

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації

(повне найменування інституту, назва факультету)

Кафедра прикладної математики та інформатики

(повна назва кафедри)

«До захисту допущено»

Зав. кафедри ПМІ

Наталія МАСЛОВА

(підпис)

(ім'я та прізвище)

«\_\_\_\_\_» \_\_\_\_\_ 2024 р.

## Випускна кваліфікаційна робота

бакалавра

(освітній ступінь)

на тему: Дослідження використання блокчейн-систем для покращення безпеки та надійності операцій з криптовалютою  
зі спецчастиною: Застосування математичної моделі для покращення безпеки та надійності блокчейн-систем

Виконала: студентка 4 курсу, групи ІПЗ-20

(шифр групи)

спеціальності 121 Інженерія програмного забезпечення

(код і назва спеціальності)

освітньої програми 121 Інженерія програмного забезпечення

Попова Анна Олександрівна

(прізвище та ініціали)

  
(підпис)

Керівник доц. каф. ПМІ, доц., к.ф.-м.н. Олена ЛЮБИМЕНКО

(посада, науковий ступінь, вчене звання, ім'я та прізвище)

(підпис)

Консультант проф. каф. ПМІ, проф., д.т.н. Людмила КОВАЛЬЧУК

(посада, науковий ступінь, вчене звання, ім'я та прізвище)

(підпис)

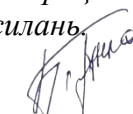
Рецензент проф. каф. ЕТКІ, проф., д.т.н. Володимир СВЯТНИЙ

(посада, науковий ступінь, вчене звання, ім'я та прізвище)

(підпис)

*Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.*

Студент

  
(підпис)

Луцьк – 2024

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації  
Кафедра прикладної математики та інформатики  
Освітній ступінь бакалавр  
Спеціальність 121 Інженерія програмного забезпечення  
Освітня програма 121 Інженерія програмного забезпечення  
(код і назва)

ЗАТВЕРДЖУЮ  
Зав. кафедри ПМІ  
/Н. Маслова/  
«\_\_» \_\_\_\_\_ 2024 року

Поповій Анні Олександрівні

(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження використання блокчейн-систем для покращення безпеки та надійності операцій з криптовалютою»

Спецчастина «Застосування математичної моделі для покращення безпеки та надійності блокчейн-систем»

Керівник роботи доц. каф. ПМІ, к.ф.-м.н. Олена ЛЮБИМЕНКО  
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом від « 17 » травня 2024 року № 198

2. Строк подання студентом роботи 12 червня 2024 року

3. Вихідні дані до роботи результати переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- 1) Огляд блокчейн технологій та областей його застосування. Методи голосування
- 2) Опис методів та засобів розв'язання задачі
- 3) Опис алгоритму та реалізації
- 4) Опис роботи системи голосування

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)  
робота містить 2 таблиці, 46 рисунків, 1 діаграму та інший графічний матеріал, у кількості, достатній для демонстрації сутності роботи

## 6. Консультанти розділів роботи

| Розділ         | Прізвище, ініціали та посада консультанта | Підпис, дата   |                  |
|----------------|-------------------------------------------|----------------|------------------|
|                |                                           | завдання видав | завдання прийняв |
| Нормоконтролер | доц. Назарова І.А.                        |                |                  |
|                |                                           |                |                  |

7. Дата видачі завдання 29 квітня 2024 року

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломної роботи                                                                                   | Строк виконання етапів роботи | Примітка |
|-------|-----------------------------------------------------------------------------------------------------------------|-------------------------------|----------|
| 1     | Огляд літератури з предметної області                                                                           | 29.04.2024 – 02.05.2024       |          |
| 2     | Збір інформації і методичних відомостей щодо блокчейну та методів голосування                                   | 03.05.2024 – 13.05.2024       |          |
| 3     | Розробка структури програмного забезпечення, вибір даних для обробки та експериментів.                          | 14.05.2024 – 02.06.2024       |          |
| 4     | Тестування системи                                                                                              | 03.06.2024 – 06.06.2024       |          |
| 5     | Оформлення пояснювальної записки                                                                                | 07.06.2024 – 12.06.2024       |          |
| 6     | Нормоконтроль пояснювальної записки, її перевірка антиплагіатною системою                                       | 12.06.2024                    |          |
| 7     | Оформлення супутніх матеріалів (розробка графічного матеріалу, написання доповіді, тощо) та рецензування роботи | 13.06.2024- 25.06.2024        |          |
| 8     | Захист роботи                                                                                                   | 26.06.2024                    |          |

Студент

  
(підпис)

Анна ПОПОВА

(ім'я та ПРІЗВИЩЕ)

Керівник роботи

\_\_\_\_\_  
(підпис)

Олена ЛЮБИМЕНКО

(ім'я та ПРІЗВИЩЕ)

## АНОТАЦІЯ

Попова Анна Олександрівна. Дослідження використання блокчейн-систем для покращення безпеки та надійності операцій з криптовалютою / Випускна кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення». – ДВНЗ ДонНТУ, Луцьк, 2024.

Об'єктом дослідження є застосування блокчейн-систем для забезпечення безпеки та надійності операцій з криптовалютою.

Предметом дослідження є застосування математичної моделі квадратичного голосування для підвищення безпеки та надійності систем голосування на блокчейні.

Мета дослідження полягає у вивченні ефективності застосування моделі квадратичного голосування для забезпечення безпеки та надійності голосування на блокчейн-платформі та розробці програмного забезпечення для її використання.

Для досягнення цієї мети використовуються методи аналізу математичних моделей, а також розробка та тестування програмного забезпечення.

Наукова новизна полягає в апробації моделі квадратичного голосування для застосування у системах голосування на блокчейні та виявленні її переваг та обмежень.

Практичне значення дослідження полягає в можливості використання розробленого програмного забезпечення для забезпечення безпеки та надійності голосування на блокчейн-платформі, що сприятиме підвищенню довіри до результатів голосувань та запобіганню можливих атак.

Ключові слова: BLOCKCHAIN, PYTHON, SOLIDITY, ГОЛОСУВАННЯ, СИМУЛЯТОР, HTML, CSS, ВЕБ-ДОДАТОК

Список публікацій здобувача:

1) Попова А.О. Застосування математичної моделі для покращення безпеки та надійності блокчейн-систем / А.О. Попова, О.М. Любименко, Н.О. Маслова // Збірник тез доповідей II Міжнародної науково-практичної конференції студентів та молодих вчених «Математика та математичне моделювання у сучасному технічному університеті», Луцьк, ДонНТУ, 2024. – С. 36–37.

## ANNOTATION

Anna Oleksandrivna Popova. Investigating the use of blockchain systems to improve the security and reliability of cryptocurrency transactions/ Bachelor's Qualification Thesis for the Educational Degree "Bachelor" in Specialty 121 "Software Engineering". – DonNTU, Lutsk, 2024.

The object of the research is the application of blockchain systems to ensure the security and reliability of cryptocurrency transactions.

The subject of the research is the application of the quadratic voting mathematical model to enhance the security and reliability of voting systems on the blockchain.

The purpose of the research is to study the effectiveness of applying the quadratic voting model to ensure the security and reliability of voting on a blockchain platform and to develop software for its implementation.

To achieve this goal, methods of mathematical model analysis, as well as software development and testing, are used.

The scientific novelty lies in the testing of the quadratic voting model for application in blockchain voting systems and identifying its advantages and limitations.

The practical significance of the research lies in the possibility of using the developed software to ensure the security and reliability of voting on a blockchain platform, which will contribute to increasing trust in voting results and preventing possible attacks.

Keywords: BLOCKCHAIN, PYTHON, SOLIDITY, VOTING, SIMULATOR, HTML, CSS, WEB APPLICATION

List of publications by the applicant:

1) Popova A.O. Application of a Mathematical Model to Improve the Security and Reliability of Blockchain Systems / A.O. Popova, O.M. Liubymenko, N.O. Maslova // Collection of Abstracts of the II International Scientific and Practical Conference of Students and Young Scientists "Mathematics and Mathematical Modeling in a Modern Technical University", Lutsk, DonNTU, 2024. – pp. 36–37.

## ЗМІСТ

|                                                                                                                        |    |
|------------------------------------------------------------------------------------------------------------------------|----|
| ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І<br>ТЕРМІНІВ .....                                            | 8  |
| ВСТУП.....                                                                                                             | 9  |
| 1 ОГЛЯД БЛОКЧЕЙН ТЕХНОЛОГІЙ ТА ОБЛАСТЕЙ ЙОГО<br>ЗАСТОСУВАННЯ. МЕТОДИ ГОЛОСУВАННЯ .....                                 | 11 |
| 1.1 Огляд блокчейн-технологій.....                                                                                     | 11 |
| 1.2 Огляд існуючих методів голосування .....                                                                           | 14 |
| 1.3 Обґрунтування вибору математичної моделі голосування для покращення<br>безпеки та надійності блокчейн-системи..... | 14 |
| 1.4 Висновки за розділом.....                                                                                          | 19 |
| 2 ОПИС МЕТОДІВ ТА ЗАСОБІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ.....                                                                      | 21 |
| 2.1 Математичне подання принципу квадратичного голосування .....                                                       | 21 |
| 2.2 Визначення вимог та функціоналу симулятора .....                                                                   | 21 |
| 2.3 Вибір технологій та інструментів для реалізації .....                                                              | 23 |
| 2.3.1 Мова програмування Python .....                                                                                  | 23 |
| 2.3.2 Мова програмування для написання смартконтрактів Solidity .....                                                  | 25 |
| 2.3.3 Фреймворки Flask та Truffle.....                                                                                 | 26 |
| 2.3.4 Засоби для створення інтерфейсу додатку: HTML, CSS, JavaScript... ..                                             | 28 |
| 2.3.5 Інструменти розробки.....                                                                                        | 28 |
| 2.4 Висновки за розділом.....                                                                                          | 30 |
| 3 ОПИС АЛГОРИТМУ ТА РЕАЛІЗАЦІЇ .....                                                                                   | 31 |
| 3.1 Проектування системи голосування.....                                                                              | 31 |
| 3.2 Розробка та розгортання смартконтракту.....                                                                        | 33 |
| 3.3 Створення основного програмного модуля.....                                                                        | 40 |
| 3.4 Розробка веб-додатку.....                                                                                          | 42 |

|                                                                      |    |
|----------------------------------------------------------------------|----|
| 3.5 Висновки за розділом.....                                        | 44 |
| 4 ОПИС РОБОТИ СИСТЕМИ ГОЛОСУВАННЯ.....                               | 46 |
| 4.1 Інтерфейс веб-додатку .....                                      | 46 |
| 4.2 Тестування роботи додатку .....                                  | 50 |
| 4.3 Аналіз результатів та визначення шляхів покращення додатку ..... | 59 |
| 4.4 Висновки за розділом.....                                        | 59 |
| ВИСНОВКИ.....                                                        | 61 |
| ПЕРЕЛІК ПОСИЛАНЬ .....                                               | 63 |
| Додаток А Зауваження нормоконтролера.....                            | 65 |
| Додаток Б Фрагменти лістингу програми.....                           | 66 |

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML – Hyper Text Markup Language

CSS – Cascading Style Sheets

QV – Quadratic Voting

ETH – Ethereum

wei – найменша одиниця вимірювання криптовалюти Ethereum

## ВСТУП

У сучасному світі блокчейн-технології швидко розвиваються і знаходять широке застосування у різних галузях, забезпечуючи високий рівень безпеки, прозорості та децентралізації. Однак, не зважаючи на їхні переваги, існує необхідність подальшого підвищення безпеки та надійності операцій, зокрема в контексті критично важливих процесів, таких як голосування.

Одним із підходів до вирішення цієї проблеми є використання математичних моделей, зокрема моделі квадратичного голосування. Ця модель дозволяє ефективно визначати рішення на основі агрегованих голосів учасників системи з урахуванням їхньої ваги та довіри.

У ході виконання даної роботи планується дослідити можливість застосування моделі квадратичного голосування для покращення безпеки та надійності голосувань у блокчейн-системах. Для досягнення цієї мети передбачено розробку програмного забезпечення системи голосування, яка буде базуватися на даній моделі.

Зростаюча популярність блокчейн-технологій породжує потребу у подальшому вдосконаленні їхніх можливостей, зокрема в області безпеки та надійності операцій. Голосування, як один із ключових процесів у суспільстві, вимагає особливої уваги до забезпечення його безпеки та правильності. Тому дослідження можливостей використання математичних моделей для оптимізації голосувань у блокчейн-системах є актуальним та важливим завданням.

Метою даного дослідження є вивчення можливостей застосування моделі квадратичного голосування для покращення безпеки та надійності голосувань у блокчейн-системах. Для досягнення цієї мети передбачено розробку програмного забезпечення системи голосування, що базується на даній моделі.

З метою вирішення поставленої проблеми були сформульовані наступні

завдання:

- проаналізувати існуючі підходи до забезпечення безпеки голосувань у блокчейн-системах;
- дослідити математичну модель квадратичного голосування;
- розробити програмне забезпечення для системи голосування, що базується на моделі квадратичного голосування;
- провести тестування та оцінку ефективності розробленої системи на практиці.

У процесі виконання роботи будуть використані різноманітні наукові методи дослідження. Зокрема, планується аналіз літературних джерел для оцінки існуючих підходів та моделей. Також буде проведено математичне моделювання для розробки квадратичної моделі голосування. Крім того, планується проведення емпіричних досліджень на основі розробленого програмного забезпечення.

Очікується, що результатом роботи буде розробка та впровадження програмного забезпечення системи голосування, що використовує модель квадратичного голосування. Це дозволить продемонструвати переваги використання зазначеної моделі у забезпеченні безпеки та надійності голосувань у блокчейн-системах. Крім того, розроблена система голосування може бути використана як платформа для подальшого дослідження та розвитку методів забезпечення безпеки голосувань у блокчейн-системах, а також для навчальних цілей.

# 1 ОГЛЯД БЛОКЧЕЙН ТЕХНОЛОГІЙ ТА ОБЛАСТЕЙ ЙОГО ЗАСТОСУВАННЯ. МЕТОДИ ГОЛОСУВАННЯ

## 1.1 Огляд блокчейн-технологій

Блокчейн – це розподілена база даних, що зберігає постійно зростаючий список записів, називаних блоками, захищених від несанкціонованих змін і злому за допомогою криптографічних методів. Ця інноваційна технологія швидко розвивається та має потенціал революціонізувати багато галузей. Блокчейн забезпечує децентралізоване, прозоре та безпечне середовище для здійснення транзакцій і зберігання даних без необхідності довіреної третьої сторони. Технологія блокчейн, вперше впроваджена з появою Bitcoin у 2008 році, стала каталізатором для нового підходу до збереження та обміну даними в онлайн-середовищі. Ця технологія, що базується на розподіленій системі зберігання даних, забезпечує безпеку, децентралізацію та прозорість [1].

Технологія блокчейн заснована на кількох ключових принципах децентралізації, незмінності та консенсусу. Відсутність центрального управління дозволяє розподілено зберігати дані на багатьох вузлах мережі. Це робить систему більш стійкою до атак та відмов. Усі транзакції є відкритими та доступними для перегляду всіма учасниками мережі, що сприяє підвищенню довіри та відкритості. Після додавання блоку до ланцюжка, внесені дані стають практично неможливими до змінення без консенсусу більшості учасників мережі. Дані в блокчейні захищені за допомогою криптографічних методів, таких як хеш-функції та цифрові підписи, що забезпечують конфіденційність і цілісність інформації.

Блокчейн має свої переваги та недоліки. Дані блокчейну зберігаються на тисячах пристроїв у розподіленій мережі вузлів, що робить систему дуже стійкою до технічних збоїв та кібератак. Кожен вузол мережі має копію бази даних, тому відключення одного вузла не загрожує безпеці всієї мережі.

Блоки в блокчейні майже ніколи не скасовуються, що ускладнює зміну

або видалення зареєстрованих даних. Блокчейн є системою, «не потребуючою довіри», оскільки він ґрунтується на розподіленій мережі вузлів, які виконують перевірку транзакцій без посередників. Це знижує ризик, пов'язаний з довірою до певної організації, і скорочує комісії за транзакції, виключаючи посередників та треті сторони [2].

Однак децентралізований характер блокчейну має й свої недоліки. Наприклад, порівняно з традиційними централізованими базами даних, блокчейни обмежені в ефективності та вимагають більший обсяг зберігання. Кожен повний вузол мережі зберігає повний обсяг даних, що може призводити до зростання обсягу потрібного дискового простору з ростом мережі. Також можна виділити такі недоліки, як, наприклад, так звана «атака 51%». Ця атака відбувається, коли атакуючій стороні вдається контролювати більше 50% всієї потужності хешування мережі блокчейн. Це може дозволити виключати або змінювати порядок транзакцій. Проте з розвитком мережі збільшується її безпека, і ймовірності успішної атаки 51% стають дуже низькими [2].

Після додавання даних їх дуже складно змінити в блокчейні. Хоча це забезпечує стабільність, зміна даних чи коду вимагає значних зусиль і часто призводить до створення нового ланцюжка блоків.

Також слід зазначити, що кожен користувач блокчейну має приватний ключ, який надає право власності на криптовалюту або інші дані блокчейну. Втрата приватного ключа може призвести до безповоротної втрати доступу до коштів, що також є великим мінусом.

Незважаючи на певні обмеження, технологія блокчейн пропонує унікальні переваги і продовжує активно розвиватися. Хоча масове впровадження ще далеке, багато галузей вже досліджують її можливості та переваги. Напевно, у найближчі роки компанії та уряди будуть експериментувати з новими способами застосування блокчейну, спрямовуючись на вигоду від цієї інноваційної технології [2].

Блокчейн складається з взаємопов'язаних блоків, де кожен новий блок містить хеш попереднього, формуючи незмінний ланцюжок. Транзакції

об'єднуються в блоки, які повинні бути перевірені і підтверджені більшістю вузлів у мережі згідно заданим правилам консенсусу. Приклад структури блокчейну приведено на рис. 1.1.

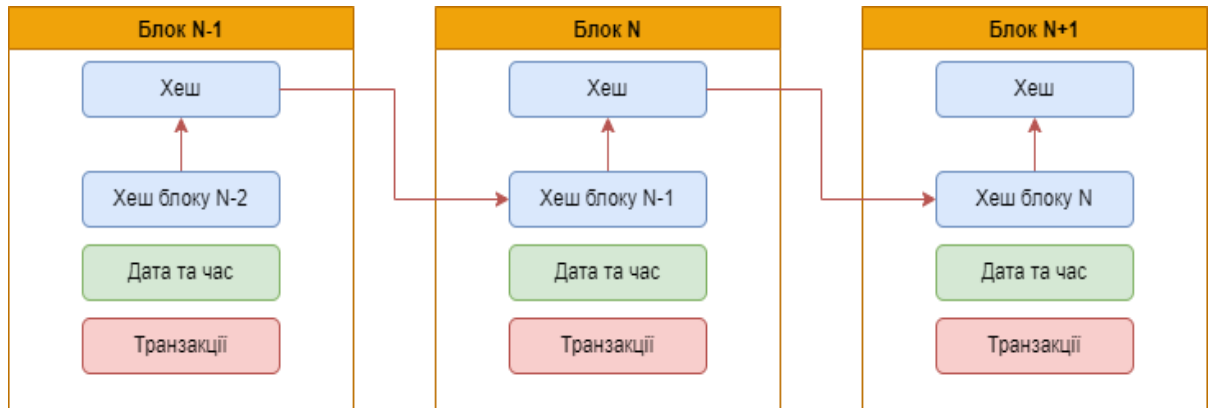


Рисунок 1.1 – Структура блокчейну

Існує кілька типів блокчейнів, кожен з яких має свої особливості та сфери застосування. Серед них можна виокремити публічні, приватні та консорціумні блокчейни.

Публічні блокчейни доступні для читання та запису для всіх учасників мережі без обмежень. Прикладами є Bitcoin та Ethereum, де кожен може брати участь у мережі.

У системах приватних блокчейнів доступ до мережі контролюється, і лише авторизовані учасники мають можливість переглядати чи додавати дані.

Консорціумні блокчейни є частково децентралізованими системами, де доступ до мережі контролюється кількома авторизованими учасниками. Вони часто застосовуються в корпоративному середовищі для побудови спільних додатків та інфраструктури.

Блокчейн знаходить широке застосування у різних галузях, таких як фінанси, охорона здоров'я, управління державою чи ланцюги постачання товарів. Криптовалюти, такі як Bitcoin та Ethereum, вже використовують блокчейн для безпечних та децентралізованих платежів. Блокчейн також може використовуватися для спрощення торгівлі цінними паперами, міжнародних

переказів та інших фінансових операцій.

Також блокчейн може відстежувати рух товарів та матеріалів по ланцюгу постачання, забезпечуючи прозорість, ефективність та запобігаючи шахрайству.

Блокчейн може використовуватися для безпечного зберігання медичних записів, покращення співпраці між медичними працівниками та надання пацієнтам більшого контролю над своїми даними.

Блокчейн може бути застосовано для безпечного та прозорого голосування, реєстрації власності та інших державних послуг.

## 1.2 Огляд існуючих методів голосування

У прийнятті рішень колективом, особливо коли йдеться про вибір проектів чи визначення пріоритетів, надзвичайно важливо мати справедливую та ефективну систему голосування. Існує багато методів голосування, кожен з яких має свої особливості та переваги, що дозволяють досягати різних цілей та враховувати різні аспекти думок учасників. Від класичного методу Борда до більш сучасного квадратичного голосування, ці методи допомагають максимально точно відобразити переваги та думки кожного учасника. Розглянемо основні існуючі методи голосування, які застосовуються для вибору проектів та прийняття колективних рішень: метод Борда, голосування за принципом більшості, кумулятивне голосування, схвалювальне голосування, оціночне голосування, квадратичне голосування та голосування з нечітким порогом [3].

Метод Борда передбачає, що кожен виборець може встановити строгий порядок переваг серед альтернатив. Виборець надає найбільш бажаній альтернативі  $N - 1$  балів, наступній —  $N - 2$  бали і так далі, де  $N$  — кількість альтернатив. Найменш бажана альтернатива отримує 0 балів. Альтернатива, яка набирає найбільшу суму балів, виграє. Цей метод враховує інтенсивність переваг виборців, де кожен крок між альтернативами рівновіддалений [3].

Серед переваг методу можна виділити врахування повного спектру

переваг виборців та забезпечення пропорційності, що може зменшити вплив крайніх думок. Однак, метод може бути складним для виборців через необхідність ранжування всіх альтернатив, потребує складного підрахунку голосів і схильний до тактичного голосування, де виборці можуть навмисно занижувати оцінки конкурентів.

Кумулятивне голосування, також відоме як "голосування відром", цей метод передбачає, що кожен виборець отримує фіксовану кількість голосів, зазвичай рівну кількості альтернатив, і може розподілити ці голоси будь-яким чином. Виборець може, наприклад, віддати всі голоси одній альтернативі або розподілити їх між кількома. Це дозволяє виразити як інтенсивність, так і широту переваг [3].

Переваги цього методу включають можливість висловити сильну підтримку окремим альтернативам та його придатність для виборів, де важлива підтримка кількох кандидатів. Недоліками є можливість надмірної концентрації голосів на одному кандидатові, що спричиняє перекося, та необхідність високого рівня стратегічного мислення від виборців.

У методі голосування за принципом більшості виборцям надається можливість віддати свій голос тільки за одну альтернативу, яка отримує 1 бал, тоді як всі інші отримують 0 балів. Це обмежує виборців у вираженні їхніх переваг між різними альтернативами [3].

Переваги цього методу включають простоту для виборців та підрахунку голосів, а також сприяння чіткому визначенню переможця. Недоліками є те, що метод не враховує інтенсивність переваг виборців, схильний до двопартійності та поляризації, а виборці можуть відчувати, що їх голос не враховується, якщо їх кандидат не виграє.

Метод оціночного голосування дозволяє виборцям оцінювати кожен альтернативу за шкалою з різною кількістю значень, визначеною залежно від здатності виборців робити змістовні розрізнення між альтернативами. Наприклад, у загальних виборах можна використовувати трьохрівневу шкалу, тоді як у випадку експертних рішень — п'ятирівневу [3].

Серед переваг цього методу є надання більш детальної інформації про переваги виборців та менша схильність до тактичного голосування, оскільки виборці можуть оцінювати всі альтернативи. Недоліками є складність підрахунку голосів та аналізу результатів, а також необхідність від виборців більшого розуміння та часу для оцінки всіх альтернатив.

Метод схвалювального голосування схожий на голосування за принципом більшості, але без обмеження в кількості альтернатив, які можуть отримати 1 бал. Виборець може схвалити будь-яку кількість альтернатив, що дає більше можливостей для вираження переваг [3].

Серед переваг цього методу є простота і зрозумілість для виборців, а також зменшення поляризації, дозволяючи виборцям підтримати кілька прийнятних кандидатів. Недоліками є те, що метод може не враховувати інтенсивність переваг, як інші методи, та може бути використаний для тактичного голосування, якщо виборці підтримують тільки найменш поляризуючих кандидатів.

Голосування з нечітким порогом використовується головним чином у криптовалютах. Цей метод передбачає, що набір переможців може включати кандидатів з нижчими балами і не включати кандидатів з вищими балами. Результат голосування залежить не тільки від балів кандидатів (тобто їх ранжування в бюлетенях виборців), але й від інших факторів, таких як бюджети відповідних проектів. Ця двофакторна залежність ускладнює теоретичний аналіз цього методу. Знання бюджетів проектів надає виборцям додаткову інформацію, яка допомагає приймати більш обдумані рішення. FTV забезпечує виконання правила Кондорсе, тобто явний лідер голосування завжди належить до набору переможців [4].

Квадратичне голосування є відносно новим методом, що дозволяє виборцям виражати інтенсивність своїх переваг. Кожному виборцю надається певна кількість кредитів, які він може розподілити між альтернативами. Вартість голосу за кожну альтернативу є квадратичною функцією кількості голосів, які виборець бажає віддати. Наприклад, щоб віддати 2 голоси за одну

альтернативу, виборець повинен витратити 4 кредити, а щоб віддати 3 голоси, потрібно витратити 9 кредитів. Це забезпечує баланс між широтою та інтенсивністю переваг виборців.

Тож, перераховані методи голосування мають свої певні особливості, для наочності порівняємо їх в таблиці 1.1.

Таблиця 1.1 – Порівняння методів голосування

| Назва методу                   | Кількість можливих оцінок              | Врахування інтенсивності переваг | Обмеження на розподіл голосів                |
|--------------------------------|----------------------------------------|----------------------------------|----------------------------------------------|
| Метод Борда                    | $N$                                    | Так                              | Немає                                        |
| Кумулятивне голосування        | $N + 1$                                | Так                              | Сума голосів дорівнює $N$                    |
| Голосування більшості          | 2                                      | Ні                               | Один голос на одну альтернативу              |
| Оціночне голосування           | Залежить від шкали                     | Так                              | Немає                                        |
| Схвалювальне голосування       | 2                                      | Ні                               | Немає                                        |
| Голосування з нечітким порогом | Залежить від системи оцінки та бюджету | Частково                         | Залежить від ранжування та бюджетів проектів |
| Квадратичне голосування        | Залежить від кредитів                  | Так                              | Квадратичні витрати кредитів                 |

Жоден метод агрегування суджень не може бути кращим за вихідні індивідуальні судження. З точки зору вимірювання, традиційні методи, такі як метод Борда, метод голосування за принципом більшості та кумулятивне

голосування, мають суттєві недоліки через обмеження можливих результатів. Тільки методи схвалювального, оціночного та квадратичного голосування вільні від таких обмежень, що робить їх єдиними справжніми кардинальними методами [3].

Квадратичне голосування та голосування з нечітким порогом пропонують збалансований підхід до вираження інтенсивності переваг, забезпечуючи додатковий рівень деталізації порівняно з іншими методами. Особливо метод голосування з нечітким порогом може бути корисним у випадках, коли додаткова інформація про бюджети проектів може вплинути на прийняття рішень.

### 1.3 Обґрунтування вибору математичної моделі голосування для покращення безпеки та надійності блокчейн-системи

Після аналізу існуючих методів голосування було обрано метод квадратичного голосування, так як він забезпечує збалансований підхід до вираження інтенсивності переваг виборців та має кілька важливих переваг, які сприяють покращенню безпеки та надійності блокчейн-систем, а саме:

- врахування інтенсивності переваг;
- протидія маніпуляціям;
- захист від сибіл-атак;
- залучення інформованих рішень;
- простота інтеграції;
- адаптивність.

Охарактеризуємо кожну з названих переваг. Квадратичне голосування дозволяє виборцям не лише вибирати між альтернативами, а й виражати ступінь своїх переваг. Це досягається завдяки квадратичній функції витрат голосів, що дозволяє виборцям розподіляти свої кредити відповідно до сили своїх переконань.

Завдяки квадратичній функції вартості голосів, система ускладнює концентрацію великої кількості голосів на одній альтернативі без значних

витрат. Це знижує можливість маніпуляцій з боку окремих виборців чи груп, забезпечуючи більш чесні результати голосування.

Квадратичне голосування ефективно захищає від сибіл-атак (коли одна особа створює багато фальшивих ідентичностей для отримання непропорційної кількості голосів). Вартість кожного додаткового голосу зростає квадратично, що робить таку атаку економічно не вигідною.

Ця модель стимулює виборців приймати більш обдумані рішення, оскільки виборці повинні ретельно зважувати свої переваги через обмеженість кредитів і зростаючу вартість додаткових голосів.

Квадратичне голосування добре підходить для інтеграції в блокчейн-системи завдяки своїй математичній ясності та простоті реалізації. Це сприяє створенню прозорої та надійної системи голосування, яка легко верифікується всіма учасниками мережі.

Квадратичне голосування може бути адаптоване до різних сценаріїв та умов, що робить його універсальним інструментом для різноманітних випадків використання у блокчейн-системах.

З огляду на ці переваги, квадратичне голосування є оптимальним вибором для покращення безпеки та надійності блокчейн-систем, сприяючи більш справедливому та стійкому процесу прийняття рішень.

#### 1.4 Висновки за розділом

У цьому розділі було проведено всебічний огляд блокчейн-технологій, їх ключових аспектів та основних областей застосування, а також методів голосування, які можуть бути інтегровані в блокчейн-системи для підвищення безпеки та надійності.

Спочатку були розглянуті основні принципи блокчейн-технологій, де було підкреслено їх децентралізовану природу, високу безпеку, прозорість та незмінність даних. Ці характеристики роблять блокчейн привабливим для різних галузей, включаючи фінанси, логістику, охорону здоров'я та урядові послуги. У фінансовому секторі, наприклад, блокчейн дозволяє значно

пришвидшити та здешевити міжнародні грошові перекази, у логістиці та управлінні ланцюгами постачання - забезпечити повну прозорість та можливість відстеження товарів на всіх етапах.

Далі були детально проаналізовані кілька методів голосування, кожен з яких має свої унікальні характеристики та обмеження. Наприклад, метод Борда дозволяє враховувати інтенсивність переваг виборців, в той час як голосування за принципом більшості є простим, але не враховує цієї інтенсивності.

Особлива увага була приділена квадратичному голосуванню, яке було обране для покращення безпеки та надійності блокчейн-систем. Цей метод дозволяє виборцям виражати інтенсивність своїх переваг за допомогою квадратичної функції витрат голосів, ускладнюючи концентрацію голосів на одній альтернативі без значних витрат і знижує можливість маніпуляцій.

Узагальнюючи, можна зробити висновок, що жоден метод агрегування суджень не може бути кращим за вихідні індивідуальні судження. Треба також зазначити, що традиційні методи голосування мають суттєві недоліки через обмеження можливих результатів. Метод квадратичного голосування, однак, забезпечує збалансований підхід до вираження інтенсивності переваг виборців та захищає від маніпуляцій.

## 2 ОПИС МЕТОДІВ ТА ЗАСОБІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ

### 2.1 Математичне подання принципу квадратичного голосування

Квадратичне голосування – це метод, що базується на квадратичному врахуванні голосів виборців. У цьому методі кожен учасник голосування має надати якусь певну кількість голосів, які потім підносяться до квадрату, щоб виразити їхню вартість. Тоді суму вкладу спонсорів за певний проект можна представити математично наступним чином, як приведено у формулі нижче [5].

$$s(p_m) = \sum_{i=1}^n c_i^2, \quad (2.1)$$

де  $p_m$  – проект, висунутий на голосування;

$m$  – загальна кількість проектів на голосуванні;

$s$  – сума вкладу спонсорів за голоси;

$n$  – загальна кількість спонсорів;

$c_i$  – кількість голосів спонсора.

В такому випадку бюджет конкурсу буде динамічним та мати залежність від вартості голосів. Його можна представити у вигляді формули, приведеної нижче.

$$b = \sum_{i=1}^m s(p_i), \quad (2.2)$$

де  $b$  – бюджет конкурсу;

$p_i$  – проект, висунутий на голосування;

$s$  – сума вкладу спонсорів за голоси;

$m$  – загальна кількість проектів на голосуванні.

Для вдалої програмної реалізації можна реалізувати більш чіткішу та інтуїтивно зрозумілу модель. Нова система буде включати два основних етапи: спонсорування проекту та голосування на основі вкладеної суми. Згідно з цією системою, спонсор спочатку робитиме внесок у проект, а кількість голосів за цей проект визначатиметься як корінь квадратний від суми внесених коштів.

Виразимо розрахунок бюджету конкурсу у формулі нижче.

$$b = \sum_{i=1}^n d_i , \quad (2.3)$$

де  $b$  – бюджет конкурсу;

$n$  – загальна кількість спонсорів;

$d_i$  – сума вкладу спонсора в бюджет конкурсу.

Далі кожен спонсор може проголосувати за проекти, розподіливши суму свого вкладу в бюджет конкурсу між проектами. Кількість голосів, яку спонсор може надати проекту визначається як корінь квадратний від суми внесених їм коштів за проект.

$$c_i = \sqrt{s_i} , \quad (2.4)$$

де  $c_i$  – кількість голосів спонсора;

$s_i$  – сума вкладу спонсора за проект.

Таким чином, виведемо формулу розрахунку кількості голосів для певного проекту. Її приведено нижче.

$$c(p_m) = \sum_{i=1}^n \sqrt{s_i} , \quad (2.5)$$

де  $c$  – кількість голосів;

$p_m$  – проект, висунутий на голосування;

$n$  – загальна кількість спонсорів;

$s_i$  – сума вкладу спонсора за проект.

Дану логіку голосування буде застосовано при реалізації програмного додатку. Вона дозволить визначати бюджет конкурсу ще до початку голосування, кількість голосів спонсорів залежатиме від того, скільки грошей вони готові виділити на проект.

## 2.2 Визначення вимог та функціоналу симулятора

Метою розробки є створення веб-симулятора для наочної демонстрації застосування математичної моделі квадратичного голосування (QV) для забезпечення безпеки та надійності голосувань у блокчейн-системах.

Проект передбачає створення веб-сайту для проведення голосувань та візуалізації результатів. Він включатиме можливість ініціювання голосувань з вибором проектів для спонсорування та використання математичної моделі QV для зарахування певної кількості голосів за проект. Ця модель дозволить визначати нечіткі пороги прийнятності та підраховувати кількість голосів, віддану спонсором.

Веб-сайт буде створено на Python з використанням Flask для побудови інтерактивного інтерфейсу. Тому він повинен мати достатню продуктивність для швидкого виконання симуляцій.

Основний функціонал симулятора включатиме можливості авторизації, спонсорування проектів, реєстрації проектів, голосування за проекти та виведення коштів переможцям. Симулятор передбачає використання моделі QV для підрахунку голосів та забезпечення чесності голосування.

Акцент буде зроблено на демонстрації ефективності QV моделі для забезпечення безпечного та надійного голосування. Візуалізація результатів голосування буде спрощеною для демонстраційних цілей, і результати будуть представлені з використанням QV моделі та порівнюватися з результатами простого підрахунку голосів.

## 2.3 Вибір технологій та інструментів для реалізації

Всі інструменти для розробки було обрано з поглядів на функціональні можливості, особистий досвід роботи з ними, та у якості найзручніших інструментів для реалізації даного завдання.

### 2.3.1 Мова програмування Python

Python – потужна та популярна мова програмування, яка має багато бібліотек та фреймворків для розробки веб-додатків, роботи з криптографією та обробки даних. Вона відмінно підходить для реалізації симулятора завдяки своїй зрозумілості, простоті та широким можливостям.

Python відомий своєю універсальністю та читабельністю коду. Його

синтаксис відрізняється лаконічністю, що дозволяє розробникам створювати програми, які займають менше місця, ніж аналогічні програми, написані на інших мовах. Однією з ключових особливостей Python є його багатоплатформовість. Програми, написані на Python, можуть працювати на різних операційних системах без необхідності вносити будь-які зміни [6].

Python має потужну стандартну бібліотеку, яка встановлюється автоматично разом з мовою. Вона містить готові інструменти для роботи з операційною системою, веб-сторінками, базами даних, різними форматами даних і навіть для створення графічного інтерфейсу програм. Python може використовуватися як для створення невеликих скриптів, так і для розробки великих систем. Ця мова є повністю безкоштовною, що робить її доступною для всіх користувачів без обмежень [6].

Однак, як і будь-яка інша мова, Python має свої недоліки. Одним з них є швидкість виконання коду, яка може бути нижчою, ніж у компільованих мов, таких як C. Це пов'язано з тим, що Python використовує інтерпретацію байт-коду, що може призвести до певного зниження продуктивності, особливо при обробці великих обчислювальних завдань. Проте зростання обчислювальної потужності сучасних комп'ютерів компенсує цей недолік для більшості застосунків. Крім того, час розробки програм на Python часто значно менший, ніж на інших мовах, завдяки його читабельності, простоті та великій екосистемі готових бібліотек і модулів [6].

Python використовується в різних областях, включаючи розробку ігор, веб-застосунків, створення внутрішніх інструментів для різних проектів, а також у наукових дослідженнях та розв'язанні прикладних завдань. У сфері веб-розробки, Python є популярним вибором для створення динамічних веб-додатків та веб-серверів, завдяки своїм фреймворкам, таким як Django і Flask.

Python також широко використовується для розробки внутрішніх інструментів у різних проектах, де його простота, швидкість розробки та розширюваність дозволяють ефективно створювати корисні програми та скрипти. У наукових дослідженнях, Python використовується для аналізу

даних, обробки зображень, моделювання та розв'язання математичних задач, завдяки багатому набору наукових бібліотек, таких як NumPy, Pandas, SciPy та інші. Загалом, Python є потужним і гнучким інструментом, який забезпечує різноманітні можливості в різних сферах розробки програмного забезпечення.

Для розробки системи голосування за проекти на основі Solidity, Python був обраний для створення основного додатку завдяки ряду ключових переваг. По-перше, Python має чудову бібліотеку web3.py, яка забезпечує зручний інтерфейс для взаємодії з блокчейнами на основі Ethereum. Це дозволяє легко інтегрувати Solidity контракти в додаток і взаємодіяти з ними.

По-друге, фреймворк Flask, який використовується в Python, є легким і гнучким інструментом для веб-розробки, що ідеально підходить для створення веб-інтерфейсу системи голосування. Його простота і розширюваність дозволяють швидко розробити і налаштувати веб-додаток. Крім того, Python відомий своєю простотою і читабельністю коду, що зменшує час розробки і спрощує підтримку коду. Це особливо важливо при розробці складних систем, де важлива легкість внесення змін і доповнень.

Додатково, Python має велику кількість сторонніх бібліотек і модулів, які можна легко інтегрувати у проект. Це дозволяє швидко додавати нові функціональні можливості і розширювати систему.

З урахуванням усіх цих переваг, Python став очевидним вибором для реалізації системи голосування, забезпечуючи ефективний розвиток, гнучкість і надійність.

### 2.3.2 Мова програмування для написання смартконтрактів Solidity

Solidity — це об'єктно-орієнтована, високорівнева мова програмування для реалізації смартконтрактів. Смартконтракти — це програми, які регулюють поведінку акаунтів у стані Ethereum. Solidity є мовою, що використовує синтаксис з фігурними дужками та призначена для роботи з Ethereum Virtual Machine (EVM). Вона була натхненна такими мовами, як C++, Python та JavaScript, і має ряд властивостей, які роблять її ідеальною для

створення смартконтрактів, зокрема статичну типізацію, підтримку наслідування, бібліотеки та складні користувацькі типи [7].

Однією з ключових переваг використання Solidity для створення системи голосування є її тісна інтеграція з блокчейном Ethereum. Це забезпечує високу ступінь децентралізації та безпеки, що є критично важливими для систем голосування. Крім того, Solidity забезпечує прозорість та незмінність даних, що гарантує чесність та точність голосування.

Іншою важливою перевагою є можливість створення смартконтрактів, які автоматично виконують певні умови без необхідності у сторонніх посередниках. У випадку системи голосування це означає, що результати будуть підраховані та опубліковані автоматично, що знижує ризик людських помилок та шахрайства.

Solidity також підтримує складні користувацькі типи даних та функціональні можливості, такі як наслідування та бібліотеки, що дозволяє створювати модульні та масштабовані контракти. Це особливо корисно при розробці системи голосування, де може знадобитися багаторазове використання коду та можливість розширення функціоналу у майбутньому.

Вибір Solidity для реалізації системи голосування був зумовлений її технічними можливостями та перевагами, які вона надає у контексті блокчейну Ethereum. Інтеграція з Python за допомогою web3.py та Flask дозволить створити повноцінний додаток, що забезпечить зручний користувацький інтерфейс та надійну роботу смартконтрактів. Такий підхід дозволить досягти високого рівня безпеки, прозорості та ефективності у системі голосування за проекти.

### 2.3.3 Фреймворки Flask та Truffle

Flask – це мікро-фреймворк для веб-розробки на Python, який надає простий і гнучкий інтерфейс для створення веб-додатків. Він відомий своєю простотою та гнучкістю, дозволяючи розробникам легко визначати маршрути URL, обробляти HTTP-запити та створювати динамічні веб-сторінки за

допомогою механізму шаблонів Jinja2. Flask також підтримує роботу з веб-формами, валідацію введення, сесії та куки, а також надає можливість працювати з реляційними базами даних через Flask-SQLAlchemy. Основними перевагами Flask є його гнучкість, зручність для початківців та наявність асинхронності у останніх версіях [8].

Flask був обраний для реалізації веб-додатку системи голосування через його простоту, гнучкість та можливість легко інтегрувати додаткові функції за допомогою розширень. Цей фреймворк дозволяє швидко розпочати розробку, маючи мінімалістичний підхід, що є ідеальним для невеликих і середніх додатків. Він забезпечує високу гнучкість у розробці, дозволяючи налаштовувати додаток під конкретні потреби проекту, що було критично важливим для інтеграції бібліотеки web3.py для взаємодії з блокчейном Ethereum. Таким чином, Flask був вибраний для реалізації веб-додатку системи голосування завдяки своїм технічним можливостям, гнучкості та простоті використання.

Truffle стане ідеальним вибором для реалізації системи голосування за проекти через свої властивості та переваги. Потужний інструментарій Truffle, який включає компіляцію, тестування та розгортання контрактів, забезпечить зручну та ефективну розробку системи голосування. Можливість автоматизованого тестування дозволить перевірити функціональність контрактів перед їх впровадженням у мережу Ethereum, що сприятиме виявленню та виправленню помилок на ранніх етапах розробки.

Крім того, зручний механізм управління версіями смартконтрактів через міграції дозволить ефективно керувати оновленням та розгортанням контрактів у різних середовищах. Завдяки широкому спектру інтеграційних можливостей, Truffle спростить інтеграцію з іншими компонентами системи голосування, що забезпечить її надійність та ефективність.

Усі ці особливості роблять Truffle оптимальним інструментом для розробки системи голосування, забезпечуючи її стабільність та успішне втілення.

#### 2.3.4 Засоби для створення інтерфейсу додатку: HTML, CSS, JavaScript

Для створення інтерактивного веб-інтерфейсу симулятора будуть використані HTML, CSS та JavaScript.

HTML визначає структуру веб-сторінки, включаючи розташування різних елементів, таких як текст, зображення та кнопки. Він є основою будь-якого веб-додатку, надаючи зрозумілу та логічну структуру для подальшого оформлення.

CSS відповідає за оформлення веб-сторінки, включаючи кольори, розміри, відступи, шрифти та інші візуальні аспекти. За допомогою CSS можна створювати привабливий та зручний дизайн для користувача, що покращить враження користувачів від взаємодії з додатком.

JavaScript – це мова програмування, яка додає інтерактивність до веб-сторінок. Вона дозволяє реагувати на події користувача, обробляти дані та маніпулювати вмістом сторінки. JavaScript використовується для створення різноманітних ефектів, анімацій та взаємодії з користувачем, що робить додаток більш привабливим та функціональним.

Такий набір технологій дозволить створити ефективний та зручний веб-інтерфейс для симулятора, що задовольнить потреби користувачів і сприятиме зручності в користуванні додатком.

#### 2.3.5 Інструменти розробки

У розробці веб-додатків та смартконтрактів для проекту буде використано різноманітні інструменти, що забезпечать зручність та ефективність роботи.

PyCharm виступає основним інструментом для розробки веб-додатків та смартконтрактів. Він є потужним середовищем розробки, призначеним для роботи з Python, HTML, CSS та JavaScript. PyCharm відрізняється широким спектром функцій, таких як редагування коду, підтримка систем контролю версій, відладчик та автодоповнення коду, що сприяє підвищенню

продуктивності та зручності розробки. Під час вибору середовища програмування було розглянуто три різні програми та після їх порівняння було остаточно обрано PyCharm. Критерії та середовища для порівняння наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння середовищ розробки

| Функції                                | Середовище розробки                         |                                        |                                        |
|----------------------------------------|---------------------------------------------|----------------------------------------|----------------------------------------|
|                                        | PyCharm                                     | Visual Studio Code                     | Sublime Text                           |
| Мова програмування                     | Python (основна), також підтримує інші мови | Підтримує багато мов, включно з Python | Підтримує багато мов, включно з Python |
| Інтеграція з Git                       | Так                                         | Так                                    | Так                                    |
| Автодоповнення коду                    | Так                                         | Так                                    | Так                                    |
| Налагодження коду                      | Так                                         | Так                                    | Немає вбудованого налагоджувача        |
| Рефакторинг коду                       | Так                                         | Обмежена функціональність              | Немає вбудованого рефакторингу         |
| Інтеграція з віртуальними середовищами | Так                                         | Потрібно встановити розширення         | Потрібно встановити розширення         |
| Інтелектуальне форматування коду       | Так                                         | Так                                    | Так                                    |
| Перевірка синтаксису                   | Так                                         | Так                                    | Так                                    |
| Безкоштовна версія                     | Так (Community Edition)                     | Так (Open Source)                      | Так                                    |

Ganache є незамінним інструментом для проведення тестів та взаємодії зі смартконтрактами. Цей інструмент надає локальну блокчейн мережу для розробки, тестування та налагодження смартконтрактів. Зручний інтерфейс та швидкий запуск дозволяють ефективно взаємодіяти з розробленими

смартконтрактами та перевіряти їх роботу перед реальним впровадженням.

Ремікс є інтегрованим середовищем розробки для Ethereum, яке надає зручний інтерфейс для написання, тестування та відлагодження смартконтрактів. Він володіє широким набором інструментів для роботи з смартконтрактами та можливість використання вбудованого відладчика для швидкого виявлення та виправлення помилок. Використання Реміксу спрощує розробку та тестування смартконтрактів, що підвищує продуктивність та швидкість розробки проекту.

## 2.4 Висновки за розділом

У цьому розділі було проведено детальний аналіз та описано методи та засоби, які будуть використовуватися для розв'язання поставленої задачі.

Розглянуто модель, що буде використовуватися у програмному продукті, з особливим акцентом на практичну використовуваність та ефективність її застосування.

Визначено вимоги до функціоналу програми та сформульовано основні критерії, які має відповідати реалізоване програмне забезпечення.

Було детально розглянуто та обґрунтовано засоби та методи, що будуть використовуватися для реалізації програмного продукту.

Модель реалізації базується на використанні мов програмування Python для розробки серверної частини та веб-інтерфейсу, а також Solidity для написання смарт-контрактів на блокчейні.

Для створення веб-інтерфейсу обрано Flask, який забезпечує зручну та ефективну розробку веб-додатків на Python.

В якості інструментів розробки обрано PyCharm для написання та налагодження коду, Remix для віртуальної розробки та тестування смарт-контрактів та Ganache для емуляції локальної блокчейн мережі.

Такий вибір засобів та технологій обґрунтований їхньою популярністю, стабільністю та зручністю використання для поставлених завдань.

## З ОПИС АЛГОРИТМУ ТА РЕАЛІЗАЦІЇ

### 3.1 Проектування системи голосування

На початку проектування системи голосування визначимося з основними можливостями користувача для взаємодії з майбутньою системою. Для цього створимо UML діаграму варіантів використання для додатку, її приведено на рис. 3.1. Виділимо основні функції та можливості користувача при використанні веб-сайту.

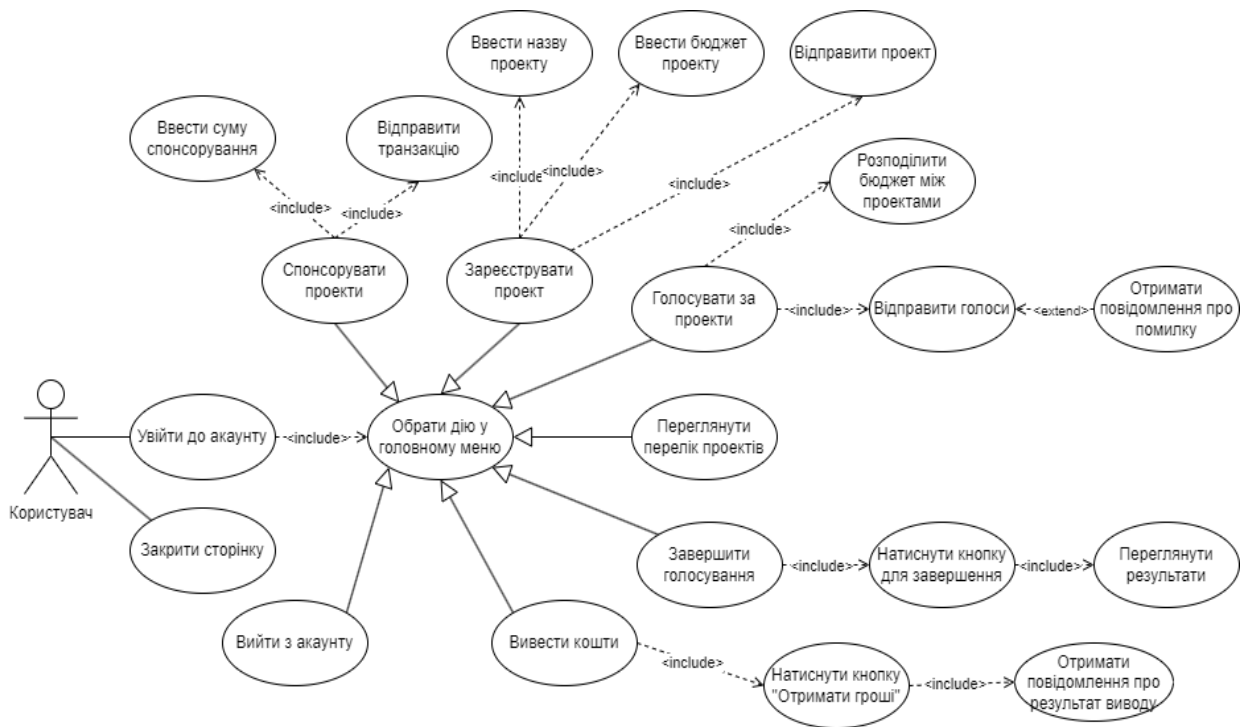


Рисунок 3.1 – Діаграма варіантів використання системи

Отже, користувач має змогу здійснювати різноманітні дії, пов'язані з управлінням проектами, акаунтами та голосуванням. Він може реєструвати нові проекти, спонсорувати їх, проходити авторизацію у системі, переглядати перелік проектів та результати голосувань.

Тепер визначимо основну структуру проекту. Він буде складатися з трьох блоків: смартконтракт на Solidity, основний додаток, написаний на мові Python та веб-інтерфейс, що будується за допомогою html та css.

На рис. 3.2 бачимо структуру проекту, що складається з декількох директорій. Директорію «truffle-sample1» створено з використанням фреймворку Truffle, що застосовується для створення децентралізованих додатків на базі блокчейну Ethereum. Її вміст було створено автоматично після ініціалізації проекту. Інші директорії створено вручну, в них зберігаються головний модуль проекту, абі код смартконтракту та елементи інтерфейсу веб-сторінки. Загалом, проект має типову ієрархію директорій та файлів, характерну для таких додатків.

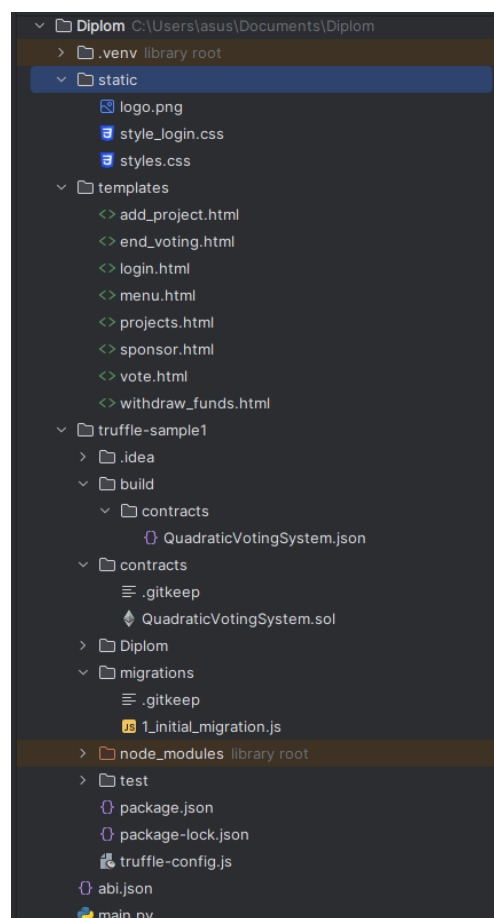


Рисунок 3.2 – Загальний вигляд структури проекту

Коренева директорія проекту містить файли конфігурації, такі як «package.json», «truffle-config.js» та «abi.json». Директорія «contracts» зберігає самі смарт-контракти, написані мовою Solidity, зокрема файл «QuadraticVotingSystem.sol», який реалізує логіку системи голосування з квадратичними вагами. Директорія «migrations» містить скрипти міграції для

розгортання смарт-контрактів у блокчейн Ethereum. Директорія «build» використовується для зберігання артефактів збірки проекту, включно зі скомпільованими смарт-контрактами у форматі JSON.

Веб-інтерфейс додатку представлений директоріями «static» та «templates». Перша містить статичні файли, такі як зображення та файли стилів, а друга – HTML-шаблони різних функціональних можливостей, як «login.html», «sponsor.html», «vote.html» тощо. Чітке розділення логіки смарт-контрактів, веб-інтерфейсу та допоміжних файлів полегшує розробку, тестування та розгортання децентралізованого додатку.

### 3.2 Розробка та розгортання смартконтракту

Для написання смартконтракту було використано мову програмування Solidity. Так як проект поки що розроблюється, застосуємо мережу в тестовому форматі. За допомогою програмного інструмента Ganache створимо середовище з віртуальними гаманцями. Для його створення у головному меню програми натискаємо на «New workspace» та у вікні, що з'явилося заповнюємо дані про назву проекту, прикріплюємо до нього вже створений на цей момент Truffle проект, який міститиме смартконтракт. У інших вкладках налаштовуємо всі необхідні параметри тестової мережі під себе. Після всіх налаштувань запускаємо мережу. Інтерфейс програми з створенням середовища приведено на рис. 3.3.

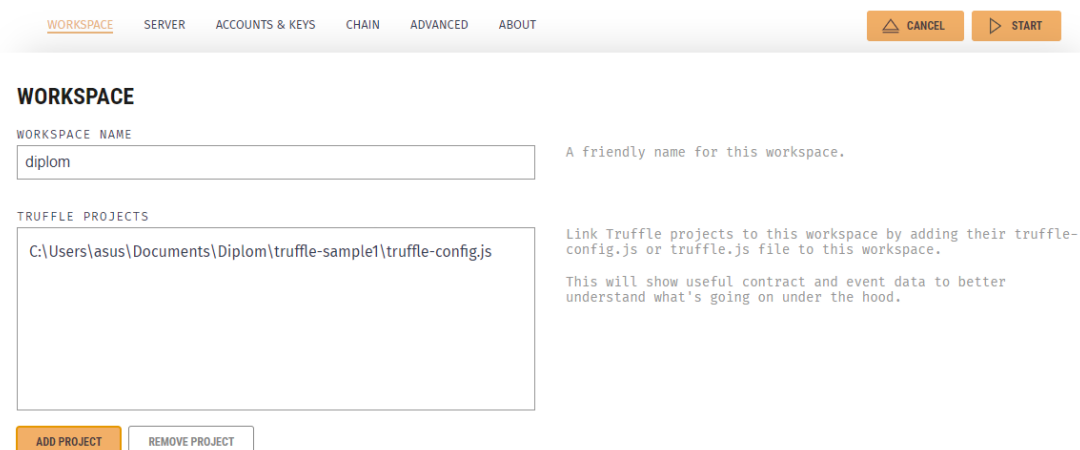
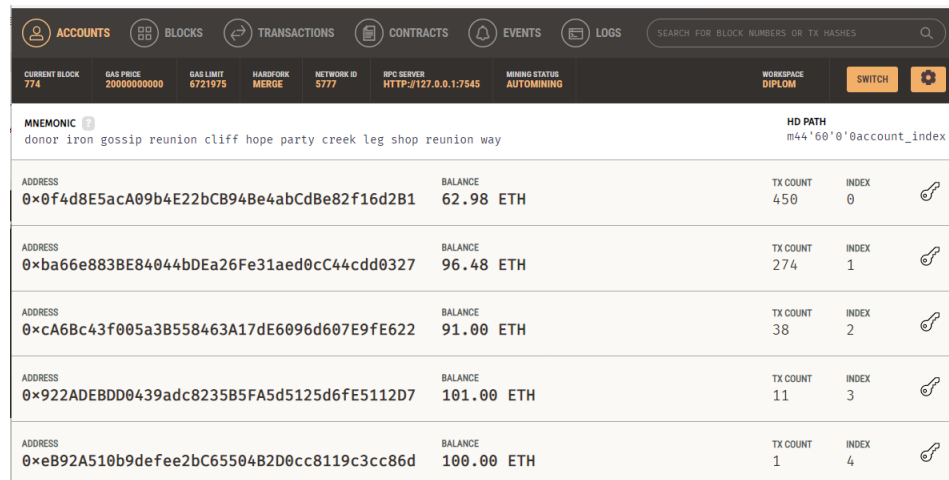


Рисунок 3.3 – Створення локальної мережі блокчейнів

Після створення мережі, отримуємо доступ до гаманців. Спочатку мережа містить налаштовану користувачем кількість гаманців та їх бюджет. Тепер гаманці цієї мережі можна використовувати для всіх тестів. Приклад інтерфейсу користувача у вкладці «акаунти» приведено на рис. 3.4.



| ADDRESS                                    | BALANCE    | TX COUNT | INDEX |
|--------------------------------------------|------------|----------|-------|
| 0x0f4d8E5acA09b4E22bCB94Be4abCdBe82f16d2B1 | 62.98 ETH  | 450      | 0     |
| 0xba66e883BE84044bDEa26Fe31aed0cC44cdd0327 | 96.48 ETH  | 274      | 1     |
| 0xcA6Bc43f005a3B558463A17dE6096d607E9fE622 | 91.00 ETH  | 38       | 2     |
| 0x922ADEBDD0439adc8235B5FA5d5125d6fE5112D7 | 101.00 ETH | 11       | 3     |
| 0xeB92A510b9defee2bC65504B2D0cc8119c3cc86d | 100.00 ETH | 1        | 4     |

Рисунок 3.4 – Доступні для тестів акаунти

Після запуску тестової мережі було написано смартконтракт під назвою «QuadraticVotingSystem». Основні елементи контракту включають масив адрес спонсорів, загальний бюджет, кількість проектів, статус завершення голосування, масив зареєстрованих проектів, а також мапінги для бюджетів спонсорів, їхніх голосів, статусу спонсора, індексів проектів за їхніми гаманцями та залишкових бюджетів.

Проект у контракті представлений структурою з полями для назви, бюджету, ваги голосів, гаманця проекту, статусу переможця та статусу зняття коштів. Це дозволяє детально фіксувати всі необхідні атрибути кожного проекту.

Функціонал контракту включає кілька ключових методів. Конструктор ініціалізує контракт з першим спонсором, яким стає його створювач. Спонсори можуть додавати себе до системи і вносити кошти через функцію «addSponsor». Кожен спонсор має можливість голосувати за проекти, використовуючи функцію vote, де кількість голосів повинна відповідати

кількості проектів. Після завершення голосування за допомогою функції «endVoting», визначаються переможці на основі ваги голосів. Переможці можуть знімати виділені їм кошти за допомогою функції «withdrawFunds».

Серед модифікаторів контракту варто відзначити «onlySponsor», який обмежує виконання певних функцій тільки для спонсорів. Контракт також включає кілька подій, таких як «SponsorAdded», «ProjectAdded», «VotingEnded» і «FundsWithdrawn», які сповіщають про важливі дії та зміни стану контракту.

Важливою особливістю є реалізація функції `sqrt`, яка обчислює квадратний корінь числа і використовується для квадратичного голосування, забезпечуючи математичну коректність підрахунку голосів. Така структура смартконтракту дозволяє обробляти всі функції, які потребує система голосування.

Після створення смартконтракту його треба скомпілювати. Спочатку контракт створюється через веб-додаток Remix, який дозволяє виконувати дії по компілюванню та розгортанню проекту прямо на сайті з використанням елементів графічного інтерфейсу. На рисунку 3.5 представлено інтерфейс з компілюванням контракту. Після компіляції з контракту можна отримати абі-код. Копіюємо його та переносимо у файл проекту «abi.json».

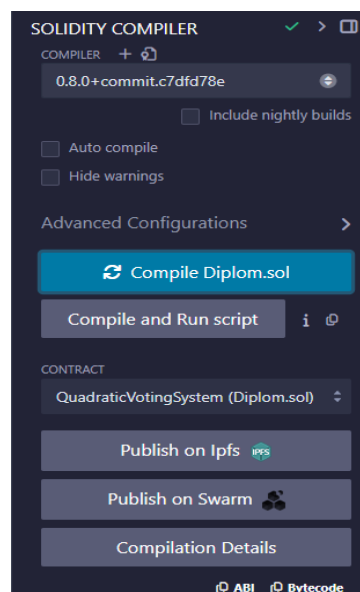


Рисунок 3.5 – Інтерфейс компіляції контракту в Remix

Після компіляції контракту переходимо на сторінку його розгортання як на рис. 3.6. У полі «середовище» підключаємося до створеної попередньо мережі. За необхідності виконавши ще додаткові налаштування розгортаємо контракт.

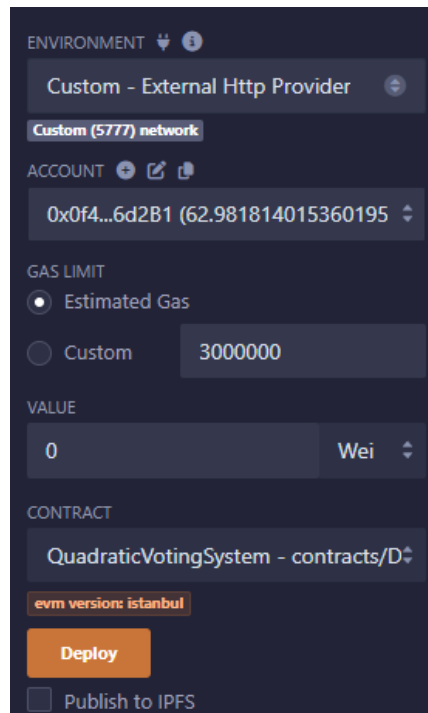


Рисунок 3.6 – Розгортання контракту

Після успішного розгортання контракту отримаємо його візуалізацію, як показано на рис. 3.7. Середовище розробки Remix надає зручні інструменти для взаємодії з контрактом. Завдяки інтерактивному інтерфейсу, користувач може легко виконувати різноманітні операції з контрактом, використовуючи спеціально створені для цього поля та кнопки. Перш за все, варто зазначити, що середовище Remix дозволяє виконувати функції контракту в режимі реального часу. Це означає, що після розгортання контракту можна одразу ж протестувати його функціонал, не виходячи з середовища розробки. Інтерфейс Remix автоматично генерує форми для введення даних і кнопки для виклику функцій контракту, що значно спрощує процес тестування та налагодження. Коли контракт успішно розгорнутий, він отримує унікальну адресу в блокчейні. Ця адреса є ключовим елементом для подальшої роботи з

контрактом, оскільки вона дозволяє ідентифікувати контракт серед усіх інших, що існують у мережі. Для подальшого використання контракту в головному додатку, написаному на мові Python, необхідно скопіювати цю адресу.

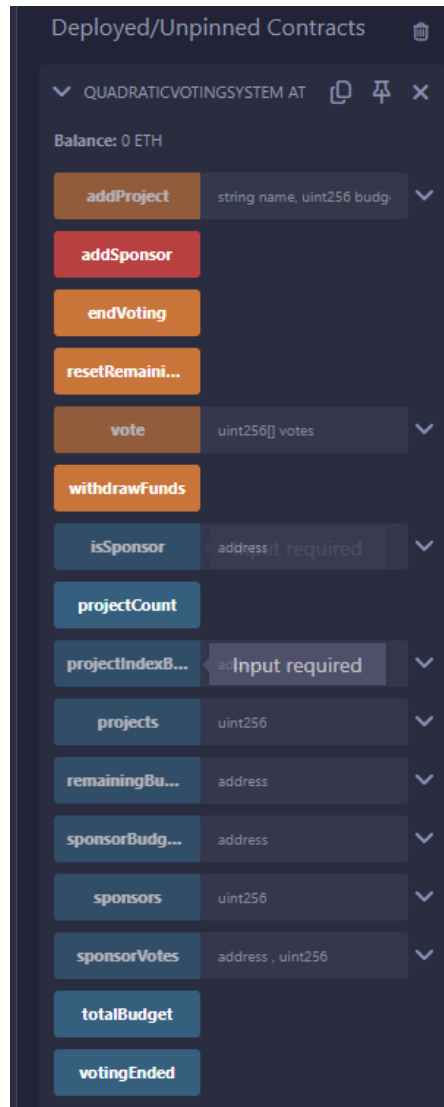


Рисунок 3.7 – Інтерфейс розгорнутого контракту

Також контракт можна скопіювати та розгорнути за допомогою фреймворку Truffle, що надасть змогу відстежувати контракти в Ganache. Після проведення тестувань у Remix перенесемо функціонал контракту туди, аби оптимізувати кількість одночасно використовуваних додатків. Після перенесення смартконтракту до локальної директорії в проєкті скопілюємо його за допомогою консолі. Алгоритм дій наведено на рис. 3.8.

```

PS C:\Users\asus\Documents\Diplom> cd truffle-sample1
PS C:\Users\asus\Documents\Diplom\truffle-sample1> truffle console
truffle(development)> compile

Compiling your contracts...

=====

> Compiling .\contracts\QuadraticVotingSystem.sol

> Artifacts written to C:\Users\asus\Documents\Diplom\truffle-sample1\build\contracts

> Compiled successfully using:
    - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang

```

Рисунок 3.8 – Компіляція контракту із застосуванням Truffle

Після цих дій бачимо, що контракт успішно скомпілювався та тепер він відображається в Ganache, але зі станом «не розгорнуто», як це видно на рис. 3.9.

| truffle-sample1 C:\Users\asus\Documents\Diplom\truffle-sample1 |              |          |
|----------------------------------------------------------------|--------------|----------|
| NAME                                                           | ADDRESS      | TX COUNT |
| QuadraticVotingSystem                                          | Not Deployed | 0        |

Рисунок 3.9 – Не розгорнутий контракт в Ganache

Для розгортання контракту створимо файл «1\_initial\_migration.js» в директорії проекту «migrations». Прописуємо там код для розгортання контракту.

```

var QuadraticVotingSystem = artifacts.require("QuadraticVotingSystem");

module.exports = function(deployer) {
  deployer.deploy(QuadraticVotingSystem);
};

```

Після цього розгортаємо контракт за допомогою команди «migrate» як на рис. 3.10 та спостерігаємо за інформацією про розгорнутий контракт, що з'явилася на рис. 3.11. Після чого перевіряємо стан контракту у Ganache як на рис. 3.12 та бачимо, що стан контракту змінився.

```

truffle(development)> migrate

Compiling your contracts...

=====

> Compiling .\contracts\QuadraticVotingSystem.sol

> Artifacts written to C:\Users\asus\Documents\Diplom\truffle-sample1\build\contracts

> Compiled successfully using:
  - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang

```

Рисунок 3.10 – Розгортання контракту

```

Starting migrations...
=====
> Network name: 'development'
> Network id: 5777
> Block gas limit: 6721975 (0x6691b7)

1_initial_migration.js
=====

Deploying 'QuadraticVotingSystem'
-----

> transaction hash: 0xfa6cc03470616bbe094b3eea53c2680aba45a235208d309ffe37fe6b9d3f0010

> Blocks: 0 Seconds: 0
> contract address: 0xbeb6f49329c0c75d98c28b6a2535d3e38c8d7fa3
> block number: 774
> block timestamp: 1718132746
> account: 0x0f4d8E5acA09b4E22bCB94Be4abCdBe82f16d2B1
> balance: 62.981814015360195142
> gas used: 2164826 (0x21085a)
> gas price: 2.500000008 gwei
> value sent: 0 ETH
> total cost: 0.005412065017318608 ETH

> Saving artifacts
-----
> Total cost: 0.005412065017318608 ETH

Summary
=====
> Total deployments: 1
> Final cost: 0.005412065017318608 ETH

```

Рисунок 3.11 – Інформація про розгорнутий контракт

| truffle-sample1 C:\Users\asus\Documents\Diplom\truffle-sample1 |                                            |          |          |
|----------------------------------------------------------------|--------------------------------------------|----------|----------|
| NAME                                                           | ADDRESS                                    | TX COUNT |          |
| QuadraticVotingSystem                                          | 0xbeb6f49329c0c75d98c28b6a2535d3e38c8d7fa3 | 0        | DEPLOYED |

Рисунок 3.12 – Розгорнутий контракт у Ganache

Тепер, після розгортання контракту, можемо взаємодіяти з ним за допомогою коду Python. Скопіюємо адресу контракту та вставимо її у код основного програмного модуля.

### 3.3 Створення основного програмного модуля

Основний модуль програми створюється із застосуванням мови Python. Код містить чимало функцій, які взаємодіють з веб-додатком, смартконтрактом та транзакціями. Розглянемо функції детальніше у порядку їх розташування у коді.

Першою розглянемо функцію «`build_and_send_transaction(txn)`» – дана функція приймає параметр «`txn`», а повертає квитанцію про транзакцію «`tx_receipt`». Основним призначенням функції є підпис транзакції за допомогою приватного ключа, відправка транзакції в блокчейн та отримання квитанції.

Друга функція «`add_sponsor(amount_eth)`» приймає деяку суму ЕТН, яку вносить спонсор. Дана функція створює та відправляє транзакцію про додавання спонсора та розмір його внеску. Наступною розглянемо «`add_project(name, budget_eth)`». Дана функція відповідає за відправлення транзакції про додавання нового проекту з визначеним бюджетом.

Ще одна з функцій, що будує та відправляє транзакції має назву «`vote(project_index)`». Вона приймає параметр «`project_index`», що містить індекс проекту, за який голосують. Після обробки дані формування транзакції дані відправляються у смартконтракт.

Наступна функція «`end_voting()`» формує та надсилає транзакцію про завершення голосування та збирає інформацію про переможців проектів.

Функція «`withdraw_funds()`» перевіряє статус проекту користувача (переміг він чи ні), потім будує і відправляє транзакцію для виведення коштів, якщо користувач є переможцем.

Наступна функція «`get_balance(address)`» отримує адресу користувача та визначає баланс ЕТН для отриманої адреси.

Функція «`check_user_project()`» перевіряє, чи є користувач власником проекту, що переміг у голосуванні. В результаті функція повертає значення «True» або «False».

Наступною розглянемо функцію «`login()`». Вона аутентифікує користувача за допомогою облікового запису та приватного ключа. В результаті виконання повертає сторінку для входу якщо користувач ввів невірні дані або перенаправляє його на сторінку головного меню, якщо було виконано вхід до облікового запису.

Функція «`menu()`» показує основне меню з інформацією про обліковий запис і баланс. На сторінці містяться функціональні кнопки управління, але цей момент буде розглянуто пізніше.

Наступною на черзі розглянемо функцію «`sponsor()`». Вона дозволяє користувачу внести певну суму грошей і стати спонсором проектів. Дана функція перенаправляє користувача або у головне меню, або повертає на сторінку голосування.

Далі функція «`add_project_route()`», вона дозволяє користувачу додати новий проект з певним бюджетом та у результаті дій користувача або повертає на сторінку додавання проектів, або перенаправляє у головне меню.

Наступною йде функція «`projects()`». Вона демонструє користувачеві всі проекти з їх деталями. В результаті дій користувача може перенаправити його на інші сторінки.

Функція «`vote_route()`» дозволяє користувачу розподілити свій бюджет і голосувати за проекти. Після відповідних дій користувача або повертає його на сторінку голосування або відображає повідомлення про помилку під час голосування.

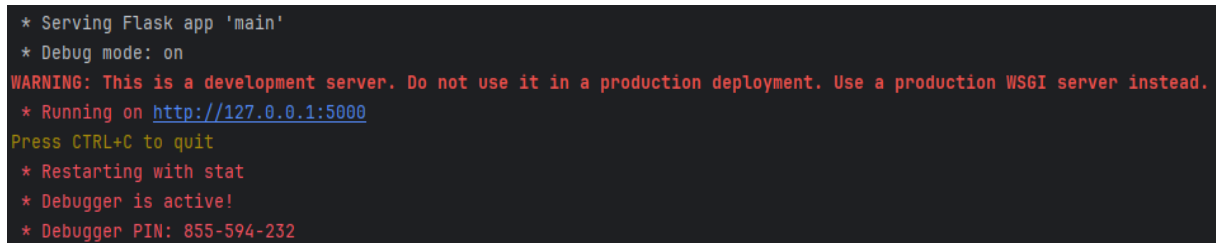
Далі функція «`end_voting_route()`», що дозволяє користувачу завершити голосування і переглянути його результати. В результаті або повертає сторінку для завершення голосування, або показує повідомлення про результат.

Наступною у черзі йде функція «`withdraw_funds_route()`», що дозволяє користувачу, проект якого є переможцем, вивести кошти. Дана функція

повертає сторінку для виведення коштів або показує повідомлення про результат виводу.

Останньою з функцій йде «logout()». Вона виконує вихід користувача і очищає сесію. В результаті спрацювання функції перенаправляє користувача на сторінку авторизації.

Під час запуску програмного додатку за допомогою фреймворку flask створюється адреса, за якою користувач може перейти на веб-сторінку для роботи з додатком. Так як додаток не прив'язаний до певного хоста, то створюється сервер для розробника. Приклад отримання посилання на веб-сайт наведено на рис. 3.13.

A screenshot of a terminal window showing the output of a Flask application running in development mode. The text is as follows:

```
* Serving Flask app 'main'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 855-594-232
```

Рисунок 3.13 – Перенаправлення розробника на веб-сторінку

Як можна спостерігати, сайт запущено та він працює на локальному хості, доступ до нього матиме лише розробник.

### 3.4 Розробка веб-додатку

Особливу вагу було приділено відтворенню інтерфейсу для програмного додатку. Для зручного користування додатком користувачем було реалізовано невеликий веб-сайт, який додає зручність та організованість використанню додатку. Для того, аби система голосування виглядала як повноцінний веб-додаток, було приділено час на його стилізацію. Перш за все, дизайн інтерфейсу був створений з урахуванням сучасних вимог та трендів у веб-дизайні.

Як відмінну рису веб-сайту було додано футер і хедер, які забезпечують структурованість та легкість навігації. Хедер містить основні елементи

навігації, такі як меню, посилання на сторінки перегляду проектів та голосування, що дозволяє користувачам швидко орієнтуватися на сайті. Також хедер містить назву веб-сайту та, на всіх сторінках окрім авторизації, інформацію про акаунт, а саме адресу гаманця та його баланс. Шаблон коду хедеру приведено нижче.

```
<div class="header">
    <h1>CryptoVote</h1>
</div>
<div class="menu">
    <div class="menu-left">
        <p>Акаунт: {{ account }}</p>
    </div>
    <p>Баланс: {{ balance }} ETH</p>
</div>
<div class="nav-buttons">
    <form action="{{ url_for('projects') }}" method="get">
        <button type="submit">Проекти</button>
    </form>
    <form action="{{ url_for('vote_route') }}" method="get">
        <button type="submit">Голосування</button>
    </form>
    <form action="{{ url_for('menu') }}" method="get">
        <button type="submit">Меню</button>
    </form>
</div>
```

Футер, своєю чергою, містить додаткову інформацію, таку як ім'я автора, інформація про проект та його логотип.

```
<footer>
    <div class="footer-container">
        <div class="left-col">
            <p>© 2024 Попова Анна</p>
        </div>
        <div class="center-col">
            
        </div>
        <div class="right-col">
```

```

        <p>Сайт зроблено під час написання дипломної роботи у</p>
        <p>Донецькому національному технічному університеті</p>
    </div>
</div>
</footer>

```

Ці елементи не лише покращують естетичний вигляд сайту, але й роблять його більш зручним для користувачів, що позитивно впливає на загальне враження від використання додатку.

### 3.5 Висновки за розділом

У даному розділі було детально розглянуто процес створення системи голосування, що включає проектування, розгортання смартконтракту, створення основного програмного модуля та розробку веб-додатку. Першим кроком стало проектування системи, де було розроблено загальну структуру проекту та діаграми використання, які наочно демонструють можливості користувачів. Ці дані допомогли зрозуміти основні сценарії взаємодії з системою та визначити ключові функції і ролі.

Наступним етапом стали розробка і розгортання смартконтракту, який забезпечує надійність і прозорість процесу голосування, використовуючи технологію блокчейн. У смартконтракті були реалізовані основні функції для реєстрації виборців, подачі голосів та підрахунку результатів, що забезпечує цілісність даних і запобігає їх фальсифікації.

Основний програмний модуль був створений на мові Python, що забезпечує ефективну взаємодію зі смартконтрактом і обробку даних голосування. У цьому модулі було реалізовано функції для підключення до блокчейн-мережі, взаємодії зі смартконтрактом та обробки транзакцій.

Останнім етапом стала розробка веб-додатку, який забезпечує зручний інтерфейс для користувачів, дозволяючи їм легко взаємодіяти з системою голосування. У додатку було створено футер та хедер, які забезпечують зручну навігацію та привабливий вигляд інтерфейсу.

Загалом, усі етапи розробки були спрямовані на створення

функціональної, надійної та безпечної системи голосування, що відповідає сучасним вимогам і стандартам. Проектування та реалізація кожного компоненту були виконані з урахуванням високої продуктивності та зручності користування, що робить систему готовою до використання у реальних умовах.

## 4 ОПИС РОБОТИ СИСТЕМИ ГОЛОСУВАННЯ

### 4.1 Інтерфейс веб-додатку

У результаті розробки було отримано веб-додаток із зручним для використання інтерфейсом. Приділимо увагу кожному з них.

Першою сторінкою, яка зустрічає користувача при запуску програми є сторінка авторизації. Її інтерфейс наведено на рис. 4.1. Тут можна спостерігати наявні хедер та футер сторінки з інформацією про проект. Посередині сторінки розташовані два поля вводу, «адреса гаманця» та «приватний ключ», куди користувач вводить відповідні дані.

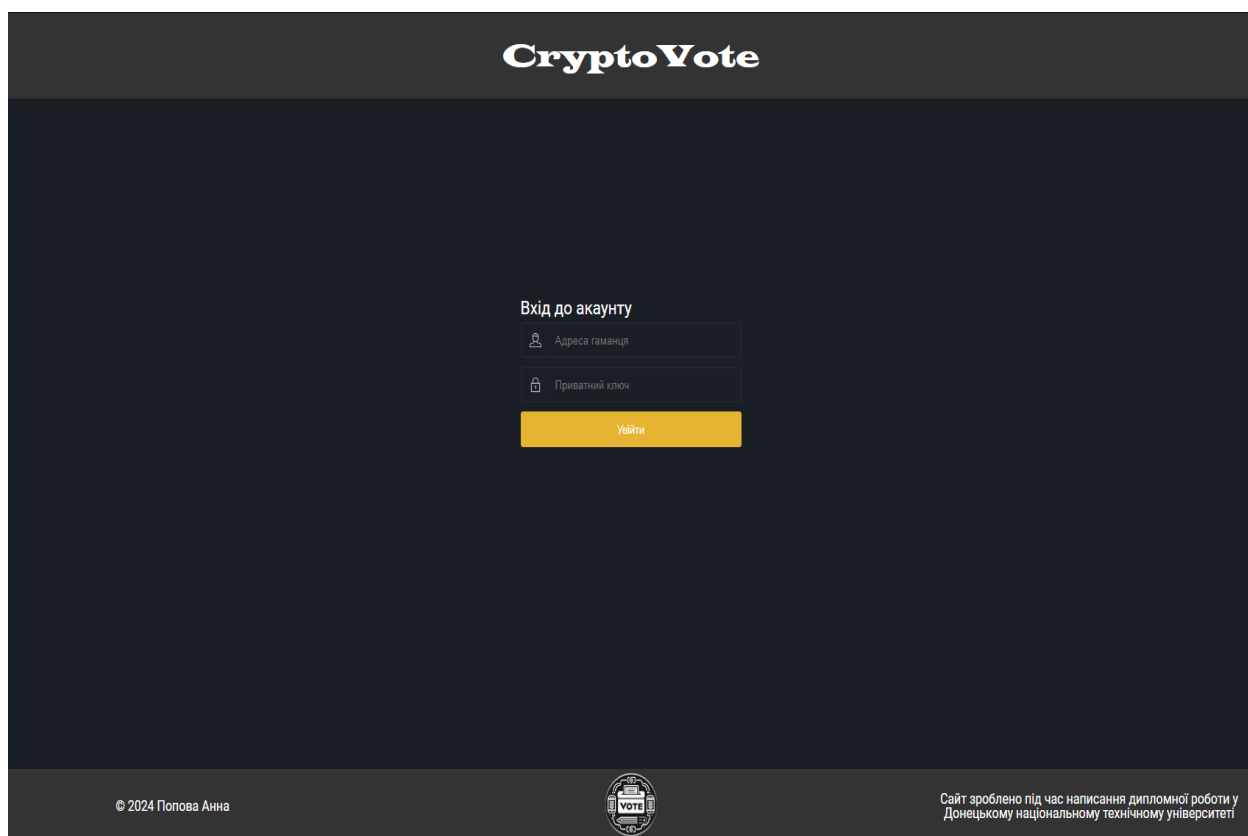


Рисунок 4.1 – Сторінка авторизації користувача

Після успішної авторизації користувач попадає на сторінку головного меню, її відображено на рис 4.2. Структура хедеру тут відрізняється. У лівому верхньому куті приведено адресу акаунту, з якого відбулася авторизація. У

протилежаю до адреси, правому куті відображається баланс гаманця користувача. Ще нижче є три основних кнопки навігації, а саме «проекти», що перенаправляє користувача на сторінку з переліком зареєстрованих проектів, також кнопка «голосування», яка направляє користувача на сторінку голосування за проекти. Та, остання в поточному рядку кнопка «меню», яка перенаправляє користувача на відповідну сторінку. Далі нижче перераховано 5 основних кнопок взаємодії з системою голосування для користувача та кнопка з функціоналом виходу з акаунту, яка перенаправляє користувача назад, на сторінку авторизації.

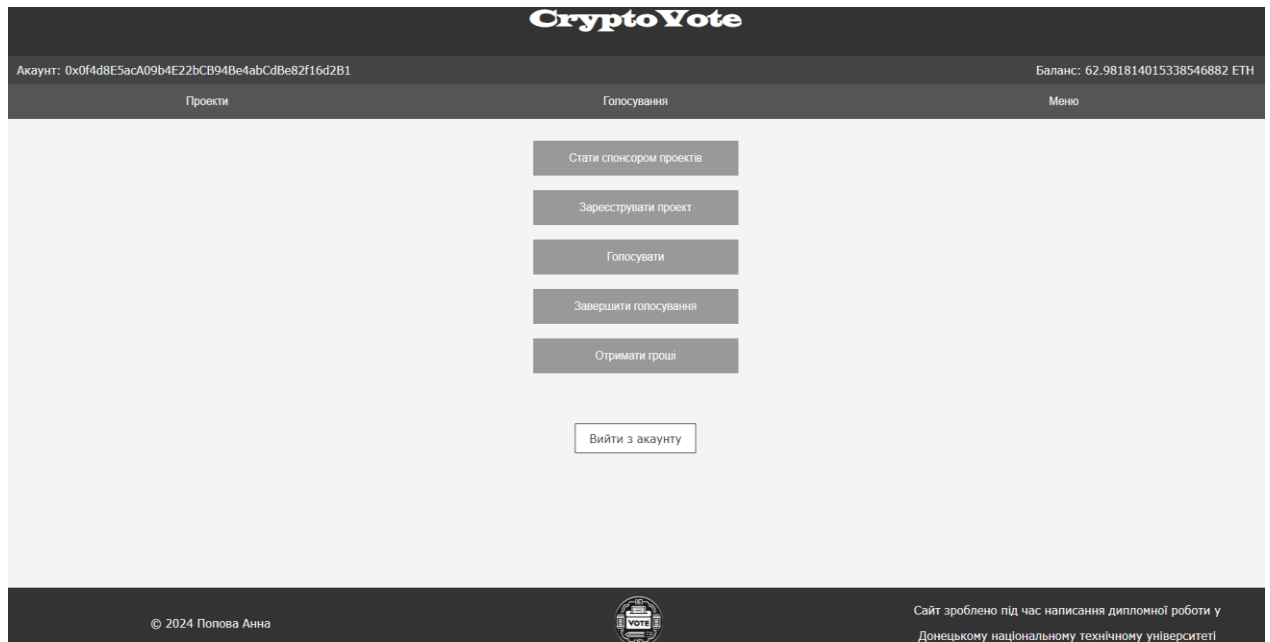


Рисунок 4.2 – Головне меню

Наступною переглянемо сторінку спонсорування проектів, її дизайн не відрізняється від дизайну минулої сторінки, різниця лише в наповненні. Сторінка так само має футер та хедер зі зручними кнопками навігації по системі. В середині сторінки міститься поле вводу, куди користувач вводить суму грошей в валюті ETH. Після введення суми, натиск на кнопку «Спонсувати» відправить гроші з рахунку користувача на рахунок контракту. Інтерфейс сторінки приведено на рис. 4.3.

**CryptoVote**

Акаунт: 0x0f4d8E5acA09b4E22bCB94Be4abCdBe82f16d2B1

Баланс: 62.981814015338546882 ETH

Проекти      Голосування      Меню

### Спонсувати проекти

Сума (в ETH):

Спонсувати

© 2024 Попова Анна

Сайт зроблено під час написання дипломної роботи у  
Донецькому національному технічному університеті

Рисунок 4.3 – Сторінка внесення грошей у бюджет конкурсу спонсором

Далі, з метою більш зручнішої демонстрації загального вигляду сторінок, зображення буде містити лише основну особливість сторінки, не включаючи в себе хедер та футер, адже він шаблонно повторюється на всіх сторінках. На рис. 4.4 зображено сторінку реєстрації нового проекту. Тут користувач вводить назву майбутнього проекту та бюджет, який потрібен для його втілення.

**CryptoVote**

6d2B1

Голосування

### Додати проект

Назва проекту:

Бюджет проекту (в ETH):

Додати проект

Рисунок 4.4 – Сторінка реєстрації нового проекту

Наступною розглянемо сторінку, приведену на рис. 4.5. Дана сторінка є сторінкою голосування. Тут буде відображатися перелік доступних для голосування проектів. Напроти назви кожного проекту буде з'являтися поле вводу, куди користувач вводитиме суму зі свого виділеного для голосування бюджету. Як можна спостерігати, на футері з даними акаунту з'явилося поле з бюджетом для голосування. Сума цього бюджету дорівнює сумі, яку спонсор відправив у бюджет контракту.

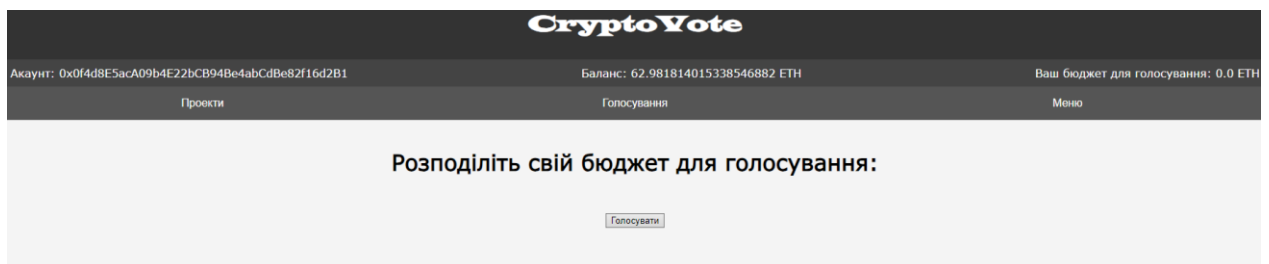


Рисунок 4.5 – Сторінка голосування

Наступною розглянемо сторінку на рис. 4.6, яка містить кнопку завершення голосування та місце для виведення переліку переможців. Після натиснення кнопки сторінка відображає ці дані.

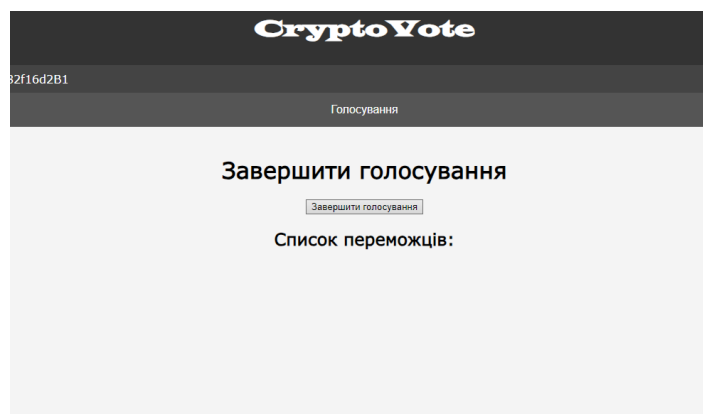


Рисунок 4.6 – Сторінка завершення голосування та виведення переможців

Такою ж сторінкою з незначним функціоналом є сторінка виведення коштів переможців, зображена на рис. 4.7. Після натиснення кнопки «отримати гроші» з'явиться відповідне до статусу користувача повідомлення.

Текст про вдале виведення коштів, або навпаки, повідомлення про те, що користувач не є переможцем.

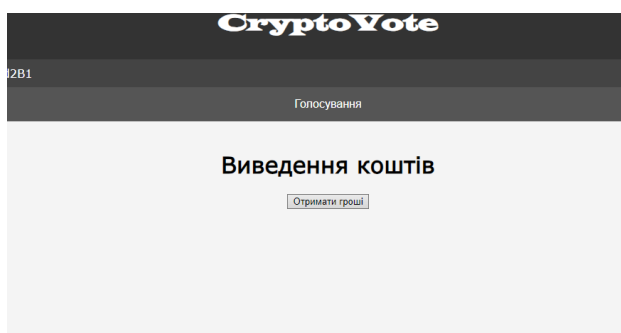


Рисунок 4.7 – Сторінка виведення коштів переможцем

Останньою розглянемо сторінку на рис. 4.8, яка відображає перелік зареєстрованих проектів. Всі нові проекти з'являються у таблиці з відповідними даними.

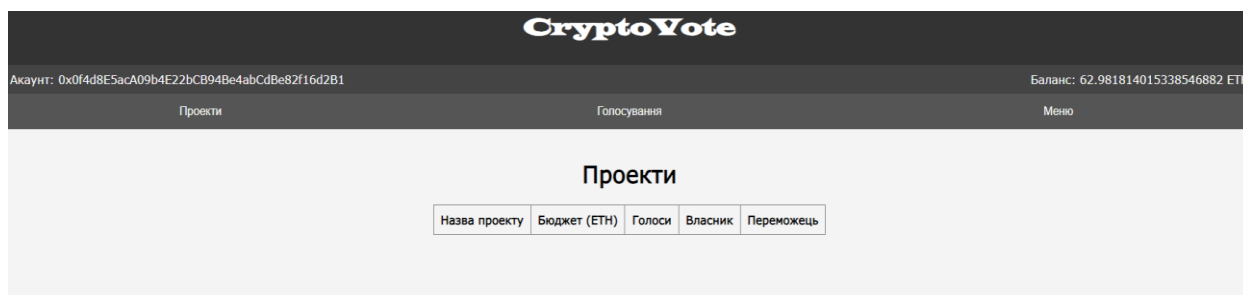


Рисунок 4.8 – Сторінка перегляду переліку проектів

Отже, реалізований веб-додаток має зручний та інтуїтивно зрозумілий інтерфейс, що покращує взаємодію користувача з ним.

## 4.2 Тестування роботи додатку

Після огляду інтерфейсу додатку, розглянемо його функціонал. Для цього спочатку створимо контракт та використаємо його адресу в основному коді. Для відстеження балансу та стану контракту використаємо програмний додаток Ganache. На рис. 4.9 відображено дані про новий розгорнутий контракт.

|                      |                          |                      |                   |                    |                                     |                             |
|----------------------|--------------------------|----------------------|-------------------|--------------------|-------------------------------------|-----------------------------|
| CURRENT BLOCK<br>775 | GAS PRICE<br>20000000000 | GAS LIMIT<br>6721975 | HARDFORK<br>MERGE | NETWORK ID<br>5777 | RPC SERVER<br>HTTP://127.0.0.1:7545 | MINING STATUS<br>AUTOMINING |
|----------------------|--------------------------|----------------------|-------------------|--------------------|-------------------------------------|-----------------------------|

← BACK

## QuadraticVotingSystem

ADDRESS  
0×beb6f49329c0C75d98c28B6A2535D3E38C8D7fA3

BALANCE  
0.00 ETH

CREATION TX  
0×Fa6cc03470616Bbe094b3EeA53C2680aBa45A235208D309Ffe37fe6B9d3f0010

### STORAGE

```

{ 10 items
  ▶ sponsors : [...] 1 item
  totalBudget : uint 0
  projectCount : uint 0
  votingEnded : bool false
  ▶ projects : [] 0 items
  ▶ sponsorBudgets : {} mapping 0 items
  ▶ sponsorVotes : {} mapping 0 items
  ▶ isSponsor : {} mapping 0 items
  ▶ projectIndexByWallet : {} mapping 0 items
  ▶ remainingBudgets : {} mapping 0 items
}

```

### TRANSACTIONS

NO TRANSACTIONS

Рисунок 4.9 – Новий контракт

Далі оберемо один з тестових гаманців в мережі та авторизуємося, як це відображено на рис. 4.10.

### Вхід до акаунту


0x0f4d8E5acA09b4E22bCB94Be4abCdB



Увійти

Рисунок 4.10 – Авторизація користувача

Далі перевіряємо баланс поточного гаманця, його відображено у правому верхньому куті, як можна спостерігати на рис. 4.11. Після чого

внесемо 2 ETH, як це відображено на рис. 4.12 та знову перевіримо стан балансу, його зображено на рис. 4.13.

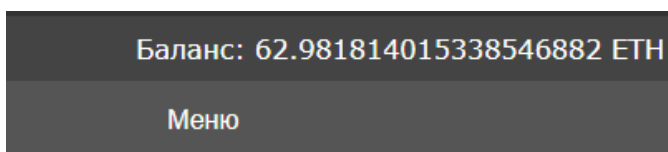


Рисунок 4.11 – Баланс акаунту до списання

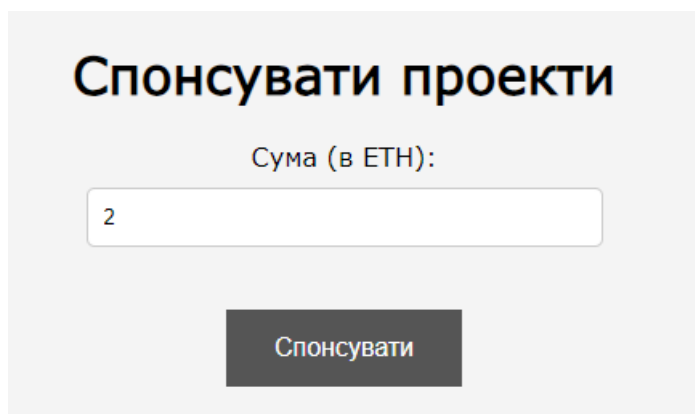


Рисунок 4.12 – Внесення суми для спонсування

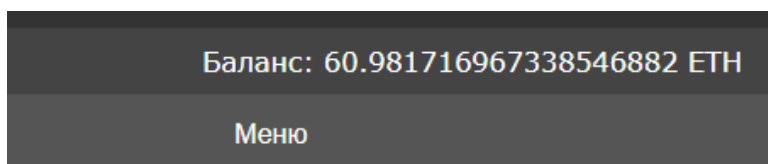


Рисунок 4.13 – Баланс акаунту після списання коштів

Як бачимо, кошти списалися вірно. Їх зарахування відбулося на рахунок контракту, це можна побачити на рис. 4.14.

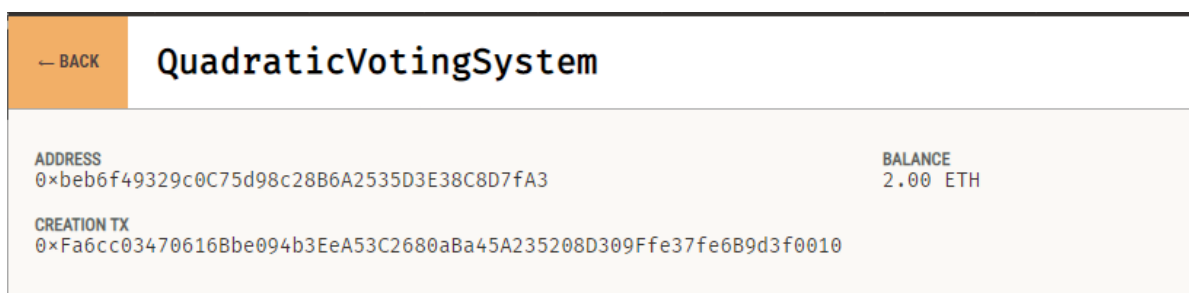


Рисунок 4.14 – Баланс контракту

Наступним кроком додамо новий проект. Введемо його назву та бюджет. Ці дії відображено на рис. 4.15.

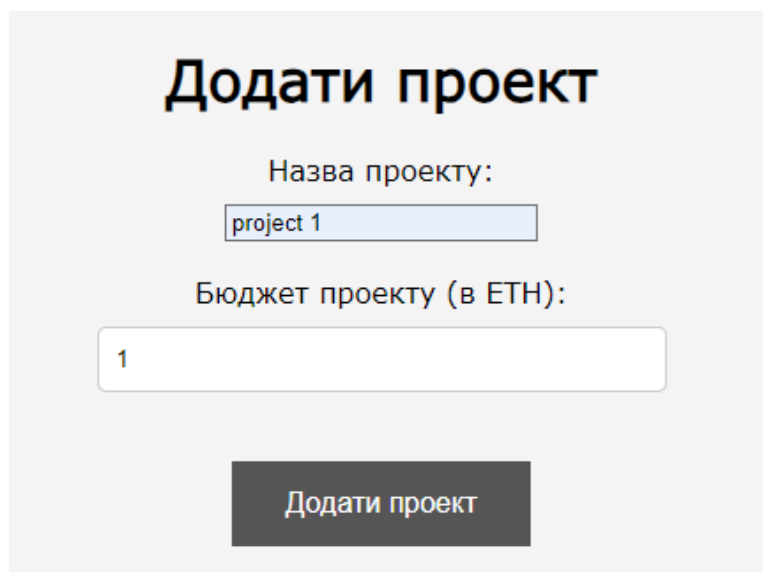


Рисунок 4.15 – Додавання нового проекту

Далі авторизуємося з іншого акаунту, пропонуємо голосування та перевіримо списання грошей. Ці дії виконано на рис. 4.16 – 4.18.

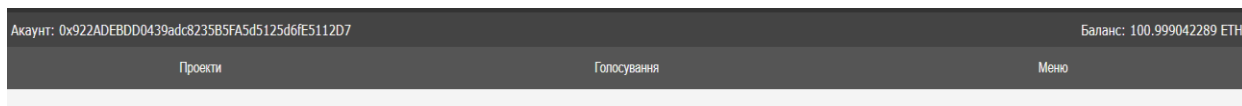


Рисунок 4.16 – Баланс другого акаунту до списання

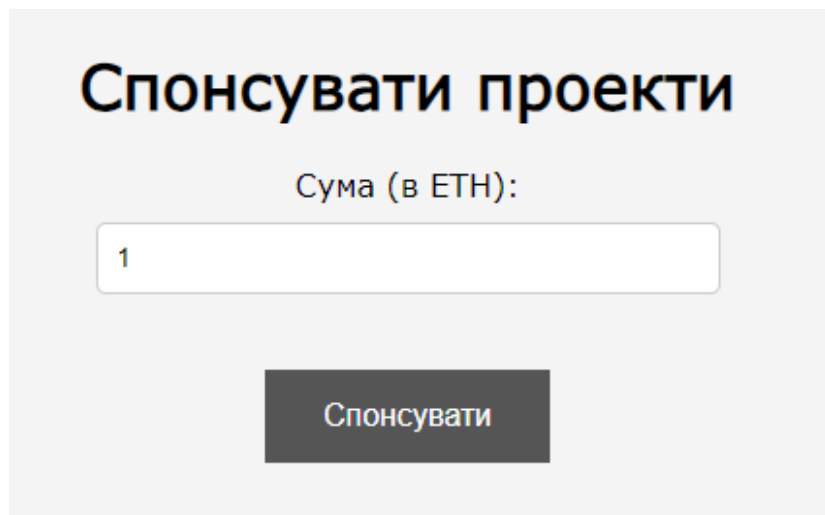


Рисунок 4.17 – Спонсування проектів

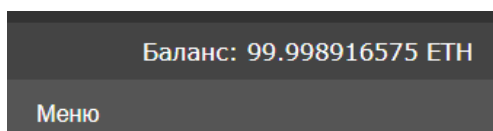


Рисунок 4.18 – Відстеження зміни балансу другого проекту

Далі знов перевіримо стан рахунку контракту як на рис. 4.19, баланс контракту і є бюджетом голосування.

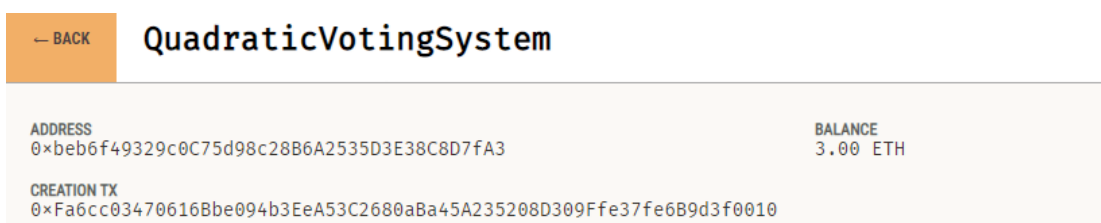


Рисунок 4.19 – Перевірка балансу контракту

Наступним кроком зареєструємо другий проект, ці дії виконані на рис. 4.20. Після цього перейдемо на сторінку перегляду даних про зареєстровані проекти як на рис. 4.21.

 A light grey rectangular form. At the top, the title 'Додати проект' is centered in bold. Below it, the label 'Назва проекту:' is followed by a text input field containing 'project 2'. Underneath, the label 'Бюджет проекту (в ETH):' is followed by a text input field containing '1'. At the bottom center is a dark grey button with the text 'Додати проект'.

Рисунок 4.20 – Реєстрація другого проекту

| Проекти       |              |        |                                            |            |
|---------------|--------------|--------|--------------------------------------------|------------|
| Назва проекту | Бюджет (ETH) | Голоси | Власник                                    | Переможець |
| project 1     | 1            | 0      | 0x0f4d8E5acA09b4E22bCB94Be4abCdBe82f16d2B1 | False      |
| project 2     | 1            | 0      | 0x922ADEBDD0439adc8235B5FA5d5125d6fE5112D7 | False      |

Рисунок 4.21 – Перелік зареєстрованих проектів

Далі проголосуємо за проекти, розподіливши бюджет для голосування між проектами. На рис. 4.22 відображено баланс для голосування другого акаунту.

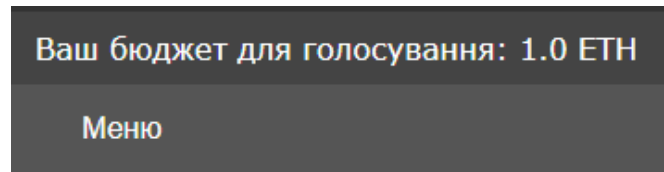


Рисунок 4.22 – Баланс для голосування другого акаунту

Як бачимо, користувач має на рахунку 1 ETH на розподілення між проектами. Для наведених полів може бути введено лише ціле число, тож даний бюджет треба розподілити між проектами, щоб одному дістався один ETH, другому нуль. Якщо спробувати розподілити на проекти не всю суму, тобто подати по нулю за проект, то голоси просто не зарахуються і виведеться повідомлення як на рис. 4.23. Це відбувається через умову логіки системи голосування, яка наголошує на тому, що розподілити слід весь бюджет.

A light gray rectangular box. At the top, it has a bold heading "Розподіліть свій бюджет для голосування:". Below the heading is a smaller line of text: "Ви повинні розподілити весь бюджет.". There are two input fields, each preceded by a label: "project 1" and "project 2". Both input fields contain the number "0". At the bottom center of the box is a button labeled "Голосувати".

Рисунок 4.23 – Спроба проголосувати при неправильному введенні даних

Далі введемо правильну суму та проголосуємо за перший проект як це показано на рис. 4.24. Перевіримо кількість голосів на сторінці переліку проектів, як це зображено на рис. 4.25.

## Розподіліть свій бюджет для голосування:

Ви повинні розподілити весь бюджет.

project 1

project 2

Голосувати

Рисунок 4.24 – Голосування за перший проект

## Проекти

| Назва проекту | Бюджет (ETH) | Голоси     | Власник                                    | Переможець |
|---------------|--------------|------------|--------------------------------------------|------------|
| project 1     | 1            | 1000000000 | 0x0f4d8E5acA09b4E22bCB94Be4abCdBe82f16d2B1 | False      |
| project 2     | 1            | 0          | 0x922ADEBDD0439adc8235B5FA5d5125d6fE5112D7 | False      |

Рисунок 4.25 – Зарахований голос в результаті голосування

Як можемо спостерігати на рис. 4.25 першому проекту нараховано  $1 * 10^9$  голосів. Це сталося з причини, що квадратний корінь з бюджету, який виділив спонсор, береться у одиниці виміру wei, яка дорівнює  $1 * 10^{-18}$  ETH.

Далі зайдемо в перший акаунт-спонсор та проголосуємо за проекти, як це зображено на рис. 4.26. Після чого знов перевіримо голоси проектів, рис. 4.27.

## Розподіліть свій бюджет для голосування:

project 1

project 2

Голосувати

Рисунок 4.26 – Голосування з першого акаунту

| Проекти       |              |            |                                            |            |
|---------------|--------------|------------|--------------------------------------------|------------|
| Назва проекту | Бюджет (ETH) | Голоси     | Власник                                    | Переможець |
| project 1     | 1            | 2414213562 | 0x0f4d8E5acA09b4E22bCB94Be4abCdBe82f16d2B1 | False      |
| project 2     | 1            | 0          | 0x922ADEBDD0439adc8235B5FA5d5125d6fE5112D7 | False      |

Рисунок 4.27 – Кількість голосів за проекти

Як можна проаналізувати, перший проект є абсолютним лідером голосування. Перейдемо у меню завершення голосування. Після натиску кнопки «завершити голосування», знизу з'явився перелік проектів – переможців. Як можемо спостерігати на рис. 4.28, тут відображено лише один проект, хоча бюджету вистачало на обидва. Причиною цього є те, що за другим проектом не було віддано жодного голосу. За логікою реалізованої програми, такий випадок несе за собою ігнорування проекту, без жодного голосу він не може стати переможцем. Стан проектів можна знову переглянути у меню «проекти», як бачимо на рис. 4.29, в першого проекту прапорець змінився на «True».

## Завершити голосування

Завершити голосування

### Список переможців:

project 1 (Бюджет: 1 ETH, Голоси: 2414213562)

Рисунок 4.28 – Завершення голосування

| Проекти       |              |            |                                            |            |
|---------------|--------------|------------|--------------------------------------------|------------|
| Назва проекту | Бюджет (ETH) | Голоси     | Власник                                    | Переможець |
| project 1     | 1            | 2414213562 | 0x0f4d8E5acA09b4E22bCB94Be4abCdBe82f16d2B1 | True       |
| project 2     | 1            | 0          | 0x922ADEBDD0439adc8235B5FA5d5125d6fE5112D7 | False      |

Рисунок 4.29 – Перелік проектів

Остання функція, яка буде перевірена це виведення коштів переможця. Спочатку спробуємо вивести кошти з другого акаунту, який не є переможцем. Як бачимо на рис. 4.30 виведення коштів є невдалим.

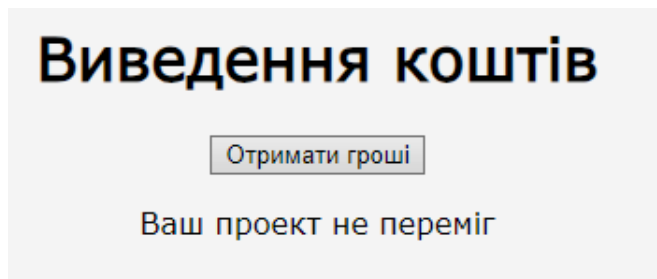


Рисунок 4.30 – Виведення коштів

Ті ж дії виконаємо з акаунту переможця. Як можемо спостерігати на рис. 4.31 виведення коштів виявилось вдалим. Перевіримо рахунок користувача і на рис. 4.32 можемо спостерігати, що баланс гаманця збільшився на 1 ЕТН, це сума вартості проекту.

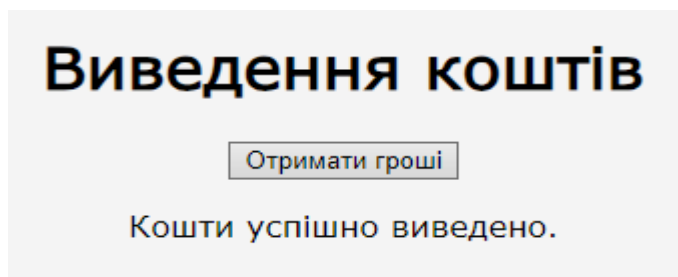


Рисунок 4.31 – Виведення коштів акаунтом-переможцем

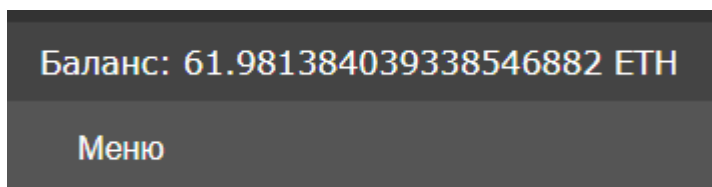


Рисунок 4.32 – Зміна балансу гаманця

Отже, як можемо судити з цього тесту, система функціонує та дозволяє спонсорувати проекти, реєструвати їх на голосування, голосувати та виводити гроші переможцем. Перераховані функції виконалися успішно.

#### 4.3 Аналіз результатів та визначення шляхів покращення додатку

У результаті проведення тестування було отримано різноманітні дані. Перш за все, можна було переконатися в тому, що система функціонує. Вона дозволяє виконувати заплановані дії, не дає стороннім людям виводити гроші, запобігає накручуванню голосів через не вигідний економічний вплив. Також що не було продемонстровано у тестах, але успішно функціонує – проголосувати повторно з одного й того самого гаманця неможливо.

Не зважаючи на те, що додаток добре працює є багато варіантів його покращення. Перш за все, не зважаючи на права доступу до особистих гаманців, сама система не має різноманітності прав доступу окрім звичайного користувача та спонсора. Тобто, користуватися функцією «завершити голосування» може кожний, вона не обмежена у правах. Для покращення цього аспекту слід було б виділити окремий модифікатор доступу типу «адміністратор» або «організатор».

Також для запобігання конфліктів в системі, можна контролювати період доступу до певної функції та зробити голосування поетапним. Наприклад, спочатку проходить термін встановлення бюджету і у цей час можна лише спонсорувати конкурс. Далі, та ж людина з модифікатором доступу керівника голосуванням, зачинає доступ до спонсування проєктів та відкриває доступ до реєстрації проєктів і так далі.

Запропоновані шляхи покращення продукту слід внести, якщо дану систему почати використовувати у реальних умовах, а не при розробці чи тестуванні. Це забезпечить захист системи під час голосування і ніхто не матиме можливості якось вплинути на його перебіг.

#### 4.4 Висновки за розділом

У даному розділі було детально розглянуто інтерфейс веб-додатку та проведено його тестування, а також проаналізовано результати голосування і визначено шляхи покращення додатку. Тестування підтвердило, що система функціонує належним чином, виконуючи всі заплановані функції та

забезпечуючи надійний захист від маніпуляцій, таких як виведення коштів сторонніми особами та накручування голосів. Важливим аспектом є те, що система не дозволяє повторне голосування з одного й того самого гаманця, що гарантує цілісність та прозорість процесу.

Однак під час тестування було виявлено деякі аспекти, що потребують вдосконалення. Зокрема, виявлено, що система має обмежене розмежування прав доступу, що створює потенційні ризики у процесі управління голосуванням. Запропоновано впровадити додаткові модифікатори доступу, такі як "адміністратор" або "організатор", для чіткішого контролю над процесом голосування.

Також були запропоновані покращення у вигляді контролю періодів доступу до певних функцій та впровадження поетапного голосування, що допоможе зменшити ризики конфліктів і забезпечити більшу цілісність процесу.

Загалом, результати аналізу показують, що система голосування здатна ефективно функціонувати в реальних умовах, але її потенціал може бути значно підвищений шляхом впровадження зазначених покращень. Це сприятиме підвищенню рівня захисту, прозорості та справедливості процесу голосування, що є ключовими вимогами для будь-якої сучасної системи електронного голосування.

## ВИСНОВКИ

У результаті виконання дипломної роботи було досягнуто значних результатів та вирішено низку поставлених завдань, хоча деякі проблеми залишилися невирішеними.

Найбільш важливі результати виконання завдання полягають у проведенні всебічного аналізу існуючих підходів до забезпечення безпеки голосувань у блокчейн-системах, зокрема, розглянуто квадратичне голосування як перспективну модель для підвищення безпеки та надійності. Модель квадратичного голосування була детально досліджена і обґрунтована для використання у блокчейн-системах. Виявлено, що цей метод дозволяє більш ефективно виражати інтенсивність переваг виборців і знижує можливість маніпуляцій.

Було розроблено програмне забезпечення для системи голосування, що включає серверну частину на Python, смарт-контракти на Solidity та веб-інтерфейс на Flask. Виконано проектування, розробку та розгортання смарт-контракту, який забезпечує прозорість і надійність процесу голосування. Проведено детальне тестування системи, яке підтвердило її функціональність та надійність. Виявлено та запропоновано шляхи вдосконалення системи, зокрема, розширення прав доступу та покращення контролю за процесом голосування.

Наукова новизна отриманих результатів полягає у застосуванні моделі квадратичного голосування в блокчейн-системах. Вперше було проведено детальне дослідження і розробка програмного забезпечення, яке інтегрує модель квадратичного голосування у блокчейн-систему. Виявлено нові аспекти використання цієї моделі для покращення безпеки та надійності електронних голосувань. Зокрема, модель дозволяє більш ефективно запобігати централізації голосів, що знижує ризик домінування однієї групи виборців. Крім того, механізм квадратичного голосування виявився

ефективним у зниженні ризиків підробки результатів та фальсифікації голосів завдяки високим витратам на маніпуляції з великою кількістю голосів.

Практичне значення розробленої системи голосування полягає в її потенціалі для використання у реальних голосуваннях у різних організаціях, що потребують високого рівня безпеки та прозорості. Система може бути впроваджена у різні галузі, включаючи урядові послуги, корпоративні голосування та громадські ініціативи. Також розроблена система може використовуватися для навчальних цілей, що сприятиме поширенню знань про блокчейн-технології та математичні моделі голосування.

Під час тестування було виявлено деякі аспекти, що потребують вдосконалення. Зокрема, система має обмежене розмежування прав доступу, що створює потенційні ризики у процесі управління голосуванням. Запропоновано впровадити додаткові модифікатори доступу, такі як "адміністратор" або "організатор", для чіткішого контролю над процесом голосування. Також були запропоновані покращення у вигляді контролю періодів доступу до певних функцій та впровадження поетапного голосування, що допоможе зменшити ризики конфліктів і забезпечити більшу цілісність процесу.

Таким чином, виконана робота підтвердила доцільність використання моделі квадратичного голосування для підвищення безпеки та надійності голосувань у блокчейн-системах та відкриває нові перспективи для подальших досліджень і практичного застосування.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Розповідаємо, що таке блокчейн простими словами – як працює мережа, як будуються блоки, що роблять майнери. URL: <https://apix-drive.com/ua/blog/useful/tehnologija-blokchejn-sho-ce-i-jak-pracjue>.
2. Positives and Negatives of Blockchain. URL: <https://academy.binance.com/ru/articles/positives-and-negatives-of-blockchain>
3. Hillinger C. Voting and the cardinal aggregation of judgments. 2004. С. 2-5.
4. Kaidalov D., Kovalchuk L., Nastenka A., Rodinko M., Shevtsov O., Oliynykov R. Ethereum Classic Treasury System Proposal / The Veritas Team. Research Report. 2017. 39 с.
5. Benhaim A., Falk B. H., Tsoukalas G. Balancing Power in Decentralized Governance: Quadratic Voting under Imperfect Information. 2023. С. 1-4.
6. Путівник мовою програмування Python. URL: <https://pythonguide.rozh2sch.org.ua>.
7. Solidity Documentation. URL: <https://docs.soliditylang.org/en/v0.8.26/>
8. Flask. URL: <https://blog.skillfactory.ru/glossary/flask/>
9. Posner E. A., Weyl E. G. Voting Squared: Quadratic Voting in Democratic Politics. 2015. С. 471–475.
10. Lalley S. P. Quadratic Voting: How Mechanism Design Can Radicalize Democracy // Article in AEA Papers and Proceedings. 2018. С. 1–2.
11. Mathematics of Voting. URL: <https://brilliant.org/wiki/mathematics-of-voting/>
12. An Introduction to Quadratic Voting. URL: <https://bobbay.hashnode.dev/an-introduction-to-quadratic-voting>
13. Fritsch R., Müller M., Wattenhofer R. Analyzing Voting Power in Decentralized Governance: Who controls DAOs? ETH Zurich, 2022. С. 4.
14. Ordinal versus cardinal voting rules: A mechanism design approach. URL: <https://www.sciencedirect.com/science/article/abs/pii/S0899825617300763>

15. A Simple Guide to Quadratic Voting. URL: <https://medium.com/tally-blog/a-simple-guide-to-quadratic-voting-327b52addde1>
16. Approval Voting.  
URL: <https://www.cambridge.org/core/journals/american-political-sciencereview/article/abs/approvalvoting/7CE5DEEE235794B0B12F76ADAE621482#access-block>
17. Chandar B. K., Weyl E. G. Quadratic Voting in Finite Populations. 2019. C.1.
18. Miller J., Weyl E. G., Erichsen L. Beyond Collusion Resistance: Leveraging Social Information for Plural Funding and Voting. 2022. C.1-2.
19. Smith W. D. Range Voting satisfies properties that no rank-order system can. Center for Range Voting. 2007. C.2-4.
20. Vasiljev S. A. Cardinal Voting: the Way to Escape the Social Choice Impossibility. 2021. C.1-2.
21. Benjamin D., Heffetz O., Kimball M., Lougee D. The Relationship Between the Normalized Gradient Addition Mechanism and Quadratic Voting. Public Choice, 2017. C.3-4.
22. Dimitri N. Quadratic Voting in Blockchain Governance. 2022. C.5.

Додаток А

Зауваження нормоконтролера

### Таблиця А.1 – Зауваження нормоконтролера

[illegible]

## Додаток Б

### Фрагменти лістингу програми

#### Лістинг смартконтракту

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.8.0;

contract QuadraticVotingSystem {
    address[] public sponsors;
    uint public totalBudget;
    uint public projectCount;
    bool public votingEnded;

    struct Project {
        string name;
        uint budget;
        uint votingWeight;
        address payable projectWallet;
        bool isWinner;
        bool fundsWithdrawn;
    }

    Project[] public projects;
    mapping(address => uint) public sponsorBudgets;
    mapping(address => mapping(uint => uint)) public sponsorVotes;
    mapping(address => bool) public isSponsor;
    mapping(address => uint) public projectIndexByWallet;
    mapping(address => uint) public remainingBudgets;

    event SponsorAdded(address sponsor, uint amount);
    event ProjectAdded(string name, uint budget, address projectWallet);
    event VotingEnded(Project[] winners);
    event FundsWithdrawn(address projectWallet, uint amount);

    modifier onlySponsor() {
        require(isSponsor[msg.sender], "Only sponsors can perform this
action");
    }
}
```

```

    }

    constructor() {
        sponsors.push(msg.sender);
        isSponsor[msg.sender] = true;
    }

    function addSponsor() public payable {
        require(msg.value > 0, "Sponsor contribution must be greater than 0");
        sponsors.push(msg.sender);
        totalBudget += msg.value;
        sponsorBudgets[msg.sender] = msg.value;
        isSponsor[msg.sender] = true;
        emit SponsorAdded(msg.sender, msg.value);
    }

    function addProject(string memory name, uint budget, address payable projectWallet) public {
        require(budget > 0, "Project budget must be greater than 0");
        projects.push(Project(name, budget, 0, projectWallet, false, false));
        projectIndexByWallet[projectWallet] = projectCount;
        projectCount++;
        emit ProjectAdded(name, budget, projectWallet);
    }

    function vote(uint[] memory votes) public onlySponsor {
        require(!votingEnded, "Voting has already ended");
        require(votes.length == projects.length, "You must vote for all projects");

        for (uint i = 0; i < votes.length; i++) {
            uint voteWeight = votes[i];
            projects[i].votingWeight += voteWeight;
            sponsorVotes[msg.sender][i] = voteWeight;
        }

        sponsorBudgets[msg.sender] = 0;
    }

```

```

function endVoting() public onlySponsor {
    require(!votingEnded, "Voting has already ended");
    require(projects.length > 0, "No projects registered");

    uint remaining = totalBudget;
    uint maxVotingWeight = 0;
    for (uint i = 0; i < projects.length; i++) {
        if (projects[i].votingWeight > maxVotingWeight) {
            maxVotingWeight = projects[i].votingWeight;
        }
    }

    for (uint i = 0; i < projects.length; i++) {
        if (projects[i].votingWeight == maxVotingWeight &&
projects[i].budget <= remaining) {
            projects[i].isWinner = true;
            remaining -= projects[i].budget;
        }
    }

    votingEnded = true;

    Project[] memory winners = new Project[] (projects.length);
    uint winnersCount = 0;
    for (uint i = 0; i < projects.length; i++) {
        if (projects[i].isWinner) {
            winners[winnersCount] = projects[i];
            winnersCount++;
        }
    }
    emit VotingEnded(winners);
}

function withdrawFunds() public {
    require(votingEnded, "Voting has not ended yet");
    uint index = projectIndexByWallet[msg.sender];
    Project storage project = projects[index];

```

```

        require(project.isWinner, "Only winning projects can withdraw
funds");
        require(!project.fundsWithdrawn, "Funds already withdrawn");

        uint amount = project.budget;
        project.budget = 0;
        project.fundsWithdrawn = true;
        project.projectWallet.transfer(amount);

        emit FundsWithdrawn(project.projectWallet, amount);
    }

    function resetRemainingBudget() public {
        require(isSponsor[msg.sender], "Only sponsors can perform this
action");
        remainingBudgets[msg.sender] = 0;
    }

    function sqrt(uint x) private pure returns (uint) {
        uint z = (x + 1) / 2;
        uint y = x;
        while (z < y) {
            y = z;
            z = (x / z + z) / 2;
        }
        return y;
    }
}

```

### Лістинг основного програмного модулю

```

import json
import secrets
from web3 import Web3
from flask import Flask, render_template, request, redirect, url_for,
session
from math import sqrt

```

```
secret_key = secrets.token_hex(32)
```

```
app = Flask(__name__)
```

```
app.secret_key = secret_key
```

```
selected_account = None
```

```
private_key = None
```

```
default_account = None
```

```
w3 = Web3(Web3.HTTPProvider('http://127.0.0.1:7545'))
```

```
contract_address = '0xbeb6f49329c0C75d98c28B6A2535D3E38C8D7fa3'
```

```
with open('abi.json', 'r') as f:
```

```
    contract_abi_json = f.read()
```

```
contract_abi = json.loads(contract_abi_json)
```

```
contract = w3.eth.contract(address=contract_address, abi=contract_abi)
```

```
def build_and_send_transaction(txn):
```

```
    signed_txn = w3.eth.account.sign_transaction(txn, private_key)
```

```
    tx_hash = w3.eth.send_raw_transaction(signed_txn.rawTransaction)
```

```
    tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)
```

```
    return tx_receipt
```

```
def add_sponsor(amount_eth):
```

```
    global selected_account, private_key
```

```
    amount_wei = w3.to_wei(amount_eth, 'ether')
```

```
    nonce = w3.eth.get_transaction_count(selected_account)
```

```
    txn = contract.functions.addSponsor().build_transaction({
```

```
        'chainId': w3.eth.chain_id,
```

```
        'gas': 300000,
```

```
        'gasPrice': w3.to_wei('1', 'gwei'),
```

```
        'nonce': nonce,
```

```
        'value': amount_wei
```

```
    })
```

```
    tx_receipt = build_and_send_transaction(txn)
```

```
print(f"Спонсора з внеском {amount_eth} ETH ({amount_wei} wei)
додано")
```

```
def add_project(name, budget_eth):
    global selected_account, private_key
    budget_wei = w3.to_wei(budget_eth, 'ether')
    nonce = w3.eth.get_transaction_count(selected_account)
    txn = contract.functions.addProject(name, budget_wei,
selected_account).build_transaction({
        'chainId': w3.eth.chain_id,
        'gas': 300000,
        'gasPrice': w3.to_wei('1', 'gwei'),
        'nonce': nonce
    })
    tx_receipt = build_and_send_transaction(txn)
    print(f"Проект '{name}' з бюджетом {budget_eth} ETH ({budget_wei}
wei) додано")
```

```
def vote(project_index):
    global selected_account, private_key
    try:
        nonce = w3.eth.get_transaction_count(selected_account)
        txn = contract.functions.vote(project_index).build_transaction({
            'chainId': w3.eth.chain_id,
            'gas': 300000,
            'gasPrice': w3.to_wei('1', 'gwei'),
            'nonce': nonce
        })
        tx_receipt = build_and_send_transaction(txn)
        print(f"Голосування за проект #{project_index} успішне")
    except Exception as e:
        if "You have already voted" in str(e):
            print("Ви вже проголосували раніше з цього гаманця.")
        else:
            print(f"Помилка при голосуванні: {e}")
```

```

def end_voting():
    global selected_account, private_key

    if not selected_account:
        return "Необхідно обрати гаманець перед завершенням голосування."

    try:
        tx = contract.functions.endVoting().build_transaction({
            'from': selected_account,
            'chainId': w3.eth.chain_id,
            'gas': 300000,
            'gasPrice': w3.to_wei('1', 'gwei'),
            'nonce': w3.eth.get_transaction_count(selected_account)
        })
        signed_tx = w3.eth.account.sign_transaction(tx, private_key)
        tx_hash = w3.eth.send_raw_transaction(signed_tx.rawTransaction)
        tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)

        winners = []
        for i in range(contract.functions.projectCount().call()):
            project = contract.functions.projects(i).call()
            if project[4]:
                name = project[0]
                budget = w3.from_wei(project[1], 'ether')
                votes = project[2]
                winners.append({"name": name, "budget": budget, "votes":
votes})

        return "Голосування завершено успішно.", winners

    except Exception as e:
        return f"Помилка при завершенні голосування: {e}", None

def withdraw_funds():
    global selected_account, private_key

    try:

```

```

user_project_status = check_user_project()

if user_project_status is None:
    return "Ви не зареєстрували проект"

elif not user_project_status:
    return "Ваш проект не перемир"


txn = contract.functions.withdrawFunds().build_transaction({
    'from': selected_account,
    'chainId': w3.eth.chain_id,
    'gas': 300000,
    'gasPrice': w3.to_wei('1', 'gwei'),
    'nonce': w3.eth.get_transaction_count(selected_account)
})

signed_txn = w3.eth.account.sign_transaction(txn, private_key)
tx_hash = w3.eth.send_raw_transaction(signed_txn.rawTransaction)
tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)


return "Кошти успішно виведено."


except Exception as e:
    if "Funds already withdrawn" in str(e):
        return "Кошти вже було виведено."

    if "No funds left to withdraw" in str(e):
        return "Немає коштів для переведення."

    return f"Помилка при переведенні коштів: {e}"


def get_balance(address):
    balance_wei = w3.eth.get_balance(address)
    balance_eth = w3.from_wei(balance_wei, 'ether')
    return balance_eth


def check_user_project():
    global selected_account

    try:
        for i in range(contract.functions.projectCount().call()):
            project = contract.functions.projects(i).call()
            if project[4] and project[3] == selected_account:

```

```

        return True # Користувач є власником проекту, що переміг

    return False # Користувач не є власником проекту, що переміг
except Exception as e:
    print(f"Помилка при перевірці проекту користувача: {e}")
    return False # Користувач не зареєстрував проект чи виникла
ПОМИЛКА

@app.route('/', methods=['GET', 'POST'])
def login():
    global selected_account, private_key
    if request.method == 'POST':
        account = request.form['account']
        private_key = request.form['private_key']
        try:
            account_obj = w3.eth.account.from_key(private_key)
            if account_obj.address == account:
                selected_account = account_obj.address
                session['account'] = account
                return redirect(url_for('menu'))
            else:
                return render_template('login.html', error='Невірний
приватний ключ')
        except Exception as e:
            return render_template('login.html', error=str(e))
    return render_template('login.html')

@app.route('/menu', methods=['GET'])
def menu():
    if 'account' not in session:
        return redirect(url_for('login'))
    account = session['account']
    balance = get_balance(account)
    return render_template('menu.html', account=account, balance=balance)

```

```

@app.route('/sponsor', methods=['GET', 'POST'])
def sponsor():
    if 'account' not in session:
        return redirect(url_for('login'))
    account = session['account']
    balance = get_balance(account)
    if request.method == 'POST':
        amount_eth = float(request.form['amount'])
        add_sponsor(amount_eth)
        return redirect(url_for('menu'))
    return render_template('sponsor.html', account=account,
balance=balance)


@app.route('/add_project', methods=['GET', 'POST'])
def add_project_route():
    if 'account' not in session:
        return redirect(url_for('login'))
    if request.method == 'POST':
        name = request.form['name']
        budget_eth = float(request.form['budget'])
        add_project(name, budget_eth)
        return redirect(url_for('menu'))
    account = session['account']
    balance = get_balance(account)
    return render_template('add_project.html', account=account,
balance=balance)


@app.route('/projects', methods=['GET'])
def projects():
    if 'account' not in session:
        return redirect(url_for('login'))

    projects = []
    project_count = contract.functions.projectCount().call()
    for i in range(project_count):
        project = contract.functions.projects(i).call()
        project_info = {

```

```

        'name': project[0],
        'budget': w3.from_wei(project[1], 'ether'),
        'votes': project[2],
        'owner': project[3],
        'is_winner': project[4]
    }
    projects.append(project_info)

    account = session['account']
    balance = get_balance(account)
    return render_template('projects.html', account=account,
                           balance=balance, projects=projects)

@app.route('/vote', methods=['GET', 'POST'])
def vote_route():
    if 'account' not in session:
        return redirect(url_for('login'))

    account = session['account']
    balance = get_balance(account)

    projects = []
    project_count = contract.functions.projectCount().call()
    for i in range(project_count):
        project = contract.functions.projects(i).call()
        projects.append(project)

    error_message = None
    sponsor_budget = contract.functions.sponsorBudgets(account).call()
    remaining_budget =
contract.functions.remainingBudgets(account).call()

    if request.method == 'POST':
        vote_values = []
        for i in range(project_count):
            wei_amount_str = request.form.get(f'wei_amount_{i}')
            if wei_amount_str:
                vote_values.append(int(wei_amount_str))

```

[illegible]

```

        return render_template('vote.html', account=account, balance=balance,
                                projects=projects, sponsor_budget=sponsor_budget,
                                remaining_budget=remaining_budget, error_message=error_message)

```

```

@app.route('/end_voting', methods=['GET', 'POST'])

```

```

def end_voting_route():

```

```

    if 'account' not in session:

```

```

        return redirect(url_for('login'))

```

```

    if request.method == 'POST':

```

```

        message, winners = end_voting()

```

```

        return render_template('end_voting.html', message=message,
                                winners=winners)

```

```

        account = session['account']

```

```

        balance = get_balance(account)

```

```

        return render_template('end_voting.html', account=account,
                                balance=balance)

```

```

@app.route('/withdraw_funds', methods=['GET', 'POST'])

```

```

def withdraw_funds_route():

```

```

    if 'account' not in session:

```

```

        return redirect(url_for('login'))

```

```

    if request.method == 'POST':

```

```

        message = withdraw_funds()

```

```

        return render_template('withdraw_funds.html', message=message)

```

```

        account = session['account']

```

```

        balance = get_balance(account)

```

```

        return render_template('withdraw_funds.html', account=account,
                                balance=balance)

```

```

@app.route('/logout', methods=['GET'])

```

```

def logout():

```

```

    session.pop('account', None)

    selected_account = None

    private_key = None

    return redirect(url_for('login'))

```

```

if __name__ == '__main__':
    app.run(debug=True)

```

## Лістинг файлу веб-інтерфейсу «login.html»

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>CryptoVote</title>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/meyer-
reset/2.0/reset.min.css">
    <link rel='stylesheet prefetch'
href='http://maxcdn.bootstrapcdn.com/font-awesome/4.1.0/css/font-
awesome.min.css'>
    <link rel="stylesheet" href="static/style_login.css">
</head>
<body>
    <div class="header_my">
        <h1>CryptoVote</h1>
    </div>

    <!-- ICONS -->
    <svg id="svg-source" height="0" version="1.1"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" style="position: absolute">
        <g id="man-people-user" data-iconmelon="Streamline Icon
Set:de32eb2621491c1a881a9fe846236da1">
            <g id="Expanded">
                <g>
                    <g>
                        <path d="M16.028,20c-4.764,0-8.639-4.486-8.639-10s3.875-
10,8.639-10c4.763,0,8.638,4.486,8.638,10s20.791,20,16.028,20z

```

```
M16.028,1.333C12,1.333,8.722,5.221,8.722,10s3.277,8.667,7.306,8.667c4.029
,0,7.306-3.888,7.306-8.667S20.057,1.333,16.028,1.333z"></path>
```

```
</g>
```

```
<g>
```

```
<path d="M31.988,32H0.012v-4.515c0-1.967,1.245-3.733,3.097-
4.39518.224-2.266v-2.77h1.333v3.785L3.51,24.361c-1.275,0.458-2.165,1.72-
2.165,3.124v3.182h29.309v-3.182c0-1.404-0.889-2.666-2.213-3.1391-9.107-
2.506v-
3.758h1.332v2.74218.178,2.251c1.9,0.677,3.145,2.442,3.145,4.409V32z"></pa
th>
```

```
</g>
```

```
<g>
```

```
<path d="M21.865,8.812c-0.045,0-0.09-0.001-0.137-0.003c-1.5-
0.055-3.25-1.004-4.361-2.287C16.59,7.513,15.48,8.15,14.106,8.383c-
2.403,0.413-5.152-0.51-5.988-1.32110.928-
0.957c0.403,0.391,2.69,1.329,4.836,0.964c1.332-0.226,2.292-0.911,2.854-
2.03410.558-
1.11410.617,1.082c0.738,1.292,2.508,2.425,3.867,2.475c0.604,0.016,1.033-
0.165,1.307-
0.57111.105,0.745C23.686,8.403,22.863,8.812,21.865,8.812z"></path>
```

```
</g>
```

```
</g>
```

```
</g>
```

```
</g>
```

```
<g id="lock-locker" data-iconmelon="Streamline Icon
Set:5d43c6f45cdbecfd5b8a12bc9e5dd33c">
```

```
<g id="Expanded">
```

```
<g>
```

```
<circle cx="16" cy="20" r="1.333"></circle>
```

```
</g>
```

```
<g>
```

```
<path d="M16,25.333c-0.369,0-0.667-0.298-0.667-0.666v-
4C15.333,20.298,15.631,20,16,20s0.667,0.298,0.667,0.667v4C16.667,25.035,1
6.369,25.333,16,25.333z"></path>
```

```
</g>
```

```
<g>
```

```
<path d="M28,32H4V12h24V32z
M5.333,30.667h21.333V13.333H5.333V30.667z"></path>
```

```
</g>
```

```
<g>
```

```

        <path d="M24,12.667h-1.333V8c0-3.676-2.991-6.667-6.667-6.667c-
3.676,0-6.667,2.99-6.667,6.667v4.667H8V8c0-4.412,3.588-8,8-
8c4.411,0,8,3.588,8,8V12.667z"></path>
    </g>
</g>
</g>
</svg>
<!-- ICONS -->

<div class="wrapper">
    <div class="header">
        <h3 class="sign-in">Вхід до акаунту</h3>
    </div>
    <div class="clear"></div>
    <form action="{{ url_for('login') }}" method="post">
        <div>
            <label class="user">
                <svg viewBox="0 0 32 32">
                    <g filter="">
                        <use xlink:href="#man-people-user"></use>
                    </g>
                </svg>
            </label>
            <input class="user-input" type="text" name="account"
placeholder="Адреса гаманця" required />
        </div>
        <div>
            <label class="lock">
                <svg viewBox="0 0 32 32">
                    <g filter="">
                        <use xlink:href="#lock-locker"></use>
                    </g>
                </svg>
            </label>
            <input type="password" name="private_key" placeholder="Приватний
ключ" required />
        </div>
        <div>
            <input type="submit" value="Увійти" />
        </div>
    </form>
</div>

```

```

    </form>

</div>

<script
src='http://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js'></
script>
</body>

<footer>
  <div class="footer-container">
    <div class="left-col">
      <p>© 2024 Попова Анна</p>
    </div>
    <div class="center-col">
      
    </div>
    <div class="right-col">
      <p>Сайт зроблено під час написання дипломної роботи у</p>
      <p>Донецькому національному технічному університеті</p>
    </div>
  </div>
</footer>
</html>

```

### Лістинг файлу веб-інтерфейсу «menu.html»

```

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>CryptoVote</title>
  <link rel="stylesheet" href="static/styles.css">
</head>
<body>
  <div class="header">
    <h1>CryptoVote</h1>
  </div>
  <div class="menu">
    <div class="menu-left">

```

```

        <p>Акаунт: {{ account }}</p>
    </div>

    <p>Баланс: {{ balance }} ETH</p>
</div>

<div class="nav-buttons">
    <form action="{{ url_for('projects') }}" method="get">
        <button type="submit">Проекти</button>
    </form>

    <form action="{{ url_for('vote_route') }}" method="get">
        <button type="submit">Голосування</button>
    </form>

    <form action="{{ url_for('menu') }}" method="get">
        <button type="submit">Меню</button>
    </form>
</div>

<div class="content">
    <div class="actions">
        <form action="{{ url_for('sponsor') }}" method="get">
            <button type="submit">Стати спонсором проєктів</button>
        </form>

        <form action="{{ url_for('add_project_route') }}"
method="get">
            <button type="submit">Зареєструвати проєкт</button>
        </form>

        <form action="{{ url_for('vote_route') }}" method="get">
            <button type="submit">Голосувати</button>
        </form>

        <form action="{{ url_for('end_voting_route') }}"
method="get">
            <button type="submit">Завершити голосування</button>
        </form>

        <form action="{{ url_for('withdraw_funds_route') }}"
method="get">
            <button type="submit">Отримати гроші</button>
        </form>
    </div>
</div>

<div class="logout">

```

```

        <a href="{{ url_for('logout') }}">Вийти з акаунту</a>
    </div>

<footer>
    <div class="footer-container">
        <div class="left-col">
            <p>© 2024 Попова Анна</p>
        </div>
        <div class="center-col">
            
        </div>
        <div class="right-col">
            <p>Сайт зроблено під час написання дипломної роботи у</p>
            <p>Донецькому національному технічному університеті</p>
        </div>
    </div>
</footer>
</body>
</html>

```

### Лістинг файлу веб-інтерфейсу «add\_project.html»

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>CryptoVote</title>
    <link rel="stylesheet" href="static/styles.css">
</head>
<body>
    <div class="header">
        <h1>CryptoVote</h1>
    </div>
    <div class="menu">
        <div class="menu-left">
            <p>Акаунт: {{ account }}</p>
        </div>
        <p>Баланс: {{ balance }} ETH</p>
    </div>

```

```

</div>

<div class="nav-buttons">
    <form action="{{ url_for('projects') }}" method="get">
        <button type="submit">Проекти</button>
    </form>
    <form action="{{ url_for('vote_route') }}" method="get">
        <button type="submit">Голосування</button>
    </form>
    <form action="{{ url_for('menu') }}" method="get">
        <button type="submit">Меню</button>
    </form>
</div>

<div class="content">
    <h1>Додати проект</h1>
    <form method="post">
        <label for="name">Назва проекту:</label>
        <input type="text" id="name" name="name" required>
        <br>
        <label for="budget">Бюджет проекту (в ETH):</label>
        <input type="number" id="budget" name="budget" step="0.01"
required>
        <br>
        <input type="submit" value="Додати проект">
    </form>
</div>

<div class="footer">
    <div class="footer-container">
        <div class="left-col">
            <p>© 2024 Попова Анна</p>
        </div>
        <div class="center-col">
            
        </div>
        <div class="right-col">
            <p>Сайт зроблено під час написання дипломної роботи у</p>
            <p>Донецькому національному технічному університеті</p>
        </div>
    </div>
</div>

```

```
</body>
</html>
```

### Лістинг файлу веб-інтерфейсу «end\_voting.html»

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>CryptoVote</title>
  <link rel="stylesheet" href="static/styles.css">
</head>
<body>
  <div class="header">
    <h1>CryptoVote</h1>
  </div>
  <div class="menu">
    <div class="menu-left">
      <p>Акаунт: {{ account }}</p>
    </div>
    <p>Баланс: {{ balance }} ETH</p>
  </div>
  <div class="nav-buttons">
    <form action="{{ url_for('projects') }}" method="get">
      <button type="submit">Проекти</button>
    </form>
    <form action="{{ url_for('vote_route') }}" method="get">
      <button type="submit">Голосування</button>
    </form>
    <form action="{{ url_for('menu') }}" method="get">
      <button type="submit">Меню</button>
    </form>
  </div>
  <div class="content">
    <h1>Завершити голосування</h1>
    <form action="/end_voting" method="POST">
      <button type="submit">Завершити голосування</button>
```

```

</form>

<h2>Список переможців:</h2>
<ul>
    {% for winner in winners %}
        <li>{{ winner.name }} (Бюджет: {{ winner.budget }} ETH,
Голоси: {{ winner.votes }})</li>
    {% endfor %}
</ul>
</div>

<div class="footer">
    <div class="footer-container">
        <div class="left-col">
            <p>© 2024 Попова Анна</p>
        </div>
        <div class="center-col">
            
        </div>
        <div class="right-col">
            <p>Сайт зроблено під час написання дипломної роботи у</p>
            <p>Донецькому національному технічному університеті</p>
        </div>
    </div>
</div>
</body>
</html>

```

### Лістинг файлу веб-інтерфейсу «projects.html»

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Projects</title>
    <link rel="stylesheet" href="static/styles.css">
</style>
</head>

```

```

<body>

  <div class="header">
    <h1>CryptoVote</h1>
  </div>

  <div class="menu">
    <div class="menu-left">
      <p>Акаунт: {{ account }}</p>
    </div>
    <p>Баланс: {{ balance }} ETH</p>
  </div>

  <div class="nav-buttons">
    <form action="{{ url_for('projects') }}" method="get">
      <button type="submit">Проекти</button>
    </form>
    <form action="{{ url_for('vote_route') }}" method="get">
      <button type="submit">Голосування</button>
    </form>
    <form action="{{ url_for('menu') }}" method="get">
      <button type="submit">Меню</button>
    </form>
  </div>

  <div class="content">
    <h1>Проекти</h1>
    <table border="1">
      <tr>
        <th>Назва проекту</th>
        <th>Бюджет (ETH)</th>
        <th>Голоси</th>
        <th>Власник</th>
        <th>Переможець</th>
      </tr>
      {% for project in projects %}
      <tr>
        <td>{{ project.name }}</td>
        <td>{{ project.budget }}</td>
        <td>{{ project.votes }}</td>
        <td>{{ project.owner }}</td>
        <td>{{ project.is_winner }}</td>
      </tr>
      {% endfor %}
    </table>
  </div>

```

```

        </table>
    </div>

    <div class="footer">
        <div class="footer-container">
            <div class="left-col">
                <p>© 2024 Попова Анна</p>
            </div>
            <div class="center-col">
                
            </div>
            <div class="right-col">
                <p>Сайт зроблено під час написання дипломної роботи у</p>
                <p>Донецькому національному технічному університеті</p>
            </div>
        </div>
    </div>
</body>
</html>

```

### Лістинг файлу веб-інтерфейсу «sponsor.html»

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>CryptoVote</title>
    <link rel="stylesheet" href="static/styles.css">
</head>
<body>
    <div class="header">
        <h1>CryptoVote</h1>
    </div>
    <div class="menu">
        <div class="menu-left">
            <p>Акаунт: {{ account }}</p>
        </div>
        <p>Баланс: {{ balance }} ETH</p>
    </div>

```

```

</div>

<div class="nav-buttons">
    <form action="{{ url_for('projects') }}" method="get">
        <button type="submit">Проекти</button>
    </form>

    <form action="{{ url_for('vote_route') }}" method="get">
        <button type="submit">Голосування</button>
    </form>

    <form action="{{ url_for('menu') }}" method="get">
        <button type="submit">Меню</button>
    </form>
</div>


<div class="content">
    <h1>Спонсувати проекти</h1>
    <form method="post">
        <label for="amount">Сума (в ETH):</label>
        <input type="number" id="amount" name="amount" step="0.01"
required>
        <br>
        <input type="submit" value="Спонсувати">
    </form>
</div>


<div class="footer">
    <div class="footer-container">
        <div class="left-col">
            <p>© 2024 Попова Анна</p>
        </div>
        <div class="center-col">
            
        </div>
        <div class="right-col">
            <p>Сайт зроблено під час написання дипломної роботи у</p>
            <p>Донецькому національному технічному університеті</p>
        </div>
    </div>
</div>
</body>
</html>

```

## Лістинг файлу веб-інтерфейсу «vote.html»

```

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>CryptoVote</title>
  <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
  <script>
    function convertEthToWei() {
      var ethAmount =
parseFloat(document.getElementById('ethAmount').value);
      var weiAmount = ethAmount * 10**18;
      document.getElementById('weiAmount').value = weiAmount;
    }
  </script>
</head>
<body>
  <div class="header">
    <h1>CryptoVote</h1>
  </div>
  <div class="menu">
    <div class="menu-left">
      <p>Акаунт: {{ account }}</p>
    </div>
    <p>Баланс: {{ balance }} ETH</p>
    <p>Ваш бюджет для голосування: {{ sponsor_budget / 10**18}}
ETH</p>
  </div>
  <div class="nav-buttons">
    <form action="{{ url_for('projects') }}" method="get">
      <button type="submit">Проекти</button>
    </form>
    <form action="{{ url_for('vote_route') }}" method="get">
      <button type="submit">Голосування</button>
    </form>
  </div>

```

```

</form>

<form action="{{ url_for('menu') }}" method="get">
    <button type="submit">Меню</button>
</form>
</div>

<div class="content">
    <h1>Розподіліть свій бюджет для голосування:</h1>
    {% if error_message %}
    <p class="error-message">{{ error_message }}</p>
    {% endif %}
    <form action="{{ url_for('vote_route') }}" method="POST">
    <ul class="projects-list">
        {% for i in range(projects|length) %}
        <li>
            <label>
                {{ projects[i][0] }}
            </label>
            <input type="number" id="ethAmount" name="eth_amount_{{ i
            }}" min="0" step="0.01" value="0" required onchange="convertEthToWei()">
            <input type="hidden" id="weiAmount" name="wei_amount_{{ i
            }}">
        </li>
        {% endfor %}
    </ul>
    <button type="submit">Голосувати</button>
</form>
</div>

<div class="footer">
    <div class="footer-container">
        <div class="left-col">
            <p>© 2024 Попова Анна</p>
        </div>
        <div class="center-col">
            
        </div>
        <div class="right-col">
            <p>Сайт зроблено під час написання дипломної роботи у</p>

```

```

        <p>Донецькому національному технічному університеті</p>
    </div>
</div>
</div>
</body>
</html>

```

### Лістинг файлу веб-інтерфейсу «withdraw\_funds.html»

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>CryptoVote</title>
    <link rel="stylesheet" href="static/styles.css">
</head>
<body>
    <div class="header">
        <h1>CryptoVote</h1>
    </div>
    <div class="menu">
        <div class="menu-left">
            <p>Акаунт: {{ account }}</p>
        </div>
        <p>Баланс: {{ balance }} ETH</p>
    </div>
    <div class="nav-buttons">
        <form action="{{ url_for('projects') }}" method="get">
            <button type="submit">Проекти</button>
        </form>
        <form action="{{ url_for('vote_route') }}" method="get">
            <button type="submit">Голосування</button>
        </form>
        <form action="{{ url_for('menu') }}" method="get">
            <button type="submit">Меню</button>
        </form>
    </div>

```

```

<div class="content">
    <h1>Виведення коштів</h1>
    <form action="/withdraw_funds" method="POST">
        <button type="submit">Отримати гроші</button>
    </form>
    {% if message %}
    <p>{{ message }}</p>
    {% endif %}
</div>

<div class="footer">
    <div class="footer-container">
        <div class="left-col">
            <p>© 2024 Попова Анна</p>
        </div>
        <div class="center-col">
            
        </div>
        <div class="right-col">
            <p>Сайт зроблено під час написання дипломної роботи у</p>
            <p>Донецькому національному технічному університеті</p>
        </div>
    </div>
</div>
</body>
</html>

```