тестування – рольова модель в проведенні голосування, містить наступні блоки для тестування:

– загальні параметри голосування;

– власник смарт-контракту;

– запуск/закінчення голосування та оператор;

– присвоєння та відчуження ролей.

На основі наявних блоків модуля рольової моделі в проведенні голосування створено тест-план та заголовки тест-кейсів для подальшого виконання (рис. 4.8).

Розроблені тест-кейси реалізовано та отримано результати автоматизованого тестування блоків модуля рольової моделі в проведенні голосування, що підтверджують його коректність функціонування та обробку усіх необхідних умов та запобіжників (рис. 4.9).

4. Role Model in Voting

401: getVotingParams() requires onlyAdmin

402: getVoter() requires onlyOperator

403: owner was set properly

404: getIsActive() requires onlyOperator

405: owner is not admin by default

406: owner is not operator by default

407: owner cannot transfer Ownership to himself

408: transferOwnership() requires onlyOwner

409: transferOwnership() works properly

410: start() requires onlyOperator

411: grantRoleAdmin() requires onlyOwner

412: grantRoleAdmin() requires address is not admin

413: grantRoleOperator() requires onlyOwner

414: grantRoleOperator() requires address is not operator

415: revokeRoleAdmin() requires onlyOwner

416: revokeRoleAdmin() requires address is admin

417: revokeRoleOperator() requires onlyOwner

418: revokeRoleOperator() requires address is operator

419: finishVoting() requires onlyOperator

Рисунок 4.8 – Тест-план модуля рольової моделі в проведенні голосування

```
4. Role Model in Voting
  √ 401: getVotingParams() requires onlyAdmin (51ms)
  √ 402: getVoter() requires onlyOperator (54ms)
  √ 403: owner was set properly (48ms)
  √ 404: getIsActive() requires onlyOperator (49ms)
  √ 405: owner is not admin by default (48ms)
  √ 406: owner is not operator by default (49ms)
  √ 407: owner cannot transfer Ownership to himself (103ms)
  √ 408: transferOwnership() requires onlyOwner (58ms)
  √ 409: transferOwnership() works properly (56ms)
  √ 410: start() requires onlyOperator (62ms)
  √ 411: grantRoleAdmin() requires onlyOwner (78ms)
  √ 412: grantRoleAdmin() requires address is not admin (58ms)
  √ 413: grantRoleOperator() requires onlyOwner (57ms)
  √ 414: grantRoleOperator() requires address is not operator (60ms)
  √ 415: revokeRoleAdmin() requires onlyOwner (59ms)
  √ 416: revokeRoleAdmin() requires address is admin (50ms)
  √ 417: revokeRoleOperator() requires onlyOwner (59ms)
  √ 418: revokeRoleOperator() requires address is operator (50ms)
  √ 419: finishVoting() requires onlyOperator (77ms)


116 passing (18s)
```

Рисунок 4.9 – Результати тестування модуля рольової моделі в проведенні голосування

В результаті, було розроблено та прогорамно реалізовано блокове тестування основних модулів системи смарт-контрактів генерування голосування, з увагою на функціональні вимоги, та вимоги безпеки та контролю доступу. Результати показали високу ефективність, надійність, стійкість смарт-контрактів, та відповідність сучасним вимогам безпеки та контролю доступу до програмних засобів на блокчейні.

4.3 Тестування веб-інтерфейсу

Для зручної взаємодії зі смарт-контрактами системи та сприянні масового розповсюдження технології слід розробити інтерфейс користувача.

Однією з вимог до інтерфейсу користувача є зручність підключення до блокчейну. Розроблена веб-сторінка дозволяє користувачу легко підключитись до інтерфейсу та блокчейну використовуючи найпопулярніші web3-криптогаманці, наприклад Metamask, та автоматично підвантажує адресу користувача на блокчейні для подальших дій (рис. 4.10).

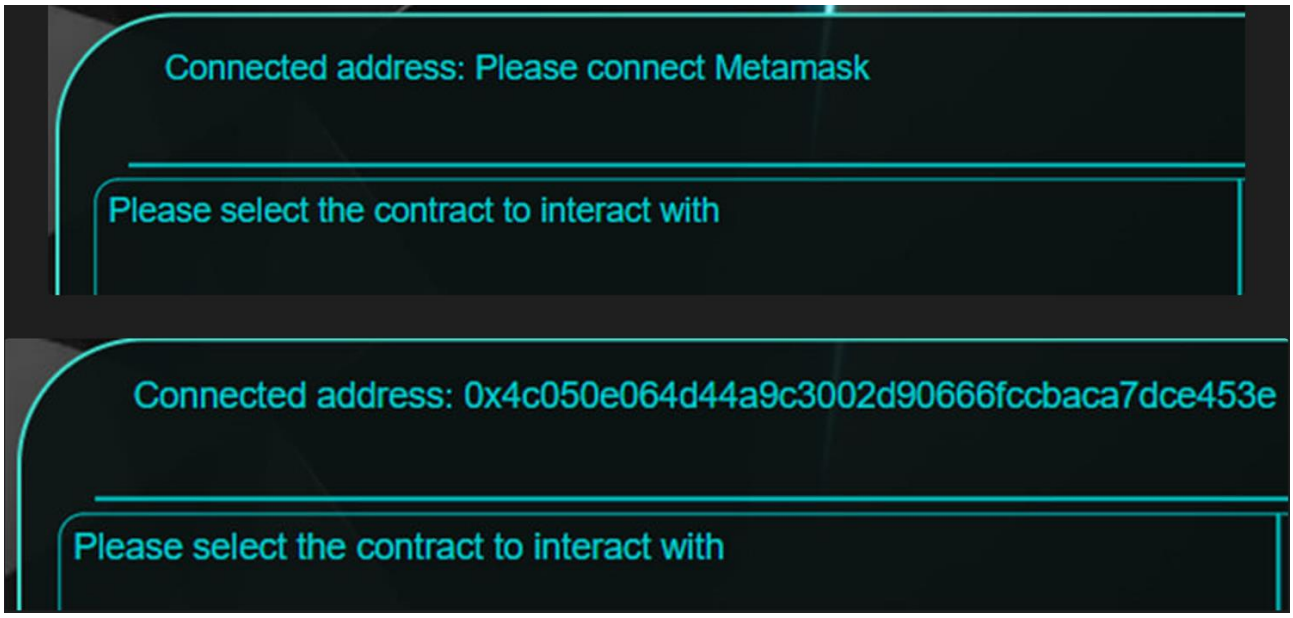


Рисунок 4.10 – Підключення до блокчейну через інтерфейс користувача

Також, важливим функціоналом інтерфейсу є можливість швидко та зручно перемикатись між смарт-контрактами генерування голосування та проведення голосування. Дана можливість реалізована, та відображає інтерфейс функціоналу саме обраного модуля системи (рис. 4.11).

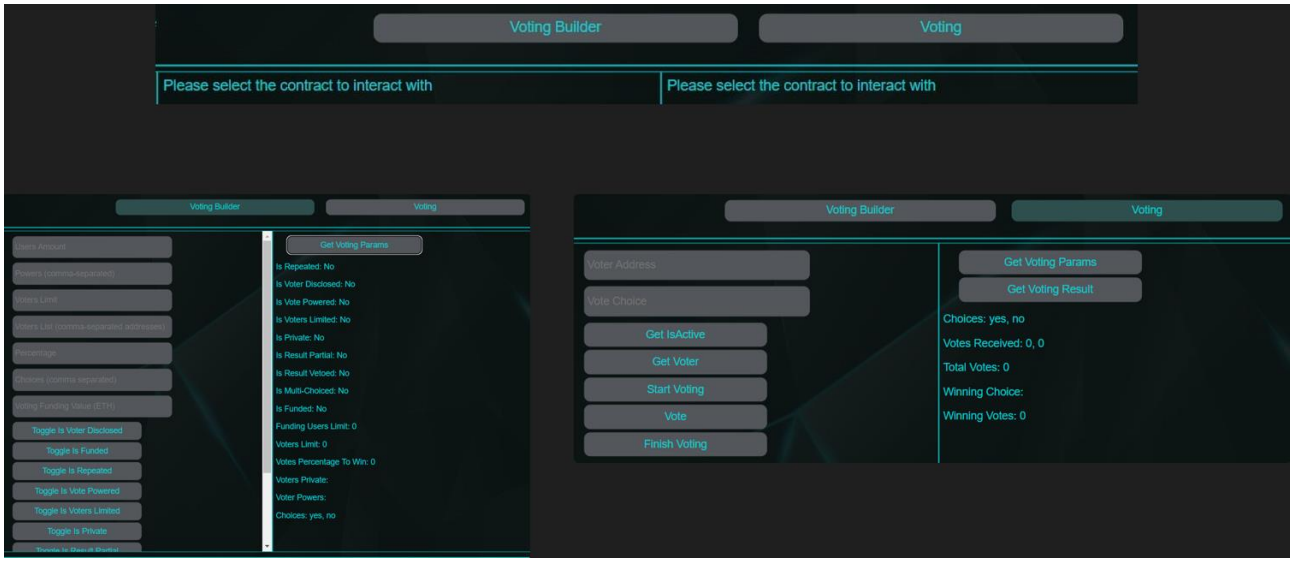


Рисунок 4.11 – Перемикання між смарт-контрактами через інтерфейс користувача

І головною вимогою до інтерфейсу користувача є повна функціональна відповідність та надання можливості взаємодіяти зі смарт-контрактами та отримувати всю необхідну інформацію без жодних додаткових дій та інструментів.

Розроблений веб-інтерфейс було розподілено на 3 основні секцій, кожна з яких виконує свою роль: секція рольової моделі, функціональна секція, інформаційна секція (рис. 4.12).

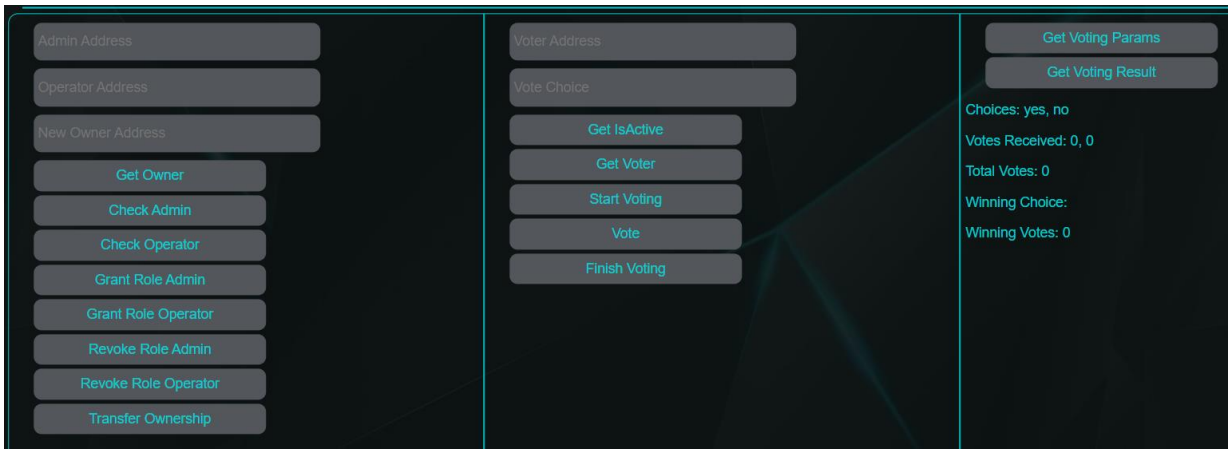


Рисунок 4.12 – Робоча область інтерфейсу користувача

Секція рольової моделі відображає усі доступні можливості викликів та запитів стосовно ролей смарт-контракту.

Функціональна секція відображає усі доступні можливості викликів та запитів стосовно основних функцій смарт-контракту, та є основною робочою областю користувача.

Інформаційна секція відображає усі запити на отримання головної інформації про функціональні параметри смарт-контракту та виступає інформаційною панеллю користувача.

В результаті, було розроблено інтерфейс користувача, що дозволяє легко підключатись до блокчейну, обирати смарт-контракт для взаємодії, та безпосередньо взаємодіяти з системою генерування голосувань.

4.4 Статичне тестування безпеки засобу

Статичне тестування безпеки (SAST) є важливою складовою в розробці смарт-контрактів, оскільки це дозволяє виявити потенційні безпекові проблеми в коді контрактів на ранній стадії розробки. SAST аналізує код без його активного виконання, виявляючи можливі уразливості, такі як проблеми з контролем доступу, недоліки в безпеці даних чи можливість витоку інформації.

Використання SAST у розробці смарт-контрактів допомагає вчасно виявляти та усувати потенційні загрози безпеці, зменшуючи ризик можливих атак чи неправильного використання контрактів у майбутньому. Цей підхід сприяє підвищенню рівня довіри до безпеки смарт-контрактів та допомагає уникнути серйозних проблем з безпекою під час їхнього функціонування в блокчейні.

Для статичного тестування безпеки засобу генерації голосувань було обрано Slither - інструмент для аналізу безпеки смарт-контрактів у мережі Ethereum. Цей інструмент має ряд вбудованих правил, які аналізують код контракту і виявляють можливі проблеми.

Slither дозволяє проаналізувати всі смарт-контракти в системі одразу, якщо вони знаходяться в одному файлі, наявні смарт-контракти: VotingRoleModel, Voting, VotingBuilder (рис. 4.13).

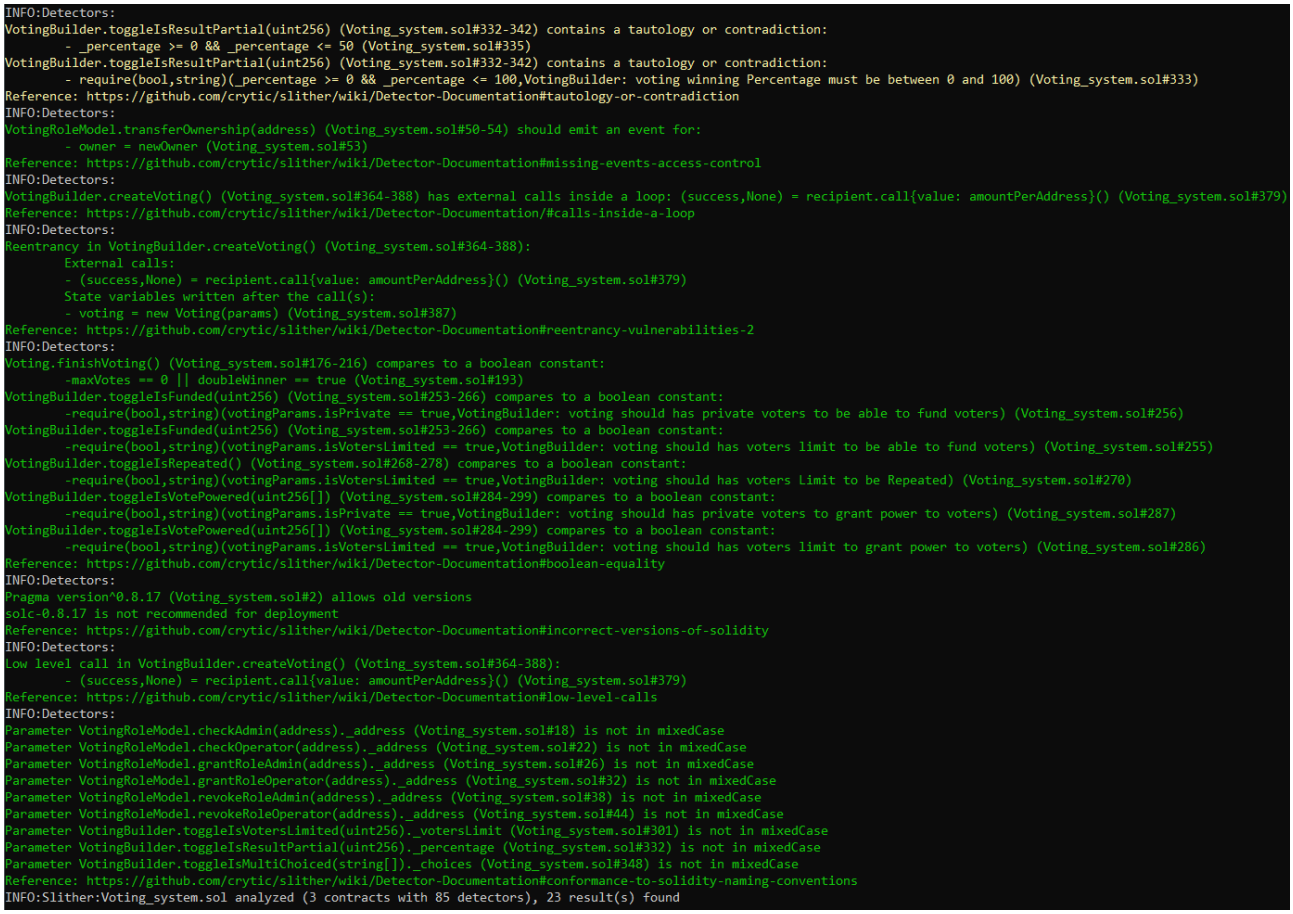


Рисунок 4.13 – Результат статичного тестування через Slither

Результат тестування за допомогою Slither не виявив будь-яких критичних вразливостей безпеки. Але розглянемо детальніше отриману в звіті інформацію.

Звіт зауважує на тавтології при перевірці чи знаходиться відсоток для перемоги в голосування в допустимих межах. Дане рішення обумовлене різними сферами даних перевірок: перша виконується для всієї процедури для перевірки введеного числа, а друга – для встановки параметру при введенні числа в певному діапазоні.

Slither вказує на те, що зміна власника контракту повинна відсилати подію в блокчейн. Це дійсно гарна практика для зручного інформування про зміни стану смарт-контракту, але має рекомендаційний характер.

Інструмент робить акцент на процес відсилання коштів учасникам при активному параметрі фінансування голосування, а саме що використовується низькорівнева функція, є вразливість до повторного виклику та спустошення балансу, та що процес виконується в циклі. Дане рішення було обумовлене тим,

що низькорівнева функція відправки криптовалюти дозволяє отримати більше контролю над процесом та чіткішу обробку помилок, а виконання в циклі необхідне для відправки коштів всім учасникам та контролюється шляхом обмеження кількості таких учасників. Також, вразливість до повторного виклику (Reentrancy) було передбачено, процедуру створення голосування поміщено в модифікатор, що блокує повторний виклик функції до її повного завершення.

Серед інших зауважень можна видітити прирівняння до константи при перевірках умов, використання не найновішої версії Solidity, та невідповідність назви смарт-контракту конвенції написання коду. Дані пункти несуть виключно рекомендаційний характер та не можуть спричинити жодної загрози, а написання назви контракту з великої літери навпаки є вимогою мови Solidity.

Отже, результат статичного тестування безпеки смарт-контрактів системи за допомогою інструменту Slither не виявив жодних критичних помилок, а виявлені зауваження є виключно рекомендаціями, або були передбачені та вже містять запобіжні рішення.

4.5 Тестування всієї системи

Отримавши результати динамічного та статичного тестування смарт-контрактів, та перевіривши працездатність інтерфейсу користувача, доцільно виконати тестування всієї системи генерування та проведення голосування на прикладі демонстрації повного робочого циклу засобу.

Виконавши підключення крипто-гаманця до інтерфейсу, користувач підключається до блокчейну та може обрати смарт-контракт для взаємодії. Послідовним робочим процесом є вибір модуля генерування голосування для налаштування параметрів та створення нового екземпляру голосування.

Однак, правилами розмежування прав доступу передбачено, що даний функціонал доступний лише для адміністраторів системи, в цілях перешкодження несанкціонованому доступу. При спробі неавторизованого

користувача виконати обмежені дії, смарт-контракт, та відповідно і інтерфейс, виведуть помилку інформуючу про причину проблеми (рис. 4.14).

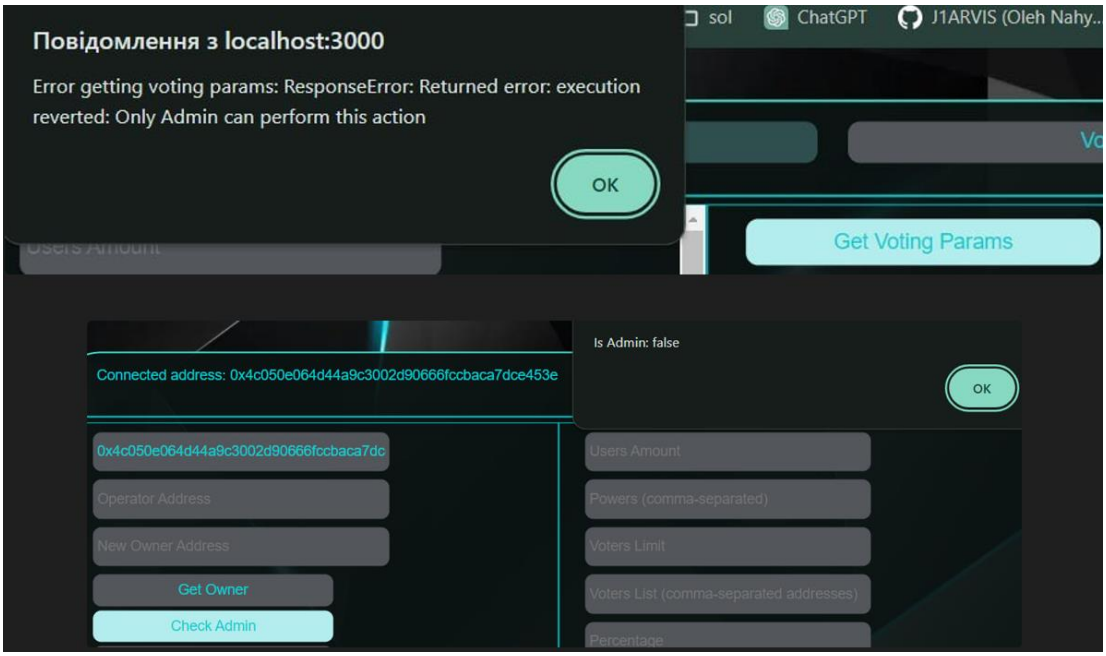


Рисунок 4.14 – Результат спроби виконання обмежених дій

Для вирішення даної перешкоди, користувач повинен звернутись до власника смарт-контракту, аби той включив його до переліку адміністраторів та надав дозвіл на виконання відповідних дій, пов’язаних з налаштуванням та генеруванням нового голосування (рис. 4.15).

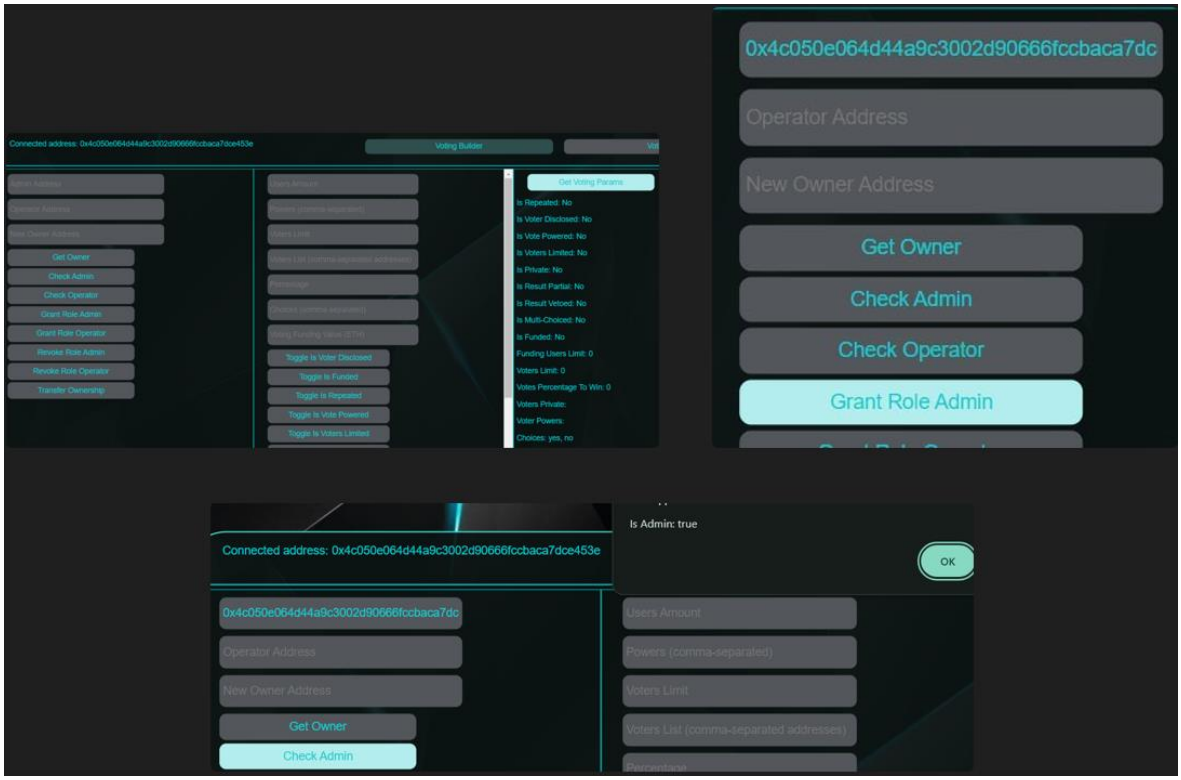


Рисунок 4.15 – Результат активації прав адміністратора

Отримавши дозвіл, користувач, а тепер вже – адміністратор, може виконати налаштування параметрів бажаного голосування використовуючи наявні виклики та перевірити результат своїх дій за допомогою запиту відображення актуальних параметрів (рис. 4.16).

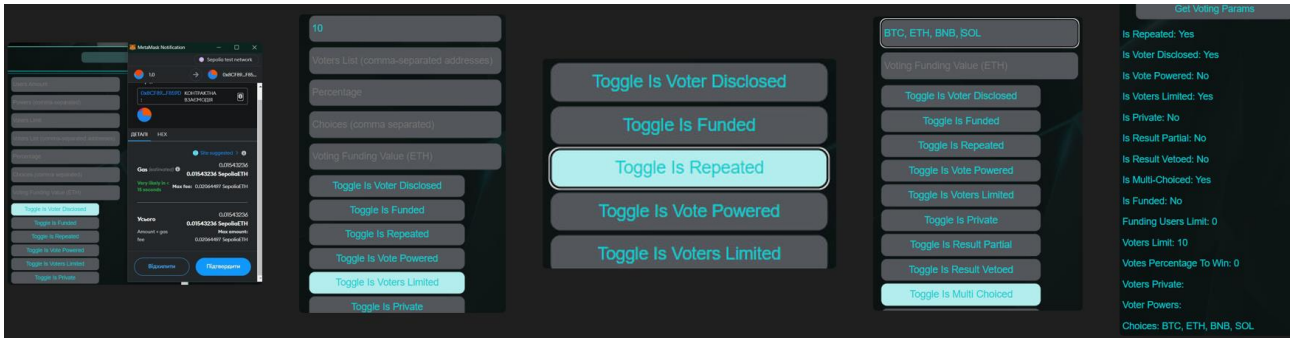


Рисунок 4.16 – Процес налаштування параметрів голосування

Отримавши бажані параметри, адміністратор може перейти до наступного кроку – генерування нового голосування, та отримання адреси нового екземпляру смарт-контракту на блокчейні (рис. 4.17).

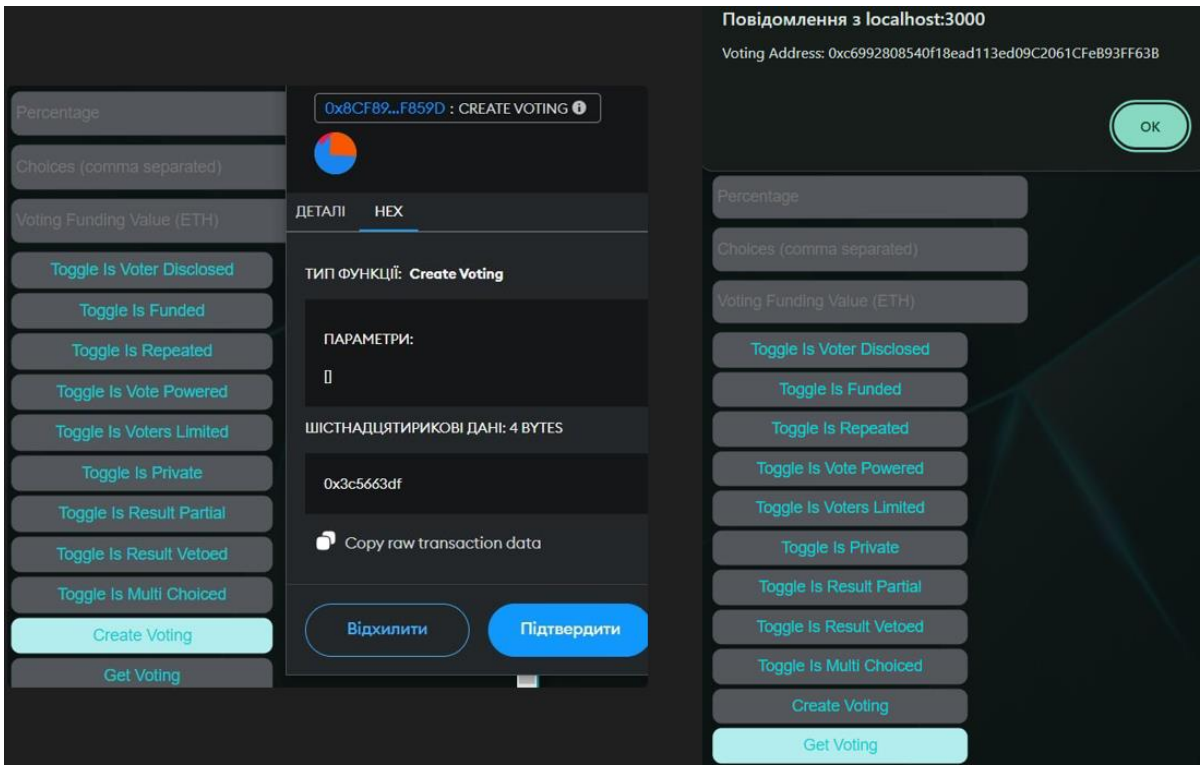


Рисунок 4.17 – Процес генерування голосування

Успішно виконавши запит на створення нового голосування, адміністратор може перевірити коректність створеного голосування та його відповідність до встановлених налаштувань.

Для цього в першу чергу потрібно перемкнути інтерфейс користувача на смарт-контракт проведення голосування, та використати отриману адресу нового екземпляру голосування для підключення до нього.

Підключившись до новоствореного голосування, користувач, що є його власником, може призначити собі роль адміністратора для змоги перевірити коректність налаштувань створеного голосування, та безпосередньо здійснити запит на параметри та порівняти їх з бажаними (рис. 4.18).

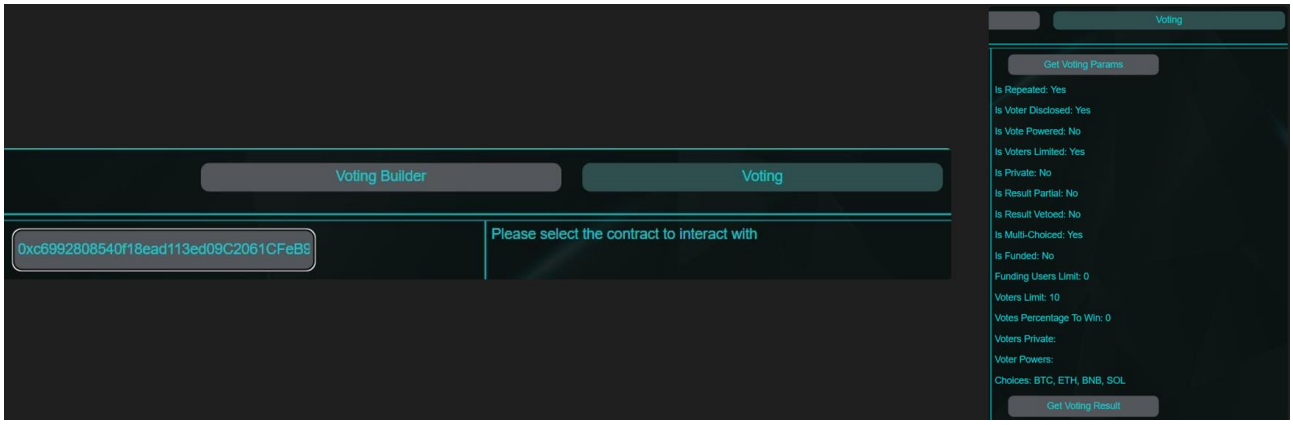


Рисунок 4.18 – Підключення та перевірка нового голосування

Впевнившись в коректності створеного голосування, користувач може назначити оператора голосування, який в свою чергу зможе розпочати процес подачі голосів (рис. 4.19).

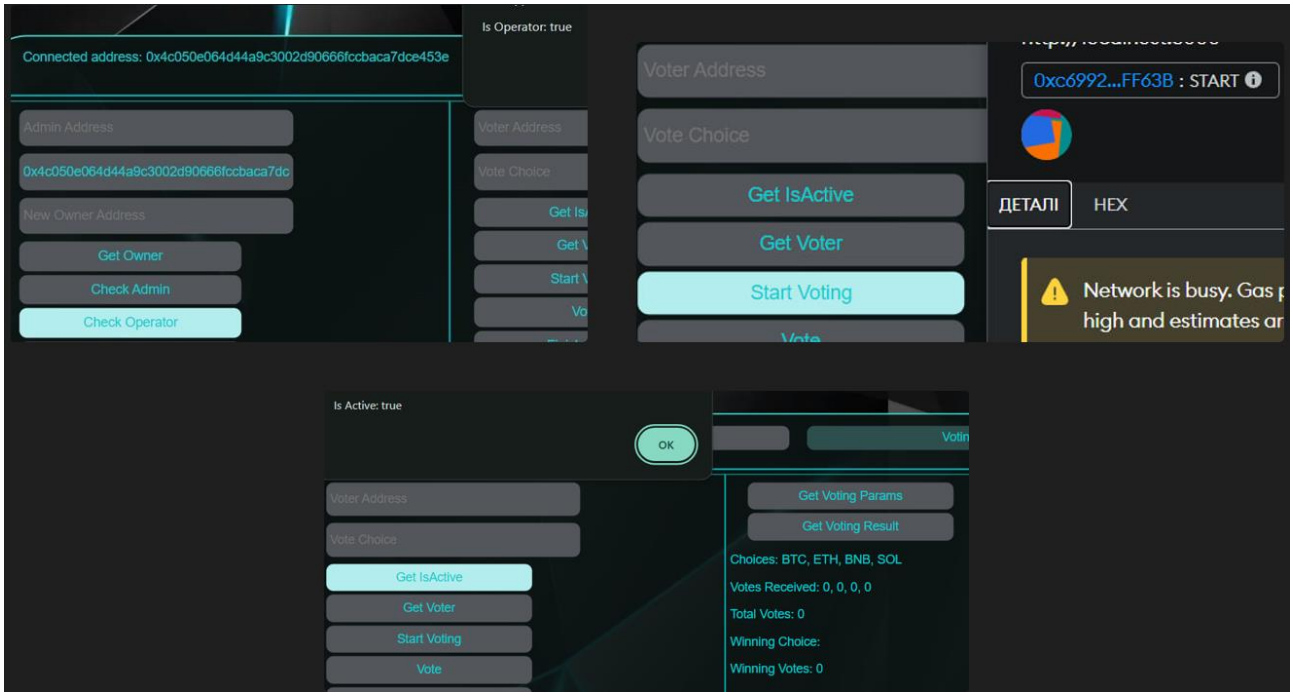


Рисунок 4.19 – Старт процесу голосування

Після старту голосування, учасники можуть подавати голоси за бажані вибори, що відображені в інформаційній секції інтерфейсу. Переглянути подану кількість голосів завжди можна здійснивши запит до результатів голосування (рис. 4.20).

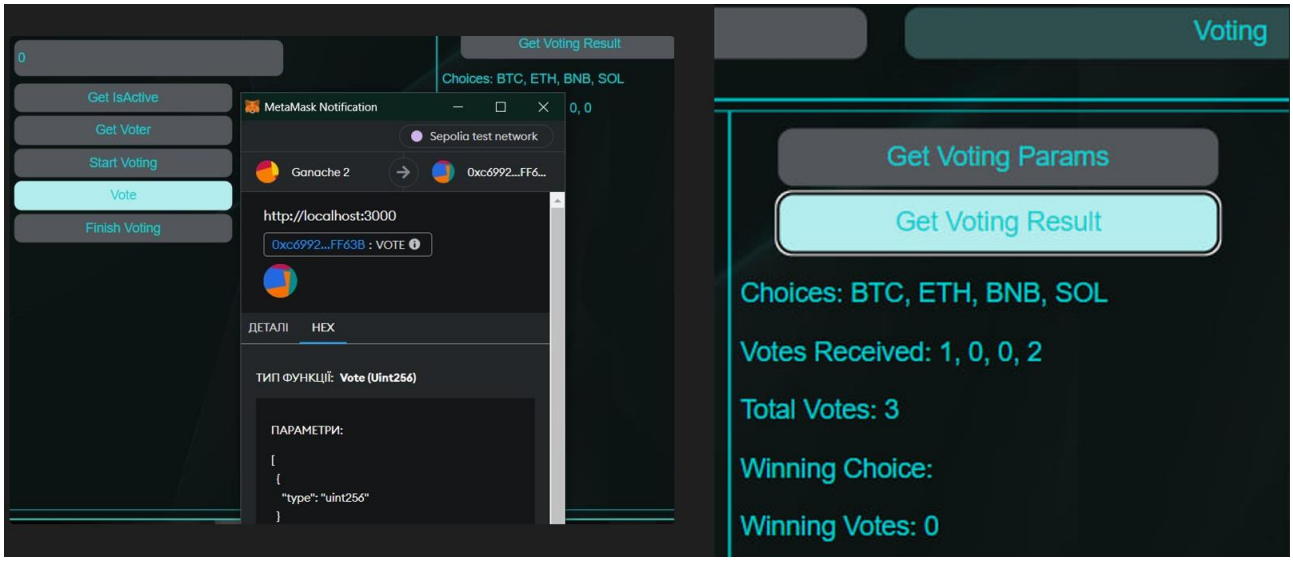


Рисунок 4.20 – Подача голосів та їх запис

В разі рішення оператора про припинення голосування, він може здійснити відповідний запит до смарт-контракту, та отримати автоматично підраховані результати (рис. 4. 21).

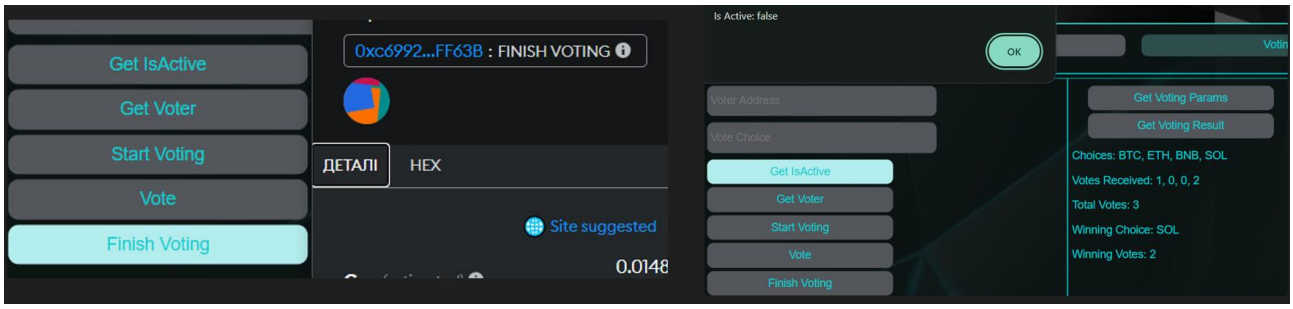


Рисунок 4.21 – Завершення голосування та підрахунок результату

В результаті, було продемонстровано та доведено працездатність засобу в робочому процесі повного циклу та його готовність до передачі в широке використання.

4.6 Висновки з розділу

На основі розробленого алгоритму та процедур було обгрунтовано засоби розробки та тестування, та безпосередньо розроблено сам програмний засіб в вигляді смарт-контрактів мовою Solidity.

Розроблено та програмно реалізовано набір автоматичних блокових тестів для розроблених смарт-контрактів, що підтвердили їх ефективність, надійність, та відповідність сучасним вимогам безпеки та контролю доступу.

Також, розроблено інтерфейс користувача, що дозволяє легко підключатись до блокчейну, обирати смарт-контракт для взаємодії, та безпосередньо взаємодіяти з системою генерування голосувань.

Продемонстровано та доведено працездатність засобу в робочому процесі повного циклу та його готовність до передачі в широке використання в публічному середовищі.

Виконано статичне тестування безпеки, що не виявило критичних помилок та вразливостей, та підтвердило можливість використання розроблених смарт-контрактів в публічному блокчейні.

5 ЕКОНОМІЧНА ЧАСТИНА

Для успішного впровадження науково-технічної розробки критично важливо, щоб вона відповідала сучасним вимогам науково-технічного прогресу та враховувала економічні аспекти. Оцінка економічної ефективності результатів науково-дослідної роботи є важливою складовою цього процесу. Дослідження, яке представлене у магістерській роботі та присвячене розробці та аналізу "Методу та інструменту генерування смарт-контрактів для захищеного голосування", віднесено до науково-технічних робіт, спрямованих на виведення на ринок. Рішення щодо комерціалізації розробки може бути прийняте протягом виконання самої роботи, відкриваючи можливості для подальшого виведення на ринок. Цей напрямок розглядається як пріоритетний, оскільки отримані результати можуть бути корисними для різних зацікавлених сторін і приносити економічні вигоди. Проте для успішної реалізації цього процесу важливо залучити зацікавленого інвестора, який виявить інтерес до втілення даного проекту, і переконати його у доцільності інвестування у цю розробку. З метою досягнення цього завдання передбачені наступні етапи виконання робіт:

1. Проведення комерційного аудиту науково-технічної розробки, включаючи визначення науково-технічного рівня та комерційного потенціалу.
2. Розрахунок витрат на реалізацію науково-технічної розробки.
3. Проведення розрахунку економічної ефективності впровадження та комерціалізації науково-технічної розробки для потенційного інвестора, а також обґрунтування економічної доцільності комерціалізації з точки зору інвестора.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Метод та засіб генерування смарт-контрактів для захищеного

голосування» є розширення функціональних можливостей програмного забезпечення для захисту файлів.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [34].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці. Для оцінки науково-технічного рівня і комерційного потенціалу розробки експертами було запрошено трьох незалежних експертів: Юрій Баришев, к.т.н., доц. кафедри «Захисту інформації» Вінницького національного технічного університету, В’ячеслав Демчук, CEO, Safelement Limited, Дмитро Тинніков, проектний менеджер, Safelement Limited

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Юрій Баришев, к.т.н., доц. каф. ЗІ, ФІТКІ, ВНТУ	В'ячеслав Демчук, CEO, Safelement Limited	Дмитро Тинніков, Проектний менеджер, Safelement Limited
	Бали, виставлені експертами:		
1. Технічна здійсненність концепції	3	3	3
2. Ринкові переваги (наявність аналогів)	2	3	2
3. Ринкові переваги (ціна продукту)	4	3	4
4. Ринкові переваги (технічні властивості)	3	3	2
5. Ринкові переваги (експлуатаційні витрати)	4	4	3
6. Ринкові перспективи (розмір ринку)	2	3	2
7. Ринкові перспективи (конкуренція)	3	2	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	3	4
Сума балів	$СБ_1$=40	$СБ_2$=39	$СБ_3$=38
Середньоарифметична сума балів $СБ_c$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{40+39+38}{3} = 39$		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [34].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній

11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Метод та засіб генерування смарт-контрактів для захищеного голосування» становить 39 бали, що, відповідно до таблиці 5.3 рівень комерційного потенціалу розробки високий, що свідчить про комерційну важливість проведення даних досліджень.

Магістерська кваліфікаційна робота «Метод та засіб генерування смарт-контрактів для захищеного голосування» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок, тобто при цьому відбувається комерціалізація науково-технічної розробки. Цей напрямок є для нас пріоритетним, оскільки результатами розробки можуть користуватися не тільки самі розробники, а й інші споживачі, отримуючи при цьому суттєвий економічний ефект.

Результатом магістерської роботи є програмний засіб для створення та проведення захищеного голосування на основі смарт-контрактів, який буде корисний комерційним компаніям, особам з доступом до інтернету, переважно які знайомі з блокчейном.

5.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

В якості аналога для розробки було обрано системи електронного голосування (eBallot). Основними недоліками аналога є відсутність впевненості на технічному рівні в прозорості, незмінності, коректності процесу голосування та підрахунку результатів. Також до недоліків можна віднести недостатню гнучкість генерування голосувань.

У розробці дана проблема вирішується шляхом використання смарт-контрактів, рольової моделі, та патерну проектування. Існують реалізовані

рішення голосування на блокчейні, а також захищеного електронного голосування, але наша розробка має переваги в багатофункціональності та захищеності на блокчейні.

Одиничний параметричний індекс розраховуємо за формулою [34]:

$$q_i = \frac{P_i}{P_{базі}}. \quad (5.1)$$

де q_i – одиничний параметричний індекс, розрахований за i-м параметром;

P_i – значення i-го параметра виробу;

$P_{базі}$ – аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 5.4.

Таблиця 5.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Доступність сервісу,%	92	100	1,09	20%
Прозорість процесу, %	74	100	1,35	30%
Вартість підтримки працездатності на одного учасника, грн	22	7	3,14	20%
Вразливість до атак, шт	15	4	3,75	15%
Варіації голосування, шт	32	160	5	15%

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою [34]:

$$I_{НП} = \prod_{i=1}^{n} q_i, \quad (5.2)$$

де $I_{нп}$ – загальний показник конкурентоспроможності за нормативними параметрами;

q_i – одиничний (частинний) показник за i-м нормативним параметром;

n – кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [34]:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \ , \tag{5.3}$$

де $I_{тп}$ – груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i – одиничний параметричний показник i-го параметра;

α_i – вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 5.4.

$$I_{тп} = 1{,}09 \cdot 0{,}2 + 1{,}35 \cdot 0{,}3 + 3{,}14 \cdot 0{,}2 + 3{,}75 \cdot 0{,}15 + 5 \cdot 0{,}15 = 2{,}55.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [34]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \ , \tag{5.4}$$

де $I_{ЕП}$ – груповий параметричний індекс за економічними показниками;

q_i – економічний параметр i-го виду;

β_i – частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m – кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП} = 0{,}75 \cdot 0{,}5 + 0{,}86 \cdot 0{,}5 = 0{,}80.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [34]**:**

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \tag{5.5}$$

$$K_{ІНТ} = 1 \cdot 2{,}55 / 0{,}80 = 3{,}2.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Метод та засіб генерування смарт-контрактів для захищеного голосування», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [34]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \tag{5.6}$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 28000 \cdot 76 / 21 = 96727 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
1. Керівник проекту розробки та дослідження інформаційної технології управління проектами	28000	1272,7	76	96727
2. Інженер контролю якості	25000	1136,4	76	86364
3. Solidity - розробник	32000	1454,5	54	78545
4. React- розробник	12000	545,5	32	17455
Всього				279091

Основна заробітна плата робітників.

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Метод та засіб генерування смарт-контрактів для захищеного голосування» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \quad (5.7)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \quad (5.8)$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6500 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [34];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1 \cdot 1{,}65 / (21 \cdot 8) = 65{,}8$ грн.

$З_{р1} = 65{,}8 \cdot 1 = 65{,}8$ грн.

Таблиця 5.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
1.Підготовчі	32	1	65,8	2105,7
2.Монтажні	40	3	88,8	3553,4
3.Складальні	40	5	111,9	4474,6
4.Налагоджувальні	40	2	72,4	2895,4
5.Випробувальні	36	4	59,8	2153,6
Всього				15182,7

Додаткова заробітна плата дослідників та робітників.

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \qquad (5.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (279091+ 15182,7) · 11 / 100% = 32379,09 грн.

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \qquad (5.10)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$Зн$ = (279091+ 15182,7+32379,09) · 22 / 100% = 71861,61 грн.

5.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Метод та засіб генерування смарт-контрактів для захищеного голосування».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \qquad (5.11)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний (А4)	180	1	180
Папір для заміток (А5)	115	1	115
Органайзер офісний	150	1	150
Всього			445
З врахуванням коефіцієнта транспортування			489,5

5.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна система підтримки підприємств малого бізнесу у сфері послуг».

Витрати на комплектуючі К, що були використані під час виконання даного етапу роботи, розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ грн.}, \quad (5.12)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – ціна комплектуючих *i*-го виду, грн.;

$К_i$ – коефіцієнт транспортних витрат, $К_i = (1{,}1 \ldots 1{,}15)$;

n – кількість видів комплектуючих.

Зроблені розрахунки бажано звести до таблиці:

Таблиця 5.8 – Витрати на комлектуючі

Найменування комплектувальних	Кількість	Ціна за штуку, грн.	Сума, грн.
HP Omen 15 2018	2	36000	72000
VS Code	1	7500	7500
Ethereum ETH	0,1	77700	7770
Testrail monthly	3	1370	4110
Всього з врахування коефіцієнт транспортних витрат			100518,

5.3.5 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.13)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (36000 \cdot 1) / (2 \cdot 12) = 4500$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
HP Omen 15 2018	36000	2	1	4500,00
Робоче місце розробника ПЗ	185700	20	2	2321,25
Всього				6821,25

5.3.6 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_е = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_е \cdot K_{вni}}{\eta_i}, \qquad (5.14)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 7,5 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_е = 0,25 \cdot 325,0 \cdot 7,5 \cdot 0,5 / 0,8 = 380,86$ грн.

5.3.7 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Метод та засіб генерування смарт-контрактів для захищеного голосування» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань

машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.15)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (279091+ 15182,7) · 20 / 100% = 58854,72 грн.

5.3.8 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.16)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (279091+ 15182,7) · 50 / 100% = 147136,79 грн.

5.3.9 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.17)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (279091+ 15182,7) · 100 / 100% = 294273,59 грн.

Витрати на проведення науково-дослідної роботи на тему «Метод та засіб генерування смарт-контрактів для захищеного голосування» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв} \quad (5.18)$$

$В_{заг}$=279091+15182,7+32379,09+71861,61+489,5+100518+6821,25+380,86 +58854,72 +147136,79 +294273,59 = 1006980,00 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.19)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,7.

ЗВ = 1006980,00 / 0,7 = 1438542,86 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Метод та засіб генерування смарт-контрактів для захищеного голосування» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 1000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо зростання на 100,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [34]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.20)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(1·100+1000·8000)·0,83·0,4·(1-0,18/100%)=1503290,3 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(1·100+1000·(8000+10000))·0,83·0,4·(1-0,18/100%)=3382464,7 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (1 \cdot 100 + 1000 \cdot (8000 + 10000 + 11000)) \cdot 0{,}83 \cdot 0{,}4 \cdot (1 - 0{,}18/100\%) = 5449465{,}4$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.21)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =18%;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 1503290{,}3 / (1+0{,}18)^1 + 3382464{,}7 / (1+0{,}18)^2 + 5449465{,}4 / (1+0{,}18)^3 =$$
$$= 6769969{,}53 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \qquad (5.22)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 1438542,86 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 1438542{,}86 = 2877085{,}72$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (5.23)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 6769969,53 грн;

PV – теперішня вартість початкових інвестицій, 2877085,72 грн.

$E_{абс} = ПП - PV$ = 6769969,53 - 2877085,72 = 3892883,82 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.24)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+3892883,82 /2877085,72)^{1/3} - 1 = 0,55.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.25)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,55 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у

впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія онтологічного моделювання бази знань з організації бібліотеки» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.26}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,55 = 1,8 р.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.5 Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Метод та засіб генерування смарт-контрактів для захищеного голосування» становить 39 балів, що, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий.

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,2 рази.

Також термін окупності становить 1,8 роки, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Метод та засіб генерування смарт-контрактів для захищеного голосування».

ВИСНОВКИ

Аналіз систем електронного голосування, технології блокчейн, та шаблонів проектування дозволив визначити ключові задачі для розробки системи захищеного електронного голосування. Досліджено існуючі системи електронного голосування та виявлено їхні переваги та недоліки, розглянуто потенційні загрози та вразливості систем електронного голосування. Висвітлено основні переваги технології блокчейн та розглянуто важливість використання твірних шаблонів проектування. Засоби на основі блокчейну покращують безпеку, прозорість та відслідковуваність, втім мають проблеми недостатньої гнучкості та проблеми прав доступу при великих масштабах. На основі цих результатів аналізу було сформовано задачу роботи.

Проведення математичного опису системи захищеного електронного голосування дозволило виокремити вхідні дані системи та кроки процесу голосування, а також виділити підтримувані типи голосування. Дослідження проблем процесу розробки та генерування голосування за допомогою смарт-контрактів наштовхнуло на рішення в вигляді використання твірного шаблону проектування та дозволило виокремити ключові кроки створення нового екземпляру голосування. Розробка архітектури засобу в результаті призвела до поділу системи на ключові функціональні модулі з описом їх призначення, та сформування правил розмежування прав доступу до модулів системи.

Розробка загального алгоритму роботи засобу генерування смарт-контрактів дозволила розробити та описати алгоритми взаємодії користувача з головними функціональними модулями. Також, завдяки цьому, було розроблено критично важливі процедури системи, як то налаштування різних параметрів голосування, генерації нового екземпляру голосування, старту голосування, подачі голосу, завершення голосування та підрахунку результатів, а також розроблено рольову модель в смарт-контрактів системи.

Для перевірки якості розробленого засобу було залучено процеси тестування з використанням середовищ та фреймворків для автоматизації даного процесу. На основі розроблених процедур та загального алгоритму було розроблено набір сценаріїв автоматичних блокових тестів для смарт-контрактів

системи. Виконане автоматичне динамічне тестування не виявило жодних помилок в реалізації розроблених процедур та підтвердило ефективність, надійність, та відповідність смарт-контрактів сучасним вимогам безпеки та контролю доступу. Окрім цього, було виконано статичне тестування безпеки, що не виявило критичних помилок та вразливостей, та підтвердило можливість використання розроблених смарт-контрактів в публічному блокчейні без ризиків загроз для користувачів та власників продукту.

На основі проведених економічних досліджень виявлено, що рівень комерційного потенціалу розробки складає 39 балів, що свідчить про високий комерційний потенціал та можливість ефективного застосування даної розробки як продукту на ринку. Термін окупності розробки складає 1,8 роки, що є меншим результатом ніж тривалість в 3 роки, що свідчить про її комерційну привабливість і високу імовірність вдалого залучення інвестицій для впровадження на ринок. Отримані результати дозволяють зробити висновок про доцільність подальшого проведення науково-дослідної роботи за даною темою.

Отриманий програмний засіб можливо використовувати для будь-якого роду голосувань в комерційних або особистих цілях. Розроблений засіб може бути корисним як компаніям та особам що знайомі з блокчейном, так і може успішно використовуватись новачками в даній сфері, що тісно не знайомі з технологіями децентралізованих додатків.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Олег Нагирняк, Валентина Каплун. Засіб генерування ключової інформації для захисту програм від несанкціонованого копіювання. Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії ВНТУ. 2021 р. 3 с. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2021/paper/view/12138/10258 (дата звернення: 15.10.2023).
2. Bilal Benseddiq Independent expertise of electronic voting systems. URL: https://www.riskinsight-wavestone.com/en/2023/05/independent-expertise-of-electronic-voting-systems/ (accessed: 03.11.2023)
3. Secret ballot - how to vote online in the company. URL: https://privote.net/blog/tips/how-to-vote-online-secret-ballot-in-the-company (accessed: 03.11.2023)
4. Dimitris A Gritzalis Principles and requirements for a secure e-voting system, 2002. URL: https://www.sciencedirect.com/science/article/abs/pii/S0167404802010143 (accessed: 03.11.2023)
5. Chantal Enguehard Transparency in Electronic Voting : the Great Challenge, 2009. 14c. URL: https://shs.hal.science/halshs-00409465/document (accessed: 03.11.2023)
6. Mark Dipinto The Benefits of Electronic Voting, 2021. URL: https://www.getquorum.com/blog/the-benefits-of-electronic-voting/ (accessed: 03.11.2023)
7. Oleksandr Markovets and Mykola Buchyn Threats of the Implementation of E-Voting and Methods of Their Neutralization. Lviv Polytechnic National University, Lviv, Ukraine, 2022. 11c.
8. The Advantages And Disadvantages Of Online Voting Systems. URL: https://electionbuddy.com/blog/2022/04/20/the-advantages-and-disadvantages-of-online-voting-systems/ (accessed: 03.11.2023)
9. Benefits and risks of e-voting and on-line voting. URL: https://decentralization.gov.ua/en/news/12905 (accessed: 03.11.2023)

10. How Does Blockchain Work? URL: https://academy.binance.com/en/articles/how-does-blockchain-work? (accessed: 03.11.2023)
11. Blockchain Voting: The Future of Elections? URL: https://builtin.com/blockchain/blockchain-voting-future-elections (accessed: 03.11.2023)
12. Uzma Jafar, Mohd Juzaiddin Ab Aziz, Zarina Shukur Blockchain for Electronic Voting System - Review and Open Research Challenges, 2021. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8434614/ (accessed: 03.11.2023)
13. Ashwani Kumar Pandey, Abhinandan Tripathi, Ashok Kumar Yadav & Shreya Yadav Voting System Based on Blockchain Technology Smart-Contracts, 2023. URL: https://link.springer.com/chapter/10.1007/978-981-99-1431-9_10 (accessed: 03.11.2023)
14. What are smart contracts on blockchain? URL: https://www.ibm.com/topics/smart-contracts (accessed: 03.11.2023)
15. What Are Smart Contracts on the Blockchain and How They Work. URL: https://www.investopedia.com/terms/s/smart-contracts.asp (accessed: 03.11.2023)
16. What is a smart contract? URL: https://www.coinbase.com/learn/crypto-basics/what-is-a-smart-contract (accessed: 03.11.2023)
17. Олександр Швець Занурення в патерни проектування, 2018. 393с.
18. Design Patterns. URL: https://sourcemaking.com/design_patterns (accessed: 03.11.2023)
19. Definitions of Voting Systems. URL: https://www.ecanz.gov.au/electoral-systems/definitions-voting-systems (accessed: 03.11.2023)
20. Message signing and transaction signing. URL: https://aag.ventures/academy/message-signing-and-transaction-signing/ (accessed: 03.11.2023)

21. Jose Aguinaga Part Three: Creating and Signing Ethereum Transactions, 2021. URL: https://medium.com/portis/part-three-creating-and-signing-ethereum-transactions-e9cca44d7e2d (accessed: 03.11.2023)
22. User Role and Permission Management: Designing Efficient Models. URL: https://frontegg.com/guides/user-role-and-permission (accessed: 03.11.2023)
23. How to Design User Role Permission Model? URL: https://www.nikhilajain.com/post/user-role-permission-model (accessed: 03.11.2023)
24. Welcome to Remix's documentation! URL: https://remix-ide.readthedocs.io/en/latest/ (accessed: 03.12.2023)
25. Remix project. URL: https://remix-project.org/ (accessed: 03.12.2023)
26. Visual Studio Code. URL: https://code.visualstudio.com/docs (accessed: 03.12.2023)
27. Introducing Tools for Solidity – VS Code Extension. URL: https://ackeeblockchain.com/blog/introducing-tools-for-solidity/ (accessed: 03.12.2023)
28. What is Truffle? URL: https://trufflesuite.com/docs/truffle/ (accessed: 03.12.2023)
29. What is Ganache? URL: https://trufflesuite.com/docs/ganache/ (accessed: 03.12.2023)
30. Hardhat Runner. URL: https://hardhat.org/docs (accessed: 03.12.2023)
31. Jest. URL: https://jestjs.io/ (accessed: 03.12.2023)
32. Mocha. URL: https://mochajs.org/ (accessed: 03.12.2023)
33. Solidity. URL: https://docs.soliditylang.org/en/v0.8.10/ (accessed: 03.12.2023)
34. В. О. Козловський, О. Й. Лесько, В. В. Кавецький Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. ВНТУ, Вінниця, 2021. 42 с.

ДОДАТКИ

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Метод та засіб генерування смарт-контрактів для захищеного голосування

Автор роботи: Нагирняк Олег Юрійович

Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ

(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 97,59 %. Схожість – 2,41 %.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

(підпис)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Олег НАГИРНЯК

(підпис)

Керівник роботи _____ Юрій БАРИШЕВ

(підпис)

ДОДАТОК Б

Текст програми смарт-контрактів

Voting_system.sol:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.17;

contract VotingRoleModel {
    mapping(address => bool) roleAdmin;
    mapping(address => bool) roleOperator;
    address owner;

    constructor() {
        owner = tx.origin;
    }

    function getOwner() public view returns(address) {
        return owner;
    }
    function checkAdmin(address _address) public view returns(bool) {
        return roleAdmin[_address];
    }

    function checkOperator(address _address) public view returns(bool) {
        return roleOperator[_address];
    }

    function grantRoleAdmin(address _address) public onlyOwner {
        require(!roleAdmin[_address], "VotingRoleModel: Role already granted to this address");

        roleAdmin[_address] = true;
    }

    function grantRoleOperator(address _address) public onlyOwner {
        require(!roleOperator[_address], "VotingRoleModel: Role already granted to this address");

        roleOperator[_address] = true;
    }

    function revokeRoleAdmin(address _address) public onlyOwner {
        require(roleAdmin[_address], "VotingRoleModel: Role is not granted to this address");

        roleAdmin[_address] = false;
    }

    function revokeRoleOperator(address _address) public onlyOwner {
        require(roleOperator[_address], "VotingRoleModel: Role is not granted to this address");

        roleOperator[_address] = false;
    }

    function transferOwnership(address newOwner) public onlyOwner {
        require(newOwner != owner, "VotingRoleModel: cannot transfer Ownership to himself");

        owner = newOwner;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only owner can perform this action");
        _;
    }

    modifier onlyAdmin() {
        require(roleAdmin[msg.sender], "Only Admin can perform this action");
        _;
    }

    modifier onlyOperator() {
        require(roleOperator[msg.sender], "Only Operator can perform this action");
        _;
    }
}

contract Voting is VotingRoleModel {
    VotingBuilder.VotingParameters votingParams;
```

```
    bool isActive = false;

    struct Voter {
        bool voted;
        string vote;
        uint256 power;
        bool isPrivate;
    }
    mapping(address => Voter) voters;
    address[] votedVoters;

    struct VotingResult {
        string[] choices;
        uint256[] votesReceived;
        uint256 totalVotes;
        string winningChoice;
        uint256 winningVotes;
    }
    VotingResult votingResult;

    constructor(VotingBuilder.VotingParameters memory _votingParams) {
        votingParams = _votingParams;
        votingResult.choices = _votingParams.choices;

        for (uint256 i = 0; i < _votingParams.choices.length; i++) {
            votingResult.votesReceived.push(0);
        }

        for(uint256 i = 0; i < _votingParams.votersPrivate.length; i++) {
            voters[_votingParams.votersPrivate[i]].isPrivate = true;

            if(_votingParams.isVotePowered) {
                voters[_votingParams.votersPrivate[i]].power = _votingParams.voterPowers[i];
            }
        }
    }

    function getVotingParams() public view onlyAdmin returns (VotingBuilder.VotingParameters memory)
{
        return votingParams;
    }

    function getIsActive() public view onlyOperator returns (bool) {
        return isActive;
    }

    function getVoter(address voter) public view onlyOperator returns (Voter memory) {
        require(votingParams.isVoterDisclosed, "Voting: Votes disclosure is not allowed");
        return voters[voter];
    }

    function getVotingResult() public view returns (VotingResult memory) {
        return votingResult;
    }

    function start() public onlyOperator {
        require(!isActive, "Voting: Voting already started");

        if (votingParams.isRepeated) {
            for (uint256 i = 0; i < votedVoters.length; i++) {
                voters[votedVoters[i]].voted = false;
            }

            delete votedVoters;
            votingResult.totalVotes = 0;
            votingResult.winningChoice = "";
            votingResult.winningVotes = 0;
            for (uint256 i = 0; i < votingParams.choices.length; i++) {
                votingResult.votesReceived[i] = 0;
            }
        }

        require(votingResult.totalVotes == 0, "Voting: Voting is already finished and is not
repeated");

        isActive = true;
    }

    function vote(uint256 choiceIndex) public onlyActive {
        require(!voters[msg.sender].voted, "Voting: Already voted");
        require(choiceIndex < votingResult.choices.length, "Voting: the selected Choice does not
exist");
```

```
        if(votingParams.isVotersLimited) {
            require(votedVoters.length < votingParams.votersLimit, "Voting: Voters Limit already
reached");
        }

        if(votingParams.isPrivate) {
            require(voters[msg.sender].isPrivate, "Voting: You are not among the allowed Private
Voters");
        }

        voters[msg.sender].voted = true;
        voters[msg.sender].vote = votingResult.choices[choiceIndex];
        votedVoters.push(msg.sender);

        if(votingParams.isVotePowered) {
            votingResult.totalVotes += voters[msg.sender].power;
            votingResult.votesReceived[choiceIndex] += voters[msg.sender].power;
        } else {
            votingResult.totalVotes++;
            votingResult.votesReceived[choiceIndex]++;
        }
    }

    function finishVoting() public onlyActive onlyOperator {
        uint256 maxVotes = 0;
        uint256 maxIndex = 0;
        bool doubleWinner = false;

        for (uint256 i = 0; i < votingResult.votesReceived.length; i++) {
            if (votingResult.votesReceived[i] >= maxVotes) {
                if(votingResult.votesReceived[i] == maxVotes) {
                    doubleWinner = true;
                    break;
                }

                maxVotes = votingResult.votesReceived[i];
                maxIndex = i;
            }
        }

        if(maxVotes == 0 || doubleWinner == true) {
            votingResult.winningChoice = "No winner: no votes or two winners";
            votingResult.winningVotes = maxVotes;
            isActive = false;
            return;
        }

        if(votingParams.isResultVetoed && 100 * maxVotes / votingResult.totalVotes != 100) {
            votingResult.winningChoice = "No winner: voting result was Votoed";
            votingResult.winningVotes = maxVotes;
            isActive = false;
            return;
        }

        if(votingParams.isResultPartial && 100 * maxVotes / votingResult.totalVotes <
votingParams.votesPercentageToWin) {
            votingResult.winningChoice = "No winner: No choice reached the required Votes Percentage
To Win";
            votingResult.winningVotes = maxVotes;
        } else {
            votingResult.winningChoice = votingResult.choices[maxIndex];
            votingResult.winningVotes = maxVotes;
        }

        isActive = false;
    }

    modifier onlyActive() {
        require(isActive, "Voting: Voting is deactivated");
        _;
    }
}
contract VotingBuilder is VotingRoleModel {
    Voting private voting;

    struct VotingParameters {
        bool isRepeated;
        bool isVoterDisclosed;
        bool isVotePowered;
        bool isVotersLimited;
        bool isPrivate;
```

```
        bool isResultPartial;
        bool isResultVetoed;
        bool isMultiChoiced;
        bool isFunded;

        uint256 fundingUsersLimit;
        uint256 votersLimit;
        uint256 votesPercentageToWin;

        address[] votersPrivate;
        uint256[] voterPowers;
        string[] choices;
    }
    VotingParameters votingParams;

    function getVotingParams() public view onlyAdmin returns (VotingParameters memory) {
        return votingParams;
    }
    function toggleIsFunded(uint256 usersAmount) public onlyAdmin {
        if (!votingParams.isFunded) {
            require(votingParams.isVotersLimited == true,"VotingBuilder: voting should has voters
limit to be able to fund voters");
            require(votingParams.isPrivate == true,"VotingBuilder: voting should has private voters
to be able to fund voters");

            require(usersAmount >= votingParams.votersLimit,"VotingBuilder: Funding users limit
cannot be lower than total voters limit");
            require(usersAmount == votingParams.votersPrivate.length,"VotingBuilder: Funding users
limit and Private Voters amount mismatch");

            votingParams.fundingUsersLimit = usersAmount;
            votingParams.isFunded = true;
        } else {
            votingParams.isFunded = false;
        }
    }
    function toggleIsRepeated() public onlyAdmin {
        if (!votingParams.isRepeated) {
            require(votingParams.isVotersLimited == true,"VotingBuilder: voting should has voters
Limit to be Repeated");
            require(votingParams.votersLimit <= 5000,"VotingBuilder: Repeated voting supports 5000
voters maximum ");

            votingParams.isFunded = false;
            votingParams.isRepeated = true;
        } else {
            votingParams.isRepeated = false;
        }
    }
    function toggleIsVoterDisclosed() public onlyAdmin {
        votingParams.isVoterDisclosed = !votingParams.isVoterDisclosed;
    }
    function toggleIsVotePowered(uint256[] memory powers) public onlyAdmin {
        if (!votingParams.isVotePowered) {
            require(votingParams.isVotersLimited == true,"VotingBuilder: voting should has voters
limit to grant power to voters");
            require(votingParams.isPrivate == true,"VotingBuilder: voting should has private voters
to grant power to voters");
            require(powers.length == votingParams.votersPrivate.length,"VotingBuilder: voter Powers
and allowed Private Voters arrays length mismatch");

            for (uint256 i = 0; i < powers.length; i++) {
                require(powers[i] >= 1,"VotingBuilder: voter Power must be at least 1");
                votingParams.voterPowers.push(powers[i]);
            }
            votingParams.isVotePowered = true;
        } else {
            delete votingParams.voterPowers;
            votingParams.isVotePowered = false;
        }
    }
    function toggleIsVotersLimited(uint256 _votersLimit) public onlyAdmin {
        if (!votingParams.isVotersLimited) {
            votingParams.votersLimit = _votersLimit;
            votingParams.isVotersLimited = true;
        } else {
            votingParams.isFunded = false;
            votingParams.isVotePowered = false;
            votingParams.isPrivate = false;
            votingParams.votersLimit = 0;
            votingParams.isVotersLimited = false;
        }
```

```
    }
    function toggleIsPrivate(address[] memory voters) public onlyAdmin {
        if (!votingParams.isPrivate) {
            require(voters.length <= 5000,"VotingBuilder: Private voting supports 5000 voters
maximum ");

            if(votingParams.isVotersLimited) {
                require(voters.length <= votingParams.votersLimit,"VotingBuilder: Private Voters
amount cannot be higher than total voters limit");
            }

            votingParams.votersPrivate = voters;
            votingParams.isPrivate = true;
        } else {
            delete votingParams.votersPrivate;
            votingParams.isFunded = false;
            votingParams.isVotePowered = false;
            votingParams.isPrivate = false;
        }
    }
    function toggleIsResultPartial(uint256 _percentage) public onlyAdmin {
        require(_percentage >= 0 && _percentage <= 100, "VotingBuilder: voting winning Percentage
must be between 0 and 100");
        if (_percentage >= 0 && _percentage <= 50) {
            votingParams.votesPercentageToWin = 0;
            votingParams.isResultPartial = false;
        } else {
            votingParams.votesPercentageToWin = _percentage;
            votingParams.isResultPartial = true;
        }
    }
    function toggleIsResultVetoed() public onlyAdmin {
        votingParams.isResultVetoed = !votingParams.isResultVetoed;
    }
    function toggleIsMultiChoiced(string[] memory _choices) public onlyAdmin {
        require(_choices.length <= 20 , "VotingBuilder: Max choices amount is 20");

        delete votingParams.choices;

        if(_choices.length <= 1) {
            votingParams.choices.push("yes");
            votingParams.choices.push("no");

            votingParams.isMultiChoiced = false;
        } else {
            votingParams.choices = _choices;
            votingParams.isMultiChoiced = true;
        }
    }
    function createVoting() public payable onlyAdmin noReentrancy {
        require(votingParams.choices.length > 1, "VotingBuilder: Voting cannot be created without
choices");
        if(votingParams.isFunded) {
            require(msg.value > 0, "VotingBuilder: No ether sent for funded voting");
            uint256 numAddresses = votingParams.votersPrivate.length;
            require(numAddresses > 0, "VotingBuilder: No addresses to distribute to");
            uint256 amountPerAddress = msg.value / numAddresses;

            bool success;
            for (uint256 i = 0; i < numAddresses; i++) {
                address payable recipient = payable(votingParams.votersPrivate[i]);
                (success, ) = recipient.call{value: amountPerAddress}("");
                require(success, string(abi.encodePacked("VotingBuilder: ether funding failed for
voter ", address(recipient))));
            }
        } else {
            require(msg.value == 0, "VotingBuilder: Funding is deactivated, TX value should be 0");
        }
        VotingParameters memory params = votingParams;
        voting = new Voting(params);
    }
    function getVoting() public view onlyAdmin returns (Voting) {
        return voting;
    }
    bool private locked;
    modifier noReentrancy() {
        require(!locked, "VotingBuilder: Reentrant call");
        locked = true;
        _;
        locked = false;
    }
}
```

ДОДАТОК В

Текст програми автоматичних тестів

voting-system-test.js:

```
const { expect } = require("chai");

let signers;
let votingbuilder;

beforeEach(async function () {
    votingbuilder = await ethers.deployContract("VotingBuilder");
    await votingbuilder.waitForDeployment();

    signers = await ethers.getSigners();
});

describe("1. Voting Builder", function () {
    it("101: check VotingParameters structure", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let params = await votingbuilder.getVotingParams();

        expect(params.isRepeated).to.be.a("boolean");
        expect(params.isVoterDisclosed).to.be.a("boolean");
        expect(params.isVotePowered).to.be.a("boolean");
        expect(params.isVotersLimited).to.be.a("boolean");
        expect(params.isPrivate).to.be.a("boolean");
        expect(params.isResultPartial).to.be.a("boolean");
        expect(params.isResultVetoed).to.be.a("boolean");
        expect(params.isMultiChoiced).to.be.a("boolean");
        expect(params.isFunded).to.be.a("boolean");

        expect(params.fundingUsersLimit).to.be.a("bigint");
        expect(params.votersLimit).to.be.a("bigint");
        expect(params.votesPercentageToWin).to.be.a("bigint");

        expect(params.votersPrivate).to.be.an("array");
        expect(params.voterPowers).to.be.an("array");
        expect(params.choices).to.be.an("array");
    });

    it("102: check VotingParameters default values", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let params = await votingbuilder.getVotingParams();

        expect(params.isRepeated).to.equal(false);
        expect(params.isVoterDisclosed).to.equal(false);
        expect(params.isVotePowered).to.equal(false);
        expect(params.isVotersLimited).to.equal(false);
        expect(params.isPrivate).to.equal(false);
        expect(params.isResultPartial).to.equal(false);
        expect(params.isResultVetoed).to.equal(false);
        expect(params.isMultiChoiced).to.equal(false);
        expect(params.isFunded).to.equal(false);

        expect(params.fundingUsersLimit).to.equal(0n);
        expect(params.votersLimit).to.equal(0n);
        expect(params.votesPercentageToWin).to.equal(0n);

        expect(params.votersPrivate).to.have.lengthOf(0);
        expect(params.voterPowers).to.have.lengthOf(0);
        expect(params.choices).to.have.lengthOf(0);
    });

    it("103: getVoting() returns 0-address before deploy", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let defaultAddr = await votingbuilder.getVoting();

        expect(defaultAddr).to.equal("0x0000000000000000000000000000000000000000");
    });

    it("104: toggleIsVotersLimited() works properly (on/off)", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let limitValue = 100;
        let params;

        await votingbuilder.toggleIsVotersLimited(limitValue);
```

```
        params = await votingbuilder.getVotingParams();

        expect(params.votersLimit).to.equal(limitValue);
        expect(params.isVotersLimited).to.equal(true);

        await votingbuilder.toggleIsVotersLimited(0);

        params = await votingbuilder.getVotingParams();

        expect(params.votersLimit).to.equal(0);
        expect(params.isVotersLimited).to.equal(false);
    });

    it("105: toggleIsVotersLimited() false disables isFunded, isVotePowered, isPrivate", async
function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);

        await votingbuilder.toggleIsVotersLimited(100);
        await votingbuilder.toggleIsVotersLimited(0);

        let params = await votingbuilder.getVotingParams();

        expect(params.isFunded).to.equal(false);
        expect(params.isVotePowered).to.equal(false);
        expect(params.isPrivate).to.equal(false);
    });

    it("106: toggleIsVoterDisclosed() works properly (on/off)", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);

        await votingbuilder.toggleIsVoterDisclosed();
        let params = await votingbuilder.getVotingParams();

        expect(params.isVoterDisclosed).to.equal(true);

        await votingbuilder.toggleIsVoterDisclosed();
        params = await votingbuilder.getVotingParams();

        expect(params.isVoterDisclosed).to.equal(false);
    });

    it("107: toggleIsVotePowered() works properly (on/off)", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let voterPowers = [2, 3, 4];

        await votingbuilder.toggleIsVotersLimited(100);
        await votingbuilder.toggleIsPrivate([signers[0], signers[1], signers[2]]);

        await votingbuilder.toggleIsVotePowered(voterPowers);
        let params = await votingbuilder.getVotingParams();

        expect(params.isVotePowered).to.equal(true);
        expect(params.voterPowers.map(Number)).to.eql(voterPowers);

        await votingbuilder.toggleIsVotePowered([]);
        params = await votingbuilder.getVotingParams();

        expect(params.isVotePowered).to.equal(false);
    });

    it("108: toggleIsVotePowered() false clears voterPowers[]", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let voterPowers = [2, 3, 4];

        await votingbuilder.toggleIsVotersLimited(100);
        await votingbuilder.toggleIsPrivate([signers[0], signers[1], signers[2]]);

        await votingbuilder.toggleIsVotePowered(voterPowers);
        await votingbuilder.toggleIsVotePowered([]);
        let params = await votingbuilder.getVotingParams();

        expect(params.voterPowers).to.eql([]);
    });

    it("109: toggleIsVotePowered() true requires isVotersLimited true", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let voterPowers = [2, 3, 4];

        let message = "VotingBuilder: voting should has voters limit to grant power to voters";
        testError(async () => { return votingbuilder.toggleIsVotePowered(voterPowers); }, message);
        let params = await votingbuilder.getVotingParams();
```

```
        expect(params.isVotePowered).to.equal(false);
        expect(params.voterPowers).to.eql([]);
    });

    it("110: toggleIsVotePowered() true requires isPrivate true", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let voterPowers = [2, 3, 4];
        await votingbuilder.toggleIsVotersLimited(100);

        let message = "VotingBuilder: voting should has private voters to grant power to voters";
        testError(async () => { return votingbuilder.toggleIsVotePowered(voterPowers); }, message);
        let params = await votingbuilder.getVotingParams();

        expect(params.isVotePowered).to.equal(false);
        expect(params.voterPowers).to.eql([]);
    });

    it("111: toggleIsVotePowered() true requires equal length powers-votersPrivate", async function
() {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let voterPowers = [2, 3, 4];

        await votingbuilder.toggleIsVotersLimited(100);
        await votingbuilder.toggleIsPrivate([signers[0], signers[1]]);

        let message = "VotingBuilder: voter Powers and allowed Private Voters arrays length
mismatch";
        testError(async () => { return votingbuilder.toggleIsVotePowered(voterPowers); }, message);
        let params = await votingbuilder.getVotingParams();

        expect(params.isVotePowered).to.equal(false);
        expect(params.voterPowers).to.eql([]);
    });

    it("112: toggleIsVotePowered() true requires power >=1", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let voterPowers = [1, 0, 4];

        await votingbuilder.toggleIsVotersLimited(100);
        await votingbuilder.toggleIsPrivate([signers[0], signers[1], signers[2]]);

        let message = "VotingBuilder: voter Power must be at least 1";
        testError(async () => { return votingbuilder.toggleIsVotePowered(voterPowers); }, message);
        let params = await votingbuilder.getVotingParams();

        expect(params.isVotePowered).to.equal(false);
        expect(params.voterPowers).to.eql([]);
    });

    it("113: toggleIsResultVetoed() works properly (on/off)", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);

        await votingbuilder.toggleIsResultVetoed();
        let params = await votingbuilder.getVotingParams();

        expect(params.isResultVetoed).to.equal(true);

        await votingbuilder.toggleIsResultVetoed();
        params = await votingbuilder.getVotingParams();

        expect(params.isResultVetoed).to.equal(false);
    });

    it("114: toggleIsResultPartial() true if % = 51-100", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let percentages = [51, 75, 100];

        for (let i = 0; i < percentages.length; i++) {
            await votingbuilder.toggleIsResultPartial(percentages[i]);
            let params = await votingbuilder.getVotingParams();

            expect(params.isResultPartial).to.equal(true);
            expect(params.votesPercentageToWin).to.equal(percentages[i]);
        }
    });

    it("115: toggleIsResultPartial() false if % = 0-50", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let percentages = [0, 25, 50];

        for (let i = 0; i < percentages.length; i++) {
```

```
        await votingbuilder.toggleIsResultPartial(percentages[i]);
        let params = await votingbuilder.getVotingParams();

        expect(params.isResultPartial).to.equal(false);
        expect(params.votesPercentageToWin).to.equal(0);
    }
});

it("116: toggleIsResultPartial() requires % = 0-100", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);

    let message = "VotingBuilder: voting winning Percentage must be between 0 and 100";
    testError(async () => { return votingbuilder.toggleIsResultPartial(111); }, message);
    let params = await votingbuilder.getVotingParams();

    expect(params.isResultPartial).to.equal(false);
    expect(params.votesPercentageToWin).to.equal(0);
});

it("117: toggleIsRepeated() works properly (on/off)", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsVotersLimited(100);

    await votingbuilder.toggleIsRepeated();
    let params = await votingbuilder.getVotingParams();

    expect(params.isRepeated).to.equal(true);

    await votingbuilder.toggleIsRepeated();
    params = await votingbuilder.getVotingParams();

    expect(params.isRepeated).to.equal(false);
});

it("118: toggleIsRepeated() true disables isFunded", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsVotersLimited(100);

    await votingbuilder.toggleIsRepeated();
    let params = await votingbuilder.getVotingParams();

    expect(params.isFunded).to.equal(false);
    expect(params.isRepeated).to.equal(true);
});

it("119: toggleIsRepeated() true requires isVotersLimited true", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);

    let message = "VotingBuilder: voting should has voters Limit to be Repeated";
    testError(async () => { return votingbuilder.toggleIsRepeated(); }, message);
    let params = await votingbuilder.getVotingParams();

    expect(params.isRepeated).to.equal(false);
});

it("120: toggleIsRepeated() true requires votersLimit <= 5000", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsVotersLimited(10000);

    let message = "VotingBuilder: Repeated voting supports 5000 voters maximum";
    testError(async () => { return votingbuilder.toggleIsRepeated(); }, message);
    let params = await votingbuilder.getVotingParams();

    expect(params.isRepeated).to.equal(false);
});

it("121: toggleIsPrivate() true requires voters <= 5000", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    let addresses = [];
    for (let i = 0; i < 5001; i++) {
        addresses.push(signers[0].address);
    }

    let message = "VotingBuilder: Private voting supports 5000 voters maximum";
    testError(async () => { return votingbuilder.toggleIsPrivate(addresses); }, message);
    let params = await votingbuilder.getVotingParams();

    expect(params.votersPrivate).to.eql([]);
    expect(params.isPrivate).to.equal(false);
});
```

```
    it("122: toggleIsPrivate() true requires voters <= votersLimit if isVotersLimited true", async
function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsVotersLimited(5);
        let addresses = [];
        for (let i = 0; i < 10; i++) {
            addresses.push(signers[0].address);
        }

        let message = "VotingBuilder: Private Voters amount cannot be higher than total voters
limit";
        testError(async () => { return votingbuilder.toggleIsPrivate(addresses); }, message);
        let params = await votingbuilder.getVotingParams();

        expect(params.isPrivate).to.equal(false);
    });

    it("123: toggleIsPrivate() works properly (on/off)", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let addresses = [];
        for (let i = 0; i < 10; i++) {
            addresses.push(signers[0].address);
        }

        await votingbuilder.toggleIsPrivate(addresses);
        let params = await votingbuilder.getVotingParams();

        expect(params.isPrivate).to.equal(true);
        expect(params.votersPrivate).to.eql(addresses);

        await votingbuilder.toggleIsPrivate([]);
        params = await votingbuilder.getVotingParams();

        expect(params.isPrivate).to.equal(false);
    });

    it("124: toggleIsPrivate() false disables isFunded, isVotePowered", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);

        await votingbuilder.toggleIsPrivate([signers[0].address]);
        await votingbuilder.toggleIsPrivate([]);
        let params = await votingbuilder.getVotingParams();

        expect(params.isFunded).to.equal(false);
        expect(params.isVotePowered).to.equal(false);
        expect(params.isPrivate).to.equal(false);
    });

    it("125: toggleIsPrivate() false clears votersPrivate[]", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);

        await votingbuilder.toggleIsPrivate([signers[0].address]);
        await votingbuilder.toggleIsPrivate([]);
        let params = await votingbuilder.getVotingParams();

        expect(params.votersPrivate).to.eql([]);
        expect(params.isPrivate).to.equal(false);
    });

    it("126: toggleIsMultiChoiced() requires choices <= 20", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let choices = [];
        for (let i = 0; i < 25; i++) {
            choices.push("choice #" + i);
        }

        let message = "VotingBuilder: Max choices amount is 20";
        testError(async () => { return votingbuilder.toggleIsMultiChoiced(choices); }, message);
        let params = await votingbuilder.getVotingParams();

        expect(params.choices).to.eql([]);
    });

    it("127: toggleIsMultiChoiced() false and sets yes/no if choices <= 1", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);

        await votingbuilder.toggleIsMultiChoiced([]);
        let params = await votingbuilder.getVotingParams();

        expect(params.choices).to.eql(["yes", "no"]);
        expect(params.isMultiChoiced).to.equal(false);
    });
```

```
    it("128: toggleIsMultiChoiced() true and sets input choices if choices 2-20", async function ()
{
        await votingbuilder.grantRoleAdmin(signers[0].address);
        let choices = [];
        for (let i = 0; i < 10; i++) {
            choices.push("choice #" + i);
        }

        await votingbuilder.toggleIsMultiChoiced(choices);
        let params = await votingbuilder.getVotingParams();

        expect(params.choices).to.eql(choices);
        expect(params.isMultiChoiced).to.equal(true);
    });

    it("129: toggleIsFunded() true requires isVotersLimited true", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);

        let message = "VotingBuilder: voting should has voters limit to be able to fund voters";
        testError(async () => { return votingbuilder.toggleIsFunded(10); }, message);
        let params = await votingbuilder.getVotingParams();

        expect(params.fundingUsersLimit).to.equal(0);
        expect(params.isFunded).to.equal(false);
    });

    it("130: toggleIsFunded() true requires isPrivate true", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsVotersLimited(100);

        let message = "VotingBuilder: voting should has private voters to be able to fund voters";
        testError(async () => { return votingbuilder.toggleIsFunded(10); }, message);
        let params = await votingbuilder.getVotingParams();

        expect(params.fundingUsersLimit).to.equal(0);
        expect(params.isFunded).to.equal(false);
    });

    it("131: toggleIsFunded() true requires usersAmount >= votersLimit", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsVotersLimited(100);
        let addresses = [];
        for (let i = 0; i < 100; i++) {
            addresses.push(signers[0].address);
        }
        await votingbuilder.toggleIsPrivate(addresses);

        let message = "VotingBuilder: Funding users limit cannot be lower than total voters limit";
        testError(async () => { return votingbuilder.toggleIsFunded(10); }, message);
        let params = await votingbuilder.getVotingParams();

        expect(params.fundingUsersLimit).to.equal(0);
        expect(params.isFunded).to.equal(false);
    });

    it("132: toggleIsFunded() true requires usersAmount = votersPrivate", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsVotersLimited(10);
        let addresses = [];
        for (let i = 0; i < 10; i++) {
            addresses.push(signers[0].address);
        }
        await votingbuilder.toggleIsPrivate(addresses);

        let message = "VotingBuilder: Funding users limit and Private Voters amount mismatch";
        testError(async () => { return votingbuilder.toggleIsFunded(15); }, message);
        let params = await votingbuilder.getVotingParams();

        expect(params.fundingUsersLimit).to.equal(0);
        expect(params.isFunded).to.equal(false);
    });

    it("133: toggleIsFunded() works properly (on/off)", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsVotersLimited(10);
        let usersAmount = 10;
        let addresses = [];
        for (let i = 0; i < 10; i++) {
            addresses.push(signers[0].address);
        }
        await votingbuilder.toggleIsPrivate(addresses);
```

```
        await votingbuilder.toggleIsFunded(usersAmount);
        let params = await votingbuilder.getVotingParams();

        expect(params.fundingUsersLimit).to.equal(usersAmount);
        expect(params.isFunded).to.equal(true);

        await votingbuilder.toggleIsFunded(0);
        params = await votingbuilder.getVotingParams();

        expect(params.isFunded).to.equal(false);
    });

    it("134: createVoting() requires choices > 1", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);

        let message = "VotingBuilder: Voting cannot be created without choices";
        testError(async () => { return votingbuilder.createVoting(); }, message);

        let addr = await votingbuilder.getVoting();
        expect(addr).to.equal("0x0000000000000000000000000000000000000000");
    });

    it("135: createVoting() requires msg.value > 0 if isFunded true", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        await votingbuilder.toggleIsVotersLimited(3);
        await votingbuilder.toggleIsPrivate([signers[1].address, signers[2].address,
signers[3].address]);
        await votingbuilder.toggleIsFunded(3);

        let message = "VotingBuilder: No ether sent for funded voting";
        testError(async () => { return votingbuilder.createVoting(); }, message);

        let addr = await votingbuilder.getVoting();
        expect(addr).to.equal("0x0000000000000000000000000000000000000000");
    });

    it("136: createVoting() requires votersPrivate > 0 if isFunded true", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        await votingbuilder.toggleIsVotersLimited(0);
        await votingbuilder.toggleIsPrivate([]);
        await votingbuilder.toggleIsFunded(0);

        let message = "VotingBuilder: No addresses to distribute to";
        testError(async () => { return votingbuilder.createVoting(); }, message);

        let addr = await votingbuilder.getVoting();
        expect(addr).to.equal("0x0000000000000000000000000000000000000000");
    });

    it("137: createVoting() funds all votersPrivate if isFunded true", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        await votingbuilder.toggleIsVotersLimited(3);
        await votingbuilder.toggleIsPrivate([signers[1].address, signers[2].address,
signers[3].address]);
        await votingbuilder.toggleIsFunded(3);

        let initialBalances = await Promise.all(
            [signers[1].address, signers[2].address, signers[3].address].map(async (address) => {
                return await ethers.provider.getBalance(address);
            })
        );

        await votingbuilder.createVoting({ value: ethers.parseEther("0.3") });

        let finalBalances = await Promise.all(
            [signers[1].address, signers[2].address, signers[3].address].map(async (address) => {
                return await ethers.provider.getBalance(address);
            })
        );

        for (let i = 0; i < initialBalances.length; i++) {
            expect(finalBalances[i]).to.be.gt(initialBalances[i]);
            expect(finalBalances[i] - initialBalances[i]).to.equal(ethers.parseEther("0.1"));
        }

        let addr = await votingbuilder.getVoting();
        expect(addr).to.not.equal("0x0000000000000000000000000000000000000000");
    });
```

```
    it("138: createVoting() requires msg.value = 0 if isFunded false", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);

        let message = "VotingBuilder: Funding is deactivated, TX value should be 0";
        testError(async () => { return votingbuilder.createVoting({ value: ethers.parseEther("0.3")
}); }, message);

        let addr = await votingbuilder.getVoting();
        expect(addr).to.equal("0x0000000000000000000000000000000000000000");
    });

    it("139: createVoting() deploys voting properly", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);

        await votingbuilder.createVoting();

        let addr = await votingbuilder.getVoting();
        expect(addr).to.not.equal("0x0000000000000000000000000000000000000000");
    });
});

describe("2. Voting", function () {
    it("201: check VotingParameters structure is equal to VotingBuilder", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        let instance = await deployVoting();

        await instance.grantRoleAdmin(signers[0].address);
        let params = await instance.getVotingParams();

        expect(params.isRepeated).to.be.a("boolean");
        expect(params.isVoterDisclosed).to.be.a("boolean");
        expect(params.isVotePowered).to.be.a("boolean");
        expect(params.isVotersLimited).to.be.a("boolean");
        expect(params.isPrivate).to.be.a("boolean");
        expect(params.isResultPartial).to.be.a("boolean");
        expect(params.isResultVetoed).to.be.a("boolean");
        expect(params.isMultiChoiced).to.be.a("boolean");
        expect(params.isFunded).to.be.a("boolean");

        expect(params.fundingUsersLimit).to.be.a("bigint");
        expect(params.votersLimit).to.be.a("bigint");
        expect(params.votesPercentageToWin).to.be.a("bigint");

        expect(params.votersPrivate).to.be.an("array");
        expect(params.voterPowers).to.be.an("array");
        expect(params.choices).to.be.an("array");
    });

    it("202: votingParams accepts params from VotingBuilder properly", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsVotersLimited(100);
        await votingbuilder.toggleIsPrivate([signers[1], signers[2], signers[3]]);
        await votingbuilder.toggleIsVotePowered([2, 3, 4]);
        await votingbuilder.toggleIsRepeated();
        await votingbuilder.toggleIsMultiChoiced([]);

        let instance = await deployVoting();
        await instance.grantRoleAdmin(signers[0].address);

        let paramsBuilder = await votingbuilder.getVotingParams();
        let paramsVoting = await instance.getVotingParams();

        expect(paramsVoting.isRepeated).to.equal(paramsBuilder.isRepeated);
        expect(paramsVoting.isVoterDisclosed).to.equal(paramsBuilder.isVoterDisclosed);
        expect(paramsVoting.isVotePowered).to.equal(paramsBuilder.isVotePowered);
        expect(paramsVoting.isVotersLimited).to.equal(paramsBuilder.isVotersLimited);
        expect(paramsVoting.isPrivate).to.equal(paramsBuilder.isPrivate);
        expect(paramsVoting.isResultPartial).to.equal(paramsBuilder.isResultPartial);
        expect(paramsVoting.isResultVetoed).to.equal(paramsBuilder.isResultVetoed);
        expect(paramsVoting.isMultiChoiced).to.equal(paramsBuilder.isMultiChoiced);
        expect(paramsVoting.isFunded).to.equal(paramsBuilder.isFunded);

        expect(paramsVoting.votersLimit).to.equal(paramsBuilder.votersLimit);
        expect(paramsVoting.votersPrivate).to.eql(paramsBuilder.votersPrivate);
        expect(paramsVoting.voterPowers).to.eql(paramsBuilder.voterPowers);
        expect(paramsVoting.choices).to.eql(paramsBuilder.choices);
    });
```

```
it("203: check VotingResult structure", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    let instance = await deployVoting();

    await instance.grantRoleAdmin(signers[0].address);
    let result = await instance.getVotingResult();

    expect(result.choices).to.be.an("array");
    expect(result.votesReceived).to.be.an("array");
    expect(result.totalVotes).to.be.a("bigint");
    expect(result.winningChoice).to.be.a("string");
    expect(result.winningVotes).to.be.a("bigint");
});

it("204: check VotingResult default values", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    let instance = await deployVoting();

    await instance.grantRoleAdmin(signers[0].address);
    let result = await instance.getVotingResult();

    expect(result.votesReceived).to.deep.equal([BigInt(0), BigInt(0)]);
    expect(result.totalVotes).to.equal(0);
    expect(result.winningChoice).to.equal("");
    expect(result.winningVotes).to.equal(0);
});

it("205: votingResult accepts choices from votingParams", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    let instance = await deployVoting();

    await instance.grantRoleAdmin(signers[0].address);
    let params = await instance.getVotingParams();
    let result = await instance.getVotingResult();

    expect(result.choices).to.eql(params.choices);
});

it("206: check Voter structure", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    await votingbuilder.toggleIsVoterDisclosed();
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    let voter = await instance.getVoter(signers[0].address);

    expect(voter.voted).to.be.a("boolean");
    expect(voter.power).to.be.a("bigint");
    expect(voter.vote).to.be.a("string");
    expect(voter.isPrivate).to.be.a("boolean");
});

it("207: check Voter default values", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    await votingbuilder.toggleIsVoterDisclosed();
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    let voter = await instance.getVoter(signers[0].address);

    expect(voter.voted).to.equal(false);
    expect(voter.vote).to.equal("");
    expect(voter.power).to.equal(0);
    expect(voter.isPrivate).to.equal(false);
});

it("208: voters accepts votersPrivate from votingParams", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    await votingbuilder.toggleIsVoterDisclosed();
    await votingbuilder.toggleIsPrivate([signers[1], signers[2], signers[3]]);
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    let voter1 = await instance.getVoter(signers[1].address);
    let voter2 = await instance.getVoter(signers[2].address);
    let voter3 = await instance.getVoter(signers[3].address);
```

```
        expect(voter1.isPrivate).to.equal(true);
        expect(voter2.isPrivate).to.equal(true);
        expect(voter3.isPrivate).to.equal(true);
    });

    it("209: voters accepts voterPowers from votingParams if isVotePowered true", async function ()
{
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        await votingbuilder.toggleIsVoterDisclosed();
        await votingbuilder.toggleIsVotersLimited(100);
        await votingbuilder.toggleIsPrivate([signers[1], signers[2], signers[3]]);
        await votingbuilder.toggleIsVotePowered([2, 3, 4]);
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        let voter1 = await instance.getVoter(signers[1].address);
        let voter2 = await instance.getVoter(signers[2].address);
        let voter3 = await instance.getVoter(signers[3].address);

        expect(voter1.power).to.equal(2);
        expect(voter2.power).to.equal(3);
        expect(voter3.power).to.equal(4);
    });

    it("210: getVoter() requires isVoterDisclosed true", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        let message = "Voting: Votes disclosure is not allowed";
        let voter = undefined;
        voter = await testError(async () => { return instance.getVoter(signers[1].address); },
message);

        expect(voter).to.equal(undefined);
    });

    it("211: isActive false by default", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        let isActive = await instance.getIsActive();

        expect(isActive).to.equal(false);
    });

    it("212: vote() requires onlyActive", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        let instance = await deployVoting();

        let message = "Voting: Voting is deactivated";
        await testError(async () => { return instance.vote(0); }, message);

        await instance.grantRoleOperator(signers[0].address);
        let result = await instance.getVotingResult();

        expect(result.votesReceived).to.deep.equal([BigInt(0), BigInt(0)]);
    });

    it("213: vote() restricts double-voting", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        await instance.start();
        await instance.vote(0);

        let message = "Voting: Already voted";
        await testError(async () => { return instance.vote(1); }, message);

        let result = await instance.getVotingResult();
        expect(result.votesReceived[0]).to.equal(1);
        expect(result.votesReceived[1]).to.equal(0);
    });
```

```
it("214: vote() requires the selected choice to exist", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    await instance.start();

    let message = "Voting: the selected Choice does not exist";
    await testError(async () => { return instance.vote(2); }, message);

    let result = await instance.getVotingResult();
    expect(result.votesReceived[0]).to.equal(0);
    expect(result.votesReceived[1]).to.equal(0);
});

it("215: vote() checks the votersLimit reached if isVotersLimited true", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    await votingbuilder.toggleIsVotersLimited(1);
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    await instance.start();
    await instance.vote(0);

    let message = "Voting: Voters Limit already reached";
    await testError(async () => { return instance.connect(signers[1]).vote(1); }, message);

    let result = await instance.getVotingResult();
    expect(result.votesReceived[0]).to.equal(1);
    expect(result.votesReceived[1]).to.equal(0);
});

it("216: vote() requires voter to be white-listed if isPrivate true", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    await votingbuilder.toggleIsPrivate([signers[0].address]);
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    await instance.start();

    let message = "Voting: You are not among the allowed Private Voters";
    await testError(async () => { return instance.connect(signers[1]).vote(0); }, message);

    let result = await instance.getVotingResult();
    expect(result.votesReceived[0]).to.equal(0);
});

it("217: vote() works properly (vote, voted, totalVotes, votesReceived)", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    await votingbuilder.toggleIsVoterDisclosed();
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    await instance.start();

    await instance.connect(signers[1]).vote(0);
    await instance.connect(signers[2]).vote(1);
    await instance.connect(signers[3]).vote(0);

    let voter1 = await instance.getVoter(signers[1].address);
    let voter2 = await instance.getVoter(signers[2].address);
    let voter3 = await instance.getVoter(signers[3].address);
    let result = await instance.getVotingResult();

    expect(voter1.voted).to.equal(true);
    expect(voter2.voted).to.equal(true);
    expect(voter3.voted).to.equal(true);
    expect(voter1.vote).to.equal("yes");
    expect(voter2.vote).to.equal("no");
    expect(voter3.vote).to.equal("yes");
    expect(result.totalVotes).to.equal(3);
    expect(result.votesReceived).to.deep.equal([BigInt(2), BigInt(1)]);
});

it("218: vote() counts power if isVotePowered true", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    await votingbuilder.toggleIsVoterDisclosed();
```

```
        await votingbuilder.toggleIsVotersLimited(100);
        await votingbuilder.toggleIsPrivate([signers[1], signers[2], signers[3]]);
        await votingbuilder.toggleIsVotePowered([2, 3, 4]);
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        await instance.start();

        await instance.connect(signers[1]).vote(0);
        await instance.connect(signers[2]).vote(1);
        await instance.connect(signers[3]).vote(0);

        let result = await instance.getVotingResult();
        expect(result.totalVotes).to.equal(9);
        expect(result.votesReceived).to.deep.equal([BigInt(6), BigInt(3)]);
    });

    it("219: start() requires isActive false", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        await votingbuilder.toggleIsVotersLimited(100);
        await votingbuilder.toggleIsRepeated();
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        await instance.start();
        await instance.vote(0);

        let message = "Voting: Voting already started";
        await testError(async () => { return instance.start(); }, message);

        let result = await instance.getVotingResult();
        expect(result.totalVotes).to.equal(1);
    });

    it("220: start() clears voted, votesReceived, totalVotes, winningChoice, winningVotes if
isRepeated true", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        await votingbuilder.toggleIsVotersLimited(100);
        await votingbuilder.toggleIsRepeated();
        await votingbuilder.toggleIsVoterDisclosed();
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        await instance.start();
        await instance.connect(signers[1]).vote(0);
        await instance.connect(signers[2]).vote(1);
        await instance.connect(signers[3]).vote(0);
        await instance.finishVoting();
        await instance.start();

        let voter1 = await instance.getVoter(signers[1].address);
        let voter2 = await instance.getVoter(signers[2].address);
        let voter3 = await instance.getVoter(signers[3].address);
        let result = await instance.getVotingResult();
        let isActive = await instance.getIsActive();

        expect(voter1.voted).to.equal(false);
        expect(voter2.voted).to.equal(false);
        expect(voter3.voted).to.equal(false);
        expect(result.totalVotes).to.equal(0);
        expect(result.winningChoice).to.equal("");
        expect(result.winningVotes).to.equal(0);
        expect(result.votesReceived).to.deep.equal([BigInt(0), BigInt(0)]);
        expect(isActive).to.equal(true);
    });

    it("221: start() requires totalVotes = 0 (not repeated)", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        await instance.start();
        await instance.connect(signers[1]).vote(0);
        await instance.connect(signers[2]).vote(1);
        await instance.connect(signers[3]).vote(0);
        await instance.finishVoting();

        let message = "Voting: Voting is already finished and is not repeated";
        await testError(async () => { return instance.start(); }, message);
```

```
    let isActive = await instance.getIsActive();
    let result = await instance.getVotingResult();
    expect(result.totalVotes).to.equal(3);
    expect(result.winningChoice).to.equal("yes");
    expect(result.winningVotes).to.equal(2);
    expect(result.votesReceived).to.deep.equal([BigInt(2), BigInt(1)]);
    expect(isActive).to.equal(false);
});

it("222: start() enables voting properly", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    await instance.start();

    let isActive = await instance.getIsActive();
    expect(isActive).to.equal(true);
});

it("223: finishVoting() requires onlyActive", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    let message = "Voting: Voting is deactivated";
    await testError(async () => { return instance.finishVoting(); }, message);

    let result = await instance.getVotingResult();
    expect(result.winningChoice).to.equal("");
});

it("224: finishVoting() returns no winner if no votes made", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    await instance.start();
    await instance.finishVoting()

    let result = await instance.getVotingResult();
    let isActive = await instance.getIsActive();
    expect(isActive).to.equal(false);
    expect(result.winningChoice).to.equal("No winner: no votes or two winners");
    expect(result.winningVotes).to.equal(0);
});

it("225: finishVoting() returns no winner if two winners", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    await instance.start();
    await instance.connect(signers[1]).vote(0);
    await instance.connect(signers[2]).vote(1);
    await instance.finishVoting()

    let result = await instance.getVotingResult();
    let isActive = await instance.getIsActive();
    expect(isActive).to.equal(false);
    expect(result.winningChoice).to.equal("No winner: no votes or two winners");
    expect(result.winningVotes).to.equal(1);
});

it("226: finishVoting() returns no winner if result was vetoed", async function () {
    await votingbuilder.grantRoleAdmin(signers[0].address);
    await votingbuilder.toggleIsMultiChoiced([]);
    await votingbuilder.toggleIsResultVetoed();
    let instance = await deployVoting();

    await instance.grantRoleOperator(signers[0].address);
    await instance.start();
    await instance.connect(signers[1]).vote(0);
    await instance.connect(signers[2]).vote(0);
    await instance.connect(signers[3]).vote(1);
    await instance.finishVoting()
```

```
        let result = await instance.getVotingResult();
        let isActive = await instance.getIsActive();
        expect(isActive).to.equal(false);
        expect(result.winningChoice).to.equal("No winner: voting result was Votoed");
        expect(result.winningVotes).to.equal(2);
    });

    it("227: finishVoting() returns no winner if no choice reached needed % of votes", async
function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        await votingbuilder.toggleIsResultPartial(75);
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        await instance.start();
        await instance.connect(signers[1]).vote(0);
        await instance.connect(signers[2]).vote(0);
        await instance.connect(signers[3]).vote(0);
        await instance.connect(signers[4]).vote(1);
        await instance.connect(signers[5]).vote(1);
        await instance.finishVoting()

        let result = await instance.getVotingResult();
        let isActive = await instance.getIsActive();
        expect(isActive).to.equal(false);
        expect(result.winningChoice).to.equal("No winner: No choice reached the required Votes
Percentage To Win");
        expect(result.winningVotes).to.equal(3);
    });

    it("228: finishVoting() returns winning votes, choice properly if isResultVetoed true", async
function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        await votingbuilder.toggleIsResultVetoed();
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        await instance.start();
        await instance.connect(signers[1]).vote(0);
        await instance.connect(signers[2]).vote(0);
        await instance.connect(signers[3]).vote(0);
        await instance.finishVoting()

        let result = await instance.getVotingResult();
        let isActive = await instance.getIsActive();
        expect(isActive).to.equal(false);
        expect(result.winningChoice).to.equal("yes");
        expect(result.winningVotes).to.equal(3);
    });

    it("229: finishVoting() returns winning votes, choice properly if isResultPartial true", async
function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        await votingbuilder.toggleIsResultPartial(60);
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        await instance.start();
        await instance.connect(signers[1]).vote(0);
        await instance.connect(signers[2]).vote(0);
        await instance.connect(signers[3]).vote(0);
        await instance.connect(signers[4]).vote(1);
        await instance.connect(signers[5]).vote(1);
        await instance.finishVoting()

        let result = await instance.getVotingResult();
        let isActive = await instance.getIsActive();
        expect(isActive).to.equal(false);
        expect(result.winningChoice).to.equal("yes");
        expect(result.winningVotes).to.equal(3);
    });

    it("230: finishVoting() returns winning votes, choice properly for two choices", async function
() {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
```

```
        await instance.start();
        await instance.connect(signers[1]).vote(0);
        await instance.connect(signers[2]).vote(0);
        await instance.connect(signers[3]).vote(1);
        await instance.connect(signers[4]).vote(1);
        await instance.connect(signers[5]).vote(1);
        await instance.finishVoting()

        let result = await instance.getVotingResult();
        let isActive = await instance.getIsActive();
        expect(isActive).to.equal(false);
        expect(result.winningChoice).to.equal("no");
        expect(result.winningVotes).to.equal(3);
    });

    it("231: finishVoting() returns winning votes, choice properly for multiple choices", async
function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced(["BTC", "ETH", "BNB"]);
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        await instance.start();
        await instance.connect(signers[1]).vote(0);
        await instance.connect(signers[2]).vote(0);
        await instance.connect(signers[3]).vote(1);
        await instance.connect(signers[4]).vote(1);
        await instance.connect(signers[5]).vote(2);
        await instance.connect(signers[6]).vote(1);
        await instance.finishVoting()

        let result = await instance.getVotingResult();
        let isActive = await instance.getIsActive();
        expect(isActive).to.equal(false);
        expect(result.winningChoice).to.equal("ETH");
        expect(result.winningVotes).to.equal(3);
    });

    it("232: finishVoting() disables voting", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await votingbuilder.toggleIsMultiChoiced([]);
        let instance = await deployVoting();

        await instance.grantRoleOperator(signers[0].address);
        await instance.start();
        await instance.finishVoting()

        let isActive = await instance.getIsActive();
        expect(isActive).to.equal(false);
    });
});

describe("3. Role Model in Voting Builder", function () {
    let onlyAdmin = "Only Admin can perform this action";
    let onlyOwner = "Only owner can perform this action";

    it("301: getVotingParams() requires onlyAdmin", async function () {
        let params = await testError(async () => { return votingbuilder.getVotingParams(); },
onlyAdmin);
        expect(params).to.equal(undefined);
    });

    it("302: getVoting() requires onlyAdmin", async function () {
        let addr = await testError(async () => { return votingbuilder.getVoting(); }, onlyAdmin);
        expect(addr).to.equal(undefined);
    });

    it("303: owner was set properly", async function () {
        let owner = await votingbuilder.getOwner();
        expect(owner).to.equal(signers[0].address);
    });

    it("304: owner is not operator by default", async function () {
        let ifOperator = await votingbuilder.checkOperator(signers[0].address);
        expect(ifOperator).to.equal(false);
    });

    it("305: owner is not admin by default", async function () {
        let ifAdmin = await votingbuilder.checkAdmin(signers[0].address);
        expect(ifAdmin).to.equal(false);
    });
```

```
    it("306: owner cannot transfer Ownership to himself", async function () {
        let message = "VotingRoleModel: cannot transfer Ownership to himself";
        await testError(async () => { return votingbuilder.transferOwnership(signers[0].address); },
message);
    });

    it("307: transferOwnership() requires onlyOwner", async function () {
        await testError(async () => { return
votingbuilder.connect(signers[1]).transferOwnership(signers[1].address); }, onlyOwner);
        let owner = await votingbuilder.getOwner();
        expect(owner).to.equal(signers[0].address);
    });

    it("308: transferOwnership() works properly", async function () {
        await votingbuilder.transferOwnership(signers[1].address);
        let owner = await votingbuilder.getOwner();
        expect(owner).to.equal(signers[1].address);
    });

    it("309: toggleIsVotersLimited() requires onlyAdmin", async function () {
        await testError(async () => { return votingbuilder.toggleIsVotersLimited(100); },
onlyAdmin);

        await votingbuilder.grantRoleAdmin(signers[0].address);
        let params = await votingbuilder.getVotingParams();
        expect(params.isVotersLimited).to.equal(false);
    });

    it("310: toggleIsVoterDisclosed() requires onlyAdmin", async function () {
        await testError(async () => { return votingbuilder.toggleIsVoterDisclosed(); }, onlyAdmin);

        await votingbuilder.grantRoleAdmin(signers[0].address);
        let params = await votingbuilder.getVotingParams();
        expect(params.isVoterDisclosed).to.equal(false);
    });

    it("311: toggleIsVotePowered() requires onlyAdmin", async function () {
        await testError(async () => { return votingbuilder.toggleIsVotePowered([1, 2, 3]); },
onlyAdmin);

        await votingbuilder.grantRoleAdmin(signers[0].address);
        let params = await votingbuilder.getVotingParams();
        expect(params.isVotePowered).to.equal(false);
    });

    it("312: toggleIsResultVetoed() requires onlyAdmin", async function () {
        await testError(async () => { return votingbuilder.toggleIsResultVetoed(); }, onlyAdmin);

        await votingbuilder.grantRoleAdmin(signers[0].address);
        let params = await votingbuilder.getVotingParams();
        expect(params.isResultVetoed).to.equal(false);
    });

    it("313: toggleIsResultPartial() requires onlyAdmin", async function () {
        await testError(async () => { return votingbuilder.toggleIsResultPartial(75); }, onlyAdmin);

        await votingbuilder.grantRoleAdmin(signers[0].address);
        let params = await votingbuilder.getVotingParams();
        expect(params.isResultPartial).to.equal(false);
    });

    it("314: toggleIsRepeated() requires onlyAdmin", async function () {
        await testError(async () => { return votingbuilder.toggleIsRepeated(); }, onlyAdmin);

        await votingbuilder.grantRoleAdmin(signers[0].address);
        let params = await votingbuilder.getVotingParams();
        expect(params.isRepeated).to.equal(false);
    });

    it("315: toggleIsPrivate() requires onlyAdmin", async function () {
        await testError(async () => { return votingbuilder.toggleIsPrivate([signers[0].address]); },
onlyAdmin);

        await votingbuilder.grantRoleAdmin(signers[0].address);
        let params = await votingbuilder.getVotingParams();
        expect(params.isPrivate).to.equal(false);
    });

    it("316: toggleIsMultiChoiced() requires onlyAdmin", async function () {
        await testError(async () => { return votingbuilder.toggleIsMultiChoiced([]); }, onlyAdmin);

        await votingbuilder.grantRoleAdmin(signers[0].address);
```

```
        let params = await votingbuilder.getVotingParams();
        expect(params.isMultiChoiced).to.equal(false);
    });

    it("317: toggleIsFunded() requires onlyAdmin", async function () {
        await testError(async () => { return votingbuilder.toggleIsFunded(10); }, onlyAdmin);

        await votingbuilder.grantRoleAdmin(signers[0].address);
        let params = await votingbuilder.getVotingParams();
        expect(params.isFunded).to.equal(false);
    });

    it("318: grantRoleAdmin() requires onlyOwner", async function () {
        await testError(async () => { return
votingbuilder.connect(signers[1]).grantRoleAdmin(signers[1].address); }, onlyOwner);

        let ifAdmin = await votingbuilder.checkAdmin(signers[1].address);
        expect(ifAdmin).to.equal(false);
    });

    it("319: grantRoleAdmin() requires address is not admin", async function () {
        await votingbuilder.grantRoleAdmin(signers[1].address);

        let message = "VotingRoleModel: Role already granted to this address"
        await testError(async () => { return votingbuilder.grantRoleAdmin(signers[1].address); },
message);
    });

    it("320: grantRoleOperator() requires onlyOwner", async function () {
        await testError(async () => { return
votingbuilder.connect(signers[1]).grantRoleOperator(signers[1].address); }, onlyOwner);

        let ifOperator = await votingbuilder.checkOperator(signers[1].address);
        expect(ifOperator).to.equal(false);
    });

    it("321: grantRoleOperator() requires address is not operator", async function () {
        await votingbuilder.grantRoleOperator(signers[1].address);

        let message = "VotingRoleModel: Role already granted to this address"
        await testError(async () => { return votingbuilder.grantRoleOperator(signers[1].address); },
message);
    });

    it("322: revokeRoleAdmin() requires onlyOwner", async function () {
        await votingbuilder.grantRoleAdmin(signers[0].address);
        await testError(async () => { return
votingbuilder.connect(signers[1]).revokeRoleAdmin(signers[0].address); }, onlyOwner);

        let ifAdmin = await votingbuilder.checkAdmin(signers[0].address);
        expect(ifAdmin).to.equal(true);
    });

    it("323: revokeRoleAdmin() requires address is admin", async function () {
        let message = "VotingRoleModel: Role is not granted to this address"
        await testError(async () => { return votingbuilder.revokeRoleAdmin(signers[1].address); },
message);
    });

    it("324: revokeRoleOperator() requires onlyOwner", async function () {
        await votingbuilder.grantRoleOperator(signers[0].address);
        await testError(async () => { return
votingbuilder.connect(signers[1]).revokeRoleOperator(signers[0].address); }, onlyOwner);

        let ifOperator = await votingbuilder.checkOperator(signers[0].address);
        expect(ifOperator).to.equal(true);
    });

    it("325: revokeRoleOperator() requires address is operator", async function () {
        let message = "VotingRoleModel: Role is not granted to this address"
        await testError(async () => { return votingbuilder.revokeRoleOperator(signers[1].address);
}, message);
    });

    it("326: createVoting() requires onlyAdmin", async function () {
        await testError(async () => { return votingbuilder.createVoting(); }, onlyAdmin);

        await votingbuilder.grantRoleAdmin(signers[0].address);
        let addr = await votingbuilder.getVoting();
        expect(addr).to.equal("0x0000000000000000000000000000000000000000");
    });
});
```

ДОДАТОК Г

ІЛЮСТРАТИВНА ЧАСТИНА

МЕТОД ТА ЗАСІБ ГЕНЕРУВАННЯ СМАРТ-КОНТРАКТІВ ДЛЯ ЗАХИЩЕНОГО ГОЛОСУВАННЯ

Актуальність, мета та задачі МКР

Актуальність теми обумовлена важливістю голосування як елементу суспільства при потребі зробити певний вибір.

Традиційні методи фізичного та електронного голосування залишаються вразливими перед загрозами (хакерські атаки, підробка голосів, порушення конфіденційності), мають низьку зручність процесів, та не пропонують універсальні рішення.

Використання блокчейну та смарт-контрактів, поруч з гнучкою системою генерації голосувань, пропонує безпечний та ефективний підхід, відповідаючи сучасним потребам.

Метою дипломної роботи є покращення спостережності електронного голосування шляхом використання смарт-контрактів.

Задачі роботи:

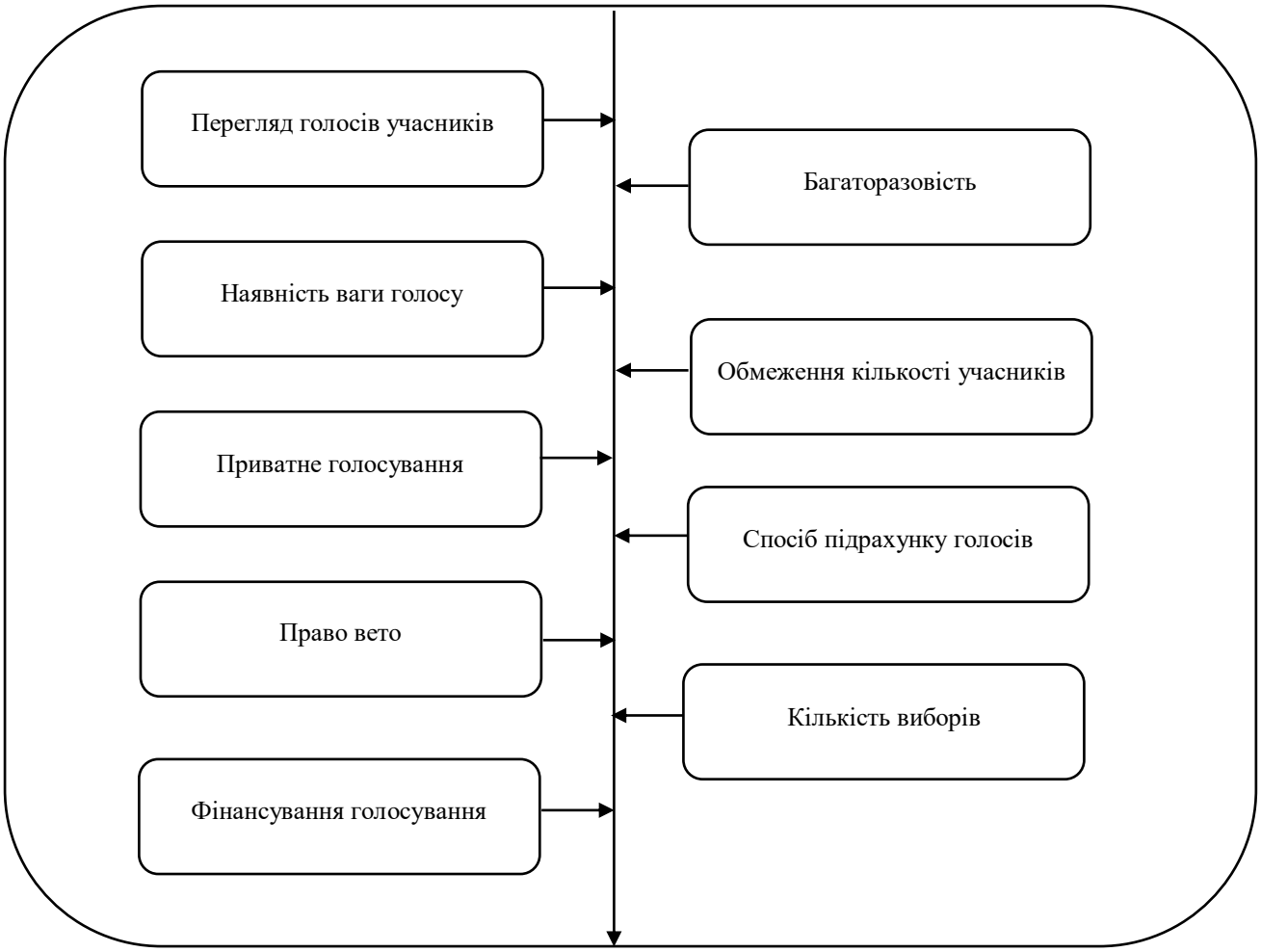
– провести аналіз існуючих методів голосування, визначити їх переваги та недоліки
– оцінити економічний ефект розробки
– виконати математичний опис системи захищеного голосування
– розробити власний метод генерації смарт-контрактів для голосування, спрямований на підвищення ефективності та гнучкості
– реалізувати засіб генерації голосування на основі розробленого методу.
– здійснити тестування та отримати результати ефективності розробленого методу та засобу

Наукова новизна та практична цінність

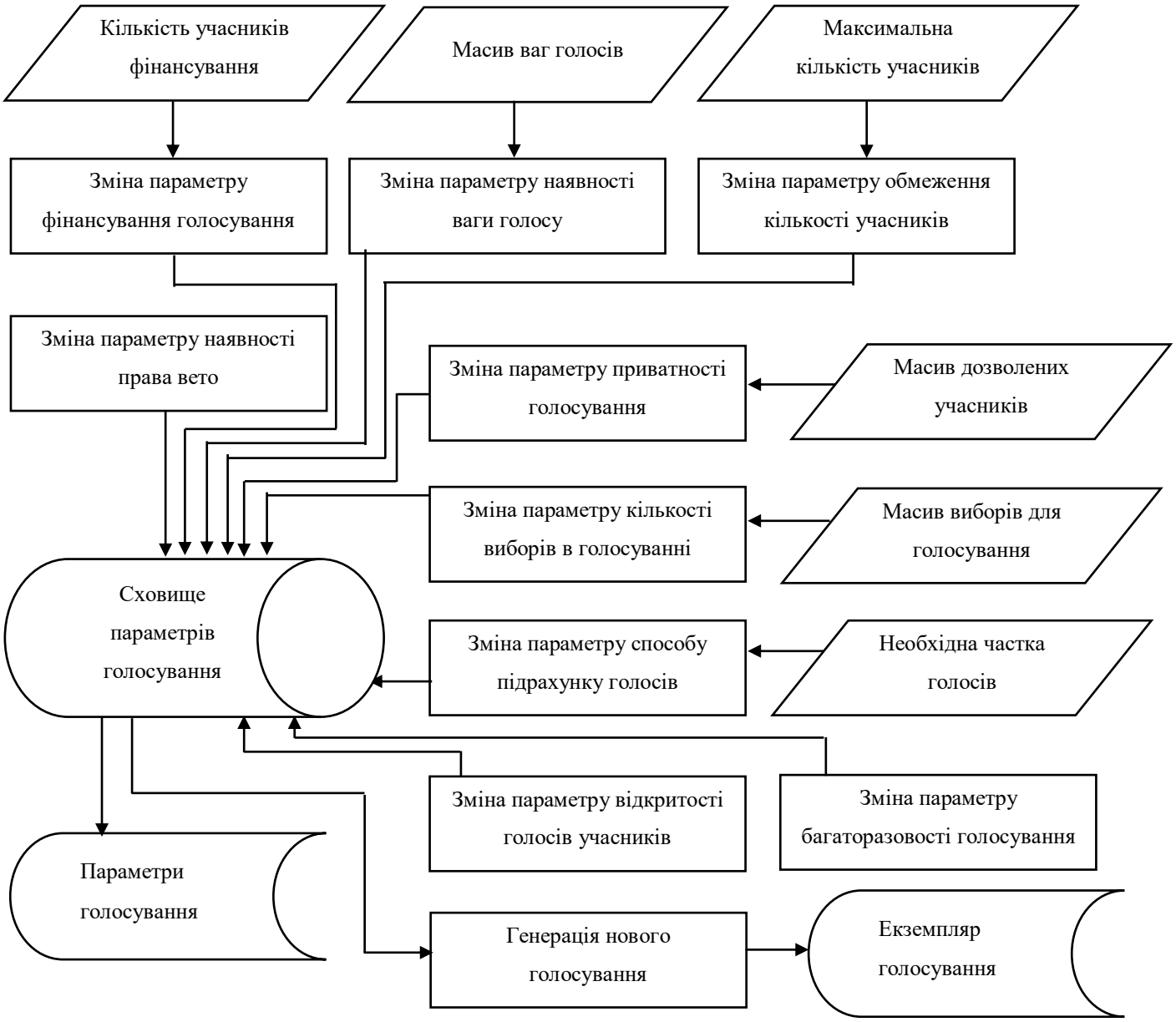
Наукова новизна: отримано подальший розвиток методів генерування голосування на основі смарт-контрактів, що відрізняється від відомих високою адаптивністю до умов використання, що дозволило збільшити кількість потенційних застосувань до 160 варіацій.

Практична цінність розробки полягає в смарт-контрактах для генерування захищених голосувань, інтерфейсі користувача для взаємодії зі смарт-контрактами, та контрольному списку блокового тестування системи

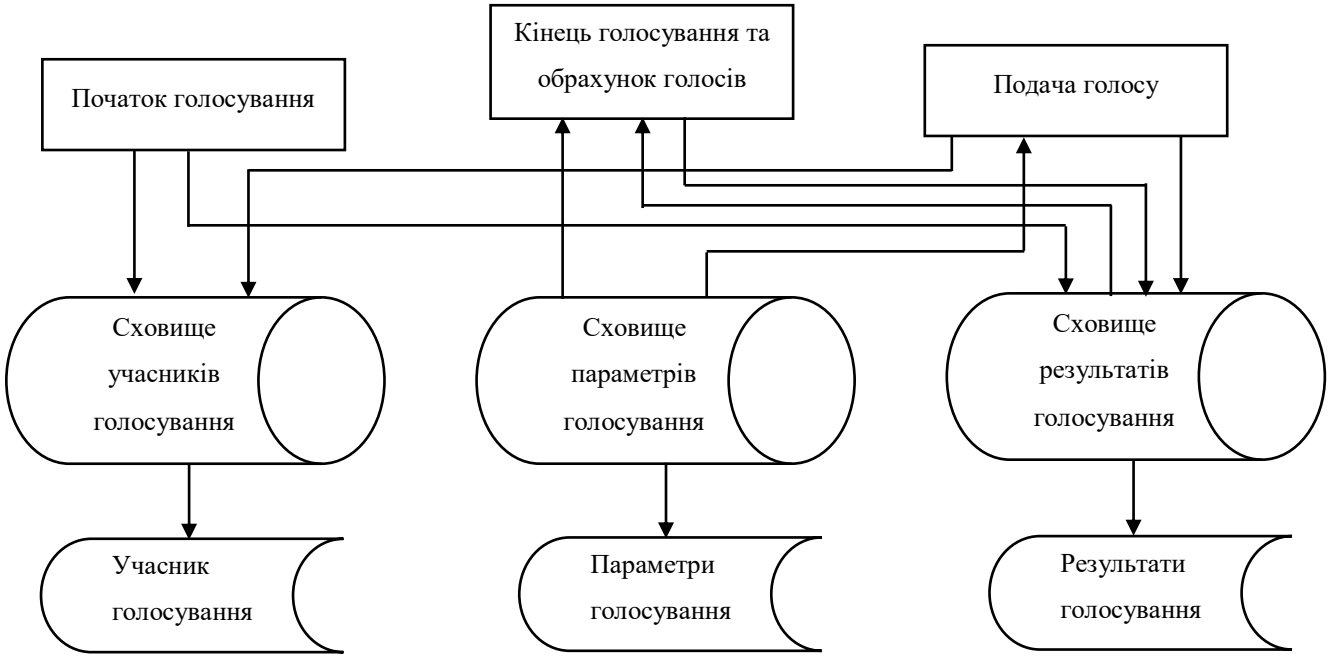
Вибір параметрів генерування голосування



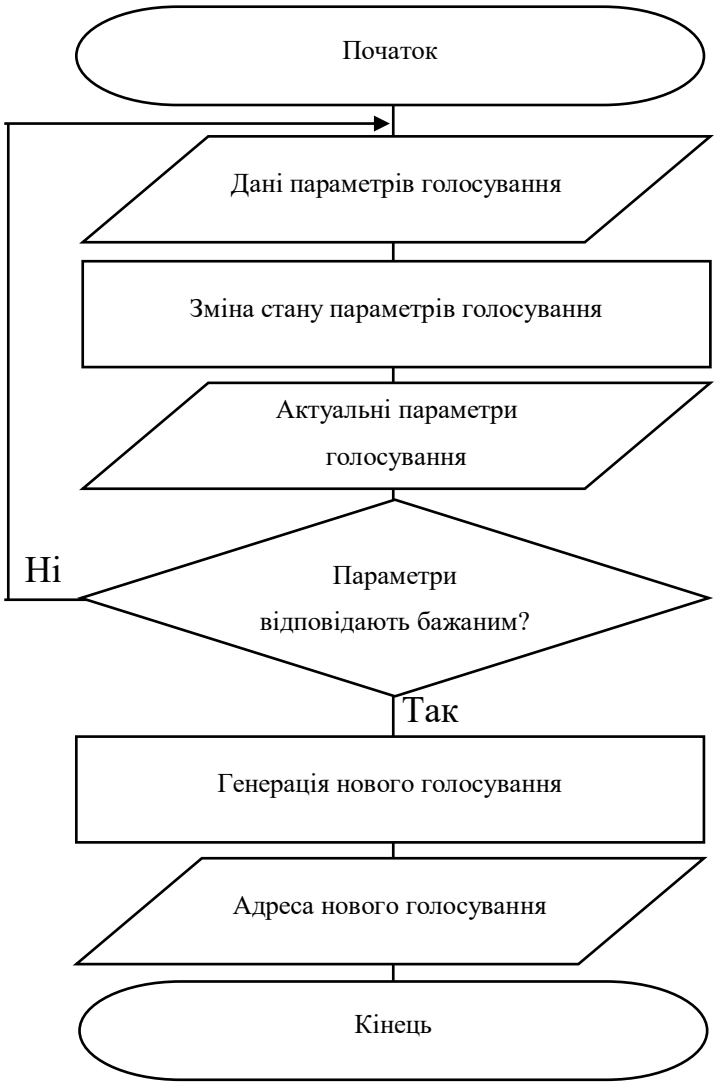
Модуль генерування голосування



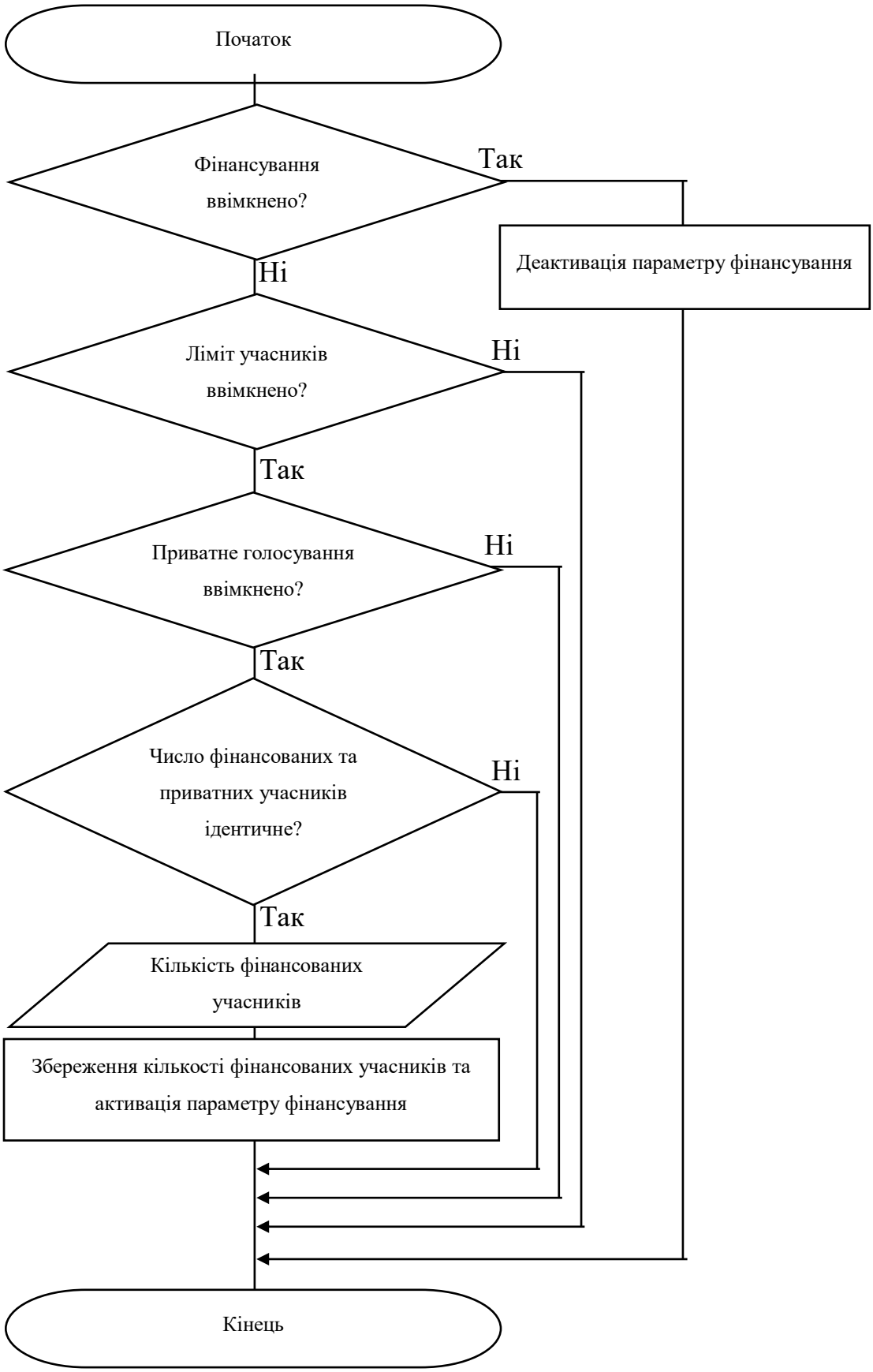
Модуль проведення голосування



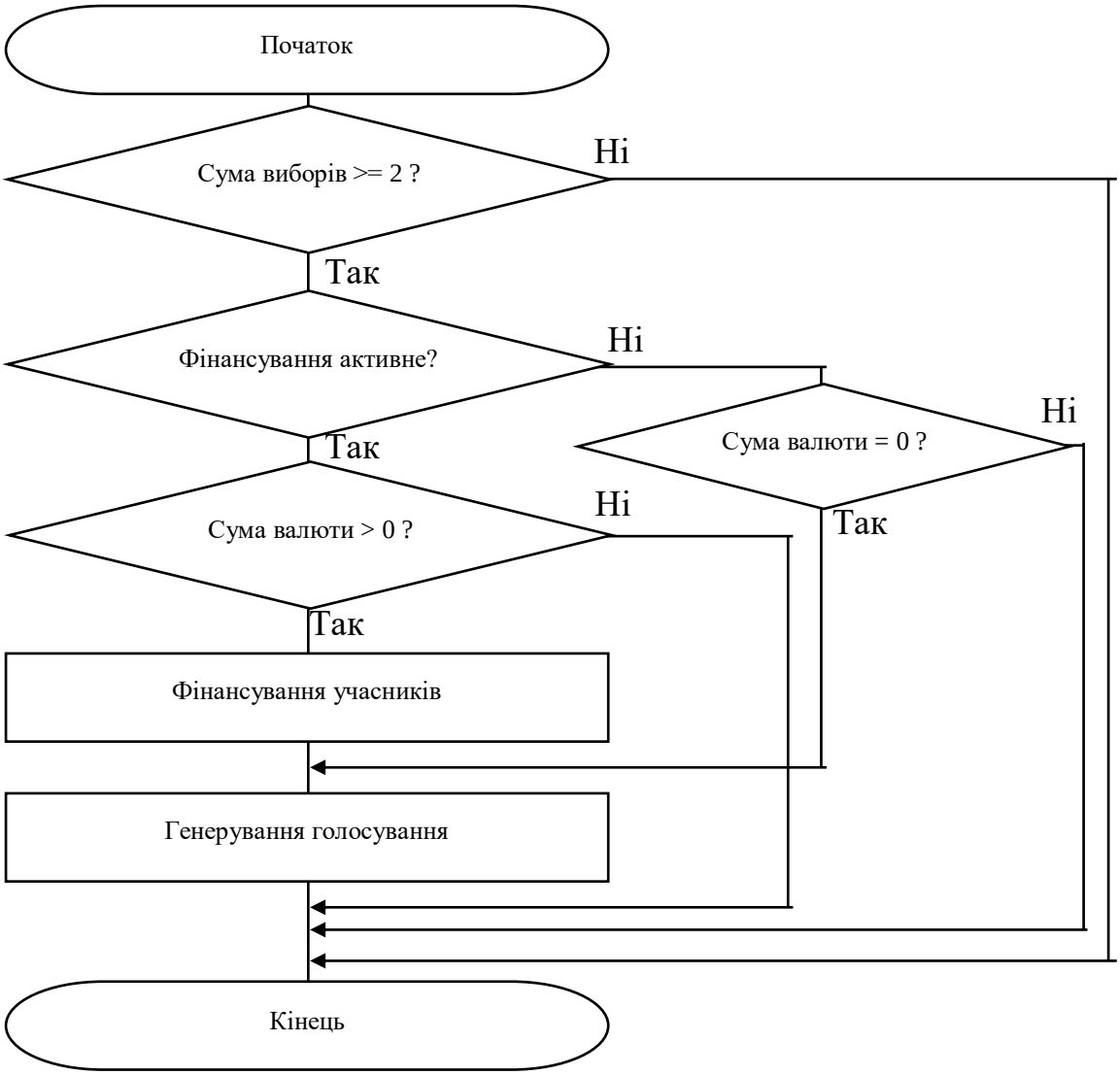
Алгоритм взаємодії з модулем генерування голосування



Процедура налаштування параметру фінансування



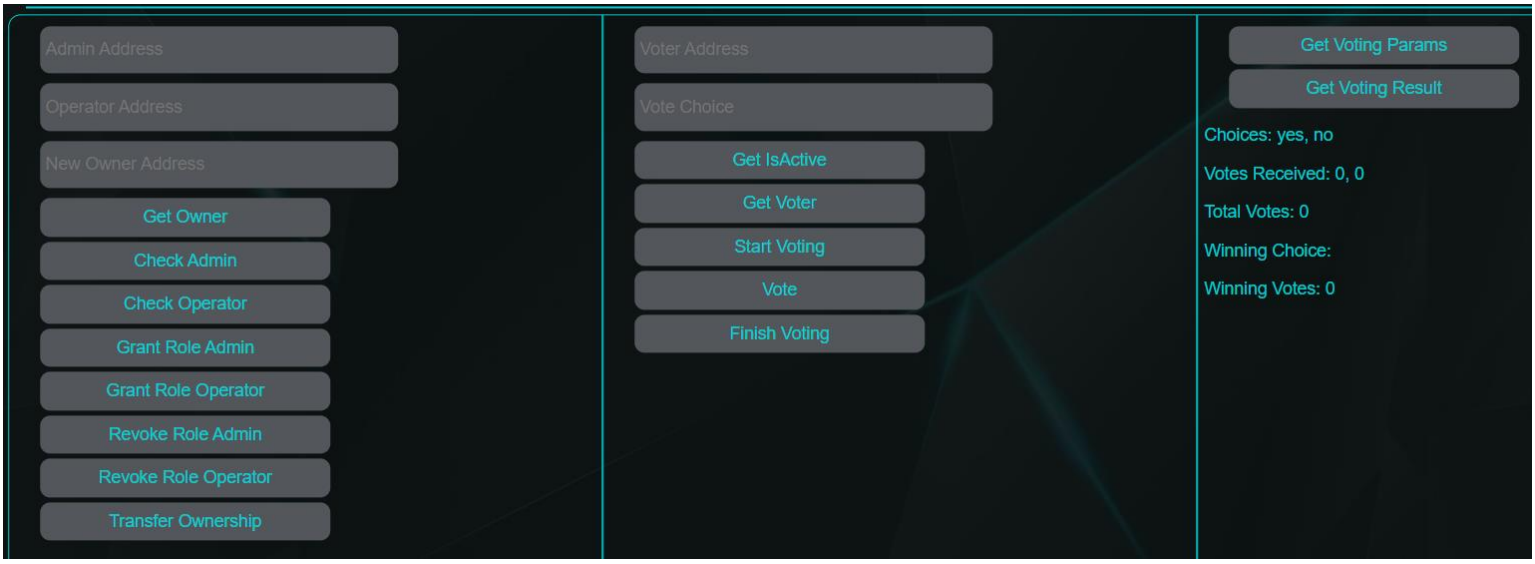
Процедура генерування голосування



Результати тестування модуля генерування голосування

```
1. Voting Builder
  √ 101: check VotingParameters structure (44ms)
  √ 102: check VotingParameters default values
  √ 103: getVoting() returns 0-address before deploy
  √ 104: toggleIsVotersLimited() works properly (on/off) (53ms)
  √ 105: toggleIsVotersLimited() false disables isFunded, isVotePowered, isPrivate (74ms)
  √ 106: toggleIsVoterDisclosed() works properly (on/off) (58ms)
  √ 107: toggleIsVotePowered() works properly (on/off) (75ms)
  √ 108: toggleIsVotePowered() false clears voterPowers[] (62ms)
  √ 109: toggleIsVotePowered() true requires isVotersLimited true
  √ 110: toggleIsVotePowered() true requires isPrivate true
  √ 111: toggleIsVotePowered() true requires equal length powers-votersPrivate
  √ 112: toggleIsVotePowered() true requires power >=1
  √ 113: toggleIsResultVetoed() works properly (on/off)
  √ 114: toggleIsResultPartial() true if % = 51-100 (52ms)
  √ 115: toggleIsResultPartial() false if % = 0-50 (49ms)
  √ 116: toggleIsResultPartial() requires % = 0-100
  √ 117: toggleIsRepeated() works properly (on/off) (39ms)
  √ 118: toggleIsRepeated() true disables isFunded
  √ 119: toggleIsRepeated() true requires isVotersLimited true
  √ 120: toggleIsRepeated() true requires votersLimit <= 5000
  √ 121: toggleIsPrivate() true requires voters <= 5000 (321ms)
  √ 122: toggleIsPrivate() true requires voters <= votersLimit if isVotersLimited true
  √ 123: toggleIsPrivate() works properly (on/off) (49ms)
  √ 124: toggleIsPrivate() false disables isFunded, isVotePowered
  √ 125: toggleIsPrivate() false clears votersPrivate[]
  √ 126: toggleIsMultiChoiced() requires choices <= 20
  √ 127: toggleIsMultiChoiced() false and sets yes/no if choices <= 1
  √ 128: toggleIsMultiChoiced() true and sets input choices if choices 2-20 (42ms)
  √ 129: toggleIsFunded() true requires isVotersLimited true
  √ 130: toggleIsFunded() true requires isPrivate true
  √ 131: toggleIsFunded() true requires usersAmount >= votersLimit (133ms)
  √ 132: toggleIsFunded() true requires usersAmount = votersPrivate (51ms)
  √ 133: toggleIsFunded() works properly (on/off) (69ms)
  √ 134: createVoting() requires choices > 1
  √ 135: createVoting() requires msg.value > 0 if isFunded true (47ms)
  √ 136: createVoting() requires votersPrivate > 0 if isFunded true (41ms)
  √ 137: createVoting() funds all votersPrivate if isFunded true (91ms)
  √ 138: createVoting() requires msg.value = 0 if isFunded false
  √ 139: createVoting() deploys voting properly (47ms)
```

Робоча область інтерфейсу користувача



Результат статичного тестування

```
INFO:Detectors:
VotingBuilder.toggleIsResultPartial(uint256) (Voting_system.sol#332-342) contains a tautolog
y or contradiction:
        - require(bool,string)(_percentage >= 0 && _percentage <= 100,VotingBuilder: voting
winning Percentage must be between 0 and 100) (Voting_system.sol#333)
VotingBuilder.toggleIsResultPartial(uint256) (Voting_system.sol#332-342) contains a tautolog
y or contradiction:
        - _percentage >= 0 && _percentage <= 50 (Voting_system.sol#335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contra
diction
INFO:Detectors:
VotingRoleModel.transferOwnership(address) (Voting_system.sol#50-54) should emit an event fo
r:
        - owner = newOwner (Voting_system.sol#53)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-acce
ss-control
INFO:Detectors:
VotingBuilder.createVoting() (Voting_system.sol#364-388) has external calls inside a loop: (
success,None) = recipient.call{value: amountPerAddress}() (Voting_system.sol#379)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loo
p
INFO:Detectors:
Reentrancy in VotingBuilder.createVoting() (Voting_system.sol#364-388):
        External calls:
        - (success,None) = recipient.call{value: amountPerAddress}() (Voting_system.sol#379)

        State variables written after the call(s):
        - voting = new Voting(params) (Voting_system.sol#387)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerab
ilities-2
INFO:Detectors:
Voting.finishVoting() (Voting_system.sol#176-216) compares to a boolean constant:
        -maxVotes == 0 || doubleWinner == true (Voting_system.sol#193)
VotingBuilder.toggleIsFunded(uint256) (Voting_system.sol#253-266) compares to a boolean cons
tant:
        -require(bool,string)(votingParams.isVotersLimited == true,VotingBuilder: voting sho
uld has voters limit to be able to fund voters) (Voting_system.sol#255)
VotingBuilder.toggleIsFunded(uint256) (Voting_system.sol#253-266) compares to a boolean cons
tant:
        -require(bool,string)(votingParams.isPrivate == true,VotingBuilder: voting should ha
s private voters to be able to fund voters) (Voting_system.sol#256)
VotingBuilder.toggleIsRepeated() (Voting_system.sol#268-278) compares to a boolean constant:

        -require(bool,string)(votingParams.isVotersLimited == true,VotingBuilder: voting sho
uld has voters Limit to be Repeated) (Voting_system.sol#270)
VotingBuilder.toggleIsVotePowered(uint256[]) (Voting_system.sol#284-299) compares to a boole
an constant:
        -require(bool,string)(votingParams.isPrivate == true,VotingBuilder: voting should ha
s private voters to grant power to voters) (Voting_system.sol#287)
```

```
VotingBuilder.toggleIsVotePowered(uint256[]) (Voting_system.sol#284-299) compares to a boole
an constant:
        -require(bool,string)(votingParams.isVotersLimited == true,VotingBuilder: voting sho
uld has voters limit to grant power to voters) (Voting_system.sol#286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Pragma version^0.8.17 (Voting_system.sol#2) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-
of-solidity
INFO:Detectors:
Low level call in VotingBuilder.createVoting() (Voting_system.sol#364-388):
        - (success,None) = recipient.call{value: amountPerAddress}() (Voting_system.sol#379)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter VotingRoleModel.checkAdmin(address)._address (Voting_system.sol#18) is not in mixe
dCase
Parameter VotingRoleModel.checkOperator(address)._address (Voting_system.sol#22) is not in m
ixedCase
Parameter VotingRoleModel.grantRoleAdmin(address)._address (Voting_system.sol#26) is not in
mixedCase
Parameter VotingRoleModel.grantRoleOperator(address)._address (Voting_system.sol#32) is not
in mixedCase
Parameter VotingRoleModel.revokeRoleAdmin(address)._address (Voting_system.sol#38) is not in
 mixedCase
Parameter VotingRoleModel.revokeRoleOperator(address)._address (Voting_system.sol#44) is not
 in mixedCase
Parameter VotingBuilder.toggleIsVotersLimited(uint256)._votersLimit (Voting_system.sol#301)
is not in mixedCase
Parameter VotingBuilder.toggleIsResultPartial(uint256)._percentage (Voting_system.sol#332) i
s not in mixedCase
Parameter VotingBuilder.toggleIsMultiChoiced(string[])._choices (Voting_system.sol#348) is n
ot in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-soli
dity-naming-conventions
INFO:Slither:Voting_system.sol analyzed (3 contracts with 85 detectors), 23 result(s) found
```

Економічна частина

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Метод та засіб генерування смарт-контрактів для захищеного голосування» становить 39 балів, що, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий.

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 3,2 рази.

Також термін окупності становить 1,8 роки, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.