Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для проведення голосувань на основі блокчейну. Фронт-енд та мобільний застосунок
(тема)

Виконав:
здобувач 4 року навчання
групи ПЗПІ-21-9

Євген ПИЛАЙКІН
(Власне ім'я, ПРІЗВИЩЕ)
Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Ірина КИРИЧЕНКО
(посада, Власне ім'я, ПРІЗВИЩЕ)

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

2025 р.

Харківський національний університет радіоелектроніки

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

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Пилайкіну Євгену Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для проведення голосувань на основі блокчейну. Фронт-енд та мобільний застосунок
Затверджена наказом по університету від 19.05. 2025р. № 397 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 11.06.2025
3. Вихідні дані до роботи методи розробки клієнт-серверних додатків, сучасні підходи до проєктування користувацьких інтерфейсів, методи інтеграції з REST API, мови програмування та фреймворки для веб- та мобільної розробки (React, TypeScript, Kotlin, Jetpack Compose).
4. Перелік питань, що потрібно опрацювати в роботі
мета роботи, аналіз проблемної галузі і постановка задачі, опис вимог до програмної системи, опис використовуваних методів та алгоритмів, опис розробленої програмної системи, аналіз можливих застосувань, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	10.05-11.05.2025	*виконано*
2	Створення специфікації ПЗ	11.05-13.05.2025	*виконано*
3	Проектування ПЗ	13.05-16.05.2025	*виконано*
4	Розробка ПЗ	16.05-01.06.2025	*виконано*
5	Тестування ПЗ	01.06-03.06.2025	*виконано*
6	Оформлення пояснювальної записки	03.06-05.06.2025	*виконано*
7	Підготовка презентації та доповіді	05.06-06.06.2025	*виконано*
8	Попередній захист	06.06-08.06.2025	*виконано*
9	Нормоконтроль, рецензування	08.06-10.06.2025	*виконано*
10	Здача роботи у електронний архів	11.06.2025	*виконано*
11	Допуск до захисту у зав. кафедри	11.06.2025	*виконано*

Дата видачі завдання «04» « квітня » 2025р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. кафедри ПІ Ірина КИРИЧЕНКО
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи, 80 стор., 28 рис., 16 джерел.

ВЕБ-ДОДАТОК, ЕЛЕКТРОННІ ГОЛОСУВАННЯ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, КРИПТОГРАФІЯ, МОБІЛЬНА РОЗРОБКА, JETPACK COMPOSE, KOTLIN, REACT, TYPESCRIPT

Об'єкт розробки – клієнтська частина платформи для проведення електронних голосувань, що включає повнофункціональний веб-інтерфейс (React) та спеціалізований мобільний додаток (Kotlin/Jetpack Compose) для довірених осіб, які взаємодіють з розподіленою мікросервісною архітектурою на основі блокчейну.

Мета роботи – розробка та впровадження клієнтських компонентів (веб-інтерфейс та мобільний додаток), що забезпечують інтуїтивно зрозумілу та безпечну взаємодію користувачів з комплексною серверною інфраструктурою системи голосувань. Основний акцент – на абстрагуванні складності блокчейн-операцій та мікросервісної логіки для кінцевого користувача.

Технічна реалізація – використання стеку React/TypeScript для веб-платформи та Kotlin/Jetpack Compose для Android-додатку; інтеграція з мікросервісним API через API-шлюз; реалізація клієнтської логіки для взаємодії зі смарт-контрактами через backend; реалізація механізмів локального зберігання ключів та криптографічних операцій; розробка адаптивних інтерфейсів для різних ролей користувачів.

У результаті розробки створено повнофункціональний веб-інтерфейс, що надає доступ до всіх можливостей системи голосувань для звичайних та преміум-користувачів, та створено спеціалізований мобільний додаток для довірених осіб, призначений для виконання завдань верифікації. Обидва клієнтські рішення забезпечують зручний та безпечний доступ до функціональності децентралізованої платформи.

ABSTRACT

CRYPTOGRAPHY, ELECTRONIC VOTING, JAVASCRIPT, JETPACK COMPOSE, KOTLIN, MOBILE DEVELOPMENT, REACT, USER INTERFACE, WEB APPLICATION

Development object – the client-side of a platform for conducting electronic voting, comprising a full-featured web interface (React) and a specialized mobile application (Kotlin/Jetpack Compose) for trusted parties, which interacts with a distributed, blockchain-based microservice architecture.

Practice objective – development and implementation of client-side components (web interface and mobile application) that ensure an intuitive and secure user interaction with the complex server-side infrastructure of the voting system. The primary focus is on abstracting the complexity of blockchain operations and microservice logic for the end-user.

Technical implementation – utilization of the React/TypeScript stack for the web platform and Kotlin/Jetpack Compose for the Android application; integration with the microservice API through an API gateway; implementation of client-side logic to interact with smart contracts via the backend; implementation of mechanisms for local key storage and cryptographic operations; development of adaptive interfaces for different user roles.

As a result of development, a full-featured web interface was created, providing access to all features of the voting system for regular and premium users, and a specialized mobile application has been created for trusted parties, intended for performing verification tasks. Both client-side solutions provide convenient and secure access to the functionality of the decentralized platform.

ЗМІСТ

ВСТУП

Темою кваліфікаційної роботи є розробка клієнтської частини для системи електронних голосувань на блокчейні. Основне завдання – забезпечити зручний доступ до функціональності системи через повнофункціональний веб-інтерфейс та спеціалізований мобільний додаток для довірених осіб.

Актуальність роботи обумовлена зростанням потреби в надійних електронних системах голосування [1]. Традиційні методи мають проблеми з явкою, витратами та вразливістю до фальсифікацій. Пандемія COVID-19 та воєнні дії в Україні посилили необхідність у дистанційних механізмах прийняття рішень в умовах обмеженої мобільності. Електронні системи на основі блокчейну вирішують ці проблеми [2], але їх ефективність залежить від якості користувацького інтерфейсу.

Метою роботи є розробка клієнтських компонентів системи – повнофункціонального веб-інтерфейсу для всіх типів користувачів та спеціалізованого мобільного додатку для верифікаторів. Пріоритетом є створення інтуїтивного інтерфейсу, що абстрагує технічну складність мікросервісної архітектури та блокчейн-взаємодій, надаючи прості елементи управління [3].

Завдання включають аналіз існуючих рішень, проєктування архітектури клієнтських додатків, розробку компонентів інтерфейсу для різних ролей користувачів та реалізацію клієнтської частини механізмів авторизації з криптографічними ключами. Важливими аспектами є реалізація інтерфейсів для різних типів голосувань та інтеграція з серверним API [4].

В умовах воєнного стану система може стати важливим інструментом для забезпечення демократичних процесів на всіх рівнях. Блокчейн гарантує незмінність результатів, а криптографія захищає приватність, що критично за підвищених безпекових ризиків. Особливо актуальною є можливість організації голосувань для громадян, які перебувають у різних регіонах України чи за кордоном.

Практична значимість полягає у створенні інструменту для прийняття рішень у різних сферах. Система особливо корисна для громад у зоні бойових дій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Користувацький інтерфейс є критичним компонентом будь-якої інформаційної системи, оскільки саме через нього відбувається взаємодія людини з цифровими продуктами. У контексті систем електронного голосування, побудованих на блокчейн-технологіях, розробка інтуїтивно зрозумілого та функціонального інтерфейсу стає особливо важливою з огляду на складність базових технологій та необхідність забезпечення довіри користувачів [5].

Розвиток систем голосування історично відображає еволюцію технологій взаємодії з користувачем – від паперових бюлетенів до механічних машин для голосування, а згодом до електронних терміналів та онлайн-платформ. Із розвитком мобільних технологій та зростанням кількості смартфонів з'явилася можливість забезпечити доступ до голосувань буквально з кишені користувача. За даними GSMA Intelligence, станом на 2025 рік прогнозується, що понад 80% населення світу матиме доступ до смартфонів, що робить мобільні платформи ключовим каналом для реалізації електронної демократії [6].

Розглядаючи інтерфейси існуючих систем електронного голосування, можна виявити кілька важливих тенденцій та підходів. Традиційні цифрові системи голосування зазвичай мають простий, мінімалістичний інтерфейс з обмеженою функціональністю. Наприклад, система електронного голосування в Естонії, одного зі світових лідерів е-демократії, пропонує користувачам базовий інтерфейс з чіткою навігацією, орієнтований на максимальну простоту та доступність. Пріоритетом таких систем є забезпечення включеності всіх категорій громадян, незалежно від їхнього віку та цифрової грамотності.

З іншого боку, новітні системи голосування на блокчейні часто страждають від надмірної технічної складності інтерфейсів. Платформи, як-от Democracy Earth або Voatz, хоча й пропонують революційні технологічні рішення, часто вимагають від користувачів розуміння складних криптографічних понять [7]. Наприклад, для взаємодії з блокчейн-системою користувачам може знадобитися створення криптографічних ключів, розуміння процесів підписання транзакцій або взаємодія

зі смарт-контрактами. В даному проєкті ця складність абстрагується на рівні backend-сервісів, проте клієнтська частина повинна коректно обробляти API-запити та надавати користувачу зрозумілий зворотний зв'язок.

Особливим викликом для розробників інтерфейсів блокчейн-систем голосування є необхідність знайти баланс між прозорістю процесу (показати користувачу, що його голос дійсно зараховано) та простотою використання. Додаткову складність створює потреба в реалізації різних рівнів приватності, особливо для закритих голосувань, де анонімність учасників є критично важливою.

Мобільний аспект взаємодії з системами голосування заслуговує особливої уваги. У даному проєкті мобільний додаток має спеціалізовану роль – він призначений для довірених осіб (верифікаторів), які підтверджують право на участь у закритих голосуваннях. Це рішення зумовлене потребою в мобільності та безпеці верифікаційних процедур. У контексті України ця тенденція посилюється успіхом застосунку "Дія", який продемонстрував ефективність мобільного підходу до надання державних послуг.

Аналізуючи інтерфейси існуючих блокчейн-рішень для голосування, можна виділити кілька платформ, які пропонують різні підходи до взаємодії з користувачем. Follow My Vote фокусується на візуалізації процесу голосування та відстеження голосів, дозволяючи користувачам буквально "слідкувати" за своїм голосом у системі. Horizon State робить акцент на корпоративному управлінні та пропонує більш складний інтерфейс для організації голосувань з різноманітними параметрами. Vocdoni, побудована на базі Ethereum, надає інструменти для створення децентралізованих голосувань та пропонує відкритий API для інтеграції з іншими системами.

Варто зазначити, що вимоги до користувацького інтерфейсу систем голосування значно різняться залежно від контексту використання. Для державних виборів пріоритетом є безпека, надійність та доступність для всіх категорій громадян. Для корпоративних голосувань важливішими можуть бути гнучкість налаштувань та інтеграція з існуючими корпоративними системами. Для

громадських ініціатив та неформальних опитувань ключовою є швидкість розгортання та простота використання.

У контексті сучасних соціальних викликів, особливо в Україні, де значна частина населення мусила покинути свої домівки через війну, мобільні рішення для голосування набувають додаткової актуальності. Вони дозволяють забезпечити участь у демократичних процесах громадян, які знаходяться в різних регіонах країни або за кордоном, що особливо важливо для прийняття рішень на рівні місцевих громад або для загальнонаціональних референдумів.

1.2 Виявлення та вирішення проблем

Розробка клієнтської частини системи електронних голосувань на основі блокчейну стикається з рядом специфічних проблем, які вимагають комплексного підходу до їх вирішення. Аналіз предметної області дозволив виявити ключові проблеми, пов'язані з інтерфейсом та взаємодією користувачів із системою.

Першою фундаментальною проблемою є складність технічних концепцій блокчейну та криптографії для пересічного користувача [8]. Більшість існуючих блокчейн-рішень вимагають від користувачів розуміння таких понять як приватні ключі, підписання транзакцій, газ (в Ethereum), підтвердження транзакцій тощо. В даному проєкті ця проблема вирішується на архітектурному рівні: всі складні операції інкапсульовані на стороні backend. Завдання клієнтської частини – надати користувачу знайомі метафори та інтерфейсні патерни. Наприклад, замість вимагати від користувача підписування транзакцій вручну, система може автоматизувати цей процес, представляючи його як звичайне "підтвердження" дії.

Другою важливою проблемою є баланс між прозорістю процесу та його зрозумілістю. З одного боку, блокчейн-технологія надає можливість повної прозорості всіх операцій, що є критичним для довіри до системи голосування [9]. З іншого боку, надання користувачу всієї цієї інформації може призвести до перевантаження інтерфейсу та когнітивного перевантаження самого користувача. Рішення полягає у створенні багаторівневого інтерфейсу, де базовий рівень надає лише найнеобхіднішу інформацію у зрозумілому форматі (наприклад, статус "Голос

зараховано"), а додаткові деталі (хеш транзакції, номер блоку) доступні "за кліком" для зацікавлених користувачів.

Третя проблема пов'язана з адаптацією інтерфейсу для різних типів голосувань та ролей користувачів. Система повинна підтримувати як прості публічні опитування, так і складні багатоетапні голосування з різними рівнями верифікації. При цьому інтерфейс для звичайного учасника (веб) суттєво відрізняється від інтерфейсу для довіреної особи (мобільний додаток). Вирішення цієї проблеми можливе через впровадження адаптивного інтерфейсу, який динамічно змінюється залежно від контексту використання та ролі користувача. Технології React для веб та Jetpack Compose для Android дозволяють ефективно реалізувати такий підхід через компонентну архітектуру та реактивне оновлення інтерфейсу.

Четверта проблема стосується мобільної адаптації. У даному проєкті мобільний додаток має чітко визначену, вузькоспеціалізовану функцію – верифікація учасників. Це рішення дозволяє уникнути проблем з обмеженими ресурсами мобільних пристроїв при взаємодії з блокчейном, оскільки всі складні операції виконуються на сервері, а додаток лише обмінюється даними через API (зчитує QR-код, відправляє запит на підтвердження).

П'ята проблема пов'язана з обробкою помилок та зворотним зв'язком. Взаємодія з блокчейном характеризується затримками (час підтвердження транзакцій), потенційними відмовами та необхідністю повторних спроб. На відміну від традиційних клієнт-серверних додатків, де відповідь зазвичай отримується миттєво, клієнтська частина повинна вміти асинхронно обробляти довготривалі операції. Вирішення цієї проблеми вимагає створення прозорої системи сповіщень та індикації стану операцій, яка інформує користувача про поточний стан його дій та очікуваний час завершення.

Шоста проблема стосується безпеки користувацьких даних та приватних ключів. Традиційний підхід до роботи з блокчейном вимагає від користувача управління приватними ключами, що створює ризики їх втрати або компрометації. У даній архітектурі приватні ключі генеруються та зберігаються на стороні сервера

в зашифрованому вигляді, що перекладає відповідальність з користувача на систему. Клієнтська частина повинна забезпечувати безпечну передачу даних для входу та підтвердження операцій, використовуючи JWT-токени.

Сьома проблема пов'язана з використанням біометричної аутентифікації. Сучасні мобільні пристрої підтримують різноманітні біометричні технології. Для мобільного додатку верифікатора це є пріоритетною функцією для захисту доступу до функціоналу підтвердження особистості. Вирішення цієї проблеми вимагає ретельного проектування архітектури безпеки, де біометричні дані використовуються лише для локальної аутентифікації на пристрої користувача, без передачі їх на сервер.

Восьма проблема стосується оффлайн-режиму роботи. Не всі користувачі мають стабільний доступ до інтернету, особливо в умовах України. Для веб-інтерфейсу це означає можливість кешування даних про голосування для перегляду. Для мобільного додатку верифікатора це може означати можливість сканування QR-коду та збереження запиту на верифікацію для подальшої відправки при відновленні з'єднання.

1.3 Постановка задачі

1.3.1 Основні задачі розробки

Для вирішення вищезазначених проблем необхідно розробити клієнтську частину для універсальної платформи електронних голосувань. Клієнтське програмне забезпечення повинно забезпечити інтуїтивно зрозумілий та безпечний доступ до складної мікросервісної та блокчейн-інфраструктури, абстрагуючи її технічні деталі від кінцевих користувачів. Система має підтримувати як веб-платформу для всіх типів користувачів, так і спеціалізований мобільний додаток для виконання верифікаційних процедур довіреними особами.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Розробити архітектуру клієнтської частини, що включає повнофункціональний веб-додаток на React та спеціалізований мобільний додаток

на Kotlin/Jetpack Compose, забезпечивши їх ефективну взаємодію з єдиним backend-API.

2. Створити адаптивні користувацькі інтерфейси для різних ролей (стандартний користувач, преміум-користувач, довірена особа, адміністратор), що динамічно відображають доступний функціонал та приховують нерелевантну інформацію.

3. Реалізувати механізми безпечної інтеграції з API, забезпечивши коректну обробку асинхронних запитів до мікросервісів, управління станом додатків (state management) та обробку помилок, що надходять від серверної частини.

4. Імплементувати клієнтську логіку для криптографічних протоколів, зокрема для генерації запитів на отримання сліпих підписів та відправки доказів з нульовим розголошенням (ZKP) через API, зберігаючи при цьому безпеку та локальність критичних операцій на пристрої користувача.

5. Розробити спеціалізований мобільний додаток для довірених осіб, що забезпечує функціонал для сканування QR-кодів, перегляду документів та підтвердження верифікації учасників закритих голосувань.

Головна задача розробки полягає у створенні клієнтського програмного забезпечення, що складається з веб-інтерфейсу та спеціалізованого мобільного додатку, яке забезпечить доступність блокчейн-технологій для широкого кола користувачів. Клієнтська частина повинна абстрагувати користувачів від складності взаємодії з мікросервісною архітектурою та блокчейном, одночасно зберігаючи всі переваги децентралізованої системи, такі як безпека, прозорість та незмінність даних.

Розробка веб-інтерфейсу на базі React є однією з ключових задач, оскільки саме через веб-браузер більшість користувачів будуть взаємодіяти з системою. Веб-інтерфейс повинен забезпечувати адаптивність для різних розмірів екранів, високу швидкість відгуку та підтримку всіх необхідних функціональних можливостей. Архітектура веб-додатку повинна бути побудована з урахуванням сучасних практик розробки, забезпечуючи чітке розділення бізнес-логіки та представлення,

модульність компонентів та ефективну роботу з асинхронними операціями через взаємодію з серверним API.

Паралельно з веб-інтерфейсом необхідно розробити спеціалізований мобільний додаток для Android з використанням Kotlin та Jetpack Compose, призначений виключно для довірених осіб (верифікаторів). Мобільний додаток повинен забезпечувати функціонал для верифікації учасників закритих голосувань, включаючи сканування QR-кодів, перегляд документів та підтвердження особи. Особлива увага має бути приділена безпеці, оптимізації використання мережевих ресурсів та можливості роботи в умовах нестабільного з'єднання.

Важливою задачею є імплементація механізмів інтеграції з backend-API, що забезпечать безпечну та ефективну взаємодію клієнтської частини з мікросервісною архітектурою. Це включає обробку відповідей від API, управління станом додатків та реалізацію клієнтської логіки для ініціації складних операцій, таких як голосування чи верифікація, які виконуються на сервері. Криптографічні операції, що вимагають участі користувача (наприклад, генерація запиту на сліпий підпис), повинні виконуватися локально на пристрої для забезпечення безпеки.

У рамках розробки клієнтської частини потрібно реалізувати інтерфейси, які працюють з криптографічними протоколами, імплементованими на backend. Зокрема, це стосується механізмів сліпих підписів та доказів з нульовим розголошенням (ZKP). Задача клієнта — коректно сформувати запит до API та обробити його результат, приховуючи складність цих операцій від кінцевого користувача, щоб він міг зосередитися на суті голосування.

1.3.2 Особливості розробки інтерфейсу користувача

Розробка інтуїтивно зрозумілих інтерфейсів для різних ролей користувачів та різних типів голосувань є критично важливою задачею. Інтерфейси повинні приховувати технічну складність блокчейну, при цьому забезпечуючи повну функціональність системи. Ключовим принципом дизайну повинна бути "прогресивна розкриття" інформації – надання користувачеві спочатку лише необхідних деталей з можливістю отримання додаткової інформації за бажанням.

Для кожної з ключових ролей користувачів (стандартний користувач, преміум-користувач, довірена особа, адміністратор) необхідно розробити специфічні інтерфейси, що відображають їхні потреби та можливості. Стандартний користувач повинен мати простий доступ до голосувань, можливість участі та перевірки свого голосу. Преміум-користувач повинен мати інструменти для створення та управління закритими голосуваннями. Мобільний додаток довіреної особи потребує спеціалізованого інтерфейсу для верифікації учасників, а адміністратор – для загального управління системою.

Особливу увагу слід приділити процесу створення голосування, який повинен проводити користувача через всі необхідні етапи, від визначення базових параметрів до налаштування питань та варіантів відповідей. Цей процес має бути реалізований у формі покрокового майстра (wizard) з валідацією даних на кожному етапі. Це особливо важливо для закритих голосувань, де потрібно налаштувати критерії участі та призначити довірених осіб.

Для процесу голосування важливо забезпечити чітке розуміння користувачем всіх аспектів – питань, варіантів відповідей, терміну голосування та механізму підтвердження. Інтерфейс повинен забезпечувати можливість ознайомлення з питаннями до початку процесу голосування, а також надавати чітке підтвердження після успішного зарахування голосу. Для закритих голосувань необхідно візуалізувати процес отримання верифікації та використання ZKP-токенів у формі, зрозумілій для нетехнічного користувача.

Візуалізація результатів голосування вимагає створення інформативних та інтерактивних графіків та діаграм, що дозволять користувачеві швидко зрозуміти підсумки. Для відкритих голосувань результати можуть оновлюватися в реальному часі, тоді як для закритих – бути доступними лише після завершення. Користувачі повинні мати можливість перевірити включення свого голосу в загальні результати без порушення анонімності.

1.3.3 Технічні та нефункціональні аспекти розробки

Забезпечення захисту користувацьких даних та управління криптографічними операціями є ключовим аспектом розробки. Оскільки приватні ключі користувачів управляються на стороні сервера, основна відповідальність клієнтської частини полягає в безпечній передачі даних аутентифікації (JWT) та запитів, що ініціюють криптографічні операції. Для веб-додатку слід використовувати HTTPS та безпечне зберігання токенів (наприклад, у HttpOnly cookie), а для мобільного додатку довіреної особи – платформенний Android KeyStore для зберігання ключів аутентифікації. Система повинна мінімізувати ризики компрометації сесійних токенів та надавати механізми відновлення доступу у випадку втрати пристрою.

Створення ефективної системи оповіщень та індикації стану є важливим для забезпечення позитивного користувацького досвіду, особливо з урахуванням специфіки асинхронної взаємодії з мікросервісами, де операції можуть мати затримку. Користувачі повинні отримувати чіткі сповіщення про статус своїх дій (наприклад, "Голос відправлено на обробку", "Голос успішно зараховано в блокчейн"), нові голосування та результати.

Забезпечення доступності інтерфейсів для користувачів з різними обмеженнями є не лише технічною, але й соціальною необхідністю для системи голосування. Це включає підтримку програм читання з екрану через використання семантичної розмітки та ARIA-атрибутів, забезпечення альтернативних методів введення, достатній контраст елементів та можливість навігації за допомогою клавіатури. Мобільний додаток повинен інтегруватися з платформенними сервісами доступності, такими як TalkBack для Android.

Продуктивність та оптимізація ресурсів є важливими аспектами розробки. Необхідно впровадити механізми кешування даних, отриманих від API, для зменшення мережевого трафіку та пришвидшення відгуку інтерфейсу. Для веб-додатку важливо оптимізувати розмір пакетів JavaScript, використовувати код-сплітінг та ліниве завантаження компонентів. Для мобільного додатку критичною

є оптимізація використання пам'яті та CPU, особливо при обробці зображень документів під час верифікації.

Реалізація підтримки офлайн-режиму дозволить користувачам продовжувати роботу з додатком навіть за відсутності постійного інтернет-з'єднання. Веб-клієнт повинен кешувати інформацію про доступні голосування для перегляду. Мобільний додаток верифікатора повинен мати можливість кешувати отримані дані про користувача та зберігати рішення про верифікацію для подальшої синхронізації з сервером. Ця функціональність особливо важлива в умовах нестабільного інтернет-з'єднання, що може бути актуальним в деяких регіонах України.

Реалізація поставлених задач дозволить створити клієнтську частину системи електронних голосувань, яка надасть користувачам зручний доступ до технологічно складного серверного рішення на базі мікросервісів та блокчейну. Система сприятиме розширенню інструментів електронної демократії в Україні, особливо в умовах воєнного стану та вимушеного переміщення значної частини населення.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Загальна концепція системи

Клієнтська частина системи електронних голосувань представляє собою комплекс програмних рішень, що забезпечує користувацький інтерфейс для взаємодії з серверною інфраструктурою, яка використовує блокчейн. Система складається з двох основних клієнтських компонентів – повнофункціонального веб-додатку для використання через браузер та спеціалізованого мобільного додатку для платформи Android, призначеного для довірених осіб.

Система підтримує два основних типи голосувань – відкриті (доступні всім зареєстрованим користувачам) та закриті (з обмеженим доступом через механізм верифікації) [10]. Відкриті голосування можуть створюватися будь-яким користувачем та не мають обмежень щодо участі. Закриті голосування створюються преміум-користувачами та вимагають верифікації учасників через довірених осіб з використанням механізму сліпих підписів.

Система підтримує різні ролі користувачів:

- Стандартний користувач – може брати участь у відкритих голосуваннях, створювати відкриті голосування, брати участь у закритих голосуваннях після отримання верифікації;
- Преміум-користувач – додатково може створювати закриті голосування та призначати довірених осіб;
- Довірена особа – може верифікувати користувачів для участі в конкретному закритому голосуванні;
- Адміністратор – управляє глобальними налаштуваннями системи та має доступ до розширеної аналітики.

Система забезпечує повний життєвий цикл голосування – від створення та налаштування, через процес верифікації учасників (для закритих голосувань), безпосереднє проведення голосування, до підрахунку та візуалізації результатів. На всіх етапах забезпечується прозорість процесу через використання блокчейну для незмінної фіксації ключових подій.

2.2 Функціональні вимоги

2.2.1 Управління обліковими записами

Система повинна забезпечувати комплексне управління обліковими записами користувачів, включаючи наступні функціональні можливості:

- система повинна забезпечувати реєстрацію користувачів з використанням електронної пошти та паролю, або через інтеграцію з зовнішніми системами аутентифікації (BankID, Дія);
- система повинна підтримувати різні ролі користувачів з відповідними правами доступу;
- користувачі повинні мати можливість редагувати свій профіль, включаючи особисту інформацію та налаштування приватності;
- система повинна забезпечувати безпечне зберігання та управління криптографічними ключами користувачів, які використовуються для взаємодії з блокчейном;
- система повинна надавати механізми відновлення доступу у випадку втрати паролю або пристрою;
- користувачі повинні мати можливість підвищити статус облікового запису до преміум-рівня через визначений системою механізм.

2.2.2 Створення та управління голосуваннями

Для ефективної організації та проведення голосувань система повинна реалізовувати наступні функціональні можливості:

- стандартні користувачі повинні мати можливість створювати відкриті голосування, вказуючи назву, опис, часові рамки та питання з варіантами відповідей;
- преміум-користувачі повинні мати можливість створювати закриті голосування з додатковими параметрами, такими як критерії участі та список довірених осіб;

- система повинна підтримувати різні типи питань: вибір одного варіанту, вибір кількох варіантів, ранжування варіантів;
- система повинна забезпечувати можливість планування голосувань на майбутнє, вказуючи дату та час початку та завершення;
- створювачі голосувань повинні мати можливість редагувати параметри голосування до його початку, а також скасовувати голосування, якщо воно ще не розпочалося;
- для закритих голосувань система повинна забезпечувати механізм призначення довірених осіб та управління їхніми правами.

2.2.3 Верифікація для закритих голосувань

Для забезпечення легітимності та анонімності закритих голосувань система повинна реалізовувати спеціальні механізми верифікації:

- довірені особи повинні мати спеціальний інтерфейс для верифікації користувачів, які бажають взяти участь у закритому голосуванні;
- система повинна реалізовувати протокол сліпих підписів для забезпечення анонімності голосів при збереженні контролю над правом участі;
- користувачі повинні мати можливість отримати верифікацію від будь-якої довіреної особи, призначеної для відповідного голосування;
- система повинна забезпечувати механізм доказів з нульовим розголошенням (ZKP) для підтвердження права на участь без розкриття особистості;
- процес верифікації повинен бути документований у блокчейні для аудиту, але без розкриття зв'язку між верифікатором, верифікованим користувачем та їхнім подальшим голосом.

2.2.4 Участь у голосуваннях

Участь користувачів у голосуваннях є ключовим аспектом системи, який повинен забезпечувати наступні функціональні можливості:

- користувачі повинні мати можливість переглядати список доступних голосувань з фільтрацією за різними критеріями (статус, тип, категорія);
- для кожного голосування система повинна надавати детальну інформацію про його параметри, питання та варіанти відповідей;
- користувачі повинні мати можливість голосувати в активних голосуваннях, вибираючи варіанти відповідей відповідно до типу питання;
- система повинна забезпечувати підтвердження успішного зарахування голосу з можливістю його подальшої верифікації в блокчейні;
- користувачі не повинні мати можливості змінити свій голос після його підтвердження;
- система повинна запобігати повторному голосуванню одного користувача в рамках одного голосування.

2.2.5 Відображення результатів

Прозорість та доступність результатів голосувань є важливим аспектом системи, який повинен забезпечуватися наступними функціональними можливостями:

- система повинна забезпечувати візуалізацію результатів голосування в реальному часі або після його завершення, залежно від налаштувань;
- користувачі повинні мати можливість перевірити включення свого голосу в загальні результати без порушення анонімності;
- для закритих голосувань система повинна надавати статистику щодо кількості верифікованих користувачів та фактичної явки.

2.3 Нефункціональні вимоги

2.3.1 Вимоги до інтерфейсу та зручності використання

Для забезпечення позитивного користувацького досвіду система повинна відповідати наступним вимогам:

- інтерфейс системи повинен бути інтуїтивно зрозумілим, не вимагаючи від користувачів спеціальних знань про блокчейн або криптографію;

- веб-інтерфейс повинен бути адаптивним, забезпечуючи зручне використання на пристроях з різними розмірами екранів (від смартфонів до десктопів);
- мобільний додаток повинен дотримуватися принципів Material Design та використовувати нативні патерни взаємодії платформи Android;
- система повинна забезпечувати чіткий зворотний зв'язок про стан операцій, особливо при взаємодії з блокчейном, де транзакції можуть мати затримку;
- критичні дії користувача повинні вимагати підтвердження для запобігання випадковим помилкам.

2.3.2 Вимоги до продуктивності та масштабованості

Для забезпечення швидкої та надійної роботи система повинна відповідати наступним вимогам до продуктивності:

- веб-додаток повинен забезпечувати час відгуку не більше 1 секунди для базових операцій, не пов'язаних з блокчейном;
- для операцій з блокчейном система повинна надавати чітку індикацію прогресу та очікуваного часу завершення;
- веб-додаток повинен працювати в останніх версіях популярних браузерів (Chrome, Firefox, Safari, Edge).

2.3.3 Вимоги до безпеки та приватності

Безпека та приватність є критично важливими аспектами системи електронних голосувань, що забезпечуються наступними вимогами:

- паролі користувачів повинні зберігатися з використанням сучасних алгоритмів хешування з сіллю;
- криптографічні ключі для взаємодії з блокчейном повинні зберігатися в зашифрованому вигляді на пристроях користувачів, з використанням апаратного захисту, якщо доступно;

- система повинна включати захист від типових веб-вразливостей (XSS, CSRF, SQL-ін'єкції);
- мобільний додаток повинен запобігати витоку конфіденційних даних при створенні скріншотів та в режимі багатозадачності;
- система повинна забезпечувати анонімність голосування, не дозволяючи відслідкувати зв'язок між користувачем та його голосом.

2.3.4 Вимоги до надійності та стійкості

Для забезпечення стабільної роботи в різних умовах система повинна відповідати наступним вимогам до надійності:

- система повинна зберігати стан незавершених операцій для можливості відновлення після збоїв;
- мобільний додаток повинен забезпечувати базову функціональність навіть за відсутності інтернет-з'єднання (перегляд кешованих даних, підготовка голосів для пізнішої синхронізації);
- система повинна коректно обробляти ситуації відмови або затримки блокчейн-операцій;
- система повинна регулярно створювати резервні копії критичних даних, що не зберігаються в блокчейні;
- система повинна забезпечувати механізми відновлення доступу користувачів у випадку втрати пристрою або ключів.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Для проєктування клієнтської частини системи електронних голосувань на базі блокчейну був використаний набір UML діаграм, які описують різні аспекти системи – від варіантів використання до структури класів та взаємодій між компонентами.

Діаграма варіантів використання (див. рис. 3.1) демонструє основні функціональні можливості, доступні користувачам через клієнтські додатки. На діаграмі представлено ключових акторів:

- Стандартний користувач;
- Преміум-користувач (як його підтип);
- Довірена особа (Верифікатор);
- Адміністратор.

Стандартні користувачі через веб-інтерфейс можуть реєструватися, переглядати голосування та брати участь у відкритих. Преміум-користувачі додатково отримують можливість створювати закриті голосування. Довірені особи використовують спеціалізований мобільний додаток для верифікації учасників. Адміністратор, через відповідний веб-інтерфейс, управляє системою. Ця діаграма слугує основою для проєктування інтерфейсів, чітко розмежовуючи сценарії для різних ролей.

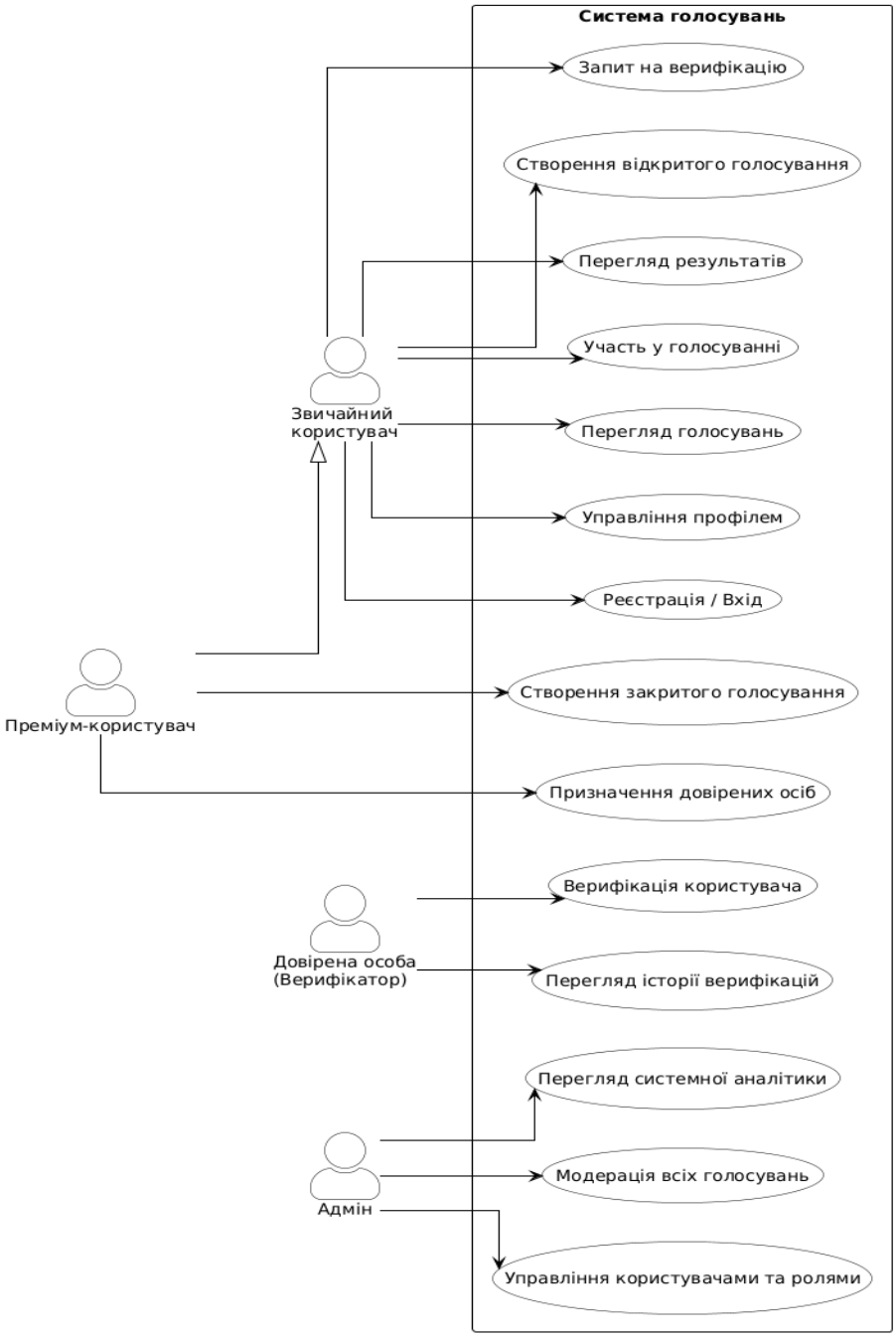


Рисунок 3.1 – Діаграма варіантів використання (Use Case Diagram) (рисунок виконаний самостійно)

Для детального розуміння процесу участі в закритому голосуванні розроблено діаграму послідовності (див. рис. 3.2). Вона ілюструє, як клієнтський додаток (веб-інтерфейс) взаємодіє з серверними сервісами. Ключові криптографічні операції (створення ZKP-доказу) виконуються на клієнті, що забезпечує безпеку. Проте сама взаємодія з блокчейном (випуск токена, запис

голосу) інкапсульована на стороні backend, з яким клієнт комунікує через API. Це дозволяє абстрагувати складність блокчейн-протоколів від користувача.

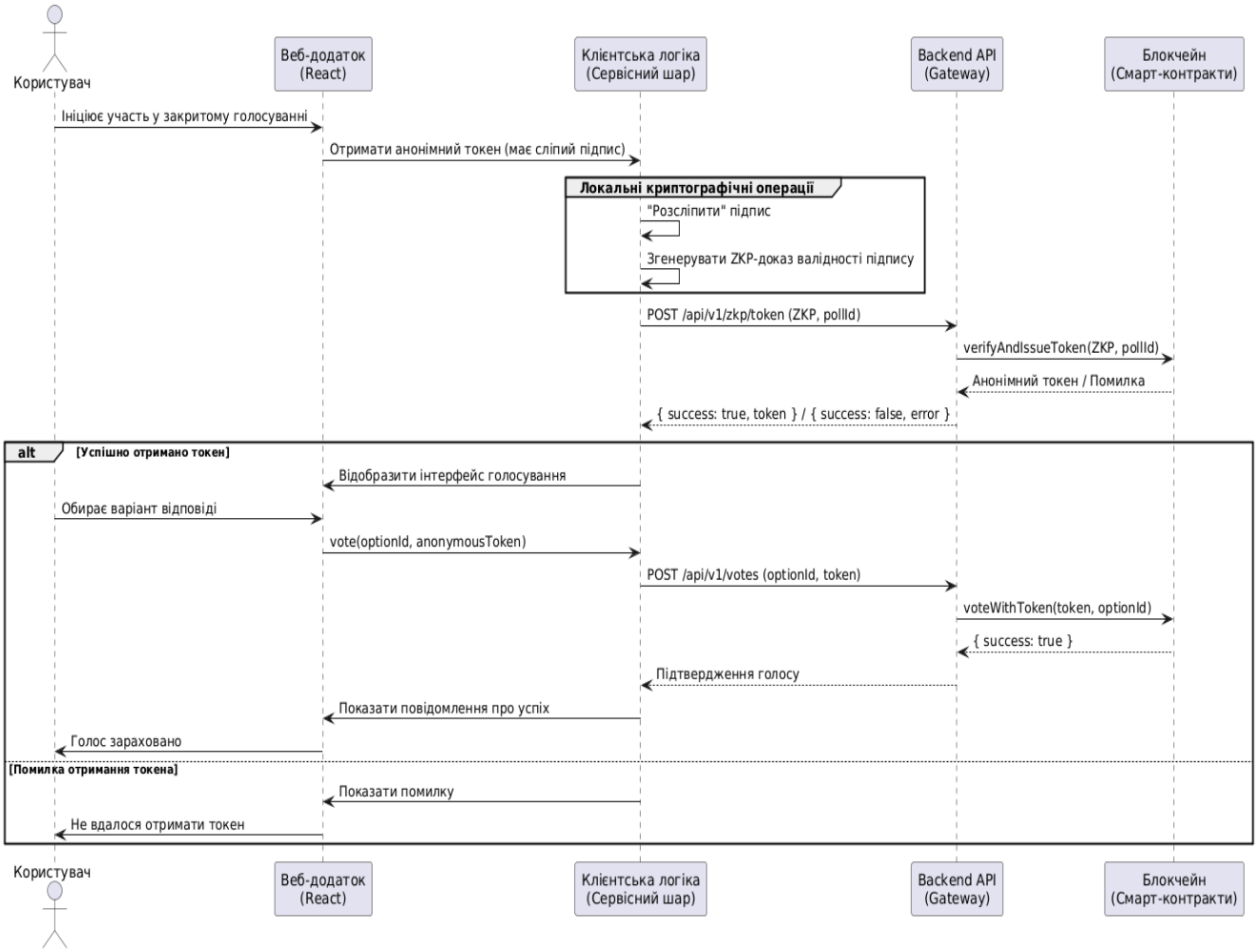


Рисунок 3.2 – Діаграма послідовності (Sequence Diagram) (рисунок виконаний самостійно)

Процес створення закритого голосування преміум-користувачем відображено на діаграмі активності (див. рис. 3.3). Діаграма показує покроковий сценарій, реалізований у веб-інтерфейсі у вигляді майстра (wizard). Користувач послідовно заповнює інформацію, додає варіанти відповідей та призначає довірених осіб. Після фінального підтвердження клієнтська логіка формує єдиний API-запит до backend, який вже відповідає за створення відповідних записів у базі даних та розгортання смарт-контракту. Це демонструє, як складний бізнес-процес декомпозується на прості кроки для користувача.

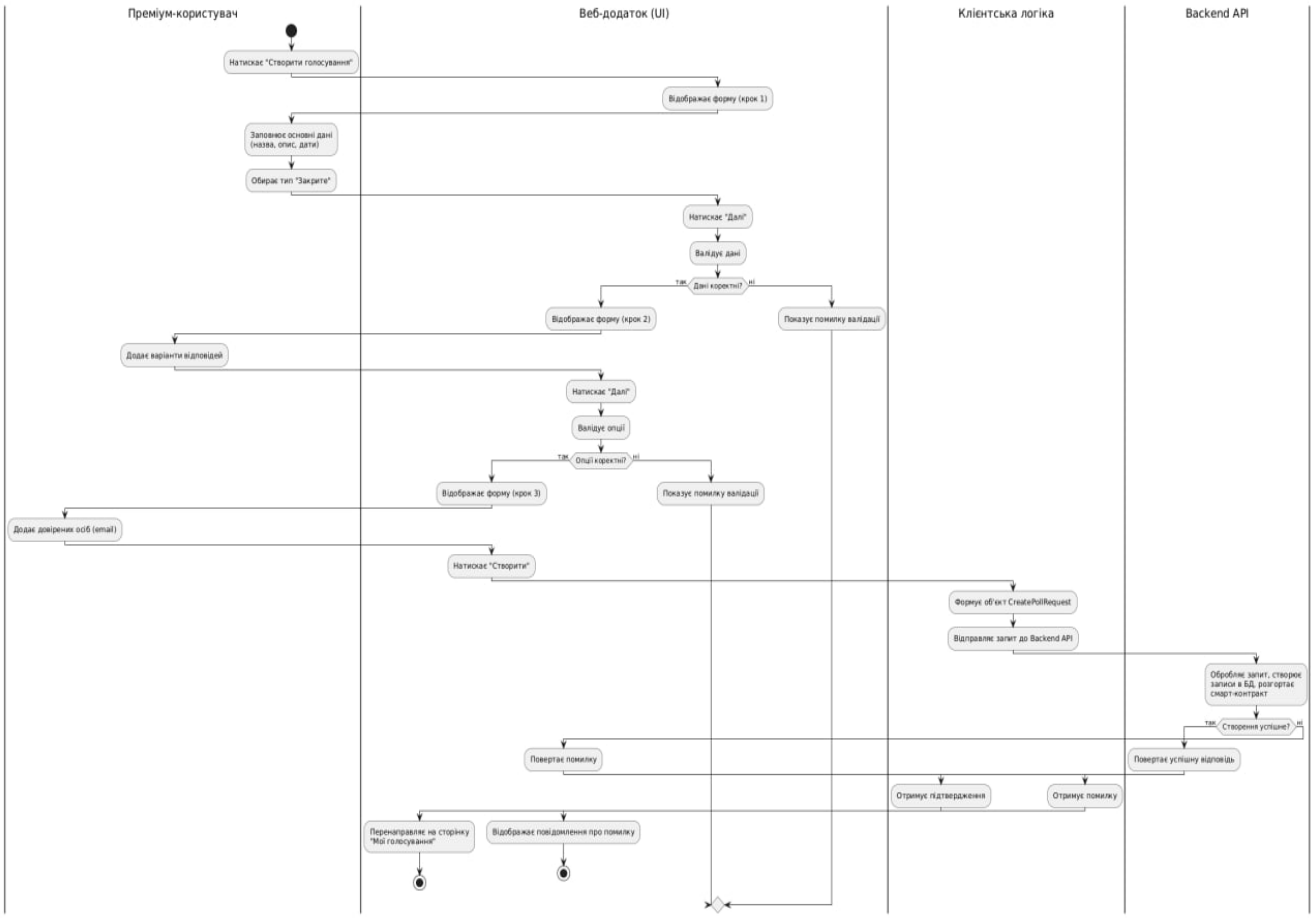


Рисунок 3.3 – Діаграма активності (Activity Diagram)) (рисунок виконаний самостійно)

Структурною основою системи є діаграма класів (див. рис. 3.4), яка відображає основні сутності системи та їх взаємозв'язки. Ця діаграма включає класи для роботи з користувачами (User), голосуваннями (Voting, OpenVoting, ClosedVoting), питаннями та варіантами відповідей (Question, Option), а також сервісні класи для роботи з криптографією (BlindSignatureService, ZKProofService) та блокчейном (BlockchainService). Особливу увагу приділено класам, що реалізують функціональність клієнтської частини – UIController, VotingRepository тощо. Важливими аспектами є наслідування (OpenVoting та ClosedVoting наслідують базовий клас Voting) та композиція (Voting містить Question, які, у свою чергу, містять Option). Діаграма класів представляє статичну структуру системи і є

основою для розробки як веб-інтерфейсу на React, так і мобільного додатку на Kotlin з Jetpack Compose.

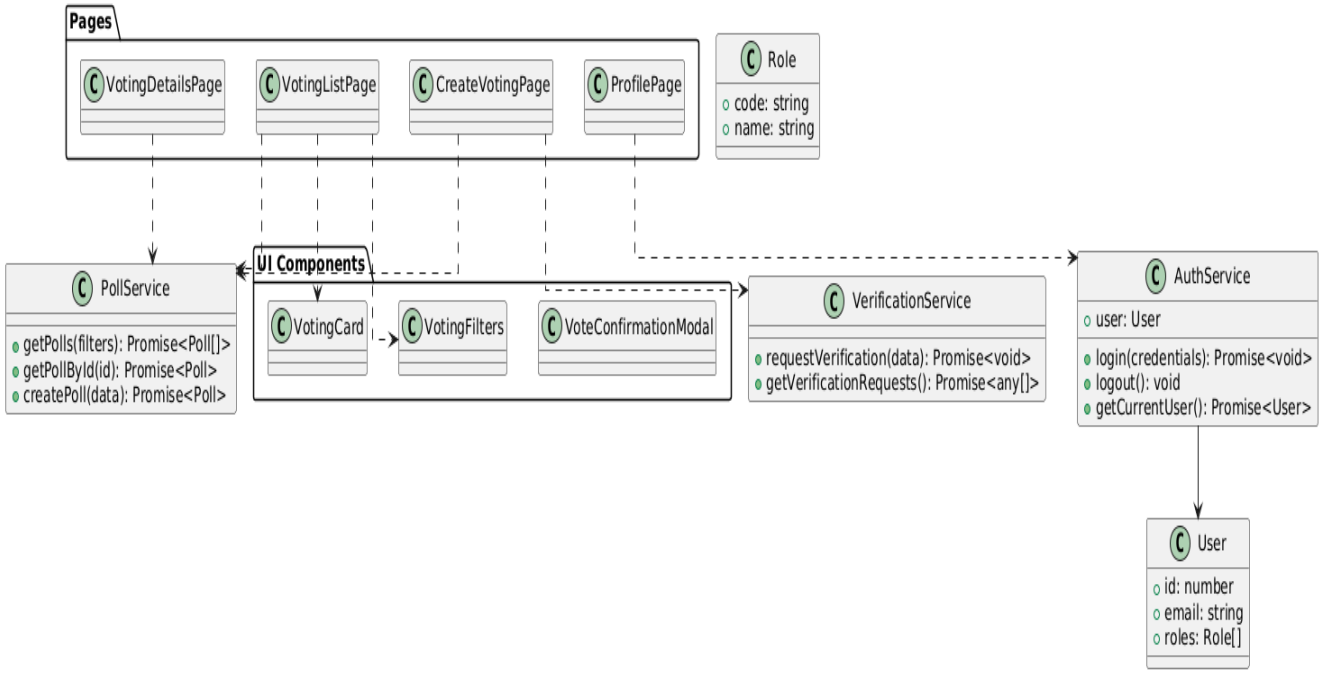


Рисунок 3.4 – Діаграма класів (Class Diagram)) (рисунок виконаний самостійно)

Для розуміння життєвого циклу голосування розроблена діаграма станів (див. рис. 3.5), яка демонструє різні стани, через які проходить голосування в системі. Після створення в системі голосування одразу отримує статус Активне (Active), але фактична можливість голосувати з'являється лише після настання вказаного часу початку. До цього моменту організатор може скасувати голосування, перевівши його у кінцевий стан Скасоване (Cancelled). Коли час голосування завершується, автоматизований процес (cron job) аналізує результати та переводить голосування в один із двох фінальних станів: Завершене (Ended), якщо воно відбулося успішно, або Провалене (Failed), наприклад, якщо не було досягнуто мінімального порогу явки.

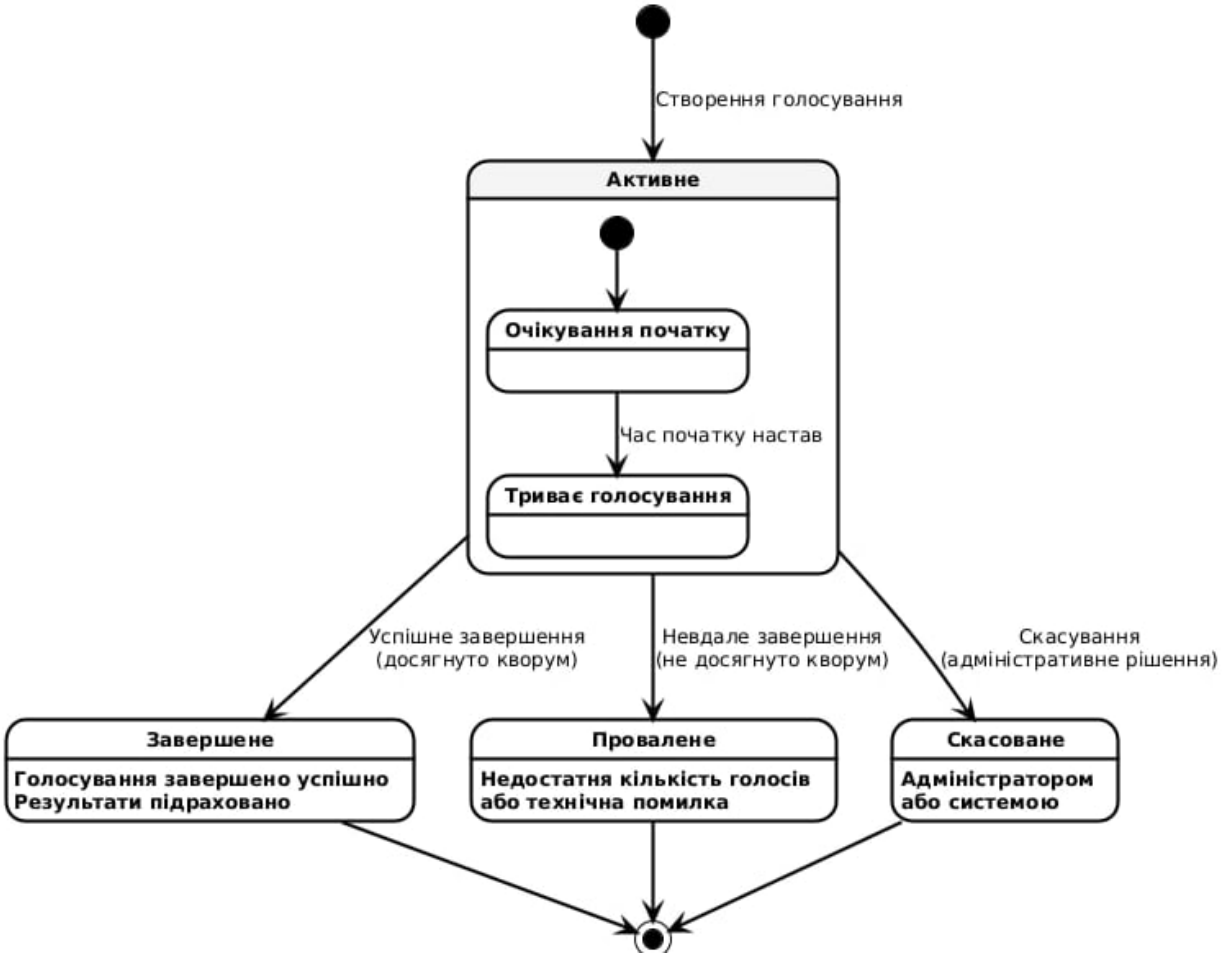


Рисунок 3.5 – Діаграма стану голосування (State Diagram) (рисунок виконаний самостійно)

3.2 Проєктування архітектури ПЗ

Архітектура клієнтської частини системи побудована на основі сучасних підходів до розробки веб- та мобільних додатків. Діаграма компонентів (див. рис. 3.6) відображає її основні складові та їх взаємодію з серверною частиною.

Клієнтська частина включає два основних компоненти – Веб-додаток (React) та Мобільний додаток (Kotlin). Обидва компоненти взаємодіють з єдиним Backend API (шлюзом), що забезпечує уніфікований доступ до мікросервісів. Криптографічні операції, що потребують участі користувача (генерація ZKP-доказів), інкапсульовані в компоненті Клієнтська логіка та сервіси. Це забезпечує повторне використання логіки та безпеку, оскільки приватні ключі не покидають пристрій. Взаємодія з блокчейном відбувається опосередковано через backend, що підвищує продуктивність та захищеність клієнтських додатків.

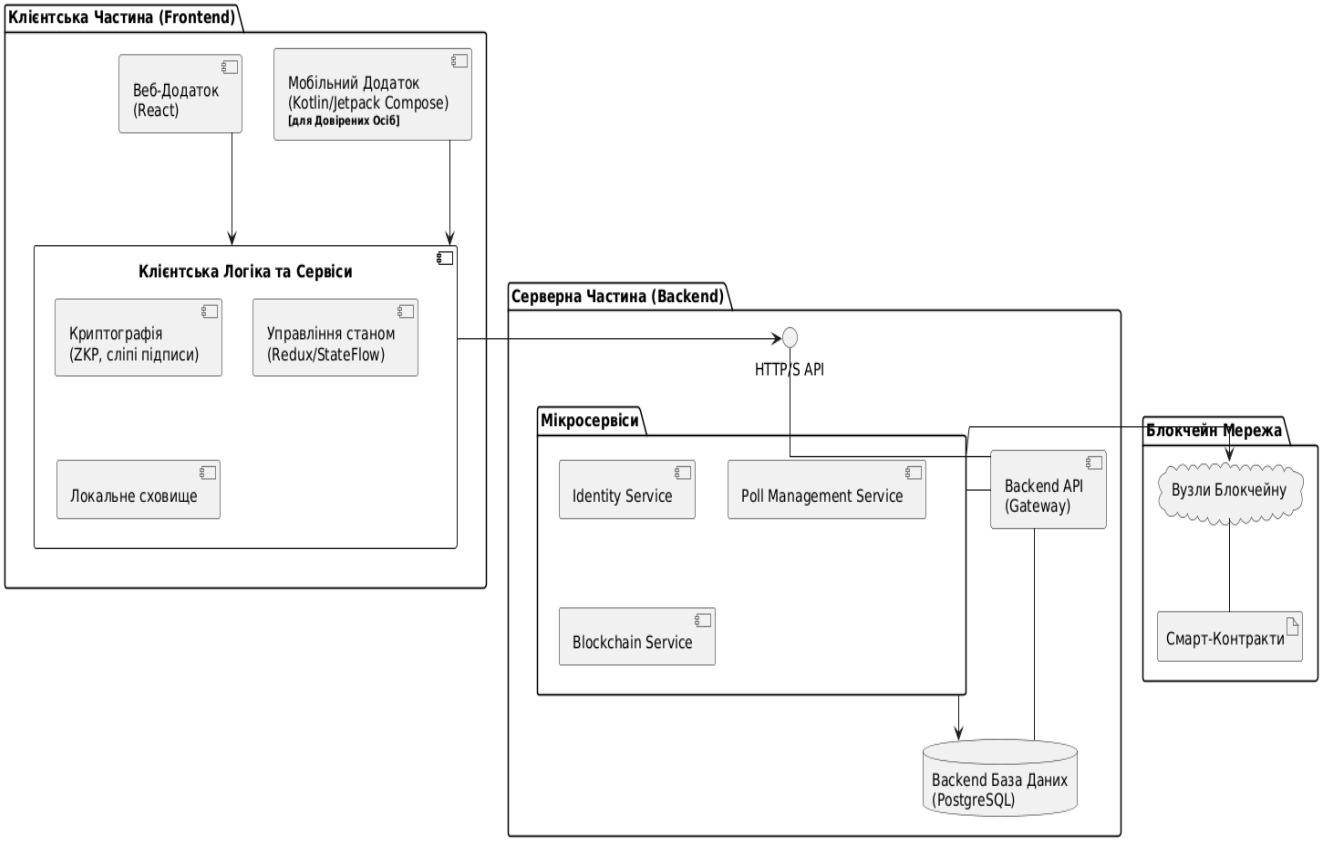


Рисунок 3.6 – Діаграма компонентів (Component Diagram) (рисунок виконаний самостійно)

Для розуміння фізичного розгортання системи розроблена діаграма розгортання (див. рис. 3.7). Вона демонструє, як компоненти системи розміщуються на фізичній інфраструктурі. Клієнтські додатки (веб-браузер та Android-пристрій) взаємодіють через HTTPS з серверами додатків, де розміщено Backend API. Серверна частина, у свою чергу, взаємодіє з базою даних та вузлами блокчейн-мережі. Важливою особливістю є відсутність прямої взаємодії клієнтських додатків з блокчейн-мережею. Уся комунікація відбувається через захищений Backend API, що значно спрощує клієнтську логіку, підвищує безпеку та дозволяє оптимізувати трафік, що є критичним для мобільних пристроїв.

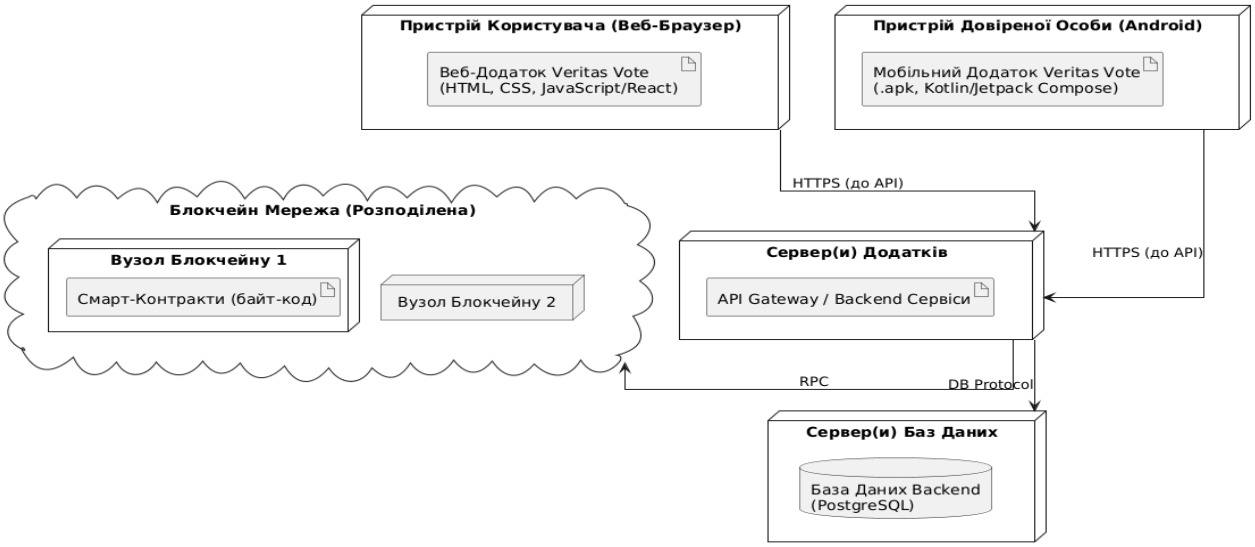


Рисунок 3.7 – Діаграма розгортання (Deployment Diagram) (рисунок виконаний самостійно)

Архітектура клієнтської частини системи електронних голосувань спроєктована на основі сучасних підходів до розробки веб- та мобільних додатків, що забезпечують гнучкість, масштабованість та надійність.

Важливим архітектурним рішенням є використання компонентного підходу в розробці інтерфейсів. Для веб-додатку цей підхід реалізовано через бібліотеку React, яка дозволяє організувати інтерфейс за принципом модульності: від базових, пере-використовуваних елементів (кнопки, поля вводу) до складних композитних компонентів (форми, сторінки). Аналогічний декларативний підхід застосовано і для мобільного додатку, де інтерфейс створюється за допомогою Jetpack Compose, що дозволяє описувати користувацький інтерфейс як результат композиції функцій стану.

Для ефективного управління даними та станом цих компонентів були обрані відповідні архітектурні шаблони. Для веб-додатку використовується архітектура з однонаправленим потоком даних, що спрощує відстеження змін стану та полегшує тестування. Для мобільного додатку на Kotlin застосовується архітектурний шаблон (MVVM), який ефективно розділяє логіку представлення від бізнес-логіки. Реактивність інтерфейсу при зміні даних досягається за допомогою інструментів Kotlin Coroutines, зокрема StateFlow та SharedFlow.

На рівні комунікації з серверною частиною реалізовано патерн "Репозиторій" (Repository), який абстрагує джерела даних від бізнес-логіки клієнтського додатку. Такий підхід дозволяє ізолювати логіку роботи з API та забезпечує гнучкість у випадку зміни серверних ендпоінтів. Для обробки асинхронних операцій, зокрема запитів до API, використовуються сучасні підходи: Promise та синтаксис async/await у веб-додатку на JavaScript/TypeScript, та співпрограми (Coroutines) у мобільному додатку на Kotlin.

Таким чином, спроєктована архітектура забезпечує чітке розділення відповідальності між шарами представлення, логіки та даних. Це створює гнучку, масштабовану та надійну основу для клієнтської частини системи електронних голосувань, дозволяючи ефективно реалізувати всі необхідні функціональні можливості на обох цільових платформах.

3.3 Приклади найцікавіших алгоритмів та методів

Одним з найскладніших аспектів у розробці клієнтської частини системи став алгоритм управління багатоетапним процесом створення голосування. На відміну від типових CRUD-операцій, цей процес вимагає від користувача послідовного введення даних на кількох кроках, валідації цих даних та фінальної відправки єдиного комплексного запиту на сервер. Для реалізації цього функціоналу у веб-додатку на React було застосовано підхід, що поєднує управління локальним станом форми, умовний рендеринг компонентів та динамічну валідацію.

Для управління всіма даними форми голосування використовується єдиний об'єкт стану, що визначається хуком useState. Цей об'єкт (formData) містить усі поля: від базових (назва, опис, дати) до динамічних масивів (варіанти відповідей, довірені особи). Такий підхід дозволяє централізовано зберігати всю інформацію та легко передавати її між кроками. Процес розбитий на логічні етапи, кожен з яких представлений окремим компонентом. Поточний крок відстежується за допомогою змінної стану currentStep. Рендеринг відповідного компонента-кроку відбувається за допомогою функції, що використовує умовну логіку (switch-case), що робить код чистим та легко розширюваним.

Ключовим елементом алгоритму є функція валідації validateStep(), яка викликається перед переходом на наступний крок. Вона перевіряє заповненість та коректність даних поточного етапу. Наприклад, для першого кроку перевіряється наявність назви, опису та коректність часових рамок, а для кроку з варіантами відповідей – що кожен варіант має текст і їх кількість не менше двох. Лише після успішної валідації користувач може перейти далі, що забезпечує цілісність даних перед фінальною відправкою.

Найцікавіша частина алгоритму проявляється у роботі з динамічними списками, такими як варіанти відповідей (options) та довірені особи (trustedPersons). Для додавання нового елемента створюється новий об'єкт з унікальним id (зазвичай на основі Date.now()) і додається до масиву у стані, що викликає перерендерінг та появу нового поля вводу в інтерфейсі. Видалення елемента реалізовано через фільтрацію масиву за id. Цей підхід, що базується на принципі незмінності стану (immutability), є стандартною практикою в React для уникнення побічних ефектів.

На фінальному кроці, перед відправкою, вся накопичена в об'єкті formData інформація перетворюється на структуру CreatePollRequest, що відповідає контракту API. Це включає перетворення дат у формат ISO, мапування масиву опцій та, у випадку закритого голосування, додавання списку email-адрес довірених осіб. Лише після цього сформований об'єкт надсилається на сервер за допомогою pollService.createPoll(). Весь цей час кнопка відправки є неактивною, поки не буде пройдена валідація останнього кроку, що запобігає відправці неповних або некоректних даних.

Нижче наведено фрагмент коду зі сторінки CreateVotingPage.tsx, який ілюструє описаний алгоритм управління кроками та валідацією.

```
// Фрагмент з src/pages/CreateVotingPage.tsx

// ... ініціалізація стану formData та currentStep ...

const validateStep = (step: number): boolean => {
  switch (step) {
    case 1:
      // Перевірка, що всі основні поля заповнені
```

```
            return    !!formData.title    &&    !!formData.description    &&
!!formData.category &&
                   !!formData.startDate && !!formData.startTime &&
                   !!formData.endDate && !!formData.endTime;
          case 2:
            // Перевірка, що всі варіанти відповідей заповнені і їх кількість
достатня
            return formData.options.every(opt => opt.text.trim() !== '') &&
formData.options.length >= 2;
          case 3:
            // Якщо голосування закрите, перевіряємо, що є хоча б одна
довірена особа
            if (formData.isPrivate) {
              return formData.trustedPersons.length > 0 &&
                     formData.trustedPersons.every(p  =>  p.email.trim()  !==
'');
            }
            return true; // для відкритих голосувань цей крок завжди валідний
          default:
            return true;
        }
      };

      const handleNextStep = () => {
        if (validateStep(currentStep)) {
          setCurrentStep(prev => Math.min(totalSteps, prev + 1));
        } else {
          // Показати повідомлення про помилку валідації
          alert('Будь ласка, заповніть всі обов\'язкові поля на поточному
кроці.');
        }
      };

      const handleSubmit = async () => {
        if (!validateStep(currentStep)) {
          alert('Форма заповнена некоректно.');
          return;
        }
        setIsSubmitting(true);
        try {
          const createPollData = { /* ... перетворення formData в API DTO ...
*/ };
          await pollService.createPoll(createPollData);
          navigate('/dashboard/my-votings');
        } catch (error) {
          // ... обробка помилки ...
        } finally {
          setIsSubmitting(false);
        }
      };

      // ... у JSX:
      //                   <Button                    onClick={handleNextStep}
disabled={!validateStep(currentStep)}>Далі</Button>
      //                    <Button                     onClick={handleSubmit}
disabled={isSubmitting}>Створити</Button>
```

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Розробка клієнтської частини системи електронних голосувань базувалася на низці ключових програмних рішень, спрямованих на створення сучасних, надійних та зручних у використанні додатків. Вибір технологій та архітектурних підходів був зумовлений необхідністю забезпечити ефективну взаємодію з комплексною мікросервісною архітектурою та абстрагувати технічну складність блокчейн-технологій від кінцевого користувача.

4.1 Архітектура та технологічний стек клієнтської частини

В основі архітектури лежить принцип розділення клієнтських додатків за їх призначенням – повнофункціональний веб-додаток для широкого кола користувачів та спеціалізований мобільний додаток для вузького кола довірених осіб. Для веб-платформи було обрано бібліотеку React у поєднанні з TypeScript. Такий вибір дозволив побудувати масштабований та надійний інтерфейс на основі компонентного підходу [11]. React забезпечує ефективну організацію компонентів та підтримку сучасних патернів розробки [12].

Перевикористовувані UI-елементи, такі як Card, Input, Badge, винесені в окремий каталог src/components/ui і слугують будівельними блоками для більш складних, функціональних компонентів. Прикладом такого комплексного компонента є VotingCard, який агрегує та відображає всю ключову інформацію про голосування в єдиному інтерактивному блоці. Його декларативна природа дозволяє передавати дані через пропси, повністю відокремлюючи логіку відображення від логіки отримання даних (див. Додаток А).

Для мобільного додатку, що виконує виключно функцію верифікації, було обрано Kotlin та Jetpack Compose. Цей стек забезпечує високу нативну продуктивність, безпеку та дозволяє створювати інтерфейс за допомогою сучасного декларативного підходу, що є оптимальним для розробки надійного та сфокусованого на одній задачі інструменту.

4.2 Взаємодія з сервером та управління даними

Фундаментальним архітектурним рішенням є те, що клієнтська частина ніколи не взаємодіє з блокчейном напряму. Вся комунікація відбувається через захищений REST API, який надається серверною частиною. Це дозволяє інкапсулювати складну логіку (роботу зі смарт-контрактами, управління газом, підписання транзакцій) на backend, що значно спрощує та убезпечує клієнтські додатки.

Для організації взаємодії з API у веб-додатку створено спеціальний сервісний шар (src/services). Класи, такі як PollService та AuthService, виступають єдиною точкою входу для всіх мережевих запитів. Це дозволяє централізовано обробляти помилки, управляти токенами авторизації та абстрагувати компоненти від деталей реалізації ендпоінтів. Наприклад, метод getPolls в PollService не лише отримує дані, але й динамічно формує параметри запиту на основі стану фільтрів в UI. (див. Додаток Б)

Управління глобальним станом, зокрема сесією користувача, реалізовано через React Context API (AuthContext). Це дозволяє будь-якому компоненту отримати доступ до даних користувача та його ролей. На основі цих даних реалізовано динамічний контроль доступу до функціоналу безпосередньо в інтерфейсі. Наприклад, кнопка для створення голосування відображається лише для авторизованих користувачів, а її функціональність (створення відкритого чи закритого голосування) залежить від ролі.

```
// Приклад використання AuthContext у компоненті
import { useAuth } from '../contexts/AuthContext';
import { Button } from '../components/ui/Button';

const SomePageComponent: React.FC = () => {
  const { user, isPremiumUser } = useAuth();

  const handleCreateClick = () => {
    if (isPremiumUser()) {
      // Логіка для створення закритого голосування
    } else {
      // Логіка для створення відкритого голосування
    }
  };
```

```
  return (
    <>
      {user && (
        <Button onClick={handleCreateClick}>
          Створити {isPremiumUser() ? 'закрите' : 'відкрите'}
голосування
        </Button>
      )}
    </>
  );
};
```

Цей підхід забезпечує гнучкість інтерфейсу та його адаптацію до прав конкретного користувача, що є ключовою вимогою системи. Всі запити до захищених ендпоінтів автоматично включають JWT, що управляється на рівні axios interceptors, забезпечуючи надійний механізм авторизації.

Крім того, для оптимізації роботи з даними та покращення користувацького досвіду застосовано патерн асинхронного завантаження та управління станами (loading, error, success). Кожна сторінка, що потребує даних з сервера, ініціалізує власний стан завантаження. Це дозволяє відображати користувачу індикатори завантаження (спіннери), поки дані отримуються, а також коректно обробляти та показувати повідомлення про помилки, якщо запит до API завершився невдало. Прикладом такої реалізації є компонент MyVotingsPage, який керує станами isLoading та error і відповідно відображає або спіннер, або повідомлення про помилку, або список голосувань. Такий підхід робить інтерфейс більш відгукливим та передбачуваним для користувача.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Модульне тестування

Тестування розробленої клієнтської частини системи голосування виконувалося за допомогою сучасного стеку технологій JavaScript для фронтенд розробки. Як основний фреймворк тестування було обрано Jest [13] – потужну платформу для тестування JavaScript додатків, що забезпечує комплексне рішення для написання, запуску та аналізу тестів із вбудованою підтримкою мокування, покриття коду та паралельного виконання. Додатково використовувалася бібліотека React Testing Library для тестування React компонентів користувацького інтерфейсу, яка дозволяє тестувати компоненти з точки зору користувача, фокусуючись на поведінці замість деталей реалізації.

Було розроблено комплексний набір модульних тестів, що охоплює ключові компоненти клієнтської частини системи. Структура тестування включає шість основних тестових наборів: тестування сервісу верифікації користувачів (verificationService.test.ts), сервісу аутентифікації (authService.simple.test.ts), сервісу управління голосуваннями (pollService.test.ts), сервісу обробки голосів (voteService.test.ts), UI компонентів (SimpleButton.test.tsx) та допоміжних функцій (helpers.test.ts). Загалом було створено 125 окремих тестових сценаріїв, які покривають найважливішу функціональність фронтенд додатку.

Запуск повного тестового покриття здійснювався командою npm run test:coverage, яка виконує всі наявні тести з формуванням детального звіту про покриття коду. Параметр --watchAll=false забезпечує одноразове виконання всіх тестів без режиму спостереження за змінами файлів. Результати виконання тестування показали стовідсоткову успішність – всі 125 тестів було успішно пройдено за час 2.454 секунди. Така швидкість виконання свідчить про ефективну архітектуру тестів та оптимізовані алгоритми перевірки клієнтської логіки.

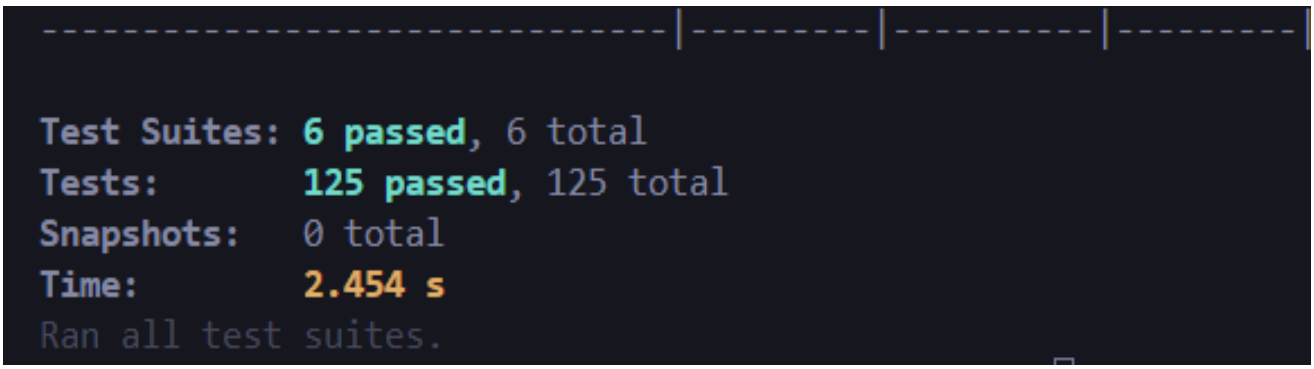
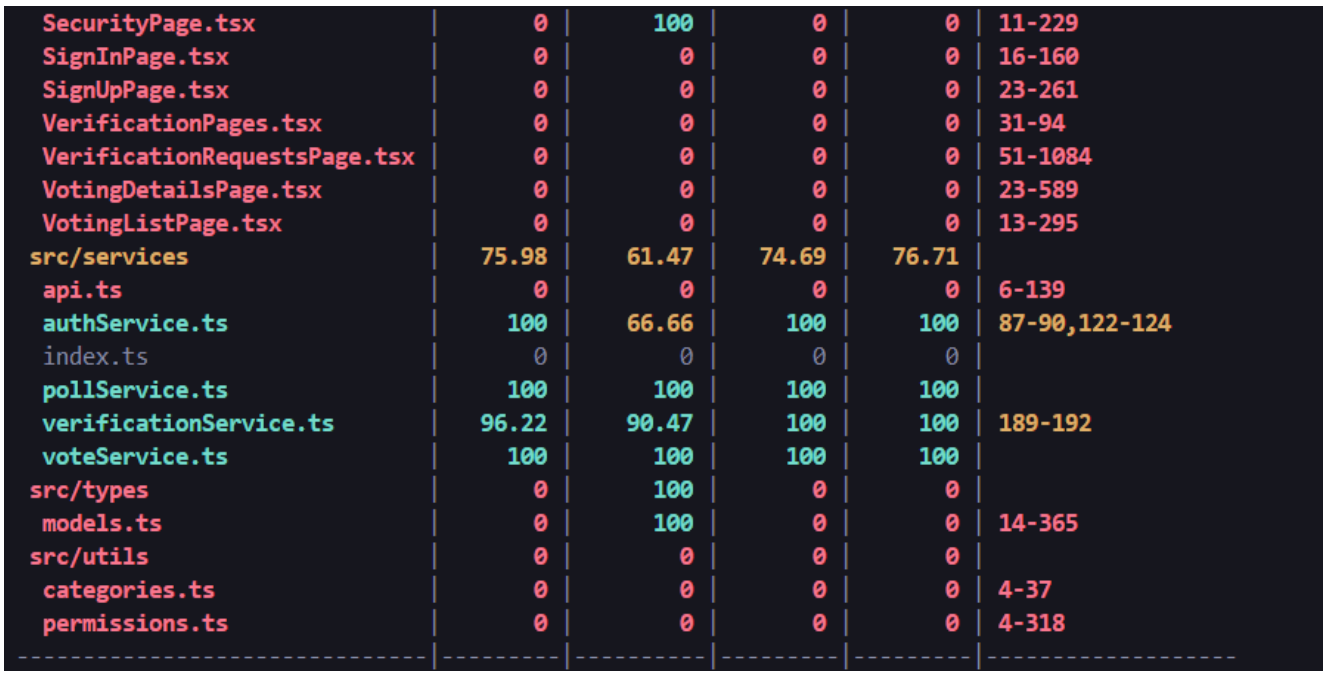


Рисунок 5.1 – Результати виконання модульних тестів системи голосування (рисунок виконаний самостійно)

Аналіз покриття коду показав характерні для фронтенд проектів результати, які потребують детального пояснення. Загальне покриття проекту становить 11.37% по операторах, 5.72% по гілках, 9.93% по функціях та 11.59% по рядках коду. На перший погляд ці цифри можуть здатися низькими, однак вони відображають типову архітектуру React додатків та стратегію тестування, орієнтовану на бізнес-логіку.

Основна причина відносно низького загального покриття полягає в тому, що значна частина кодової бази представлена React компонентами користувацького інтерфейсу, які на поточному етапі розробки мають нульове покриття тестами. До таких компонентів належать всі файли в директоріях src/components/, src/pages/, включаючи LandingSidebar.tsx, VotingCard.tsx, ProfilePage.tsx, CreateVotingPage.tsx та багато інших. Ця стратегія є виправданою для проектів на стадії MVP (Minimum Viable Product), де пріоритет надається тестуванню критично важливої бізнес-логіки.

Натомість було зосереджено увагу на тестуванні сервісного шару додатку, який відповідає за взаємодію з API, обробку даних, аутентифікацію та основну функціональність системи голосування. Саме ці компоненти формують основу надійності всієї системи та потребують найретельнішого тестування.



SecurityPage.tsx	0	100	0	0	11-229
SignInPage.tsx	0	0	0	0	16-160
SignUpPage.tsx	0	0	0	0	23-261
VerificationPages.tsx	0	0	0	0	31-94
VerificationRequestsPage.tsx	0	0	0	0	51-1084
VotingDetailsPage.tsx	0	0	0	0	23-589
VotingListPage.tsx	0	0	0	0	13-295
src/services	75.98	61.47	74.69	76.71	
api.ts	0	0	0	0	6-139
authService.ts	100	66.66	100	100	87-90,122-124
index.ts	0	0	0	0	
pollService.ts	100	100	100	100	
verificationService.ts	96.22	90.47	100	100	189-192
voteService.ts	100	100	100	100	
src/types	0	100	0	0	
models.ts	0	100	0	0	14-365
src/utils	0	0	0	0	
categories.ts	0	0	0	0	4-37
permissions.ts	0	0	0	0	4-318

Рисунок 5.2 – Детальний звіт покриття коду по модулях фронтенд системи (рисунок виконаний самостійно)

Результати покриття ключових сервісів демонструють відмінні показники, що підтверджує високу якість розробленого коду. Сервіс аутентифікації (authService.ts) має стовідсоткове покриття по операторах, функціях та рядках коду, з покриттям гілок на рівні 66.66%. Неповне покриття гілок пояснюється наявністю деяких умовних конструкцій для обробки edge cases, які важко відтворити в тестовому середовищі. Сервіси управління голосуваннями (pollService.ts) та обробки голосів (voteService.ts) досягли ідеального покриття у всіх категоріях – 100% по всіх метриках, що свідчить про ретельну проробку тестових сценаріїв.

Сервіс верифікації користувачів (verificationService.ts) показав найвищі результати з покриттям 96.22% по операторах, 90.47% по гілках та повне покриття по функціях і рядках. Невелике зниження покриття пов'язане з наявністю складної логіки обробки різних типів документів для верифікації, де деякі сценарії залежать від зовнішніх факторів.

Окрему увагу приділено тестуванню допоміжних функцій (helpers.test.ts), які забезпечують коректну роботу утилітарних операцій у всьому додатку. Ці функції включають форматування чисел, обчислення відсотків, валідацію електронних

адрес та генерацію унікальних ідентифікаторів. Стовідсоткове покриття цих функцій гарантує надійність базових операцій системи.

Тестування React компонентів представлено тестовим набором для SimpleButton.test.tsx, який демонструє підхід до перевірки інтерфейсних елементів. Тести перевіряють правильність відображення контенту кнопки, обробку подій кліку, поведінку в різних станах (disabled, loading), застосування CSS класів та можливості кастомізації зовнішнього вигляду. Цей приклад служить шаблоном для розширення тестового покриття інших UI компонентів у майбутньому.

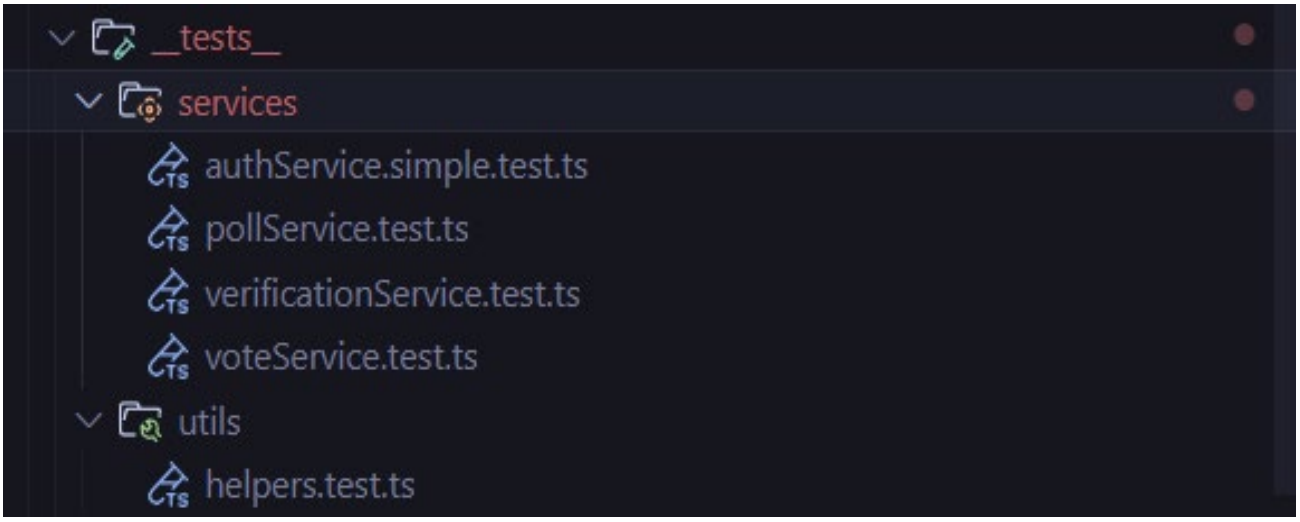


Рисунок 5.3 – Структура тестових сценаріїв для фронтенд сервісів (рисунок виконаний самостійно)

Архітектура сервісного шару спроектована таким чином, що всі сервіси працюють з мокованими HTTP клієнтами, що дозволяє тестувати логіку обробки запитів та відповідей незалежно від реального API. Сервіс аутентифікації покривається тестами, що перевіряють весь життєвий цикл роботи з користувачами: реєстрацію з валідацією вхідних даних, процес входу в систему з обробкою JWT токенів, отримання інформації про поточного користувача, перевірку системи ролей та дозволів, оновлення профілю, зміну паролів, видалення облікових записів та безпечний вихід з системи.

Кожен тестовий сценарій структурований за принципом Arrange-Act-Assert, що забезпечує чітке розділення між підготовкою тестових даних, виконанням

тестованої операції та перевіркою отриманих результатів. Використання мокування дозволяє повністю контролювати поведінку зовнішніх залежностей та тестувати як успішні сценарії, так і різноманітні помилкові ситуації.

Сервіс управління голосуваннями охоплює тестами створення нових голосувань з валідацією всіх параметрів, отримання списків голосувань з можливістю фільтрації за різними критеріями, управління життєвим циклом голосувань від створення до завершення, перевірку дозволів користувачів на створення приватних голосувань та роботу з системою категоризації. Особлива увага приділяється тестуванню часових обмежень та бізнес-правил доступу до голосувань.

Методологія тестування базується на сучасних практиках розробки програмного забезпечення та найкращих підходах до тестування React додатків. Активно використовується інверсія залежностей через dependency injection для забезпечення тестованості коду. Всі зовнішні API викликі мокуються за допомогою можливостей Jest, що дозволяє створювати передбачувані та ізольовані тестові середовища.

```
describe('Вихід з системи', () => {
  it('повинен очистити токени', () => {
    authService.logout();

    expect(mockAuthApi.clearToken).toHaveBeenCalled();
    expect(mockVotingApi.clearToken).toHaveBeenCalled();
  });
});

describe('Перевірка аутентифікації', () => {
  it('повинен повернути true коли токен існує', () => {
    mockAuthApi.getToken.mockReturnValue('token');

    expect(authService.isAuthenticated()).toBe(true);
  });

  it('повинен повернути false коли токена немає', () => {
    mockAuthApi.getToken.mockReturnValue(null);

    expect(authService.isAuthenticated()).toBe(false);
  });
});

describe('Методи для роботи з ролями', () => {
  const mockUser = {
    id: 1,
    uuid: 'uuid-123',
    email: 'test@example.com',
    phone_number: '+380123456789',
    country_code: 'UA',
    status: 'active',
    created_at: '2024-01-01T00:00:00Z',
    roles: [
      {
        id: 1,
        name: 'User',
        code: 'user' as const,
```

Рисунок 5.4 – Приклад коду тестування сервісу аутентифікації (рисунок виконаний самостійно)

Інтеграція TypeScript у процес тестування забезпечує додатковий рівень безпеки та підвищує якість тестового коду. Строга типізація тестових даних, функцій та очікуваних результатів дозволяє виявляти потенційні помилки на етапі компіляції, що значно підвищує надійність тестів. Використання TypeScript також покращує читабельність тестів та спрощує їх підтримку в довгостроковій перспективі.

ВИСНОВКИ

Результатом виконання роботи є проведення комплексного аналізу предметної галузі електронних голосувань на основі блокчейну. Аналіз показав, що існуючі системи електронних голосувань мають ряд суттєвих недоліків: традиційні централізовані системи не забезпечують необхідного рівня прозорості та захисту від маніпуляцій, а блокчейн-системи відзначаються надмірною технічною складністю інтерфейсів, що створює бар'єр для широкого впровадження[14].

Досліджено найсучасніші платформи (Follow My Vote, Horizon State, Vocdoni, Democracy Earth) та виявлено їх переваги і недоліки. Встановлено, що особливо важливою в контексті України є можливість використання мобільних пристроїв для голосування, враховуючи тенденції розвитку цифрових сервісів та вимушене переміщення значної частини населення.

На основі проведеного аналізу спроектовано архітектуру клієнтської частини системи, що включає веб-інтерфейс та мобільний додаток, з використанням відповідних UML-діаграм, які формалізують різні аспекти системи: варіанти використання для різних ролей користувачів, структуру класів, послідовність взаємодій, життєвий цикл голосувань та активності користувачів.

На заключному етапі розроблено клієнтську частину системи, що включає веб-інтерфейс з використанням React та мобільний додаток для Android на базі Kotlin і Jetpack Compose.

Основними проблемами існуючих систем є надмірна технічна складність для пересічних користувачів, незбалансованість між прозорістю процесу та простотою використання, а також недостатня адаптація для мобільних пристроїв, які стають основним каналом взаємодії з цифровими сервісами. Спроектовано архітектуру системи з використанням патернів проектування для ефективного управління станом та даними — Redux для веб-додатку та MVVM з використанням StateFlow для мобільного додатку.

Реалізовано криптографічні алгоритми на клієнтській стороні, зокрема механізми сліпих підписів та доказів з нульовим розголошенням, що забезпечують анонімність голосування при збереженні контролю над правом участі. Розроблено

UML-діаграми, які формалізують різні аспекти системи: варіанти використання для різних ролей користувачів, структуру класів, послідовність взаємодій, життєвий цикл голосувань та активності користувачів при взаємодії з системою.

Особливу увагу приділено інтерфейсу користувача, який приховує технічну складність блокчейну за знайомими метафорами та інтерфейсними патернами, забезпечуючи доступність системи для користувачів з різним рівнем технічної грамотності. Реалізовано механізми безпеки для захисту приватних ключів та персональних даних користувачів, включаючи локальне шифрування, біометричну аутентифікацію та концепцію соціального відновлення доступу.

Впроваджено підтримку роботи в умовах нестабільного інтернет-з'єднання через механізми локального кешування та можливість підготовки голосів у режимі офлайн, що особливо актуально в контексті України, де внаслідок воєнних дій інфраструктура може бути пошкоджена. Розроблена система має значний практичний потенціал як інструмент електронної демократії для проведення голосувань різного рівня – від загальнодержавних референдумів до місцевих рішень громад та корпоративного управління, забезпечуючи участь громадян, які знаходяться в різних регіонах країни або за кордоном.

Інноваційність розробки полягає в успішному поєднанні технологій блокчейну, що гарантують незмінність та прозорість результатів, із сучасними підходами до створення інтуїтивно зрозумілих інтерфейсів, що робить складні криптографічні механізми доступними для широкого кола користувачів. Водночас важливо враховувати потенційні ризики впровадження інтернет-голосування в Україні, включаючи питання кібербезпеки та необхідність поступового впровадження таких систем [15].

Апробація роботи проведена на І Міжнародній науково-технічній конференції «Сучасні інформаційні технології та системи штучного інтелекту» MIT@AIS-2025 (див. дод. Д), що засвідчує актуальність і практичну цінність її результатів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Blockchain-Based E-Voting Systems: A Technology Review – URL: https://www.mdpi.com/2079-9292/13/1/17 (дата звернення: 10.05.2025).

2. Blockchain for Electronic Voting System—Review and Open Research Challenges – URL: https://www.mdpi.com/1424-8220/21/17/5874 (дата звернення: 10.05.2025).

3. Blockchain-based electronic voting systems: A case study in Morocco – URL: https://www.sciencedirect.com/science/article/pii/S2666603024000046 (дата звернення: 08.05.2025).

4. Secure Digital Voting System based on Blockchain – URL: https://core.ac.uk/download/pdf/155779036.pdf (дата звернення: 07.05.2025)

5. Mobile Design Best Practices – URL: https://uxplanet.org/mobile-design-best-practices-2d16d37ecfe (дата звернення: 05.05.2025).

6. Designing for mobile: 5 best practices for UI designers – URL: https://www.uxdesigninstitute.com/blog/designing-for-mobile-best-practices/ (дата звернення: 03.05.2025).

7. E-Voting using Blockchain Technology – URL: https://www.researchgate.net/publication/342932007_E-Voting_using_Blockchain_Technology (дата звернення: 01.05.2025).

8. Securing e-voting based on blockchain in P2P network – URL: https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-019-1473-6 (дата звернення: 29.04.2025).

9. Blockchain based E-voting System – URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3648870 (дата звернення: 25.04.2025).

10. A Systematic Review of Challenges and Opportunities of Blockchain for E-Voting – URL: https://www.mdpi.com/2073-8994/12/8/1328 (дата звернення: 22.04.2025).

11. Adam Freeman. Pro React 16: Монографія. Berkeley, CA: Apress, 2019. 768 с.

12. Banks A., Porcello E. Learning React: Модерн. 2 вид. Sebastopol, CA: O'Reilly Media, 2020. 310 с.

13. Jest: Delightful JavaScript Testing. Jest. URL: https://jestjs.io/docs/getting-started (дата звернення: 15.05.2025).

14. Tereshchenko G., Kyrychenko I. Analysis and justification of the use of existing blockchain solutions for the protection of digital assets. Innovative technologies and scientific solutions for industries. 2024. № 1 (27). С. 164–178. URL: https://doi.org/10.30837/itssi.2024.27.164 (дата звернення: 12.05.2025)

15. The risks of rushing to internet voting in Ukraine. Atlantic Council. URL: https://www.atlanticcouncil.org/blogs/ukrainealert/the-risks-of-rushing-to-internet-voting-in-ukraine/ (дата звернення: 16.05.2025).

16. GitHub. NurePylaikinYevhen/2025_B_PI_PZPI-21-9_Pylaikin_Y_O URL: https://github.com/NurePylaikinYevhen/2025_B_PI_PZPI-21-9_Pylaikin_Y_O