МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПОЛІСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ, ОБЛІКУ ТА ФІНАНСІВ

Кафедра комп'ютерних технологій і моделювання систем

Кваліфікаційна робота
на правах рукопису

Передера Владислава Дмитровича

УДК 004.75:324:004.056

КВАЛІФІКАЦІЙНА РОБОТА

Розроблення децентралізованої системи електронного голосування на основі блокчейну

(тема роботи)

F3(122) Комп’ютерні науки

(шифр і назва спеціальності)

Подається на здобуття освітнього ступеня магістр

кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Передер В. Д.

(підпис, ініціали та прізвище здобувача вищої освіти)

Керівник роботи
Лапін Андрій Валерійович
(прізвище, ім’я, по батькові)
к. е. н. доцент
(науковий ступінь, вчене звання)

Житомир – 2025

Висновок кафедри комп'ютерних технологій і моделювання систем:
за результатами попереднього захисту: _____

Протокол засідання кафедри комп'ютерних технологій і моделювання систем
№ ____ від «_____» _____ 20__ р.

Завідувач кафедри комп'ютерних технологій і моделювання систем
к.п.н., доцент _____ М. О. Ковальчук
(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)
«_____» _____ 20___ р.

Результати захисту кваліфікаційної роботи

Здобувач вищої освіти Передер Владислав Дмитрович захистив
(прізвище ,ім'я, по батькові)
кваліфікаційну роботу з оцінкою:

сума балів за 100-бальною шкалою _____
за шкалою ECTS _____
за національною шкалою _____

Секретар ЕК
лаборант кафедри _____ В. В. Корольчук
(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)

АНОТАЦІЯ

Передер В. Д. – Кваліфікаційна робота на правах рукопису.

Кваліфікаційна робота на здобуття освітнього ступеня магістр за спеціальністю 122 – Комп'ютерні науки. – Поліський національний університет, Житомир, 2025.

Обсяг кваліфікаційної роботи: 42 сторінок, кількість рисунків: 13, кількість таблиць: 0, кількість додатків: 4, кількість використаних джерел: 33.

Ключові слова: блокчейн, децентралізована система, електронне голосування, смартконтракти.

У кваліфікаційній роботі розглянуто підходи до створення децентралізованих систем електронного голосування та обґрунтовано доцільність використання блокчейн-технологій як основи для підвищення прозорості й контрольованості виборчих процедур. Основну увагу зосереджено на моделі неанонімного голосування з попередньою верифікацією учасника поза блокчейном та подальшим включенням його адреси до whitelist, що дає змогу поєднати організаційний контроль із технічною незмінністю результатів.

Практичне значення роботи полягає у створенні навчально-дослідного прототипу, придатного для використання у внутрішніх голосуваннях закладів вищої освіти або інших організаційних спільнот із чітко визначеним колом учасників. Результати можуть бути використані як основа для подальшого розвитку системи з розширенням ролей доступу, покращенням механізмів ідентифікації та інтеграцією з інституційними сервісами.

SUMMARY

Pereder V. D. – Qualification thesis manuscript.

Qualification thesis submitted for the Master's degree in Specialty 122 – Computer Science. – Polissia National University, Zhytomyr, 2025.

Thesis length: 42 pages, number of figures: 13, number of tables: 0, number of appendices: 4, number of references: 33.

Keywords: blockchain, decentralized system, electronic voting, smart contracts.

The qualification thesis examines approaches to building decentralized electronic voting systems and substantiates the feasibility of using blockchain technology as a basis for increasing the transparency and controllability of electoral procedures. The main focus is on a non-anonymous voting model with prior participant verification performed off-chain and subsequent inclusion of the participant's address in a whitelist, which makes it possible to combine organizational control with the technical immutability of results.

The practical significance of the work lies in developing an educational and research prototype suitable for use in internal voting within higher education institutions or other organizational communities with a clearly defined group of participants. The results may serve as a basis for further development of the system through expanded access roles, improved identification mechanisms, and integration with institutional services.

ЗМІСТ

ВСТУП

Цифрова трансформація суспільства поступово змінює те, як організовуються управлінські та демократичні процеси. У цьому контексті електронне голосування сприймається як інструмент, що може зробити ухвалення рішень оперативнішим і зручнішим, а процедури - прозорішими. Водночас на практиці ключовими залишаються питання довіри до результатів, технічної стійкості, контролю доступу та можливості незалежної перевірки підсумків.

Актуальність теми зумовлена тим, що централізовані системи голосування часто залишають простір для сумнівів щодо цілісності даних і чесності підрахунку. Коли потрібно переконливо довести, що голоси не були підмінені, а результат не змінювався після завершення процедури, особливо цінними стають механізми, які зменшують вплив людського фактора та фіксують події у незмінному вигляді. Саме тому блокчейн-технології розглядаються як основа для систем, де записи зберігаються у розподіленому середовищі та можуть бути перевірені незалежно.

У цій роботі електронне голосування розглядається передусім у прикладному вимірі університетського середовища (вибори органів студентського самоврядування, внутрішні колективні рішення). Такий контекст зручний для створення дослідницького прототипу: він дозволяє зосередитися на ключових принципах - верифікації виборців, одноразовості голосу, прозорості журналу подій і перевірності результатів - без претензії на масштаб загальнодержавних виборів.

Обрана модель у межах дослідження є неанонімною: особа учасника верифікується поза блокчейном, після чого його адресу додають до whitelist. У смартконтракті при цьому зберігається лише блокчейн-адреса, тобто система забезпечує технічну ідентифікацію в межах протоколу, але не розкриває персональні дані в публічному реєстрі. Це є практичним компромісом для

сценарію, де організатор має підстави знати склад учасників, а сама процедура лишається прозорою та контрольованою.

Мета дослідження полягає у теоретичному обґрунтуванні та практичному проєктуванні прототипу децентралізованої системи електронного голосування на основі блокчейну для університетського середовища, із використанням смартконтрактів і whitelist-верифікації. Для досягнення мети передбачено: проаналізувати підходи до е-голосування та вимоги до перевірності й захисту результатів; обґрунтувати вибір блокчейну/смартконтрактів; спроєктувати архітектуру з ролями організатора й виборця та механізмами доступу; визначити смартконтрактну взаємодію (створення голосувань, керування whitelist, реєстрація голосів і підрахунок); описати модель безпеки; реалізувати й апробувати прототип у тестовому середовищі з демонстрацією клієнтського інтерфейсу та читанням подій блокчейну.

Об’єктом дослідження є процес організації та проведення електронного голосування у розподілених інформаційних системах, а предметом - методи й програмні засоби побудови децентралізованого прототипу на основі блокчейну (архітектура смартконтрактів і whitelist-верифікація). Методологічну основу становлять аналіз і узагальнення наукових джерел, системне проєктування та прототипування з практичним моделюванням взаємодії компонентів. Наукова новизна роботи полягає у формалізації університетської моделі голосування на смартконтрактах із whitelist-контролем доступу та в поєднанні компонентів «фабрики голосувань», контрактів голосування й клієнтського інтерфейсу з індексацією подій. Робота має теоретичну й практичну значущість як основа для подальшого розвитку та навчально-дослідних демонстрацій, а її структура охоплює теоретичний аналіз, проєктування архітектури та опис реалізації прототипу з підсумковими висновками й додатками.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ МОДЕЛЕЙ ЕЛЕКТРОННОГО ГОЛОСУВАННЯ ТА МОЖЛИВОСТЕЙ ЇХ РЕАЛІЗАЦІЇ НА ОСНОВІ БЛОКЧЕЙНУ

1.1 Поняття електронного голосування та його основні моделі

Електронне голосування сьогодні дедалі частіше розглядають не як разову технологічну новинку, а як закономірний етап розвитку демократичних і управлінських процедур у цифровому суспільстві. У найзагальнішому розумінні йдеться про процес подання, обробки та підрахунку голосів із використанням цифрових засобів, які частково або повністю замінюють паперові інструменти. Водночас у сучасних дослідженнях наголошується, що електронне голосування - це не лише «оцифрований бюлетень», а ширша зміна організаційної логіки ухвалення рішень. Технологія впливає на те, як фіксується волевиявлення, як регламентується процедура і як вибудовується довіра до підсумків у середовищі, де швидкість і прозорість стають такими ж важливими, як і формальна правильність процесу.

З практичної точки зору інтерес до електронного голосування зрозумілий. Традиційні формати часто потребують значних часових і організаційних ресурсів: підготовки фізичної інфраструктури, комплектації комісій, логістики бюлетенів, збору протоколів та ручного підрахунку. Цифрові системи дозволяють автоматизувати частину цих процедур, зробити результати більш оперативними та зменшити кількість технічних помилок, що виникають у процесі рутинної роботи. Однак у літературі підкреслюється, що ефект від електронного голосування не є гарантованим «за замовчуванням»: він залежить від архітектури системи, якості регламентів і зрозумілої для учасників логіки контролю та перевірки результатів [1].

У міжнародних стандартах е-голосування важливий акцент робиться на тому, що цифрова форма не повинна послаблювати демократичні принципи, які історично забезпечували легітимність голосування [4]. Електронні рішення мають зберігати сенс процедури: голосування повинно бути організованим, зрозумілим, стабільним і таким, що допускає внутрішній або зовнішній контроль. Саме тому сучасні підходи оцінюють е-голосування не лише за «функціональністю», а за тим, чи підсилює технологія довіру і чи не створює нових зон невизначеності для учасників процесу.

Найпоширеніший підхід до класифікації електронного голосування пов’язаний зі способом подання голосу [1]. Передусім виділяють локальне електронне голосування, коли волевиявлення здійснюється на дільниці або у визначеній фізичній точці за допомогою електронних терміналів. Ця модель зберігає традиційну організаційну рамку, але переводить частину операцій у цифровий формат. У багатьох дослідженнях її розглядають як відносно «м’який» шлях модернізації, що дозволяє впроваджувати нові інструменти без радикальної перебудови звичних процедур [1].

Іншою важливою моделлю є дистанційне інтернет-голосування, яке передбачає можливість подання голосу поза межами виборчої дільниці - через мережу Інтернет із використанням персонального пристрою. Сам підхід виглядає логічним продовженням загальної цифровізації суспільних процесів: він націлений на те, щоб зменшити залежність громадянина від фізичної присутності, розширити доступність процедури та зробити участь у голосуванні більш гнучкою за часом і місцем [4]. У практичному сенсі ця модель часто асоціюється з підвищенням зручності, особливо в ситуаціях, коли інституції прагнуть охопити ширше коло учасників або спростити організацію великої кількості голосувань у короткі терміни.

Водночас у наукових джерелах підкреслюється, що дистанційний формат передбачає складніший рівень відповідальності за організацію процесу та технологічні гарантії цілісності результатів [34]. Тому в реальних сценаріях ця модель особливо потребує зваженого підходу до вибору технологічної платформи, прозорого регламенту проведення процедури та механізмів внутрішнього контролю. У цьому сенсі дистанційне голосування не стільки «заміщує» інші моделі, скільки доповнює їх, формуючи ширшу палітру інструментів для різних рівнів і форматів ухвалення рішень.

Окремо варто виділити **організаційні та корпоративні електронні голосування**, що застосовуються у компаніях, університетах, професійних об’єднаннях та інших структурах. На практиці саме цей сегмент є одним із найбільш “живих” і динамічних, адже потреба у швидкому та прозорому ухваленні рішень виникає тут регулярно: від виборів органів самоврядування й затвердження внутрішніх регламентів до бюджетних, кадрових або стратегічних рішень. Для багатьох організацій електронне голосування в такому форматі - це насамперед інструмент ефективного управління, який дозволяє зменшити часові витрати на збори, підвищити дисципліну процедур і зробити результати більш наочними для учасників процесу.

Важливо й те, що в межах організаційних голосувань правові рамки зазвичай є гнучкішими, а порядок ухвалення рішень визначається статутами, положеннями або внутрішніми політиками. Саме тому ця сфера природно сприймається як зручний простір для технологічних експериментів: тут легше апробувати нові підходи до збереження та аудиту результатів, порівняти різні моделі електронних платформ і оцінити, наскільки технологія справді покращує якість управлінського рішення. У багатьох випадках цінність таких систем полягає не лише у швидкості, а й у тому, що процес ухвалення рішень стає більш прозорим, а роль одного адміністратора чи централізованої бази

даних - менш критичною. Саме з цієї причини інтерес до децентралізованих технологій у даному контексті виглядає цілком закономірним.

Загалом, моделі електронного голосування відрізняються не лише технічними особливостями, а й організаційною логікою їх застосування [4]. Саме в таких середовищах стає особливо помітним ефект від прозорої фіксації рішень, можливості швидкого формування підсумків і зниження залежності від централізованих адміністративних механізмів.

1.2 Правові та організаційні можливості впровадження електронного голосування в Україні

Оскільки тема роботи орієнтована на створення прототипу, здатного працювати в національному середовищі, важливо зрозуміти, у яких сферах цифрові механізми волевиявлення вже є допустимими, а де для цього потрібні додаткові нормативні кроки.

1.2.1 Чинне законодавство як рамка для цифрового волевиявлення

Правові умови використання цифрових інструментів у голосуваннях в Україні можна умовно поділити на два взаємозалежні блоки. Перший охоплює процедурні норми - усе, що визначає порядок проведення виборів, правила місцевого самоврядування та механізми ухвалення рішень у різних типах організацій. Базовим кодифікованим актом у виборчій сфері є Виборчий кодекс України [3], а спеціальним законом щодо парламентських виборів - Закон України «Про вибори народних депутатів України» [4].

Другий блок формує технологічну основу довіри: це норми про електронну ідентифікацію, електронні підписи, захист персональних даних і фіксацію юридично значущих дій у цифровому середовищі. Ключовим у цьому сегменті є Закон України «Про електронну ідентифікацію та електронні

довірчі послуги» [5], який визначає правові та організаційні засади електронної ідентифікації та використання довірчих послуг.

У сфері політичних виборів Україна й надалі дотримується традиційної моделі голосування. Така позиція виглядає логічною: виборчі процеси мають найвищу суспільну ціну, а ризики кібервтручання чи процедурних маніпуляцій здатні підірвати довіру до легітимності результатів. Саме тому говорити про швидке, пряме впровадження блокчейн-голосування на загальнодержавному рівні сьогодні було б передчасно - у цій площині потрібен поступовий нормативний і організаційний перехід із урахуванням норм виборчого законодавства [3 - 4].

Разом із тим розвиток правового поля електронної довіри вже створив міцний фундамент для локальних або корпоративних цифрових процедур. Окрім базового профільного закону, важливу роль відіграють підзаконні акти, що регламентують функціонування національної е-ідентифікаційної інфраструктури, зокрема Положення про інтегровану систему електронної ідентифікації [6].

Практичними інструментами, що можуть використовуватись як легальний рівень автентифікації у майбутніх моделях е-голосування, є державні та галузеві рішення: BankID НБУ (правові засади - Положення НБУ про Систему BankID [7]) та MobileID як послуга е-ідентифікації й КЕП, що застосовується в межах загального режиму електронної ідентифікації і підтримується державною екосистемою е-послуг.

Для цієї роботи це принциповий акцент: блокчейн-рішення повинне не підміняти правові механізми ідентифікації, а коректно інтегруватися з ними, використовуючи їх як вхідний рівень довіри для подальшої фіксації, збереження й аудиту результатів у децентралізованому реєстрі. Такий підхід узгоджується з логікою чинного регулювання електронних довірчих послуг.

Отже, чинна правова база не відкриває універсального «вікна» для політичного онлайн-голосування, проте цілком дозволяє розглядати децентралізовані моделі як реалістичний інструмент для тих сегментів, де ризики нижчі, а регламенти гнучкіші.

1.2.2 Сфери, де електронне голосування вже є організаційно і правово реалістичним

Найбільш придатними для практичного застосування децентралізованих технологій є ті типи голосувань, де правила ухвалення рішень визначаються статутами, внутрішніми положеннями або спеціальним корпоративним і організаційним законодавством.

Передусім це стосується корпоративних голосувань. Для товариств з обмеженою та додатковою відповідальністю правовою основою є Закон України «Про товариства з обмеженою та додатковою відповідальністю» [8]. Для акціонерних товариств - Закон України «Про акціонерні товариства» [9], який передбачає сучасні формати участі акціонерів у загальних зборах і допускає використання електронних механізмів голосування у визначених законом формах.

Схожа логіка актуальна і для об'єднання співвласників багатоквартирного будинку. Базовим нормативним актом тут є Закон України «Про об'єднання співвласників багатоквартирного будинку» [10]. Окремі зміни останніх років спрямовані на спрощення управління багатоквартирними будинками та допускають ширше використання електронних документів у процедурах прийняття рішень співвласниками, що створює додаткові можливості для впровадження цифрових і, зокрема, блокчейн-орієнтованих рішень у цій сфері [11].

Важливим напрямом є студентські та університетські голосування. Правову основу участі студентів у формуванні органів самоврядування встановлює Закон України «Про вищу освіту», зокрема норми про студентське самоврядування та порядок виборів його органів [12]. Це дозволяє закладам вищої освіти деталізувати процедури на рівні внутрішніх положень, що робить університетське середовище організаційно придатним для пілотування сучасних цифрових систем голосування.

Не менш перспективними є голосування в громадських об’єднаннях - професійних асоціаціях, спілках, громадських організаціях. Загальні організаційні засади їх утворення та діяльності визначає Закон України «Про громадські об'єднання» [13], а конкретні механізми волевиявлення зазвичай деталізуються у статутах. Це створює легальний простір для застосування технологій, що підсилюють прозорість і перевірність внутрішніх рішень, включно з блокчейном.

Окрему групу становлять консультаційні голосування та опитування органів місцевого самоврядування. Базові форми безпосередньої участі жителів, зокрема громадські слухання, закріплені в Законі України «Про місцеве самоврядування в Україні» [14]. Додатково розвиток електронних інструментів участі підтримується через інститут електронних петицій, який закріплений у Законі України «Про звернення громадян» та змінах до нього [15]. Такі механізми не є виборами у класичному сенсі, однак формують практичне середовище для розгортання технологічних моделей е-участі з підвищеною прозорістю результатів.

1.3 Блокчейн-технологія як основа для створення децентралізованої системи електронного голосування

Після аналізу правових і організаційних передумов електронного голосування в Україні доцільно перейти до вибору технологічної основи

майбутнього рішення. У межах цього підрозділу важливо не «ідеалізувати» блокчейн, а показати його як технічне ядро, що може підсилити прозорість і перевірність процедури лише за умови коректного процедурного дизайну та збереження ролі правових і організаційних механізмів контролю. Саме тому блокчейн у голосуванні слід трактувати як інструмент фіксації та перевірки дій, а не як заміну регламентів чи відповідальних інституцій [16].

1.3.1. Властивості блокчейну, важливі для систем голосування

Ключова перевага блокчейну для голосувань полягає в незмінності записів: події та результати фіксуються так, що їх практично неможливо непомітно підмінити «заднім числом», а спроби втручання стають виявними на рівні мережевого консенсусу. Це створює технічну основу довіри там, де критично довести цілісність підсумків порівняно з централізованими реєстрами.

Другою важливою властивістю є прозорість і можливість незалежного аудиту: за правильно визначеної моделі доступу учасники або аудитори можуть перевіряти коректність процесу за даними транзакцій і подій, а не покладатися на заяви адміністратора. Водночас децентралізація не є автоматичною гарантією безпеки - без коректного протоколу та контролю ризиків технічні переваги можуть бути нівельовані [16].

Нарешті, блокчейн підтримує перехід від довіри до інституції до довіри до перевірної процедури: результати можуть бути верифіковані технічно, а не лише підтверджені організатором. Такий підхід (коли підсумки потенційно може перевірити будь-хто за правилами системи) розглядають як перспективний напрям застосування блокчейну у голосуванні [17].

1.3.2. Обґрунтування використання блокчейну в українському контексті

В українському контексті питання довіри до цифрових процедур є особливо чутливим, тому блокчейн у межах цієї роботи доцільно розглядати як модель, де довіра спирається на перевірність дій і незмінність даних, а не лише на «статус адміністратора» чи внутрішні гарантії платформи. Це концептуально зменшує підозри щодо можливих маніпуляцій принаймні на рівні фіксації та контролю ключових подій.

Практично важливо й те, що в Україні існують середовища, де електронні механізми ухвалення рішень є більш реалістичними з організаційної та правової точок зору - зокрема корпоративні й університетські голосування. Такі сценарії можуть слугувати «полігонами» для апробації підходу без перенесення його одразу на рівень загальнодержавних виборів.

1.3.3. Реалізації системи у вигляді смарт-контрактів

Реалізація блокчейн-голосування через смарт-контракти полягає у формалізації правил процедури у коді, який виконується однаково для всіх учасників. У такій моделі ключові події фіксуються як транзакції або зміни стану смарт-контракту, що забезпечує прозору історію процесу й зменшує можливість «непомітних правок» результатів.

Критичним елементом є контроль права на участь: для багатьох практичних сценаріїв доцільною є модель, де особа верифікується поза блокчейном, а право голосу в системі задається включенням адреси до whitelist. Смарт-контракт перевіряє дозвіл адреси та факт попереднього голосування, що детерміновано блокує повторні спроби без «людського втручання».

Підсумки можуть формуватися автоматично після завершення голосування або залишатися як публічно перевірний набір даних, який можна незалежно перерахувати. Водночас блокчейн не скасовує потреби в

кіберзахисті, процедурному контролі та якісному правовому регламенті: безпековий ефект з’являється лише за умови коректної загальної моделі системи, а не «сам по собі» від факту використання розподіленого реєстру [19].

Висновки до першого розділу

У першому розділі проаналізовано основні моделі електронного голосування та їхні організаційні й технологічні особливості. Показано, що е-голосування варто розглядати не лише як «цифровий бюлетень», а як ширшу зміну підходів до ухвалення рішень у цифровому середовищі. Розмежування локального, дистанційного та організаційно-корпоративного голосування дозволило чіткіше окреслити відмінності у ризиках і вимогах до довіри.

Окремо розглянуто правові й організаційні передумови впровадження електронного голосування в Україні. Обґрунтовано, що для загальнодержавних політичних виборів потрібні значно жорсткіші гарантії безпеки й подальший нормативний розвиток, тоді як у менш ризикових сферах вже існує практичний простір для легітимного використання електронних механізмів волевиявлення.

На цій основі сформульовано концепцію майбутньої системи на смартконтрактах: фіксація подій у розподіленому реєстрі, використання попередньо верифікованого списку учасників, алгоритмічне запобігання повторному голосуванню та отримання публічно перевірних результатів. Таким чином, перший розділ сформував теоретико-методологічну та нормативно-організаційну базу дослідження і логічно підводить до другого розділу, де акцент переходить на системне проєктування архітектури прототипу та деталізацію механізмів реалізації.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ДЕЦЕНТРАЛІЗОВАНОЇ СИСТЕМИ ЕЛЕКТРОННОГО ГОЛОСУВАННЯ

2.1. Обґрунтування вибору підходу до побудови системи

Після розгляду моделей та загального контексту електронного голосування природно постає практичне питання: який підхід є водночас реалістичним для впровадження, зрозумілим для демонстрації в межах магістерської роботи та достатньо надійним за логікою довіри. Саме тому в цьому підпункті фіксується не «набір технологій», а послідовність концептуальних рішень, які стануть основою для подальшого проєктування архітектури, алгоритмів і реалізації прототипу.

У межах обраної концепції ключовими є п’ять опорних тез: доцільність децентралізованої моделі, вибір Ethereum-подібної платформи, перенесення правил процедури у смартконтракти, фокус на університетському сценарії як найбільш керованому та реалістичному середовищі апробації, а також застосування whitelist-механізму як практичного способу контролю доступу у прототипі. Такий «скелет» одразу задає рамку для всіх наступних технічних рішень і не дозволяє розпорошитися на деталі, які не підсилюють головну ідею системи.

Передусім, децентралізація тут важлива як спосіб зняти базову проблему довіри. У голосуванні критично, щоб учасники вірили не «адміністратору» чи «серверу», а правилам процедури. У централізованих рішеннях навіть за сумлінної організації завжди лишається сумнів щодо можливості непомітного втручання, тоді як децентралізований підхід переносить акцент на публічно визначені правила, які виконуються однаково для всіх.

Як технологічна основа для цього підходу, блокчейн дає дві принципові переваги: прозорість із можливістю незалежного аудиту та

зменшення залежності від одного адміністративного центру. Якщо модель доступу й оприлюднення даних визначена коректно, перевірка підсумків може бути доступною не лише організаторам, а й зовнішнім спостерігачам або самим учасникам. Водночас важливо підкреслити: децентралізація не є «автоматичною гарантією безпеки» - вона працює лише разом із продуманим процедурним дизайном і контролем ризиків.

Далі, Ethereum-подібна архітектура обирається через зрілість екосистеми та практичну відтворюваність [20]. У цьому середовищі є стандарти, бібліотеки, інструменти тестування й базова культура аудиту, що критично для академічного прототипу. Окрема цінність - гнучкість середовищ (тестові мережі та інші сумісні рішення), яка дозволяє демонструвати систему без надмірних витрат, зберігаючи сумісність із загальною логікою децентралізованих застосунків.

Смартконтракти [21] у цій роботі виступають не «частиною блокчейну», а механізмом формалізації правил голосування. Їхня роль полягає в тому, щоб «вшити» регламент у програмну логіку: тоді система не просто декларує правила, а фактично виконує їх у стабільний та однаковий спосіб для всіх учасників. Для дослідницького формату це важливо ще й тим, що дозволяє чітко пояснити правила процедури й показати, як саме вони реалізуються у децентралізованому середовищі.

Практичним контекстом апробації обрано університетське голосування, оскільки це реалістичне середовище з відносно керованими ризиками: існує зрозумілий перелік учасників, є внутрішні правила та природна потреба в процедурах ухвалення рішень (зокрема в студентському самоврядуванні). Це дозволяє зосередитися на ключових принципах (одноразовість голосу, прозорість, перевірюваність), не змішуючи прототип із надвисокими ризиками та складністю загальнодержавних виборів.

Нарешті, whitelist-верифікація обґрунтовується як найбільш практичний механізм для неанонімної моделі [22] в межах прототипу: ідентифікація відбувається поза блокчейном (на рівні інституції), а в смартконтракті зберігається лише адреса як маркер права участі. Такий підхід не намагається «перенести персональні дані в блокчейн», але зберігає баланс між організаційним контролем і технічною перевірюваністю дій.

Отже, обрана концепція є збалансованою: вона одночасно показує, як блокчейн-інструменти підсилюють довіру та прозорість, і при цьому залишається реалістичною для навчально-наукового прототипу. Саме це створює логічний перехід до подальшого опису архітектури системи та її компонентів у наступних підпунктах розділу.

2.2 Проєктування архітектури децентралізованої системи електронного голосування

Після обґрунтування загального підходу варто перейти до того, як саме обрана концепція складається у цілісну систему. Тут важливо показати не стільки технічні дрібниці, скільки структуру рішення: які компоненти ми виділяємо, де проходять межі відповідальності між університетом і блокчейном, як взаємодіють користувачі, інтерфейс і смартконтракти.

2.2.1 Загальна архітектурна модель системи

На рис. 2.1 наведено загальну архітектурну модель децентралізованої системи університетського електронного голосування, побудовану на основі Ethereum-подібної блокчейн-логіки та смартконтрактів. Схема демонструє базові компоненти та їхню взаємодію у найпростішому й водночас найбільш показовому вигляді, коли критично важливі дії голосування фіксуються у блокчейні, а користувацька взаємодія відбувається через веб-інтерфейс. Такий рівень узагальнення є типовим для проєктного підрозділу, оскільки він

дозволяє пояснити логіку системи без переходу до технічних деталей реалізації.

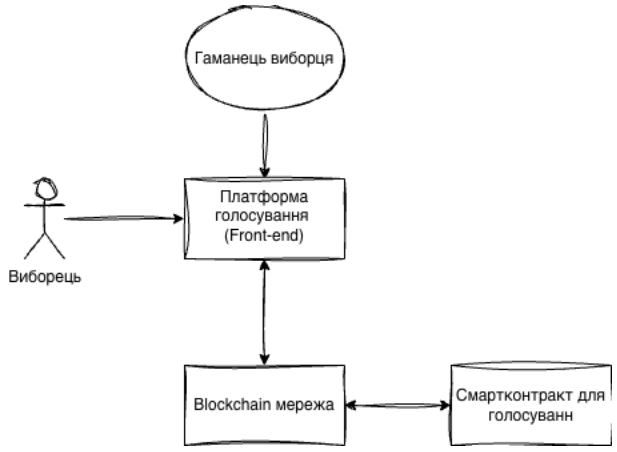

Рисунок 2.1 Загальна архітектура системи

На схемі видно, що центральною ланкою користувацької взаємодії виступає платформа голосування (Front-end). Саме вона формує зрозумілий інтерфейс для виборця, де можна переглянути активну процедуру, ознайомитися з варіантами і виконати власне волевиявлення. Важливо, що фронтенд у цій моделі не є “джерелом істини” щодо результатів - він виконує роль зручного інструмента доступу до децентралізованої логіки.

Виборець на схемі показаний як кінцевий учасник процесу, який здійснює взаємодію з платформою. У контексті університетського сценарію це студент, що має право участі у голосуванні. Його прямий контакт із системою реалізований через інтерфейс, що робить процедуру максимально звичною на рівні користувацького досвіду: студент не повинен розуміти внутрішню будову блокчейну, щоб коректно взяти участь у голосуванні.

Окремим важливим елементом є гаманець виборця. На архітектурному рівні він виконує роль механізму особистого підтвердження дії: саме через гаманець користувач підписує транзакцію голосування. Це принципово підсилює довіру до процесу, оскільки голос не “відправляється системою за користувача”, а підтверджується ним власноруч як власна цифрова дія.

Далі схема показує зв’язок із blockchain мережею, яка виступає середовищем виконання і збереження незмінних записів. У межах цієї архітектури блокчейн є тим рівнем, де фактично “живе” достовірний перебіг голосування: права доступу, фіксація голосу, неможливість повторної участі та прозора перевірка підсумків.

Нарешті, смартконтракт для голосування відображає ядро логіки системи. Саме він визначає правила процедури та гарантує, що вони будуть застосовані однаково до кожного учасника. Архітектурно важливо, що взаємодія смартконтракту відбувається не напряму з користувачем, а через блокчейн-мережу, що підкреслює децентралізовану природу рішення і відсутність залежності від централізованого сервера як основного носія істини.

Таким чином, подана схема ілюструє мінімально достатню, але концептуально завершену модель децентралізованого університетського голосування:

- інтерфейс забезпечує зручність і зрозумілість участі;
- гаманець гарантує персональне підтвердження дії;
- блокчейн забезпечує незмінність і перевірність;
- смартконтракт формалізує правила голосування.

2.2.2. Архітектура смартконтрактів

На рис. 2.2 подано концептуальну архітектуру смартконтрактної частини системи університетського голосування. Представлена модель підкреслює, що рішення не зводиться до одного “великого” контракту: смартконтрактна логіка організована як набір взаємопов’язаних компонентів із чітким розмежуванням функцій - створення голосувань, керування переліком учасників, виконання процедури та формування підсумків. Така

організація є доцільною для сценарію, де одночасно важливо забезпечити прозорість, контроль доступу та можливість масштабування під різні внутрішні виборчі процеси.

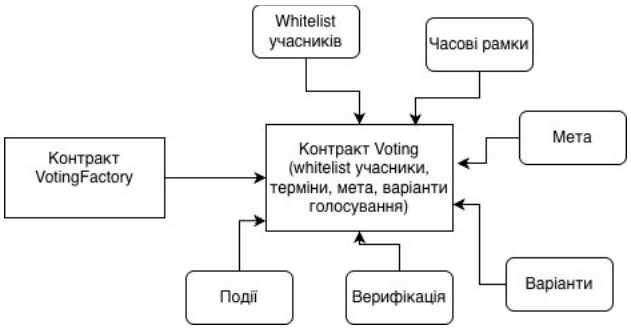

Рисунок 2.2 Архітектура смартконтрактів

Ключовим “входом” у смартконтрактний контур є контракт VotingFactory, який доцільно трактувати як організаційний шлюз системи. Саме він відповідає за створення нових голосувань у вигляді окремих екземплярів контракту Voting, тобто кожна процедура має власний контракт і власний стан. Такий підхід зменшує ризик змішування даних різних голосувань і спрощує перевірку: аудитору або учасникам достатньо аналізувати конкретний контракт конкретної процедури, не “відфільтровуючи” зайве із загального стану системи.

Окремі контракти Voting у межах цієї моделі виступають центральними носіями правил процедури: вони відображають життєвий цикл голосування (часові межі), перевіряють допустимість дій учасників та підтримують керований доступ через попередньо визначений перелік адрес. Відповідно, організатор працює з процедурою як із самостійною “сесією”, а учасники взаємодіють із чітко визначеними правилами, зафіксованими в коді, що відповідає загальній ідеї формалізації виборчого процесу на рівні смартконтрактів.

Окремо варто підкреслити, що винесення створення голосувань у фабрику також підтримує організаційну зручність: система може

накопичувати набір незалежних процедур у різні періоди (наприклад, для різних підрозділів університету), не ускладнюючи базову логіку та не перевантажуючи один контракт функціями, які природніше розділяти між компонентами.

2.2.3. Модель безпеки

У проєктуванні децентралізованої системи університетського голосування безпека є базою довіри до всієї процедури. Навіть у локальному інституційному сценарії учасники очікують гарантій: неможливості повторного голосування, відсутності «тихих» змін результатів і прозорої перевірності підсумків. Тому в архітектурі смартконтрактів ці механізми мають бути закладені одразу - як правила, що виконуються автоматично, а не як сподівання на «організаційну чесність».

Критичний елемент - захист від повторного голосування [23]. Концептуально це означає жорстку одноразовість дії для кожної дозволеної адреси: система фіксує факт участі у сесії та блокує будь-які повторні спроби голосування з тієї ж адреси. Таке обмеження зменшує ризик replay-сценаріїв, коли користувач або зловмисник намагається повторити вже виконану дію, щоб вплинути на результат. Іншими захисними елементами є перевірки на доступність у списку whitelist, часові рамки голосування.

2.2.4. Логіка верифікації користувача

Оскільки у роботі обрана неанонімна модель університетського голосування, верифікація виборця розглядається як двоетапний процес із чітким розподілом відповідальності. Це важливий концептуальний момент: система не намагається “зробити ідентифікацію всередині блокчейну”, а використовує блокчейн як механізм прозорої фіксації права участі та самої дії

голосування. Такий підхід є найбільш практичним для академічного прототипу і відповідає логіці інституційних процедур.

Перший етап відбувається до потрапляння адреси у whitelist. На цьому рівні університет (або уповноважена комісія) перевіряє, що конкретний студент має право участі у виборах, після чого отримує від нього адресу криптогаманця. Саме цей організаційний процес формує легітимну основу списку учасників і знімає ризик участі сторонніх осіб.

Другий етап реалізується вже у смартконтракті. Принципово важливо, що у блокчейні не зберігаються персональні дані, а лише адреса, включена до whitelist. Отже, контракт виконує не “ідентифікацію особи”, а перевірку права адреси на участь. Це дозволяє зберегти розумний баланс між приватністю для студентів у межах спільноти та необхідним рівнем контролю з боку університету як організатора процедури.

2.3. Опис алгоритмів функціонування системи

У межах запропонованої архітектури система університетського електронного голосування працює як послідовність прозорих і формалізованих етапів, де організаційні дії університету поєднуються з незмінною фіксацією ключових подій у блокчейні.

Першим етапом є формування та ініціація голосування. Уповноважений організатор (університет або студентська виборча комісія) визначає мету голосування, формує перелік варіантів, встановлює часові межі процедури та створює нову сесію голосування через відповідний контракт (наприклад, через VotingFactory).

Другим етапом виступає реєстрація учасників і внесення адрес до whitelist. Студенти проходять організаційно визначену перевірку права участі, після чого надають адресу власного криптогаманця. Організатор додає ці адреси до whitelist з боку смартконтракту.

Третій етап - процедура голосування. Студент підключає гаманець до платформи, обирає варіант і підтверджує дію транзакцією. Перед фіксацією голосу смартконтракт перевіряє, чи входить адреса до whitelist, чи не голосував користувач раніше, а також чи перебуває процедура у межах дозволеного часу. Лише після проходження цих перевірок голос записується у стан контракту, а відповідна подія фіксується у блокчейні, що забезпечує прозорість і можливість незалежного аудиту.

Таким чином, алгоритм функціонування системи є цілісним і практично придатним для університетського контексту: організаційний рівень забезпечує легітимний склад учасників, а блокчейн і смартконтракти гарантують одноразовість голосування, незмінність записів і прозорість результатів. У наступному розділі ці алгоритми можуть бути підтверджені описом реалізації та тестування прототипу.

2.4. Вибір технологій та середовища розгортання

У проєктному розділі важливо не просто перелічити інструменти, а показати, що кожен з них логічно підтримує загальну модель системи та дозволяє коректно продемонструвати результати реалізації.

Для смартконтрактної частини обрано мову Solidity [24], оскільки саме вона є базовим інструментом для розробки контрактів у мережах Ethereum-подібної архітектури. Це забезпечує відповідність логіки голосування концепції децентралізованого виконання правил, а також дозволяє реалізувати механізми доступу, whitelist-верифікації та фіксації результатів у формі прозорої контрактної моделі. Solidity є найбільш популярним вибором [25] для такої теми, оскільки підтримує усталені практики написання та тестування контрактів, що важливо для академічного прототипу.

Як основний інструмент розробки та тестування смартконтрактів використовується Foundry [26]. Його вибір зумовлений зручністю швидкого

циклу розробки, можливістю написання структурованих тестів та відносно простим налаштуванням середовища для експериментальної апробації.

Середовище розгортання обрано у форматі Sepolia Testnet [27], що є виправданим для дослідницького завдання. Використання тестової мережі дозволяє перевірити працездатність смартконтрактної логіки без фінансових ризиків і при цьому зберегти технологічну сумісність із основною Ethereum-екосистемою.

Користувацьку частину системи запроєктовано на основі Next.js [28], що дає змогу створити практичний і зручний веб-інтерфейс для університетського сценарію голосування. У межах прототипу саме фронтенд забезпечує доступність системи для студентів, а також реалізує логіку підключення гаманця, відображення статусу участі та інтерпретації результатів. Це важливий елемент, оскільки демонструє не лише “чисту” блокчейн-частину, а й те, як децентралізована модель працює у зрозумілій для користувача формі.

Для взаємодії інтерфейсу зі смартконтрактами використовується ethers.js [29] - бібліотека, яка дозволяє коректно виконувати читання стану контрактів, надсилати транзакції та обробляти відповіді мережі.

Для отримання та структурування подій голосування застосовується підхід на основі GraphQL [30], а саме Subgraph [31]. У контексті системи це дозволяє впорядковано працювати з подіями, які формують прозорий журнал перебігу процедур: створення голосування, внесення адрес до whitelist, факт подання голосу та завершення сесії. Такий механізм забезпечує більш зручне відображення ходу голосування на рівні інтерфейсу.

Як основний гаманець користувача використовується MetaMask [32], що є найбільш поширеним та зручним варіантом для систем у межах Ethereum-екосистеми [33]. Його застосування дозволяє студентам підтверджувати транзакції голосування як особисту цифрову дію.

Отже, обраний технологічний стек є узгодженим із запропонованою архітектурою та практичними вимогами університетського сценарію. Він забезпечує повний цикл створення прототипу: від формалізації правил голосування у вигляді смартконтрактів до зручного користувацького інтерфейсу та прозорого відстеження подій. Це створює достатні умови для подальшого опису реалізації, тестування та оцінки результатів у наступних частинах роботи.

Висновки до другого розділу

У другому розділі сформовано проєктну основу децентралізованої системи електронного голосування на базі блокчейну для університетського сценарію. Обґрунтовано вибір децентралізованої моделі та Ethereum-подібної платформи як середовища, здатного забезпечити прозору фіксацію ключових подій голосування. Визначено роль смартконтрактів як механізму формалізації правил процедури, що дозволяє підвищити довіру до результатів завдяки виконанню визначених алгоритмічних обмежень.

Запроєктовано загальну архітектуру системи та структуру смартконтрактної частини із розподілом відповідальності між модулем створення голосувань, керуванням whitelist та логікою підрахунку результатів. Окреслено базову модель безпеки, яка передбачає захист від повторного голосування, збереження цілісності даних і незмінність історії голосів, а також враховує ризики централізації адміністративної ролі.

Уточнено логіку верифікації користувача відповідно до неанонімної моделі: ідентифікація студента відбувається до включення адреси у whitelist, тоді як у смартконтракті зберігається лише адреса як технічний маркер права участі. Також обґрунтовано вибір технологій і середовища розгортання (Solidity, Foundry, Sepolia Testnet, Next.js, ethers.js, GraphQL, MetaMask) як узгодженого стеку для реалізації та демонстрації працездатного прототипу. У

сукупності це створює логічний перехід до третього розділу, де буде представлено практичну реалізацію та перевірку запропонованих рішень.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АПРОБАЦІЯ ПРОТОТИПУ ДЕЦЕНТРАЛІЗОВАНОЇ СИСТЕМИ ЕЛЕКТРОННОГО ГОЛОСУВАННЯ

3.1 Реалізація програмного прототипу системи

У межах третього розділу виконано перехід від спроєктованої в попередньому розділі архітектури до працездатного програмного прототипу децентралізованої системи електронного голосування.

Прототип реалізовано як набір взаємопов’язаних компонентів, що охоплюють смартконтрактний рівень, рівень читання/агрегації подій та клієнтський інтерфейс із взаємодією через криптогаманець. На смартконтрактному рівні система складається з контракту створення голосувань (фабрики) та контрактів окремих голосувань, які відповідають за реєстрацію стану процедури, допуск адрес, фіксацію голосів і формування підсумків. Повний програмний код ключових смартконтрактів наведено у додатках (див. дод. А та Б).

Для ефективного отримання даних про перебіг голосувань (створення процедур, додавання учасників, факти голосування, завершення) у прототипі передбачено окремий механізм індексації подій і читання стану через GraphQL (див. дод. В). Такий підхід дає змогу не покладатися на пряме “сканування” блокчейну клієнтом, а отримувати структуровані дані для інтерфейсу у зручній формі запитів.

Клієнтська частина прототипу забезпечує користувацькі сценарії (перегляд активних голосувань, перевірка доступу, подання голосу, перегляд результатів) і взаємодіє з блокчейном через криптогаманець: підписання транзакцій та ініціювання викликів смартконтрактів відбувається зі сторони користувача, тоді як читання даних для відображення стану та історії подій може виконуватися через індексаційний шар У результаті прототип

демонструє завершений цикл “запис у блокчейн → поява подій → читання й відображення в інтерфейсі”, що придатно для апробації у тестовому середовищі.

3.2 Логіка смартконтрактів виборчого процесу

Організаційний рівень реалізовано через контракт VotingFactory, який відповідає за створення та реєстрацію голосувань. Фабрика дозволяє ініціювати окрему контрактну сесію для кожної виборчої процедури функцію якої (див. дод. Г, рис. Г.1) із заданими параметрами: організатором, метою, часовими межами, набором варіантів та початковим whitelist. Після створення голосування його адреса зберігається у реєстрі фабрики разом із базовими метаданими, що дає можливість отримувати список усіх голосувань або фільтрувати їх за організатором чи часовим періодом. Цей підхід забезпечує масштабованість рішення й відповідає практиці, коли університет може проводити кілька різних голосувань у межах одного організаційного циклу.

Безпосереднє виконання процедури голосування реалізує контракт Voting, який акумулює всю логіку конкретної сесії. У конструкторі контракту (див. дод. Г, рис. Г.2) здійснюється первинна валідація параметрів: коректність адреси організатора, узгодженість часових меж, наявність варіантів голосування та допустима кількість опцій.

Механізм whitelist-верифікації реалізовано через збереження дозволених адрес у структурі даних контракту. Організатор має можливість додавати й видаляти учасників до початку голосування, що логічно узгоджується з університетською процедурою формування списку студентів, які мають право участі. Важливо, що після старту голосування внесення змін до списку заблоковано на рівні контрактної логіки. Це підсилює довіру до процедури, оскільки склад учасників не може змінюватися “в процесі”, а отже зменшуються ризики маніпуляцій із доступом під час активної фази.

Ключова функція подання голосу реалізована з урахуванням трьох обов’язкових перевірок (див. дод. Г, рис. Г.3). По-перше, виборець повинен бути присутнім у whitelist. По-друге, система перевіряє, чи не було голосування з цієї адреси раніше, що забезпечує одноразовість участі. По-третє, валідується коректність обраного варіанту. Лише після виконання цих умов контракт фіксує голос, оновлює лічильники відповідного варіанту та збільшує загальну кількість голосів.

Підрахунок голосів у межах прототипу інтегровано в контракт Voting. Кожен варіант містить власний лічильник, що оновлюється під час голосування, а результати можуть бути отримані як для окремої опції, так і у вигляді повного набору підсумкових даних. Додатково передбачено логіку визначення переможця після завершення голосування (див. дод. Г, рис. Г.4). Таким чином результати формуються на основі стану контракту, що усуває необхідність зовнішнього підрахунку як джерела довіри.

Важливим елементом реалізації є модифікатори доступу та часові обмеження, які задають формальний життєвий цикл голосування. Організатор має обмежені повноваження для адміністративних операцій, а дії виборців дозволені лише у межах активної фази. Публікація результатів можлива тільки після завершення визначеного часу.

Окремо слід відзначити подієву модель (див. дод. Г, рис. Г.5), яка супроводжує ключові етапи процедури: створення голосування, зміни у whitelist, факт подання голосу та публікацію результатів. Події є важливими не тільки для прозорості, а й для інтеграції з клієнтською частиною та механізмами індексації. Саме вони дозволяють інтерфейсу коректно відображати стан системи, формувати список голосувань і візуалізувати перебіг процедури без втрати зв’язку з ончейн-джерелом даних.

3.3 Клієнтська частина та взаємодія виборців

Клієнтська частина прототипу є повноцінним провідником користувача через процедуру голосування: від входу в систему й підключення гаманця до подання голосу та перегляду результатів. У межах цього модуля важливо було забезпечити дві речі: по-перше, зрозумілий та послідовний UX для студентів, а по-друге - коректне відображення станів, що випливають зі смартконтрактної логіки (часові вікна, доступ через whitelist, статус активності, результати).

Початковою точкою взаємодії є головна сторінка вебзастосунку (див. дод. Г, рис. Г.6), яка формує контекст і підводить користувача до цільового сценарію - участі в наявних процедурах голосування. Фактично це “вхідний екран”, що допомагає не занурювати користувача в технічні аспекти блокчейну, а одразу переводить його до практичної дії: перегляду активних голосувань та подальшої участі.

Ключовим етапом взаємодії є підключення криптогаманця, оскільки саме адреса користувача виступає технічним ідентифікатором участі у неанонімній whitelist-моделі. Стартовий стан сторінки “Активні голосування” (див. дод. Г, рис. Г.7) навмисно передбачає запит на підключення: доки адреса не визначена, система не повинна показувати потенційно нерелевантні стани або створювати враження “порожнього” інтерфейсу. Після натискання кнопки з’являється стандартне діалогове вікно MetaMask (див. дод. Г, рис. Г.8), де користувач обирає акаунт і надає застосунку доступ до адреси.

Після успішного підключення інтерфейс переходить у персоналізований режим: на сторінці “Активні голосування” (див. дод. Г, рис. Г.9) відображається факт з’єднання та скорочена адреса користувача, а також стає доступним перелік процедур у форматі карток. Кожна картка містить базові метадані (назва, адреса контракту, часові межі) і відображає статус активності, що напряму відповідає часовій логіці смартконтракту. Такий

підхід зручний для університетського середовища, де паралельно можуть існувати кілька голосувань у різних часових інтервалах, і користувачеві важливо одразу бачити, які з них доступні саме зараз.

Найбільш показовим з точки зору “закінченого сценарію” є сторінка конкретного голосування (див. дод. Г, рис. Г.10). На ній зібрано ключові елементи, необхідні для прозорого сприйняття процедури: блок результатів із наочним розподілом голосів, інформаційна панель із параметрами голосування (опис, організатор, час, адреса контракту), а також індикатори стану (активне/завершене). Додатково інтерфейс відображає часову логіку та таймер - це підкреслює, що клієнтська частина підпорядковується правилам контракту і не дозволяє “взаємодії поза межами дозволеного періоду”.

Окремим елементом прозорості є секція “Останні голоси”, де демонструються події у вигляді скорочених адрес. Це підсилює перевірність перебігу процедури, але водночас не суперечить обраному підходу, оскільки інтерфейс не розкриває персональні дані - лише блокчейн-ідентифікатори.

Інтерактивний етап подання голосу (див. дод. Г, рис. Г.11) реалізовано як вибір одного з варіантів із подальшим підтвердженням дії користувачем. Така побудова інтерфейсу має прикладне значення: користувач не “натискає кнопку в системі”, а здійснює особисто підтверджену дію через гаманець, що узгоджується з загальною логікою Web3-взаємодії та обраною моделлю неанонімного голосування.

Висновки до третього розділу

У межах третього розділу було реалізовано програмний прототип децентралізованої системи електронного голосування для університетського середовища, який відтворює повний цикл проведення процедури: від створення голосування до відображення результатів. На відміну від другого розділу, де було сформовано концепцію й архітектуру, тут показано практичну

реалізацію взаємодії смартконтрактів, клієнтського інтерфейсу та механізму індексації подій, що забезпечує цілісність і завершеність сценарію використання прототипу.

Смартконтрактний рівень прототипу представлено двома ключовими контрактами - VotingFactory та Voting, що відповідає закладеній ідеї модульності: фабрика створює окремі незалежні сесії голосувань, а контракт Voting зосереджує у собі правила процедури, whitelist, часові рамки та підрахунок голосів. Така структура робить модель придатною для масштабування в межах студентського самоврядування й одночасно спрощує аудит кожного окремого голосування як автономної сутності.

Клієнтська частина прототипу виконує роль зрозумілого інтерфейсу для студентів: стартова сторінка формує контекст і підводить користувача до сценарію участі у процедурах, а сторінка конкретного голосування демонструє результати, метадані, статус і журнал останніх голосів як елемент прозорості. Підключення та підтвердження дій реалізовано через MetaMask, що забезпечує персоналізоване підтвердження транзакції й логічно узгоджується з неанонімною whitelist-моделлю. Для структурування та зручного відображення перебігу процедур застосовано індексацію подій і роботу з ними через GraphQL, що підсилює можливість перегляду історії та перевірності результатів на рівні інтерфейсу.

ВИСНОВКИ

У цій кваліфікаційній роботі розглянуто проблему підвищення прозорості, довіри та керованості локальних електронних процедур волевиявлення завдяки застосуванню блокчейн-технологій. Актуальність теми пов’язана з тим, що в університетському й організаційному середовищі зростає потреба у швидких і перевірюваних механізмах ухвалення колективних рішень, де водночас необхідно витримати баланс між відкритістю підсумків, технічною надійністю та контрольованим доступом учасників.

Метою дослідження було розроблення та обґрунтування підходів до створення програмного прототипу децентралізованої системи електронного голосування на основі блокчейну. Для досягнення мети виконано ключові завдання: проаналізовано моделі е-голосування та вимоги до перевірності результатів, обґрунтовано доцільність блокчейну й смартконтрактів, обрано неанонімну whitelist-модель для університетського контексту, спроєктовано архітектуру та реалізовано прототип із демонстрацією його роботи.

У першому розділі систематизовано теоретичні підходи до електронного голосування та визначено передумови використання блокчейну як інфраструктури, що забезпечує незмінність результатів і відтворювану перевірку процедур у локальних сценаріях. У другому розділі сформовано концептуальні засади системи, обґрунтовано вибір Ethereum-подібної платформи та запропоновано архітектуру з розподілом відповідальностей між клієнтським застосунком, мережею та смартконтрактним рівнем; визначено базові механізми безпеки й логіку whitelist-верифікації, де первинна ідентифікація відбувається поза блокчейном, а в контракті фіксується лише адреса гаманця як маркер права участі. У третьому розділі реалізовано та апробовано програмний прототип: смартконтракти для створення й проведення голосувань, керування допуском і фіксації результатів, клієнтську

частину з інтеграцією гаманця та індексацією подій, що спрощує відображення голосувань і підсумків у інтерфейсі.

Практичне значення роботи полягає у створенні узгодженого прототипу з прозорою логікою голосування та контрольованим доступом через whitelist, який поєднує інституційну відповідальність організатора з перевагами незмінного журналу подій у блокчейні. Такий підхід зменшує залежність процедури від ручних або неформалізованих механізмів підрахунку й фіксації волевиявлення та підсилює довіру до результатів у регламентованих локальних сценаріях.

Перспективами подальших досліджень є розширення ролей і прав доступу, розвиток механізмів делегування та аудиту, за потреби - вдосконалення приватності для окремих сценаріїв, а також формалізація інтеграції з внутрішніми реєстрами університету й оцінка продуктивності системи при зростанні кількості учасників і варіантів голосування.

Отже, поставлену мету роботи досягнуто, а основні завдання виконано: обґрунтовано доцільність застосування блокчейн-технологій для локальних електронних процедур, спроєктовано архітектуру децентралізованої системи та реалізовано прототип, придатний для демонстрації практичної реалізації запропонованого підходу в університетських голосуваннях.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Peter W. Introducing electronic voting: Essential considerations. Stockholm : International Institute for Democracy and Electoral Assistance, 2011. 36 с.

2. CoE Search - CM. *CoE Search - Public Search*. URL: https://search.coe.int/cm?i=0900001680726f6f (дата звернення: 17.12.2025).

3. Виборчий кодекс України : Кодекс України від 19.12.2019 № 396-IX : станом на 31 груд. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/396-20#Text (дата звернення: 17.12.2025).

4. Про вибори народних депутатів України : Закон України від 17.11.2011 № 4061-VI : станом на 1 січ. 2022 р. URL: https://zakon.rada.gov.ua/laws/show/4061-17#Text (дата звернення: 17.12.2025).

5. Про електронні довірчі послуги : Закон України від 05.10.2017 № 2155-VIII : станом на 18 груд. 2024 р. URL: https://zakon.rada.gov.ua/laws/show/2155-19#Text (дата звернення: 17.12.2025).

6. Про затвердження Положення про інтегровану систему електронної ідентифікації : Постанова Каб. Міністрів України від 19.06.2019 № 546 : станом на 31 груд. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/546-2019-п#Text (дата звернення: 17.12.2025).

7. Про затвердження Положення про Систему BankID Національного банку України : Постанова Нац. банку України від 17.03.2020 № 32 : станом на 30 трав. 2024 р. URL: https://zakon.rada.gov.ua/laws/show/v0032500-20#Text (дата звернення: 17.12.2025).

8. Про товариства з обмеженою та додатковою відповідальністю : Закон України від 06.02.2018 № 2275-VIII : станом на 28 серп. 2025 р.

URL: https://zakon.rada.gov.ua/laws/show/2275-19#Text (дата звернення: 17.12.2025).

9. Про акціонерні товариства : Закон України від 27.07.2022 № 2465-IX : станом на 28 серп. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/2465-20#Text (дата звернення: 17.12.2025).

10. Про об'єднання співвласників багатоквартирного будинку : Закон України від 29.11.2001 № 2866-III : станом на 10 листоп. 2023 р. URL: https://zakon.rada.gov.ua/laws/show/2866-14#Text (дата звернення: 17.12.2025).

11. Про внесення змін до деяких законів України щодо спрощення управління багатоквартирними будинками : Закон України від 14.07.2023 № 3270-IX. URL: https://zakon.rada.gov.ua/laws/show/3270-20#Text (дата звернення: 17.12.2025).

12. Про вищу освіту : Закон України від 01.07.2014 № 1556-VII : станом на 22 верес. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/1556-18#Text (дата звернення: 17.12.2025).

13. Про громадські об'єднання : Закон України від 22.03.2012 № 4572-VI : станом на 3 верес. 2024 р. URL: https://zakon.rada.gov.ua/laws/show/4572-17#Text (дата звернення: 17.12.2025).

14. Про місцеве самоврядування в Україні : Закон України від 21.05.1997 № 280/97-ВР : станом на 31 жовт. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/280/97-вр#Text (дата звернення: 17.12.2025).

15. Про внесення змін до Закону України "Про звернення громадян" щодо електронного звернення та електронної петиції : Закон України від 02.07.2015 № 577-VIII. URL: https://zakon.rada.gov.ua/laws/show/577-19#Text (дата звернення: 17.12.2025).

16. Blockchain-Based E-Voting Systems: A Technology Review / M. Hajian Berenjestanaki та ін. *Electronics*. 2023. Т. 13, № 1. С. 17. URL: https://doi.org/10.3390/electronics13010017 (дата звернення: 18.12.2025).

17. Blockchain for securing electronic voting systems: a survey of architectures, trends, solutions, and challenges / H. O. Ohize та ін. *Cluster Computing*. 2024. Т. 28, № 2. URL: https://doi.org/10.1007/s10586-024-04709-8 (дата звернення: 18.12.2025).

18. ethereum.org. Introduction to smart contracts | ethereum.org. *ethereum.org*. URL: https://ethereum.org/uk/developers/docs/smart-contracts (дата звернення: 17.12.2025).

19. Going from bad to worse: from Internet voting to blockchain voting / S. Park та ін. *Journal of Cybersecurity*. 2021. Т. 7, № 1. URL: https://doi.org/10.1093/cybsec/tyaa025 (дата звернення: 18.12.2025).

20. Alchemy Documentation - Build anything onchain. *Alchemy Documentation*. URL: https://www.alchemy.com/docs/what-is-ethereum (дата звернення: 17.12.2025).

21. Що таке смарт-контракти на блокчейні? 4 реальні приклади використання. *Kraken Learn*. URL: https://www.kraken.com/uk/learn/what-are-smart-contracts (дата звернення: 17.12.2025).

22. Whitelisting Techniques for Smart Contracts. *Medium Hira Siddiqui*. URL: https://medium.com/coinmonks/whitelisting-techniques-for-smart-contracts-ba3998f5d5ba (дата звернення: 17.12.2025).

23. A Guide to Smart Contract Security. *Hedera Learning*. URL: https://hedera.com/learning/smart-contracts/smart-contract-security (дата звернення: 17.12.2025).

24. What is Solidity?. *Alchemy*. URL: https://www.alchemy.com/overviews/solidity (дата звернення: 17.12.2025).

25. Top 6 Smart Contract Languages in 2024 | Chainlink. *Chainlink: The Industry-Standard Oracle Platform*. URL: https://chain.link/education-hub/smart-contract-programming-languages (дата звернення: 17.12.2025).

26. Introduction to Foundry | Quicknode Guides. *High-Performance Blockchain Infrastructure*.
URL: https://www.quicknode.com/guides/ethereum-development/smart-contracts/intro-to-foundry (дата звернення: 17.12.2025).

27. ethereum.org. Networks | ethereum.org. *ethereum.org*. URL: https://ethereum.org/uk/developers/docs/networks/#ethereum-testnets (дата звернення: 17.12.2025).

28. Vercel. Next.js Docs | Next.js. *Next.js by Vercel - The React Framework*. URL: https://nextjs.org/docs (дата звернення: 17.12.2025).

29. Documentation Ethers. URL: https://docs.ethers.org/v6/getting-started/ (дата звернення: 17.12.2025).

30. What Is GraphQL and How It Works - GraphQL Academy. *Hygraph: The Headless CMS That's Fast to Implement and Scale | Hygraph*. URL: https://hygraph.com/learn/graphql (дата звернення: 17.12.2025).

31. What is a subgraph? (2023 Guide). *Alchemy*. URL: https://www.alchemy.com/overviews/what-is-a-subgraph (дата звернення: 17.12.2025).

32. Web3 Wallets: A Complete Overview | Quicknode Guides. *High-Performance Blockchain Infrastructure*. URL: https://www.quicknode.com/guides/web3-fundamentals-security/basics-to-web3-wallets (дата звернення: 17.12.2025).

33. The 15 Best Web3 Wallets for 2025 (Must Read). *Alchemy*. URL: https://www.alchemy.com/overviews/web3-wallets (дата звернення: 17.12.2025).

ДОДАТКИ

Додаток А. Код смарткконтракту VotingFactory

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

import {Voting} from "./Voting.sol";

/**
 * @title VotingFactory
 * @notice Фабрика для створення нових екземплярів контрактів голосування
 * @dev Організаційний "шлюз" системи, що відповідає за створення та реєстрацію голосувань
 */
contract VotingFactory {
    // Custom Errors
    error AccessDenied();
    error InvalidOwner();
    error VotingNotFound();
    error InvalidPeriod();

    // Структура для зберігання інформації про створене голосування
    struct VotingInfo {
```

```
address votingAddress;
        address organizer;
        string goal;
        uint256 createdAt;
        bool exists;
    }

    // Власник фабрики (може бути університет або адміністрація)
    address public immutable owner;

    // Реєстр створених голосувань
    mapping(address => VotingInfo) public votings;
    address[] public votingAddresses;

    // Події
    event VotingCreated(
        address indexed votingAddress,
        address indexed organizer,
        address indexed creator,
        string goal,
        uint256 startTime,
```

```
        uint256 endTime,
        uint256 timestamp
    );

    event FactoryOwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    // Модифікатор для перевірки власника
    modifier onlyOwner() {
        if (msg.sender != owner) {
            revert AccessDenied();
        }
        _;
    }

    /**
     * @notice Конструктор фабрики
     * @param _owner Адреса власника фабрики
     */
    constructor(address _owner) {
        if (_owner == address(0)) revert InvalidOwner();
        owner = _owner;
```

```
}

/**
 * @notice Створення нового голосування
 * @param _organizer Адреса організатора голосування
 * @param _startTime Час початку голосування (Unix timestamp)
 * @param _endTime Час завершення голосування (Unix timestamp)
 * @param _goal Мета/призначення голосування
 * @param _description Додатковий опис голосування
 * @param _optionNames Масив назв варіантів голосування
 * @param _whitelist Масив адрес учасників для whitelist
 * @return votingAddress Адреса створеного контракту голосування
 */
function createVoting(
    address _organizer,
    uint256 _startTime,
    uint256 _endTime,
    string memory _goal,
    string memory _description,
    string[] memory _optionNames,
    address[] memory _whitelist
) external returns (address votingAddress) {
```

```
        // Створення нового екземпляру контракту Voting
          Voting newVoting = new Voting(_organizer, _startTime, _endTime, _goal, _description, _optionNames, _whitelist);

        votingAddress = address(newVoting);

        // Реєстрація голосування в фабриці
        votings[votingAddress] = VotingInfo({
          votingAddress: votingAddress, organizer: _organizer, goal: _goal, createdAt: block.timestamp, exists: true
        });

        votingAddresses.push(votingAddress);

       emit VotingCreated(votingAddress, _organizer, msg.sender, _goal, _startTime, _endTime, block.timestamp);

        return votingAddress;
    }

    /**
     * @notice Отримання інформації про голосування
     * @param _votingAddress Адреса контракту голосування
```

```
     * @return info Структура з інформацією про голосування
     */
        function getVotingInfo(address _votingAddress) external view returns (VotingInfo memory info) {
        if (!votings[_votingAddress].exists) revert VotingNotFound();
        return votings[_votingAddress];
    }

    /**
     * @notice Перевірка, чи існує голосування
     * @param _votingAddress Адреса контракту голосування
     * @return exists Чи існує голосування
     */
    function votingExists(address _votingAddress) external view returns (bool exists) {
        return votings[_votingAddress].exists;
    }

    /**
     * @notice Отримання загальної кількості створених голосувань
     * @return count Кількість голосувань
     */
```

```
    function getVotingCount() external view returns (uint256 count) {
        return votingAddresses.length;
    }

    /**
     * @notice Отримання списку всіх адрес голосувань
     * @return addresses Масив адрес голосувань
     */
    function getAllVotingAddresses() external view returns (address[] memory addresses) {
        return votingAddresses;
    }

    /**
     * @notice Отримання голосувань за організатором
     * @param _organizer Адреса організатора
     * @return addresses Масив адрес голосувань організатора
     */
    function getVotingsByOrganizer(address _organizer) external view returns (address[] memory addresses) {
        uint256 count = 0;
```

```
    // Підрахунок кількості голосувань організатора
    for (uint256 i = 0; i < votingAddresses.length; i++) {
        if (votings[votingAddresses[i]].organizer == _organizer) {
            count++;
        }
    }

    // Створення масиву з адресами
    addresses = new address[](count);
    uint256 index = 0;

    for (uint256 i = 0; i < votingAddresses.length; i++) {
        if (votings[votingAddresses[i]].organizer == _organizer) {
            addresses[index] = votingAddresses[i];
            index++;
        }
    }

    return addresses;
}

/**
```

```
 * @notice Отримання голосувань, створених у певний період
 * @param _fromTimestamp Початковий timestamp
 * @param _toTimestamp Кінцевий timestamp
 * @return addresses Масив адрес голосувань
 */
function getVotingsByPeriod(uint256 _fromTimestamp, uint256 _toTimestamp)
    external
    view
    returns (address[] memory addresses)
{
    if (_fromTimestamp > _toTimestamp) revert InvalidPeriod();

    uint256 count = 0;

    // Підрахунок кількості голосувань у періоді
    for (uint256 i = 0; i < votingAddresses.length; i++) {
        uint256 createdAt = votings[votingAddresses[i]].createdAt;
        if (createdAt >= _fromTimestamp && createdAt <= _toTimestamp) {
            count++;
        }
    }
```

```
        // Створення масиву з адресами
        addresses = new address[](count);
        uint256 index = 0;

        for (uint256 i = 0; i < votingAddresses.length; i++) {
            uint256 createdAt = votings[votingAddresses[i]].createdAt;
            if (createdAt >= _fromTimestamp && createdAt <= _toTimestamp) {
                addresses[index] = votingAddresses[i];
                index++;
            }
        }

        return addresses;
    }
}
```

Додаток Б. Код смартконтракту Voting

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

/**
 * @title Voting
  * @notice Контракт для проведення голосування з whitelist учасників, термінами та варіантами
 * @dev Центральний елемент архітектури, що містить всю логіку голосування
 */
contract Voting {
    // Custom Errors
    error InvalidOrganizer();
    error InvalidTimeframe();
    error EndTimeInPast();
    error NoOptions();
    error TooManyOptions();
    error VotingNotActive();
    error VotingNotFinished();
    error AccessDenied();
    error InvalidAddress();
    error ParticipantAlreadyInWhitelist();
```

```
error CannotAddAfterStart();
error ParticipantNotInWhitelist();
error CannotRemoveAfterStart();
error ParticipantAlreadyVoted();
error AddressNotInWhitelist();
error AlreadyVoted();
error InvalidOption();
error NoVotes();

// Структура для зберігання інформації про варіант голосування
struct Option {
    string name;
    uint256 voteCount;
}

// Структура для зберігання метаданих голосування
struct VotingData {
    string goal; // Мета голосування
    string description; // Додатковий опис
    address creator; // Адреса створювача
}
```

```
// Модифікатор для перевірки, чи голосування активне
modifier onlyActiveVoting() {
    if (block.timestamp < startTime || block.timestamp > endTime) {
        revert VotingNotActive();
    }
    _;
}

// Модифікатор для перевірки, чи голосування завершене
modifier onlyFinishedVoting() {
    if (block.timestamp <= endTime) {
        revert VotingNotFinished();
    }
    _;
}

// Модифікатор для перевірки доступу організатора
modifier onlyOrganizer() {
    if (msg.sender != organizer) {
        revert AccessDenied();
    }
    _;
```

```
    }

    // Публічні змінні
    address public immutable organizer; // Організатор голосування
    uint256 public immutable startTime; // Час початку голосування
    uint256 public immutable endTime; // Час завершення голосування
    VotingData public meta; // Метадані голосування
    uint256 public totalVotes; // Загальна кількість голосів
    uint256 public optionCount; // Кількість варіантів

    // Внутрішні структури даних
    mapping(address => bool) public whitelist; // Whitelist учасників
    mapping(address => bool) public hasVoted; // Перевірка, чи голосував учасник
    mapping(address => uint256) public votes; // Голос конкретного учасника
    mapping(uint256 => Option) public options; // Варіанти голосування
    address[] public whitelistedAddresses; // Масив адрес для ітерації

    // Події
    event VotingCreated(
        address indexed votingAddress, address indexed organizer, string goal, uint256 startTime, uint256 endTime
    );
```

```
    event ParticipantAdded(address indexed participant);

    event ParticipantRemoved(address indexed participant);

        event VoteCast(address indexed voter, uint256 indexed optionId, string optionName, uint256 timestamp);

    event VotingFinished(address indexed votingAddress, uint256 totalVotes, uint256 timestamp);

        event ResultsPublished(address indexed votingAddress, uint256[] optionIds, uint256[] voteCounts, uint256 timestamp);

    /**
     * @notice Конструктор контракту Voting
     * @param _organizer Адреса організатора голосування
     * @param _startTime Час початку голосування (Unix timestamp)
     * @param _endTime Час завершення голосування (Unix timestamp)
     * @param _goal Мета/призначення голосування
     * @param _description Додатковий опис голосування
     * @param _optionNames Масив назв варіантів голосування
     * @param _whitelist Масив адрес учасників для whitelist
```

```
 */

constructor(
    address _organizer,
    uint256 _startTime,
    uint256 _endTime,
    string memory _goal,
    string memory _description,
    string[] memory _optionNames,
    address[] memory _whitelist
) {
    if (_organizer == address(0)) revert InvalidOrganizer();
    if (_startTime >= _endTime) revert InvalidTimeframe();
    if (_endTime <= block.timestamp) revert EndTimeInPast();
    if (_optionNames.length == 0) revert NoOptions();
    if (_optionNames.length > 50) revert TooManyOptions();

    organizer = _organizer;
    startTime = _startTime;
    endTime = _endTime;
```

```
            meta = VotingData({goal: _goal, description: _description, creator: msg.sender});

        // Ініціалізація варіантів голосування
        optionCount = _optionNames.length;
        for (uint256 i = 0; i < _optionNames.length; i++) {
            options[i] = Option({name: _optionNames[i], voteCount: 0});
        }

        // Додавання адрес до whitelist
        for (uint256 i = 0; i < _whitelist.length; i++) {
            if (_whitelist[i] != address(0) && !whitelist[_whitelist[i]]) {
                whitelist[_whitelist[i]] = true;
                whitelistedAddresses.push(_whitelist[i]);
                emit ParticipantAdded(_whitelist[i]);
            }
        }

        emit VotingCreated(address(this), _organizer, _goal, _startTime, _endTime);
    }

    /**
```

```
     * @notice Додавання учасника до whitelist
     * @param _participant Адреса учасника для додавання
     */
    function addParticipant(address _participant) external onlyOrganizer {
        if (_participant == address(0)) revert InvalidAddress();
        if (whitelist[_participant]) revert ParticipantAlreadyInWhitelist();
        if (block.timestamp >= startTime) revert CannotAddAfterStart();

        whitelist[_participant] = true;
        whitelistedAddresses.push(_participant);

        emit ParticipantAdded(_participant);
    }

    /**
     * @notice Видалення учасника з whitelist
     * @param _participant Адреса учасника для видалення
     */
    function removeParticipant(address _participant) external onlyOrganizer {
        if (!whitelist[_participant]) revert ParticipantNotInWhitelist();
        if (block.timestamp >= startTime) revert CannotRemoveAfterStart();
        if (hasVoted[_participant]) revert ParticipantAlreadyVoted();
```

```
    whitelist[_participant] = false;

    emit ParticipantRemoved(_participant);
}

/**
 * @notice Подача голосу за конкретний варіант
 * @param _optionId ID варіанту (починається з 0)
 */

function vote(uint256 _optionId) external onlyActiveVoting {
    // Верифікація: перевірка whitelist
    if (!whitelist[msg.sender]) revert AddressNotInWhitelist();

    // Верифікація: перевірка, чи не голосував раніше
    if (hasVoted[msg.sender]) revert AlreadyVoted();

    // Верифікація: перевірка валідності варіанту
    if (_optionId >= optionCount) revert InvalidOption();

    // Реєстрація голосу
```

```
        hasVoted[msg.sender] = true;
        votes[msg.sender] = _optionId;
        options[_optionId].voteCount++;
        totalVotes++;

                emit VoteCast(msg.sender, _optionId, options[_optionId].name, block.timestamp);
    }

    /**
     * @notice Отримання результату голосування для конкретного варіанту
     * @param _optionId ID варіанту
     * @return name Назва варіанту
     * @return voteCount Кількість голосів за варіант
     */
        function getOptionResult(uint256 _optionId) external view returns (string memory name, uint256 voteCount) {
        if (_optionId >= optionCount) revert InvalidOption();
        return (options[_optionId].name, options[_optionId].voteCount);
    }

    /**
```

```
     * @notice Отримання всіх результатів голосування
     * @return optionIds Масив ID варіантів
     * @return optionNames Масив назв варіантів
     * @return voteCounts Масив кількості голосів
     */
    function getAllResults()
        external
        view
        returns (uint256[] memory optionIds, string[] memory optionNames, uint256[] memory voteCounts)
    {
        optionIds = new uint256[](optionCount);
        optionNames = new string[](optionCount);
        voteCounts = new uint256[](optionCount);

        for (uint256 i = 0; i < optionCount; i++) {
            optionIds[i] = i;
            optionNames[i] = options[i].name;
            voteCounts[i] = options[i].voteCount;
        }

        return (optionIds, optionNames, voteCounts);
```

```
}

/**
 * @notice Отримання переможця (варіант з найбільшою кількістю голосів)
 * @return winnerId ID переможця
 * @return winnerName Назва переможця
 * @return winnerVotes Кількість голосів переможця
 */
function getWinner()
    external
    view
    onlyFinishedVoting
    returns (uint256 winnerId, string memory winnerName, uint256 winnerVotes)
{
    if (totalVotes == 0) revert NoVotes();

    uint256 maxVotes = 0;
    uint256 winningOptionId = 0;

    for (uint256 i = 0; i < optionCount; i++) {
        if (options[i].voteCount > maxVotes) {
            maxVotes = options[i].voteCount;
```

```
            winningOptionId = i;
        }
    }

    return (winningOptionId, options[winningOptionId].name, maxVotes);
}

/**
 * @notice Публікація результатів (викликається після завершення)
 */

function publishResults() external onlyFinishedVoting {
    uint256[] memory optionIds = new uint256[](optionCount);
    uint256[] memory voteCounts = new uint256[](optionCount);

    for (uint256 i = 0; i < optionCount; i++) {
        optionIds[i] = i;
        voteCounts[i] = options[i].voteCount;
    }

    emit VotingFinished(address(this), totalVotes, block.timestamp);
    emit ResultsPublished(address(this), optionIds, voteCounts, block.timestamp);
```

```
    }

    /**
     * @notice Перевірка статусу голосування
     * @return isActive Чи активне голосування
     * @return isFinished Чи завершене голосування
     * @return timeRemaining Секунд до завершення (0 якщо завершене)
     */
    function getVotingStatus() external view returns (bool isActive, bool isFinished, uint256 timeRemaining) {
        isActive = block.timestamp >= startTime && block.timestamp <= endTime;
        isFinished = block.timestamp > endTime;

        if (block.timestamp < endTime) {
            timeRemaining = endTime - block.timestamp;
        } else {
            timeRemaining = 0;
        }
    }

    /**
     * @notice Отримання кількості учасників у whitelist
```

```
     * @return count Кількість учасників
     */
    function getWhitelistCount() external view returns (uint256 count) {
        return whitelistedAddresses.length;
    }

    /**
     * @notice Перевірка, чи може адреса голосувати
     * @param _address Адреса для перевірки
     * @return canParticipantVote Чи може голосувати
     * @return reason Причина, якщо не може
     */
            function canVote(address _address) external view returns (bool canParticipantVote, string memory reason) {
        if (!whitelist[_address]) {
            return (false, unicode"Адреса не в whitelist");
        }

        if (hasVoted[_address]) {
            return (false, unicode"Вже проголосовано");
        }
```

```
        if (block.timestamp < startTime) {
            return (false, unicode"Голосування ще не почалося");
        }

        if (block.timestamp > endTime) {
            return (false, unicode"Голосування вже завершене");
        }

        return (true, "");
    }
}
```

Додаток В. GraphQl схема для відслідковування подій на контрактах

```
type FactoryOwnershipTransferred @entity(immutable: true) {
  id: Bytes!
  previousOwner: Bytes! # address
  newOwner: Bytes! # address
  blockNumber: BigInt!
  blockTimestamp: BigInt!
  transactionHash: Bytes!
}

type VotingCreated @entity(immutable: true) {
  id: Bytes!
  votingAddress: Bytes! # address
  organizer: Bytes! # address
  creator: Bytes! # address
  goal: String! # string
  startTime: BigInt! # uint256
  endTime: BigInt! # uint256
  timestamp: BigInt! # uint256
  blockNumber: BigInt!
  blockTimestamp: BigInt!
  transactionHash: Bytes!
```

```
}

type ParticipantAdded @entity(immutable: true) {
  id: Bytes!
  votingAddress: Bytes! # address
  participant: Bytes! # address
  blockNumber: BigInt!
  blockTimestamp: BigInt!
  transactionHash: Bytes!
}

type ParticipantRemoved @entity(immutable: true) {
  id: Bytes!
  votingAddress: Bytes! # address
  participant: Bytes! # address
  blockNumber: BigInt!
  blockTimestamp: BigInt!
  transactionHash: Bytes!
}

type VoteCast @entity(immutable: true) {
  id: Bytes!
```

```
  votingAddress: Bytes! # address
  voter: Bytes! # address
  optionId: BigInt! # uint256
  optionName: String! # string
  timestamp: BigInt! # uint256
  blockNumber: BigInt!
  blockTimestamp: BigInt!
  transactionHash: Bytes!
}

type VotingFinished @entity(immutable: true) {
  id: Bytes!
  votingAddress: Bytes! # address
  totalVotes: BigInt! # uint256
  timestamp: BigInt! # uint256
  blockNumber: BigInt!
  blockTimestamp: BigInt!
  transactionHash: Bytes!
}

type ResultsPublished @entity(immutable: true) {
  id: Bytes!
```

```
  votingAddress: Bytes! # address
  optionIds: [BigInt!]! # uint256[]
  voteCounts: [BigInt!]! # uint256[]
  timestamp: BigInt! # uint256
  blockNumber: BigInt!
  blockTimestamp: BigInt!
  transactionHash: Bytes!
}
```

Додаток Г.

```
function createVoting(
    address _organizer,
    uint256 _startTime,
    uint256 _endTime,
    string memory _goal,
    string memory _description,
    string[] memory _optionNames,
    address[] memory _whitelist
) external returns (address votingAddress) {
    // Створення нового екземпляру контракту Voting
    Voting newVoting = new Voting(_organizer, _startTime, _endTime, _goal, _description, _optionNames, _whitelist);

    votingAddress = address(newVoting);

    // Реєстрація голосування в фабриці
    votings[votingAddress] = VotingInfo({
        votingAddress: votingAddress, organizer: _organizer, goal: _goal, createdAt: block.timestamp, exists: true
    });

    votingAddresses.push(votingAddress);

    emit VotingCreated(votingAddress, _organizer, msg.sender, _goal, _startTime, _endTime, block.timestamp);

    return votingAddress;
}
```

Рисунок 1. Функція створення нового контракту голосування

```
constructor(
    address _organizer,
    uint256 _startTime,
    uint256 _endTime,
    string memory _goal,
    string memory _description,
    string[] memory _optionNames,
    address[] memory _whitelist
) {
    if (_organizer == address(0)) revert InvalidOrganizer();
    if (_startTime >= _endTime) revert InvalidTimeframe();
    if (_endTime <= block.timestamp) revert EndTimeInPast();
    if (_optionNames.length == 0) revert NoOptions();
    if (_optionNames.length > 50) revert TooManyOptions();

    organizer = _organizer;
    startTime = _startTime;
    endTime = _endTime;

    meta = VotingData({goal: _goal, description: _description, creator: msg.sender});

    // Ініціалізація варіантів голосування
    optionCount = _optionNames.length;
    for (uint256 i = 0; i < _optionNames.length; i++) {
        options[i] = Option({name: _optionNames[i], voteCount: 0});
    }

    // Додавання адрес до whitelist
    for (uint256 i = 0; i < _whitelist.length; i++) {
        if (_whitelist[i] != address(0) && !whitelist[_whitelist[i]]) {
            whitelist[_whitelist[i]] = true;
            whitelistedAddresses.push(_whitelist[i]);
            emit ParticipantAdded(_whitelist[i]);
        }
    }

    emit VotingCreated(address(this), _organizer, _goal, _startTime, _endTime);
}
```

Рисунок 2 Первинна валідація параметрів при створенні голосування

```
function vote(uint256 _optionId) external onlyActiveVoting {
    // Верифікація: перевірка whitelist
    if (!whitelist[msg.sender]) revert AddressNotInWhitelist();

    // Верифікація: перевірка, чи не голосував раніше
    if (hasVoted[msg.sender]) revert AlreadyVoted();

    // Верифікація: перевірка валідності варіанту
    if (_optionId >= optionCount) revert InvalidOption();

    // Реєстрація голосу
    hasVoted[msg.sender] = true;
    votes[msg.sender] = _optionId;
    options[_optionId].voteCount++;
    totalVotes++;

    emit VoteCast(msg.sender, _optionId, options[_optionId].name, block.timestamp);
}
```

Рисунок 3 Логіка функції подачі голосу

```
function publishResults() external onlyFinishedVoting {
    uint256[] memory optionIds = new uint256[](optionCount);
    uint256[] memory voteCounts = new uint256[](optionCount);

    for (uint256 i = 0; i < optionCount; i++) {
        optionIds[i] = i;
        voteCounts[i] = options[i].voteCount;
    }

    emit VotingFinished(address(this), totalVotes, block.timestamp);
    emit ResultsPublished(address(this), optionIds, voteCounts, block.timestamp);
}
```

Рисунок 4 Функція визначення переможця

```
// Події
event VotingCreated(
    address indexed votingAddress, address indexed organizer, string goal, uint256 startTime, uint256 endTime
);

event ParticipantAdded(address indexed participant);

event ParticipantRemoved(address indexed participant);

event VoteCast(address indexed voter, uint256 indexed optionId, string optionName, uint256 timestamp);

event VotingFinished(address indexed votingAddress, uint256 totalVotes, uint256 timestamp);

event ResultsPublished(address indexed votingAddress, uint256[] optionIds, uint256[] voteCounts, uint256 timestamp);
```

Рисунок 5 Список подій на контракті голосування

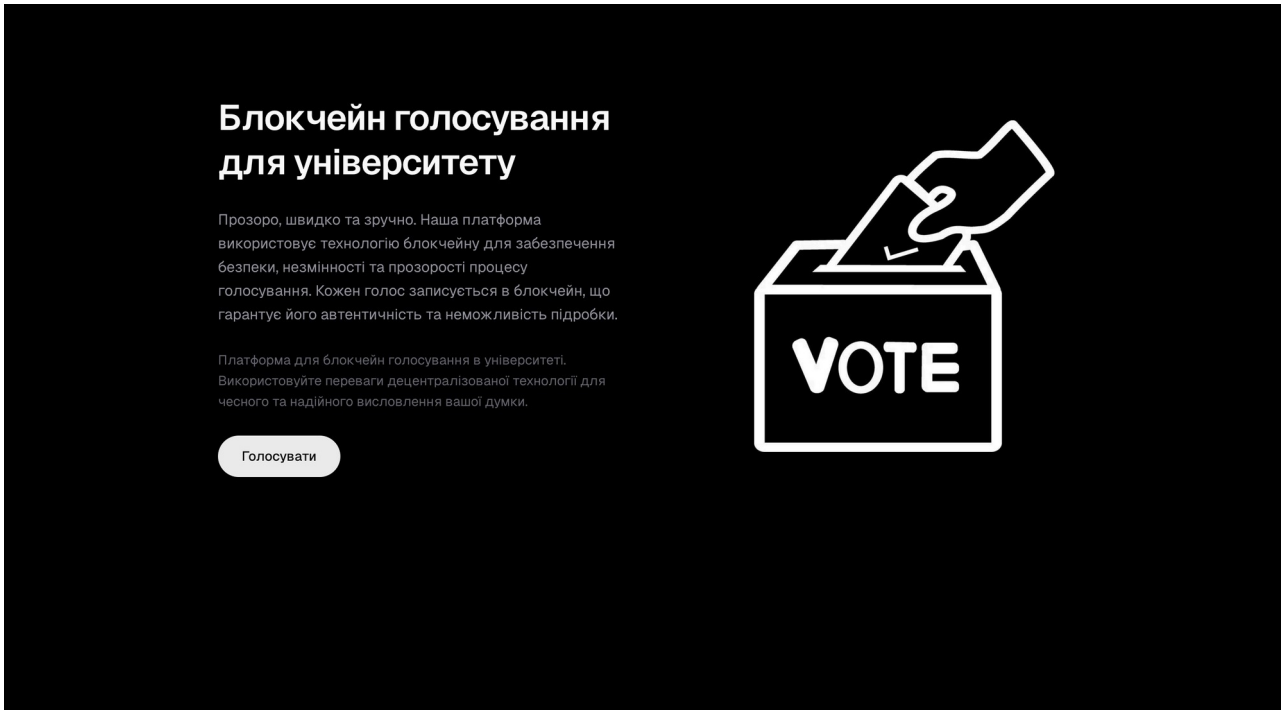

Рисунок 6 Головна сторінка

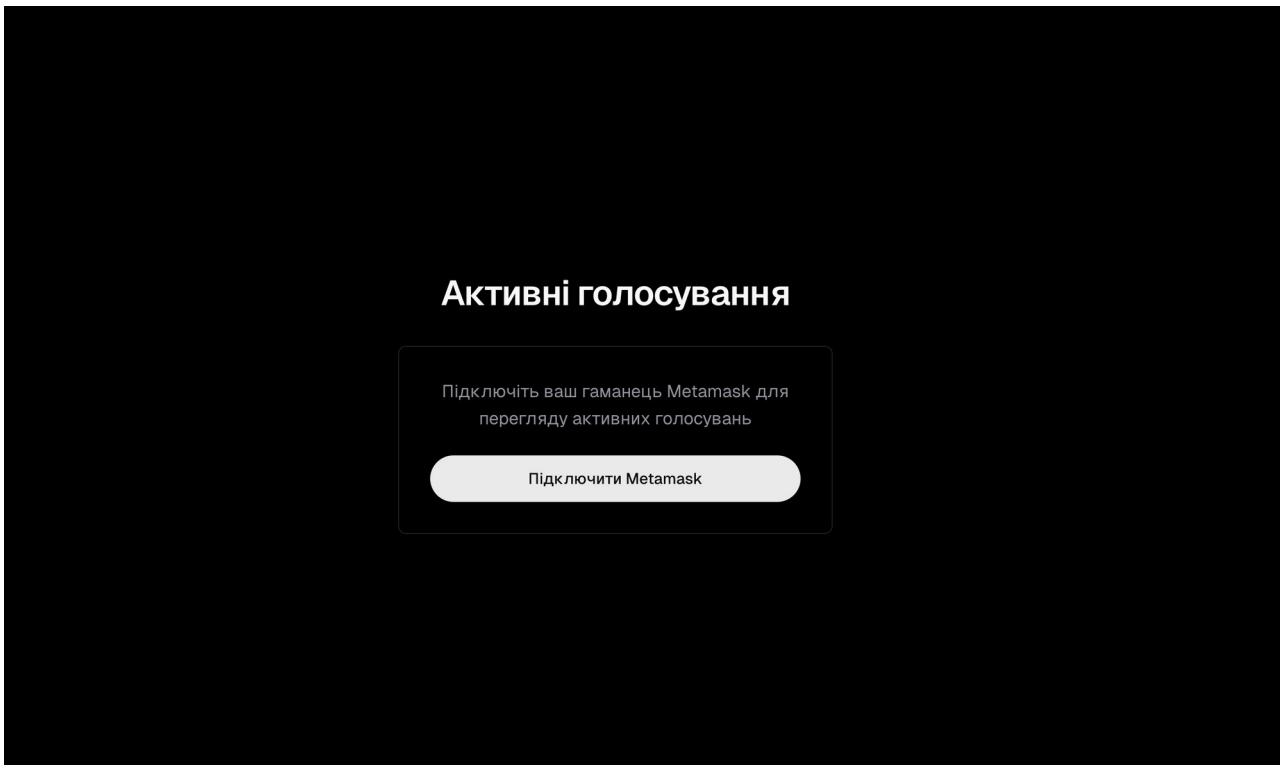

Рисунок 7 Сторінка підключення гаманцю

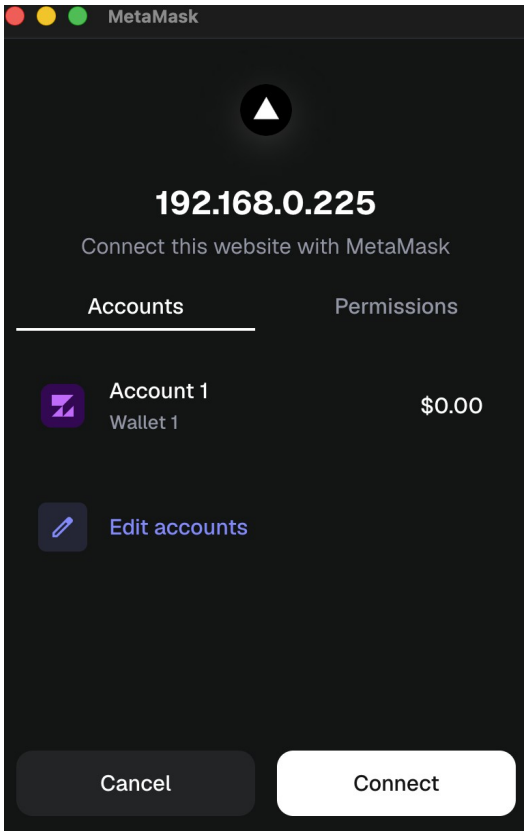

Рисунок 8 Підключення гаманця Metamask

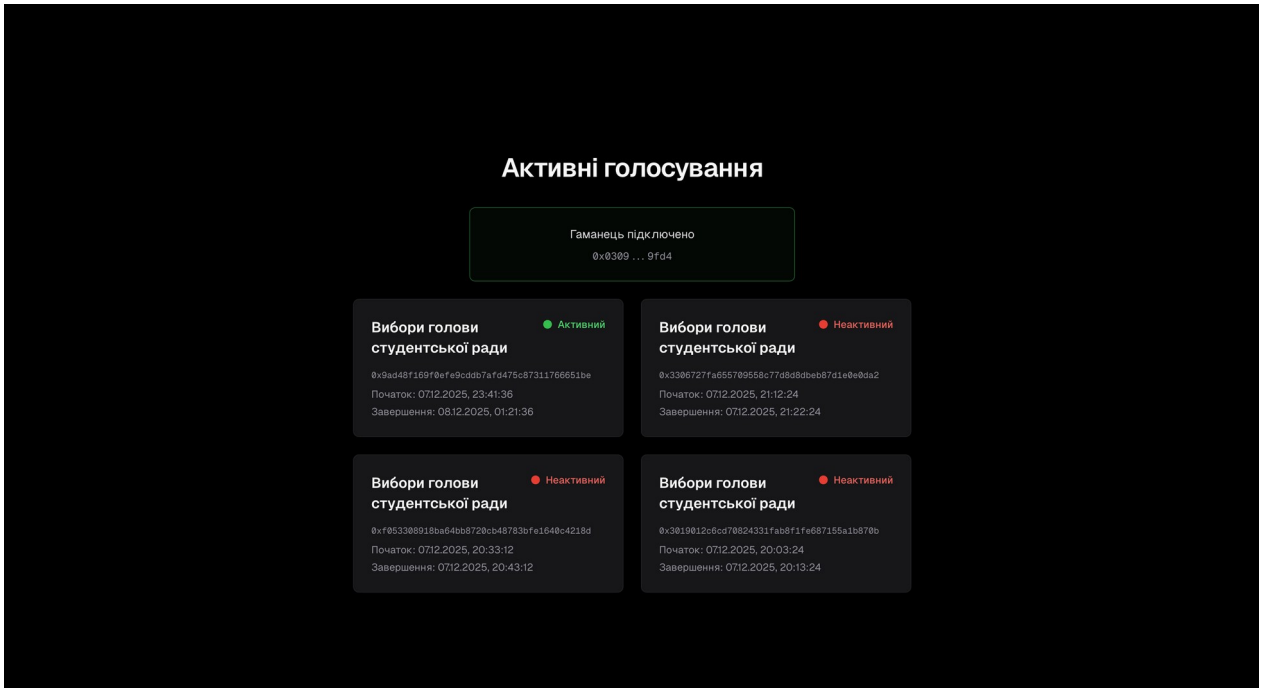

Рисунок 9 Сторінка з сесіями голосування

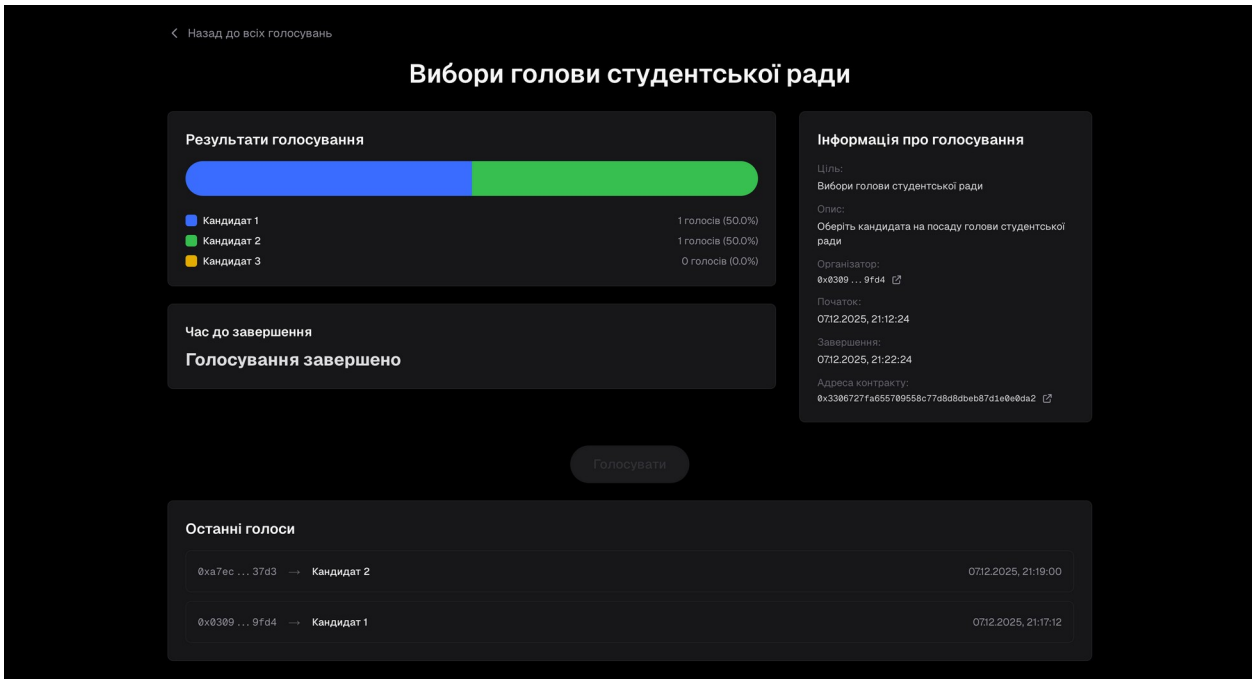

Рисунок 10 Сторінка конкретного голосування

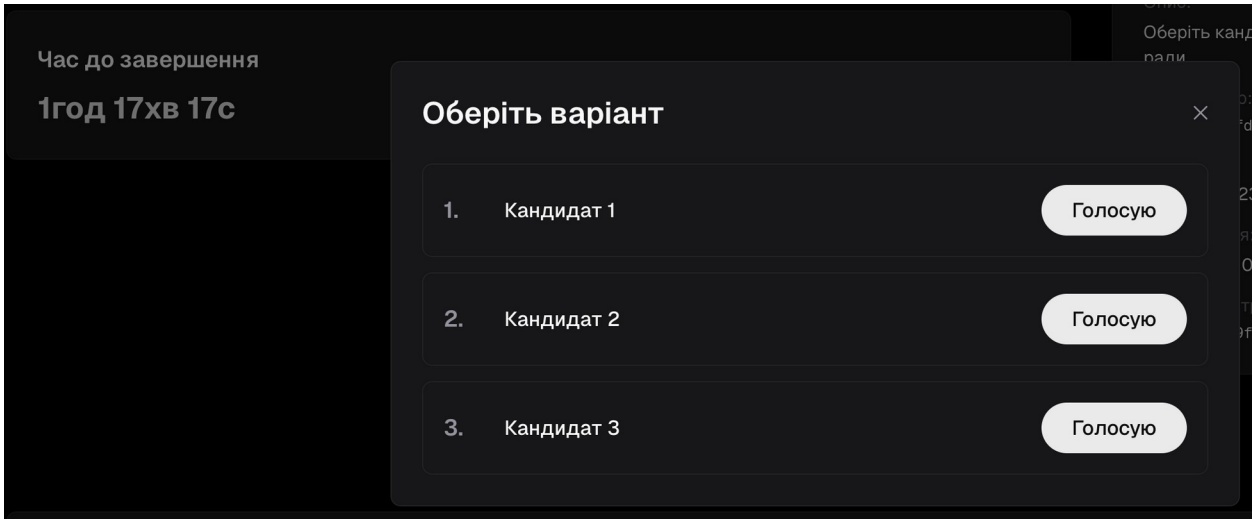

Рисунок 11 Інтерактивний етап вибору варіанту