Міністерство освіти і науки України

Харківський національний університету імені В.Н. Каразіна

Навчально-науковий інституту комп’ютерних наук та штучного інтелекту

Спеціальність 125 «Кібербезпека»

Освітня програма «Кібербезпека»

В.о. зав. кафедрою КІСМіТ

Марина ЄСІНА

“Допущено до захисту”

« »_____2025р.

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Побудова робочого місця адміністратора децентралізованої системи захищеного голосування»

Оцінка «_____»

Голова ЕК

Мичуда Л.З.

Керівник: д.т.н, проф. Олійников Р. В

Рецензент: PhD доц. Родінко М.Ю.

Виконавець: студент групи КБ-41

Кириченко Д.Д.

Харків 2025

РЕФЕРАТ

Пояснювальна записка до проєкту бакалавра містить 56 сторінок, 11 рисунків, 2 таблиці, 1 додаток, 48 посилань на джерела.

Мета роботи полягає у розробці та дослідженні функціоналу робочого місця адміністратора децентралізованої системи захищеного голосування, що базується на принципах розподілених обчислень і забезпечення високого рівня безпеки. Розглянуто процеси реєстрації адміністратора, формування переліку голосувань, учасників, власників нод, а також встановлення параметрів консенсусу та завантаження системи на p2p-сервер.

Об’єкт дослідження – процес організації децентралізованого захищеного голосування.

Предмет дослідження – структура та функціональні компоненти інтерфейсу адміністратора, зокрема можливості керування голосуваннями, нодами та учасниками системи, а також особливості розгортання P2P-мережі.

Основними методами дослідження є проєктування, моделювання архітектури системи, аналіз існуючих рішень у сфері електронного голосування, а також тестування безпечності та ефективності реалізованого функціоналу.

У роботі досліджено: можливості використання національних сервісів електронної ідентифікації (Дія, BankID), принципи побудови консенсусу в децентралізованих системах, механізми безпечного зберігання та передачі даних, а також роль адміністратора в керуванні розподіленою мережею.

Результати роботи можуть бути використані для створення національних або корпоративних платформ електронного голосування з підвищеним рівнем захищеності.

Ключові слова: ДЕЦЕНТРАЛІЗОВАНЕ ГОЛОСУВАННЯ, РОБОЧЕ МІСЦЕ АДМІНІСТРАТОРА, ЗАХИЩЕНЕ ГОЛОСУВАННЯ, КОНСЕНСУС, P2P-МЕРЕЖА, НОДИ, УЧАСНИКИ ГОЛОСУВАННЯ, BANKID, ДІЯ, ЕЛЕКТРОННА ІДЕНТИФІКАЦІЯ, АДМІНІСТРУВАННЯ СИСТЕМИ.

ABSTRACT

The explanatory note to the bachelor's project consists of 56 pages, 11 figures, 2 tables, 1 appendix, and 48 references.

The aim of the work is to develop and study the functionality of the administrator’s workplace in a decentralized secure voting system based on principles of distributed computing and high-level data protection. The project examines the processes of administrator registration, creation of voting lists, participant and node owner management, consensus parameter configuration, and deployment of the system on a peer-to-peer (P2P) network.

The object of the study is the process of organizing decentralized secure voting.

The subject of the study is the structure and functional components of the administrator's interface, including the capabilities for managing votings, nodes, and participants, as well as the peculiarities of deploying a P2P network.

The main research methods include system design, architecture modeling, analysis of existing electronic voting solutions, and testing of security and effectiveness of the implemented functionality.

The work explores the potential use of national digital identity services (Diia, BankID), consensus mechanisms in decentralized systems, secure data storage and transmission methods, and the administrator’s role in managing a distributed network.

The results of this work may be used to create national or corporate electronic voting platforms with enhanced security.

Keywords: DECENTRALIZED VOTING, ADMINISTRATOR’S WORKPLACE, SECURE VOTING, CONSENSUS, P2P NETWORK, VOTING PARTICIPANTS, NODES, BANKID, DIIA, ELECTRONIC IDENTIFICATION, SYSTEM ADMINISTRATION.

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ЕЦП	– електронний цифровий підпис
ОС	– операційна система
ПЗ	– програмне забезпечення
API	– Application Programming Interface
BFT	– Byzantine Fault Tolerance
CVS	– Concurrent Versions System
DDoS	– Distributed Denial of Service
IEEE	– Institute of Electrical and Electronics Engineers
IFES	– International Foundation for Electoral Systems
IPFS	– InterPlanetary File System
JSON	– JavaScript Object Notation
PoA	– Proof of Authority
PoS	– Proof of Stake
PoW	– Proof of Work
P2P	– Peer-to-Peer
RSA	– Rivest–Shamir–Adleman
TLS	– Transport Layer Security
VVPAT	– Voter-Verified Paper Audit Trail
VPN	– Virtual Private Network

ВСТУП

В останні десятиліття світ стикається з потребою реформування виборчих процесів шляхом їх цифровізації. Стрімкий розвиток інформаційних технологій, а також зростаюча потреба в прозорих, швидких і безпечних формах волевиявлення призводять до активного впровадження електронного голосування в різних сферах життя. Особливої актуальності це набуває в умовах військової загрози, пандемій чи інших кризових ситуацій, коли фізична присутність виборців неможлива або небажана. Проте звичайні централізовані електронні системи голосування часто залишаються вразливими до атак, фальсифікацій або цензури, що ставить під сумнів легітимність результатів виборів [1].

Одним із перспективних напрямів у цій галузі є застосування децентралізованих систем голосування, які базуються на p2p-мережах, блокчейні та криптографічних методах захисту [2]. Такі системи забезпечують прозорість, незмінність, відсутність єдиного контролюючого центру та довіру до результатів. Ключовою складовою подібної системи є робоче місце адміністратора, через яке здійснюється підготовка до голосування: формування списку виборів, учасників, нод та конфігурація основних параметрів. Саме створенням такого інструменту присвячена дана кваліфікаційна робота.

Актуальність теми зумовлена нагальною потребою у створенні інтерфейсу керування децентралізованим голосуванням, що є не лише безпечним, але й зручним у використанні. Без такого адміністративного модуля запуск голосування стає або неможливим, або надто складним для практичного впровадження. Враховуючи, що в Україні активно розвиваються електронні сервіси, такі як «Дія» та BankID, в роботі також буде розглянута можливість інтеграції з цими платформами для безпечної автентифікації адміністратора [3].

Метою дипломної роботи є створення робочого місця адміністратора децентралізованої системи захищеного голосування, що забезпечує налаштування

всіх ключових параметрів виборчого процесу з подальшим збереженням інформації на сервері p2p-мережі.

Об’єкт дослідження – децентралізовані системи електронного голосування.

Предмет дослідження – інтерфейс адміністратора для організації процесу децентралізованого голосування.

Реальність і практичність розробки. У процесі підготовки цієї дипломної роботи буде реалізовуватися описаний функціонал поступово, з використанням доступних технологій (наприклад, JavaScript/Java/Python для інтерфейсу, JSON як формат обміну даними, REST або WebSocket для взаємодії з p2p-мережею).

Завдання дослідження:

1) Провести аналіз сучасних підходів до реалізації децентралізованих систем голосування (зокрема, з використанням блокчейну, IPFS, криптографічних механізмів тощо) [4].

2) Визначити основні вимоги до інтерфейсу адміністратора, враховуючи зручність використання, надійність збереження даних, захищеність доступу.

3) Реалізувати:

- реєстрацію адміністратора (з перевіркою через умовну авторизацію; опціонально – етап автентифікації через Дію/BankID – як модель або прототип);
- створення переліку голосувань з можливістю вказати назву, опис і список варіантів вибору;
- формування списку учасників голосування – за іменами, ID, email або іншими унікальними ідентифікаторами;
- формування списку власників нод та задання кількості нод, які вони отримують (наприклад, для моделювання ваги голосу);
- задання параметрів консенсусу – таких як тип алгоритму (наприклад, PoW, PoS, Raft, PBFT), кворум, час голосування тощо;
- експорт конфігурації у форматі JSON або подібному, з подальшим завантаженням на сервер p2p-мережі.

4) Протестувати створену систему в умовах моделювання виборчого процесу.

5) Підготувати рекомендації для подальшого розвитку інтерфейсу та масштабування системи.

Очікуваний результат – повністю функціональний інструмент (робоче місце адміністратора), який дозволяє запускати голосування в тестовій децентралізованій мережі. Цей результат може бути розширено у майбутньому для використання в студентському самоврядуванні, громадських.

1 АКТУАЛЬНИЙ СТАН І ПЕРСПЕКТИВИ РОЗВИТКУ СИСТЕМ ЕЛЕКТРОННОГО ГОЛОСУВАННЯ

1.1 Огляд сучасних систем електронного голосування

На сьогодні існує декілька впроваджених систем електронного голосування, що демонструють різні підходи до організації виборчого процесу. Серед найвідоміших прикладів – система інтернет-голосування в Естонії та експериментальні опитування через застосунок «Дія» в Україні. Кожна з цих систем має власні характеристики, архітектуру та рівень безпеки [5].

Естонська система інтернет-голосування. Естонія стала першою країною, що запровадила національне інтернет-голосування ще у 2005 році. Голосування відбувається дистанційно: виборець завантажує на свій комп'ютер спеціальну програму і авторизується за допомогою електронного цифрового підпису (ЕЦП), вбудованого в ID-картку громадянина. Після цього виборець може заповнити електронний бюлетень і підписати свій вибір ЕЦП, шифруючи голос. Для підвищення довіри система дозволяє переголосування – виборець може змінити свій електронний голос, і зарахованим буде останній поданий вибір (це зменшує ризик примусу, адже під тиском можна проголосувати, а пізніше змінити голос самостійно). Естонська система інтегрована з національним реєстром і забезпечує криптографічний захист передавання даних. Програмне забезпечення серверної частини пройшло незалежні аудити, частину коду відкрито для громадськості. За даними на 2023 рік, понад половину голосів на парламентських виборах Естонії було подано онлайн [6], що підтверджує довіру населення до цієї технології. Водночас система постійно вдосконалюється з урахуванням рекомендацій експертів з кібербезпеки.

Експериментальні голосування в системі «Дія». В Україні роль платформи для електронного волевиявлення починає виконувати мобільний застосунок «Дія». Наразі через «Дію» реалізовано кілька пілотних електронних голосувань консультативного характеру. Прикладом є опитування під час національного

відбору на пісенний конкурс «Євробачення-2023», коли за одну годину понад 166 тисяч українців проголосували в застосунку [7] за свого фаворита. Користувач «Дії» проходить електронну ідентифікацію (через BankID чи NFC-зчитування чипу ID-картки), після чого отримує доступ до електронного бюлетеня в додатку і може зробити вибір у кілька кліків. Узагальнення цього можна побачити в таблиці 1.1.

Таблиця 1.1 - Сучасна система електронного голосування

Система	Технології та безпекові заходи	Реалізація (практичне застосування)
Естонське інтернет-голосування	Дистанційне голосування через інтернет; централізовані сервери під управлінням Держвиборчкому; автентифікація виборців за допомогою ID-картки та ЕЦП.	Застосовується на національних виборах в Естонії з 2005 р.; на парламентських виборах 2023 року понад 51% голосів подано онлайн
«Дія» (електронне голосування)	Мобільний додаток для електронного голосування; централізована обробка даних державним сервером; автентифікація через BankID/MobileID або ID-картку; результати підраховуються автоматично і швидко.	Пілотні опитування громадської думки в Україні (неофіційні вибори). Напр., через «Дію» проголосували 166 тис. громадян під час відбору на «Євробачення-2023».

Результати такого голосування обробляються централізовано державним хмарним сервісом і можуть бути отримані практично миттєво. Перевагою є зручність для користувачів – участь у голосуванні можлива віддалено з будь-якого місця, без необхідності приходити на дільницю. Цей інструмент вже використовувався також для опитувань щодо складу журі національного відбору «Євробачення-2024» тощо. Хоча голосування в «Дії» поки що не мають юридично зобов'язувальної сили, держава відпрацьовує технічні та організаційні аспекти е-голосування. У перспективі розглядається можливість проведення consultative референдумів або виборів через «Дію», але для цього необхідно вирішити питання безпеки і довіри виборців до системи [6].

Як видно з таблиці, сучасні е-системи можуть суттєво різнитися. Естонська модель спирається на державну інфраструктуру та ЕЦП, забезпечуючи контроль з боку виборчих органів. Український експеримент із «Дією» показує прагнення інтегрувати голосування в існуючу цифрову екосистему послуг.

1.2 Проблемні аспекти розгорнутих систем електронного голосування

Незважаючи на успіхи впровадження, існуючі електронні системи голосування мають суттєві проблемні аспекти. На жаль, науковий консенсус наразі свідчить, що сучасні технології не забезпечують повністю надійного і захищеного інтернет-голосування [8].

Аналіз практики показує кілька ключових проблем електронного голосування:

- Вразливість до кібератак. Електронні системи теоретично піддаються таким загрозам, як несанкціонований доступ до серверів, злом програмного забезпечення голосування, маніпуляції з боку інсайдерів, віруси та шкідливі програми на пристроях виборців тощо. Дослідники відзначають, що зловмисник може віддалено атакувати як серверну інфраструктуру, так і клієнтські пристрої виборців; розробивши один метод атаки, його можна масштабувати на велику кількість голосів, причому майже непомітно для системи контролю. Наприклад, шкідливе ПЗ на комп'ютері виборця

теоретично здатне змінити його голос до шифрування, і це буде надзвичайно важко виявити.

- Недостатність традиційних заходів безпеки. Класичні вимоги до виборів – таємність бюлетеня, перевірка правильності підрахунку, унеможливлення повторного голосування – важко одночасно реалізувати в електронній формі. Багато інтернет-систем покладаються на довіру до центрального серверу та користувацьких пристроїв. Це створює єдині точки відмови: якщо сервер скомпрометовано або пристрої виборців масово уражені, результати голосування можуть бути змінені. Як відзначив один з аудиторів естонської системи, вона «фактично сліпо довіряє серверній частині і комп'ютерам виборців», що може бути привабливою ціллю для атак державного рівня.
- Проблема підкупу та контрольованого голосування. Дистанційне голосування через інтернет позбавлене природного захисту від примусу, який існує на звичайних дільницях (де голосування відбувається таємно в кабіні). У розгорнутих системах з'являються ризики підкупу та зовнішнього тиску на виборців. Як зазначають фахівці, можливий сценарій, коли зловмисники скуповують голоси: достатньо «взяти ноутбук, мішок грошей і ходити по адресах, допомагаючи людям проголосувати» – при інтернет-голосуванні це технічно спростило б підкуп, і протидіяти такій схемі майже неможливо. Виборець, перебуваючи поза контрольованим середовищем, може голосувати під тиском родичів, роботодавців або третіх осіб, і система це не відстежить.
- Зниження прозорості та довіри. Традиційні паперові вибори мають певний рівень спостережуваності: представники партій та громадськості можуть бути присутніми при підрахунку бюлетенів, пересвідчитися у фізичній наявності урн, пломб тощо. В електронному голосуванні процес підрахунку відбувається у «чорній скриньці» програмного забезпечення, що викликає скепсис у частини суспільства. Необхідні механізми перевірки правильності підрахунку – такі як публічні аудити, відкрита публікація хешів голосів, протоколи end-to-end верифікації –

але навіть їх впровадження вимагає від виборців довіряти складним криптографічним процедурам. У ряді країн (Німеччина, Нідерланди) Конституційні суди заборонили або обмежили використання електронного голосування саме через те, що середній виборець не може зрозуміти і проконтролювати електронний процес, а отже, принцип публічності і підконтрольності виборів порушується.

- Обмеження продуктивності та масштабованості. Деякі технології, застосовані в електронних виборах, мають обмеження по швидкодії при великих обсягах даних. Наприклад, складні криптографічні протоколи (міксування бюлетенів, багаторазове шифрування, Zero Knowledge докази) можуть вимагати значних ресурсів і часу при мільйонах виборців. Блокчейн-платформи стикаються з проблемою пропускної здатності (кількість транзакцій у секунду) та затримок підтвердження блоку, особливо якщо використовуються публічні блокчейни. Хоча у приватних (permissioned) блокчейнах продуктивність вища, швидкість обробки все одно є фактором: потрібно забезпечити одночасне голосування великих масивів користувачів без зависань і збоїв.
- Проблеми правового і етичного характеру. Запровадження е-голосування піднімає питання відповідності законодавчим нормам. Також постає питання збереження даних: електронні бюлетені, навіть зашифровані, зберігаються на серверах чи в блокчейні – потрібно визначити, як довго їх зберігати, хто має до них доступ, як запобігти аналізу метаданих (наприклад, часу голосування) для можливого розкриття вибору конкретної особи.

Окремо слід згадати про блокчейн-рішення в електронному голосуванні. Спочатку висловлювалася думка, що розподілений реєстр вирішить головні проблеми – забезпечить незмінність даних і довіру без необхідності центрального авторитету. Дійсно, блокчейн здатен зробити підрахунок прозорим – учасники мережі можуть самостійно переконатися, що дані про голоси не змінювалися після запису. Проте досвід і дослідження показали, що блокчейн не є панацеєю. По-перше, навіть у блокчейн-системах клієнтські додатки та пристрої залишаються

такими ж вразливими, як і в звичайному інтернет-голосуванні (вразливості на рівні браузера, ОС тощо). По-друге, вимоги до таємності виборів важко суміщаються з публічністю блокчейну: аби гарантувати, що ніхто не дізнається, як проголосував конкретний користувач, доводиться ускладнювати протокол (анонімні токени, шифрування голосів тощо), і це частково нівелює прозорість. Експерти з MIT та Гарварду звертають увагу, що блокчейн-системи не усувають фундаментальних загроз електронних виборів, більше того – додають нові точки відмови. Наприклад, у permissioned-блокчейні (де обмежене коло вузлів) потенційно простіше скомпрометувати усі сервери, якщо вони працюють на схожому ПЗ. Можливі й специфічні атаки: форкування ланцюга (коли частина вузлів починає альтернативний ланцюг блоків) може викликати плутанину, яка версія є дійсною. Також постає проблема керування ключами: якщо виборець втратив приватний ключ – він втратить можливість голосувати; якщо ключ викрадуть – зловмисник проголосує замість виборця.

1.3 Модель загроз

Для побудови захищеної системи електронного голосування важливо визначити модель загроз – перелік потенційних небезпек та векторів атак, на які система повинна бути стійкою [9]. В контексті децентралізованої системи голосування (як і будь-якої е-системи виборів) можна виділити такі ключові загрози:

- Порушення конфіденційності голосування. Загроза: розкриття відповідності між виборцем і поданим ним голосом (порушення таємниці голосування). Наслідки: втрата анонімності підриває принцип вільних виборів, відкриває шлях до тиску на виборців ex post facto (після голосування) або до продажу голосів. Можливі вектори: злом бази даних, де міститься інформація про те, хто як проголосував (якщо така інформація десь існує в системі); перехоплення трафіку або шпигунське ПЗ на пристрої, що фіксує вибір користувача; недоліки криптографічних протоколів, що дозволять зіставити зашифрований голос з особою виборця.

- Порушення цілісності та правильності результатів. Загроза: несанкціонована зміна, видалення або додавання голосів, що призводить до спотворення підсумків виборів. Наслідки: сфальсифікований результат виборів (перемога невірного кандидата/опції), підрив легітимності влади, суспільні заворушення. Вектори: кібератака на сервери підрахунку або вузли блокчейн-мережі з метою модифікації даних (включно з атакою типу «маніпуляція в пам'яті» – коли атакувальник змінює дані до їх запису на носій або у блокчейн); атака на алгоритми підрахунку (вбудовування бекдору, використання вразливостей у криптобібліотеках); компрометація ключів шифрування, що дозволить розшифровувати або підробляти підписи голосів; колізійна атака на функцію хешування (теоретично – підробка доказів цілісності). До цієї категорії належить і повторне голосування однієї особи (multi-voting) – спроба проголосувати більше разу через різні облікові записи чи експлойт системи.
- Порушення доступності (відмова у обслуговуванні). Загроза: неможливість для легітимних користувачів отримати доступ до системи голосування у потрібний час. Наслідки: збої або зрив виборчого процесу, потенційна потреба продовження голосування або анулювання результатів дільниць, падіння довіри. Вектори: DDoS-атаки на веб-портал або сервери системи, що перевантажують їх і роблять недоступними для виборців; саботаж мережевої інфраструктури (відключення інтернету, блокування протоколів); масове виведення з ладу клієнтських додатків через оновлення з вірусом; атака на електромережу чи дата-центр, де розміщені сервери (фізичне вимкнення).
- Соціальна інженерія та атаки на інфраструктуру довіри. Сюди відносяться загрози, пов'язані не з алгоритмами голосування напряму, а з допоміжними компонентами і людським фактором. Наприклад, фішингові атаки на виборців чи адміністраторів: якщо нападник обманом змусить користувача встановити підробний додаток для голосування або перейти на фейковий веб-сайт, він може викрасти облікові дані чи перенаправити голоси.

Також це атаки на сервери реєстрів виборців: злом бази даних виборців може дозволити додати фіктивних виборців чи видалити справжніх (внаслідок чого деякі люди не зможуть проголосувати, а «мертві душі» проголосують замість них).

- Атаки інсайдерів. Не можна виключати загрозу, коли частина системи скомпрометована зсередини – напряму або опосередковано. Це може бути адміністратор, який має привілейований доступ до серверів або ключів і який навмисне (або через вимушений тиск) змінює дані. Або ж розробник, що залишив бекдор у коді. Навіть член виборчої комісії, який має право запускати процес підрахунку чи доступ до проміжних результатів, – потенційне джерело загроз.
- Підрив легітимності та довіри. Окрема категорія загроз – цілеспрямовані дії, метою яких є не стільки вплинути на конкретний результат, скільки скомпрометувати саму ідею виборів [10]. Наприклад, атака, що залишає сліди втручання (навіть без масштабної фальсифікації), може бути використана для заяв: «систему зламано, результати недійсні». Такі дії можуть вчинятися, щоб викликати хаос, перевибори або політичну кризу. До цього ж можна віднести масове поширення дезінформації про роботу системи, фейкових доказів «злому» тощо.

На основі цієї моделі загроз формуються вимоги до системи: збереження таємниці голосів, цілісності даних, доступності сервісів, верифіковуваність результатів (можливість перевірити відсутність фальсифікацій), автентифікація виборців і захист від повторного голосування, стійкість до вірусних атак, захист від внутрішнього порушника тощо [11]. У наступному пункті розглянемо модель зловмисника – тобто, хто саме може реалізувати зазначені загрози і які в нього потенційно є можливості.

1.4 Модель зловмисника

Модель зловмисника описує потенційних атакувальників системи, їхні типи, мотивацію та ресурси [12]. Розглянемо основні типи зловмисників:

1) Зовнішній хакер-одинак або мала група. Це можуть бути технічно обізнані особи, що діють самостійно або невеликою командою, без підтримки держави чи великої організації [13]. Мотиви – хуліганство, особистий інтерес, фінансова вигода (напр. заради викупу чи за наймом). Ресурси таких зловмисників обмежені, але вони можуть скористатися загальнодоступними інструментами злому, шкідливим ПЗ з даркнету, ботнет-мережами для DDoS-атак напрокат тощо.

2) Внутрішній зловмисник (інсайдер). Це особа, що має законний доступ до системи з середини: адміністратор серверів, член виборчої комісії з правами доступу до проміжних даних, розробник системи, співробітник центру сертифікації, що випускає електронні ключі тощо. Мотиви можуть бути різні – від ідеологічних (бажання вплинути на результат виборів) до корисливих (хабар за фальсифікацію) чи вимушених (шантаж з боку інших сил). Інсайдер небезпечний тим, що має поглиблені знання про систему і можливість обійти деякі зовнішні рівні захисту. Наприклад, адміністратор може прямо змінити базу даних з голосами, якщо немає належних механізмів мульти-підпису або логування; розробник може вставити бекдор ще на етапі створення ПЗ [14].

3) Організована злочинна група або комерційна структура. Це сценарій, коли атакою займається група фахівців, можливо найнятих кимось, з метою вплинути на вибори [15]. Ресурси у них більші, може бути фінансування на придбання 0-day експлойтів, підкуп інсайдерів, розгортання великих ботнетів. Мотив зазвичай фінансовий або на замовлення третіх осіб (наприклад, нечесний кандидат чи бізнес, зацікавлений у певному результаті референдуму). Така група може поєднувати різні вектори атак: одночасно і кібератаки, і підкуп виборців або членів комісій, і пропаганду для дезорієнтації.

4) Державний або квазі-державний суб’єкт (внутрішній чи зовнішній). Це найпотужніший тип зловмисника. Сюди входять іноземні розвідки, які можуть намагатися вплинути на вибори в іншій країні, або

авторитарна влада в середині країни, що бажає контролювати результати. Ресурси практично необмежені: висококваліфіковані спеціалісти, великі фінанси, доступ до апаратних закладок, можливість тиску на компанії-розробники чи хмарних провайдерів, навіть можливості на рівні інфраструктури інтернету (перехоплення трафіку на магістралях, компрометація сертифікаційних центрів і т.п.). Мотив – геополітичний або збереження влади. Такий атакувальник може використати невідомі науці вразливості (zero-day exploits), кіберарсенал на кшталт державних APT-груп, і діяти дуже приховано [16]. Він може атакувати ланцюжок постачання (supply chain) – наприклад, впровадити бекдор в криптографічну бібліотеку, якою користується система, задовго до виборів.

5) Несвідомий зловмисник (помилка або випадкова подія). Формально це не «зловмисник», але при моделюванні треба врахувати, що система може постраждати від ненавмисних дій або збігів обставин: програмні збої, людські помилки адміністраторів, збої обладнання, випадкове відключення електрики саме в день виборів, природні катаклізми. Хоч це і нецілеспрямовані дії, їх вплив аналогічний атаці на доступність чи цілісність.

В сукупності, ці образи зловмисників визначають рівень захисту, який необхідно забезпечити. Наприклад, якщо система позиціонується для загальнонаціональних виборів, вона *мусить* бути стійкою хоча б до ключових методів атаки державного актору (інакше використання такої системи небезпечне). Якщо ж йдеться про голосування в невеликій організації, модель зловмисника може вважати малоймовірною участь іноземних спецслужб, але все одно повинна врахувати внутрішнього адміністратора і звичайних хакерів.

Для децентралізованої блокчейн-системи специфічними є зловмисники, що атакують сам механізм консенсусу [17]. Наприклад, «атака 51» – якщо зловмисник (чи змовна група) контролює більшість вузлів або частку стейку (в Proof-of-Stake системі), він може переписати історію блокчейну.

1.5 Перспективи розвитку систем децентралізованого електронного голосування і постановка задач розробки

Незважаючи на окреслені проблеми, електронне голосування продовжує розвиватися, і новітні технології дають надію на підвищення його безпечності та надійності. Одним з перспективних напрямів є саме децентралізовані системи електронного голосування, зокрема на основі блокчейну та розподілених реєстрів. Їхній головний потенціал – усунення єдиної точки відмови та концентрованої довіри. Ідея полягає в тому, що жоден окремий вузол чи орган не контролює всі дані виборів; натомість, підтвердження результатів є колективним, а будь-яка підробка залишить сліди в розподіленому леджері. Уже сьогодні реалізовано кілька пілотів блокчейн-голосування (Швейцарія, Японія, Канада на муніципальному рівні), які показують технічну можливість такого підходу [18] [19].

Перспективи розвитку децентралізованих е-виборчих систем охоплюють кілька напрямів:

- Удосконалення криптографічних методів. Сучасна наука пропонує інструменти, що можуть забезпечити одночасно і таємницю, і перевірюваність голосування. Зокрема, методи гомоморфного шифрування дозволяють рахувати зашифровані голоси, не розшифровуючи їх (тобто отримати загальний результат, не знаючи індивідуальних виборів). Це означає, що навіть якщо зловмисник отримає доступ до бази голосів, але без ключа розшифровки – він не дізнається, хто як проголосував, а підсумок можна перевірити альтернативним шляхом. Також активно досліджуються zero-knowledge proofs (докази з нульовим розголошенням) для виборчих задач – наприклад, доказ того, що зашифрований бюлетень справді містить допустимий вибір і був врахований у сумі, без розкриття самого вибору. Усі ці методи поки що складні у реалізації на великому масштабі, але поступово переходять з теоретичної площини в практичні протоколи [20].
- Комбінування з фізичними гарантіями. Перспективним є гібридний підхід, коли електронне голосування поєднується з фізичними підтвердженнями. Наприклад, система VVPAT – це коли кожен електронний

голос супроводжується роздруком на папері, який виборець може перевірити і який зберігається для можливого перерахунку. У контексті децентралізації можна передбачити, що кожен вузол мережі зберігатиме частковий паперовий архів або що виборцю видаватиметься зашифрований паперовий чек, який неможливо підробити без знання ключа. Ця концепція зараз обговорюється, адже папір слугує довіреним резервом, до якого можна звернутися, якщо електронні дані поставлено під сумнів. Інший фізичний рівень – спеціалізовані пристрої для голосування (hardware tokens, смарт-карти, захищені планшети для голосування).

- Стандартизація і незалежна перевірка. Щоб технології е-голосування стали зрілими, потрібні міжнародні стандарти безпеки і прозорості. Наразі в різних країнах використовуються різні протоколи, часто пропрієтарні рішення. Перспективою є розробка відкритих стандартів (стандарт TLS для шифрування в інтернеті) для виборчих систем. Вже з’являються ініціативи проєктів IEEE щодо стандартів для end-to-end verifiable voting. Стандарти повинні охоплювати модель загроз, вимоги до криптографії, вимоги до аудиту, процедури сертифікації системи перед виборами. Одночасно, практика відкритих тестувань (public penetration testing) перед впровадженням – теж тенденція: Швейцарія, наприклад, оголошувала публічні тестування свого e-boting з винагородами за знайдені вразливості. Такий підхід дозволяє виявити проблеми до реального голосування і підвищує довіру суспільства, адже залучаються незалежні експерти. У майбутньому можна очікувати, що децентралізовані системи голосування з відкритим вихідним кодом стануть нормою, а їхня безпека буде підтверджуватися спільнотою фахівців.

Покращення інтерфейсів і процесів для користувачів. Не менш важливий напрям – зробити електронне голосування зрозумілим і зручним для виборців та адміністраторів. Це включає локалізацію інтерфейсів різними мовами, адаптацію для людей з інвалідністю, інтуїтивно зрозумілу подачу інформації про кандидатів, підтвердження вибору тощо. Також необхідно навчання користувачів безпечним

практикам (не повідомляти нікому свої облікові дані, перевіряти сертифікати сайту тощо). Для адміністраторів – інструменти моніторингу процесу виборів у реальному часі, виявлення аномалій (наприклад, раптовий наплив тисяч голосів з однієї IP-адреси), функції екстреного реагування (продовжити час голосування на окремих дільницях у разі збою) тощо [21].

2 ТЕОРЕТИЧНІ ЗАСАДИ ПОБУДОВИ РОБОЧОГО МІСЦЯ АДМІНІСТРАТОРА ДЕЦЕНТРАЛІЗОВАНОЇ СИСТЕМИ ГОЛОСУВАННЯ

2.1 Роль адміністратора в системі електронного голосування

Адміністратор відіграє ключову роль у забезпеченні працездатності та надійності системи електронного голосування. У децентралізованій моделі адміністратор не є єдиним керівним центром, однак саме він виконує організаційні та технічні завдання, що забезпечують запуск та підтримку голосувань [22].

Основні функції адміністратора включають: – створення та конфігурацію нових голосувань (визначення назви, опису, дати, варіантів вибору); – формування списків учасників голосування із зазначенням їх ідентифікаційних параметрів; – визначення власників вузлів p2p-мережі та призначення кількості нод для кожного з них; – налаштування параметрів консенсусу, які визначають алгоритм досягнення згоди між вузлами системи (наприклад, PoS, PoA); – ініціалізація процесу поширення конфігураційних даних через p2p-мережу [23]; – контроль реєстрації та ідентифікації користувачів, включаючи інтеграцію з державними сервісами (Дія, BankID); – контроль журналів подій, аудит системних змін, реагування на інциденти; – забезпечення конфіденційності та цілісності даних у середовищі, де адміністратор не може повністю довіряти іншим учасникам У деяких реалізаціях адміністратор може також відповідати за ініціалізацію криптографічних ключів та розподілення сертифікатів доступу.

У класичних централізованих системах адміністратор мав практично необмежені повноваження щодо обробки голосів, перегляду результатів і керування системою. Натомість у децентралізованих платформах важливо розмежовувати доступи та мінімізувати довіру до одного оператора, аби запобігти зловживанням [24].

У деяких сучасних рішеннях, голосування не вимагає постійного контролю з боку адміністратора – всі результати автоматично зберігаються в блокчейні, що мінімізує ризики маніпуляції. Проте адміністратор залишається відповідальним за

конфігурацію параметрів перед початком голосування, а також за технічну підтримку процесу і допомогу користувачам [25].

Таким чином, роль адміністратора не зводиться до простої технічної підтримки: це стратегічна фігура, що координує взаємодію учасників мережі, забезпечує дотримання технічних, безпекових та етичних стандартів. При цьому реалізація функціоналу адміністратора повинна відповідати принципу «мінімально необхідного доступу» (least privilege principle), щоб уникати потенційних ризиків маніпуляції.

2.2 Вимоги до функціональності адміністративного інтерфейсу

Інтерфейс адміністратора має бути функціонально повним, інтуїтивно зрозумілим, захищеним і придатним для роботи з критично важливими даними [26]. Його структура повинна враховувати як базові адміністративні дії, так і специфіку децентралізованого середовища.

До ключових функціональних вимог можна віднести:

– гнучке створення голосувань, адміністратор повинен мати можливість створювати голосування з різними типами бюлетенів (один вибір, декілька виборів, ранжування), а також вказувати дату початку, тривалість, опис та умови участі;

– формування та управління списками учасників, передбачено імпорт списків із зовнішніх джерел (CSV, бази даних), призначення унікальних ідентифікаторів, а також можливість ручного додавання або блокування користувачів;

– налаштування параметрів безпеки, передбачено вибір типу автентифікації (наприклад, BankID, Дія, логін/пароль, електронний підпис), налаштування рівнів доступу, багатофакторну автентифікацію, моніторинг спроб несанкціонованого входу;

– інтеграція з інфраструктурою блокчейн або p2p, інтерфейс має дозволяти адміністратору зв’язуватись із вузлами, ініціювати синхронізацію, переглядати статус вузлів, а також вручну перезапускати компоненти у разі збоїв;

– конфігурація алгоритмів консенсусу, для адміністраторів передбачено вибір (або перегляд) механізму досягнення згоди в мережі: Proof-of-Work, Proof-of-

Authority, Proof-of-Stake, тощо. Інтерфейс повинен дозволяти налаштувати частку голосу кожного вузла відповідно до обраної моделі [27];

– Аудит та звітність, повний журнал дій адміністратора, включаючи дату, час, IP-адресу та суть події. Передбачено експорт логів у зашифрованому вигляді. Також – автоматичне створення підсумкових звітів про кількість користувачів, поданих голосів, інциденти тощо;

– інтерфейс взаємодії з користувачами, можливість надсилати повідомлення учасникам голосування (наприклад, push- або email-нотифікації), запускати тестові голосування для перевірки системи, змінювати параметри до моменту старту [28].

Крім технічних аспектів, важливою є ергономічність інтерфейсу –розміщення елементів управління, зручна панель навігації, кольорові акценти для швидкого розпізнавання статусів та повідомлень, підтримка темної та світлої тем. Наявність системи підказок (tooltips), автоматичного збереження конфігурацій, підтвердження критичних дій (наприклад, видалення голосування) –усе це позитивно впливає на якість адміністрування та знижує ризик помилок при роботі.

У контексті безпеки, доцільним є впровадження концепції Zero Trust: навіть адміністратор повинен проходити постійну верифікацію при виконанні чутливих операцій. Це може включати повторну автентифікацію, підтвердження через мобільний пристрій або тимчасове посилення політик доступу на час критичних змін [29].

Також важливо передбачити можливість журналювання доступу до інтерфейсу та спроб його використання в режимі офлайн. Це дозволяє виявити спроби компрометації середовища або несанкціоновані дії зі сторони локального персоналу, що має фізичний доступ до пристрою.

Останнім критичним елементом є масштабованість –інтерфейс має зберігати стабільність і продуктивність навіть у випадку одночасної конфігурації кількох сотень вузлів або тисяч користувачів. Це передбачає ефективну роботу з базою даних, використання кешування, асинхронну обробку запитів і оптимізовану побудову логіки інтерфейсу на рівні інтерфейсу користувача [30].

2.3 Особливості побудови децентралізованих систем голосування

Децентралізовані системи електронного голосування відрізняються від традиційних централізованих моделей тим, що не мають єдиного серверного центру, який обробляє усі дані. Натомість інформація зберігається та обробляється у розподіленій мережі вузлів (нод), що взаємодіють між собою без посередника. Це значно підвищує стійкість системи до зовнішніх атак, цензури та фальсифікацій.

Ключовою технологією, що забезпечує децентралізацію, є блокчейн. Вона дозволяє записувати всі транзакції (в контексті голосування –подані голоси) у вигляді послідовного ланцюга блоків, кожен з яких містить хеш попереднього. Це робить зміну або видалення даних практично неможливим без згоди більшості вузлів.

Крім блокчейну, у децентралізованих голосуваннях можуть застосовуватись технології IPFS для розподіленого зберігання, а також протоколи типу Libp2p або WebRTC для прямого зв’язку між вузлами. Це дозволяє обходити традиційні сервери і забезпечувати більшу конфіденційність та відмовостійкість системи [31].

У таких системах важливо правильно обрати механізм досягнення консенсусу між вузлами. Найпоширеніші підходи включають: – Proof-of-Work (PoW) – вимагає від вузлів виконання складних обчислень для підтвердження транзакцій; – Proof-of-Stake (PoS) – базується на кількості монет або ресурсів, які учасник готовий «поставити» для участі в підтвердженні; – Proof-of-Authority (PoA) – передбачає, що лише довірені вузли мають право підтвердження транзакцій [32].

Децентралізована архітектура має низку переваг: – відсутність єдиного пункту відмови; – прозорість усіх операцій для користувачів; – складність зовнішнього втручання без зламу більшості вузлів; – можливість автоматизованого аудиту через відкриті дані блокчейну. Разом із тим існують і виклики: складність масштабування, висока вимогливість до пропускної здатності мережі, проблема затримок при підтвердженні транзакцій, а також потреба у спеціалізованих знаннях для адміністрування. У випадку голосування це означає необхідність проектувати системи таким чином, щоб навіть складна блокчейн-інфраструктура залишалась «прозорою» для звичайного користувача і зрозумілою для адміністратора [33].

Таким чином, децентралізація є перспективним напрямом розвитку систем голосування, що поєднує високу стійкість до атак, прозорість процесів та надійне збереження результатів, однак потребує комплексного технічного підходу до побудови інтерфейсів, безпеки та синхронізації даних.

2.4 Архітектура децентралізованої системи голосування

Архітектура децентралізованої системи голосування визначає логічну та фізичну організацію її компонентів, їхні зв’язки, ролі та функції у процесі підготовки, проведення й завершення виборів. Основною метою побудови архітектури є забезпечення цілісності, конфіденційності, доступності та прозорості електронного волевиявлення [34].

Типова децентралізована система складається з таких ключових елементів:

– клієнтський модуль виборця – веб або мобільний інтерфейс, через який учасник подає голос. Має функції автентифікації, перевірки виборчих прав та шифрування голосу;

– клієнт адміністратора – інтерфейс для створення виборів, внесення списків учасників, налагодження вузлів, налаштування консенсусу;

– вузли p2p-мережі (ноди) – окремі сервери або комп’ютери, які зберігають, ретранслюють та підтверджують дані голосування. Вони працюють синхронно, обмінюючись блоками та повідомленнями через захищений канал зв’язку [35];

– блокчейн-реєстр – ланцюг транзакцій (блоків), у якому фіксуються всі голоси. Використовується для гарантування незмінності й відкритої перевірки результатів;

– сховище метаданих (наприклад, IPFS) – для збереження додаткових даних: списків виборців, конфігурацій, повідомлень системи;

– модуль консенсусу – набір алгоритмів, що дозволяють вузлам досягати згоди щодо валідності голосів (PoS, PoA тощо);

У цій архітектурі критично важливим є забезпечення:

– відмовостійкості: кожен вузол зберігає копію даних, а тому збій одного елемента не порушує функціонування всієї мережі;

– криптографічного захисту: голос шифрується на стороні клієнта, передається у мережу у зашифрованому вигляді, підписується цифровим підписом;

– перевірюваності: учасник може переконатися, що його голос враховано, не розкриваючи змісту вибору;

– масштабованості: архітектура повинна підтримувати розширення кількості вузлів без зниження продуктивності.

Таким чином, архітектура визначає не лише технічну побудову системи, але і логіку взаємодії суб'єктів процесу голосування. Її проєктування вимагає врахування вимог безпеки, юзабіліті, незалежності компонентів і можливості незалежної перевірки [36].

2.5 Проєктування інтерфейсу адміністратора

Проєктування інтерфейсу адміністратора децентралізованої системи голосування має враховувати не лише зручність використання, але й безпекові аспекти, надійність і ефективність взаємодії з іншими компонентами системи. Інтерфейс є центральним елементом управління, тому від його якості залежить як коректність конфігурації голосувань, так і захищеність усіх процесів [37].

Перш за все, необхідно розробити логічну структуру інтерфейсу. Вона може складатися з таких основних модулів:

– Панель огляду: загальний дашборд з поточними голосуваннями, статусами вузлів, повідомленнями безпеки.

– Модуль створення голосування: форми для введення назви, опису, типу голосування, дати початку і завершення, завантаження списків учасників.

– Керування вузлами: відображення активних вузлів, інформація про їхній стан, опція перезапуску, оновлення конфігурацій.

– Налаштування консенсусу: вибір алгоритму (PoA, PoS), встановлення ваги кожного вузла, перегляд журналу рішень.

– Звіти та аудит: доступ до логів дій, аналітика за результатами голосування, експортування.

– Службова інформація: моніторинг доступів, логінів, спроб автентифікації, збоїв та технічних сповіщень.

Інтерфейс має бути побудований з урахуванням принципів юзабіліті, зокрема:

– послідовність розташування елементів;

– зрозумілі назви та підказки;

– підтвердження перед критичними діями;

– мінімальна кількість кроків для виконання ключових завдань.

З точки зору безпеки, обов'язковими є: – обмеження доступу до окремих розділів за ролями; – використання захищених протоколів (HTTPS, TLS); – журналювання усіх змін конфігурацій та голосувань; – контроль часу сесій і повторна автентифікація після періоду бездіяльності [38].

Інтерфейс повинен бути адаптивним – доступним як із десктопів, так і з мобільних пристроїв, у тому числі через мобільний застосунок. Важливою є також підтримка мультимовності для можливого використання на міжнародному рівні. Таким чином, проєктування інтерфейсу адміністратора – це багаторівневе завдання, що вимагає узгодження естетики, зручності, продуктивності й захисту. Результат має бути інтерфейс, який дозволяє керувати голосуваннями швидко, безпечно й без помилок навіть у критичних ситуаціях.

2.6 Моделі зберігання та обміну даними у p2p-мережах

Однією з ключових технічних основ децентралізованих систем є організація зберігання та обміну даними між вузлами (нодами) у p2p-мережі. У системах електронного голосування це має критичне значення, оскільки гарантує стійкість до втрат даних, незалежність від єдиного центру, а також безпеку й перевірюваність переданих повідомлень [39].

Основні моделі обміну даними у p2p-середовищі зображені на таблиці 2.1:

– Push-модель – вузли надсилають повідомлення до інших учасників одразу після їх генерації. Підходить для трансляції подій (наприклад, надходження голосу) в реальному часі.

– Pull-модель – вузли періодично перевіряють наявність нових даних у сусідів. Забезпечує зменшення навантаження на мережу, але знижує оперативність.

– Гібридна модель – поєднує push- і pull-підходи: критичні події передаються негайно, а другорядні оновлення –за запитом

У контексті зберігання часто застосовуються такі моделі:

– Full replication – кожен вузол зберігає повну копію всіх блоків і метаданих. Забезпечує високу надійність, але потребує великих ресурсів.

– Sharding – дані розділяються між групами вузлів, кожна з яких відповідає за свою частину. Підвищує масштабованість, але ускладнює верифікацію.

– Distributed Hash Table (DHT) – децентралізований індекс, за яким вузли зберігають і знаходять дані за хешами. Застосовується в IPFS і BitTorrent.

Таблиця 2.1 - Моделі зберігання та обміну даними

Модель	Переваги	Недоліки
Full replication (Повна реплікація)	Забезпечує максимальну надійність	Спричиняє значне навантаження на пам’ять кожного вузла
Sharding (Шардінг)	Підвищує масштабованість системи за рахунок розподілу даних між вузлами. Значно економить ресурси	Ускладнює верифікацію цілісності даних, оскільки інформація зберігається фрагментовано
DHT (Distributed Hash Table – Розподілена хеш-таблиця)	Забезпечує гнучкий механізм пошуку та зберігання даних за ключем (хешем), особливо ефективний у великих p2p-мережах	Ефективність залежить від точності розподілу ключів і рівномірності навантаження між вузлами

Для зберігання великих обсягів виборчих документів, бюлетенів або логів доцільним є використання IPFS –протоколу, який дозволяє зберігати контент за його

хешем, створюючи децентралізовану файлову систему. У такій системі документи неможливо змінити без зміни хешу, що гарантує їхню цілісність.

Також можливо комбінувати блокчейн (для запису голосів) з IPFS (для зберігання списків, журналів, конфігурацій), використовуючи хеші як посилання між структурами даних. Це дозволяє зменшити об’єм блокчейну, зберігаючи при цьому доступність даних [40].

Надійність p2p-зберігання підвищується шляхом: – шифрування кожного фрагмента даних; – дублювання файлів на декількох вузлах; – перевірки контрольних сум перед відтворенням інформації; – автоматичного оновлення вмісту через підписані транзакції.

Таким чином, p2p-зберігання та обмін у децентралізованому голосуванні – це поєднання структурованих та реплікованих даних, криптографічного контролю та масштабованого транспорту, який дає змогу досягти безпеки та відмовостійкості без центрального вузла.

3. БЕЗПЕКА ТА АКТУАЛЬНІСТЬ СИСТЕМИ ГОЛОСУВАННЯ

3.1 Актуальність використання захищених систем голосування

У сучасному світі питання довіри до виборчих процесів набуло особливого значення. У багатьох країнах, включаючи Україну, спостерігається зниження довіри громадян до результатів голосувань, особливо в умовах війни, внутрішньої нестабільності, інформаційних атак і втручання зовнішніх сил. За даними дослідження IFES, понад 40% виборців у країнах, що розвиваються, сумніваються в прозорості національних виборів [41].

У таких умовах забезпечення прозорості, достовірності та доступності виборів стає критично важливим. Децентралізовані системи електронного голосування пропонують рішення, яке дозволяє забезпечити високу стійкість до фальсифікацій, технічних збоїв і навмисного втручання. На відміну від централізованих систем, де результат голосування часто залежить від цілісності одного серверного вузла, децентралізовані рішення базуються на мережі незалежних нод. Це дає змогу знизити ризик маніпуляцій і підвищити довіру до результатів. Особливої актуальності такі системи набувають в умовах: – військового або надзвичайного стану, коли фізичне голосування може бути неможливим; – великої кількості виборців за кордоном (мігранти, біженці); – широкої цифровізації державних сервісів (наприклад, «Дія»); – потреби оперативного проведення голосувань у громадах, освітніх чи корпоративних установах.

На додачу, відповідно до досвіду Естонії, де вже кілька років функціонує система інтернет-голосування, понад 50% громадян надають перевагу онлайн-участі у виборах, і рівень задоволеності такими системами є високим. Також не менш важливим є фактор молодого покоління виборців, яке очікує зручного цифрового сервісу. Платформи з сучасним інтерфейсом, інтеграцією з мобільними технологіями та онлайн-доступністю мають значно більший шанс бути прийнятими та використаними широкими масами [42].

3.2 Основні принципи побудови захищеної системи

Забезпечення безпеки в електронному голосуванні базується на фундаментальних принципах інформаційної безпеки. Основу становить концепція CIA-трикутника, яка охоплює:

– Конфіденційність (Confidentiality): гарантує, що голос кожного виборця не буде розкритий третім особам. Для цього використовуються шифрування, анонімізація, одноразові токени та захищені канали зв'язку (TLS, VPN).

– Цілісність (Integrity): передбачає, що голос не може бути змінений або викривлений після його подання. Цього досягають за допомогою криптографічних хеш-функцій, цифрових підписів та перевірки контрольних сум блоків даних.

– Доступність (Availability): система повинна працювати безперервно, навіть за умови атаки чи технічних збоїв. Для цього використовують реплікацію даних, балансування навантаження, відмовостійкі p2p-архітектури.

Окрім базових принципів, у системі електронного голосування критично важливими є:

– анонімність – зв'язок між виборцем та його голосом не повинен зберігатися, щоб уникнути тиску або маніпуляцій. Анонімність реалізується через застосування сліпих підписів, мікшування транзакцій та гомоморфне шифрування;

– невідмовність (Non-repudiation) – жоден учасник системи не може заперечити свою дію (наприклад, подання голосу), що забезпечується цифровим підписом та логуванням дій у блокчейні;

– верифікованість (Verifiability) – кожен виборець повинен мати змогу впевнитися, що його голос був врахований і не змінений, не розкриваючи змісту вибору. У системах на базі блокчейну це реалізується за рахунок прозорості ланцюга блоків;

Усі ці принципи повинні реалізовуватися як на рівні інтерфейсу, так і в архітектурі зберігання та обробки даних. Їх недотримання може призвести до втрати довіри до всієї системи – навіть якщо голоси були враховані технічно правильно.

3.3 Технології криптографічного захисту голосування

Криптографія є фундаментом безпеки електронного голосування. Вона забезпечує конфіденційність голосів, перевірку справжності повідомлень, захист від фальсифікацій та гарантує цілісність усіх операцій у системі [43].

Серед основних технологій, що застосовуються у криптографічному захисті електронного голосування:

– Асиметричне шифрування (RSA): дозволяє зашифрувати голос на стороні клієнта і розшифрувати лише у призначеній точці підрахунку. Публічний ключ використовується для шифрування, приватний –для розшифрування. Це забезпечує односторонню безпеку обміну.

– Цифровий підпис: гарантує, що повідомлення (голос, журнал дій, конфігурація) було створене саме визначеним користувачем і не було змінене. Цей механізм також забезпечує юридичну відповідальність (невідмовність).

– Гомоморфне шифрування: дозволяє виконувати математичні операції (наприклад, підрахунок голосів) над зашифрованими даними без їх розшифровки. Це забезпечує анонімність і прозорість водночас. Ця технологія активно досліджується для використання в системах електронного голосування, де надзвичайно важливою є незалежна перевірка підсумків без розкриття змісту голосів.

– Сліпі підписи (blind signatures): дозволяють серверу підписувати голос без знання його змісту. Використовується для збереження анонімності при верифікації. Такі підписи уперше були запропоновані Дейвідом Чаумом і з того часу застосовуються в різноманітних цифрових протоколах для уникнення простежуваності дій користувачів.

– Одноразові токени та QR-коди: використовуються як спосіб підтвердження участі без розкриття особистих даних. Можуть бути застосовані у поєднанні з мобільними застосунками. У контексті голосування вони можуть видаватися через застосунок або SMS і використовуватися один раз для входу чи підтвердження дії.

– Криптографічні хеш-функції (SHA-3, Blake2): забезпечують перевірку цілісності даних, допомагають виявити зміну або фальсифікацію бюлетенів чи

записів у блокчейні. Вони широко використовуються для створення цифрових відбитків (fingerprints), які зберігаються в блокчейні разом із транзакціями або голосами.

Застосування криптографії не є лише технічним аспектом – це ключовий елемент побудови довіри користувачів до системи. Кожен виборець має знати, що його голос не буде викрадений, підмінений або відкритий. У добре спроєктованій децентралізованій системі криптографія поєднується з протоколами консенсусу, механізмами автентифікації та мережевим захистом для формування повної моделі цифрової безпеки.

3.4 Методи автентифікації та авторизації в електронному голосуванні

Автентифікація та авторизація є фундаментальними процесами, що визначають безпеку та легітимність електронного голосування. Їх головна мета – гарантувати, що лише зареєстровані користувачі мають право доступу до системи, а також забезпечити, щоб кожен учасник виконував лише ті дії, які дозволені його роллю. Надійна автентифікація запобігає участі сторонніх осіб у голосуванні, фальсифікації результатів і багаторазовому голосуванню. У сучасних умовах зростаючої цифровізації найефективнішими є мультифакторні системи автентифікації, які поєднують декілька механізмів перевірки: знання (пароль), володіння (пристрій або токен) і біометричні дані (відбиток пальця, розпізнавання обличчя) [44].

В Україні зручним і широко підтримуваним рішенням є інтеграція з державними сервісами, зокрема BankID та Дія ID. BankID дозволяє підтвердити особу через обслуговуючий банк, не розкриваючи зайвих персональних даних. Дія ID, у свою чергу, забезпечує просту автентифікацію через мобільний застосунок із можливістю накладення електронного підпису. Електронний цифровий підпис (ЕЦП) надає найвищий рівень захисту і може використовуватися для підтвердження поданого голосу з юридичною значущістю. Водночас, через складність використання, його застосування доцільне переважно для адміністративних або критичних дій.

У міжнародній практиці поширеним підходом є використання протоколів OAuth 2.0 та OpenID Connect, які дозволяють автентифікуватися через зовнішні платформи – Google, Facebook, державні портали [45]. Це особливо ефективно для неформальних або внутрішніх голосувань. Двофакторна автентифікація додає додатковий рівень захисту, вимагаючи підтвердження входу через SMS-код або застосунок-генератор кодів. Біометричні методи, як-от сканування обличчя або пальців, забезпечують високу унікальність доступу, хоча і вимагають відповідного обладнання.

Після автентифікації важливо застосувати механізми авторизації –визначити, які саме дії доступні кожному користувачу. Для цього застосовують наступну модель: виборець може лише голосувати, адміністратор – створювати голосування та переглядати звіти, а спостерігач – лише переглядати хід процесу. Усі ці дії повинні бути зафіксовані в журналі подій – лог-файлах, які дають змогу відстежувати підозрілі дії та порушення політик безпеки [46].

3.5 Захист p2p-інфраструктури від мережевих атак

Однією з найбільших переваг децентралізованих систем голосування є їхня відмовостійкість і розподіленість. Проте саме ці характеристики роблять їх привабливою ціллю для низки мережевих атак. У системах на основі p2p-архітектури безпека вузлів, каналів зв’язку та алгоритмів взаємодії між ними набуває критичного значення. Насамперед, необхідно захистити мережу від атаки типу DoS/DDoS, яка спрямована на виведення з ладу окремих вузлів або перевантаження каналів передачі даних. Захист досягається за допомогою фаєрволів, систем виявлення аномалій трафіку, а також застосуванням розподілу навантаження між вузлами. У блокчейн-системах також використовуються механізми адаптивного обмеження запитів (rate-limiting) та автоматичного блокування шкідливої активності [47].

Другим важливим аспектом є протидія атаці «людина посередині» (MITM), коли зловмисник намагається перехопити або змінити передані між вузлами повідомлення. Для запобігання цьому необхідно шифрувати весь мережевий трафік

з використанням TLS або аналогічних протоколів, а також застосовувати перевірку автентичності вузлів через цифрові сертифікати. Особливу небезпеку становлять атаки на цілісність даних, коли один або кілька вузлів можуть передавати сфальсифіковану інформацію, видаючи її за правдиву. Для захисту від цього в системі реалізується механізм досягнення консенсусу: жодне оновлення не вважається дійсним, поки не буде підтверджено більшістю вузлів. Це може бути реалізовано через алгоритми типу PoA, BFT (Byzantine Fault Tolerance) або інші стратегії, адаптовані до особливостей системи голосування.

Ще однією актуальною загрозою є спроби контролю більшості вузлів мережі (атака 51%) [48]. Щоб уникнути цього, важливо дотримуватися принципів децентралізації: розподіляти вузли між незалежними власниками, контролювати географічне розташування та використовувати репутаційні системи для оцінки надійності учасників.

Також слід враховувати ризики з боку фізичного доступу до вузлів. У разі встановлення нод на незахищених пристроях виникає загроза компрометації всієї мережі. Для зниження ризиків застосовуються методи апаратного шифрування, безпечного завантаження (secure boot) і постійного моніторингу стану системи.

Крім того, важливим напрямом є журналювання подій (audit logging) – фіксація кожної взаємодії вузлів, результатів голосування, відхилених транзакцій тощо. Це дозволяє проводити післяінцидентний аналіз і вчасно виявляти аномальні дії. Захист p2p-інфраструктури вимагає комплексного підходу: від базової мережевої безпеки до криптографічного забезпечення зв’язку, від організаційних заходів до постійного аудиту. Лише комбінація технічних і процедурних механізмів забезпечує реальну стійкість системи голосування до сучасних загроз.

4 РОЗРОБКА РОБОЧОГО МІСЦЯ АДМІНІСТРАТОРА СИСТЕМИ ГОЛОСУВАННЯ

4.1. Постановка задачі та вимоги до автоматизованого робочого місця

На основі аналізу сучасних підходів до реалізації електронного голосування, а також огляду вже існуючих систем, сформульовано технічне завдання на створення веб-застосунку – робочого місця адміністратора децентралізованої системи захищеного голосування. Основна мета – створити інтуїтивно зрозумілий, функціональний інтерфейс, який дозволить адміністратору повністю контролювати всі етапи підготовки до виборчого процесу в p2p-мережі.

Завдання, реалізовані у межах розробки:

- реєстрація адміністратора, у системі реалізовано базову автентифікацію за допомогою логіну й паролю. Після успішного входу адміністратор отримує доступ до всіх функціональних розділів. У перспективі можлива інтеграція з сервісами електронної ідентифікації, такими як Дія ID або BankID;
- формування переліку голосувань, передбачена форма створення нового голосування, яка включає введення назви, опису, дати початку й завершення, типу доступу (відкрите або закрите), а також варіантів відповіді. Усі голосування зберігаються локально в масиві та синхронізуються з localStorage;
- формування переліку учасників, у системі реалізовано механізм додавання учасників вручну з можливістю вказати:
    - ПІБ або псевдонім;
    - ідентифікатор (ID, email);
    - роль у системі (виборець, спостерігач, оператор);
    - кількість нод, закріплених за учасником. Також реалізовано можливість видалення учасників.

- формування списку власників нод, цей функціонал реалізовано як частину модуля учасників: для кожного учасника можна задати кількість вузлів (нод), які він контролює. Ця інформація в майбутньому може використовуватись для розрахунку ваги голосу або формування структури участі в консенсусі.
- для визначення параметрв, створено окремий інтерфейс, у якому адміністратор може задати глобальні параметри голосування:
  - вибір алгоритму PoA, PoS, PoW;
  - відсоток кворуму (кількість голосів для прийняття рішення);
  - час створення блоку (інтервал синхронізації). Параметри зберігаються в об’єкті consensusParams, який використовується при кожному новому голосуванні.
- завантаження конфігурації, реалізовано модуль, що об’єднує всі дані в єдиний конфігураційний JSON-об’єкт. Його структура відповідає вимогам до передачі інформації у вузли p2p-мережі або інші системи. Дані можна переглянути у форматі, придатному до подальшого збереження або експорту.

Реалізація кожного модуля описана у наступних розділах, із прикладами структури даних, взаємодії компонентів та механізмами збереження інформації.

4.2. Реалізація модуля автентифікації адміністратора

Першим логічним кроком у реалізації системи стало створення модуля авторизації адміністратора. Його головна мета – обмежити доступ до внутрішніх функцій застосунку лише для довірених осіб, що мають відповідні повноваження. У поточній реалізації було впроваджено просту форму автентифікації, яка складається з полів для логіну та паролю. Дані перевіряються на відповідність фіксованим значенням (наприклад, логін: admin, пароль: 1234). Після введення коректної комбінації виконується оновлення стану isLoggedIn, що дає змогу

переглянути інші компоненти системи. Усі спроби доступу до модуля адміністрування до автентифікації блокуються.

Компонент авторизації реалізовано як AdminLogin.jsx, де:

- поля вводу обробляються через useState;
- при натисканні кнопки «Увійти» перевіряється правильність введених даних;
- у разі успіху відбувається доступ до головного компонента App.jsx, де розгортається вся логіка застосунку.

У межах інтерфейсу реалізовано простий візуальний зворотний зв'язок – при неправильному введенні логіну або паролю виводиться відповідне повідомлення. Крім того, після успішної авторизації користувач автоматично перенаправляється до модуля створення голосування. Важливо, що реалізований підхід дозволяє у майбутньому розширити цей компонент. Наприклад:

- підключити справжню внутрішню логіку застосунку із зберіганням користувачів у базі даних;
- використовувати токен-авторизацію для захищеної передачі сесії;
- додати двофакторну автентифікацію BankID/Дія ID.

Таким чином, створений модуль автентифікації забезпечує необхідний базовий рівень безпеки, відповідає принципам ізоляції доступу та дає змогу швидко розгортати систему в тестовому або навчальному середовищі.

4.3. Формування та керування голосуваннями

Модуль створення та керування голосуваннями є центральним елементом функціональності адміністративного інтерфейсу системи. Його реалізація забезпечує можливість створення, збереження, перегляду та видалення голосувань з урахуванням визначених параметрів.

Основу функціоналу становить компонент VotingForm.jsx, у якому реалізована форма створення нового голосування. Вона дозволяє адміністратору задати наступні параметри:

- назва голосування (обов'язкове текстове поле);

- опис голосування (необов’язково);
- час початку та завершення (вибір дати та часу);
- варіанти вибору (щонайменше два, можливість динамічно додавати чи видаляти);
- тип доступу – публічне або приватне;
- параметри консенсусу (додаються автоматично зі збереженого глобального стану).

Перед збереженням голосування проходить базову перевірку: непусте ім’я, принаймні дві опції, коректні дати. Після цього голосування додається до глобального масиву votings та записується у localStorage.

Всі збережені голосування відображаються на сторінці VotingList.jsx у вигляді табличного списку. Кожен запис містить такі дані:

- назва та опис голосування;
- дата початку та завершення;
- тип (відкрите/закрите);
- кількість варіантів для голосування;
- вказані параметри консенсусу (алгоритм, блок-тайм, кворум);
- кнопка для видалення голосування.

Компонент має адаптивний дизайн і дозволяє зручно переглядати створені голосування незалежно від роздільної здатності екрана. У майбутньому тут може бути додано:

- редагування існуючих голосувань;
- зміна статусу голосування (чернетка, активне, завершене);
- дублювання записів для створення нових на основі попередніх.

Уся логіка створення та виводу голосувань реалізована через React useState, useEffect та пропси, які передають функції між компонентами. Для кожного голосування автоматично додаються актуальні параметри консенсусу, задані адміністратором у ConsensusParams.jsx. Дані зберігаються у localStorage, що забезпечує незалежність від серверної інфраструктури. При бажанні систему можна доповнити REST API для зв’язку з сервером або базою даних. Також

реалізована можливість перегляду всієї структури даних голосувань у JSON-форматі на сторінці P2PUpload.jsx, що дає змогу підготувати інформацію до подальшого експорту або передачі до p2p-мережі.

4.4. Робота з учасниками голосування

Окрему важливу роль у системі виконує модуль керування учасниками голосування. Саме через нього адміністратор формує список осіб, які матимуть право участі у голосуванні. Цей список є критичним з точки зору прозорості, легітимності та безпеки голосування, особливо у децентралізованих рішеннях, де кожен учасник має можливість впливу на результат.

Функціональність реалізована через компонент Participants.jsx, який включає інтерфейс для ручного введення нових учасників. У формі можна вказати наступні параметри:

- ім’я або псевдонім користувача;
- унікальний ідентифікатор (наприклад, email або умовний ID);
- роль користувача у системі (виборець, спостерігач або оператор);
- кількість нод (вузлів), які контролює учасник.

Після натискання кнопки «Додати» учасник додається до локального масиву participants, який зберігається у localStorage. Одночасно учасник з’являється у таблиці нижче, де доступні наступні дії:

- перегляд основних параметрів;
- видалення учасника зі списку;
- у майбутньому – редагування параметрів і зміна статусу).

Інтерфейс побудований з урахуванням адаптивності та зручності користування. Всі елементи компонента мають валідацію: перевіряється, чи введено непорожні значення, чи не дублюється ID, чи введена кількість нод є числовою.

З технічної точки зору, логіка реалізована за допомогою хуків React (useState, useEffect) та передачі функцій як пропсів. Усі зміни у масиві participants автоматично зберігаються у локальному сховищі браузера, що забезпечує

незалежність від серверної інфраструктури. Компонент учасників тісно пов’язаний з іншими частинами системи. Зокрема, у майбутньому список учасників буде прив’язуватись до конкретних голосувань, а їх кількість нод – враховуватись у механізмах розрахунку консенсусу.

У перспективі модуль може бути розширений такими функціями:

- імпорт списку учасників із CSV або JSON;
- валідація через BankID/Дія;
- прив’язка учасника до окремих голосувань;
- ведення журналу змін і дій адміністратора;
- механізм підтвердження особи або електронного підпису.

Отже, модуль роботи з учасниками забезпечує повноцінну підтримку внесення, перегляду й базового керування ключовими суб’єктами голосування. Це створює надійну основу для масштабування системи та підвищення її безпечності й довіри.

4.5. Формування переліку власників нод та керування вузлами

У межах реалізації інтерфейсу адміністратора було також створено модуль, що дозволяє керувати вузлами децентралізованої мережі. У системах захищеного голосування на базі p2p-архітектури кожен вузол (нод) є важливою складовою інфраструктури: він відповідає за обробку, розповсюдження та збереження даних. Таким чином, адміністратор повинен мати можливість формувати та контролювати склад таких вузлів.

Реалізація цього модуля здійснена шляхом додавання атрибуту «кількість нод» до кожного учасника у компоненті Participants.jsx. Таким чином, кожному користувачу (виборцю, оператору або спостерігачу) може бути призначена певна кількість нод, які він контролює. Це значення відображається у таблиці користувачів та може бути змінене або видалене адміністратором.

Такий підхід дозволив реалізувати базову модель розподілу ролей у мережі, де кожен учасник несе відповідальність за певну кількість вузлів. Це особливо

важливо в майбутній реалізації алгоритмів консенсусу, де вага голосу може визначатись кількістю підконтрольних вузлів.

Форма додавання нод до користувача включає:

- унікальний ідентифікатор користувача;
- роль у системі (як і в загальному списку);
- поле введення кількості вузлів;
- валідацію правильності введених даних (має бути додатне число).

Дані автоматично зберігаються у локальному сховищі (localStorage) та синхронізуються з іншими компонентами системи. Наприклад, при формуванні конфігурації для P2PUpload.jsx ці значення передаються у структуру конфігураційного JSON-об’єкта.

У майбутньому цей модуль може бути доповнений:

- окремим інтерфейсом для управління вузлами;
- прив’язкою вузлів до географічного розташування;
- класифікацією нод за типом (спостереження, підрахунок, архівування);
- автоматичним аудитом доступності нод у реальному часі (наприклад, через ping);
- графічною візуалізацією структури p2p-мережі.

Таким чином, реалізований підхід до формування переліку власників вузлів дозволяє закласти основу для побудови більш складної системи розподіленого управління голосуванням, де структура та статус кожної точки мережі будуть визначальними для безпеки та надійності процесу.

4.6. Формування загальних параметрів консенсусу

У децентралізованій системі голосування механізм консенсусу відіграє ключову роль у досягненні згоди між вузлами щодо результатів голосування. Для забезпечення гнучкості й прозорості цього процесу, у реалізованому інтерфейсі передбачено окремий модуль для налаштування параметрів консенсусу.

Цей функціонал реалізований у компоненті ConsensusParams.jsx, що дозволяє адміністратору встановлювати такі параметри:

- алгоритм консенсусу – обирається з-поміж доступних варіантів: Proof-of-Authority (PoA), Proof-of-Stake (PoS) або Proof-of-Work (PoW). Алгоритм визначає, яким чином учасники мережі досягають згоди;
- кворум – встановлюється як відсоткове значення (наприклад, 51%), що вказує, скільки учасників (вузлів) повинні підтримати рішення для його затвердження;
- час створення блоку – задається у секундах і визначає частоту генерації нових блоків у системі.

Всі ці параметри зберігаються в об’єкті consensusParams у локальному сховищі (localStorage). Після налаштування вони автоматично підставляються у нові голосування, які створюються через компонент VotingForm.jsx. Таким чином, адміністратор встановлює єдину політику консенсусу, яка діє для всіх подальших виборчих процесів. Кожне поле у формі супроводжується підказками, що пояснюють вплив відповідного параметра на безпеку та ефективність системи. Наприклад, нижчий кворум забезпечує швидше ухвалення рішень, однак знижує стійкість до атак. Високий кворум підвищує безпеку, але вимагає участі більшої кількості вузлів.

У майбутньому можливе доповнення цього модуля такими функціями:

- розширення типів алгоритмів (наприклад, Delegated Proof-of-Stake, BFT);
- призначення ролей нод (ініціатори, валідатори, архіватори);
- таймаути та механізми повторного голосування у разі неповного кворуму;
- збереження конфігурації у підписаному вигляді (цифровий підпис адміністратора);
- інтеграція з блокчейн-мережами або смарт-контрактами.

5 РОЗРОБЛЕННЯ ВЕБ-ЗАСТОСУНКУ

5.1. Загальний опис функціональних можливостей розробленої системи

У межах практичної реалізації дипломного проєкту було розроблено повноцінний веб-застосунок, який виконує функції адміністративного інтерфейсу для децентралізованої системи захищеного голосування. Його основне призначення – забезпечити адміністратора можливістю ефективно управляти процесами електронного голосування, створювати голосування з детальними параметрами, контролювати доступ учасників, задавати консенсусні правила та готувати дані до інтеграції в p2p-середовище. Розробка велась з використанням JavaScript-бібліотеки React, що дозволяє реалізовувати SPA (Single Page Application) з підтримкою динамічного рендерингу, реактивного оновлення інтерфейсу, ізоляції компонентів і високого рівня повторного використання коду. Компоненти були побудовані із врахуванням принципів гнучкості, модульності й масштабованості.

Система складається з шести функціональних модулів:

- Авторизація адміністратора – реалізована базова форма логіну, яка приймає логін і пароль, після чого дозволяє доступ до повного інтерфейсу керування. Для простої автентифікації застосовується внутрішній стан isLoggedIn, який активується при правильному введенні даних.
- Створення голосувань – адміністратор заповнює форму, яка включає:
    - назву голосування (наприклад, «Обрання голови»);
    - опис (детальна інформація про мету голосування);
    - час початку і завершення (у форматі datetime-local);
    - перелік варіантів вибору (мінімум два);
    - тип доступу – «публічне» (відкрите голосування) або «приватне» (доступ за списком). Кожне голосування, після збереження, зберігається у глобальному масиві votings, а також записується в localStorage для довготривалого збереження.

- Формування списку учасників – окремий розділ дозволяє додати будь-яку кількість учасників, кожен з яких має наступні параметри:
  - ПІБ або псевдонім;
  - унікальний ідентифікатор (email, ID або інше);
  - роль: виборець (той, хто голосує), спостерігач (має лише перегляд), оператор (має розширені права);
  - кількість нод (кількість точок, які він контролює в p2p-мережі). Усі учасники зберігаються у масиві participants, що автоматично оновлюється у localStorage.
- Параметри консенсусу – передбачено конфігурування глобальних параметрів для всієї мережі. Всі значення вводяться через окрему форму й зберігаються у глобальному об’єкті consensusParams. Серед параметрів:
  - відсоток кворуму (наприклад, 51% – рішення вважається прийнятим, якщо голоси «за» перевищують цю межу);
  - час створення блоку (в секундах);
  - обраний алгоритм консенсусу (PoA, PoW або PoS).
- Перегляд голосувань – всі створені голосування виводяться в таблиці, де вказуються всі технічні та організаційні параметри. Для кожного голосування відображаються: назва, опис, дати, тип, кількість варіантів, а також параметри консенсусу (алгоритм, блок-тайм, кворум).
- Збір усіх даних – передбачено окремий розділ, де всі дані – учасники, голосування, консенсус – формуються у форматований JSON-об’єкт, що може бути переданий на вузол або інше середовище обробки у рамках p2p.

5.2. Архітектура застосунку та логіка його роботи

Система побудована за принципом односторінкового застосунку, структуру проєкта зображена на рисунку 5.1. Це означає, що всі дії відбуваються в межах однієї HTML-сторінки, а всі розділи (сторінки) – це логічні компоненти, які динамічно підвантажуються й перемикаються залежно від значення стану view.

Центральною точкою застосунку є компонент App.jsx, у якому реалізована логіка навігації, зберігання даних, синхронізації з локальним сховищем та передача пропсів у компоненти.

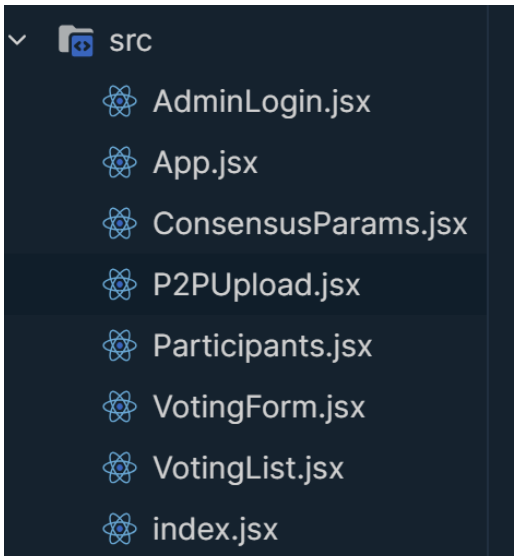

Рисунок 5.1 – Структура проєкту

Кожен функціональний блок реалізований у вигляді окремого компонента:

- AdminLogin.jsx – містить логіку авторизації. При правильному введенні логіну/паролю (наприклад, admin/1234), відбувається оновлення стану isLoggedIn, після чого рендериться основний інтерфейс. Вікно адиіна зображене на рисунку 5.2.

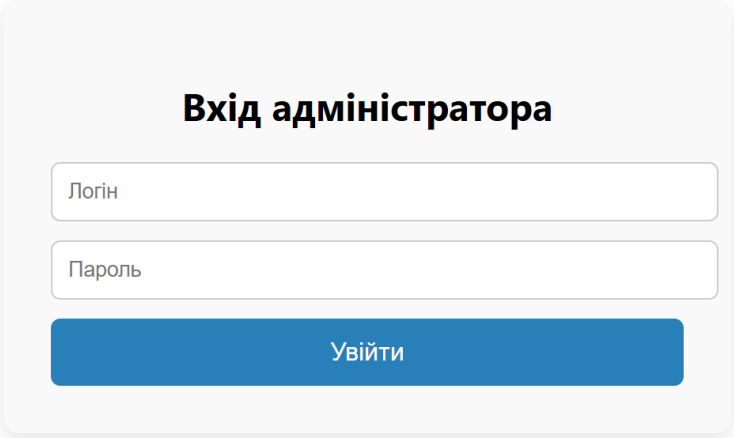

Рисунок 5.2 – Вхід адміністратора

- VotingForm.jsx – форма створення голосування зображена на рисунку 5.3 та рисунку 5.4. Всі поля контролюються локальним станом. Коли форма

заповнена, об’єкт голосування відправляється у батьківський компонент через функцію onCreate, де й додається до глобального масиву votings. Одночасно в голосування вставляється копія активних параметрів консенсусу. Після створення голосування висвічується вікно о створенні голосування, зображено на рисунку 5.5.

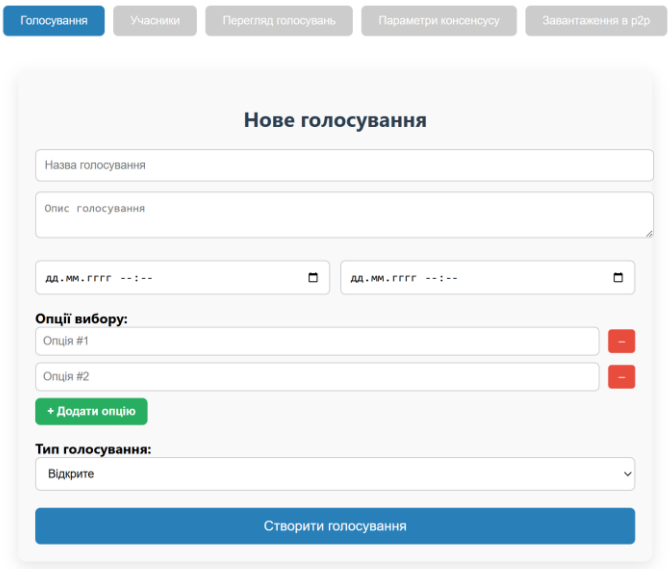

Рисунок 5.3 – Створенная нового голосування

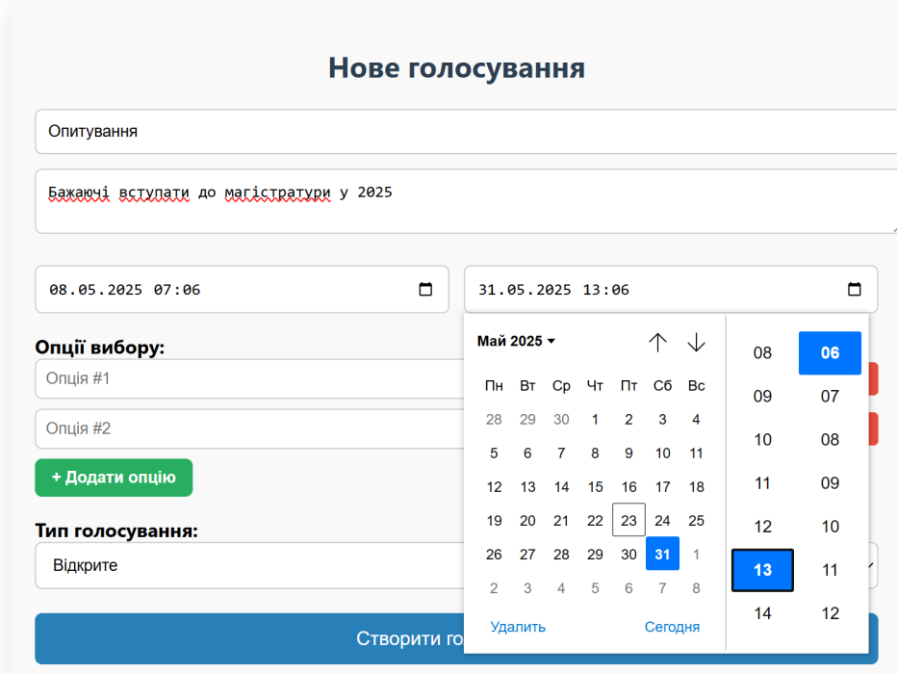

Рисунок 5.4 – Обмеження голосування по часу

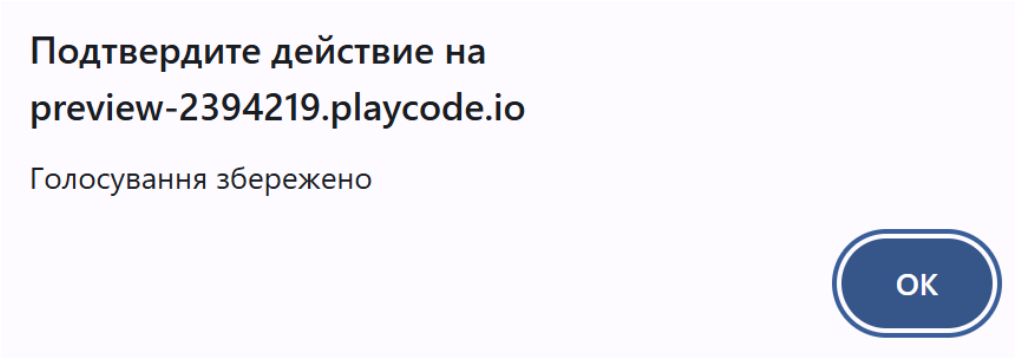

Рисунок 5.5 – Підтвердження створення голосування

- Participants.jsx – дозволяє вводити та редагувати список учасників. При натисканні на кнопку «Додати» учасник з’являється в таблиці. Є можливість видалити будь-якого з них. Усі зміни автоматично зберігаються у localStorage. Функціонал вкладки учасники голосувань зображені на рисунку 5.6, рисунку 5.7 та рисунку 5.8.

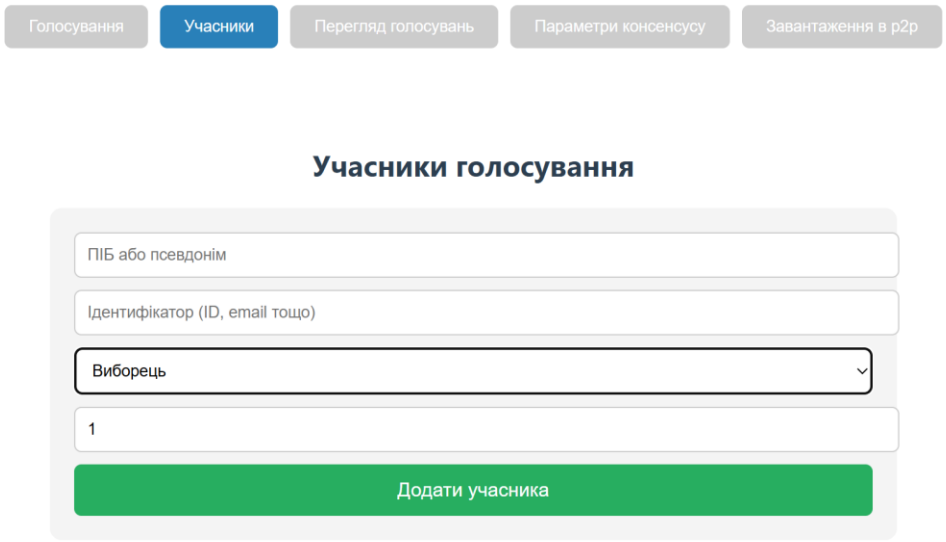

Рисунок 5.6 – Додавання учасників

Рисунок 5.7 – Типи учасників

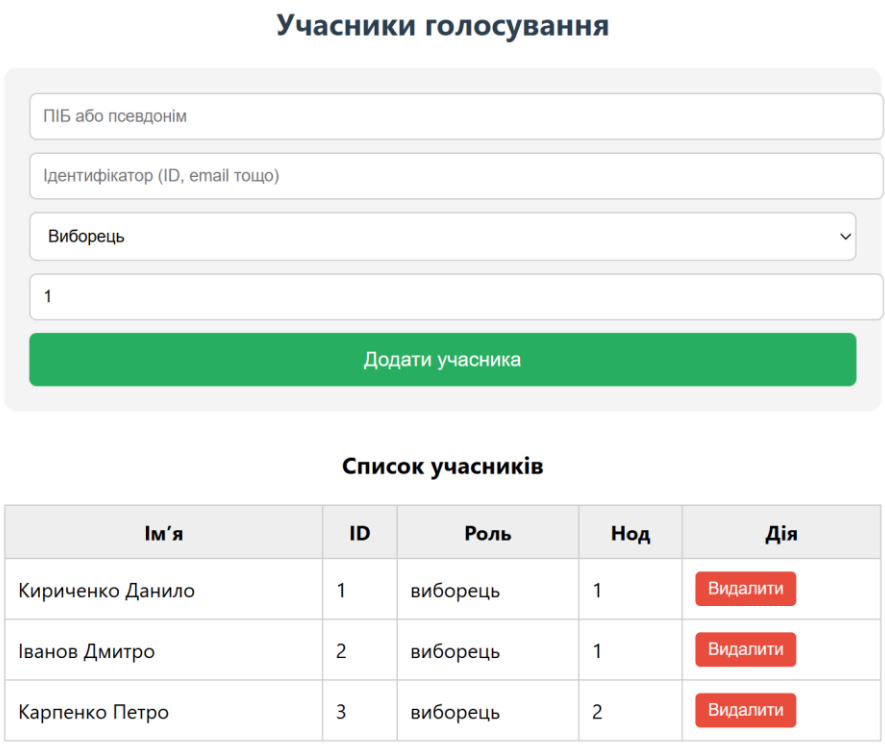

Рисунок 5.8 – Перелік виборців

- VotingList.jsx – табличне відображення всіх голосувань, рисунок 5.9, створених в системі. Дані рендеряться у зручному для перегляду форматі, що включає і технічні (алгоритм консенсусу, блок-тайм, кворум), і логічні параметри (назва, тип, дата). Для кожного рядка реалізована кнопка видалення.

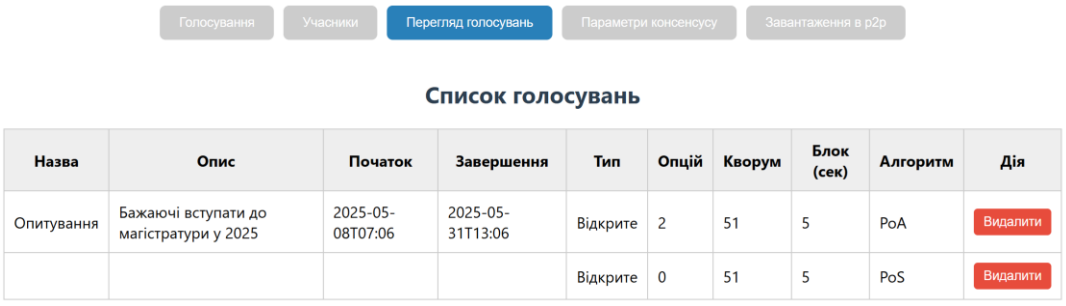

Рисунок 5.9 – Створенні голосування

- ConsensusParams.jsx – тут адміністратор може задати або змінити глобальні параметри. При майбутньому доопрацюванні компонент синхронізується з локальним сховищем і впливає на всі подальші голосування. Сторінка з параметрами зображена на рисунку 5.10.

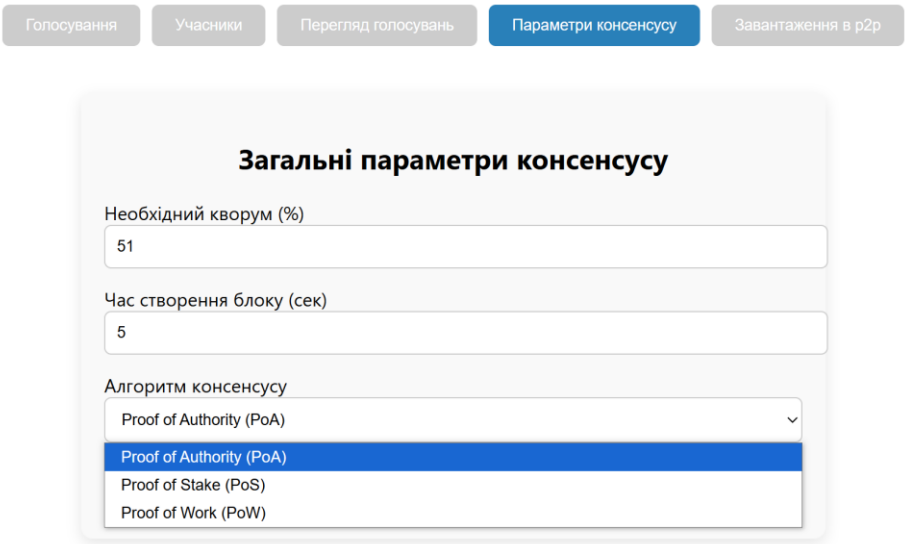

Рисунок 5.10 – Параметри консенсусу

- P2PUpload.jsx – формується об’єкт, рисунок 5.11, який включає масив votings, масив participants і об’єкт consensusParams. Вивід реалізовано через тег <pre>, який дозволяє переглянути всю структуру у зручному форматі.

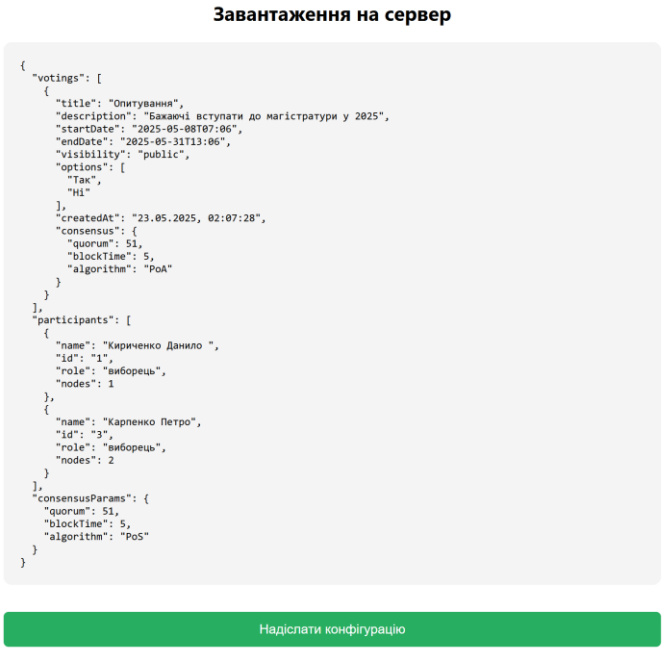

Рисунок 5.11 – Структура даних для завантаження на сервер

Передача даних між компонентами реалізована через пропси. Зберігання відбувається через useState та useEffect, які стежать за змінами й одразу оновлюють localStorage. Це дозволяє системі працювати навіть при втраті підключення або перезавантаженні браузера.

5.3. Механізми збереження даних, їх обробка та підготовка до використання

Весь обмін даними в межах застосунку реалізовано без використання внутрішньої логіки застосунку. Усі об’єкти, створені користувачем, зберігаються у локальному сховищі браузера (localStorage). Це дозволяє:

- зберігати дані довгостроково;
- зчитувати їх при повторному вході в застосунок;
- уникати використання бази даних або серверного API;
- працювати в автономному режимі (offline-first).

Для кожної категорії даних передбачено свій ключ у localStorage:

- participants – масив об’єктів учасників;
- votings – масив голосувань;
- consensusParams – глобальні параметри.

Усі зміни здійснюються через useEffect, який слідкує за оновленням відповідного стану. Зміна хоча б одного учасника або параметру одразу оновлює сховище. При створенні нового голосування дані консенсусу копіюються із загального об’єкта, що забезпечує послідовність. Це гарантує, що кожне голосування буде мати повний набір параметрів, незалежно від того, коли й ким воно було створене.

Формат вихідного об’єкта конфігурації:

```
{
  "timestamp": "2025-05-19T10:10:10.000Z",
  "votings": [ {...} ],
  "participants": [ {...} ],
  "consensusParams": { ... }
}
```

Цей формат відповідає стандарту JSON і може бути використаний для подальшої обробки на стороні сервера, збереження на вузлах p2p-мережі або передачі іншим адміністраторам системи. Такий підхід дозволяє інтегрувати клієнтське рішення в розподілену інфраструктуру без потреби у централізованому сервері.

ВИСНОВКИ

У результаті проведеного теоретичного дослідження було проаналізовано сучасні системи електронного голосування, виявлено їхні ключові недоліки та визначено перспективні напрями розвитку. Особливу увагу приділено децентралізованим підходам, які дозволяють підвищити рівень прозорості, стійкості до атак і довіри до результатів голосування. Було сформовано модель потенційних загроз і профіль зловмисника, що дало змогу обґрунтувати необхідність побудови гнучкої та безпечної інфраструктури. На основі цього сформульовано вимоги до функціональності робочого місця адміністратора, яке відіграє ключову роль у керуванні процесом голосування, учасниками та параметрами p2p-мережі.

Також досліджено архітектурні принципи децентралізованих систем, методи зберігання даних, особливості обміну інформацією та підходи до проєктування зручного адміністративного інтерфейсу. У третьому розділі окреслено засади побудови безпечної системи голосування, проаналізовано механізми криптографічного захисту, автентифікації та методи протидії мережевим загрозам у p2p-середовищі.

У результаті виконання практичної частини дипломної роботи було розроблено, реалізовано та протестовано прототип адміністративного інтерфейсу для децентралізованої системи захищеного голосування. Основною метою проєкту було створення зручного та ефективного робочого місця адміністратора, який відповідає за повний цикл підготовки голосування: від автентифікації користувача до формування переліку учасників, параметрів голосування, конфігурацій вузлів і визначення алгоритмів консенсусу.

Система побудована на сучасній технології React і реалізована у вигляді односторінкового застосунку. Кожен компонент функціонально ізольований, що забезпечує масштабованість, зрозумілу архітектуру та можливість подальшого розширення. Завдяки використанню локального сховища (localStorage) вдалося

реалізувати автономну модель зберігання даних без необхідності серверної частини.

В межах практичного розділу були виконані такі ключові етапи:

- реалізовано базову автентифікацію адміністратора;
- створено гнучкий механізм формування та перегляду голосувань;
- забезпечено додавання, перегляд та видалення учасників системи;
- реалізовано структуру обліку нод для подальшого врахування у p2p-консенсусі;
- розроблено інтерфейс для конфігурування основних параметрів механізму узгодження (алгоритм, кворум, час блоку);
- реалізовано фінальну сторінку з відображенням готової конфігурації системи у форматі JSON.

Отримані результати підтверджують можливість побудови ефективного та логічного інтерфейсу адміністратора для децентралізованих голосувальних систем. Дана реалізація може бути основою для подальшого розгортання повноцінної системи голосування з підключенням реальних вузлів, авторизаційних сервісів, захищеного серверного API, цифрового підпису та блокчейн-технологій. Таким чином, поставлені цілі дипломної роботи повністю досягнуті, а розроблений інструмент відповідає сучасним вимогам безпечного й гнучкого адміністрування процесів електронного голосування.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System [Електронний ресурс]. –2008. – Режим доступу: https://bitcoin.org/bitcoin.pdf (дата звернення: 10.04.2025).

2. Barnes A. Blockchain-Based Voting: An Exploration of Security and Trust // Journal of E-Government Studies. – 2021. (дата звернення: 10.04.2025).

3. Закон України «Про електронні довірчі послуги» № 2155-VIII від 05.10.2017 [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2155-19#Text (дата звернення: 12.04.2025).

4. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» № 80/94-ВР від 05.07.1994 [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/80/94-вр#Text (дата звернення: 15.04.2025).

5. Міністерство цифрової трансформації України – Голосування у застосунку «Дія» на Євробаченні-2023. [Електронний ресурс] – Режим доступу: https://diia.gov.ua/news (дата звернення: 15.04.2025).

6. e-Estonia – How did Estonia carry out the world's first mostly online national elections? – e-estonia.com, 07.03.2023. [Електронний ресурс] – Режим доступу: https://e-estonia.com/how-did-estonia-carry-out-the-worlds-first-mostly-online-national-elections/ (дата звернення: 15.04.2025).

7. Springall D. et al. – Security Analysis of the Estonian Internet Voting System. ACM CCS 2014. [Електронний ресурс] – Режим доступу: https://jhalderm.com/pub/papers/ivoting-ccs14.pdf (дата звернення: 15.04.2025).

8. Goodman N. – The Security Challenges of Internet Voting. Carnegie Endowment for International Peace. [Електронний ресурс] – Режим доступу: https://carnegieendowment.org (дата звернення: 15.04.2025).

9. National Institute of Standards and Technology (NIST) – Cybersecurity Framework. [Електронний ресурс] – Режим доступу: https://www.nist.gov/cyberframework (дата звернення: 18.04.2025).

10. Rivest R., Wack J. – On the Notion of 'Software Independence' in Voting Systems. U.S. Election Assistance Commission. [Електронний ресурс] – Режим доступу: https://www.nist.gov/publications/software-independence-voting-systems (дата звернення: 20.04.2025).

11. Krebs B. – Inside a Massive Cyberattack. KrebsOnSecurity.com, 2021. [Електронний ресурс] – Режим доступу: https://krebsonsecurity.com (дата звернення: 20.04.2025).

12. Bishop M. – Introduction to Computer Security. Addison-Wesley, 2005. (дата звернення: 22.04.2025).

13. Rid T. – Active Measures: The Secret History of Disinformation and Political Warfare. Farrar, Straus and Giroux, 2020. (дата звернення: 25.04.2025).

14. Rivest R., Wack J. – On the Notion of 'Software Independence' in Voting Systems. U.S. Election Assistance Commission. [Електронний ресурс] – Режим доступу: https://www.nist.gov/publications/software-independence-voting-systems (дата звернення: 27.04.2025).

15. Nakamoto S. – Bitcoin: A Peer-to-Peer Electronic Cash System. bitcoin.org, 2008. [Електронний ресурс] – Режим доступу: https://bitcoin.org/bitcoin.pdf (дата звернення: 27.04.2025).

16. Blockchain Technology: Insights from Scientific Literature and Lessons for the FinTech Industry. TNO, 2017. (дата звернення: 27.04.2025).

17. Міністерство цифрової трансформації України – BankID та інтеграція з Дія ID. [Електронний ресурс] – Режим доступу: https://diia.gov.ua/services (дата звернення: 27.04.2025).

18. Koens T., Poll E., Jansen W. – Blockchain Technology: Insights from Scientific Literature and Lessons for the FinTech Industry. TNO, 2017. (дата звернення: 27.04.2025).

19. Закон України «Про основні засади забезпечення кібербезпеки України» № 2163-VIII від 05.10.2017 [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2163-19#Text (дата звернення: 28.04.2025).

20. НД ТЗІ 3.7-003-2005. Порядок проведення робіт із створення комплексної системи захисту інформації в інформаційно-телекомунікаційних системах [Електронний ресурс]. – Режим доступу: https://tzi.ua/nd-tzi-3-7-003-2005 (дата звернення: 28.04.2025).

21. CEUR Workshop Proceedings, Vol-3925. Savorona N., Kovalenko Y., Pavlenko M. Децентралізована система опитувань на основі блокчейну [Електронний ресурс] // CH&CMiGIN 2024. – 2024. – Режим доступу: http://ceur-ws.org/Vol-3925/paper20.pdf (дата звернення: 28.04.2025).

22. Daraghmi E., Karray M., Alkhodre A., et al. Decentralizing Democracy: Secure and Transparent E-Voting Systems with Blockchain Technology [Електронний ресурс] // Future Internet. –2023. –Vol. 16, No. 11. –Article 388. – Режим доступу: https://www.mdpi.com/1999-5903/16/11/388 (дата звернення: 01.05.2025).

23. Singh J., Pooja, Sharma N. Blockchain-based decentralized voting system security perspective: safe and secure for digital voting [Електронний ресурс] // arXiv preprint. – 2023. – arXiv:2303.06306. – Режим доступу: https://arxiv.org/abs/2303.06306 (дата звернення: 01.05.2025).

24. Ethereum Improvement Proposal (EIP-225). Clique Proof-of-Authority Consensus Protocol [Електронний ресурс]. –Режим доступу: https://eips.ethereum.org/EIPS/eip-225 (дата звернення: 02.05.2025).

25. ISO/IEC 30170:2012. Information technology –Blockchain and distributed ledger technologies –Reference architecture [Електронний ресурс]. – Режим доступу: https://www.iso.org/standard/78162.html (дата звернення: 05.05.2025).

26. NIST SP 800-53 Rev.5 – Security and Privacy Controls for Information Systems and Organizations. National Institute of Standards and Technology, 2020. (дата звернення: 05.05.2025).

27. Buterin V. – A Next-Generation Smart Contract and Decentralized Application Platform. Ethereum Whitepaper, 2014. (дата звернення: 05.05.2025).

28. ISO/IEC 27001:2022 – Information technology – Security techniques – Information security management systems –Requirements. (дата звернення: 10.05.2025).

29. Nielsen J. – Usability Engineering. Academic Press, 1994. (дата звернення: 11.05.2025).

30. Kindervag J. – Build Security Into Your Network’s DNA: The Zero Trust Network Architecture. Forrester Research, 2010. (дата звернення: 12.05.2025).

31. O'Reilly – Designing Web APIs: Building APIs That Developers Love. 2020. (дата звернення: 12.05.2025).

32. Nakamoto S. – Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. [Електронний ресурс] –Режим доступу: https://bitcoin.org (дата звернення: 15.05.2025).

33. Benet J. – IPFS - Content Addressed, Versioned, P2P File System. 2014. (дата звернення: 15.05.2025).

34. Buterin V. – Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. Ethereum Foundation, 2014. (дата звернення: 17.05.2025).

35. EU Blockchain Observatory – Blockchain and the Future of Digital Voting. 2020. (дата звернення: 18.05.2025).

36. Swan M. – Blockchain: Blueprint for a New Economy. O'Reilly Media, 2015. (дата звернення: 18.05.2025).

37. NIST – Framework for Cyber-Physical Systems. Release 1.0. National Institute of Standards and Technology, 2016. (дата звернення: 18.05.2025).

38. OWASP Foundation – OWASP Application Security Verification Standard (ASVS) 4.0.3. 2021. (дата звернення: 20.05.2025).

39. Nielsen J. – Designing Web Usability: The Practice of Simplicity. New Riders Publishing, 2000. (дата звернення: 20.05.2025).

40. Androutsellis-Theotokis S., Spinellis D. – A survey of peer-to-peer content distribution technologies. ACM Computing Surveys, 2004. (дата звернення: 21.05.2025).

41. Zamani M., Movahedi M., Raykova M. – RapidChain: Scaling Blockchain via Full Sharding. ACM CCS, 2018. (дата звернення: 22.05.2025).

42. Benet J. – IPFS - Content Addressed, Versioned, P2P File System. 2014. (дата звернення: 23.05.2025).

43. Cachin C., Vukolić M. – Blockchain Consensus Protocols in the Wild. arXiv, 2017. (дата звернення: 24.05.2025).

44. International Foundation for Electoral Systems (IFES). 2022. Trust in Elections: Global Trends and Challenges. – Режим доступу: https://www.ifes.org/news/trust-elections (дата звернення: 24.05.2025).

45. Chaum D. – Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM, 1981. (дата звернення: 24.05.2025).

46. Gentry C. – Fully Homomorphic Encryption Using Ideal Lattices. STOC, 2009. (дата звернення: 24.05.2025).

47. OWASP Authentication Cheat Sheet. OWASP Foundation. – Режим доступу: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (дата звернення: 24.05.2025).

48. Cachin C., Vukolić M. – Blockchain Consensus Protocols in the Wild. (дата звернення: 24.05.2025).

ДОДАТОК А

ВИХІДНИЙ КОД ЗАСТОСУНКУ

APP.JSX

```
import React, { useState, useEffect } from 'react';
import VotingForm from './VotingForm';
import Participants from './Participants';
import VotingList from './VotingList';
import ConsensusParams from './ConsensusParams';
import AdminLogin from './AdminLogin';
import P2PUpload from './P2PUpload';
export default function App() {
  const [view, setView] = useState('voting');
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  const [participants, setParticipants] = useState(() => {
    const saved = localStorage.getItem('participants');
    return saved ? JSON.parse(saved) : [];
  });
  const [votings, setVotings] = useState(() => {
    const saved = localStorage.getItem('votings');
    return saved ? JSON.parse(saved) : [];
  });
  const [consensusParams, setConsensusParams] = useState(() => {
    const saved = localStorage.getItem('consensusParams');
    return saved ? JSON.parse(saved) : { quorum: 51, blockTime: 5, algorithm: 'PoA' };
  });
  useEffect(() => {
    localStorage.setItem('participants', JSON.stringify(participants));
  }, [participants]);
```

```
  useEffect(() => {
    localStorage.setItem('votings', JSON.stringify(votings));
  }, [votings]);
  useEffect(() => {
    localStorage.setItem('consensusParams', JSON.stringify(consensusParams));
  }, [consensusParams]);
  const handleAddParticipant = (participant) => {
    setParticipants([...participants, participant]);
  };
  const handleDeleteParticipant = (index) => {
    setParticipants(participants.filter((_, i) => i !== index));
  };
  const handleDeleteVoting = (index) => {
    setVotings(votings.filter((_, i) => i !== index));
  };
  const handleCreateVoting = (votingData) => {
    const votingWithConsensus = { ...votingData, consensus: consensusParams };
    setVotings([...votings, votingWithConsensus]);
  };
  if (!isLoggedIn) {
    return <AdminLogin onLogin={() => setIsLoggedIn(true)} />;
  }
  return (
    <div style={{ fontFamily: 'Segoe UI, sans-serif' }}>
      <nav style={{ display: 'flex', justifyContent: 'center', flexWrap: 'wrap', gap: '10px',
margin: '20px 0' }}>
        <button
          onClick={() => setView('voting')}
          style={{ padding: '10px 20px', backgroundColor: view === 'voting' ? '#2980b9' :
'#ccc', color: 'white', border: 'none', borderRadius: '6px' }}
        >
```

```
        Голосування
      </button>
      <button
        onClick={() => setView('participants')}
        style={{ padding: '10px 20px', backgroundColor: view === 'participants' ? '#2980b9' : '#ccc', color: 'white', border: 'none', borderRadius: '6px' }}
      >
        Учасники
      </button>
      <button
        onClick={() => setView('list')}
        style={{ padding: '10px 20px', backgroundColor: view === 'list' ? '#2980b9' : '#ccc', color: 'white', border: 'none', borderRadius: '6px' }}
      >
        Перегляд голосувань
      </button>
      <button
        onClick={() => setView('consensus')}
        style={{ padding: '10px 20px', backgroundColor: view === 'consensus' ? '#2980b9' : '#ccc', color: 'white', border: 'none', borderRadius: '6px' }}
      >
        Параметри консенсусу
      </button>
      <button
        onClick={() => setView('upload')}
        style={{ padding: '10px 20px', backgroundColor: view === 'upload' ? '#2980b9' : '#ccc', color: 'white', border: 'none', borderRadius: '6px' }}
      >
        Завантаження в p2p
      </button>
    </nav>
    {view === 'voting' && <VotingForm onCreate={handleCreateVoting} />}
```

```
      {view === 'participants' && (
        <Participants
          participants={participants}
          onAdd={handleAddParticipant}
          onDelete={handleDeleteParticipant}
        />
      )}
      {view === 'list' && <VotingList votings={votings}
onDelete={handleDeleteVoting} />}
      {view === 'consensus' && (
        <ConsensusParams params={consensusParams}
setParams={setConsensusParams} />
      )}
      {view === 'upload' && (
        <P2PUpload
          votings={votings}
          participants={participants}
          consensusParams={consensusParams}
        />
      )}
    </div>
  );
}
```

index.jsx

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
```

```
    <App />
  </React.StrictMode>
);
```

VotingForm.jsx

```
import React, { useState } from 'react';
export default function VotingForm({ onCreate }) {
  const [formData, setFormData] = useState({
    title: '',
    description: '',
    startDate: '',
    endDate: '',
    visibility: 'public',
  });
  const [options, setOptions] = useState(['', '']);
  const updateField = (field, value) => {
    setFormData({ ...formData, [field]: value });
  };
  const updateOption = (index, value) => {
    const updated = [...options];
    updated[index] = value;
    setOptions(updated);
  };
  const addOption = () => setOptions([...options, '']);
  const removeOption = (index) => setOptions(options.filter((_, i) => i !== index));
  const handleSubmit = () => {
    const payload = {
      ...formData,
      options: options.filter(opt => opt.trim() !== ''),
      createdAt: new Date().toLocaleString()
```

```
    };
    onCreate(payload);
    setFormData({ title: '', description: '', startDate: '', endDate: '', visibility: 'public' });
    setOptions(['', '']);
    alert('Голосування збережено');
  };
  return (
    <div style={{ maxWidth: '700px', margin: '40px auto', padding: '20px', background: '#f9f9f9', borderRadius: '10px', boxShadow: '0 6px 18px rgba(0, 0, 0, 0.1)', fontFamily: 'Segoe UI, sans-serif' }}>
      <h2 style={{ textAlign: 'center', color: '#2c3e50', marginBottom: '20px' }}>Нове голосування</h2>
      <input
        style={{ width: '100%', padding: '10px', marginBottom: '12px', border: '1px solid #ccc', borderRadius: '6px' }}
        placeholder="Назва голосування"
        value={formData.title}
        onChange={(e) => updateField('title', e.target.value)}
      />
      <textarea
        style={{ width: '100%', padding: '10px', marginBottom: '12px', border: '1px solid #ccc', borderRadius: '6px' }}
        placeholder="Опис голосування"
        value={formData.description}
        onChange={(e) => updateField('description', e.target.value)}
      />
      <div style={{ display: 'flex', gap: '12px', margin: '10px 0' }}>
        <input
          type="datetime-local"
          style={{ flex: 1, padding: '10px', borderRadius: '6px', border: '1px solid #ccc' }}
          value={formData.startDate}
          onChange={(e) => updateField('startDate', e.target.value)}
```

```
        />
        <input
          type="datetime-local"
          style={{ flex: 1, padding: '10px', borderRadius: '6px', border: '1px solid #ccc' }}
          value={formData.endDate}
          onChange={(e) => updateField('endDate', e.target.value)}
        />
      </div>
      <div style={{ marginTop: '16px' }}>
        <label style={{ fontWeight: 'bold' }}>Опції вибору:</label>
        {options.map((opt, idx) => (
          <div key={idx} style={{ display: 'flex', alignItems: 'center', gap: '10px',
marginBottom: '8px' }}>
            <input
              style={{ flex: 1, padding: '8px', borderRadius: '6px', border: '1px solid #ccc' }}
              placeholder={`Опція #${idx + 1}`}
              value={opt}
              onChange={(e) => updateOption(idx, e.target.value)}
            />
            <button onClick={() => removeOption(idx)} style={{ padding: '6px 12px',
backgroundColor: '#e74c3c', color: 'white', border: 'none', borderRadius: '4px' }}>–
</button>
          </div>
        ))}
        <button onClick={addOption} style={{ padding: '8px 14px', backgroundColor:
'#27ae60', color: 'white', border: 'none', borderRadius: '6px', fontWeight: 'bold' }}>+
Додати опцію</button>
      </div>
      <div style={{ marginTop: '16px' }}>
        <label style={{ fontWeight: 'bold' }}>Тип голосування: </label>
        <select
```

```
        style={{ width: '100%', padding: '10px', borderRadius: '6px', border: '1px solid
#ccc' }}
        value={formData.visibility}
        onChange={(e) => updateField('visibility', e.target.value)}
       >
        <option value="public">Відкрите</option>
        <option value="private">Приватне</option>
       </select>
      </div>
      <button
       onClick={handleSubmit}
       style={{ width: '100%', marginTop: '20px', padding: '12px', backgroundColor:
'#2980b9', color: 'white', border: 'none', borderRadius: '6px', fontSize: '16px' }}
      >
       Створити голосування
      </button>
    </div>
  );
}
```

Participants.jsx

```
import React, { useState } from 'react';
export default function Participants({ participants, onAdd, onDelete }) {
  const [newParticipant, setNewParticipant] = useState({ name: '', id: '', role: 'виборець',
nodes: 1 });
  const handleChange = (field, value) => {
    setNewParticipant({ ...newParticipant, [field]: value });
  };
  const handleAdd = () => {
    if (!newParticipant.name.trim() || !newParticipant.id.trim()) {
      alert('Усі поля обов’язкові');
```

```
    return;
  }
  onAdd(newParticipant);
  setNewParticipant({ name: '', id: '', role: 'виборець', nodes: 1 });
 };
 return (
  <div style={{ maxWidth: '700px', margin: '40px auto', padding: '20px', fontFamily: 'Segoe UI, sans-serif' }}>
   <h2 style={{ textAlign: 'center', color: '#2c3e50', marginBottom: '20px' }}>Учасники голосування</h2>
   <div style={{ background: '#f4f4f4', padding: '20px', borderRadius: '10px' }}>
    <input
     style={{ width: '100%', padding: '10px', marginBottom: '10px', border: '1px solid #ccc', borderRadius: '6px' }}
     placeholder="ПІБ або псевдонім"
     value={newParticipant.name}
     onChange={(e) => handleChange('name', e.target.value)}
    />
    <input
     style={{ width: '100%', padding: '10px', marginBottom: '10px', border: '1px solid #ccc', borderRadius: '6px' }}
     placeholder="Ідентифікатор (ID, email тощо)"
     value={newParticipant.id}
     onChange={(e) => handleChange('id', e.target.value)}
    />
    <select
     style={{ width: '100%', padding: '10px', marginBottom: '10px', borderRadius: '6px', border: '1px solid #ccc' }}
     value={newParticipant.role}
     onChange={(e) => handleChange('role', e.target.value)}
    >
     <option value="виборець">Виборець</option>
```

```
          <option value="спостерігач">Спостерігач</option>
          <option value="оператор">Оператор</option>
        </select>
        <input
          type="number"
          min="1"
          style={{ width: '100%', padding: '10px', marginBottom: '10px', border: '1px solid #ccc', borderRadius: '6px' }}
          placeholder="Кількість нод"
          value={newParticipant.nodes}
          onChange={(e) => handleChange('nodes', parseInt(e.target.value) || 1)}
        />
        <button
          onClick={handleAdd}
          style={{ width: '100%', padding: '12px', backgroundColor: '#27ae60', color: 'white', border: 'none', borderRadius: '6px', fontSize: '16px' }}
        >
          Додати учасника
        </button>
      </div>
      {participants.length > 0 && (
        <div style={{ marginTop: '30px' }}>
          <h3 style={{ textAlign: 'center' }}>Список учасників</h3>
          <table style={{ width: '100%', borderCollapse: 'collapse', marginTop: '20px' }}>
            <thead>
              <tr style={{ backgroundColor: '#eee' }}>
                <th style={{ padding: '10px', border: '1px solid #ccc' }}>Ім’я</th>
                <th style={{ padding: '10px', border: '1px solid #ccc' }}>ID</th>
                <th style={{ padding: '10px', border: '1px solid #ccc' }}>Роль</th>
                <th style={{ padding: '10px', border: '1px solid #ccc' }}>Нод</th>
                <th style={{ padding: '10px', border: '1px solid #ccc' }}>Дія</th>
```

```
            </tr>
          </thead>
          <tbody>
            {participants.map((p, i) => (
              <tr key={i}>
                <td style={{ padding: '10px', border: '1px solid #ccc' }}>{p.name}</td>
                <td style={{ padding: '10px', border: '1px solid #ccc' }}>{p.id}</td>
                <td style={{ padding: '10px', border: '1px solid #ccc' }}>{p.role}</td>
                <td style={{ padding: '10px', border: '1px solid #ccc' }}>{p.nodes}</td>
                <td style={{ padding: '10px', border: '1px solid #ccc' }}>
                  <button
                    onClick={() => onDelete(i)}
                    style={{ padding: '6px 10px', backgroundColor: '#e74c3c', color: 'white',
border: 'none', borderRadius: '4px' }}
                  >
                    Видалити
                  </button>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    )}
  </div>
 );
}
```

VotingList.jsx

```
import React from 'react';
```

```
export default function VotingList({ votings, onDelete }) {
  return (
    <div style={{ maxWidth: '1100px', margin: '40px auto', fontFamily: 'Segoe UI, sans-serif' }}>
      <h2 style={{ textAlign: 'center', color: '#2c3e50', marginBottom: '20px' }}>Список голосувань</h2>
      {votings.length === 0 ? (
        <p style={{ textAlign: 'center', color: '#777' }}>Немає створених голосувань.</p>
      ) : (
        <table style={{ width: '100%', borderCollapse: 'collapse' }}>
          <thead>
            <tr style={{ backgroundColor: '#eee' }}>
              <th style={{ padding: '10px', border: '1px solid #ccc' }}>Назва</th>
              <th style={{ padding: '10px', border: '1px solid #ccc' }}>Опис</th>
              <th style={{ padding: '10px', border: '1px solid #ccc' }}>Початок</th>
              <th style={{ padding: '10px', border: '1px solid #ccc' }}>Завершення</th>
              <th style={{ padding: '10px', border: '1px solid #ccc' }}>Тип</th>
              <th style={{ padding: '10px', border: '1px solid #ccc' }}>Опцій</th>
              <th style={{ padding: '10px', border: '1px solid #ccc' }}>Кворум</th>
              <th style={{ padding: '10px', border: '1px solid #ccc' }}>Блок (сек)</th>
              <th style={{ padding: '10px', border: '1px solid #ccc' }}>Алгоритм</th>
              <th style={{ padding: '10px', border: '1px solid #ccc' }}>Дія</th>
            </tr>
          </thead>
          <tbody>
            {votings.map((voting, index) => (
              <tr key={index}>
                <td style={{ padding: '10px', border: '1px solid #ccc' }}>{voting.title}</td>
                <td style={{ padding: '10px', border: '1px solid #ccc' }}>{voting.description}</td>
```

```
            <td style={{ padding: '10px', border: '1px solid #ccc'
}}>{voting.startDate}</td>
            <td style={{ padding: '10px', border: '1px solid #ccc'
}}>{voting.endDate}</td>
            <td style={{ padding: '10px', border: '1px solid #ccc' }}>{voting.visibility
=== 'public' ? 'Відкрите' : 'Приватне'}</td>
            <td style={{ padding: '10px', border: '1px solid #ccc'
}}>{voting.options.length}</td>
            <td style={{ padding: '10px', border: '1px solid #ccc'
}}>{voting.consensus?.quorum ?? '-'}</td>
            <td style={{ padding: '10px', border: '1px solid #ccc'
}}>{voting.consensus?.blockTime ?? '-'}</td>
            <td style={{ padding: '10px', border: '1px solid #ccc'
}}>{voting.consensus?.algorithm ?? '-'}</td>
            <td style={{ padding: '10px', border: '1px solid #ccc' }}>
              <button
                onClick={() => onDelete(index)}
                style={{ padding: '6px 10px', backgroundColor: '#e74c3c', color: 'white',
border: 'none', borderRadius: '4px' }}
              >
                Видалити
              </button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  )}
 </div>
 );
}
```

ConsensusParams.jsx

```
import React, { useState } from 'react';
export default function ConsensusParams({ params, setParams }) {
  const [localParams, setLocalParams] = useState(params || {
    quorum: 51,
    blockTime: 5,
    algorithm: 'PoA'
  });
  const handleChange = (field, value) => {
    setLocalParams({ ...localParams, [field]: value });
  };
  const handleSave = () => {
    setParams(localParams);
    alert('Параметри консенсусу збережено.');
  };
  return (
    <div style={{ maxWidth: '600px', margin: '40px auto', padding: '20px', background:
'#f9f9f9', borderRadius: '10px', boxShadow: '0 4px 12px rgba(0,0,0,0.1)', fontFamily:
'Segoe UI, sans-serif' }}>
      <h2 style={{ textAlign: 'center', marginBottom: '20px' }}>Загальні параметри
консенсусу</h2>
      <label>Необхідний кворум (%)</label>
      <input
        type="number"
        min="1"
        max="100"
        value={localParams.quorum}
        onChange={(e) => handleChange('quorum', parseInt(e.target.value))}
        style={{ width: '100%', padding: '10px', marginBottom: '15px', borderRadius:
'6px', border: '1px solid #ccc' }}
      />
      <label>Час створення блоку (сек)</label>
      <input
```

```
        type="number"
        min="1"
        value={localParams.blockTime}
        onChange={(e) => handleChange('blockTime', parseInt(e.target.value))}
        style={{ width: '100%', padding: '10px', marginBottom: '15px', borderRadius:
'6px', border: '1px solid #ccc' }}
      />
      <label>Алгоритм консенсусу</label>
      <select
        value={localParams.algorithm}
        onChange={(e) => handleChange('algorithm', e.target.value)}
        style={{ width: '100%', padding: '10px', marginBottom: '20px', borderRadius:
'6px', border: '1px solid #ccc' }}
      >
        <option value="PoA">Proof of Authority (PoA)</option>
        <option value="PoS">Proof of Stake (PoS)</option>
        <option value="PoW">Proof of Work (PoW)</option>
      </select>
      <button
        onClick={handleSave}
        style={{ width: '100%', padding: '12px', backgroundColor: '#2980b9', color:
'white', border: 'none', borderRadius: '6px', fontSize: '16px' }}
      >
        Зберегти параметри
      </button>
    </div>
  );
}
```

AdminLogin.jsx

```
import React, { useState } from 'react';
```

```
export default function AdminLogin({ onLogin }) {
  const [admin, setAdmin] = useState({ username: '', password: '' });
  const [error, setError] = useState('');
  const handleChange = (field, value) => {
    setAdmin({ ...admin, [field]: value });
  };
  const handleSubmit = () => {
    // Простий приклад авторизації: логін admin, пароль 1234
    if (admin.username === 'admin' && admin.password === '1234') {
      onLogin();
    } else {
      setError('Невірний логін або пароль');
    }
  };
  return (
    <div style={{ maxWidth: '400px', margin: '100px auto', padding: '30px', background: '#f9f9f9', borderRadius: '10px', boxShadow: '0 4px 12px rgba(0,0,0,0.1)', fontFamily: 'Segoe UI, sans-serif' }}>
      <h2 style={{ textAlign: 'center', marginBottom: '20px' }}>Вхід адміністратора</h2>
      <input
        type="text"
        placeholder="Логін"
        value={admin.username}
        onChange={(e) => handleChange('username', e.target.value)}
        style={{ width: '100%', padding: '10px', marginBottom: '12px', borderRadius: '6px', border: '1px solid #ccc' }}
      />
      <input
        type="password"
        placeholder="Пароль"
        value={admin.password}
```

```
        onChange={(e) => handleChange('password', e.target.value)}
        style={{ width: '100%', padding: '10px', marginBottom: '12px', borderRadius:
'6px', border: '1px solid #ccc' }}
      />
      {error && <p style={{ color: 'red', textAlign: 'center' }}>{error}</p>}
      <button
        onClick={handleSubmit}
        style={{ width: '100%', padding: '12px', backgroundColor: '#2980b9', color:
'white', border: 'none', borderRadius: '6px', fontSize: '16px' }}
      >
        Увійти
      </button>
    </div>
  );
}
```

P2PUpload.jsx

```
import React, { useState } from 'react';
export default function P2PUpload({ votings, participants, consensusParams }) {
  const [status, setStatus] = useState('');
  const handleUpload = () => {
    const payload = {
      timestamp: new Date().toISOString(),
      votings,
      participants,
      consensusParams
    };
    console.log('Надсилання до P2P-серверу:', payload);
    setStatus('Конфігурацію успішно надіслано');
  };
  return (
```

```
    <div style={{ maxWidth: '800px', margin: '40px auto', fontFamily: 'Segoe UI, sans-serif' }}>
      <h2 style={{ textAlign: 'center', marginBottom: '20px' }}>Завантаження на сервер</h2>
      <pre style={{ background: '#f4f4f4', padding: '20px', borderRadius: '10px', overflowX: 'auto' }}>
        {JSON.stringify({ votings, participants, consensusParams }, null, 2)}
      </pre>
      <button
        onClick={handleUpload}
        style={{ marginTop: '20px', width: '100%', padding: '12px', backgroundColor: '#27ae60', color: 'white', border: 'none', borderRadius: '6px', fontSize: '16px' }}
      >
        Надіслати конфігурацію
      </button>
      {status && <p style={{ textAlign: 'center', marginTop: '10px', color: 'green' }}>{status}</p>}
    </div>
  );
}
```