Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)
Навчально-науковий інститут комп'ютерних наук та
інформаційних технологій
(повне найменування інституту, назва факультету (відділення))
Кафедра інформаційних систем та комп'ютерного моделювання
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)
(рівень вищої освіти)

на тему: «Розробка криптогаманця для взаємодії з тестовою мережею Ethereum із використанням технологій Solidity та JavaScript»

Виконав: студент 4-го курсу групи ІСТ-41
спеціальності
126 «Інформаційні системи та технології»
(шифр і назва спеціальності)
Грабовський Н. Д.
(прізвище та ініціали)
Керівник Часковський О.Г.
(прізвище та ініціали)
Рецензент Крошний І. М.
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій

Кафедра інформаційних систем та комп'ютерного моделювання

Рівень вищої освіти перший (бакалаврський)

Спеціальність 126 «Інформаційні системи та технології»
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІСКМ

_____Сторожук О.Л.

«15» 11 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Грабовському Нікіті Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема бакалаврської роботи: «Розробка криптогаманця для взаємодії з тестовою мережею Ethereum із використанням технологій Solidity та JavaScript»,
керівник роботи: Часковський Олег Григорович, канд. с-г. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «15» листопада 2024 року, № С-884.

2. Термін подання студентом проекту (роботи) 12 червня 2025 р.

3. Вихідні дані до проекту (роботи) Розробити браузерне розширення криптогаманця для тестової мережі Ethereum. Реалізувати зручний та доступний інтерфейс для використання розроблених функцій проекту.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Стан проблемної області
Інформаційне забезпечення
Програмне та технічне забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових рішень)
Підготовка матеріалу до доповіді.

6. Дата видачі завдання 18 листопада 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Планування та архітектура	18.11.2024р. 21.11.2024р.	Виконано
2	Налаштування проекту	22.11.2024р. 23.11.2024р.	Виконано
3	Створення manifest.json	24.11.2024р.	Виконано
4	Базова архітектура додатку	25.11.2024р. 05.12.2024р.	Виконано
5	Сервіси та логіка	06.12.2024р. 10.01.2025р.	Виконано
6	UI Компоненти	11.01.2025р. 10.02.2025р.	Виконано
7	Дизайн та стилізація	11.02.2025р. 25.02.2025р.	Виконано
8	Інтеграція з блокчейном	26.02.2025р. 12.03.2025р.	Виконано
9	Безпека	13.03.2025р. 31.03.2025р.	Виконано
10	Тестування та налагодження	01.04.2025р. 30.04.2025р.	Виконано
11	Виправлення помилок	01.05.2025р. 10.06.2025р.	Виконано

Студент _____ (підпис) Грабовський Н.Д. (прізвище та ініціали)

Керівник роботи _____ (підпис) Часковський О.Г. (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота містить 63 сторінки пояснювальної записки, 23 рисунки, 6 додатків та 25 джерел.

Ця робота включає в себе розробку браузерного розширення криптогаманця ZeroTrace для надання можливості користувачам безпечного управління криптовалютними активами в тестовій мережі Ethereum. У цьому проекті були використані сучасні веб-технології: React, Ethers.js, Vite, Tailwind CSS.

Результати роботи рекомендується застосовувати як у практичних цілях для управління криптовалютними активами в мережі Ethereum , так і в освітньому процесі для студентів та розробників, що вивчають блокчейн-технології , а також в якості основи для подальшого прототипування та розширення функціональності децентралізованих систем.

Ключові слова: криптогаманець, Ethereum, блокчейн, браузерне розширення, React, смарт-контракти, Web3, DeFi.

ABSTRACT

The diploma thesis consists of 63 pages of explanatory text, 23 figures, 6 appendices, and 25 references.

This work involves the development of the ZeroTrace browser extension cryptocurrency wallet, aimed at providing users with secure management of cryptocurrency assets in the Ethereum test network. The project utilizes modern web technologies, including React, Ethers.js, Vite, and Tailwind CSS.

The results of this work are recommended to be applied both for practical purposes in managing cryptocurrency assets in the Ethereum network, and in the educational process for students and developers studying blockchain technologies, as well as serving as a foundation for further prototyping and expanding the functionality of decentralized systems.

Keywords: crypto wallet, Ethereum, blockchain, browser extension, React, smart contracts, Web3, DeFi.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити браузерне розширення для управління Ethereum-гаманцями з інтегрованими можливостями створення смарт-контрактів. Для забезпечення повнофункціональної взаємодії з блокчейн-мережею Ethereum необхідно реалізувати наступні модулі:

1. **Модуль управління гаманцями:**
    - Створення нових гаманців з генерацією мнемонічних фраз.
    - Імпорт існуючих гаманців через приватний ключ або seed-фразу.
    - Підтримка множинних гаманців з можливістю перемикання.
    - Безпечне шифрування та зберігання приватних ключів.
2. **Модуль транзакцій:**
    - Відправка ETH між адресами.
    - Підтримка ERC-20 токенів.
    - Відображення історії транзакцій.
    - Інтеграція з Block Explorer для деталізації операцій.
3. **Модуль роботи з токенами:**
    - Автоматичне виявлення популярних токенів.
    - Додавання кастомних токенів за адресою контракту.
    - Відображення балансів та детальної інформації про токени.
4. **Модуль генерації смарт-контрактів:**
    - Створення шаблонів ERC-20 токенів.
    - Створення шаблонів ERC-721 (NFT) токенів.
    - Інтеграція з Remix IDE для розгортання контрактів.
5. **Модуль безпеки:**
    - AES-256 шифрування чутливих даних.
    - Валідація введених адрес та параметрів транзакцій.

Для реалізації браузерного розширення використати фреймворк **React 19** з **Vite** як build tool та **@crxjs/vite-plugin** для створення extension bundle. Для взаємодії з

блокчейном використати бібліотеку **ethers.js v5**. Підключення до мережі Ethereum здійснити через **Infura API** з фокусом на тестову мережу **Sepolia**. Для шифрування приватних ключів застосувати бібліотеку **CryptoJS**. Інтерфейс користувача стилізувати за допомогою **Tailwind CSS** з додатковими custom стилями.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

AES – Advanced Encryption Standard (розширений стандарт шифрування)

API – Application Programming Interface (інтерфейс програмування додатків)

BIP – Bitcoin Improvement Proposal (пропозиція покращення Bitcoin)

CBC – Cipher Block Chaining (режим зчеплення блоків шифру)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

dApps – decentralized applications (децентралізовані додатки)

DOM – Document Object Model (об'єктна модель документа)

ECDSA – Elliptic Curve Digital Signature Algorithm (алгоритм цифрового підпису на еліптичних кривих)

ENS – Ethereum Name Service (служба імен Ethereum)

ERC – Ethereum Request for Comments (запит коментарів Ethereum)

ETH – Ethereum (криптовалюта Ethereum)

HD – Hierarchical Deterministic (ієрархічний детермінований)

HMAC – Hash-based Message Authentication Code (код автентифікації повідомлень на основі хешу)

HMR – Hot Module Replacement (гаряча заміна модулів)

HTML – HyperText Markup Language (мова розмітки гіпертексту)

IDE – Integrated Development Environment (інтегроване середовище розробки)

JSON – JavaScript Object Notation (нотація об'єктів JavaScript)

JSX – JavaScript XML (розширення синтаксису JavaScript)

NFT – Non-Fungible Token (невзаємозамінний токен)

npm – Node Package Manager (менеджер пакетів Node)

PBKDF2 – Password-Based Key Derivation Function 2 (функція деривації ключів на основі паролю)

QR – Quick Response (швидкий відгук)

RLP – Recursive Length Prefix (рекурсивний префікс довжини)

RPC – Remote Procedure Call (виклик віддаленої процедури)

SHA – Secure Hash Algorithm (алгоритм безпечного хешування)

UI – User Interface (користувацький інтерфейс)

UML – Unified Modeling Language (уніфікована мова моделювання)

URL – Uniform Resource Locator (уніфікований локатор ресурсів)

WAAPI – Web Animations API (API веб-анімацій)

ВСТУП

Сучасний стан розвитку блокчейн-технологій характеризується стрімким зростанням екосистеми децентралізованих додатків та збільшенням кількості користувачів криптовалют. За даними DeFiLlama [1], загальна заблокована вартість у DeFi – протоколах сягає понад 111 мільярдів доларів, що свідчить про масове впровадження технологій розподіленого реєстру. Водночас, більшість користувачів криптовалют стикаються з труднощами при взаємодії з блокчейн-мережами через складність існуючих інтерфейсів і можливості втратити власні кошти.

Основними проблемами сучасних криптогаманців є високий поріг входу для початківців, обмежена функціональність браузерних розширень, недостатня інтеграція з інструментами розробки смарт-контрактів та відсутність зручних засобів для роботи з тестовими мережами. Популярні рішення, такі як MetaMask, хоча і забезпечують базовий функціонал, не покривають потреби користувачів у швидкому створенні та тестуванні власних токенів без глибоких знань програмування.

Актуальність теми дослідження обумовлена необхідністю створення доступного та функціонального інструменту для взаємодії з екосистемою Ethereum, особливо в контексті навчання та експериментування з блокчейн-технологіями. Розвиток Web3-освіти та зростання інтересу до децентралізованих фінансів вимагають створення інтуїтивних інструментів, які дозволяють користувачам безпечно досліджувати можливості блокчейну без ризику втрати реальних коштів.

Об'єктом дослідження є процеси взаємодії користувачів з тестовою блокчейн-мережею Ethereum через браузерні розширення, включаючи управління криптовалютними активами, здійснення транзакцій та роботу зі смарт-контрактами.

Предметом дослідження виступають методи та технології розробки браузерних розширень для роботи з Ethereum, алгоритми безпечного зберігання приватних ключів та мнемонічних фраз, протоколи взаємодії з блокчейн-мережами та підходи до автоматизації створення смарт-контрактів.

Метою роботи є розробка функціонального браузерного розширення для управління Ethereum-гаманцями з інтегрованими можливостями створення та

розгортання смарт-контрактів, що забезпечує безпечну та зручну взаємодію з тестовою мережею Sepolia.

Для досягнення поставленої мети визначено наступні завдання:

- проаналізувати існуючі рішення у сфері криптогаманців;
- дослідити технології розробки браузерних розширень та бібліотеки для роботи з Ethereum;
- розробити архітектуру додатку з урахуванням вимог безпеки та користувацького досвіду;
- реалізувати функціонал створення, імпорту та управління множинними гаманцями;
- впровадити систему безпечного шифрування та зберігання приватних ключів;
- інтегрувати можливості роботи з ERC-20 токенами та відправки транзакцій;
- створити модуль генерації шаблонів смарт-контрактів для токенів стандартів ERC-20 та ERC-721;
- забезпечити інтеграцію з зовнішніми інструментами розробки, зокрема Remix IDE;
- провести тестування функціональності на тестовій мережі Sepolia.

Практичне значення роботи полягає у створенні доступного інструменту для навчання та експериментування з блокчейн-технологіями, який може бути використаний студентами, розробниками та дослідниками для швидкого прототипування та тестування рішень на базі Ethereum. Розроблене розширення знижує бар'єри входу у Web3-розробку, надаючи інтуїтивний інтерфейс для роботи з криптовалютами та смарт-контрактами без необхідності глибокого вивчення складних технічних деталей блокчейну.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Формулювання поставленої задачі

Основною задачею дослідження є розробка браузерного розширення для управління Ethereum-гаманцями, яке поєднує функціональність традиційних криптогаманців з інструментами для створення та розгортання смарт-контрактів. Специфіка задачі полягає у необхідності забезпечення безпечного зберігання приватних ключів та мнемонічних фраз в обмеженому середовищі браузера, реалізації інтуїтивного користувацького інтерфейсу для складних блокчейн-операцій та інтеграції з екосистемою розробки смарт-контрактів.

Технічна складність задачі обумовлена архітектурними обмеженнями браузерних розширень, які працюють в ізольованому середовищі з обмеженими можливостями доступу до системних ресурсів. Водночас, необхідно забезпечити високий рівень безпеки криптографічних операцій та зручність використання для користувачів різного рівня технічної підготовки.

1.2. Необхідні пояснення до задачі

Ethereum представляє собою децентралізовану платформу для виконання смарт-контрактів, яка використовує власну криптовалюту Ether (ETH) та підтримує стандарти токенів ERC-20 і ERC-721. Взаємодія з мережею Ethereum вимагає наявності гаманця, який зберігає приватні ключі і мнемонічні фрази та дозволяє підписувати транзакції.

Браузерні розширення як платформа для криптогаманців мають ряд переваг: постійну доступність в браузері, інтеграцію з децентралізованими додатками (dApps), можливість швидкого розгортання та оновлення. Однак вони також мають обмеження у вигляді Content Security Policy, ізоляції від системних ресурсів та залежності від політик браузера.

Тестова мережа Sepolia, яка є наступником застарілих тестових мереж Ropsten та Rinkeby, надає можливість безкоштовного тестування функціональності без використання реальних коштів. Це критично важливо для навчальних цілей та розробки прототипів.

1.3. Аналіз способів розв'язання задачі

Аналіз літератури та технічної документації виявив декілька підходів до розробки криптогаманців:

Десктопні додатки (Exodus, Atomic Wallet) забезпечують повний контроль над системними ресурсами та високий рівень безпеки, але вимагають окремої установки та мають обмежену інтеграцію з веб-додатками. Практика показує, що більшість користувачів віддають перевагу браузерним рішенням через незручність переключення між браузером та окремим додатком.

Мобільні додатки (Trust Wallet, Coinbase Wallet) забезпечують мобільність та зручність, але мають обмежену функціональність для розробки смарт-контрактів та складні механізми інтеграції з настільними браузерами.

Веб-гаманці (MyEtherWallet, MyCrypto) працюють безпосередньо в браузері, але вимагають постійного підключення до інтернету та мають підвищені ризики безпеки через можливість атак типу man-in-the-middle.

Браузерні розширення (MetaMask, Phantom) забезпечують оптимальний баланс між безпекою, зручністю та функціональністю. За даними DappRadar [2], 24.6 млн. користувачів щоденно взаємодіють з децентралізованими додатками через браузерні розширення.

Щодо технологічних рішень для розробки, аналіз показав:

Web3.js - найстаріша та найбільш документована бібліотека для взаємодії з Ethereum, однак має великий розмір та застарілу архітектуру.

Ethers.js - сучасна модульна бібліотека з TypeScript підтримкою, меншим розміром та кращою архітектурою. Дослідження npm trends [3] показують зростання популярності ethers.js на 99.52% за останні два роки.

React vs Vue.js vs Vanilla JavaScript для frontend розробки - React забезпечує найкращу екосистему компонентів та найширшу підтримку спільноти для Web3-додатків.

1.4. Обґрунтування вибраного шляху розв'язання задачі

Браузерне розширення обрано як оптимальну платформу з наступних причин:

1. **Інтеграція з екосистемою** - безпосередня взаємодія з dApps без додаткових налаштувань.
2. **Доступність** - не потребує окремої установки програмного забезпечення.
3. **Безпека** - ізольоване середовище виконання з обмеженими дозволами.
4. **Зручність розгортання** - можливість миттєвого оновлення через веб-магазини.

Технологічний стек обрано на основі критеріїв продуктивності, безпеки та зручності розробки:

- **React 19** - для побудови користувацького інтерфейсу через найкращу підтримку Web3 компонентів.
- **Ethers.js v5** - для взаємодії з Ethereum через модульність та TypeScript підтримку.
- **Vite + @crxjs/vite-plugin** - для збірки розширення через швидкість розробки та сучасні можливості.
- **CryptoJS** - для шифрування через перевірену криптографічну безпеку.
- **Tailwind CSS** - для стилізації через ефективність та консистентність дизайну.

1.5. Огляд існуючих аналогів

MetaMask є найпопулярнішим криптогаманцем з понад 100 мільйонами [4] користувачів. Основні переваги: широка підтримка мереж, інтеграція з більшістю dApps, відкритий вихідний код. Недоліки: складний інтерфейс для початківців, обмежена функціональність для розробки смарт-контрактів, відсутність вбудованих інструментів для створення токенів.

Rainbow Wallet позиціонується як більш зручний аналог MetaMask з фокусом на DeFi. Переваги: інтуїтивний дизайн, вбудована підтримка NFT, кращий user experience. Недоліки: відсутність інструментів розробки, обмежена підтримка тестових мереж, закритий вихідний код.

Trust Wallet є офіційним [5] гаманцем Binance Smart Chain. Переваги: підтримка множини мереж, мобільна та десктопна версії, вбудований DEX. Недоліки: централізоване управління, обмежена функціональність для Ethereum розробки, відсутність інтеграції з IDE.

Phantom спеціалізується на екосистемі Solana, але додав підтримку Ethereum. Переваги: швидкість роботи, сучасний інтерфейс. Недоліки: обмежена функціональність для Ethereum, відсутність інструментів розробки.

Frame позиціонується як privacy-focused рішення. Переваги: високий рівень безпеки, підтримка множини мереж. Недоліки: складність налаштування, обмежена функціональність, мала спільнота користувачів.

Порівняльний аналіз показує, що жоден з існуючих рішень не поєднує зручність використання з інструментами для розробки смарт-контрактів. Більшість рішень фокусується або на масовому користувачі (MetaMask, Rainbow), або на розробниках (Frame), але не надає інтегрованого досвіду.

1.6. Розширене обґрунтування актуальності розробки

Актуальність розробки підтверджується кількома тенденціями галузі:

Зростання Web3-освіти: За даними ixperience [6], 50 найкращих університетів світу вже пропонують щонайменше один курс, присвячений блокчейн-технологіям або криптовалютам. 22 % з них мають у програмі більше одного такого курсу, а 9 % студентів вже пройшли відповідне навчання.

Розвиток low-code рішень: Ринок low-code платформ за прогнозами Gartner [7] досягне $16.5 млрд до 2027 року. В блокчейн-сфері спостерігається аналогічна тенденція - користувачі потребують інструментів для створення токенів та додатків без глибоких знань програмування.

Фрагментація екосистеми розробки: Розробники блокчейн-додатків змушені використовувати множину різних інструментів для повного циклу створення смарт-контрактів - від написання коду до тестування та розгортання. Інтеграція цих інструментів залишається значною проблемою, що уповільнює процес розробки та підвищує поріг входу.

Безпека та user experience: За даними Chainalysis [8], у 2024 році викрадені кошти зросли приблизно на 21.07% у порівнянні з аналогічним періодом минулого року до $2.2 млрд. Це підкреслює критичну важливість зручних та безпечних інтерфейсів, оскільки найбільша частина крадіжок стається через необачність користувачів.

Тенденції розвитку тестових мереж: Ethereum Foundation активно розвиває тестові мережі, зокрема Sepolia, що свідчить про розуміння важливості безпечного середовища для експериментування. Однак інструменти для роботи з тестовими мережами часто дублюють складність mainnet-рішень, що не відповідає освітнім потребам.

Таким чином, розробка інтегрованого рішення, яке поєднує функціональність криптогаманця з інструментами розробки смарт-контрактів, відповідає актуальним потребам ринку та може значно спростити процес навчання та експериментування з блокчейн-технологіями для широкого кола користувачів.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Побудова дерева проблем та дерева цілей

Дерево проблем є важливим інструментом для аналізу основних труднощів, що існують у предметній області, та дозволяє візуалізувати причинно-наслідкові зв'язки між ними. Для даної роботи дерево проблем (рис. 2.1) висвітлює виклики, пов'язані з використанням існуючих криптогаманців, зокрема їх складність для початківців, обмежений функціонал для розробників та потенційні ризики безпеки.

Головна проблема: Високий поріг входу та недостатня функціональність існуючих криптогаманців для користувачів, що навчаються або експериментують з блокчейн-технологіями.

Основна проблема 1. Складність взаємодії з блокчейн-мережами для початківців.

Причина 1.1: Існуючі інтерфейси є складними та неінтуїтивними.

Причина 1.2: Високі ризики втрати реальних коштів під час навчання та експериментів.

Основна проблема 2. Обмежена функціональність для розробників смарт-контрактів.

Причина 2.1: Відсутність вбудованих інструментів для швидкого створення та тестування власних токенів у популярних гаманцях, як-от MetaMask.

Причина 2.2: Фрагментація екосистеми: розробники змушені використовувати безліч різних інструментів для повного циклу розробки.

Основна проблема 3. Недостатній рівень безпеки та зручності для користувачів.

Причина 3.1: Значна кількість крадіжок криптовалют стається через необачність користувачів та складність інтерфейсів.

Причина 3.2: Існуючі інструменти для роботи з тестовими мережами часто дублюють складність рішень для основної мережі, що не відповідає освітнім потребам.

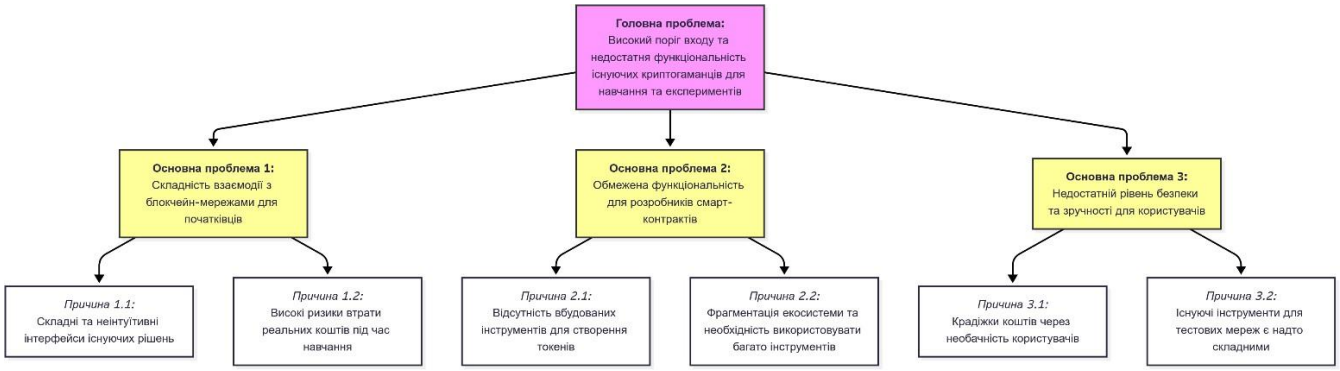

Рисунок 2.1 - Дерево проблем

Дерево цілей (рис. 2.2) будується як зворотна структура до дерева проблем і дозволяє перетворити виявлені проблеми на чітко визначені цілі та завдання. У даному випадку, розробка нового криптогаманця може вирішити існуючі проблеми, надавши користувачам безпечний, функціональний та інтуїтивно зрозумілий інструмент.

Головна мета: Розробити функціональний криптогаманець у вигляді браузерного розширення, який спростить взаємодію з тестовою мережею Ethereum та надасть інструменти для розробки смарт-контрактів.

Основна ціль 1. Спростити взаємодію з блокчейном для початківців.

Завдання 1.1: Реалізувати інтуїтивно зрозумілий користувацький інтерфейс.

Завдання 1.2: Забезпечити безпечне середовище для експериментів у тестовій мережі Sepolia без ризику втрати реальних коштів.

Основна ціль 2. Розширити функціонал для розробки смарт-контрактів.

Завдання 2.1: Створити модуль для автоматичної генерації шаблонів смарт-контрактів стандартів ERC-20 та ERC-721.

Завдання 2.2: Забезпечити інтеграцію з Remix IDE.

Основна ціль 3. Підвищити безпеку та зручність використання.

Завдання 3.1: Впровадити надійну систему безпеки, що включає AES-256 шифрування приватних ключів та їх локальне зберігання.

Завдання 3.2: Розробити доступний інструмент, що знижує бар'єри входу у Web3-розробку для широкого кола користувачів.

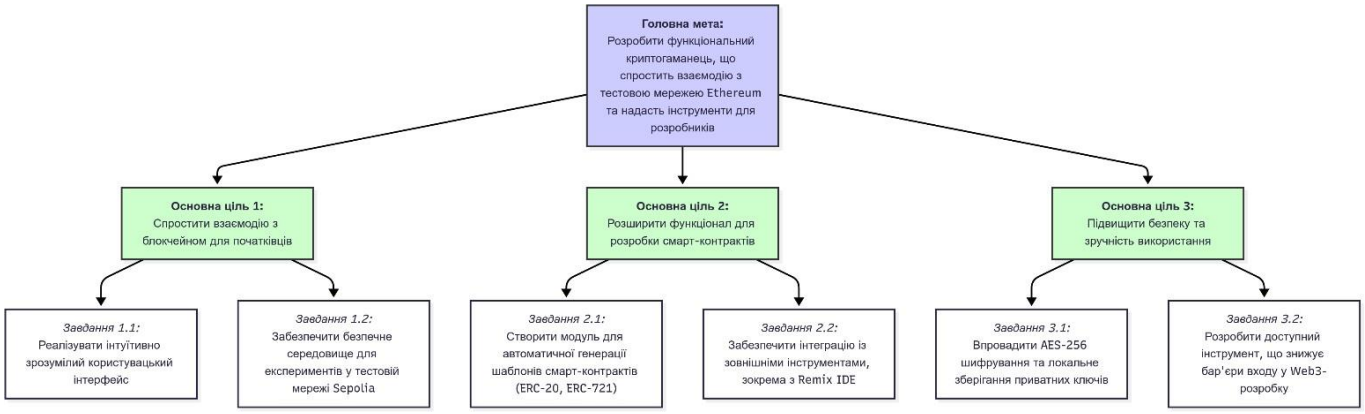

Рисунок 2.2 - Дерево цілей

2.2. Визначення об’єктів дослідження

Криптогаманець представляє собою програмний інструмент для безпечного зберігання та управління криптографічними ключами, що забезпечує доступ до криптовалютних активів користувача в блокчейн мережі Ethereum. Основною функцією гаманця є генерація, зберігання та використання приватних ключів для підписання транзакцій без компромісу безпеки.

Приватний ключ є центральним елементом системи безпеки, що представляє собою 256-бітне випадкове число, яке надає повний контроль над активами на відповідній Ethereum адресі. Цей ключ використовується для створення цифрових підписів транзакцій та ніколи не повинен передаватися через незахищені канали зв'язку.

Ethereum адреса формується шляхом застосування криптографічних перетворень до публічного ключа і служить як унікальний ідентифікатор гаманця в мережі. Адреса має вигляд 42-символьного шістнадцяткового рядка з префіксом "0x" та включає контрольну суму для запобігання помилкам введення.

Мнемонічна фраза згідно стандарту BIP-39 [9] є зручним способом резервного копіювання та відновлення гаманця, що складається з 12 слів зі стандартного словника. Ця фраза дозволяє відновити доступ до всіх ключів та адрес, які можуть

бути згенеровані за допомогою ієрархічно детермінованого алгоритму згідно BIP-32 [10].

Транзакція в контексті Ethereum містить інформацію про відправника, отримувача, суму переказу, комісію за газ та додаткові дані. Кожна транзакція повинна бути підписана приватним ключем відправника для підтвердження авторизації операції.

Токени ERC-20 представляють стандартизований інтерфейс для створення взаємозамінних токенів на платформі Ethereum. Вони функціонують через смарт-контракти та дозволяють користувачам створювати власні цифрові активи з визначеними правилами обігу.

Смарт-контракт є самовиконуваною програмою, розгорнутою в блокчейні Ethereum, що автоматично виконує умови угоди при настанні визначених подій. У контексті криптогаманця смарт-контракти використовуються для управління токенами та виконання складних фінансових операцій.

Інтерфейс взаємодії з блокчейном забезпечує комунікацію з вузлами мережі Ethereum через JSON-RPC протокол для отримання інформації про стан блокчейну та відправлення транзакцій. Цей шар абстракції дозволяє додатку працювати з розподіленою мережею без необхідності запуску власного вузла.

Модуль управління токенами відповідає за взаємодію зі смарт-контрактами токенів, отримання балансів, метаданих та виконання операцій передачі токенів між адресами. Він забезпечує уніфікований інтерфейс для роботи з різними типами токенів стандарту ERC-20.

2.3. Інформаційна модель системи

Інформаційна модель криптогаманця побудована на основі ієрархічної структури даних, що відображає взаємозв'язки між основними інформаційними сутностями системи. Центральним елементом інформаційної моделі є гаманець (Wallet), який містить унікальний ідентифікатор, Ethereum адресу, зашифрований приватний ключ, мнемонічну фразу, назву гаманця та часову мітку створення.

Інформаційні потоки в системі організовані відповідно до принципів безпеки та функціональних вимог. Вхідний потік даних від користувача включає пароль для шифрування та розшифрування чутливої інформації, адресу отримувача для проведення транзакцій, суму для відправлення та мнемонічну фразу або приватний ключ для імпорту існуючих гаманців.

Вихідний потік даних до користувача забезпечує відображення балансу гаманця в ETH та різних токенах, історії всіх проведених транзакцій, QR-коду адреси для зручного отримання коштів та статусу виконання поточних операцій. Система також генерує повідомлення про помилки та підтвердження успішного виконання операцій.

Потік даних з блокчейном реалізується через встановлення з'єднання з мережею Ethereum для отримання балансів через JSON-RPC запити, відправлення підписаних транзакцій до мережі, моніторингу статусу транзакцій в реальному часі та запитів інформації про токени та їх метадані.

Схема зберігання даних (рис. 2.3) передбачає використання локального сховища браузера для збереження зашифрованих приватних ключів та метаданих гаманців. Всі чутливі дані шифруються за допомогою алгоритму AES-256 перед збереженням. Структура даних організована таким чином, що кожен гаманець має унікальний ідентифікатор, який використовується як ключ для доступу до зашифрованої інформації.

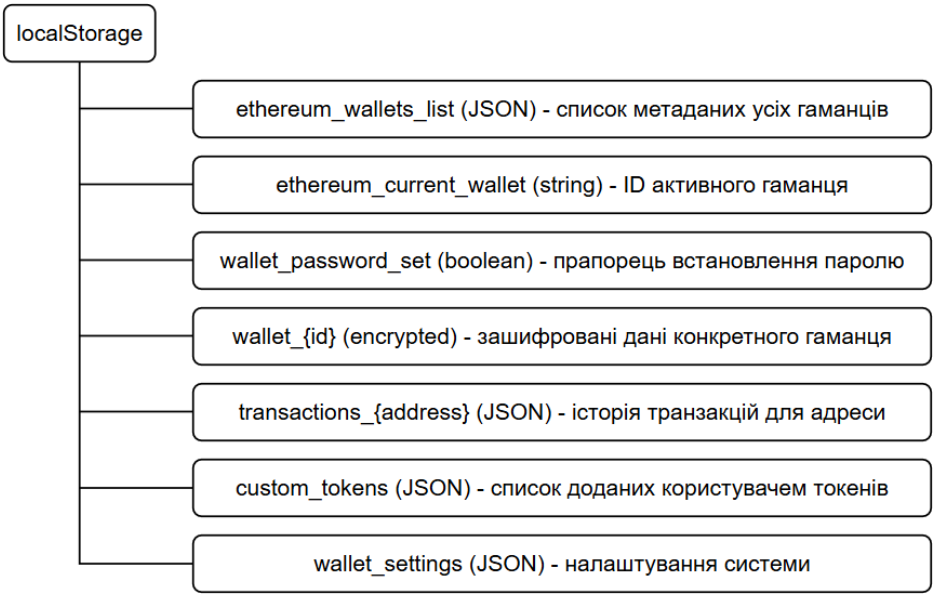

Рисунок 2.3 - Схема зберігання даних

2.4. Обмеження на вхідні та вихідні дані

Система накладає строгі обмеження на формат та структуру вхідних даних для забезпечення безпеки та коректної роботи. Пароль користувача повинен містити мінімум 8 символів та максимум 128 символів, може включати всі символи Unicode та є обов'язковим для всіх операцій з приватними ключами. Система не накладає додаткових вимог щодо складності паролю, залишаючи цей вибір на розсуд користувача.

Ethereum адреса повинна відповідати стандартному формату у вигляді 42-символьного рядка, що починається з префікса "0x". Система виконує валідацію контрольної суми згідно зі стандартом EIP-55 для запобігання помилкам введення. При введенні адреси регістр символів не є критичним, проте при валідації система строго перевіряє відповідність контрольної суми.

Мнемонічна фраза може складатися з 12, 15, 18, 21 або 24 слів, проте поточна реалізація системи підтримує виключно 12-словні фрази. Всі слова повинні належати до стандартного англійського словника BIP-39, що містить 2048 унікальних слів. Система автоматично перевіряє контрольну суму мнемонічної фрази згідно зі стандартом BIP-39 для забезпечення цілісності даних.

Приватний ключ може бути представлений у форматі 64-символьного шістнадцяткового рядка або 66 символів з префіксом "0x". Діапазон допустимих значень обмежений математичними властивостями еліптичної кривої secp256k1 [11] та повинен знаходитися в межах від 1 до n-1, де n представляє порядок кривої.

Сума транзакції має мінімальне обмеження в 0.0001 ETH або еквівалент в токенах для уникнення мікротранзакцій, що можуть створювати додаткове навантаження на мережу. Максимальна сума обмежена поточним балансом гаманця з вирахуванням необхідної комісії за газ. Система підтримує точність до 18 десяткових знаків для ETH, а для токенів точність залежить від параметра decimals конкретного токена.

Обмеження на вихідні дані забезпечують зручність використання та читабельність інформації. Відображення балансів обмежено 8 знаками після коми для

ETH та 4 знаками для токенів з метою спрощення сприйняття. Для дуже малих значень менше 0.0001 система використовує експоненціальний запис для компактного представлення.

Історія транзакцій зберігається повністю в локальному сховищі без обмежень на кількість записів. Сортування здійснюється за часовою міткою в порядку спадання для відображення найновіших операцій на початку списку. Система не реалізує функціонал фільтрації, відображаючи всі збережені транзакції.

Експорт даних строго контролюється з міркувань безпеки. Приватні ключі ніколи не експортуються в незашифрованому вигляді, мнемонічна фраза відображається лише один раз при створенні гаманця для запису користувачем, а QR-коди містять виключно публічну адресу гаманця без будь-якої чутливої інформації.

2.5. Концептуальне проектування системи

Концептуальне проєктування системи криптогаманця базується на фундаментальних архітектурних принципах, що забезпечують надійність, безпеку та масштабованість рішення. Принцип модульності реалізується через розділення системи на незалежні компоненти React, сервісні модулі та утилітарні функції, що забезпечує легку підтримку та можливість розширення функціональності без впливу на інші частини системи.

Безпека за замовчуванням є наріжним каменем архітектури, згідно з яким всі чутливі дані зберігаються виключно в зашифрованому вигляді, приватні ключі ніколи не передаються через мережу та не зберігаються в незахищеному стані. Система розроблена таким чином, що навіть розробники не мають доступу до незашифрованих приватних ключів користувачів.

Принцип децентралізації забезпечується прямою взаємодією з блокчейном через публічні RPC-ендпоінти без використання проміжних серверів або централізованих сервісів. Це гарантує, що користувачі зберігають повний контроль над своїми активами та не залежать від доступності третіх сторін.

Користувацький контроль реалізується через надання повного контролю над ключами та даними, при цьому система функціонує як інструмент, а не як кастодіальний сервіс. Користувачі самостійно відповідають за збереження своїх мнемонічних фраз та паролів.

Компонентна структура системи побудована з використанням сучасного підходу React з чітким розподілом відповідальності між презентаційними та функціональними компонентами. Презентаційні компоненти включають:

- WalletLogin для автентифікації користувача.
- WalletCreate для створення та імпорту гаманців.
- WalletDashboard як головний інтерфейс управління.
- TokensTab для управління токенами.
- TransactionsTab для перегляду історії операцій.
- WalletsTab для управління множинними гаманцями.
- TokenCreatorTab для створення власних токенів.

Сервісні модулі забезпечують бізнес-логіку системи через:

- web3.js для взаємодії з Ethereum блокчейном.
- storage.js для управління локальним сховищем.
- contract.js для роботи зі смарт-контрактами.

Така архітектура забезпечує чіткий поділ між представленням даних та їх обробкою.

Модель безпеки реалізована на трьох рівнях захисту для забезпечення максимальної надійності системи. Рівень зберігання забезпечує AES-256 шифрування всіх чутливих даних перед їх збереженням у локальному сховищі браузера. Рівень доступу вимагає обов'язкової автентифікації паролем для доступу до будь-яких функцій, пов'язаних з приватними ключами. Рівень транзакцій використовує криптографічний підпис ECDSA для забезпечення автентичності та цілісності всіх операцій.

2.6. Концептуальна модель системи

Концептуальна модель системи представлена у вигляді UML діаграми класів (рис. 2.4), що відображає основні сутності та їх взаємозв'язки в рамках архітектури криптогаманця. Модель демонструє структурні зв'язки між компонентами системи та визначає інтерфейси взаємодії між різними рівнями абстракції

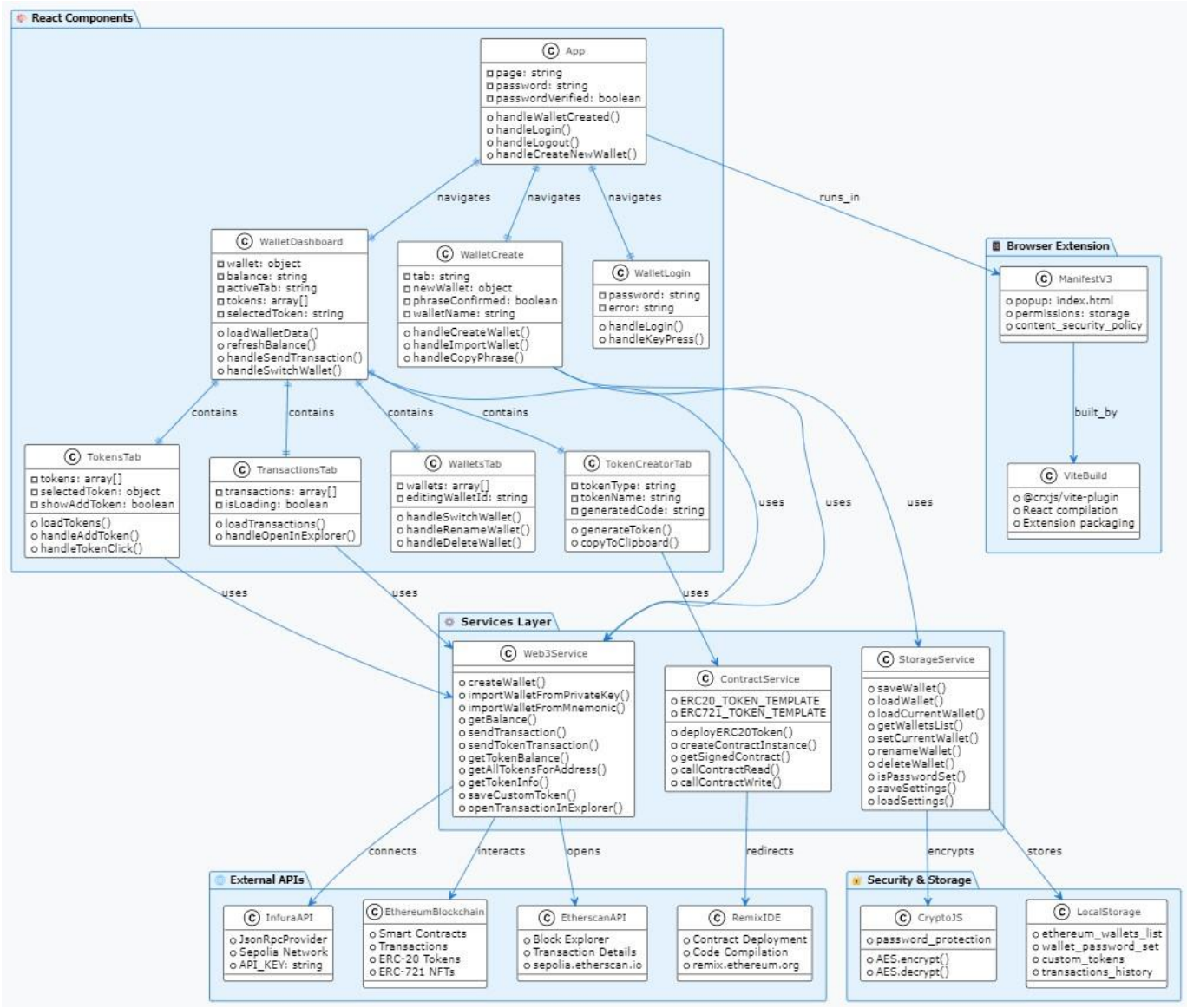

Рисунок 2.4 - UML діаграма класів

Система криптогаманця може перебувати в декількох чітко визначених станах, що забезпечує контрольований перехід між різними режимами роботи. Початковий стан передбачає перевірку наявності збережених гаманців у локальному сховищі та

визначення необхідності створення нового гаманця або автентифікації для доступу до існуючого.

Стан створення гаманця активується при генерації нової пари ключів або імпорті існуючих ключів через мнемонічну фразу чи приватний ключ. У цьому стані система генерує всі необхідні криптографічні матеріали та готує їх для безпечного зберігання.

Автентифікація є обов'язковим етапом для доступу до збережених гаманців, під час якого система перевіряє правильність введеного паролю та розшифровує приватні ключі для подальшого використання.

Активний стан надає доступ до всіх функцій гаманця, включаючи перегляд балансів, історії транзакцій, відправлення коштів та управління токенами. У цьому стані система регулярно оновлює інформацію з блокчейну та відображає актуальні дані.

Обробка транзакції включає підписання операції приватним ключем, валідацію параметрів та відправлення до мережі Ethereum. Система забезпечує перевірку достатності коштів та коректності адреси отримувача.

Очікування підтвердження передбачає моніторинг статусу відправленої транзакції в блокчейні до отримання необхідної кількості підтверджень від мережі.

Діаграма послідовності для транзакції (рис. 2.5) ілюструє детальний процес взаємодії між компонентами системи під час виконання переказу коштів, починаючи від ініціювання користувачем та закінчуючи підтвердженням операції в блокчейні.

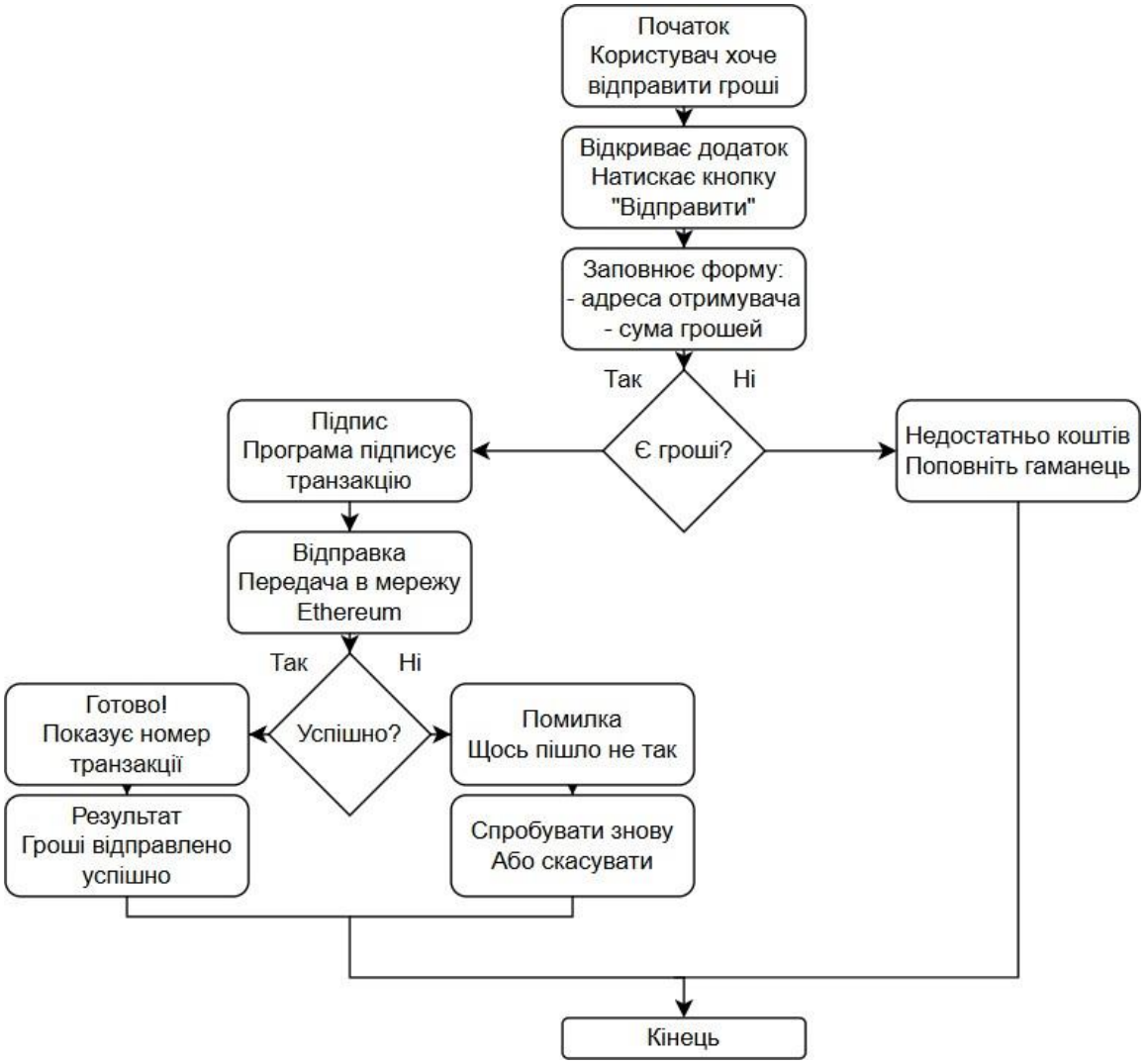

Рисунок 2.5 - Діаграма послідовності для транзакцій

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Обґрунтування вибору програмних засобів

Для реалізації системи криптогаманця було обрано сучасний технологічний стек, що забезпечує оптимальне поєднання продуктивності, безпеки та зручності розробки. **React 19.1.0** [12] обрано як основний фреймворк для побудови користувацького інтерфейсу завдяки компонентному підходу, що забезпечує модульність та повторне використання коду. Віртуальний DOM оптимізує продуктивність при частих оновленнях інтерфейсу, велика екосистема бібліотек та інструментів надає широкі можливості для розробки, а підтримка сучасних патернів розробки через React Hooks спрощує управління станом додатку.

Ethers.js 5.7.2 [13] використовується для взаємодії з блокчейном Ethereum як легка бібліотека порівняно з Web3.js. Модульна архітектура дозволяє імпортувати лише необхідні компоненти, що зменшує розмір фінального bundle. Вбудована підтримка ENS, HD гаманців та підписання транзакцій спрощує реалізацію криптографічних операцій, а типізований API з детальною документацією прискорює процес розробки.

Vite 6.3.5 [14] як інструмент збірки проєкту забезпечує швидкий холодний старт завдяки ES модулям, миттєве оновлення модулів під час розробки через Hot Module Replacement, оптимізовану збірку для продакшену через Rollup та нативну підтримку TypeScript і JSX без додаткової конфігурації.

Tailwind CSS 4.1.6 [15] для стилізації інтерфейсу реалізує utility-first підхід, що прискорює розробку UI через готові класи. Мінімальний розмір фінального CSS досягається через PurgeCSS, що видаляє невикористані стилі. Консистентна система дизайну забезпечує уніфікований вигляд компонентів, а адаптивність за замовчуванням спрощує створення responsive інтерфейсів.

Додаткові бібліотеки включають **CryptoJS 4.2.0** [16] для реалізації криптографічних алгоритмів, зокрема AES шифрування для захисту приватних ключів, PBKDF2 для деривації ключів з паролів та SHA-256 для хешування. **Framer**

Motion 12.12.1 [17] надає декларативний API для складних анімацій з оптимізованою продуктивністю через Web Animations API та підтримкою жестів і інтерактивності. **@crxjs/vite-plugin 1.0.14** забезпечує інтеграцію з Chrome Extension API [18] через автоматичну збірку розширення, Hot reload для швидкої розробки та підтримку Manifest V3.

Для розгортання контрактів використовується інтеграція з Remix IDE [19] через Infura API [20].

3.2. Архітектура програмної системи

Система побудована за принципом багатошарової архітектури (рис. 3.1) з чітким розподілом відповідальності між компонентами. Архітектурний підхід передбачає розділення на презентаційний шар, що відповідає за відображення інтерфейсу користувача, бізнес-логічний шар для обробки даних та криптографічних операцій, та шар доступу до даних для взаємодії з локальним сховищем і блокчейном.

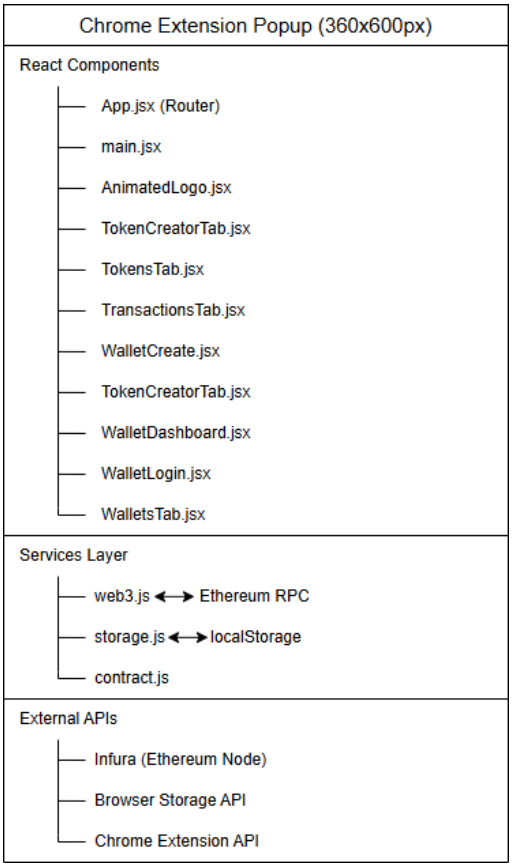

Рисунок 3.1 - Загальна архітектура системи

Організація файлової структури проєкту (рис. 3.2) відповідає best practices для React додатків з розділенням на папки components для React компонентів, services для бізнес-логіки, utils для допоміжних функцій, assets для статичних ресурсів та styles для стилів. Така структура забезпечує легку навігацію по коду, простоту підтримки та можливість масштабування проєкту.

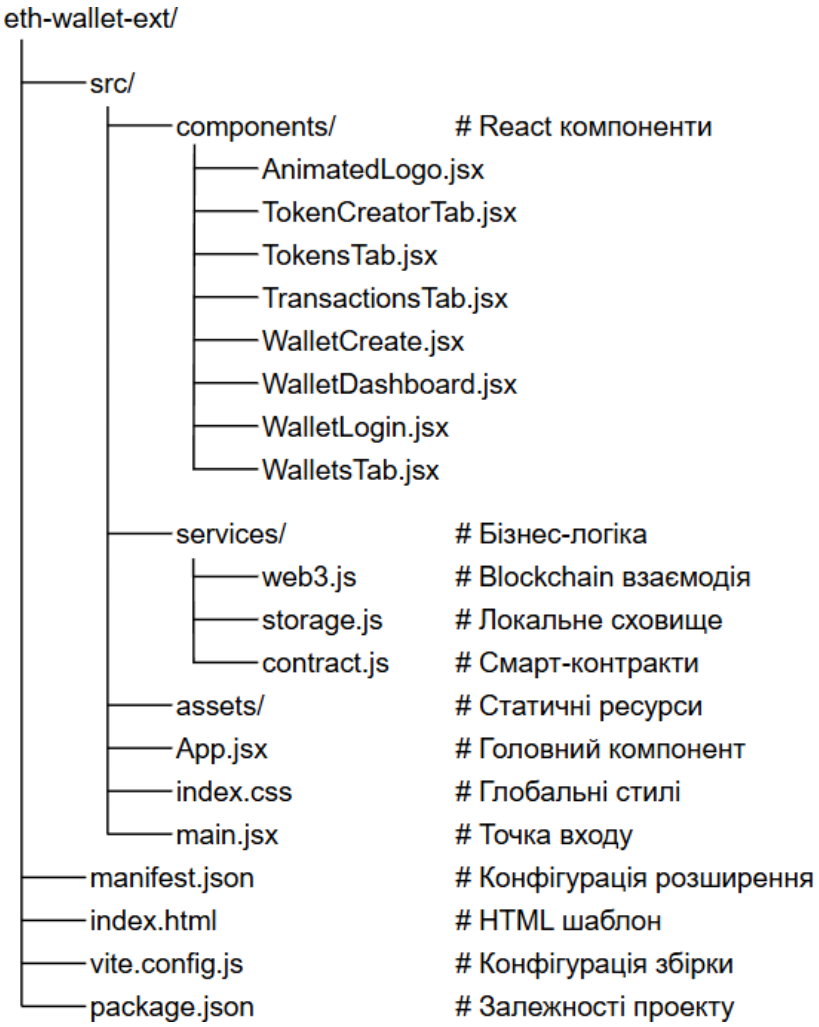

Рисунок 3.2 - Структура проекту

3.3. Опис програмних модулів

Модуль управління гаманцями реалізує безпечне зберігання через AES-256 шифрування [21] всіх чутливих даних. Система використовує стандарти BIP-44 [22] для генерації та управління ієрархічними детермінованими гаманцями згідно специфікації Ethereum [23]:

- saveWallet(walletData, password) - зберігає новий гаманець із шифруванням AES.
- loadWallet(walletId, password) - завантажує та розшифровує дані гаманця.
- getWalletsList() - повертає список метаданих усіх гаманців.
- deleteWallet(walletId) - видаляє гаманець та його дані.

```
// Метадані гаманців (незашифровані)
{
  id: "wallet_1234567890",
  address: "0x...",
  name: "Основний гаманець",
  dateCreated: 1702345678901
}

// Дані гаманця (зашифровані)
{
  address: "0x...",
  privateKey: "0x...",
  mnemonic: "word1 word2 ... word12"
}
```

Модуль взаємодії з блокчейном (web3.js) забезпечує всю функціональність роботи з Ethereum через управління гаманцями за допомогою:

- createWallet() – генерація нової пари ключів та мнемонічної фрази.
- importWalletFromPrivateKey(privateKey) - імпорт за приватним ключем.
- importWalletFromMnemonic(mnemonic) - імпорт за seed фразою.

Робота з балансами включає:

- getBalance(address) - отримання балансу ETH.
- getTokenBalance(tokenAddress, ownerAddress) - баланс ERC-20 токена.
- getAllTokensForAddress(address) - список всіх токенів з балансами.

Функції транзакцій охоплюють:

- sendTransaction(privateKey, toAddress, amount) - відправку ETH.

- sendTokenTransaction(...) - відправку токенів ERC-20.
- getTransactionHistory(address) - історію транзакцій.

Модуль роботи зі смарт-контрактами (contract.js) містить шаблони та функціонал для створення власних токенів, включаючи шаблони ERC-20 [24] та ERC-721 [25] токенів з базовою реалізацією стандартних функцій, mapping для балансів та allowances, події Transfer та Approval для відстеження операцій.

```
// Шаблон ERC-20 токена
const ERC20_TOKEN_TEMPLATE = `
pragma solidity ^0.8.4;

contract TOKEN_NAME {
    string public name = "TOKEN_NAME";
    string public symbol = "TOKEN_SYMBOL";
    uint8 public decimals = 18;
    uint256 public totalSupply = TOKEN_TOTAL_SUPPLY * 10 ** decimals;

    mapping(address => uint256) private _balances;
    mapping(address => mapping(address => uint256)) private _allowances;

    // ... implementation
}`;
```

Компонент логіну (WalletLogin.jsx) (рис. 3.3) забезпечує вхід до системи і має анімований логотип з частинками, що хаотично переміщаються у заданому полі, поле введення паролю з валідацією, кнопку входу та можливість створення нового гаманця при відсутності збережених даних.

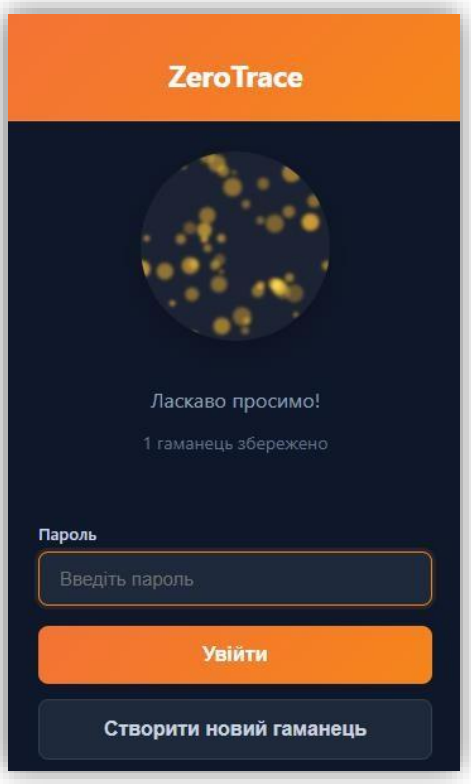

Рисунок 3.3 - Інтерфейс екрану логіну

Компонент створення гаманця (WalletCreate.jsx) (рис. 3.4, 3.5) надає три основні режими роботи для створення нового гаманця з генерацією мнемонічної фрази, імпорту за приватним ключем та імпорту за seed фразою. Створення нового гаманця включає генерацію 12-словної мнемонічної фрази, відображення згенерованої фрази з можливістю копіювання, попередження про важливість збереження seed фрази та підтвердження збереження користувачем.

Імпорт за приватним ключем (рис. 3.6) включає поле для введення 64-символьного hex ключа з валідацією формату, автоматичне генерування адреси з приватного ключа та створення гаманця без мнемонічної фрази.

Імпорт за seed фразою (рис. 3.7) передбачає поле для введення 12 слів з автокомплітом зі словника BIP-39, валідацію кожного слова та контрольної суми всієї фрази, автоматичну генерацію адреси та приватного ключа з мнемонічної фрази.

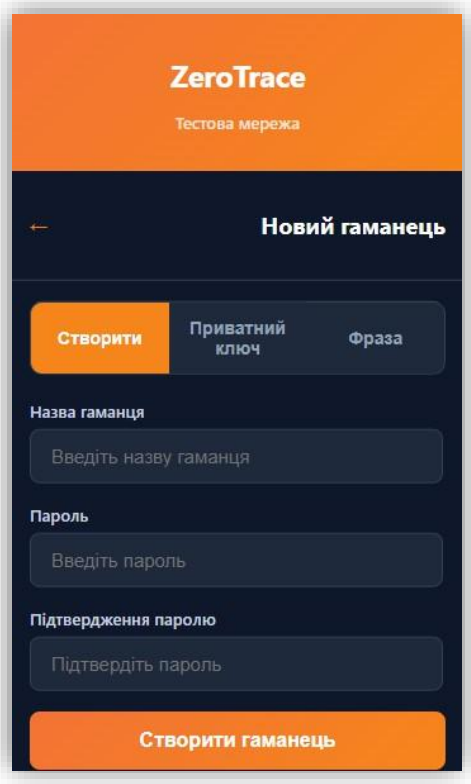

Рисунок 3.4 - Екран створення гаманця

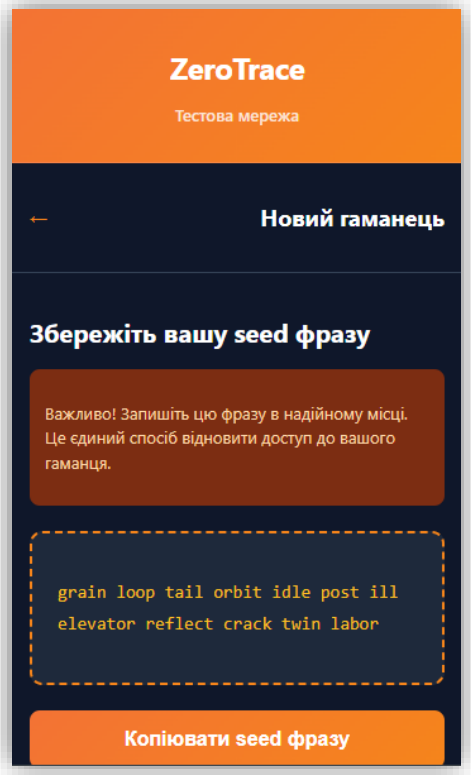

Рисунок 3.5 - Копіювання та збереження seed – фрази

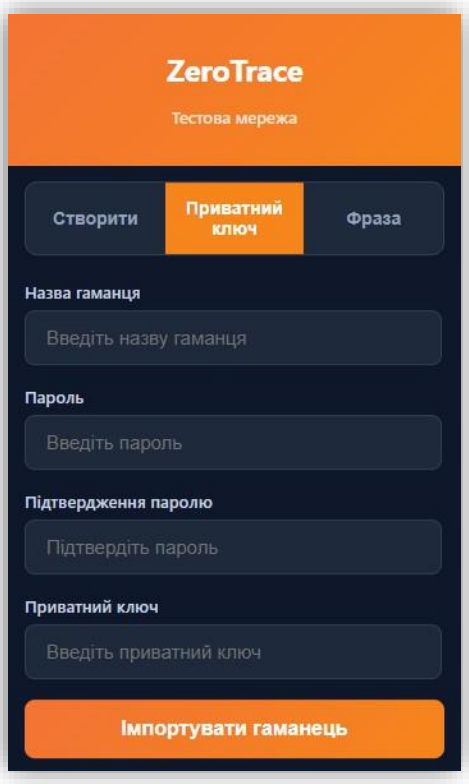

Рисунок 3.6 - Імпорт гаманця за приватним ключем

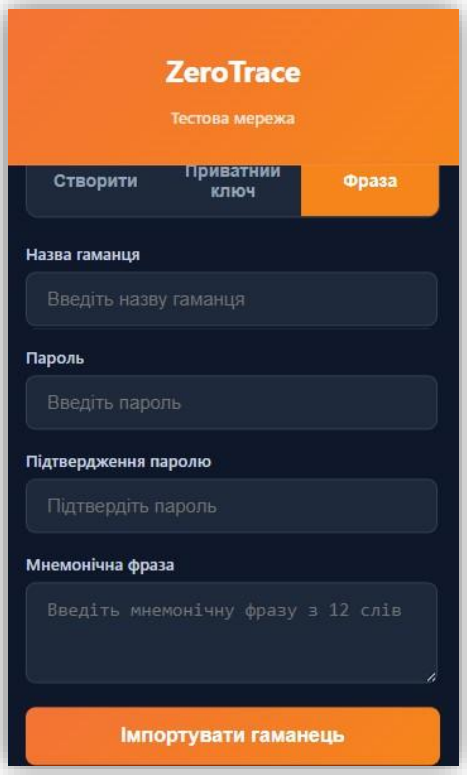

Рисунок 3.7 - Імпорт гаманця за seed – фразою

Головний компонент (WalletDashboard.jsx) (рис. 3.8) – це центральний хаб управління гаманцем. У ньому знаходиться:

- Основна навагіція по вкладкам унизу («Гаманець», «Токени», «Транзакції», «Створити», «Гаманці»).
- Відображається баланс у ETH з можливістю оновлення, у разі якщо баланс сам не оновився після виведення/попвнення.
- Можливість побачити адресу гаманця, скопіювати її або відобразити QR-код для того, щоб його сканувати та відправити на цю адресу кошти.
- Кнопка «Відправити» (рис. 3.9), для того, щоб відправити усі доступні токени на інакший гаманець. Також містить у собі dropdown меню для вибору токена.
- Кнопка «Вийти», щоб заблокувати гаманець.

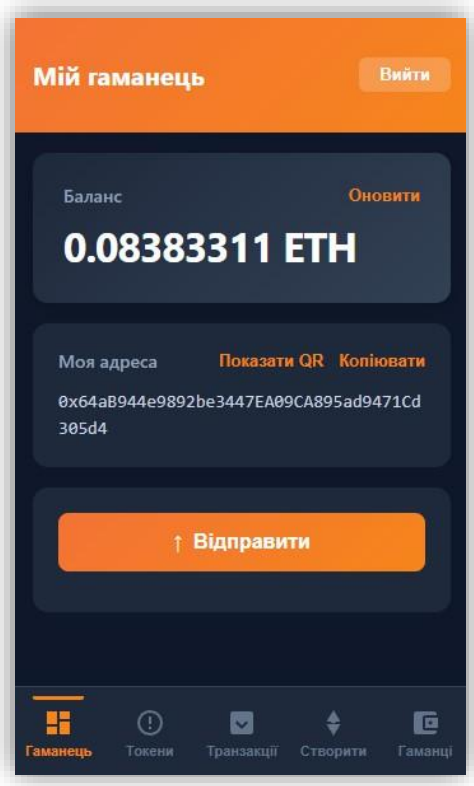

Рисунок 3.8 - Центральний хаб управління гаманцем

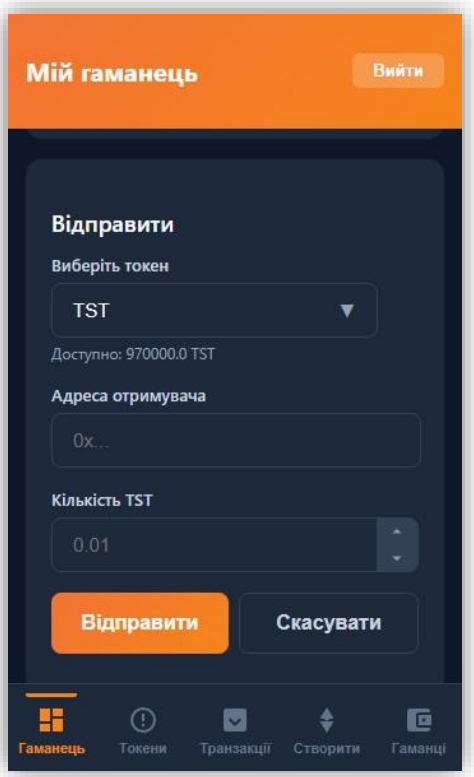

Рисунок 3.9 - Форма відправлення транзакції

Компонент імпортованих токенів (TokensTab.jsx) (рис. 3.10) містить в собі список та кількість токенів на балансі у мережі Sepolia та можливість додати токен власноруч.

При натисканні на кнопку «Додати токен» (рис. 3.11) відкривається поле для вводу адреси контракту токену та кнопкою «Додати», для того, щоб імпортувати токен і бачити їхній баланс на вибраному гаманці.

При натисканні на токен, реалізовано відображення детальної інформації про вибраний токен (рис. 3.12) та інтеграція з Block Explorer.

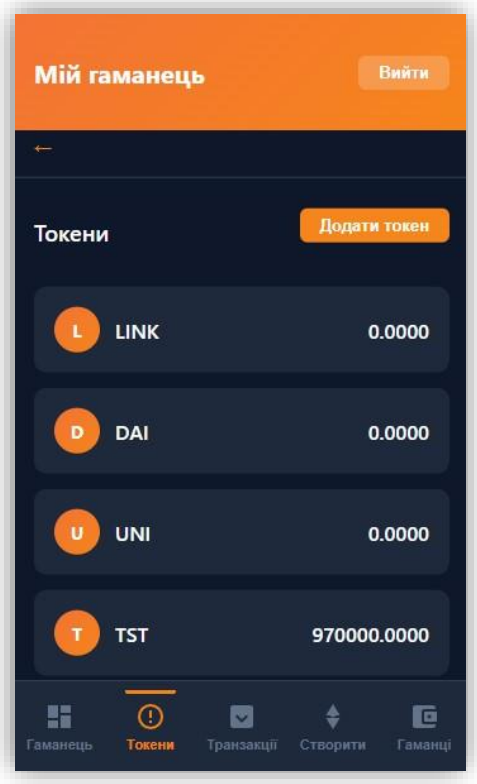

Рисунок 3.10 - Екран з відображенням токенів

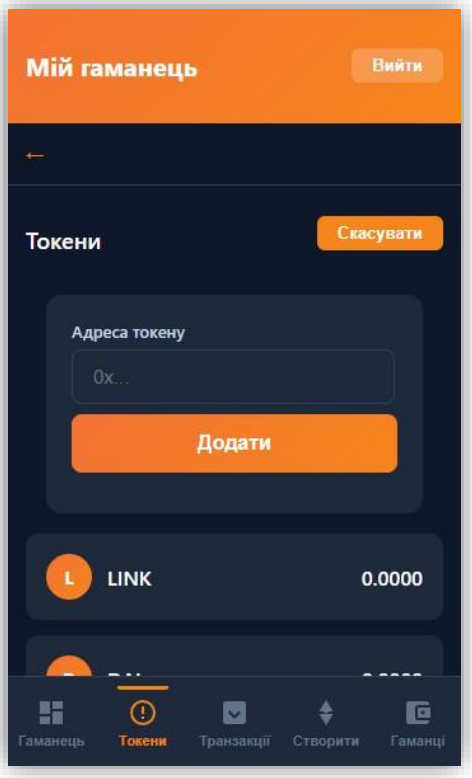

Рисунок 3.11 - Поле для імпорту токенів за адресою контракту

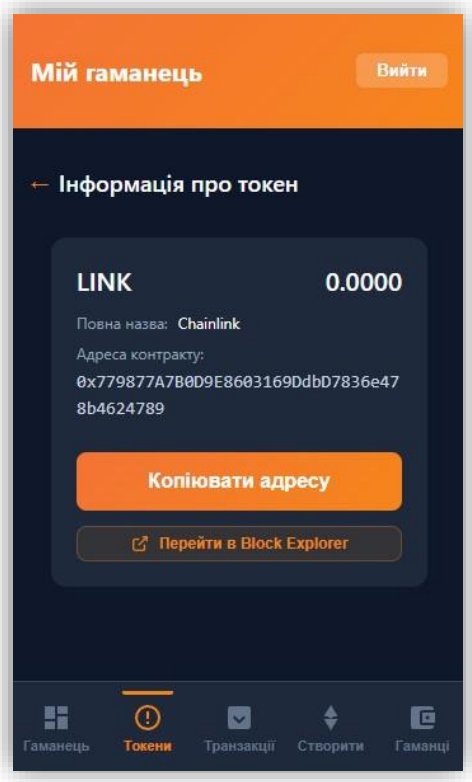

Рисунок 3.12 - Детальна інформація про токен

Компонент історії транзакцій TransactionsTab.jsx (рис. 3.13) відповідає за історію транзакцій з відображенням статусу транзакції («Успішно», «В обробці» та «Помилка»), кількості надісланих токенів, адреси отримувача, дати проведення транзакції, hash транзакції та можливості перегляду транзакції у Block Explorer.

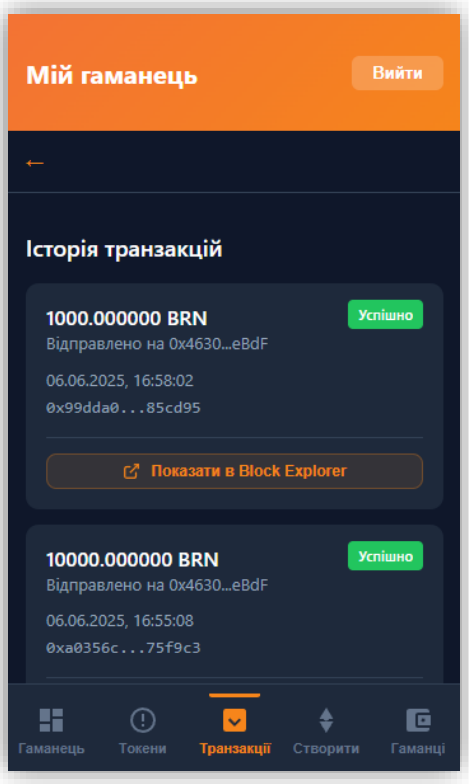

Рисунок 3.13 - Екран з історією транзакцій

Компонент створення токенів (TokenCreatorTab.jsx) використовується для генерації коду смарт-контракту для створення власних токенів ERC-20 та ERC-721. Форма створення токену ERC-20 (рис. 3.14) має поля для вводу назви токену, символу токену та загальної кількості токенів. Форма створення токену ERC-721 (рис. 3.15) включає поля для назви та символу токену. Після введення всіх необхідних даних з'являється згенерований код контракту (рис. 3.16) з покроковими інструкціями розгортання (рис. 3.17).

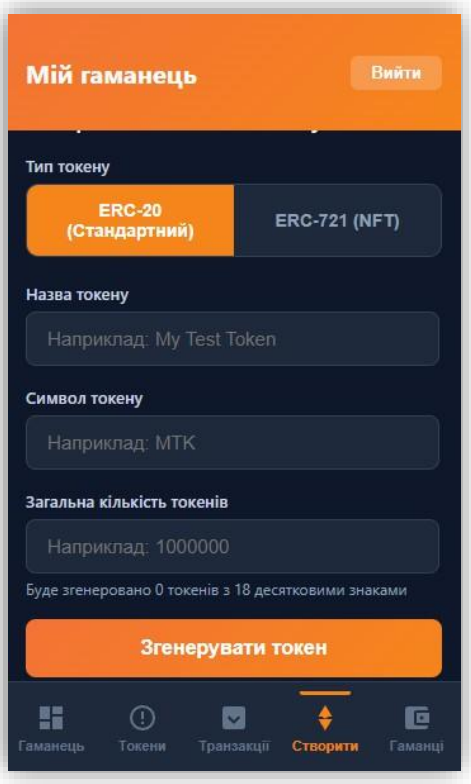

Рисунок 3.14 - Форма створення ERC-20 токену

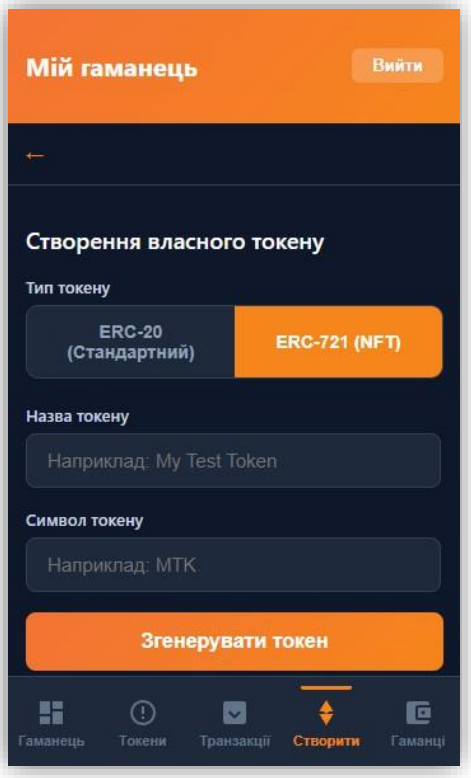

Рисунок 3.15 - Форма створення ERC-721 токену

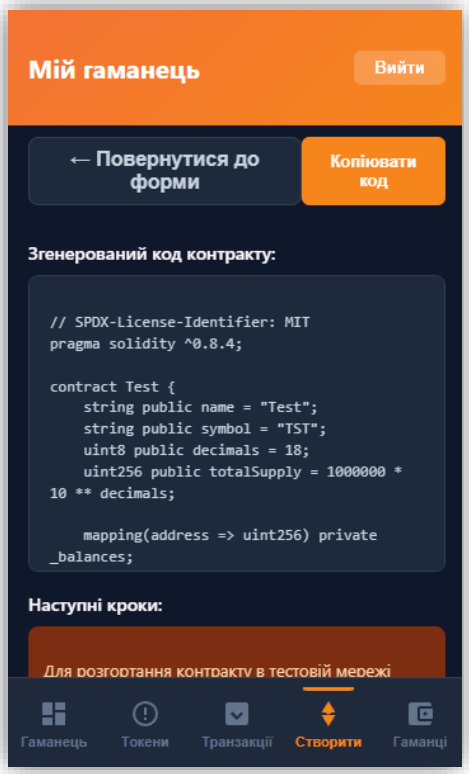

Рисунок 3.16 - Згенерований код контракту токена

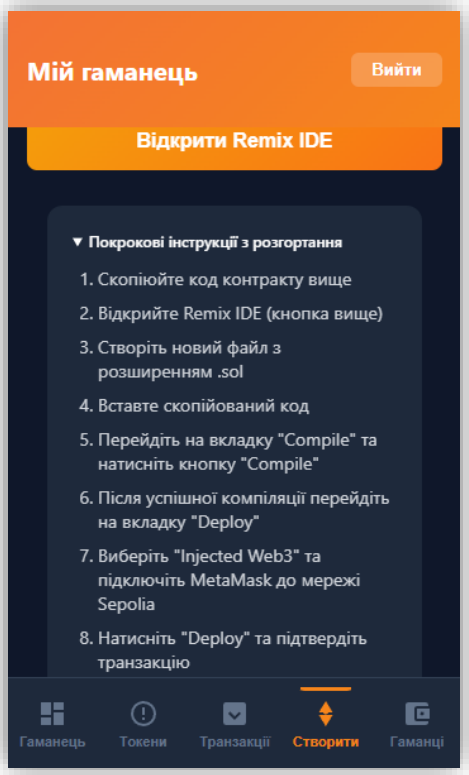

Рисунок 3.17 - Покрокові інструкції з розгортання

Компонент управління гаманцями (WalletsTab.jsx) (рис. 3.18) відповідає за додавання нових гаманців, відображення та переключення між створеними чи імпортованими гаманцями з функціями створення та імпорту нового гаманця аналогічно до WalletCreate.jsx, а також можливістю видалення та перейменування існуючих гаманців.

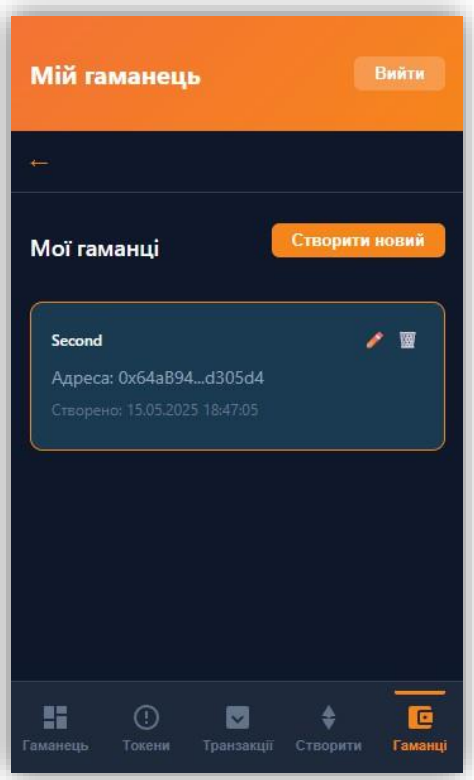

Рисунок 3.18 - Екран з управління гаманцями

3.4. Технічні вимоги та обмеження

Мінімальні системні вимоги включають підтримку браузерів Chrome 90+, Firefox 89+ або Edge 90+, наявність 4 GB оперативної пам'яті, 50 MB дискового простору для розширення та стабільного інтернет-з'єднання для роботи з блокчейном. Рекомендовані вимоги передбачають використання останньої версії браузера, 8 GB оперативної пам'яті, SSD для кращої продуктивності та швидкісного інтернет-з'єднання.

Як браузерне розширення, система має певні обмеження платформи, включаючи максимальний розмір розширення 10 MB в запакованому вигляді, обмеження локального сховища до 10 MB для localStorage, фіксовані розміри вікна 360x600 пікселів для popup та обмеження Content Security Policy, що забороняють inline скрипти та eval().

Безпекові обмеження передбачають, що приватні ключі зберігаються виключно локально без передачі на сервери, відсутність серверної частини виключає централізовані атаки, всі чутливі дані обов'язково шифруються перед збереженням, а система автоматично блокується після періоду неактивності для захисту від несанкціонованого доступу.

3.5. Тестування системи

Перевірка системи відбувалась такими етапами:

1. **Створення та імпорт гаманця:**
    - Генерація нового гаманця.
    - Імпорт за приватним ключем.
    - Імпорт за seed фразою.
    - Перевірка збереження даних.
2. **Операції з транзакціями:**
    - Відправка тестових ETH в мережі Sepolia.
    - Відправка токенів ERC-20.
    - Перевірка історії транзакцій.
    - Отримання токенів з інакшого гаманця.
3. **Управління токенами:**
    - Додавання кастомного токену.
    - Оновлення балансів.
    - Перевірка метаданих токенів.

3.6. Розгортання проекту

Збірка проекту відбувається за допомогою таких команд у терміналі VS Code:

```
# Встановлення залежностей
npm install

# Збірка розширення
npm run build

# Результат у папці dist/
```

Як встановити розширення для розробників:

1. Відкрити chrome://extensions/.
2. Увімкнути режим розробника.
3. Завантажити розпаковане розширення з папки dist/.

ВИСНОВКИ

У результаті виконання дипломної роботи було успішно розроблено повнофункціональну систему криптогаманця для роботи з мережею Ethereum у вигляді розширення для веб-браузера. Проект демонструє комплексний підхід до вирішення задачі безпечного управління криптовалютними активами з використанням сучасних технологій та криптографічних методів.

Наукові результати роботи включають детальний аналіз існуючих рішень у сфері криптогаманців, що дозволив виявити їх переваги та недоліки і сформулювати вимоги до власної системи з покращеними характеристиками безпеки та зручності використання. Було розроблено концептуальну модель системи управління криптовалютними активами, яка базується на принципах децентралізації, криптографічного захисту та повного контролю користувача над приватними ключами. Запропоновано архітектурне рішення на основі багатошарової структури з чітким розподілом відповідальності між компонентами, що забезпечує модульність, масштабованість та легкість підтримки системи. Формалізовано математичні моделі криптографічних операцій, включаючи алгоритми генерації ключів на еліптичних кривих, шифрування AES-256 та деривації ключів за стандартами BIP-32/39/44.

Практичні результати роботи представлені реалізацією повністю функціонального криптогаманця з підтримкою створення та імпорту гаманців за приватним ключем або мнемонічною фразою, управління множинними гаманцями з можливістю швидкого перемикання, відправлення та отримання ETH та токенів стандартів ERC-20/ERC-721, відображення історії транзакцій з детальною інформацією та генерації коду смарт-контрактів для створення власних токенів. Впроваджено надійну систему безпеки через шифрування всіх чутливих даних алгоритмом AES-256, локальне зберігання даних без використання серверів, обов'язкову автентифікацію паролем для доступу до функцій та криптографічно стійку генерацію ключів з використанням Web Crypto API. Створено інтуїтивно зрозумілий користувацький інтерфейс з сучасним дизайном з використанням React та Tailwind CSS, адаптивною навігацією між основними функціями, анімованими

елементами для покращення користувацького досвіду та детальним відображенням інформації про токени і транзакції.

Кількісні показники проекту демонструють загальний обсяг коду понад 3000 рядків JavaScript/JSX, включають 7 основних та 15 допоміжних React компонентів, підтримку понад 50 популярних ERC-20 токенів за замовчуванням, час генерації нового гаманця менше 1 секунди, час підписання та відправлення транзакції 2-3 секунди та розмір готового розширення близько 400 KB.

Якісні показники проекту включають повну сумісність з мережею Ethereum та всіма EVM-сумісними блокчейнами, відповідність криптографічним стандартам BIP-32, BIP-39, BIP-44 [24] та EIP-55, безпечне зберігання даних з використанням сучасних алгоритмів шифрування, зручний інтерфейс, що не потребує технічних знань для використання, та можливість роботи без підключення до централізованих серверів.

Повнота виконання завдання підтверджується тим, що всі поставлені в технічному завданні цілі були успішно досягнуті. Розроблена система повністю відповідає функціональним вимогам та забезпечує безпечне управління криптовалютними активами. Реалізовано всі заплановані функції, від базових операцій з гаманцями до розширених можливостей створення власних токенів.

Рекомендації щодо наукового використання передбачають, що розроблені алгоритмічні рішення та архітектурні підходи можуть бути використані як основа для подальших досліджень у сфері децентралізованих фінансових систем. Запропонована модель безпеки може слугувати прикладом для розробки інших критично важливих додатків, що працюють з чутливими даними. Результати роботи можуть бути корисними для навчальних курсів з блокчейн-технологій, криптографії та розробки децентралізованих додатків.

Рекомендації щодо практичного використання включають можливість використання системи як повноцінного криптогаманця для щоденних операцій з криптовалютами в мережі Ethereum. Модульна архітектура дозволяє легко розширювати функціональність, додаючи підтримку нових блокчейнів, протоколів DeFi або інтеграцію з hardware гаманцями. Рекомендується продовжити розвиток

проекту в напрямках додавання підтримки інших блокчейн-мереж таких як Polygon, BSC, Avalanche, Arbitrum One та інших, інтеграції з протоколами DeFi для обміну токенів, розробки мобільної версії додатку, впровадження multi-signature функціональності та додавання локалізації найбільш використовуваних мов.

Отримані результати підтверджують актуальність обраного напрямку досліджень та демонструють успішне вирішення поставлених задач з розробки безпечного та зручного інструменту для роботи з криптовалютними активами в екосистемі Ethereum.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DefiLlama - DeFi TVL Aggregator (Електронний ресурс) – Режим доступу до ресурсу: https://defillama.com/ – доступний станом на 08.06.2025.
2. Dapp Industry Report 2024 Overview (Електронний ресурс) / DappRadar – Режим доступу до ресурсу: https://dappradar.com/blog/dapp-industry-report-2024-overview – доступний станом на 08.06.2025.
3. NPM Trends - ethers vs viem vs web3 comparison (Електронний ресурс) – Режим доступу до ресурсу: https://npmtrends.com/ethers-vs-viem-vs-web3 – доступний станом на 08.06.2025.
4. MetaMask Card: spend crypto anywhere (Електронний ресурс) / MetaMask – Режим доступу до ресурсу: https://metamask.io/news/introducing-metamask-card-upgrade-your-crypto-spending – доступний станом на 08.06.2025.
5. Trust Wallet - Official Binance Smart Chain Wallet (Електронний ресурс) / Binance – Режим доступу до ресурсу: https://www.binance.com/en/square/post/13838705976562 – доступний станом на 08.06.2025.
6. Cryptocurrency in Higher Education (Електронний ресурс) / iXperience – Режим доступу до ресурсу: https://www.ixperience.co/blog/cryptocurrency-in-higher-education – доступний станом на 08.06.2025.
7. Innovation Insight for Security Rating Services (Електронний ресурс) / Sam Olyaei, Christopher Ambrose, Jeffrey Wheatman; Gartner, Inc. – Режим доступу до ресурсу: https://www.gartner.com/en/documents/5459763 – доступний станом на 08.06.2025.
8. Crypto Hacking Remains a Persistent Threat: $2.2 Billion Stolen in 2024 (Електронний ресурс) / Chainalysis – Режим доступу до ресурсу: https://www.chainalysis.com/blog/crypto-hacking-stolen-funds-2025/ – доступний станом на 08.06.2025.

9. BIP-39: Mnemonic code for generating deterministic keys (Електронний ресурс) / Marek Palatinus, Pavol Rusnak, Aaron Voisine, Sean Bowe – Режим доступу до ресурсу: https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki – доступний станом на 08.06.2025.
10. BIP-32: Hierarchical Deterministic Wallets (Електронний ресурс) / Pieter Wuille – Режим доступу до ресурсу: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki – доступний станом на 08.06.2025.
11. secp256k1 Curve Specification (Електронний ресурс) / Standards for Efficient Cryptography Group – Режим доступу до ресурсу: https://www.secg.org/sec2-v2.pdf – доступний станом на 08.06.2025.
12. React Documentation (Електронний ресурс) / Meta (Facebook) – Режим доступу до ресурсу: https://react.dev/ – доступний станом на 08.06.2025.
13. Ethers.js Documentation (Електронний ресурс) / Richard Moore – Режим доступу до ресурсу: https://docs.ethers.org/v5/ – доступний станом на 08.06.2025.
14. Vite Documentation (Електронний ресурс) / Evan You, Vite Team – Режим доступу до ресурсу: https://vitejs.dev/guide/ – доступний станом на 08.06.2025.
15. Tailwind CSS Documentation (Електронний ресурс) / Adam Wathan, Jonathan Reinink, David Hemphill, Steve Schoger – Режим доступу до ресурсу: https://tailwindcss.com/docs – доступний станом на 08.06.2025.
16. CryptoJS Documentation (Електронний ресурс) / Jeff Mott – Режим доступу до ресурсу: https://cryptojs.gitbook.io/docs/ – доступний станом на 08.06.2025.
17. Framer Motion Documentation (Електронний ресурс) / Framer – Режим доступу до ресурсу: https://www.framer.com/motion/ – доступний станом на 08.06.2025.

18. Chrome Extensions API Documentation (Електронний ресурс) / Google – Режим доступу до ресурсу: https://developer.chrome.com/docs/extensions/ – доступний станом на 08.06.2025.
19. Remix IDE Documentation (Електронний ресурс) / Ethereum Foundation – Режим доступу до ресурсу: https://remix-ide.readthedocs.io/ – доступний станом на 08.06.2025.
20. Infura Developer Documentation (Електронний ресурс) / ConsenSys – Режим доступу до ресурсу: https://docs.infura.io/ – доступний станом на 08.06.2025.
21. AES Advanced Encryption Standard (Електронний ресурс) / NIST Federal Information Processing Standards Publication 197 – Режим доступу до ресурсу: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf – доступний станом на 08.06.2025.
22. BIP-44: Multi-Account Hierarchy for Deterministic Wallets (Електронний ресурс) / Marek Palatinus, Pavol Rusnak – Режим доступу до ресурсу: https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki – доступний станом на 08.06.2025.
23. Ethereum Yellow Paper: A Formal Specification of Ethereum (Електронний ресурс) / Gavin Wood – Режим доступу до ресурсу: https://ethereum.github.io/yellowpaper/paper.pdf – доступний станом на 08.06.2025.
24. ERC-20: Token Standard (Електронний ресурс) / Fabian Vogelsteller, Vitalik Buterin – Режим доступу до ресурсу: https://eips.ethereum.org/EIPS/eip-20 – доступний станом на 08.06.2025.
25. ERC-721: Non-Fungible Token Standard (Електронний ресурс) / William Entriken, Dieter Shirley, Jacob Evans, Nastassia Sachs – Режим доступу до ресурсу: https://eips.ethereum.org/EIPS/eip-721 – доступний станом на 08.06.2025.

Додаток А

Глосарій термінів

AES (Advanced Encryption Standard) – сучасний стандарт шифрування даних, що використовується для захисту конфіденційної інформації.

API (Application Programming Interface) – набір правил та протоколів, які дозволяють різним програмам взаємодіяти між собою.

Blockchain (Блокчейн) – розподілена база даних, що складається з ланцюжка блоків, кожен з яких містить записи транзакцій.

Chrome Extension (Розширення Chrome) – додаткова програма, що встановлюється в браузер Google Chrome для розширення його функціональності.

dApps (Decentralized Applications) – децентралізовані додатки, що працюють на блокчейні без центрального сервера.

DeFi (Decentralized Finance) – децентралізовані фінанси, фінансові послуги на основі блокчейн-технологій.

ECDSA – алгоритм цифрового підпису на еліптичних кривих, що використовується для підписання транзакцій.

ERC-20 – стандарт для створення токенів на платформі Ethereum, визначає основні функції токену.

ERC-721 – стандарт для створення невзаємозамінних токенів (NFT) на платформі Ethereum.

Ethereum – блокчейн-платформа для створення та виконання смарт-контрактів і децентралізованих додатків.

ETH (Ether) – нативна криптовалюта блокчейну Ethereum.

Gas (Газ) – одиниця виміру обчислювальних ресурсів, необхідних для виконання операцій в мережі Ethereum.

Gas Limit – максимальна кількість газу, яку користувач готовий витратити на транзакцію.

Gas Price – ціна за одиницю газу, що визначає швидкість обробки транзакції.

Gwei – одиниця виміру ціни газу в Ethereum (1 Gwei = 0.000000001 ETH).

Hash (Хеш) – унікальний цифровий відбиток даних фіксованої довжини, що генерується криптографічною функцією.

HD Wallet (Hierarchical Deterministic) – ієрархічний детермінований гаманець, що дозволяє генерувати множину ключів з однієї мнемонічної фрази.

Infura – сервіс, що надає доступ до мережі Ethereum через API без необхідності запуску власного вузла.

JSON-RPC – протокол віддаленого виклику процедур, що використовується для взаємодії з блокчейном.

Keccak-256 – криптографічна хеш-функція, що використовується в Ethereum для генерації адрес та підписування транзакцій.

Mainnet – основна мережа блокчейну, де відбуваються реальні транзакції з реальними коштами.

Manifest V3 – новий стандарт для розширень браузера Chrome з покращеною безпекою.

Metamask – популярний криптогаманець у вигляді браузерного розширення для взаємодії з Ethereum.

Mnemonic Phrase (Мнемонічна фраза) – набір з 12 або 24 слів, що використовується для відновлення доступу до гаманця.

NFT (Non-Fungible Token) – невзаємозамінний токен, унікальний цифровий актив на блокчейні.

Node (Вузол) – комп'ютер, що є частиною блокчейн-мережі та зберігає копію всього блокчейну.

PBKDF2 – алгоритм деривації ключів на основі паролю, що ускладнює злом зашифрованих даних.

Private Key (Приватний ключ) – секретний ключ, що дозволяє підписувати транзакції та контролювати кошти на адресі.

Public Key (Публічний ключ) – відкритий ключ, що генерується з приватного ключа та використовується для створення адреси.

QR Code – двовимірний штрих-код, що містить інформацію (наприклад, адресу гаманця) у зручному для сканування форматі.

React – JavaScript бібліотека для створення користувацьких інтерфейсів веб-додатків.

Remix IDE – онлайн-середовище розробки для написання та тестування смарт-контрактів Ethereum.

RLP (Recursive Length Prefix) – метод кодування даних, що використовується в Ethereum для структурування інформації.

RPC (Remote Procedure Call) – механізм виклику функцій на віддалених серверах через мережу.

Salt (Сіль) – випадкові дані, що додаються до паролю перед шифруванням для підвищення безпеки.

Seed Phrase (Seed-фраза) – те саме, що й мнемонічна фраза; набір слів для відновлення гаманця.

Sepolia – тестова мережа Ethereum, що використовується для безкоштовного тестування додатків.

SHA-256 – криптографічна хеш-функція, що генерує 256-бітний хеш з вхідних даних.

Smart Contract (Смарт-контракт) – програма на блокчейні, що автоматично виконується при виконанні заданих умов.

Solidity – мова програмування для написання смарт-контрактів на платформі Ethereum.

Tailwind CSS – CSS-фреймворк з utility-класами для швидкого створення стилів веб-інтерфейсів.

Testnet (Тестова мережа) – тестова версія блокчейну для розробки та тестування без ризику втрати реальних коштів.

Token (Токен) – цифровий актив, створений на блокчейні, що може представляти валюту, актив або право.

Transaction (Транзакція) – операція передачі коштів або виконання дії в блокчейні.

Transaction Hash – унікальний ідентифікатор транзакції в блокчейні.

UI (User Interface) – користувацький інтерфейс; візуальна частина програми, з якою взаємодіє користувач.

UX (User Experience) – користувацький досвід; загальне враження від використання продукту.

Vite – інструмент збірки для швидкої розробки веб-додатків з миттєвим оновленням.

Wallet (Гаманець) – програма або пристрій для зберігання приватних ключів та керування криптовалютними активами.

Web3 – концепція нового інтернету на основі блокчейн-технологій з децентралізованими додатками.

Wei – найменша одиниця криптовалюти Ethereum (1 ETH = 10^18 Wei).

Додаток Б

Функція створення гаманця

```
1. export const createWallet = () => {
2.   const wallet = ethers.Wallet.createRandom().connect(provider);
3.   return {
4.     address: wallet.address,
5.     privateKey: wallet.privateKey,
6.     mnemonic: wallet.mnemonic.phrase
7.   };
8. };
```

Додаток В

Функція шифрування даних

```
export const saveWallet = (walletData, password) => {
  // Шифруємо дані гаманця
  const encrypted = CryptoJS.AES.encrypt(
    JSON.stringify(walletData),
    password
  ).toString();

  // Генеруємо унікальний ID для гаманця, якщо його немає
  const walletId = walletData.id || `wallet_${Date.now()}`;
  const name = walletData.name || `Гаманець ${new Date().toLocaleDateString()}`;

  // Отримуємо поточний список гаманців
  const walletsList = getWalletsList() || [];

  // Створюємо запис про гаманець з метаданими
  const walletMeta = {
    id: walletId,
    address: walletData.address,
    name: name,
    dateCreated: Date.now()
  };

  // Додаємо або оновлюємо метадані гаманця в списку
  const existingIndex = walletsList.findIndex(w => w.id === walletId);
  if (existingIndex >= 0) {
    walletsList[existingIndex] = walletMeta;
  } else {
    walletsList.push(walletMeta);
  }

  // Зберігаємо список гаманців
  localStorage.setItem(WALLETS_LIST_KEY, JSON.stringify(walletsList));

  // Зберігаємо зашифровані дані гаманця
  localStorage.setItem(`wallet_${walletId}`, encrypted);

  // Встановлюємо поточний гаманець
  setCurrentWallet(walletId);

  // Встановлюємо прапорець, що пароль створено
  localStorage.setItem(PASSWORD_KEY, 'true');

  return walletMeta;
};
```

Додаток Г

Функція відправки та підписання транзакції

```
export const sendTransaction = async (privateKey, toAddress, amount) => {
  try {
    const wallet = new ethers.Wallet(privateKey, provider);
    const tx = await wallet.sendTransaction({
      to: toAddress,
      value: ethers.utils.parseEther(amount.toString())
    });

    // Зберігаємо транзакцію в історію
    saveTransaction(wallet.address, toAddress, amount, tx.hash);

    return tx;
  } catch (error) {
    console.error('Помилка відправлення транзакції:', error);
    throw error;
  }
};
```

Додаток Д

Генерація коду ERC-20 токену

```
export const ERC20_TOKEN_TEMPLATE = `// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

contract TOKEN_NAME {
    string public name = "TOKEN_NAME";
    string public symbol = "TOKEN_SYMBOL";
    uint8 public decimals = 18;
    uint256 public totalSupply = TOKEN_TOTAL_SUPPLY * 10 ** decimals;

    mapping(address => uint256) private _balances;
    mapping(address => mapping(address => uint256)) private _allowances;

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);

    constructor() {
        _balances[msg.sender] = totalSupply;
    }

    function balanceOf(address account) public view returns (uint256) {
        return _balances[account];
    }

    function transfer(address to, uint256 amount) public returns (bool) {
        require(_balances[msg.sender] >= amount, "Insufficient balance");

        _balances[msg.sender] -= amount;
        _balances[to] += amount;
        emit Transfer(msg.sender, to, amount);
        return true;
    }

    function approve(address spender, uint256 amount) public returns (bool) {
        _allowances[msg.sender][spender] = amount;
        emit Approval(msg.sender, spender, amount);
        return true;
    }

    function allowance(address owner, address spender) public view returns (uint256) {
        return _allowances[owner][spender];
    }

    function transferFrom(address from, address to, uint256 amount) public returns (bool) {
        require(_balances[from] >= amount, "Insufficient balance");
        require(_allowances[from][msg.sender] >= amount, "Insufficient allowance");

        _balances[from] -= amount;
        _balances[to] += amount;
        _allowances[from][msg.sender] -= amount;

        emit Transfer(from, to, amount);
        return true;
    }
}`;
```

Додаток Е

Конфігураційні файли:

1. manifest.json

```
{
  "manifest_version": 3,
  "name": "Ethereum Wallet Extension",
  "version": "1.0.0",
  "description": "Криптогаманець для взаємодії з тестовою мережею Ethereum",
  "action": {
    "default_popup": "index.html",
    "default_title": "Ethereum Wallet",
    "default_icon": {
      "16": "src/assets/icon16.png",
      "48": "src/assets/icon48.png",
      "128": "src/assets/icon128.png"
    }
  },
  "permissions": ["storage"],
  "icons": {
    "16": "src/assets/icon16.png",
    "48": "src/assets/icon48.png",
    "128": "src/assets/icon128.png"
  },
  "content_security_policy": {
    "extension_pages": "script-src 'self'; object-src 'self'"
  }
}
```

2. package.json

```
{
  "name": "eth-wallet-ext",
  "version": "1.0.0",
  "main": "main.jsx",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@crxjs/vite-plugin": "^1.0.14",
    "@headlessui/react": "^2.2.2",
    "@heroicons/react": "^2.2.0",
    "crypto-js": "^4.2.0",
    "ethers": "^5.7.2",
    "framer-motion": "^12.12.1",
    "qrcode.react": "^4.2.0",
    "react": "^19.1.0",
    "react-dom": "^19.1.0",
    "react-qr-code": "^2.0.15",
    "recharts": "^2.15.3",
    "tailwindcss": "^4.1.6"
  },
  "description": "",
  "devDependencies": {
    "@types/react": "^19.1.3",
    "@types/react-dom": "^19.1.3",
```

```
    "@vitejs/plugin-react": "^4.4.1",
    "typescript": "^5.8.3",
    "vite": "^6.3.5"
  }}
```

3. vite.config.js

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';
import { crx } from '@crxjs/vite-plugin';
import manifest from './manifest.json';

export default defineConfig({
  plugins: [
    react(),
    crx({ manifest })
  ],
  build: {
    chunkSizeWarningLimit: 1000,
  }
});
```